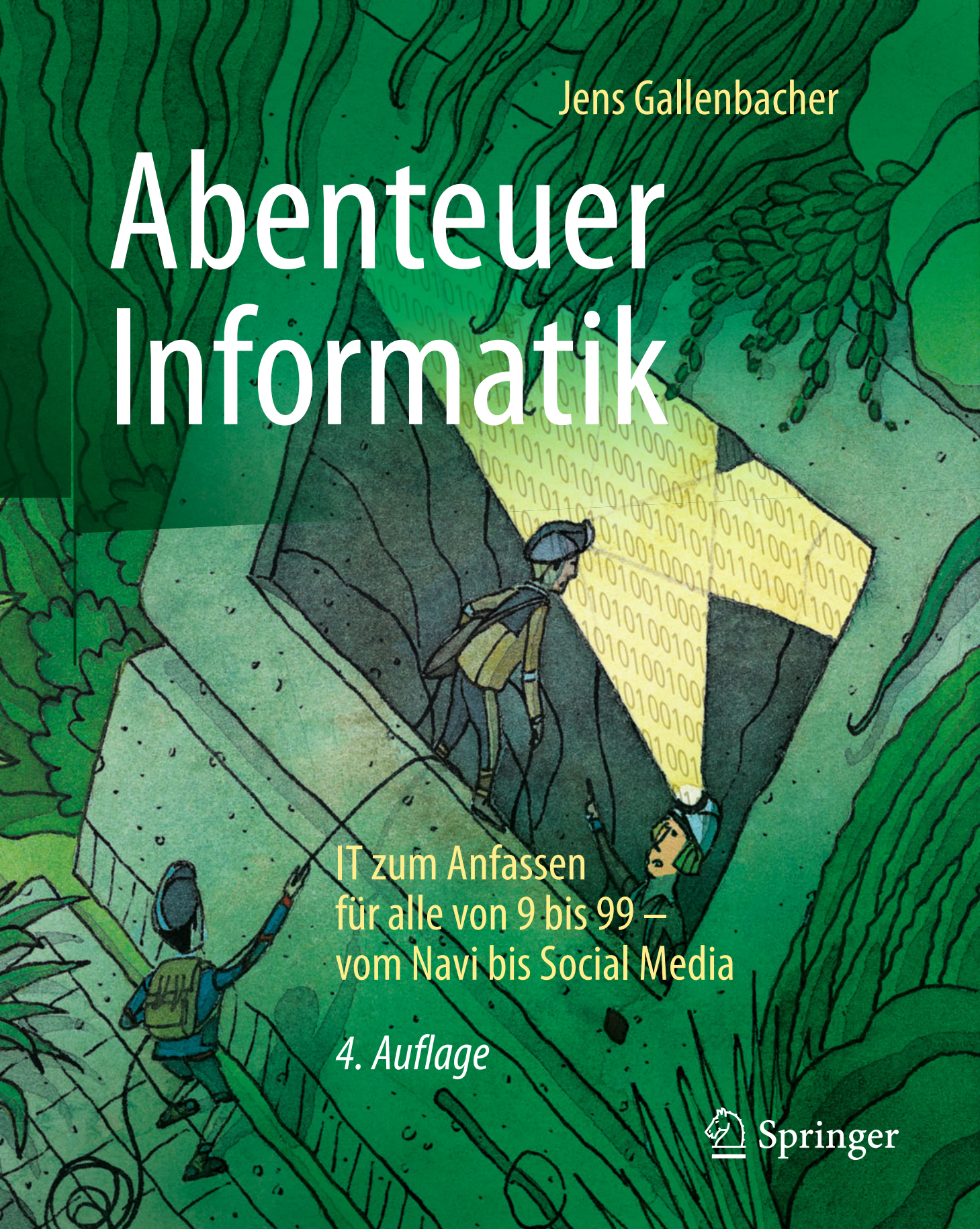

Jens Gallenbacher
Abenteuer Informatik
IT zum Anfassen
für alle von 9 bis 99 –
vom Navi bis Social Media
4. Auflage
Springer

Abenteuer Informatik

Jens Gallenbacher

Abenteuer Informatik

IT zum Anfassen für alle von 9 bis 99 – vom Navi bis Social Media

4. Auflage

Springer

Prof. Dr.-Ing. Jens Gallenbacher
Technische Universität Darmstadt
Didaktik der Informatik
Darmstadt
Deutschland
abenteuer@gallenbacher.de

Ergänzendes Material zu diesem Buch finden Sie auf www.abenteuer-informatik.de

ISBN 978-3-662-53964-4 ISBN 978-3-662-53965-1 (eBook)
DOI 10.1007/978-3-662-53965-1

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer

Planung: Dr. Andreas Rüdinger
Einbandabbildung: © getty-images, wsp design Werbeagentur GmbH, Heidelberg

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer ist Teil von Springer Nature
Die eingetragene Gesellschaft ist Springer-Verlag GmbH Berlin Heidelberg
Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Inhalt

Einleitung

Informationstechnik ist unzweifelhaft ein dominierender Faktor unserer Zivilisation: Sie steckt in normalem Haushaltsgeräten so selbstverständlich wie in Kleidung und Spielen. Technische Geräte wie Autos und Telefone kommen schon lange nicht mehr ohne IT aus. Vernetzung bestimmt bereits heute den alltäglichen Umgang, in dem wir überall und jederzeit Zugriff auf Informationen haben. Auch wenn es nicht möglich scheint: Die Zukunft wird davon noch viel stärker geprägt sein, wenn elektronische Komponenten ganz selbstverständlich untereinander interagieren – solche, die als Geräte sichtbar sind, wie Mobiltelefone, Fernseher oder Autoschlüssel, aber auch solche, von denen wir diese Funktionalität heute noch nicht im Blick haben, wie Schuhe oder Briefumschläge.

Der Fortschritt von Informationstechnik ist paradoxerweise vor allem an ihrer „Unsichtbarkeit“ messbar – ermöglicht durch Miniaturisierung und höhere Leistungsfähigkeit, aber insbesondere auch durch die Gestaltung von Benutzungsschnittstellen, die eine sehr „natürliche“ Bedienung ermöglichen. Gesten wie „Wischen“ oder Sprachbefehle zum Mobiltelefon sind uns nicht mehr fremd. Ist das Informationstechnik? Aber die Systeme agieren zunehmend auch völlig autonom, indem sie unsere Bedürfnisse anhand bestimmter Messwerte erkennen – oder zumindest das, was die Schöpfer dieser Geräte als unsere Bedürfnisse identifizieren.

Genau aus diesem Grund sollte insbesondere immer unsichtbarer werdende Technik zum Wohle von uns allen gestaltet werden, was nur durch die Mitwirkung bzw. zumindest die informierte Meinungsbildung dazu gelingen kann. Die Entscheidung für oder gegen die Nutzung einer neuen App, eines coolen Rückerstattungsangebots oder des Sportarmbands für einen günstigeren Krankenkassentarif muss auf Basis der Kenntnis um die Wirkprinzipien und damit der Möglichkeit eines kritischen Umgangs mit der Technologie gefällt werden.

Informatik ist der Schlüssel zum Verständnis der Informationstechnik und damit der Schlüssel zum Verstehen unserer modernen Umwelt!

Die gute Nachricht: Informatik ist nicht nur cool, sondern auch eine Disziplin, die von Menschen nach menschlichen Prinzipien gestaltet ist! Überraschend viele Prinzipien sind Ihnen also schon bekannt. Schemata großer Datenbanksysteme beruhen auf den gleichen Vorgehensweisen, wie sie auch Grundschülerinnen und Grundschüler beim Sortieren von Spielkarten ganz implizit entwickeln, andere Ideen können wir uns aus der Natur abschauen. Für das Verständnis komplexer Zusammenhänge haben wir eigene Anschauungen entwickelt, wovon die Sprache der Mathematik nur eine ist.

Dieses Buch hilft Ihnen dabei, diese Potentiale zu erkennen und die Informatikerin bzw. den Informatiker in sich zu finden. Dabei werden einerseits alltägliche Geräte wie das Navi im Auto beleuchtet, andererseits bewusst spielerische Herangehensweisen gewählt.

Ein Prinzip durchzieht dabei allerdings alle Kapitel: Das wörtliche „Begreifen“ steht nicht nur am Anfang, sondern ist das zentrale Element. Das geht am besten ohne Computer: mit Papier, Bleistift, einer Schere und den Vorlagen aus dem Buch. Die

„komplizierten“ Dinge brauchen dann noch ein Stück Holz, Nägel, Faden und einen Hammer ...

Das Prinzip hat sich seit der ersten Auflage, die 2006 erschienen ist, bewährt und ich habe es daher noch konsequenter in allen Kapiteln umgesetzt. Überhaupt ist diese vierte Auflage nun zum ersten Mal nicht nur ergänzt, sondern komplett überarbeitet. An dieser Stelle möchte ich auch den vielen Leserinnen und Lesern danken, die einerseits Fehler, andererseits auch Missverständnisse bei bestimmen Formulierungen zurückgemeldet und damit geholfen haben, das Buch zu verbessern.

Im spielerischen Teil, bei dem es vor allem um den Spaß, das Experiment und die Aha-Effekte geht, werden Sie nicht im Detail lernen, wie man einen Computer programmiert, aber Sie werden verstehen, was hinter vielen Standardprogrammen steckt.

Der Experimentierteil ist sehr einfach aufgebaut, erfordert praktisch kein Vorwissen und ist daher auch für Kinder (etwa ab der 3. Klasse) geeignet, wenn er mit den Eltern zusammen durchgearbeitet wird. Der Weg zur Lösung ist hier das Ziel: Wie geht ein Informatiker vor, um ein Problem zu knacken?

Auch die Begründung ist meist noch recht einfach zu verstehen: Warum funktioniert das Verfahren?

Wer dann (hoffentlich) ganz versessen darauf ist, noch mehr zu erfahren, kann am Ende jedes Kapitels eine mehr wissenschaftlich gehaltene Zusammenfassung und Vertiefung lesen. Dieser Teil ist mit „Was steckt dahinter?“ übertitelt. Im Mittelpunkt stehen hier die Fragen, warum das Verfahren immer funktioniert (also der Beweis), wo weitere Anwendungen liegen und wie in der Praxis das Verfahren noch verbessert wird. Hier sind zum Verständnis oft Mathematikkenntnisse vonnöten, wie sie in der gymnasialen Oberstufe vermittelt werden.

Dieses Buch kann also auf ganz verschiedene Weise gelesen bzw. – wie ich lieber sagen möchte – erlebt werden. Um das zu unterstützen, sind ein paar kleine Details eingebaut, die ich hier beschreiben möchte:

Beim Lesen von „Abenteuer Informatik“ können Sie eine mehr passive oder mehr aktive Rolle wählen. Selbstverständlich ist es spannend und auch lehrreich, auf wichtige Erkenntnisse der Informatik selbst zu kommen – zum Beispiel nach einem eigenhändig durchgeführten Experiment.

Sehr oft finden Sie daher eine Frage, einen Denkanstoß oder einen Arbeitsauftrag in blauer Schrift. Direkt dahinter steht das „Denk-Köpfchen“.

Es soll Sie kurz daran hindern, hier schon weiterzulesen. Sie können sich nun entscheiden, ob Sie selbst versuchen, zu einer Lösung zu kommen oder diese einfach im nächsten Abschnitt lesen. Lassen Sie sich davon leiten, wie spannend das Thema für Sie ist.

Eine Quelle zusätzlicher Information ist die kleine linke bzw. rechte Spalte des Buches. Hier finden Sie kurze Lebensläufe wichtiger Persönlichkeiten, Anekdoten und anderes Wissenswerte im Zusammenhang mit dem Haupttext. Das Verständnis des Kapitels ist allerdings auch ohne diese Spalte gewährleistet. Sie können sich daher diesen Teil für ein zweites Durchlesen aufsparen oder ihn als willkommene Ablenkung ansehen.

Den Anfang neuer Kapitel finden Sie im Buch übrigens recht einfach, wenn Sie es von der Seite betrachten: Das große Bild auf der linken Seite ist auch hier zu erkennen und lässt sich auf diese Weise als eine Art Index nutzen.

Wichtige Begriffe

Im Text sind immer wieder grundlegende Begriffe und Methoden der Informatik kurz erklärt. Um sie leichter wiederzufinden, stehen sie in blauen Kästchen. Wenn Sie möchten, legen Sie sich doch ein eigenes Merkblatt mit den Inhalten dieser Kästchen an. Ganz hinten im Buch sind allerdings ein paar wichtige Schlagwörter in einem Glossar zusammengefasst auch übersichtlich nachzuschlagen.

Vorderseite

Rückseite

Die Reihenfolge der Kapitel ist übrigens recht lose. Auch wenn Sie das Buch nicht von vorne nach hinten durchlesen, sondern sich von Ihrer „Abenteuerlust“ leiten lassen, kann man die meisten Texte verstehen! Manchmal ist allerdings ein Hinweis auf eines der vorhergehenden Kapitel gegeben, das dann eine wichtige Grundlage beinhaltet. Mein Tipp ist, auf jeden Fall das erste Kapitel, vielleicht auch noch das zweite Kapitel zu Beginn zu lesen. Danach können Sie unbeschwert stöbern.

Kommen wir zu einem sehr wichtigen Teil, der dieses Buch auch von den allermeisten anderen Informatik-Büchern unterscheidet: der Bastelbogen! Er ermöglicht das wörtliche „Begreifen“ ohne zu viel Aufwand in der Herstellung eigener Materialien. Es genügt meistens, diese auszuschneiden.

Jedes Kapitel enthält daher am Ende entsprechende Kopiervorlagen. Manche Spielmaterialien – wie die Karten der binären Magie – brauchen zwingend Vorder- und Rückseite. Da selbst Kopierer und Drucker mit Duplex-Einheit oft nicht schaffen, die beiden Seiten wirklich deckungsgleich zu Papier zu bringen, habe ich diese Vorlagen so angelegt, dass man sie nach dem Ausschneiden an der durchgezogenen Linie noch an der gestrichelten Linie faltet und entweder zusammenklebt oder gefaltet laminiert.

Abbildung
Gedruckte Bastelbögen

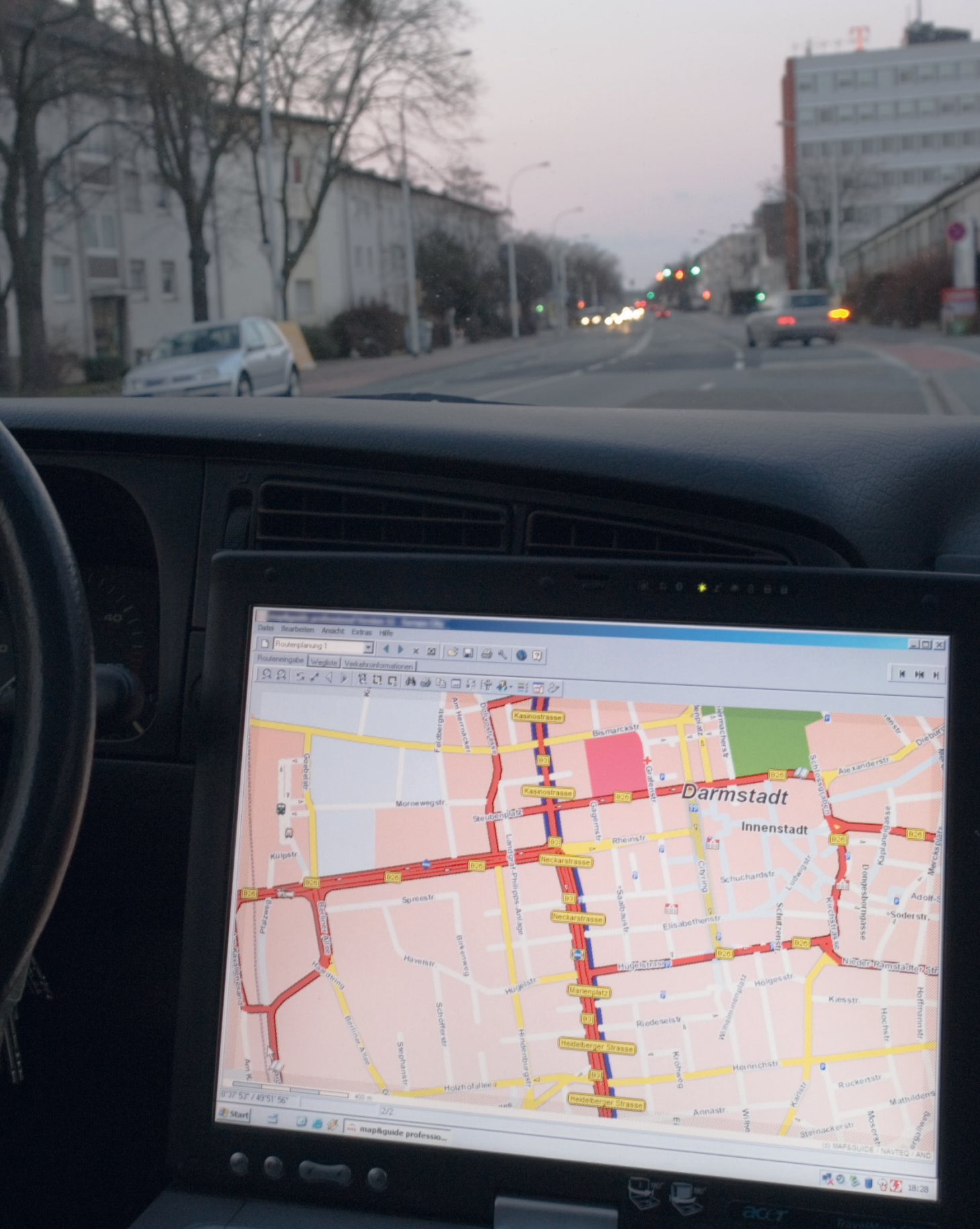
Noch einfacher ist selbstverständlich, die fertigen Bastelbögen zu benutzen. Um noch mehr Inhalt in das Buch zu bringen und die elektronische Ausgabe zu ermöglichen, sind diese nicht mehr integraler Bestandteil, sondern gesondert zu beziehen. Das erleichtert auch Lehrerinnen und Lehrern, gleich einen ganzen Klassensatz zu bekommen. Leserinnen und Lesern, die dieses Buch in einer Bücherei ausgeliehen haben, können so ebenfalls unbeschwert schneiden und kleben. Die Bedingungen für den Bezug sind zum Zeitpunkt des Drucks noch nicht ganz geklärt, bitte informieren Sie sich über die Webseite

www.abenteuer-informatik.de

Diese ist auch die zentrale Seite für Korrekturen und zusätzliche Informationen zum Buch sowie der gleichnamigen Ausstellung, die inzwischen auf Wanderschaft in Universitäten, Schulen und auch Häusern wie dem Heinz Nixdorf Forum in Paderborn oder dem Ars Electronica Center in Linz ist. Im Science-Center ExperiMINTa in Frankfurt am Main ist sie dauerhaft zu sehen.

Damit genug der Nutzungshinweise – ich wünsche Ihnen viel Spaß bei Ihrem ganz persönlichen Abenteuer Informatik!

Jens Gallenbacher


Datei Bearbeiten Ansicht Extras Hilfe
Routenplanung 1
Routeneingabe
Wegliste
Verkehrsinformationen
Darmstadt
Innenstadt
Kasinostrasse
Bismarckstr.
Alexanderstr.
Mornewegstr.
Steubenplatz
Rheinstr.
Neckarstrasse
Schuchardstr.
Spreestr.
Elisabethenstr.
Hügelstrasse
Marienplatz
Holgestr.
Kiesstr.
Riedeselstr.
Heidelberger Strasse
Heinrichstr.
Holzhofallee
Annastr.
Rückertstr.
Soderstr.
Nieder-Ramstädter Str.
Start
map&guide professio...
18:28

1. Sag mir wohin ...

Einführung

Routenplaner gehören heute schon fast zum Alltag: Viele Autos haben sie bereits eingebaut und auch wer keinen im Fahrzeug hat, lässt sich den günstigsten Weg zu seinem Ziel oft auf dem heimischen PC ermitteln und druckt ihn aus.

Versuchen wir doch gleich einmal das nachzuvollziehen: Nehmen Sie einen großen Straßenatlas und ermitteln Sie die günstigste Strecke von Stockheim nach Weilheim!

Zu viel Arbeit? Kein Atlas? Also gut – ich hatte in der Einleitung ja versprochen, dass alle notwendigen Materialien hier im Buch zu finden sind. Daher arbeiten wir erst einmal mit der folgenden kleinen Welt nach Abbildung 1.1.

Die Karte zeigt Orte, zwei Autobahnen, größere und kleinere Landstraßen. Die roten und blauen Zahlen geben dabei immer die Länge der Straße an, wenn man sie mit dem Auto fährt. Man kann sehen, dass kleine Straßen meistens viel länger sind, als sie scheinen, weil die zahlreichen Kurven und Berge nicht eingezeichnet sind.

Welches ist denn nun der günstigste Weg von Imstadt nach Oppenheim?

Erster Ansatz wäre, zur nächsten Autobahn zu fahren. Aber geht das am besten über Pappstadt oder die Auffahrt Budingen? Außerdem führt ja von Pappstadt aus auch eine direkte Landstraße zum Ziel. Die ist aber wohl gewunden und ziemlich lang. Oder doch lieber die gelbe Straße zum Flughafen und von dort die Autobahn nach Oppenheim nehmen?

Versuchen Sie, die Lösung selbst herauszufinden. Hinweis: Sie müssen 24,6 km zurücklegen.

Geschafft? Gut! Dann lehnen Sie sich zurück und genießen Sie den Erfolg.

Wie sind Sie vorgegangen? Sie haben wahrscheinlich alle möglichen Wege durchprobiert und die Entfernung zum Ziel ermittelt. Dann haben Sie sich für den günstigsten Weg entschieden.

Dieses Verfahren gibt es auch bei Computern – es hat sogar einen Namen: die Brute-Force-Methode, also etwa „Brutale Macht“. Warum? Weil auf diese Weise etwas umfangreichere Aufgaben nur mit extrem großer Rechenkraft gelöst werden können. Überlegen Sie einmal, wie viele verschiedene Wege Sie schon bei der gegebenen, sehr übersichtlichen Karte durchspielen mussten. Stellen Sie sich nun vor, wie das mit 1000 und mehr Städten wäre – hier wären normale Rechner gar nicht mehr zur Lösung fähig.

Außerdem besitzt ein Rechner keine Intelligenz: Während Sie beim Durchprobieren unbewusst alle absurden und unwahrscheinlichen Möglichkeiten verwerfen, muss er diese durchrechnen.

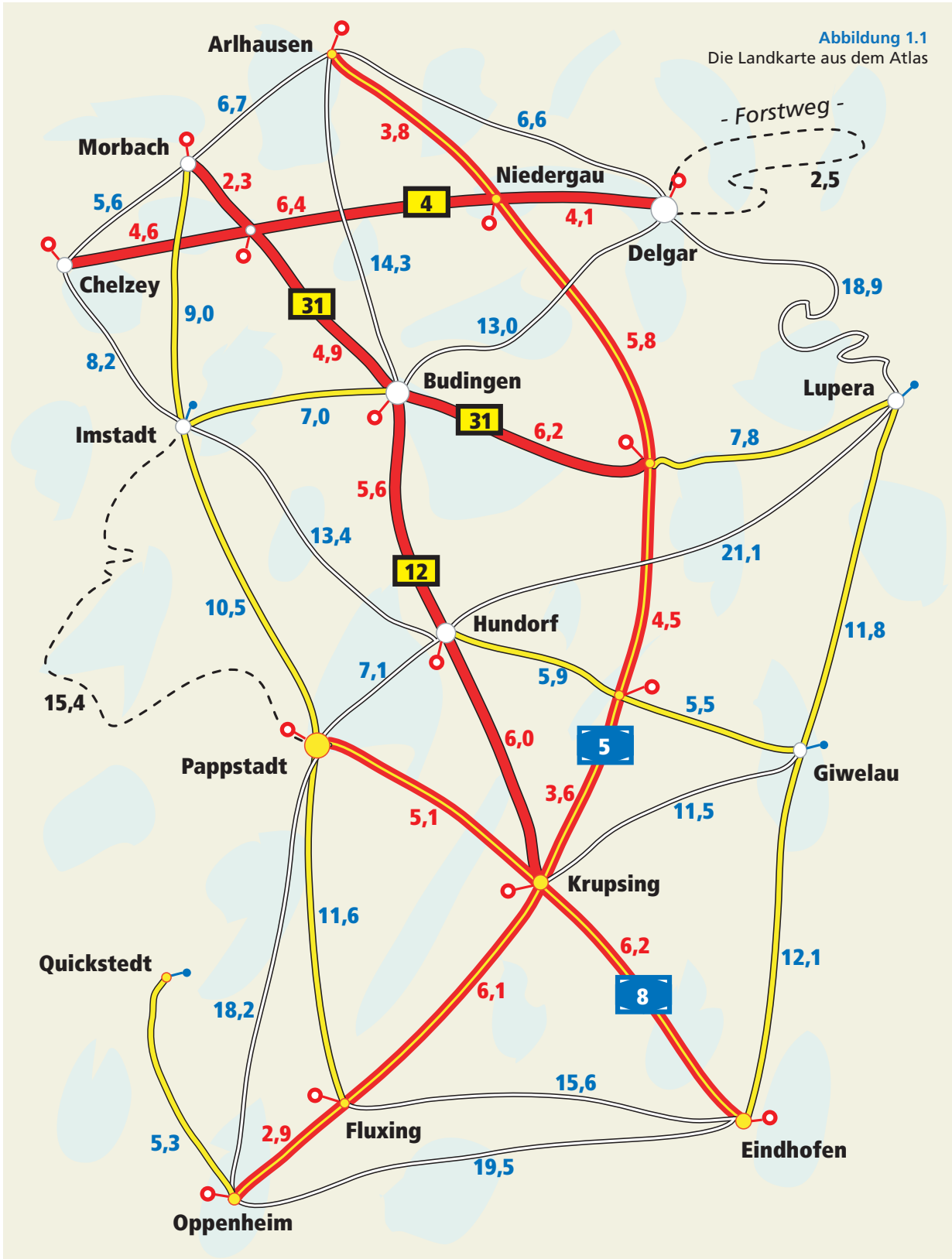


Abbildung 1.1
Die Landkarte aus dem Atlas

Vorüberlegungen

Wie kommt ein Informatiker nun zu einer besseren Lösung?

Zunächst sollte man mit der Methode der Abstraktion arbeiten!

> **Methode der Abstraktion**
>
> In zur Verfügung stehender Information stecken sowohl relevante als auch unwesentliche Anteile. Durch Abstraktion reduzieren Sie die Information auf das für die aktuelle Problemlösung Wesentliche: Dadurch können Sie sich besser auf Ihre Aufgabe konzentrieren.

Man könnte auch sagen: Werfen Sie alles Überflüssige über Bord und konzentrieren Sie sich auf das Wesentliche. Was ist aber bei der gestellten Aufgabe wesentlich?

Alle gegebenen Informationen stecken in der Karte. Welche Typen von Informationen kann man erkennen? Erstellen Sie eine Liste, bevor Sie weiterlesen.

Die Informationen der Karte sind in folgender Tabelle zusammengefasst.

	wichtig?
Namen der Städte	☐
Position der Städte	☐
Größe der Städte	☐
Verlauf der Straßen	☐
Länge der Straßen	☐
Namen und Nummern der Straßen	☐
Straßentyp	☐
Straße führt von ... nach ...	☐
Landschaftliche Information	☐

An dieser Stelle fällt Ihnen eventuell auch bereits auf, dass unsere bisherige Formulierung der Aufgabe, den „günstigsten" Weg zu finden, nicht präzise genug ist. Genau genommen suchen wir den (Strecken-)kürzesten Weg!

Überlegen Sie weiter, welche dieser Informationen wir benötigen, um den kürzesten Weg zwischen zwei Städten zu suchen. Markieren Sie für jede Information, ob diese Ihrer Meinung nach für die Aufgabenstellung wichtig oder nicht wichtig ist. Der kürzeste Weg soll sich hierbei auf die zu fahrende Strecke beziehen, nicht auf die gefahrene Zeit.

Auch **abstrakte Malerei** ist die Konzentration auf das Wesentliche. Der Künstler stellt die Aspekte in den Mittelpunkt, die ihn bewegen, zum Beispiel ein Gefühl oder ein Ereignis. Details wie die realistische Darstellung treten dadurch in den Hintergrund. Hier abgebildet ist das „Blaue Pferd" von Franz Marc (1911).

Abstraktion begegnet uns ständig! Wegweiser, Straßenschilder, Fahrpläne, Infobroschüren und viele andere Dinge des Alltags sind so aufbereitet, dass die nötigen Informationen möglichst offen und gut erkennbar dargestellt sind.

Mein Ergebnis ist folgendes:

	wichtig?
Namen der Städte	☑ Wenn man nicht weiß, welche Stadt wie heißt, kann auch nicht der kürzeste Weg zwischen Imstadt und Oppenheim bestimmt werden.
Position der Städte	☐ Es ist uns egal, wo sich die Städte genau befinden. Relevant sind nur die Straßen zwischen den Städten.
Größe der Städte	☐ Kommt in unserer Aufgabenstellung nirgendwo vor.
Verlauf der Straßen	☐ Es kommt nur auf die Strecke an, nicht auf den Verlauf.
Länge der Straßen	☑ Um die Reisestrecke zu summieren, benötigen wir die einzelnen Strecken zwischen den Orten.
Namen und Nummern der Straßen	☐ Zumindest zur Bestimmung der kürzesten Strecke irrelevant.
Straßentyp	☐ Da es nur auf die Entfernungen, nicht auf Zeit ankommt, ist egal, ob Autobahn oder Feldweg gefahren wird.
Straße führt von ... nach ...	☑ Wir benötigen die Information, von welcher Stadt zu welcher anderen eine Straße führt.
Landschaftliche Information	☐ Offensichtlich ...

Nun kann die Karte neu gezeichnet werden, und zwar so, dass möglichst alle irrelevanten und damit störenden Informationen fehlen. Versuchen Sie es einmal selbst, bevor Sie weiterlesen!

Aus Gründen der Übersichtlichkeit habe ich hier in Abbildung 1.2 die Städte auf ihren ersten Buchstaben reduziert. Das ist jedoch nicht unbedingt notwendig. Wenn Ihre Lösung die ausgeschriebenen Namen enthält, können Sie auch damit weiterarbeiten.

Man sieht, dass es noch ein paar Besonderheiten gibt: An vier Stellen kreuzen sich zwei Straßen, ohne dass es Auf- oder Abfahrten von einer zur anderen gäbe (z. B. eine Autobahnbrücke über einer Landstraße). Dies ist mit einem Bogen dargestellt. Darüber hinaus gibt es weitere drei Stellen, an denen sich zwei Straßen schneiden und es Auf- und Abfahrten gibt – gekennzeichnet mit einem Punkt. Außerdem ist unklar, welches nun die „Straße zwischen I und P“ ist, denn hierfür gibt es zwei Kandidaten.

Informatiker wollen sich jedoch prinzipiell nicht mit Sonderfällen beschäftigen und lieben es daher übersichtlich: Zu viele spezielle Fälle und Unterscheidungen machen das Denken schwierig. Wie bei Mathematikern, die einen Bruch erst einmal auf den gleichen Nenner bringen, wird hier auch versucht, ein Problem möglichst gleichförmig darzustellen.

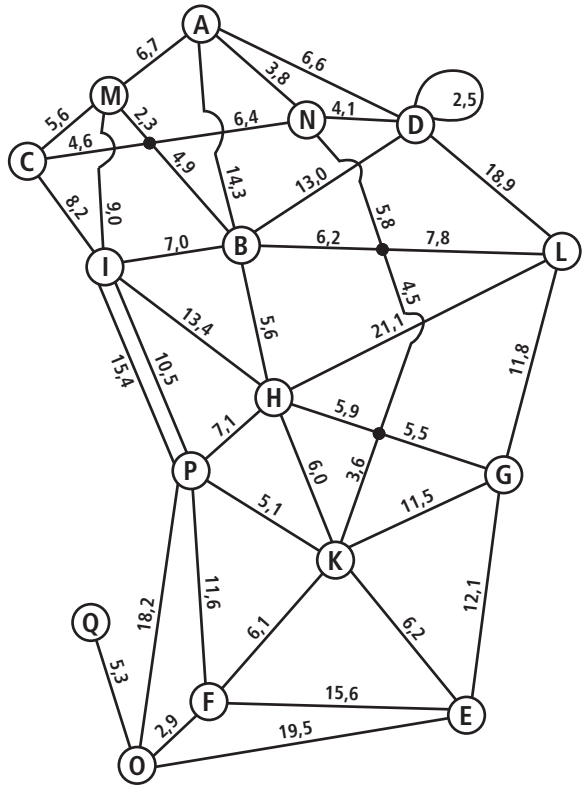


Abbildung 1.2
Abstraktere Form der Landkarte

Methode der Gleichformung

Versuchen Sie, die verschiedenen Facetten eines Problems auf die gleichen Grundelemente zurückzuführen. Dadurch wird einerseits das Problem übersichtlicher und andererseits benötigt man weniger Lösungsansätze: Für gleichförmige Teilprobleme kann der gleiche Lösungsansatz verwendet werden.

Können Sie die Karte auf diese Weise noch einfacher gestalten?

Überlegen Sie, welches die Eigenschaften der „normalen" Elemente sind und ob man die speziellen Elemente nicht auch als normale Elemente darstellen kann.

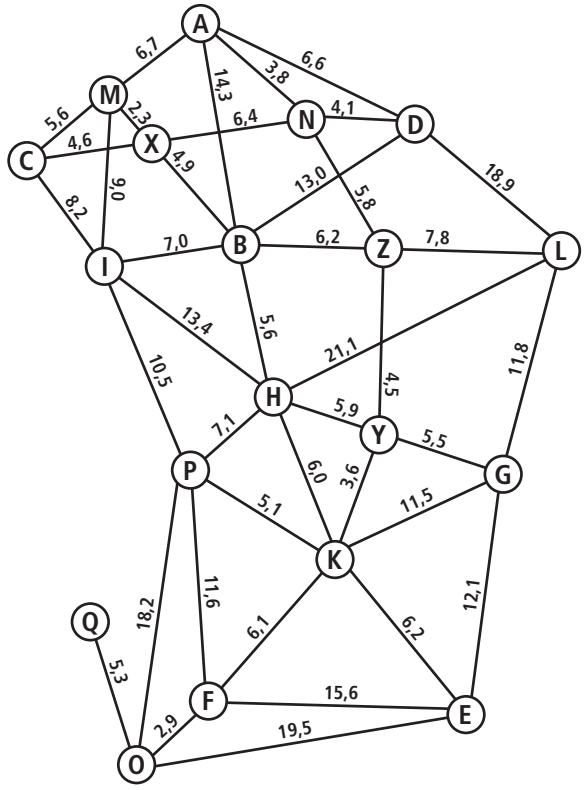
Auf der Karte sind Städte als Kreise eingezeichnet. Ohne dass dies speziell vermerkt ist, kann man offenbar bei Städten problemlos von einer Straße auf eine angrenzende Straße wechseln.

Genau das soll auch an den mit Punkt gekennzeichneten Stellen möglich sein. Also tun wir einfach so, als ob sich dort Städte befinden. Um sie nicht mit den anderen Städten zu verwechseln, nennen wir sie Ⓧ, Ⓨ und Ⓩ.

An allen weiteren Kreuzungspunkten zwischen zwei Straßen ist jetzt kein Wechsel mehr möglich. Es ist daher auch keine Unterscheidung, also auch keine Kennzeichnung durch Bögen mehr notwendig.

Für die Bestimmung des kürzesten Weges kommt von zwei möglichen Strecken nur die kürzeste in Frage. Rundfahrten wie von Delgar aus und zurück spielen hier keine Rolle. Beides kann daher in unserer Arbeitskarte berücksichtigt werden. Den fertigen Plan zeigt Abbildung 1.3.

Abbildung 1.3
Landkarte mit durch virtuelle Orte ersetzten Knotenpunkten



Leonhard Euler (1707 – 1783) Euler hat schon 1736 als Erster Wegeprobleme durch abstrakte Darstellung vereinfacht, als er das Königsberger Brückenproblem löste: „Gibt es einen Rundweg, der alle sieben Brücken über den Fluss Pregel genau einmal überquert und wieder zum Ausgangspunkt führt?" Euler bewies, dass es keinen solchen Weg geben kann.

Die Städte sind immer noch auf ihrer geographischen Position eingezeichnet. Dadurch ergeben sich an manchen Stellen Ballungszentren. Die Straßenführung wird unübersichtlich. Die geographische Position der Städte haben wir jedoch weiter vorne als irrelevant deklariert. Um es Ihnen so einfach wie möglich zu machen, habe ich die Karte in Abbildung 1.4 etwas entzerrt.

Bitte überprüfen Sie, dass die Inhalte immer noch übereinstimmen, also die Verbindungen zwischen den Städten gleich sind und die gleiche Längenangabe aufweisen. Lediglich die Darstellung hat sich geändert. Sie können auch erkennen, dass es sich noch um die gleiche Karte handelt wie am Anfang des Kapitels – lediglich mit weniger Detailinformationen.

Mit dieser Karte werden wir ab jetzt weiterarbeiten, daher ist sie am Ende des Kapitels als Abbildung 1.K1 nochmals besonders groß abgedruckt – zum Beispiel als Kopiervorlage. Sie können sich vorstellen, dass wir zur Bestimmung des kürzesten Weges auch viel rechnen müssen, daher habe ich es uns zur Übung auch noch etwas einfacher gemacht und die einzelnen Wegstrecken gerundet.

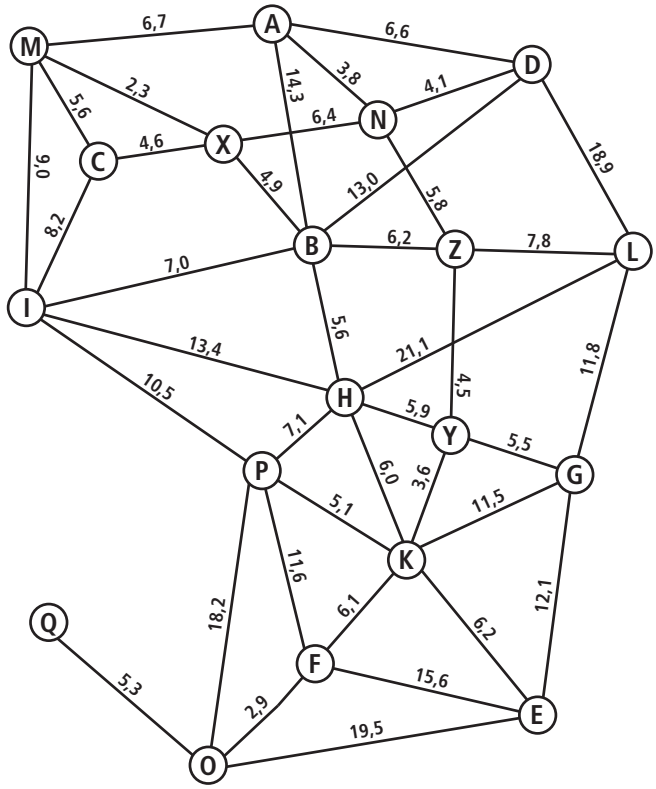


Abbildung 1.4
Entzerrte Landkarte

Dijkstra und die Ameisen

Wie ermitteln wir denn aber nun den kürzesten Weg von Imstadt nach Oppenheim?

Der direkte Ansatz, immer ganze Wege zu betrachten, ist ja bereits aufgrund des hohen Aufwands gescheitert. Versuchen wir etwas anderes – vielleicht können wir ja hier von der Natur lernen:

Ein Stamm Ameisen hat auf der Suche nach Futter ein ähnliches Problem: Eine Kundschafterin findet ein großes Stück Fleisch. Welchen Weg sollen die Arbeiterinnen nehmen, um die Beute am schnellsten zu sichern?

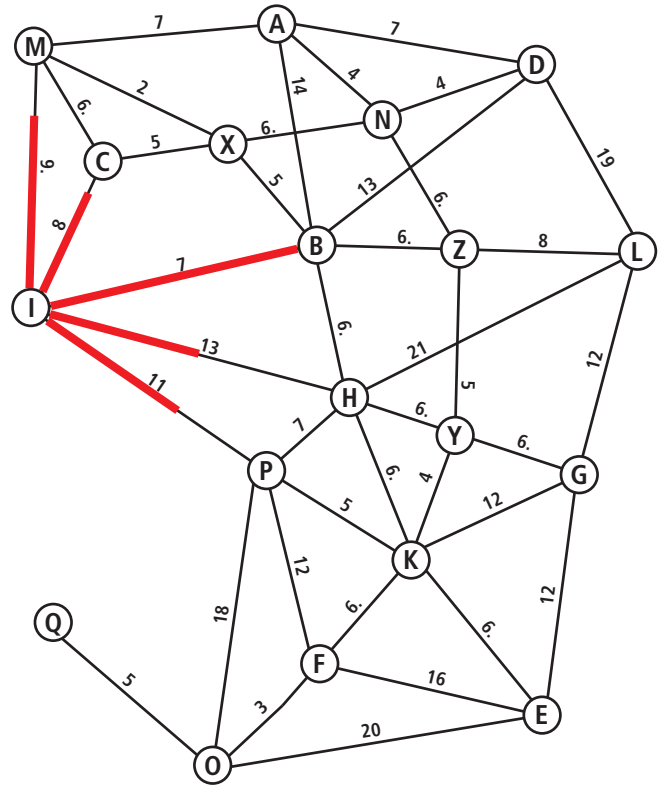
Setzen wir also den Stamm Ameisen auf unseren Ausgangspunkt Imstadt bzw. Ⓘ. Fünf Wege führen von dort weg, also teilen sich unzählige Ameisen auf, um diese zu erkunden. Wir nehmen jetzt einmal an, dass alle Ameisen gleich schnell sind, zum Beispiel 1 km pro Minute (okay, für unser Problem setzen wir offenbar Turbo-Ameisen ein ...).

Mit dem Finger auf der Landkarte verfolgen wir den Weg der Ameisen. Abbildung 1.5 zeigt in Rot ihren Fortschritt nach 7 Minuten.

Die Ameisen haben Ⓑ erreicht. Auf den anderen Strecken sind sie noch auf dem Weg. Bitte lassen Sie sich nicht davon täuschen, dass die Strecke mit 7 km am längsten aussieht: Wir verwenden ja explizit keinen maßstabsgetreuen Plan!

Was haben wir dadurch bisher von den Ameisen gelernt?

Abbildung 1.5
Die Ameisen auf dem Weg von Imstadt aus



Genau! Um von Ⓘ nach Ⓑ zu kommen, gibt es garantiert keinen günstigeren Weg als den mit 7 km. Die Ameisen haben ja sämtliche bisher für sie möglichen Wege ausprobiert und sind nach 7 km zuerst bei Ⓑ angekommen.

Wie geht es jetzt weiter? Die Ameisen, die bisher nirgendwo angekommen sind, setzen ihren Weg einfach fort. Die Ameisen bei Ⓑ teilen sich erneut auf: Wieder sind fünf Wege möglich. Den bisherigen Erfolg dokumentieren sie, indem sie die bisher zurückgelegte Strecke bei Ⓑ vermerken. Abbildung 1.6 zeigt den Plan der Ameisen.

Nach insgesamt 8 Minuten kommt der nächste Ameisentrupp bei Ⓒ an. Die Insekten sehen, dass sie die Ersten sind, markieren die Strecke, verzeichnen die Anzahl der bisher gelaufenen Kilometer und teilen sich auf die zwei bei Ⓒ weitergehenden Wege auf.

Am Ende der neunten Minute kommt dann auch der Trupp bei Ⓜ als Erster an. Auch dieser Weg wird vermerkt (siehe Abbildung 1.7). Von hier aus sind drei weitere Strecken zu erkunden.

So weit verlief alles nach dem gleichen Schema. Die kürzesten Strecken zu den Städten Ⓑ, Ⓒ und Ⓜ stehen nun fest.

Vielleicht haben Sie bemerkt, dass nun Ameisentrupps sowohl von Ⓜ als auch von Ⓒ ausgehend unterwegs sind – zwischen beiden Städten auf Kollisionskurs.

Was passiert jetzt, wenn sie sich irgendwo auf der Strecke dazwischen begegnen? Welche Informationen können sie austauschen? Bringt ihnen das etwas für ihr Ziel, das Gelände zu erkunden?

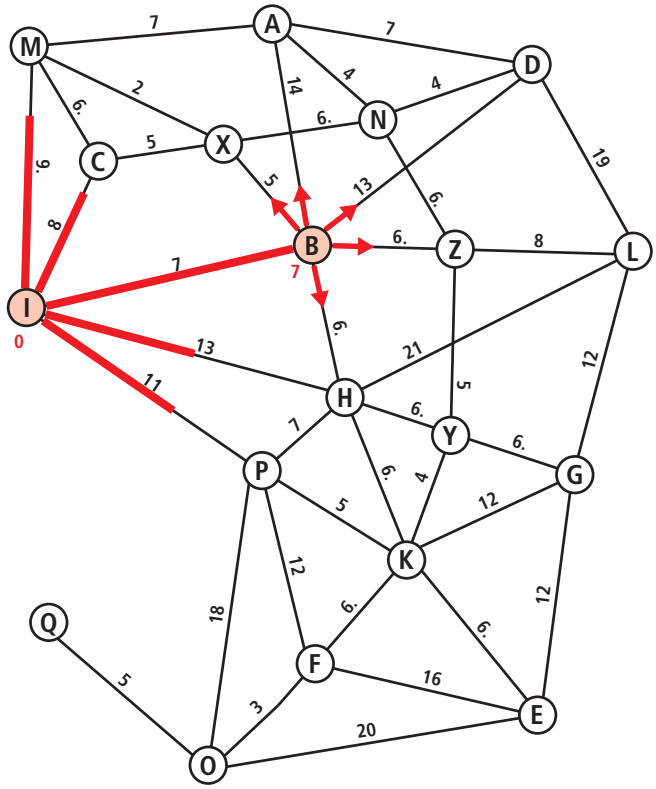


Abbildung 1.6
Die Ameisen bewegen sich von Budingen aus in alle Richtungen.

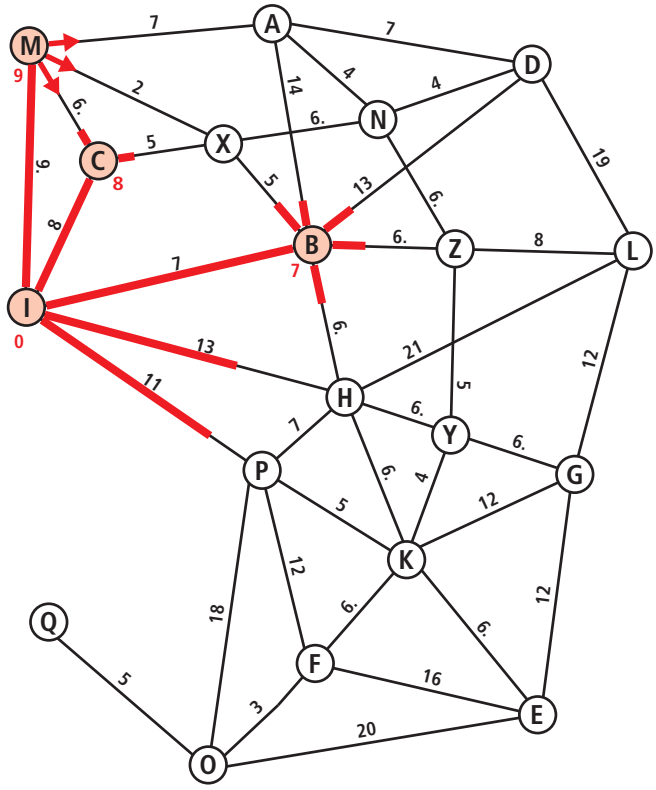


Abbildung 1.7
Drei weitere Wege werden von Morbach aus erkundet.

Richtig! Der Trupp von Ⓒ weiß, dass dieses Ziel bereits erreicht ist, die kürzeste Strecke dorthin also feststeht. Der Trupp, der von Ⓜ kommt, kann das Gleiche von seinem Ausgangspunkt berichten. Es ist also sinnlos, weiterzumarschieren. Die Strecke wird als „unbrauchbar" markiert, die Ameisen können wieder zu ihrem Stamm zurück, sie sind sozusagen aus dem Rennen. Abbildung 1.8 zeigt das gleich mit einem dicken roten Kreuz.

Als Nächstes kommen zwei Erkundungstrupps gleichzeitig an: In der 11. Minute erreichen die Ameisen Ⓟ und Ⓧ. Ⓟ erreichen sie auf direktem Wege von Ⓘ aus. Bei Ⓧ kommt der Trupp an, der über Ⓜ unterwegs ist (9 km bis Ⓜ plus 2 bis Ⓧ).

Wieder teilen sich die Ameisen auf. Von Ⓧ gibt es nur noch einen erfolgversprechenden Weg, bei den anderen treffen sie recht schnell auf Kameraden und geben die Strecke auf. Die von Ⓟ ausgehenden Touren sind alle noch offen. Abbildung 1.9 zeigt den aktuellen Stand.

Haben Sie das Prinzip verstanden?

Statt immer nur einen Weg auszuprobieren und wieder zu verwerfen, wenn sich ein besserer gefunden hat, erkunden die Ameisen gleichzeitig alle sich bietenden Möglichkeiten.

Kommen sie als Erste bei einer Stadt an, wissen sie, dass der genommene Weg der kürzeste ist, denn sonst wäre ja ein anderer Erkundungstrupp bereits da (zur Erinnerung: Alle bewegen sich mit der gleichen Geschwindigkeit.)

Treffen die Ameisen irgendwo auf Artgenossen, dann wissen sie, dass ihre Reise zu Ende ist, weil sie sich gegenseitig ein „Ich bin schon da" berichten können. Andere haben also das anvisierte Ziel früher erreicht.

Führen Sie zur Übung das Verfahren noch zu Ende und zeichnen Sie den Weg der Ameisen, die gefundenen Strecken sowie verworfene Wege in die Landkarte ein!

Können Sie sich vielleicht gleichzeitig auch schon vorstellen, wie ein Computer das Ameisenprinzip adaptiert?

Die Lösung wird in Abbildung 1.10 dargestellt. Haben Sie ein anderes Ergebnis? Kein Problem: Die Beispielaufgabe ist bewusst so gestellt, dass es zu manchen Orten unterschiedliche Wege mit gleicher kürzester Strecke gibt. So kommt man etwa zu Ⓗ sowohl über ⒾⒷⒽ als auch direkt über ⒾⒽ in jeweils 13 km. Falls Ihre Lösung jedoch Abweichungen in den rot eingetragenen Entfernungen aufweist, sollten Sie diese nochmal kontrollieren.

Welche Informationen haben wir dadurch jetzt eigentlich gewonnen?

Um von Imstadt zu einem beliebigen anderen Ort zu kommen, folgen Sie dem Pfad der Ameisen:

Von Imstadt nach Oppenheim kommt man so am günstigsten über Pappstadt, Krupsing und Flughafen. Die Gesamtstrecke beträgt 25 km. Ein günstigerer Weg existiert nicht!

Sie haben aber nicht nur die ursprüngliche Aufgabe gelöst, sondern sozusagen als Abfallprodukt noch die kürzesten Wege von Imstadt zu allen weiteren Städten ermittelt.

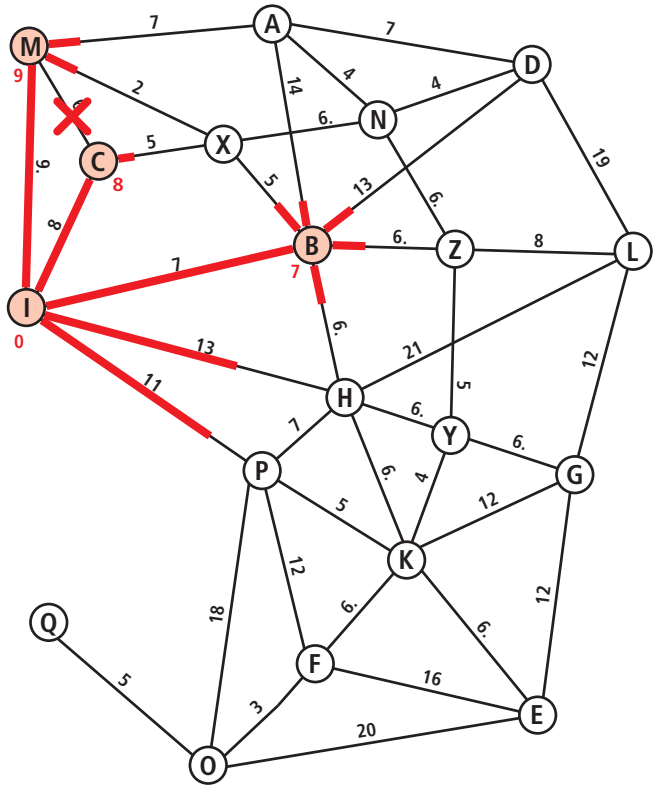


Abbildung 1.8
Der Weg zwischen Morbach und Chelzey wird als unbrauchbar markiert.

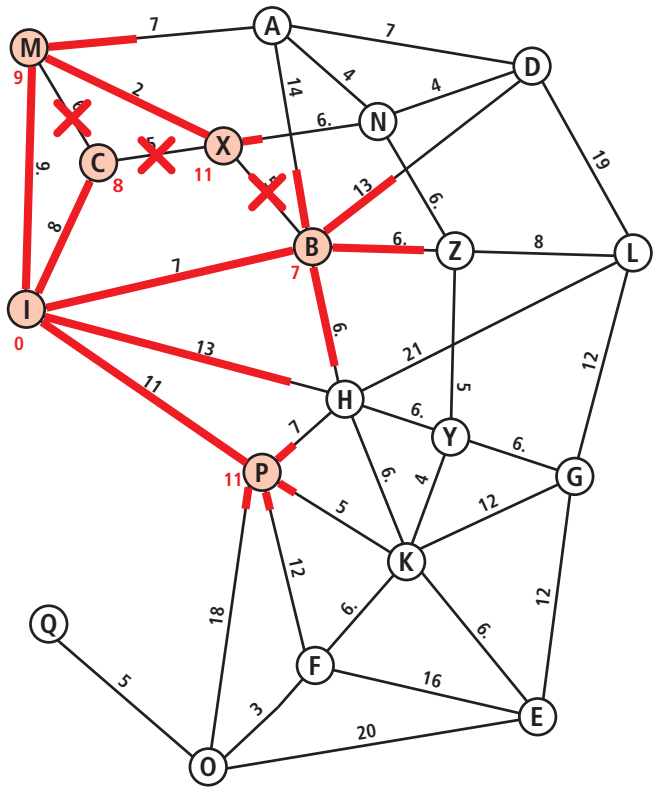
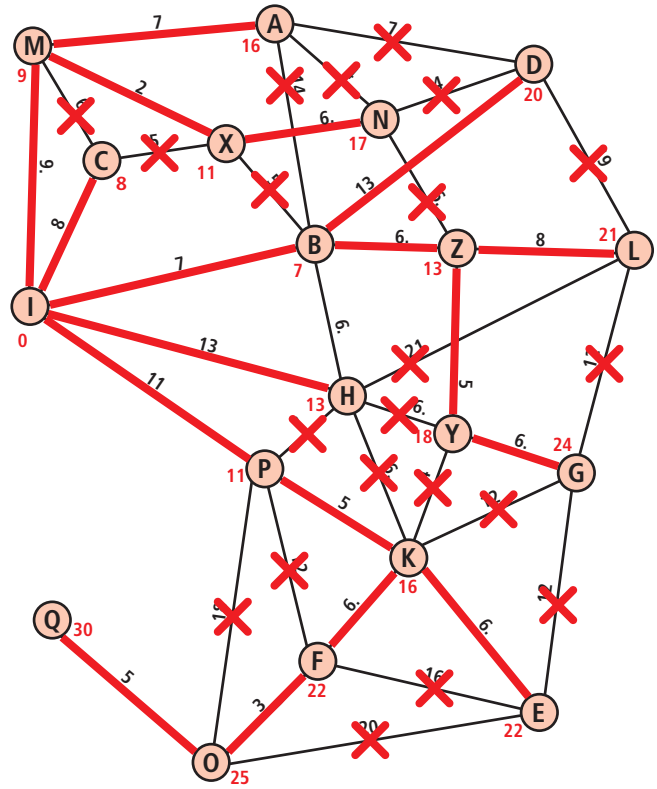


Abbildung 1.9
Weiterer Fortschritt der Ameisen

Abbildung 1.10
Die fertig von den Ameisen erkundete Landkarte



So fährt man zum Beispiel nach Giwelau am besten über Buding und die beiden Autobahnkreuze, die wir mit Ⓩ und Ⓨ bezeichnet haben.

Beachten Sie bitte auch die „0“, die nun der Vollständigkeit halber bei Imstadt steht, denn der Weg von Imstadt nach Imstadt beträgt ja ganz offensichtlich 0 km.

Warum ist das Ameisen-Prinzip für einen Informatiker interessant?

- Es führt in absehbarer Zeit zum Ziel. Da die Ameisen ständig in Bewegung sind und keine Strecke doppelt gehen, müssen sie recht bald alles erkundet haben (maximal nach der Zeit, die dem kürzesten Weg zur am weitesten entfernten Stadt entspricht).
- Es werden immer wieder die gleichen, sehr einfachen Anweisungen benutzt, um die Ameisen zu steuern:

  ❶ Teile den Trupp auf und folge allen Routen.

  ❷ Wenn ein Ort erreicht wird: kürzeste Strecke dorthin gefunden, weiter bei ❶.

  ❸ Wenn man einem anderen Trupp begegnet: Strecke verwerfen, Ende.

Solche Verfahren lassen sich (normalerweise) gut und einfach für den Computer umsetzen. Eine entsprechend formulierte Vorschrift nennt man dann Algorithmus.

Algorithmus

Ein Algorithmus ist eine Handlungsvorschrift zur Lösung eines Problems bzw. einer Kategorie von Problemen. Diese Handlungsvorschriften lassen sich im Allgemeinen in ein Computerprogramm umsetzen. Hierfür müssen sie hinreichend genau formuliert sein.

Wie lösen also unsere Routenplaner im Auto das Problem des kürzesten Weges? Enthalten sie einen Ameisen-Simulator und spielen die genaue Vorgehensweise nach?

Sicherlich wäre das vorteilhafter als die ganz zu Anfang beschriebene „Brute-Force-Methode". Trotzdem ist der Informatiker hier erneut gefragt, das gefundene Verfahren für den Computer zu optimieren.

Können Sie sich vorstellen, wie?

Das Ameisen-Prinzip liegt uns Menschen sehr, weil wir uns gut vorstellen können, wie die kleinen Krabbeltiere die Pfade erkunden. Dadurch erschließt sich auch recht deutlich, warum das Verfahren wirklich die günstigsten Wege hervorbringt.

Wann immer wir so ein „aus dem Leben gegriffenes" Verfahren im Computer benötigen und es daher in Bits und Bytes umsetzen, hilft wieder das Prinzip der Abstraktion:

Was ist vom Ameisen-Prinzip für die Problemlösung relevant? Bedenken Sie: Computer benötigen keine Anschauung, diese Anteile sind also zu eliminieren.

Ob Edsger W. Dijkstra, zuletzt Professor in Texas, 1959 über eine Ameisenstraße lief, ist unbekannt. Er hat jedoch bereits zu diesem Zeitpunkt ein Verfahren vorgestellt, das unser Ameisen-Prinzip im Computer zur Berechnung eines kürzesten Weges nutzt. Es ist bis heute nach ihm benannt.

Doch versuchen Sie zunächst einmal selbst, ein solches Verfahren herauszufinden. Vielleicht kommen Sie ja sogar auf eine bessere Methode, die dann nach Ihnen benannt wird.

Der Dijkstra-Algorithmus

Fangen wir noch einmal ganz von vorne an: Sie sitzen in Imstadt und wollen nach Oppenheim.

Die Ameisen unseres letzten Experiments liefen nun einfach in alle anderen direkt erreichbaren Städte, um zu ermitteln, wie lange sie dorthin unterwegs sind. Die hierfür notwendigen Zeiten braucht ein Computer jedoch gar nicht zu ermitteln, er liest einfach die Strecken ab. Wie aus Abbildung 1.11 ersichtlich, sind dies ja einfach die verzeichneten Längen der Verbindungslinien zwischen den Städten.

In der Graphik ist außerdem bei allen Zielpunkten markiert, woher diese kommen, damit die beschriebene Entfernung zutrifft.

Algorithmus

Wussten Sie, dass die Herkunft dieses Begriffes nie richtig geklärt wurde? Entweder kommt er vom griechischen „arithmos", was Zahl bedeutet, oder er ist der veränderte Name des persisch-arabischen Mathematikers „Al-Charismi", der – englisch ausgesprochen – in etwa wie „algorithm" klingt. Erste Algorithmen wurden übrigens bereits lange vor der Zeitenwende hauptsächlich von griechischen Mathematikern entwickelt: Etwa im 4. Jahrhundert v. Chr. das euklidische Verfahren zur Bestimmung des größten gemeinsamen Teilers zweier Zahlen oder im 3. Jahrhundert v. Chr. das Sieb des Eratosthenes zur Ermittlung von Primzahlen. Beide Algorithmen werden heute noch in Computern eingesetzt!

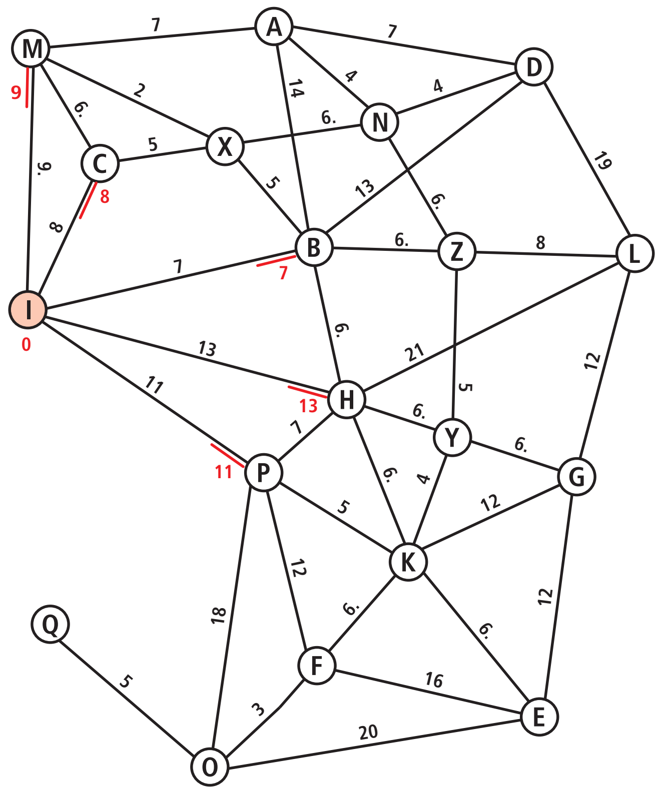


Abbildung 1.11
Potentiell vom Start in Imstadt erreichbare Städte und die Weglängen dorthin

Wie ging es mit den Ameisen weiter? Der Trupp, der zuerst bei einer Stadt ankam, markierte die genommene Strecke als „günstig“ und verteilte sich auf die umliegenden Strecken.

Für den Computer ist nun gar kein Problem, diese Stadt zu bestimmen: Es ist diejenige, die mit der kleinsten Zahl bezeichnet ist, also Ⓑ mit der Zahl 7. Abbildung 1.12 zeigt das, der Knoten ist mit roter Farbe als „besucht“ markiert.

Von dieser Stadt aus werden wiederum die Entfernungen zu allen Nachbarn bestimmt. Da die Ameisen jedoch bis nach Ⓑ bereits 7 km unterwegs waren, müssen diese addiert werden. Versuchen Sie es! Stoßen Sie auf ein Problem?

Genau! Für Ⓧ, Ⓐ, Ⓓ und Ⓩ können wir die Methodik einfach durchführen, aber was ist mit Ⓗ? Hier steht bereits ein Wert! Wie ist hier zu verfahren?

Ziehen wir die Ameisen zurate: Der Weg von Ⓘ über Ⓑ nach Ⓗ ist 13 km lang, der direkte Weg von Ⓘ nach Ⓗ ebenfalls 13 km. Die mögliche neue Strecke ist zwar nicht schlechter, aber auch nicht besser – wir können die alte Markierung behalten.

Daher gilt für das Dijkstra-Verfahren die folgende Regel für den Fall, dass an einer Nachbarstadt bereits eine Zahl steht:

- Wenn die Zahl, die neu hingeschrieben werden soll, kleiner ist, dann wird die alte durch die neue Zahl ersetzt und dementsprechend auch die Marke, woher die Ameisen kommen.
- Wenn die neue Zahl größer oder gleich ist, passiert gar nichts.

Ergebnis ist daher die Karte aus Abbildung 1.13.

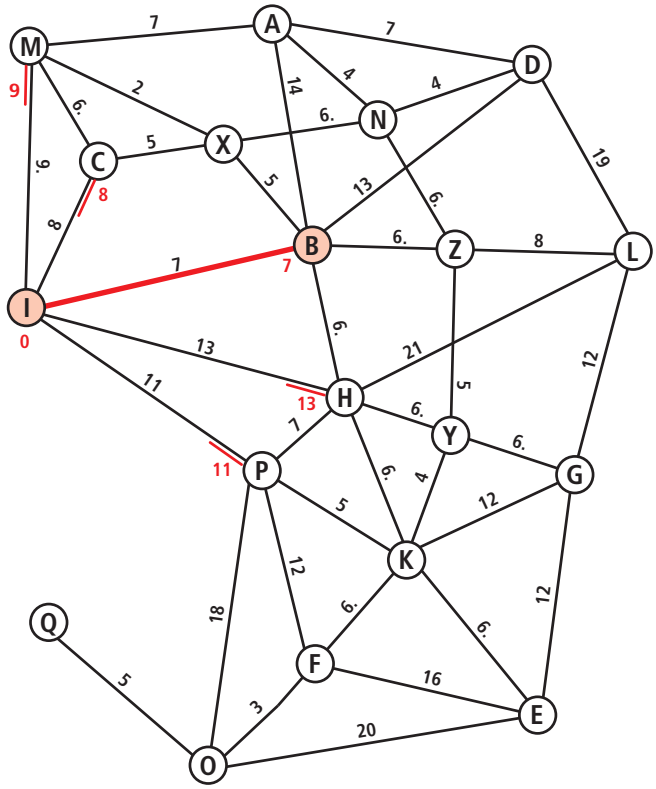


Abbildung 1.12
Der erste kürzeste Weg von Imstadt aus wurde ermittelt.

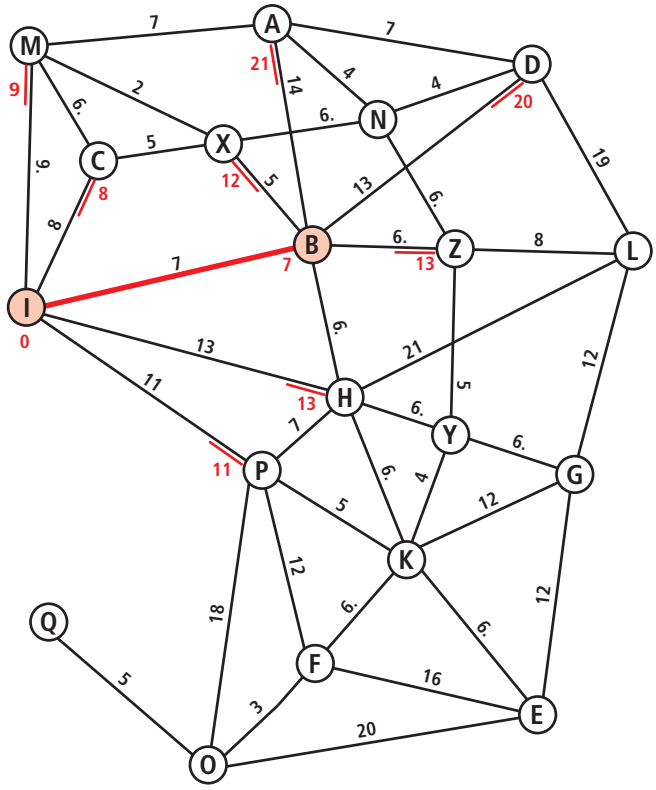


Abbildung 1.13
Erreichbare Städte sind durch eine rote Zahl mit der gefundenen Weglänge markiert.

Wie geht es nun weiter?

Prinzipiell wie am Anfang: Aus allen mit Zahlen markierten Städten, die noch nicht von Ameisen besucht wurden (noch nicht rot gefärbt), wird die mit der kleinsten Zahl herausgesucht. Dort kommen die Ameisen als Nächstes an. In diesem Fall ist das Ⓒ.

Von Ⓒ aus werden wieder alle benachbarten Städte betrachtet. Nach Ⓜ käme man in 14 km, nach Ⓧ nach 13 km. Beides wird jedoch von der bereits vorhandenen Zahl unterboten, also passiert gar nichts weiter, die nächste nichtmarkierte Stadt mit der kleinsten Zahl wird gesucht. Das Ergebnis sehen Sie in Abbildung 1.14.

Lassen Sie uns noch einen weiteren Schritt detailliert zusammen unternehmen: Der noch nicht rot markierte Knoten mit der kleinsten roten Zahl ist Ⓜ. Diesen erreichen wir also als Nächstes. In der Karte wird er zusammen mit der Straße zu ihm rot eingefärbt, die Entfernung von 9 km ist damit fest ermittelt.

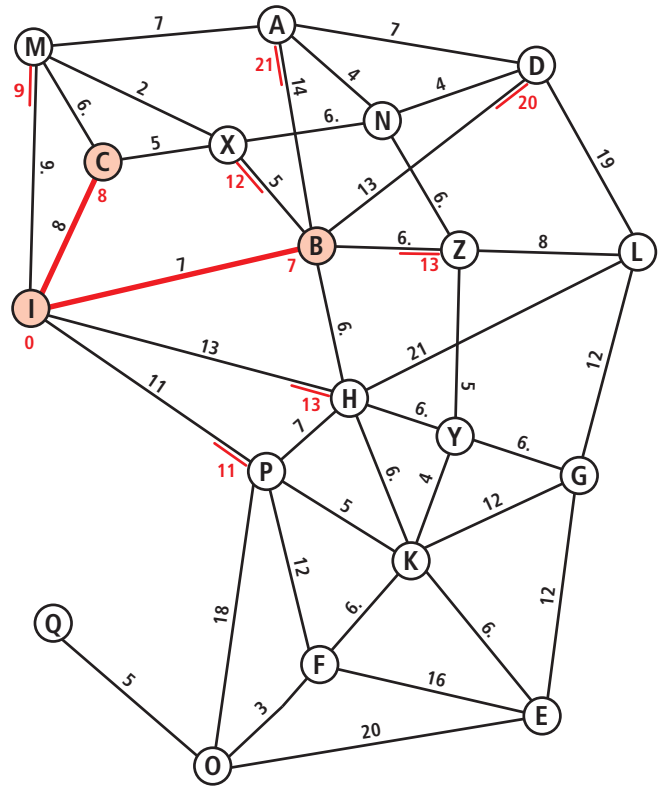
Nun betrachten wir die Städte, die von Ⓜ aus erreichbar sid. Ⓒ und Ⓘ sind bereits markiert – hier waren wir also schon. Kandidaten sind demnach Ⓐ und Ⓧ. Tatsächlich lassen sich die Wegstrecken gegenüber den bereits angenommenen verbessern, indem wir über Ⓜ gehen.

Abbildung 1.15 zeigt das – die alten Wegmarkierungen sind hier einfach ausgestrichen.

Bevor Sie nun weitermachen, versuchen Sie doch einmal, den Algorithmus zur Bestimmung kürzester Wege aufzuschreiben.

Algorithmen spielen in der Informatik eine sehr große Rolle: Ein Großteil der vorkommenden Aufgaben aus Wirtschaft und Wissenschaft lässt sich mit Algorithmen

Abbildung 1.14
Erneut wurde ein kürzester Weg gefunden.



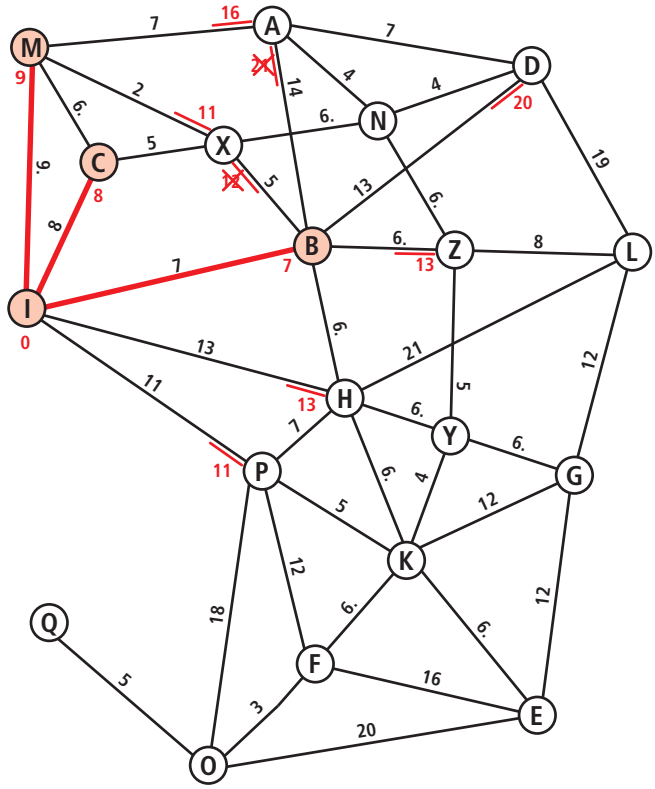


Abbildung 1.15
Im Verlauf werden alle günstigeren Wege gefunden, bevor eine Stadt rot markiert wird.

lösen, die schon vor längerer Zeit entwickelt wurden (dazu mehr am Ende des Kapitels).

Es gibt auch diverse formale Methoden, die Algorithmen zu Papier zu bringen. Eigentlich geht es jedoch darum, die Beschreibung so genau zu machen, dass jemand anderes den Algorithmus danach durchführen kann.

Behalten Sie das im Hinterkopf, wenn Sie jetzt versuchen, den beschriebenen Dijkstra-Algorithmus insgesamt zu formulieren.

Es gibt natürlich unzählige korrekte Lösungen. Ihre sollte im Kern mit Abbildung 1.16 übereinstimmen.

An einem Blockdiagramm wie diesem erkennt man besonders gut Wiederholungen von Anweisungsfolgen. Es ist aber genauso gut, einfach Text zu schreiben oder verschiedene Anweisungen mit Pfeilen zu verbinden. Alles ist erlaubt, was Ihnen ermöglicht, sich strukturiert Gedanken über den Ablauf zu machen. Wenn das Ergebnis dann auch noch für andere lesbar und verständlich ist – umso besser!

Führen Sie nun den Algorithmus für das gegebene Beispiel fort.

Abbildung 1.16
Algorithmus als Blockdiagramm

0. Markiere die **Startstadt** rot, weise ihr die Kennzahl 0 zu. Bezeichne diese als **aktuelle Stadt**.

1. Gehe aus von der **aktuellen Stadt** zu allen direkt erreichbaren **Nachbarstädten** ...

... und führe das Folgende für jede **Nachbarstadt** durch, die noch nicht rot markiert ist:

Errechne die **Summe** aus der roten Kennzahl an der **aktuellen Stadt** und der Streckenlänge zur **Nachbarstadt**.

- Hat die **Nachbarstadt** keine **Kennzahl**, weise ihr die **Summe** als Kennzahl zu. Markiere die Strecke zur **aktuellen Stadt**.
- Hat die **Nachbarstadt** eine **Kennzahl** kleiner oder gleich der **Summe**, mache nichts.
- Hat die **Nachbarstadt** eine **Kennzahl** größer der **Summe**, streiche die dortige **Kennzahl** sowie die Markierung. Weise ihr danach die **Summe** als neue **Kennzahl** zu. Markiere die Strecke zur **aktuellen Stadt**.

2. Betrachte alle Städte, die zwar eine rote **Kennzahl** haben, aber nicht rot markiert sind. Suche die mit der kleinsten **Kennzahl**.

3. Bezeichne diese als **aktuelle Stadt**. Weisen mehrere Städte die kleinste Kennzahl auf, wähle eine beliebige davon.

4. Markiere die **aktuelle Stadt** rot, zeichne die dort markierte Strecke komplett rot nach.

5. Falls die **Zielstadt** noch nicht rot markiert ist, weiter bei (1.)

Edsger Wybe Dijkstra (1930 – 2002) arbeitete nach der Ausbildung zunächst als Professor an der Universität in Eindhoven und wechselte 1984 nach Texas. Neben seinen Arbeiten zur Berechnung des kürzesten Weges hat er auch einen wesentlichen Beitrag zur Einführung der strukturierten Programmierung geleistet. Dafür erhielt er 1972 auch den Turing-Preis, der das Pendant des Nobelpreises für Informatiker ist.

Das Ergebnis der Weiterführung sehen Sie in Abbildung 1.17. Erkennen Sie die Gemeinsamkeiten mit dem Ergebnis bei Nutzung des Ameisen-Prinzips?

Bemerken Sie, dass alle Städte bis auf Ⓠ rot markiert sind? In unserer Algorithmusbeschreibung haben wir festgelegt, dass wir aufhören dürfen, wenn die Zielstadt erreicht ist. Auf diese Weise sparen wir uns eventuell, einige Städte zu betrachten. Und auch wenn trotzdem fast alle Städte rot markiert sind, hat die Durchführung nicht allzu viel Zeit in Anspruch genommen.

Üben Sie nun mit dem neuen Algorithmus und bestimmen Sie den kürzesten Weg von Lupera nach Eindhofen. Wie wäre es, wenn Sie dafür die genaueren Wegstreckenangaben nutzen, die am Ende des Kapitels in Abbildung 1.K2 als Kopiervorlage zu finden sind?

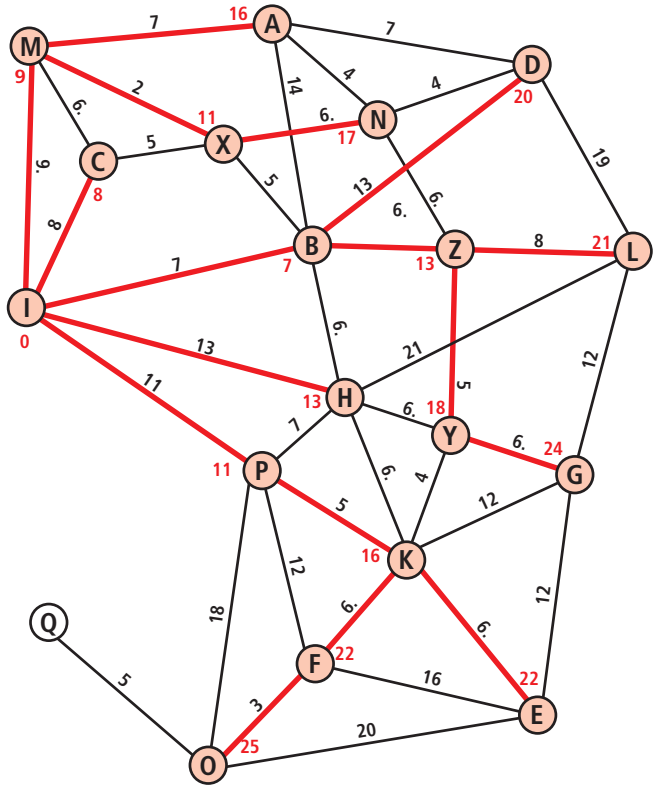


Abbildung 1.17
Fertige Landkarte nach Durchlauf des Dijkstra-Algorithmus

Wenn Sie alles richtig gemacht haben, sollte in etwa Abbildung 1.18 herauskommen. Die durchgestrichenen roten Zahlen und Markierungen sind absichtlich noch sichtbar, um den Verlauf besser nachvollziehen zu können. Ein Routenplaner würde Ihnen die Strecke als wahrscheinlich präsentieren wie in Abbildung 1.19.

Weil es ab und zu Verwechslungen gibt: Bitte denken Sie daran, dass das dargestellte rote Netz nur die kürzesten Wege von Lupera aus repräsentiert! Wenn Sie es zum Beispiel nutzen würden, um über rote Strecken von Imstadt nach Fluxing zu gelangen, wäre dieser Weg über Budingen, die Kreuze Ⓩ und Ⓨ sowie Krupsing mit 27,4 km viel zu weit. Aus Abbildung 1.K2 können Sie ablesen, dass dies auch in 21,7 km zu schaffen ist.

Dijkstra bestimmt immer die kürzesten Wege von einem Startpunkt zu vielen anderen Punkten, ist aber nicht geeignet, um daraus Schlüsse auf die Situation mit anderen Startpunkten zu ziehen.

Herzlichen Glückwunsch – Sie wissen nun, wie so ein Routenplaner funktioniert! Da Sie sicher ganz heiß darauf sind, Ihr neues Wissen gleich praktisch anzuwenden, hier gleich die nächste Aufgabenstellung ...

Abbildung 1.18
In 22,1 km kommt man von Lupera nach Eindhofen.

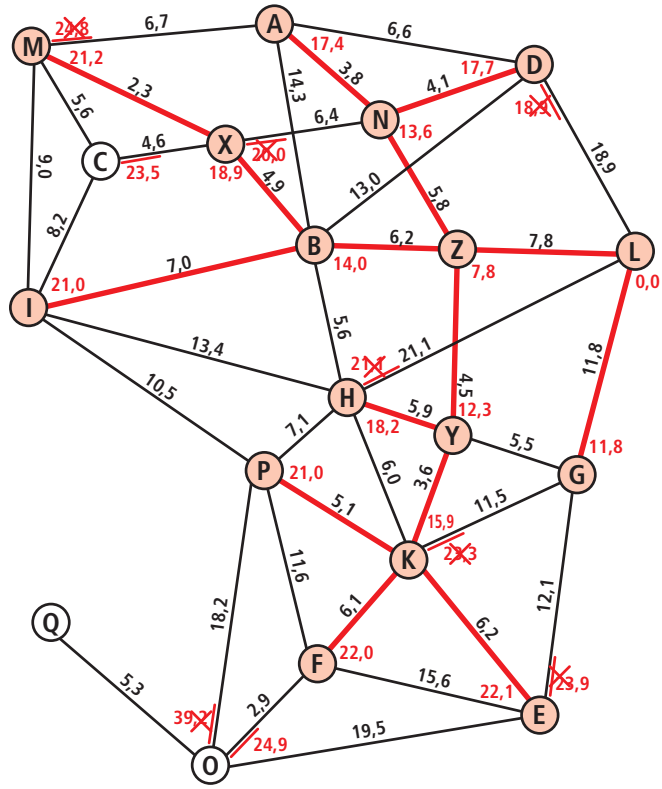
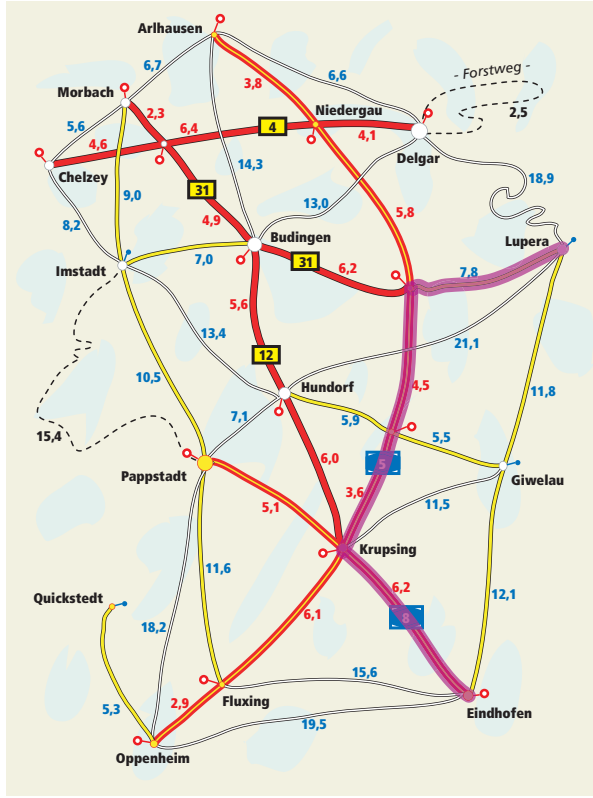


Abbildung 1.19
Ein Navigationsgerät würde das Ergebnis eventuell auf diese Weise visualisieren.



Die Schilda-Rallye

Sehen Sie hier in Abbildung 1.20 den Touristen-Stadtplan von Schilda, mit seinen acht Hotels und vier Pensionen.

Wie es so kommt, hat der Stadtrat vor Kurzem beschlossen, alle Straßen zu Einbahnstraßen zu machen (ganz im Gegensatz zu den traditionellen Gepflogenheiten sind trotzdem noch alle Orte erreichbar).

Die Fahrzeuge von Schilda-Taxi warten vor den Hotels Adler und Gozo sowie auf dem Parkplatz der Pension Kapitol. Aufgrund der neuen Verkehrsführung benötigen die Fahrer einen Plan, wie sie auf dem kürzesten Weg von ihrem Standort zu allen anderen Hotels kommen. Eine Entfernungstabelle ist auch zur Berechnung der neuen Tarife notwendig.

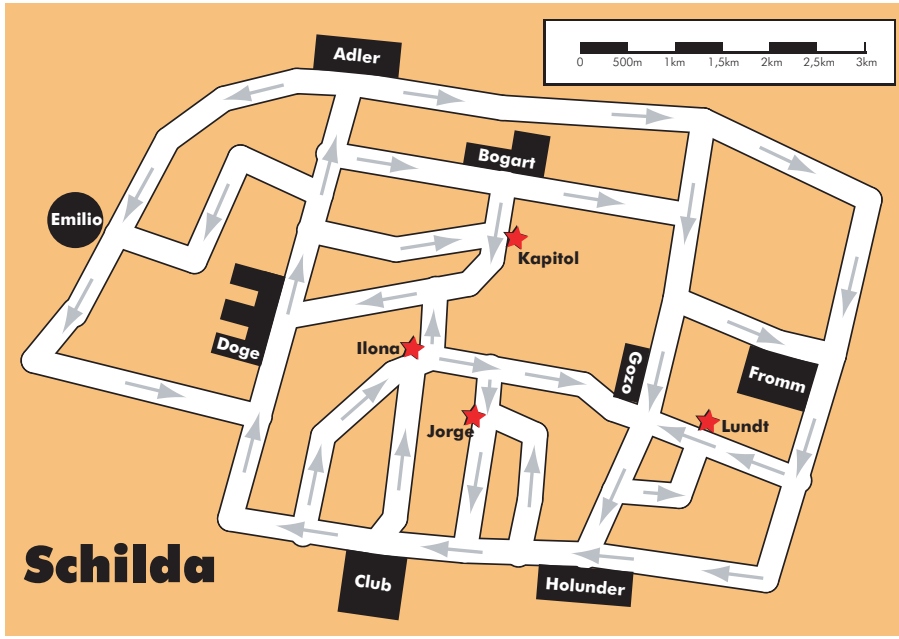


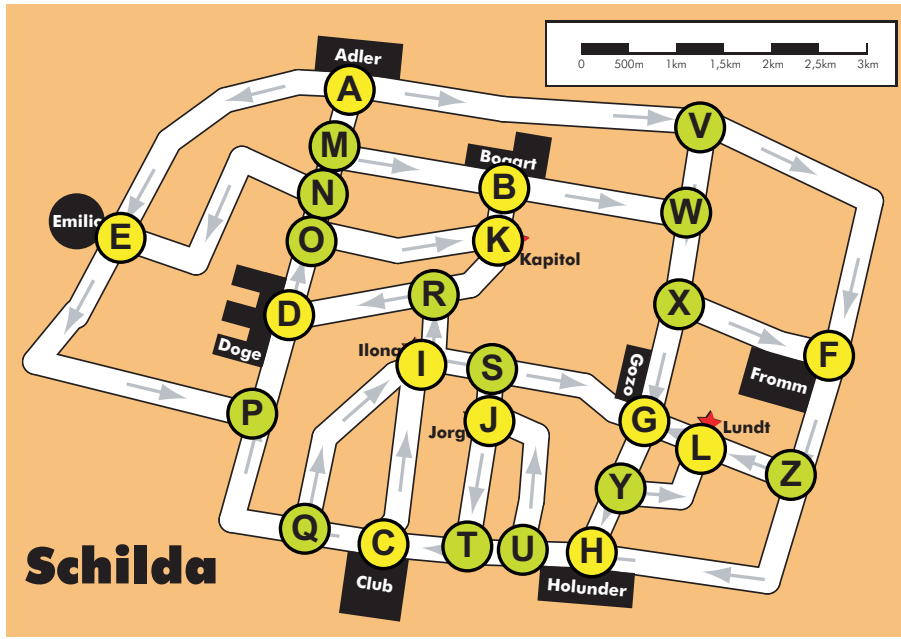
Abbildung 1.20
Stadtplan von Schilda

Ihre Aufgabe: Erstellen Sie drei Tabellen mit den Entfernungen der Hotels Adler und Gozo sowie der Pension Kapitol zu allen anderen Gästehäusern.

Tipp: Gehen Sie zunächst genauso vor, wie Sie es gerade gelernt haben. Einen Unterschied gibt es allerdings. Erkennen Sie ihn? Versuchen Sie das Problem so weit zu lösen, wie Sie selbst kommen. Wenn es wirklich nicht mehr weitergeht, schauen Sie in die hier vorgeschlagene Lösung.

Zunächst wird wieder eine Abstraktion benötigt. Die Definition von Orten ist der erste Schritt. Selbstverständlich sind die Positionen aller Hotels relevante Orte. Relevant sind jedoch auch alle Orte, an denen sich die Straße teilt, also ein Abbiegen möglich ist.

Abbildung 1.21
Stadtplan von Schilda mit markierten Straßenkreuzungen



In dieser Lösung nach Abbildung 1.21 habe ich die Orte für die Hotels (ihrem Namen entsprechend) Ⓐ bis Ⓛ genannt und in der Karte gelb markiert, Straßenkreuzungen ohne Hotel Ⓜ bis Ⓩ mit einer hellgrünen Markierung. Andere Nomenklaturen sind selbstverständlich genauso möglich.

Haben Sie weitere Biegungen als „Orte" markiert, etwa zwischen Ⓔ und Ⓟ oder zwischen Ⓥ und Ⓕ? Das ist kein Fehler, jedoch nicht notwendig, denn ein Auto kann sowieso nur dem Straßenverlauf folgen.

Welches ist der nächste Schritt? Stellen Sie einen Plan auf, ähnlich dem Plan für das letzte Routenplaner-Problem!

Der Unterschied zum letzten Problem sind die Einbahnstraßen – diese wichtige Information muss auch in einer abstrakteren Darstellung der Karte erfasst werden, aber das stellt ja kein Hindernis dar. Abbildung 1.22 auf der nächsten Doppelseite zeigt die Lösung.

Erinnern Sie sich noch an das Prinzip der Ameisen? Es lässt sich immer noch anwenden – nur müssen die Ameisen nun die Regeln der Einbahnstraßen beachten, also immer nur in Pfeilrichtung laufen. Für den Dijkstra-Algorithmus bedeutet dies, dass „Nachbarorte" immer nur solche sind, die in Pfeilrichtung erreichbar sind.

Fangen Sie damit an, die Wege-Tabelle vom Hotel Adler aus zu erstellen!

Sie sind ja ganz sicher selbst auf die Lösung in Abbildung 1.23 auf der nächsten Doppelseite gekommen ...

Üben Sie nun weiter, indem Sie die Entfernungen vom Hotel Gozo zu jedem anderen Hotel und von der Pension Kapitol aus bestimmen.

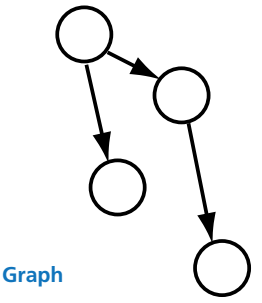
Die Lösung für das Hotel Gozo entnehmen Sie Abbildung 1.24.

Die Entfernungen von der Pension Kapitol aus entnehmen Sie Abbildung 1.25.

Auch diese Abbildungen finden Sie auf der nächsten Doppelseite – als kleiner Schutz vor dem „Spicken".

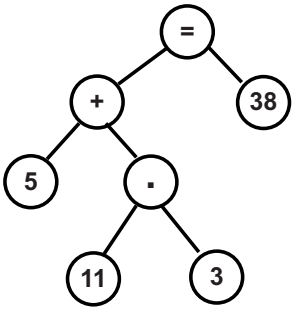
Was steckt dahinter?

Die besprochenen Probleme gehören in das Umfeld der sogenannten Graphenalgorithmen. Ein Graph besteht hierbei aus einer Menge von Knoten und einer Menge von Kanten, die zwischen den Knoten verlaufen. Graphen werden oft wie in diesem Beispiel dargestellt: die Knoten als Kringel und die Kanten als Linien oder Pfeile dazwischen. Man unterscheidet ungerichtete Graphen, also solche, bei denen die Verbindung zweier Knoten in beide Richtungen geht (ohne Pfeil), und gerichtete Graphen (mit Pfeil).



Graph

Wozu ist aber ein Graph gut? Wie andere Modelle der Informatik auch kann der Graph einen Ausschnitt der Wirklichkeit modellieren, um diese einfacher zu verstehen und zu analysieren. Prominentes Beispiel für einen Graphen ist die Landkarte bzw. der Stadtplan, wie hier im Kapitel gezeigt. Knoten sind hierbei die Orte, Kanten die Verbindungswege. Genauso könnte der Graph jedoch auch ein Computernetzwerk darstellen. Auch andere abstrakte Konzepte wie Produktionspläne oder eine mathematische Gleichung lässt sich per Graph sehr übersichtlich darstellen – in diesem Fall $5 + (11 \cdot 3) = 38$.



Mathematische Gleichung

Sowohl Knoten als auch Kanten können Markierungen tragen. Im Beispiel der Landkarte sind das Ortsnamen für die Knoten und Weglängen für die Kanten. Aufgrund der vielfältigen Anwendungen von Graphen gibt es auch eine Menge von Algorithmen, die auf Graphen arbeiten und Aufgaben – in der Informatik nennen wir sie oft „Probleme" – mit Hilfe einer übersichtlichen Darstellung als Graph lösen. Der besprochene Dijkstra-Algorithmus ist hier ein sehr prominenter Vertreter.

Was zeichnet einen guten Algorithmus aus? Einerseits muss er natürlich die gestellte Aufgabe lösen. Wichtiges Kriterium ist jedoch außerdem, dass er die gestellte Aufgabe schnell löst und so auch bei großen Problemen nicht in die Knie geht.

Man rechnet daher aus, wie stark der Zeitbedarf mit der sogenannten Problemgröße ansteigt. Auch dies kann man am Beispiel der Landkarte sehr schön demonstrieren.

Eine brauchbare Problemgröße ist die Anzahl der Städte auf der Landkarte.

Rekapitulieren wir zunächst den anfänglich kurz erwähnten Brute-Force-Algorithmus zur Bestimmung des kürzesten Weges:

„Bestimme alle möglichen Wege vom Start zum Ziel und suche davon den kürzesten."

Im schlechtesten Fall müssen also von einem Graphen alle von einem Punkt ausgehenden Wege bestimmt werden. Außerdem sind im schlechtesten Fall alle Knoten mit

Abbildung 1.22
Stadtplan von Schilda

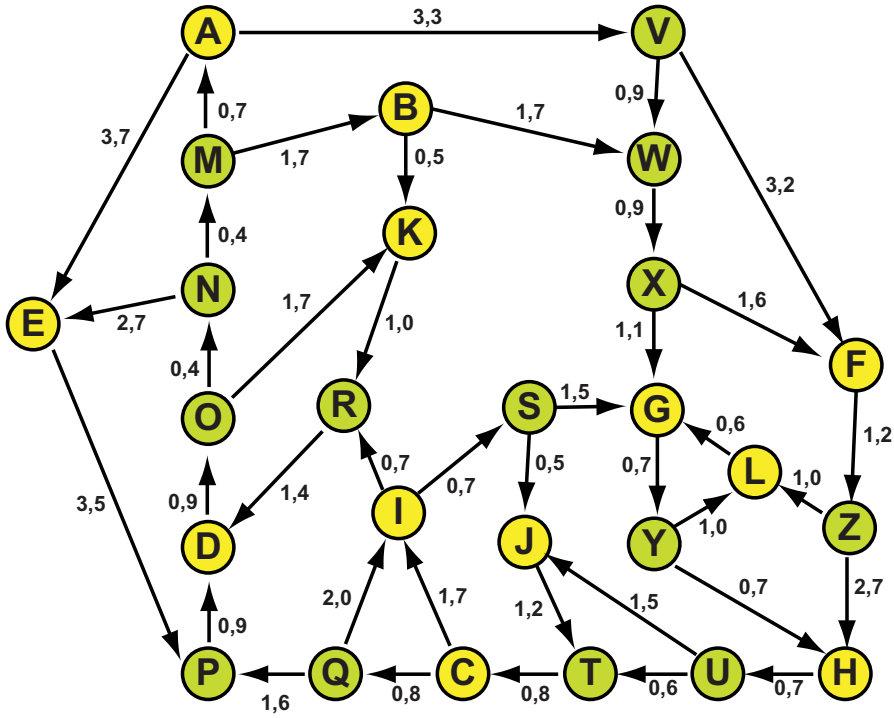
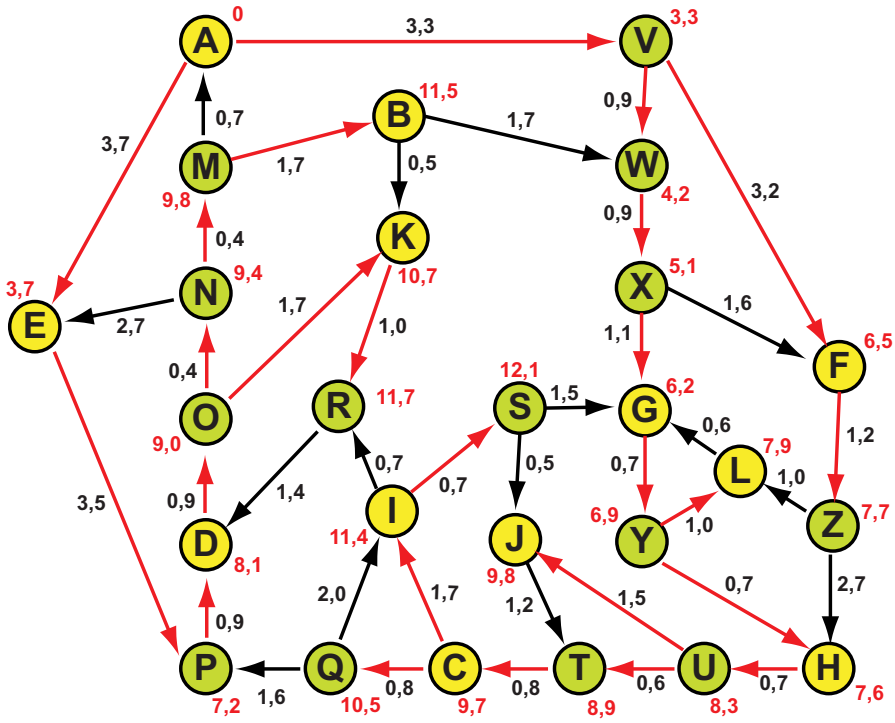


Abbildung 1.23
Kürzeste Entfernungen vom Hotel Adler. Sie ergeben folgende Tabelle:

Hotel	Entfernung
Adler	0,0
Bogart	11,5
Club	9,7
Doge	8,1
Emilio	3,7
Fromm	6,5
Gozo	6,2
Holunder	7,6
Ilona	11,4
Jorge	9,8
Kapitol	10,7
Lundt	7,9



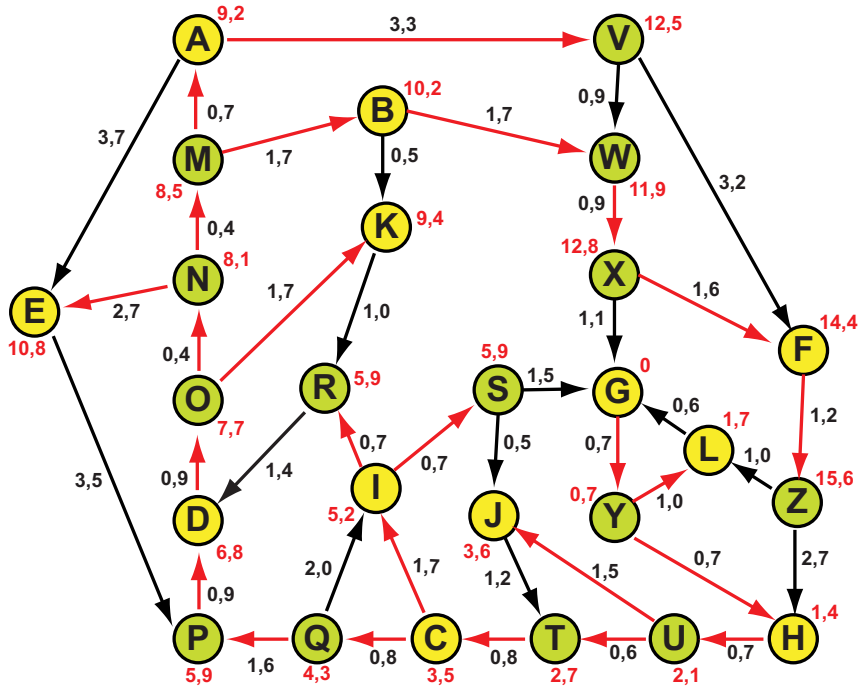


Abbildung 1.24
Kürzeste Wege vom Hotel Gozo. Sie ergeben folgende Tabelle:

Hotel	Entfernung
Adler	9,2
Bogart	10,2
Club	3,5
Doge	6,8
Emilio	10,8
Fromm	14,4
Gozo	0,0
Holunder	1,4
Ilona	5,2
Jorge	3,6
Kapitol	9,4
Lundt	1,7

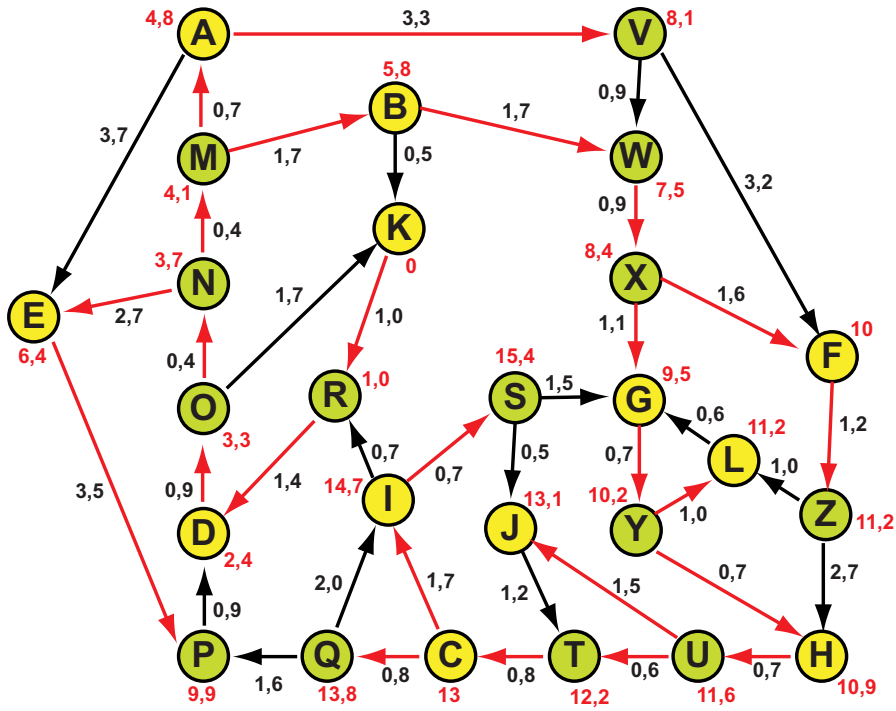


Abbildung 1.25
Kürzeste Wege vom Hotel Kapitol. Sie ergeben folgende Tabelle:

Hotel	Entfernung
Adler	4,8
Bogart	5,8
Club	13
Doge	2,4
Emilio	6,4
Fromm	10,0
Gozo	9,5
Holunder	10,9
Ilona	14,7
Jorge	13,1
Kapitol	0,0
Lundt	11,2

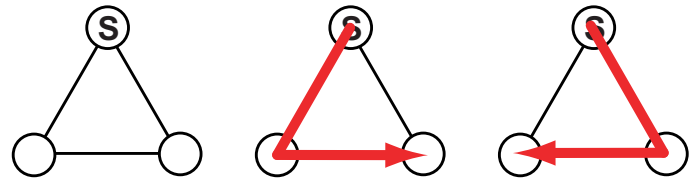


Abbildung 1.26
Von S gehen zwei unterschiedliche Wege aus.

allen anderen Knoten verbunden (der Fachausdruck dafür ist „vollständiger Graph“). Für drei Knoten ist es noch kein Problem, die Anzahl möglicher Wege vom Startpunkt S aus zu bestimmen. Abbildung 1.26 zeigt die zwei Lösungen.

Versuchen Sie das jetzt auch mit den Graphen aus Abbildung 1.27 mit vier bis sieben Knoten.

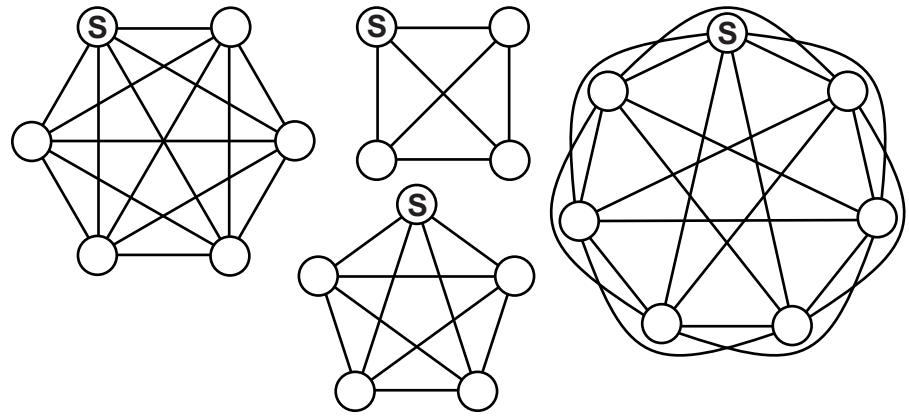


Abbildung 1.27
Vollständige Graphen mit vier bis sieben Knoten. Bei einem vollständigen Graphen ist jeder Knoten mit jedem anderen durch eine Kante verbunden.

Knoten	Schritte
3	4
4	18
5	96
6	600
7	4.320
8	35.280
9	322.560
10	3.265.920
15	1.220. 496.076.800
20	2.311.256.907. 767.808.000
50	29.805.811.337. 679.110.482. 740.356.002. 743.473.467. 490.088.737. 581.301.760. 000.000.000

Falls Sie angefangen haben, die Lösung zeichnerisch zu bestimmen, werden Sie bereits festgestellt haben, wie viel länger die Suche dauert, wenn auch nur ein Knoten hinzukommt. Man kann die Anzahl der Möglichkeiten auch rechnerisch bestimmen:

Für den Graphen mit vier Knoten gilt, dass man ihn aus zwei Komponenten zusammensetzen kann: einem einzelnen Knoten Ⓢ plus einem Graphen mit drei Knoten. ◯ kann dann mit dem Graphen verbunden werden, wie in Abbildung 1.28 dargestellt.

Da der „bekannte“ Graph drei Knoten besitzt, kann vom „neuen Knoten Ⓢ“ auf drei Arten ein Weg zum bekannten Graphen begonnen werden, nämlich jeweils zu einem der Knoten. Stellen Sie sich vor, dass im 3er-Graphen dann auf die bereits ermittelte Weise ein Weg gesucht wird.

Daher ist die Anzahl möglicher Wege im 4er-Graphen $3 \cdot 2 = 6$. Für den 5er-Graphen gibt es nach dem gleichen Schema vier Möglichkeiten, Wege vom neuen Knoten zum 4er-Graphen zu beginnen. Die Anzahl der Wege insgesamt beträgt daher $4 \cdot 6 = 24$. Auf diese Weise kann man ableiten, dass in einem vollständigen Graphen mit n Knoten $(n-1)!$ verschiedene Wege von einem gesetzten Startpunkt ausgehen.

Da für jeden der Wege $n-1$ Streckenabschnitte eingerechnet werden müssen, bedarf es für den Brute-Force-Algorithmus ungefähr $(n-1) \cdot (n-1)!$ Berechnungen, um den kürzesten Weg zu finden. Die nebenstehende Tabelle enthält die Anzahlen für ein paar Problemgrößen.

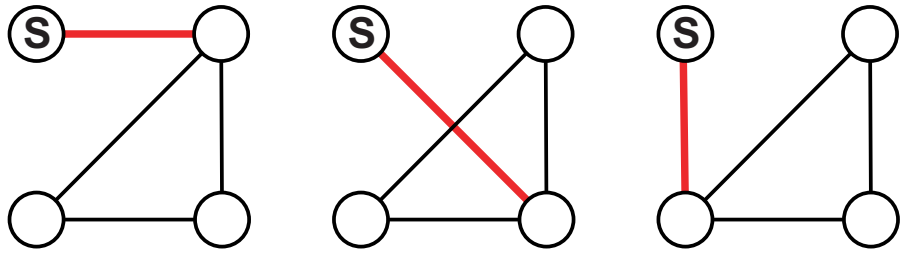


Abbildung 1.28
Der Startknoten S kann auf drei Arten mit dem vorhandenen Graphen verbunden werden.

Für 100 Knoten sind im schlimmsten Fall bereits

92.392.953.289.504.711.154.882.246.467.704.033.485.808.808.581.737.805.253.907.034.256.265.423.993.297.616.452.852.049.336.394.953.103.391.160.941.618.951.520.668.673.358.807.695.360.000.000.000.000.000.000.000

Berechnungen vonnöten. Um das einzuschätzen: Ein Googol wurde bereits in der ersten Hälfte des 20. Jahrhunderts definiert als Obergrenze für Zahlen, die natürliche Vorgänge beschreiben, denn mit 10^{100} ist ein Googol größer als die geschätzte Zahl von Elementarteilchen im gesamten Universum. Für die 100 Knoten kommen wir immer noch auf

9.239.295.328.950.471.115.488.224.646.770.403.348.580.880.858.173.780.525.390 Googol

oder 9.239 Nonillionen Googol Berechnungen.

Wir können das aber noch weiter auf die Spitze treiben: Wenn auch nur alle 12.903 Gemeinden Deutschlands (Stand: Ende 2003) als Knoten in die Suche einbezogen werden, ergibt sich für die Anzahl der Berechnungen eine unglaubliche Zahl, die ich Ihnen nicht in ihrer vollen Schönheit vorenthalten möchte. Sehen Sie selbst in Abbildung 1.29.

Anders ausgedrückt, handelt es sich um etwa $9{,}88 \cdot 10^{47438}$.

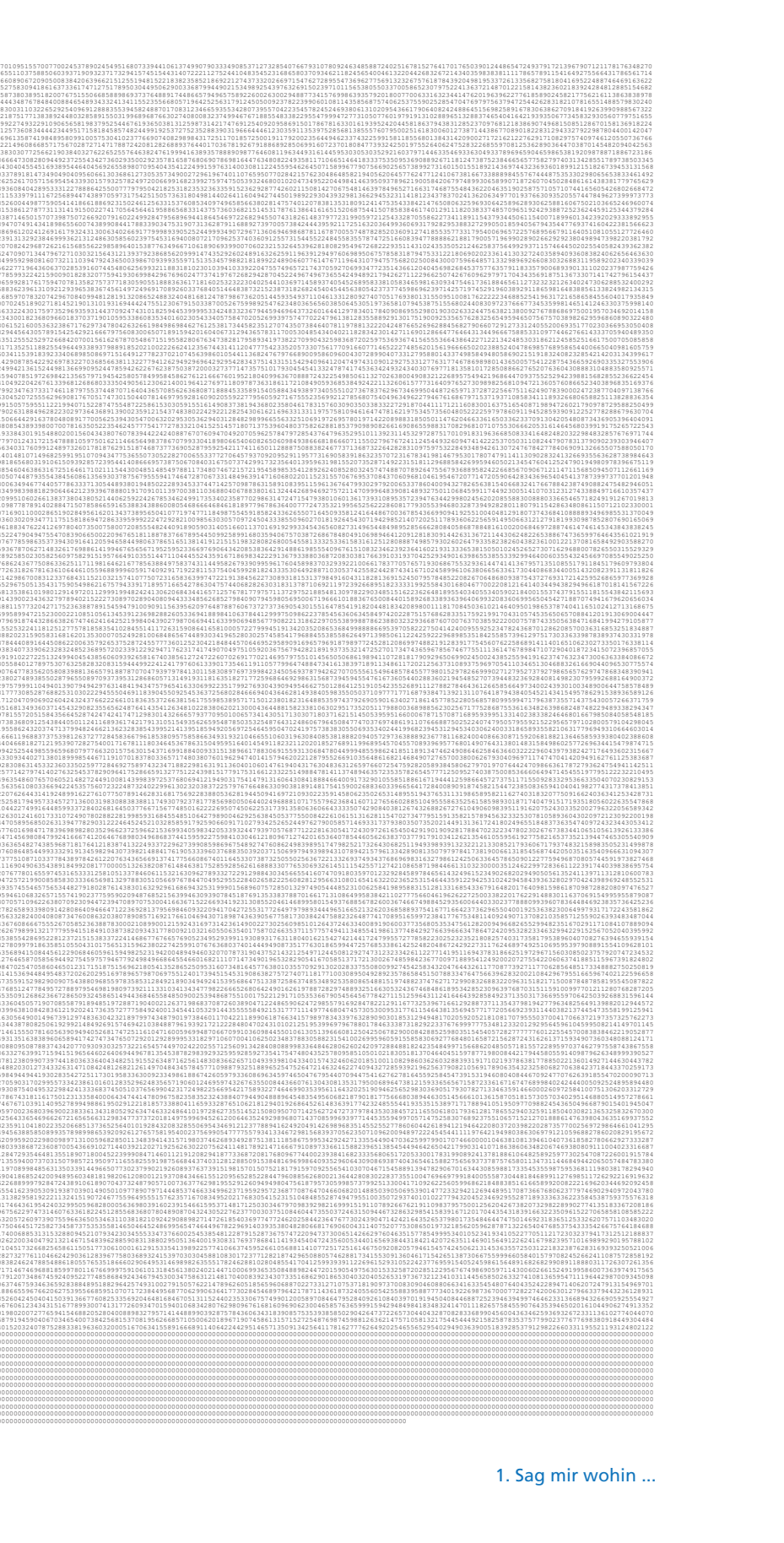
Wenn wir diese Zahl in der normalen Schriftart hier abgedruckt hätten, würde sie 14 Seiten vollständig eng beschrieben füllen. Um sich diese Summe wenigstens einigermaßen vorstellen zu können, möchte ich sie mit einer Analogie verdeutlichen.

Wenn sich die verfügbaren Rechenleistungen wie momentan jedes Jahr fast verdoppeln, haben im Jahre 2020 alle Rechner der Erde zusammengenommen etwa die Möglichkeit, 128 Exaflops durchzuführen, also 128 Trillion Fließkommaoperationen pro Sekunde. Nehmen wir an, wir vereinigen alle Rechner zu einer gigantischen Maschine, die sich dann um unser Wegeproblem kümmert. Dann würde sie immer noch etwa $4 \cdot 10^{47412}$ Jahre für das Ergebnis benötigen.

Das Alter des Universums wird heute auf etwa 12 Milliarden Jahre geschätzt. Die zusammengezogene Macht der Rechner würde also immer noch etwa $333 \cdot 10^{47397}$ mal so lange benötigen, wie das gesamte Universum alt ist – ausgedrückt mit der Einheit Googol von vorher über eine Million Googol Googole von Universumszeitaltern.

Was soll an diesem Beispiel verdeutlicht werden? Es gibt Aufgabenstellungen, die wir zwar mit schierer Rechenleistung lösen können, wenn sie sehr klein sind (so klein, dass wir meistens gar keinen Computer benötigen), aber nicht mehr, wenn sie relevante Größenordnungen annehmen.

Abbildung 1.29
Rechenschritte mit Brute-Force für den kürzesten Weg bei 12.309 Gemeinden



In diesem Fall wächst der Aufwand zur Lösung exponentiell an, was bedeutet: Wenn n die Problemgröße ist, kann man den Aufwand nur noch mit x^n nach oben hin abschätzen (mit einem beliebigen x).

Optimal wäre ein linearer Aufwand. Das bedeutet, dass zur Lösung eines Problems der Größe n der Aufwand $x \cdot n$ beträgt (mit konstantem x). Leider gibt es kaum Algorithmen, die so gut sind. Betrachten wir einmal den Aufwand von Dijkstra:

Vom Startknoten aus werden alle benachbarten Knoten (also die, zu denen eine Verbindung besteht) betrachtet. Bei einem vollständigen Graphen mit n Knoten sind das $(n - 1)$. Ein Beispiel mit 7 Knoten sehen Sie in Abbildung 1.30.

Danach ist der Startknoten vollständig „aus dem Rennen", er wird nicht mehr betrachtet. Nun wird der Knoten mit der kleinsten Markierung gesucht und von dort geht es wiederum zu allen benachbarten Knoten – das sind $(n - 2)$. Danach ist auch dieser „aus dem Rennen".

Dies setzt sich fort. Daher ist der Aufwand für den Dijkstra-Algorithmus:

$(n - 1) + (n - 2) + (n - 3) + \ldots + (1)$

Wir können dies wiederum in einer Tabelle wie nebenstehend dokumentieren.

Knoten	Schritte
3	3
4	6
5	10
6	15
7	21
8	28
9	36
10	45
15	105
20	190
50	4.950
12.903	83.237.253

Bei 12.903 Knoten ergeben sich 83.237.253 Rechenschritte. Das erledigt bereits ein normales Smartphone ohne große Schwierigkeiten in kürzester Zeit. An der Tabelle kann man ablesen, dass die Rechenzeit auch schneller als linear mit der Problemgröße anwächst, aber lange nicht so stark wie bei Brute-Force.

Sie erkennen, dass in der Informatik manchmal scheinbare Lösungen nicht praktikabel sind, weil sie bei realen Aufgabenstellungen zu viel Rechenzeit beanspruchen. Ein Informatiker muss daher fast immer die sogenannte Komplexität einer Lösung abschätzen. Die Komplexität gibt an, wie stark die Rechenzeit in Relation zur Problemgröße anwächst. Im Falle von Brute-Force ist die Komplexität n^n, während Dijkstra eine Komplexität von n^2 aufweist.

Die Informatik unterscheidet dabei verschiedene Klassen von Problemen:

Alles, was sich in polynomieller Laufzeit lösen lässt, gehört zur Klasse „P". Das bedeutet, die Laufzeit steigt im Verhältnis zur Problemgröße in einem Verhältnis an, das

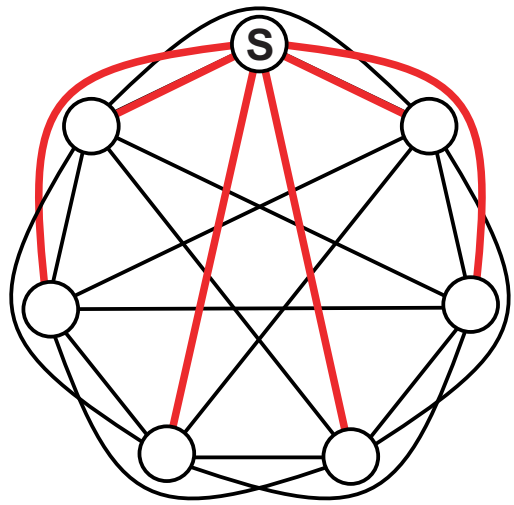


Abbildung 1.30
Im vollständigen Graphen mit 7 Knoten hat jeder Knoten (also auch der Startknoten) 6 Nachbarn.

sich durch ein Polynom ausdrücken lässt. Die Problemgröße steht also im Fall von Hoch-Rechnung immer unten.

Probleme mit polynomieller Laufzeit könnten etwa n^2, $n^4 + n^2 + n$, $n\log n$ oder auch n^{100} als Komplexität haben.

Darüber hinaus gibt es die Klasse der „**N**icht **P**raktisch berechenbaren" Probleme oder „NP". Diese haben eine Laufzeit, die sich nicht durch ein Polynom ausdrücken lässt. Dies bedeutet, dass in der Komplexitätsfunktion die Problemgröße irgendwo in einem Exponenten auftaucht.

NP-Komplexitäten sind 2^n, $n!$ oder n^n.

Die Lösung dieser Frage ist momentan der Heilige Gral der Informatik – oft gesucht, (bisher) nicht gefunden. Genauer beschäftigen wir uns mit dieser Frage später im Buch.
Und wenn Sie Informatiker sind und „NP" daher statt „nicht praktisch lösbar" als „nichtdeterministisch polynomiell lösbar" ausschreiben, haben Sie selbstverständlich recht! Auf allen real existierenden Computern läuft das aber auf die Frage „praktisch berechenbar" hinaus und die Frage P = NP wäre auf jeden Fall mit einer Lösung des Problems des Handungsreisenden in polynomieller Zeit ein für alle Male geklärt.

Allerdings haben NP-Probleme noch eine gute Seite: Man kann eine Antwort raten und dann in polynomieller Zeit überprüfen, ob diese Antwort korrekt ist.

Zu welcher Klasse von Problemen gehört das gerade besprochene Problem des kürzesten Weges? Selbstverständlich in die Klasse „P", denn mit dem Dijkstra-Algorithmus existiert eine Lösung, die in polynomieller Laufzeit arbeitet.

Übrigens gilt dies nicht für das alte „Traveling Salesman"-Problem:

„Ein Vertreter aus Darmstadt muss Kunden in den Städten Berlin, Chemnitz, Frankfurt, Hamburg, Hannover, Marburg, München und Stuttgart besuchen. Bestimmen Sie eine Route, die von seinem Wohnort durch alle genannten Städte und zurück führt und die kürzer ist als 1000 km!"

Für dieses Problem gibt es immer noch keinen Algorithmus mit polynomieller Laufzeit. Vielleicht finden Sie ja einen ... Allerdings ist auch einzusehen, dass es sich um ein NP-Problem handelt: Rät man eine Strecke, kann man in polynomieller Zeit feststellen, ob sie der Bedingung „kürzer als 1000 km" genügt.

Eine weitere Klasse von Problemen ist nicht in polynomieller Zeit berechenbar, und selbst wenn man eine Lösung rät, kann man die Richtigkeit nicht in polynomieller Zeit überprüfen. Diese Erkenntnisse gehen jedoch weit über die praktischen Anwendungen der Informatik hinaus, daher möchte ich hier im Buch nicht weiter darauf eingehen.

Eine der größten Fragen der Mathematik und Informatik ist immer noch kurz gesagt „P = NP?", also ob prinzipiell alle Probleme der NP-Klasse auch in polynomieller Zeit lösbar sind (und nur noch kein entsprechender Algorithmus gefunden wurde) oder ob es Probleme gibt, die grundsätzlich nicht in polynomieller Zeit lösbar sind. Es gibt immer wieder Veröffentlichungen in dieser Richtung, aber sie haben sich bisher als informationstechnische Pendants der Kernfusion im Reagenzglas herausgestellt (also als Flop).

Wenn Ihnen jetzt P und NP noch völlig unklar sind, dann machen Sie sich keine Sorgen – sehr vielen Informatikern geht es ähnlich!

Der schnellste Weg und weitere Anwendungen

Kann man nun den Dijkstra-Algorithmus immer nur zur Berechnung kürzester Wege verwenden?

Ein Informatiker versucht natürlich, für einen so schönen Algorithmus weitere Anwendungen zu finden, denn es ist ja effektiver, für neue Aufgaben bereits bekannte (und bewährte) Verfahren zu adaptieren, als das Rad immer wieder neu zu erfinden.

Betätigen Sie sich doch einmal in dieser Disziplin und versuchen Sie, die folgenden Aufgabenstellungen mit dem Dijkstra-Algorithmus zu lösen.

Wie sieht es etwa aus, wenn wir für unsere erste Karte nicht den kürzesten, sondern den schnellsten Weg suchen?

Dabei gehen wir zur Berechnung davon aus, dass wir mit dem Auto auf der Autobahn im Schnitt 130 km/h schaffen, auf der Landstraße (rot und gelb) jeweils 100 km/h und auf den kleinen Gemeindestraßen (weiß) nur 50 km/h. Zusätzlich veranschlagen wir für jede Ortsdurchfahrt pauschal 8 Minuten (der Einfachheit halber auch bei den Orten, durch die wir auf der Autobahn fahren, auch hier gibt es ja oft Tempolimits aufgrund Lärmschutz etc.).

Finden Sie den schnellsten Weg von Imstadt zu jeder anderen Stadt!

Erstellen Sie einfach eine Karte, auf der an den Straßen nicht die Strecke, sondern die Fahrzeit in Minuten steht. Dazu teilen Sie die Strecke durch die angenommene Geschwindigkeit. Für die Strecke zwischen Imstadt und Budingen ergibt sich auf diese Weise:

$$7\,\text{km} : 80\frac{\text{km}}{\text{h}} = 7\,\text{km} : \frac{80\,\text{km}}{60\,\text{min}} = 7\,\text{km} \cdot \frac{60\,\text{min}}{80\,\text{km}} = 5{,}25\,\text{min}$$

Außerdem müssen für jede Ortsdurchfahrt 8 min addiert werden (also für jeden Ort an einem Ende der Strecke 4 min, da wir mangels anderer Information davon ausgehen, dass das Ein- und Ausfahren gleich lange benötigt).

Daher muss man für die besagte Strecke 13,25 min veranschlagen.

Achtung: Strecken, die an einem Ende die Nicht-Orte Ⓧ, Ⓨ und Ⓩ haben, bekommen natürlich nicht die zusätzliche Zeit für die Ortsdurchfahrt angerechnet.

Abbildung 1.31 zeigt die modifizierte Karte.

Mit dieser Karte kann dann der ganz normale Dijkstra-Algorithmus durchgeführt werden. Das Ergebnis sehen Sie in Abbildung 1.32.

Wenn Sie das Ergebnis mit der Lösung für die kürzeste Wegstrecke nach Abbildung 1.17 vergleichen, stellen Sie fest, dass es recht ähnlich, aber doch nicht identisch ist.

Das ist Informatik: Probleme lösen

Das bisherige Kapitel hat es Ihnen vielleicht schon demonstriert: Informatik macht Spaß, wenn man Aufgaben mit viel Kreativität und Intuition löst. Der wichtigste Schritt zur Lösung ist dann schon getan, wenn man sich etwas zutraut und nicht vor der Herausforderung zurückschreckt. Sie haben das gemacht, indem Sie bis hierhin durchgehalten haben! Verschiedene Untersuchungen haben gezeigt, dass in der Informatik die Tendenz zum Aufgeben für „interessierte Laien“ besonders hoch ist: In der

Abbildung 1.31
Karte mit den angenommenen Wegen in Minuten

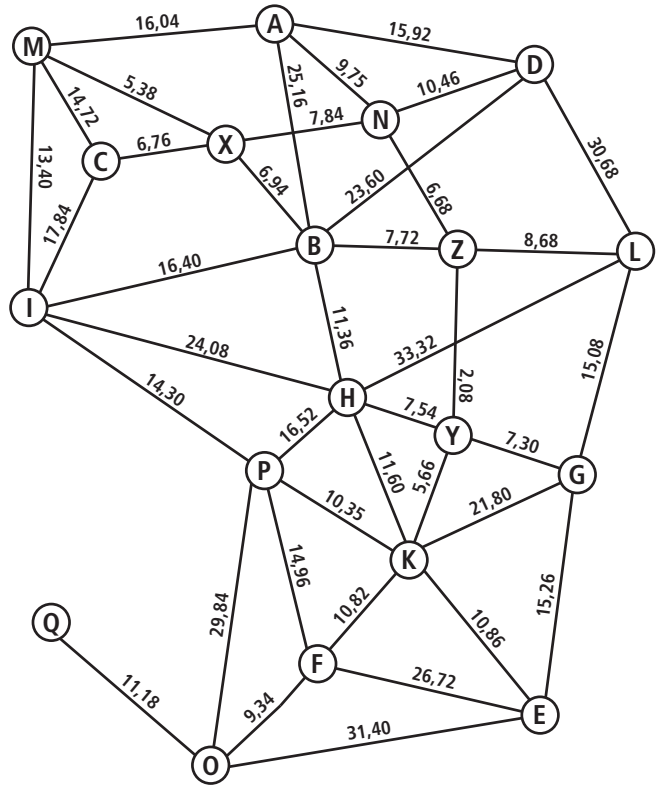
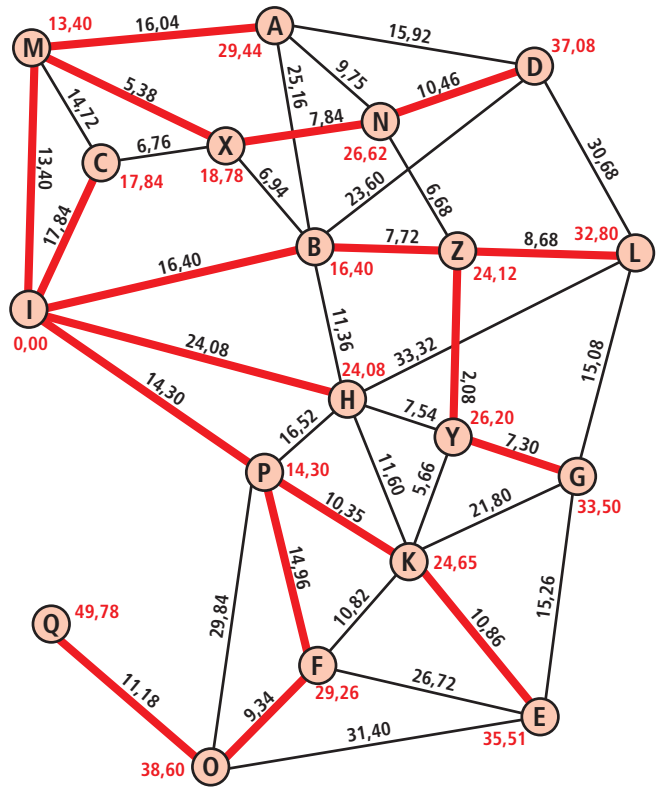


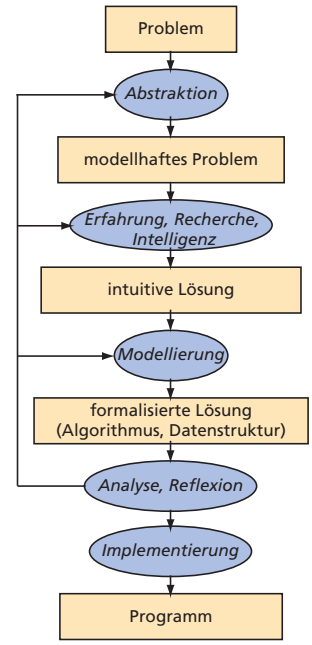
Abbildung 1.32
Die Karte mit allen zeitkürzesten Wegen von Imstadt aus



öffentlichen Wahrnehmung hat sich das Klischee des typischen „Nerds“ etabliert, der zwar sozial eher unterbelichtet, aber eben dadurch hoch kompetent für die speziellen Anforderungen der Informatik ist – eine für „Normalos“ unüberwindliche Hürde …

Dieses Buch soll auch zeigen, dass man sich besonders in der Informatik der Lösung sehr vieler Aufgaben durch konstruktive, gestaltende Herangehensweise sehr gut nähern kann. Der Unterschied zwischen „Normalen“ und „Hochleistern“ besteht häufig nur darin, dass letztere in der Lage sind, einen Ansatz zu finden und diesen konsequent zu verfolgen.

In der Informatik nutzen wir statt „Aufgabe“ gerne einen erweiterten Begriff, den ich in diesem Kapitel bereits verwendet habe, ohne ihn zu definieren: Ein Problem unterscheidet sich von einer einfachen Aufgabe dadurch, dass der Bearbeiter kein Standardverfahren zur Bewältigung kennt. Die Lösung eines Problems ist also immer mit einem schöpferischen Moment verbunden und dem produktiven Denken zuzuordnen. Manchmal fällt es gerade dabei jedoch schwer, den Anfang zu finden. Eine „Ankerlinie“, an der man sich zur Lösung der Aufgabe entlanghangeln kann, stellt daher das allgemeine Problemlöseschema sein, wie in der nebenstehenden Abbildung dargestellt.



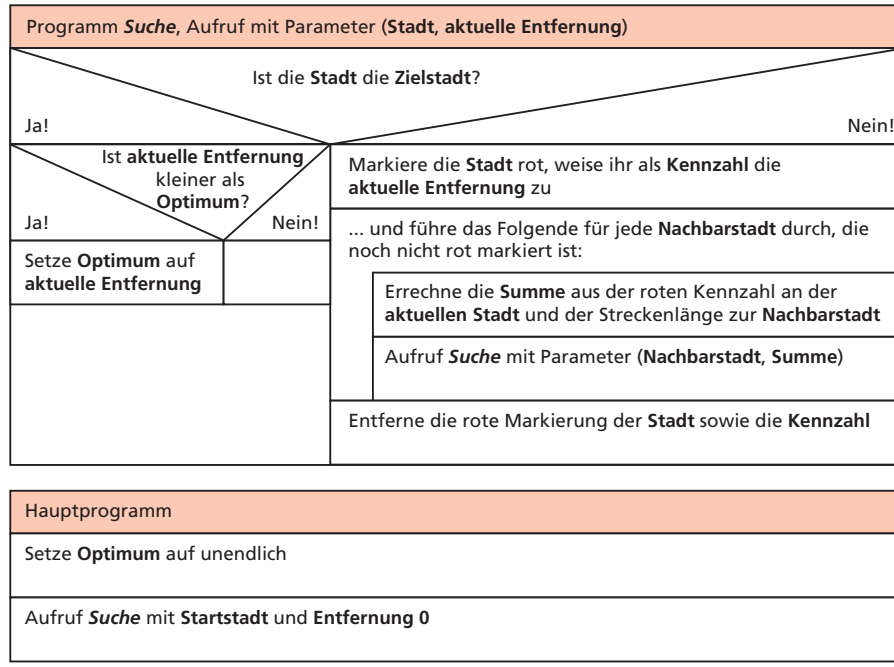
Wenn Sie nun denken, dass das doch der oben aufgeführten Definition eines Problems widerspricht, weil es damit genau kein Standardverfahren zur Lösung geben darf, so haben Sie natürlich recht: Es handelt sich auch nicht um ein Lösungsverfahren, sondern nur um einen Vorschlag für die Herangehensweise, eine Lösung zu finden. Im Prinzip haben wir im Verlauf des ersten Kapitels exakt dieses Problemlöseschema verfolgt, um kürzeste Wege zu finden.

Sie merken vielleicht bereits: Dieser Abschluss des ersten Kapitels ist etwas trockener als der Beginn und wird im Folgenden hauptsächlich die gewonnenen Erkenntnisse nochmal auf eine andere Art darstellen, um sie zu reflektieren und dann auch zu erweitern. Da hier jedoch weniger direkt experimentiert wird: Entscheiden Sie selbst, ob Sie dies bereits jetzt lesen möchten oder diesen Abschnitt zunächst überspringen. Ich verspreche Ihnen, dass hier keine Grundlagen vorkommen, die zum Verständnis des weiteren Buches nötig wären!

Ordnen wir nun die Vorgehensweise beim Nachvollziehen des Dijkstra-Algorithmus in das Problemlöseschema: Zunächst haben wir eine Abstraktion vorgenommen. In diesem Fall konnten wir uns des Standardwerkzeugs „Graph“ bedienen, um die Komplexität der Aufgabenstellung auf das für die Problemlösung minimale Maß zu reduzieren. Der Graph ist ein entscheidendes geistiges Werkzeug, in der Informatik gibt es aber noch viele weitere Standardwerkzeuge für unterschiedliche Arten der Modellierung. In den meisten Fällen ist die Anwendung zwar nicht nach „Schema F“, aber doch sehr mechanisch intuitiv durchführbar.

Der nächste Schritt ist derjenige, der tatsächlich den „kreativen Funken“ benötigt. Effektiv haben wir Menschen normalerweise eine sehr gute Intuition zur Lösung eines Problems – zumindest die Brute-Force-Methode des Alles-Ausprobierens ist im Regelfall leicht vorstellbar. Diese lässt sich auch sehr einfach modellieren und formaler als Algorithmus fassen. Als Darstellung eignet sich dafür alles, was man selbst übersichtlich und lesbar findet. Das kann Text sein, ein Blockdiagramm wie in Abbildung 1.16 oder z. B. ein sogenanntes Struktogramm wie in Abbildung 1.33. Es ist am Beispiel recht selbsterklärend, bei Bedarf finden Sie jedoch genügend Quellen, die die genaue Bedeutung der unterschiedlichen graphischen Elemente aufzeigen.

Abbildung 1.33
Struktogramm für die Brute-Force-Methode, den kürzsten Weg zu finden



Nächster Schritt war dann die Analyse. Der wichtigste Aspekt dabei ist die Einschätzung, ob man mit den zur Verfügung stehenden Ressourcen überhaupt eine Chance hat, die Lösung des Problems auf die vorgeschlagene Weise zu erreichen. Das läuft auf eine Komplexitätsabschätzung hinaus, die anhand des modellierten Algorithmus machbar ist, besonders unter der Perspektive einer Worst-Case-Betrachtung. Das konnten wir mit dem Graphen, in dem alle Städte direkt verbunden sind, modellieren und sind auf die astronomisch hohe Abschätzung für die Laufzeit gekommen.

Die Analyse hat uns also eine eventuell aufwendige Implementierung erspart. Glücklicherweise gibt es im allgemeinen Problemlöseschema auch Pfeile zurück, die uns ermöglichen, einen der vorhergehenden Schritte nochmals anzugehen, um eine verbesserte Lösung zu bekommen. In diesem Fall erkennt man, dass durch eine veränderte Modellierung der Lösung durch Ausprobieren sicher keine Verbesserung erzielt werden kann: Die Analyse lässt sich auf alle entsprechenden Lösungen übertragen.

Im Verlauf des Kapitels haben wir die „intuitive Lösung“ mit Hilfe der Methode der Ameisen gemeistert. Manchmal lässt allerdings bei der Problemlösung ein entsprechender Geistesblitz auf sich warten. Eine Alternative besteht daher darin, das neue Problem auf bereits (meistens von anderen) gelöste Probleme zurückzuführen. Im Zweifel führt Ausprobieren verschiedener bekannter Algorithmen oft zum Erfolg! Im Fall unseres Wegeproblems ist das die Breitensuche. Ich überlasse es Ihnen, das Thema zu recherchieren. Es wird recht schnell offensichtlich, wie man Aufgaben löst, bei denen alle Wege zwischen den Orten gleich lang sind. Als Anregung für die Übertragung auf allgemeine Landkarten schauen Sie sich bei Bedarf Abbildung 1.34 an. Hier sind alle Strecken auf die Länge 1 normiert, indem beim Graphen (mit ganzzahlig gerundeten Entfernungen) entsprechend viele Zwischenknoten eingefügt wurden. Führen Sie hierauf eine ganz normale Breitensuche aus, dann kommen Sie auf das gleiche Ergebnis wie beim Experimentieren mit den Ameisen vorne im Kapitel und (hoffentlich) auch auf Dijkstras Idee. Ein Ergebnis mit eingefärbten Knoten statt Zahlen sehen Sie in Abbildung 1.35.

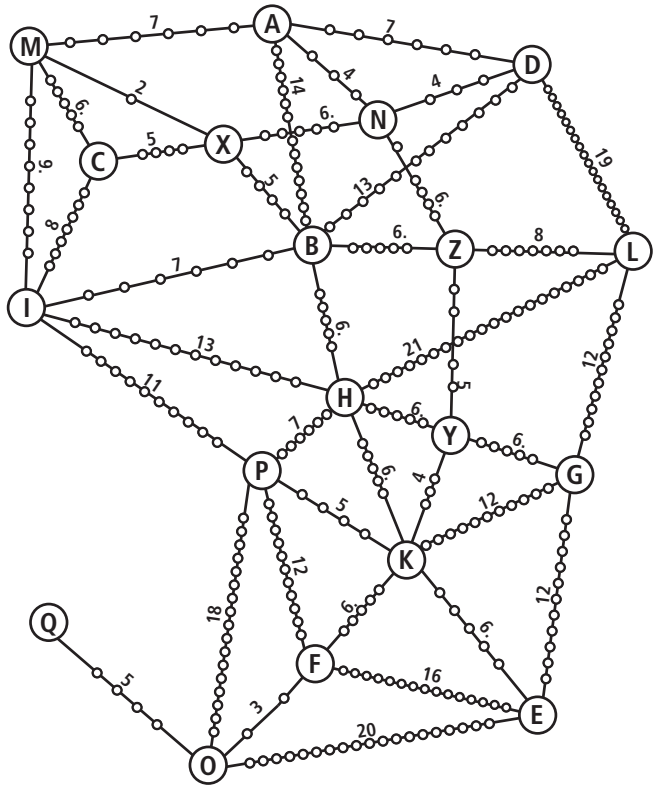


Abbildung 1.34
Karte, auf der durch Zwischenknoten alle Wegstrecken die Länge 1 besitzen. Die Gesamtstrecke könnte weggelassen werden, ist aber zum Vergleich angegeben.

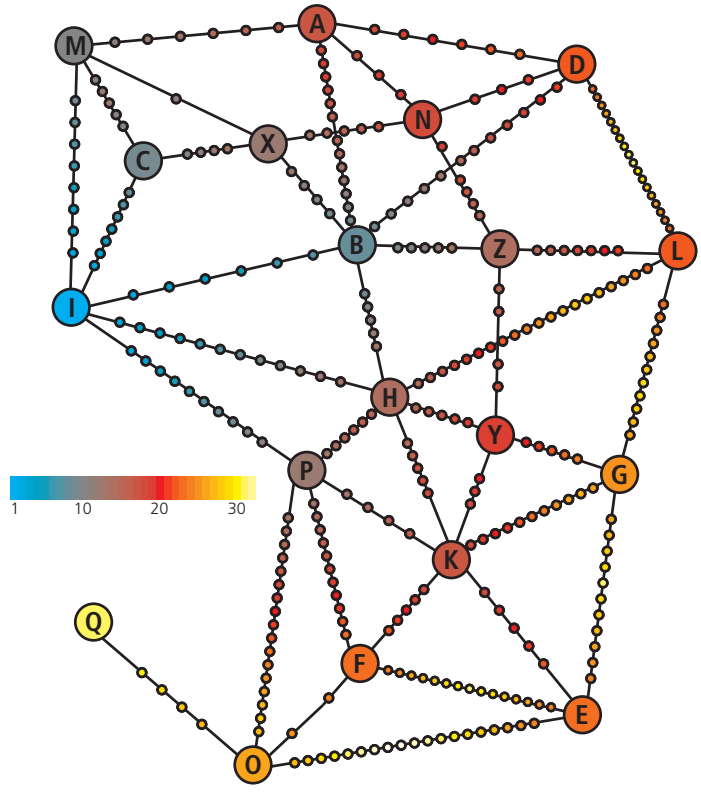


Abbildung 1.35
Abstände als Farben

Bei der anschließenden Analyse ermitteln Sie auch hier die sogenannte quadratische Laufzeit. Dass der originale Dijkstra-Algorithmus sogar noch schneller läuft – in sogenanntem $O(n \cdot \log n)$ – hängt damit zusammen, dass man noch etwas in Bezug auf die verwendeten Datenstrukturen optimieren kann. Das möchte ich hier im Buch jedoch nicht weiter vertiefen, Sie können es selbst unter dem Stichwort „Prioritätswarteschlange“ recherchieren. Zusammen werden wir das allgemeine Problemlöseschema weiter anwenden, indem wir doch noch einmal grundlegender darüber nachdenken, ob sich unsere Lösung weiter verbessern lässt.

Stellen Sie sich vor, Sie nutzen den Dijkstra-Algorithmus zur Bestimmung des kürzesten Weges zwischen zwei Städten in einem Navigationssystem. Wenn Sie die Route von Frankfurt am Main nach München bestimmen, wird das Verfahren – wie man das von einer Breitensuche erwartet – etwa kreisförmig um Frankfurt herum nach dem Ziel suchen. Abbildung 1.36 zeigt das. Viel Rechenaufwand wird also hineingesteckt, um zum Beispiel den Weg über Hannover zu optimieren, obwohl Hannover ganz sicher nicht auf der Route nach München liegt!

Wenn wir im Problemlöseschema zurück gehen, finden wir eventuell eine leicht verbesserte Lösung: Um die beiden Städte wird ein Rechteck, eine „Bounding-Box“ gezeichnet und die Suche nach dem kürzesten Weg begrenzen wir auf Strecken innerhalb dieses Rechtecks. Gegebenenfalls erweitert man das Kästchen noch großzügig, um auch die Autobahnauffahrt zu finden, für die man zunächst ein kurzes Stück in die „falsche Richtung“ fahren muss. Abbildung 1.37 zeigt ein Beispiel.

An dieser Stelle können wir zunächst formal darüber nachdenken, zu welchem Schritt des Problemlöseschemas wir für diesen Vorschlag zurückgehen müssen. Ist es eine neue, kreative Problemlösung? Effektiv nutzen wir allerdings Informationen, die wir

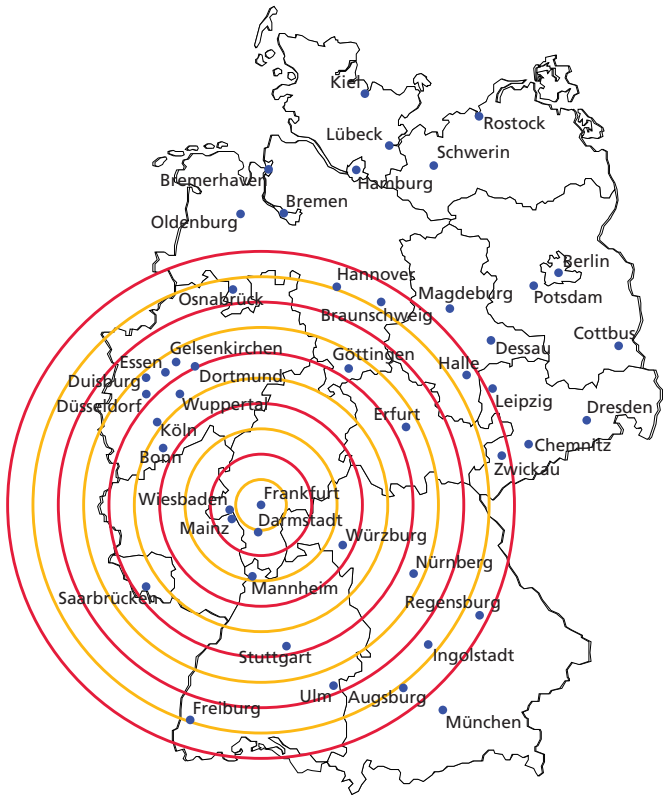


Abbildung 1.36
Dijkstra nutzt recht uneingeschränkt die Breitensuche und erkundet den Graphen daher ohne Bevorzugung einer Richtung.

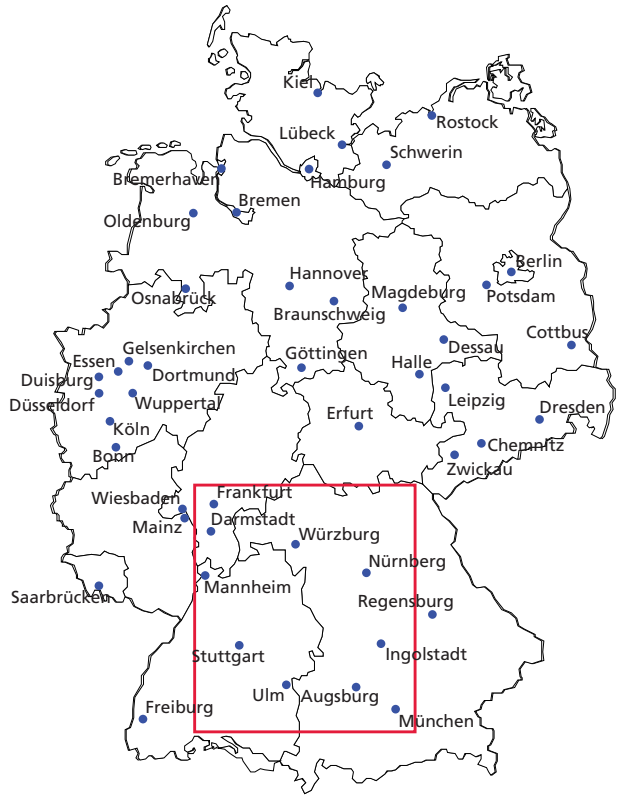


Abbildung 1.37
Ein gedachtes Rechteck um Start und Ziel grenzt die Suche ein, indem nun alle Orte außerhalb des Rechtecks nicht mehr in das Verfahren einbezogen werden.

bereits im Abstraktionsschritt zuvor weggelassen hatten: die geographischen Positionen der Orte! Diese benötigen wir natürlich, um zu bestimmen, ob ein Ort innerhalb oder außerhalb des Rechtecks liegt. Daher fangen wir hier sogar noch weiter vorne an!

Die Methode der Bounding-Box könnte dazu führen, die beste Strecke gar nicht zu finden: Woher wissen wir, dass es nicht günstiger ist, ein paar Kilometer in Gegenrichtung zu fahren, um dann eine schnurgerade Autobahn zu erreichen, auf der man den Umweg leicht wieder hereinholt? In einigen Fällen könnte die Bounding-Box sogar verhindern, dass man überhaupt eine Strecke findet. Die nebenstehende Abbildung stellt einen Teil Süditaliens dar. Möchte man von Lecce nach Catanzaro navigieren, findet man innerhalb der rot eingezeichneten Bounding-Box keinen Weg (außer man schwimmt).

Besser ist, zwar den Fokus der Suche in Richtung des Ziels zu lenken, aber trotzdem noch alle Optionen offenzuhalten. Wie so oft in der Informatik ist es auch hier sinnvoll, sich selbst bei der manuellen Suche nach einer günstigen Fahrtroute auf die Finger zu schauen. Wenn wir vom Start ausgehen und den nächsten Ort ansteuern, sind uns solche Orte am liebsten, die uns näher zum Ziel bringen. Wir könnten daher vielleicht die Länge der direkten Strecke zum Ziel (Luftlinie) als eine Art „Lustfaktor“ in den Algorithmus einbeziehen: Je kürzer die Luftlinie, desto williger steuern wir einen Ort an, werden aber die anderen Orte auch nicht ignorieren.

Im Problemlöseschema haben wir der neuen Anforderung nach dem Versuch mit der Bounding-Box bereits Rechnung getragen und die für die Bestimmung der Luftlinie nötigen Informationen hinzugenommen: die genaue geographische Position der Städte.

Nun lässt sich die Länge der Luftlinie zum Ziel sehr einfach aus den geographischen Koordinaten berechnen. Abbildung 1.38 zeigt den Graphen, in den mit hellblauer Schrift an jedem Knoten diese Luftlinienentfernung zum Ziel eingetragen ist. Es handelt sich erneut um unser Standardbeispiel von Imstadt nach Oppenheim. Wie kommt man nun auf eine weitere Verbesserung für einen bereits bekannten Algorithmus? Man führt ihn durch und prüft in jedem Schritt, ob sich dieser in Bezug auf eine Verbesserung in der geplanten Weise eignet.

Der erste Schritt ist die Ermittlung aller Nachbarknoten und deren Entfernungen von der Startstadt aus. Hier fällt es uns schwer, etwas einzusparen. Aber wir können auch stringenter denken: Eigentlich wollen wir ja zum Ziel, die Nachbarstädte sind daher wahrscheinlich nur Wegpunkte. Und wir wissen, welche minimale Entfernung jeder Wegpunkt zum Ziel hat. Also können wir aus der Summe der tatsächlich ermittelten Entfernung zum Wegpunkt und der minimalen Entfernung des Wegpunktes zum Ziel eine minimale Strecke zum Ziel bestimmen für den Fall, dass wir den entsprechenden Wegpunkt benutzen. Abbildung 1.39 stellt die entsprechenden Berechnungen mit violetter Summe dar.

Im Prinzip funktioniert unser neuer Algorithmus – in der Fachliteratur bekannt als A-Stern – wie „Dijkstra“. Allerdings bestimmen wir den nächsten zu besuchenden Knoten ein klein wenig anders: Gemäß der neuen Idee, betrachten wir nun nicht mehr nur die ermittelte Entfernung von der Startstadt, sondern die Summe aus dieser Entfernung und der Luftlinie zum Zielort. Diese Summe stellt quasi die minimal zurückzulegende Strecke dar, die man benötigt, um vom Start zum Ziel zu kommen, wenn man die entsprechende Stadt als Teilziel wählt.

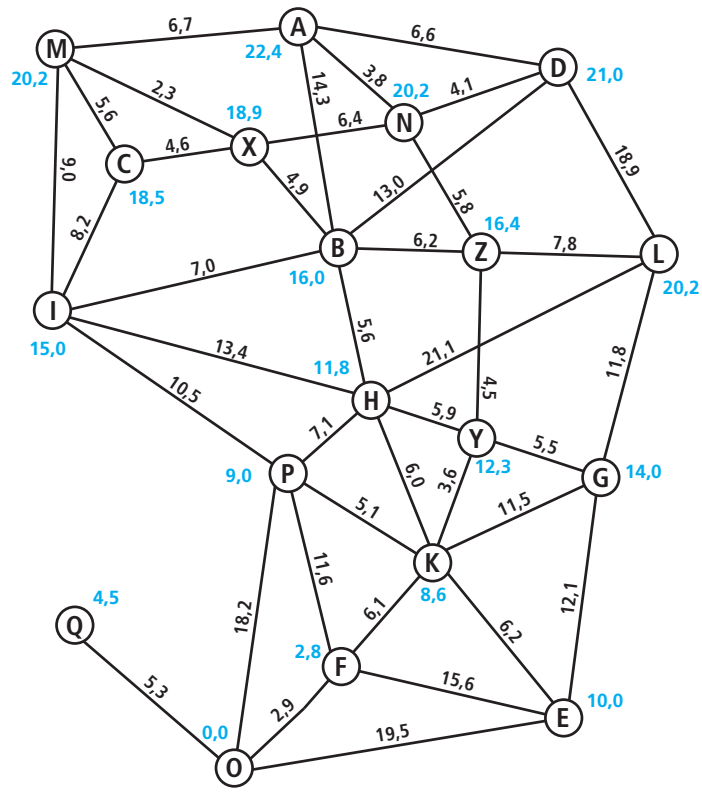


Abbildung 1.38
Landkarte als Graph mit zusätzlichen Knotenmarkierungen für die Luftlinie zum Ziel Oppenheim

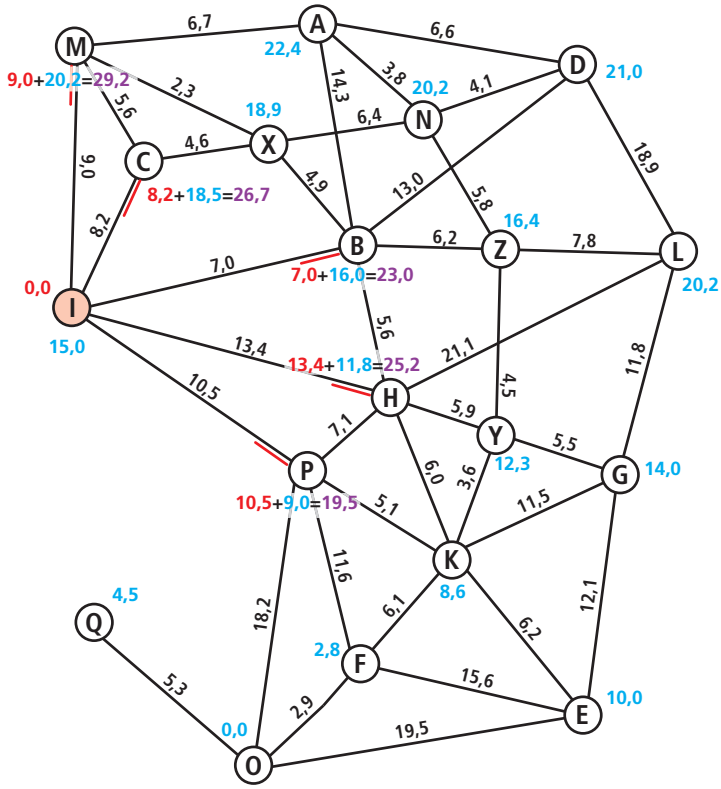


Abbildung 1.39
Neben den tatsächlich ermittelten Strecken zu den Orten vom Start aus, können wir auch die Summe (violett) aus tatsächlicher Strecke vom Start (rot) und minimaler Strecke zum Ziel (blau) ermitteln. Das ist gleichzeitig eine minimale Strecke zum Ziel bei Nutzung des Ortes als Teilziel.

Das Blockdiagramm des neuen Algorithmus nach Abbildung 1.40 ist praktisch identisch mit dem, das Sie aus Abbildung 1.16 bereits als Dijkstra-Algorithmus kennen. Den einzigen Unterschied habe ich zur Verdeutlichung rot markiert. Bei einer Implementierung muss die hier zur Verdeutlichung in den Graphen aufgenommene blaue Luftlinie genau wie die violette Summe natürlich nicht als eigene Knotenmarkierung gespeichert werden, sondern kann mit dem Satz des Pythagoras aus den Koordinaten der Orte bei Bedarf ganz einfach berechnet werden.

Hier wird auch ein ganz wesentlicher Vorteil der sehr allgemeinen, abstrakten Beschreibung eines Algorithmus deutlich, die in unserem Fall die Form des Blockdiagramms hat: Verbesserungen und Veränderungen sind so meistens nur sehr geringfügig. Es wird sofort offensichtlich, an welchen Stellen die Programmierer auch die Implementierung überarbeiten müssen.

Modellbildung

Ein informatisches Modell beschreibt genau die Teile einer Problemstellung übersichtlich, die zur Lösung des Problems nötig sind. Das gilt insbesondere auch für die Modellierung der Problemlösung als Algorithmus oder als Datenstruktur. Modellbildung findet auf vielen unterschiedlichen Ebenen statt: Selbst die Implementierung in einer gängigen Programmiersprache stellt eine Modellierung dar, da sie bestimmte Konzepte und Denkweisen impliziert.

Abbildung 1.40
A-Stern Algorithmus als Blockcode

0. Markiere die **Startstadt** rot, weise ihr die Kennzahl 0 zu. Bezeichne diese als **aktuelle Stadt**.

1. Gehe aus von der **aktuellen Stadt** zu allen direkt erreichbaren **Nachbarstädten** ...

... und führe das Folgende für jede **Nachbarstadt** durch, die noch nicht rot markiert ist:

Errechne die **Summe** aus der roten Kennzahl an der **aktuellen Stadt** und der Streckenlänge zur **Nachbarstadt**.

- Hat die **Nachbarstadt** keine **Kennzahl**, weise ihr die **Summe** als Kennzahl zu. Markiere die Strecke zur **aktuellen Stadt**.
- Hat die **Nachbarstadt** eine **Kennzahl** kleiner oder gleich der **Summe**, mache nichts.
- Hat die **Nachbarstadt** eine **Kennzahl** größer der **Summe**, streiche die dortige **Kennzahl** sowie die Markierung. Weise ihr danach die **Summe** als neue **Kennzahl** zu. Markiere die Strecke zur **aktuellen Stadt**.

2. Betrachte alle Städte, die zwar eine rote **Kennzahl** haben, aber nicht rot markiert sind. Suche diejenige mit der kleinsten **Summe aus Kennzahl und Luftlinie zum Ziel**.

3. Bezeichne diese als **aktuelle Stadt**. Weisen mehrere Städte die kleinste Kennzahl auf, wähle eine beliebige davon.

4. Markiere die **aktuelle Stadt** rot, zeichne die dort markierte Strecke komplett rot nach.

5. Falls die **Zielstadt** noch nicht rot markiert ist, weiter bei (1.)

Abbildung 1.41 zeigt den nächsten Schritt des Algorithmus zum Nachvollziehen: Knoten Ⓟ hat die kleinste violette Markierung und wird als Nächstes rot markiert, nun ist allerdings die tatsächliche Strecke dorthin bestimmt und sowohl blaue als auch violette Markierungen werden unerheblich. Für die Nachbarn werden die roten Kennzahlen ermittelt, daraus ergeben sich wiederum violette Summen.

Die Strategie wird nun konsequent verfolgt. Abbildung 1.42 stellt einen weiteren Schritt dar. Knoten Ⓑ wird rot markiert. Beachten Sie, dass sich für Knoten Ⓗ dadurch die Bewertungen verändern.

Nun möchte ich Ihnen aber gar nicht den Spaß verderben, die restlichen Schritte des Algorithmus selbst nachzuvollziehen – ganz selbstständig oder mit Hilfe der weiteren Abbildungen bis 1.46.

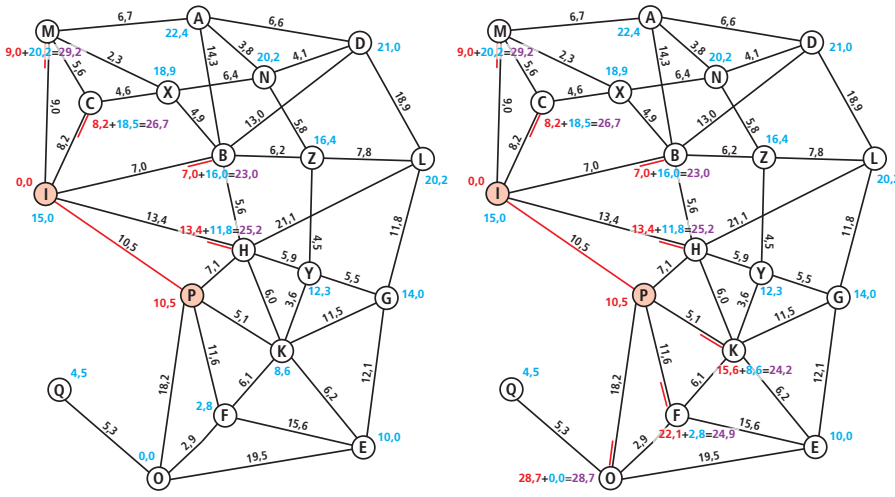


Abbildung 1.41
Nächster Schritt von A-Stern

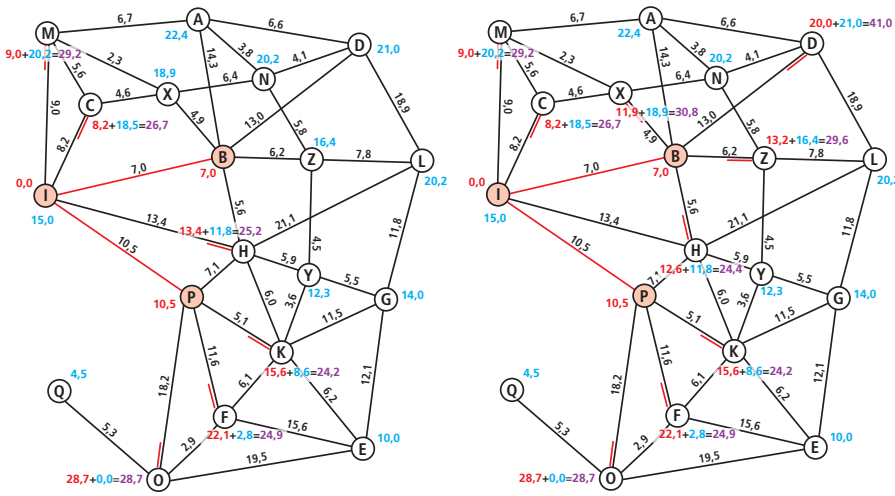


Abbildung 1.42
A-Stern

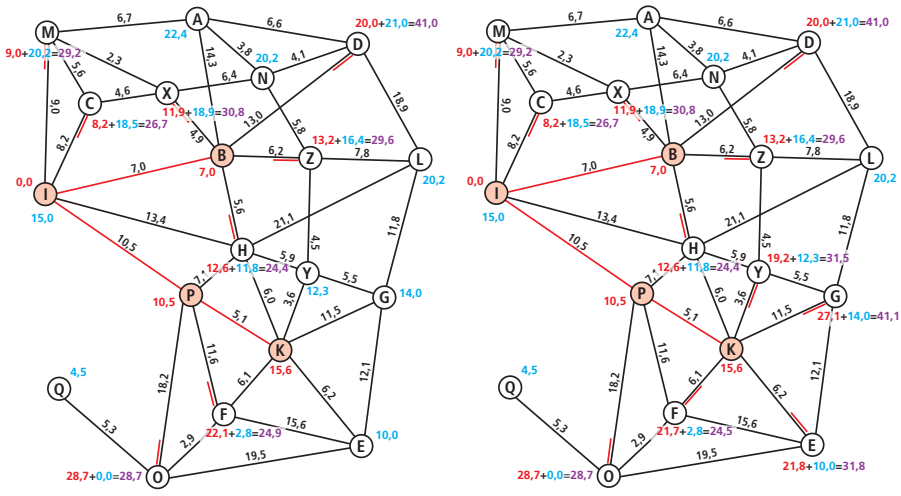


Abbildung 1.43
A-Stern

Abbildung 1.44
A-Stern

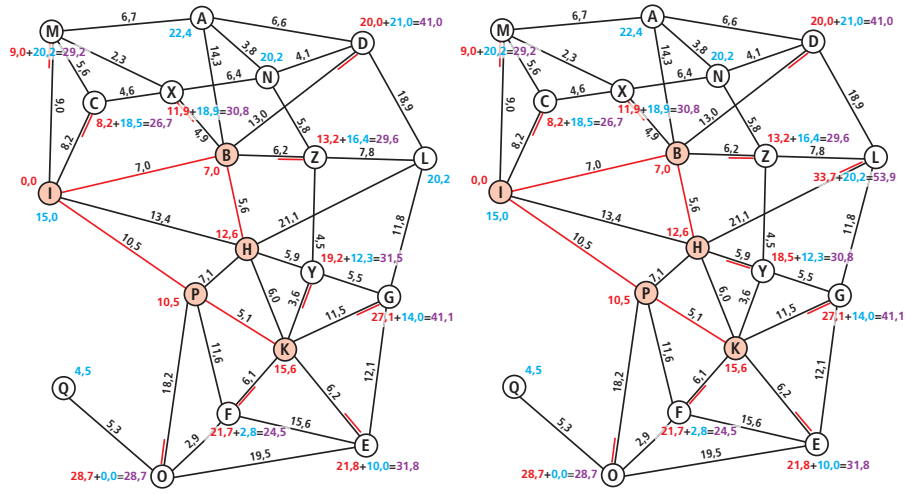
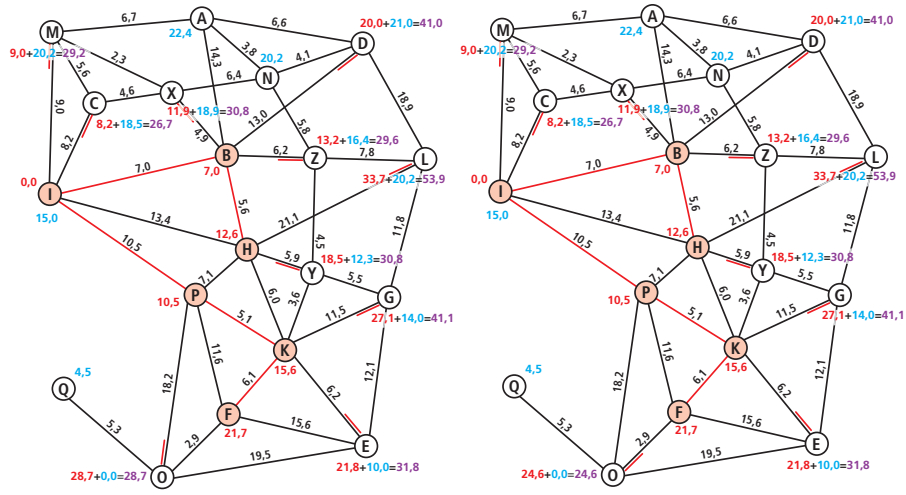


Abbildung 1.45
A-Stern



Unser Wunsch nach besserer (Ausführungs-)Geschwindigkeit ist mit A-Stern erfüllt worden: Statt in alle Richtungen gleichermaßen zu suchen, bevorzugt der neue Ansatz die Richtung zum Ziel. Im Beispiel kommt der Algorithmus auch zum gleichen Ergebnis wie Dijkstra. Die Frage ist allerdings, ob er auch immer das Optimum findet, denn unser Argument mit dem Ameisenprinzip gilt leider nicht mehr, da wir ja die Besuchsreihenfolge verändert haben. Der Analyseschritt der allgemeinen Problemlösestrategie ist also teilweise noch offen, solange die Frage nach der Optimalität nicht geklärt ist.

In Abbildung 1.47 ist der „erkundete“, also rot markierte Teilgraph zur Verdeutlichung gelb umrandet. Bemerken Sie, dass alle Nachbarknoten, also alle potentiellen Teilziele eine rote und damit auch eine violette Markierung tragen? Diese entspricht der **minimal** zurückzulegenden Strecke, wenn wir diesen Knoten als Teilziel nutzen. Und jede dieser Zahlen ist größer als die tatsächlich ermittelte Strecke zum Ziel. Wäre eine der Zahlen kleiner, hätten wir den entsprechenden Knoten gemäß unserem Algorithmus ja ebenfalls rot markiert!

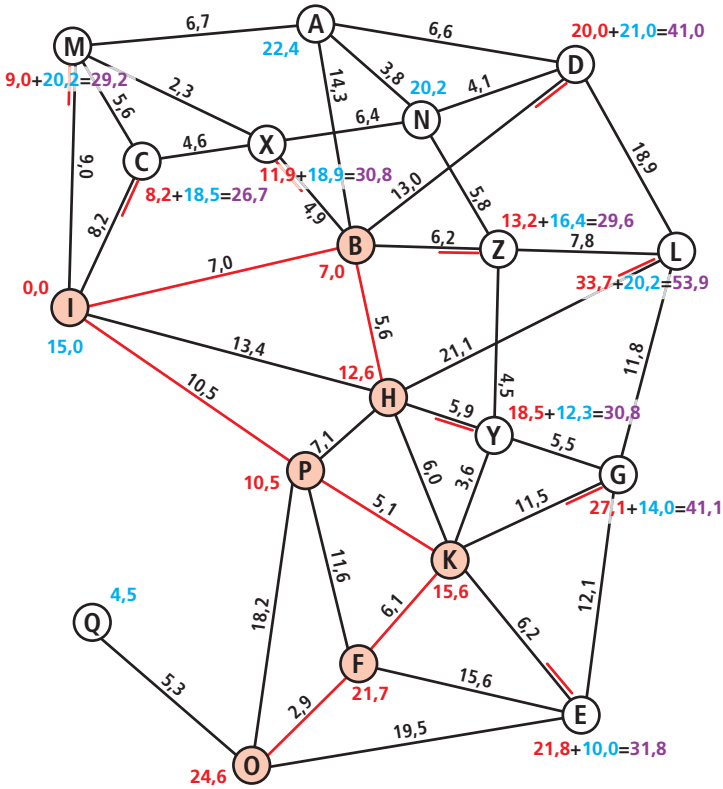


Abbildung 1.46
Letzter Schritt des A-Stern Algorithmus: Das Ziel Oppenheim wurde erreicht in 24,6 km.

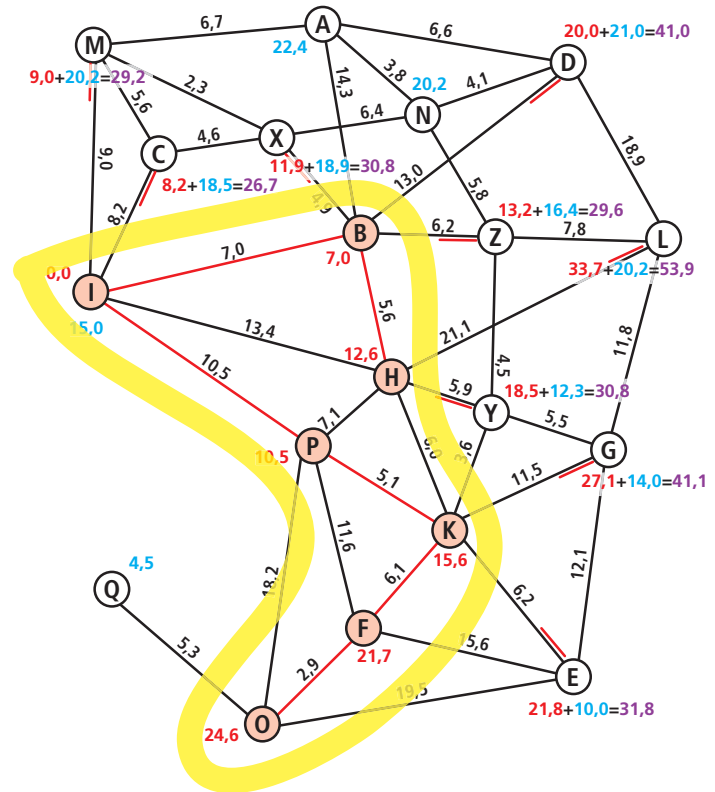


Abbildung 1.47
Alle Knoten in der Nachbarschaft des erkundeten Pfades wurden als Teilziele bewertet und nicht berücksichtigt.

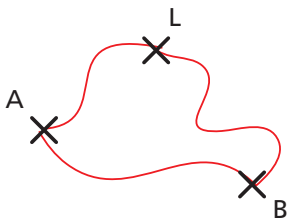
Damit ist nachgewiesen, dass alle „Abwege“ eine Verschlechterung bedeutet hätten und damit unsere gefundene Strecke – wie bei Dijkstra – die kürzeste ist.

A-Stern ist ein sehr schönes Beispiel dafür, dass wir in der Informatik oft einen bekannten Algorithmus nur leicht zu erändern brauchen, um ihn für einen bestimmten Zweck deutlich zu verbessern. Effektiv haben in diesem Fall wenige Zeichen Code ausgereicht. A-Stern wurde übrigens schon 1968 von den Peter Hart, Nils Nilsson und Bertram Raphael veröffentlicht, fand aber erst in den 1990ern erste echte Anwendungen – in Computerspielen.

Das Thema „Bestimmung des kürzesten Weges“ ist damit aber immer noch nicht am Ende: Auch die von A-Stern mit der Luftlinie als minimale Entfernung zum Ziel erreichte Steigerung der Laufzeit des Verfahrens reicht immer noch nicht aus, damit bekannte Portale effizient Millionen von Routen pro Sekunde berechnen können. Selbst wenn das Argument „dann stellen wir eben noch ein paar zusätzliche Server hin“ auf den ersten Blick einen gewissen Charme hat: Jede weitere Steigerung der Effizienz führt bei Millionen von Anfragen gleich auch zu einer spürbaren Entlastung unserer Umwelt, weil erhebliche Mengen Strom und damit CO_2 eingespart werden können.

Für Enthusiasten lässt sich daher noch nachvollziehen, wie man weiter vorgeht, um auch A-Stern weiter zu verbessern. Knackpunkt ist die erwähnte „Luftlinie“. Damit unsere Argumentation für die Optimalität von oben noch greift, muss dieser Wert auf jeden Fall kleiner oder gleich der tatsächlich zu fahrenden Strecke zwischen zwei Orten sein. Die Luftlinie hat diese Eigenschaft, ist aber oft keine wirklich gute untere Schranke. Denken Sie etwa an das Beispiel Lecce–Catanzaro aus Abbildung 1.40. Die Luftlinie läuft mitten durch das Mittelmeer, ist also sehr viel kürzer als die tatsächlich zurückzulegende Strecke drumherum. A-Stern würde in diesem Fall immer noch sehr viel Sucharbeit in die falsche Richtung investieren. Wie bekommt man aber eine bessere Abschätzung nach unten?

Andrew Goldberg, Haim Kaplan und Renato Werneck arbeiten dafür mit wenigen Leuchtturmpunkten („Landmarks“), zu denen sie die tatsächliche Entfernung vorberechnen. Während eine Tabelle mit der zurückzulegenden Strecke von jedem Punkt der Erde zu jedem anderen die Kapazität jeder Datenbank sprengen würde, ist dies von jedem Punkt der Erde zu 10 bis 20 vordefinierten „Leuchttürmen“ durchaus problemlos.



Die Idee dafür sei hier nur skizziert: Für drei beliebige Punkte bzw. Orte der Landkarte – wir nennen sie A, B und L – gilt die Dreiecksungleichung. Der zurückzulegende Weg zwischen zwei beliebigen der Punkte ist immer maximal so lang wie die Summe der beiden übrigen Wege. Der Beweis ist sehr einfach: Nehmen wir zum Beispiel an, der Weg |AB| sei länger als |LA|+|LB|. Dann könnten wir von A nach B einfach über L fahren und damit genau |LA|+|LB| zurücklegen, was die Annahme widerlegt.

Für die Skizze am Rand gelten also bezüglich der kürzesten Strecken zwischen den drei Punkten A, B und L alle Dreiecksungleichungen:

$$|AB| \leq |LA| + |LB|$$

$$|LA| \leq |LB| + |AB|$$

$$|LB| \leq |AB| + |LA|$$

Die unteren beiden kann man umformen in

$$|AB| \geq ||LA| - |LB||$$

Für die kürzeste mögliche Strecke zwischen zwei beliebigen Punkten A und B ist also die Differenz der kürzesten Strecken von A und B zu einem Leuchtturm L eine untere Schranke. Rechnet man diese untere Schranke für alle möglichen Leuchttürme aus, ist das Maximum dieser Schranken immer noch eine untere Schranke, die normalerweise aber schon recht gut ist.

Resümee

Anhand der Lösung des Wegeproblems haben wir wichtige Werkzeuge der Erkenntnisgewinnung in der Informatik kennen gelernt. Abstraktion und Modellbildung – das gezielte Weglassen irrelevanter Information und die übersichtliche Darstellung der relevanten – sind die wichtigsten Voraussetzungen, um eine Aufgabe zu lösen. Wenn man durch Intuition, Analogiebildung oder Verwendung bekannter Algorithmen einen Ansatz gefunden hat, hilft es wiederum, diesen formal zu beschreiben, um durch Analyse herauszufinden, ob er für die zur Verfügung stehenden Ressourcen wie Rechenzeit und Speicherplatz auch funktioniert.

Ein allgemeines Problemlöseschema mit den aufgeführten Schritten hilft bei der Bewältigung recht komplexer informatischer Probleme, weil es eine Ankerlinie darstellt, an der man sich entlanghangeln kann. Selbstverständlich muss man trotzdem noch sehr viel Kreativität und logisches Denkvermögen einsetzen, was die meisten Menschen glücklicherweise auch besitzen. Sie müssen sich nur bei der Lösung „beobachten“ und die Vorgehensweise formalisieren – schon ist der Algorithmus zu Papier gebracht und danach auch verhältnismäßig schnell implementiert.

Eine einmal gefundene Lösung kann dann auf viele Aufgaben angewendet werden. So kann man mit Dijkstra nicht nur den kürzesten Weg, sondern allgemein den „besten“ Weg berechnen, wobei man beliebige Qualitätskriterien annehmen kann. Dabei modellieren wir unsere Welt immer explizit und implizit. Die angenommenen Zeiten für die fahrbaren Geschwindigkeiten und die Zuschläge für die Ortsdurchfahrten beeinflussen das Ergebnis in ganz erheblicher Weise. Stimmen sie nicht, kann unser restliches Lösungsverfahren perfekt funktionieren – eine realistische Lösung wird nicht herauskommen.

Fast immer lässt sich ein Verfahren noch weiter verbessern, etwa durch eine neue Lösungsstrategie oder – wie im Fall A-Stern – durch ein verändertes Modell der Welt.

In den folgenden Kapiteln wird uns die hier erprobte „Informatiker-Denke“ immer wieder begegnen.

Abbildung 1.K1
Abstrakte Karte als Grundlage für unsere Überlegungen zur Lösung der Aufgabe

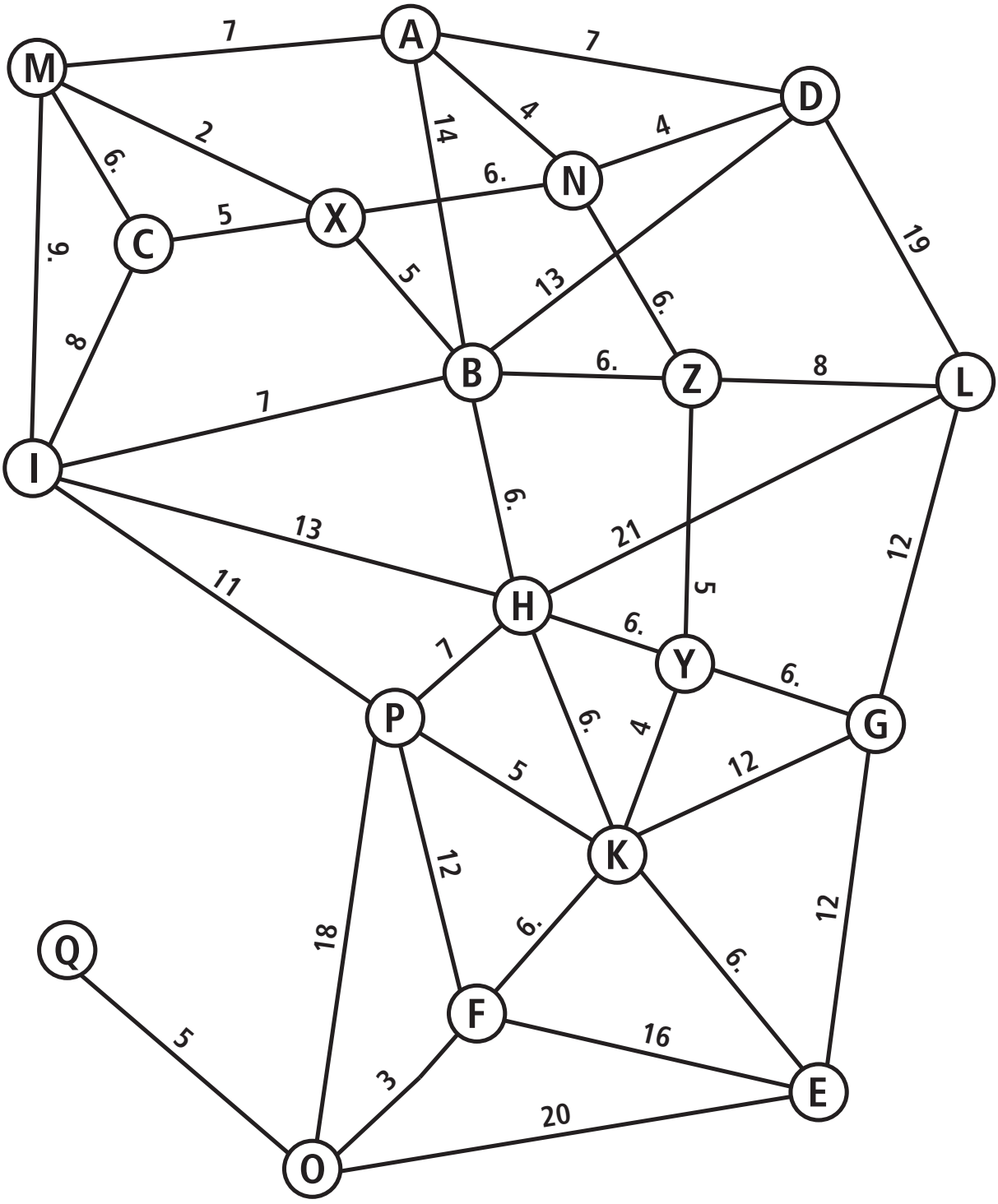
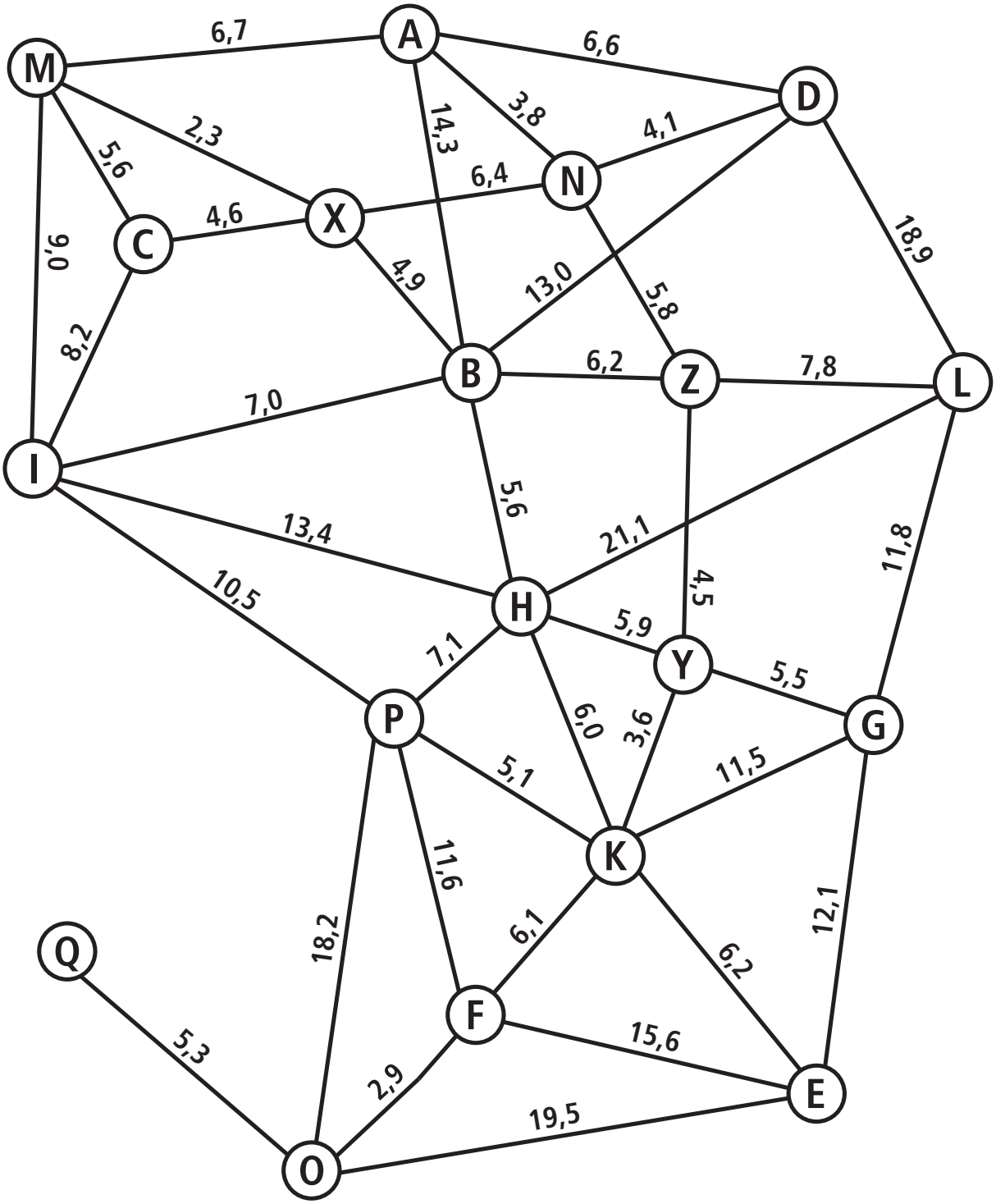


Abbildung 1.K2
Abstrakte Karte mit genaueren Wegstrecken




AIESEC-LAESTE-
REDAKTION
AKAKRAFT-1323
IBZ
ELEKTROTECHNISCHER-VEREIN-
STUDENTENWERK
HRZ-TELEFONZENTRALE
HAUSHALTSSTEL
PERSONALRAT
KASSE
Bibliothek-Institut für Pädagogik
ZENTR. TECHN. DIENST
ASTA-LICHTW
Lichtwiese

2. Ordnung muss sein!

Einführung

Ordnung ist das halbe Leben. Schön ist jedoch, wenn man nicht das halbe Leben damit zubringen muss, sie zu halten! Wie gut, dass es heute Computer gibt, die einen darin unterstützen.

Wie aus dem deutschen Kunstwort „Informatik“ schon ersichtlich wird, hat das Geschäft eines Informatikers sehr viel mit Information zu tun: Diese soll nicht einfach nur irgendwo abgelegt werden, sondern man möchte sie auch möglichst schnell wieder auffinden.

Das Grundproblem unterscheidet sich dabei nicht von der Frage, die auch für das Arbeitszimmer, die Küche und andere Bereiche des Lebens gilt: „Wer Ordnung hält, ist nur zu faul zum Suchen“ oder, etwas vornehmer ausgedrückt, „Der Aufwand, Ordnung zu schaffen, muss dadurch gerechtfertigt werden, dass man daraus erhöhten Komfort zieht“ – zum Beispiel mehr Platz, schnelleres Auffinden von Dingen usw.

Normalerweise versucht man daher, das Chaos erst gar nicht aufkommen zu lassen – jeden neuen Gegenstand also gleich an den richtigen Platz zu setzen. In verschiedenen Situationen funktioniert das jedoch nicht: Post kommt meistens als ungeordneter Stapel an, Spielkarten werden gemischt ausgegeben usw. Hier wird eine vollständige Sortierung notwendig. In diesem Kapitel beschäftigen wir uns damit, wie Computer die Aufgabe lösen, eine unsortierte Menge von Daten so zu organisieren, dass der Zugriff auf einen einzelnen Datensatz sehr schnell geht.

Um das wiederum experimentell nachzuvollziehen, eignen sich Spielkarten sehr gut: Jeder hat Erfahrung damit, hat selbst schon einmal die zugeteilten Karten auf der Hand sortiert, um dann Doppelkopf, Rommé oder Canasta präziser und schneller spielen zu können. In Abbildung 2.K1 am Ende des Kapitels finden Sie Kopiervorlagen für ein paar ganz spezielle Spielkarten zum Ausschneiden, die statt Kreuz, Karo, Ass und Dame normale Zahlen enthalten, denn dies sind die Größen, mit denen ein Computer meistens rechnen muss: Geburtsdaten und Sozialversicherungsnummern für Rentenauszahlung, Matrikelnummern von Studierenden für die Prüfungsverwaltung, Kontonummern und viele mehr. Schneiden Sie die Karten aus – Mischen ist erst einmal nicht notwendig!

Tipp:
Die Vorlagen für die Bastelmaterialien können Sie auch im Internet herunterladen. Dort finden Sie zusätzlich Bezugsquellen für fertige Bastelbögen auf Pappe:

www.abenteuer-informatik.de

Das Sortierproblem

Bei allen Aufgaben, die man mit dem Computer lösen möchte, ist ein guter Ansatz, sie erst einmal manuell zu erforschen, um ein Gefühl dafür zu bekommen und die Problemstellung zu erfassen. Wie wir beim Auffinden des kürzesten Weges im letzten

Kapitel gesehen haben, ist dabei oft die Problemgröße relevant. Das ist in unserem Fall die Anzahl zu sortierender Objekte.

Machen Sie daher einen ersten Versuch:

Nehmen Sie etwa die Hälfte der Karten mit orangem Symbol auf der Rückseite und sortieren Sie sie nach der blauen Zahl. Ergebnis sollte eine zusammenhängende Reihe von Karten (ohne Lücken) auf Ihrem Schreibtisch sein. Falls hier nicht genug Platz ist, dürfen selbstverständlich auch mehrere Reihen untereinander die Sortierung ergeben.

Eigentlich war das aber nur die Übung für den eigentlichen Versuch: Nehmen Sie nun 120 der Karten mit orangem Rücken (einschließlich der gerade sortieren) und mischen Sie erneut. Bilden Sie dann sechs Stapel zu jeweils 20 Karten, die Sie bereitlegen. Sie benötigen außerdem eine Uhr mit Sekundenangabe oder eine Stoppuhr: Messen Sie die Zeit, die Sie zum Sortieren des ersten Stapels nach den blauen Zahlen benötigen, und schreiben Sie das Ergebnis auf.

Danach sammeln Sie die Karten wieder ein, nehmen einen weiteren Stapel hinzu, so dass es jetzt 40 Karten sind. Sortieren Sie nun nach den roten Zahlen. Auf diese Weise sparen Sie das Mischen zwischendurch, weil die roten Zahlen unsortiert sind, wenn die blauen sortiert sind, und umgekehrt. Notieren Sie auch hier die benötigte Zeit. Wiederholen Sie das mit 60, 80, 100 und 120 Karten.

Stellen Sie nun die Ergebnisse graphisch dar: Nehmen Sie ein Stück kariertes Papier und übertragen Sie die nebenstehende Abbildung! Machen Sie dann für jeden Ihrer Messwerte ein Kreuz über der entsprechenden Anzahl von Karten (20, 40, 60, ...) in der Höhe, die der benötigten Zeit zum Sortieren entspricht, und verbinden Sie die Kreuze mit Linien!

Vielleicht haben Sie ja auch noch willige „Opfer“ in Ihrer Umgebung, mit denen Sie den gleichen Versuch durchführen können: Lassen Sie sie zunächst ebenfalls eine Anzahl von Karten (ungefähr 60) sortieren, um sich an den Vorgang zu gewöhnen, und messen Sie dann die Zeit zum Sortieren von 20, 40, 60 usw. Karten. Übertragen Sie die Ergebnisse möglichst jeweils mit einer eigenen Farbe in Ihr Diagramm. Ein Beispiel dafür sehen Sie in der Abbildung 2.1.

Nehmen Sie sich nach jedem Versuch auch kurz Zeit, um die Strategie des Sortierers mit ein paar Stichworten festzuhalten.

Wie können wir das Diagramm interpretieren? Die erste Erkenntnis ist, dass die Zeit zum Sortieren mit der Anzahl der Karten im Allgemeinen ansteigt. Das ist jetzt noch keine bahnbrechende Entdeckung, denn das kennen wir von allen möglichen alltäglichen Vorgängen: Das Spülen von 20 Tellern dauert auch länger als das von zehn Stück. Aber vielleicht gibt es ja noch andere Faktoren für den Zeitbedarf? Allgemein ist für die Informatik die Identifizierung einer Problemgröße sehr wichtig.

Die Problemgröße

Ein wichtiger Vorgang bei der Lösung von Aufgaben aus der Informationstechnik ist das Bestimmen der sogenannten Problemgröße. Sie stellt ein Maß dar, wie schwierig bzw. umfangreich die Aufgabe bzw. das Problem ist. Daher hängt von der Problemgröße entscheidend der zu leistende Aufwand ab.

In vielen Fällen ist die Problemgröße recht offensichtlich – wie die Anzahl von Städten einer Landkarte für die Bestimmung eines kürzesten Weges. In anderen Fällen

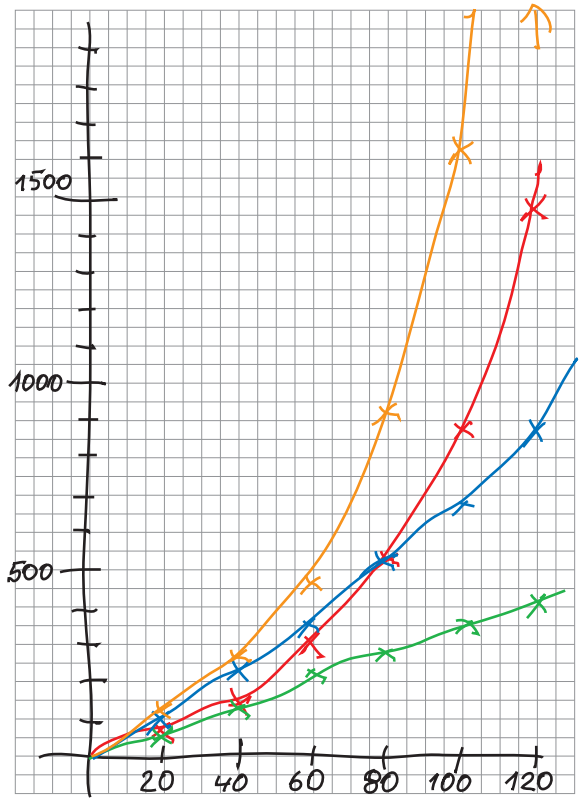


Abbildung 2.1
Graph Anzahl der Karten vs. Zeit in Sekunden.

müssen wir erst einmal darüber nachdenken, und die Problemgröße hängt nicht nur von der Aufgabe selbst ab, sondern auch davon, was sich im Allgemeinen bei gleich bleibendem Aufgabentyp an der tatsächlichen Aufgabenstellung ändert:

Beim Sortieren von Objekten ist ein wesentlicher Faktor die Anzahl der Objekte! Das konnten wir in unserem kleinen Experiment feststellen. Aber auch die Größe der zu sortierenden Elemente könnte als Problemgröße dienen: Es ist sicherlich aufwendiger, 100-stellige Zahlen zu sortieren als 6-stellige ...

Daher müssen wir uns überlegen, was in einer Aufgabenstellung aus der Realität normalerweise feststeht und was variabel ist. Die Größe der zu sortierenden Elemente ist dabei meistens recht stabil: Geburtsdaten, Kontonummern, Namen usw. haben eine recht definierte Länge. Wenn wir als IT-Unternehmen ein neues Verfahren für die Sortierung von Daten anbieten, könnten wir also mit dem Slogan „Wir sortieren Kontonummern mit der doppelten Länge in der gleichen Zeit wie unsere Konkurrenten“ sicherlich weniger punkten als mit der Aussage „Wir sortieren die doppelte Anzahl von Kontonummern in der gleichen Zeit wie unsere Konkurrenten.“

Für das Sortieren nehmen wir daher im Folgenden immer die Anzahl der Objekte als Problemgröße an.

Problemgrößen und Aufwand

Von der Problemgröße hängt der Aufwand zum Erfüllen der Aufgabe ab. Wie verhält sich das denn nun in unserem Sortier-Experiment? Normalerweise geht man ja

intuitiv vom sogenannten linearen Verlauf aus: Pro Objekt wird eine bestimmte Zeit benötigt – zum Spülen von 100 Tellern benötigt man also 100-mal die Zeit, die für einen einzelnen Teller notwendig ist. Unter Umständen sogar etwas weniger, weil man ja langsam Übung bekommt und so für die letzten Teller kürzer braucht.

Schauen Sie sich daraufhin einmal Ihr Diagramm bzw. Ihre Zahlen an. Benötigten Sie zum Sortieren von 40 Karten in etwa doppelt so lange wie für 20? Haben Sie 100 Karten in der fünffachen Zeit von 20 sortiert? Im Beispiel wird das von der grünen Linie ausgedrückt. Wenn Sie das wirklich geschafft haben, sind Sie schon ein Spitzen-Sortierer und heben sich von 99 % aller anderen positiv ab.

Normal ist, wenn Sie für die Sortierung von 40 Karten knapp dreimal so lange benötigen wie für 20, nicht die doppelte Zeit. Für 60 Karten ist die fünf- bis sechsfache Zeit fällig usw. Das sieht man im Beispiel in etwa an der roten und der blauen Linie. Manche Sortierer verdoppeln jeweils sogar die benötigte Zeit, wenn sie einen weiteren Stapel Karten mit sortieren. Das trifft ungefähr für die gelbe Linie zu.

Versuchen Sie selbst, dieses Phänomen zu erklären! Was ist der entscheidende Unterschied zwischen dem Sortieren von Karten und dem Spülen von Tellern?

Divide et impera
ist weder eine Erfindung der Informatiker noch etwas, das (wie man bei lateinischen Sprüchen denkt) bereits die Römer in die Welt setzten. Der Satz wird König Ludwig XI. von Frankreich (1423 – 1483) zugeschrieben, der ihn als Maxime hatte. Es handelte sich ursprünglich um eine Militärstrategie, den Gegner zu entzweien, um ihn dann leichter zu beherrschen. Die Informatik hat ihn dann für ihre friedlichen Zwecke umfunktioniert: eine Aufgabenstellung beherrschbarer zu machen, indem man sie einteilt.

Der Unterschied lässt sich auf eine sehr grundlegende Methode der Informatik zurückführen: „divide et impera“ oder zu Deutsch „teile und herrsche“. Diese Methode ist allerdings gar nicht so neu, sondern wird von fast jedem seit Jahrhunderten für die Erledigung alltäglicher Arbeiten genutzt. Lesen Sie selbst:

Das Prinzip „divide et impera“

Sehr häufig hat man im Leben Probleme zu lösen, die zu unüberschaubar und groß sind, um sie in einem Ansatz zu lösen. Vielmehr teilen wir das Gesamtproblem auf in mehrere, handhabbare Stücke, die wir lösen. Die Teillösungen werden danach nur noch zusammengefasst.

Dieses Prinzip wird auch in der Informatik sehr stark verwendet: Ein Programm zerlegt die gestellte Aufgabe zunächst in mehrere kleinere Einheiten, genannt Teilprobleme (divide = teile), und weist danach andere Programme an, diese zu lösen (impera = herrsche, befehlige). Dabei ist sehr wichtig, dass die Teilprobleme unabhängig voneinander gelöst werden können, denn sonst müssten die Programme miteinander kommunizieren, unter Umständen auf Lösungen voneinander warten, was den Aufwand wiederum sehr erhöht.

Unterschiedliche Probleme können unterschiedlich gut aufgeteilt werden. Das Tellerwasch-Problem stellt hier keine Herausforderung dar: „100 Teller waschen“ hat direkt als Teilproblem „1 Teller waschen“. Dieses wird 100-mal ausgeführt. Jeder Teller kann unabhängig voneinander gewaschen werden, der Gesamtaufwand besteht also in der Summe der einzelnen Aufwände, also 100-mal der Zeit, die zum Waschen eines Tellers benötigt wird. Man kann die Zeit sogar durch Parallelisierung optimieren: Stellen wir 100 Tellerwäscher ein, schaffen wir das Problem in der Zeit, die man zum Waschen eines einzelnen Tellers benötigt ...

Wie sieht das aber beim Sortieren von Karten aus? Probieren wir, auch dieses Problem auf die gleiche Weise aufzuteilen. Wir gehen davon aus, pro Arbeitsschritt eine (un-

sortierte) Karte auf die Hand zu nehmen, die wir dann in die Folge bereits sortierter Karten einordnen.

„100 Karten sortieren“ hat direkt als Unterproblem „eine Karte sortieren“. Wie lange benötigen wir aber, um eine Karte korrekt zu sortieren? Das hängt sehr stark von der Anzahl Karten ab, die bereits auf dem Tisch liegen!

Die erste Karte geht sehr schnell, wir können sie einfach hinlegen. Bei der zweiten Karte müssen wir bereits die Entscheidung treffen, ob sie links oder rechts angelegt wird – der Aufwand hat sich vergrößert.

Befinden sich bereits 99 Karten auf dem Tisch, müssen wir die neue Karte mit sehr vielen der bereits sortierten Karten vergleichen, um die richtige Position herauszufinden.

Hier liegt der große Unterschied: Der Aufwand zum Lösen eines Teilproblems ist nicht mehr unabhängig von der Problemgröße, wie das beim Tellerwaschen der Fall war! Je mehr Karten wir sortieren, desto länger dauert im Mittel das Sortieren einer Karte. Daher steigt der Aufwand mit jeder Karte, die zu unserer Sortierung hinzukommt, einerseits um den Faktor „die Anzahl der Karten ist höher, daher muss die Anzahl der Sortiervorgänge erhöht werden“, aber zusätzlich auch um den Faktor „jeder Sortiervorgang dauert im Mittel etwas länger“.

Eine der wesentlichen Aufgaben eines Informatikers ist, diese Erhöhung des Zeitaufwands zu erforschen und dieses Wissen zu nutzen, um die verwendeten Programme zu verbessern, so dass auch hohe Problemgrößen noch bewältigt werden können.

Sortieren auf Computerisch ...

Egal, welche verschiedenen Strategien Sie und Ihre Versuchspersonen bisher angewandt haben – diese sind sicherlich in der einen oder anderen Weise auch in der Informatik bekannt, es existieren also Sortieralgorithmen dafür.

Allerdings fällt es schwer, diese zu erkennen, wenn man jemandem beim Sortieren von Karten zuschaut. Ein Mensch hat einfach zu viele Freiheitsgrade, muss beim Agieren keine strikten Regeln einhalten. Hier hilft oft, einfach Computer zu spielen:

Überlegen Sie, welchen Beschränkungen ein Computer unterworfen ist, wenn er in seinem Speicher Zahlen sortieren soll. Wie könnte man diese Beschränkungen in eine Art „Spiel“ umsetzen, so dass man mit ähnlichen Voraussetzungen die Karten sortiert? Wenn Sie ungefähr wissen, wie der Speicher eines Computers funktioniert, versuchen Sie zunächst, die Frage selbst zu beantworten, ansonsten lesen Sie bitte einfach weiter.

Der größte Unterschied zwischen Mensch und Maschine besteht beim Sortieren in der Speicherung: Der menschliche Kartensortierer besitzt quasi überall „Speicher“ – auf der Hand, als Stapel auf dem Tisch, ausgelegt auf dem Tisch usw. Beim Computer ist der Speicher streng in Positionen organisiert und jede Position, genannt Speicherstelle oder Speicheradresse, kann genau einen Wert bzw. in unserem Spiel eine Karte beinhalten.

Das können wir glücklicherweise als Experiment simulieren. Die Doppelseite mit Abbildung 2.K2 ist Ihr eigener Computerspeicher für die Karten aus der Kopiervorlage. Zunächst kommen wir mit dem gelben Bereich mit den Speicherstellen **#32** bis **#63** aus. Wenn Sie wollen, können Sie aber den Bogen mehrfach kopieren, um einen noch viel längeren Bereich zu erhalten! Die grünen Buchstaben auf den ersten Speicherstellen sollen genauso wie die roten Zahlenbereiche erst einmal unbeachtet bleiben!

Bringen Sie nun ein paar Karten in den neuen Speicher. Für Sie gilt selbstverständlich die Spielregel: Karten dürfen nur genau auf Speicherpositionen liegen, nicht dazwischen. Legen Sie zwei beliebige Karten auf die Adressen **#32** und **#33**.

Den Inhalt der Speicheradressen können Sie auslesen – einfach darauf schauen. Eine Besonderheit gibt es allerdings beim Schreiben! Was passiert, wenn Sie einen zusätzlichen Wert auf Position **#33** schreiben (also eine Karte auf diese legen)? Bei unserem

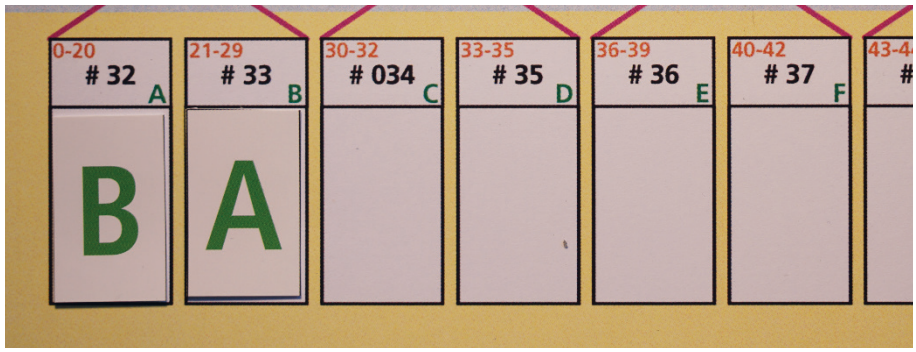


Abbildung 2.2
Papierspeicher mit unsortierten Karten

Papierspeicher passiert nicht viel: Eine Speicheradresse kann prinzipiell als Stapel beliebig viele Karten fassen. Nimmt man die oberen Karten wieder weg, kommen die unteren wieder zum Vorschein.

Ein Computer funktioniert anders. Jede Adresse kann genau einen Wert speichern. Gibt man einen neuen Wert hinein, überschreibt man damit den alten Inhalt. Wir müssen also noch als Spielregel festlegen, dass auf jeder Speicheradresse nur eine Karte liegen darf: Sobald eine neue Karte auf einer Position abgelegt wird, kommt die eventuell bereits dort liegende Karte auf den Ablagestapel und ist aus dem Spiel.

Probieren Sie das versuchsweise anhand eines sehr einfachen Experiments: Die Speicheradressen #32 und #33 sind momentan jeweils mit einer Karte belegt. Tauschen Sie den Inhalt dieser beiden Positionen aus und halten Sie sich dabei genau an die Spielregeln.

Haben Sie das Problem erkannt? Normalerweise würden wir für den Austausch einfach eine der Karten in die Hand nehmen, die andere verschieben und dann die Karte aus der Hand auf den verbliebenen Platz legen. Dadurch hätten wir aber die Hand als „Speicher“ genutzt – das ist gegen die Spielregel, nach der Karten nur auf Speicherpositionen des Papierspeichers liegen dürfen.

Nehmen wir dagegen eine der Karten und legen sie direkt auf die andere Position, wird diese überschrieben, die Karte mit dem anderen Wert kommt aus dem Spiel – sie ist verloren und der Austausch nicht mehr möglich.

Folgerung daraus ist, dass wir den zusätzlichen Speicherplatz („die Hand“) wirklich benötigen: Legen Sie die Karte von Speicherstelle **#33** zunächst auf Position **#34** ab. Danach können Sie die Karte von **#32** auf **#33** verschieben und jetzt erst die Karte von **#34** auf **#32** legen. Das entspricht unseren Spielregeln und exakt so funktioniert die Sache auch im Computer.

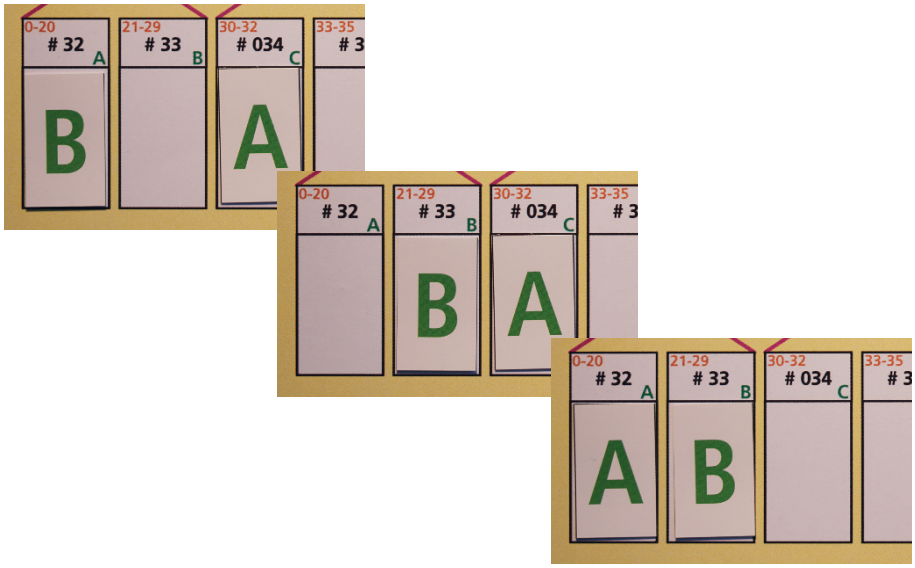


Abbildung 2.3
Kartentausch nur mit Hilfe einer zusätzlichen Speicherstelle

Für viele Vorgänge wird ein Zwischenspeicher benötigt. Dafür besitzen die meisten Computer auch spezielle Vorkehrungen – Speicheradressen, die besonders schnell und komfortabel abgefragt werden können. Wie setzen wir das im Papierspeicher um? Wo haben wir im echten Leben einen komfortablen Zwischenspeicher für Karten? Wie wäre es mit folgender Erweiterung der Spielregel: Sie dürfen nun maximal eine Karte in der Hand behalten, während Sie die anderen Karten im Speicher verschieben.

Zeit für unsere nächsten Schritte als Computer-Simulator. Wählen Sie zufällig sechs der grünen Spielkarten (mit einfachen Buchstaben) aus und legen Sie diese unsortiert auf die Adressen **#32** bis **#37**. Abbildung 2.4 zeigt ein Beispiel.

Nun wollen wir die Folge alphabetisch sortieren. Eine Möglichkeit dafür ist, von vorne Ordnung zu schaffen. **A** muss also ganz nach vorne. Nehmen Sie daher **A** in die Hand, Ihren legalen Zwischenspeicher, und schieben Sie nacheinander **T** auf die Position **#36**, **S** auf **#35**, **E** auf **#34** und **L** auf **#33**. Nun ist **#32** leer, wir können **A** aus dem Zwischenspeicher dort ablegen. Das Verfahren können wir für **D** wiederholen, also **D** aufnehmen, **LEST** von hinten nach vorne je um eine Position verschieben und dann **D** auf der freien Position **#33** ablegen. Abbildung 2.5 zeigt, wie die Folge jetzt sein sollte.

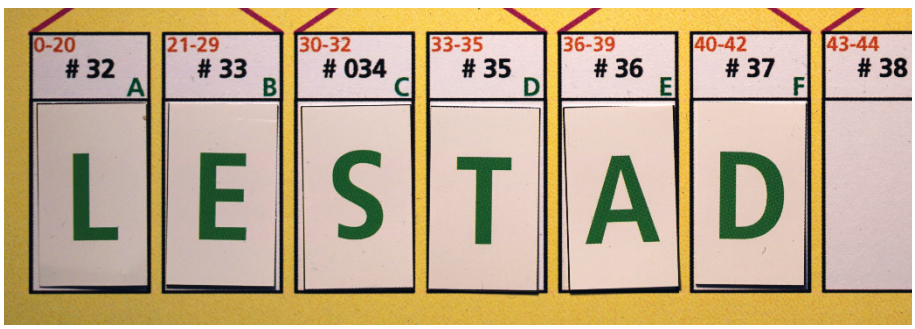


Abbildung 2.4
Unsortierte Karten im Speicher

Abbildung 2.5
Nach den ersten Sortierschritten

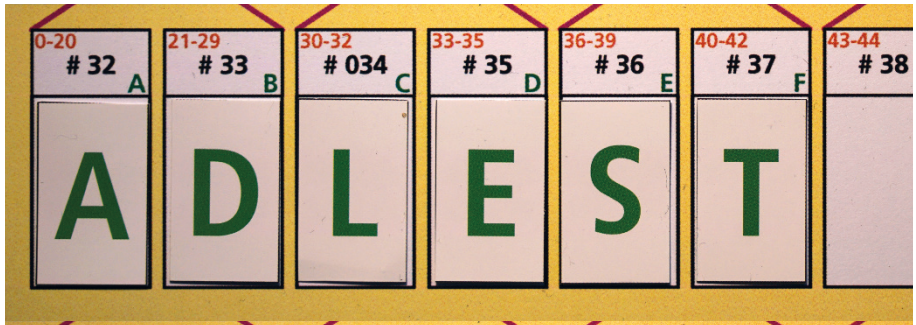
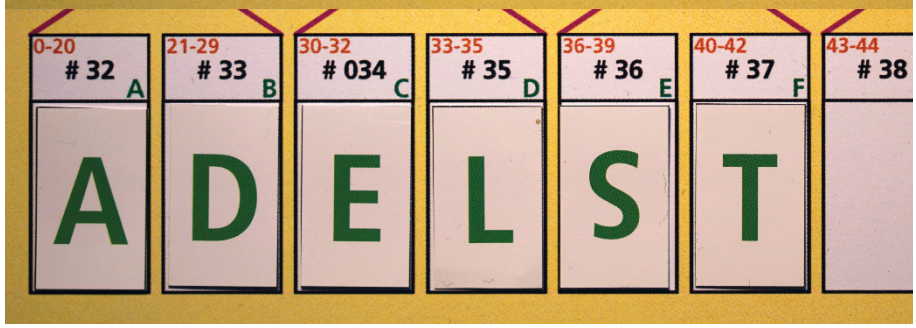


Abbildung 2.6
Die sortierte Folge



Nun müssen nur noch **L** und **E** getauscht werden und die Folge ist perfekt sortiert, wie Abbildung 2.6 zeigt.

War das eine korrekte Vorgehensweise, wenn man einen Computer simulieren möchte? Betrachten Sie für die Antwort erst einmal nur den allerersten Schritt: **A** muss ganz nach vorne.

Zwei Erkenntnisse haben wir hier genutzt, die wir als Menschen sofort haben, die der Computer aber nicht ohne Weiteres besitzen kann:

1. **A** ist im Speicher die Karte mit dem alphabetisch kleinsten Buchstaben.

2. Die Karte mit dem kleinsten Buchstaben liegt auf Position **#36**.

Woher wissen wir das? Weil wir natürlich alle Karten gleichzeitig sehen und bereits unser Unterbewusstsein spontan die Situation für uns analysiert.

Ein Computer kann zwar auf seinen gesamten Speicher zugreifen, aber im Normalfall immer nur gleichzeitig eine Adresse anschauen. Es ist für ihn daher im Normalfall ein aufwendiger Prozess, herauszufinden, welche der vorhandenen Karten denn nun ganz nach vorne muss.

Wir fügen das auch noch unseren Spielregeln hinzu. Dazu benötigen wir die Pfeile aus den Bastelbögen.Wenn wir eine Speicherstelle auslesen wollen, muss immer ein Pfeil auf die entsprechende Adresse zeigen. Alle anderen Adressen gelten als unsichtbar.

Ich verspreche: Jetzt haben wir wirklich alle Regeln festgelegt, um Computer zu spielen, und dürfen mit der Umsetzung des ersten Sortierverfahrens beginnen.

Alles, was Sie hier erforschen, einschließlich der offenbar ungünstigeren Vorgehensweisen, ist übrigens tatsächlich in heutigen Computersystemen wiederzufinden. Besonders in Bereichen, in denen an korrekten Programmen viel Geld hängt, folgt man gerne dem Informatik-Motto „never change a running system“, also „Verändere nie ein System, wenn es (doch) läuft.“ Daher finden sich in den Programmen oft noch Prinzipien, die an anderen Stellen bereits durch neue Ideen ersetzt wurden.

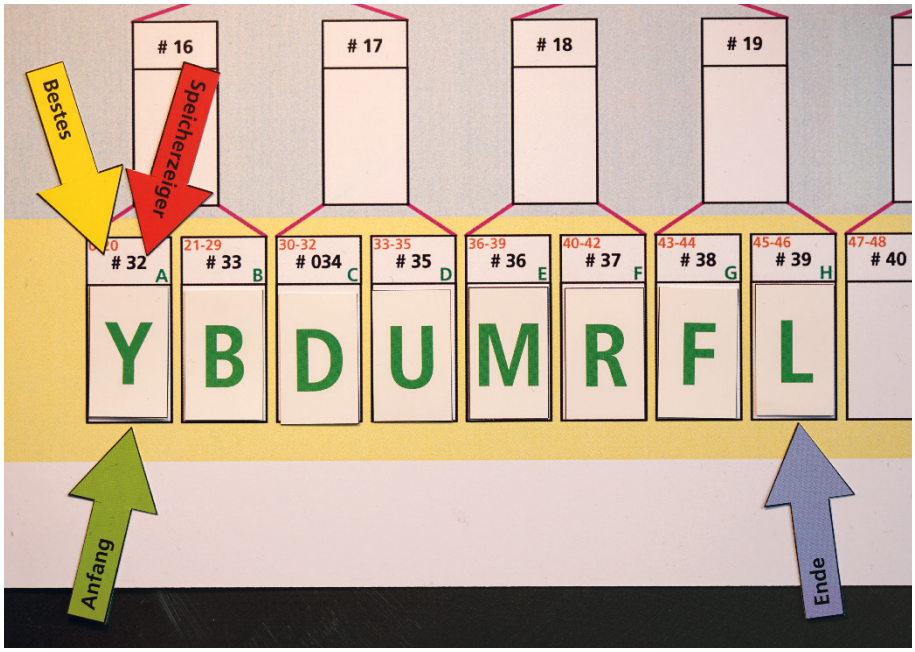


Abbildung 2.7
Sortierfolge mit Zeigern

Selection-Sort

Aufgabe ist, die Folge von sechs Buchstabenkarten im Speicher von Anfang bis zum Ende zu sortieren. Abbildung 2.7 zeigt die Aufstellung mit den zur Verfügung stehenden Zeigern „Speicherzeiger“ sowie „Anfang“, „Ende“ und „Bestes“.

Wie bereits oben dargestellt, ist ein immer wiederkehrendes Prinzip der Informatik die Unterteilung eines großen Problems in kleinere, handhabbare Portionen. Eine Unterteilung des Sortierproblems könnte also lauten:

Sortieren von n Karten:

- Sortiere die Karte mit dem kleinsten Buchstaben an den Anfang
- Sortiere die restlichen $n - 1$ Karten

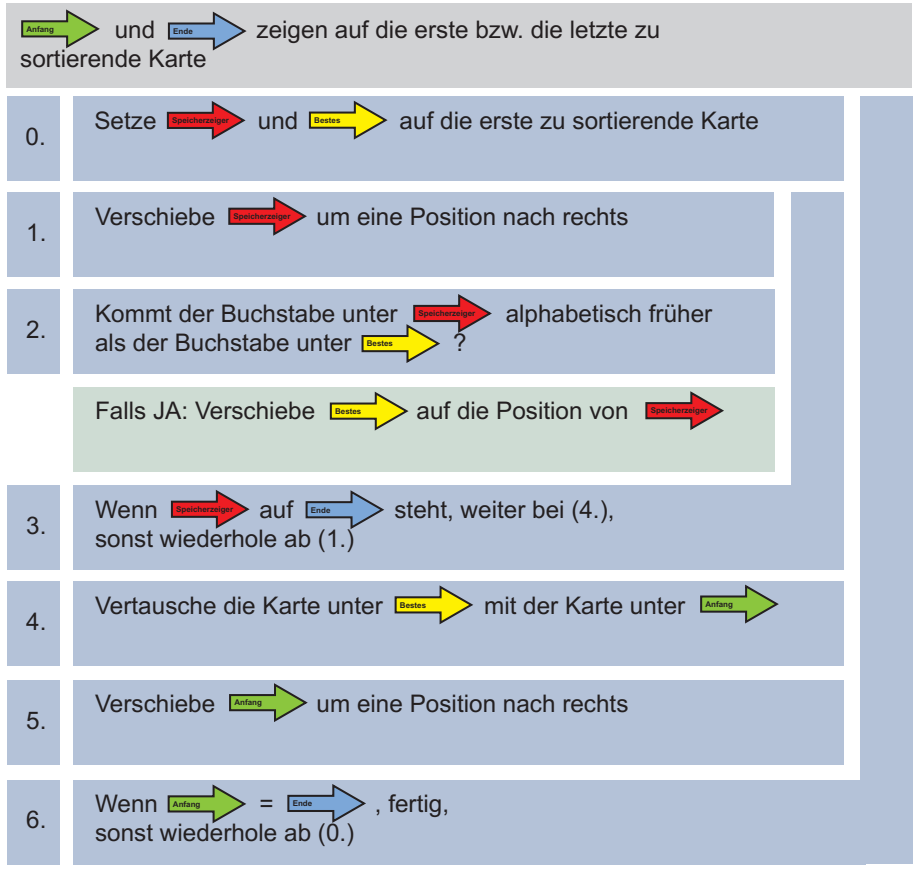
Diese Vorgehensweise nennt man auch „Sortieren durch Auswahl“ oder mit englischem Fachbegriff „Selection-Sort“, weil immer eine Karte ausgewählt und korrekt einsortiert wird.

Versuchen Sie nun, diese Vorschrift zu konkretisieren. Spielen Sie mit den Materialien und formulieren Sie einen allgemeinen Algorithmus für Selection-Sort. Die Zeiger helfen Ihnen dabei, Positionen zu markieren. Sie dürfen sie im Algorithmus benutzen.

Sie können eine aus dem ersten Kapitel bekannte Schreibweise verwenden oder einfach die Vorschrift so aufschreiben, wie es für Sie selbst am besten nachvollziehbar ist.

Ein Vorschlag ist die ausformulierte Vorschrift nach Abbildung 2.8.

Abbildung 2.8
Die hier verwendete Notation für Algorithmen ist eine vereinfachte Form der von Isaac Nassi und Ben Shneiderman entwickelten modularen Darstellungsart. Im deutschen Sprachraum ist sie unter den Bezeichnungen „Nassi-Shneiderman-Diagramm" oder „Struktogramm" bekannt.



In den Anweisungen (1.) bis (3.) wird der kleinste Buchstabe gesucht und gefunden. Dieser muss ganz an den Anfang und wird daher getauscht mit der Karte, die den Platz dort belegt (4.).

Nun weiß man, dass die Karte an der ersten Position korrekt liegt, dass diese also nie mehr verschoben werden muss: Sie gehört nicht mehr zum zu sortierenden Bereich. Daher rücken wir bei (5.) den Anfangszeiger um eine Position nach rechts und wiederholen das Sortieren so lange, bis die unsortierte Folge nur noch aus einer einzelnen Karte besteht. Eine Karte ist aber sowieso immer sortiert, daher sind wir fertig!

Ich will das Vorgehen hier gar nicht weiter erklären. „Begreifen" Sie lieber den Algorithmus, indem Sie mehrere zufällige Folgen von Karten sortieren. Nehmen Sie hierfür auch gerne die Karten mit den echten Zahlen. Sie können sowohl die rote als auch die blaue Reihe nutzen – wenn Sie das abwechselnd tun, sparen Sie sich das Mischen zwischendurch.

Besser durch Blubberblasen?

Damit wir noch etwas zum Spielen haben, erkläre ich gleich ein weiteres, sehr bekanntes Sortierverfahren, das wegen seiner Einfachheit besonders in älteren Programmen recht häufig eingesetzt wird.

Die generelle Idee ist dabei: Gehe die Reihe mit Karten von Anfang bis Ende durch. Vergleiche dabei immer zwei direkt nebeneinander liegende Karten. Sind diese richtig sortiert, mache gar nichts, ansonsten vertausche beide.

Können Sie wieder das grundlegende Prinzip der Informatik erkennen? Jede Vertauschung stellt etwas mehr Ordnung her, bis alle kleinen Fortschritte zusammen die gesamte Arbeit vollbracht haben.

Versuchen Sie das anhand einer zufälligen Reihe, wie in Abbildung 2.9.

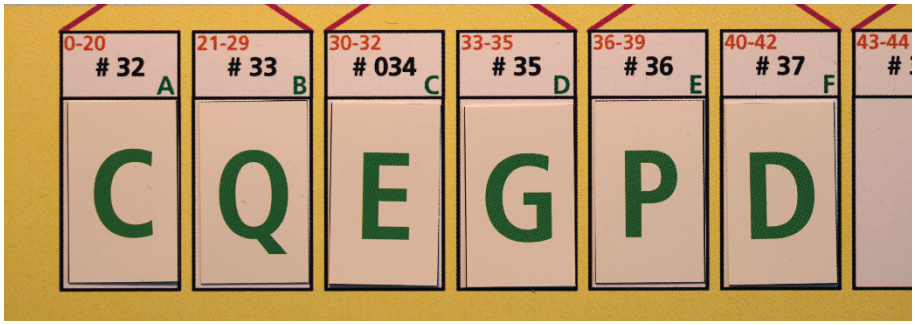


Abbildung 2.9
Unsortierte Folge

Nach dem Durchgehen der Reihe ist die Folge bereits etwas mehr sortiert (Abbildung 2.10).

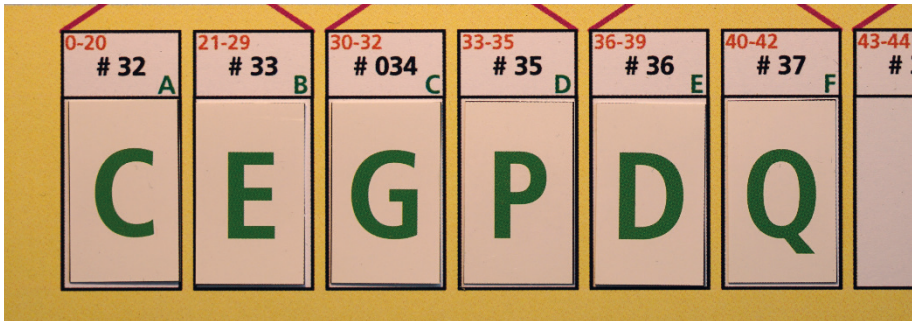


Abbildung 2.10
Folge nach einem Durchlauf des Verfahrens

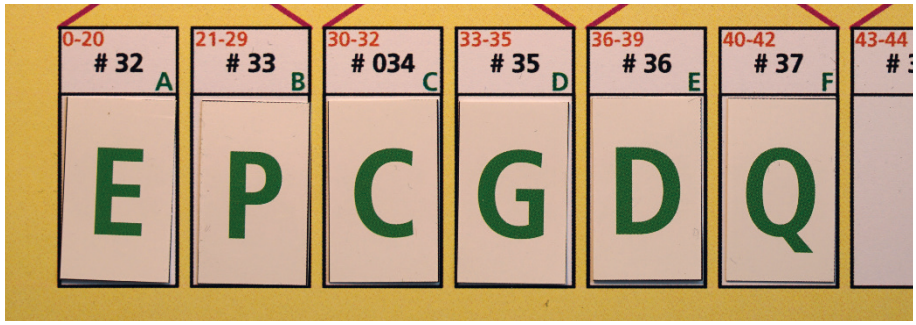
Überlegen Sie, was genau Sie über die Folge nach dem ersten Durchlauf sagen können und ob man diese Aussage so formulieren kann, dass sie für jede beliebige Kombination von Karten gilt, nachdem Sie den Algorithmus einmal durchgeführt haben.

Überlegungen dieser Art müssen Informatiker oft anstellen: das Schließen auf eine allgemeine Regel aufgrund eines Beispiels.

In diesem Fall sind nach dem ersten Durchlauf zwei Karten korrekt sortiert: **C** und **Q**. Ist das aber immer so? Nein! Nimmt man als Ausgangsfolge zum Beispiel **PEQCGD**, kommt nach dem ersten Durchlauf **EPCGDQ** heraus, wie in Abbildung 2.11 dargestellt. **C** steht also nicht am Anfang, aber wiederum **Q** am Ende.

Wenn Sie darüber nachdenken, muss das auch so sein: Der Reihe nach wird durch das Vertauschen zweier Karten die größere der beiden immer nach rechts gebracht. Die größte Karte wechselt daher auf die Position ganz rechts, wo sie hingehört.

Abbildung 2.11
Unsortierte Folge



Auch dieses Sortierverfahren bringt also pro Durchlauf eine Karte an die richtige Stelle. Wenn wir es so oft durchführen, wie Karten vorhanden sind, müssen also danach alle Karten sortiert sein.

Versuchen Sie nach diesem Prinzip einen Algorithmus zu formulieren, in ähnlicher Weise wie oben Selection-Sort. Sie können wieder die Zeiger Start, Ende und Speicherzeiger verwenden, um den Bereich zu markieren, der noch unsortiert ist.

Eine mögliche, in der Praxis häufig verwendete Formulierung sehen Sie in Abbildung 2.12: Wenn in einer Anweisung „Wiederhole Folgendes ...“ vorkommt, bezieht sich „Folgendes“ auf den eingerückten Text darunter.

Der Algorithmus nennt sich übrigens „Bubblesort“, weil man sich vorstellt, dass die Karten durch den immerwährenden Tausch an die korrekte Position kommen, so wie Luftblasen im Wasser an die Oberfläche steigen.

Gut, besser, bester?

Das Finden verschiedener Algorithmen zur Lösung eines Problems ist für Informatiker sehr wichtig! Mindestens genauso wichtig ist jedoch, unter allen möglichen Algorithmen den besten zu finden.

Für einen guten Algorithmus kann es selbstverständlich sehr viele verschiedene Kriterien geben, zum Beispiel wie viel Programmieraufwand man benötigt oder wie elegant ein entsprechendes Computerprogramm aussieht. Für manche Anwendungen gilt sogar, dass ein unübersichtliches Programm sehr viel besser ist als ein übersichtliches, weil es von Konkurrenten schwerer abgeschrieben werden kann.

Normalerweise untersuchen wir jedoch immer die Zeit, die ein Algorithmus zur Lösung einer bestimmten Problemgröße benötigt – je schneller, desto besser. Genau wie bei den Menschen gibt es selbstverständlich auch Computer, die mehr oder weniger zügig arbeiten, und so kann das gleiche Verfahren auf unterschiedlichen Rechnern unterschiedlich viel Zeit in Anspruch nehmen. Daher müssen wir eine vergleichbare, nur auf den Algorithmus, nicht auf die Maschine zutreffende Basis schaffen:

Hierfür legen wir einfach fest, welche Schritte des Algorithmus im Computer Zeit kosten. Man spricht hier auch von Operationen. Diese werden dann bei der Durchführung einfach gezählt.

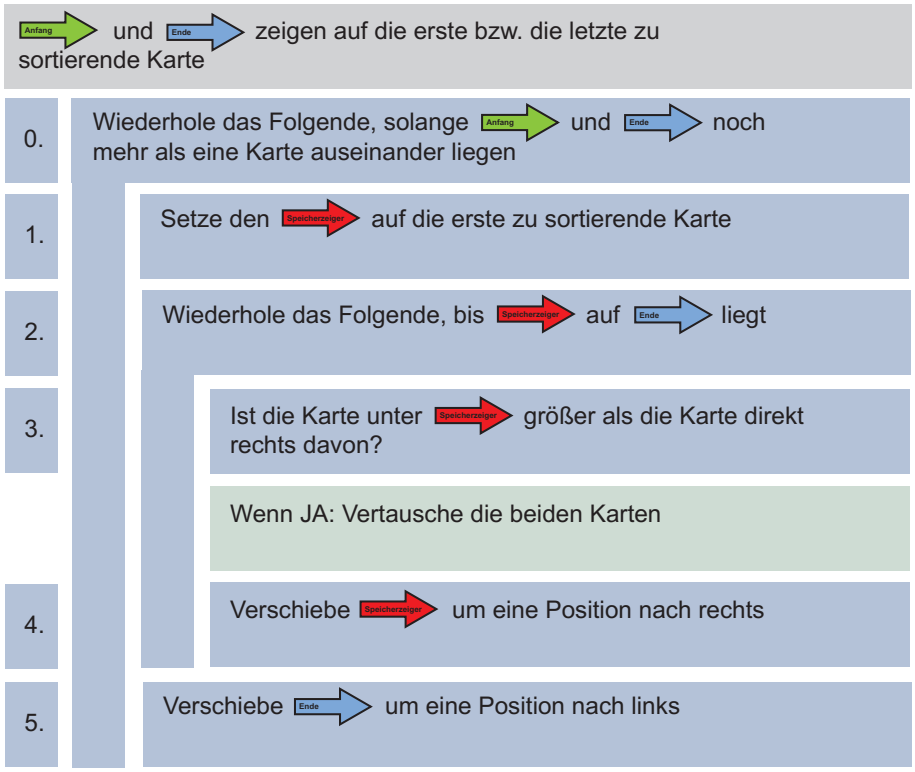


Abbildung 2.12
Ein mögliches Struktogramm für Bubblesort

Da normalerweise Zugriffe auf den Speicher immer besonders zeitintensiv sind, können für unseren Papierspeicher folgende Schritte ausgewählt werden:

- Verschieben einer Karte auf eine neue Speicherposition
- Vergleich zweier Karten

Selbstverständlich gibt es noch viele weitere Operationen, wie zum Beispiel Verschieben eines Zeigers, Wenn-dann-Anweisung usw. Wie bereits oft in diesem Buch verwenden wir jedoch das Prinzip der Abstraktion, um uns das Leben so einfach wie möglich zu machen – in diesem Fall, indem wir nur die relevanten Operationen auswählen.

Elementaroperationen

Operationen bzw. Arbeitsschritte, die wir bei der Bestimmung der Laufzeit eines Algorithmus für wichtig bzw. zeitkritisch erachten. Welche Operationen hier ausgewählt werden, ist nicht global definiert, sondern vom jeweiligen Anwendungsfall abhängig. Oft handelt es sich um Speichervorgänge (Lesen und Schreiben im Speicher).

Benchmark-Tests
Wir beschäftigen uns hier damit, die Laufzeit von Algorithmen durch theoretische Betrachtungen abzuschätzen. Ein anderer Ansatz ist, es einfach auszuprobieren. Das nennt man „Benchmarking" (von engl. Benchmark = Maßstab). Hier werden die zu testenden Programme (oder auch ganze Systeme) mit Testdaten laufen gelassen. Der benötigte Zeitaufwand zum Lösen der Aufgaben wird gemessen und verglichen.

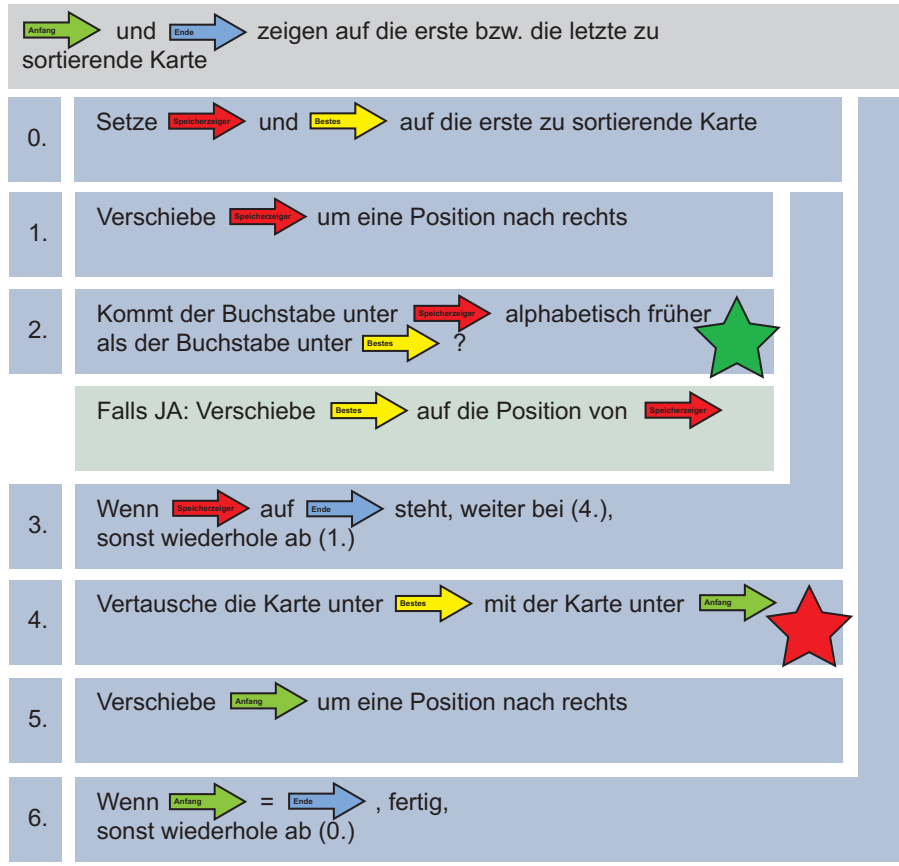
Preisfrage: Wie viele unserer Elementaroperationen benötigt das weiter oben beschriebene Verfahren zum Tausch zweier Karten im Papierspeicher?

Um die Karten auf den Speicherpositionen A und B zu tauschen, sind drei Verschiebeoperationen nötig:

- Verschiebe Karte von A auf Zwischenspeicher
- Verschiebe Karte von B nach A
- Verschiebe Karte von Zwischenspeicher nach B

Wenn wir auf diese Weise die bisher aufgeschriebenen Algorithmen für Selection-Sort und Bubblesort analysieren, können wir an den mit farbigen Sternen markierten Stellen Elementaroperationen identifizieren. Grün steht für einen Vergleich (1 Elementaroperation), Rot für den Austausch zweier Speicherpositionen (3 Elementaroperationen). Die Abbildungen 2.13 und 2.14 zeigen die Struktogramme.

Abbildung 2.13
Selection-Sort mit markierten Elementaroperationen



Jetzt sind Sie an der Reihe: Überprüfen Sie experimentell, welches Sortierverfahren das bessere ist, indem Sie verschiedene Kartenanzahlen mit beiden Verfahren sortieren. Notieren Sie sich dann (zum Beispiel mit einer Strichliste) jedes Mal den Vergleich zweier Karten und (dreifach) den Austausch zweier Karten. Wenn Sie die bunten Sortierkarten nutzen, können Sie abwechselnd die roten und blauen Zahlen nutzen, um dazwischen nicht immer wieder mischen zu müssen.

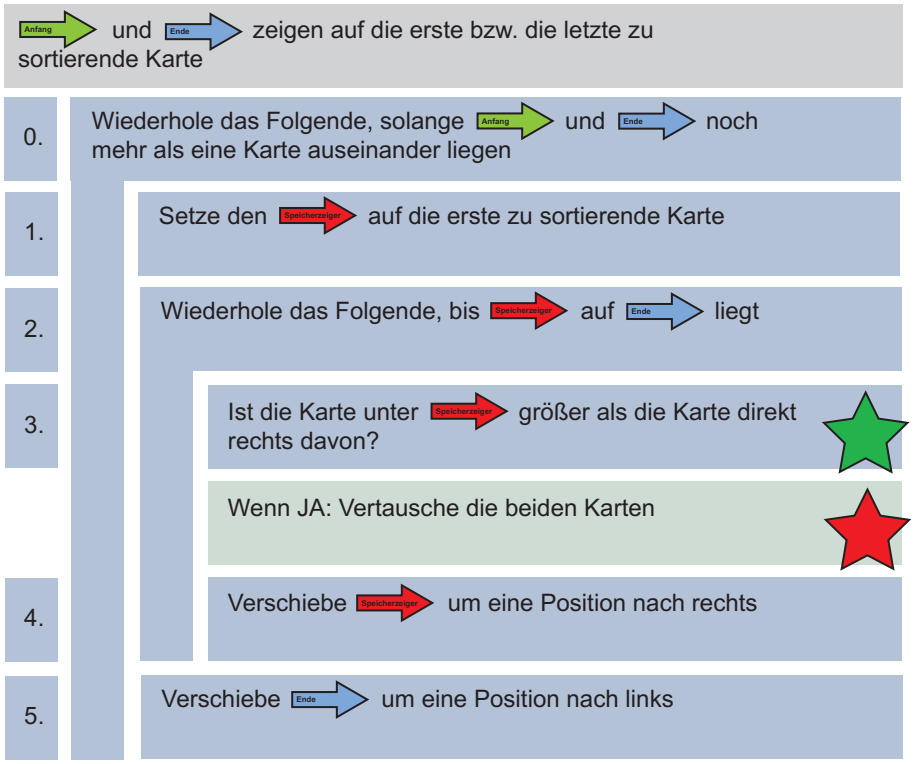


Abbildung 2.14
Bubblesort mit markierten Elementaroperationen

Selbstverständlich ist das Ergebnis davon abhängig, wie gut die Karten jeweils gemischt waren. Wenn man zufällig eine bereits sortierte Folge als Ausgangssituation hat, dann benötigt man zum Beispiel gar keine Vertauschungen. Bis auf solche Spezialfälle kann man an der entstehenden Tabelle jedoch recht gut ablesen, wie die Rechenzeit in Abhängigkeit der Problemgröße ansteigt.

Bei meinen Versuchen mit bis zu 35 Karten bin ich auf Ergebnisse nach der Tabelle auf der nächsten Seite gekommen. Dabei habe ich in gesonderten Spalten auch den „Aufwand pro Karte“ vermerkt, indem ich die Anzahl der Elementaroperationen nochmals durch die Anzahl der Karten geteilt habe.

Nur einige der vielen bekannten Sortieralgorithmen:
Bogosort
Bubblesort
Cocktailsort
Combsort
Gnomesort
Heapsort
Insertionsort
Introsort
Mergesort
Quicksort
Selection-Sort
Shellsort
Smoothsort
Stoogesort
Tournament-Sort

Wie kann man dies interpretieren? Genau wie beim komplett manuellen Sortieren, wie wir es ganz am Anfang durchgeführt haben, steigt der Aufwand pro Karte mit der Anzahl der Karten. Man sagt, die Verfahren sind nicht linear, was bedeutet, dass sich der Aufwand nicht mit einem konstanten Faktor aus der einfachen Problemgröße ableiten lässt.

Weiterhin kann man ablesen, dass Bubblesort fast durchgehend schlechter ist als Selection-Sort: Der Aufwand pro Karte ist bei Bubblesort ungefähr doppelt so groß wie bei Selection-Sort.

Der folgende Abschnitt (bis zur nächsten Überschrift) erfordert etwas mathematisches Vorwissen. Daher ist es nicht schlimm, wenn Sie ihn gegebenenfalls nicht komplett verstehen oder gleich ganz überspringen möchten.

Insgesamt wächst der Aufwand pro Karte bei beiden Verfahren etwa linear zur Problemgröße n. Bei Selection-Sort kann man ihn etwa mit $0{,}6 \cdot n$ abschätzen. Bei Bubblesort sind es etwa $1{,}3 \cdot n$.

Tabelle
Aufwand für das Sortieren verschiedener Kartenzahlen

Karten (=Problemgröße)	Selection-Sort	Bubblesort	Selection-Sort pro Karte	Bubblesort pro Karte
2	4	4	2,0	2,0
3	9	12	3,0	4,0
4	15	6	3,8	1,5
5	22	22	4,4	4,4
6	30	33	5,0	5,5
7	39	57	5,6	8,1
8	49	79	6,1	9,9
9	60	111	6,7	12,3
10	72	96	7,2	9,6
11	85	127	7,7	11,5
12	99	201	8,3	16,8
13	114	183	8,8	14,1
14	130	241	9,3	17,2
15	147	273	9,8	18,2
16	165	309	10,3	19,3
17	184	403	10,8	23,7
18	204	366	11,3	20,3
19	225	441	11,8	23,2
20	247	535	12,4	26,8
21	270	594	12,9	28,3
22	294	534	13,4	24,3
23	319	616	13,9	26,8
24	345	756	14,4	31,5
25	372	828	14,9	33,1
26	400	787	15,4	30,3
27	429	861	15,9	31,9
28	459	1014	16,4	36,2
29	490	1006	16,9	34,7
30	522	1074	17,4	35,8
31	552	1194	17,8	38,5
32	586	1251	18,3	39,1
33	620	1188	18,8	36,0
34	653	1335	19,2	39,3
35	690	1426	19,7	40,7

Für den Gesamtaufwand T müssen wir natürlich noch mit der Anzahl der Karten n multiplizieren und kommen auf

$T_{\text{Selection-Sort}}(n) \approx 0{,}6 \cdot n^2$ sowie

$T_{\text{Bubblesort}}(n) \approx 1{,}3 \cdot n^2$.

Wie sehr können wir eigentlich den Faktoren 0,6 bzw. 1,3 vertrauen? Dazu können Sie ein weiteres Experiment durchführen: Nehmen wir an, die Sortierung würde auf einem speziellen Computer durchgeführt, der die Operation „vertauschen“ hardwareunterstützt durchführen kann. Daher koste diese nun – wie der Vergleich – nur noch eine Elementaroperation. Wiederholen Sie nun das Experiment für beide Sortierverfahren Selection-Sort und Bubblesort!

Während sich bei Selection-Sort am Ergebnis wahrscheinlich nicht viel verändert hat, ist sicherlich Bubblesort deutlich „besser“ geworden: Selection-Sort nutzt ja genau eine Vertausch-Operation pro Karte. Bubblesort nutzt hingegen, abhängig von der gegebenen Vorsortierung, deutlich häufiger die Vertauschung, weshalb sich der Faktor durch die neue Annahme deutlich verbessert. Bei mir ergaben sich:

$T_{\text{Selection-Sort}}(n) \approx 0{,}5 \cdot n^2$ sowie

$T_{\text{Bubblesort}}(n) \approx 0{,}7 \cdot n^2$.

Daraus lässt sich eine ganz entscheidende Erkenntnis ableiten: Konstante Faktoren sind im Allgemeinen sehr stark abhängig von den verwendeten Rechnersystemen und von den Annahmen, die wir für die notwendigen Rechenzeiten treffen. Letztlich nehmen wir hier auch wiederum eine Modellierung vor. Wenn wir diese Modellierung gewissenhaft durchführen, verändert sich in der Regel der höchste Exponent hinter der Problemgröße n nie.

Hinzu kommt, dass konstante Faktoren durch die Wahl eines schnelleren Computers immer ausgeglichen werden können. Wären zum Beispiel 1.000.000 Karten zu sortieren, bräuchte mit den ursprünglichen Annahmen Selection-Sort ungefähr 600 Milliarden Elementaroperationen, Bubblesort 1.300 Milliarden. Selbstverständlich würde man in der Praxis den schnelleren Algorithmus einsetzen. Andererseits kann das Manko jederzeit durch Einsatz eines etwa doppelt so schnellen Computers ausgeglichen werden. Für bestimmte Gegebenheiten mag sogar Bubblesort besser geeignet sein: Es werden immer nur direkt benachbarte Speicherstellen miteinander verglichen. Denkt man

daran, dass die Daten auf einer Festplatte liegen, geht also der Vergleich ohne große Bewegung des Schreib-/Lesekopfes vonstatten. Bei Selection-Sort hat man diesen Vorteil nicht.

Wesentlich ist allerdings, wie stark der Zeitaufwand anwächst, wenn die Problemgröße anwächst! Dieses Verhältnis spiegelt sich im höchsten Exponenten und lässt sich nicht durch Wahl eines stärkeren Rechners verändern – genauso wenig wie durch die Feinjustage des Algorithmus!

Wenn wir einen besseren Algorithmus finden möchten, ist es also wesentlich, dieses Verhältnis zwischen Problemgröße und Aufwand zu verbessern. Konstante Faktoren spielen eine untergeordnete Rolle! Um Algorithmen miteinander zu vergleichen, lässt man daher normalerweise konstante Faktoren weg. Mit dieser Betrachtungsweise sind beide Sortierverfahren gleichwertig, sie haben quadratische Laufzeit. Wer es formal nicht so genau nimmt, sagt zu ihnen „quadratische Sortierverfahren".

Nicht nur in der Informatik, auch in anderen Wissenschaften sucht man immer möglichst einfache Beschreibungsmöglichkeiten. Eine kurze, prägnante Formel ist immer vorzuziehen gegenüber einem komplizierten und langen Ausdruck. Wichtig ist, dass alle wesentlichen Aspekte berücksichtigt wurden.

Wie unterscheidet sich denn nun ein Sortierverfahren, das zu einer anderen Kategorie gehört? Nehmen wir an, es gäbe eines, das kubisch, also proportional zu Problemgröße-hoch-3 funktioniert. Das würde für die 1.000.000 Karten bereits etwa 1 Trillion Elementaroperationen benötigen, wäre also nicht nur etwas schlechter, sondern würde gleich um Größenordnungen länger benötigen, die Aufgabe zu lösen. Glücklicherweise haben wir ja bereits bessere Verfahren kennen gelernt.

Schön wäre allerdings, wenn wir ein Verfahren hätten, das noch weniger Rechenschritte benötigt. Hier kann ich vorwegnehmen: Die Elite der „normalen" Sortierverfahren läuft proportional zu $n \cdot \log_2 n$ mit n als Problemgröße. Damit lassen sich 1.000.000 Karten ungefähr mit einem Vielfachen von 10.000 Elementaroperationen sortieren, was wiederum um Größenordnungen besser ist als Bubble- oder Selection-Sort.

Die Aufwandsabschätzung

Mit der Aufwandsabschätzung wird in der Informatik oft die Qualität von Algorithmen bestimmt. Diese Abschätzung gibt an, wie stark die notwendige Rechenzeit in Bezug zur Problemgröße anwächst. Konstante Faktoren werden hier nicht berücksichtigt. Auf diese Weise kann für kleine Problemgrößen die tatsächliche Rechenzeit eines „schlechteren" Algorithmus unter der des besseren liegen.

Mit der hohen Kunst der Aufwandsabschätzung wollen wir nun einen noch besseren Sortieralgorithmus finden.

Weitere Quantensprünge?
Die hier herausgearbeiteten Prinzipien für Aufwandsabschätzung gelten für herkömmliche Computer, die immer genau eine Elementaroperation auf einmal durchführen können (oder mit mehreren Prozessorkernen eine genau definierte Zahl von Elementaroperationen auf einmal).
Manche Forscher versuchen jedoch, Computer zu bauen, die völlig anders – auf Basis von Wahrscheinlichkeiten – operieren. Für unsere konventionellen Berechnungen würde das dann so aussehen, als ob diese Computer bei höheren Problemgrößen auch eine höhere Zahl von Elementaroperationen gleichzeitig verarbeiten könnten. Nach diesem Prinzip arbeiten zum Beispiel Quantencomputer. Andere Forscher legen jedoch Ergebnisse vor, nach denen Quantencomputer in sinnvollen Größenordnungen nie funktionieren werden.

nlogn oder die Kür des Sortierens

Im folgenden Abschnitt werde ich der Kürze halber immer n als Bezeichnung der Problemgröße verwenden, ohne das jedes Mal erneut zu erwähnen. Das ist der unter Informatikern übliche Buchstabe dafür.

Im letzten Abschnitt haben wir experimentell nachvollzogen, warum die bisher durchgesprochenen Sortierverfahren eine quadratische Aufwandsabschätzung (also n^2) besitzen. Man kann das jedoch auch anders begründen: Ein Sortierschritt der Verfahren sorgt jeweils dafür, dass ein einzelnes Element korrekt positioniert wird (am Ende der Folge). Wir benötigen also n Sortierschritte, bis alle Karten korrekt liegen.

Betrachten wir nun einen einzelnen Sortierschritt: Hier wird die gesamte Liste durchgegangen – bei Selection-Sort, um das größte Element zu finden, bei Bubblesort, um jeweils Paare in die korrekte Reihenfolge zu bringen. Im ersten Schritt müssen wir also n Karten anfassen, im zweiten Schritt eine weniger und so weiter, bis im letzten Schritt nur eine Karte betrachtet wird. Im Durchschnitt sind also pro Schritt etwa $\frac{n}{2}$ Karten anzufassen (was einem Vielfachen von n Operationen entspricht).

Insgesamt gibt es also

- n Durchläufe
- mit je n Operationen.

Multipliziert ergibt das einen Aufwand von $\frac{n^2}{2}$ Operationen. Da wir konstante Faktoren nicht betrachten wollten, fassen wir das zu einer Aufwandsabschätzung von n^2 zusammen.

Widmen wir uns der nächsten großen Aufgabe: Ein besseres Sortierverfahren soll gefunden werden!

Das können wir offenbar erreichen, indem wir

- entweder die Anzahl der Durchläufe reduzieren oder
- den Aufwand pro Durchlauf reduzieren.

Wie schon im letzten Kapitel können wir hier auch wieder alltägliche Situationen als Vorbild nutzen – in diesem Fall ein im Sport typischer Wettkampf im K.-o.-Verfahren. Allerdings lassen wir in unserem Fall nicht Sportler, sondern Karten gegeneinander

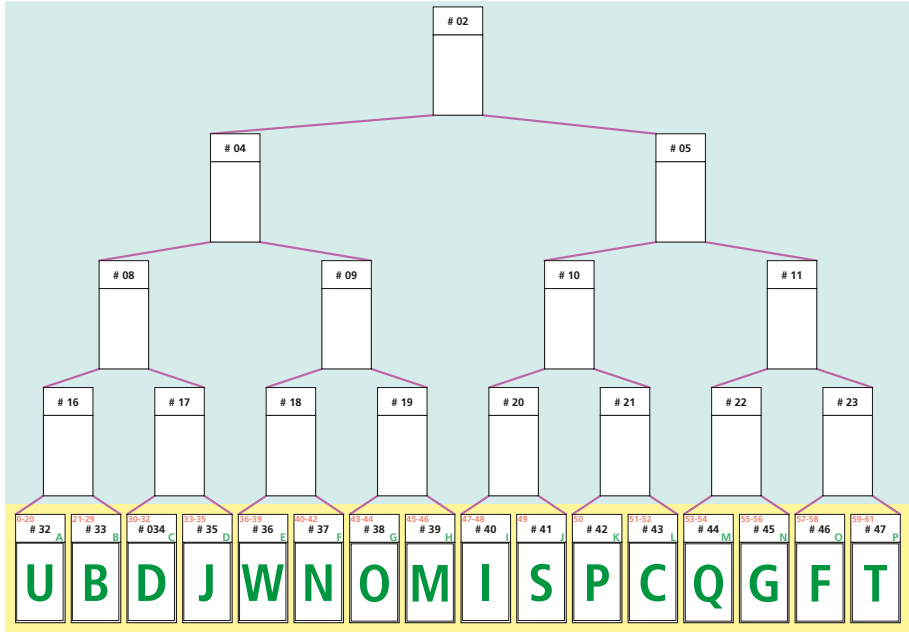


Abbildung 2.15
Die Karten treten paarweise gegeneinander an.

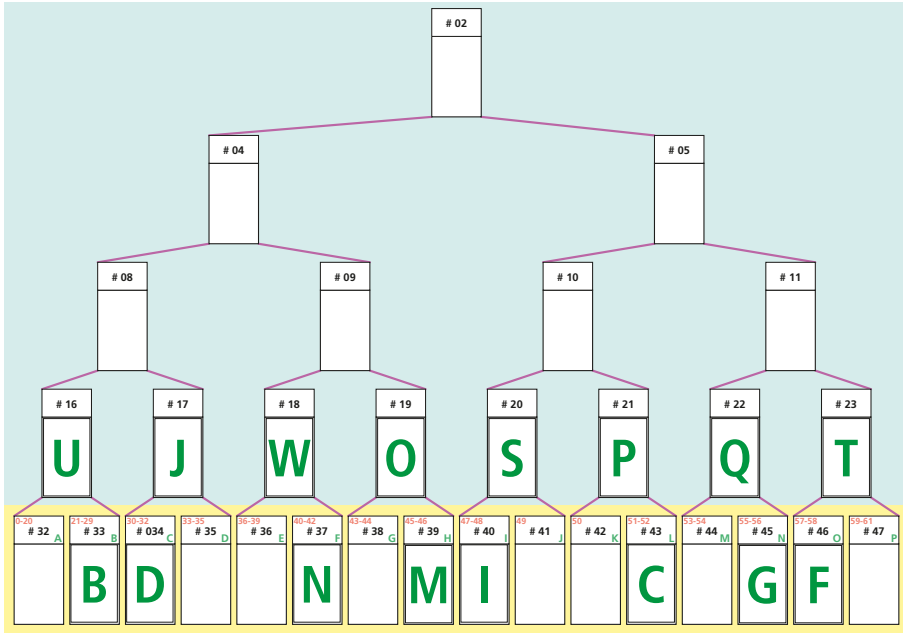


Abbildung 2.16
Das Achtelfinale wurde ausgetragen.

antreten. Verwenden Sie den Papierspeicher für den Wettkampf – diesmal einschließlich der Speicherstellen **#01** bis **#31**. Legen Sie aber zunächst die unsortierte Folge auf die bereits bekannten Speicheradressen **#32** bis **#47**, also die Hälfte des Papierspeichers mit **#02** als oberstem Element. Ein Beispiel sehen Sie in Abbildung 2.15.

Daraufhin spielen wir die erste K.-o.-Runde aus. Die jeweils größere Karte eines Paares „gewinnt“ und steigt eine Runde auf, wie in Abbildung 2.16. Im Sport entspräche das dem Achtelfinale, dessen acht Sieger um den Einzug ins Viertelfinale spielen.

Als Nächstes rücken die jeweils größeren Karten des Viertelfinales auf die vier höheren Felder vor, quasi als Halbfinalisten. Abbildung 2.17 zeigt das.

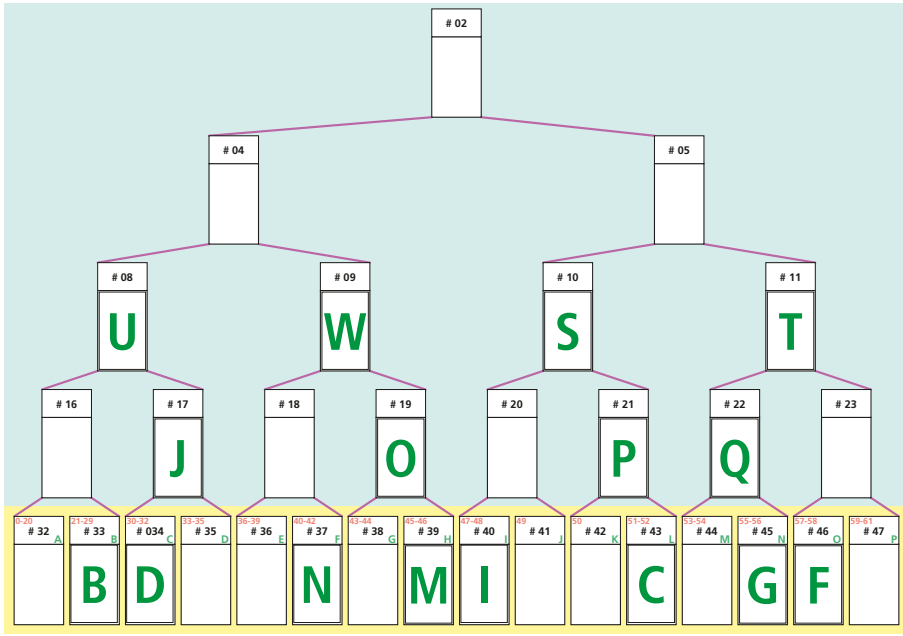
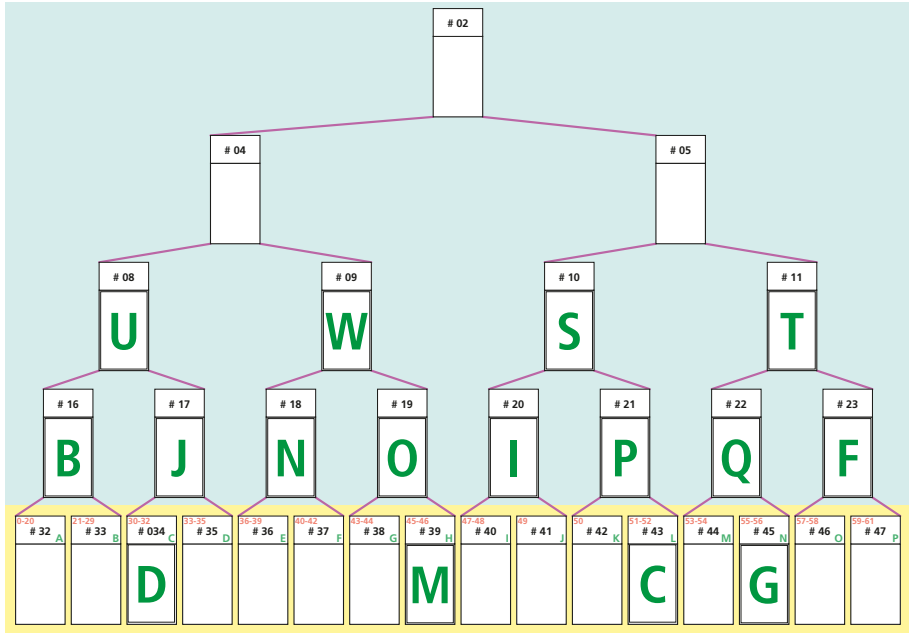


Abbildung 2.17
Die Sieger des Achtelfinales stehen im Viertelfinale

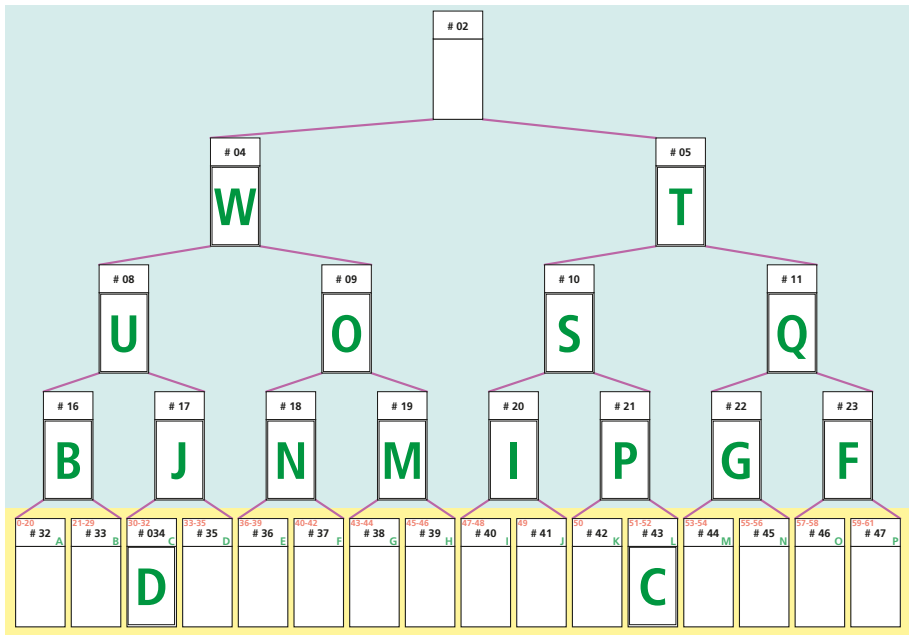
Abbildung 2.18
Belegung des Viertelfinales mit nachgerückten Kandidaten des Achtelfinales



Im Sport wären die ausgeschiedenen Mannschaften bereits nach Hause gefahren. Beim Sortieren wäre es allerdings ungünstig, die „Verlierer" gar nicht mehr weiter zu berücksichtigen. Daher werden nun die frei gewordenen Plätze des Viertelfinales aufgefüllt: Von unten rücken die entsprechenden Karten nach. In Abbildung 2.18 sehen Sie, wie zum Beispiel **B** von **#33** nach **#16** aufgestiegen ist.

Nun wird nach dem gleichen Schema das Halbfinale ausgespielt. Es gewinnen W und T, sie rücken ins Finale auf. Beachten Sie, dass wiederum die frei gewordenen Plätze des Halbfinales belegt werden. Dafür stehen nun jeweils zwei Kandidaten zur Verfügung und es wird „ausgespielt", wer aufsteigt. In diesem Fall kommt **O** auf **#09** und **Q**

Abbildung 2.19
Vor dem Finale



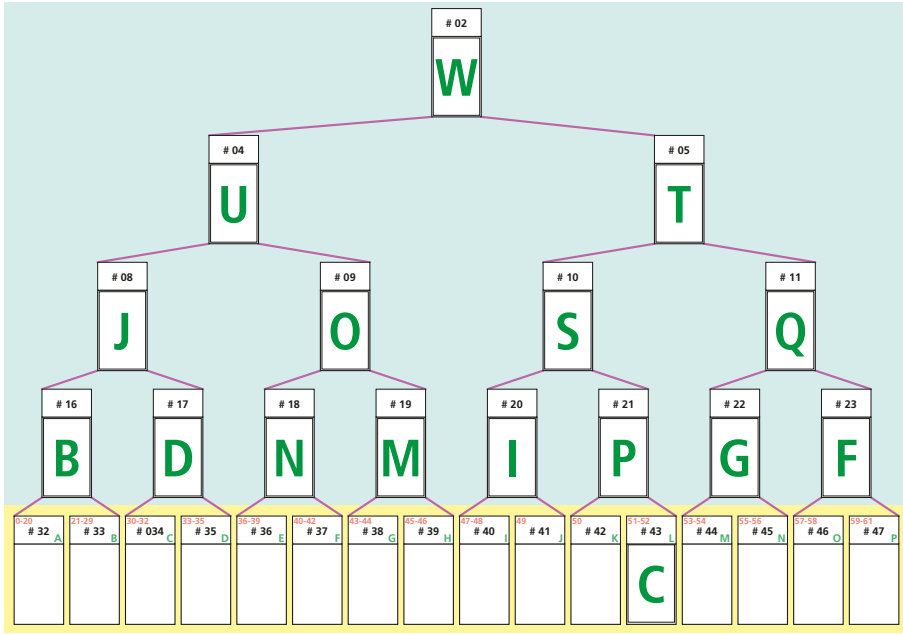


Abbildung 2.20
Fertig ausgespieltes K.-o.-System

auf **#11**. Falls möglich, werden danach noch die verwaisten Plätze des Viertelfinales von unten aufgefüllt. Abbildung 2.19 zeigt das Ergebnis.

Nun kommt nur noch das Finale – **W** gewinnt und kommt auf das Siegerfeld **#02**. Die entstehende Lücke wird selbstverständlich sofort von den jeweils nächstplatzierten Karten der darunterliegenden Ebenen ausgefüllt, die dadurch entstandene Lücke auch usw. In diesem Fall rückt die Karte **U** nach, dann **J** und **D**. Abbildung 2.20 zeigt das fertige K.-o.-System. Spielt man mit dem kompletten Papierspeicher, dauert es bis zum Finale eine Runde länger und das Siegerfeld ist die rot markierte Position **#01**.

Was bringt uns das Ganze jetzt? Durch das K.-o.-System haben wir garantiert die größte Karte gefunden, denn die liegt auf dem Siegerfeld. Sie gehört an das Ende der sortierten Liste. Dort schieben wir sie daher auch hin. In diesem Fall ist das **W**.

Der Siegerplatz ist nun vakant und selbstverständlich rücken die nächsten Karten nach, indem nacheinander das Finale sowie je eine Partie des Halbfinales und Viertelfinales ausgetragen werden, bis kein Kandidat zum Nachrücken von unten mehr da ist.

Versuchen Sie selbst, die Situation zu ermitteln, nachdem W an die Endposition der sortierten Liste verschoben wurde.

Ihr Resultat können Sie mit Abbildung 2.21 vergleichen. Genau das gleiche Prinzip verwenden Sie nun, um immer wieder den nächsten Sieger auf die letzte freie Position der fertigen Sortierung zu verschieben und das Siegerfeld neu zu füllen, indem Sie die Karten darunter nachrücken lassen. Probieren Sie es aus! Zur Kontrolle habe ich Ihnen in Abbildung 2.22 noch ein letztes Zwischenergebnis dargestellt.

Sie werden jetzt vielleicht überlegen, wozu der ganze Aufwand eigentlich gut ist, denn wir haben ja bereits zwei gut funktionierende Sortierverfahren kennen gelernt. Was ist anders am K.-o.-Verfahren, das übrigens nach dem englischen Wort für Wettbewerb

Abbildung 2.21
Situation nach dem „Ausscheiden" des Siegers und Nachrücken der nächsten Kandidaten

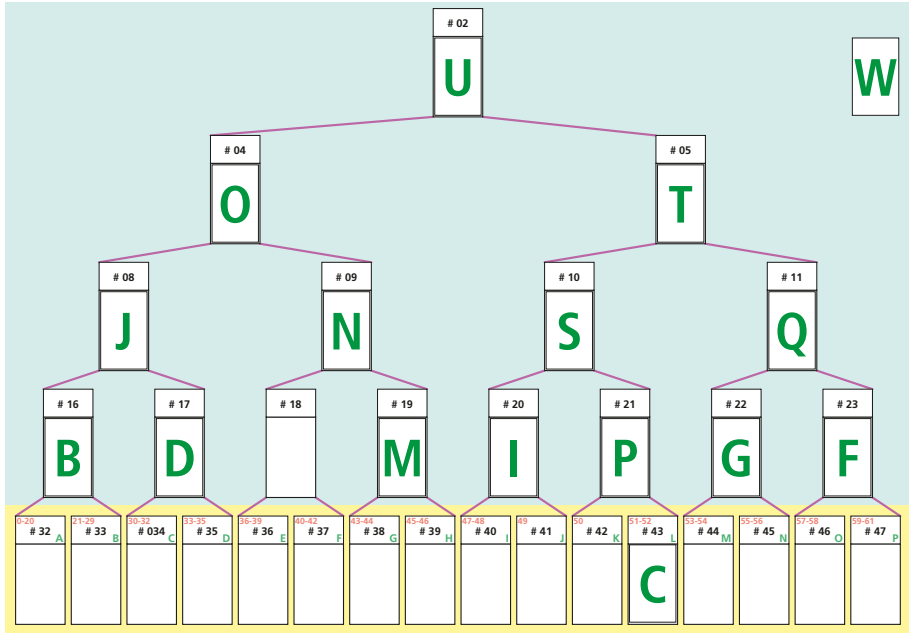
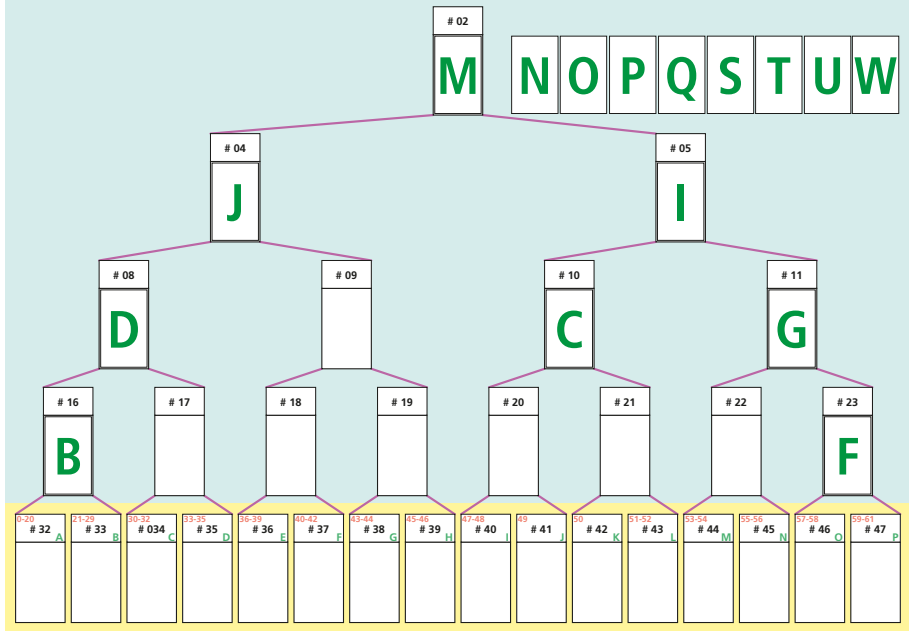


Abbildung 2.22
Die Hälfte der Karten ist nun sortiert.



„Tournament-Sort" genannt wird? Für diese Analyse benötigen wir erst einmal die genaue formale Fassung des Algorithmus.

Im Aufstellen formaler Algorithmen sind Sie ja inzwischen bereits Meister. Wollen Sie das ein weiteres Mal unter Beweis stellen, indem Sie für Tournament-Sort eine Ablaufvorschrift erstellen?

Meinen Vorschlag finden Sie in Abbildung 2.23.

0. Gehe die leeren Ebenen von unten nach oben durch und fülle jedes Feld mit der größeren Karte der beiden zugehörigen Wettbewerber

1. Fülle dabei weiter unten entstehende Lücken sofort wieder mit einer der Karten darunter auf:

2. bei zwei Wettbewerbern mit der größeren Karte | bei nur einem Wettbewerber mit diesem | bei keinem Wettbewerber bleibt das Feld leer

3. Mache Folgendes so lange, bis keine Karte mehr im Siegerfeld liegt

4. Verschiebe die Siegerkarte auf das erste freie Feld am Ende der sortierten Liste

5. Fülle dabei weiter unten entstehende Lücken sofort wieder auf wie oben beschrieben

Abbildung 2.23
Fertig ausgespieltes K.-o.-System

Versuchen Sie nun, diesen Algorithmus genau so zu analysieren, wie wir dies bereits oben getan haben. Wie viele Durchläufe benötigt er für die Sortierung? Wie hoch ist der Aufwand pro Durchlauf?

Kleiner Hinweis: Im Prinzip besteht der Algorithmus aus einer Art Vorbereitungsschritt und der eigentlichen Sortierung. Beides können wir auch getrennt voneinander betrachten.

Im ersten Teil wird die K.-o.-Tabelle vorbereitet, die erste Karte auf dem Siegerfeld ermittelt. Hierfür muss jedes (anfangs leere) Feld im oberen Bereich der K.-o.-Tabelle mit einer Karte belegt werden. Wie viele leere Felder gibt es übrigens für eine K.-o.-Tabelle mit n Sortierkarten?

Auf der untersten Ebene im gelben Bereich befinden sich n Felder mit den unsortierten Karten. Auf der untersten K.-o.-Ebene sind es halb so viele, da je eine von zwei Karten gewinnt. Es folgen wiederum halb so viele, also ein Viertel usw., bis nur noch ein Feld, das Siegerfeld, in der obersten Ebene steht.

Insgesamt gibt es also $\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\ldots+1$ Felder auf den Siegerebenen.
Einer mathematischen Rechenregel nach ergibt diese sogenannte geometrische Reihe ungefähr wieder n, also $\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\ldots+1\approx n$.
Sie können sich das einfach auch durch Ausprobieren klarmachen: Legen Sie eine Anzahl Karten in einer Reihe aus und dann für jedes Feld der K.-o.-Tabelle eine Münze darüber – also in der Reihe oberhalb eine Münze für je zwei Karten, in der Reihe darüber wieder eine Münze für je zwei Münzen darunter usw.

Zählen Sie die Anzahl der Münzen. Je nachdem, ob die Anzahl der Karten gerade ist bzw. wie oft sie durch zwei teilbar ist, ist sie etwas kleiner oder etwas größer als die Zahl der Münzen, aber immer ungefähr gleich.

Die gesamte Anzahl der Siegerplätze eines K.-o.-Systems ist immer etwa gleich groß wie die Anzahl der Wettbewerber.

Wir haben nun also herausgefunden, dass jedes Feld der leeren K.-o.-Tabelle gefüllt werden muss und dass es insgesamt ungefähr n Felder gibt! Nennen wir das einmal n Durchläufe.

Beim Füllen eines Feldes müssen gegebenenfalls weiter unten frei werdende Felder mit Nachrückern bestückt werden. Maximal passiert das so oft, wie es Ebenen in der Tabelle gibt. Man nennt die Anzahl der Ebenen auch die Höhe der Tabelle oder einfach h.

Der Aufwand unseres Tournament-Sorts wird also bestimmt von

- n Durchläufen mit jeweils
- h Verschiebeoperationen

oder Aufwand = $n \cdot h$.

Es wäre nun schön, wenn wir die Höhe irgendwie ausrechnen könnten. Ermitteln Sie zunächst durch Probieren, welche Höhe eine Tabelle mit 8, 9, 10, 15, 16, 32 und 64 Karten besitzt.

8 Karten ergeben 4 Ebenen. Bei 9, 10, 15 oder 16 Karten sind es immer 5 Ebenen. Die K.-o.-Tabelle mit 32 Karten hat 6 Ebenen, wie auch einfach am Bastelbogen ablesbar ist. 64 Karten können auf 7 Ebenen sortiert werden.

Klar wird dies, wenn man bedenkt, dass jede Ebene immer halb so viele Karten enthält (ggf. muss aufgerundet werden) wie die darunterliegende Ebene. Die oberste Ebene enthält eine Karte. Umgekehrt kann also jede untere Ebene (maximal) doppelt so viele Karten enthalten wie die darüber liegende Ebene.

Wenn wir die oberste Ebene mit der Nummer 1 versehen, ergibt sich für das Fassungsvermögen also die Tabelle am Rand.

Ebene	Anzahl Karten
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
...	...
h	2^{h-1}
$h+1$	2^h

Eine Tabelle mit 2^h Karten besitzt nach dieser Tabelle also $h+1$ Ebenen!

Da wir die Anzahl der Karten mit n bezeichnet haben, gilt also nun der Zusammenhang $n \approx 2^h$.

Das Ungefähr-Zeichen verwende ich hier, weil Gleichheit nur gelten würde, wenn die Anzahl der Karten einer „glatten“ Zweierpotenz, also 2, 4, 8, 16, ... entspräche.

Mathematiker sehen nun sofort, dass man die Formel mit dem Logarithmus auflösen kann:

$\log_2 n \approx h$

Für alle anderen: Sie konnten sich experimentell eine Vorstellung davon machen, wie die Anzahl der Ebenen mit der Anzahl der Karten wächst: Je mehr Ebenen bereits vorhanden sind, desto mehr Karten kann man hinzufügen, ohne die Anzahl der Ebenen zu erhöhen. Mit anderen Worten: Je mehr Karten vorhanden sind, desto langsamer wächst die Anzahl der Ebenen. Dieses Verhältnis wird mathematisch vom Logarithmus ausgedrückt!

Kommen wir jetzt zu unserer Betrachtung zurück, wie viel Rechenzeit der Algorithmus in Bezug zur Anzahl der Karten benötigt.

Oben hatten wir bereits zum Aufbau der K.-o.-Tabelle einen Aufwand von $n \cdot h$ bestimmt. Die Höhe h der Tabelle können wir nun mit $\log_2 n$ angeben. Daher haben wir für diesen ersten Teil des Algorithmus einen Aufwand von $n \cdot \log_2 n$ oder einfach $n \log n$.

Im zweiten Teil des Algorithmus wird jeweils die Siegerkarte in die sortierte Liste verschoben und dann das leere Feld wieder gefüllt. Das passiert so oft, wie Karten vorhanden sind, also n-mal. Jedes Mal muss das leere Feld gefüllt werden und die Anzahl der Verschiebungen entspricht wiederum maximal der Höhe, also $\log_2 n$. Wieder ergibt sich daher insgesamt ein Aufwand von $n \cdot \log_2 n$. Man könnte jetzt sagen, der Aufwand beider Teile zusammen beträgt $2 \cdot n \cdot \log_2 n$, aber konstante Faktoren lassen wir ja einfach weg, daher kommen wir auf den gewünschten Algorithmus mit $n \log n$ Aufwand. Sie können diese viel bessere Laufzeit auch experimentell nachvollziehen:

Sortieren Sie unterschiedliche Kartenzahlen mit Tournament-Sort. Wenn der vorhandene Papierspeicher für Ihren Eifer nicht ausreicht, können Sie leicht mit Haftnotizen auf einer Tischplatte Ihre eigene Version mit mehr Platz bauen. Zählen Sie anhand einer Strichliste jeweils, wie viele Verschiebungen und wie viele Vergleiche Sie durchführen mussten.

In der Tabelle auf der nächsten Seite sind zum Vergleich nochmals die Werte von Selection-Sort aufgeführt, also des besseren der beiden zuerst betrachteten Sortieralgorithmen. Außerdem sind ein paar Werte für recht hohe Kartenanzahlen eingetragen, die ich allerdings nicht per Hand, sondern per Computer ermittelt habe. Diese zeigen deutlich, dass sich ein Verfahren mit einem geringeren Aufwand besonders bei hohen Problemgrößen auszahlt – bei sehr kleinen Kartenzahlen ist Tournament-Sort sogar schlechter!

Bei Selection-Sort steigt der Aufwand pro Karte linear mit der Gesamtanzahl Karten, bei Tournament-Sort lediglich logarithmisch. Sie sehen, dass daher auch große Mengen mit Tournament-Sort noch verarbeitet werden können. Bei bereits 500 Karten benötigt dieser Algorithmus nur noch etwa 7 % der Elementaroperationen von Selection-Sort für die gleiche Aufgabe.

Was steckt dahinter?

Dieses Kapitel stand voll im Zeichen dessen, was in der Informatik Aufwandsabschätzung oder auch Komplexität genannt wird. Bereits im letzten Kapitel war es notwendig, die Qualität zweier Algorithmen gegeneinander abzuwiegen. Dort ging es jedoch hauptsächlich darum, die gegebene Aufgabe des Routenplaners überhaupt zu bewältigen. Einer der Algorithmen – Brute-Force – versagte dabei bereits für kleine Beispiele.

In diesem Kapitel haben Sie nun mehrere Algorithmen kennen gelernt, die auf den ersten Blick gleichwertig sind, da sie für überschaubare Beispiele alle recht gut funktionieren.

Computer werden jedoch oft gerade in Bereichen eingesetzt, die für uns Menschen nicht mehr überschaubar sind. Der Zentralrechner einer Bank muss nicht 30, 500 oder 10.000 Datensätze sortieren, sondern teilweise Milliarden von Transaktionen in die korrekte Reihenfolge bringen. Hierbei ist zwingend erforderlich, die dafür verwendeten Algorithmen nicht intuitiv zu beurteilen, sondern anhand genauerer Maßstäbe. Im Allgemeinen wird die Abschätzung der Komplexität als Maßstab verwendet!

Tabelle
Aufwand für das Sortieren verschiedener Kartenzahlen

n	Selection	Tournament	Selection pro Karte	Tournament pro Karte
2	4	5	2,0	2,5
3	9	12	3,0	4,0
4	15	16	3,8	4,0
5	22	28	4,4	5,6
6	30	33	5,0	5,5
7	39	41	5,6	5,9
8	49	48	6,1	6,0
9	60	68	6,7	7,6
10	72	73	7,2	7,3
11	85	80	7,7	7,3
12	99	90	8,3	7,5
13	114	97	8,8	7,5
14	130	110	9,3	7,9
15	147	115	9,8	7,7
16	165	126	10,3	7,9
17	184	161	10,8	9,5
18	204	168	11,3	9,3
19	225	179	11,8	9,4
20	247	190	12,4	9,5
25	372	244	14,9	9,8
30	522	289	17,4	9,8
50	1.372	577	27,4	11,5
100	5.247	1.368	52,5	13,7
150	11.622	2.300	77,5	15,3
200	20.497	3.096	102,5	15,5
250	31.872	3.931	127,5	15,7
300	45.747	5.197	152,5	17,3
350	62.122	6.091	177,5	17,4
400	80.997	7.073	202,5	17,7
450	102.372	7.943	227,5	17,7
500	126.247	8.869	252,5	17,7

Wie bereits oben dargestellt, ist dabei oft gar nicht so wichtig, ob ein Algorithmus doppelt so lange benötigt wie ein anderer. Wesentlich ist, wie viel mehr Rechenzeit ein Algorithmus benötigt, wenn die Problemgröße anwächst. Genau dieses Verhalten macht sich nämlich besonders stark bemerkbar, wenn wir mit sehr großen Datenmengen arbeiten.

Wenn wir die Verfahren auf diese Weise analysieren, bekommen wir heraus, welches für unsere Problemstellung am besten geeignet ist. Die Sache hat aber auch noch einen weiteren Vorteil: Wenn wir die Analyse durchführen, machen wir uns besonders deutlich, an welchen Stellen des jeweiligen Verfahrens Rechenzeit verbraucht wird. Das liefert gute Hinweise darauf, wo wir noch etwas verbessern können, um zu einem günstigeren Ergebnis zu gelangen.

Im Beispiel oben hatten wir festgestellt, dass sowohl Selection-Sort als auch Bubblesort grob gesagt n Durchläufe mit jeweils n Operationen benötigen, also insgesamt eine Komplexität von n^2 besitzen. Die Folge war, dass wir auf ein Verfahren hinarbeiten konnten, in dem die Anzahl der Durchläufe gleich blieb, aber die Anzahl der Operationen pro Durchlauf deutlich optimiert wurde: Tournament-Sort.

Einen anderen Ansatz zur Optimierung verfolgt übrigens der von Tony Hoare 1960 erfundene Quicksort-Algorithmus, der auch heute noch in vielen Formen eingesetzt wird: Hier benötigt jeder Durchlauf prinzipiell noch n Operationen, aber die Anzahl der Durchläufe wird reduziert. Auch Quicksort hat daher im Durchschnitt die Komplexität von $n \log n$.

Großes Oh

Im bisherigen Verlauf des Kapitels habe ich die Freiheit der populärwissenschaftlichen Darstellung genutzt und den Algorithmen einfach einem bestimmten Aufwand zugeordnet, also etwa n^2 oder $n \log n$.

Der wirkliche Aufwand eines Algorithmus hängt aber von vielen Faktoren ab: wie die Daten am Anfang sortiert sind, wie das Verfahren genau auf einem Computer umgesetzt wurde usw. Da dieser tatsächliche Aufwand daher gar nicht oder nur sehr schwer zu bestimmen ist, spricht man gewöhnlich von der Ordnung eines Aufwands, wenn man ihn so abschätzt, wie wir das bisher getan haben.

Bubblesort und Selection-Sort sind von der Ordnung n^2, während Tournament-Sort und Quicksort von der Ordnung $n \log n$ sind.

Informatiker bedienen sich hierfür des Buchstabens O, abgeleitet vom griechischen Omikron, das der Mathematiker Paul Bachmann bereits 1892 verwendete, um die Ordnung von Funktionen zu definieren.

Man sagt daher genauer, Bubblesort und Selection-Sort haben eine Komplexität von $O(n^2)$, gesprochen „groß-Oh von n Quadrat“. Quicksort und Tournament-Sort haben eine Komplexität von $O(n \log n)$. Diese Notation wird Ihnen im Großteil der Informatik-Fachliteratur begegnen.

Tony Hoare
Er wurde am 11. Januar 1932 in Sri Lanka geboren. Er studierte Statistik und beschäftigte sich mit der automatisierten Übersetzung von Fremdsprachen. Berühmt ist er durch die Entwicklung des Quicksort-Algorithmus, der bis heute als das effizienteste vergleichsbasierte Sortierverfahren für die meisten realen Datenmengen gilt, sowie für das „Hoare-Kalkül“, das verwendet wird, wenn man die Korrektheit eines Algorithmus beweisen möchte.

O(n) oder: Streben nach dem perfekten Algorithmus

Lange Zeit galten Tournament-Sort & Co. als die besten möglichen Sortierverfahren. Man kann sogar mathematisch beweisen, dass es für das Sortieren, so wie wir es machen, gar keine besseren Verfahren als solche von der Komplexität $O(n \log n)$ geben kann. Informatiker sind da jedoch recht flexibel und oft lassen sich auch die strengsten mathematischen Beweise dadurch umgehen, dass man die Voraussetzungen dafür abändert. Das haben wir im ersten Kapitel nachvollzogen, als plötzlich die geographischen Positionen der Orte (und damit ihre Entfernungen) in die Betrachtung einbezogen wurden. Bei Proxmap-Sort berücksichtigt man, dass in realen Anwendungen keine Schlüssel vorkommen, die nur verglichen werden können. Was ich damit meine, möchte ich Ihnen erläutern:

Proxmap-Sort ist eigentlich bei genauerer Betrachtung gar nichts Revolutionäres, sondern etwas, das Generationen von Menschen bereits beim Sortieren von Karten oder ähnlichen Objekten intuitiv durchführen.

Nehmen Sie ein Päckchen gewöhnlicher Spielkarten und einen großen Tisch. Ihre Aufgabe ist, die Karten nach ihren Werten zu sortieren. Die Herangehensweise nach den bisherigen Algorithmen wäre, die Karten erst einmal genauso, wie sie sind, auf dem Tisch auszulegen und dann einen der besprochenen Algorithmen durchzuführen.

Effektiv gehen aber in der Realität die wenigsten Menschen auf diese Weise vor. Vielmehr nutzen wir bereits beim Auslegen der Karten unser Wissen über deren Verteilung. Die Verteilung besagt, wie viele Karten welches Typs im Stapel sind. Normalerweise wissen wir also, dass jede Karte von As bis König in einer Farbe genau einmal vorkommt. Nehmen wir zu Beginn eine 7 auf, dann legen wir sie nicht auf die linke Seite, sondern bereits ungefähr an die Position, an die sie gehört, also etwa in die Mitte des Tisches. Ein As kommt an den Anfang, ein König ans Ende. Auf diese Weise brauchen wir im Idealfall jede Karte nur ein einziges Mal anzufassen und legen sie direkt an die korrekte Position.

Genau das Gleiche macht Proxmap: Für jede Karte wird „erraten“, wo diese in einer Liste hinkommt, und dort wird sie dann direkt abgelegt. Ist die Stelle ausnahmsweise

bereits besetzt, muss man eventuell eine oder zwei Karten verschieben, um Platz zu schaffen. Im Wesentlichen kommt der Algorithmus jedoch damit aus, jede Karte nur ein einziges Mal zu betrachten.

Probieren Sie das gleich einmal mit den blauen Zahlen auf Ihren Sortierkarten aus: Mischen Sie 10 der Karten und versuchen Sie, diese direkt auf den ersten 10 Feldern Ihres Papierspeichers zu verteilen.

Das Vorgehen ist klar: Die Karten zeigen Werte zwischen 0 und 999.999. Daher liegt die Vermutung nahe, dass die erste Stelle bereits ein guter Anhaltspunkt für die Position der Karte ist.

log*n* = *c* ?

Den interessierten Lesern möchte ich nicht verschweigen, dass einige Informatiker mit einem Augenzwinkern behaupten, es gäbe keinen Unterschied zwischen O(*n*) und O(*n*log*n*). Die Argumentation ist recht einfach: Unsere Erde hat geschätzt 6 · 10^{49} Atome. Mehr Objekte werden ganz sicher nie zu sortieren sein. Der Zweierlogarithmus dieser wahrlich astronomischen Zahl beträgt trotzdem nur knapp schnöde 166. Das ließe sich doch quasi als Konstante ansehen. Sie können sich ja schon einmal eine Argumentation zurechtlegen – nur für den Fall …

Sie werden bemerken, dass hier trotzdem oft noch sogenannte Kollisionen auftreten: Mehrere Karten kommen eigentlich auf den gleichen Platz. Diese müssen dann untereinander mit einem der konventionellen Verfahren sortiert werden.

Mit der geringen Anzahl von 10 Karten ist die Chance sehr hoch, dass wir zufällig recht ähnliche Karten ziehen. Ein ähnliches Problem haben Meinungsforscher, wenn sie zu wenige Testpersonen haben. Ein repräsentatives Ergebnis bekommt man nur mit einer entsprechend hohen Stichprobe.

Die Statistik können wir aber zu unseren Gunsten nutzen, wenn wir ein paar mehr Speicherplätze zur Verfügung stellen, als Karten einzusortieren sind.

Verteilen Sie daher in einem weiteren Experiment 15 gemischte Karten auf den 20 vorderen Speicherplätzen #32 bis #51 des gesamten Papierspeichers und sortieren Sie dabei wieder nach der blauen Zahl.

Hinweis: Wenn Sie Mußе haben, können Sie später auch den Papierspeicher noch auf beliebig viele Plätze erweitern – mit einem weiteren (z. B. kopierten) Bastelbogen oder einfach mit aufgeklebten Haftnotizen. Beschriften Sie die Speicherstellen und füllen Sie diese zu ca. 75 % mit Ihrer initial unsortierten Zahlenfolge.

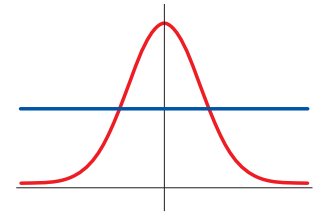
Nun ist auch noch die zweite Stelle relevant: Zahlen zwischen 0 und 49.999 kommen auf **#32**, solche zwischen 50.000 und 99.999 auf **#33** usw.

Kollisionen sind jetzt sehr selten, obwohl wir immerhin 75 % des Speichers füllen! Sie müssen am Ende die Lücken nur noch schließen, indem Sie die Karten zusammenschieben. Das kostet ebenfalls maximal eine Verschiebeoperation pro Karte.

Macht man den Speicherplatz noch größer bzw. die Anzahl zu sortierender Karten kleiner, werden Kollisionen äußerst unwahrscheinlich und es gilt tatsächlich, dass jede Karte im Durchschnitt nur zwei Mal angefasst werden muss – einmal für das Einsortieren und nochmals für das Zusammenschieben.

Damit ist der Aufwand „2 · Anzahl der Karten“ oder $2n$. Da wir konstante Faktoren prinzipiell weglassen, kommen wir für Proxmap-Sort auf $O(n)$.

Alles ganz normal?



Die Gauß-Verteilung bzw. Normalverteilung (rot) gegenüber der Gleichverteilung (blau). Die x-Achse bildet dabei den Wertebereich ab, die y-Achse die Wahrscheinlichkeit. Während die blaue Linie die gleiche Wahrscheinlichkeit aller Werte anzeigt, hat die rote Linie ein deutliches Maximum und fällt dann in beide Richtungen stark ab. Wegen ihrer Form wird die Gauß-Verteilung auch Glockenkurve genannt.

Versuchen Sie nun, genau die gleichen 15 Karten nochmals auf die Speicherplätze zu verteilen, jetzt aber aufgrund der roten Zahlen. Plötzlich funktioniert unser Verfahren nicht mehr so gut. In den allermeisten Fällen gibt es jetzt deutlich mehr Kollisionen. Woran liegt das?

Auf den ersten Blick sind die roten Zahlen genauso zufällig verteilt wie die blauen. Hier trügt jedoch der Augenschein: Die blauen Zahlen sind gleich verteilt. Das bedeutet, jede Zahl zwischen 0 und 999.999 kommt mit derselben Wahrscheinlichkeit vor. Auf diese Weise ergibt sich auch eine gleichmäßige Verteilung der Karten auf die Speicherplätze – ein Garant für wenige Kollisionen.

Die roten Zahlen sind jedoch normal verteilt (auch Gauß-verteilt genannt). Hier kommen die Zahlen im mittleren Bereich um 500.000 deutlich häufiger vor als solche am Rand des Zahlenbereichs, wie 100.000 oder 900.000. Daher kommt es um die entsprechenden Speicherstellen auch vermehrt zu Kollisionen – Proxmap-Sort kann nicht mehr optimal funktionieren.

Ein Beispiel, das dies noch klarer macht, ist die alphabetische Sortierung tatsächlicher Wörter. Die grünen Wörter auf den Sortierkarten sind repräsentativ für die deutsche Sprache. Sie sind in regelmäßigen Abständen aus dem Duden entnommen.

Die grünen Buchstaben auf den Speicherstellen 0 bis 25 können zur Proxmap-Sortierung der Wörter verwendet werden. Testen Sie aus, wie viele Kollisionen es mit 20 Karten gibt ...

Offenbar ist unsere Sprache ebenfalls nicht gleich verteilt: Selbstverständlich kommen Wörter mit den Anfangsbuchstaben A oder S viel häufiger vor als solche mit X oder Y.

Egal ob normal verteilte Zahlen oder Wörter: Proxmap-Sort scheint damit nicht recht zu funktionieren ... oder können wir den Algorithmus so modifizieren, dass er besser läuft?

Den Teufel mit dem Belzebub austreiben? In der Informatik funktioniert es zumindest, unregelmäßig verteilte Karten mit einer unregelmäßigen Verteilung der Felder des Sortierverfahrens auszugleichen.

Sortieren Sie nochmals 15 Karten nach den roten Zahlen auf den ersten 20 Speicherplätzen – diesmal aber gemäß der roten Beschriftung der Speicherstellen.

Plötzlich funktioniert die Sache wieder mit wenigen Kollisionen. Die rote Beschriftung des Papierspeichers berücksichtigt die Verteilung der Karten und reserviert für wahrscheinlichere Intervalle auch mehr Speicherplätze als für unwahrscheinliche.

Sie merken also, dass Proxmap-Sort dann arbeiten kann, wenn die Verteilung der zu sortierenden Größen bekannt ist und man dadurch für jede Größe die Position im Speicher erraten kann. In der Praxis analysiert das Verfahren erst die zu sortierenden Daten, um die Verteilung kennen zu lernen. Damit kann der Computer recht gut raten, wo ein neuer Datensatz in den Speicher gehört.

Proxmap – ein Sortierverfahren?

Weiter oben hatte ich geschildert, dass eigentlich bei $O(n \log n)$ Schluss ist mit der Optimierung von Sortierverfahren. Mit Proxmap-Sort erreicht man aber unter günstigen Bedingungen $O(n)$, nur eben – wie bereits erwähnt – unter veränderten Voraussetzungen. Nach den letzten Betrachtungen sollten Sie auch den Unterschied zwischen den klassischen Sortierverfahren und Proxmap erkennen. Welcher ist das?

Für die klassischen Sortierverfahren muss man lediglich eine sogenannte Ordnungsrelation zwischen zwei zu sortierenden Größen kennen: Für je zwei Karten muss klar sein, welche die größere ist bzw. ob beide als gleichwertig anzusehen sind. Dabei ist egal, wie viel größer oder kleiner die eine Karte gegenüber der anderen ist. Auch die Verteilung der Werte spielt keine Rolle – die Sortierverfahren laufen ohne weiteres Wissen darüber zuverlässig und mit gleicher Leistungsfähigkeit ab. Sie basieren ausschließlich darauf, immer wieder zwei Kartenwerte miteinander zu vergleichen.

Proxmap-Sort benötigt zusätzliche Informationen: Wie wir Menschen beim Sortieren unser Wissen über die Verteilung der Karten implizit nutzen, so versucht auch Proxmap die korrekte Position einer Karte anhand ihres Wertes zu erraten. Das setzt voraus, dass man überhaupt etwas über die Verteilung der Größen kennt. Falls dies nicht der Fall ist, kann Proxmap-Sort auch nicht sinnvoll verwendet werden.

Vielleicht überlegen Sie nun, was für Objekte man sortieren könnte, über die man keine Informationen außer der Vergleichbarkeit hat. Konstruieren lässt sich so etwas: Denken Sie an die Reihenfolge beim morgendlichen Aufstehen. „Kleiner als“ sei definiert als „früher“. Dann würde sich zum Beispiel ergeben

aufwachen < duschen < anziehen < frühstücken < schlüssel_nehmen

Ein Computer würde sich sehr schwertun, etwa dem Vorgang „anziehen“ den korrekten Platz zuzuweisen unter all den anderen Vorgängen. Eventuell kommen ja noch Tätigkeiten hinzu wie „zähneputzen“, und auch wenn die Reihenfolge irgendwie im Computerspeicher hinterlegt wird, kann man wohl keinem einzelnen Vorgang eine genaue Durchführungszeit zuweisen.

Letztlich muten solche Beispiele aber immer sehr künstlich an, denn sie enthalten Dinge, die man normalerweise auch nicht per Computer sortieren möchte. Für alle „echten“ Beispiele, die mir einfallen, lässt sich recht gut erraten, wohin ein Objekt in eine Sortierung gehört. Daher funktioniert Proxmap-Sort auch praktisch immer.

Sortieren nach Gewicht
Dosen gleichen Aussehens werden paarweise verglichen

Ein weiteres Szenario, für das Proxmap nicht funktioniert, können Sie übrigens auch einfach basteln: Nehmen Sie eine Menge gleich aussehender, undurchsichtiger Dosen, zum Beispiel leere Farbdosen, und füllen Sie diese mit unterschiedlichen Gewichten. Ein netter Wettbewerb ist nun, die Dosen zu vermischen und danach möglichst schnell wieder dem Gewicht entsprechend zu sortieren. Dazu bleibt einem nicht viel anderes übrig, als immer zwei Dosen miteinander zu vergleichen – in der Hand oder auf einer Waage. Versuchen Sie es!

Proxmap – der Algorithmus

Vielleicht haben Sie bemerkt, dass wir bisher bei Proxmap noch nicht ganz korrekt gearbeitet haben: Kam es zu einer Kollision, konnte plötzlich unser Papierspeicherplatz

Abbildung 2.24
Proxmap-Sort

trotzdem mehrere Werte fassen. Um das Kapitel abzuschließen, erhalten Sie als Abbildung 2.24 noch einen Vorschlag für die „saubere“ Implementierung des Algorithmus.

Versuchen Sie es – das Verfahren funktioniert. Testen Sie das auch wieder mit einer Strichliste für jede Speicher- und Verschiebeoperation und ermitteln Sie dadurch die Qualität experimentell. Achten Sie darauf, dass Sie den Speicher immer nur zu etwa 75 % füllen, weil sonst Proxmap-Sort nicht mehr günstig laufen kann: Es gibt zu viele Kollisionen.

Resümee

Einerseits wissen Sie nun, wie ein Computer auf verschiedene Arten große Datenmengen sortieren kann, um dann später wieder schnell darauf zuzugreifen. Fast noch

wesentlicher ist jedoch die Erkenntnis, dass man das gleiche Ziel fast immer auf sehr unterschiedliche Arten und Weisen erreichen kann.

Um zu beurteilen, welche dieser Möglichkeiten nun die beste ist, benötigt man einen Qualitätsmaßstab. Im Fall von Algorithmen ist die Ordnung der Komplexität ein solcher Maßstab. Sie gibt an, wie stark die Rechenzeit in Abhängigkeit der Problemgröße anwächst.

Oft kann man zusätzliche Informationen über die gegebenen Daten dazu verwenden, günstigere Algorithmen zu verwenden, wie im Fall von Proxmap-Sort, wo das Wissen um die Verteilung der Werte zu einem Verfahren mit der Komplexität $O(n)$ führt, während bei den klassischen, nur auf Vergleichen beruhenden Verfahren der günstigste Fall bei $O(n \log n)$ liegt.

Eventuell können Sie an dem einfachen Beispiel dieses Kapitels am besten ablesen, was den ingenieurwissenschaftlichen Teil der Informatik ausmacht: Das Sortieren, basierend nur auf Vergleichen, strahlt eine gewisse mathematischen „Schönheit“ aus, weil es universell ist: Alles, was sortierbar ist, kann sortiert werden. Trotzdem sind Informatiker damit in der Regel nicht zufrieden, sondern suchen ein Verfahren, das zwar an bestimmten, konstruierten Daten scheitert, aber in allen realen Fällen effizienter funktioniert.

Abbildung 2.K1
Kopiervorlage für Karten

187650	302782	503710	861867	212088	522107	38897	955099	96588	424261	747690	44948	581176	133986	467199	145423
aber	Ablage	absurd	Acker	Anbau	Anlage	Anteil	Arbeit	Aufruf	auftun	Ausland	Auslauf	Auster	Ausweg	Bank	Beere
276637	338210	425565	471452	544520	582278	260249	562184	444082	409121	559505	495493	163105	124890	448463	677562
488893	968277	154571	165413	9859	112092	553475	869328	885439	475340	598622	391932	539477	54218	369430	570546
Bein	bereit	Besitz	Beule	Birne	Boden	Brot	Busch	Dank	denn	Docht	Duft	Durst	ein	einmal	einzeln
551904	578884	708129	343721	310313	792094	497819	762015	328439	596164	578754	551017	303040	825829	431497	686943
495973	501884	737014	92280	251907	386655	903955	272409	902289	761556	245698	79474	666701	190206	525643	555035
Ente	Erde	erneut	Erz	falsch	Fell	Filter	Flug	frei	Fuhre	Garn	gefeit	Geist	genehm	Gesetz	gewagt
313203	545038	511851	416671	196973	348971	541849	456940	206299	511274	472166	541510	527355	501389	454812	533659
417230	770853	122091	220073	799190	316190	323682	791873	801774	645869	727378	242709	285853	468831	433707	822890
gleich	Graf	Gruppe	halb	Happen	Hefe	herb	Heu	hinter	hold	Idee	intern	jung	Karte	Kind	Klette
503130	493004	469414	502542	373529	286658	567975	411534	769872	587632	365762	500126	454879	504286	981432	656819
47703	689770	494105	338328	10320	230696	853090	999775	42536	601352	840530	344842	80693	173243	938664	442995
Komma	Korb	Kreuz	Kur	lang	Leber	Leute	Los	Mais	Maus	mild	Mohn	nach	Narr	nie	Oase
387491	485489	563432	716338	628007	657758	456570	261525	655509	557321	484208	408488	649042	593174	443699	955147
737455	825946	407769	194723	101144	64452	889619	179025	298597	614559	693337	901096	26685	569868	663440	544775
Ozon	Perle	Platte	Presse	Qualm	Raub	Regel	Report	Rost	Saal	schade	Schein	Schlag	Schliff	Schrank	Schutz
635066	616474	356123	394035	627000	33834	299566	429514	434058	871139	536598	746586	532031	180319	356039	285673
621512	512644	265569	1546	633803	875185	341378	922111	658614	379917	716582	208537	704238	919143	832882	406012
schwer	selbst	Sinn	sozial	Spitze	Stadt	Stein	Stock	streng	Suppe	Tausch	toll	treu	Uhr	Umkehr	Umzug
570621	304128	534281	685547	511749	437755	819903	622771	791314	345638	608609	293607	415283	739765	534179	525784
783299	813083	715258	229762	532702	450663	759881	626457	67142	585876	890751	971956	987582	371795	941393	352600
Unheil	untreu	Vehikel	Verein	Verleih	Vers	Verweis	voll	Vorteil	Wand	Weide	Wert	Winter	Wurm	Zeit	Zirkus
460043	520134	473739	491187	308833	473574	482518	468563	409135	430459	66580	715557	642427	356724	381937	491846
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Q	R	S	T	U	V	W	X	Y	Z	0	1	3	7	12	13
18	24	31	37	40	45	51	65	66	72	73	75	78	80	84	87
92	96	98	99												

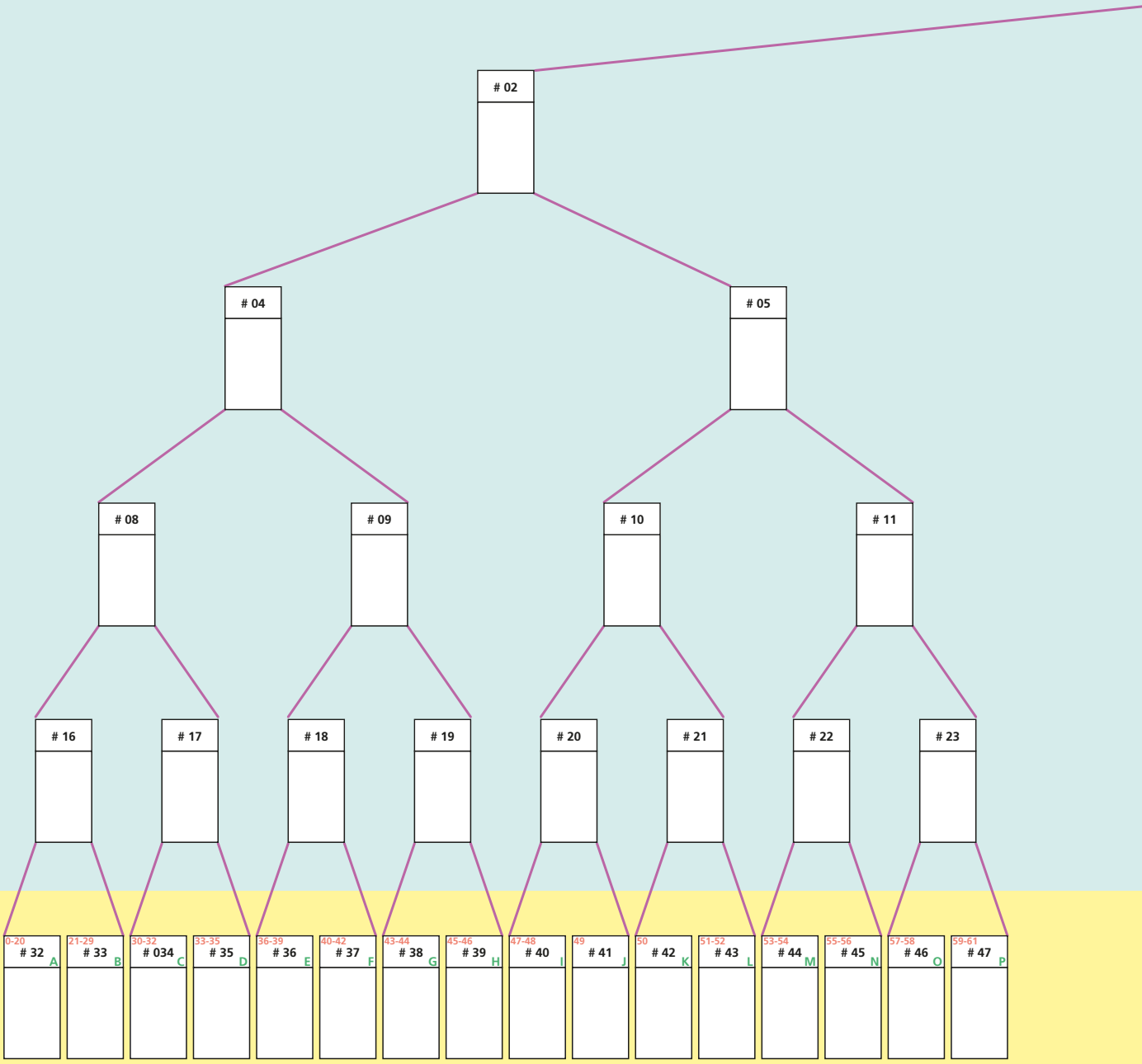
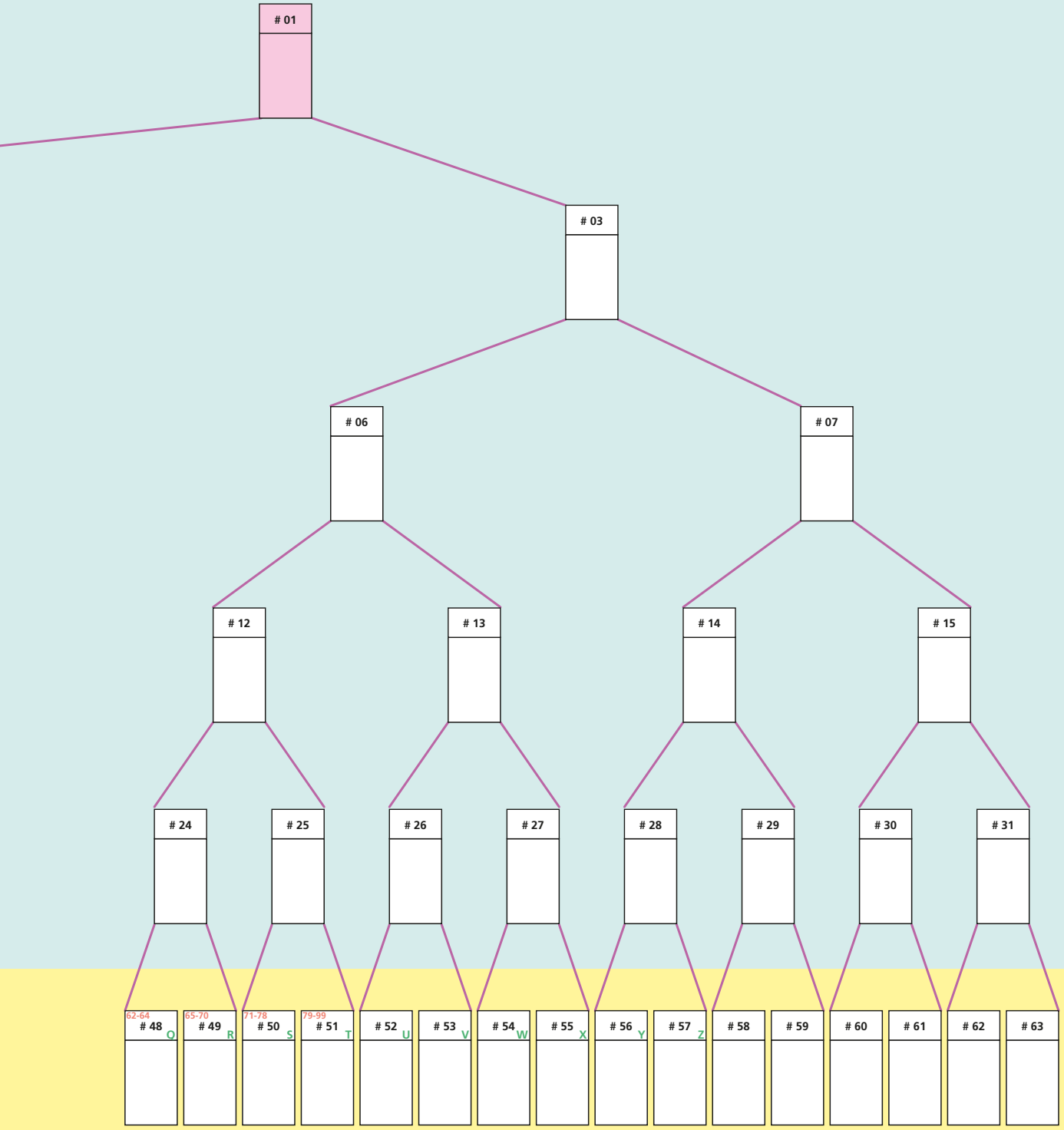
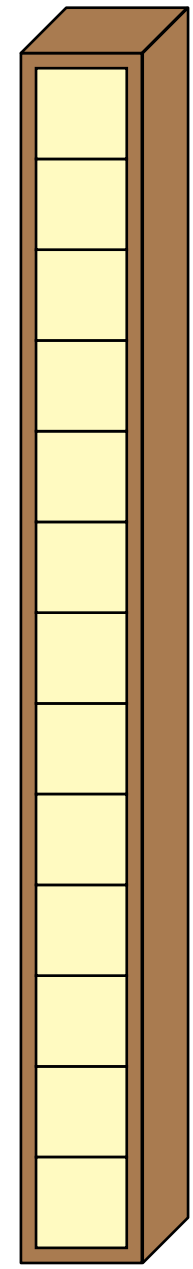

02
04
05
08
09
10
11
16
17
18
19
20
21
22
23
0-20 # 32 A
21-29 # 33 B
30-32 # 034 C
33-35 # 35 D
36-39 # 36 E
40-42 # 37 F
43-44 # 38 G
45-46 # 39 H
47-48 # 40 I
49 # 41 J
50 # 42 K
51-52 # 43 L
53-54 # 44 M
55-56 # 45 N
57-58 # 46 O
59-61 # 47 P

Abbildung 2.K2
Der Papierspeicher



3. Ich packe meinen Koffer und ...



Es war einmal ein tapferer Mann aus dem Volk, der rettete die Prinzessin Tausendschön vor dem Räuber Abi Lala und seinen zweiundvierzig Schergen. Leider musste sich der König nun etwas als Belohnung ausdenken: Wäre der Mann ein Prinz gewesen, hätte er ihm einfach die Hand der Prinzessin anbieten können. Für einen gewöhnlichen Bürger würde der König aber wohl oder übel seine Schatzkammer öffnen müssen.

Um den Schaden jedoch zu begrenzen, gab der König dem Mann eine Holzkiste und sprach: „Du darfst dir alles aus der Schatzkammer nehmen, was in die Kiste passt, so dass sich der Deckel noch schließen lässt." Er dachte, dass jener sicher so von den Schätzen geblendet wäre, dass er eher billigen Tand als wirklich wertvolle Stücke auswählen würde.

Hier hat er jedoch nicht mit unserer Beteiligung gerechnet! Die Informatik hat dem Mann aus dem Volk sogar eine eigene Kategorie von Aufgaben gewidmet, genannt „Rucksackprobleme".

Das Rucksackproblem

Bereits in den vorherigen Kapiteln haben wir sehr viel mit Abstraktion gearbeitet und das Problem zunächst vereinfacht, um darüber besser nachdenken zu können. Hier nehme ich diesen Schritt schon jetzt vorweg und biete Ihnen die gesammelten „abstrakten" Schätze als Bastelbogen oder Kopiervorlage 3.K1 – rechteckig und aus einer glatten Zahl von Quadraten zusammengesetzt.

Schneiden Sie sie aus und versuchen damit, die Kiste am Buchrand möglichst optimal zu füllen. Wir gehen hier einfach davon aus, dass jedem Schatz ein Zertifikat mit einem genauen Wert (selbstverständlich einheitlich in Golddublonen bemessen) beiliegt – dieser steht als ganze Zahl neben der Abbildung.

Die sogenannten „Trivialfälle" sind hier übrigens bereits eliminiert: Nehmen wir an, es existierten zwei Gegenstände mit exakt der gleichen Größe, aber unterschiedlichem Wert. Dann legt man selbstverständlich den billigeren Gegenstand im Vorfeld bereits als „uninteressant" ab. Daher steht hier nur ein Objekt pro Größe zur Wahl (eben das jeweils teuerste).

Sie sehen bereits an dem einfachen Beispiel, dass es gar nicht so einfach ist, das Optimum aus allen Möglichkeiten herauszufinden: Es passen sechs Edelsteine in die Kiste, aber auch eine Vase und zwei Ringe. Vielleicht sind aber drei Münzen wertvoller? Oder gar die goldene Waage – dann passt aber sonst nicht viel mehr in die Kiste ...

Wie auch bereits zuvor, können wir hier die Lösung durch stures Ausprobieren finden: Alle Möglichkeiten werden systematisch durchgegangen, die wertvollste Kombination wird beibehalten. Das ist für das vorliegende Beispiel machbar, aber bereits sehr zeit-

aufwendig. Überlegen Sie, wie viele Möglichkeiten Sie bereits hätten, wenn die Kiste auch nur doppelt so groß wäre ...

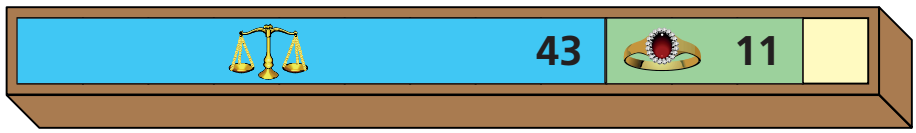


Abbildung 3.1
Gefüllte Schatzkiste mit einem Wert von 54 Dublonen. Geht das noch besser?

Über die Problematik des sturen Probierens hatten wir bereits im Kapitel über Routenplanung philosophiert, das möchte ich an dieser Stelle nicht wiederholen. Allerdings versichere ich Ihnen, dass der Aufwand zum Probieren hier mit der Größe der Kiste ebenfalls sehr stark anwächst. Die Lösungsansätze werden damit schnell unübersichtlich und sind zumindest in sinnvoller Zeit nicht zielführend.

Mehr ist weniger

Divide et impera
„Teile und herrsche" – erinnern Sie sich noch daran? Wenn nicht: In Kapitel 2 wird dieses Prinzip ausführlich erörtert.

Statt herumzuprobieren wollen wir daher das Problem konsequent angehen. Eine entscheidende Strategie der Informatik haben Sie ja bereits mit „divide et impera" kennengelernt: Wenn das Gesamtproblem zu groß zum Lösen ist, versuchen wir es in kleinere Pakete zu zerteilen. Das machen wir so lange, bis die Teilprobleme handhabbar klein sind.

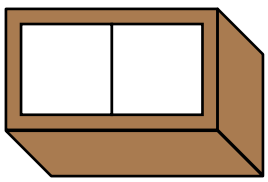
Um das Problem in kleinere Happen zu unterteilen, müssen wir allerdings zunächst feststellen, wodurch überhaupt ein Problem dieser Art als „groß" oder „klein" gilt – wir bestimmen die Problemgröße. Beim Routenplaner ist das zum Beispiel die Anzahl der Orte auf einer Landkarte, beim Sortieren die Anzahl der zu sortierenden Objekte.

Ihre Aufgabe: Überlegen Sie, was beim Packen des Rucksacks die Problemgröße ist.

Zugegeben: Die Frage war vielleicht ein wenig unfair formuliert, aber ansonsten hätte ich schon zu viel verraten. Sie sind bestimmt trotzdem auf die korrekte Antwort gekommen.

- Ist „Größe der Kiste" vielleicht eine Problemgröße?
- Ist „Anzahl unterschiedlicher Gegenstände" vielleicht eine Problemgröße?

Beides ist richtig: Wie bereits oben ausgeführt, wird eine Lösung immer schwieriger, je größer die Kiste ist und je mehr Gegenstände man daher gleichzeitig einpacken muss (das gilt übrigens nur, wenn man wirklich an einer exakten Lösung interessiert ist, aber dazu mehr weiter unten). Zusätzlich steigt die Problemgröße selbstverständlich auch an, wenn man mehr unterschiedliche Gegenstände zur Wahl hat und ausprobieren muss.



Sie können Ihre Vermutung über eine Problemgröße normalerweise leicht untermauern: Überlegen Sie, welches das „kleinste Problem" ist, wenn Sie Ihre Überlegungen zugrunde legen.

„Größe der Kiste": Nehmen Sie eine sehr kleine Kiste, zum Beispiel die nebenstehende. Ist hier die Lösung des Problems besonders schwierig? Nein, denn es gibt überhaupt nur zwei Möglichkeiten:

- die Kiste leer lassen (und das wäre wirklich dumm) oder
- einen Edelstein hineinladen, was immerhin sechs Dublonen bringt.

„Anzahl unterschiedlicher Gegenstände“: Benutzen Sie immer die große Kiste, aber gehen Sie davon aus, in der Schatzkammer befände sich nur eine einzige Sorte Schatz, zum Beispiel Edelsteine. Auch in diesem Fall gibt es nicht viel zu entscheiden: Man lädt so viele Edelsteine in die Kiste, wie hineinpassen. Abbildung 3.2 zeigt eine solche Kiste.

Abbildung 3.2
Schatzkiste nur mit Edelsteinen, immerhin 42 Dublonen wert

Beim Sortieren von Karten war eine mögliche Vorgehensweise, mit einer einzelnen Karte anzufangen (die per Definition sortiert ist). Dann wurde die Problemgröße jeweils um eine Karte erweitert, die in die bereits sortierte Folge leicht eingefügt werden konnte.

Vielleicht finden Sie ja auf dieser Basis auch eine Lösung für das Rucksackproblem? Versuchen Sie es: Fangen Sie mit kleinen Kisten oder wenigen unterschiedlichen Objekten an und erhöhen Sie dann die Problemgröße langsam.

Die Lösung gestaltet sich schwieriger als erwartet:

Fängt man mit einer kleinen Kiste an und steigert deren Größe, kommt man recht schnell auf optimale Ergebnisse für Kisten mit 2, 3, 4 und 5 Quadraten (wir nennen diese fortan 2er-Kiste, 3er-Kiste usw. oder Kisten der Größen 2, 3, 4 usw.). Danach fängt das Rätseln an: Welche der Teillösungen müssen kombiniert werden, um eine noch größere Kiste zu packen? Auch hier können wir nur alle Möglichkeiten ausprobieren, was uns zum Ausgangspunkt unserer Betrachtungen zurückwirft.

Versuchen wir also, gleich die große 13er-Kiste zu nehmen, fangen aber zunächst an, sie ausschließlich mit Edelsteinen zu füllen, wie schon in Abbildung 3.2 gezeigt. Danach erweitern wir die Problemgröße: Auch Ringe sind nun erlaubt. Recht schnell kann man feststellen, dass vier Ringe (mit 44 Dublonen Wert) besser sind als die sechs Goldbarren (die nur 42 Dublonen wert waren).

Offenbar sind also Edelsteine nicht so günstig wie Ringe, wir können sie also für die folgenden Betrachtungen außer Acht lassen. Weiter geht es und auch die Goldtaler mit 16 Dublonen Wert sollen nun im Angebot sein. Wiederum können wir den Wert unserer Kiste steigern, indem wir die Ringe entsorgen und stattdessen drei Goldtaler einladen. 48 Dublonen sind nun das Ergebnis.

Als Nächstes nehmen wir den wertvollen Kompass hinzu, der so ausladend ist, dass er fünf Plätze ausfüllt. In unsere Kiste passen daher nur zwei Kompasse mit einem Gesamtwert von 48 Dublonen. Das ist genauso viel, wie wir bereits haben. Daher bleiben wir bei den Goldtalern, oder?

Ganz so einfach ist es nicht, denn legen wir zu den zwei Kompassen noch einen Ring (was gut passt), steigt der Wert unserer Kiste sogar auf 59. Unsere Entscheidung, die Ringe als „zu billig“ außer Acht zu lassen, war also ziemlich voreilig! Auch mit diesem Lösungsansatz müssen wir im späteren Verlauf der Problemlösung immer wieder

ganz an den Anfang zurückspringen, um das Optimum herauszuholen, was effektiv auch einfach das Durchspielen aller Möglichkeiten bedeutet.

Ist hier also „divide et impera“ gar nicht zu gebrauchen? Müssen wir uns bei diesem Problem geschlagen geben?

Selbstverständlich nicht: Die Informatik hat noch ein paar Tricks auf Lager. In diesem Fall heißt der Trick: „Mehr ist weniger“, oder anders gesagt „Wenn man mehr Probleme löst, hat man letztlich weniger Arbeit.“

Wie ist das genau zu verstehen? Das Teilen von Problemen ist manchmal schwierig und man weiß nicht genau, welche Teilprobleme man genau lösen muss, um sie dann zu einer Gesamtlösung zu kombinieren. In solchen Fällen ist es manchmal hilfreich, einfach alle möglichen Teilprobleme zu lösen und danach zu entscheiden, welche man kombiniert. So etwas nennt man dynamische Programmierung.

Richard Ernest Bellman (1920 – 1984) studierte Mathematik und beschäftigte sich danach mit theoretischer Physik. Ab 1952 befasste er sich dann bei der Rand-Corporation mit Optimierungsproblemen und adaptierte ein vorher nur von Physikern verwendetes Prinzip für die Informatik, das er „dynamische Programmierung“ nannte. Heute ist ein Algorithmus zur Erstellung optimaler Binärbäume nach ihm benannt.

Dynamische Programmierung

Strategie zur Lösung komplexer Probleme, besonders zur Lösung von Optimierungsaufgaben. Man löst zunächst viele Teilprobleme und verwendet diese Bausteine – optimale Zwischenergebnisse –, um die nächstgrößeren Probleme zu lösen usw., bis das Gesamtproblem gelöst ist.

Um dieses Prinzip auf unser Rucksackproblem anzuwenden, benötigen wir viele Kisten: von einer ganz kleinen Kiste (mit überhaupt keinem Platz darin) bis zur größten Kiste mit 13 Quadraten Platz. Sie finden dies im Bastelbogen und Abbildung 3.K2 – sogar mit zwei weiteren Feldern für Ihre eigenen Experimente. Ich nenne das Gebilde im Folgenden „Maxikiste“, da die Kisten aller ganzer Größen enthalten sind.

Jetzt fangen wir ganz klein an: Gehen Sie davon aus, es gäbe nur die kleinste Sorte Schatz zur Auswahl – Edelsteine. Gemäß dem Prinzip der dynamischen Programmierung ermitteln wir nun für den anderen Typ Problemgröße – die Kistengröße – nicht nur eine Lösung, sondern gleich alle bis zur Größe 13. Die entsprechenden Kisten finden wir in der Maxikiste.

Mit nur einer Sorte Schatz ist es leicht, die Kisten entsprechend optimal zu füllen. Machen Sie das! Zur Verdeutlichung können Sie links auf Klebezetteln daneben den Wert jeder Kiste vermerken.

Abbildung 3.3 zeigt die Maxikiste. Selbstverständlich kann man in die kleinste Kiste mit der Größe 0 nichts hineingeben. Auch in die Kiste mit nur einem Quadrat Platz passt kein Edelstein – sie bleibt daher frei. Die restlichen Kisten können mit ein bis sechs Edelsteinen gefüllt werden, man stopft einfach so viele hinein, wie passen. Die größte Kiste enthält sechs Barren zu je sieben Dublonen, hat also einen Wert von 42 Dublonen.

Um es noch einmal festzuhalten: Die Maxikiste zeigt nun die optimale Lösung mit der kleinsten Problemgröße – nur die kleinsten Schätze dürfen ausgewählt werden.

Wie können wir diese Ergebnisse verwenden, um nun den nächstgrößeren Schatz einzubeziehen? Diesmal zeige ich Ihnen, was dafür zu tun ist, und wir überlegen hinterher zusammen, warum diese Strategie sinnvoll ist.

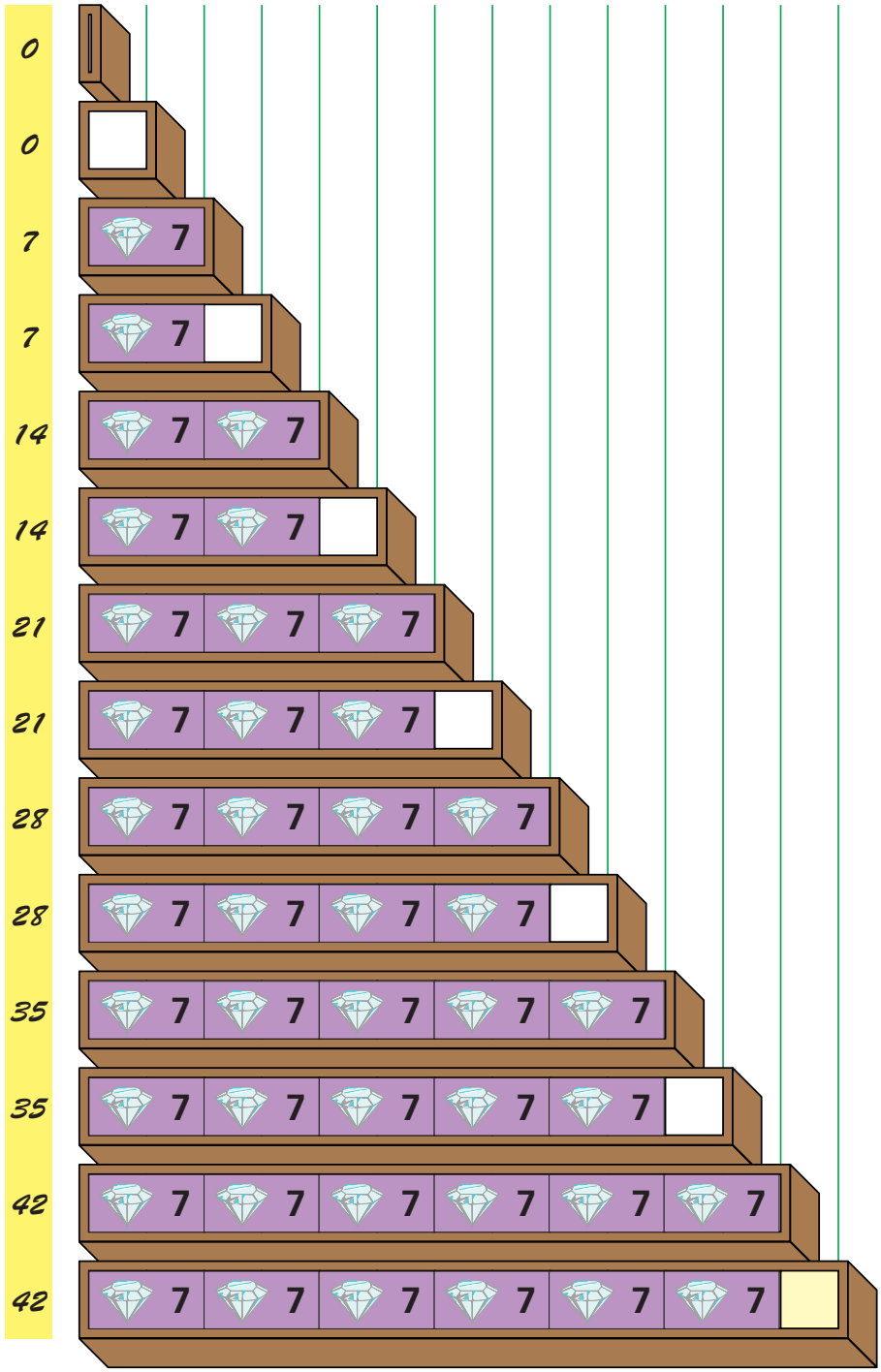
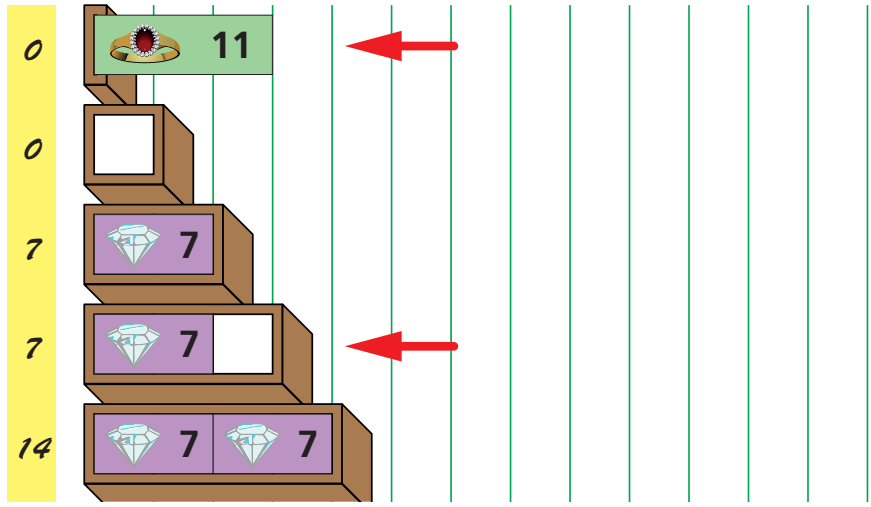


Abbildung 3.3
Lösung des Rucksackproblems bis zur Kistengröße 13, wenn nur Edelsteine zur Wahl stehen

Nehmen Sie einen Schatz „Ring“ und legen ihn rechts an die kleinste Kiste an – das ist die Kiste mit Größe 0. Sie haben jetzt sozusagen den neuen Schatz zu der Kiste hinzugelegt. In welche (größere) Kiste würden die Inhalte nun passen? Selbstverständlich in die Kiste der Größe 3, wie Sie auch ablesen können, wenn Sie die grünen Linien nach unten verfolgen. Abbildung 3.4 zeigt das.

Abbildung 3.4
Den „Inhalt" der Kiste mit Größe 0 und einen Schatz der Größe 3 könnten wir zusammen in die Kiste der Größe 3 packen.



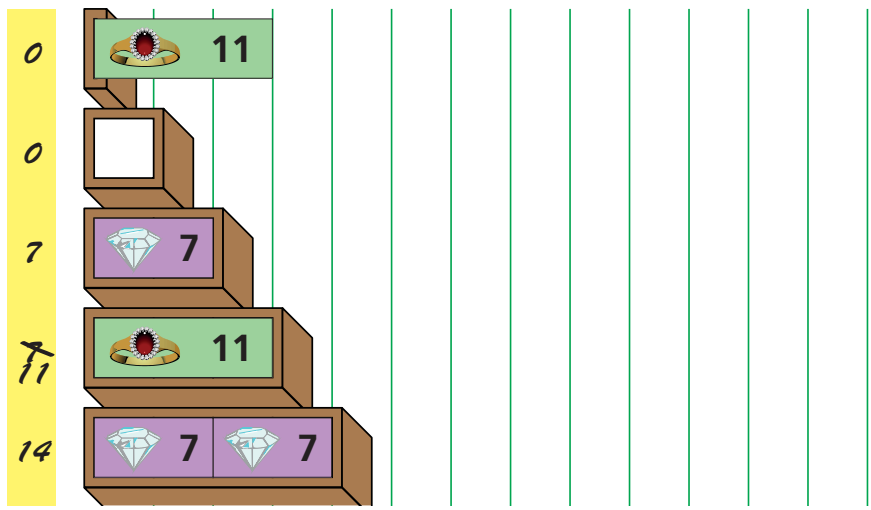
Bestimmen Sie nun den Wert beider markierter Zeilen: Der Wert der unteren Zeile steht bereits links davon, er beträgt sieben Dublonen. Der Wert der oberen Zeile ist die Summe aus dem Wert der Kiste (0) und dem Wert des neuen Schatzes – 11 Dublonen.

Es gibt nun zwei Möglichkeiten:

- Der Wert der oberen Zeile ist höher. Die neue Kombination aus dem vorigen Inhalt der kleineren Kiste plus dem neuen Schatz ist also wertvoller als der vorher bestimmte „optimale" Inhalt. Dieser ist daher unter Berücksichtigung des neuen Schatzes nicht mehr optimal, wir tauschen ihn aus gegen die obere Kombination (das heißt, wir entfernen die Schätze aus der Kiste und setzen neue vom Schatzvorrat ein, bis die untere markierte Kiste genau die gleichen Schätze enthält wie die obere plus dem neuen Schatz).
- Der Wert der oberen Zeile ist nicht höher. Offenbar gilt das alte Optimum auch noch unter Berücksichtigung des neuen Schatzes. Es wird nichts verändert.

Im Beispiel beträgt der Wert der oberen Zeile 11 Dublonen, der Wert der unteren Zeile liegt bei lediglich sieben Dublonen. Es lohnt sich also, die Kiste neu zu packen. Tun Sie das! Selbstverständlich müssen Sie auch den Wert links von der Kiste auf den neuesten Stand bringen. Das Ergebnis sehen Sie in Abbildung 3.5.

Abbildung 3.5
Der „Inhalt" der Kiste mit Größe 0 und ein Schatz der Größe 3 wurden nun in die Kiste der Größe 3 gepackt.



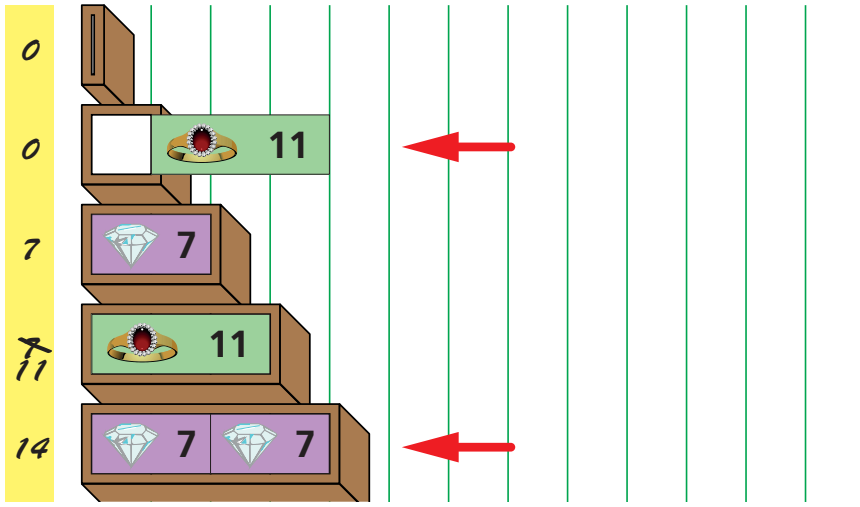


Abbildung 3.6
Ist die vorhandene 4er-Kiste mit zwei Goldbarren günstiger als die neue Kombination aus Geldbündel und dem (leeren) Inhalt der 1er-Kiste?

Nun verschieben wir den Schatz um eine Position nach unten und legen ihn rechts an die nächstgrößere Kiste an. Jetzt vergleichen wir den neuen, kombinierten Inhalt mit dem vorhandenen Inhalt der 4er-Kiste. Abbildung 3.6 zeigt das Vorgehen.

Wieder vergleichen Sie den Wert der bisher optimalen 4er-Kiste (14) und den Wert der neuen Kombination (11). Diesmal ist die alte Kombination günstiger und Sie verändern den Inhalt der Kisten nicht.

Erneut wird der neue Schatz eine Position nach unten verschoben. Versuchen Sie diesmal, erst selbst die Entscheidung zu treffen, ob Kisteninhalte ausgetauscht werden (und wenn ja, welche).

Selbstverständlich ist hier der neue Inhalt vorzuziehen, da er einen Wert von 18 gegenüber dem vorhandenen Wert von 14 hat. Wieder wird das Geldbündel eine Zeile tiefer geschoben. Nun liegt es an einer Kiste, die wir bereits umgepackt haben. Abbildung 3.7 zeigt die Situation.

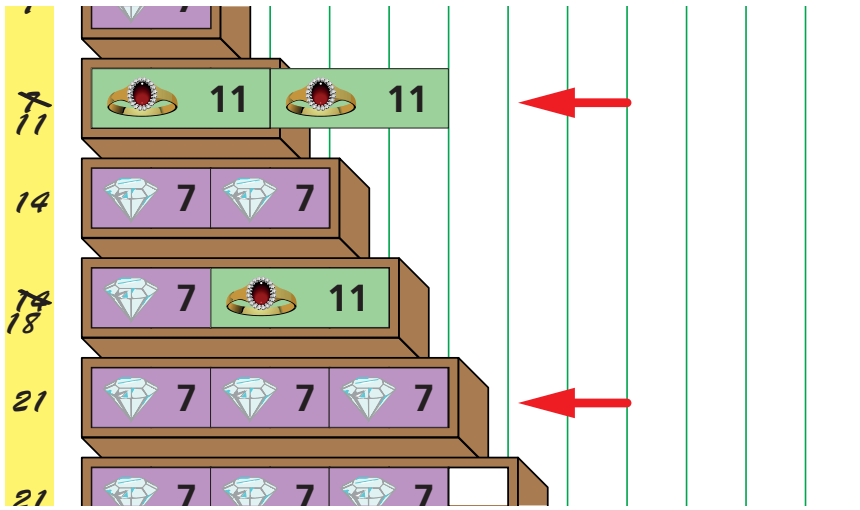


Abbildung 3.7
Kann die 6er-Kiste noch günstiger gefüllt werden, indem man den Inhalt der optimalen 3er-Kiste mit dem Schatz „Ring“ kombiniert?

Das ist aber gar kein Problem – folgen Sie einfach dem bisherigen Schema und vergleichen Sie die Kombination mit der 6er-Kiste. Auch hier tauschen wir den Inhalt aus, weil wir eine Verbesserung erzielen, wie in Abbildung 3.8 zu sehen ist.

Sehen Sie an diesem Beispiel, dass es klug ist, die Kisten von klein nach groß zu füllen? Auf diese Weise hatten wir bereits die neue, optimal gefüllte 3er-Kiste als Grundlage der Neubetrachtung für die 6er-Kiste zur Verfügung. Wäre in der 3er-Kiste noch der alte Inhalt gewesen (mit sieben Dublonen Wert), hätten wir den Tausch verworfen – eine in diesem Fall ungünstige Variante.

Führen Sie nun das Verfahren für die gesamte Maxikiste durch!

Das Ergebnis sehen Sie in Abbildung 3.9. Die meisten Kisten enthalten nun einen oder mehrere wertvolle Ringe. Der Übersichtlichkeit halber habe ich die Werte direkt neu eingetragen, ohne die alten auszustreichen.

Weiter geht es und der nächstgrößere Schatz – der Goldtaler – steht nun ebenfalls zur Auswahl. Füllen Sie gleich die gesamte Maxikiste neu.

Sie sehen das Ergebnis in Abbildung 3.10 auf der nächsten Doppelseite. Die 2er-, 3er- und 5er-Kisten bleiben bestehen, bei allen anderen lohnt es sich, den neuen Schatz einfach oder mehrfach zu verwenden.

Bitte richten Sie Ihr besonderes Augenmerk auch auf die Veränderung bei der 6er-Kiste: Dort war im zweiten Schritt der Edelstein zugunsten zweier Ringe weggefallen. Unsere Vorgehensweise macht das jedoch keinesfalls endgültig, denn im dritten Schritt wurden die Ringe wieder durch einen Edelstein und einen Goldtaler ersetzt.

Als Nächstes ist der Kompass an der Reihe. Wie sieht die optimale Maxikiste nun aus?

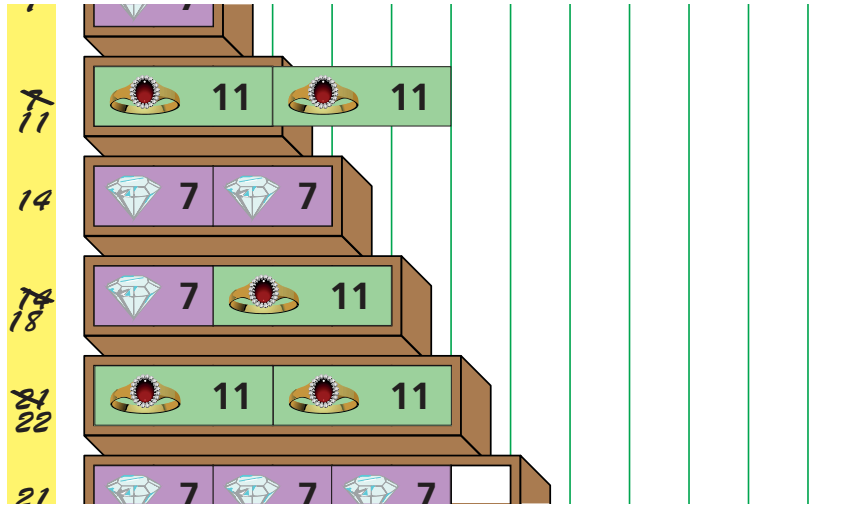


Abbildung 3.8
Die 6er-Kiste hat nach dem Tausch des Inhalts einen Wert von 22.

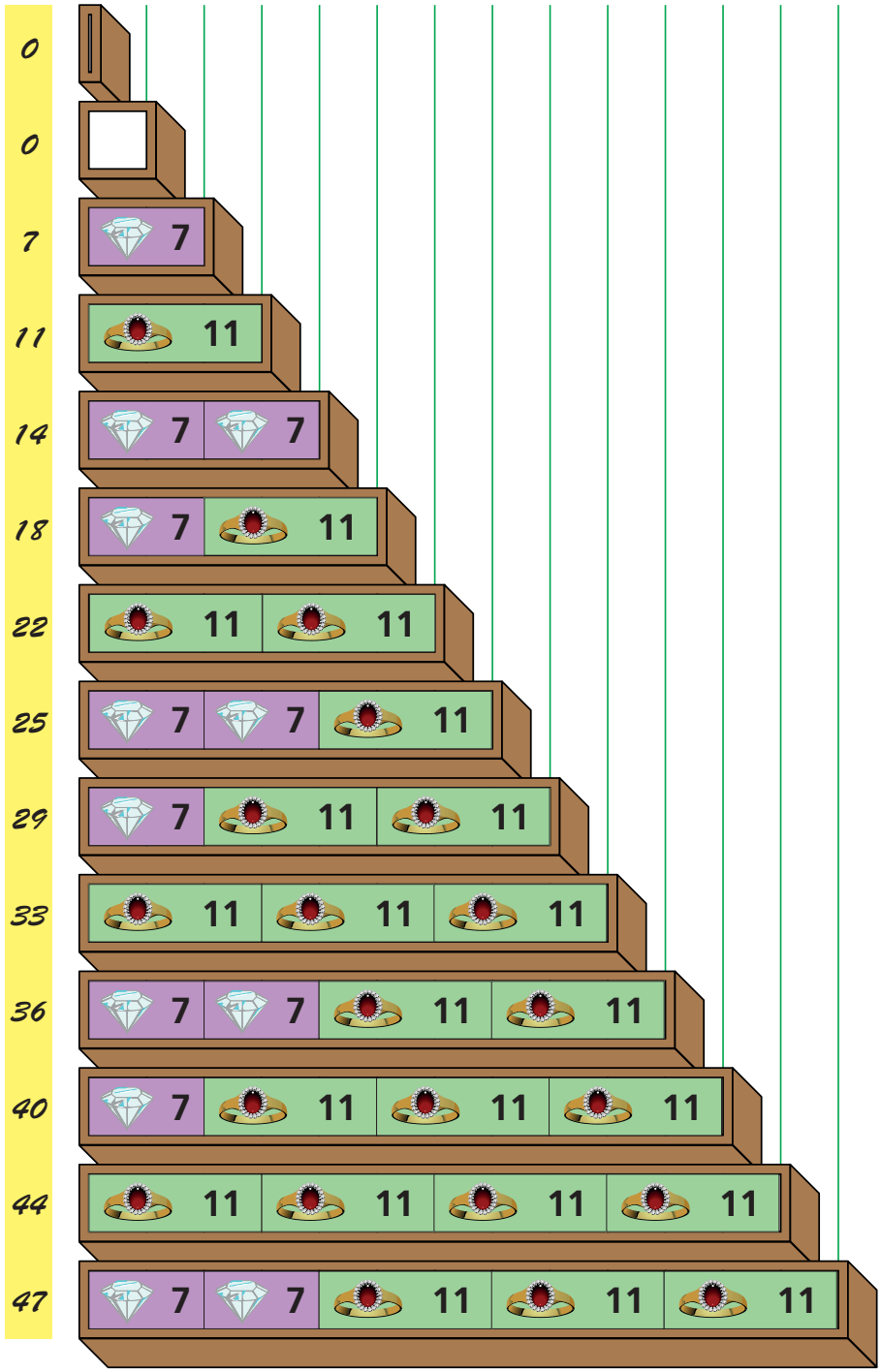
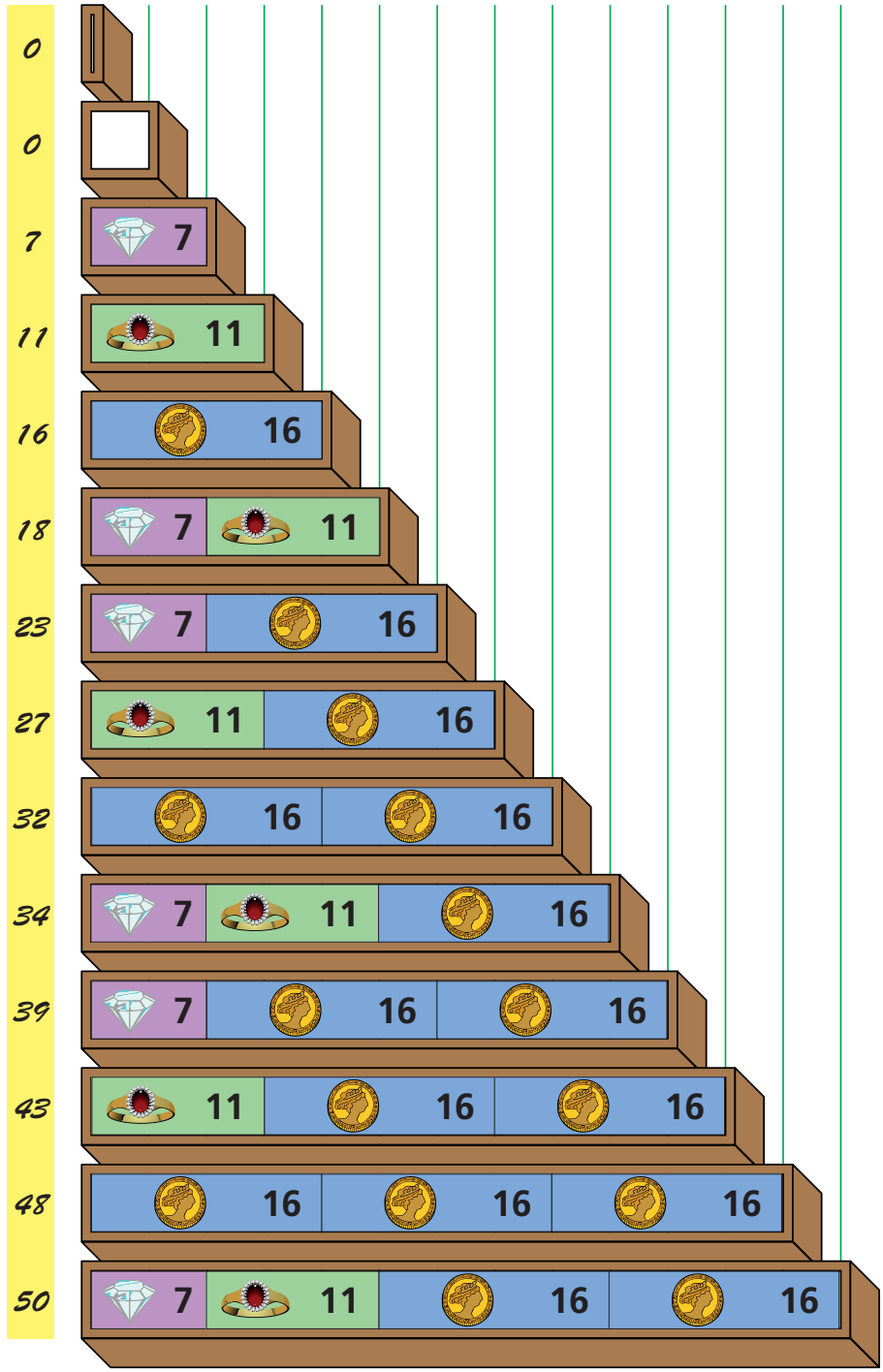


Abbildung 3.9
Nun ist auch das Geldbündel als Schatz berücksichtigt.

In Abbildung 3.11 sehen Sie, dass eine optimal gefüllte Kiste keinesfalls randvoll sein muss. Der Kompass ist offenbar so viel wert, dass es sich sogar lohnt, bei 6er- und 11er-Kisten ein Feld frei zu lassen. Das zeigt auch, warum es in unserem Verfahren sehr wichtig ist, die 1er-Kiste jedes Mal zu betrachten, auch wenn diese immer leer bleibt: Nur durch Anlegen des Kompasses an die 1er-Kiste konnte die Verbesserung des Wertes der 6er-Kiste erreicht werden.

Abbildung 3.10
Die Schmuckschatulle verändert die optimalen Ergebnisse erneut.



Es stehen aber immer noch größere Schätze zur Verfügung. Als Nächstes sind die bunten Vasen an der Reihe.

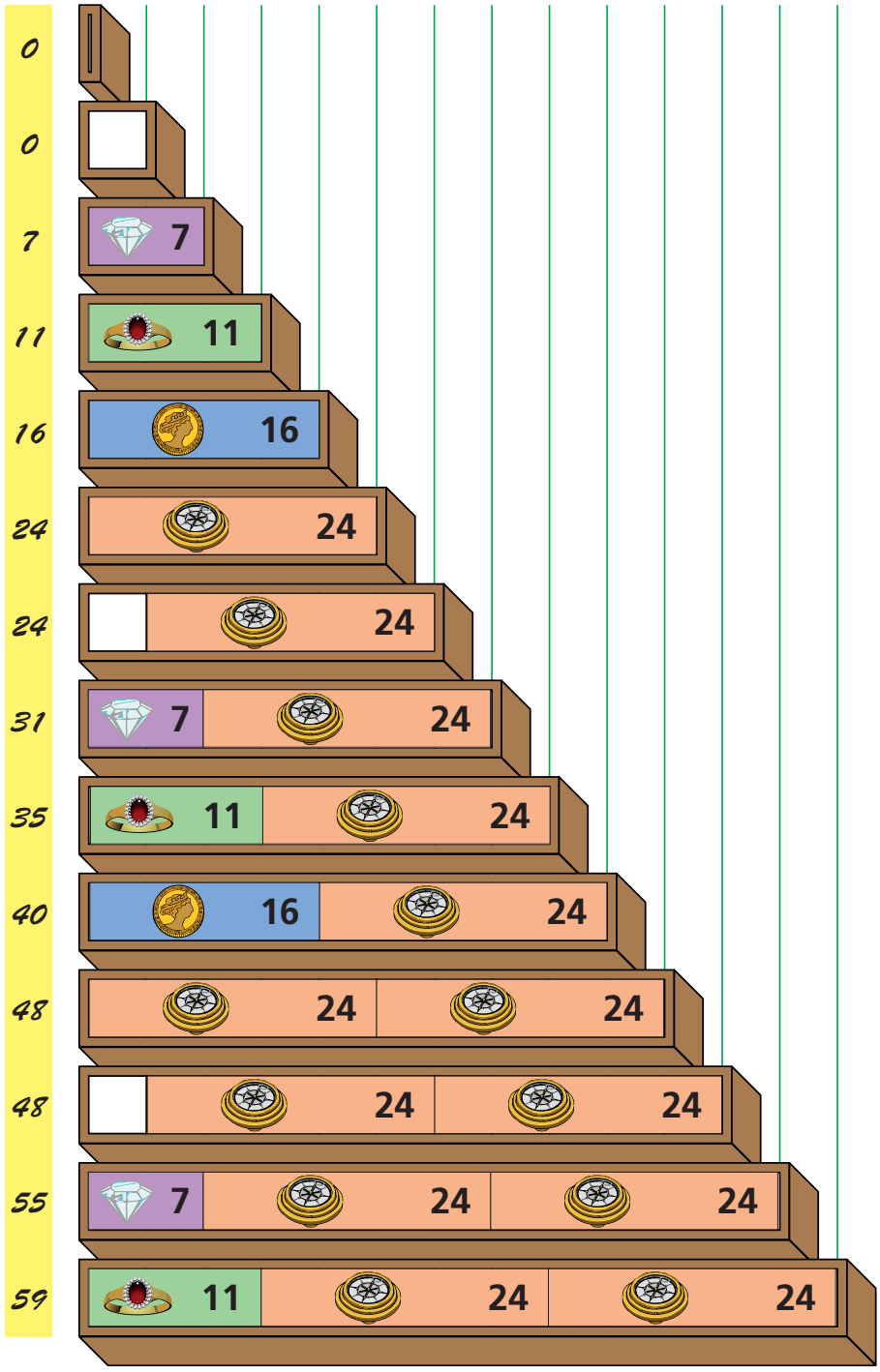
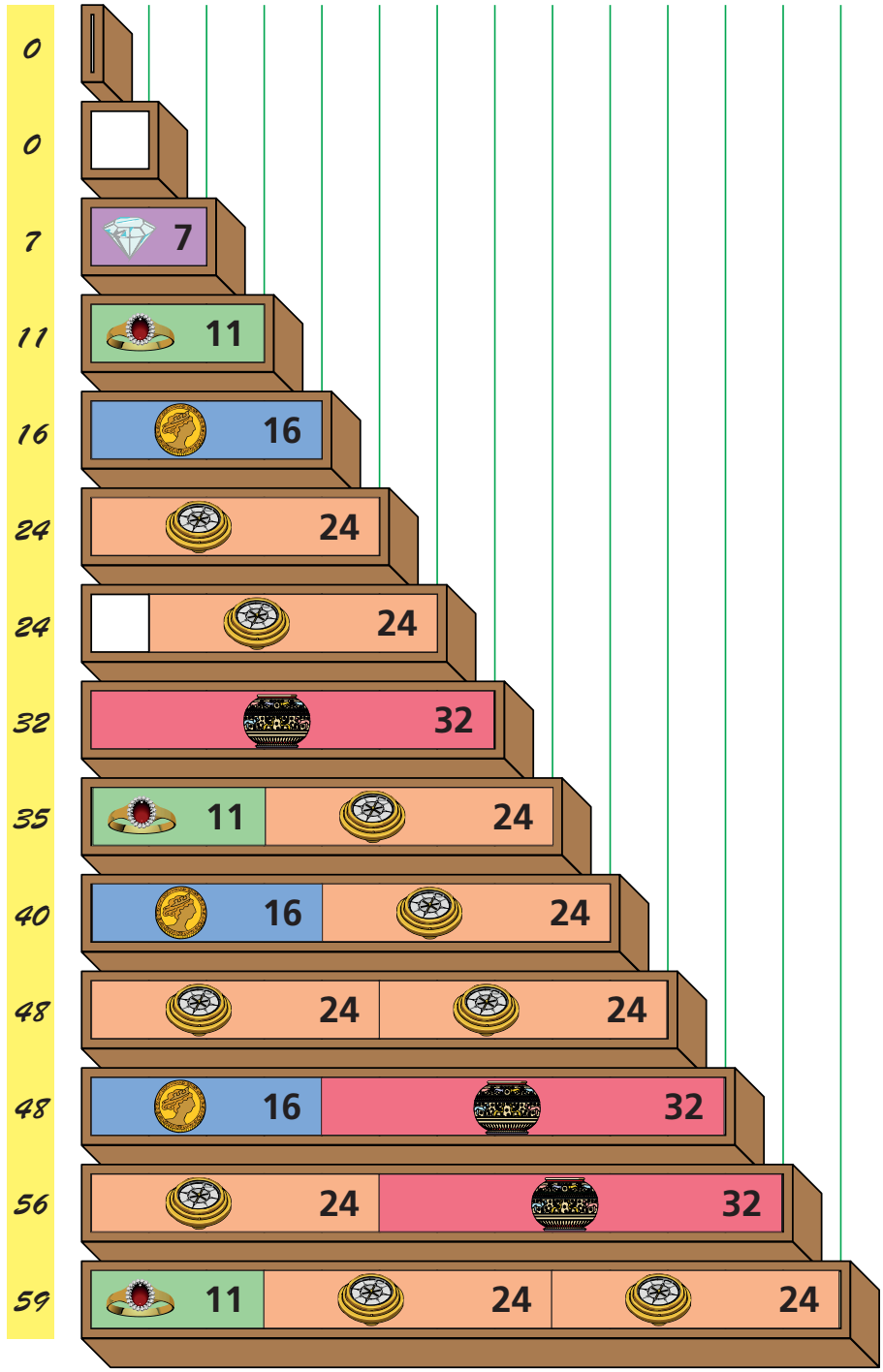


Abbildung 3.11
Der Kompass verändert die optimalen Füllungen nochmals.

Sie kommen nur in wenige Kisten, wie Abbildung 3.12 zeigt. Über eine weitere Besonderheit sind Sie wahrscheinlich gestolpert, als Sie die 11er-Kiste optimiert haben. Hier ist der Wert ohne und mit dem neuen Schatz absolut identisch. Sie können sich daher frei entscheiden oder aber zusätzliche Kriterien definieren wie etwa die möglichst vollständige Ausnutzung des Platzes. In diesem Fall würde man dann – wie im Beispiel geschehen – die Schätze tauschen.

Abbildung 3.12
Die Vase verändert die optimalen Ergebnisse leicht.



Führen Sie nun noch die letzten beiden Schritte durch und ergänzen nacheinander die Krone und die Waage.

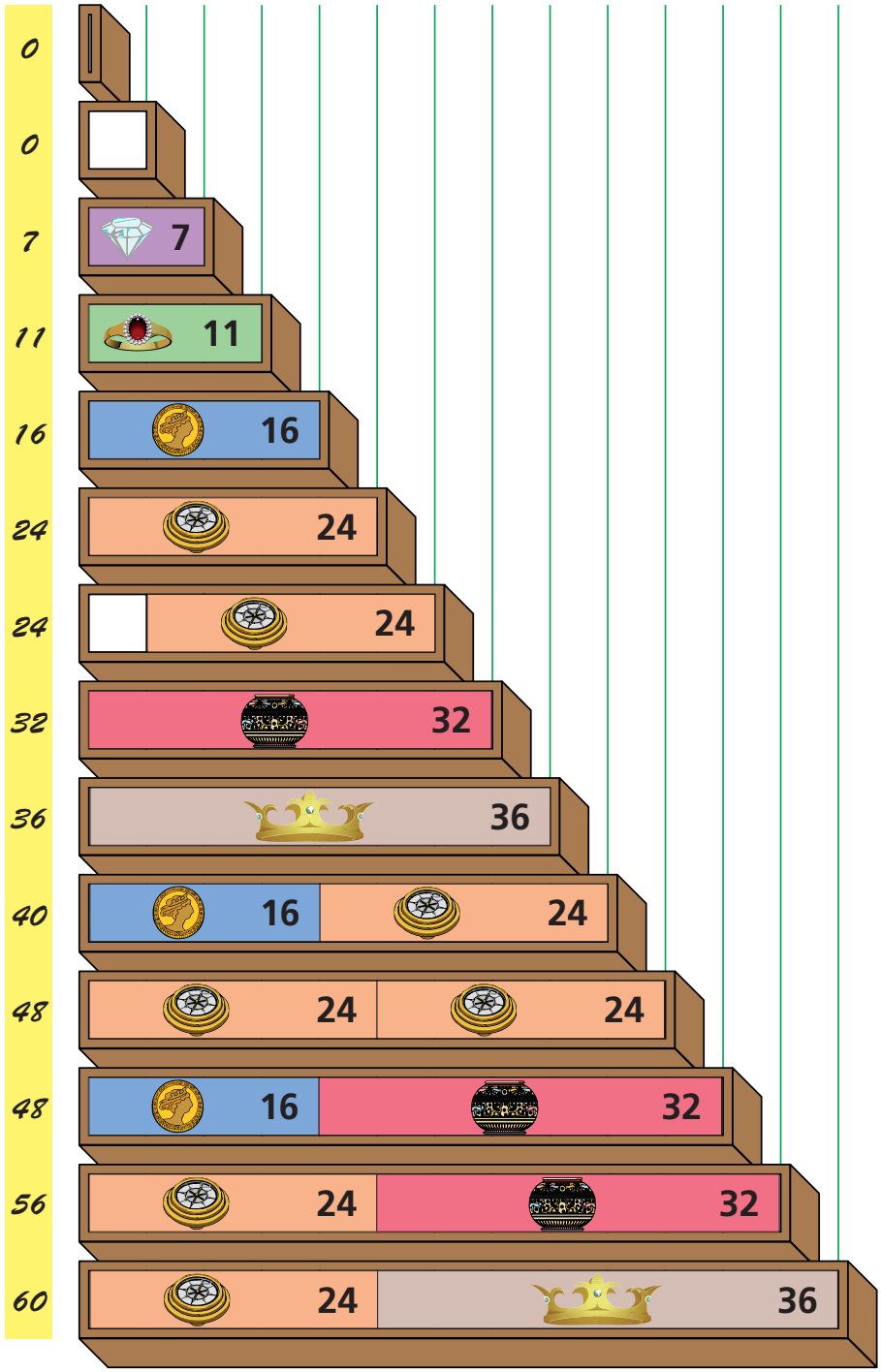
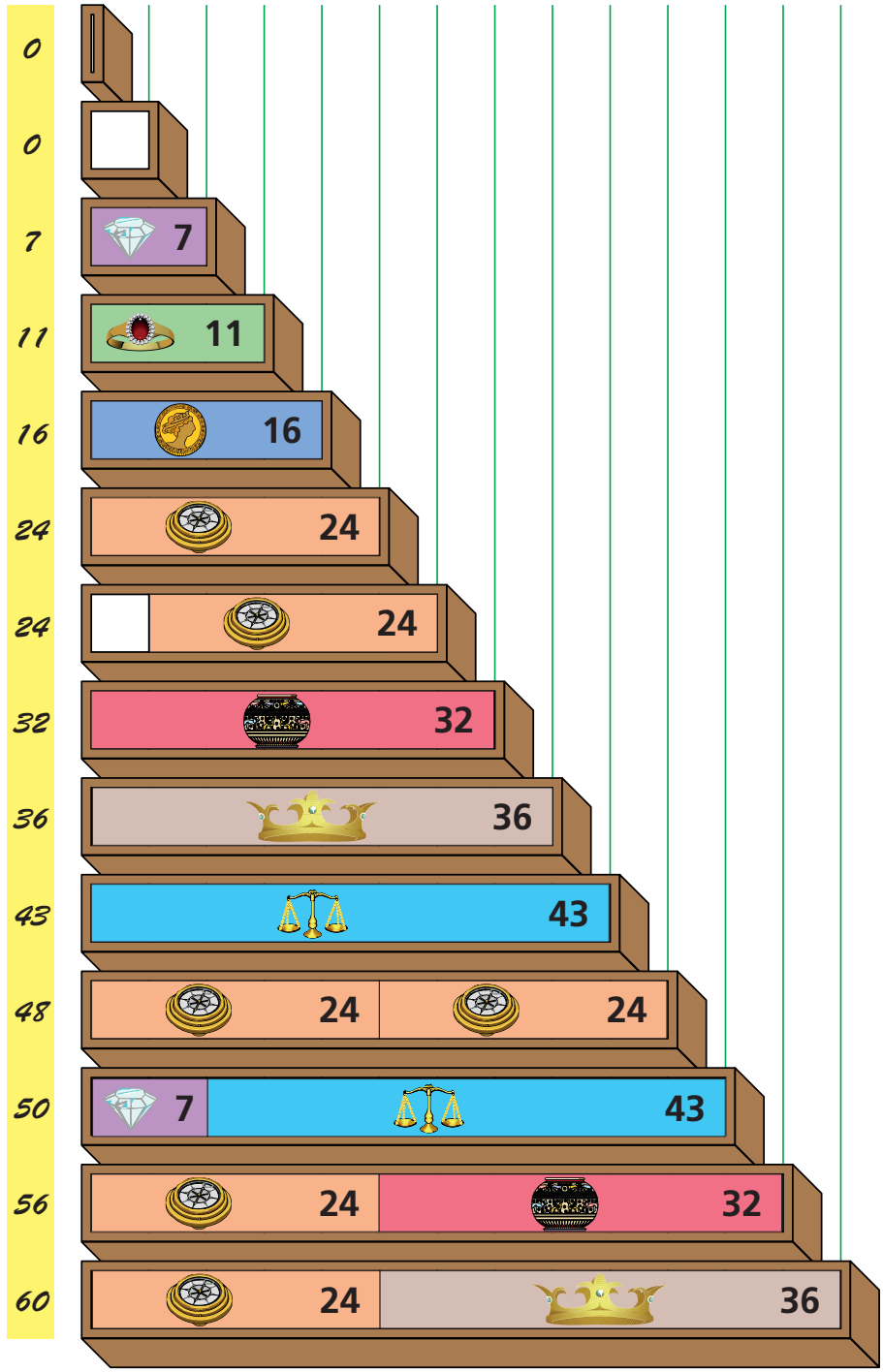


Abbildung 3.13
Mit Krone

Die Abbildungen 3.13 und 3.14 zeigen den nächsten Zwischenschritt sowie das Endergebnis. Obwohl die Krone nur zwei Kisten beeinflusst, ist die entscheidende 13er-Kiste dabei, deren Wert nun auf 60 gesteigert werden kann. Die Waage sorgt danach noch für eine Verbesserung bei 9er- und 11er-Kiste, was aber am Resultat nichts verändert, da uns ja letztlich die 13er-Kiste interessiert und alle anderen nur dazu gedient haben, uns zum optimalen Ergebnis zu führen.

Abbildung 3.14
Endergebnis



Unser Held vom Beginn des Kapitels sollte sich also einen Kompass und eine Krone einpacken und erhält damit einen Belohnung im Gegenwert von 60 Dublonen. Unser Algorithmus nach dem Prinzip der dynamischen Programmierung hat sich bereits ausgezahlt!

Funktioniert das denn auch immer?

Bei der Durchführung des Verfahrens oben bin ich Ihnen noch eine Begründung schuldig geblieben, ob, warum und wie es funktioniert.

Gehen wir nochmals zum Start zurück, als es sozusagen nur einen Schatz gab: den Edelstein.

Die einzelnen Kisten sind bis zum Rand mit Edelsteinen – dem einzig verfügbaren Schatz – gefüllt, mehr passen nicht hinein. Wir können daher festhalten, dass alle Kisten optimal gefüllt sind. Jetzt nehmen wir den nächstgrößeren Schatz hinzu. Wir fangen mit der kleinsten (optimal gepackten) Kiste an und schauen, was passiert, wenn man deren Inhalt mit dem neuen Schatz kombiniert. Das machen wir nacheinander mit allen Kisten, bis der Inhalt zu groß wird.

Allgemein gesprochen kombinieren wir also den Inhalt einer Kiste A mit dem neuen Schatz S. Um das unterzubringen, benötigen wir eine Kiste B mit der Größe von Kiste A plus der Größe des Schatzes S.

Wie erreicht man, dass die Kiste B optimal gepackt ist? Hierfür gibt es nur zwei Möglichkeiten:

In einer optimal gepackten Kiste B kommt Schatz S nicht vor – in diesem Fall besteht die optimale Kiste aus dem, was schon vor der Betrachtung von Schatz S als optimal festgestellt wurde: dem Inhalt der Kiste aus dem letzten Durchlauf.

Kiste B benötigt Schatz S, um optimal gepackt zu sein – in diesem Fall packen wir schon einmal Schatz S in die Kiste (denn wir haben diesen ja gerade als nötig identifiziert). Wie viel Platz haben wir jetzt noch? Genau! Es steht noch so viel Platz zur Verfügung, wie auch Kiste A bietet. Kiste A war aber nach der Voraussetzung bereits optimal gepackt, also liegt es nahe, deren Inhalt zusätzlich zu Schatz S in die Kiste B zu sortieren. Genau das machen wir auch in obigem Verfahren!

Abbildung 3.15 zeigt das graphisch. Indem wir also immer bei den kleineren Kisten anfangen, stellen wir sicher, dass diese bereits optimal sind, wenn wir zu den größeren Kisten kommen. Auf diese Weise können wir die kleineren Kisten schon zur „Konstruktion“ der größeren benutzen. Für jede Kistengröße, in die der neue Schatz passt, beantworten wir die einfache Frage, um die neue, optimal gefüllte Kiste einschließlich ggf. des neuen Schatzes zu ermitteln.

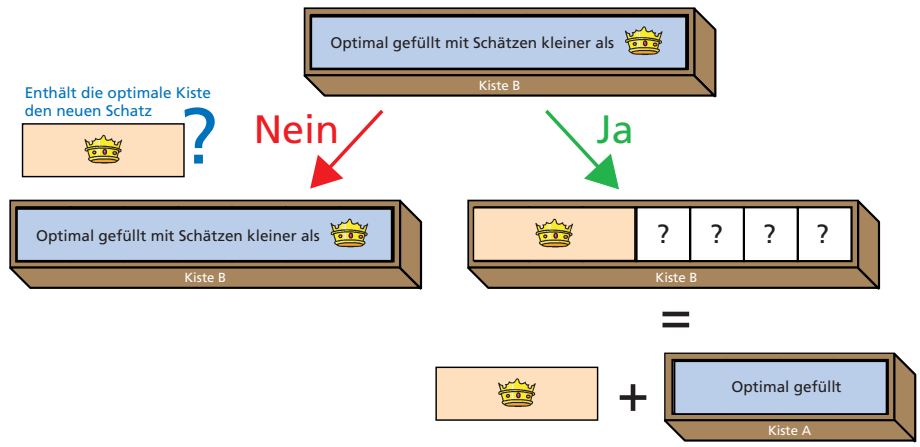


Abbildung 3.15
Entweder der neue Schatz ist sinnvoll in der Kiste unterzubringen oder nicht – eine weitere Möglichkeit gibt es nicht.

Wozu dient diese Erkenntnis?

Könige sind rar geworden! Die Schatzkammern sind Bankkonten gewichen, und die stehen meistens unter der starken Überwachung der Parlamente. Prinzessinnen sind emanzipiert und lassen sich nicht mehr leichtfertig von bösen Räubern entführen.

Wozu also die Vorbereitung auf einen Gang durch die Schatzkammer? Selbstverständlich gibt es für die Lösung des Rucksackproblems sehr viele ernsthafte Anwendungen.

Auf der Hand liegt der Einsatz bei einer Spedition: Ein Fuhrunternehmer hat einen LKW zur Verfügung und kann verschiedene Waren mit unterschiedlichen Gewinnspannen transportieren. Wie packt er den LKW, um den größtmöglichen Nutzen zu haben?

Bei einer erweiterten Version geht es gleich um eine ganze LKW-Flotte: Stellen Sie sich vor, der Spediteur hat einen Auftrag angenommen, eine ganze Reihe unterschiedlicher Objekte zu transportieren – egal wie viele Fuhren er hierfür benötigt. Wenn er es dank dichter Packung schafft, die Ladung optimal auf die LKWs aufzuteilen (also mit wenig ungenutztem Raum), kann er unter Umständen ganze Fuhren einsparen und damit seinen Gewinn steigern.

Denken Sie auch an Arbeitsabläufe: Ein Softwareprojekt muss in einem Monat fertig werden. Wie verteilt man die verschiedenen Aufgabenpakete auf die zehn vorhandenen Mitarbeiter, so dass möglichst alle gleichmäßig bis zum Ende ausgelastet sind und somit das Projekt in der kürzesten Zeit fertig wird?

Während man bei menschlichen Mitarbeitern nicht immer genau schätzen kann, wie lange sie tatsächlich für die Durchführung eines Arbeitspakets brauchen, geht das bei Computern deutlich besser: Eine entsprechende Aufgabe muss gelöst werden, wenn ein Programm von einem Computer mit vielen Prozessoren möglichst schnell ausgeführt werden soll. Auch hier gilt es, die vorhandenen Teilprogramme so auf die Prozessoren zu verteilen, dass alle ungefähr die gleiche Last haben und so alle nahezu gleichzeitig fertig werden. Das funktioniert natürlich sowohl im menschlichen als auch im Computer-Fall nur, wenn die einzelnen Teilaufgaben voneinander unabhängig bearbeitet werden können.

Der Algorithmus

Jetzt sind Sie wieder gefragt: Können Sie aus dem Beispiel von oben einen allgemeinen Algorithmus ableiten, der dann künftig für alle Rucksackprobleme dieser Art verwendbar ist?

Um die Reihenfolge der einzelnen Operationen besser kenntlich zu machen, dürfen Sie gerne auch wieder Zeiger verwenden, zum Beispiel „Kiste A“ und „Kiste B“ aus dem Bastelbogen!

Tipp: In der Lösung werden Sie einige sogenannte „Schleifen“ benutzen. Dies sind die Konstrukte der Art: „Wiederhole das Folgende, bis eine bestimmte Bedingung zutrifft.“ In der bisher verwendeten Notation für Algorithmen können Sie diese „kopfgesteuerten Schleifen“ darstellen wie in Abbildung 3.16. Wenn die Bedingung von Anfang an nicht stimmt, werden die Anweisungen aus dem grauen Kästchen kein einziges Mal ausgeführt.

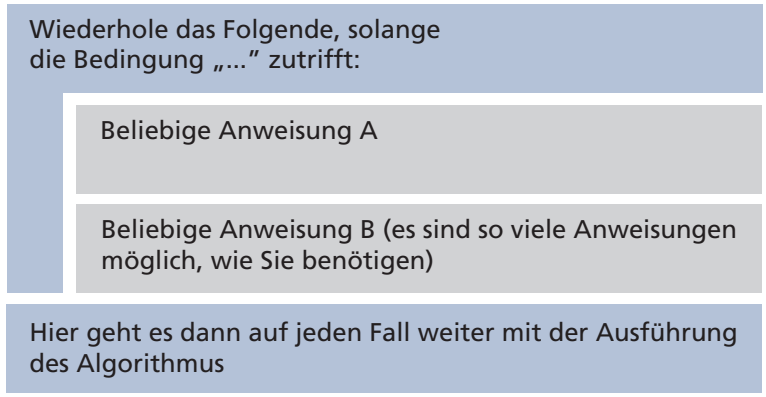


Abbildung 3.16
Notation für die kopfgesteuerte Schleife

Im Gegensatz dazu stehen die „fußgesteuerten Schleifen“ der Art: „Wiederhole das Obenstehende, solange eine bestimmte Bedingung zutrifft.“ Die Notation sehen Sie in Abbildung 3.17. Hier werden die Anweisungen aus dem grauen Kästchen auf jeden Fall einmal ausgeführt, bevor die Bedingung überprüft wird.

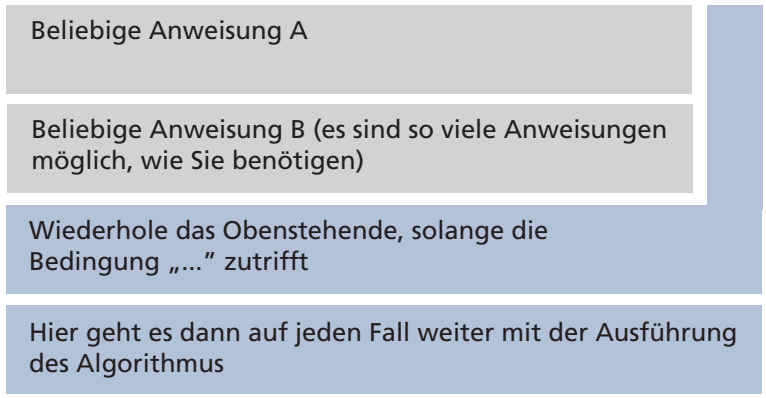


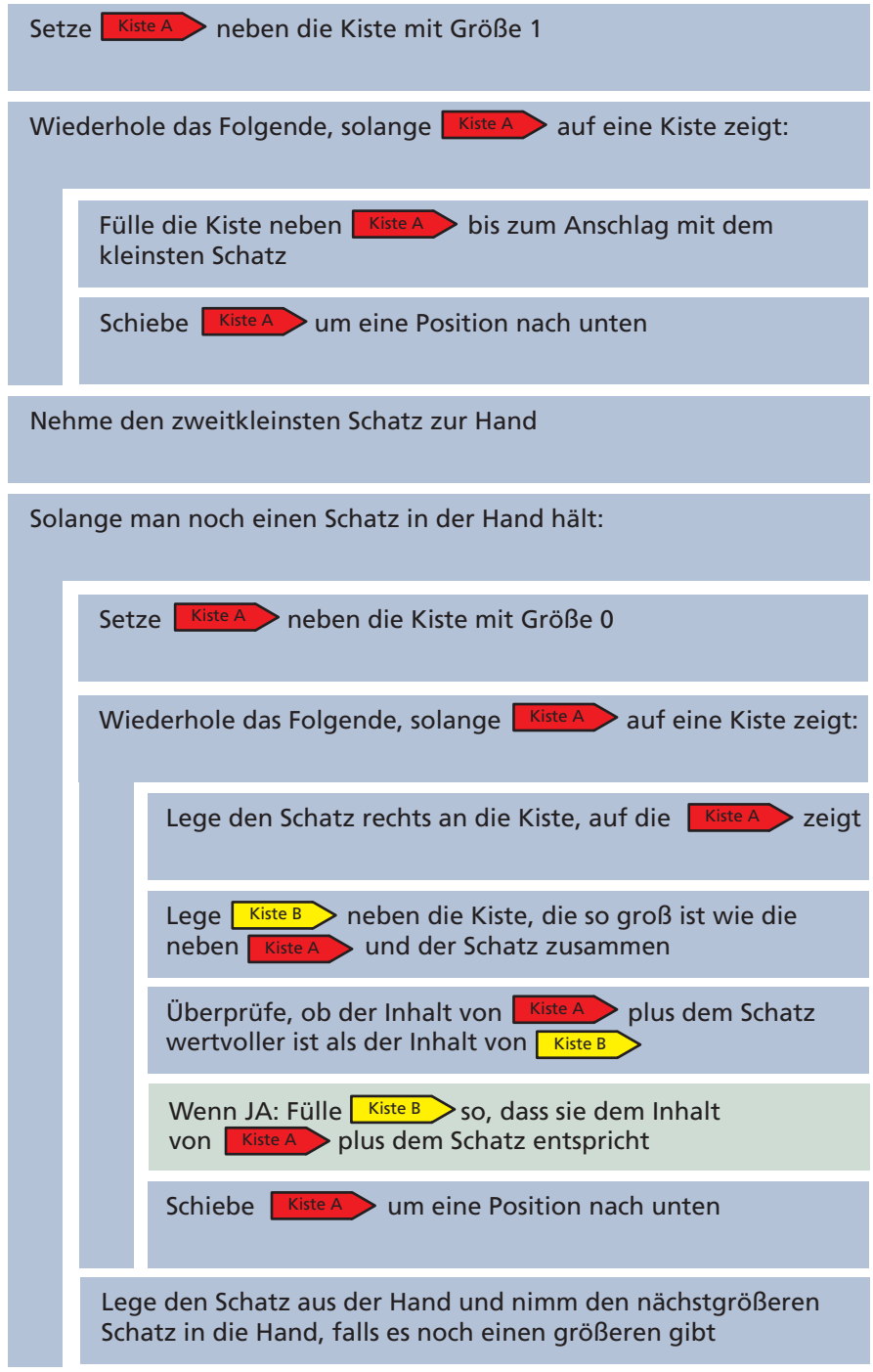
Abbildung 3.17
Notation für die fußgesteuerte Schleife

Sie dürfen die Zeiger verwenden, um die Kisten zu markieren, die miteinander verglichen werden sollen. Der Algorithmus kann dann wie in Abbildung 3.18 auf der nächsten Seite aussehen.

Wenn Sie bereits firm in der Durchführung der Algorithmen sind und sich merken, welche Kisten Sie gerade betrachten, lässt sich das Verfahren auch etwas knapper darstellen wie in Abbildung 3.19. Diese Beispiele sollen unterstreichen, dass es keinen allgemeingültigen Weg gibt, Ihre Ideen aufzuschreiben. Gute Notationen unterscheiden sich in Ausführlichkeit, Formtreue und anderen Parametern. Sie sind dann gut, wenn sie für den jeweiligen Leser verständlich sind.

Sie sehen: Es ist günstig, dieses Buch auch im nächsten Urlaub griffbereit zu haben, wenn es darum geht, die wertvollsten Schnäppchen und Mitbringsel im knappen Fluggepäck unterzubringen ...

Abbildung 3.18
Der Rucksack-Algorithmus



Was steckt dahinter?

Vielleicht erkennen Sie, dass Sie das Prinzip der dynamischen Programmierung bereits kennen gelernt haben: Beim Routenplaner war es ebenfalls einfacher, sämtliche Routen zu Orten zu bestimmen, die weniger weit weg als unser Ziel waren.

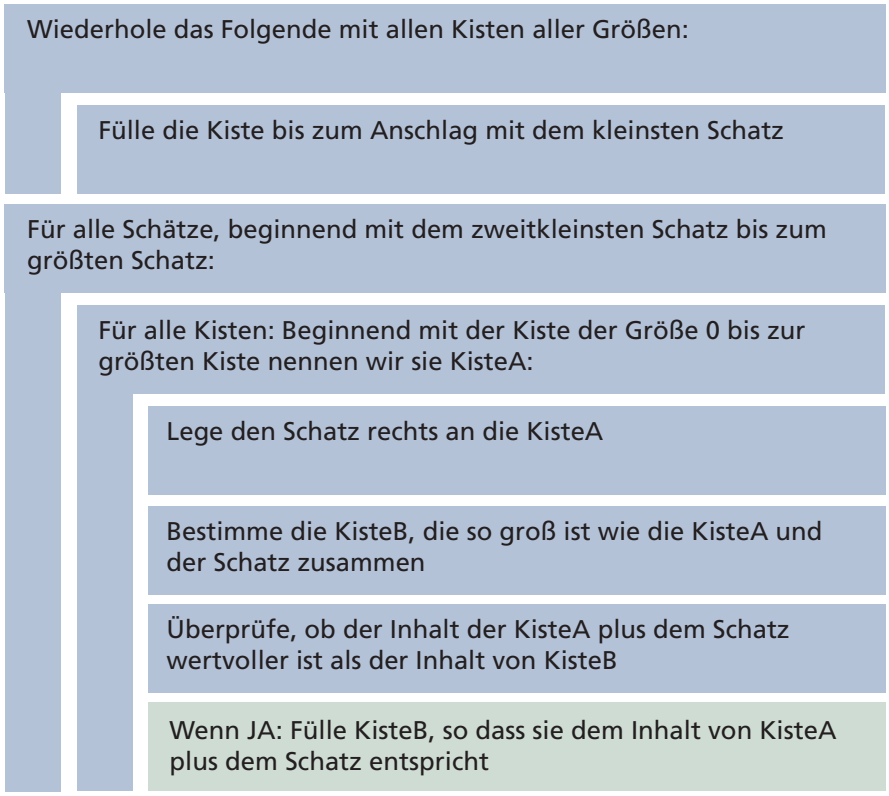


Abbildung 3.19
Der Rucksack-Algorithmus, etwas kürzer gefasst

Genau das machten auch die Ameisen: Sie erkundeten gleichzeitig alle möglichen Orte, bis sie am Ziel ankamen. Etwas Entsprechendes haben wir auch beim Packen der Kisten getan: Wir haben alle kleineren Kisten gepackt und darauf aufbauend den optimalen Inhalt der größeren bestimmt.

Das erinnert stark an ein Prinzip, das in der Mathematik bereits sehr lange bekannt ist und auch in der Informatik eine entscheidende Rolle spielt: die vollständige Induktion.

Vollständige Induktion

Die vollständige Induktion ist ein zweistufiges mathematisches Beweisverfahren. Es funktioniert insbesondere für Aussagen in Bezug auf ganze Zahlen. Man beweist, dass die Aussage für eine bestimmte kleine Zahl a gilt. Ferner beweise man: Falls die Aussage für die Zahl $n - 1$ gilt, dann gilt sie auch für die Zahl n.

Damit hat man dann bewiesen, dass die Aussage für alle ganzen Zahlen größer oder gleich a gilt.

Vollständige Induktion kennt man prinzipiell, seit 1654 der französische Mathematiker und Philosoph Blaise Pascal mathematische Beweise nach diesem Schema durchgeführt hat. Erst mit der Einführung eines formalen Systems für natürliche Zahlen um 1880 wurde daraus dann ein allgemeingültiges Beweisverfahren, das heute auch in der Informatik häufig eingesetzt wird.

Beispielsweise möchten wir beweisen, dass folgender Satz gilt:

$$1+2+3+\ldots+n=\frac{n\cdot(n+1)}{2}$$

Der kleine Gauß ist eine bekannte Bezeichnung für den Satz, der hier als Beispiel für vollständige Induktion genutzt wird. Hintergrund ist eine Geschichte über den jungen Karl Friedrich Gauß, der in der Volksschule als Beschäftigungstherapie die Zahlen 1 bis 100 addieren sollte. Er legte das Ergebnis seinem Lehrer in kürzester Zeit vor, indem er die kleinste und die größte Zahl addierte (=101) und diese Summer mit 50 multiplizierte, denn 1 + 100 = 2 + 99 = 3 + 98 usw., und von solchen Paarungen gibt es 50 Stück.

Zunächst kommt die sogenannte Induktionsverankerung: der Beweis, dass der Satz für eine bestimmte kleine Zahl gilt. In diesem Fall wählen wir $n = 1$. Die Rechnung ist

$$1 = \frac{1 \cdot (1+1)}{2}$$

Wir sehen leicht, dass diese Rechnung stimmt. Als Nächstes kommt der sogenannte Schluss. Dabei nehmen wir an, dass wir den Satz für alle Fälle bis zur Zahl $n - 1$ bereits bewiesen haben und versuchen, ihn mit dieser Annahme ebenfalls für die Zahl n zu beweisen.

Als Formel ausgedrückt nehmen wir als gegeben an:

$$1 + 2 + 3 + \ldots + (n-1) = \frac{(n-1) \cdot n}{2}$$

Zu beweisen ist:

$$1 + 2 + 3 + \ldots + n = \frac{n \cdot (n+1)}{2}$$

Den ersten Teil vor dem Gleichheitszeichen können wir erweitern und $(n - 1)$ explizit aufführen. Dadurch ändert sich an der Aussage nichts:

$$1 + 2 + 3 + \ldots + (n-1) + n = \frac{n \cdot (n+1)}{2}$$

Den ersten Teil der neuen linken Seite kennen wir bereits aus unserer Annahme. Wir setzen hierfür die Formel ein, die wir bereits als bewiesen angenommen hatten:

$$\frac{(n-1) \cdot n}{2} + n = \frac{n \cdot (n+1)}{2}$$

Eine kleine Umformung der rechten Seite, bei der wir den einzelnen Summanden n in den Bruch holen, führt zu:

$$\frac{(n-1) \cdot n + 2 \cdot n}{2} = \frac{n \cdot (n+1)}{2}$$

Nun kann man die beiden Summanden im linken Zähler leicht zusammenführen und sehen, dass die Gleichung gilt. Wir haben bewiesen, dass der Satz für n gilt unter der Voraussetzung, dass er auch für $n - 1$ gilt.

Da wir ihn für $n = 1$ bewiesen hatten, folgt daraus also, dass er ebenfalls für $n = 1 + 1 = 2$ gilt. Wenn er für $n = 2$ gilt, ist er ebenfalls für $n = 3$ bewiesen usw. Insgesamt können wir also sagen, dass die Behauptung für alle $n \geq 1$ gilt.

Kommen wir nach diesem mathematischen Intermezzo zur dynamischen Programmierung zurück. Auch hier gibt es quasi eine Verankerung, bei der das Problem für eine kleine Problemgröße gelöst wird. Im Laufe des Algorithmus leiten wir dann jeweils aus einer kleineren Lösung die nächstgrößere ab.

Im Beispiel des Rucksackproblems bestand die Verankerung darin, das Problem für den Fall zu lösen, dass nur ein einziger Schatz zur Wahl steht: der kleinste.

Danach konstruierten wir aus der Problemlösung der Größe n (eine optimal gefüllte Kiste der Größe n) und dem nächsten Schatz der Größe s die Problemlösung der Größe $n + s$: Die Kiste der Größe $n + s$ wird optimal gefüllt, wenn man sie entweder so lässt, wie sie ist, oder mit dem Inhalt der Kiste n + Schatz s füllt – je nachdem, welche der Möglichkeiten wertvoller ist.

Die Begründung habe ich bereits weiter oben angeführt. Sie setzt voraus, dass man tatsächlich eine optimale Kiste der Größe n hat. Genau genommen handelt es sich hier

um eine Art Induktionsschluss. Auf diese Weise wird gezeigt, dass das Verfahren für immer höhere Problemgrößen ebenfalls korrekt funktioniert.

Versuchen Sie jetzt, für den Dijkstra-Algorithmus aus dem ersten Kapitel die Verankerung und den Induktionsschluss zu finden!

Verankerung:

Der kürzeste Weg vom Startort S zu sich selbst ist 0.

Schluss:

Angenommen, wir haben den (tatsächlich) kürzesten Weg vom Startort S zu den Orten O_1 bis O_{n-1} bestimmt. Dann ist der nächste Ort O_n derjenige, der einem Ort O_x aus der Menge der Orte O_1 bis O_{n-1} benachbart ist und bei dem die Summe aus dem (bereits bestimmten optimalen) Weg von S nach O_x und dem Weg von O_x nach O_n minimal ist.

Sie erkennen hier, dass der Übergang zwischen Informatik und Mathematik fließend ist. Viele Teilgebiete haben beide Wissenschaften gemeinsam, so die diskrete Mathematik (oder Mathematik ganzer Zahlen), die auch hier verwendet wurde.

Das verflixte NP

Eines der großen Probleme bei der vorgestellten Lösung des Rucksackproblems lässt sich aber an der Beziehung zwischen dynamischer Programmierung und vollständiger Induktion auch ablesen:

Die hier vorgestellte Lösung funktioniert nur für „ganzzahlige" Problemgrößen.

Was bedeutet das? Alle Kisten waren sozusagen „linear", sie bestanden aus einer ganzzahligen Anzahl von Quadraten, ebenso die Schätze. Hier konnte man leicht von einer kleineren Kiste auf eine entsprechend größere schließen.

Reale Probleme arbeiten selbstverständlich mit dreidimensionalen Kisten. Jeder „Schatz" hat unterschiedliche Ausmaße in Länge, Breite und Höhe (oder ist sogar völlig unregelmäßig, wie bei einer lose Krone). Außerdem lassen sich reale Schätze nicht in ganzzahlige Kästchen (zum Beispiel glatte Zentimeter-Werte) einteilen. Darüber hinaus geben die echten Schatzkammern oft nicht beliebig viele Schätze einer Sorte her. Was passiert, wenn unser Algorithmus anzeigt, dass man zwei Kronen einpacken soll, aber nur eine da ist?

Polynomiell, nicht polynomiell
Auch in Kapitel 1 haben wir im Abschnitt „Was steckt dahinter?" bereits einführend über dieses Thema philosophiert. Lesen Sie am besten dort nochmals nach, falls hier etwas unklar bleiben sollte! Dort steht übrigens auch, was das Travelling-Salesman-Problem eigentlich ist ...

Hier versagt das Verfahren! Sie sehen, dass oft eine kleine Veränderung der Aufgabenstellung ausreicht, um aus einem „einfachen" Problem eines zu machen, für dessen Lösung kein Algorithmus existiert, der in polynomieller Zeit arbeitet. Bereits im Zusammenhang mit dem Dijkstra-Algorithmus diskutierten wir ausführlich darüber.

Man hat eine ganze Menge von Problemen, die man momentan nicht in polynomieller Zeit lösen kann, in eine Kategorie gepackt, die sogenannten „NP-vollständigen Probleme". Dazu gehören zum Beispiel das allgemeine Rucksackproblem und auch das Travelling-Salesman-Problem (allerdings sind längst nicht alle Probleme, die nicht in polynomineller Zeit lösbar sind, NP-vollständig).

Man konnte bisher für keines der Probleme eine Lösung in polynomieller Zeit finden, aber auch keinen Beweis dafür, dass eine solche Lösung nicht existieren könnte. Gleichzeitig hat man aber nachgewiesen, dass – sollte es eine entsprechende Lösung für eines der Probleme geben – alle Probleme dieser Art in polynomieller Zeit lösbar sind.

Das Thema wird uns weiter beschäftigen, zum Beispiel im Zusammenhang mit dem Affenpuzzle. Vielleicht dient es als kleiner Appetitmacher, wenn ich hier schon einmal erwähne, dass ein Preisgeld von einer Million Dollar dafür ausgesetzt ist, die Frage endgültig zu klären, ob es möglich ist, Probleme der Kategorie „NP-vollständig“ in polynomieller Zeit zu lösen.

Im Klartext: Wenn Sie sich hinsetzen und für eines der beschriebenen Probleme einen Algorithmus basteln, der mit einer (beliebig schlechten, aber) polynomiellen Rechenzeit im Verhältnis zur Problemgröße auskommt, sind Sie der Held der Informatik für die nächsten Jahre! Aber dazu später mehr.

Resümee

Sie haben in diesem Kapitel mit der dynamischen Programmierung erfahren, dass es manchmal durchaus effektiver sein kann, alle Lösungen eines Problems bis zu einer Größe *n* zu bestimmen, als sich nur auf die Lösung der gewünschten Problemgröße zu konzentrieren.

Neben dem vorher behandelten Lösungsschema „top down“, bei dem man große Aufgaben Schritt für Schritt handhabbar macht, indem man sie immer weiter aufteilt, sollten Sie also auch „bottom up“ im Blick haben: Hier fangen Sie gleich ganz unten an und lösen zunächst einfachste Bausteine, um diese dann zur „großen“ Problemlösung zusammenzusetzen.

Gleichzeitig ist das hier behandelte Rucksackproblem ein schönes Beispiel dafür, wie wenige Veränderungen der Aufgabenstellung manchmal ausreichen, aus einem einfachen Problem eines zu machen, das quasi nur noch durch mehr oder weniger geschicktes Ausprobieren lösbar ist.

Abbildung 3.K1
Kopiervorlage für Schätze, Sie brauchen diese drei Mal

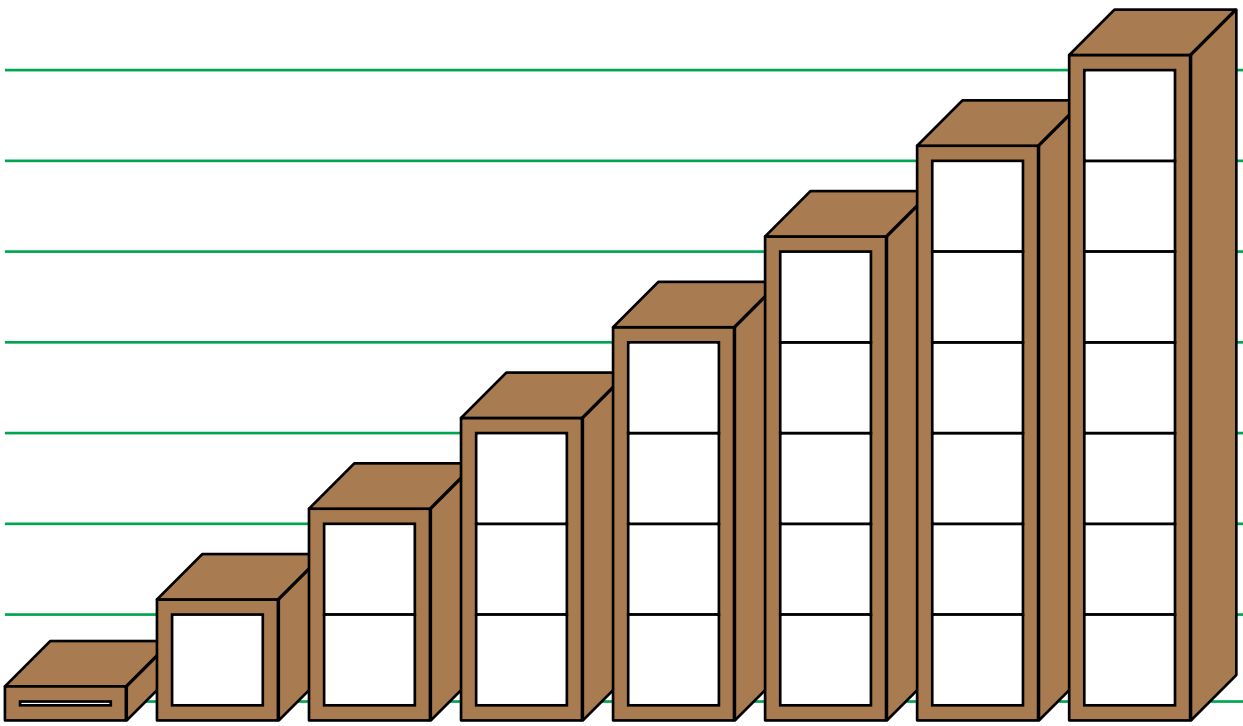
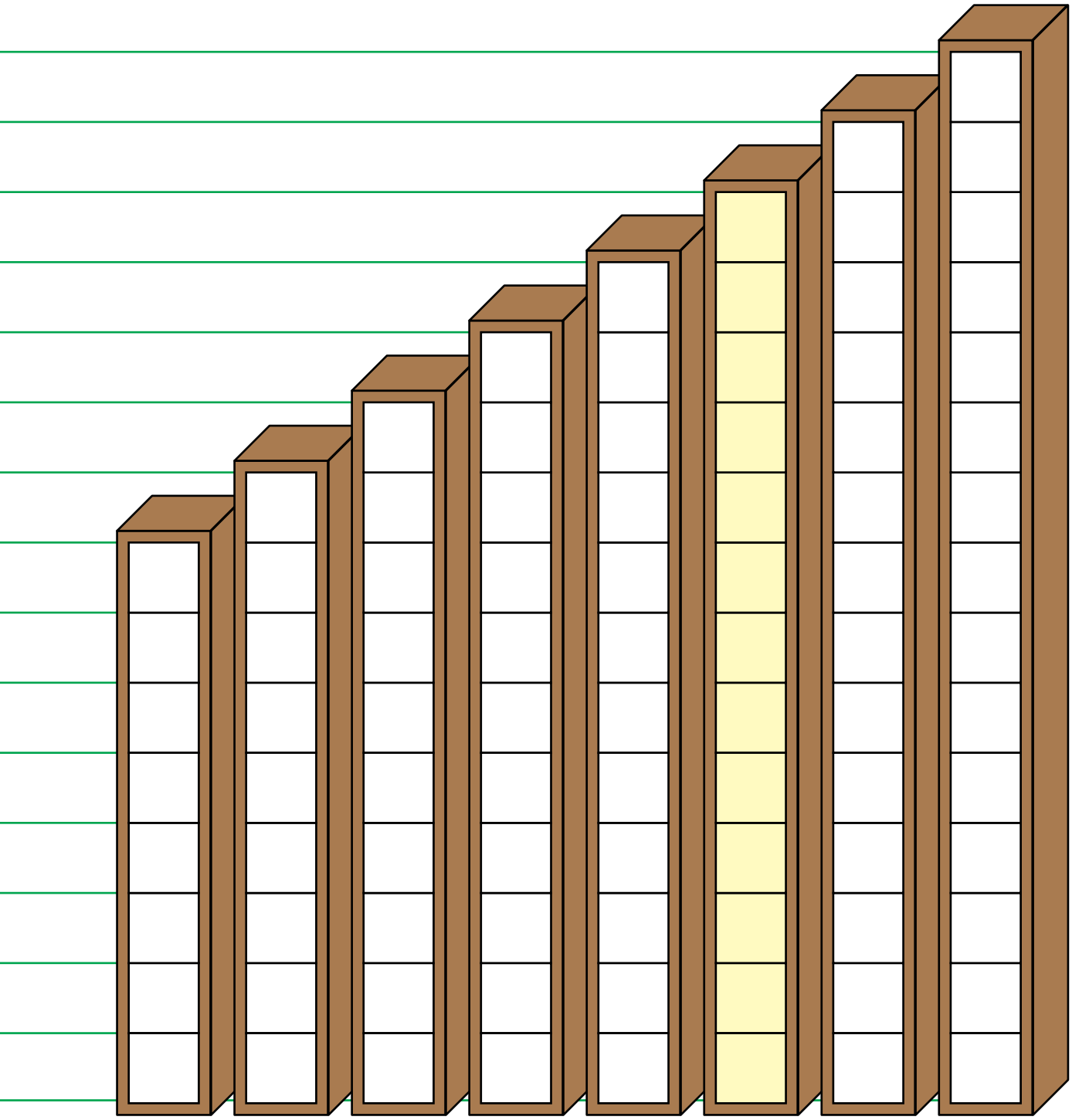


Abbildung 3.K2
Maxikiste als Spielfeld



4. Der Trick mit dem Binären

Es ist allgemein bekannt: Computer rechnen nicht mit dem uns vertrauten Zahlensystem, das wir wahrscheinlich dem Umstand verdanken, zehn Finger zu besitzen. Computer rechnen binär. Auf die Frage, warum das so ist, gibt es verschiedene Antworten. Eine sehr gebräuchliche ist, dass Relais nun einmal zwei Schaltzustände haben und daher der erste funktionierende Computer – der aus diesen Bauteilen bestand – automatisch auf ein Zahlensystem festgelegt war, das mit zwei Zuständen auskommt. Einen meiner Meinung nach noch viel gewichtigeren Grund lernen Sie allerdings erst im nächsten Kapitel kennen. Vorerst wollen wir uns damit begnügen, etwas Spaß und Übung mit diesem Binärsystem zu haben.

Uhr oder keine Uhr?

Abbildung 4.1 zeigt vier seltsame Diagramme aus roten, grünen und blauen Punkten. Unter der Adresse am Buchrand können Sie diese sogar bewegt abrufen, die Punkte leuchten auf und verschwinden wieder. Kann es sich hier um eine Uhr (bzw. die Darstellung einer Uhrzeit) handeln?

http://www.abenteuer-informatik.de/bu.html

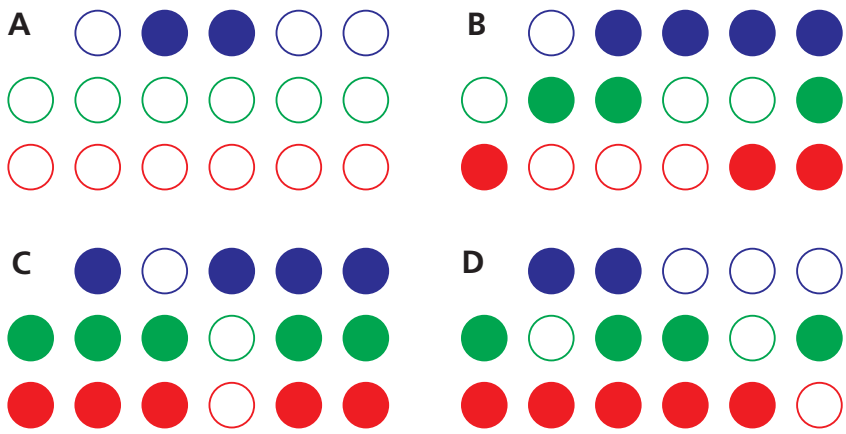


Abbildung 4.1
Uhren?

Rufen Sie die Internetseite auf und suchen Anhaltspunkte, die dafür sprechen.

Es spricht schon einiges dafür: Die roten Punkte wechseln ihren Zustand im Sekundentakt, die grünen minütlich und – auch wenn Sie etwas Geduld brauchen, um das festzustellen – die blauen jede Stunde. So weit entspricht das Verhalten offenbar einer Uhr. Eine Uhr zählt allerdings Sekunden, Minuten und Stunden, und dieser Zählvorgang wird nicht auf den ersten Blick klar.

Daher machen wir noch ein zweites Experiment: Basteln Sie noch die gelben Binärkarten aus Vorlage 4.K1 oder dem Bastelbogen. Die Vorderseite zeigt dabei jeweils eine Zahl von Punkten, die Rückseite ist leer. Wenn Sie die Kopiervorlage nutzen, können Sie jeweils ein Rechteck aus einem Quadrat mit Punkten und einem ohne Punkte ausschneiden, es an der Kante zwischen den Quadraten sauber knicken und dann zusammenkleben oder laminieren.

Versuchen Sie, die Karten nacheinander so auf den Tisch zu legen, dass diese insgesamt 13 Punkte zeigen. Versuchen Sie auch andere Zahlen darzustellen. Bekommen Sie 25 Punkte, 44 und auch 68?

Sicher hatten Sie keine Schwierigkeiten mit 13, 25 und 44 Punkten. Da die Gesamtzahl aller Punkte auf den Kärtchen aber nur 63 beträgt, sind 64 oder mehr unmöglich darzustellen. Abbildung 4.2 zeigt 44 Punkte.

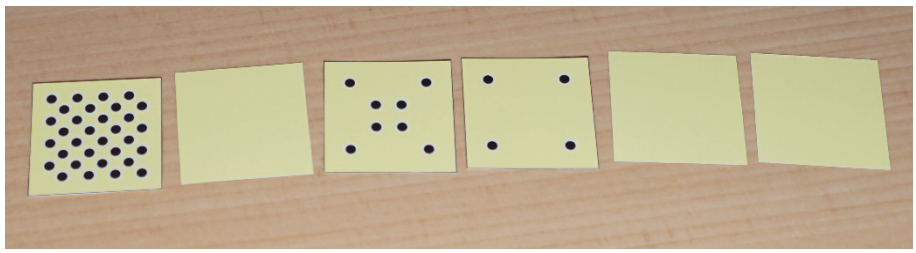


Abbildung 4.2
Die Karten zeigen 44 Punkte.

Nun wollen wir konsequent erforschen, ob auch wirklich alle möglichen Zahlen mit den Kärtchen gelegt werden können. Dafür zählen wir durch, legen also nacheinander alle Zahlen 1, 2, 3 usw. bis zur 63. Bitte sortieren Sie dabei die Karten nach der Zahl der aufgedruckten Punkte – von 32 bis 1 absteigend, unabhängig davon, ob diese gerade sichtbar sind oder nicht. Achten Sie beim Zählen darauf, wie oft Sie die einzelnen Karten jeweils umdrehen.

Sie haben es geschafft? Super! Und wie oft wurde nun die 1er-Karte umgedreht? Genau: jedes Mal. Und die mit den zwei Punkten? Jedes zweite Mal. Und die mit den vier Punkten? Jedes dritte Mal? Nein – jedes vierte Mal!

Wenn Sie nun wieder die blauen, grünen und roten Punkte vom Anfang betrachten, stellen Sie fest, dass auch diese nach dem gleichen Schema wechseln: ganz rechts jedes Mal, links davon jedes zweite Mal, dann jedes vierte usw.

Wenn diese Diagramme nun nach dem gleichen Schema „ticken“ wie das Zählen mit den Binärkarten, kann man vermuten, dass diese auch nach dem gleichen Schema zählen. Versuchen wir also, den Punkten Wertigkeiten zuzuordnen, die der Anzahl an Punkten auf den entsprechenden Karten entsprechen, von rechts nach links also 1, 2, 4, 8, 16, 32.

Können Sie nun die aktuelle Uhrzeit aus der Webseite ablesen? Welche Uhrzeiten verstecken sich in Abbildung 4.1? Finden Sie die falsche Uhrzeit?

An der Binäruhr erkennen Sie, dass man eigentlich gar keine Punkte benötigt, wenn man die Wertigkeit konsequent mit der Position verknüpft. Genau dasselbe gilt ja auch im uns gewohnten Dezimalsystem: Die Ziffer 4 ganz rechts steht einfach nur für eine Vier, während sie eine Position links davon bereits Vierzig repräsentiert.

Auf diese Weise können Sie auch Abbildung 4.1 interpretieren: Uhrzeit A hat von den blauen Stunden die Punkte für 4 und für 8 eingeschaltet, zeigt also 12:00:00 Uhr. B zeigt 15:25:35. Uhrzeit C ist mit 23:59:59 ganz kurz vor Mitternacht. Teil D zeigt allerdings mit 24:45:62 eine Uhrzeit, die es so normalerweise gar nicht gibt.

„Punkt nicht gesetzt“ und „Punkt gesetzt“ können wir mit allem ersetzen, was zwei sichtbar unterschiedliche Zustände hat. Ein besonderer Spaß ist zum Beispiel das binäre Zählen mit den Fingern. An einer Hand schaffen wir es auf diese Weise, mit den fünf Fingern von 0 bis 31 zu zählen. Im Titelbild dieses Kapitels sehen Sie die entsprechenden Fingerstellungen. Mit beiden Händen kommen wir immerhin bis 1023. Abbildung 4.3 zeigt zum Beispiel 879.

Stellen wir die Zustände mit den Symbolen 0 und 1 dar, befinden wir uns in der üblichen Notation für das Binärsystem, die auch in den folgenden Kapiteln verwendet wird.

Zaubern mit Informatik

Verschiedene verblüffende Zauber- und Gedächtniskunststücke beruhen darauf, dass man Zusammenhänge in unterschiedlichen Zahlensystemen unterschiedlich gut wahrnehmen kann. Abbildung 4.K2 ist die Kopiervorlage für sieben Karten, die jeweils Zahlen zwischen 1 und 100 enthalten. Die Karten sind zum Ausschneiden auch auf dem Bastelbogen zu finden.

Um den Trick zu erklären, nehmen wir einmal an, Sie möchten mit Ihrer Zauberkunst Margit beeindrucken. Bitten Sie Margit, sich eine beliebige Zahl zwischen 1 und 100 auszudenken, diese jedoch nicht laut auszusprechen. Margit bekommt nun die sieben Zauberkarten in die Hand und soll Ihnen nur die zurückgeben, die die entsprechende Zahl enthält. Alternativ lassen Sie Margit auch nur auf die Karten zeigen oder die entsprechende Farbe nennen.

Ohne Zögern können Sie nun die Zahl nennen und der Trick hat funktioniert.

Ach so? Sie möchten auch noch wissen, wie Sie das Kunststück zuwege bringen? Also gut: Nehmen Sie von allen bezeichneten Karten die Zahl links oben, also die kleinste Zahl, und addieren Sie diese – fertig.

Führen Sie den Trick ein paar Mal vor und versuchen Sie zu ergründen, warum er funktioniert. Kleiner Tipp: Es hat etwas mit dem Binärsystem zu tun.

Abbildung 4.3 zeigt noch einmal die Zahlen auf den Karten – diesmal aber im Binärsystem. Sie erkennen, dass auf der gelben Karte alle Zahlen stehen, deren letzte Stelle, also die 1er-Stelle, „eingeschaltet“ bzw. 1 ist. Auf der orangen Karte stehen alle Zahlen mit eingeschalteter 2er-Stelle. Grün steht für die 4er-Stelle, Blau für die 8er-Stelle usw.

Wenn Sie die Bastelbögen verwenden, können Sie die richtige Zahl sogar ermitteln, obwohl Sie nur die Rückseiten zu sehen bekommen, denn dort steht die kleinste Zahl nochmals binär und Sie haben nun doch kein Problem mehr im Entziffern von Binärzahlen, oder?
Im Beispiel oben kommen Sie ganz leicht auf die Zahl 0111100 binär, was 60 dezimal entspricht.

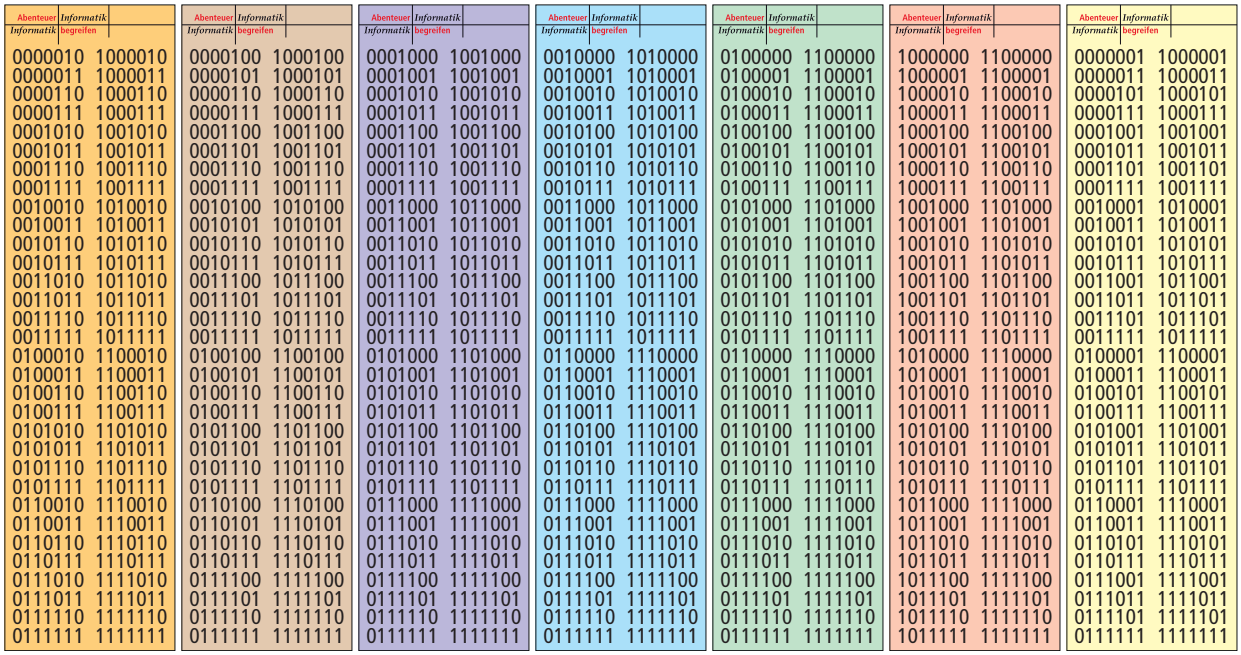


Abbildung 4.3
Zauberkarten mit Zahlen im Binärsystem

Bei der kleinsten Zahl jeder Kategorie sind dann selbstverständlich alle anderen Stellen „ausgeschaltet“ oder 0. Jetzt können Sie sich vorstellen, dass eine Zahl „zusammengesetzt“ werden kann, indem man alle Zahlen, die jeweils eine 1 an den entsprechenden Stellen repräsentieren, zusammenzählt.

Wenn Sie immer noch unsicher sind, nehmen Sie einfach an, die Zahlen in Abbildung 4.3 seien Dezimalzahlen, die eben nur 0 und 1 enthalten. Die Zahlen sind also Eins, Zehn, Elf, Hundert, Hunderteins, Hundertzehn, Hundertelf, Tausend usw.

Wenn Margit sich nun die Zahl 100101 ausdenkt, findet Sie diese auf der gelben, der grünen und der braunen Karte. Die Zahlen oben links stellen dann sozusagen die „Komponenten“ der einzelnen Stellen dar und addieren sich wieder zur ganzen gesuchten Zahl.

Was steckt dahinter?

Jedes Zahlensystem beruht letztlich auf dem Zählen. Dazu kann man zum Beispiel die Finger benutzen, womit man bis 10 kommt (oder eben zehn unterschiedliche „Ziffern“ 0 bis 9 darstellt). Das alte Volk der Maya hat wahrscheinlich zusätzlich zu den Fingern noch die Zehen genutzt und daher ein 20er-System entwickelt.

In jedem Fall gibt es eine begrenzte Zahl auf diese Weise darstellbarer Ziffern – um darüber hinaus weiter zählen zu können, muss man sich etwas einfallen lassen. In additiven Systemen wie dem römischen wurden dann spezielle Zahlzeichen für größere Zahlen entwickelt, also z. B. X für 10 oder C für 100. Auch dann ist der Vorrat aber irgendwann einmal erschöpft. Daher gehen Stellenwertsysteme wie unser Dezimalsystem einen anderen Weg: Ist beim Zählen die Zahl der Ziffern erschöpft, erhöht man die Ziffer links davon um eins und beginnt dafür bei der rechten Ziffer wieder bei 0. Ist man dort ebenfalls bei der letzten Ziffer angekommen, geht man noch eine weitere Stelle nach links usw.

Für unser bekanntes Dezimalsystem ist das Ihnen selbstverständlich geläufig: Das Zählen funktioniert mit den uns bekannten Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – nun sind diese erschöpft und wir müssen die Stelle links davon erhöhen. Diese ist nicht vorhanden, wir nehmen aber implizit eine unerschöpfliche Anzahl von Nullen an. Daher geht es weiter mit 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 und wir müssen erneut die linke Ziffer erhöhen, so dass wir auf 20 kommen usw.

Auf diese Weise erhöhen wir die Stelle ganz rechts jedes Mal, die Stelle links davon jedes 10te Mal, die Stelle links davon jedes 100te Mal usw. Wir sagen dazu 1er-, 10er,- und100er-Stelle. Der Wert der Stellen kann daher mit Potenzen der Basis unseres Zahlensystems, also Potenzen von 10 beschrieben werden:

$$dcba = d \cdot 1000 + c \cdot 100 + b \cdot 10 + a \cdot 1 = d \cdot 10^3 + c \cdot 10^2 + b \cdot 10^1 + a \cdot 10^0$$

Das Herumreiten auf Ihnen längst bekannten Fakten war wichtig, um zu zeigen, dass im Binärsystem im Prinzip alles genau gleich abläuft. Basis ist hier allerdings die 2. Daher gibt es nur die Ziffern 0 und 1. Beim Zählen gehen diese Ziffern also viel schneller aus und man muss viel früher mit dem sogenannten „Übertrag“ die Stelle links davon erhöhen:

0, 1, 10, 11, 100, 101, 110, 111, 1000 usw.

Die Wertigkeiten der Stellen ergeben sich dann genau so wie beim Dezimalsystem, aber natürlich auf Basis der 2 – binär ausgedrückt 10. Um die dezimale 10 von der binären 10 zu unterscheiden, wird manchmal die Zahlenbasis (dezimal ausgedrückt) als Index an die Zahl gehängt:

$$2_{10} = 10_2$$

Oft wird dabei der Index 10 für das Dezimalsystem auch weggelassen. Die Rechnung mit den Wertigkeiten der Stellen von oben gilt dann genauso für das Binärsystem:

$$dcba = d \cdot 1000_2 + c \cdot 100_2 + b \cdot 10_2 + a \cdot 1_2 = d \cdot 2^3 + c \cdot 2^2 + b \cdot 2^1 + a \cdot 2^0$$

In dieser Rechnung können Sie dann auch die Anzahl von Punkten auf den gelben Kärtchen von Abbildung 4.K1 erkennen: Jeweils die Zweierpotenzen.

Da im Prinzip alles so ist, wie wir es von unserem „normalen“ Zahlensystem kennen, aber eben nur mit zwei Ziffern, funktionieren auch (fast) alle bekannten Arten zu rechnen und ergeben die gleichen Resultate. Sie müssen sich lediglich daran gewöhnen, den Übertrag beim Addieren schon etwas früher zu nutzen. Für die schriftliche binäre Addition, die Sie in Abbildung 4.4 sehen, gilt $0+0=0;\ 0+1=1;\ 1+0=1;\ 1+1=10$. Bereits bei 1+1 gibt es also einen Übertrag.

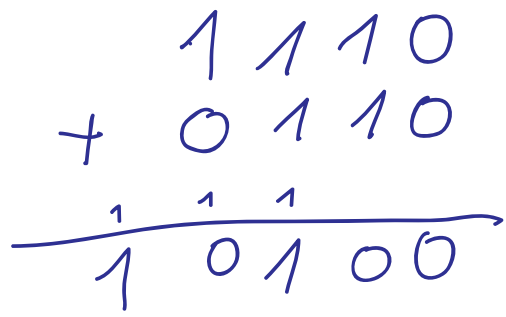


Abbildung 4.4
Addieren im Binärsystem mit Überträgen

Resümee

Binärzahlen sind keine fremde Welt, sondern lediglich eine andere Repräsentation der uns bestens bekannten Zahlen durch die Kombination von nur zwei unterschiedlichen Symbolen.

Manchmal ist es nötig, die Sichtweise zu verändern, um bestimmte Sachverhalte besser zu erkennen. In unserem Fall war es auf diese Weise möglich, einen bekannten Zaubertrick zu erklären.

Noch nicht wirklich geklärt ist, warum Binärzahlen und Informatik fest zusammengehören. In den folgenden Kapiteln werden wir daher immer wieder auf die binäre Darstellung zurückkommen, um die Informatik besser zu verstehen und dieser Sache auf den Grund zu gehen.

Abbildung 4.K1
Punktkarten

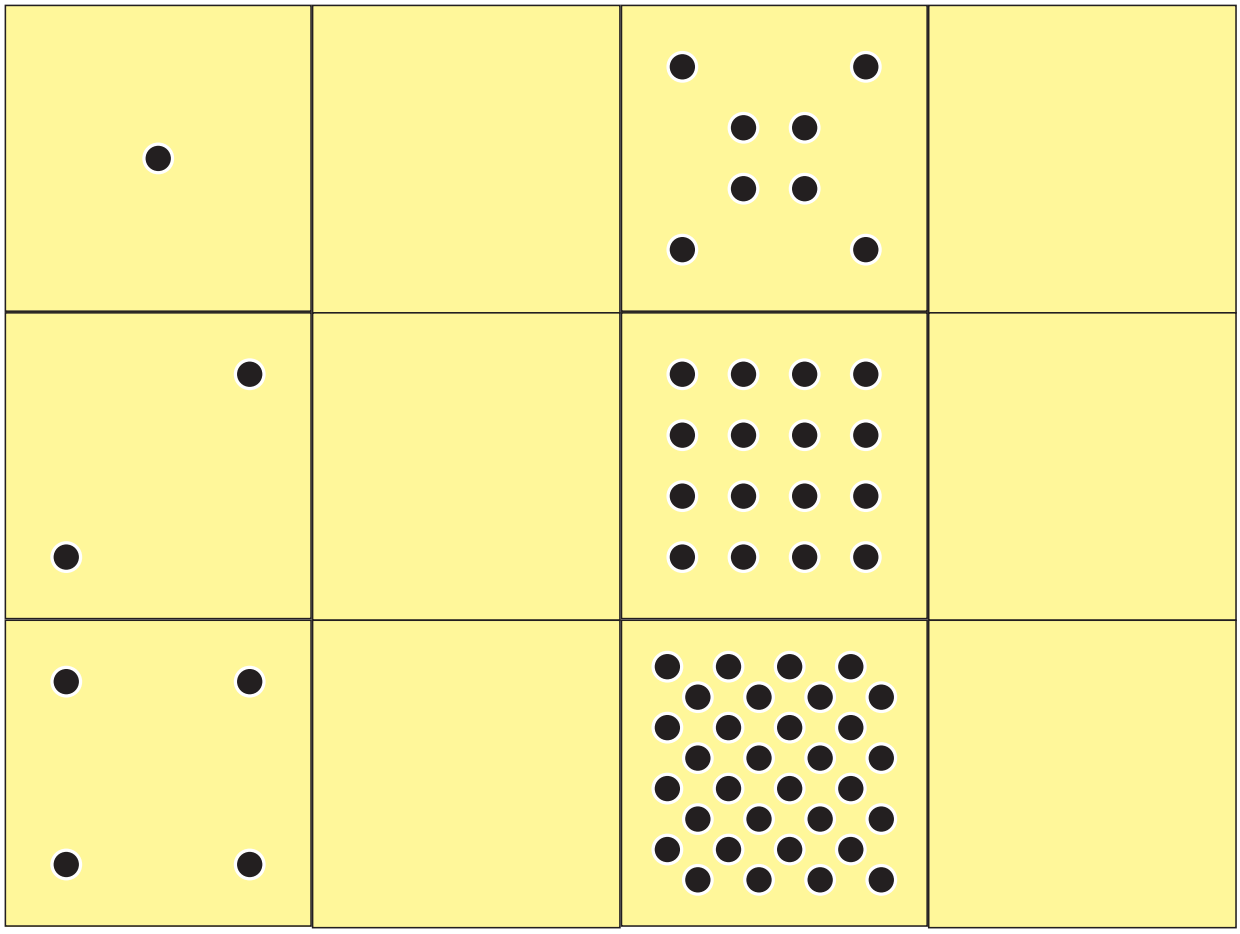
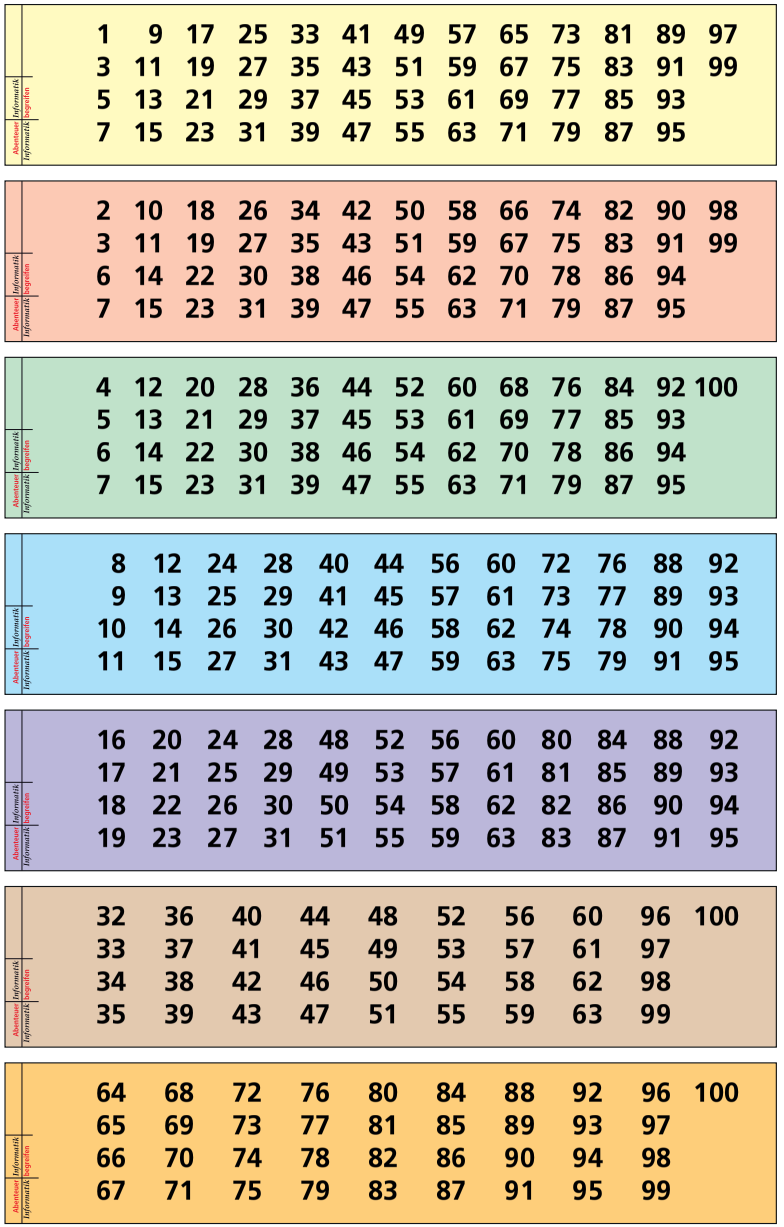


Abbildung 4.K2
Zauberkarten




GIZEH-WERK, BERGNEUSTADT
Nr.14.0409.1
GIZEH
GIZEH-WERK, BERGNEUSTADT 1
Ziffernkarte 3

5. 100000000000 Jahre Informatik?

100 Milliarden Jahre Informatik? Nein, nicht ganz: Binärzahlen haben Sie ja gerade näher kennen gelernt und die Zahl 100000000000 entspricht im üblichen Dezimalsystem der 2048. Viele Leserinnen und Leser wundern sich bestimmt trotzdem noch: Eventuell ist der Name Konrad Zuse bekannt, der in den 30er-Jahren des 20. Jahrhunderts den ersten funktionierenden elektrischen Computer gebaut hat, aber 2048 Jahre ist das noch nicht her!

In den letzten Kapiteln konnten Sie jedoch nachvollziehen, dass Informatik eine Disziplin des menschlichen Denkens und der menschlichen Kreativität ist. Computer stellen dabei die wesentlichen Werkzeuge dar, sind aber genau genommen nicht Gegenstand unserer Wissenschaft!

Auf den folgenden Seiten werde ich daher die für viele unbekannte Geschichte der Informatik etwas näher beleuchten und darlegen, warum es fast unmöglich ist, einen genauen Beginn der Informatik festzulegen. Auf jeden Fall liegt dieser aber deutlich weiter zurück als Konrad Zuses erster Erfolg beim Bau des Computers. Machen Sie sich selbst ein Bild! Natürlich dürfen Sie dabei wieder fleißig experimentieren. Dafür erhebt dieses Kapitel keinesfalls Anspruch auf historische Vollständigkeit. Vielmehr soll es durch einige ausgewählte Schlaglichter anregen, selbst noch mehr herauszufinden. Viel Vergnügen bei lebendiger Geschichte!

Rechenmaschinen

Bereits seit Jahrtausenden bauten Menschen Maschinen, um sich Arbeiten des täglichen Lebens zu erleichtern bzw. diese überhaupt zu ermöglichen. Hebelkonstruktionen ließen unsere Urahnen bereits sehr große Gewichte bewegen. Transportwagen brachten Güter zu entfernten Orten. Spezielle Öfen erlaubten es, Ton wasserdicht zu brennen und Eisenerz zu schmelzen. Die Geschichte menschlichen Erfindungsreichtums ist umfangreich.

Der Einsatz von Apparaten beschränkte sich aber durchaus nicht nur auf mechanische Tätigkeiten. Den ersten „Rechenapparat“ bildeten sicherlich die menschlichen Finger, zusammen mit einfachen Regeln für die Addition und Subtraktion, so wie sie heute auch noch von Kindern auf der ganzen Welt verwendet werden. Die plausible Begründung für unser heutiges Zahlensystem mit zehn verschiedenen Ziffern sind unsere zehn Finger.

Kerbholz, wie es seit Jahrtausenden genutzt wurde. Als „Quittung“ konnte es längs auseinandergebrochen werden, dann hatte jeder Vertragspartner die Möglichkeit, nachträgliche Manipulation festzustellen.

Kerbhölzer dienten den Babyloniern schon vor über 30.000 Jahren als Zähl- und Rechenhilfe. Zum Addieren ritzte man nacheinander die Summanden ein und zählte danach alle Kerben.

Vielleicht fängt die Geschichte der Informatik aber auch erst an, wenn die Hilfsmittel für das Rechnen ein Stellenwertsystem repräsentieren: Die Perlen im über 3000 Jahre alten Abakus stehen je nach Position für 1, 10, 100, ... Chinesen und Römer nutzten einen Typ Abakus wie den in Abbildung 5.1, der auch noch jeweils Perlen für den Faktor

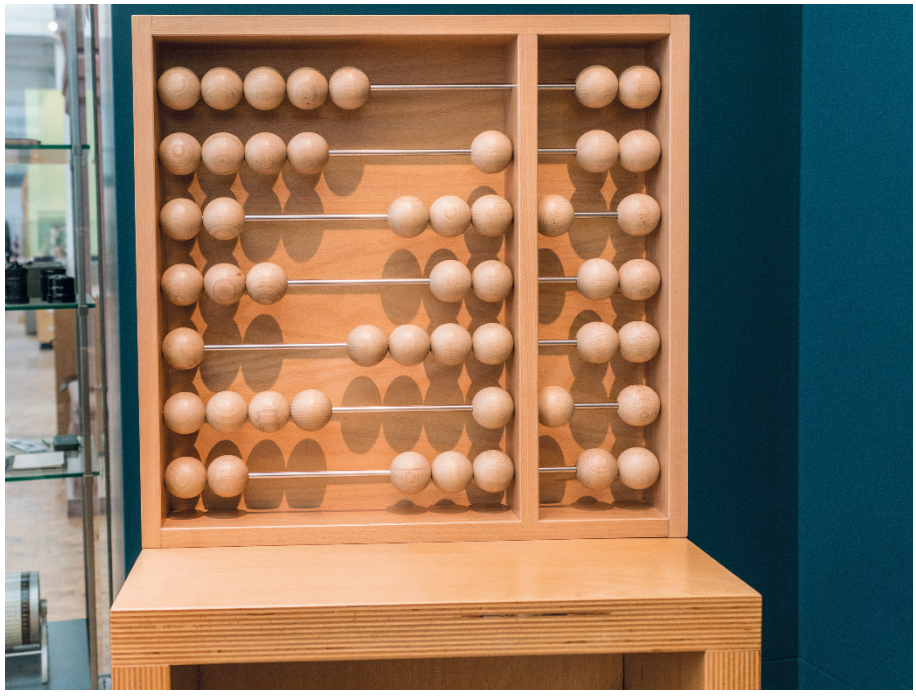


Abbildung 5.1
Abakus aus Holz – nach römischer bzw. chinesischer Bauart. Die dargestellte Zahl ist 182.463.

Wilhelm Schickard (1592–1635) war Universalgelehrter und Professor für biblische Sprachen, Mathematik und Astronomie an der Universität Tübingen.

Gottfried Wilhelm Leibniz (1646–1716) war als Jurist, Naturwissenschaftler, Politiker, Philosoph, Historiker, Theologe und Diplomat. Er verband immer die Mathematik und besonders Zahlen mit der Mythologie. 1673 stellte er seine Rechenmaschine vor, die nicht nur Addition und Subtraktion, sondern auch Multiplikation und Division beherrschte.

5 hatte, um sich die Arbeit mit den Grundrechenarten zu vereinfachen. Die Ausführung wurde über die Zeit immer besser, aber an der grundsätzlichen Funktionsweise hat sich bis heute nichts verändert. Geht man in China in einen kleinen Laden, zum Markt oder auch in ein modernes Kaufhaus, wird dort heute noch oft der Preis mit einem Abakus bestimmt.

Können Sie mit dem Abakus umgehen? Zum Rechengerät wird er nämlich erst in Verbindung mit festen Abläufen, die man erlernen muss, um ihn zu bedienen. Der Mensch ist hier noch von entscheidender Bedeutung.

Mit der Zeit wurden dann auch Geräte entwickelt, die schon mehr nach „Maschine“ aussahen. So konstruierte 1623 der Tübinger Professor Wilhelm Schickard einen Apparat, der automatisch addieren und subtrahieren konnte. Diese Maschine gilt als erste urkundlich erwähnte Rechenmaschine mit einem Zahnradgetriebe (Abbildung 5.2). Selbst Multiplikation und Division sind (allerdings unter starker menschlicher Mithilfe) möglich.

Innerhalb weniger Jahre zogen unzählige andere berühmte Wissenschaftler mit eigenen Modellen nach, so Blaise Pascal und Gottfried Wilhelm Freiherr von Leibniz. Die Nachbauten sind heute noch funktionsfähig im Deutschen Museum in München zu bewundern.

Auch wenn die beschriebenen Maschinen geistige statt körperlicher Arbeit verrichten oder erleichtern konnten, so waren sie doch vom Typ her eher mit dem Hebelkran oder dem Pferdewagen vergleichbar als mit einem Computer von heute. Warum?

Bei all diesen Maschinen war von ihrer Erbauung an klar, wofür sie gebraucht wurden: Man konnte mit einer Maschine Lasten transportieren, mit der anderen Additionen ausführen. Niemand ist auch nur auf die Idee gekommen, zum Beispiel mit einer Rechenmaschine Musik zu komponieren oder Texte zu schreiben – hierfür gab es wiederum spezielle Apparaturen.

Abbildung 5.2
Rekonstruktion der Schickard'schen Rechenmaschine im Technoseum Mannheim

Ein Umbau zu einer Spieluhr oder einer Satzmaschine wäre vielleicht möglich gewesen – es wäre dabei aber ein völlig neuer Apparat entstanden, der dann wiederum nur eine vorbestimmte Funktionsweise besessen hätte.

Blaise Pascal (1623–1662). Sein Vater war ein hoher Steuerbeamter und für ihn erfand Pascal 1642 eine Rechenmaschine, die „Pascaline". Anfangs konnte sie nur addieren, nach zehn weiteren Jahren Entwicklungsarbeit auch subtrahieren. Pascal ist uns heute hauptsächlich als Namensgeber des Pascal'schen Dreiecks und des „Satzes von Pascal" geläufig, beides war allerdings bereits lange vor seiner Zeit bekannt. Die Programmiersprache wurde seiner Rechenmaschine zu Ehren nach Pascal benannt, ebenso die Einheit des Luftdrucks, um seine diesbezüglichen Experimente zu würdigen.

Babbage und seine Idee

Beim heutigen Computer ist alles anders: Seine Herstellung legt noch überhaupt nicht fest, ob er später einmal auf einem Schreibtisch für Textverarbeitung verwendet wird oder in einer Industriehalle Roboter steuert oder an Bord eines Flugzeugs den Piloten bei der Navigation unterstützt: Seine Funktionsweise wird weitgehend nicht von seiner Bauart bestimmt, sondern von der Software, die darauf läuft.

Charles Babbage war im England des 19. Jahrhunderts als Mathematiker und Naturwissenschaftler tätig. Er entwickelte für die Krone verschiedene Rechenapparate, zum Beispiel einen, mit dem man mechanisch Quadratzahlen bestimmen konnte, um auf diese Weise genaue Rechentabellen herzustellen. Solche Tabellen waren wiederum für die Navigation auf See unerlässlich und ein Großteil der britischen Macht beruhte auf dem zielgenauen Einsatz der Flotte.

Babbages Rechenapparate funktionierten gut und der Staat finanzierte den Erfinder dafür. Allerdings verwendete dieser immer mehr Entwicklungszeit auf seine eigentliche Vision: die Analytical Engine (deutsch „untersuchende Maschine"). Im Wesentlichen handelte es sich um einen komplexen mechanischen Apparat, der statt einer festen Funktion eine Programmierung durch Lochkarten erlaubte.

Diese Idee war revolutionär: Nicht nur, dass hier eine Maschine entworfen wurde, ohne ihre genaue Funktion zu umreißen, sie hatte auch bereits wichtige Eigenschaften

Abbildung 5.3
Lochkarten zur Steuerung eines automatischen Webstuhls. Sie enthalten die Information für das Muster des Stoffes.

Charles Babbage (1791–1871) war Mathematiker und Naturwissenschaftler und – wie man sagen könnte – einer der ersten Ingenieure. Er entwickelte auf Basis seiner fundierten wissenschaftlichen Ausbildung Maschinen für verschiedene Tätigkeiten, die vorher dem Menschen vorbehalten waren, etwa das Erstellen von Tabellen für die Navigation von Schiffen. Er hatte als Erster die Idee für eine „universelle Maschine", deren Zweck zum Zeitpunkt des Erbauens der Maschine noch nicht feststand, die also nachträglich programmiert werden konnte (und musste).

heutiger Computer. So unterschied Babbage zwischen dem „Store", also dem Speicher für die Daten, und der „Mill", die einem Rechenwerk, also etwa einem heutigen Prozessor entspricht. Abbildung 5.4 zeigt eine vereinfachte Konstruktionszeichnung von der „Mill". Mit den Lochkarten konnte man dann programmieren, welche mathematischen Grundoperationen das Rechenwerk auf bestimmten Speicherstellen ausführen sollte und welche Teile dieses Programms unter welchen Bedingungen zu wiederholen sind.

Lochkarten sind keinesfalls eine Erfindung der Informationstechnik, sondern werden seit 1745 dazu genutzt, automatische Webstühle mit einem Muster für Stoffe und Teppiche zu „programmieren". Das Verfahren stammt aus Frankreich und wurde von Jacques de Vaucanson entwickelt, 1804 von Joseph-Marie Jacquard verbessert, nach dem es heute noch benannt ist.

Die Karten für besonders hübsche Stoffmuster oder -bilder waren sehr viel wert und wurden häufiger entwendet. Erste Formen der „Software-Piraterie"? Abbildung 5.3 zeigt Lochkarten, die in einen automatischen Webstuhl eingelegt sind.

Interessanterweise war damals die Frauenquote in der Informationstechnologie sehr hoch, denn neben Babbage beschäftigte sich nur noch eine Person mit der Analytical Engine: Ada Augusta, Countess of Lovelace. Sie war die Tochter des englischen Dichters Lord Byron und kombinierte sein poetisches Erbe mit einer fundierten wissenschaftlichen Ausbildung.

Als eine von sehr wenigen erkannte sie das große Potential von Babbages Erfindung und unterstützte ihn fortan – einerseits als sein Sprachrohr zur englischen „High Society" und andererseits, indem sie bereits ausführliche Programme schrieb. Die so erstellte Software diente einerseits zur Berechnung mathematischer Größen wie den Bernoulli-Zahlen, andererseits aber auch zur Komposition komplexer Musik. Ada ging daher als erste Programmiererin in die Geschichte ein. 1979 wurde eine Pro-

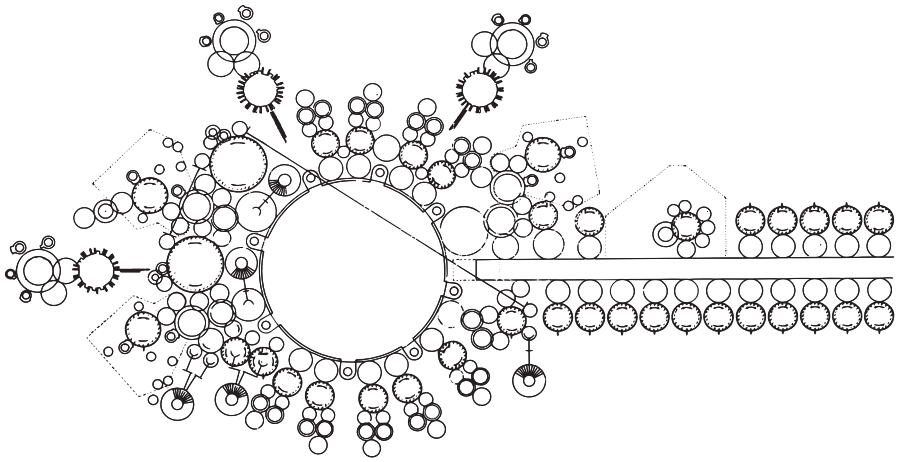


Abbildung 5.4
Babbages „Analytical Engine" hatte ein mechanisches Pendant zum modernen Rechenwerk. Er nannte dies „Mill", weil die Zahnradkonstruktion an eine Mühle erinnert.

grammiersprache nach ihr benannt. Sie wird noch heute als Symbolfigur genutzt, um mehr Frauen für technische Studien und Berufe zu begeistern.

Tragisch: Obwohl die meisten der Programme lauffähig waren, erlebte Ada die Ausführung auf einer echten Maschine nie. Die Analytical Engine blieb in der Planungsphase stecken. Wie so viele große Erfinder beschäftigte sich Babbage nicht genügend mit der Akquise von Mitteln für die Realisierung. Bei seinem Tod 1871 hatte er noch nicht einmal einen Prototyp komplett fertiggestellt und seine Idee geriet lange Zeit in Vergessenheit.

Eventuell hat bei Babbages Maschine jedoch auch ein wichtiges Konzept gefehlt. Welches könnte das sein? Wir haben ja bereits besprochen, dass darin ganz wichtige Ideen moderner Computer durchaus vorhanden waren: die Aufteilung in unterschiedliche Werke, eine Art Programmablaufspeicher, ein Antrieb usw.

Allerdings wollte Babbage, dass seine Maschine in dem uns Menschen vertrauten Dezimalsystem arbeitet. Um zu ergründen, warum das eventuell eine schlechte Entscheidung ist, gehen wir in der Geschichte vorübergehend noch einmal viel weiter zurück: in die Zeit antiker Hochkulturen in Ägypten und Äthiopien.

Ada Augusta Byron King, Countess of Lovelace (1815–1852) war für ihre Zeit eine Ausnahme, da sie – unterstützt von ihrer Mutter und später von ihrem Ehemann – als Frau in Mathematik und Naturwissenschaften ausgebildet war. Ada gilt als erste Programmiererin der Welt, da sie nicht nur Babbages Ideen dokumentierte und ergänzte, sondern auch für die damals nie gebaute Analytical Engine erste Programme verfasste.

Die Äthiopische Multiplikation

Abbildung 5.5 zeigt zwei Händler, die sich über den Preis für 37 Amphoren mit Datteln geeinigt haben: Jede Amphore soll 13 Shekel kosten. Aber wie viel muss nun insgesamt bezahlt werden? Weder Käufer noch Verkäufer konnten multiplizieren.

Die einfachste Möglichkeit wäre sicher gewesen, neben jeder Amphore 13 Shekel zu deponieren und diese dann einzusammeln. Tatsächlich ist jedoch ein anderes Verfahren überliefert, das heute unter den Bezeichnungen „Äthiopische Multiplikation", „Ägyptische Multiplikation" oder „Russische Bauernmultiplikation" bekannt ist.

Es beruht auf zwei Spalten mit Kuhlen, die in den Sand gegraben und mit unterschiedlichen Anzahlen von Steinchen befüllt werden. Sie können die Kuhlen aus Abbildung 5.K1 oder dem Bastelbogen verwenden. Als Ersatz für Steinchen eignen sich zum Beispiel Linsen oder Steckperlen. Sie können natürlich auch einfach Zahlen schreiben.

Abbildung 5.5
Händler verhandeln einen Preis – sie können allerdings nicht rechnen.

Ich zeige die durchzuführenden Rechenschritte anhand des Beispiels der Multiplikanden 37 und 13, bitte vollziehen Sie diese mit anderen, selbst ausgedachten Zahlen nach. Falls Sie tatsächlich kleine Objekte zum Zählen verwenden, sollten Sie auf jeden Fall kleinere Zahlen als im Beispiel benutzen – warum, werden Sie bald sehen.

Führen Sie nun den folgenden Algorithmus durch. Abbildung 5.6 zeigt zum Vergleich die Schritte mit den Beispielzahlen.

1. Die beiden Multiplikanden kommen in die oberen beiden Kuhlen

2. Wir betrachten nur die **linken** Kuhlen: In jede leere Kuhle kommen halb so viele Steinchen wie in die Kuhle darüber. Geht das nicht auf, runden wir ab. Das kann man machen, indem man so viele Steinchen in die Hand nimmt wie darüber, und dann nur jedes zweite Steinchen in die Kuhle darunter abzählt. Bleibt abgerundet kein Steinchen übrig, hören wir auf.

3. Wir betrachten nur die **rechten** Kuhlen: In jede leere Kuhle kommen doppelt so viele Steinchen wie in die Kuhle darüber. Das machen wir in allen Kuhlen, für die in der Kuhle links davon auch Steinchen liegen. Das kann man machen, indem man zwei Mal jeweils so viele Steinchen abzählt wie in der Kuhle darüber, und sie in die Kuhle darunter gibt.

4. Noch einmal nur in der **linken** Spalte: Aus allen Kuhlen so lange immer zwei Steinchen auf einmal herausnehmen, bis sich nur noch eines oder gar keines darin befindet. Bleibt keines übrig, die Steine der Kuhle rechts daneben auch gleich entfernen.

5. Alle verbleibenden Steinchen der rechten Spalte zusammenzählen. Sie haben das Ergebnis.

Gemeinsamkeit von Computer und Wüstenhändler

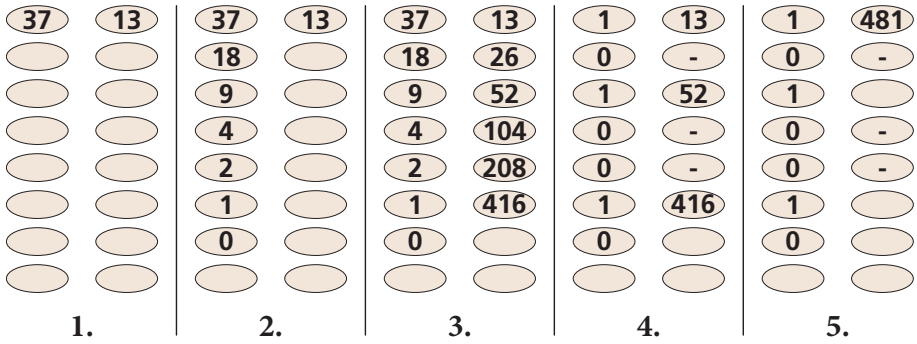


Abbildung 5.6
Äthiopische Multiplikation 37 mal 13

Äthiopisches Multiplizieren macht Spaß – keine Frage. Trotzdem möchten Sie sicherlich wissen, was dieses Verfahren in einem Buch über Informatik zu suchen hat. Denken Sie darüber zunächst einmal selbst nach!

Auf jeden Fall handelt es sich um einen Algorithmus, der eine komplizierte Operation durch eine Zahl deutlich einfacherer Operationen erledigt: Das Multiplizieren größerer Zahlen ist schwierig, aber halbieren, verdoppeln und zählen lernen wir schon in der ersten Klasse! Genau solche Vorschriften zu finden, die Kompliziertes in Einfaches zerlegen, ist eine wichtige Aufgabe der Informatiker.

Dass diese Art der Algorithmisierung aber nicht nur etwas für Informatiker, sondern etwas sehr Menschliches ist, zeigt nicht nur die Äthioptische Multiplikation, sondern viele Vorgehensweisen des Alltags. Betrachten wir doch einmal unser normales, schriftliches Multiplizieren. Wenn wir etwa 431 und 305 schriftlich multiplizieren, zerlegen wir es auch in kleinere Schritte. Je nach Grundschule gehen wir den ersten oder den zweiten Multiplikanden Ziffer für Ziffer durch und multiplizieren mit dem anderen Multiplikanden. Nebenstehend werden die Ziffern von 305 von hinten nach vorne jeweils mit 431 multipliziert. Dabei muss man durch Verschieben oder Ergänzen von Nullen auch noch die Stellenwertigkeit repräsentieren. Selbstverständlich kann man diese Rechnungen auch noch weiter zerlegen, also etwa 431 · 5 in 400 · 5 + 30 · 5 + 1 · 5, das wollen wir uns aber hier sparen.

```
   ↓↓↓
431 · 305
2155   = 431 · 5
  00   = 431 · 00
129300 = 431 · 300
131455
```

Aus dem letzten Kapitel wissen wir, dass Rechnungen im Binärsystem in der Regel genauso ausgeführt werden können wie Rechnungen im uns bekannten Dezimalsystem.Versuchen Sie das doch und multiplizieren Sie die (binären) Zahlen 1101 und 100101 schriftlich.

Die Rechnung mit dem Ergebnis sehen Sie am Rand der nächsten Seite. War diese nun besonders kompliziert? Wohl kaum! Statt des kleinen Einmaleins benötigten Sie lediglich die Multiplikation mit 0 oder mit 1.

```
1101 · 100101
1101
00
110100
0000
00000
110100000
111100001
```

Die Binärzahlen, die wir hier genommen haben, entsprechen genau den dezimalen Zahlen im Beispiel für die Äthiopische Multiplikation – 13 und 37. Das Ergebnis ist dann auch das binäre Pendant zu 481.

Grund genug, dass wir die Rechnung der Wüstenvölker nochmals betrachten, diesmal allerdings ebenfalls mit binär dargestellten Zahlen.

Wir beginnen, wie nebenstehend abgebildet, mit den beiden obersten Kuhlen. Wir gehen nun die linke Spalte von oben nach unten durch und halbieren jeweils die Zahl.

100101	1101

Was bedeutet halbieren? Im Binärsystem heißt das, durch die Basis des Zahlensystems zu dividieren. Um sich das vorzustellen, denken Sie bitte an das vertraute Dezimalsystem: Was passiert, wenn Sie hier eine beliebige ganze Zahl durch die Basis des Zahlensystems – also 10 – dividieren (ohne Rest)?

Genau: Die letzte Stelle fällt einfach weg. Das Gleiche passiert auch im Binärsystem. Von Kuhle zu Kuhle wird die Zahl also einfach immer um die letzte Stelle kürzer. Auch dieser Schritt ist am Rand zu sehen.

In der rechten Spalte verdoppeln wir – das bedeutet, mit der Basis des Zahlensystems zu multiplizieren. Wieder können wir unsere Erfahrungen vom Dezimalsystem übertragen: Es wird jeweils einfach eine Null angehängt.

100101	1101
10010	
1001	
100	
10	
1	

Nun nochmals zur linken Spalte: Es werden aus jeder Kuhle paarweise Steinchen eliminiert – also immer so viele, wie die Basis des Zahlensystems ist. Auch hier versuchen wir aus dem Dezimalsystem zu schließen, was passiert. Wenn wir eine beliebige, ganze Zahl immer wieder um 10 reduzieren, werden alle Stellen ab der Zehnerstelle so lange heruntergezählt, bis nichts mehr übrig bleibt als die Einerstelle, die davon ja nicht betroffen ist.

100101	1101
10010	11010
1001	110100
100	1101000
10	11010000
1	110100000

Bei der Äthiopischen Multiplikation bleibt also jeweils nur die letzte Stelle der binären Repräsentation der Zahl stehen. Vergleichen Sie: Von oben nach unten gelesen ergeben die Ziffern in den Kuhlen die Originalzahl.

Steht links eine Null, wird die rechte Zahl ebenfalls entfernt, sonst bleibt sie stehen – es handelt sich um die Zahl von ganz oben rechts, also den zweiten Multiplikanden, ergänzt um eine entsprechende Zahl von Nullen.

Zum Schluss addieren Sie alle verbleibenden Zahlen der rechten Spalte.

1	1101
0	00
1	110100
0	0000
0	00000
1	110100000

Vergleichen Sie die Vorgehensweise mit der schriftlichen binären Addition. Was stellen Sie fest?

Die alten Wüstenvölker haben genau nach dem gleichen Prinzip multipliziert wie wir, allerdings quasi im binären Zahlensystem. Sicherlich nicht, weil es so kompliziert ist, sondern ganz im Gegenteil: Weil es so schön einfach ist!

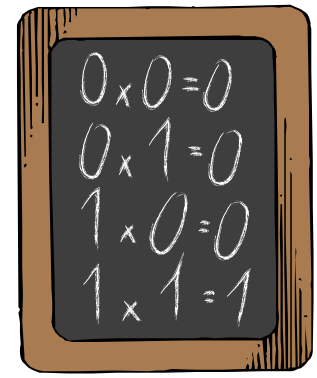
1	1101
0	00
1	110100
0	0000
0	00000
1	110100000
	111100001

Stellen Sie sich vor, Sie hätten in der Grundschule auch nur das kleine Einmaleins des Binärsystems lernen müssen. Genau vier einfache Zeilen und fertig! Keine peinlichen Momente, wenn einem gerade ´mal entfallen war, ob 7 mal 9 eher 61 oder 63 sind...

Genau aus diesem Grund multiplizieren Computer noch heute prinzipiell nach dem gleichen Verfahren wie die Wüstenhändler es vor einigen Jahrtausenden niedergeschrieben haben. Alles, was wir als Menschen auswendig lernen müssen, kann auch für den Computer nicht als Programm abgeleitet werden – es muss fest in Hardware

gebaut werden. Wie das genau funktioniert, werden wir später im Buch ergründen. Zunächst glauben Sie mir aber sicherlich, dass die Implementierung des kleinen Einmaleins des Dezimalsystems deutlich mehr Platz beansprucht und erheblich höhere Kosten verursacht als das kleine Einmaleins des Binärsystems.

Auch wenn die eigentlichen Berechnungen im Binärsystem etwas länger werden, weil die Zahlen mehr Stellen haben: Computer arbeiten auch heute noch mit dem Binärsystem, weil es so einfach und deshalb kostengünstig und effektiv zu implementieren ist.



Das Binärsystem wurde übrigens bereits 1697 von Gottfried Wilhelm Leibniz vorgestellt. Damals war jedoch eine direkte Anwendung noch nicht denkbar und so leitete Leibniz kurzerhand von der Existenz eines so elementaren Zahlensystems einen Beweis für die Erschaffung der Welt aus dem Nichts ab.

Von Zuse zur Univac

Erst über 60 Jahre nach dem Tod Babbages wandte sich wiederum ein Einzelkämpfer der Idee einer universellen Rechenmaschine zu: Konrad Zuse baute mit Hilfe seiner Eltern und Freunde im Wohnzimmer seinen ersten Computer, die Z1. Diesen stellte er 1938 auf Basis von mechanischen Schaltern (später Relais) fertig. Als Bauingenieur musste er sehr viele Berechnungen von Hand durchführen – zum Beispiel um die Auslegung von Brückentragwerken zu bestimmen. So meinte Zuse später zu seiner Erfindung: „Ich war zu faul zum Rechnen und erfand den Computer."

Auch diese Maschine wurde wiederum von der Trennung zwischen Speicher und Rechenwerk bestimmt. Entscheidend war jedoch die Idee, das bisher für Rechenapparate verwendete Dezimalsystem durch das Binärsystem zu ersetzen.

Bald nach Fertigstellung der Z1 brach allerdings der Zweite Weltkrieg aus. Anfangs wurde Zuses Computerentwicklung durch die Wehrmacht gefördert. So hatte er bis 1941 einen brauchbaren Rechner gebaut, der allerdings etwa fünf Sekunden für eine Addition benötigte. Unzufrieden mit der Geschwindigkeit, wollte er nun einen deutlich schnelleren Computer entwickeln. Sein Student Helmut Schreyer schlug zu diesem Zweck vor, die Relais durch Röhren zu ersetzen. Damit wäre eine Steigerung der Geschwindigkeit um den Faktor 1000 möglich gewesen.

Die Verwirklichung dieser bahnbrechenden Idee fiel in Deutschland allerdings dem Kriegswahn zum Opfer: Zuse veranschlagte für die Konstruktion und den Bau eines Nachfolgecomputers zwei Jahre. Die deutschen Machthaber gingen aber davon aus,

Konrad Zuse (1910 – 1995) gilt als Erfinder des ersten funktionsfähigen digitalen Computers.

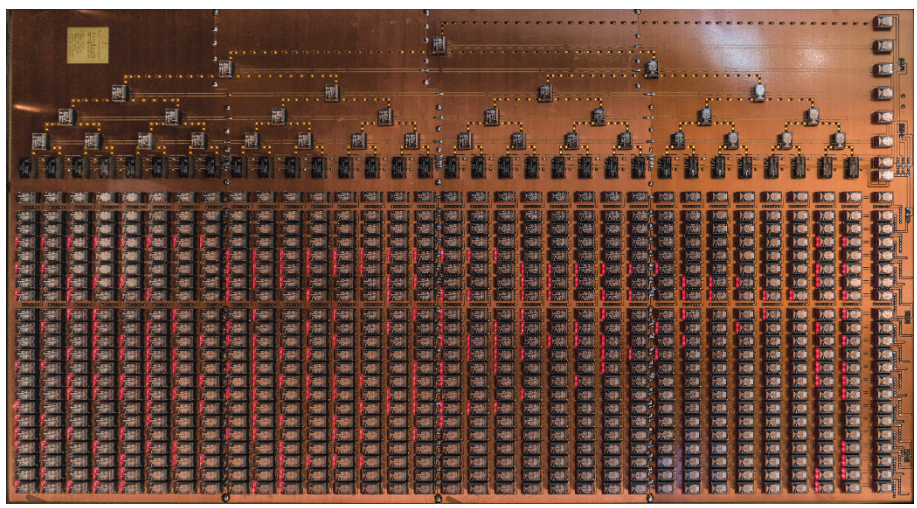


Abbildung 5.7
Nachbau des Speicherwerks der Zuse Z3 mit modernen Relais durch Raúl Rojas, FU Berlin

John Presper Eckert (1919–1995, oberes Bild), und John William Mauchly (1907–1980, unteres Bild) haben mit der ENIAC nicht nur einen der ersten Computer der Welt entwickelt, sondern waren auch Pioniere dabei, diese neue Technologie von der reinen militärischen und wissenschaftlichen Anwendung in die Wirtschaft zu transferieren: 1951 verkauften sie den ersten Computer, die UNIVAC, an das US-amerikanische Büro für Volkszählungen.

den Krieg viel schneller zu gewinnen, was das Projekt militärisch uninteressant machte. Für zivile Ziele hatte man kein Geld. Daher musste Zuse seine führende Rolle in der IT bald abtreten.

An der Universität Pennsylvania in den USA wurde 1942 das Projekt ENIAC begonnen. Von der Armee finanziert (hier erkannte man auch das militärische Potential), entstand ein Computer auf Basis von Elektronenröhren, wie sie in Radios und Verstärkern eingesetzt wurden. Den Krieg konnte ENIAC nicht mehr entscheiden, denn die Fertigstellung erfolgte erst 1946.

Aufgrund von Patentstreitigkeiten mit der Universität wurden die Erfinder J. Presper Eckert und John W. Mauchly aber zu einem Schritt gezwungen, der die Computertechnik weiter revolutionierte: Sie mussten noch 1946 den wissenschaftlichen Betrieb verlassen und das Konzept in ein wirtschaftlich ertragreiches Produkt umsetzen: Die UNIVAC (Universal Automatic Computer) wurde geboren und 1951 an das US-amerikanische Büro für Volkszählungen ausgeliefert.

Davon alarmiert, stieg jetzt auch IBM in das Geschäft mit der universellen Maschine ein. Vorher stellten sie hauptsächlich Büromaschinen her, die mit mechanischen Lochkarten arbeiteten. Insbesondere dank einer Armee von ausgezeichnet geschulten und motivierten Handelsvertretern konnte IBM trotz leistungsmäßig unterlegener Technologie bald Marktführer werden.

Der Computer als Massenware

Damit fing eine neue Revolution an: Bisher war der Computer ein wissenschaftliches oder militärisches Gerät gewesen – es gab nicht viele auf der Welt. Amerikanische Wirtschaftsprofessoren sagten noch 1948 voraus, eine Handvoll Computer reiche aus, um den Bedarf zu decken: sechs in Amerika und drei in Europa.

Plötzlich änderte sich alles: Viele Firmen sahen das unglaubliche Potential der neuen Technik und bestellten ihre eigenen Computer. IBM verkaufte allein im ersten Jahr 1951 über tausend davon. Das führte jedoch zur Verknappung eines ganz anderen „Rohstoffs“ ...

Computer arbeiten mit einer ganz eigenen Sprache, die im Wesentlichen Befehle enthält wie:

- Lese die Speicherstelle 394 aus
- Addiere Speicherstelle 395
- Schreibe das Ergebnis in Speicherstelle 396
- Kopiere den Inhalt der Speicherstelle 390 nach 394
- Fahre mit dem Programm bei Speicherstelle 119 fort

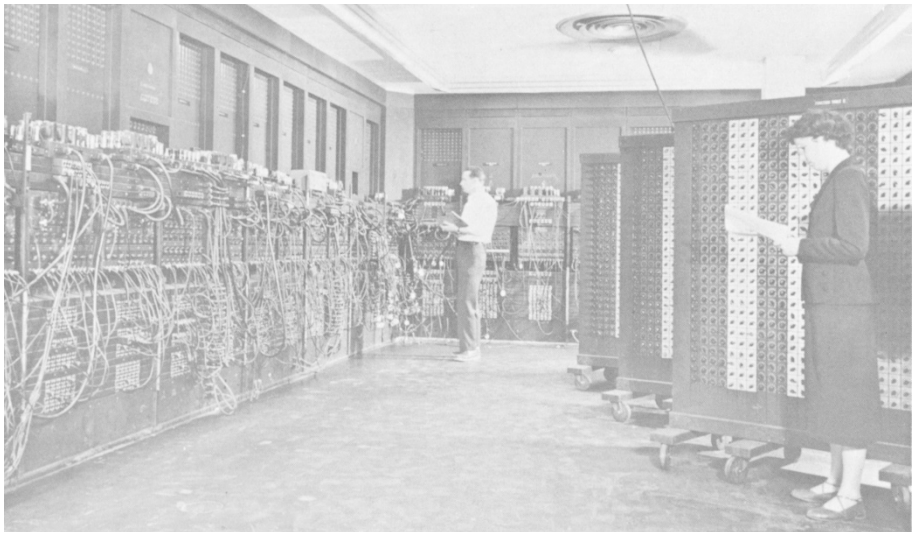


Abbildung 5.8
Die ENIAC (Electronic Numerical Integrator And Computer) hatte über 18.000 Elektronenröhren, was alleine schon in der Wartung eine große Herausforderung darstellte. Sie benötigte die Energie eines kleinen Kraftwerks und die Fläche einer Drei-Zimmer-Wohnung und konnte immerhin 5000 Additionen in der Sekunde durchführen.

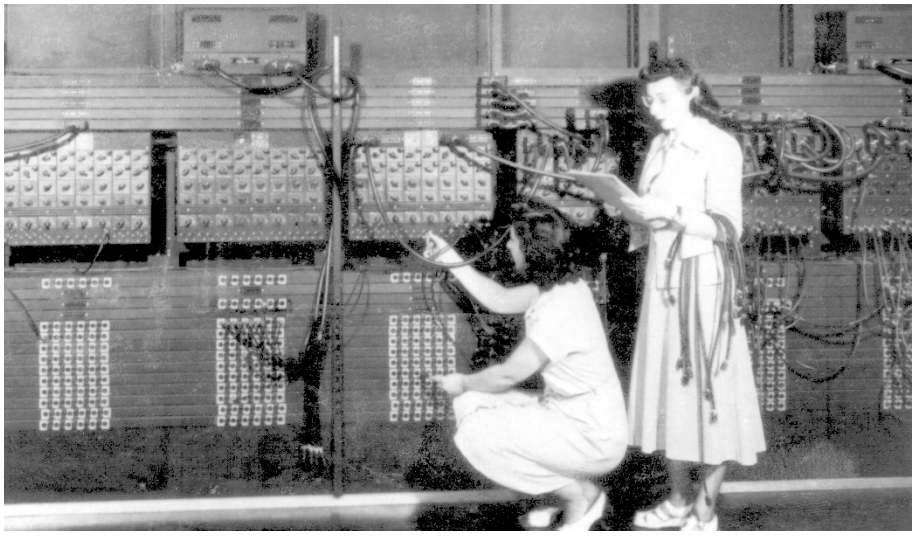


Abbildung 5.9
Wissenschaftlerinnen der Universität Pennsylvania beim „Programmieren" der ENIAC

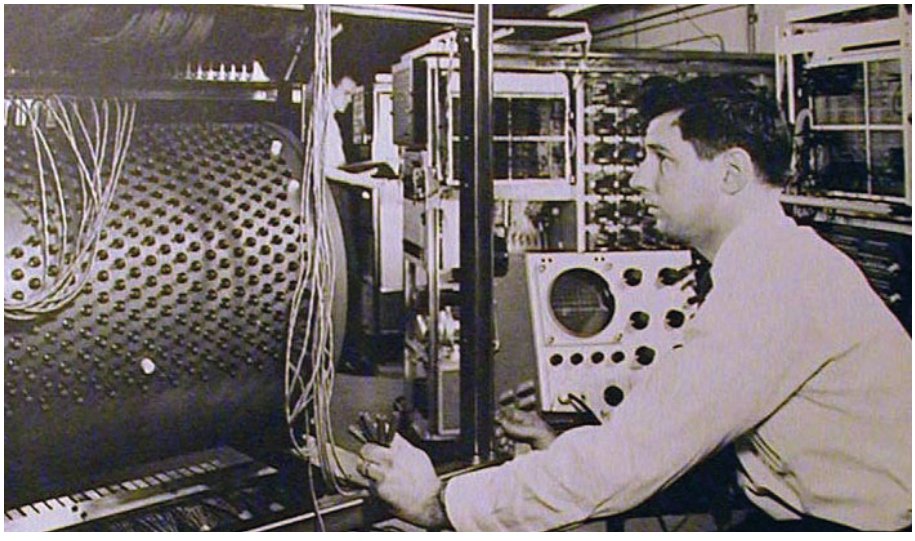


Abbildung 5.10
Teil einer UNIVAC in einer Werbeanzeige von 1956

Software
Der Begriff „Software", also direkt übersetzt „weiche Ware" oder „Computerprogramm", ist übrigens auch hochoffiziell vom Deutschen Institut für Normung definiert worden. Unter DIN 44300 als „Gesamtheit oder Teil der Programme für Rechensysteme, wobei die Programme zusammen mit den Eigenschaften der Rechensysteme den Betrieb der Rechensysteme, die Nutzung der Rechensysteme zur Lösung gestellter Aufgaben oder zusätzliche Betriebs- und Anwendungsarten der Rechensysteme ermöglichen."

Tatsächlich handelt es sich hier freilich nicht um richtige Befehle in einer uns verständlichen Sprache, sondern um Zahlen im binären System (hier als Beispiel schon in Gruppen angeordnet):

- 10001000 00000001 10001010
- 11001001 00000001 10001011
- 10001001 00000001 10001100
- 00111010 00000001 10000110 00000100
- 00000101 01110111

Die Umsetzung in den sogenannten Hex-Code (Zahlen im 16er-System) und die Einführung der Mnemonics – natürlichsprachliche Abkürzungen für die Befehle – waren erste Schritte, das Erlernen dieser Sprache und damit das Schreiben von Programmen zu erleichtern. Da die Entwicklung hauptsächlich in den USA betrieben wurde, beziehen sich die Abkürzungen selbstverständlich auf die englische Sprache.

- RD 018A
- ADD 018B
- WRI 018C
- CPY 0186 +4
- JMP 77

Menschen, die hier durchblickten, waren rar und auch in anderen – abwechslungsreicheren und besser bezahlten – Berufen hoch geschätzt. Daher kostete die Entwicklung der Software bald das Drei- bis Vierfache der (bereits sehr teuren) Maschinen. Die Computerrevolution wurde ausgebremst durch eine Knappheit an Programmierern.

John Warner Backus (1924–2007) war als einer der ersten Informatiker der Kopf des Teams, das die Programmiersprache Fortran und damit die erste höhere Programmiersprache überhaupt entwickelt hat. Später beschäftigte er sich dann mit dem Konzept „Sprache" allgemeiner und entwickelte zusammen mit Peter Naur eine formale Beschreibungssprache für Sprachen – die Backus-Naur-Form.

Was macht man, wenn es nicht genug Leute gibt, die bereit sind, mit den Rechnern im Binärcode (oder ähnlichem „Kauderwelsch") zu sprechen? Man versucht dem Rechner eine Sprache beizubringen, die näher an der normalen Sprache der Menschen ist.

FORTRAN wurde geboren. Es ist die Abkürzung für „Formular Translation" und wurde besonders gerne von Mathematikern und Ingenieuren genutzt. Auch in dieser Sprache bestanden Programme aus einem Ablaufplan mit vielen Sprüngen und Verzweigungen, aber technische Formeln und Gleichungen konnten weitgehend als solche beschrieben werden.

Das folgende Beispiel ist aus einem Original-Handbuch von 1956 entnommen („Fortran 1 Programmer's Reference Manual" von IBM). Es liest Zahlen ein und gibt die größte davon aus. Auch wenn man nie gelernt hat, FORTRAN zu programmieren, kann man die Struktur und den Sinn doch recht leicht erkennen.

```
   DIMENSION A(999)
   FREQUENCY 30(2,1,10), 5(100) READ 1, N, (A(I), I*1, N)
 1 FORMAT (I3/(12F6.2)) BIGA = A(1)
 5 DO 20 I = 2, N
30 IF (BIGA A(1)) 10,20,30
10 BIGA = A(1)
20 CONTINUE PRINT 2, BIGA
 2 FORMAT (23H DIE GROESSTE ZAHL IST F7.2) STOP 7777
```

Selbstverständlich wünschten sich nun auch Banker und Geschäftsleute, dass der Computer ihre Sprache der Wirtschaft versteht. COBOL wurde aus der Taufe geho-

ben, die Abkürzung steht für „Common Business Oriented Language“. Der Sprachstandard wurde 1960 verabschiedet.

Bei diesen höheren Programmiersprachen übersetzt der Computer die Symbole erst einmal in einen für ihn verständlichen Binärcode und führt ihn dann aus.

Das Sprachproblem war gelöst und auch ein neues elektronisches Bauteil leistete der florierenden Computerindustrie weiteren Vorschub: der Transistor. Während Röhren störungsanfällig waren, häufig ausgetauscht werden mussten und sehr viel Energie verbrauchten, tat der neue Baustein auf Basis von Silizium den gleichen Job deutlich sparsamer, effizienter und zuverlässiger. Außerdem sind Transistoren sehr viel kleiner. Der erste Computer auf Transistor-Basis wurde 1956 verkauft.

Grace Brewster Murray Hopper (1906–1992) war als Mathematikerin und Physikerin nicht nur eine der ersten Computeranwenderinnen der Welt, sondern leistete auch wesentliche Vorarbeiten zur Programmiersprache Cobol

Atlas und die Alptraum-Kabel ...

Diese technologische Revolution ließ die Ingenieure noch kleinere und vor allem sehr viel komplexere Maschinen bauen. Schnell stieß man an die nächste Grenze: Alle verwendeten Bauteile mussten irgendwie verkabelt werden. Der in Manchester gebaute „Atlas I“ enthielt auf zehn „Kleiderschränke“ verteilt ca. 80.000 Transistoren und 300.000 Dioden. Immerhin konnte er bereits 1.000.000 Befehle pro Sekunde ausführen, aber die Verkabelung der Bauteile und damit auch die Wartung bei Störungen erwies sich als ein einziger Alptraum. So war mit der bestehenden Technologie erst einmal kein größerer Computer möglich.

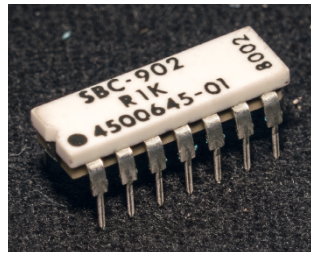
Es gab jedoch bereits einen Ausweg: 1959 waren die ersten integrierten Schaltkreise verfügbar. Beim Urahnen der heutigen Computerchips wurden Transistoren und die verbindenden Leiterbahnen gemeinsam durch ein fotochemisches Verfahren hergestellt – und das mit einem der billigsten Rohmaterialien: Silizium, das leicht aus einfachem Sand gewonnen wird.

Niklaus Wirth (geb. 1934) ist der erste deutschsprachige Turing-Preisträger. Er entwickelte mehrere Programmiersprachen, von denen PASCAL die bekannteste ist, die besonders in den 1990er-Jahren aufgrund ihrer konsequenten Struktur gerne in der Informatik-Ausbildung eingesetzt wurde.

Interessanterweise zeigten die Computerfirmen jedoch lange Zeit ziemlich wenig Interesse an der Erfindung: Aufgrund des geringen Absatzes waren die neuen Schaltkreise trotz geringer Rohstoffpreise einfach zu teuer. Das änderte sich erst 1962, also John F. Kennedy der Welt verkündete, dass die USA bis zum Ende des Jahrzehnts Menschen zum Mond und sicher zurückfliegen wollten.

Der NASA war klar, dass man zum Manövrieren im Weltraum einen Computer benötigte. Zur Landung auf dem Mond muss man den Erdtrabanten erst einmal umkreisen. Auf der abgewandten Seite besteht dann allerdings keine Funkverbindung. Somit musste ein entsprechend leistungsstarker Computer auch an Bord sein, nicht nur in der Bodenstation.

Ein Atlas wog über 20 Tonnen – viel zu schwer für eine Rakete, die kaum ihre drei Astronauten tragen konnte. Die einzige Lösung war die Verwendung integrierter Schaltkreise. Die nächste technologische Revolution war damit vorprogrammiert: Da die Chips nun in hohen Stückzahlen gekauft wurden, konnten sie günstiger produziert werden und bald gab es kaum noch Computer ohne die „kleinen schwarzen oder weißen Käfer“.



Ein integrierter Schaltkreis

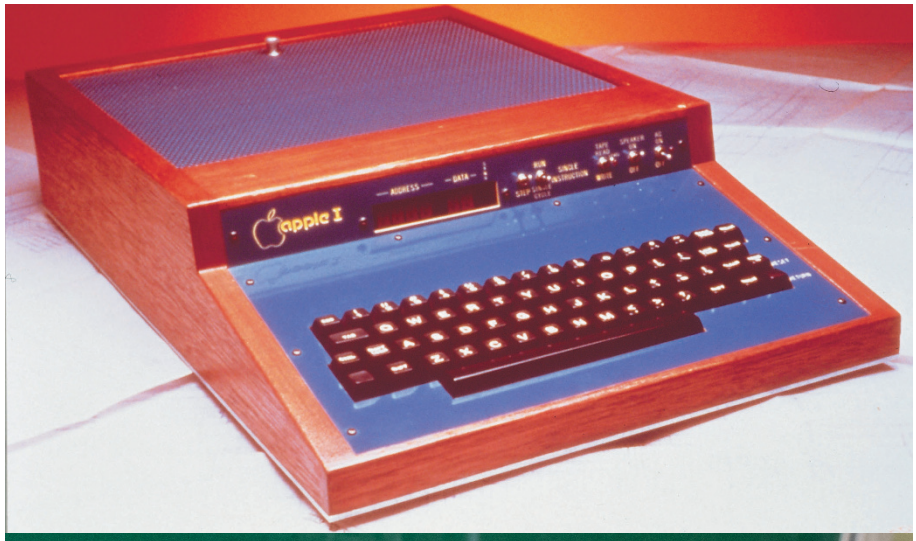
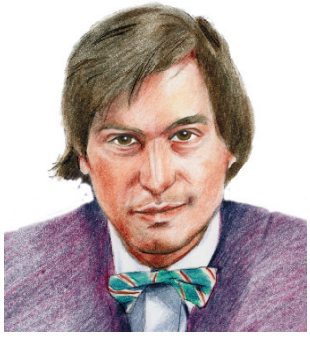


Abbildung 5.11
Der erste Apple-Computer mit einem in der heimischen Garage handgefertigten Gehäuse (oben), aber bereits ähnlich zum unten abgebildeten typischen Design, das über zehn Jahre lang sehr erfolgreich verkauft wurde.



Steve Jobs (1955–2011, oben) und Stephan Gary Wozniak (*1950, unten) träumten von einem eigenen Computer in einer Zeit, als Computer noch professionelle, teure Großgeräte waren. Sie brachten mit ihrem Apple die PC-Revolution in Gang.

Die PC-Revolution

Die kostengünstige Herstellung leistungsfähiger und kleiner elektronischer Schaltkreise ermöglichte dann auch die Verwirklichung eines Traums, den viele junge Leute in den 1960er-Jahren träumten: irgendwann einmal einen eigenen Computer zu besitzen. Unter ihnen waren auch Steven Jobs und Steve Wozniak. Ihre 1976 gegründete Firma Apple produzierte den ersten ernst zu nehmenden Computer für Privatleute und kleine Firmen. Der Apple II verkaufte sich in den folgenden Jahren millionenfach.

Erst 1981 stieg IBM dann auch in das Geschäft mit den kleinen Rechnern ein und brachte den ersten PC auf dem Markt. Das Ende dieser Revolution ist noch nicht erreicht – auch heute werden immer kleinere Computer mit immer mehr Leistung für immer weniger Geld verkauft.

Abbildung 5.12
Die erste Maus mit Holzgehäuse

Mensch und Computer

Eine Revolution ganz anderer Art begann ebenfalls in den 1970ern: Immer mehr Menschen fragten sich, ob Computer wirklich die hochkomplizierten, nur von speziell geschultem Personal bedienbaren Maschinen bleiben müssen. Für sie war das vornehmliche Ziel, den Rechner menschlicher zu machen. Das bedeutet, dass er sich nicht unbedingt menschlich verhalten muss, sich aber den menschlichen Denk- und Kommunikationsgewohnheiten anpasst.

Nehmen wir einmal die heute alltägliche Computer-Maus. Wussten Sie, dass das erste Exemplar dieser Gattung bereits 1963 das Licht der Welt erblickte? Douglas C. Engelbart entwickelte sie damals an der Universität Stanford. Sie hatte keine Kugel und auch nicht die heute übliche optische Abtastung der Tischoberfläche, sondern zwei separate Räder an der Unterseite.

Die heute noch übliche Form der Maus mit Kugel wurde dann in den frühen 1970ern am Xerox PARC (Palo Alto Research Center) entwickelt. Ihr zugrunde liegt die Beobachtung, dass Menschen im Alltag Dinge anfassen, bewegen und an anderer Stelle wieder ablegen. Wenn der Computer menschlicher werden soll, muss er Fähigkeiten ausnutzen, die der Bediener bereits besitzt, und so wurden in der ersten GUI (Graphical User Interface = graphische Bedienoberfläche) genau diese Fertigkeiten auch zum Manipulieren von Bildern, Dateien und Graphik im Computer herangezogen: Man klickt (greift) einen Gegenstand (z. B. eine Datei), hält diesen dann fest (Maustaste bleibt gedrückt), verschiebt ihn (Mausbewegung) und lässt ihn am Ziel fallen (Maustaste loslassen).

Letztlich waren das die Anfänge der Bedienoberfläche des viel verkauften Computers Apple Macintosh, später wurde das Konzept dann auch von Microsoft für Windows übernommen. Diese uns vertraute Arbeitsumgebung auf dem Rechner hat also bereits eine Geschichte von einem halben Jahrhundert. Heute ist „Human Computer Interaction“ (deutsch „Mensch-Maschine-Kommunikation“) eine eigene Fachdisziplin in der Wissenschaft Informatik. Hier versucht man, den Computer noch besser an die Bedürfnisse des Menschen anzupassen.

Douglas Carl Engelbart (1925–2013) war Elektrotechniker und kümmerte sich zeitlebens besonders intensiv um sogenannte Mensch-Maschine-Schnittstellen (auch wenn der Begriff erst später geprägt wurde). Neben entscheidenden Ideen zu graphischen Bedienoberflächen (GUI = Graphical User Interface) hatte er auch die Idee der Eingabe durch natürliche Handbewegungen, für die er eine erste Maus erfand.

Resümee

Es könnten hier noch unzählige Geschichten mehr erzählt werden: vom Turing-Test, mit dem man herausfinden möchte, ob Computer eine Art „Intelligenz“ entwickeln können, von neuronalen Netzen, die das menschliche Denken simulieren sollten, von riesigen Rechnern und ganz kleinen, von Ansätzen, Chips nicht mehr herzustellen, sondern als eine Art Biorechner anzubauen, und so weiter und so weiter ...

Hier historisch akkurat zu berichten, würde den Rahmen dieses Buchs sprengen, das ja für das Mitmachen begeistern möchte. Daher sind in den folgenden Mitmach-Kapiteln kleine historische Querbezüge enthalten. Oft stehen diese in der Randspalte. Falls Sie diese bisher noch nicht beachtet haben: Schauen Sie doch einmal hinein.

Für dieses Kapitel war mir wichtig, zu zeigen, dass die Informatik bereits eine recht lange und illustre Geschichte aufweist. Viele der heute noch verwendeten Verfahren wie etwa die binäre Multiplikation haben ihren Ursprung tausende von Jahren in der Vergangenheit. Informatik ist eine Disziplin menschlicher Kreativität und menschlichen Denkens!

Sie wissen nun auch, dass viele Situationen der letzten Jahre sich bereits schon einmal ähnlich ereignet haben: Immer wieder gab es Grenzen bei der Fortentwicklung von Hardware und Software und immer wieder wurden die scheinbaren Barrieren durch neue Techniken oder durch kreativen Einsatz der vorhandenen Möglichkeiten durchbrochen. Und das ist doch eigentlich das Spannende an einer Wissenschaft: Es gibt immer die Möglichkeit, dass Sie etwas ganz Neues entdecken. Daher: Entdecken Sie doch gleich mit dem nächsten Kapitel die Informatik für sich neu!

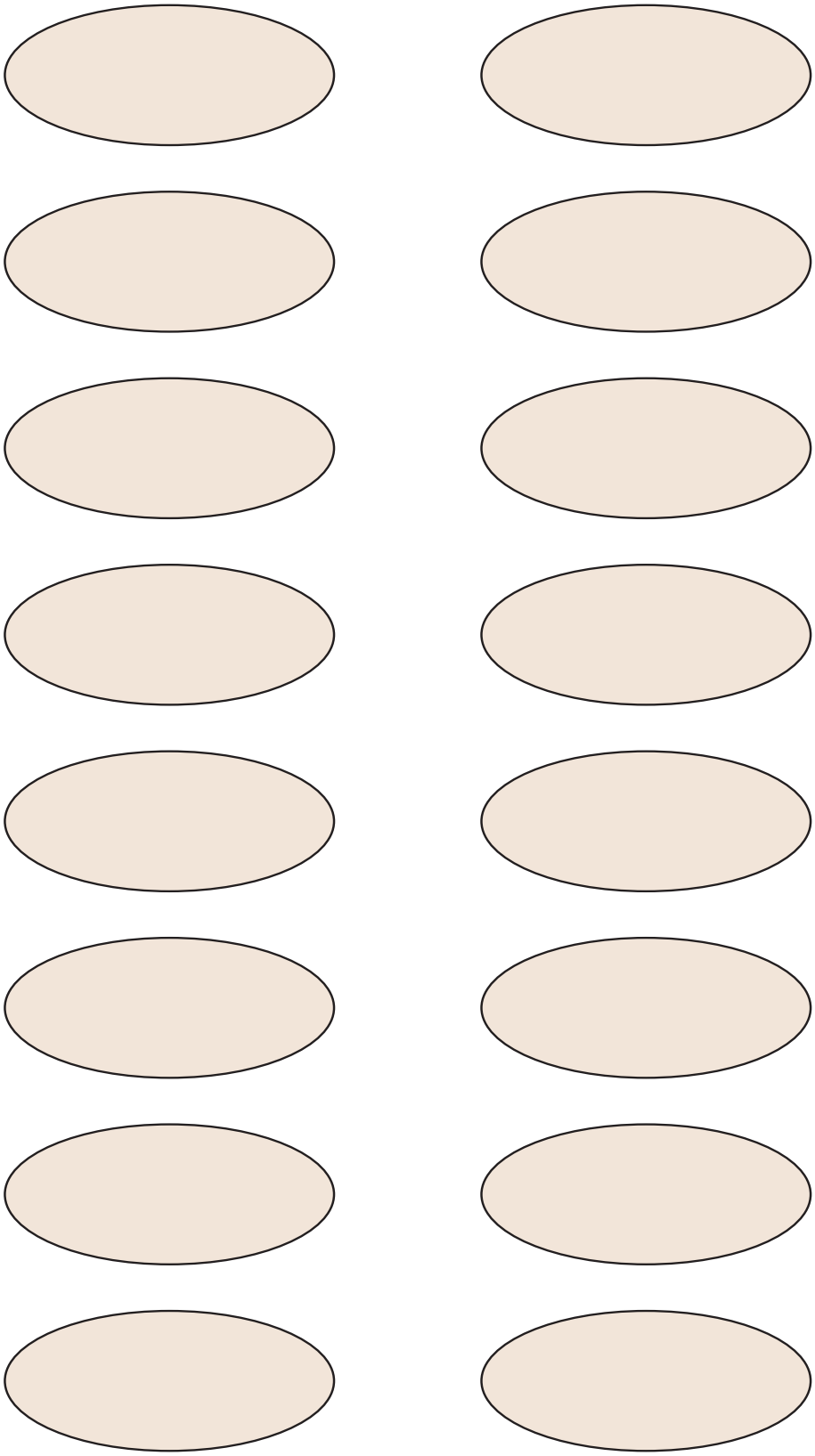
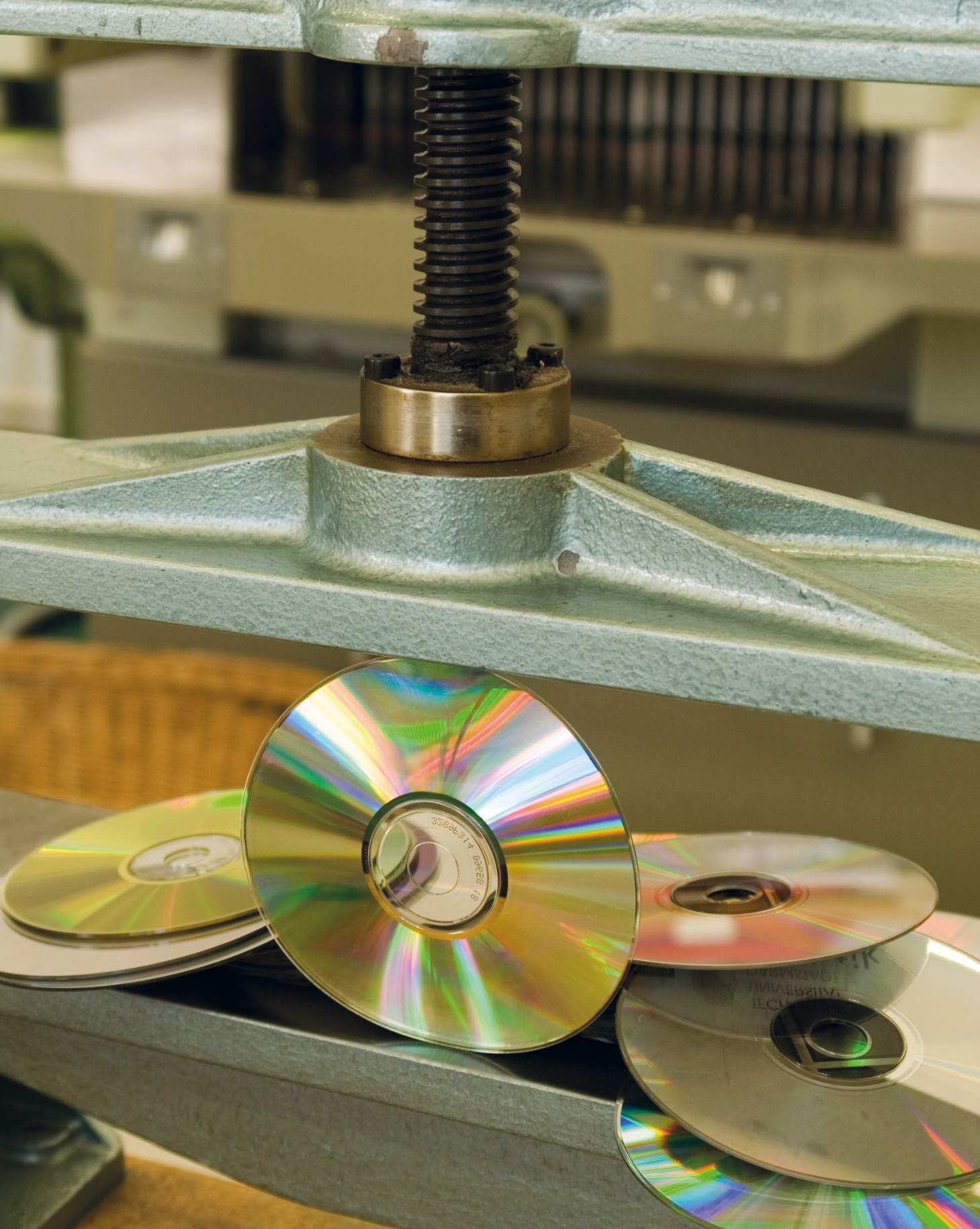


Abbildung 5.K1
Ersatz für Löcher im Parkett, wenn Sie die Äthiopische Multiplikation zuhause nachstellen möchten



6. Von Kamelen und dem Nadelöhr

In der Bibel sagt Jesus: „Es ist leichter, dass ein Kamel durch ein Nadelöhr gehe, denn dass ein Reicher ins Reich Gottes komme." Die Redensart ist damit eine Analogie für etwas Unmögliches geworden.

Für alle Reichen, die bisher vergeblich an die Himmelspforte klopfen mussten, kommt heute jedoch Hilfe aus der Informatik: Moderne Codierung lässt Daten (zum Beispiel Bilder von Kamelen) auf Nadelöhrgröße schrumpfen.

Dieses Kapitel erklärt, wie das überhaupt funktionieren kann und wo die Möglichkeiten sowie ihre Grenzen liegen.

Buchstaben und Leitungen

Stellen Sie sich vor, Sie haben nur die Deckenbeleuchtung nebst Lichtschalter zur Verfügung, um Ihrem Nachbarn die neuesten Erkenntnisse der Informationstechnik mitzuteilen. Hier wird das elementare Grundproblem der Codierung sichtbar: Menschen und Computer sprechen eine unterschiedliche Sprache!

Während wir in komplexen Symbolen kommunizieren, kennt ein Computer nur „An" und „Aus" – aus gutem Grund, wie wir bereits im letzten Kapitel klären konnten.

Zählt man hingegen unser Alphabet mit Ziffern und gebräuchlichen Satzzeichen zusammen, kommt man etwa auf 48 Symbole. Unterscheidet man zusätzlich Groß- und Kleinschrift sowie Umlaute, werden es noch deutlich mehr. Es gilt nun, diese Zeichen in einer Sprache auszudrücken, die nur zwei Zustände kennt. Der Einfachheit halber drücken wir diese wiederum mit den Symbolen „0" und „1" aus.

Codierung

Verfahren, die Symbole einer Nachricht in eine andere Form zu wandeln, ohne den Informationsgehalt der Nachricht einzuschränken. Die Codierung impliziert, dass eine entsprechende Decodierung möglich ist, um die Nachricht wieder in den Originalzustand zu versetzen.

Codierung bedeutet in der Informationstechnik meistens die Umwandlung von der für den Menschen verständlichen Form in eine für Maschinen verarbeitbare und über Netzwerke kommunizierbare Version.

Selbstverständlich funktioniert diese Umsetzung nicht eins zu eins – Wenn wir für ein Zeichen des menschlichen Alphabets nur ein Zeichen der Computersprache zur Verfügung hätten, müssten wir z. B. die Buchstaben A–M durch eine „0", die Buchstaben N–Z durch eine „1" ausdrücken.

Zeichen	Blockcode
_ (leer)	00000
A	00001
B	00010
C	00011
D	00100
E	00101
F	00110
G	00111
H	01000
I	01001
J	01010
K	01011
L	01100
M	01101
N	01110
O	01111
P	10000
Q	10001
R	10010
S	10011
T	10100
U	10101
V	10110
W	10111
X	11000
Y	11001
Z	11010
Ä	11011
Ö	11100
Ü	11101
ß	11110
.	11111

Versuchen Sie nach diesem Schema die folgende Nachricht zu decodieren: 0101100100

Die ursprüngliche Nachricht sollte „INFORMATIK“ heißen. Aufgrund der großen Spielräume bei der Codierung könnte hier aber auch ein ganz anderes Wort wie etwa „ANESTHESIA“ gemeint sein. Der Arbeitsauftrag war also nicht besonders sinnvoll, sondern sollte nur eine Erkenntnis bringen: Wir benötigen zur Sicherstellung der Decodierung für jeden Nachrichtenteil der Originalnachricht einen eindeutigen Code.

Bei der Umsetzung unseres Alphabets in einen Code aus den Symbolen 0 und 1, also einen Binärcode, muss daher jedes Zeichen der Ursprungsnachricht in mehrere Zeichen des Codes gewandelt werden.

Binärcode

Ein Code, der nur zwei unterschiedliche Symbole enthält. Diese werden üblicherweise durch „0“ und „1“ repräsentiert. Binärcodes sind im Computerbereich üblich, weil sie sich direkt durch eine Elektronik verarbeiten lassen, die zwei Zustände („eingeschaltet“ und „ausgeschaltet“) versteht.

Einer der bekanntesten Codes ist ASCII (American Standard Code for Information Interchange). Es handelt sich quasi um eine Tabelle, in der alle Zeichen des (amerikanischen) Alphabets entsprechenden Sequenzen aus 0 und 1 zugeordnet werden.

Eine vereinfachte Form von ASCII ist in der nebenstehenden Tabelle angegeben.

Codieren Sie zur Übung doch einfach mal einige Zeichen, zum Beispiel das Wort „INFORMATIK“. Welche Weisheit mag sich wohl hinter dem Code „01001000110100000000001000010101110010110010111100000000000101100100110111100010010011100000000100100011010000“ verbergen?

INFORMATIK lässt sich selbstverständlich in den Code „0100101110001100111110010011010000110100010010101 1“ wandeln. Sehen Sie, wie Sie diesen Code per Lichtschalter übermitteln könnten? Zum Beispiel können Sie mit Ihrem Nachbarn eine Uhrzeit ausmachen. Jede Sekunde ab dieser Uhrzeit wird ein Zeichen „gesendet“, indem das Licht entweder ausgeschaltet („0“) oder eingeschaltet („1“) ist.

Der zweite Code lässt sich in den Spruch „ICH DENKE, ALSO BIN ICH“ decodieren. Oft wird von Codekonstrukteuren das Leerzeichen vergessen (hier „00000“). Dies ist jedoch für unsere Sprachverständigung sehr wichtig und hat daher den gleichen Stellenwert wie jedes Zeichen des Alphabets auch!

Die Decodierung mit Tabelle ist etwas aufwendig, daher gibt es für den Code auch eine etwas andere Darstellung, die man „Codebaum“ nennt. Abbildung 6.1 zeigt diese Darstellung für den bereits verwendeten Code.

Wenn alle Codes die gleiche Länge haben, diese also in der Tabelle oder im Codebaum einen optischen „Block“ bilden, nennt man sie „Blockcode“.

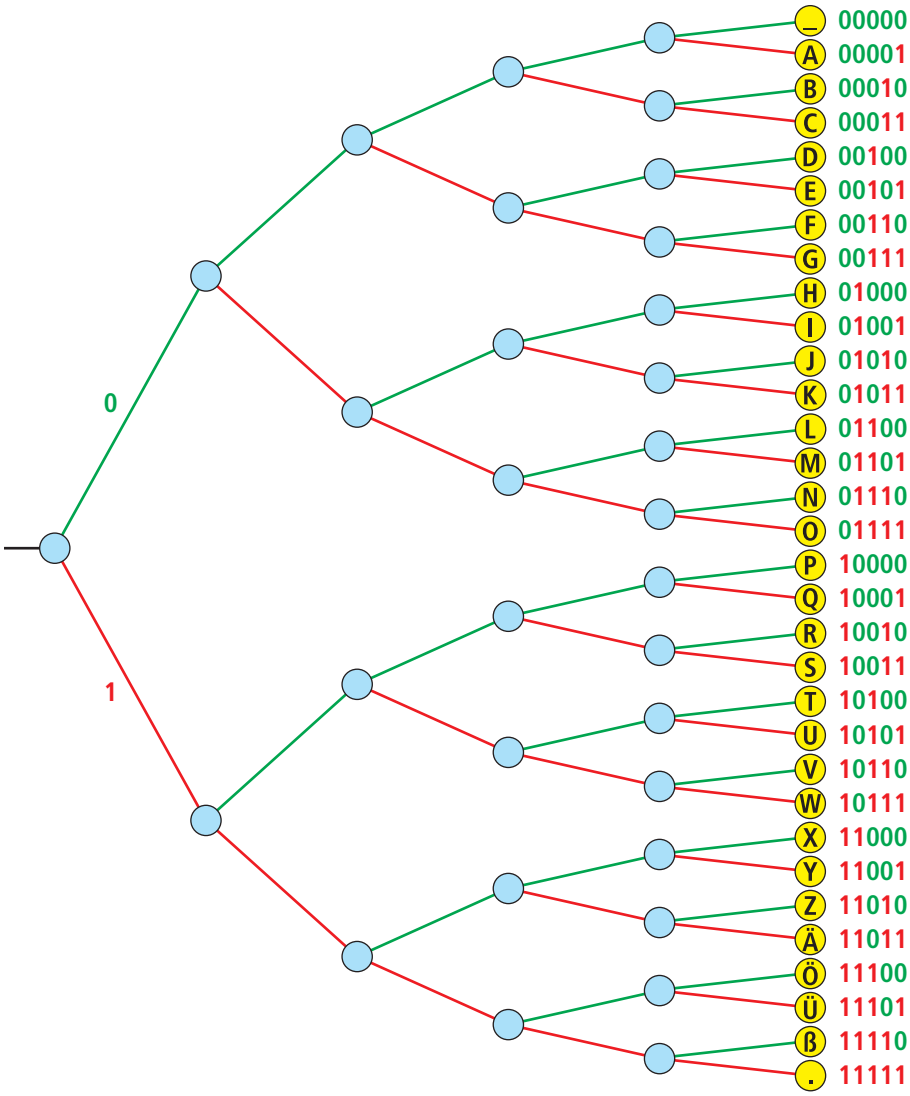


Abbildung 6.1
Codebaum für 32 Zeichen

Wenn Sie nun einen Code in Klartext wandeln möchten, fangen Sie immer links an und gehen Sie – je nachdem, ob das erste Codezeichen eine 0 oder eine 1 ist – nach oben oder unten. In der Abbildung 6.1 wird das zusätzlich durch die grüne Farbe für 0 und die rote Farbe für 1 verdeutlicht. Das machen Sie mit dem nächsten Zeichen an der nächsten „Abzweigung" genauso usw., bis Sie zu einem Buchstabenfeld gelangen. Diesen können Sie nun hinschreiben – er ist entziffert. Mit dem nächsten Zeichen fangen Sie nun wieder ganz links an.

Decodieren Sie die Nachricht „10111101010111000100001011001000010000011 0010" mit dem Codebaum!

Code
Der Begriff kommt übrigens vom lateinischen „codex", was so viel wie „abgeschlagener oder gespaltener Baum" bedeutet. Das war das Grundmaterial für Schreibtafeln. Von den Schreibtafeln wurde der Begriff dann auf Bücher übertragen, unter anderem auch auf die Codebücher, die zum Verschicken verschlüsselter Botschaften notwendig waren ...

Bits und Bytes
Ein Byte (engl. „Bissen") ist sozusagen ein Happen aus dem Datensalat eines Computers. Es stellt eine kleine Informationseinheit dar – normalerweise exakt so groß, um eine Zahl zwischen 0 und 255 zu beinhalten. Bit ist englisch als Verkleinerungsform von Byte zu sehen und – als BIT normalerweise in Großbuchstaben geschrieben – auch die Abkürzung für „Binary Digit", also deutsch „Ziffer im Binärsystem".

Hier kommt „WUNDERBAR" heraus.

Beachten Sie bitte, dass der von uns verwendete Code für alle Zeichen die gleiche Länge hat. Wir können daher ganz leicht ausrechnen, wie viele Bits ein Code für eine Nachricht mit 100 Zeichen hat: 100 Zeichen mal 5 Bit pro Zeichen = 500 Bit.

Können Sie selbst einen Binärcode konstruieren?

Nichts leichter als das! Für jedes Zeichen des Alphabets muss eindeutig eine Folge von Bits angegeben werden. Das wollen wir doch gleich einmal üben:

Ein Genforscher bittet Sie, einen Binärcode für Gensequenzen zu entwickeln. Gensequenzen werden im Allgemeinen durch Ketten aus nur vier verschiedenen Zeichen A C G T dargestellt.

Eine solche Gensequenz ist also zum Beispiel „AGATGCCGTTACGA".

Helfen Sie dem Genforscher und entwickeln Sie einen entsprechenden Code! Codieren Sie auch gleich die angegebene Gensequenz.

Eine mögliche Lösung ist sicherlich, einfach den Code von oben zu benutzen, denn die Zeichen A, C, G und T kommen in ihm ja auch vor. Die Gensequenz würde dann als „00001001110000110100001110001100011001111010010100000010001100111100001" codiert.

Wie schon mehrfach in diesem Buch gezeigt, ist das sogar eine Möglichkeit, die Informatiker sehr gerne wählen, denn man braucht nichts neu zu entwickeln, sondern verwendet nur bereits bekannte Verfahren in einem neuen Zusammenhang. Allerdings wäre das sehr verschwenderisch und auch Informatiker versuchen, eine möglichst optimale Lösung zu finden.

Bevor ich darauf eingehe, was eigentlich verschwenderisch an der Lösung ist, möchte ich einen anderen Code vorstellen, auf den Sie wahrscheinlich ebenfalls gekommen sind (bzw. auf einen ähnlichen). Abbildung 6.2 zeigt den entsprechenden Codebaum.

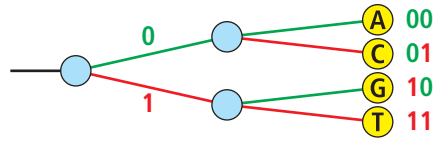


Abbildung 6.2
Codebaum für die vier Buchstaben der Genetik

Nimmt man ihn, um die Gensequenz zu codieren, kommt man auf „00100011100101101111000110000". Dies ist wesentlich kürzer, löst jedoch die Aufgabe ebenfalls: Pro Zeichen sind hier nur zwei Bit notwendig.

Vielleicht können Sie noch recherchieren, wie viele Bits Sie benötigen, um die gleiche Gensequenz im bereits erwähnten ASCII zu codieren.

Da ASCII 8 Bit pro Zeichen verwendet, kommen wir hier auf 112 Bit – also vier Mal mehr als mit unserem kleinen Codebaum. Die Frage ist nun, warum die Anzahl der Bits pro Zeichen in unterschiedlichen Codes differiert.

Von mehr und weniger gehaltvollen Buchstaben

Hier kommt der Begriff „Information“ ins Spiel: Jedes Zeichen eines Alphabets trägt in sich eine Information – aber wie viel ist das? Kann man diese irgendwie messen? Wir wollen versuchen, uns der Antwort auf diese Frage zu nähern, indem wir das hauptsächliche Werkzeug des täglichen Informationsaustauschs näher betrachten: die Sprache!

„Alle Dinge der Welt passen in 26 Zeichen“, sagt eine Redensart. Die deutsche Sprache kennt sogar 30 verschiedene Buchstaben (mit Umlauten). Um alle Dinge der Welt damit auszudrücken, muss jeder Begriff aus mehreren Buchstaben zusammengesetzt werden. „LIEBE“ besteht zum Beispiel aus fünf solcher Buchstaben. In China gibt es viel mehr Schriftzeichen. Dafür kann ein Begriff wie „LIEBE“ mit einem einzigen Schriftzeichen dargestellt werden. Offenbar ist also die Information, die in einem Zeichen enthalten ist, im Chinesischen größer als im Deutschen.

Chinesisches Schriftzeichen für „Liebe“

Ein weiteres Beispiel:

Im ersten Code ganz oben gibt es keine Groß- und Kleinschreibung. Nehmen wir an, Sie erhalten mit dem Code die folgende Nachricht:

„ICH HABE LIEBE GENOSSEN“

Diese ist nicht eindeutig, denn je nach Groß- und Kleinschreibung ergeben sich völlig unterschiedliche Bedeutungen: „Ich habe liebe Genossen“ versus „Ich habe Liebe genossen“.

Beide Deutungen enthalten genauso viele Zeichen wie die Originalbotschaft, aber mehr Information. Jedes Zeichen trägt also offenbar mehr Information, wenn das Alphabet Groß- und Kleinbuchstaben enthält.

Umgekehrt enthält jedes Zeichen der Gensequenz also weniger Information, da das Alphabet ja nur aus vier Zeichen besteht. Es ist also nur konsequent, wenn im Code pro Zeichen weniger Bit gebraucht werden.

Auf diese Weise sehen wir auch, warum das Bit die kleinste Informationseinheit darstellt: Es existiert in einem Alphabet mit nur zwei Zeichen, 0 und 1. Ein Alphabet mit nur einem Zeichen würde gar keine Information mehr enthalten – denken Sie an ein Licht, das dauernd ausgeschaltet ist. Zwei Zeichen stellen also die kleinste Möglichkeit für ein sinnvolles Alphabet dar.

Somit kann die Anzahl der Bits in einem Binärcode auch als Anhaltspunkt für den Informationsgehalt genommen werden. Das setzt natürlich voraus, dass der Code sinnvoll gewählt wurde, denn die obige Gensequenz konnten wir einerseits ungeschickt mit 70 Bit oder gar 112 Bit oder geschickt durch 28 Bit ausdrücken. Als Maß für den Informationsgehalt schreibt man bit dann normalerweise komplett klein.

Offenbar entspricht der Informationsgehalt eines Zeichens im Genalphabet etwa 2 bit, während ein Zeichen des kleinen Alphabets vom Anfang des Kapitels etwa 5 bit entspricht.

Dabei ordnet man allerdings jedem Zeichen den gleichen Informationsgehalt zu. Ob dies zu Recht geschieht, versuchen wir anhand eines Experiments selbst herauszufinden.

bit
In der Informationstheorie hat sich noch eine weitere Deutung von „bit“ etabliert: Komplett klein geschrieben steht es für „basic indissoluble information unit“ – wobei das „t“ der Abkürzung etwas unkonventionell dem letzten Buchstaben zugeordnet wird. Das bedeutet auf Deutsch in etwa „unteilbare“ oder „atomare Informationseinheit“. In der Informationstheorie versucht man auf diese Weise, den echten Informationsgehalt einer Nachricht mit Zahlen auszudrücken, indem man überlegt, auf wie viele bits die Nachricht zusammengefasst werden kann, so dass die Information immer noch enthalten, aber eine weitere Reduktion nicht möglich ist.

Zeichen	Morsecode
A	· –
B	– · · ·
C	– · – ·
D	– · ·
E	·
F	· · – ·
G	– – ·
H	· · · ·
I	· ·
J	· – – –
K	– · –
L	· – · ·
M	– –
N	– ·
O	– – –
P	· – – ·
Q	– – · –
R	· – ·
S	· · ·
T	–
U	· · –
V	· · · –
W	· – –
X	– · · –
Y	– · – –
Z	– – · ·

Lesen Sie den Text in Abbildung 6.3. Es handelt sich um einen bekannten deutschen Text, dem 145 Buchstaben abhandengekommen sind. Können Sie etwas entziffern?

Verschiedentlich erkennt man ein Wort, etwa „antwortete“, aber der Sinn des Textes als Ganzes bleibt den meisten verborgen.

Versuchen Sie es daher nochmals mit dem Text aus Abbildung 6.6, die Sie nach einem Mal umblättern an genau der gleichen Stelle finden.

Auch diesem Text fehlen 145 Buchstaben. Er ist jedoch deutlich besser verständlich. Zugrunde liegt in beiden Fällen der gleiche Originaltext der Gebrüder Grimm. Falls Sie mit beiden Texten noch Schwierigkeiten haben, lesen Sie sich den Originaltext in Abbildung 6.10 durch und versuchen danach, ihn in den beiden verstümmelten Versionen wiederzufinden. Dafür müssen Sie noch einmal weiter blättern.

Wodurch kommt die unterschiedliche Lesbarkeit zustande? Der erste Text enthält lediglich die Buchstaben, die in unserer Sprache am häufigsten vorkommen: alle Vokale (A, E, I, O, U) und die Konsonanten N, R und T. Aus dem zweiten Text sind hingegen alle Vokale entfernt worden.

Wenn man unterschiedliche Buchstaben aus der Sprache weglässt, entfernt man offenbar unterschiedlich viel Informationsgehalt. Das Weglassen häufiger Buchstaben scheint nicht so schlimm zu sein wie das seltener Buchstaben.

Vielleicht ist das aber ja eine Spezialität unserer deutschen Sprache! Versuchen Sie es doch gleich nochmal auf Englisch und entziffern den Text aus Abbildung 6.4. Er enthält wiederum die häufigen Buchstaben. Wenn Sie sich dann genügend lange die Zähne daran ausgebissen haben, versuchen Sie es mit Abbildung 6.9 ein Blatt weiter.

Um den kompletten Text zu lesen, müssen Sie nochmal umblättern. Offenbar gilt das beobachtete Phänomen auch für englische Texte!

Fassen wir noch einmal konsequent zusammen: Der erste Text ist nicht lesbar – enthält also eigentlich keine oder zumindest nur sehr wenig Information für uns. Der zweite Text ist mit etwas Mühe ganz gut lesbar, enthält also deutlich mehr Informatio-

Abbildung 6.3
Bekannter deutscher Text, der nur noch häufige Buchstaben enthält (ohne Satzzeichen)

an atte ieen are ei eine errn eient a ra er u i err eine eit it eru nun ote i erne ieer ei u einer utter et ir einen on er err antortete u at ir treu un eri eient ie er ient ar o o er on ein un a i ein tüo a o roß a anen o ar an o ein Tüein au er Tae iete en uen inein ette in au ie uter un ate i au en e na au

nen. Der erste Text enthält 242 häufige Buchstaben. Wenn wir die wenige Information des Textes auf die Buchstaben verteilen, bleibt also pro Buchstabe auch nur wenig Informationsgehalt. Der zweite Text enthält 242 seltene Buchstaben, die offenbar jeweils mehr Informationsgehalt haben.

Das Experiment zeigt uns: Offenbar tragen nicht alle Buchstaben des Alphabets die gleiche Menge an Information – häufige Buchstaben haben einen kleineren Informationsgehalt als seltene. Es wäre daher nur konsequent, wenn ein Code auch nicht für jeden Buchstaben die gleiche Anzahl von Bits besäße. Unter Umständen lässt sich auf diese Weise auch Zeit bei der Übermittlung einer Nachricht sparen.

Bestes Beispiel hierfür ist das Morsen: Der häufigste Buchstabe unseres Alphabetes, das „E", wird nur als einzelner kurzer Ton gesendet, während etwa „Q" durch die deutlich größere Folge „lang lang kurz lang" repräsentiert ist. Allerdings ist das Morsen kein binärer Code, da außer den Zeichen DIT (kurz) und DAH (lang) auch noch die Länge der Pause eine wichtige Rolle spielt und Information trägt.

Die Idee, auch bei der binären Codierung die Häufigkeiten und damit den Informationsgehalt zu berücksichtigen, um Speicherplatz zu sparen, liegt aber sehr nahe.

Zunächst wollen wir aber anhand eines Beispiels ausprobieren, ob sich die Mühe denn wirklich lohnt: Bilden Sie ein paar Sätze (in deutscher Sprache) und codieren Sie diese mit Hilfe der Tabelle am Seitenrand. Lassen Sie zwischen den Bitfolgen für die Buchstaben keine Lücken – auch auf einer Datenleitung werden die Bits direkt hintereinander gesendet.

Codieren Sie nun „ABENTEUER_INFORMATIK" und decodieren Sie danach „00111100001011000101011100011000010011011111101100011111001100010111001110001001001111110101011111111011010101110001111001010101010100001001100100101000010000111101000010011100011010000101000101001010011000001101011000011101000001100100101010111101001000001011001010101011110001001011111100010100110101000000110101100001111000010101111101" mit dem Flexcode. Funktioniert das in beide Richtungen? Vergleichen Sie, welche Codelängen Sie mit dem Blockcode benötigt hätten.

Manche Menschen sind verblüfft, wenn sie merken, dass man mit dem Flexcode trotz unterschiedlicher Bitlängen der Codes für die einzelnen Zeichen problemlos codieren und decodieren kann. Bei „ABENTEUER_INFORMATIK" spart man auch schon Bits ein: Mit dem Blockcode würden die 20 Zeichen glatte 100 Bits beanspruchen, während wir mit dem Flexcode lediglich 88 brauchen.

Zeichen	Flexcode
_ (leer)	000
A	00100
B	00101
C	00110
D	00111
E	010
F	01100
G	01101
H	0111
I	1000
J	1001000
K	1001001
L	100101
M	10011
N	1010
O	101100
P	1011010
Q	1011011
R	10111
S	1100
T	1101
U	1110
V	1111000
W	1111001
X	11110100
Y	11110101
Z	1111011
Ä	11111000
Ö	11111001
Ü	11111010
ß	11111011
.	111111

hen aune as tee eas o the enhantess shut he in a toe hih a in a oest The toe ha no oo ut hih u as a ino hen the enhantess ante to o in she ae hese eo the ino an ie aune aune et on ou hai aune ha aniient on hai ine as sun o hen she hea the oie o the enhantess she oun he hai oun a hoo o the ino The hai e tent as on an the enhantess ie u it

Abbildung 6.4
Bekannter englischer Text, der nur noch häufige Buchstaben enthält (ohne Satzzeichen)

Die kryptische Bitfolge danach kann man entziffern als „DIESER SATZ WIRD KÜRZER, WENN MAN IHN MIT EINEM CODE FLEXIBLER LÄNGE CODIERT“. Mit dem Blockcode kämen wir auf eine Länge von 375 statt den hier benötigten 333 Bits.

Betrachten Sie den Flexcode, so stellen Sie fest, dass hier – wie auch beim Morsecode – seltene Buchstaben einen längeren Code haben und häufige Buchstaben wie das E dafür nur durch drei Bits repräsentiert werden. Es lohnt sich also offensichtlich wirklich, die Zeichen mit Codes unterschiedlicher Länge darzustellen.

Wie kommt man auf einen entsprechenden Code? Zeit für ein weiteres Experiment, das wir zunächst an unserem kleinen Beispiel mit den Gencodes starten. Nehmen wir an, Sie möchten folgende Gensequenz „kostengünstig“ übermitteln:

GACGTGAGGAGCGTGAGTAGGCGA

Die 24 Zeichen können mit dem Codebaum nach Abbildung 6.2 in 48 Bit codiert werden. Diesen wollen wir nun mit den Häufigkeiten der Zeichen flexibler gestalten. Dazu schneiden Sie bitte die Buchstabenplättchen aus Kopiervorlage 6.K1 oder dem Bastelbogen aus. Falls Sie daran Spaß haben, können Sie natürlich die Kreise genau nachschneiden, sonst reicht auch das umrahmende Quadrat.

Legen Sie die Buchstabenplättchen für A, C, G und T auf die markierten Startpositionen von Abbildung 6.5. Der resultierende Code entspricht genau dem Blockcode aus Abbildung 6.2. Unserem Experiment zufolge sollten ja häufigere Buchstaben durch einen kürzeren Binärcode repräsentiert werden. Welches Zeichen ist das mit Abstand häufigste in der obigen Genkombination?

Genau: G. Um einen kürzeren Code zu erhalten, schieben Sie das Plättchen einfach um einen Kreis nach links. Abbildung 6.7 zeigt das. Die Bedingung ist nun erfüllt und G hat den Code „1“. Allerdings gibt ein noch ein großes Problem: Wenn man die Nachricht „11“ empfängt, ist ungewiss, ob diese nun „GG“ oder „T“ bedeutet. Um solche Unklarheiten zu vermeiden, darf sich im Baum hinter einem korrekt erkannten Buchstaben kein weiterer Buchstabe befinden.

Nehmen Sie also das „T“ weg, um den Missstand zu beseitigen. Gibt es einen freien Platz, wo wir es platzieren könnten? Offensichtlich nicht – jede mögliche Verzweigung

Abbildung 6.5
Codebaum für Gencodes

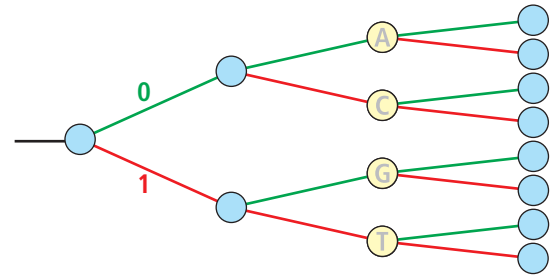


Abbildung 6.6
Bekannter deutscher Text, der nur noch seltene Buchstaben enthält (ohne Satzzeichen)

Hns htt sbn Jhr b snm Hrrn gdnt d sprch r z hm Hrr mn Zt st hrm nn wllt ch grn wdr hm z mnr Mttr gbt mr mnn Lhn Dr Hrr ntwrtt d hst mr tr nd hrlch gdnt w dr Dnst wr s sll dr Lhn sn nd gb hm n Stck Gld ds s grß ls Hnsns Kpf wr Hns zg n Tchln s dr Tsch wcklt dn Klmpn hnn stzt hn f d Schltr nd mcht sch f dn Wg nch Hs

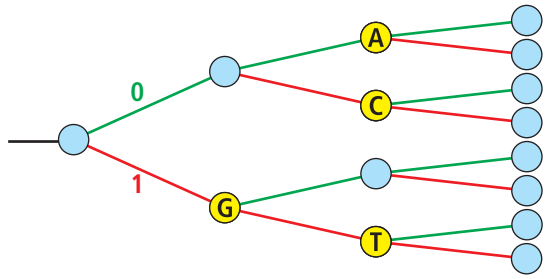


Abbildung 6.7
Liest man eine „1“ ein, weiß man mit diesem Codebaum nicht, ob man das Zeichen „G“ korrekt erkannt hat oder ob eine weitere „1“ für das „T“ fehlt.

führt zu einem Buchstaben. Da „T“ in unserem Text vorkommt, wir also darauf nicht verzichten können, müssen wir diesen freien Platz also schaffen!

Wenn wir den Code für häufige Buchstaben verkürzen, indem wir das entsprechende Plättchen nach links schieben, können wir natürlich auch Codes für seltene Buchstaben verlängern, indem wir das Plättchen nach rechts schieben. „C“ ist in der Nachricht nur drei Mal enthalten, also recht selten. Schieben Sie es daher auf eines der nachgeordneten freien Felder nach rechts oben oder rechts unten.

Nun ist ein Platz frei, auf den Sie das Plättchen für „T“ platzieren können. Abbildung 6.8 zeigt der fertigen Codebaum. Die Gensequenz oben kann nun ohne Probleme in „1000101011100110010101011100101100110101 00“ codiert und wieder decodiert werden. Sie benötigen dafür mit dem neuen Codebaum lediglich 42 Bit, haben also bereits in diesem recht einfachen Beispiel 6 Bit gespart.

Nach diesem sehr betreuten Experimentieren sind Sie richtig an der Reihe: Nehmen Sie die große Codebaum-Kopiervorlage aus Abbildung 6.K2 und legen die Buchstabenplättchen in der bekannten Reihenfolge auf die grauen Startfelder. Die Vorlage ist nun identisch mit dem Codebaum aus Abbildung 6.1.

Die Codelänge für ein Zeichen entspricht dem Abstand von der linken Seite zum entsprechenden Zeichen im Baum. Momentan sind alle Längen gleich, nämlich 5. Das möchten wir – genau wie im kleinen Beispiel – ändern: Der Buchstabe „E“ ist in der deutschen Sprache am häufigsten. Von 100 Zeichen sind innerhalb unserer kleinen

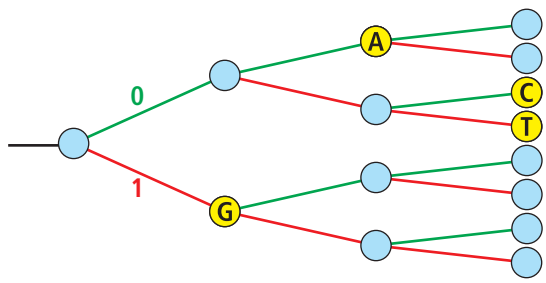


Abbildung 6.8
Optimaler Codebaum für die Beispielsequenz

Whn Rpnzl ws twlv yrs ld th nchntrss sht hr n twr whch ly n frst Th twr hd n dr bt hgh p ws wndw Whn th nchntrss wntd t g n sh plcd hrslf blw th wndw nd crd Rpnzl Rpnzl lt dwn yr hr Rpnzl hd mgnfcnt lng hr fn s spn gld Whn sh hrd th vc f th nchntrss sh wnd hr hr rnd hk f th wndw Th hr fll twnty yrds dwn nd th nchntrss clmbd p by t

Abbildung 6.9
Bekannter englischer Text, der nur noch seltene Buchstaben enthält (ohne Satzzeichen)

Zeichen	Häufigkeit / 100
_ (leer)	17,38
A	3,97
B	1,50
C	3,18
D	3,90
E	12,56
F	1,40
G	2,22
H	5,39
I	6,55
J	0,15
K	0,86
L	3,19
M	2,57
N	7,74
O	1,74
P	0,73
Q	0,03
R	5,38
S	5,33
T	5,03
U	3,24
V	0,51
W	1,49
X	0,03
Y	0,22
Z	0,83
Ä	0,40
Ö	0,27
Ü	0,52
ß	0,39
.	1,30

Auswahl im Mittel 12,56 „E“. Schieben Sie daher das entsprechende Plättchen gleich zwei Positionen nach links und sorgen dafür, dass die nun nachgeordneten Buchstabenplättchen anderweitig untergebracht werden, indem Sie seltene Buchstaben nach rechts schieben. Sie dürfen dabei auch gerne die Reihenfolge durcheinanderbringen – dem Computer ist es letztlich beim Codieren und Decodieren egal, ob die Codes für A und für B ähnlich sind.

Stellen Sie durch Verschieben einen Codebaum her, der Ihrer Meinung nach gut zu der am Buchrand dargestellten Häufigkeitsverteilung (in Prozent) der Buchstaben in der deutschen Sprache passt.

Ein mögliches Ergebnis des Buchstabenschiebens ist in Abbildung 6.12 auf der nächsten Doppelseite zu sehen. Es entsteht, wenn man die Buchstaben, nach Häufigkeit sortiert, von oben nach unten einfüllt und dabei mit sinkender Häufigkeit immer weiter nach rechts platziert. Sie können aber auch beliebige andere Systeme nutzen oder das Verschieben einfach aus dem Bauch heraus erledigen. Die Tabelle mit den Flexcodes entsteht zum Beispiel, wenn man die Buchstaben von oben nach unten in der alphabetischen Reihenfolge lässt, damit die manuelle Codierung und Decodierung leichter von der Hand geht.

Bevor wir weitermachen, denken Sie bitte kurz darüber nach, warum die so entstehenden Codes immer „funktionieren“, also beim Codieren und Decodieren eindeutig sind. Wir sorgen dafür, dass im Baum kein Buchstabenplättchen untergeordnet hinter einem anderen Buchstabenplättchen zu finden ist. Übertragen auf die resultierenden Binärcodes bedeutet dies, dass kein Code der Anfang oder Präfix einen anderen Codes ist. Man nennt einen solchen Code „präfixfrei“ und er entsteht mit unserem Verfahren ganz automatisch.

Präfix

Ein Präfix ist ein Wort oder eine Zeichenfolge, die mit dem Anfang einer anderen Zeichenfolge identisch ist. In der Codierung versucht man Präfixe zu vermeiden, da man dann codierte Nachrichten nicht mehr direkt decodieren kann.

Hans hatte sieben Jahre bei seinem Herrn gedient da sprach er zu ihm Herr, meine Zeit ist herum nun wollte ich gerne wieder heim zu meiner Mutter gebt mir meinen Lohn Der Herr antwortete du hast mir treu und ehrlich gedient wie der Dienst war so soll der Lohn sein und gab ihm ein Stück Gold das so groß als Hansens Kopf war Hans zog ein Tüchlein aus der Tasche wickelte den Klumpen hinein setzte ihn auf die Schulter und machte sich auf den Weg nach Haus

Abbildung 6.10
Anfang des Märchens „Hans im Glück“ der Gebrüder Grimm (ohne Satzzeichen)

Probieren Sie mit verschiedenen deutschen Texten aus, wie viele Bits Sie benötigen, um diese in Blockcode zu codieren, und stellen Sie Ihren eigenen Code dagegen.

Als Beispiel habe ich das mit folgendem berühmten Text (gleich in Großbuchtaben und ohne Zeichen, die nicht in unserer Tabelle enthalten sind) und dem Codebaum nach Abbildung 6.12 gemacht:

HABE NUN ACH PHILOSOPHIE JURISTEREI UND MEDIZIN UND LEIDER AUCH THEOLOGIE DURCHAUS STUDIERT MIT HEIßEM BEMÜHEN. DA STEH ICH NUN ICH ARMER THOR UND BIN SO KLUG ALS WIE ZUVOR. HEIßE MAGISTER HEIßE DOCTOR GAR UND ZIEHE SCHON AN DIE ZEHN JAHR HERAUF HERAB UND QUER UND KRUMM MEINE SCHÜLER AN DER NASE HERUM UND SEHE DASS WIR NICHTS WISSEN KÖNNEN. DAS WILL MIR SCHIER DAS HERZ VERBRENNEN.

Der Text hat insgesamt 383 Zeichen, für den Blockcode mit konstanter Codelänge von 5 Bit benötigen wir also 1915 Bit.

Wenn wir mit dem Codebaum 6.12 arbeiten, benötigen wir für Leerzeichen und E jeweils 3 Bit. Diese Zeichen kommen im Text genau 111-mal vor, das entspricht 333 Bits. Auf die gleiche Weise zählen wir N, I, H, R, S und T zusammen (138) und multiplizieren mit 4 Bits. A, U, D, C und L kommen 74-mal vor, M, G, O, B, W, F, Punkt, K und Z sind 47-mal vorhanden. Von den restlichen Buchstaben, die im Codebaum durch 7 Bits repräsentiert werden, gibt es lediglich 13 Stück. Daher kommen wir auf $111 \cdot 3 + 138 \cdot 4 + 74 \cdot 5 + 47 \cdot 6 + 13 \cdot 7 = 1628$ Bits – immerhin bereits eine Ersparnis von etwa 15 %.

Zum Vergleich: Mit dem Flexcode von vorher passt der Text in 1639 Bit. Schafft „Ihr" Codebaum es mit noch weniger?

Durch die Wahl eines Codes mit variabler Codelänge gegenüber dem mit fester Codelänge kann also Speicherplatz bzw. Übertragungskapazität eingespart werden. Die Einsparungen nutzen dabei aus, dass verschiedene Zeichen häufiger vorkommen als andere, und weisen den häufigeren kleinere Codelängen zu.

Frühe Datenkomprimierung
Das Konzept, häufige Zeichen einzusparen und dafür weniger häufig genutzte zu verlängern, ist bereits sehr alt. Ein Beispiel ist Sanskrit, eine Schriftsprache aus Indien. Der Vokal „a" kommt so häufig hinter dem „t", dass er gar nicht mehr geschrieben wird:

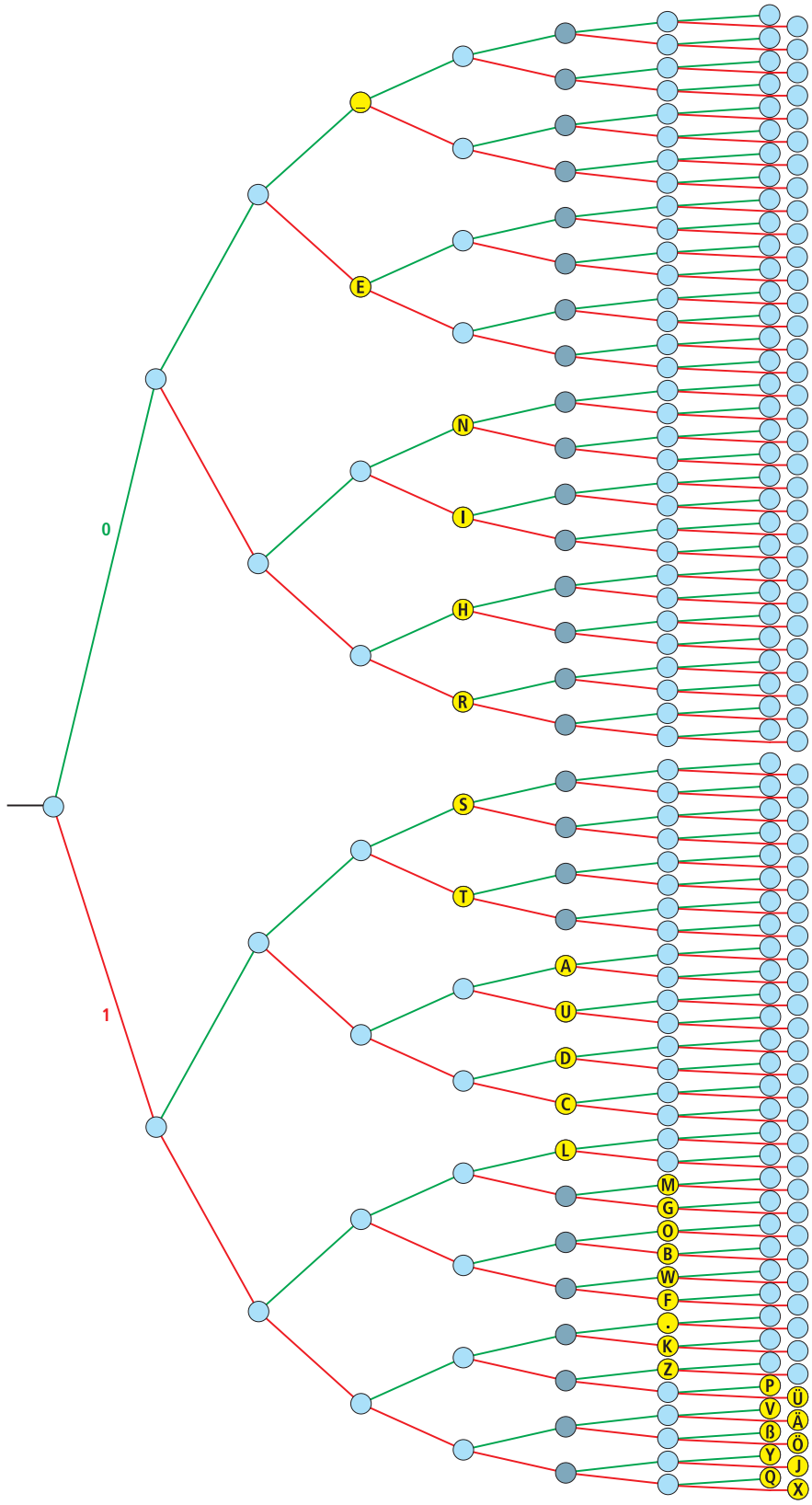
Möchte man – was selten vorkommt – lediglich „t" ohne „a" schreiben, muss man sogar etwas anhängen:

Das entspricht genau unserer Datenkomprimierung mit dem Codebaum: Häufig vorkommende Kombinationen werden durch einfachere Codes repräsentiert als seltene.

When Rapunzel was twelve years old the enchantress shut her in a tower which lay in a forest The tower had no door but high up was a window When the enchantress wanted to go in she placed herself below the window and cried Rapunzel Rapunzel let down your hair Rapunzel had magnificent long hair fine as spun gold When she heard the voice of the enchantress she wound her hair round a hook of the window The hair fell twenty yards down and the enchantress climbed up by it

Abbildung 6.11
Anfang des Märchens „Rapunzel" der Gebrüder Grimm in englischer Sprache (ohne Satzzeichen)

Abbildung 6.12
Mögliches Ergebnis des „Buchstabenschiebens“



Mit diesem Verfahren kann trotz kleinerer Datenmenge die eigentliche Nachricht wieder zu 100 % decodiert werden. Man nennt solche Verfahren daher auch verlustfreie Komprimierungen.

Verlustfreie Komprimierung

Populäres Synonym für eine Codierung, bei der die Datenmenge im durchschnittlichen Fall reduziert wird. Da es sich um eine Codierung handelt, ist gewährleistet, dass die Originaldaten aus dem Code wiederhergestellt werden können.

Entsprechende Verfahren werden von den gängigen Programmen wie Zip, ARJ, RAR usw. eingesetzt: Sie nutzen aus, dass in Daten immer bestimmte Zeichen oder auch ganze Wörter häufiger vorkommen als andere. Diese bekommen dann kürzere Codes als solche, die nur wenige Male in der Datenquelle enthalten sind.

Zip & Co

ZIP ist übrigens die Abkürzung für „Zigzag Inline Package", deutsch so etwas wie „zeilenweise im Zickzack gefaltetes (Daten-)Paket". Es wurde 1989 von Phil Katz geschrieben, der es als PKZIP (Phil Katz's Zip) vertrieb. Zip ist so bis heute der meistverbreitete Standard zur verlustfreien Datenkomprimierung. Andere bekannte Komprimierer sind ARJ (nach seinem Erfinder „Archiver von Robert Jung" genannt) und RAR (auch nach dem Erfinder „Eugene Roshal Archiver" benannt).

Faxe(n) machen

Schön und gut: Texte können wir nun Speicherplatz schonend ablegen und haben auch bereits eine gute Vorstellung davon, wie das gemacht wird. Was ist aber mit anderen Daten, zum Beispiel Bildern?

Hier behaupte ich einfach, dass sich das gleiche Verfahren nicht nur eignet, sondern dass dieses auch tatsächlich bei allen Faxgeräten der Welt eingesetzt wird.

Wenn Sie nun darüber grübeln, was eigentlich ein Faxgerät ist, gehören Sie zu den jüngeren Lesern dieses Buches. Was während der ersten Auflage von „Abenteuer Informatik" noch „State of the Art" war, ist inzwischen immer seltener in Büros zu sehen.

Das erste Faxgerät (Fax = Abkürzung für Faksimile, also „gleich Gemachtes") der Welt kam in den 1960er-Jahren aus Deutschland von der Firma Siemens. Mit dem Faxgerät können Briefe mit Text oder Zeichnungen über die Telefonleitung übertragen werden, indem diese beim Sender gescannt, digital übertragen und beim Empfänger wieder ausgedruckt werden.

Faxgeräte nutzen für die sparsame Übermittlung der Bilder Techniken, die auch in vielen anderen Bereichen der Bildverarbeitung verwendet werden – oft in einer verfeinerten Variante. Das Prinzip lässt sich daher immer noch besonders gut am Original erklären, weshalb wir hier beim „guten alten Fax" bleiben.

Abbildung 6.13
Fax-Telefon-Kombination aus den 1990er-Jahren für Thermopapierrollen

Um es zu verstehen, müssen wir zunächst anschauen, wie ein Bild überhaupt im Computer bzw. in einem Faxgerät aussieht. Nehmen wir an, Sie haben ein Bild Ihres Traumhauses gezeichnet, zum Beispiel das aus Abbildung 6.14, und wollen dieses an einen Architekten faxen, damit er es verwirklicht.

Das Faxgerät kann nicht erkennen, ob ein Text, eine Skizze oder etwas anderes verschickt werden soll, es erstellt aus der Seite ein sogenanntes Rasterbild. Das bedeutet, über die Seite wird ein Gitternetz, das Raster, gelegt. Sie sehen das in Abbildung 6.15 mit einem roten Raster.

Für jedes kleine Kästchen, auch Rasterpunkt genannt, muss nun die Entscheidung getroffen werden, ob es schwarz oder weiß ist. Eine weitere Aufteilung von Rasterpunkten gibt es nicht, andere Farben sind nicht möglich. Wenn man bei der Skizze des Traumhauses alle Rasterpunkte als schwarz definiert, durch die eine Linie verläuft, kommt man auf Abbildung 6.16.

Sie sehen, dass durch diese Maßnahme die Qualität leidet. Besonders die Details um die Fenster herum gehen in diesem Beispiel verloren. Beim Fax-Standard sind pro Millimeter etwa acht Rasterpunkte festgelegt. Vielleicht haben Sie auch bei echten Fax-Sendungen oder Ausdrucken von gescannten Dokumenten schon festgestellt, dass kleine Schrift teilweise kaum mehr lesbar ist.

Die Rasterung ist allerdings unbedingt notwendig, denn nun kann das Bild auch gespeichert werden: Jeder Rasterpunkt ist einer Speicherstelle zugeordnet, die dann die Information „Schwarz“ oder „Weiß“ beinhaltet.

Wenn wir versuchen, das Bild über eine Leitung zu übertragen, können wir das einfach Zeile für Zeile von links nach rechts tun. Die Übertragung würde für die ersten vier Zeilen dann zum Beispiel lauten:

WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWSSWWWWWWWWWWWW
WWWWWWWWSSSSWWWWWWWW

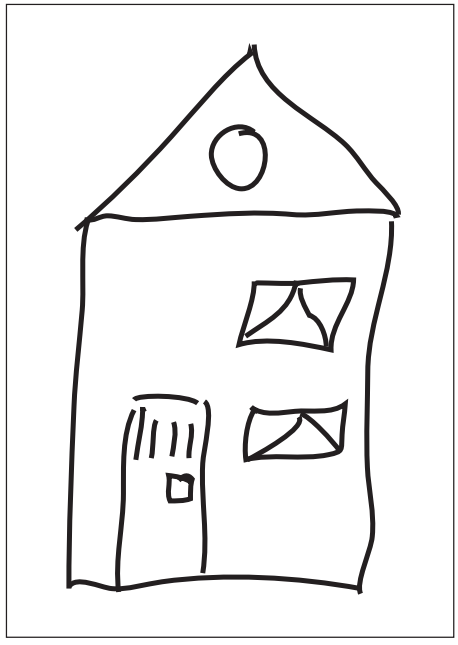


Abbildung 6.14
Die Seite mit Ihrer Traumhausskizze

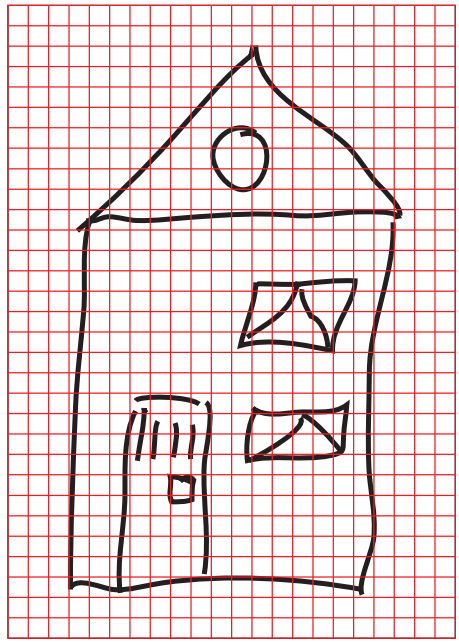


Abbildung 6.15
Raster über dem Traumhaus

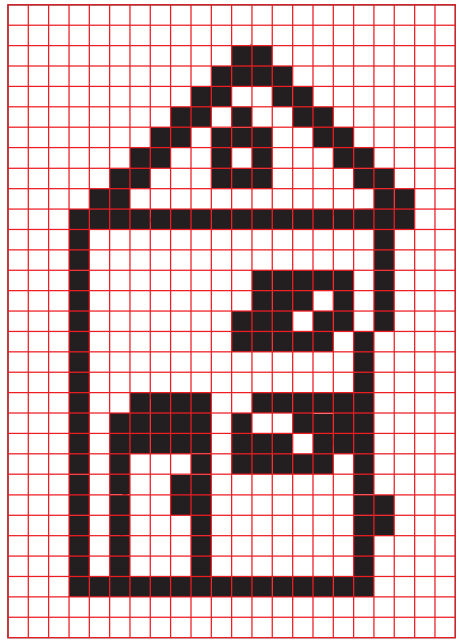


Abbildung 6.16
Gerastertes Traumhaus

Vielleicht fällt Ihnen bereits anhand dieses Beispiels eine Verbesserungsmöglichkeit ein?

Bei den Zeichnungen wechseln sich immer sehr lange Reihen von weißen Rasterpunkten mit kürzeren Reihen schwarzer Rasterpunkte ab. Wenn man nun einfach die Anzahl dieser Punkte zählt und abwechselnd für Weiß und Schwarz übermittelt, sieht die Übertragung schon sehr viel übersichtlicher aus:

55-2-19-4-8

Diese Darstellung nennt man übrigens „Lauflängen“ bzw. englisch „run-length“.

Eine Lauflängencodierung bzw. deren Weiterentwicklung LZW (nach Lemple, Zif und Welch) wird in vielen bekannten Bild- und Dokumentformaten auf dem Computer standardmäßig eingesetzt, um Speicherplatz zu sparen. Darunter sind BMP, GIF, TIFF, PDF und PostScript.

Lauflängencodierung

Bei der Lauflängencodierung werden nicht einzelne Bildpunkte gespeichert bzw. übermittelt, sondern die Anzahl gleicher Bildpunkte hintereinander. Englisch lautet der entsprechende Fachausdruck „run-length“. In manchen Bildverarbeitungsprogrammen bekommen Sie beim Speichern die Möglichkeit, „RLC“ auszuwählen, was ausgeschrieben „run-length coding“ bzw. Lauflängencodierung bedeutet.

Spielen Sie doch einfach einmal Faxgerät und wandeln die gesamte Traumhaus-Zeichnung in Zahlen um, die der jeweiligen Länge von weißen und schwarzen Bereichen entsprechen. Dabei können Sie vielleicht auch schon mal darüber nachdenken, wie man hier noch weiter einsparen könnte.

Den Anfang kennen Sie ja bereits. Die vollständige Zahlenreihe lautet:

55-2-19-4-17-2-2-2-15-2-1-1-2-2-13-2-1-3-2-2-11-2-2-1-1-1-3-2-9-2-3-3-4-2-7-

2-12-2-5-17-5-1-14-1-6-1-14-1-6-1-8-5-1-1-6-1-8-3-1-1-1-1-6-1-7-3-1-2-1-1-6-

1-7-5-1-1-7-1-13-1-7-1-13-1-7-1-2-4-2-6-7-1-1-5-1-1-2-4-7-1-1-5-1-3-1-3-7-1-

1-1-3-1-1-5-1-1-7-1-1-1-2-2-7-1-7-1-1-1-2-2-7-2-6-1-1-1-3-1-7-2-6-1-1-1-3-1-

7-1-7-1-1-1-3-1-7-1-7-15-48

Selbstverständlich können wir diese Zahlen einfach übermitteln. Eine Zahl zwischen 0 und 63 benötigt 6 Bit, wie Sie spätestens seit Kapitel 4 auch selbst ausrechnen können. Insgesamt sind für das Dokument 159 Zahlen zu übermitteln, also 954 Bit.

Auch wenn die Übertragung übersichtlicher geworden ist, gespart haben wir nicht: Wenn man die Bildpunkte mit je einem Bit einzeln übermittelt hätte, nämlich als „0“ für Weiß und „1“ für Schwarz, hätten wir für 31 Zeilen mal 22 Spalten mal 1 Bit lediglich 682 Bit benötigt.

Wenn Sie jetzt das Wissen vom Anfang dieses Kapitels nutzen: Vielleicht kommen Sie auf die zündende Idee, wie die Lauflängencodierung doch noch zum Sparen von Speicherplatz dienen könnte ...

Schauen wir uns doch einfach einmal an, wie oft welche Lauflängen in unserem Beispiel vorkommen:

Lauflänge	Anzahl	Lauflänge	Anzahl	Lauflänge	Anzahl	Lauflänge	Anzahl
1	66	7	18	13	3	19	1
2	27	8	2	14	2	...	0
3	12	9	1	15	2	48	1
4	4	10	0	16	0	...	0
5	7	11	1	17	2	55	1
6	8	12	1	18	0	...	0

Selbstverständlich können wir jetzt wieder das Prinzip des Codebaums mit variabler Codelänge nutzen – mit Lauflängen statt Buchstaben des Alphabets. Auf diese Weise kann 1 durch einen sehr kurzen Code und zum Beispiel 9 durch einen sehr langen Code repräsentiert werden.

Einen möglichen Codebaum sehen Sie in Abbildung 6.17.

Codieren Sie die Zahlenreihe nun von Neuem – diesmal mit Hilfe des Codebaums. Wie viele Bits benötigen Sie nun für das Bild?

Die korrekte Codierung lautet:

1101011100110000110111011001110010010011001010000100100110100100010101
0010010111111100100000101010010111110100101010101011101001111001100000
1001101111001111011011000101011001100010101100101111011011001011001011
1101010000010110011110100100000101100111110110011101101000111011010000 11
1010010111010010110111001101100100101110111001101101010010101110001010
0011011001110001001001110111000100100111100101100001010011110010110000
1010011101110001010011101111100101101010

Sie hat lediglich 460 Bit, was eine Einsparung von etwa einem Drittel gegenüber der vollständigen Übertragung darstellt.

Jedes Faxgerät funktioniert nach (fast) genau diesem Prinzip. Die ursprünglichen Entwickler haben von zehn Dokumenten, die sie als „normale“ Vorlagen angenommen haben, die Lauflängen der schwarzen und weißen Bereiche berechnet und daraus – für

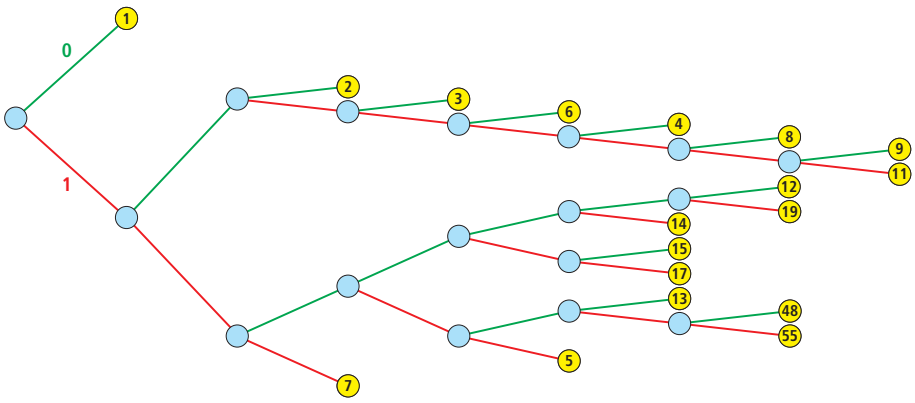


Abbildung 6.17
Codebaum für das Faxgerät

Claude Elwood Shannon (1916–2001) gilt als Begründer der Informationstheorie. Er beschäftigte sich professionell als Mathematiker und Elektrotechniker mit der Übermittlung von Nachrichten und schuf ein Modell des Informationsbegriffs, um die Übermittlung fehlertolerant und effizient zu machen. Daneben hat Shannon auch eine ganze Reihe außergewöhnlicher, kreativer Apparate erfunden und gebaut – etwa eine Jonglagemaschine oder eine mechanische Maus, die aus einem Labyrinth finden konnte.

Schwarz und Weiß getrennt – optimale Codebäume erstellt. Alle Faxgeräte der Welt haben diese Codebäume eingespeichert, so dass sie ein Fax nach dem Standard codieren und auch decodieren können.

Falls Sie irgendwo noch ein altes, funktionstüchtiges Faxgerät auftreiben, können Sie das auch experimentell nachvollziehen: Schicken Sie zunächst einen normalen Brief und stoppen Sie die Zeit, die das Faxgerät benötigt, um diesen zu senden. Das geht recht flott! Als Nächstes schicken Sie ein sehr ungewöhnliches Dokument. Dazu eignet sich zum Beispiel die gemusterte Rückseite einer Gehaltsabrechnung oder ein Blatt Papier, das Sie willkürlich flächendeckend mit Kugelschreiber verkritzelt haben. Alle anderen Vorlagen mit einem zufälligen, sehr dicht ausgeprägten Muster gehen ebenso. Stoppen Sie wiederum die Zeit! Dieses Fax benötigt deutlich länger.

Falls Sie kein altes Gerät auftreiben können, das den Inhalt beim Einziehen des Papiers gleich wegschickt und keinen Zwischenspeicher besitzt, funktioniert das auch mit modernen Geräten, etwa einem Router mit eingebauter Faxfunktion. Wichtig ist, dass Sie die Dauer des Sendens verfolgen können.

Sehr schön! Sie wissen jetzt, wie unser Verfahren sogar für die sparsame Übermittlung von Bildern eingesetzt werden kann.

Rhabarberbarbara

Eventuell sind Sie immer noch etwas unzufrieden: Mit unserem Buchstabenschieben haben wir bisher maximal 15 % eingespart, im letzten Abschnitt wurde dann behauptet, dass z. B. Zip auf diese Weise funktioniert. Zip schafft aber eine deutlich höhere Einsparung und ich hoffe, Sie sind neugierig, wie das zustande kommt. Das Geheimnis liegt in der Berücksichtigung ganzer Wörter oder Datensequenzen. Allein wenn wir Buchstabenpaare betrachten, liegt auf der Hand, dass (von Abkürzungen und zusammengesetzten Wörtern einmal abgesehen) etwa Kombinationen wie NR oder TD sehr selten sind.

Nun kann man einen riesigen Codebaum erstellen, der statt 32 Buchstaben alle 1024 möglichen Kombinationen enthält – mit kleinen Buchstaben sind es entsprechend noch mehr. Hier möchte ich eine Art Trick auf Basis eines Verfahrens von Michael Burrows und David Wheeler vorstellen. Die Burrows-Wheeler-Transformation (BWT) wird in modernen Codierungsverfahren eingesetzt, um die Daten vor der eigentlichen Codierung möglichst „einfach“ zu machen. Das passiert auf Basis der unterschiedlichen Häufigkeiten von Buchstabenkombinationen.

Die Idee der Geschichte der Rhabarberbarbara stammt aus dem Internet, wo Sie viele Unterschiedliche Versionen und Umsetzungen finden.

Als Beispiel widmen wir uns einer Dame namens Barbara, die so guten Rhabarbersaft herstellte, dass sie überall nur noch Rhabarberbarbara genannt wurde. Ihre Saftschenke hieß daher nach kurzer Zeit Rhabarberbarbararhabarberbar. Unter den Stammkunden befand sich ein Barbar mit prächtigem Bart – der Bartbarbar. Als er irgendwann nur noch in Rhabarberbarbaras Bar angetroffen wurde, nannte man ihn kurzerhand Rhabarberbarbararhabarberbarbartbarbar.

Warum habe ich Ihnen diese Geschichte erzählt? Einerseits ist sie lustig und enthält gleich mehrere Zungenbrecher, andererseits können wir das resultierende Wort hervorragend als Beispiel gebrauchen, um die BWT zu demonstrieren, denn hier kommen auf kleinem Raum bestimmte Buchstabenkombinationen besonders häufig vor.

Die BWT macht aus

*RHABARBERBARBARARHABARBERBARBARTBARBAR

die Kombination:

RHHBBBBBBBBRBRRTRRAARRRBBRRAAAAEEAAA*AAR

Insbesondere nach unserem Exkurs in Lauflängencodierung können Sie sich vorstellen, dass man letzteren Text nochmal deutlich sparsamer verschicken kann, indem man Folgen aus gleichen Buchstaben besonders behandelt. Das Verblüffende ist, dass die BWT in beide Richtungen funktioniert, man also aus dem zweiten Wort wieder ohne Probleme das erste herstellen kann. BWT basiert darauf, dass gleiche Buchstabenfolgen Wiederholungen eines Buchstaben provozieren.

Wir wollen das anhand des Wortes BARTBARBAR ausprobieren. Die BWT funktioniert folgendermaßen:

1. Ergänzen Sie das Wort um ein Startzeichen vorne, das im Wort nicht vorkommt.

2. Bilden Sie alle Rotationen des Wortes, also alle Wörter, die entstehen, wenn Sie jeweils einen Buchstaben vom Anfang ans Ende schieben.

3. Sortieren Sie diese Rotationen lexikalisch.

4. Nehmen Sie in der sortierten Reihenfolge jeweils den letzten Buchstaben. Diese Buchstaben ergeben von oben nach unten das BWT-Wort.

Führen wir dieses Verfahren an unserem Beispiel durch. Das funktioniert sehr schön mit kariertem Papier, wie hier beschrieben, aber selbstverständlich auch direkt, z. B. in einem Texteditor.

Zunächst benötigen wir alle Rotationen. Um diese akkurat zu erstellen, schreiben wir das Wort mit vorgestelltem Startzeichen X zwei Mal direkt hintereinander auf das Karopapier.

Dieses können Sie nun durch Verschieben um jeweils eine Position nach links nutzen, um alle Rotationen untereinander auf ein neues Stück Karopapier zu schreiben. Abbildung 6.18 auf der nächsten Seite zeigt das.

Nun schneiden Sie alle Rotationen als schmale Streifen aus und sortieren diese alphabetisch wie in Abbildung 6.19.

Die letzte Spalte von Buchstaben von oben nach unten gelesen, ergibt das BWT-Wort:

BBBTXRAAARR

Erkennen Sie, wann hier mehrere gleiche Buchstaben direkt hintereinander stehen? Die Sortierung ordnet Passagen mit gleichem Präfix hintereinander. Der letzte Buchstabe ist durch die Rotation effektiv der, der eigentlich direkt davor steht. Wenn also gleiche Passagen implizieren, dass der gleiche Buchstabe vorangestellt ist, äußert sich das in einer Sequenz aus gleichen Buchstaben im BWT-Wort.

Das Verfahren zur Rückwandlung des BWT-Worts in das ursprüngliche Wort ist sehr einfach, aber etwas aufwendiger. Bereiten Sie dazu zunächst so viele dünne Streifen Karopapier vor, wie Buchstaben im BWT-Wort enthalten sind.

Abbildung 6.18
Alle Rotationen unseres Wortes

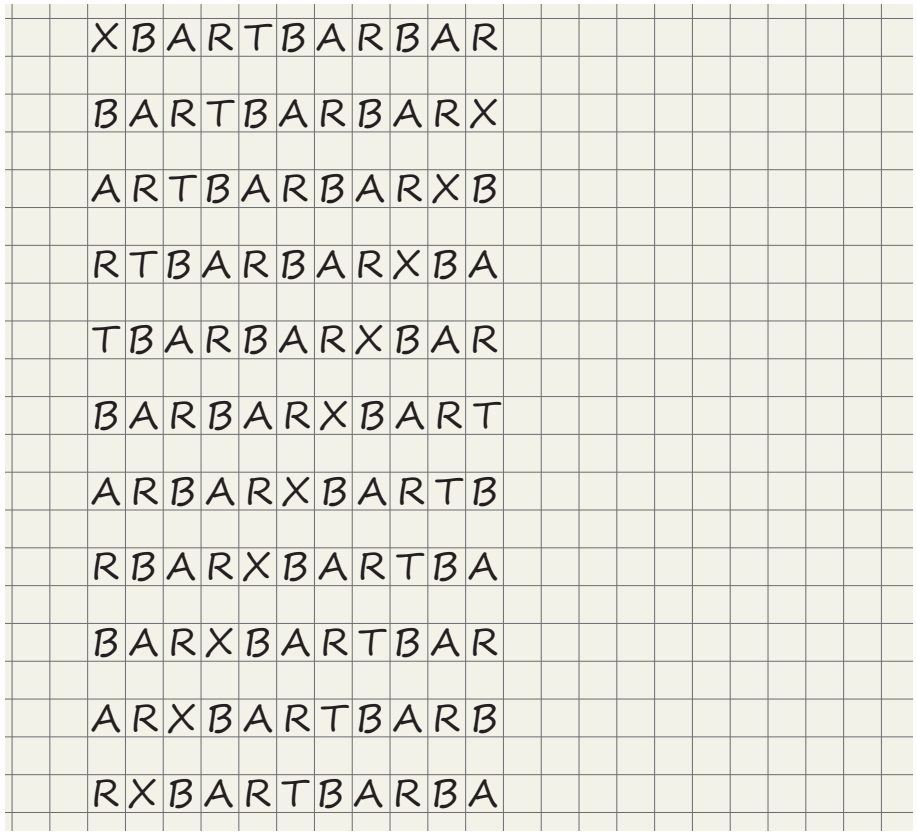


Abbildung 6.19
Sortierte Rotationen

Schreiben Sie das BWT-Wort dann von oben nach unten auf ein frisches Blatt. In Abbildung 6.20 habe ich das zur Verdeutlichung in Rot dargestellt.

Die Streifen Karopapier kommen direkt hinter jeden Buchstaben. Wir übertragen die Buchstaben auf die hinterste Position des Karopapiers. Auch das ist in der Abbildung schon gemacht.

Als Nächstes sortieren Sie die Streifen alphabetisch von oben nach unten. Übertragen Sie danach wiederum die roten Buchstaben in die letzte freie Spalte der Streifen. Abbildung 6.21 zeigt das.

Diesen Schritt – also sortieren und anschließend die roten Buchstaben wieder in die letzte freie Spalte übertragen – wiederholen Sie nun, bis die Streifen alle so viele Zeichen enthalten wie das BWT-Wort. Abbildung 6.22 zeigt das Ergebnis: alle Rotationen des ursprünglichen Wortes. Dank des Startzeichens können Sie sofort die „richtige" davon identifizieren.

Probieren Sie das ruhig ein paar Mal mit eigenen Beispielen aus – es macht Spaß! Wenn Sie beim Schritt des Sortierens aufmerksam sind, werden Sie feststellen, dass dieses nach kurzer Zeit immer in exakt gleicher Weise abläuft, die Streifen müssen also jedes Mal auf die gleiche Weise vertauscht werden. Das ist ein kleiner Hinweis darauf, wie das BWT-Verfahren dann tatsächlich auf einem Computer abläuft, wenn nicht nur ein Wort, sondern ein ganzer Text oder eine andere Datei durcheinandergewirbelt und wieder geordnet wird.

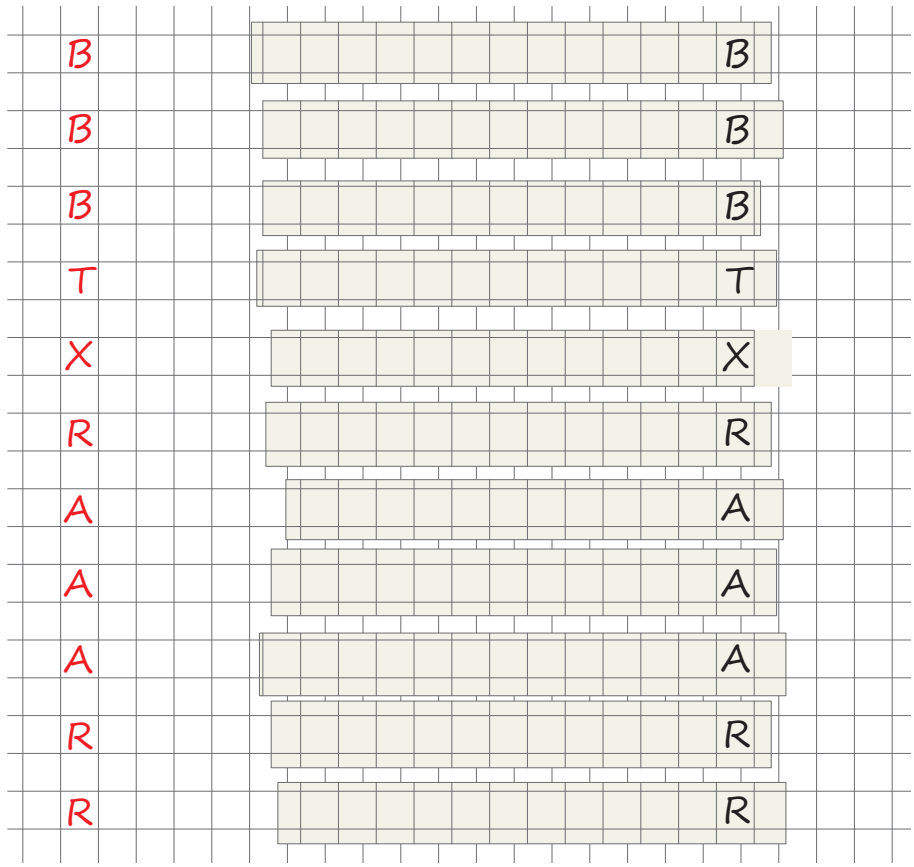
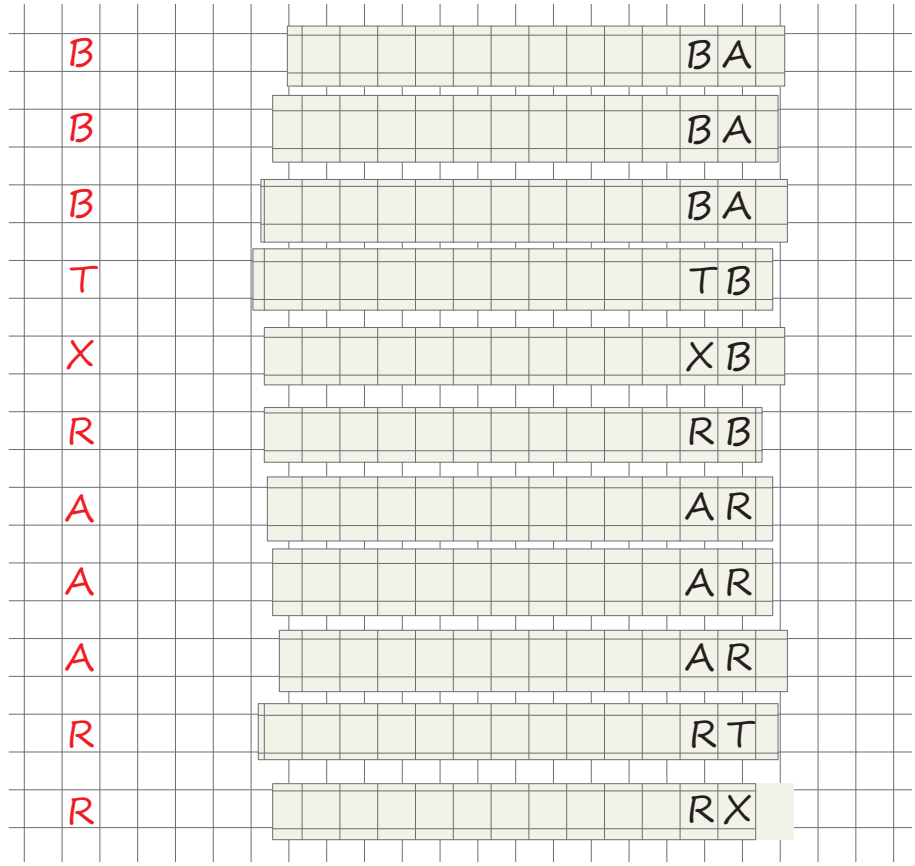


Abbildung 6.20
Das BWT-Wort wird in die letzte Spalte übertragen.

Abbildung 6.21
Nach dem Sortieren und erneuten Übertragen der roten Buchstaben



Mit BWT haben Sie nun ein Verfahren kennen gelernt, das selbst die Datenmenge eines Textes nicht reduziert, sondern lediglich die Anordnung der Zeichen verändert. Wiederholungen von Sequenzen werden auf diese Weise so deutlich, dass der Text danach wesentlich besser codiert werden kann. Zusammen mit dem vorher durchgeführten Buchstabenschieben im Codebaum ergibt die BWT eines der besten bekannten Verfahren zur verlustfreien Komprimierung.

Was steckt dahinter?

Ein guter Codebaum ist wesentlich für eine gute verlustfreie Komprimierung. Je besser der Codebaum an die tatsächlich übermittelten Daten angepasst wird, desto weniger Speicherplatz benötigen wir für diese.

In den letzten Abschnitten haben wir den Codebaum eher intuitiv durch Legen bzw. Verschieben von Buchstabenplättchen auf dem Standard-Codebaum erzeugt. Das ist allerdings noch etwas unbefriedigend: Woher wissen wir, dass es nicht noch eine bessere Möglichkeit zur Codierung gibt? Außerdem wäre es sinnvoll, wenn das Aufstellen des Codebaums nicht manuell erfolgen müsste, sondern ebenfalls per Algorithmus vom Computer übernommen würde.

Wie üblich brauchen wir also eine zündende Idee für ein Verfahren. Bisher sind wir recht häufig fündig geworden, wenn wir das informationstechnische Problem mit etwas aus dem täglichen Leben verglichen haben. Probieren wir es hier auch einmal.

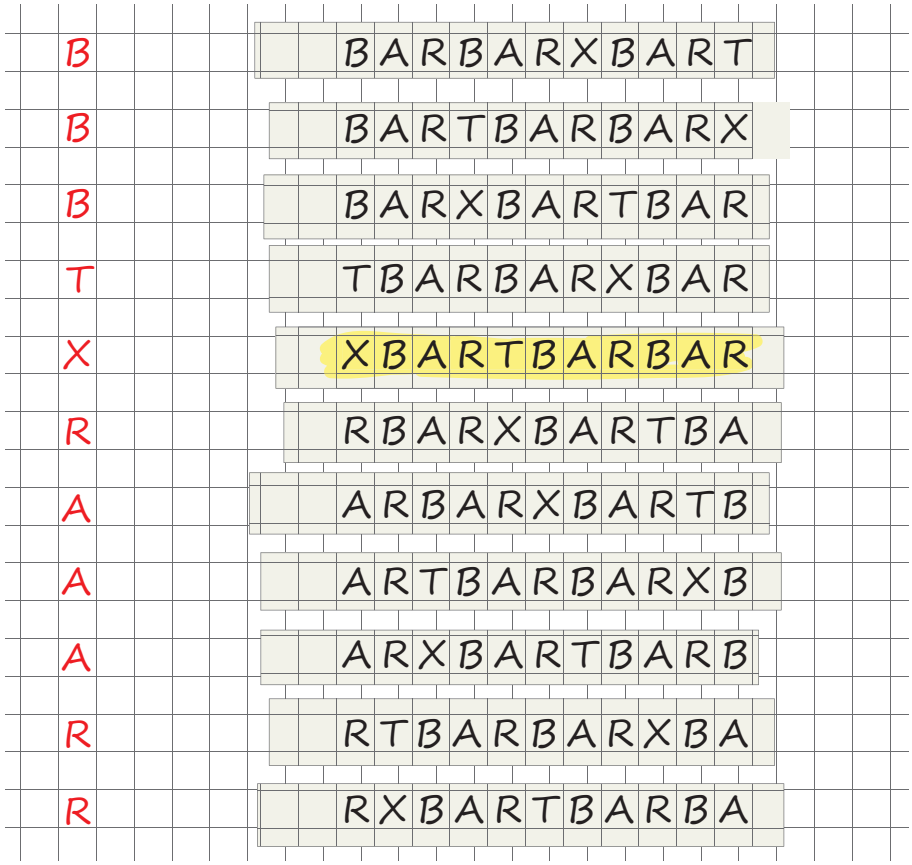


Abbildung 6.22
Das ursprüngliche Wort ist wieder lesbar.

Wissen Sie, wie ein Mobile gebastelt wird? Es hat Querstangen, an denen mit einem Faden jeweils ein Gegenstand oder eine weitere Querstange befestigt ist. Abbildung 6.23 zeigt ein Mobile mit Querstangen, die genau in der Mitte die Öse für den Faden haben.

Was können wir über dieses Mobile sagen? Alle Querstangen sind im Gleichgewicht. Denken Sie an eine Balkenwaage, wie sie auf Marktplätzen gebraucht wird: Das be-

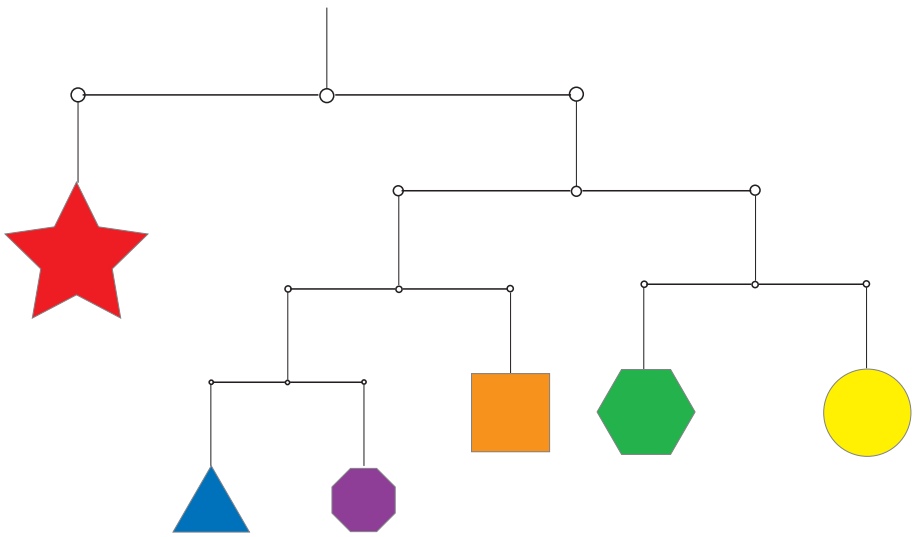


Abbildung 6.23
Ein einfaches Mobile

deutet, dass links und rechts jeweils gleich große Gewichte hängen. Hängt an einer Seite mehr als ein Gewicht, muss bei jeder Querstange die Summe der Gewichte links gleich der Summe der Gewichte rechts sein. Das Eigengewicht der Stangen und Fäden vernachlässigen wir der Einfachheit halber.

Kleines Rätsel: Wenn alle Gewichte in Abbildung 6.23 zusammen 80 Gramm wiegen, wie viel wiegt dann jedes Gewicht einzeln?

Zur Lösung können Sie nun ein Gleichungssystem aufstellen oder aber einfach raten. Es gibt nur eine Antwort, mit der sich das Mobile komplett im Gleichgewicht befindet. Abbildung 6.24 gibt Auskunft.

Vergleichen Sie nun das Mobile mit Ihrem optimierten Codebaum! Beachten Sie dabei die Häufigkeitsverteilung der einzelnen Zeichen!

Drehen Sie einfach den Codebaum um ein Viertel im Uhrzeigersinn. Sehen Sie die Ähnlichkeiten mit dem Mobile? Genau wie beim Mobile die kleinsten Gewichte ganz unten hängen, sind das beim Codebaum die Zeichen mit geringer Häufigkeit. Je weiter unten ein Zeichen hängt, desto länger ist der entsprechende Code.

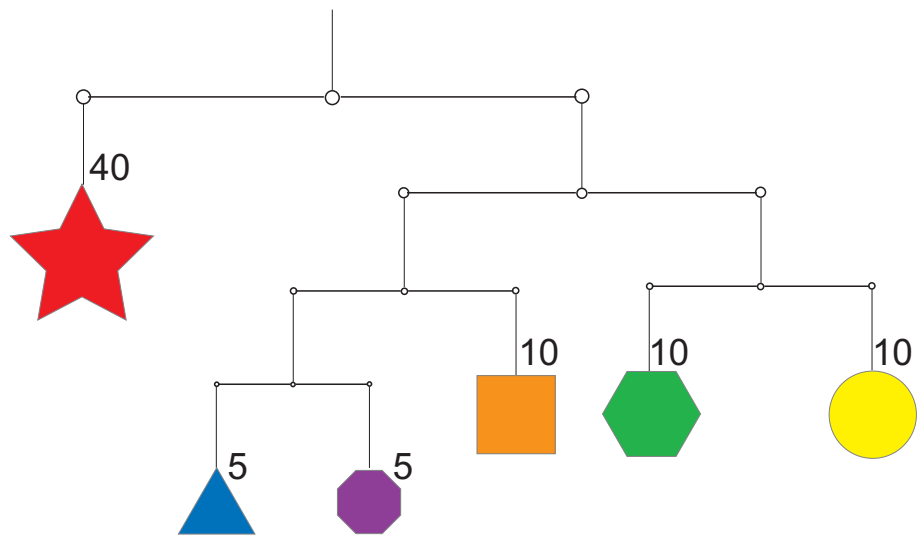
Genau das wollen wir aber erreichen: Je seltener ein Zeichen vorkommt, desto länger darf der Code sein. Aufgrund der Ähnlichkeit zwischen dem Gewicht beim Mobile und der Häufigkeit beim Codebaum spricht man hier übrigens auch von gewichtsbalancierten Bäumen. Um also einen guten Codebaum zu konstruieren, können wir uns daran orientieren, wie wir ein Mobile bauen:

Die kleinsten Gewichte müssen ganz nach unten. Daher suchen wir zunächst die beiden Objekte mit den kleinsten Gewichten und verbinden sie mit einer Querstange. Damit haben wir jetzt ein neues Objekt mit einem größeren Gewicht geschaffen.

Wiederum suchen wir die beiden Objekte bzw. Gebinde mit dem kleinsten Gewicht und verbinden sie. Das geht so weiter, bis alle Objekte im Mobile aufgehängt sind.

Die Abbildungen 6.25 bis 6.31 zeigen an einem Beispiel die Schritte mit Objekten sehr unterschiedlichen Gewichts.

Abbildung 6.24
Gewichtsverteilung im Mobile



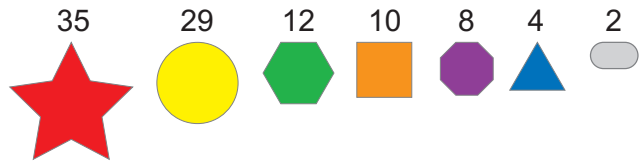


Abbildung 6.25
Sieben Objekte mit deren Gewichten

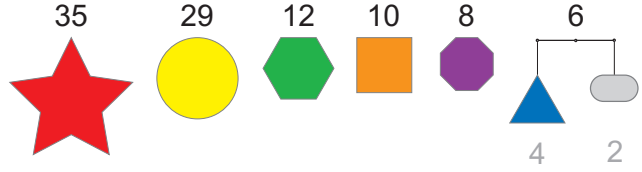


Abbildung 6.26
Zunächst werden das blaue Dreieck und das graue Rechteck zusammengefasst. Der Verbund wiegt nun 6 Gramm.

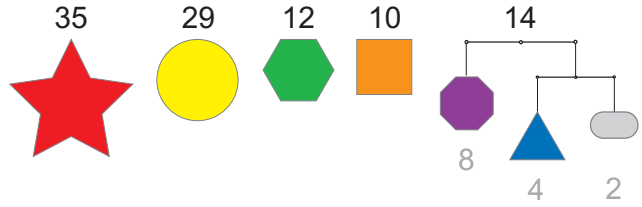


Abbildung 6.27
Wieder müssen die leichtesten Objekte gesucht werden, das sind der bisherige Verbund und das violette Achteck. Zusammen bringen sie nun 14 Gramm auf die Waage.

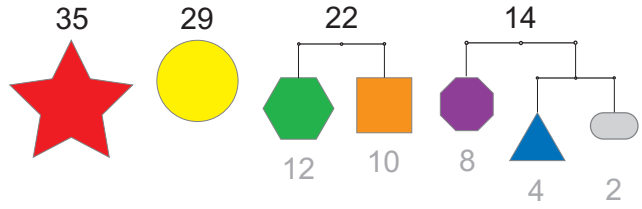


Abbildung 6.28
Nun sind zwei neue Objekte die leichtesten: Grünes Sechseck und oranges Quadrat werden zusammengefasst.

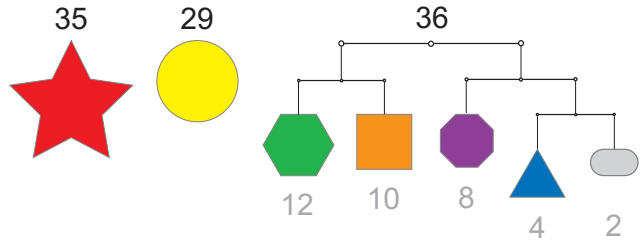


Abbildung 6.29
Danach sind beide Verbünde die leichtesten Objekte, sie werden zu einem 36 Gramm wiegenden Verbund zusammengefasst.

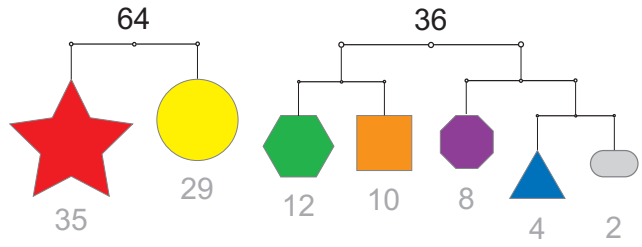


Abbildung 6.30
Als Nächstes müssen gelber Kreis und roter Stern einen Verbund bilden.

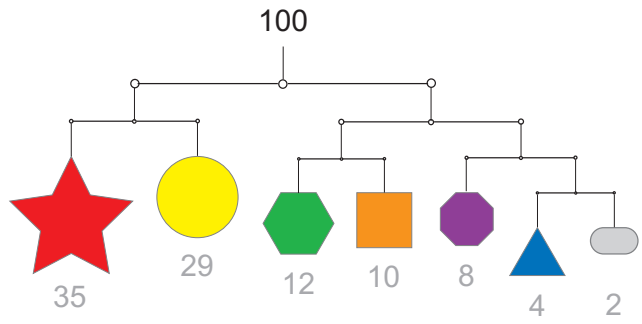


Abbildung 6.31
Im letzten Arbeitsschritt wird das komplette Mobile fertiggestellt. Es wiegt 100 Gramm.

Sie können erkennen, dass an jeder Querstange links und rechts ähnlich große Gewichte hängen, so gut das mit den gegebenen Objekten eben geht. Durch das Bauen von unten nach oben werden leichte und schwere Objekte an ihren korrekten Platz eingesetzt.

Die Vorgehensweise beim Aufstellen eines optimalen Codebaums ist absolut identisch: Fassen Sie jeweils die beiden Zeichen mit den geringsten Häufigkeiten zusammen. Beide gelten nun als „Superzeichen" mit der entsprechenden Häufigkeitssumme.

Diese Vorgehen heißt auch „Huffman'scher Algorithmus".

Die Abbildungen 6.32 bis 6.40 zeigen, wie man mit seiner Hilfe auf den kleinen Codebaum für das Faxgerät kommt. Noch nicht zusammengefasste Zeichen sind dabei gelb dargestellt. Der Übersicht halber verzichte ich auf die Beschriftung „0" und „1". Wir gehen einfach davon aus, dass „nach oben" bzw. eine grüne Verbindung einer „0" entspricht und „nach unten" bzw. eine rote Verbindung der „1"!

Abbildung 6.32
Ausgangslage – die Zeichen bzw. Lauflängen (in den Kreisen) und deren Häufigkeit (violett)

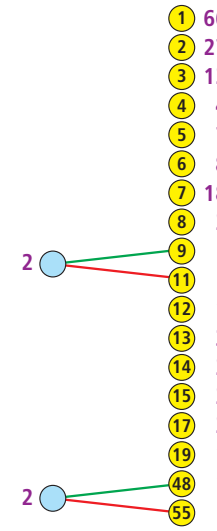


Abbildung 6.33
Die kleinsten Häufigkeiten werden zusammengefasst, das sind jeweils zwei beliebige Lauflängen mit einer Häufigkeit von 1.

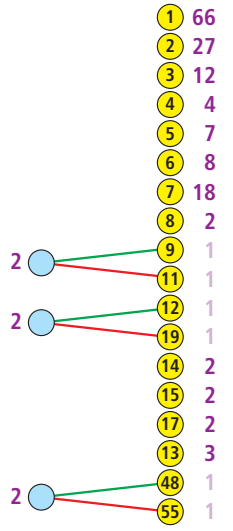


Abbildung 6.34
Der Übersicht halber werden die Lauflängen 13 und 19 vor dem Zusammenfassen getauscht.

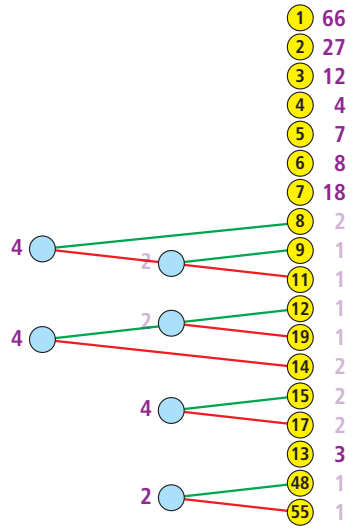


Abbildung 6.35
Zusammenfassen der Häufigkeiten 4. Der Bequemlichkeit halber nehmen wir die nebeneinanderliegenden Einträge. Technisch wäre es genauso gut, wenn wir das über Kreuz machen würden.

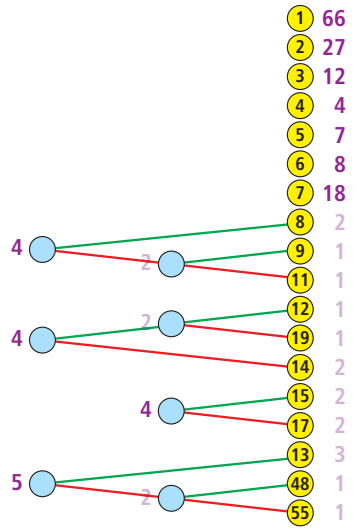


Abbildung 6.36
Nun sind die verbliebenen Häufigkeit 2 und die 3 an der Reihe.

Abbildung 6.37
Zusammenfassen der Häufigkeiten 4

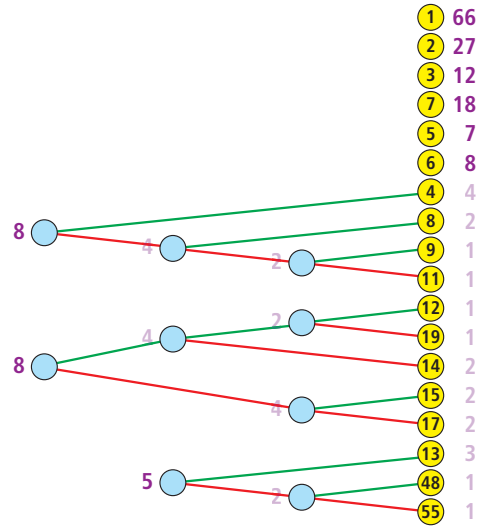


Abbildung 6.38
Da nun 5 und 7 zusammengefasst werden müssen, wird das Plättchen mit der Häufigkeit 7 der Übersicht halber nach unten gelegt.

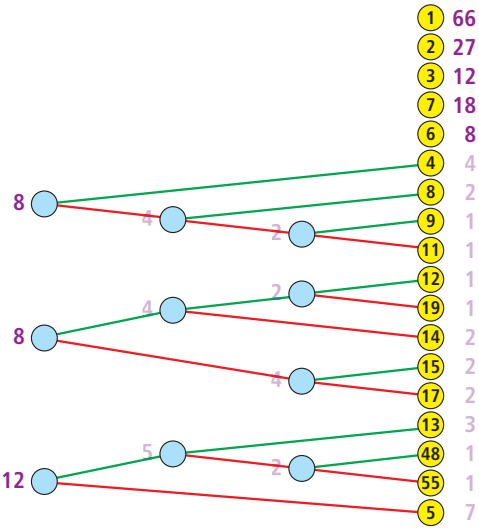
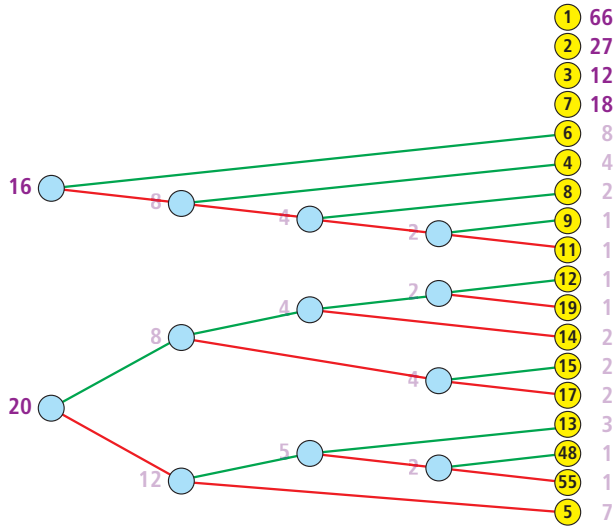


Abbildung 6.39
Als Nächstes sind 8 und 12 die kleinsten Häufigkeiten.



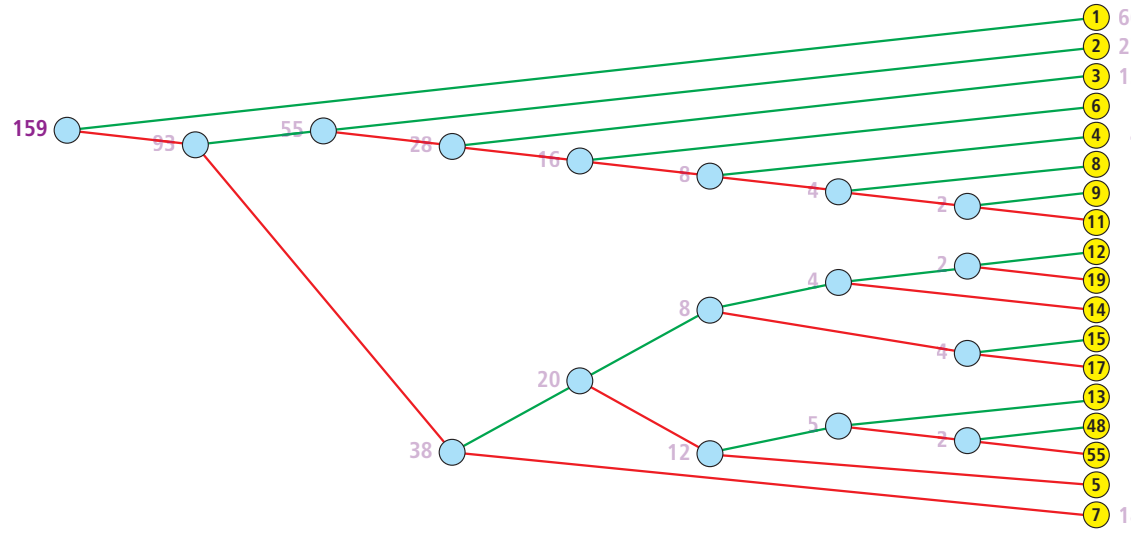


Abbildung 6.40
So geht das Verfahren weiter, bis der Codebaum fertig ist. Er entspricht dem in Abbildung 6.17.

Jetzt haben wir auch nachvollzogen, wie man auf den Codebaum für die Lauflängen weiter oben gekommen ist. Bitte beachten Sie, dass dies nicht die einzige Lösung ist! Da immer wieder gleiche Häufigkeiten auftraten, musste oft zufällig ausgewählt werden, welche Zeichen man miteinander kombinierte. Durch den Huffman'schen Algorithmus können also viele verschiedene Codebäume entstehen. Alle sind jedoch optimal.

Sie können gleich an einem richtigen Projekt üben: Wenden Sie den Huffman'schen Algorithmus an, um einen optimalen Codebaum für die Buchstaben des Alphabets mit den Häufigkeiten vom Anfang des Kapitels zu bekommen. Das geht am besten mit einem Stift und Papier. Eine entsprechende Vorlage finden Sie in Abbildung 6.K3. Sie dürfen gerne die Schere nutzen, um die Zeichen zunächst nach Häufigkeiten zu sortieren.

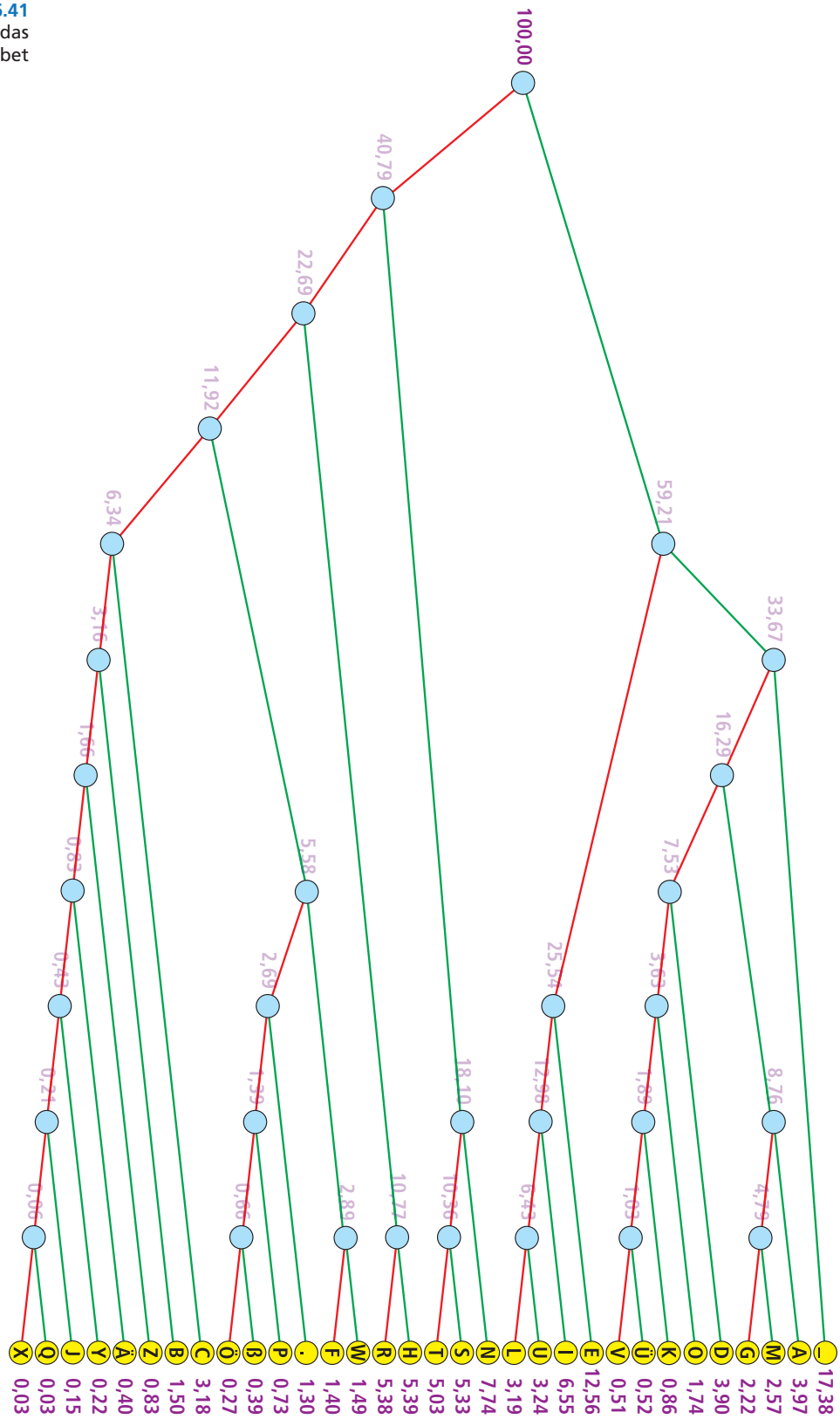
Ein mögliches Ergebnis ist in Abbildung 6.41 dargestellt.

Nochmal Shannon

Claude Elwood Shannon ist uns schon in der Spalte am Buchrand begegnet. Seine Erkenntnisse über Information sind aber so interessant, dass ich ihnen auch noch einen separaten Abschnitt widmen möchte.

Stellt man Menschen die Frage „Was ist Information?", wird diese oft nur mit einem Schulterzucken quittiert. Antworten, die man bekommt, sind meistens sehr unterschiedlich. Eine allgemeine Definition fällt zunächst schwer. Shannon verknüpfte Information mit Ungewissheit: Wer allwissend ist, kann keine Information erhalten, denn ihm ist ja alles wohlbekannt. Wenn Sie allerdings bezüglich eines Schverhaltes unsicher sind, dann räumt Information die damit verbundene Ungewissheit ggf. aus – je größer die ausgeräumte Ungewissheit, desto größer die Information.

Abbildung 6.41
Möglicher Codebaum für das kleine Alphabet



Information

Information ist das Beseitigen von Ungewissheit.

Das entspricht prinzipiell auch vielen weiteren Definitionen, etwa: „Information ist der Teil einer Nachricht, der für den Empfänger neuartig ist." Wir werden uns später auch noch ausführlicher mit der Unterscheidung von Daten und Information beschäftigen.

Bitte beachten Sie, dass Sie sich nur in der Informatik und den Ingenieurdisziplinen auf diese Auslegung der Begriffe verlassen können – einige Autoren verwendet in der Soziologie etwa die Begriffe „Daten" und „Information" genau andersherum!

Auch Shannon nutzte – wie wir am Anfang des Kapitels – die Sprache als Experimentierfeld und legte Versuchspersonen Texte mit Lücken vor. Wenn dann zum Beispiel einzelne Buchstaben fehlten, stellte er fest, dass die häufigen Buchstaben von den allermeisten Menschen einfach ergänzt werden konnten. Bei den seltenen Buchstaben war die Ungewissheit deutlich größer. Seltene Buchstaben füllen demnach größere „Informationslücken", tragen mehr Information.

Shannon verknüpfte den Informationsbegriff mit dem aus der Physik gebräuchlichen Maß für Unordnung: der Entropie. Etwas Geordnetes, Übersichtliches ist einfach zu durchschauen und füllt daher deutlich weniger „Ungewissheit" als etwas Chaotisches. Verknüpft man diese Anschauung mit dem Zusammenhang zwischen Informationsgehalt und Wahrscheinlichkeit des Auftretens, kommt man auf eine Formel wie die von Shannon vorgeschlagene:

$$H(x) = \log_2\left(\frac{1}{p(x)}\right)$$

$H(x)$ steht dabei für die Entropie, den Informationsgehalt eines Ereignisses x – in unseren Beispielen meistens ein Zeichen einer Nachricht. $p(x)$ ist die Wahrscheinlichkeit dafür. Die Einheit für Entropie ist bit.

David Albert Huffman (1925–1999) war amerikanischer Elektrotechniker und beschäftigte sich intensiv mit Informationsübermittlung. Die heute noch nach ihm benannte Art der Codierung hat er 1952 als Seminararbeit für sein Promotionsstudium entwickelt.

Bei unseren Genkombinationen aus vier Zeichen kommt zum Beispiel das Zeichen „A" an einer bestimmten Stelle mit der Wahrscheinlichkeit 0,25 vor. Eingesetzt in die Shannon'sche Formel trägt es eine Information von 2 Bit. Voilà – genauso viele Bits benötigen wir auch, um es in einem Blockcode mit vier Zeichen binär darzustellen. Und so funktioniert die Formel mit allen Blockcodes, deren Anzahl an Zeichen eine Zweierpotenz darstellt, etwa 32, 64, 128 oder 256 wenn man davon ausgeht, dass alle Zeichen gleich häufig verwendet werden: Die Shannon'sche Entropie in „bit" ergibt den tatsächlichen Bedarf an „Bit".

Grund genug, auch für unsere unterschiedliche Häufigkeit von Zeichen in der deutschen Sprache die Entropie auszurechnen und mit den Bitlängen des ermittelten Huffman-Codes zu vergleichen. Die Tabelle unten macht das für den Code aus Abbildung 6.41. Die Bitlänge des Huffman-Codes für ein Zeichen in der letzten Spalte entspricht in den allermeisten Fällen der gerundeten Entropie. Die Fälle, für die das nicht gilt, sind violett gefärbt.

Wenn Sie annehmen, dass Sie 1.000.000 Zeichen mit den angegebenen Häufigkeiten codieren, kommen Sie mit dem Blockcode (offensichtlich) auf 5.000.000 Bit. Mit unserem Huffman-Code sparen Sie etwas und liegen nur noch bei 4.222.800 Bit.

x	p(x)	H(x)	Huffman-Code	Bits
_ (leer)	17,38	2,52	000	3
A	3,97	4,65	00100	5
B	1,50	6,06	111110	6
C	3,18	4,97	11110	5
D	3,90	4,68	00110	5
E	12,56	2,99	010	3
F	1,40	6,16	111001	6
G	2,22	5,49	001011	6
H	5,39	4,21	1100	4
I	6,55	3,93	0110	4
J	0,15	9,38	1111111110	10
K	0,86	6,86	0011110	7
L	3,19	4,97	01111	5
M	2,57	5,28	001010	6
N	7,74	3,69	100	3
O	1,74	5,84	001110	6
P	0,73	7,10	1110110	7
Q	0,03	11,70	11111111110	11
R	5,38	4,22	1101	4
S	5,33	4,23	1010	4
T	5,03	4,31	1011	4
U	3,24	4,95	01110	5
V	0,51	7,62	00111111	8
W	1,49	6,07	111000	6
X	0,03	11,70	11111111111	11
Y	0,22	8,83	111111110	9
Z	0,83	6,91	1111110	7
Ä	0,40	7,97	11111110	8
Ö	0,27	8,53	11101111	8
Ü	0,52	7,59	00111110	8
ß	0,39	8,00	11101110	8
.	1,30	6,27	111010	6

Indem Sie die Häufigkeiten mit der Shannon'schen Entropie multiplizieren, erhalten Sie so etwas wie die theoretische Untergrenze für einen optimalen Code. Diese gilt nur für die angenommenen Rahmenbedingungen – wie hier, dass wir jedes Zeichen einzeln codieren möchten. Für unsere Häufigkeiten kommen wir für die 1.000.000 Zeichen auf 4.178.261. Mit dem Huffman-Code sind wir also bereits sehr gut!

Resümee

In diesem Kapitel haben Sie einen ersten Einblick in die Informationstheorie genommen. Durch die Sichtweise, dass der Informationsgehalt seltener Ereignisse höher ist als der häufiger Ereignisse, konnten wir Nachrichten so geschickt codieren, dass diese zwar immer noch alle Informationen enthalten, jedoch deutlich weniger Speicherplatz in Anspruch nehmen.

Auch wenn wir unsere Experimente hauptsächlich mit Sprache durchgeführt haben, gelten die Zusammenhänge auch für andere Daten, etwa Bilder.

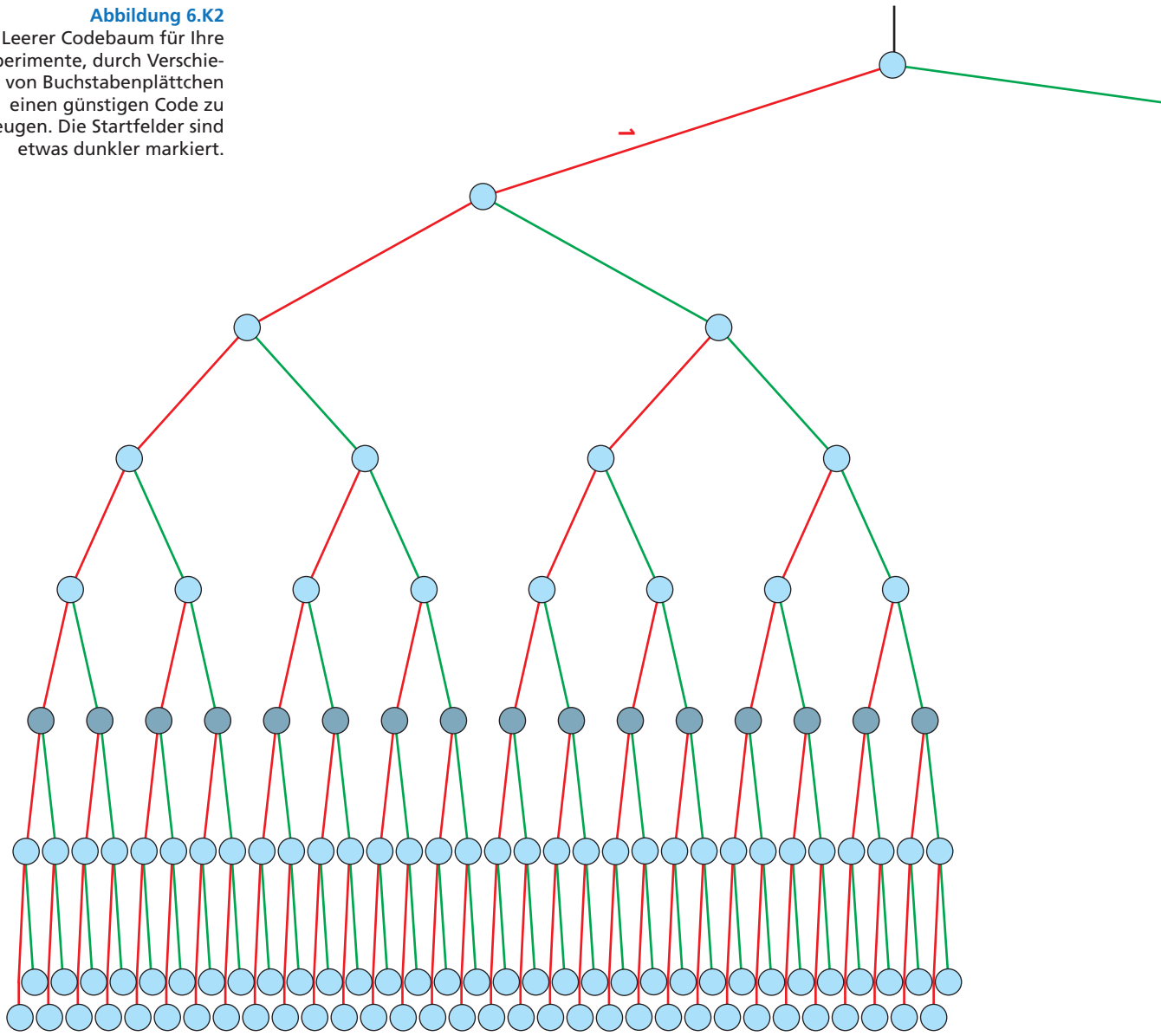


Abbildung 6.K1
Buchstabenplättchen. Der Einfachheit halber können Sie diese auch an den dünnen Linien als Quadrate ausschneiden – hier dreifach, falls Sie sich „verschneiden“.

Abbildung 6.K2
Leerer Codebaum für Ihre Experimente, durch Verschieben von Buchstabenplättchen einen günstigen Code zu erzeugen. Die Startfelder sind etwas dunkler markiert.

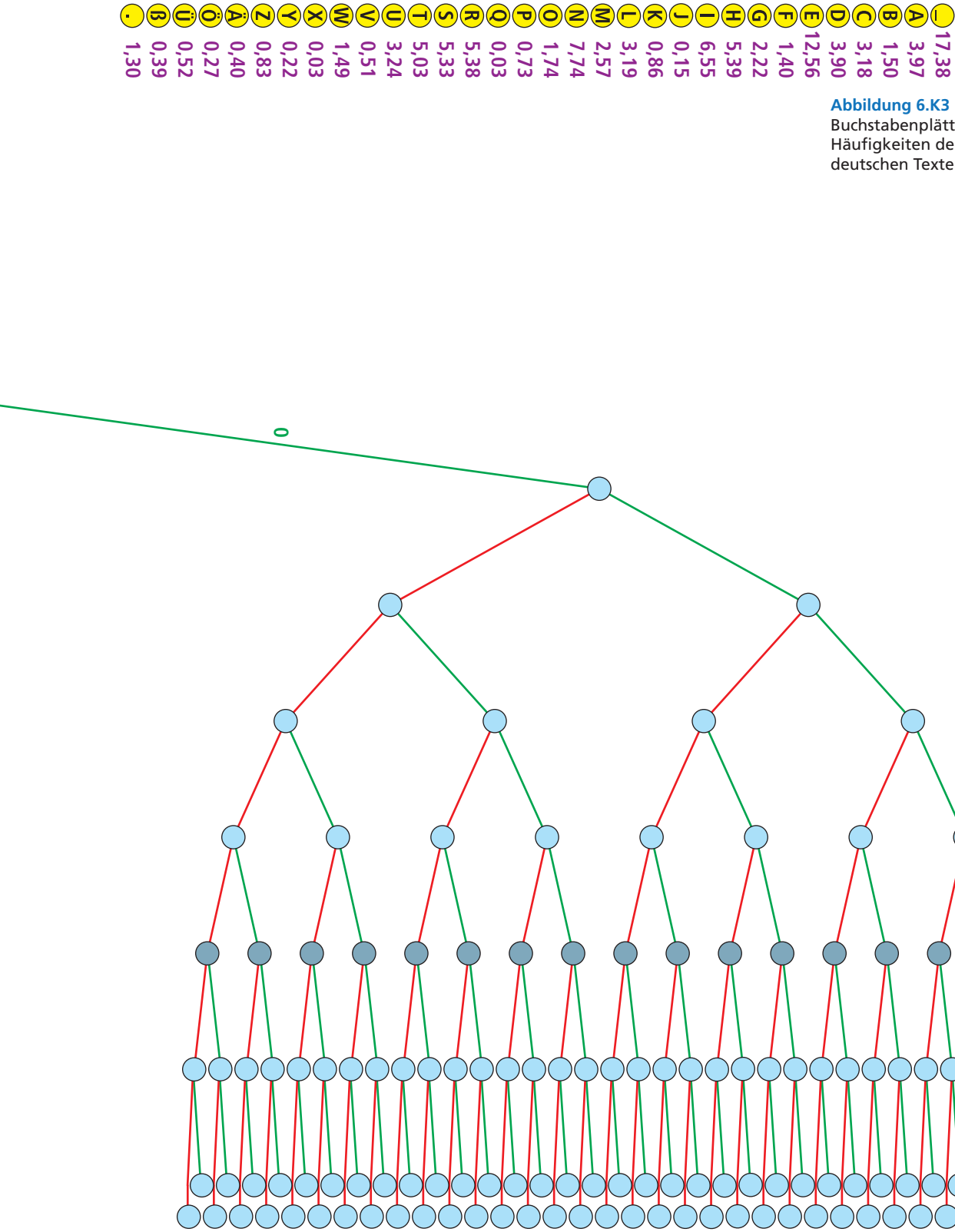


Abbildung 6.K3
Buchstabenplättchen mit Häufigkeiten der Zeichen in deutschen Texten (in Prozent)

7. Verluste gibt es doch immer!

Wenn man Verfahren wie den Huffman'schen Algorithmus zusammen mit der Burrows-Wheeler-Transformation geschickt anwendet, kann man Texte auf sehr wenige Daten schrumpfen (ca. 10 % bis 20 %). Das funktioniert leider nicht gleich effektiv mit multimedialen Daten, also Bildern, Tönen (Musik) oder Filmen. Außerdem sind solche Daten oft so immens groß, dass sich hier eine noch weitergehende Reduzierung der Datenmenge lohnt. Die Frage ist, wie man so etwas zustande bringt.

Hierzu schauen Sie sich bitte die Abbildungen 7.1 und 7.2 an.

Der Mensch als Faktor

Erkennen Sie, was dargestellt ist? Nein? Dann halten Sie das Buch einmal ein Stück weg und kneifen zusätzlich die Augen zusammen. Erkennen Sie es nun?

Abbildung 7.1
Was wird hier dargestellt?

Abbildung 7.2
Was wird hier dargestellt?

Anhand dieses Experiments sehen wir, dass sich farbige Bilder offenbar ähnlich wie ein Fax als Raster gut darstellen lassen: Den Elefanten erkennen wir, obwohl er nur aus groben Bildpunkten besteht, von denen jeder eine bestimmte Farbe aufweist.

Während Abbildung 7.1 ziemlich sicher von allen erkannt wird, scheiden sich bei Abbildung 7.2 die Geister: Obwohl das Bild identisch aufgelöst ist wie der Elefant, erkennen hier einige Menschen nur mit Mühe ein Gebäude, während andere sofort das Taj Mahal vor Augen haben. Offenbar spielt hier die Erfahrung eine entscheidende Rolle.

Sehen ist also mehr als eine exakte Abbildung der Realität: Diese besteht ja eigentlich nur aus bunt zusammengewürfelten Kästchen. Je weiter wir uns entfernen und je mehr wir die Sicht künstlich verschlechtern, desto mehr denkt unser Gehirn, das Bild verbessern zu müssen. Aus unserer Erfahrung heraus, wie ein Elefant auszusehen hat, ergänzt es die fehlenden Teile – und wer schon einmal das Glück hatte, vor dem vielleicht schönsten Gebäude der Welt zu stehen, wird auch dieses Bild immer wieder aus dem Gedächtnis ergänzen.

Selbstverständlich ist es deutlich sparsamer, so ein grobes Bild zu speichern oder zu übertragen als ein sehr feinkörniges. In diesem Fall wurden immer 80 · 80, also 6400 Bildpunkte zu einem großen „Klotz“ zusammengefasst. Man sieht also, dass Elefant und Taj auch mit unter 0,1 % der Information noch zu erkennen ist.

Abbildung 7.3
Elefant

Abbildung 7.4
Taj Mahal

Allerdings: Auch wenn die Information „Elefant“ übertragen wird – schön ist ein solches Bild nicht! Niemand kommt auf die Idee, Bilder nur noch so zu übertragen, dass man zum Betrachten die Augen zusammenkneifen muss ...

Machen wir ein weiteres Experiment: Die Abbildungen 7.3 bis 7.6 zeigen Elefant und Taj in einer genaueren Weise. Vergleichen Sie die rechte und linke Version genau. Fällt Ihnen daran etwas Bestimmtes auf? Können Sie entscheiden, welche „besser“ ist?

Sehen alle Bilder gut aus? Sowohl die Konturen als auch die Farben sind – fast – in Ordnung. Selbst Details wie die Haare oder Blätter sind gut zu erkennen. Trotzdem sind die linken Bilder sehr viel gröber als die rechten. Besonders gut sieht man das bei der Umrandung des Wasserlaufs in Abbildung 7.4. Schauen Sie nochmals ganz genau hin, falls es Ihnen bisher nicht aufgefallen ist.

Abbildung 7.5
Elefant

Abbildung 7.6
Taj Mahal

Abbildung 7.7
Vergrößerung eines Ausschnitts

Erst mit einer Lupe oder in der Vergrößerung (Abbildung 7.7 Mitte) erkennen Sie genau, dass Bereiche, die wir vermeintlich detailliert wahrgenommen hatten, gar nicht wirklich detailliert sind: Während die Konturen, also die Helligkeitsunterschiede, im Bild sehr fein und wie im Original rechts dargestellt werden, sind die Farben im gleichen groben Raster gesetzt wie im Bild links! In Fachsprache nennt man solche Blockstrukturen, die man ja eigentlich nicht sehen möchte, „Artefakte".

Wenn man jetzt noch die Information berücksichtigt, dass in einem normalen, elektronischen Bild die Speicherung der Grauwerte, also der Helligkeitsinformationen, nur etwa 1/3 ausmacht und 2/3 für die Farbe „ausgegeben" werden, lässt sich daraus eine exzellente Einsparung ableiten:

Offenbar nimmt das Auge Farbe nicht so detailliert wahr wie Helligkeit! Das haben wir nun experimentell nachgewiesen! Was spricht also dagegen, die Helligkeitsinformation unangetastet zu lassen, die Farbinformation hingegen drastisch zu vergröbern?

Unter Verlusten

Genau das passiert zum Beispiel beim Farbfernsehen: Innerhalb bewegter Bilder ist unser Auge noch viel weniger in der Lage, Farben detailreich wahrzunehmen. Daher wird die recht präzise Information, die bereits Schwarz-Weiß-Fernseher bekamen, gemischt mit einer sehr groben Farbinformation. Dem Genuss eines Filmes ist das jedoch nicht abträglich, auch wenn tatsächlich etwas fehlt!

Am Anfang des Kapitels haben wir die Datenmenge z. B. von Text reduziert, konnten jedoch immer wieder das Originaldokument aus den reduzierten Daten restaurieren. Das funktioniert hier nicht mehr. Wenn wir ein Bild aus dem Fernseher stark vergrößern, tritt der Bluff zutage. Genau wie bei unserem Taj-Bild: Wenn man sich für die echte Farbe der Wassereinfassung interessiert, kann man das Bild bearbeiten, so viel man möchte, man wird diese nicht wiederherstellen können.

JPEG ist eigentlich die Abkürzung für ein Gremium, das sich 1986 gebildet hat, um Verfahren zur Bildübertragung über Telekommunikationsleitungen zu erstellen: „Joint Photographic Experts Group". 1992 brachte es dann das gleichnamige komprimierte Bildformat heraus. „.jpg" ist die davon abgeleitete Endung für das entsprechende Dateiformat. JPEG ist das wohl bekannteste verlustbehaftete Bildformat.

Das Gleiche ist übrigens ebenfalls den Ottern vom Kapiteleingangsbild widerfahren: Auch hier wurden große Blocks der Farbinformation zusammengefasst – mit Absicht so deutlich, dass man es bemerkt. Wenn Sie möchten, spielen Sie das mit eigenen Bildern und einem beliebigen Programm zur Bildverarbeitung durch. Dazu müssen Sie nur die Farben nach Lab wandeln lassen. Der L-Kanal bezeichnet dabei den Helligkeitswert, a und b stehen für die Farbkanäle. Abbildung 7.8 zeigt, wie der Elefant in diese Bestandteile zerlegt und nach der Rasterung der Farbkanäle wieder zusammengesetzt wird.

Abbildung 7.8
Das Bild wird in die einzelnen Farbkanäle zerlegt. Danach werden die farbigen Kanäle grob gerastert. Im wieder zusammengesetzten Bild merkt man das kaum.

Abbildung 7.9
Taj Mahal mit den gleichen Informationen wie bei Abbildung 7.4, aber mit durch Unschärfe kaschierten Artefakten

Um den Effekt zu „vertuschen" kann man vor dem Zusammensetzen die groben Farbkanäle auch noch unscharf machen. Dadurch vermeidet man Artefakte. Abbildung 7.9 zeigt zum Vergleich das Taj-Bild auf diese Weise.

In der Praxis werden noch viele weitere Eigenschaften unseres Sehvermögens verwendet, um scheinbar irrelevante Informationen einfach wegzulassen. Ein paar davon können Sie im Folgenden „am eigenen Leib" ausprobieren:

Vergleichen Sie die beiden blauen Streifen, dann die beiden grünen Streifen in Abbildung 7.10 aus normaler Leseentfernung. Was fällt auf?

Beim grünen unteren Streifen ist durch genaues Hinschauen erkennbar, dass die Farbe abgestuft ist: Statt eines glatten Verlaufs (was bei Computerbildern normalerweise 256 unterschiedliche Schattierungen impliziert) wurden nur 20 verschiedene Grüntöne aneinander gehängt. Beim unteren blauen Streifen sieht man auf Anhieb keinen Unterschied zum oberen. Geht man sehr nah an das Buch, erkennt man allerdings auch

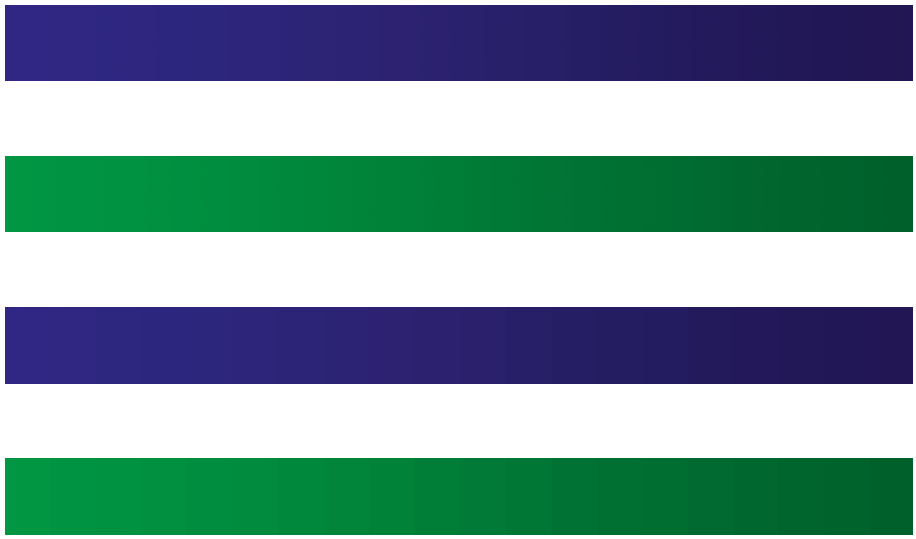


Abbildung 7.10
Farbverläufe

Abbildung 7.11
Elefantenbild mit nur 32 unterschiedlichen Farben

hier sehr feine Abstufungen, denn auch der untere blaue Streifen besteht lediglich aus 20 verschiedenen Tönen.

Die Empfindlichkeit des Auges für die Farbe Grün ist höher als für Rot und viel höher als für Blau. Daher benötigt man für einen blauen Gegenstand wesentlich weniger unterschiedliche Schattierungen der Farbe als z. B. für einen grünen. Bilder vom Ozean sind also nicht nur schön, sondern sie können auch in weniger Speicher abgelegt werden als Dschungelbilder. Auf diese Weise sieht auch Abbildung 7.11 noch einigermaßen ansehnlich aus – obwohl es aus nur noch 32 unterschiedlichen Farben zusammengesetzt ist.

Zwischen der Abbildung der Wirklichkeit auf unserer Netzhaut und dem, was unser Geist daraus macht, liegen also offenbar Welten. So werden die meisten Informationen verworfen und nur bestimmte, für das Wiedererkennen relevante Merkmale registriert und abgespeichert. Dazu gehören etwa Bildkanten. Obwohl diese nur sehr wenig Platz beanspruchen, verlässt sich unser Gehirn fast komplett auf sie, wenn es um das Erkennen von Objekten geht.

Wahrscheinlich sind wir auch nur auf diese Art in der Lage, uns die gesammelten Erinnerungen eines Lebens zu merken.

Einen visuellen „Beweis“ hierfür liefere ich in Abbildung 7.12: Die gezeichneten Figuren bestehen aus primitiven und nicht besonders akkurat gezeichneten Konturen, die noch dazu nicht im korrekten Größenverhältnis zueinander stehen – trotzdem erkennen wir die dargestellten Tiere auf Anhieb.

Abbildung 7.12
Strich-Zoo

Eine Anleitung zum Zeichnen ähnlicher Tierchen finden Sie bei Gerolf Steiner: „Tierzeichnungen in Kürzeln“; ISBN 3-8274-1702-3

Film ab

Sie haben nun Methoden kennen gelernt, wie Bilder scheinbar gleich bleiben, aber dennoch durch Weglassen von normalerweise sowieso nicht sichtbarer Information deutlich sparsamer gespeichert werden können. JPEG & Co. nutzen dies aus.

Ein Film besteht aus sehr vielen einzelnen Bildern, die in schneller Folge gezeigt werden. Selbstverständlich spart man daher bei Blu-Ray, DVD und digitalem Fernsehen mit den gleichen Verfahren, wie sie auch bei einzelnen Bildern genutzt werden. Hier geht es allerdings noch besser: Betrachten Sie den Filmstreifen am Buchrand mit seinen vielen Einzelbildern. Abbildung 7.13 zeigt zur Verdeutlichung einen Ausschnitt mit zwei aufeinanderfolgenden Bildern in groß. Erkennen Sie den Unterschied?

Wir sehen bereits, dass sich die Bahn bewegt hat, während die Umgebung gleich geblieben ist. Um jedoch ganz sicher zu sein, schauen wir das Differenzbild in Abbildung 7.14 an, in dem die Unterschiede zwischen den beiden Einzelbildern dargestellt sind. Sie erkennen recht gut die Bahn mit Insassen sowie den Schatten.

Bei dem sogenannten MPEG-Standard, wie er für praktisch alle modernen Filmaufnahmen und vor allem die Übertragung genutzt wird, folgen auf ein vollständiges Bild – das sogenannte Keyframe – immer nur die wesentlichen Abweichungen. Wenn etwa das linke Bild aus Abbildung 7.13 als Keyframe genutzt würde, könnte danach eine Folge reiner Differenzbilder kommen, von denen Abbildung 7.14 das erste darstellt. Es ist leicht einsichtig, dass dieses Bild mit deutlich weniger Aufwand übertragbar ist.

Das Bild mit dem Fahrzeug demonstriert noch eine weitergehende Möglichkeit für Einsparungen: Wie in den meisten Filmaufnahmen ändern sich selbst bewegte Objekte nicht komplett – sie werden quasi nur auf der Filmebene verschoben. In Abbildung 7.15 wurde der Ausschnitt mit der sich bewegenden Bahn als komplettes Rechteck verschoben, so dass er mit dem folgenden Bild nahezu identisch ist. Beim „echten" MPEG-Verfahren könnte man diesen Ausschnitt auch noch drehen und vergrößern oder verkleinern. Obwohl wir die Möglichkeiten hier also gar nicht komplett ausgenutzt haben, sehen Sie bereits am rechten Differenzbild, wie wenige Unterschiede nun noch bleiben.

Auch die noch deutlich sichtbaren Schatten könnten auf diese Weise einfach verschoben werden – selbstverständlich funktioniert das Verfahren mit praktisch beliebig vielen sich verändernden Bildausschnitten. Denken Sie zum Beispiel an einen Tierfilm, in dem in der unteren Bildhälfte zwei Füchse durch das Gras streifen, während oben ein Vogel fliegt.

Für einen Computer ist es sehr aufwendig zu ermitteln, welche Ausschnitte für eine Verschiebung in Frage kommen und wie diese sich von Bild zu Bild verändern. Ist das dann aber erst einmal passiert, lässt sich die komprimierte Bildfolge wieder sehr schnell zusammensetzen. Auf diese Weise muss nur bei der Produktion einmalig viel Zeit eingesetzt werden, um die maximale Kompression zu erreichen. Beim Anschauen auf dem Fernseher oder dem Smartphone bedarf es dann nicht mehr viel Rechenleistung.

Wichtig ist dabei aber immer, dass kleine Veränderungen zugunsten von Speicherplatz in Kauf genommen werden, denn unser Auge nimmt sie in der fließenden Bewegung sowieso nicht wahr. Auch hier dient das bewusste Weglassen von Informationen also der schnelleren Übertragung. Als Filmproduzent können Sie das auch ganz genau

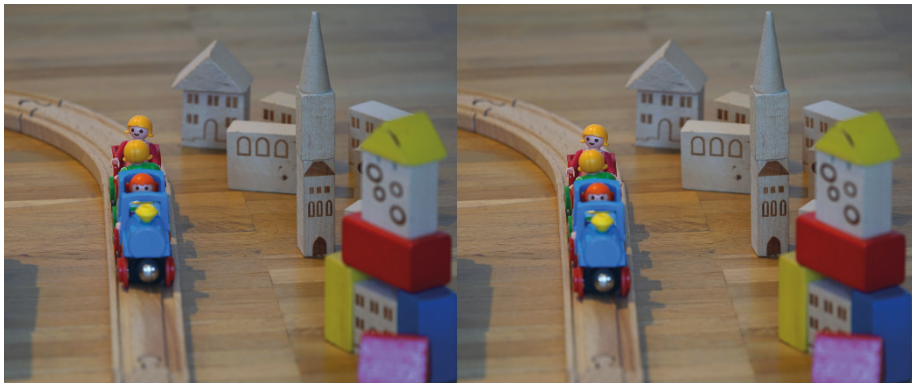


Abbildung 7.13
Aufeinanderfolgende Bildausschnitte des Films

Abbildung 7.14
Differenzbild

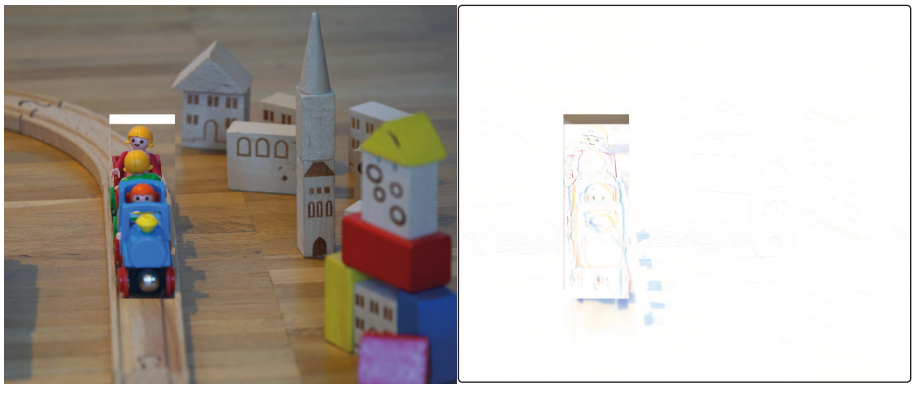


Abbildung 7.15
Verschieben eines Blocks vor dem Rechnen der Differenz

steuern und zum Beispiel in der Version für Smartphones mehr Details weglassen als in der Version für hochauflösende Fernsehgeräte.

Ton läuft

Auch wenn man das in einem Buch nicht so gut vermitteln kann, dürfen Sie sich die Komprimierung von Musik und MP3 sehr ähnlich zu den entsprechenden Verfahren bei Bildern vorstellen.

Machen Sie ein kleines Experiment: Stellen Sie einen Freund vor irgendein lautes Gerät, zum Beispiel einen Küchenmixer, gehen Sie zehn Schritte weg und lassen Sie ihn im Flüsterton etwas erzählen. Hören Sie etwas anderes als den Küchenmixer?

Sicherlich nicht – auch wenn die Informationen eigentlich in Ihrem Ohr ankommen. So kann die Polizei aus Abhörbändern immer wieder wichtige Informationen ziehen, indem elektronisch die Hintergrundgeräusche eliminiert werden.

Das durchschnittliche Ohr ist dazu jedoch selbst nicht in der Lage – die geflüsterte Sprache geht genauso im Mixerlärm unter wie ein einzelnes Zupfen an einer Geige in einem vollen Orchester (das allerdings geschulte Musiker durchaus hören).

Genau hier setzt die Audiokomprimierung ein: Alles, was der durchschnittliche Mensch sowieso nicht wahrnimmt, wird einfach weggelassen, der notwendige Speicherplatz für diese Information eingespart – ganz analog zu den hier vorgestellten Verfahren zur Bildkomprimierung.

Gleichzeitig kann man sich vorstellen, dass sich linker und rechter Kanal eines Stereo-Signals nicht komplett unterscheiden müssen, um einen räumlichen Eindruck zu geben. Hier werden daher lediglich die Differenzen extrahiert und übertragen – ähnlich dem Verfahren bei Filmen.

Was steckt dahinter?

Nach der Codierung oder verlustfreien Komprimierung haben Sie in diesem Kapitel nun „echte“ Komprimierungsverfahren entdeckt: Die ursprüngliche Information bleibt hierbei nicht intakt, sondern es werden verzichtbare Teile entfernt. Welche das sind, hängt sehr stark von unserem Bild des Menschen ab, der die Informationen in Form von Bildern, Tönen oder anderen Medien konsumieren möchte.

MP3
Das ist die abgekürzte Form einer anderen Abkürzung: „MPEG-1 Audio Layer 3“. MPEG ist dabei die „Moving Picture Experts Group“ bzw. deren gleichnamiger Standard für die komprimierte Übermittlung bewegter Bilder. Um auch den zugehörigen Ton platzsparend abzuspeichern und zu verschicken, wurde 1987 der entsprechende Audio-Standard am Fraunhofer-Institut für Integrierte Schaltungen in Erlangen entwickelt. Heute ist MP3 längst ein eigenständiges Format, das vor allem mit portabler Musik in Verbindung gebracht wird.

Komprimierung

Bei der Komprimierung wird eine Datenmenge reduziert, indem zwischen relevanten und irrelevanten Informationen unterschieden wird und lediglich die relevanten Informationen beibehalten werden. Diese Unterscheidung wird oft im Medienbereich anhand physiologischer Studien vorgenommen, nach denen bestimmte Informationsteile nicht oder nur von wenigen Menschen wahrgenommen werden können.

Auch wenn wir es nicht merken – Datenkomprimierung wird quasi in allen Bereichen um uns herum vorgenommen. Überlegen Sie, wo in folgenden Anwendungen Codierung, wo Komprimierung steckt:

- telefonieren
- Postleitzahl statt Ort
- digitales Fernsehen
- Bilder mit der Digitalkamera machen
- eine CD hören
- eine DVD oder Blu-Ray ansehen
- sich an den letzten Urlaub erinnern

- Wartemusik am Telefon klingt immer etwas quäkig – weil die Telefongesellschaften Geld sparen, indem sie nur den Frequenzbereich der Sprache vernünftig übermitteln. Das ist eindeutig eine Komprimierung!
- Postleitzahlen bilden bestimmte Bereiche der Landkarte ab. Man kann von der Postleitzahl wieder zurück auf die Stadt schließen, also eine Codierung.
- Fernsehen, DVD, MP3 und viele andere Medien sind nur möglich, indem die Datenmenge auf einen Bruchteil reduziert wird. Eindeutig ein Fall von Komprimierung.
- Fast alle Digitalkameras speichern Bilder im Format JPEG ab. Dieses Format nutzt genau die oben im Kapitel beschriebenen Eigenschaften unseres Sehens, um mehr Bilder auf weniger Speicher unterzubringen. Allerdings können viele Digitalkameras auch Bilder in einem Rohformat speichern (RAW), das keine Komprimierung nutzt, so dass die Bilder auch noch problemlos weiterverarbeitet werden können. Oft werden diese Bilder dann codiert, um Speicherplatz zu sparen.
- Die CD ist ein inzwischen recht altes Medium. Bei ihrer Entwicklung war die Leistungsfähigkeit von Computern noch nicht sehr hoch: Der CD-Player hätte mehrere Stunden benötigt, ein im heutigen MP3-Verfahren komprimiertes Stück von einigen Minuten zu dekomprimieren und abzuspielen. Da man aber natürlich einen Titel von drei Minuten Länge auch gerne in drei Minuten abgespielt haben möchte, enthält die CD die digitalen Klänge unkomprimiert.
- Im Gegensatz dazu sind sowohl die Bild- als auch Tondaten auf einer DVD oder Blu-Ray einerseits komprimiert, andererseits aber auch mit zusätzlichen Daten zur Fehlerkorrektur versehen. Lesen Sie weiter hinten im Buch mehr darüber!
- Der letzte Punkt ist etwas Besonderes: Hier geht es zur Abwechslung nicht um Technik, sondern um das menschliche Gedächtnis. Und tatsächlich kann man davon ausgehen, dass Erinnerungen über die Zeit immer weiter komprimiert werden: Auch sie werden unterbewusst in wichtige und unwichtige Informationen aufgeteilt. Über die Zeit merken wir uns dann nur wichtige Anteile und vergessen die unwichtigen, um wieder Platz für neue Erkenntnisse und Eindrücke zu schaffen. Diesen Vorgang nennt man Abstraktion. Im Wesentlichen geht es aber genau um das Gleiche wie bei der Komprimierung von Musikstücken – der verfügbare Speicherplatz ist für die gesamte Datenflut nicht ausreichend, also versuchen wir, die wichtigsten Informationen zu behalten.

Warum ist diese Erkenntnis für die Informatik so relevant? Ob es um Daten, um Algorithmen oder um Erinnerungen geht: In jedem Fall muss in vielen Situationen Unwichtiges und Wichtiges voneinander getrennt werden. Denken Sie an das Kapitel über Routenplaner und wie wir an die Lösung des Problems herangegangen sind: Hier war Abstraktion entscheidend wichtig.

Auch in diesem Fall ist der Mensch mit seinen Eigenschaften, seinen Stärken und seinen Schwächen die Grundlage für die Trennung zwischen „wichtig“ und „unwichtig“. Er spielt in der Informatik die Hauptrolle!

Resümee

Die Betrachtung der menschlichen Physiologie ermöglicht es, Bilder, Filme und Musik auf wenige Prozente der ursprünglichen Datenmenge zu schrumpfen, ohne dass wir überhaupt einen nennenswerten Unterschied feststellen.

Sie haben allerdings bei den vergrößerten Kontrollaufnahmen immer auch gesehen, dass dies nur gilt, wenn man nicht bewusst ganz genau hinschaut und das Material auch nicht weiter bearbeitet. Bei einer Reduzierung der Datenmenge ist daher immer vorsichtig abzuwägen, ob bestimmte Anteile wirklich verzichtbar sind oder ob das nur für die momentane Anwendung oder Zielgruppe gilt.

Diese Erkenntnis ist dann auch auf die menschliche Erinnerung und insbesondere auf die Vorgehensweise beim Entwickeln neuer Algorithmen übertragbar: Die Kunst ist, genau das richtige Maß zu finden.

8. Erkennungsdienst

Wenn man dem Prinzip „divide et impera“ folgt, zerlegt man große Aufgabenstellungen in kleinere, handhabbare Teile. Bisher hatten wir dabei die Größe der einzelnen Teile nicht weiter beachtet – um dieses Detail wollen wir uns jetzt mit Hilfe eines Spiels kümmern.

Darf es eine Frage mehr sein?

Die Regeln von „Erkennungsdienst“ sind sehr einfach: Finden Sie eine Mitspielerin oder einen Mitspieler. Ein Startspieler wird bestimmt und sucht sich frei aus, welche Person er oder sie sein möchte. Dabei muss man natürlich dem eigenen Geschlecht nicht treu bleiben. Die Gegenpartei versucht nun, durch Fragen herauszufinden, um welche Person es sich handelt. Die Fragen dürfen allerdings nur mit „Ja“ oder „Nein“ beantwortet werden. Pro Frage wird ein Minuspunkt notiert. Man kann jederzeit eine Vermutung der Art „du bist Sabrina“ äußern. Ist diese korrekt, ist die Runde zu Ende. Ist sie falsch, werden zwei Minuspunkte notiert.

Jetzt wechseln die Rollen. Nach einer vorher ausgemachten Zahl von Runden – zum Beispiel zehn – gewinnen Sie, wenn Sie weniger Minuspunkte angesammelt haben.

Abbildung 8.1 zeigt die 16 Identitäten, die bei Spielversion A ausgewählt werden können. Wenn Sie sichergehen möchten, dass Personen nicht aufgrund bekannter Vorlieben (zum Beispiel für blonde Haare) vorzeitig entdeckt werden, können Sie die entsprechenden Karten von Kopiervorlage 8.K1 ausschneiden und Ihre wechselnden Identitäten verdeckt ziehen.

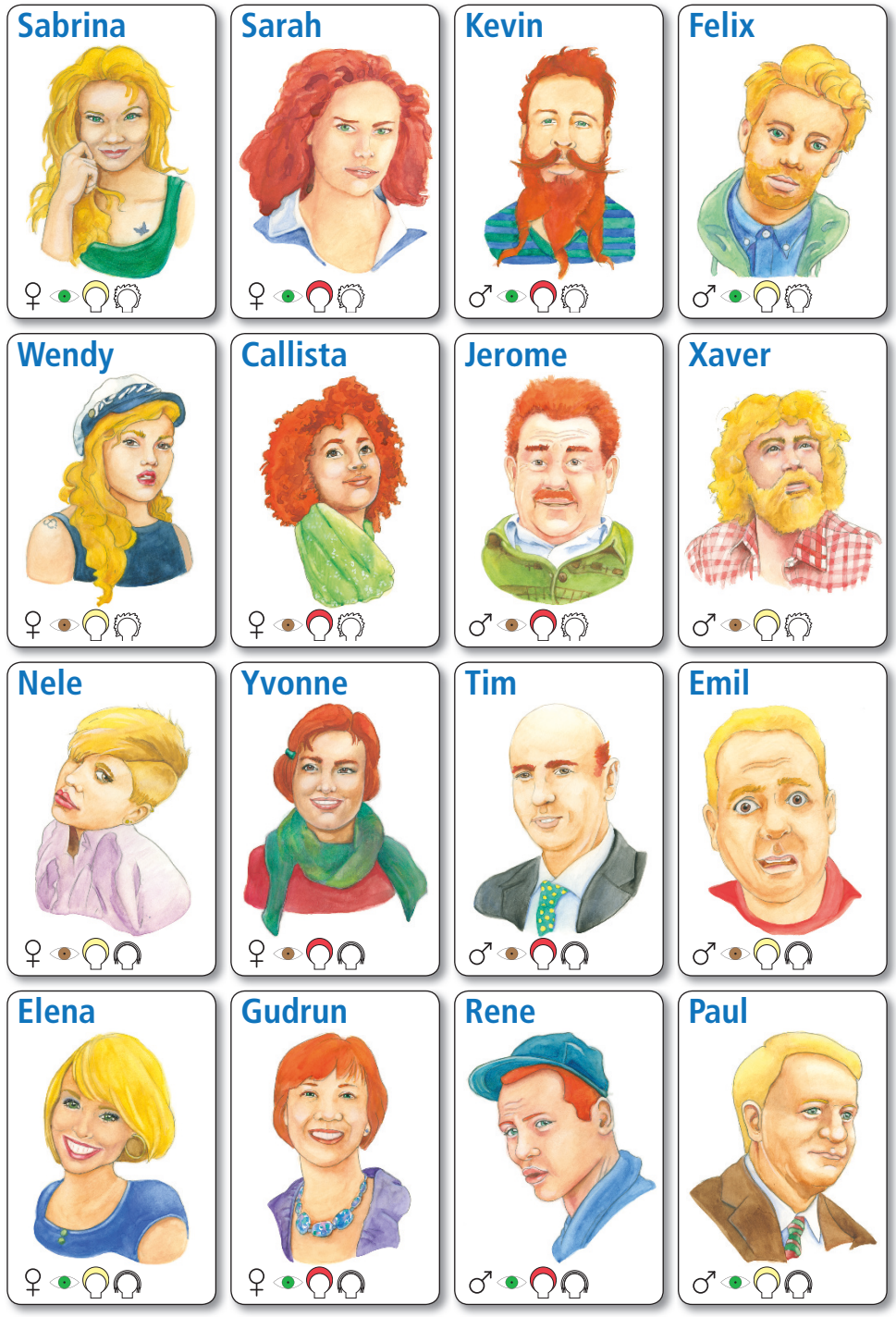
Als Hilfe sind vier Eigenschaften unterschiedlicher Ausprägung nochmals als Symbol unter den Bildern angegeben:

- Ist es eine Frau ♀ oder ein Mann ♂ ?
- Ist die Augenfarbe blau, braun oder grün ?
- Sind die Haare rot, blond, braun oder schwarz ?
- Sind die Haare glatt oder gelockt ?

Als kleine zusätzliche Hilfe tragen übrigens alle unsere gelockten Männer gleichzeitig einen Bart.

Spielen Sie Version A. Versuchen Sie auch einmal bewusst nur nach den vorgegebenen Eigenschaften zu fragen. Ist das Spiel interessant? Wer gewinnt?

Abbildung 8.1
Erkennungsdienst, Version A



Ich gehe davon aus, das Spiel war sehr langweilig, da niemand gewinnen konnte. Wahrscheinlich haben Sie immer beide genau vier Fragen benötigt, um eine Person zu identifizieren, und damit immer exakt vier Minuspunkte kassiert.

Warum ist das so?

Bei den zur Verfügung stehenden Identitäten liegt die Spielstrategie quasi auf der Hand: Es gibt genau acht Frauen und acht Männer, acht mit grünen Augen und acht mit braunen Augen, acht Rothaarige und acht Blonde, acht mit glatten Haaren, acht mit Locken. Vielleicht ist Ihnen auch schon aufgefallen, dass diese in der Abbildung 8.1 auch noch sehr geordnet sind – links und rechts, oben und unten bzw. innen und außen.

In den vorangegangenen Kapiteln haben wir unsere Strategien bereits häufig mit graphischen Mitteln verdeutlicht. Das wollen wir hier wieder machen. Abbildung 8.2 zeigt die Strategie in einer Baumstruktur, wie wir sie ebenfalls bereits kennen gelernt haben. Man fängt mit der ganz oben symbolisierten Frage an. Lautet die Antwort „Nein", folgt man dem linken Pfad, bei „Ja" dem rechten und stellt die Frage dort. Die Abbildung 8.2 zeigt es: Man kommt mit der immer gleichen (und praktisch beliebigen) Abfolge von Fragen immer mit vier Fragen zum Ziel. Eine bessere Vorgehensweise gibt es nicht – und man muss sich auch sehr anstrengen, eine ungünstigere zu finden.

Die Visualisierung wird auch Entscheidungsbaum genannt, weil sie eine Vorgehensweise auf Basis von Entscheidungen darstellt – hier abhängig von der jeweiligen gegnerischen Antwort „Nein" oder „Ja". Es handelt sich um eine vollständige Beschreibung der Strategie des aktiven Spielers.

Der Entscheidungsbaum zeigt es übersichtlich: Auf jeder Ebene finden sich die gleichen Symbole – mit der immer identischen Abfolge von vier Fragen identifiziert man sicher jede Person. Versuchen Sie, einen entsprechenden Baum für eine beliebige andere Reihenfolge aufzustellen. Im Resultat führt auch dieser in vier Schritten zum korrekten Ergebnis. Unsere Diagnose lautet also immer noch: Langweilig!

Testen Sie nun mit identischen Spielregeln Version B in Abbildung 8.3. Hier sind nur noch Frauen auswählbar. Was hat das für Auswirkungen auf Ihre Strategie? Versuchen Sie, diese in einem neuen Entscheidungsbaum darzustellen!

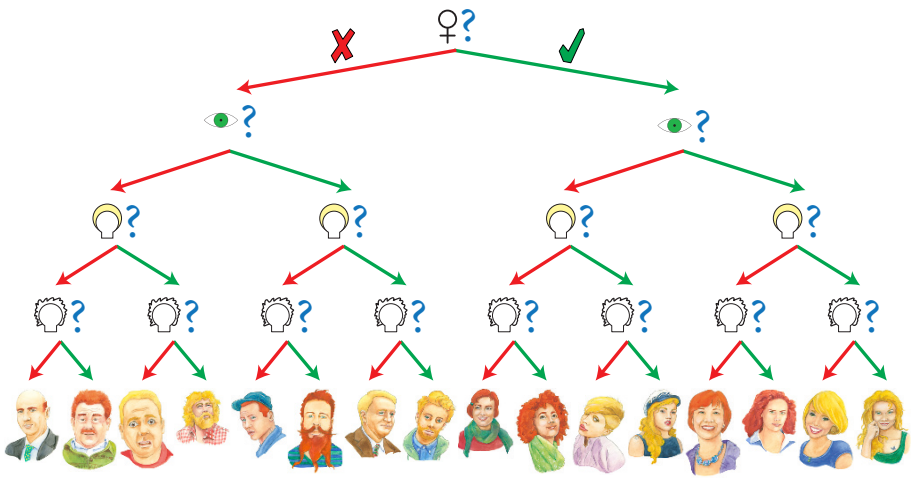


Abbildung 8.2
Entscheidungsbaum für Version A

Abbildung 8.3
Erkennungsdienst, Version B

Sie können sich nun sparen, nach dem Geschlecht zu fragen. Dafür gibt es jetzt vier unterschiedliche Haarfarben. Eventuell führt das zu einer Strategie, wie sie in Abbildung 8.4 als Entscheidungsbaum dargestellt ist, mit der zunächst die Haarfarbe bestimmt wird und dann die weiteren Eigenschaften.

Strategisch gewinnen

Das Spiel ist nun also bereits interessanter, da man in einigen Fällen die Person bereits nach drei Fragen identifiziert hat, das aber dadurch erkauft, dass man in anderen Fällen sogar fünf Fragen benötigt. Je nach Strategie und Auswahl einer Identität kann man nun also gewinnen oder verlieren.

Wenn man wüsste, dass der Spielpartner nach der abgebildeten Strategie spielte, wäre es sinnvoll, eine blonde oder rothaarige Identität anzunehmen, weil diese erst nach fünf Fragen gefunden würde. Mit dieser Spielvariante sind aber tatsächlich etliche Strategien möglich, die sich auch voneinander unterscheiden.

Versuchen Sie zur Übung, auch für diese Version eine Strategie zu finden, die in jedem Fall genau vier Fragen benötigt, um eine Person zu identifizieren, egal welche.

Sicherlich sind Sie auf den entscheidenden „Trick“ gekommen: Die Regeln besagen lediglich, dass die Fragen mit „Ja“ oder „Nein“ zu beantworten sind. Die Komplexität der Fragen wird nicht eingeschränkt. Daher können Sie die Runde mit der Frage „Hast du braune oder schwarze Haare?“ einleiten. Abbildung 8.5 zeigt die entsprechende Strategie als Entscheidungsbaum.

Spielen Sie nun probehalber die beiden Strategien gegeneinander! Welche gewinnt auf Dauer?

In der Regel sollten Sie mit der Strategie nach Abbildung 8.5 besser fahren: Wenn wir annehmen, dass alle Personen mit der gleichen Wahrscheinlichkeit gewählt werden,

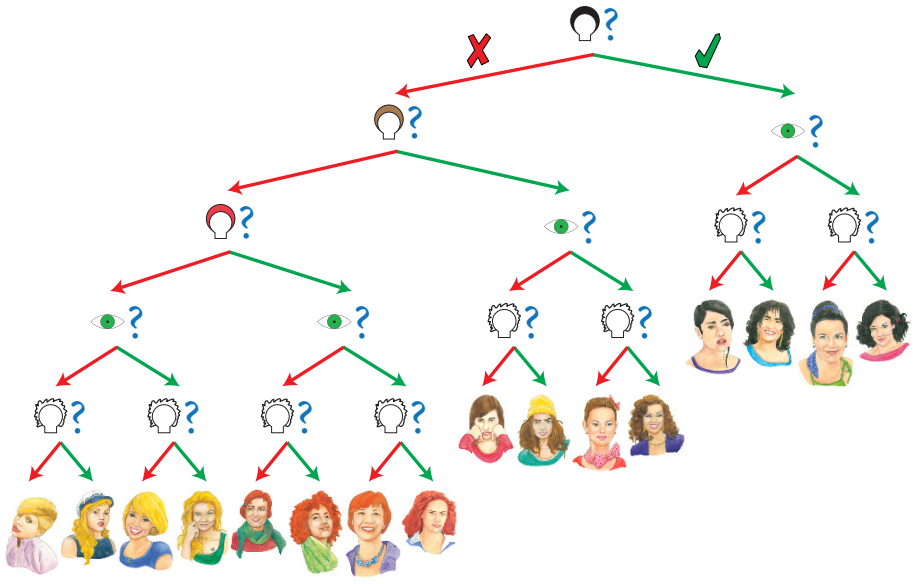
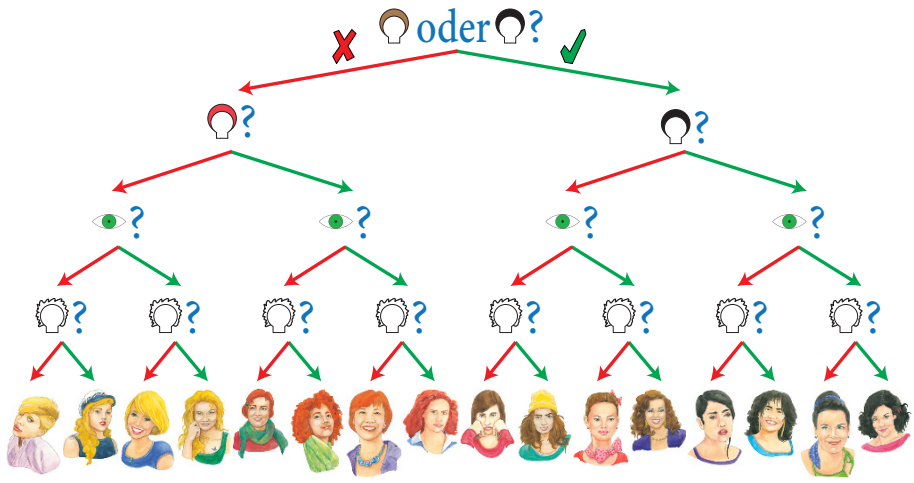


Abbildung 8.4
Entscheidungsbaum für Version B mit einer Strategie, die je nach Identität mit unterschiedlich vielen Fragen zum Ziel führt.

Abbildung 8.5
Entscheidungsbaum für Version B mit einer Strategie, die in jeder Situation mit vier Fragen auskommt.



ist die Strategie nach Abbildung 8.4 in vier Fällen besser, in vier Fällen gleich stark und in acht Fällen schlechter!

Steckt dahiner eventuell eine Regel? Testen Sie das mit Version C des Spiels, die in Abbildung 8.7 fast die gesamte nächste Doppelseite in Anspruch nimmt.

Finden Sie eine Strategie, die in den meisten Fällen die Nase vorne hat, und zeichnen Sie einen Entscheidungsbaum dafür.

Treten Sie gerne versuchsweise mit der Strategie nach Abbildung 8.6 an! Sie sollten damit auf Dauer zumindest unentschieden spielen!

Um weiter konsequent zu testen, steht Ihnen in den Kopiervorlagen 8.K1 und 8.K2 ein vollständiger Satz mit 96 Personen zur Verfügung, in dem nicht nur alle Kombinationen der bereits besprochenen Eigenschaften vertreten sind, sondern zusätzlich noch Brillenträger identifiziert sind. Der ergänzende Bastelbogen hat sogar noch eine weitere Eigenschaft „Sommersprossen", die den Satz auf 192 Identitäten erweitert. Mischen Sie den Kartensatz und spielen mit 32 zufällig gewählten Identitäten. Mit einem der erweiterten Sätze können Sie sogar ein Profispiel mit 64 Identitäten wagen, zusammen mit den Sommersprossen auch 128, was aber bereits sehr unübersichtlich wird und wirklich nur sinnvoll ist, wenn Sie Spaß an der Knobelei haben. Wer schafft es, mit der eigenen Strategie möglichst viele Spiele zu gewinnen?

Was steckt dahinter?

In unseren Versuchen sollten Sie festgestellt haben, dass es im Normalfall günstiger ist, eine Strategie zu verwenden, die für alle möglichen Fälle die gleiche Zahl an Fragen vorsieht. Sie haben sicherlich auch bereits erkannt, dass man durch geschickte Verknüpfung von Eigenschaften immer Fragen für eine solche Strategie formulieren kann. Eine Frage bleibt allerdings: Kann es nicht doch sein, dass in irgendeinem besonderen Fall eine Strategie besser ist, die bestimmte Identitäten schneller findet als andere?

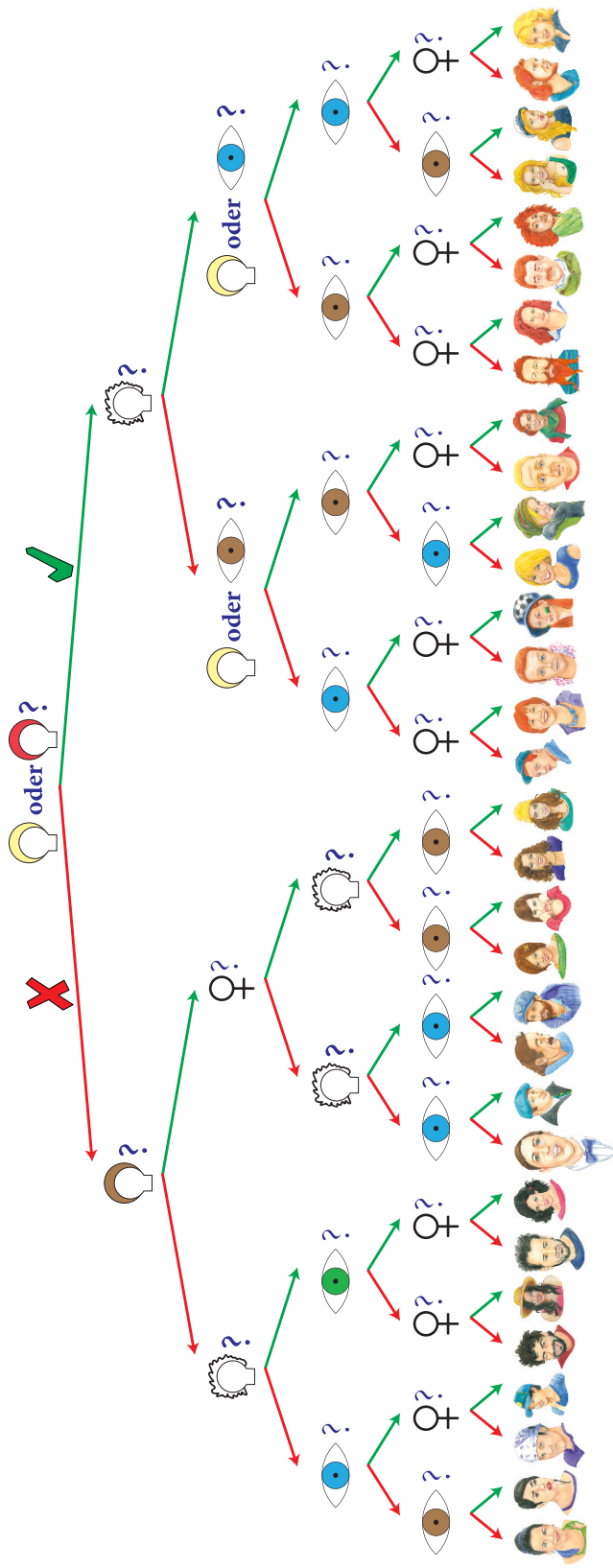
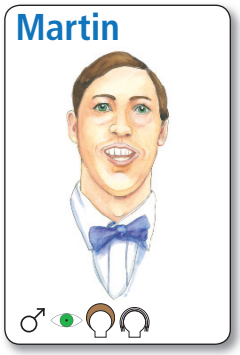


Abbildung 8.6
Entscheidungsbaum für Version C mit einer Strategie, die immer mit genau fünf Fragen zum Ziel führt.

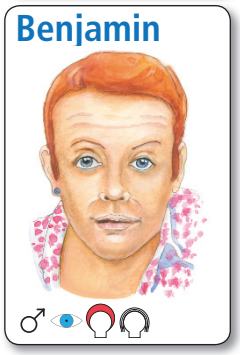
Abbildung 8.7
Erkennungsdienst, Version C mit 32 Identitäten auf der gesamten Doppelseite


Martin


Claudia


Levi


Katharina


Benjamin

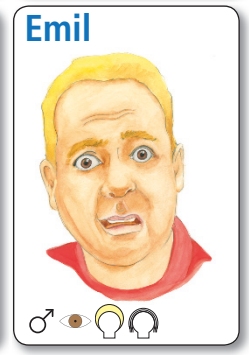

Sabrina


Marie


Hanna


Kevin


Yvonne


Emil


Wendy


Mia


Patrick


Elena


Zoe

Um dieser Theorie auf den Zahn zu fühlen, gehen wir von einer ausgeglichenen Strategie aus. Abbildung 8.8 zeigt einen entsprechenden Entscheidungsbaum. Die tatsächlichen Fragen spielen keine Rolle, daher werden diese nur durch Fragezeichen symbolisiert, die Identitäten durch schematische Köpfe unterschiedlicher Farbe.

Möchten wir nun eine Identität mit weniger Fragen herausfinden, müssen wir diese nach oben rücken. Abbildung 8.9 zeigt das. Damit „belegt" diese allerdings den Platz, den sich bisher zwei Identitäten geteilt haben. Eine ist nicht mehr auffindbar. In der Abbildung sieht man, wie sie sich darüber ärgert.

Um sie wieder unterzubringen, muss irgendwo Platz geschaffen werden. Das gelingt, indem eine beliebige andere Identität eine Stufe tiefer positioniert wird. Um diese zu finden, muss man damit künftig eine Frage mehr stellen. Die heimatlose Identität findet nun direkt daneben Platz, wie aus Abbildung 8.10 ersichtlich ist.

Vergleichen wir nun die veränderte Strategie mit unserer ursprünglichen, ausgeglichenen: Einen einzelnen Fall, für den wir nun weniger Fragen benötigen und damit gewinnen, bezahlen wir mit zwei Fällen, in denen wir mehr Fragen benötigen und damit verlieren. Insgesamt ergibt sich also mit jedem Ungleichgewicht eine Verschlechterung!

Welche Einsichten bringt uns das für die Informatik? Genau genommen folgen wir mit unseren Strategien dem Prinzip „divide et impera", das wir bereits im ersten Kapitel kennen gelernt haben: Mit jeder Frage schaffen wir kleinere Teilaufgaben, weil die Menge an noch möglichen Identitäten kleiner wird. Irgendwann besteht diese Menge nur noch aus einer Identität und wir können sie einfach benennen, die Aufgabe damit lösen.

„Erkennungsdienst" liefert uns nun auch noch einen Hinweis, wie wir bei „divide et impera" die Teilaufgaben gestalten sollten, um die Gesamtaufgabe möglichst effizient zu lösen: möglichst gleichmäßig. Das wollen wir nach dem spielerischen Beispiel an einer knallharten Problemlösung aus der Informatik nachvollziehen.

Wer suchet ...

In Kapitel 2 haben wir diverse Objekte sortiert, ohne nach dem Sinn dieser Aktivität zu fragen. Stellen Sie sich einen Karteikasten mit 1000 Kärtchen vor, auf denen Namen und die zugehörigen Telefonnummern notiert sind. Sortiert ist der Kasten natürlich nach den Namen.

Nehmen wir aber an, Sie haben die Nummer 019834299876 auf der Anruferliste des Telefons entdeckt und möchten feststellen, ob der Anruf von einer Person aus dem Karteikasten kam. Dafür bleibt Ihnen nichts anderes übrig, als alle Karten der Reihe nach durchzuschauen. Wenn Sie pro Kärtchen auch nur eine Sekunde benötigen, sind Sie trotzdem über 15 Minuten beschäftigt.

In einem zweiten, nach Nummern sortierten Karteikasten würden Sie in wenigen Sekunden fündig (oder stellten fest, dass die Nummer nicht enthalten ist). Bleibt trotzdem die Frage, wie Sie in einem solchen Karteikasten suchen.

Wenn Sie Informationen über die enthaltenen Nummer haben, können Sie natürlich versuchen, die Position zu raten, und sich dann im kleinen Bereich zum Ziel vortasten. Wenn Sie beispielsweise wissen, dass die meisten Ihrer Kunden eine Festnetz-

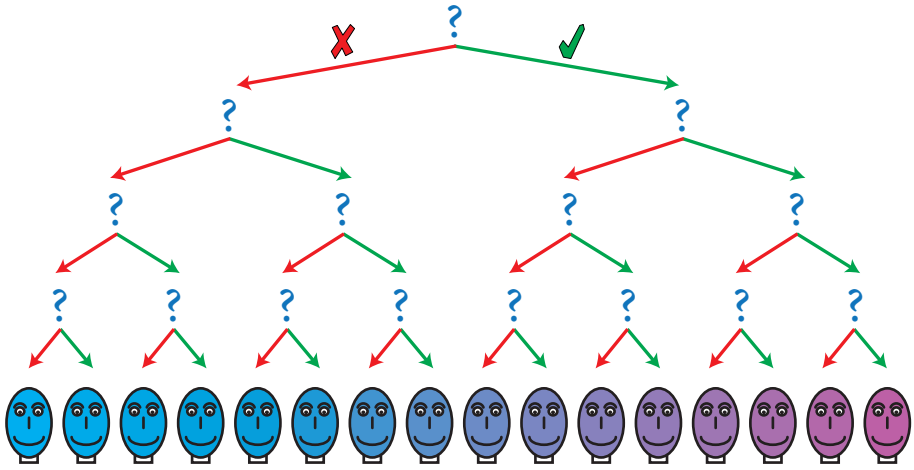


Abbildung 8.8
Allgemeiner Entscheidungsbaum für 16 Identitäten, die alle mit der gleichen Zahl an Fragen enttarnt werden können.

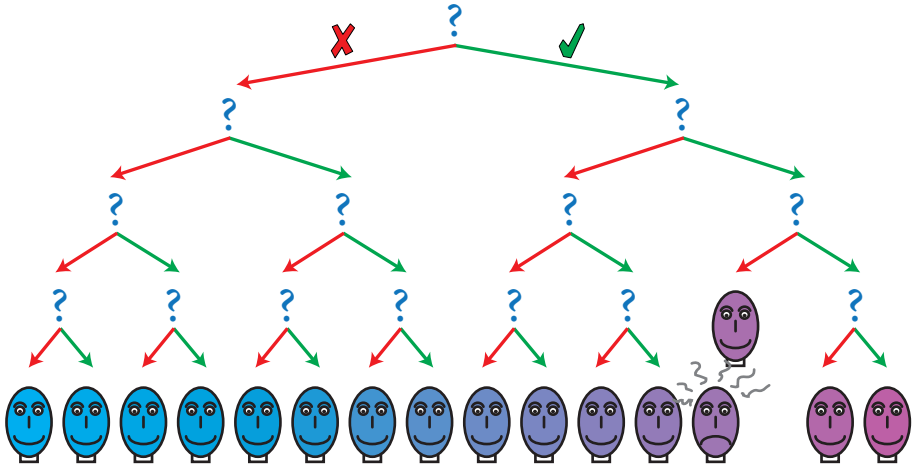


Abbildung 8.9
Eine Identität kann nun mit einer Frage weniger enttarnt werden, dafür ist eine andere gar nicht mehr auffindbar. Was für echte Agenten wünschenswert wäre, ist nicht Ziel des Spieles, daher ist die Identität sichtlich verärgert.

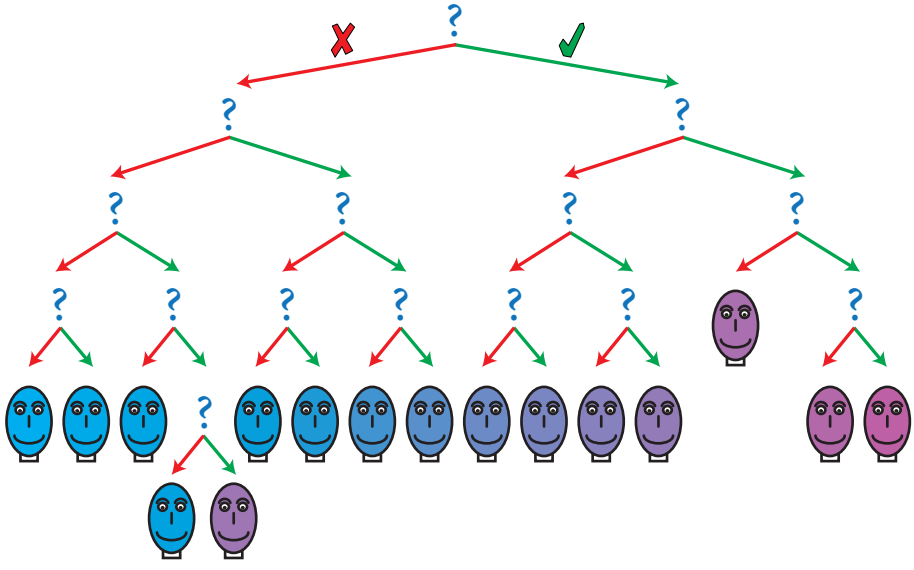


Abbildung 8.10
Die verlorene Identität bekommt an anderer Stelle einen Platz im Entscheidungsbaum. Damit benötigt man aber für zwei Identitäten jeweils eine Frage mehr, bis man sie eindeutig enttarnt hat.

nummer haben, die mit „08“ beginnt, ist es sinnvoll, nach der mit „01“ startenden Nummer oben ganz vorne suchen.

Wenn Sie allerdings nichts über die Verteilung der Nummern wissen, kann diese Strategie auch schiefgehen: Sie suchen ganz vorne, aber in der Kartei befinden sich hauptsächlich Mobilrufnummern, die mit 015, 016 oder 017 starten. Damit ist die gesuchte Nummer tatsächlich ganz hinten.

Die beste Strategie folgt daher dem in diesem Kapitel erforschten Prinzip: Mit jeder Karteikarte, die man im sortierten Kasten anschaut, stellt man nicht nur fest, ob sie die gesuchte Nummer enthält, man kann auch entweder alle Karteikarten davor oder alle Karteikarten danach ausschließen – je nachdem, ob die gesuchte Nummer größer oder kleiner ist. Abbildung 8.11 zeigt das.

Welche Karte sollten Sie nun zuerst anschauen? Nach den bisherigen Erkenntnissen diejenige, mit der man die Aufgabe „Suchen“ in etwa gleich große Teilaufgaben splittet – hier ist dies auch die mittlere Karteikarte. Abbildung 8.12 zeigt, dass der komplette obere Bereich für die weitere Suche ausscheidet.

Es geht nun weiter, indem die verbleibende Zahl in Frage kommender Karten wiederum halbiert wird wie in Abbildung 8.13 und so fort, bis Sie die gesuchte Nummer entweder finden oder keine Karte mehr übrig ist.

Eventuell denken Sie darüber nach, ob nicht eine andere Strategie geschickter sei. Beispielsweise könnten wir zunächst nicht die mittlere Karte anschauen, sondern eine im vorderen Bereich. Eventuell hätten wir dann Pech, die gesuchte Nummer wäre größer und wir müssten im Anschluss mehr Karten durchsuchen als bei der ersten Strategie. Vielleicht hätten wir aber auch Glück und die gesuchte Nummer wäre kleiner. Dann kämen wir in kürzerer Zeit zu unserem Ergebnis. Es heißt doch: „Wer wagt, gewinnt“, oder?

Wir verlassen uns dann doch lieber auf handfestere Argumente. Der mit den Abbildungen 8.8 bis 8.10 geführte Beweis ist auch auf das Suchen übertragbar und legt nahe, dass es durchschnittlich immer günstiger ist, die mittlere Karte anzuschauen. Weil dieses Verfahren die verbleibende Menge an Karten oder anderen Suchobjekten in etwa durch zwei teilt, wird es in der Informatik „binäre Suche“ genannt.

Resümee

Wir haben in diesem Kapitel an einem ganz einfachen Beispiel das Teilen großer Aufgaben in kleinere, handhabbare Aufgaben nachvollzogen und festgestellt, dass es sich in der Regel lohnt, die Teilaufgaben möglichst gleichmäßig zu gestalten.

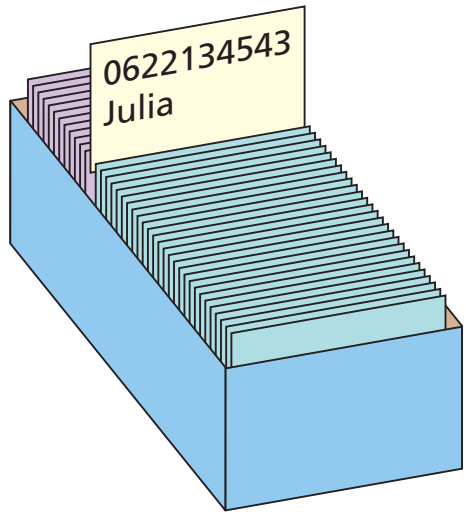


Abbildung 8.11
Sortierter Karteikasten. Im türkis markierten Bereich befinden sich nur Nummern, die in der Sortierung vor 0622134543 stehen, im violetten solche, die dahinter folgen.

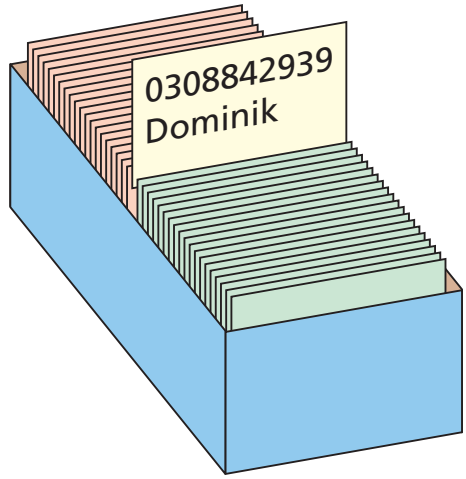


Abbildung 8.12
Auf der Suche nach 019834299876 scheiden alle roten Karten aus. Man kann im grünen Bereich weiter suchen.

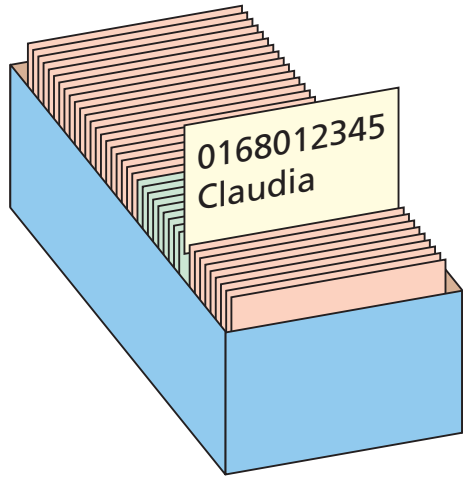


Abbildung 8.13
Auf der Suche nach 019834299876 scheiden alle roten Karten aus. Man kann im grünen Bereich weiter suchen.

Abbildung 8.K1
Kopiervorlage mit 48 möglichen Identitäten ohne Brille. Alle Kombinationen der Eigenschaften sind vertreten

Tobias

Quentin

Benjamin

Levi

Daniel

Christian

Andre

Emil

Xaver

Tim

Jerome

Alexander

Peter

Noah

Patrick

Paul

Felix

Rene

Kevin

Martin

David

Marcel

Luca

Leon

Abbildung 8.K2
Kopiervorlage mit 48 möglichen Identitäten mit Brille. Alle Kombinationen der Eigenschaften sind vertreten

Torben

Richard

Bernd

Lloyd

Dorian

Constantin

Amon

Enzo

Zeus

Titus

Jeff

Arthur

Pavel

Norbert

Pablo

Werner

Fabian

Ralf

Kenneth

Manuel

Dirk

Merlin

Ludwig

Lars


Telegärtner
43
44
46
47
49
50
47X
48X
51
52
SERIES
CAT-6 26AWG×4PR

9. Paketpost

Das Internet ist heute allgegenwärtig. Innerhalb eines Jahrzehnts wurde aus einem weitgehend wissenschaftlich genutzten Kommunikationsmittel ein Medium, das gleichberechtigt neben Fernsehen und Rundfunk steht. Heute hat es die traditionellen Medien in der Bedeutung bereits längst überholt. Statt eigener Festplatten und anderer persönlicher Datenspeicher haben viele Menschen heute überall Zugriff auf ihren Teil der „Cloud", die das Internet als hauptsächliche Infrastruktur nutzt.

Trotzdem ist kaum bekannt, was dieses Gefüge aus Rechnern zusammenhält. Wie wird sichergestellt, dass eine Nachricht korrekt am Bestimmungsort ankommt – trotz einiger Millionen Kilometer Leitung und hunderttausender Vermittlungscomputer, die teilweise verwirrend miteinander vernetzt sind? Kommen Sie mit auf eine Erkundungstour!

Zu diesem Thema selbst Experimente durchzuführen, würde recht aufwendig. Ich möchte daher versuchen, es anhand eines Rollenspiels anschaulich zu machen, das Sie nach Wahl mit Freunden oder einfach in Gedanken nachspielen können. Begleiten Sie mich in eine Zeit handschriftlicher Botschaften!

Informatix

Willkommen am Hof des Königs Informatix. Er mag seinen Hofstaat gerne gut organisiert, und das erfordert hervorragende Kommunikation der Untertanen miteinander.

In kleinen Einheiten funktioniert das sehr gut – sozusagen auf Zuruf. Hören wir daher einmal kurz in ein typisches Gespräche des Hofküchenpersonals hinein. Küchenchef Tom führt dort mit seinen vier Gehilfen Adam, Bert, Chris und Dieter das Regiment.

Adam: „Chris, sollen die Mohrrüben für deinen Auflauf in Scheiben oder in Würfel geschnitten werden?"

Chris: „Weiß nicht. Hallo Meister Tom – ist es relevant, wie die Mohrrüben für den Karottenauflauf geschnitten werden?"

Tom: „Nein Chris, aber es sollte einheitlich sein!"

Tom: „Alle mal herhören – hat schon jemand Mohrrüben geschnitten?"

Dieter: „Ja, Tom, ich habe sie in Scheiben geschnitten!"

Tom: „Chris, wir sollten einheitlich Scheiben nehmen."

Chris: „Adam, also Scheiben bitte."

Adam: „Okay, Chris."

Können Sie sich nach diesem Beispiel bereits denken, wie die Gespräche in der Küche allgemein ablaufen?

- Wer spricht zu wem?
- Wer kann zuhören?
- Wer hört tatsächlich zu?

Gesprochen wird abwechselnd. Jeder Sprecher wendet sich dabei eindeutig an:

- eine bestimmte Person („Chris, wir sollten ...“)
- oder alle Personen („Alle mal herhören – ...“)

Zuhören könnte prinzipiell jeder im Raum, allerdings sind nicht alle Gespräche für jeden interessant, und so wird normalerweise nur der Angesprochene tatsächlich aufmerksam.

Genauso wie die Personen in der Hofküche können Sie sich das interne Computernetzwerk einer kleinen Firma oder zwischen den Rechnern eines Haushaltes vorstellen: Jeder Computer kann abwechselnd etwas sagen, also eine Nachricht senden. Alle anderen Computer können diese empfangen und darauf reagieren. In Abbildung 9.1 sehen Sie entsprechend das Netzwerk der „Küchenmeister GmbH“. Im Gegensatz zu Menschen bevorzugen Computer allerdings Zahlen statt Namen.

Versuchen Sie, den Sinn des Gesprächs der Computer aus den Gesprächsstücken herauszubekommen!

Abbildung 9.1
Gesprächige Computer

Offenbar möchte Nummer 1 von Nummer 3 die Personaldatei. Nummer 2 verwaltet die Zugriffsberechtigungen, Nummer 4 verwaltet die Kennworte.

Da Computer Nummer 1 erst fragt, ohne das Kennwort zu nennen, wird ihm die Auskunft verweigert. Erst nachdem er Nummer 4 nach dem Kennwort gefragt hat und dieses bei seiner erneuten Anfrage mitliefert, bekommt er von Nummer 3 die gewünschten Daten.

Zurück an den Hof: Wenn ein neuer Lehrling in die Küche kommt, stellt das kein Problem dar. Er stellt sich vor, zum Beispiel als „Emil", und wird daraufhin in alle Gespräche eingebunden. Die Kommunikation funktioniert wie zuvor tadellos.

Was passiert aber nun, wenn König Informatix eine neue Rationalisierungsmaßnahme ausprobieren möchte: In der Hofküche ist viel ungenutzter Platz, daher soll die Hofbäckerei ebenfalls dort einziehen. Die alte Bäckerstube kann Informatix dann zur Unterbringung seiner Sammlung alter Rechenapparate nutzen. Bäckermeister Ulf wirkt mit seinen sieben Gesellen nun ebenfalls in der Hofküche, so dass nun 14 Menschen miteinander kommunizieren. Lauschen wir kurz hinein:

Gerd: „Inge, für den Kuchen benötigen wir einen dunklen Zuckerguss."

Tom: „Dieter, die Schokolade gleichmäßig umrühren."

Hans: „Gerd, ist der Ofen angeheizt?"

Tom: „Gerd, ich brauche ein paar Plätzchen als Verzierung für die Schokospeise!"

Gerd: „Ja Tom!"

Inge: „Okay, Gerd, kann ich anrühren!"

Gerd: „Nein, Hans."

Sie sehen bereits an diesem sehr kleinen Beispiel, dass es schwierig wird, den Gesprächen sinnvoll zu folgen und den Überblick zu behalten, wer nun mit wem kommuniziert. Überlegen Sie, was passiert, wenn sich eine ganze Halle voller Menschen durch ständige Zurufe über ihre Arbeit austauschen muss. Im Fall, dass jeder nur selten etwas von jemand anderem möchte, funktioniert die Sache noch. Muss aber viel kommuniziert werden, wird die gegenseitige Störung so groß, dass man eventuell gar nicht mehr oder nur sehr verzögert zu Wort kommt. Die Leistungsfähigkeit des gesamten Betriebes leidet dann darunter. So passiert das auch am Hof von Informatix – seit der Zusammenlegung der Küche mit der Bäckerei muss der König den Gürtel deutlich enger schnallen, weil viel weniger produziert wird. Davon ist er natürlich alles andere als begeistert, denn Essen ist seiner Meinung nach die wichtigste Tätigkeit des Tages.

Helfen Sie dem hungrigen König! Wie könnte er – trotz Rationalisierungszwang – die Produktivität wieder steigern?

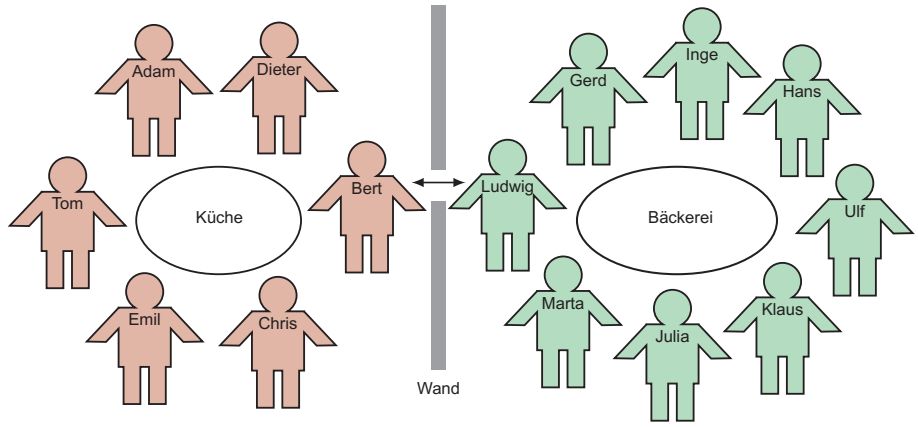
Die Analyse der Situation ist eindeutig: Zu viele Leute sind am Gespräch beteiligt. Die Gesprächskreise stören sich gegenseitig, weil sie das gleiche „Medium", in diesem Fall denselben Raum benutzen.

Die Lösung ist eine schalldämmende Trennwand! Die Küche wird aufgeteilt. Nun können beide Arbeitsgruppen wieder ungestört miteinander reden.

Allerdings ist das noch nicht die ganze Lösung! Wenn Sie den Dialog oben genau angeschaut haben, ist Ihnen sicher aufgefallen, dass zum Beispiel Tom und Gerd miteinander geredet haben – sie sind in unterschiedlichen Gruppen. Einen Vorteil hatte die Zusammenlegung nämlich schon: Köche und Bäcker konnten ihre Aufträge koordinieren und sich gegenseitig zuarbeiten. Die hierfür notwendige Kommunikation ist nun aber durch die Zwischenwand blockiert.

Eine Lösung ist, in die Wand ein kleines Fenster einzubauen, durch das sich gerade zwei Personen unterhalten können. Je ein Geselle aus den Gruppen wird dann zum „Kommunikator" ernannt. Die Kommunikatoren sitzen vor dem Fenster und können sich durch die Wand unterhalten, ohne dass die Gespräche in den jeweiligen Räumen beeinträchtigt werden. Abbildung 9.2 zeigt das schematisch. In den vorangegangenen Kapiteln haben wir sehr ausführlich über Abstraktion gesprochen. Daher werde ich im Folgenden auch abstrakte Darstellungen verwenden, um die Sachverhalte möglichst klar darzustellen.

Abbildung 9.2
Wieder in zwei Werkstätten ...



Die Hofangestellten können nun wieder uneingeschränkt in ihren Gruppen miteinander reden. Um mit jemandem aus der anderen Gruppe zu sprechen, müssen sie über Bert und Ludwig gehen, die den Auftrag haben, die Verbindung zu halten.

Überlegen Sie nun, wie die letzte Kommunikation unter den neuen räumlichen Gegebenheiten abgelaufen wäre. Am besten notieren Sie es kurz.

Im Dialog sind die veränderten Passagen rot markiert. Es hat sich selbstverständlich nur dort etwas verändert, wo die Kommunikation nun zwischen zwei Räumen stattfindet.

Gerd: „Inge, für den Kuchen benötigen wir einen dunklen Zuckerguss."

Tom: „Dieter, die Schokolade gleichmäßig umrühren."

Hans: „Gerd, ist der Ofen angeheizt?"

Tom: „Bert, sag' Gerd, ich brauche ein paar Plätzchen als Verzierung für die Schoko-Speise!"

Bert: „Ludwig, sag' Gerd, Tom benötigt ein paar Plätzchen als Verzierung für eine Schoko-Speise!"

Ludwig: „Gerd, Tom benötigt ein paar Plätzchen als Verzierung für eine Schoko-Speise!"

Gerd: „Ludwig, sag' Tom: Ja!"

Ludwig: „Bert, sag' Tom von Gerd: Ja!"

Bert: „Tom, Gerd sagt: Ja!"

Inge: „Okay, Gerd, kann ich anrühren!"

Gerd: „Nein, Hans."

Die Entlastung der einzelnen Räume hat also ihren Preis: Möchte man mit jemandem sprechen, der nicht mit im Raum sitzt, geht das immer über mehrere Ecken. In unserem Fall wird eine einfache Nachricht verdreifacht – sie muss jeweils von Bert und von Ludwig wiederholt werden, um den Adressaten zu erreichen.

Genau die gleichen Effekte kann man auch in Computernetzwerken beobachten: Statt einfacher Gespräche werden kurze Datenblöcke verschickt, die sogenannten Datenpakete. Das Versenden eines Datenpakets blockiert das Netzwerk, genau wie die Kommunikation durch Worte andere Gespräche im Raum unterbricht.

Bei den ersten Computernetzwerken konnten prinzipiell alle angeschlossenen Rechner gleichzeitig miteinander reden, was sehr schnell zur Überlastung führte. Weiterer Malus war die Tatsache, dass prinzipiell jeder Teilnehmer alle Gespräche innerhalb eines Netzes belauschen konnte.

Oft waren die Programme auch noch sehr „gesprächig". So fragten einige in kurzen Abständen das gesamte Netzwerk so etwas wie: „Hallo, mein Name ist Hugo, wer ist denn noch so alles angeschlossen?", woraufhin alle anderen Computer im Netz eine Antwort schickten. Auf diese Weise kam es häufig vor, dass ein Netzwerk mit zum Beispiel 20 Computern tadellos lief – schloss man jedoch einen einzigen weiteren Rechner an, ging gar nichts mehr: Die Leitungen waren komplett mit „Hallo"-Meldungen wie der obigen belegt, so dass sinnvolle Anfragen keine Chance hatten.

Die Lösung dieses Problems ist prinzipiell so einfach wie in Informatix' Hofküche: Kleine Computernetzwerke, in denen direkt jeder mit jedem anderen sprechen kann, dürfen eine gewisse Anzahl von Teilnehmern nicht überschreiten. Man spricht hier übrigens auch von LAN oder Local Area Network: Das sind die Computer, die in einer kleinen Firma oder einer Abteilung eng miteinander verbunden sind.

Ein WAN oder Wide Area Network verbindet mehrere LANs miteinander. In der Hofküche ist der Job von Bert und Ludwig, das WAN aufrechtzuerhalten.

Effektiv gibt es heute im Internet sehr viele Hierarchiestufen zwischen dem LAN, das Ihren Rechner vielleicht mit Ihrem Drucker verbindet, und dem WAN, das die ganz großen Internet-Anbieter miteinander verbindet.

Zurück zum Hofstaat: König Informatix ist begeistert von den Möglichkeiten, die die Trennwand mit Loch bietet, und überträgt das sofort auf seinen gesamten Hofstaat: Nicht nur Bäcker und Köche sollen miteinander reden können, sondern alle anderen auch. Es würde sicher nicht funktionieren, alle in einen einzigen Raum mit Trennwänden zu setzen, daher wird pro Arbeitsgruppe ein Bote bestimmt, der als Kommunikator den Kontakt zu den anderen Kommunikatoren hält, indem er Nachrichten überbringt. Abbildung 9.3 zeigt, wie die Gruppen über ihre Kontaktpersonen Gunter, Olga, Bert, Ludwig, Lina und Rolf Nachrichten austauschen können – diese bilden quasi ein eigenes Netzwerk auf höherer Ebene.

Stellen Sie sich vor, es soll nun auch Kontakt zu den Angestellten in den Palästen der benachbarten Könige Technokratix und Computix gehalten werden. Hierfür würden wiederum Kommunikatoren bestimmt, die die Verbindung herstellen. In diesem Netzwerk könnte dann jemand für die Kommunikation mit dem Kaiserpalast abgestellt werden usw. Auf diese Weise ist man in der Lage, ein riesiges Netzwerk aufzubauen. Allerdings stoßen unsere Boten im Beispiel schnell auf ein Problem, das auch die Post in den Anfängen hatte: Auch wenn die hier verwendeten Vornamen eindeutig sind, kann man das für reale Verhältnisse kaum annehmen.

Selbst wenn man dies gewährleisten könnte, würde es für alle Beteiligten sehr verwirrend. So müsste sich jeder Beteiligte immer merken, mit welchen Gesprächspartnern er direkt reden kann und für welche er die Kontaktperson anzusprechen hat.

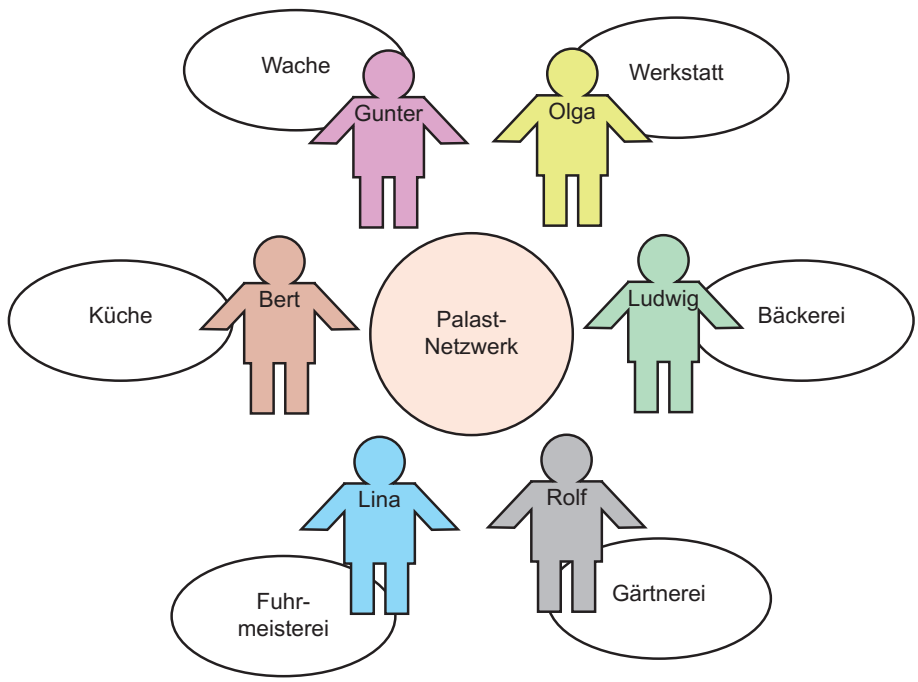


Abbildung 9.3
Alle Palastangestellten tauschen sich über ihre Kontaktpersonen aus.

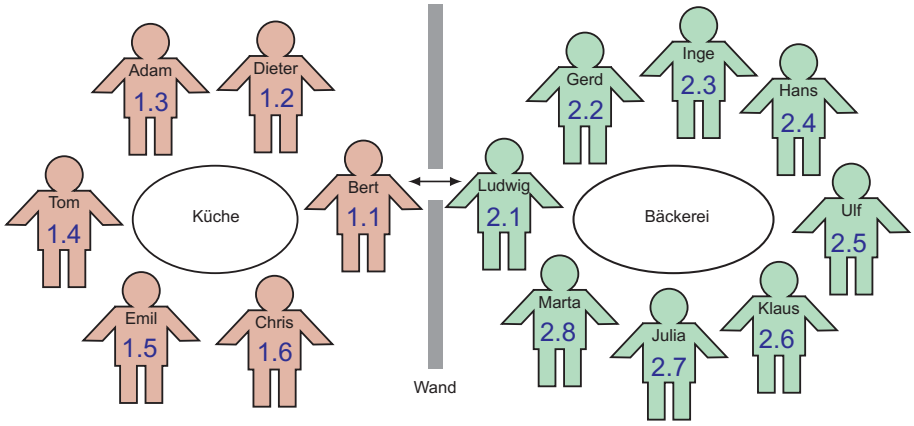


Abbildung 9.4
Jeder Gesprächspartner hat nun eine Nummer.

Die Post hat dieses Dilemma durch die Einführung von Zahlen als eindeutiges Identifikationsmerkmal gelöst – zumindest in Teilbereichen der Adresse: Postleitzahl und Hausnummer.

Es wäre nun möglich, einfach jedem möglichen Gesprächspartner zu seinem Namen eine beliebige Nummer zu verpassen. Dann hätte man allerdings die Vorteile der Zahlen nicht ausgenutzt: Postleitzahlen oder zum Beispiel auch Telefonnummern folgen zusätzlich auch einem Schema. Die ersten Ziffern grenzen die Region ein und erst die letzten Ziffern bestimmen den Ort bzw. die Person genauer.

Postleitzahl und Hausnummer
Übrigens: Die Postleitzahl gibt es für die Öffentlichkeit in Deutschland seit 1944, als die Bevölkerung aufgefordert wurde, zwecks schnellerer Beförderung die Nummer der Oberpostdirektion zur Adresse anzugeben, zum Beispiel 5b für Oberpreußen.
Die Hausnummer wurde bereits viel früher „erfunden“: Kaiserin Maria Theresia verfügte sie 1770 zu Zwecken der leichteren Volkszählung in Wien.

Ein solches Verfahren kann sich auch König Informatix zunutze machen. Das kommt seinem Ordnungsdrang sehr entgegen. Schauen wir uns hierfür nochmals den kleinen Gesprächskreis in der Palastküche an. Jeder Angestellte bekommt eine Nummer zugewiesen. Dabei ist es sinnvoll, den Kommunikatoren eine einheitliche Nummer – zum Beispiel die 1 – zu geben. Eine vorangestellte weitere Nummer bestimmt dann, in welchem Küchenteil sich der Teilnehmer befindet – 1 für die eigentliche Küche und 2 für die Bäckerei. Sie sehen das in Abbildung 9.4. Auf diese Weise könnten auch die anderen Bereiche des Palastes ihre eigene Nummer erhalten: 3 für die Werkstatt, 4 für die Wache usw.

Wie helfen aber die Nummern bei der Kommunikation? Nehmen wir einmal an, Inge möchte mit Maria reden. Inge hat die Nummer 2.3, Maria die 2.8. Inge erkennt, dass Maria sozusagen die gleiche „Vorwahl“ hat wie sie, daher kann sie ihr Anliegen einfach in den Raum rufen, Maria hört sie.

IP
Internet-Protokoll. Dieser Standard ist bereits im September 1981 bekanntgegeben worden. Ursprünglich für die Forschungsbehörde im amerikanischen Pentagon entwickelt, ist IP bis heute die Grundlage der Kommunikation zwischen Computern im Internet. Die 1998 veröffentlichte Weiterentwicklung IPv6 konnte sich trotz deutlich größerer Möglichkeiten bis heute nicht klar durchsetzen.

Wenn Inge mit Dieter sprechen möchte, ist das etwas anderes: Dieter hat die Nummer 1.2, befindet sich also in einem anderen Gesprächsnetz. Daher sagt sie ihr Anliegen ihrem Kommunikator, der 2.1. Dieser weiß, dass der zuständige Kommunikator für Nachrichten an die 1.1 Bert ist, und gibt die Nachricht an ihn weiter. Dieser leitet sie schließlich an Dieter mit der 1.2.

Auf die gleiche Weise funktioniert das Internet. Jeder Computer ist eindeutig anhand einer Zahlenkombination identifizierbar: Es handelt sich heute normalerweise um vier Nummern zwischen 0 und 255, die meistens mit Punkt getrennt dargestellt sind, zum Beispiel 130.83.242.159. Die ganze Kombination nennt man IP-Nummer oder IP-Adresse.

Diese Art der Adressierung aus verschiedenen, hierarchisch geordneten Zahlen kennen Sie übrigens auch vom täglichen Gebrauch: Eine vollständige Telefonnummer besteht aus einer Landesvorwahl (49 für Deutschland), einer Ortsvorwahl (z. B. 6151 für

Darmstadt) und einer Anschlussnummer (z. B. 123456). Die gesamte Nummer lautet dann +49 6151 123456.

Genau wie in unserem Beispiel haben alle Rechner in einem kleinen Netzwerk, z. B. einer Firma, dieselben vorderen Zahlen als IP-Nummer. Sie unterscheiden sich lediglich in der hinteren Zahl. Die vorderen Zahlen bestimmen dann, um welches Netzwerk es sich handelt. Die Rolle der Kommunikatoren im obigen Beispiel übernehmen im Computernetz die sogenannten Router. Es sind spezialisierte Rechner, die Datenpakete im Internet in die richtige Richtung weiterleiten.

Abbildung 9.5 zeigt schematisch einen Ausschnitt des Internets. Die Geschäftsführer der Informatix AG haben sich die Ordnung des alten Königs zunutze gemacht und sie auf ihre heutige Infrastruktur übertragen.

Beachten Sie, wie die Router (in der Skizze als flache Kästchen dargestellt) zwei oder mehr IP-Adressen besitzen, weil sie ja auch in mehreren Computernetzen „sprechen" müssen.

Im Informatix-Beispiel wären das etwa Bert oder Ludwig, die einerseits innerhalb ihrer Gruppe kommunizieren und dafür eine Nummer aus dem entsprechenden Bereich besitzen. Andererseits sprechen sie jedoch auch zusätzlich noch mit den anderen Kommunikatoren und haben hierfür eine weitere Adresse.

Übertragen Sie Ihr Wissen der Kommunikationsstruktur in Informatix' altem Königreich auf das Internet: Setzen Sie sich an den Rechner mit der IP-Adresse 10.43.3.32

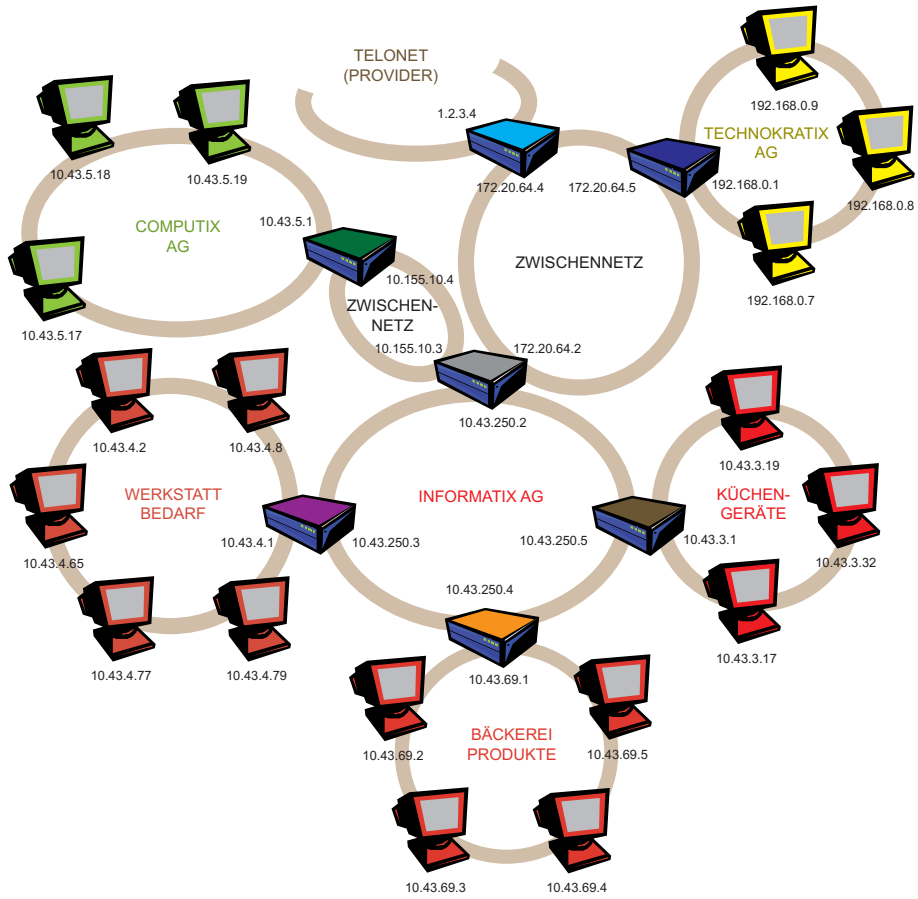


Abbildung 9.5
Internetausschnitt mit der Informatix AG

in der Abteilung für Küchengeräte. Wie laufen die Datenpakete zu den Computern 10.43.3.19, 192.168.0.9 und 10.43.4.2? Schildern Sie genau, welche Nachrichten die Computer bzw. Router untereinander austauschen!

Fall 1: Ausgangs-Adresse: 10.43.3.32, Ziel-Adresse: 10.43.3.19

Der Computer erkennt, dass die ersten drei Zahlen von Ausgangs- und Ziel-Adresse identisch sind, und schickt die Nachricht direkt an die Ziel-Adresse ... fertig:

Von: 10.43.3.32 – An: 10.43.3.19 – Nachricht

Fall 2: Ausgangs-Adresse: 10.43.3.32, Ziel-Adresse: 192.168.0.9

Hier sind die ersten Zahlen komplett unterschiedlich, der Computer schickt die Nachricht also an seinen zugeordneten Router, der sie dann immer weiter zum Ziel schickt:

Von: 10.43.3.32 – An: 10.43.3.1 – Nachricht von 10.43.3.32 für 192.168.0.9

Von: 10.43.250.5 – An: 10.43.250.2 – Nachricht von 10.43.3.32 für 192.168.0.9

Von: 172.20.64.2 – An: 172.20.64.5 – Nachricht von 10.43.3.32 für 192.168.0.9

Von: 192.168.0.1 – An: 192.168.0.9 – Nachricht von 10.43.3.32 für 192.168.0.9

Der zugeordnete Router ...
... heißt bei heutigen Computersystemen übrigens auch „Standard-Gateway“ oder einfach „Gateway“.

An diesem Beispiel sehen Sie, dass die eigentliche Nachricht nochmals Sender und Empfänger beinhalten muss. Ansonsten wüssten die Router nicht, an wen sie diese weiterleiten sollen. Der Empfänger kann auch nur so feststellen, wer der Absender eigentlich war, denn direkt empfängt er die Nachricht ja von seinem Router 192.168.0.1.

Fall 3: Ausgangs-Adresse: 10.43.3.32, Ziel-Adresse: 10.43.4.2

Dieser Fall funktioniert genau wie Fall 2.

Von: 10.43.3.32 – An: 10.43.3.1 – Nachricht von 10.43.3.32 für 10.43.4.2

Von: 10.43.250.5 – An: 10.43.250.3 – Nachricht von 10.43.3.32 für 10.43.4.2

Von: 10.43.4.1 – An: 10.43.4.2 – Nachricht von 10.43.3.32 für 10.43.4.2

Bis hierher ist die Sache sehr einfach: Die Nachricht wird so lange von Netzwerk zu Netzwerk geschickt, bis sie am Ziel ankommt. Anhand des Planes können Sie auch genau bestimmen, welche Stationen hierbei passiert werden müssen.

Allerdings besitzen die beteiligten Computer keinen solchen Plan, und das ist auch gar nicht notwendig. Überlegen Sie daher, welche Informationen die einzelnen Computer (nicht die Router) benötigen, um am Internet teilzunehmen. Welche anderen Netzwerkadressen müssen sie kennen?

Tatsächlich muss natürlich jeder seine eigene IP-Adresse kennen. Hinzu kommt die Information, dass sich alle Computer mit gleichen ersten drei Zahlen in der IP-Adresse im selben Netzwerk befinden und man sie direkt ansprechen kann.

Zusätzlich ist nur noch die Adresse des zugeordneten Routers notwendig. An diesen werden alle anderen Nachrichten geschickt! Auch die normalen Bediensteten von König Informatix brauchten nur zu wissen, wem sie eine Nachricht geben mussten, wenn

diese für jemanden außerhalb des Zimmers bestimmt war. Den Rest erledigten dann die Kommunikatoren. Im Internet sind das die Router.

Für die einzelnen Computer ist die Welt also einfach – die eigentliche Vermittlungsarbeit wird von den Routern erledigt. Diese besitzen Tabellen, in denen steht, wo sie eine Nachricht hinschicken müssen, damit diese näher zum Ziel kommt. So eine Tabelle kann sehr einfach sein.

Betrachten wir zum Beispiel die Tabelle für den grünen Router der Computix AG.

Nachricht von	Nachricht an	Schicke an
10.43.5.*	irgendjemand	10.155.10.3
irgendjemand	10.43.5.*	Ziel-Adresse

Die Tabelle ist so einfach, weil der Router lediglich zwei Netze miteinander verbindet. Alles, was von drinnen kommt, muss an den nächsten Router weitergeleitet werden. Das besagt die erste Regel. Das Sternchen ist hierbei ein Platzhalter, der so viel wie „irgendeine Zahl“ bedeutet.

Kleine Welt
Das Kleine-Welt-Phänomen ist 1967 vom Soziologen Stanley Milgram formuliert worden. Eigentlich sagt es etwas über die große Welt aus: Wenn man die Bekanntschaftsbeziehungen jedes Menschen aufzeichnet, dann sind überraschenderweise fast alle Menschen der Welt über sehr wenige „Umwege“ miteinander bekannt. Wir leben also sozusagen doch in einer kleinen Welt! Das Gleiche gilt auch für das Internet: Obwohl jeder Computer und Netzverbindungsknoten nur wenige andere kennt, sind alle über sehr wenige Strecken oder „Hops“ erreichbar. Das bringt nicht immer nur Vorteile mit sich: Zum Beispiel haben Computerviren sehr kurze „Übertragungswege“.

Die zweite Regel legt dann fest, dass Pakete an ein bekanntes Ziel (nämlich eines aus dem inneren Netzwerk der Computix AG) direkt dorthin zugestellt werden. Mehr Regeln benötigt der grüne Router nicht!

Vielleicht wundern Sie sich, warum nicht alle möglichen Fälle abgedeckt sind. Was macht der Router mit einer Nachricht von 192.168.0.7 an 10.43.69.2?

Diese Nachricht würde in keiner sinnvollen Konstellation beim grünen Router vorbeikommen! Entweder es handelt sich um einen Irrläufer oder ein Computer aus dem Netz der Computix AG hat einen falschen Absender angegeben (zum Beispiel um Zugang zu einer Webseite zu bekommen, die nur für Computer der Technokratix AG freigegeben ist). In beiden Fällen gibt es nur eine richtige Reaktion: Er ignoriert die Nachricht einfach.

Überlegen Sie nun, welche Tabelle der braune Router besitzen muss.

Diese Tabelle ist bereits etwas komplizierter. Der Router kennt drei weitere Router, an die er Nachrichten schicken kann. Für jeden dieser Router gibt es eine Regel.

Nachricht von	Nachricht an	Schicke an
10.43.3.*	10.43.69.*	10.43.250.4
10.43.3.*	10.43.4.*	10.43.250.3
10.43.3.*	irgendjemand	10.43.250.2
irgendjemand	10.43.3.*	Ziel-Adresse

Beachten Sie die dritte Zeile: Diese widerspricht sich mit den ersten beiden, da „irgendjemand“ natürlich auch die explizit angegebenen Zieladressen umfasst. Wir gehen daher immer davon aus, dass eine spezieller gefasste Regel höhere Priorität als die

allgemeinere hat: Es wird immer die für den gegebenen Fall speziellste Regel angewendet.

Steigern wir unseren Regulierungsdrang weiter: Stellen Sie nun die Tabelle für den grauen Router in der Mitte auf! Sie können davon ausgehen, dass alle Nachrichten für Computer außerhalb der Technokratix AG, der Computix AG und der Informatix AG über den Provider Telonet verschickt werden.

Hier ist die Sache erneut etwas komplizierter. Der Router hat sogar drei Anschlüsse in verschiedenen Netzen und in seinem Bereich steht auch der blaue Router, der die Verbindung mit allen nicht eingezeichneten Computern im Internet herstellen kann.

Nachricht von	Nachricht an	Schicke an
irgendjemand	10.43.5.*	10.155.10.4
irgendjemand	10.43.3.*	10.43.250.5
irgendjemand	10.43.4.*	10.43.250.3
irgendjemand	10.43.69.*	10.43.250.4
irgendjemand	192.168.0.*	172.20.64.5
10.43.5.*	irgendjemand	172.20.64.4
10.43.4.*	irgendjemand	172.20.64.4
10.43.3.*	irgendjemand	172.20.64.4
10.43.69.*	irgendjemand	172.20.64.4

An diesen Beispielen sehen Sie, dass das Internet prinzipiell durch das Zusammenspiel sehr vieler Router funktioniert. Diese wiederum vermitteln Nachrichten aufgrund fester Routing-Tabellen. Solche Tabellen können kleiner oder größer ausfallen. Da die Router jedoch immer nur einen kleinen Ausschnitt des Internets abdecken, bleiben diese übersichtlich.

Erkennen Sie, dass auch das Internet „typisch informatisch“ aufgebaut ist? Das Prinzip „divide et impera“, das dieses Buch schon über mehrere Kapitel durchzieht, können Sie auch hier ganz deutlich erkennen: Das Internet als Ganzes ist unübersichtlich und kaum zu durchschauen. Es besteht jedoch aus einzelnen Komponenten, die jeweils nur einen kleinen Teil des Internets selbst kennen. Die Funktionsweise jeder dieser Komponenten ist recht leicht zu verstehen. Nur auf diese Weise kann ein solch riesiges Netz funktionieren!

Ach wie gut, dass jeder weiß ...

Rumpelstilzchen möchte im Märchen seinen Namen um jeden Preis verbergen. Genau das Gegenteil trifft auf die Computer im Internet zu – fast jeder Internet-Teilnehmer hat heute zusätzlich zur IP-Nummer noch einen IP-Namen.

Wofür braucht man diesen? Die Computer sind anhand ihrer Nummer eindeutig identifizierbar und außerdem arbeiten Computer doch sehr gerne mit Zahlen!

Wie groß ist das Internet?
Diese Frage beschäftigte schon mehrere Studien. Allerdings kann man sie ganz verschieden interpretieren: Wie viele Computer sind angeschlossen? Wie viele Computer haben eine eigene Adresse? Wie viele Computer sind „Server", bieten also irgendwelche Dienste rund um die Uhr an? Wie viele Webseiten gibt es?
Die Graphik unten zeigt, wie viele Computer in den unterschiedlichen Jahren einen Namen hatten, der fest einer Adresse zugewiesen ist. Da nahezu jeder Computer, der zeitweise am Internet teilnehmen möchte, einen Namen bekommt, ist das ungefähr die Menge von Computern, die prinzipiell (aber nicht dauerhaft) am Internet angeschlossen sind. Die Skala ist in Millionen Computern eingeteilt.

Stimmt! Tatsächlich kommunizieren die Rechner miteinander auch ausschließlich auf Basis der IP-Nummern. Ein Problem haben allerdings die Benutzer: Könnten Sie sich vorstellen, im Internet zu surfen und dabei ausschließlich mit IP-Nummern zu arbeiten? Wenn Sie auf die Seiten von „Abenteuer Informatik" schauen möchten, rufen Sie nicht http://www.abenteuer-informatik.de/ auf, sondern http://194.175.173.45/. Stellen Sie sich die riesigen Telefonbücher vor, in denen die IP-Nummern aller Computer im Internet verzeichnet sind.

Glücklicherweise können uns Computer aber Arbeit abnehmen, und so hat man 1983 den sogenannten „Domain Name Service (DNS)" erfunden, übersetzt etwa „Namensbereichsdienst". Spezielle Computer im Internet – die DNS-Server (oder einfach Nameserver) – übernehmen die Rolle des Telefonbuchs. Wenn Sie einen Namen in der Adresszeile Ihres Internet-Browsers eintippen, kann damit der Computer erst einmal gar nichts anfangen. Er benötigt zur Kommunikation mit diesem Rechner unbedingt die entsprechende IP-Nummer.

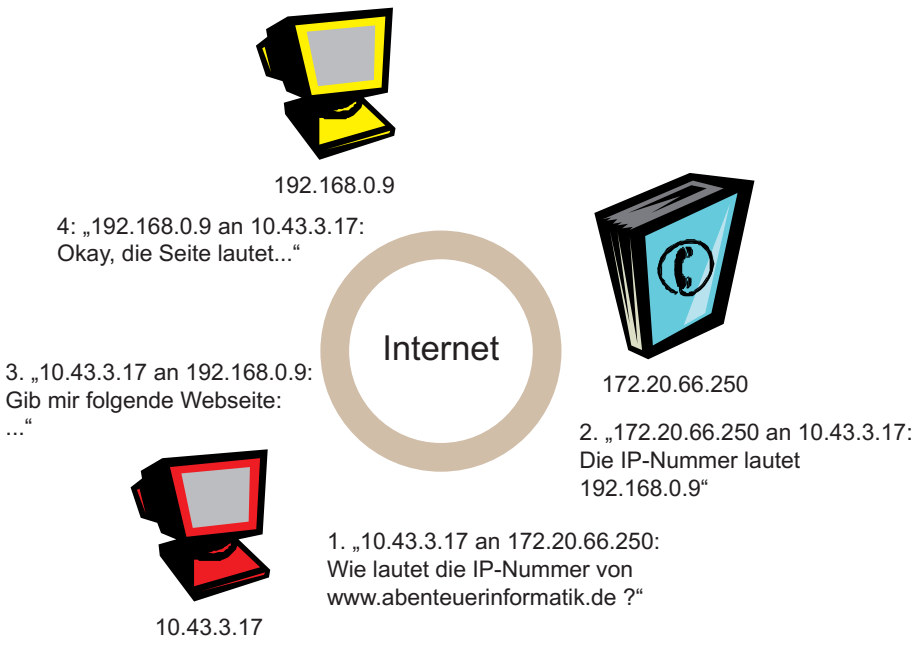
Er muss diese daher in Erfahrung bringen und fragt einen DNS-Server danach. Von diesem muss er allerdings die IP-Nummer kennen, sonst könnte er ihn ja auch nicht erreichen. Abbildung 9.6 zeigt die Kommunikation schematisch.

Im Beispiel möchte der Benutzer des roten Computers die Internetseite von „www.abenteuer-informatik.de" aufrufen. Der Computer kennt die IP-Nummer des Zielrechners nicht und fragt beim Nameserver (in der Graphik blau) nach. Dieser antwortet mit der korrekten Nummer, woraufhin die eigentliche Frage nach der Webseite gestellt werden kann.

Wenn Sie also beim Recherchieren im World Wide Web eine Bezeichnung wie „www.irgendetwas.de" benutzen, ist dies immer der Name eines Computers, nicht etwa die Bezeichnung für ein spezielles Dokument!

Jetzt wissen Sie auch, warum Ihr Computer die IP-Nummer eines Nameservers kennen muss, damit Sie sinnvoll im Internet arbeiten können!

Abbildung 9.6
Kommunikation mit dem DNS-Server



Obwohl dieses Buch eher Verfahren der Informatik beschreibt, möchte ich der Vollständigkeit halber kurz auf die Verteilung von IP-Nummer und IP-Namen eingehen. Die gesamte Kommunikation im Internet beruht darauf, dass eine IP-Nummer bzw. ein IP-Name ganz eindeutig einem einzelnen Computer zugewiesen ist. Nur auf diese Weise können Nachrichten gezielt an den Adressaten zugestellt werden.

Das gleiche Problem gibt es auch bei der normalen Post: Wenn drei Herren mit dem Namen „Peter Müller" in einem Hochhaus wohnen, weiß der Briefträger nicht, wem er entsprechende Post zustellen soll. Hier muss eine zusätzliche Angabe (z. B. „1. Stock") für Klarheit sorgen. Während der Briefträger gegebenenfalls nachfragen könnte, würden im Internet die Nachrichten entweder falsch oder gar nicht zugestellt werden.

Eine Organisation namens IANA (Internet Assigned Numbers Authority) vergibt daher IP-Nummern auf Antrag an lokale Behörden, Internet-Provider oder Organisationen. Dabei werden nie einzelne Nummern, sondern immer ganze Blöcke zugeteilt. Diese können dann vom „Besitzer" weiter unterteilt und vergeben werden.

Normalerweise spricht man von einer Class-A-Adresse, wenn nur die erste Zahl von der IANA festgelegt wird, zum Beispiel ist der Bereich 13.*.*.* komplett der Firma Xerox zugeteilt. Die Sternchen stehen für beliebige Zahlen. Xerox muss (und darf) diese selbst verwalten und innerhalb des eigenen Bereichs dafür sorgen, dass es keine Computer mit doppelter IP-Nummer gibt. Im Bereich einer Class-A-Adresse können ungefähr 256^3, das heißt ca. 16 Millionen Rechner eine individuelle IP-Nummer haben.

Bei der Class-B-Adresse sind die beiden ersten Zahlen festgelegt. So ist 130.83.*.* der Technischen Universität Darmstadt zugewiesen. Innerhalb dieses Bereichs können etwa 256^2, also ca. 65.000 Computer mit eigener Nummer angeschlossen werden. Es gibt auch noch Class-C-Netze mit drei festgelegten Zahlen. In den Anfängen des Internets war man auf diese drei Bereichseinteilungen festgelegt. Heute können die Nummernbereiche auch individueller eingeteilt und vergeben werden, zum Beispiel ist auch ein Bereich mit nur vier Adressen möglich. Der Einfachheit halber werden wir hier jedoch immer von den klassischen Bereichen ausgehen.

Auch Namen werden zentral zugeteilt. Während die Hierarchie bei den IP-Nummern von vorne nach hinten verläuft, ist das bei den IP-Namen genau umgekehrt: Von der ICANN (Internet Corporation for Assigned Names and Numbers) werden Domains, also Namensbereiche, an Personen und Organisationen vergeben. Diese können dann innerhalb ihres Bereichs beliebig weitere Namen vergeben. Einzige Einschränkung: Der gesamte Name darf mit Punkten nicht mehr als 255 Zeichen haben.

Beispielsweise ist der gesamte Namensbereich mit Endung „.de" der deutschen Organisation DENIC (Deutsches Network Information Center) übertragen worden. Möchte man den Bereich „abenteuer-informatik.de" für sich beanspruchen, kann man dies bei der DENIC anmelden. Diese reservieren dann gegen Gebühr die Domain und Sie können weitere Unterbereiche vergeben, zum Beispiel „verwaltung.abenteuer-informatik.de" oder „buch.abenteuer-informatik.de". An erster Position steht immer der Name des Computers, der auch oft die Bezeichnung seiner Funktion enthält, also zum Beispiel „www.buch.abenteuer-informatik.de" für den Web-Server, der sich mit der Unterrubrik „buch" beschäftigt. Ein Computer kann auch mehrere Namen tragen, zum Beispiel wenn er gleichzeitig Web-Server und Mail-Server ist.

Alle Organisationen, die einen Namensbereich weiter administrieren und für andere Unterbereiche zur Verfügung stellen, betreiben einen DNS-Server oder teilen sich ei-

Meilensteine der Internet-Verwaltung

1972 Gründung der IANA und einheitliche Vergabe von IP-Nummern

1985 Erste IP-Namen werden vergeben.

1992 InterNIC wird gegründet, um die IP-Namen zu verwalten.

1998 Die ICANN wird gegründet, um InterNIC als oberste „Namensbehörde" abzulösen.

2000 ICANN@large wird etabliert, eine Art Mitbestimmungsgremium: Zum ersten Mal kann jeder Internet-Benutzer Vorstandsmitglieder der ICANN wählen.

nen DNS-Server mit einem Partner. Dieser enthält dann das Telefonbuch mit der Name-Nummer-Übersetzung für die meisten Computer im Internet. Auf jeden Fall muss er die Name-Nummer-Übersetzungen für die Computer im eigenen Namensbereich enthalten. Auf diese Weise können die DNS-Server sich gegenseitig nach Nummern fragen, die sie selbst nicht gespeichert haben. Spätestens der DNS-Server der für den Namensbereich verantwortlichen Organisation kann dann eine korrekte Auskunft erteilen.

Das wird an einem Beispiel am klarsten:

Nehmen wir an, die Kantine der Informatix AG möchte selbst ins Internet und meldet daher den Namen „kantine.informatix.de“ an. Diese neue Domain ist anfangs lediglich dem DNS-Server der Informatix AG selbst bekannt. Was passiert nun, wenn ein Teilnehmer vom Rechner „king.technokratix.com“ aus die Seite „www.kantine.informatix.de“ abruft?

1. **king.technokratix.com an dns.technokratix.com: Wie lautet die IP-Nummer von www.kantine.informatix.de?**
2. **dns.technokratix.com an dns.com: Wie lautet die IP-Nummer von www.kantine.informatix.de?**
3. **dns.com an dns.de: Wie lautet die IP-Nummer von www.kantine.informatix.de?**
4. **dns.de an dns.informatix.de: Wie lautet die IP-Nummer von www.kantine.informatix.de?**
5. **dns.informatix.de an dns.de: Die Nummer lautet 10.49.88.17**
6. **dns.de an dns.com: Die Nummer lautet 10.49.88.17**
7. **dns.com an dns.technokratix.com: Die Nummer lautet 10.49.88.17**
8. **dns.technokratix.com an king.technokratix.com: Die Nummer lautet 10.49.88.17**

Sie sehen also, dass unter Umständen eine einfache Anfrage nach einer Nummer DNS-Server quer durch die Welt beschäftigen kann. Vielleicht haben Sie auch schon einmal bemerkt, dass der Abruf der allerersten Webseite eines bestimmten Servers sehr lange dauert, während dann die folgenden Seiten sehr schnell geladen sind. In diesem Fall wurde die Verzögerung wahrscheinlich durch die langwierige Anfrage beim DNS-Server verursacht.

Große neue Welt: IPv6

Die bisherigen Beispiele bezogen sich auf die vierte Version des IP-Standards, die heute noch weitgehend genutzt wird. Prinzipiell ließen sich damit auch noch lange alle existierenden Computer verbinden: Rechnerisch könnte man über vier Milliarden Computer adressieren. Allerdings sorgt das hierarchische System der Vergabe von Nummern dafür, dass sehr viele brach liegen, weil sie den falschen „Besitzer“ haben ...

Abhilfe soll ein neues Nummernsystem schaffen, das auch bereits seit 1998 zum Standard erhoben ist, sich aber erst sehr langsam durchsetzt.

Es beruht auf einer deutlich längeren IP-Nummer von 128 Bit oder 16 Byte Länge. Damit kann man im wahrsten Sinne des Wortes astronomisch viele Adressen vergeben. Vielleicht können Sie es sich anhand eines Vergleichs besser vorstellen: Es gäbe genug Adressen, um wirklich jedem einzelnen Kubikmillimeter der Erde über 300 Millionen Adressen zuzuweisen.

Idee dahinter ist, nicht nur „echte" Computer, sondern auch Haushaltsgeräte, intelligente Kleidungsstücke, Fahrzeuge und andere Dinge unserer Umwelt internetfähig zu machen. Dann könnte zum Beispiel die Jacke im Winter schon einmal die kalte Außentemperatur an den Teekocher melden, der dann über das Smartphone den Träger fragt, ob er beim Heimkommen eine heiße Tasse Lieblingstee haben möchte. Die sonstigen Prinzipien einschließlich des Routings bleiben aber auch in IPv6 weitgehend erhalten. Sie können daher die Experimente hier in Ruhe mit kleinem, übersichtlichem Adressraum durchführen und wissen trotzdem über die große neue Welt des Internets bescheid.

Was steckt dahinter?

Sie haben bisher in diesem Kapitel kennen gelernt, wie Nachrichten im Internet geroutet werden. Das ist ein für das Verständnis sehr wichtiger Teil, aber nicht das Einzige, was es zur Kommunikation zwischen Computern zu sagen gibt.

Wie bei allen komplexen Systemen muss auch das Internet so zerlegt werden, dass die einzelnen Teile handhabbar bleiben. So konnten wir erkennen, dass sowohl die einzelnen teilnehmenden Computer als auch die Verbindungsglieder – die Router – immer nur einen kleinen Ausschnitt der Vermittlungsarbeit übernehmen und daher auch immer nur einen Ausschnitt des gesamten Wissens um die Verbindungen im Internet benötigen.

Bisher haben wir immer von „Nachrichten" oder „Datenpaketen" gesprochen, die Computer austauschen. Damit die Kommunikation jedoch funktioniert, müssen alle Internet-Teilnehmer die gleiche Sprache sprechen. Und hier gibt es wiederum eine Vielzahl technischer Details auf verschiedenen Ebenen: vom Aufbau der WWW-Seiten bis hin zur elektrischen Spannung in den Kabeln, die zwischen einzelnen Komponenten verlaufen.

Auch das kann man in seiner Gesamtheit nicht überblicken und es muss daher aufgeteilt werden. Hier gilt das hierarchische OSI-Modell. Ausgeschrieben bedeutet das „Open Systems Interconnection Reference Model", also sinngemäß übersetzt etwa „Allgemeines Modell zur Verbindung von Computersystemen".

Es teilt die Kommunikation hierarchisch in sieben Schichten ein. Um diese zu verstehen, begeben wir uns ein weiteres Mal in das Königreich von Informatix. Seine Tochter Juliana hat sich in den Prinzen Romero des benachbarten Königreichs Technokratien verliebt und tauscht nun heiße Nachrichten mit diesem aus. Das funktioniert weitgehend über die normalen Botendienste des Palastes, da die Familien der Turteltauben kein Problem miteinander haben, ganz im Gegensatz zu den Verwandten eines ähnlich klingenden Paares ...

Schicht 1

Normalerweise nutzen die beiden hierfür Tinte und Papier. Die Botschaften werden dann von den jeweiligen Kommunikatoren der Paläste überbracht. Die Paläste befinden sich allerdings in Sichtweite und so kann Juliana nachts von ihrem Fenster aus auch Lichtzeichen mit einer hellen Laterne geben (Abbildung 9.7).

Bei Papier und Tinte oder bei den Lichtzeichen handelt es sich um ein physikalisch vorhandenes Übertragungsmedium. Alle diesbezüglichen Aspekte sind im OSI-Layer 1 (Physical Layer = physikalische Schicht) festgelegt. Bei Computern ist das zum Beispiel die genaue Beschreibung der verwendeten elektrischen Signale im Netzwerkkabel oder die Farbe des verwendeten Lichtes in Glasfaserkabel.

Schicht 2

Julianas und Romeros Nachrichten werden von unterschiedliche Boten transportiert, die die Nachrichten schneller oder langsamer weiterleiten. Manchmal passiert es auch, dass ein unzuverlässiger Zusteller im Wirtshaus hängen bleibt und am nächsten Morgen den Brief vergisst. Daher nummerieren die beiden Liebenden ihre Schreiben durch. Auf diese Weise können sie feststellen, wenn einer fehlt. Der andere kann ihn dann nochmals schicken (Abbildung 9.8).

Die Nummerierung ist eine Art rudimentäre Fehlerkorrektur. Bei der Verbindung zwischen zwei Computern kann ebenfalls vieles schiefgehen. Ein Datenpaket kann ganz verloren gehen oder aber auch durch Störungen auf der Leitung verfälscht ankommen. Daher werden zusätzliche Daten wie Nummern, Prüfsummen usw. eingefügt, mit denen man kontrollieren kann, ob eine Nachricht unverändert angekommen ist. Wenn nicht, wird sie zum Beispiel nochmals angefordert. Wie das genau funktioniert, ist in OSI-Layer 2 (Data Link Layer = Datenverbindungsschicht) festgelegt. Auch wird beschrieben, wie die einzelnen Datenpakete prinzipiell aufgebaut sind und wie ein Rechner einen anderen im gleichen LAN anspricht.

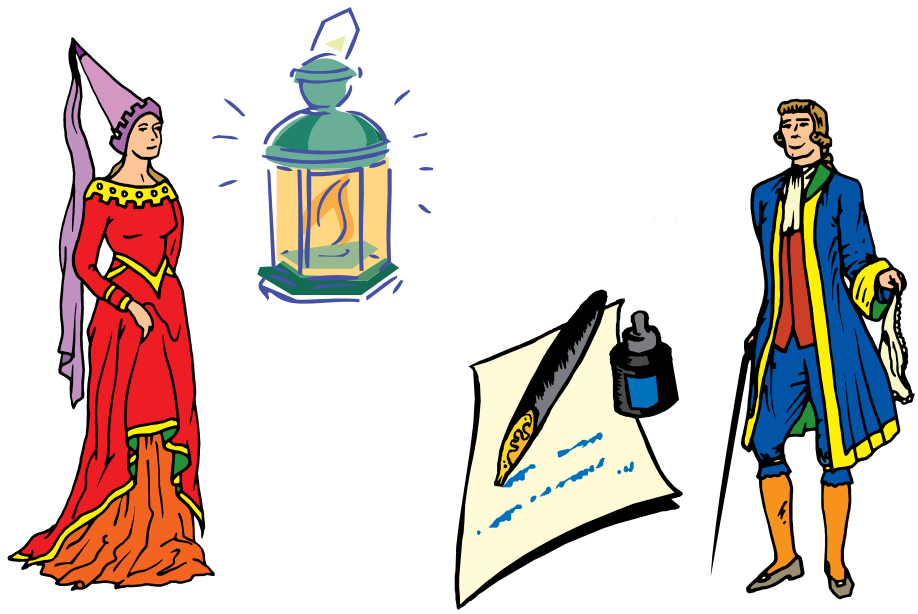


Abbildung 9.7
OSI-Layer 1: Die physikalische Schicht beschreibt die Signale des Übertragungsmediums.

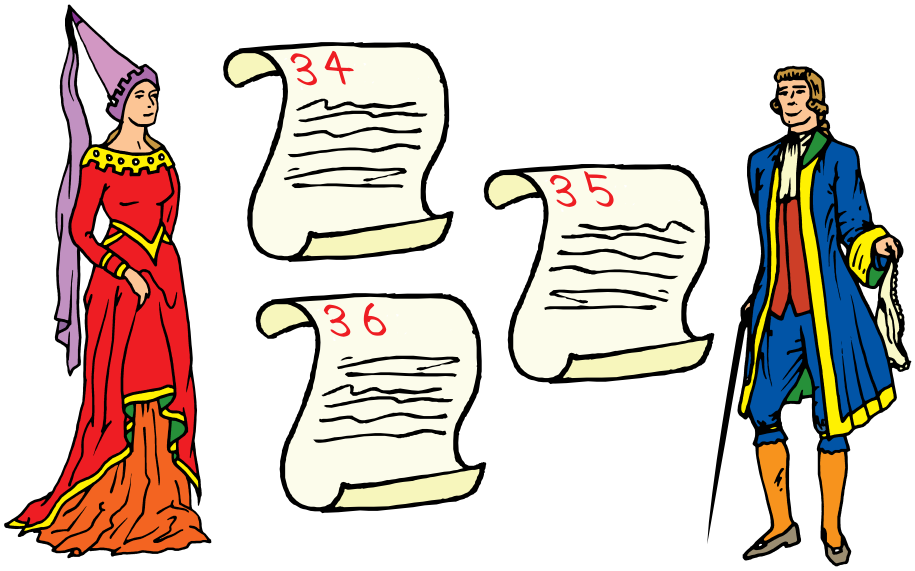


Abbildung 9.8
OSI-Layer 2: Die Datenverbindungsschicht sorgt für die zuverlässige Übertragung einer Nachricht.

Schicht 3

Mittelgroße Netzwerke können heute auch allein auf Basis dieser zweiten Schicht effektiv arbeiten und benötigen keinen Router. Ein sogenannter Switch verbindet die einzelnen Computer und leitet Datenpakete nur an die korrekten Adressaten weiter. Er stellt dabei meistens automatisch fest, welcher Computer an welchem seiner Anschlüsse hängt, so dass kaum administrativer Aufwand nötig ist.

In unserem Beispiel müssen die beiden Turteltauben natürlich ihre Briefe korrekt adressieren, damit diese ankommen und vom Botendienst richtig zugestellt werden. Der Adressat und der Absender müssen daher auf dem Umschlag stehen (Abbildung 9.9).

Die prinzipielle Kommunikation der Computer in einem großen Netz wie dem Internet ist in OSI-Layer 3 (Network Layer = Netzwerkschicht oder Paketschicht) geregelt. Die Erklärungen vom Anfang des Kapitels beziehen sich fast alle auf die Netzwerkschicht, da das sogenannte „Routing“ hier beschrieben ist, also die Art und Weise, wie

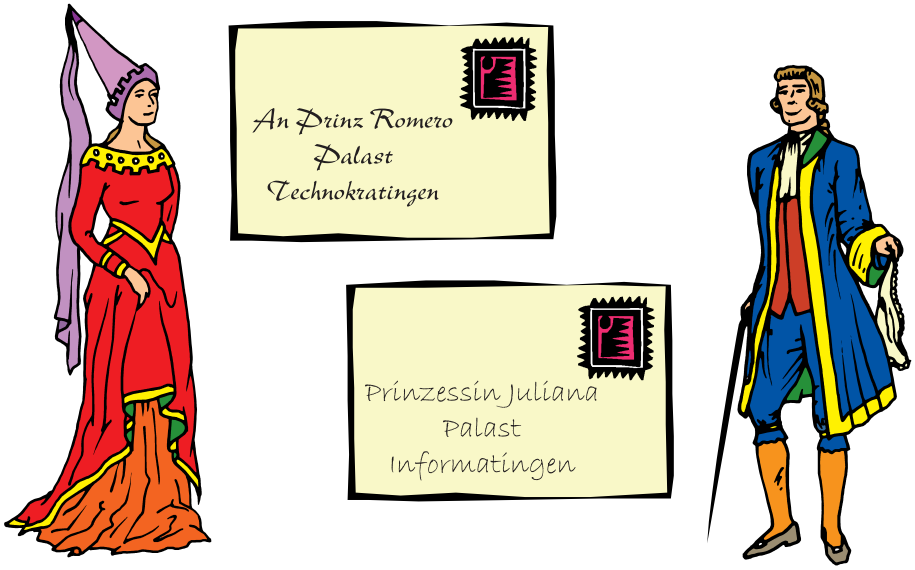


Abbildung 9.9
OSI-Layer 3: Die Netzwerkschicht sorgt für die Vermittlung der Nachrichten im Internet.

Datenpakete im Netz hin- und hergeschickt werden, um irgendwann auf wundersame Art und Weise am Ziel anzukommen.

Schicht 4

Juliana und Romero sind nicht die Einzigen, die in den Palästen Briefe schreiben. Auch die gesamte Verwaltung der Königreiche, vom Steuereintreiber bis zum Küchenchef, beruht auf schriftlicher Kommunikation. Daher hat der Oberpostmeister viel zu tun. Er muss dafür sorgen, dass auch bei hoher Belastung jeder Brief transportiert wird. Wenn alle verfügbaren Boten unterwegs sind, gibt er erst die Nachrichten mit einer hohen Priorität weiter, also etwa persönliche Briefe des Königs oder solche, die mit „Dringend“ beschriftet sind (Abbildung 9.10).

Bei den „echten“ Netzwerken wird dies von OSI-Layer 4 (Transport Layer = Transportschicht) erledigt. Hier wird die komplette Vermittlung der Datenpakete vom Ursprung zum Ziel geregelt. Auch hier kann man bestimmte „Vorfahrtsregeln“ einplanen. Wenn Sie zum Beispiel mit jemandem über das Netzwerk telefonieren, enthalten die Datenpakete jeweils einen kleinen Teil Ihrer digitalisierten Sprache. Kleine Verzögerungen von Sekundenbruchteilen führen dann bereits zu einer Verstümmelung der Sprache. Andere Daten haben wiederum eine niedrigere Priorität: Ob eine WWW-Seite innerhalb einer zehntel oder einer halben Sekunde angezeigt wird, registrieren wir normalerweise gar nicht.

Schicht 5

Romero hat sich ein Spiel ausgedacht: Er schreibt den ersten Absatz einer Geschichte und schickt ihn an Juliana, sie verfasst dann den nächsten Absatz und so weiter, bis einer von beiden die Geschichte beendet. Oft dauert es etwas, bis einer von beiden einen neuen Absatz geschrieben hat, und daher schreiben sich beide währenddessen auch „normale“ Briefe. Manchmal kommt es sogar vor, dass die beiden zwei oder mehr Geschichten gleichzeitig bearbeiten. Damit man sofort erkennt, welche Fortsetzungsgeschichte in einem Brief weitergeführt wird, malen Romero und Juliana ein individuelles Symbol für jede ihrer Geschichten auf den Umschlag (Abbildung 9.11).

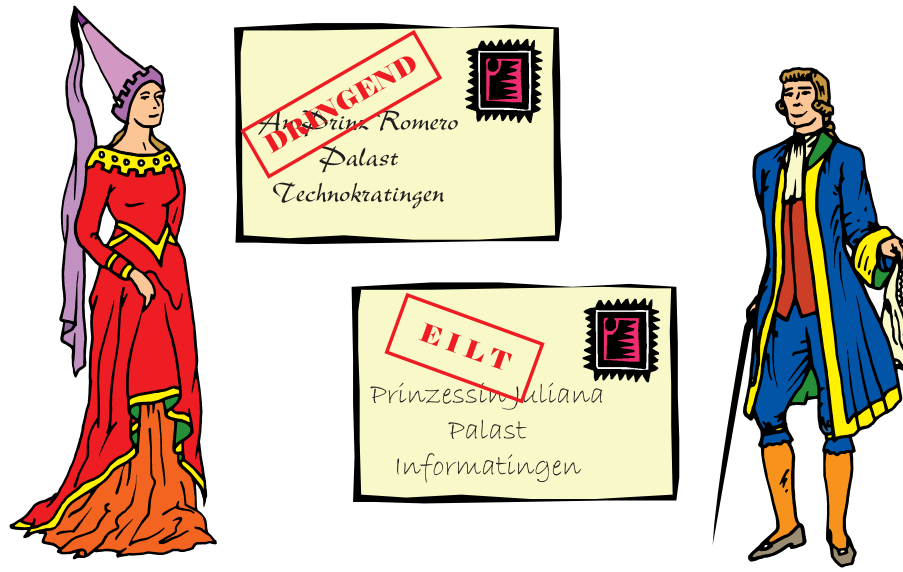


Abbildung 9.10
OSI-Layer 4: Die Transportschicht regelt den Verkehr der Datenpakete.

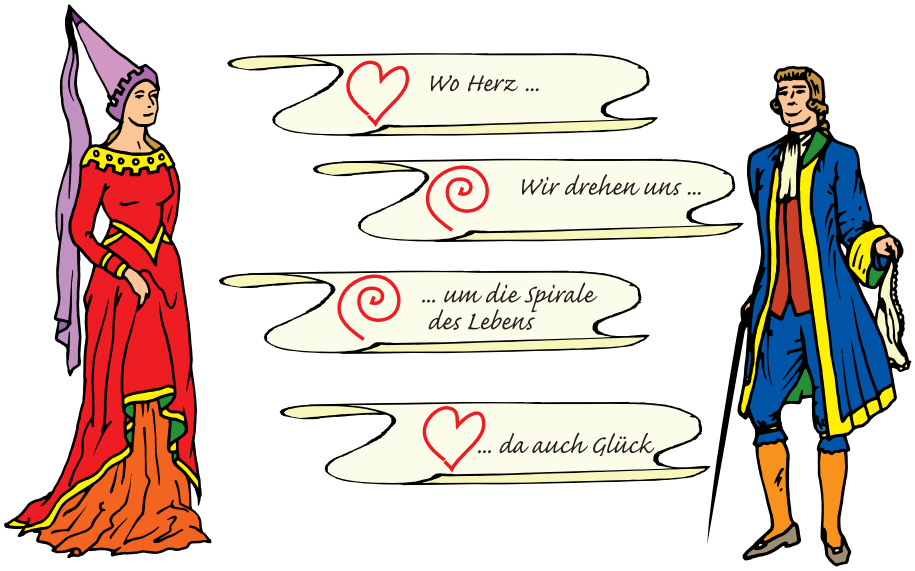


Abbildung 9.11
OSI-Layer 5: Die Sitzungsschicht hält virtuelle Verbindungen zwischen zwei Computern im Internet.

Das Internet ist paketorientiert. Das bedeutet, dass die Kommunikation über kleinere Datenpakete stattfindet und nicht durch eine feste Verbindung zwischen zwei Computern. Wenn Sie Ihrem Nachbarn eine E-Mail schicken, könnte es theoretisch passieren, dass das erste und dritte Datenpaket auf kürzestem Wege zugestellt werden, während das zweite den Umweg über Australien nimmt. Router sollen oft die Last auf den Datenleitungen gleichmäßig verteilen und prinzipiell könnte dadurch so etwas zustande kommen. Das zweite Paket würde natürlich auch gegenüber dem ersten und dritten verzögert ankommen, wahrscheinlich kommt das dritte Paket sogar vor dem zweiten am Ziel an.

Trotzdem soll die E-Mail am anderen Ende wieder zusammengesetzt werden, so als wäre sie über eine einzige, feste Verbindung geschickt worden. Die gesamte Übertragung findet hierfür in einer „Sitzung“ statt, die von OSI-Layer 5 (Session Layer = Sitzungsschicht) beschrieben wird. Ähnlich dem Symbol, das Romero und Juliana auf die Briefe malen, beschreibt im Internet eine eindeutige Nummer die Zugehörigkeit eines Datenpakets zu einer Sitzung.

Schicht 6

Nachts sind die Botendienste beider Königreiche geschlossen. Daher haben die zwei Liebenden ausgemacht, sich dann ihre Nachrichten anhand von Lichtzeichen mit einer starken Laterne zu schicken. Da nur „Licht an“ und „Licht aus“ erkennbar ist, verwenden sie das Morsealphabet (s. Kapitel „Von Kamelen und dem Nadelöhr“), um die einzelnen Zeichen zu codieren (Abbildung 9.12).

Computer übermitteln untereinander in den bisherigen fünf OSI-Schichten prinzipiell nur Zahlen. Erst OSI-Layer 6 (Presentation Layer = Darstellungsschicht) legt fest, wie diese Zahlen in Buchstaben oder zum Beispiel auch Zeichnungen gewandelt werden, um für uns Menschen verständlicher zu sein. Eine wesentliche Vereinbarung ist hier zum Beispiel, welche Zeichentabelle genutzt wird (zwei sehr bekannte sind ASCII und Unicode). Auch die Komprimierung von Daten, um das Netz besser auszunutzen, wird in dieser Schicht beschrieben.

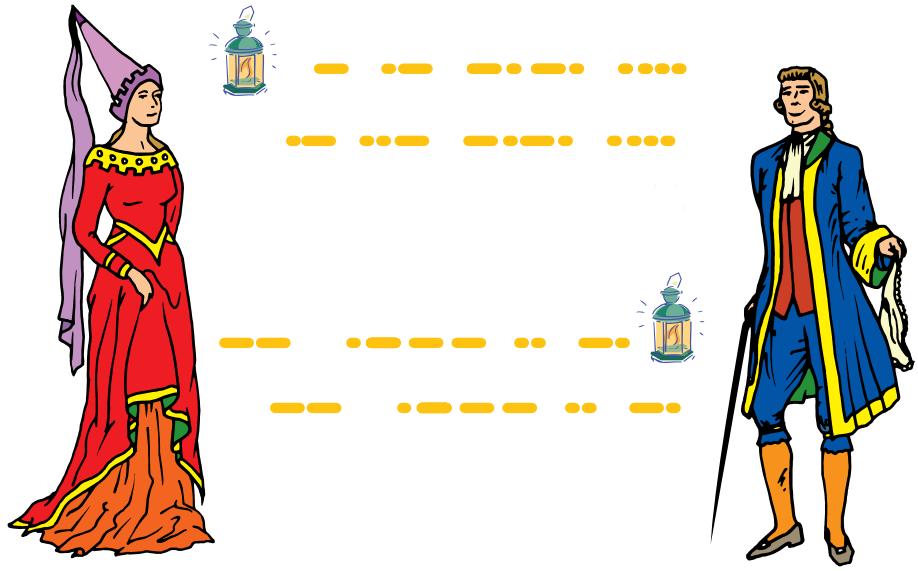


Abbildung 9.12
OSI-Layer 6: Die Darstellungsschicht beschreibt die Codierung der Nachrichten.

Schicht 7

Glücklicherweise liegen die beiden Königreiche von Informatix und Technokratix so dicht zusammen, dass in ihnen die gleiche Sprache gesprochen wird. Allerdings studiert Juliana Philosophie und Romero Physik. So kommt es manchmal zu kleinen Missverständnissen, wie sie zwischen Geistes- und Naturwissenschaftlern üblich sind. Am Ende können beide sich selbstverständlich immer in der Sprache der Liebe verständigen und verstehen sich wieder (Abbildung 9.13).

Menschen spezialisieren sich oft auf spezielle Fachgebiete und übernehmen die dort verbreiteten, ganz eigenen Sprachen. So sind sie für Außenstehende manchmal kaum zu verstehen. In viel stärkerem Maße trifft das auf Computer zu. Für jede Art von Dienstleistung, die sie erbringen, kommunizieren sie mit einem ganz eigenen Vokabular (der Fachbegriff ist „Protokoll"). So gibt es eigene Sprachen für das Web (HTTP = Hypertext Transfer Protocol), E-Mail (SMTP = Small Mail Transfer Protocol), den Austausch von Dateien (FTP = File Transfer Protocol) oder sogar für das Computer-Pendant der Zeitansage (NTP = Network Time Protocol). Diese sind in OSI-Layer 7 (Application Layer = Anwendungsschicht) spezifiziert. Damit zwei Computer miteinander reden können, müssen sie sich also auch auf dieser Ebene verstehen. Ein E-Mail-Programm hat zum Beispiel Schwierigkeiten, mit einem WWW-Server zu reden, weil es SMTP spricht, der Server aber nur HTTP versteht.

Für verschiedene Zwecke werden einer Nachricht also immer weitere Daten hinzugefügt, um sie im Internet zu verschicken – genau wie bei einem „echten" Brief: Er wird geschrieben, unterschrieben, in einen Umschlag gepackt, mit den Adressen versehen. Eine Marke wird aufgeklebt. Die Post stempelt den Umschlag usw.

Sicherlich haben Sie auch in weiteren Bereichen einschlägige Erfahrung bezüglich der Kommunikation mit anderen Menschen gemacht. Nehmen Sie sich Romero und Juliana zum Vorbild und versuchen, Ihnen bekannte Prozesse in das OSI-Modell einzuordnen.

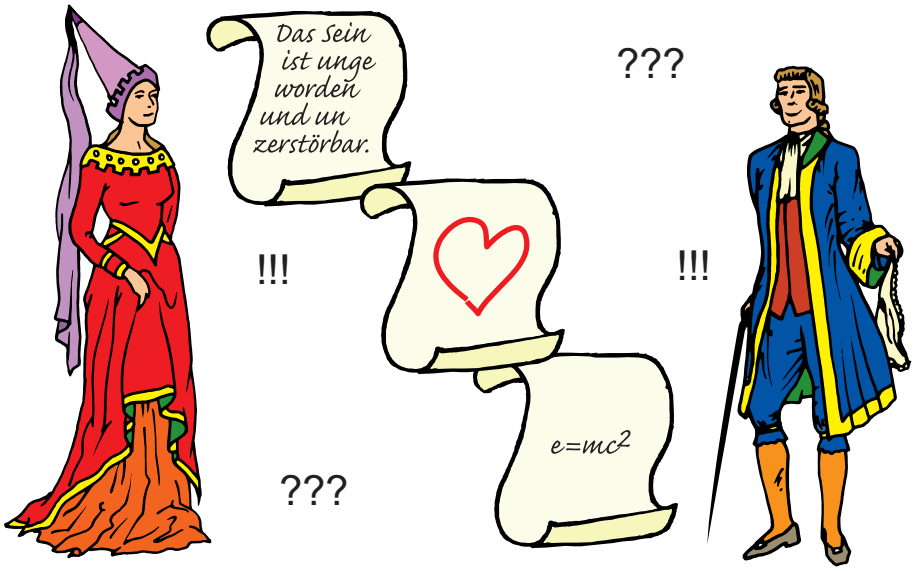


Abbildung 9.13
OSI-Layer 7: Die Anwendungsschicht interpretiert die Inhalte der Datenpakete.

Vielleicht fragen Sie nun, warum das OSI-Modell so wichtig ist, dass ich es hier in diesem populärwissenschaftlichen Buch aufgreife. Die erste Antwort lautet, dass das Internet heute eines der wichtigsten Kommunikationsmittel geworden ist und das Wissen um den grundlegenden Aufbau viele Probleme mit der Bedienung oder dem Aufbau privater Netzwerkkomponenten vereinfacht.

Es gibt aber meiner Meinung nach noch einen anderen, gewichtigeren Aspekt: Ob Computer vernetzt werden oder ob es sich um Kommunikationsprozesse zwischen natürlichen Gesprächspartnern handelt – die Prinzipien der Informatik funktionieren auch hier und sorgen dafür, komplexe Sachverhalte und unübersichtliche Systeme einfach und handhabbar zu machen. Das gelingt nur, weil die Informatik immer wieder Prinzipien des menschlichen Denkens aufgreift und auf technische Systeme überträgt.

Auch wenn das Beispiel mit Romero und Juliana bewusst märchenhaft gehalten ist: Die Prinzipien kennen wir aus unserem Alltag. Wenn wir etwa einen Brief verschicken, werfen wir ihn in den nächsten Briefkasten und gehen davon aus, dass er schon irgendwie korrekt zugestellt werden wird. Das gesamte System dazwischen – Abholung, Verteilzentrum, Zustellung usw. – ist für uns lediglich eine sogenannte Black Box mit Namen „Post“. Das ist keine Missachtung, sondern eine Tugend, weil sich auf diese Weise alle auf das konzentrieren können, was sie am besten beherrschen. Auch das ist eine Spielart der Modellbildung.

In vielen Wissenschaften gibt es das Prinzip der Modellbildung: Die Wirklichkeit wird durch ein Modell beschrieben, so dass die eigenen Beobachtungen damit möglichst vollständig erklärbar sind. Die Modelle verfeinern sich selbstverständlich. Für unsere archaischen Vorfahren war ein Weltbild völlig plausibel, in dem die flache Erde im Mittelpunkt steht, mit an den Himmel gehefteten Sternen und feurigen Rennwagen, die dort umherkreisen. Im Laufe der Zeit hat sich das Modell dann den neuen Erkenntnissen und Anforderungen angepasst: Zunächst rückte die Sonne in den Mittelpunkt, die Erde wurde auf die Umlaufbahn verwiesen. Im heutigen Modell ist auch unsere Sonne nur ein winziges Licht einer großen Galaxis.

Auch in der Informatik dreht sich alles um Modelle. Meistens nicht, um die Wirklichkeit möglichst exakt zu beschreiben, sondern um einen für die intendierte Problemlösung möglichst passenden Ausschnitt zu beschreiben. Erinnern Sie sich noch an das

Prinzip der Abstraktion? So stellen wir unsere Welt zum Beispiel im Computer genau so dar, dass alle notwendigen Informationen zum Berechnen eines kürzesten Weges vorhanden sind – als Graph.

Am Internet sehen Sie, dass dies auch umgekehrt möglich ist: Hier werden bekannte Strukturen der Realität verwendet, um ein technisches System nach dem gleichen Modell aufzubauen. Man folgt quasi dem Beispiel der Wirklichkeit, und dafür gibt es einen speziellen Begriff: Paradigmenbildung.

Paradigma

Das Paradigma ist ein (Denk-)Muster oder Vorbild, anhand dessen ein technisches System oder ein abstraktes Konzept konstruiert wird. Dadurch wird es für die Anwender leichter, sich im System zurechtzufinden.

Bestes Beispiel für Paradigmenwechsel in der Informatik sind die Programmiersprachen: Zunächst sehr stark auf die direkte Umsetzung der Bedürfnisse der technischen Systeme zugeschnitten, wurden nach und nach immer mehr „menschliche" Konzepte übernommen, zum Beispiel mit der objektorientierten Programmierung.

Bei dieser besteht die Software in einer Ansammlung von Objekten, die miteinander kommunizieren – genau wie an einer „echten" Arbeitsstelle: Dort sind die meisten Angestellten damit beschäftigt, Tätigkeiten an andere zu delegieren. Wenn sie das gut machen, läuft auch die Produktion gut (zugegebenerweise nur, wenn die Arbeit auch irgendwo unten in der Hierarchie wirklich verrichtet wird, idealerweise von Maschinen ...). Diesem Beispiel folgend, besteht die Programmierung weitgehend darin, den Objekten richtiges Delegieren „beizubringen".

Resümee

Der Aufbau des Internets zeigt, wie menschlich letztlich die Informationstechnologie ist: Im Prinzip bedient sie sich jahrhundertealter Strukturen und Vorgehensweisen. Man darf dies auf keinen Fall mit einer Schwäche verwechseln:

Neue technische Möglichkeiten verlangen zunächst immer von uns Menschen, dass wir uns anpassen, um die Vorteile zu genießen. Beispiel hierfür ist die Entwicklung der Computer, die zunächst nur von hoch spezialisierten Experten bedient werden konnten. Inzwischen sind die Computer der menschlichen Denkart weitgehend angepasst (wenn auch viele Menschen das stark anzweifeln, sobald der DVD-Player mal wieder nicht so funktioniert wie erwartet).

Genauso ist dies auch mit Datennetzen: Zunächst technisch sehr kompliziert und aufwendig gestaltet, waren sie nicht zuverlässig nutzbar. Man hat sich daher Strukturen ausgedacht, die „menschlicher" ausgeprägt und dadurch beherrschbar sind. Das heutige, moderne Internet ist aus diesem Grund prinzipiell immer noch mit der Hierarchie einer mittelalterlichen Pferdepost vergleichbar! Nur viel größer und natürlich viel schneller ...

10. Alles im Fluss

Viel Wasser bedeutet auch viel Abwasser. Das erkannte bereits der römische König Tarquinius Priscus im 5. Jahrhundert v. Chr. und ließ die berühmte Cloaca Maxima, das in Teilen heute noch verwendete Abwassersystem, anlegen. Durch sie konnten die Sümpfe zwischen den berühmten sieben Hügeln trockengelegt werden – Voraussetzung für Wachstum, Wohlstand und ein Weltreich, das die Geschichte über ein Jahrtausend geprägt hat.

Wie groß muss aber ein Abwasserkanal dimensioniert werden? Genaue Berechnungen konnte man nicht anstellen und so sind alle Leitungswege der Antike und des Mittelalters nach dem Prinzip „lieber zu groß als zu klein" eher gigantisch gebaut worden. Trotzdem konnte es immer wieder zu Engstellen kommen, weil sich das Wasser an Stellen sammelte, die man so nicht vorhergesehen hatte.

Heute haben wir (meistens) nicht mehr die Ambition, Monumentalbauten zu errichten, vielmehr wollen wir sie so auslegen, dass sie dem gewünschten Zweck genau entsprechen. Mehr noch: Wir wissen heute, dass es insbesondere für die Frischwasserversorgung relevant ist, diese sehr genau auf den Bedarf auszurichten. Ansonsten besteht die Gefahr, dass sich im Trinkwasser aufgrund zu geringer Bewegung Keime bilden – es steht dann im wahrsten Sinne des Wortes ab.

Daher stellen wir uns auch heute noch die oben erwähnte Frage nach der Dimensionierung. Sie ahnen es bereits: Auch hier kann die Informatik eine entscheidende Hilfe sein.

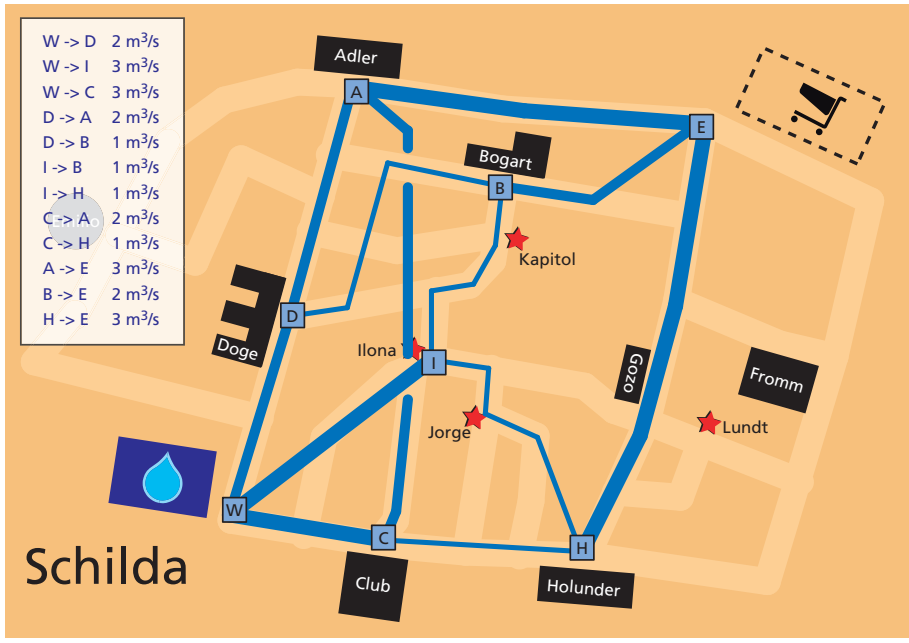
Vereinfachen wir das Problem zunächst: Wir gehen davon aus, dass irgendwo Wasser gepumpt wird und wir dieses durch ein verzweigtes Leitungsnetz an eine andere Stelle transportieren möchten – zum Beispiel in Ihr Haus. Jede Leitung des Netzes hat dabei eine Fließrichtung (weil Wasser sich nun einmal freiwillig immer nach unten bewegt) und eine fest vorgegebene Kapazität, die von allen möglichen Umständen abhängt, die wir gar nicht im Detail betrachten möchten: Durchmesser der Rohre, Gefälle, Leistung zusätzlich installierter Pumpen usw.

Viel Wasser den Berg hinunter …

Abbildung 10.1 zeigt ein Beispiel für ein solches Problem: Schilda kennen Sie bereits aus dem ersten Kapitel. Ein neues Einkaufszentrum „IT Discount" soll gebaut und muss mit Wasser versorgt werden. Die Architekten veranschlagen, dass hier zu Spitzenzeiten etwa 7 m^3 pro Sekunde verbraucht werden. Im Wasserwerk kann man diese Menge ohne Probleme bereitstellen. Die Frage ist nun, ob das Leitungsnetz dieses Wasser auch transportieren kann oder ob man zusätzliche Leitungen verlegen muss.

In der Karte sind die möglichen Wasserleitungen eingezeichnet sowie die Kapazität, mit der zusätzlich etwas fließen kann. Da die Schildaer Wasserversorgung keine Pumpen kennt, ist man auf ein Gefälle angewiesen, so dass man die Leitungen nur in eine

Abbildung 10.1
Schildas Wasserleitungen



Richtung benutzen kann. Die Legende in Abbildung 10.1 gibt an, zwischen welchen Verbindungsschächten welche zusätzlichen Mengen fließen können.

Sie kennen aus dem ersten Kapitel bereits die Vorgehensweise eines Informatikers, wenn sich er ein solches Problem vornimmt – Abstraktion: Zunächst werden alle Informationen weggelassen, die offensichtlich zur Problemlösung nicht wichtig sind. Machen Sie dies und zeichnen Sie den Stadtplan ohne störende Details.

Sicherlich spielen die Hotels und die Straßen Schildas keine Rolle für die Wasserversorgung, die unterirdisch erfolgt. Wichtig sind die Verbindungsschächte und die Information, wie viel Wasser zwischen ihnen fließen kann. Dabei sind die Länge und Form einer Leitung nicht relevant, denn die Kapazitäten werden daraus nicht hergeleitet, sondern stehen durch die Legende fest.

Wir können also ein Diagramm zeichnen, das – ähnlich wie bei der Berechnung des kürzesten Weges – nicht mehr maßstabsgetreu, dafür aber übersichtlich ist. Abbildung 10.2 zeigt eine mögliche Lösung.

Die Pfeile symbolisieren dabei eine Wasserleitung sowie ihre Fließrichtung, die Beschriftung gibt die Kapazität der Leitung in Kubikmetern pro Sekunde an. Momentan werden die Leitungen noch nicht genutzt, es fließt nichts hindurch!

Ⓦ ist dabei der Punkt, an dem das Trinkwasser bereitgestellt wird – die Quelle. Am Punkt Ⓔ wird das Wasser verbraucht, man spricht hier von einer „Senke".

Vielleicht fragen Sie sich jetzt, wo das eigentliche Problem liegt: Offenbar gehen von Ⓦ Leitungen mit einer Gesamtkapazität von 8 m^3 pro Sekunde los und bei Ⓔ kommen Leitungen mit einer Kapazität von 8 m^3 pro Sekunde an. Demnach können wir also genügend Wasser losschicken und es kommt auch genügend Wasser an. Problem gelöst?

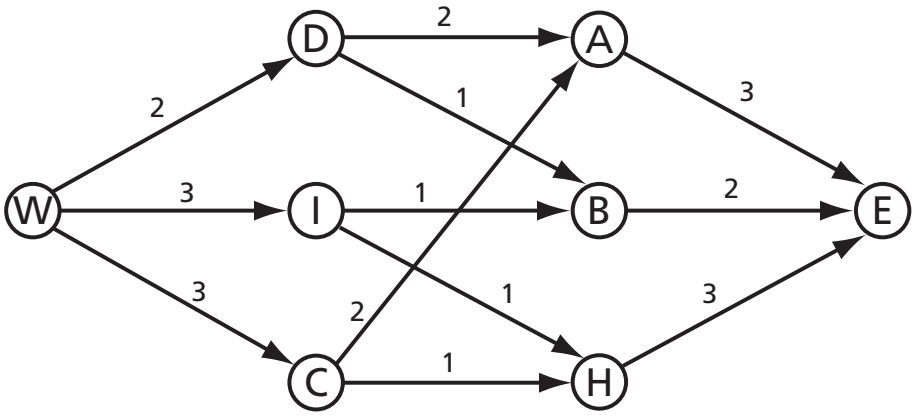


Abbildung 10.2
Die Leitungswege in einem ordentlichen Diagramm

Die nächste Abbildung 10.3 macht klar, warum die Kapazitäten, die von der Quelle weggehen bzw. an der Senke ankommen, nicht das einzige Kriterium für das Funktionieren oder Scheitern der Aufgabe sind.

Hier sind die Kapazitäten der Leitungen im mittleren Teil deutlich herabgesenkt. Überlegen Sie, was passiert, wenn wir tatsächlich mehr als 4 m³ Wasser pro Sekunde vom Wasserwerk losschicken.

Richtig! Wenn 2 m³ Wasser von Ⓦ nach Ⓓ zufließen, aber durch die Leitungen von Ⓓ aus nur 1,5 m³ abfließen können, gibt es einen See am Punkt Ⓓ. Pro Sekunde stauen sich hier 0,5 m³ Wasser!

Obwohl die Leitungen von Ⓦ ausgehend und bei Ⓔ zugehend genügend Kapazität für die Aufgabe haben, kann durch das Leitungsnetz aus Abbildung 10.3 nicht genügend Wasser geleitet werden!

Wenden wir uns wieder der ursprünglichen Aufgabe zu. Ein weiteres Prinzip der Informatik ist das Auffinden von Invarianten. Das kann entscheidende Hinweise auf eine Lösungsstrategie geben.

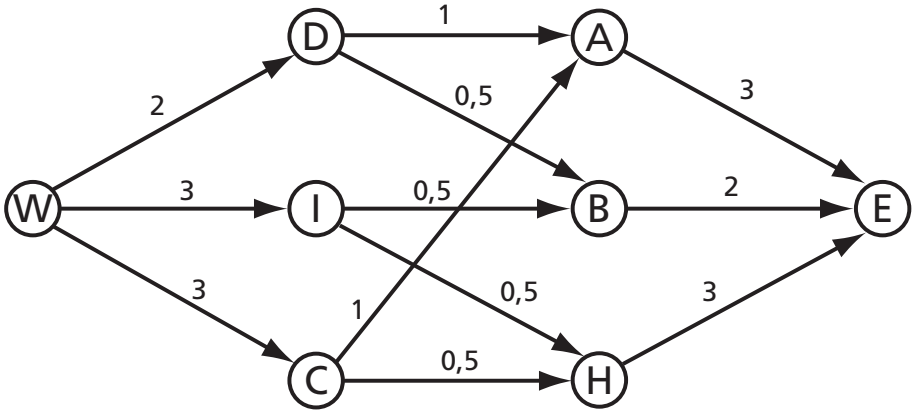


Abbildung 10.3
In dieser Konstellation wird ganz offensichtlich, dass maximal 4 m³ durch das System von Ⓦ nach Ⓔ fließen können.

Invariante

Eine Invariante ist eine Größe oder Gesetzmäßigkeit, die während der Lösung eines Problems bzw. während der Durchführung eines Algorithmus immer gleich bleibt.

Wie wir an dem kleinen Beispiel oben gesehen haben, spielt offenbar für die Lösung unseres Problems der tatsächliche Fluss durch die Wasserleitungen eine Rolle. Die Kapazität stellt hierfür die Obergrenze dar. Welche Gesetzmäßigkeiten können wir für die Flüsse untereinander annehmen? Das Betrachten von Beispielen kann Klarheit verschaffen.

In Abbildung 10.4 habe ich tatsächliche Flüsse auf den Leitungen eingezeichnet. Welche der drei Möglichkeiten a bis c sind realistisch, welche können so in Realität nicht vorkommen?

Die Möglichkeiten (a) und (b) scheiden aus! Sie haben sicherlich zwei Invarianten erkannt:

Haben Sie es bemerkt?
In geschlossenen Systemen folgt selbstverständlich die speziellere erste Invariante aus der zweiten, denn wenn eine Menge Wasser losfließt und nirgendwo versickern oder sich vermehren kann, muss sie wohl irgendwann in gleicher Menge am Ziel ankommen.

- Die Menge Wasser, die aus Ⓦ fließt, muss genauso hoch sein wie die Menge Wasser, die bei Ⓔ ankommt. Käme weniger an, gäbe es ein Leck in den Leitungen, käme mehr an, müsste ein zusätzlicher Zulauf für die Zunahme sorgen. Beides wollen wir jedoch ausschließen.
- Die Menge Wasser, die in einen beliebigen Knoten hineinfließt, muss identisch sein mit der Menge Wasser, die hinausfließt. Ansonsten gäbe es entweder eine Überschwemmung oder das Wasser vermehrte sich auf wundersame Weise.

Nur in Lösung (c) sind alle Invarianten erfüllt.

Die nächste Aufgabe ist, eine Vorgehensweise zur Lösung der eigentlichen Aufgabe zu finden, die alle Invarianten berücksichtigt und den maximalen Gesamtfluss feststellt:

Gegeben sei ein Leitungsnetzwerk, durch das nichts fließt, dessen Leitungen aber eine Kapazitätsgrenze besitzen. Resultat soll das Netzwerk sein, durch das eine maximale Menge Wasser zwischen der definierten Quelle und der definierten Senke fließt. Denken Sie wieder an die Sicht des Informatikers: Ein großes Problem ist besser handhabbar, wenn man es in kleinere Teilprobleme aufteilen kann.

Welche Teilprobleme können Sie hier entdecken?

Das Gesamtproblem besteht darin, den Wasserfluss zwischen dem Wasserwerk und dem Einkaufszentrum von null so zu erhöhen, dass er maximal wird. Ein Teilproblem ist also, den Wasserfluss ein wenig zu erhöhen. Wenn man oft genug den Fluss zwischen Quelle und Senke ein bisschen erhöht, so dass er nicht mehr weiter erhöhbar ist, hat man offenbar das Gesamtproblem gelöst.

Und welches Teilproblem ist noch kleiner, als den Fluss ein wenig zu erhöhen? Genau: den Fluss um eine Einheit zu erhöhen. Wie erhöht man den Fluss aber konform mit

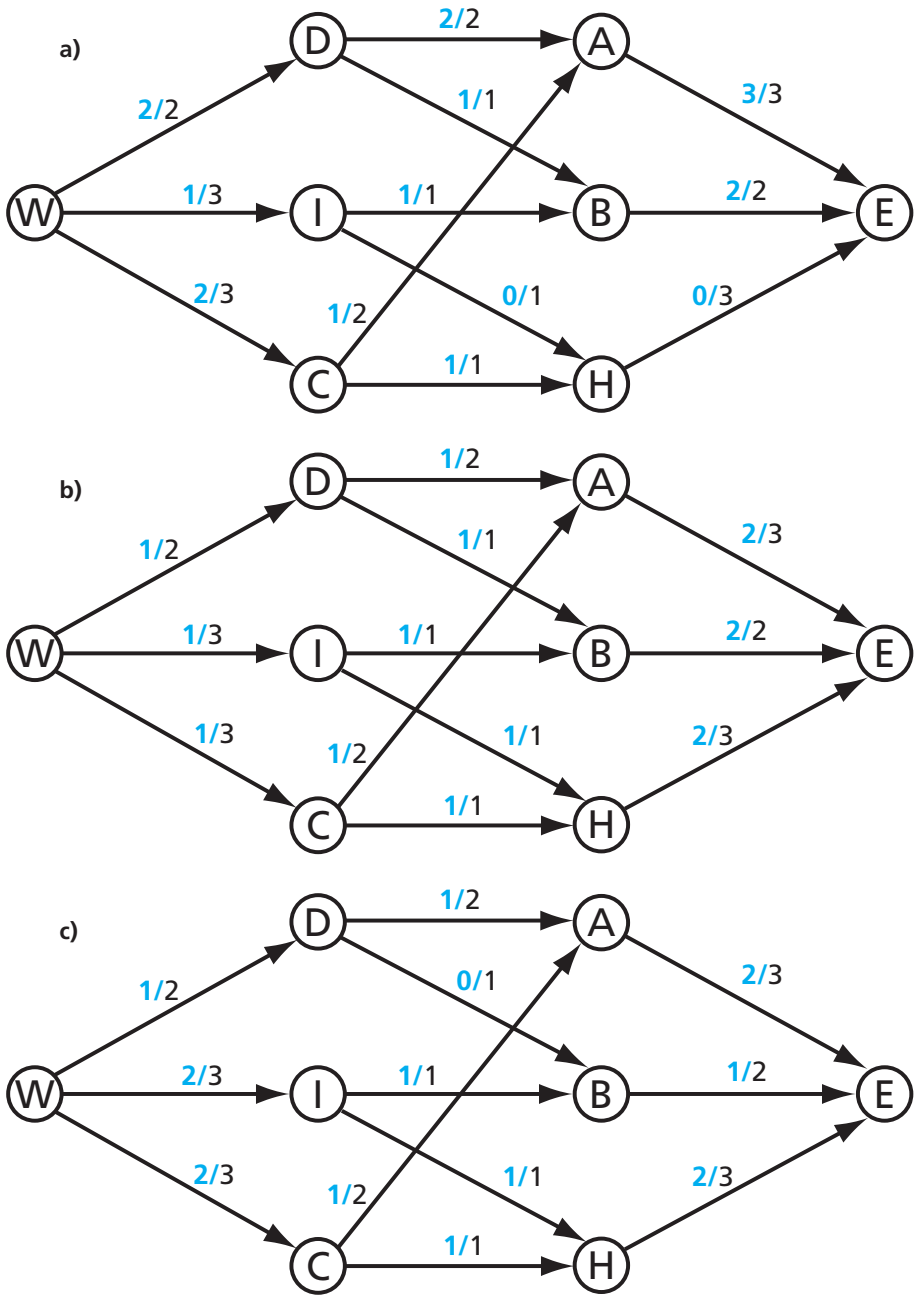
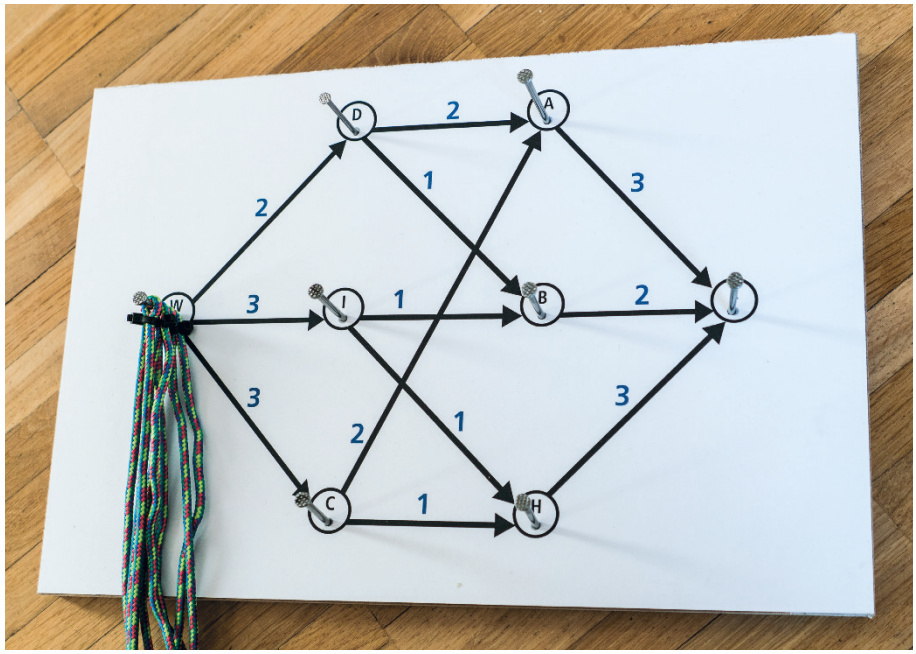


Abbildung 10.4
Die blauen Zahlen stellen die tatsächliche Wassermengen dar, die durch die Leitungen fließen.

den Invarianten um eine Einheit? Wie so oft in der Informatik hilft hier die Anschauung, und diese wollen wir über eine kleine Bastelarbeit herstellen.

Kopiervorlage 10.K1 zeigt den nun bereits mehrfach verwendeten Graphen zum Stadtplan von Schilda in einem etwas breiteren Format. Nehmen Sie diesen und kleben ihn auf ein kleines Brettchen, zum Beispiel ein Frühstücksbrettchen. Schlagen Sie danach in alle mit einem kleinen grauen Kreis markierten Stellen Nägel. Nun brauchen Sie nur noch ein paar Fäden – acht Stück reichen aus, die Sie an den Nagel bei Ⓦ knoten. Fertig ist der analoge Flusscomputer. Abbildung 10.5 zeigt das Ergebnis.

Abbildung 10.5
Schildas Wasserleitungen als Experiment



Falls Sie kein Brett griffbereit haben, können Sie auch etwas aus Pappe und gebogenen Büroklammern basteln oder Ihre Kreativität in einer andere Konstruktion fließen lassen. Wichtig ist, dass am Ende die Schnüre über die Nägel von der Quelle Ⓦ zur Senke Ⓔ gespannt werden können.

Stellen Sie sich vor, jede Schnur steht für einen Kubikmeter Wasser pro Sekunde. Sie belegen eine Rohrleitung, indem Sie die Schnur von Ⓦ nach Ⓔ spannen. So können Sie zum Beispiel die erste Schnur über Ⓦ – Ⓓ – Ⓐ – Ⓔ verlaufen lassen. Die Kapazitäten sind damit aber noch nicht ausgeschöpft und Sie können eine zweite Schnur über den gleichen Weg legen. Nun sind die Leitungen zwischen Ⓦ und Ⓓ sowie zwischen Ⓓ und Ⓐ ausgeschöpft, zwischen Ⓐ und Ⓔ könnte prinzipiell noch 1 m^3/s zusätzlich fließen.

Merken Sie, wie die Schnüre ganz implizit sicherstellen, dass die Invarianten eingehalten werden? Sie spannen die Schnur zwischen Ⓔ und Ⓦ. Damit wird automatisch die erste Invariante sichergestellt: Ein von der Quelle ausgehender Fluss kommt auch bei der Senke an. Gleichzeitig gilt für alle Knoten dazwischen, dass die Schnur dort weder anfängt noch endet. Damit muss sie – wenn sie den Knoten überhaupt tangiert – dort hin-, aber auch wieder wegführen. Hurra – die zweite Invariante ist ebenfalls sichergestellt.

Machen Sie – ausgehend von diesen beiden, in Bild 10.6 erkennbaren, bereits festgelegten Wasserflüssen – weiter und versuchen Sie, so viele Schnüre wie möglich so zu verlegen, dass die Kapazitäten nicht überlastet werden. Wie viel Wasser kann pro Sekunde vom Wasserwerk zum Einkaufszentrum geleitet werden?

Wahrscheinlich kommen Sie auch auf 6 m^3/s, Abbildung 10.7 (auf der übernächsten Seite) zeigt die Situation. Für das geplante Einkaufszentrum bedeutet das also nichts Gutes – die geforderte Menge Wasser kann nicht angeliefert werden.

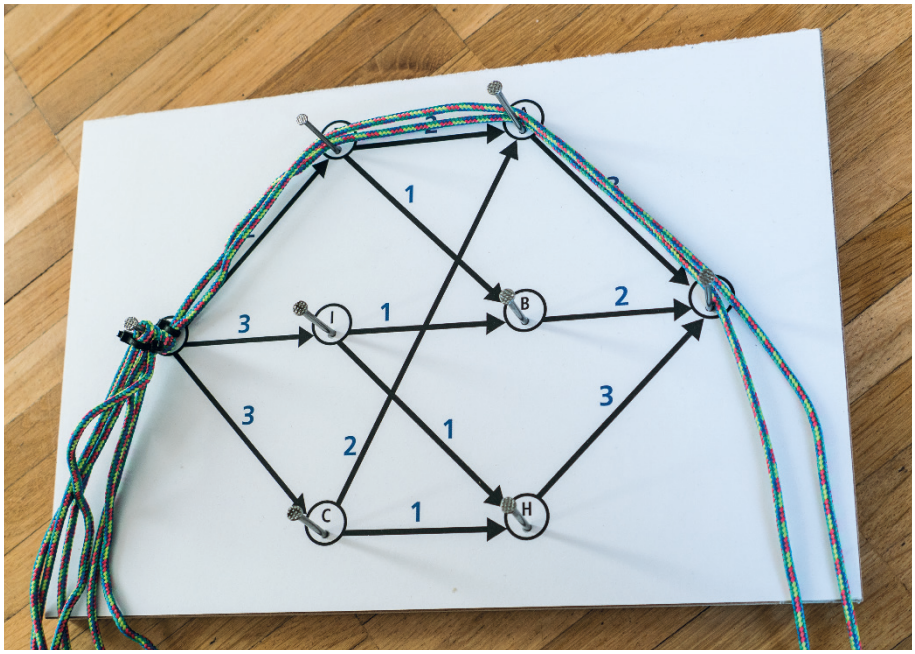


Abbildung 10.6
Zwei Kubikmeter Wasser sind bereits durch Schnüre festgelegt.

Einen zweiten Versuch sollten wir aber trotzdem machen, bevor wir aufgeben: Nehmen Sie alle Schüre zurück und fangen Sie diesmal von unten an. Kommen Sie nun auf ein besseres Ergebnis?

Nun konnten Sie sicherlich mehr Flüsse legen:

Ⓦ – Ⓒ – Ⓗ – Ⓔ

Ⓦ – Ⓒ – Ⓐ – Ⓔ

Ⓦ – Ⓒ – Ⓐ – Ⓔ

Ⓦ – Ⓘ – Ⓗ – Ⓔ

Ⓦ – Ⓘ – Ⓑ – Ⓔ

Ⓦ – Ⓓ – Ⓑ – Ⓔ

Ⓦ – Ⓓ – Ⓐ – Ⓔ

Nun sind es doch 7 m^3/s geworden. Offenbar spielt es also eine Rolle, in welcher Reihenfolge die Flüsse etabliert werden. Manchmal blockiert eine bereits getroffene Entscheidung das Erreichen des Optimums.

Für Informatiker ist diese Situation unbefriedigend: Wenn man Entscheidungen eventuell zurücknehmen muss, um zu besseren Ergebnissen zu kommen, bedeutet das immer sehr hohe Rechenzeiten. Man probiert quasi alle Möglichkeiten aus. Bevor man ein solches Verfahren umsetzt, sollte man also zunächst weiter darüber nachdenken, ob es noch bessere Strategien gibt.

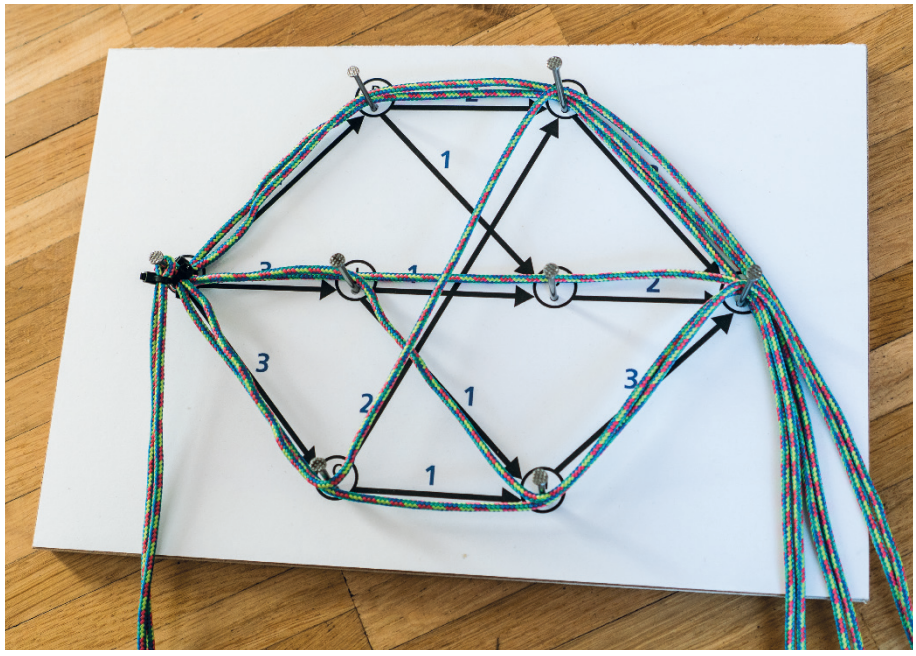


Abbildung 10.7
Wasser marsch – leider nicht genügend

Wasser, das den Berg hoch fließt

Um auf die richtige Idee zu kommen, stellen Sie zunächst die „ungünstige" Situation nach Abbildung 10.7 wieder her. Basteln Sie nun mit Hilfe einer Haftnotiz eine zusätzliche Pumpstrecke, mit der maximal 2 m^3/s Wasser von Ⓐ nach Ⓓ gepumpt werden können.

Schaffen Sie es nun, einen zusätzlichen Fluss von der Quelle zur Senke zu etablieren?

Mit Hilfe der Pumpe klappt das tatsächlich: Eine zusätzliche Schnur lässt sich über die Strecke Ⓦ – Ⓒ – Ⓐ – Ⓓ – Ⓑ – Ⓔ spannen. Die Aufgabe ist erfüllt!

Vielleicht sind Sie aber damit doch noch nicht ganz zufrieden: Von der Möglichkeit, eine Pumpe zu installieren, hatte die Stadtverwaltung in Schilda keinen Ton gesagt.

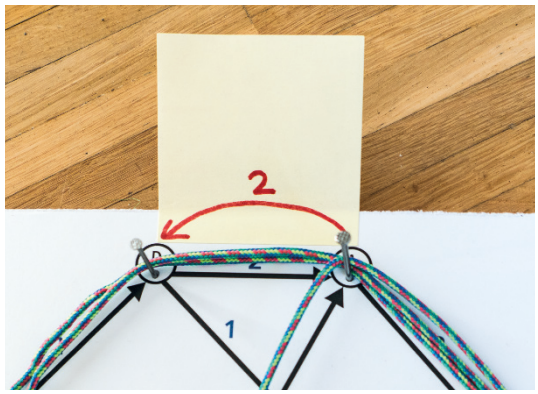


Abbildung 10.8
Virtuelle Pumpstrecke mit Hilfe einer Haftnotiz und eines roten Pfeils

Außerdem waren die Kosten dafür ganz sicher auch nicht einkalkuliert. Ich behaupte, dass unsere auf diese Weise gefundene Lösung doch sehr brauchbar ist.

Abbildung 10.9 zeigt der Übersicht halber nochmals sowohl die zuletzt gefundene Lösung mit Pumpe als auch die „gute“ Lösung durch Probieren schematisch. Vergleichen Sie beide Lösungen genau. Fällt Ihnen etwas in Bezug auf die Notwendigkeit einer Pumpe auf?

Zunächst einmal sehen die Lösungen fast identisch aus! Einen Unterschied gibt es lediglich zwischen den Knoten Ⓓ und Ⓐ. Wenn Sie die Lösung mit der roten Kante betrachten, sehen Sie, dass dort 2 m^3/s Wasser von Ⓓ nach Ⓐ fließen und 1 m^3/s über die Pumpstrecke wieder zurückgepumpt wird. Nicht besonders sinnvoll: Ein Kubikmeter wird pro Sekunde lediglich im Kreis herumgeschickt.

Stattdessen könnte man auch ein Ventil auf der schwarzen Kante zwischen Ⓓ und Ⓐ etwas weiter schließen und dafür sorgen, dass nur 1 m^3/s fließt – dann braucht man das „überschüssige“ Wasser auch nicht zurückzupumpen. Effektiv entspricht das dann genau auch der durch Probieren entstandenen unteren Lösung.

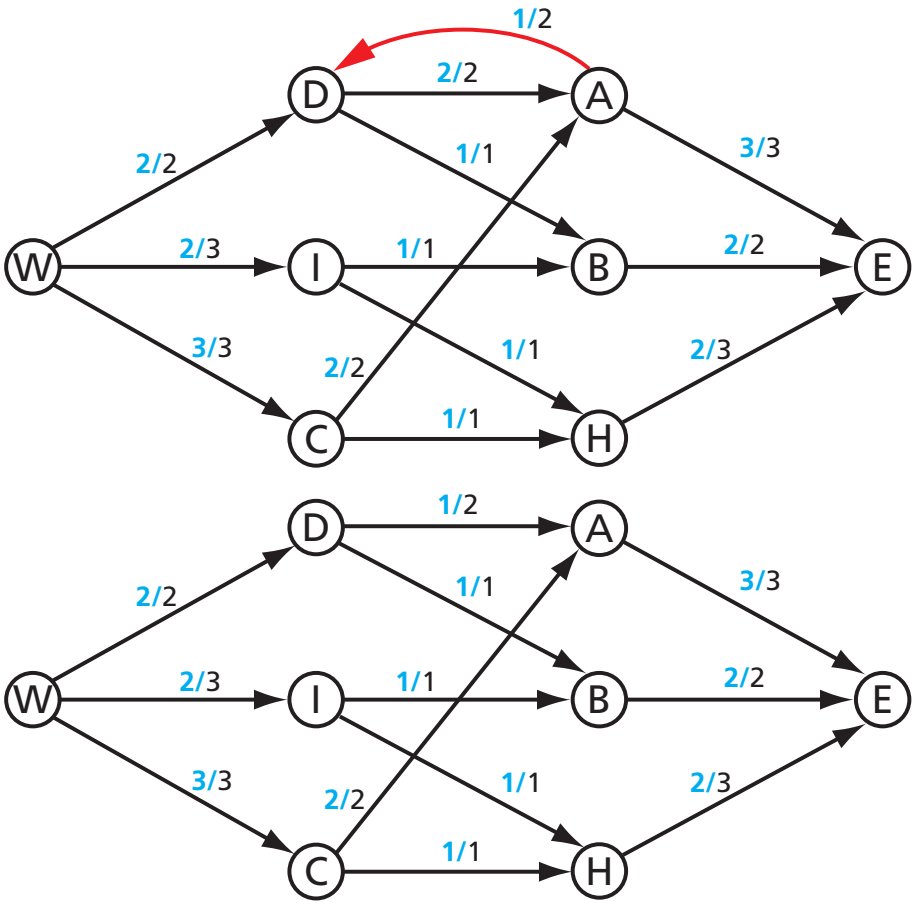


Abbildung 10.9
Die auf unterschiedlichen Wegen entstandenen „optimalen“ Lösungen mit jeweils maximalem Fluss von Ⓦ nach Ⓔ

Abbildung 10.10
Eine Pumpstrecke mit Kapazität 2 darf man einzeichnen, ohne Gefahr zu laufen, dann auch wirklich eine Pumpe einbauen zu müssen.

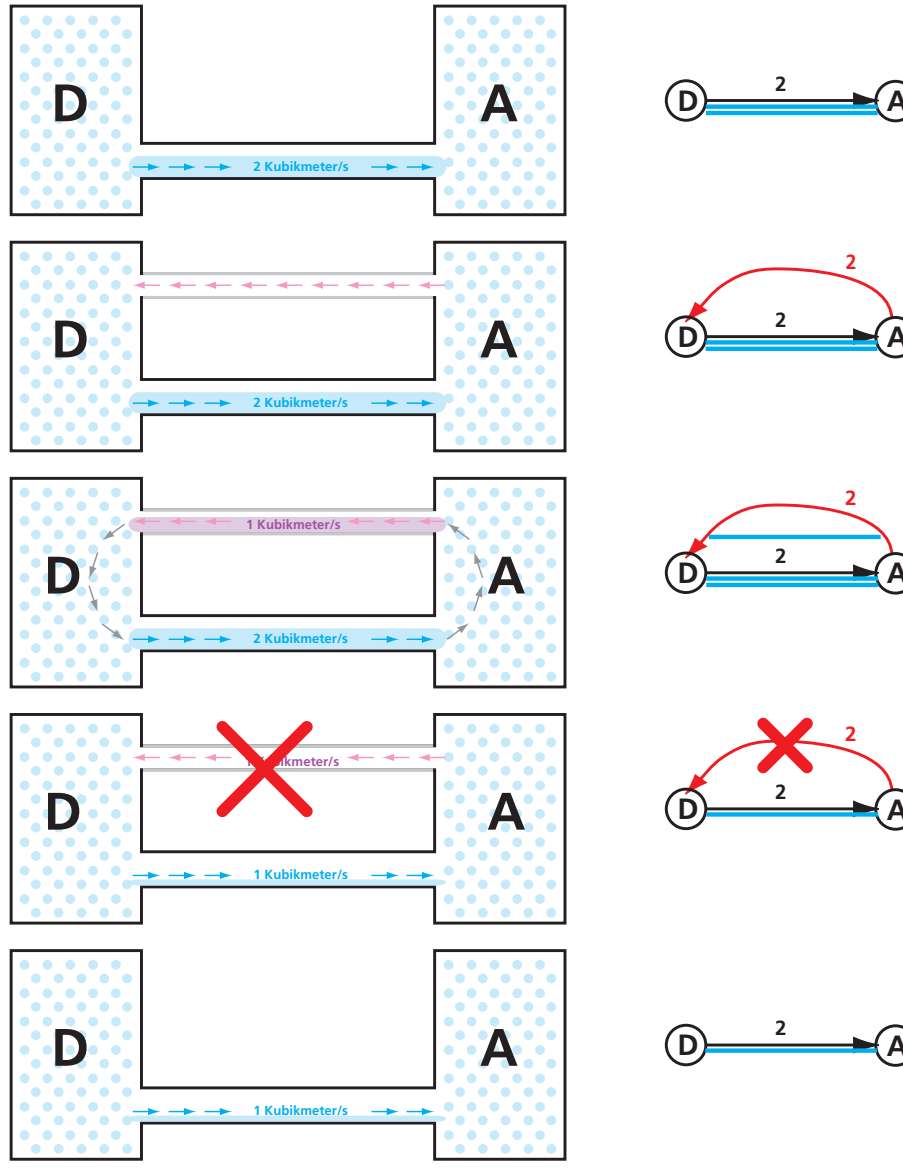


Abbildung 10.10 zeigt zur Verdeutlichung noch einmal schematisch, was zwischen Knoten Ⓓ und Ⓐ passiert und warum man eigentlich gar keine rote Kante benötigt, auch wenn man diese zunächst einzeichnet.

Wir haben es also letztlich geschafft, den Durchfluss weiter zu vergrößern, ohne zusätzliche Rohre zu verlegen. Hierfür haben wir implizit eine Erkenntnis genutzt, die uns erlaubt, eine imaginäre Rückleitung zwischen zwei Punkten zu verlegen, falls bereits Wasser zwischen ihnen fließt. Die Kapazität darf so groß sein wie der Fluss in die Gegenrichtung, denn so viel „Rückfluss“ können wir durch schlichte Reduktion des „Hinflusses“ realisieren.

Auf diese Weise korrigieren wir den anfangs gemachten „Fehler“. Es wird trotz der einfachen, immer voranschreitenden Vorgehensweise sichergestellt, dass jeder zusätzliche Fluss aufgespürt wird, selbst wenn ein zu benutzendes Rohr scheinbar durch eine vorherige Entscheidung blockiert scheint.

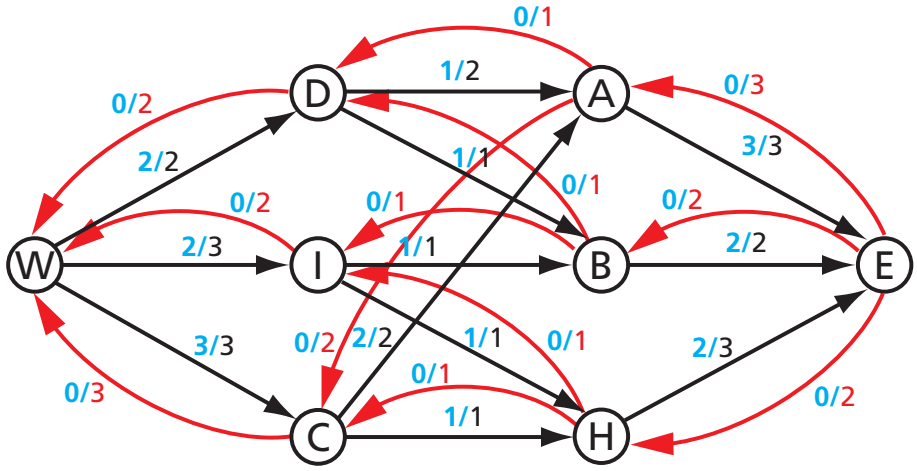


Abbildung 10.11
Wenn alle möglichen Rückkanten eingezeichnet sind, wird die Aufgabe, einen weiteren möglichen Fluss zu finden, schon unübersichtlicher.

In Abbildung 10.11 sehen Sie zur Verdeutlichung nochmal den kompletten Graphen mit allen möglichen Rückkanten. Finden Sie hier noch einen Weg von der Quelle zur Senke?

Das ist eine Herausforderung: nicht nur weil die vielen Pfeile das Diagramm unübersichtlich machen, sondern auch weil nun Kreise entstehen. So könnten wir ohne Probleme einen Kubikmeter Wasser zusätzlich von Ⓦ nach Ⓘ leiten und direkt über die rote Kante wieder zurück. Das macht aber freilich keinen Sinn, denn wir möchten ja einen zusätzlichen Fluss bis zur Senke Ⓔ etablieren.

Wenn das Verfahren auf einem Computer ablaufen soll, müssen wir uns also später noch überlegen, wie das digitale Pendant zum effektiven Verlegen der Schnüre am besten funktioniert.

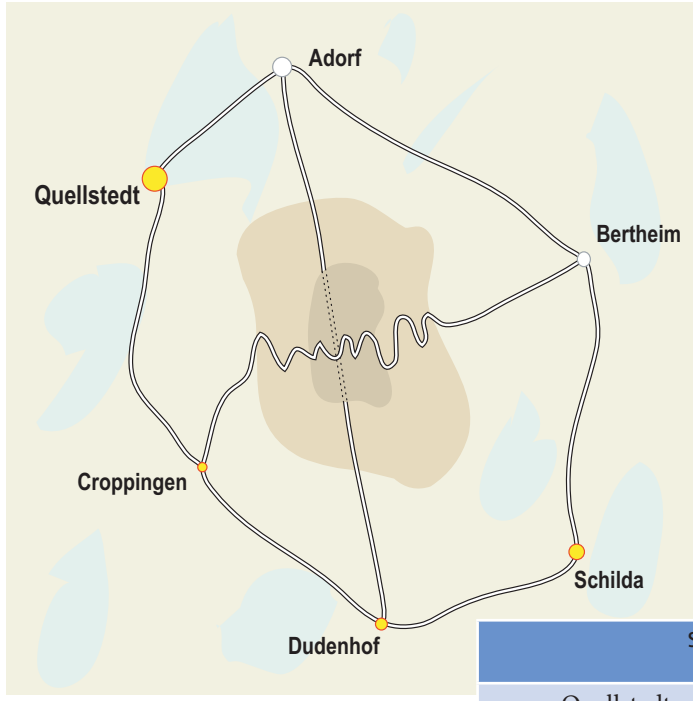
Verkehr unter Kontrolle

Bevor Sie den Algorithmus allgemein formulieren, wenden wir die gewonnenen Erkenntnisse aber nochmals auf ein neues Beispiel aus der Praxis an: In heutiger Zeit wird es immer wichtiger, auch Verkehr zu lenken und zu leiten. Abbildung 10.12 zeigt die Verbindungen zwischen Quellstedt und Schilda. Ein Berg zwischen den Orten verhindert eine direkte Verbindungsstraße, der Berufsverkehr teilt sich auf zwischen den Umfahrungen auf beiden Seiten des Berges. Die Strecken über den Pass bzw. durch den Tunnel werden aufgrund der Länge eigentlich nur von Touristen genutzt.

Verkehrszählungen haben ergeben, dass die einzelnen Streckenabschnitte Kapazitäten aufweisen, wie sie in der Tabelle angegeben sind. Ein Verkehrsleitsystem sorgt dafür, dass diese Kapazitäten der Abschnitte nicht überschritten werden, um den Verkehrsinfarkt zu vermeiden.

Zur abendlichen Rushhour fahren sehr viele Autos von Quellstedt nach Schilda. Die oben schon erwähnte Verkehrszählung hat ergeben, dass dann die Strecke Quellstedt – Croppingen – Dudenhof – Schilda mit 50 Autos pro Minute voll ausgelastet ist. Über Quellstedt – Adorf – Bertheim – Schilda fahren immerhin 10 Autos pro Minute. Die Bergstraßen werden normalerweise zu dieser Zeit gar nicht nennenswert genutzt – hier nehmen wir daher 0 Autos pro Minute an.

Abbildung 10.12
Landkarte mit den Wegen zwischen Quellstedt und Schilda



Strecke	Kapazität Autos/min
Quellstedt – Adorf	45
Quellstedt – Croppingen	50
Adorf – Bertheim	20
Adorf – Dudenhof	35
Bertheim – Croppingen	35
Bertheim – Schilda	45
Croppingen – Dudenhof	50
Dudenhof – Schilda	50

In einer Woche sollen die Meisterschaften im Südfruchtbalancieren in Schilda stattfinden. Der Magistrat rechnet während der Rushhour mit einem zusätzlichen Verkehr von 30 Autos pro Minute, die von Quellstedt aus unterwegs sind. Ihre Aufgaben:

- Ermitteln Sie, ob dieses Verkehrsaufkommen auf den Strecken überhaupt möglich ist.
- Wenn ja, welche Mengen an Autos pro Minute muss das Verkehrsleitsystem auf jeder Strecke zulassen?

Gehen Sie hierfür genauso vor, wie wir das im obigen Beispiel zusammen getan haben: Zunächst zeichnen Sie eine abstraktere Darstellung der Landkarte und beschriften jede Strecke mit den korrekten Flüssen sowie Kapazitäten. Danach ermitteln Sie zusätzliche Flüsse von Quellstedt nach Schilda. Vergessen Sie die Rückleitungen nicht …

Der entstehende abstrakte Plan ist noch einfacher als der im letzten Beispiel, denn wir kommen mit sechs Verbindungspunkten aus, die wir nach den Städten Ⓠ, Ⓢ,

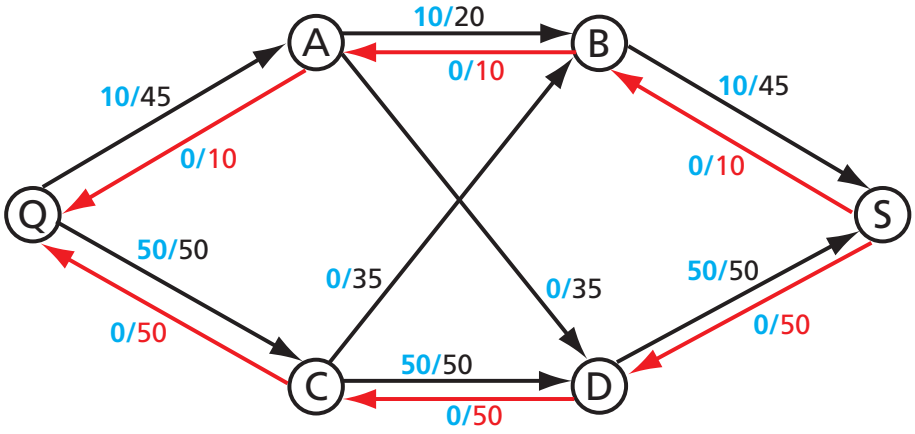


Abbildung 10.13
Verkehrsplanungsdiagramm für die Strecke Quellstedt – Schilda einschließlich der Pfeile für virtuelle Rückflüsse

Ⓐ, Ⓑ, Ⓒ und Ⓓ nennen. Dazwischen gibt es acht Strecken. Eine besondere Abstraktion können wir hier noch vornehmen: Die Straßen sind laut Aufgabenstellung prinzipiell keine Einbahnstraßen, daher sind sie eigentlich in beide Richtungen befahrbar. Allerdings geht es uns um den Verkehrsfluss von Q nach S. Daher betrachten wir die Strecken nur in „positiver" Richtung, das heißt, wir gehen davon aus, dass ein Auto immer in Richtung Schilda unterwegs ist. Abbildung 10.13 zeigt die Lösung mit entsprechenden Pfeilen! Außerdem sind in Rot auch bereits mögliche Rückflüsse eingezeichnet – wie vorher ausgeknobelt immer mit der Kapazität, die dem Fluss in die eigentliche Richtung entspricht. Bitte verwechseln Sie dies nicht mit Fahrten in die Gegenrichtung (die ja bei Straßen auch möglich ist): Während Wasser „geduldig" ist und auch ohne Murren folgen würde, wenn wir es im Kreis pumpen, könnten wir kaum Autofahrer überzeugen, unsinnigerweise immer im Kreis zu fahren, weil das unserer Verkehrsplanung entgegenkommt.

Der erste zusätzliche Fluss ist aus dem Diagramm sehr leicht auffindbar: Abbildung 10.14 zeigt in Blau die Strecke über Ⓠ – Ⓐ – Ⓑ – Ⓢ. Erkennen Sie hier gleich eine weitere Vereinfachung zum Spannen von Schnüren von vorhin? Wenn wir alle Restkapazitäten auf dem gesamten Weg anschauen, können wir sofort sehen, dass zusätzlich nicht nur ein Auto, sondern 10 Autos pro Minute fahren können – das ist das Minimum der Restkapazitäten. Statt nacheinander 10 Schnüre zu spannen, erhöhen wir daher auf der gesamten Strecke den Fluss gleich um 10. Das Ergebnis sehen Sie in Abbildung 10.15.

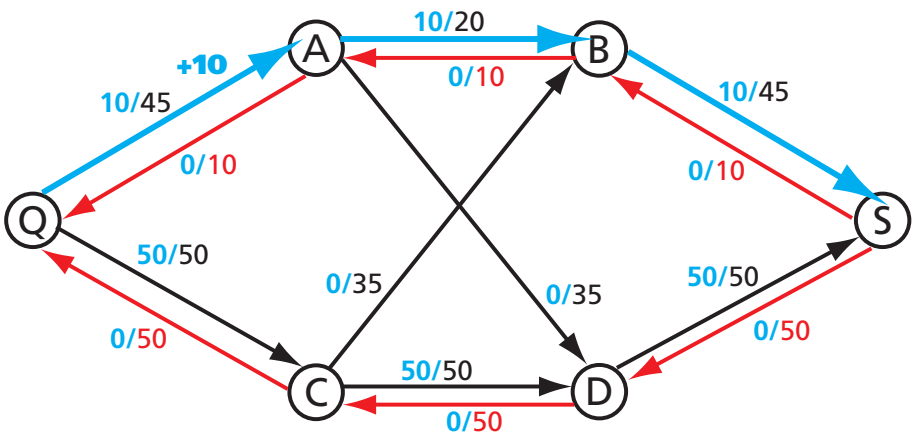


Abbildung 10.14
Zusätzlich sind über die blau markierte Strecke 10 Autos pro Minute möglich.

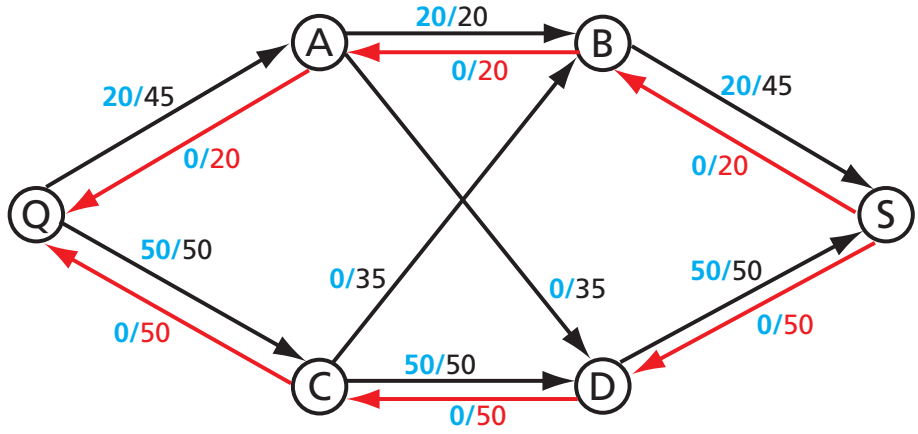


Abbildung 10.15
Die Flüsse wurden entsprechend an allen Pfeilen erhöht.

Wie verfahren wir nun? Um den Verkehrsfluss weiter zu erhöhen, müssen Sie die Rückkante zwischen Ⓒ und Ⓓ ausnutzen. Abbildung 10.16 zeigt das. Die kleinste Restkapazität auf dem Weg beträgt 25. In Abbildung 10.17 ist dieser zusätzliche Fluss eingezeichnet.

Zwischen Ⓒ und Ⓓ wird der Verkehr nun allerdings noch im Kreis geleitet, was – wie schon besprochen – kaum ein Autofahrer mit sich machen ließe. Daher müssen wir noch den virtuellen Verkehrsfluss auf der Rückkante auflösen. Das Endergebnis ist in Abbildung 10.18 dargestellt.

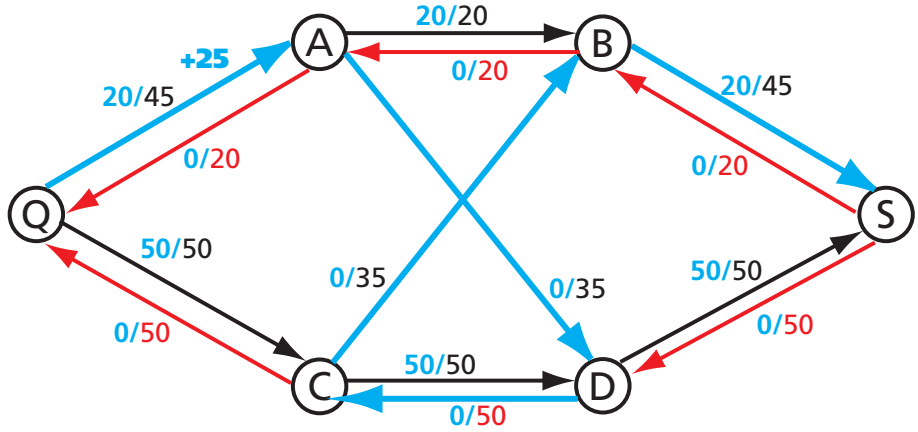


Abbildung 10.16
Weitere Steigerung ist möglich, wenn wir den Rückfluss berücksichtigen.

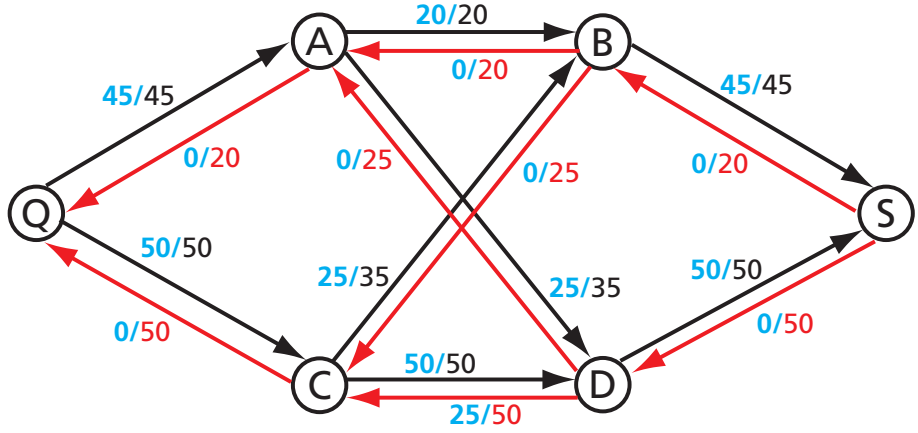


Abbildung 10.17
Die Flüsse wurden entsprechend an allen Pfeilen erhöht.

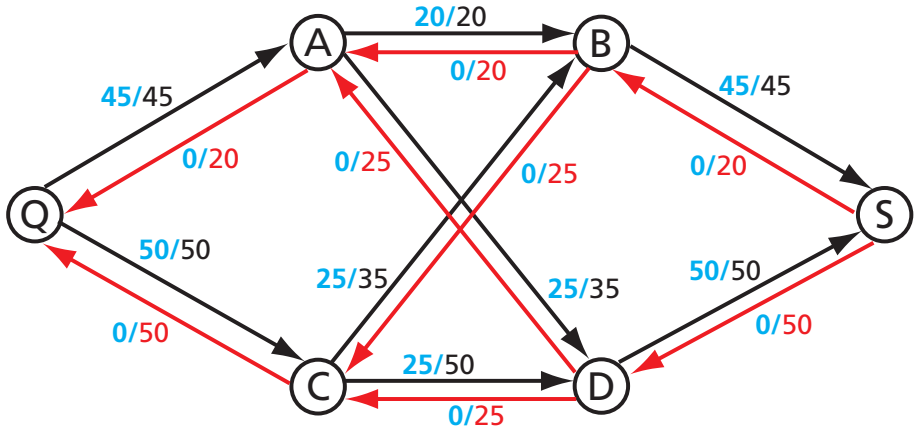


Abbildung 10.18
Verkehrsplanungsdiagramm zur Erhöhung des Verkehrsaufkommens zwischen Quellstedt und Schilda

Damit ist die Aufgabe gelöst: Zwischen Quellstedt und Schilda können 95 Autos pro Minute ungehindert verkehren. Beim Fußballspiel werden 90 Autos pro Minute erwartet, es gibt also sogar einen „Puffer" von 5 Autos pro Minute.

Um diesen Fluss zu erreichen, müssen Sie allerdings das Verkehrsleitsystem so einstellen, dass zwischen Croppingen und Dudenhof maximal 25 Autos pro Minute fahren können! Den höheren Gesamtfluss erreichen Sie also durch Reduktion des Flusses auf einer Teilstrecke!

Mehr Verkehrsfluss ...
... entstehe bei niedrigerem Tempolimit – behaupten einige Veröffentlichungen über die modernen Schilderbrücken. Das lässt sich aber schon aufgrund der Faustformel für den Sicherheitsabstand widerlegen: Dieser soll der in 2 Sekunden gefahrenen Strecke entsprechen (etwa halber Tachowert). Wenn sich alle sogar exakt daran halten, bedeutet dies, dass an einem Punkt alle 2 Sekunden ein Auto vorbeikommt, der mögliche Fluss also unabhängig von der Geschwindigkeit konstant bleibt.

Damit haben Sie auch das wesentliche Wirkprinzip unserer Verkehrsleitsysteme auf Autobahnen entdeckt: Um den Gesamtfluss in einem System zu erhöhen muss eventuell der Fluss auf einer Teilstrecke verringert werden. Für den Straßenverkehr ist hierfür eine wirksame Methode, die Strecke unattraktiv zu machen. Aus diesem Grund zeigen die Schilder dann auch in der Rushhour schon mal 80 km/h an, selbst wenn in der momentanen Situation die Autobahn komplett frei ist. Hier sollen die Fahrer „lernen", die Strecke zu dieser Zeit zu meiden.

Sie haben vielleicht in Ihrer unmittelbaren Umgebung bereits feststellen können, ob die Städte- und Verkehrsplaner einen guten Informatiker an Ihrer Seite hatten. Eine neue Umgehungsstraße führt oft nur dann zu einer verbesserten Verkehrssituation, wenn der Verkehrsfluss auf einer anderen Strecke verkleinert wird – etwa durch Verkehrsberuhigung. Es ist jedoch entscheidend, an welcher Stelle diese Beruhigung stattfindet – man kann sie durch das hier vorgestellte Verfahren ermitteln. Häufig spielen in der Kommunalpolitik aber andere Kriterien wie „Stadtrat X wohnt an der Straße Y" eine größere Rolle für diese Entscheidung, was weniger optimale Ergebnisse für alle zur Folge hat …

Abbildung 10.19
Schilderbrücke mit variablem Tempolimit auf der Autobahn

Was steckt dahinter?

In diesem Kapitel haben Sie auf der Suche nach dem maximalen Fluss grundsätzlich unterschiedliche Entwurfsstrategien für Algorithmen angewendet. Zunächst haben wir nach dem bereits bekannten Prinzip „divide et impera“ die große Aufgabenstellung in kleine, handhabbare Teile zerlegt, bis wir pro Durchlauf den Gesamtfluss nur noch um eine Einheit erhöhen mussten.

Nach der mehrfachen Erhöhung um eine Einheit hatten wir dann eine Gesamtlösung, die wir zunächst nicht mehr weiter verbessern konnten, von der wir allerdings auch nicht mit Sicherheit sagen konnten, ob es die optimale Lösung ist.

Gewissheit darüber, dass es nicht die optimale Lösung war, brachte dann ein Neuanfang über andere Wege von der Quelle zur Senke. „Gezieltes Probieren“ ist in der Informatik absolut nicht verwerflich, sondern eine valide Entwurfsstrategie namens Backtracking.

Backtracking

Wichtige Entwurfsstrategie der Informatik: „Versuche so lange Teillösungen zu errechnen und zu einer Gesamtlösung zu vereinigen, bis diese entweder gefunden ist oder klar wird, dass diese auf dem aktuellen Weg nicht gefunden werden kann. Gehe dann gezielt einige Schritte zurück und versuche andere Teillösungen. Mache das, bis Du mit der gefundenen Lösung zufrieden bist.“

Backtracking kann sehr effizient gestaltet werden, wenn man Wege, die nicht zum Ziel führen, möglichst schnell erkennt und dann frühzeitig abbricht. Diese Vorgehensweise ist dann auch unter „Branch and Bound“ bekannt. Im Vergleich mit anderen Strategien beanspruchen solche mit Backtracking jedoch in der Regel deutlich mehr Rechenzeit.

Da uns als Menschen intelligentes Ausprobieren von Kleinkindertagen an geläufig ist und wir damit erfolgreich sind, versuchen wir auch in der Informatik zunächst viele Aufgaben mit dieser Strategie zu lösen. Allerdings wissen wir ebenfalls aus Erfahrung, dass größere Aufgaben mit Ausprobieren oft gar nicht mehr zu bewältigen sind. Im Fall der Fluss-Maximierung haben wir den Algorithmus daher ergänzt und den Umweg über die Rückkanten ermöglicht.

Mit dieser Ergänzung bleibt ein einmal erreichter Stand immer bestehen. Daher nennt man die entsprechende Entwurfsstrategie für Algorithmen „Greedy-Prinzip“. Greedy ist das englische Wort für gierig. Die Laufzeiten solcher gieriger Aufgabenlösungen sind deutlich besser. Auch wenn die Bezeichnung zunächst etwas negativ anmutet, ist also eine Greedy-Lösung anzustreben – falls man eine solche findet.

Greedy-Prinzip

Folgt ein Verfahren der Informatik dem Greedy-Prinzip, geht es über einen einmal erreichten Stand nicht mehr zurück. Das Greedy-Prinzip ist damit quasi das Gegenstück zum Backtracking. Algorithmen, die dem Greedy-Prinzip folgen, benötigen in der Regel eine kurze Laufzeit. Oft ist es daher günstiger, nur eine Näherungslösung mit Greedy zu berechnen als das Optimum per Backtracking.

Der Algorithmus, der dem hier vorgestellten Verfahren mit Rückkanten folgt, ist nach den Mathematikern Lester Randolph Ford und Delbert Ray Fulkerson benannt, die ihn 1956 entwickelten.

Lester Randolph Ford Jr. (* 1927) und Delbert Ray Fulkerson (1924–1976) beschäftigten sich als Angestellte der Rand-Corporation besonders mit Optimierungsproblemen und „Operations Research", also mit für die industrielle Nutzung vorgesehenen Problemlösungen.
1956 wurde ihr Artikel über das „Max-Flow-Min-Cut"-Problem veröffentlicht, dessen wesentliche Teile hier im Kapitel dargestellt sind.

Seit 1956 haben sich sehr viele Mathematiker und Informatiker mit dem gleichen Problem beschäftigt, da es für viele Bereiche praktische Anwendung findet:

- Der Einsatz bei der Planung von Rohrleitungen für den Transport von Flüssigkeiten oder Gas liegt auf der Hand. Das können Wasser- und Abwassersysteme, Be- und Entlüftungsschächte oder z. B. Ölpipelines sein.
- Die Anwendung bei der Verkehrsplanung haben wir bereits angesprochen.
- Insbesondere Computernetzwerke benötigen auch eine genaue Planung im Hinblick auf die zur Verfügung stehenden Kapazitäten zwischen zwei beliebigen Rechnern. Für zeitkritische Systeme ist das von größter Wichtigkeit, zum Beispiel Telefonie über Internet: Hier wäre es nicht akzeptabel, wenn zu irgendeinem Zeitpunkt so wenige Daten übertragen werden könnten, dass die Sprache abgehackt klingt.

Daher gibt es einige neuere Lösungen, die ein schnelleres Auffinden des maximalen Flusses ermöglichen oder noch weitergehende Ergebnisse liefern (zum Beispiel, wo der günstigste Punkt wäre, um eine Leitung zu erweitern, falls der ermittelte maximale Fluss nicht ausreichen sollte). Viele Lösungen beruhen jedoch im Grunde auf den Ergebnissen von Ford und Fulkerson und dem damit verbundenen, recht einfachen Algorithmus.

Ich möchte auf eine weitere Besonderheit eingehen, die in vielen Problemlösungen der Informatik eine große Rolle spielt. Hierfür schalte ich zunächst voll auf den Wortschatz aus Fachbegriffen um, den ich im ersten Kapitel eingeführt habe.

Das Problem lässt sich demnach durch einen Graphen beschreiben, der aus Knoten und Kanten besteht. Sowohl Knoten als auch Kanten können in verschiedener Art markiert werden. Zur Verdeutlichung nochmals die Analogien der Begriffe in unserem ersten Beispielszenario:

Knoten sind die Versorgungsschächte.

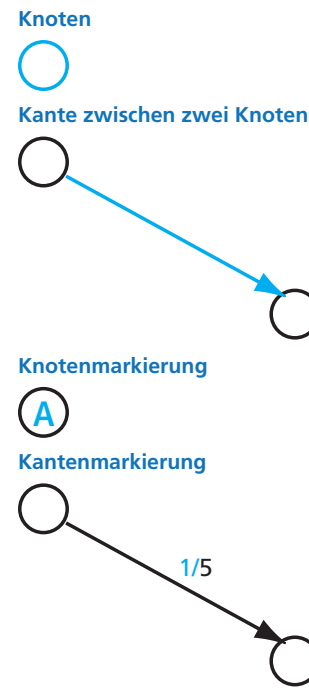
Kanten sind die Wasserleitungen dazwischen. In diesem Fall handelt es sich, genauer ausgedrückt, um gerichtete Kanten, da sie Pfeile besitzen und so das Wasser nur in einer Richtung laufen kann.

Knotenmarkierungen sind die schwarzen oder farbigen Beschriftungen sowie das Ausmalen der Knoten.

Kantenmarkierungen sind die Beschriftungen der Wasserleitungen mit der Kapazität und dem Fluss. Außerdem stellt das Einfärben der Kanten in Rot oder Blau auch eine Art der Kantenmarkierung dar.

Wesentlicher Bestandteil des Algorithmus von Ford und Fulkerson ist das Auffinden eines zusätzlichen Flusses zwischen dem als Quelle und dem als Senke bezeichneten Knoten. Im Fachjargon nennt man dies einen „zunehmenden Weg".

Wir haben das einfach mit einer Schnur gemacht und unsere Intuition eingesetzt, um zu ermitteln, wo wir diese noch verlegen können. Einem Computer müssen wir das allerdings erst beibringen. Wie im ersten Kapitel „wandern" wir dabei durch den Graphen und markieren auf dem Weg Knoten und Kanten, bis wir entweder das Ziel erreichen oder eben nicht. Eine solche Vorgehensweise nennt man „Traversierung".

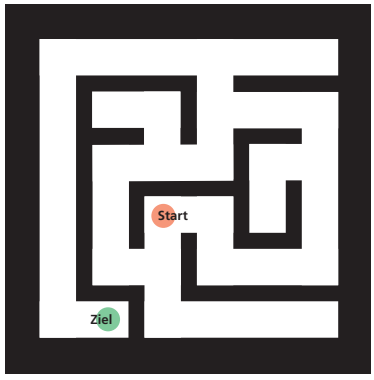


Die Tiefensuche wird nach der englischen Bezeichnung „Depth First Search“ auch als DFS abgekürzt. Die Breitensuche wird nach der englischen Bezeichnung „Breadth First Search“ auch als BFS abgekürzt.

Prinzipiell gibt es zwei unterschiedliche Strategien für eine solche Traversierung: die Tiefensuche und die Breitensuche. Bei der Tiefensuche besucht man vom Startknoten aus den nächsten Knoten, von dort direkt wieder den nächsten usw. Die Tiefensuche führt also meistens nach kurzer Zeit weit weg vom Startknoten. Erst wenn wir in einer „Sackgasse“ angelangt sind, machen wir wieder weiter bei einem der vorhergehenden Knoten, an dem noch weitere Kanten zu erkunden waren. Erkennen Sie die Parallele zum Backtracking?

Bei der Breitensuche werden erst alle Nachfolger des Startknotens besucht, dann deren Nachfolger komplett usw. Die Breitensuche bleibt also zunächst in direkter Nachbarschaft zum Startknoten und führt erst langsam in die weiter entfernten Gebiete des Graphen.

Abbildung 10.20
Labyrinth mit Startpunkt und Zielpunkt



Am besten kann man sich den Unterschied anhand eines Labyrinthes vorstellen: Sie befinden sich irgendwo im Labyrinth und wollen den Ausgang finden. Ein Beispiel sehen Sie in Abbildung 10.20. Wie gehen Sie vor? Eine Möglichkeit ist die Tiefensuche:

„Gehe vom Ausgangspunkt bis zur nächsten Weggabelung. Wähle dort einen beliebigen, bisher nicht besuchten Weg und laufe diesen weiter. Mache dies so lange, bis eine Sackgasse erreicht ist. Dann kehre zur letzten Kreuzung zurück. Wähle dort einen bisher unbesuchten Weg usw.“

Abbildung 10.21 zeigt die Vorgehensweise im Beispiel, wenn wir der Regel „Immer erst den rechten Weg probieren, dann den linken“ folgen. Wir müssen zunächst an vier Kreuzungen die Entscheidung „rechts“ treffen – das wird vom blauen Kreis mit Fragezeichen symbolisiert. Danach kommen wir in eine Sackgasse und verfolgen unseren Weg zurück bis zur letzten Kreuzung. Von dort aus gehen wir den linken, bisher nicht beachteten Weg und kommen am Ausgangspunkt an, was auch einen Fehlschlag bedeutet. Wiederum verfolgen wir unseren Weg zurück – diesmal sogar zwei Kreuzungen weit – und verfolgen wieder den unbekannten Weg. Nach einer weiteren Sackgasse hat uns dann glücklich dieses Verfahren ans Ziel gebracht!

Abbildung 10.22 zeigt demgegenüber das Vorgehen bei einer Breitensuche. Hier suchen wir zunächst alle Kreuzungen von unserem Startpunkt aus und nummerieren sie, ohne weiterzugehen (auch dies machen wir erst einmal nach der Regel „erst rechts, dann links“). Danach arbeiten wir die Nummern ab, beginnend bei 1. Wiederum erforschen wir die abgehenden Gänge bis zu einer neuen Kreuzung bzw. bis zu einer Sackgasse. Neue Kreuzungen werden weiter fortlaufend nummeriert, aber immer erst weiter betrachtet, wenn sie an der Reihe sind. Auch mit diesem Algorithmus finden wir mühelos das Ziel.

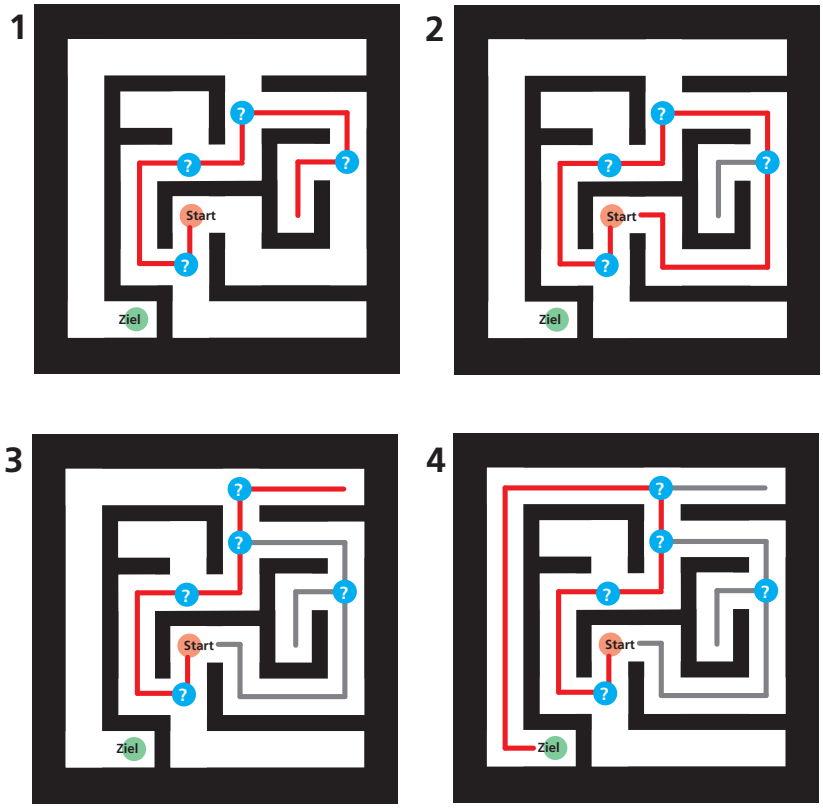


Abbildung 10.21
Tiefensuche mit der Regel „erst rechts, dann links“

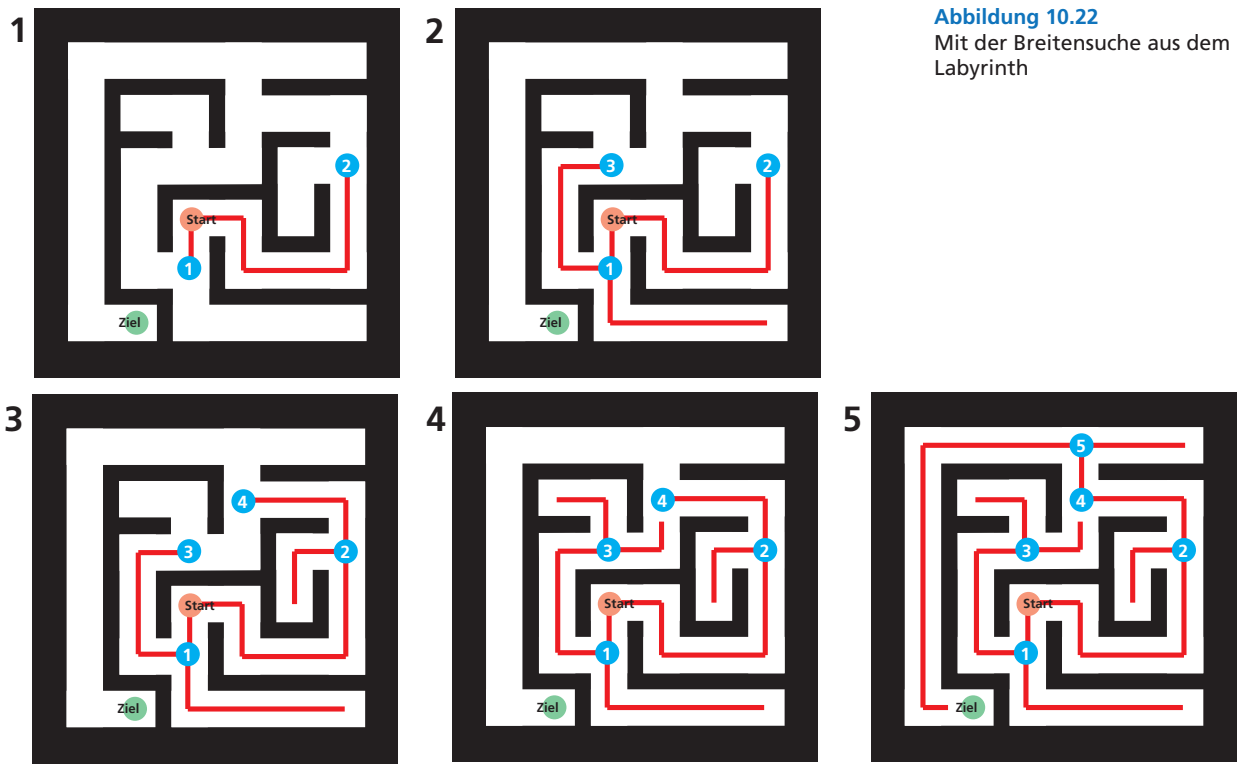


Abbildung 10.22
Mit der Breitensuche aus dem Labyrinth

In genau gleicher Weise geht man vor, um einen Graphen mit Tiefensuche bzw. per Breitensuche zu traversieren. Denken Sie zurück an das Dijkstra-Verfahren aus dem ersten Kapitel! Auch bei der Suche nach einem kürzesten Weg besuchten wir alle Knoten des Graphen nacheinander, bis wir am Ziel angekommen waren. Handelte es sich eher um eine Tiefensuche oder um eine Breitensuche?

Selbstverständlich war das eher eine Breitensuche: Wir betrachteten weiter entfernte Knoten erst, wenn alle näheren bereits besucht waren. Genau genommen handelte es sich um eine abgewandelte Breitensuche, denn die Knotenmarkierungen spielten bei der Besuchsreihenfolge auch noch eine entscheidende Rolle!

Kommen wir aber zum Thema dieses Kapitels zurück! Bei der Suche nach einem zunehmenden Weg (zur Erinnerung: So nennen wir nun den zusätzlichen Fluss) wird der Graph von der Quelle aus traversiert, bis wir bei der Senke angekommen sind. Was meinen Sie: Geht dies mit einer Tiefensuche oder einer Breitensuche?

Beides ist möglich! Eine Tiefensuche würde in einem sehr weit verzweigten Graphen allerdings unter Umständen einen sehr komplizierten und langen Weg zwischen Quelle und Senke suchen, während bei der Breitensuche immer der nächstkürzeste Weg herauskommt.

Abbildung 10.23 zeigt einen weiteren Flussgraphen, mit dem Sie herumspielen können, zum Beispiel um Breiten- und Tiefensuche miteinander zu vergleichen und um selbst auf den Algorithmus zu kommen, der den maximal möglichen Fluss zwischen Quelle und Senke schnell findet.

Die Abbildung ist nochmals etwas größer und ohne eingezeichnete Flüsse als Kopiervorlage auf der nächsten Seite wiederholt. Spielen Sie darauf nach Herzenslust!

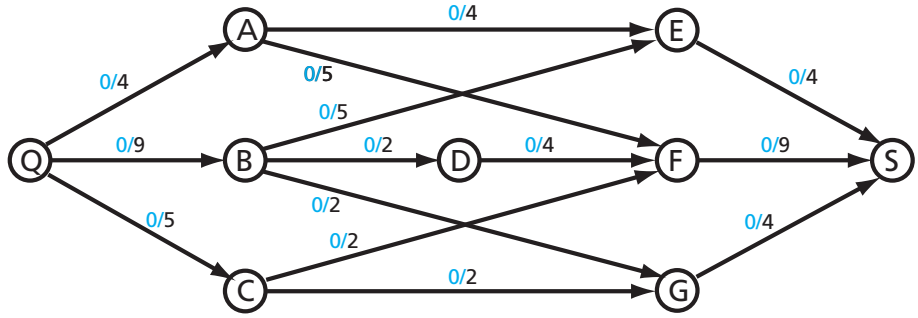


Abbildung 10.23
Weiteres Beispiel zum Ausprobieren

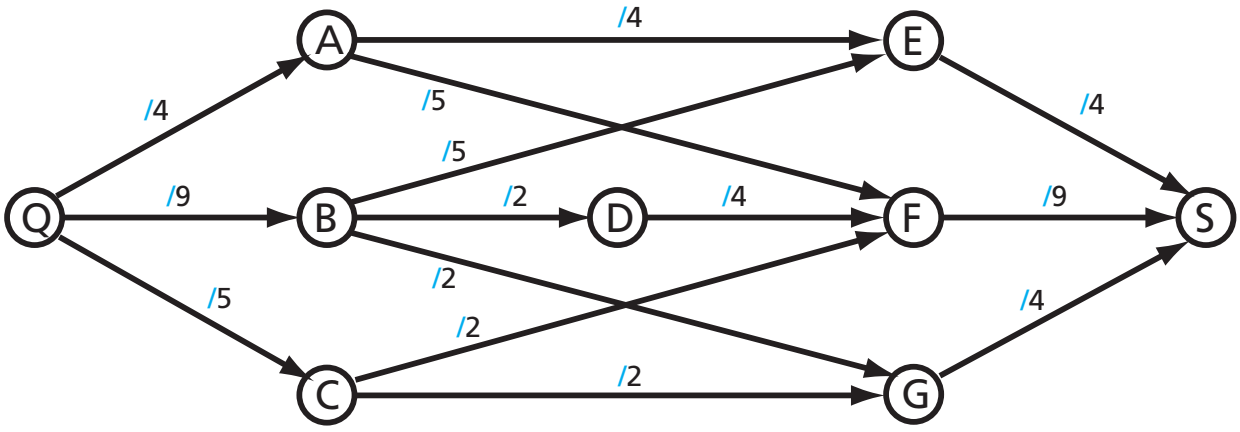


Abbildung 10.24
Beispiel zum Spielen

Der Algorithmus

Es wird Zeit, das anhand von Beispielen erlernte Verfahren allgemein zu formulieren. Sie haben hier ja bereits Übung! Versuchen Sie einen Algorithmus zu formulieren, der dann jedem hilft, einen maximalen Fluss in einem Netz von Wegen zu ermitteln. Denken Sie daran, wirklich jeden einzelnen Schritt (wie im ersten Beispiel exerziert) zu beschreiben!

Da Sie nun schon halber (oder ganzer) Informatiker sind, verwenden Sie bitte die besprochenen Fachbegriffe „Knoten“ für Versorgungsschächte, Orte und ähnliches sowie „Kanten“ für die Leitungen bzw. Wege dazwischen.

Tipp: Es erleichtert die Sache, wenn Sie zu Beginn jeder Suche nach einem zusätzlichen Fluss an die Quelle in Blau ein „unendlich“ schreiben …

Dies ist der komplizierteste Algorithmus dieses Buches. Daher machen Sie sich keine Gedanken, wenn Sie ihn nicht beim ersten Anlauf perfekt hinbekommen! In der „Musterlösung“ sind zur prägnanten Formulierung einige Bezeichnungen definiert, etwa „Knoten X“. Das funktioniert selbstverständlich genauso gut mit entsprechenden Umschreibungen. Auch sind bereits einige Dinge zusammengefasst, die man nach den Experimenten intuitiv erst einmal umständlicher realisiert – etwa die Traversierung mit den beiden unterschiedlichen Knotenmarkierungen (Zahl und Färbung). Es existieren sehr viele unterschiedliche korrekte Lösungen!

Der „äußere“ Algorithmus ist eigentlich recht einfach, es werden – solange es geht – zusätzliche Flüsse ermittelt. Abbildung 10.25 zeigt ihn als Blockdiagramm.

Abbildung 10.25
„Äußere Schleife" des Algorithmus zur Bestimmung des maximalen Flusses

Gegeben sei ein Netz aus Knoten und (gerichteten) Kanten. Jede Kante sei mit einem Fluss und einer Kapazität beschriftet. Ein Knoten ist als Quelle, ein weiterer als Senke definiert.

Bestimme einen zusätzlichen Fluss von der Quelle zur Senke.

Falls es noch einen zusätzlichen Fluss gab, wiederhole diesen Schritt.

Fertig! Der mögliche maximale Fluss kann abgelesen werden, indem man die Flüsse zusammenaddiert, die von der Quelle abgehen oder die bei der Senke zugehen.

Trickreich ist es, den hellblauen Kasten als Algorithmus zu formulieren. In der vorgeschlagenen Lösung nach Abbildung 10.26 wird der Einfachheit halber zwischen Markierung und Färbung der Knoten unterschieden, obwohl natürlich eine Färbung auch eine Markierung darstellt. Erkennen Sie, was die Teile des Verfahrens in den hellblauen Kästen und die in den rötlichen bewirken? Die hellblauen Bereiche beschreiben die Behandlung der „normalen" Kanten, während die hellroten Bereiche die Rückkanten berücksichtigen.

Wenden Sie nun den Algorithmus auf das Beispiel nach Abbildung 10.23 an! Können sie herausfinden, ob dieser nach dem Prinzip der Tiefensuche oder dem der Breitensuche vorgeht?

Die Abbildungen 10.27 bis 10.33 zeigen einen möglichen Lösungsverlauf, wenn man davon ausgeht, dass die Knoten immer in zyklischer alphabetischer Reihenfolge ihrer Namen ausgewählt werden. Sie sehen immer den Zustand nach einer kompletten Durchführung der inneren Schleife in Abbildung 10.26.

Abbildung 10.34 stellt das Endergebnis dar: Insgesamt kann ein Fluss von 16 Einheiten realisiert werden.

Sie haben ganz sicher beim Ausprobieren festgestellt, dass hier der formulierte Algorithmus überhaupt keine genauen Angaben macht. Aus diesem Grund ist die Traversierung auch nicht eindeutig auf Breiten- oder Tiefensuche festgelegt! Im Original-Algorithmus von Ford und Fulkerson wird immer eine Breitensuche durchgeführt. Da prinzipiell jedoch beide Traversierungsstrategien zum Erfolg führen, habe ich hier bewusst eine etwas weniger restriktive, dafür übersichtlichere Variante formuliert.

Wird das Verfahren auf einem Computer ausgeführt, hält dieser sich selbstverständlich implizit an eine strikte „Besuchsreihenfolge".

Außerdem ist genau genommen nicht ausgeführt, wie der Computer den im letzten Schritt aufgeführten „durchgehenden Weg" finden soll. Diese Abwägung zwischen ausführlicher Formulierung jedes einzelnen Schrittes und einer übersichtlichen Darstellung der generellen Vorgehensweise müssen Informatiker sehr häufig vornehmen.

Lösche die Markierung aller Knoten. Markiere die Quelle mit ∞.
Färbe alle Knoten weiß. Färbe alle Kanten schwarz.

Solange an der Senke keine Markierung steht und es noch Knoten mit Markierung gibt, die nicht rot gefärbt sind, mache Folgendes:

Suche einen beliebigen Knoten mit Markierung, der nicht rot gefärbt ist. Diesen nennen wir Knoten X, seine Markierung m.

Färbe Knoten X rot.

Für alle Kanten von Knoten X zu einem Knoten Y, der bisher noch keine Markierung hat, mache Folgendes:

Markiere Y mit zusätzlichem Fluss (Kapazität minus Fluss), der über die Kanten von X nach Y möglich ist, maximal aber mit m.
Nimm keine Markierung vor, falls diese 0 wäre!

Falls eine Markierung vorgenommen wurde:
Färbe die Kante von X nach Y blau.

Für alle Kanten von einem Knoten Y ohne Markierung zum Knoten X, mache Folgendes:

Markiere Y mit dem Fluss über die Kanten von Y nach X, maximal jedoch mit m.
Nimm keine Markierung vor, wenn diese 0 wäre!

Falls eine Markierung vorgenommen wurde:
Färbe die Kante von Y nach X rot.

Falls die Senke keine Markierung hat:
Es ist kein zusätzlicher Fluss möglich, ABBRUCH!

Nenne die Markierung der Senke z.
Verfolge den durchgehenden Weg blau oder rot gefärbter Kanten von der Quelle zur Senke. Addiere bei allen blauen Kanten dieses Weges z zum vorhandenen Fluss. Subtrahiere bei allen roten Kanten dieses Weges z vom vorhandenen Fluss.

Abbildung 10.26
„Innere Schleife" des Algorithmus zur Bestimmung des maximalen Flusses

Abbildung 10.27
Erster zunehmender Fluss von vier Einheiten

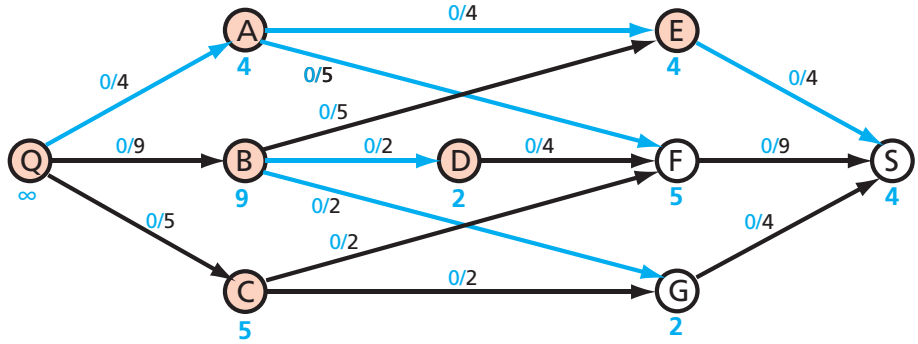


Abbildung 10.28
Zweiter zunehmender Fluss von zwei Einheiten

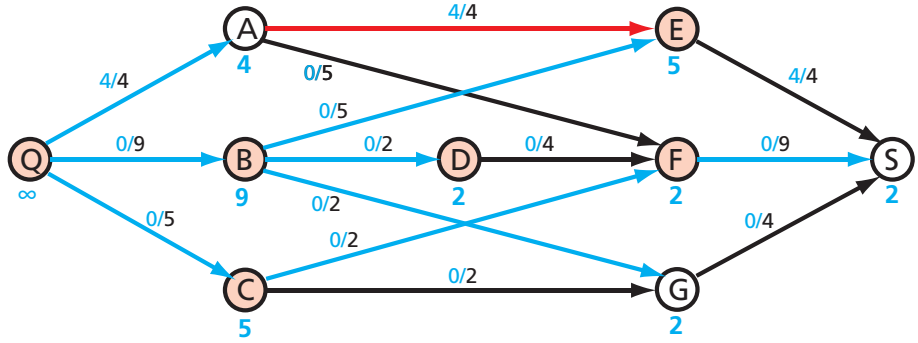


Abbildung 10.29
Dritter zunehmender Fluss von zwei Einheiten

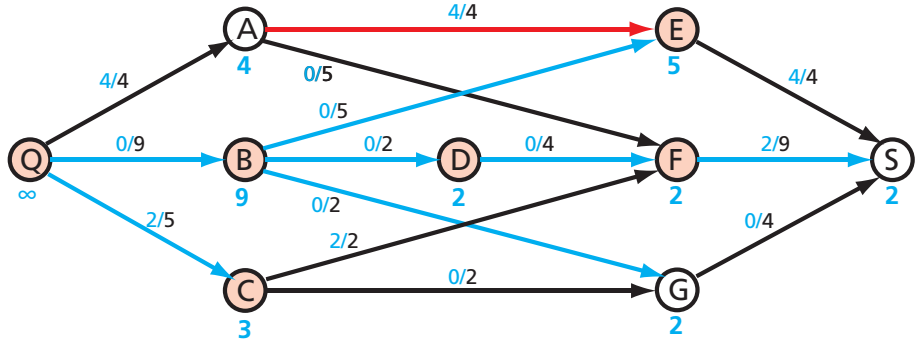
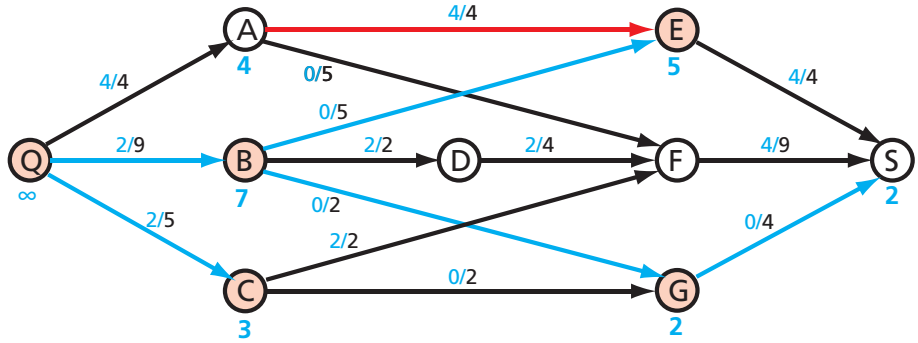


Abbildung 10.30
Vierter zunehmender Fluss von zwei Einheiten



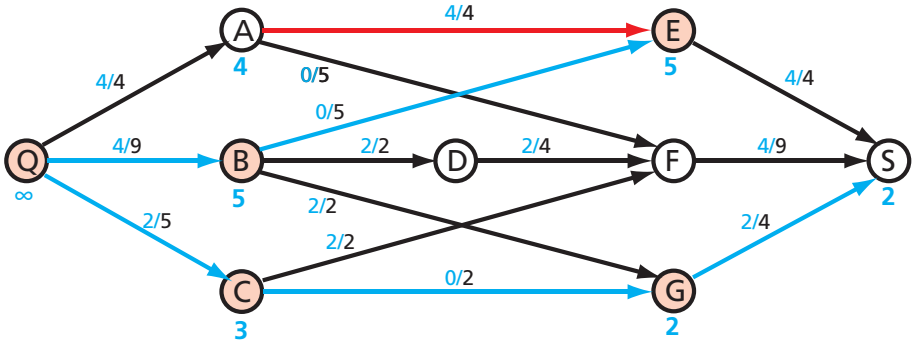


Abbildung 10.31
Fünfter zunehmender Fluss von zwei Einheiten

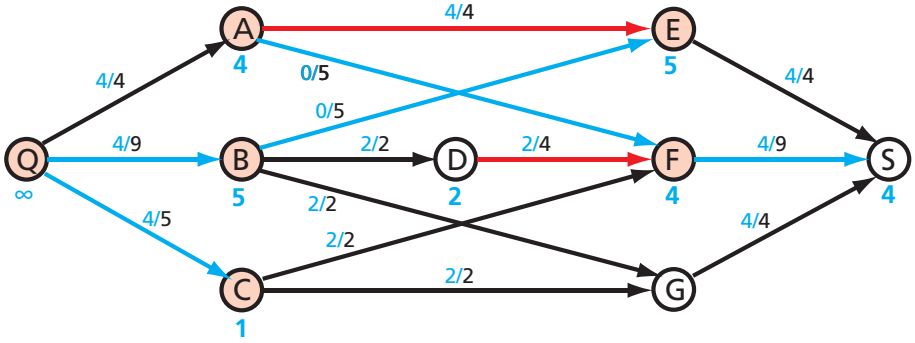


Abbildung 10.32
Sechster zunehmender Fluss von vier Einheiten

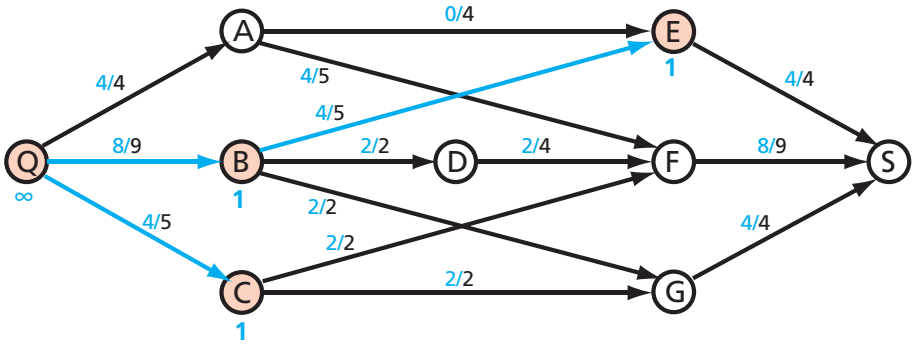


Abbildung 10.33
Kein weiterer zunehmender Fluss kann gefunden werden. Der Algorithmus wird abgebrochen, da der maximale Fluss erreicht ist.

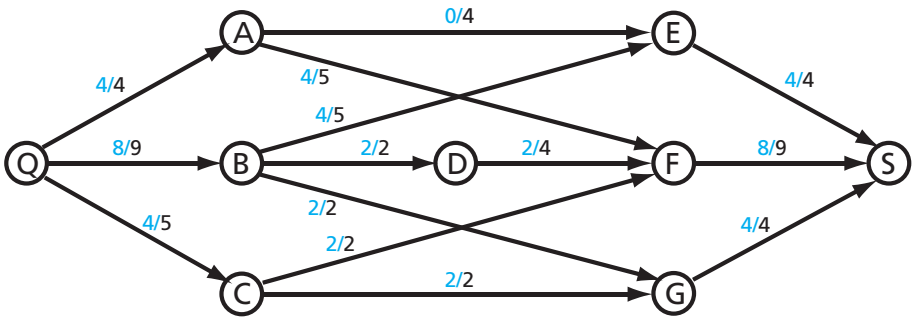


Abbildung 10.34
Endresultat: Zwischen Quelle und Senke können 16 Einheiten fließen.

Hochzeitsglocken

Informatiker haben für die Lösung einer Aufgabe immer die Möglichkeit, entweder die Aufgabe grundlegend neu zu bearbeiten – mit Hilfe von Entwurfsstrategien wie „divide et impera“ oder dem Greedy-Prinzip. Als Alternative dazu können sie die Aufgabenstellung aber auch so darstellen, dass sie sich mit einem bereits bekannten Verfahren lösen lässt.

Mit einem besonders eindrucksvolles, weil unerwarteten Beispiel dieser Art möchte ich das Kapitel abschließen. Versuchen Sie die folgende Aufgabe mit dem Algorithmus von Ford und Fulkerson zu lösen! Ich gebe zu, das ist knifflig, aber ich bin mir sicher, Sie schaffen das, ohne in die Lösung zu spicken …

Die Agentur „Ich und Du“ veranstaltet regelmäßig Speed-Dating-Events. Am Ende haben sich alle Teilnehmerinnen und Teilnehmer kennen gelernt und bekommen eine Liste mit den Namen aller potentiellen Partner bzw. Partnerinnen. Dort kreuzen alle an, mit wem man sich eine Heirat vorstellen könnte. Die Agentur wertet dann die Listen aus. Ergebnis ist eine lange Liste mit möglichen Paarungen.

Das Unternehmen veranstaltet nach dem Speed-Dating das Romance-Dinner, bei dem nur potentielle Paare teilnehmen, also Paare, die sich gegenseitig auf den Listen angekreuzt haben. Da „Ich und Du“ am Romance-Dinner am meisten verdient, möchte die Agentur selbstverständlich diesen Gewinn maximieren und möglichst viele Paare gleichzeitig zusammenbringen. Das erscheint zunächst schwierig, da die Reihenfolge der Paarbildung offensichtlich eine Rolle spielt.

Ein einfaches Beispiel verdeutlicht das grundsätzliche Problem: Es stehen Anna, Berta und Christine sowie Markus, Norbert und Olaf zur Disposition. Die folgende Tabelle gibt alle möglichen Paarungen an.

Frau	Mann
Anna	Norbert
Anna	Olaf
Berta	Markus
Christine	Norbert

Verkuppeln wir nun zu Beginn Anna mit Norbert, ist noch das Paar „Berta und Markus“ möglich, Christine und Olaf gehen leer aus, die Agentur kann nur zwei Tickets für das Romance-Dinner verkaufen. Verkuppeln wir allerdings die Paare „Anna und Olaf“, „Berta und Markus“ sowie „Christine und Norbert“, sind alle glücklich und es gibt auch mehr Einnahmen für die Agentur.

Ein kleines Indiz für die Ähnlichkeit zur Ermittlung des maximalen Flusses ist, dass auch hier offenbar die Reihenfolge der Bearbeitung eine Rolle spielt und man zunächst an eine Lösung nach dem Prinzip Backtracking denkt, um die maximale Zahl an Paarungen für das Romance-Dinner zu finden.

Nun sind Sie an der Reihe: Helfen Sie der Agentur, ihren Gewinn auch im nächsten Beispiel mit neun Frauen und neun Männern zu maximieren! Überlegen Sie vielleicht erst, wie man die möglichen Paarungen günstig darstellen kann, so dass der Algorithmus von Ford und Fulkerson darauf anwendbar wird. Wenn Sie auf keine Lösung kommen, spielen Sie erst das kleinere Beispiel (mit sechs Personen) durch.

Frau	Mann
Anna	Paul
Anna	Rudolf
Berta	Olaf
Berta	Udo
Christine	Norbert
Christine	Theo
Doris	Markus
Doris	Theo

Frau	Mann
Doris	Wilhelm
Elise	Markus
Elise	Rudolf
Elise	Theo
Elise	Wilhelm
Francine	Norbert
Francine	Viktor
Gudrun	Paul

Frau	Mann
Gudrun	Viktor
Herta	Markus
Herta	Olaf
Herta	Viktor
Inge	Paus
Inge	Udo
Inge	Viktor

Sie sind sicherlich bereits auf die Lösung gekommen. Trotzdem möchte ich an dieser Stelle erst einen Tipp geben, so dass auch diejenigen, die die Aufgabe noch nicht komplett gelöst haben, noch die Chance bekommen, selbst etwas weiter zu denken!

Wir nehmen hierfür das Beispiel mit nur sechs Personen. Prinzipiell war es für alle Aufgaben dieses Kapitels notwendig, einen Graphen zu zeichnen, das Problem also irgendwie durch Knoten (Kringel) und Kanten (Verbindungen zwischen den Kringeln) darzustellen. Versuchen wir es doch einmal mit Knoten für jede Person und mit Kanten für jede mögliche Paarung. Abbildung 10.35 zeigt, wie das dann aussieht. Dabei sind die Knoten der Übersicht halber mit den Anfangsbuchstaben der Namen beschriftet.

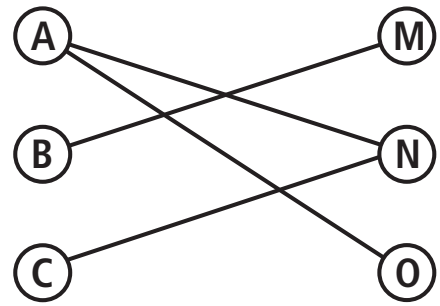
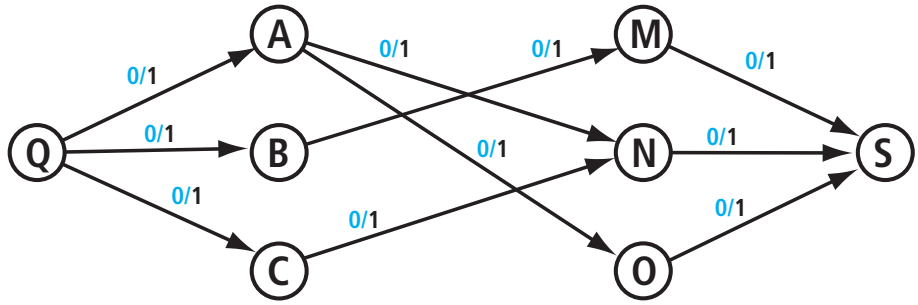


Abbildung 10.35
Das Hochzeitsproblem als Graph

Das ist jetzt allerdings noch kein Flussgraph, den wir mit dem Algorithmus von Ford und Fulkerson lösen könnten. Dafür fehlen noch ein paar Kleinigkeiten. Kommen Sie darauf, welche?

Abbildung 10.36
Das Hochzeitsproblem als Flussgraph



Nehmen wir den Graphen mit den Paarungen als Leitungsnetz mit der Fließrichtung von links nach rechts, fehlt uns nur noch eine Quelle und eine Senke. In Abbildung 10.36 sind diese eingezeichnet (als Knoten Q und S). Können Sie das Problem nun lösen?

Führen wir doch den Algorithmus einfach einmal durch, indem wir bei der Traversierung immer erst oben, dann unten suchen. Abbildung 10.37 zeigt die einzelnen zunehmenden Wege und das Ergebnis. In den ersten beiden Schritten sehen Sie die „normale" Vorgehensweise, im dritten Schritt wurde eine Rückkante verwendet, um den Fluss weiter zu steigern.

Wie kann man diese nun als Lösung für das Hochzeitsproblem interpretieren? Ganz einfach: Jede Leitung zwischen einer Frau und einem Mann steht für eine mögliche Paarung. Fließt Wasser hindurch, findet eine Hochzeit statt, sonst nicht.

Warum entspricht der maximale Fluss aber einer günstigen Paarungssituation? Für jede durchgeführte Paarung fließt eine Einheit Wasser von der Quelle zur Senke. Wenn die Menge an Wasser von der Quelle zur Senke also maximal ist, gilt dies auch für die Anzahl der gebildeten Paare. Das Wasser symbolisiert also quasi den Gewinn der Agentur „Ich und Du" beim Romance-Dinner.

Nach diesem Schema können (und sollen) Sie nun auch das große Problem mit neun Paaren lösen.

Die Abbildungen 10.38 bis 10.45 ab der nächsten Doppelseite zeigen die Lösung des Problems. Die Kapazitäten werden nicht extra aufgeführt, sie sind ohnehin jeweils 1. Das Gleiche gilt für die Pfeile – wir gehen davon aus, dass alle von links nach rechts verlaufen. Wir wollen den Algorithmus sehr konsequent durchlaufen und verwenden daher auch die Breitensuche. Die entsprechenden Schritte sind in den Abbildungslegenden beschrieben.

Alle Teilnehmer konnten also mit dem Algorithmus unter die Haube gebracht werden, die Agentur „Ich und Du" kann sich über maximale Gewinne freuen! Die Paarungen sind zur Verdeutlichung in der nebenstehenden Tabelle zusammengefasst.

Solche Paarbildungen sind übrigens nicht nur für Eheanbahnungsinstitute wichtig! Stellen Sie sich zum Beispiel vor, dass in einem Firmennetzwerk jeweils zwei Computer gegenseitig Sicherungskopien anfertigen und hier günstigerweise Paare gefunden werden müssen, die mit schnellen Leitungen untereinander verbunden sind.

Frau	Mann
Anna	Rudolf
Berta	Olaf
Christine	Norbert
Doris	Wilhelm
Elise	Theo
Francine	Viktor
Gudrun	Paul
Herta	Markus
Inge	Udo

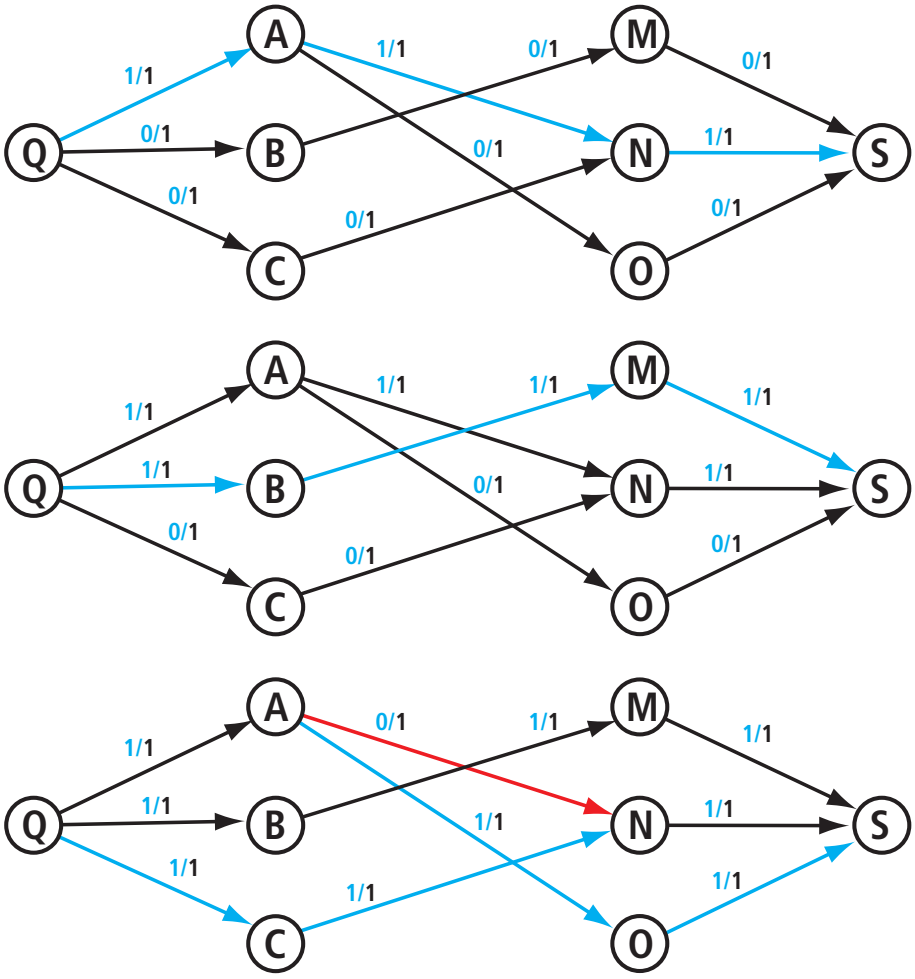


Abbildung 10.37
Zunehmende Wege in der Repräsentation des Hochzeitsproblems als Flussgraph

Bei der Planung von Projekten haben verschiedene Mitarbeiter unterschiedliche Kompetenzen für Teilaufgaben. Der vorgestellte Algorithmus kann auch verwendet werden, um die Aufgaben den Mitarbeitern ihrer Kompetenz entsprechend optimal zuzuordnen.

Wenn Sie noch Lust haben, sollten Sie das auch an weiteren selbst ausgedachten Beispielen ausprobieren.

Abbildung 10.38
Der anfängliche Flussgraph für das große Hochzeitsproblem

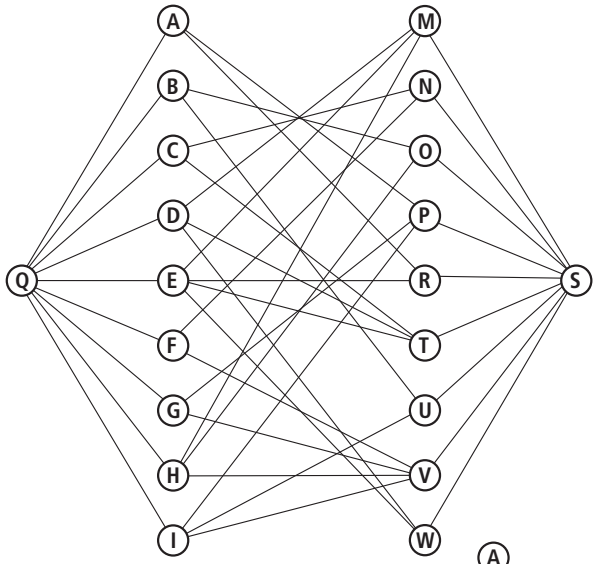


Abbildung 10.39
Die ersten sechs Paarbildungen verlaufen zunächst problemlos. Die Flüsse sind einfach durch die Blaufärbung der Leitung dargestellt. Sie können sich jeweils die Beschriftung „1/1“ hinzudenken oder diese auch noch einfügen. Alle schwarzen Linien müssen dann die Beschriftung „0/1“ tragen.

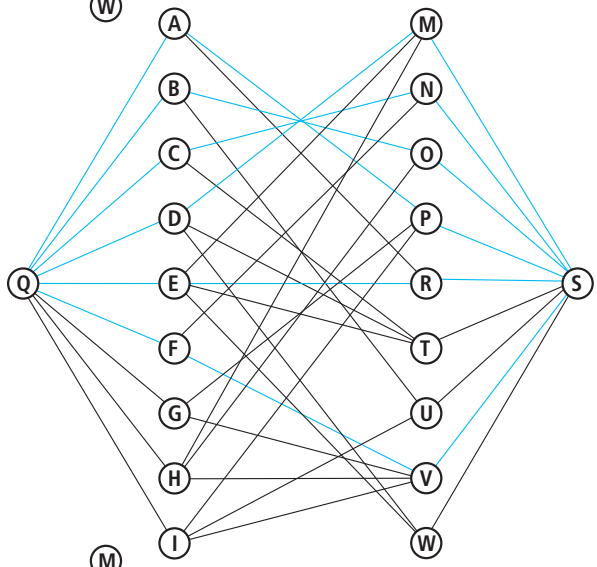
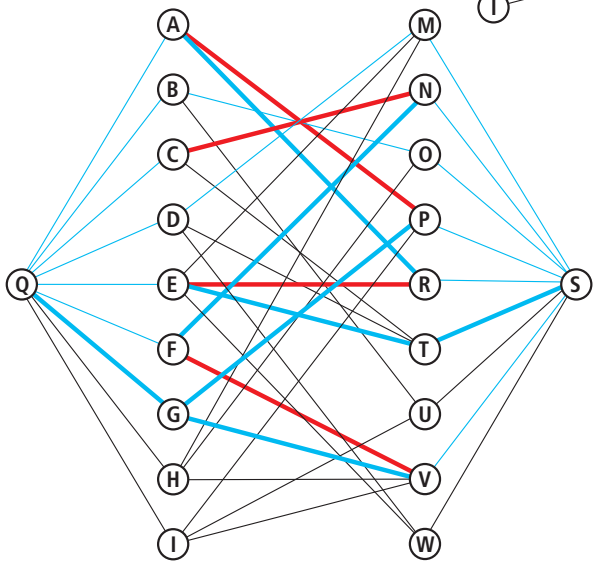


Abbildung 10.40
Bei der Suche nach einem Partner für Gudrun müssen wir nun einige vorher getroffene Entscheidungen wieder rückgängig machen: Wenn Sie die Breitensuche verwenden, finden Sie die dick gezeichneten zunehmenden Wege. Der Algorithmus findet die Senke dann zuerst über die Knoten Q, G, P, A, R, E, T, S.



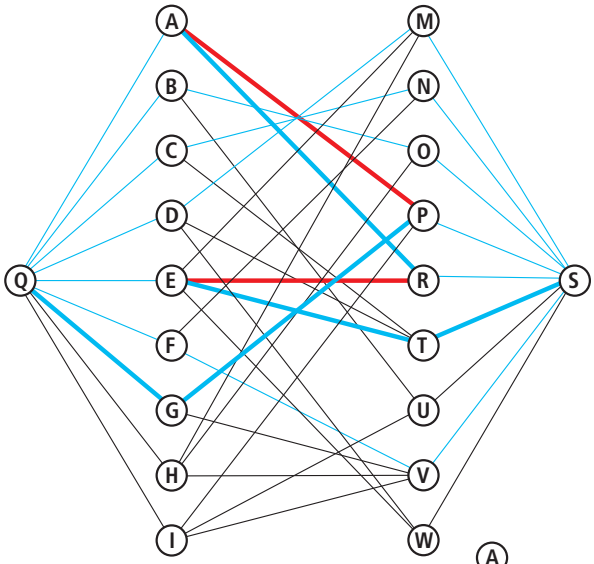


Abbildung 10.41
Nochmals zur Verdeutlichung: Beim Traversieren mit Breitensuche findet der Algorithmus den hier eingezeichneten Weg zwischen Quelle und Senke zuerst.

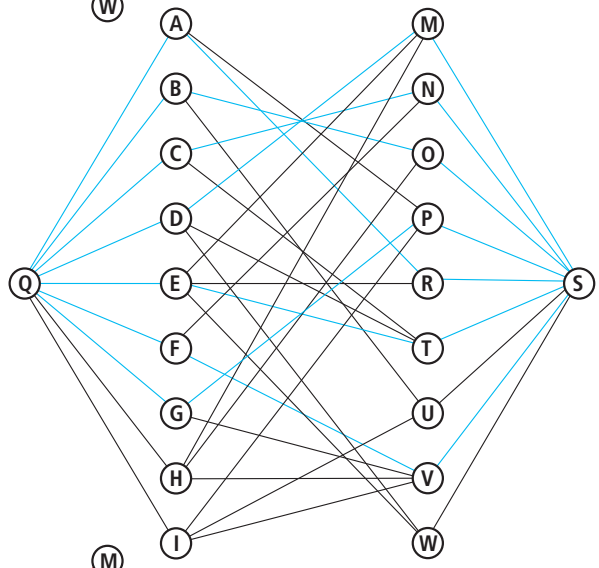


Abbildung 10.42
Die ersten sieben Paare sind unter der Haube … vorerst. Erinnern Sie sich, was die blauen und die roten Kanten zu sagen hatten? Die blauen Kanten bilden neue Flüsse (Markierung „1/1"), während bei den roten der bestehende Fluss rückgängig gemacht wird.

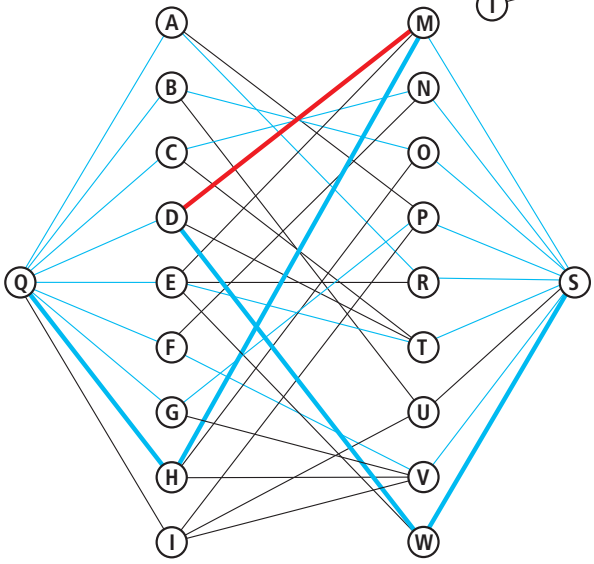


Abbildung 10.43
Als Nächstes finden wir auch für Knoten mehrere zunehmende Wege mit Rückwärtskanten.

Abbildung 10.44
In dieser Abbildung wurde der erfolgreiche Weg mit Rückkante verwirklicht.

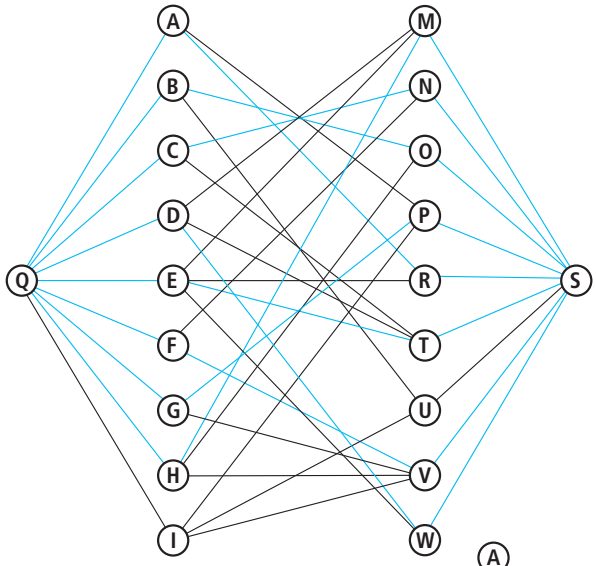
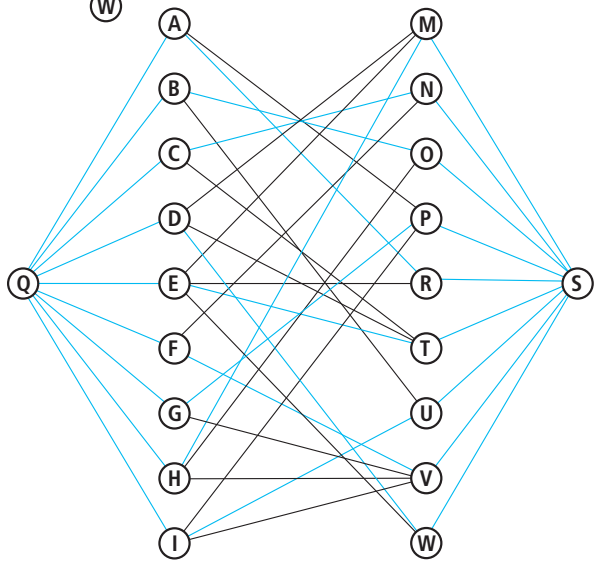


Abbildung 10.45
Die letzte Paarung lässt sich wieder einfach herstellen. Sie sehen hier den fertigen Graphen. Die blauen Kanten symbolisieren einen Fluss, also eine gültige Paarung.



Resümee

In diesem Kapitel haben Sie gelernt, wie ein am Ende recht einfacher Algorithmus dazu dient, Leitungsnetze besser zu planen und zu handhaben. Der Weg, diesen Algorithmus zu entwickeln, war jedoch gar nicht so einfach. Verschiedene Entwurfsschemata wie Backtracking und das Greedy-Prinzip konnten uns jedoch als roter Faden dabei helfen.

Sehr wichtig ist auch die Erkenntnis, dass bestehende Algorithmen oft für Aufgaben einsetzbar sind, mit denen sie auf den ersten Blick gar nichts zu tun haben. So konnten Sie mit einem Verfahren sowohl Wassermengen in einem Leitungsnetz optimieren als auch Partner für ein Heiratsinstitut zuordnen.

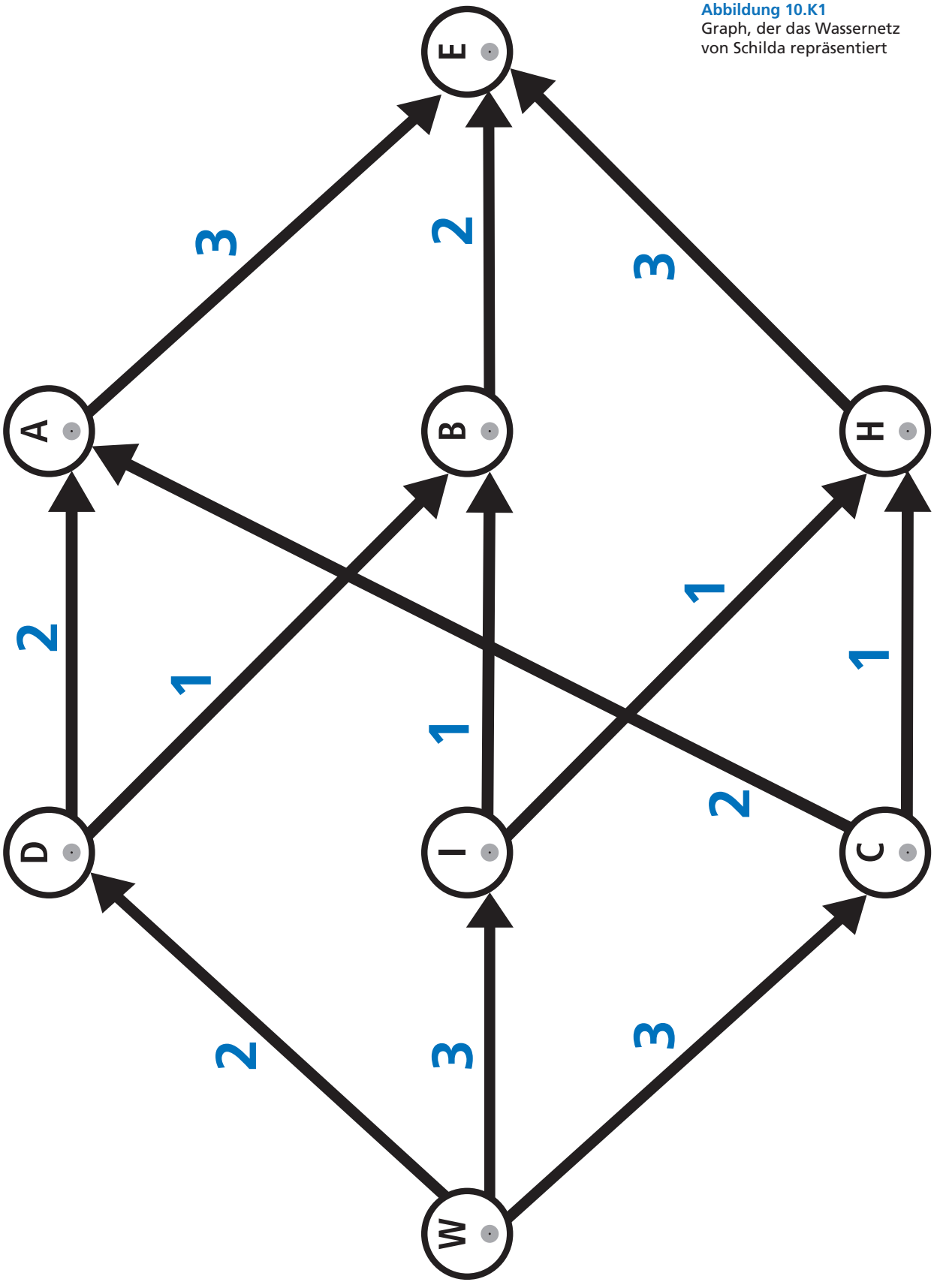


Abbildung 10.K1
Graph, der das Wassernetz von Schilda repräsentiert

11. Ordnung im Chaos

Viele Chefs glauben, dass Chaos auf dem Schreibtisch der Mitarbeiterinnen und Mitarbeiter für Chaos in deren Kopf steht. Verschiedene Abhandlungen über Karriereplanung empfehlen daher dringend, Ordnung zu halten. Albert Einstein und Sigmund Freud waren beide eher Befürworter des Chaos: Laut Berichten von Zeitgenossen hatten sie den typischen „Krater“ aus Dokumenten und angefangenen Schriftstücken auf ihrem Arbeitsplatz. Für Einstein war dieses Chaos der beste Beweis für die Sätze der Thermodynamik, nach denen Chaos der natürliche Zustand ist. Sigmund Freud vertrat sogar etwas weitergehende Theorien, was den psychologischen Hintergrund eines Ordnungsliebhabers angeht, aber das möchte ich hier gar nicht vertiefen.

Wichtig ist, dass es offenbar zwei Geisteshaltungen bezüglich des Arbeitsplatzes gibt – die der offensichtlichen Ordnung und die des offensichtlichen Chaos. Die Informatik ist bereits auf vielen anderen Gebieten Vermittler zwischen unterschiedlichen Disziplinen und vielleicht kann sie auch in diesem Fall eine Brücke bilden. Eine der wichtigen Methoden, um Daten im Computerspeicher abzulegen und schnell wieder darauf zuzugreifen, führt nämlich zu einer chaotischen Anordnung! Trotzdem können einzelne Datensätze in Windeseile wieder hervorgeholt werden. Trügt hier also der Schein? Im folgenden Kapitel werden wir uns dieser Thematik näher widmen.

Es wäre übrigens günstig, wenn Sie das Kapitel „Sortieren“ vorher durcharbeiten, da das Folgende größtenteils daran anknüpft.

Warum Ordnung?

Im zweiten Kapitel haben wir uns zunächst mit dem Sortieren beschäftigt, ohne wirklich darüber nachzudenken, warum das eigentlich sinnvoll ist. Eine Begründung folgte dann am Ende des achten Kapitels. Hier wollen wir nun gezielt anknüpfen.

Wir versetzen uns ein weiteres Mal in die Lage des Computers. Packen Sie den Papierspeicher aus und 16 Kärtchen mit Zahlen darauf. Mischen Sie die Karten und verteilen Sie diese auf den ersten 16 Speicherpositionen wie in Abbildung 11.1. Die Zeiger „An-

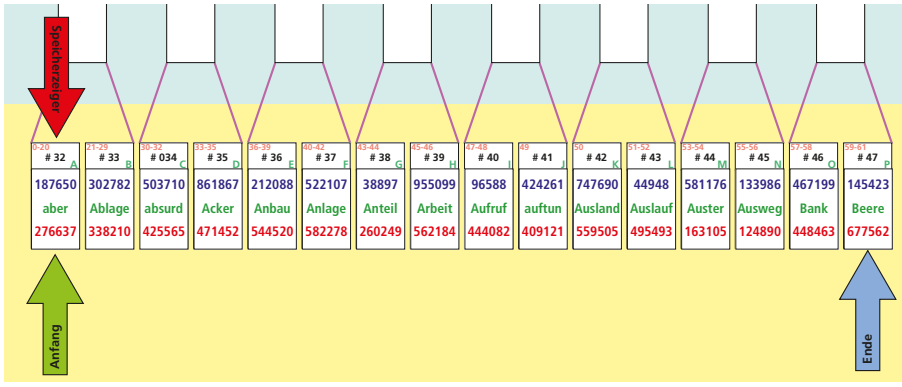


Abbildung 11.1
Papierspeicher mit chaotischem Inhalt

fang“, „Ende“ und „Speicherzeiger“ erleichtern das Nachspielen der Vorgehensweise des Computers. Denken Sie daran: Der Computer kann immer nur die Karte unter einem der Zeiger „sehen“ und muss den Zeiger neu setzen, um eine andere Karte anzuschauen.

Finden Sie als Computer eine bestimmte vorhandene Karte (zum Beispiel die mit der blauen 38897). Versuchen Sie dabei möglichst geschickt vorzugehen. Schreiben Sie auf, wie viele „Rechenschritte“ Sie benötigt haben, und wiederholen den Versuch ein paar Mal. Auf diese Weise können Sie abschätzen, wie lange die Suche im Mittel dauert.

Egal, wie Sie es drehen und wenden, letztendlich bleibt bei einem chaotischen Speicherinhalt immer nur die Möglichkeit, eine Karte nach der anderen zu überprüfen, bis man die gesuchte gefunden hat – zum Beispiel nach dem Algorithmus in Abbildung 11.2. Man nennt das auch „sequentielle Suche“, weil man der Reihe nach sucht.

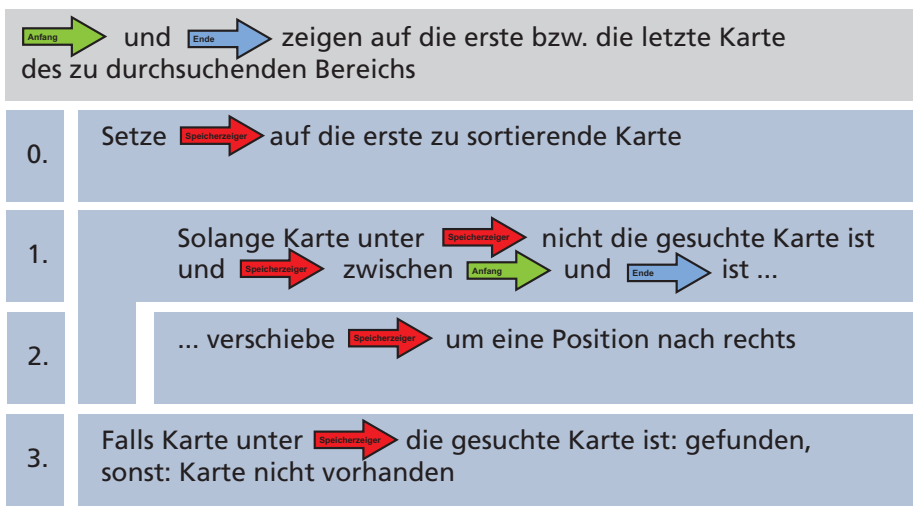
Im Mittel muss man etwa die Hälfte der Karten anschauen, bis man fündig wird, denn die gesuchte Karte kann sich ja zufällig ganz am Anfang oder auch ganz am Ende befinden. Bei 16 Karten wird man also durchschnittlich etwa nach 8 Karten fertig sein.

Als Nächstes sortieren Sie bitte die Karten, zum Beispiel nach der blauen Zahl. Verwenden Sie zum Üben am besten eines der Verfahren aus dem zweiten Kapitel. Selbstverständlich dürfen Sie die Karten auch „einfach so ordnen“. Abbildung 8.3 zeigt eine sortierte Folge.

Versuchen Sie nun wiederum, in möglichst wenigen Schritten eine bestimmte Karte zu finden. Das Verfahren haben wir am Ende von Kapitel 8 bereits verwendet. Kleiner Tipp: Fangen Sie mit dem Speicherzeiger auf der Position #39 an, also etwa in der Mitte.

Vielleicht erinnern Sie sich noch an das Zahlenratespiel: Ein Spieler denkt sich eine Zahl zwischen 0 und 100 aus und ein anderer soll die Zahl in möglichst wenigen Versuchen raten. Nach jedem Versuch sagt der erste Spieler, ob die gesuchte Zahl größer oder kleiner ist.

Abbildung 11.2
Algorithmus für sequentielle Suche



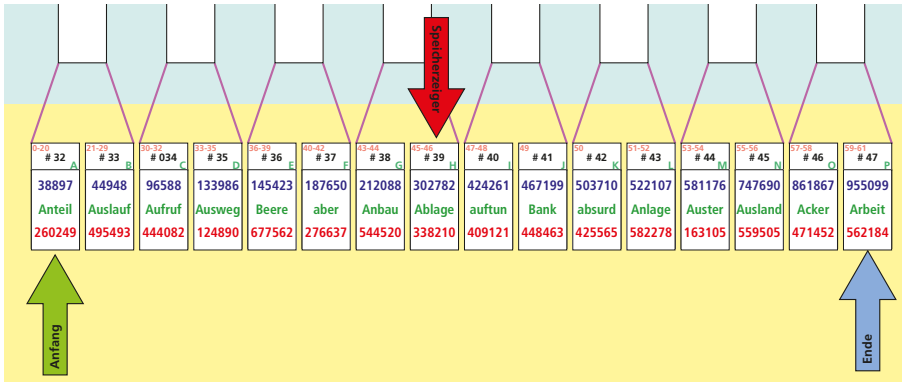


Abbildung 11.3
Papierspeicher mit sortierter Kartenfolge

Hier beginnt man am besten in der Mitte und halbiert dann immer die Anzahl noch möglicher Zahlen. Genau das gleiche Prinzip können wir auch mit dem Papierspeicher zum Suchen einer Karte benutzen, falls die Karten im Speicher sortiert sind:

Wir beginnen in der Mitte und kontrollieren, ob die Karte die gesuchte ist. Wenn nein, schauen wir, ob die Zahl größer oder kleiner als die gesuchte ist. Ist die gesuchte Zahl größer, muss sie sich rechts vom Speicherzeiger befinden. Ist sie kleiner, muss sie links sein. Das ergibt sich direkt aus der Sortierung!

Auf diese Weise haben wir bereits die Hälfte der vorhandenen Karten ausgeschlossen und brauchen nur noch in der anderen Hälfte zu suchen. Wir setzen den Speicherzeiger wiederum in die Mitte und wenden das gleiche Verfahren an, um erneut die Hälfte der Karten auszuschließen.

Schaffen Sie es, dafür den Algorithmus zum Beispiel in der Art von Abbildung 11.2 zu formulieren und aufzuschreiben?

Einen entsprechenden Algorithmus sehen Sie in Abbildung 11.4. Das Verfahren heißt auch „binäre Suche“ in einer sortieren Liste, weil man den Suchbereich immer in zwei Hälften teilt. Auch dieser Algorithmus funktioniert übrigens nur korrekt, wenn sich die gesuchte Karte auf jeden Fall im Speicher befindet.

Wenn Sie die binäre Suche ein paar Mal durchführen, stellen Sie fest, dass Sie hier durchschnittlich vier, maximal fünf Vergleiche benötigen, um die Karte zu finden. Das Verfahren bringt uns also schneller ans Ziel.

Das macht sich bei größeren Datenmengen noch viel stärker bemerkbar: Die binäre Suche schließt pro Schritt die Hälfte aller Daten aus. Umgekehrt ausgedrückt benötigen wir also für die Suche in einer doppelt so großen Datenbank nur einen einzigen Schritt mehr. Bei der sequentiellen Suche bräuchte der Algorithmus auch tatsächlich doppelt so viele Schritte.

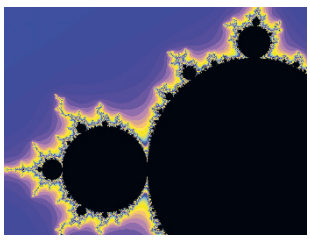
Die Experten wissen natürlich bereits aus Kapitel 2, dass sich der Suchaufwand bei n Karten formal durch $\log_2 n$ ausdrücken lässt.

Abbildung 11.4
Algorithmus zur binären Suche

Anfang und Ende zeigen auf die erste bzw. die letzte Karte des zu durchsuchenden Bereichs

0.	Setze Speicherzeiger auf die Mitte zwischen Anfang und Ende (falls es keine „genaue" Mitte gibt, nimm die Karte links davon)
1.	Ist die gesuchte Zahl größer als die Zahl auf der Karte unter Speicherzeiger ?
2.	Wenn JA: Lege Anfang auf die Karte rechts von Speicherzeiger
3.	Wenn NEIN: Lege Ende auf die Karte links von Speicherzeiger
4.	Wiederhole das ab Schritt (0.), bis die Karte unter Speicherzeiger die gesuchte Karte ist oder Ende links von Anfang steht
5.	Falls Karte unter Speicherzeiger die gesuchte Karte ist: gefunden, sonst: Karte nicht vorhanden

Das Chaos im Wandel der Zeit
Das Wort kommt aus dem Griechischen und meint „den unendlichen leeren Raum" oder „die gestaltlose Urmasse des Weltraums". Biblisch wurde die Welt aus dem Chaos geschaffen. Früher verstand man also etwas Ursprüngliches, noch nicht Geformtes unter dem Chaos.
Heute bezeichnet man jede Form von Unordnung – besonders auch die von uns allen geschaffene – als Chaos. Die Chaostheorie beschäftigt sich hingegen eher mit dem Gegenteil: Ordnung soll in scheinbar chaotischen Systemen gefunden werden. Bekanntestes Beispiel aus der Informatik sind die selbstähnlichen Fraktale – zum Beispiel die aus einer sehr einfachen Formel gewonnene Mandelbrot-Menge („Apfelmännchen"), die auch beliebig fein dargestellt immer wieder die gleichen, sich wiederholenden Grundfiguren enthält.



Solche Bilder kann man leicht mit ein paar Zeilen Programmcode selbst herstellen.

Ordnung oder Chaos?

Wir haben damit nochmals bestätigt, dass Ordnung nicht nur einen ästhetischen Wert für uns Menschen darstellt, sondern selbst für Maschinen und deren Algorithmen echte Vorteile bringt. Als Nächstes möchte ich Ihnen die Frage stellen, was Ordnung eigentlich ist. Hierfür machen wir ein kleines Spiel:

Die Spalten in Abbildung 11.5 enthalten immer die gleichen zehn Zahlen in unterschiedlicher Reihenfolge. Welche Spalten sind sortiert, welche nicht? Können Sie den Grad der Unordnung in den Ihrer Meinung nach unsortierten Spalten beurteilen?

Selbstverständlich erkennen Sie die gelbe Reihe A als sortiert, die anderen scheinen dagegen auf den ersten Blick etwas chaotisch zu sein. Vielleicht haben Sie aber doch bereits eine Ordnung ausmachen können.

Schlüssel hierfür ist das Sortierkriterium. Bei der gelben Reihe ist das ganz offensichtlich – die Zahlen sind der Größe nach von der kleinsten zur größten aufgeführt. Die rote Reihe ist da schon etwas subtiler: Betrachten Sie nur die drei hinteren Stellen der Zahlen, also die letzten drei Ziffern. Stimmt nun die Sortierung wieder?

Ich behaupte jetzt, dass auf die gleiche Weise auch für die blaue und die grüne Reihe je ein Sortierkriterium existiert. Es ist allerdings eine ziemlich harte Nuss, darauf zu kommen. Wenn Sie Lust haben, versuchen Sie die Regel herauszufinden, sonst lesen Sie einfach weiter.

0	727	932018		973412
1	213121	213121	213121	727
2	545955	832213	903512	961347
3	832213	961347	961347	545955
4	903512	973412	979834	945630
5	932018	903512	932018	903512
6	945630	945630	545955	
7	961347	727	727	213121
8	973412	979834	973412	832213
9	979834	545955	945630	979834
10			832213	932018

Abbildung 11.5
Jede Spalte enthält die gleichen zehn Zahlen. Welche davon ist geordnet?

In der blauen Reihe steckt jeweils die doppelte Quersumme: Nehmen Sie die Ziffern der Zahlen einzeln und addieren Sie sie. Das Gleiche machen Sie nochmals mit dem Ergebnis. Die Resultate liegen bei den gegebenen Zahlen zwischen 1 und 10 und genau danach sind sie sortiert.

Beispiele:

- Die Quersumme von 945630 ist 9 + 4 + 5 + 6 + 3 + 0 = 27
- Die Quersumme von 27 ist 2 + 7 = 9

Daher ist 945630 in Zeile 9 eingeordnet.

Um das Geheimnis der grünen Reihe zu knacken, denken Sie an die schriftliche Division mit Rest aus der Grundschule zurück. Dort rechnet man nicht mit Brüchen, sondern nur mit ganzen Zahlen. Dabei kann immer ein Divisionsrest übrig bleiben.

27 geteilt durch 12 ergibt 2, Rest 3

In der Informatik braucht man häufig nur den Rest dieser Division und gar nicht das Ergebnis, daher hat man die Berechnung des Restes zur eigenen Rechenart gemacht, genannt „modulo“ oder abgekürzt „mod“. Man schreibt zum Beispiel:

27 mod 12 = 3

Damit gleich Verwechslungen vermieden werden: Dies ist ein sehr verwandter, aber trotzdem etwas anderer Gebrauch von „mod“ als in der Mathematik!

Resteverwertung

So wie für das Enträtseln der grünen Spalte benötigen wir den Divisionsrest noch häufiger in diesem Kapitel und auch insgesamt für verschiedene Verfahren in der Informatik. In Zeiten des Taschenrechners verlernen wir sehr schnell die schriftliche Division mit Rest, wie man sie in der Grundschule betreibt. Das ist völlig normal – selbst bei den Studierenden der Informatik oder Mathematik gibt es sehr viele, die sich diese Technik erst wieder aneignen müssen. Daher hier eine kleine Wiederholung für alle, die sich nicht mehr genau daran erinnern ...

Prinzipiell nutzen wir unser Dezimalsystem aus, um größere Divisionen der Form

Dividend : *Divisor* = *Ergebnis*

in kleinere, leichter zu beherrschende Operationen zu unterteilen – ein sehr Informatik-nahes Vorgehen, wie zum Beispiel auch die Äthiopische Multiplikation weiter vorne im Buch! Am Beispiel lässt das sich immer am besten lernen, daher versuchen wir es mit:

498737894 : 13 = ?

Zunächst nehmen wir so viele der ersten Ziffern des Dividenden, dass die entstehende Zahl gerade größer wird als der Divisor – in diesem Fall also „49". Wir teilen diese Zahl durch den Divisor und schreiben das (einstellige) Ergebnis hinter das Gleichheitszeichen. Dabei beachten wir nur ganze Anteile des Ergebnisses.

49 8737894 : 13 = 3

Die Zahl ist aber meistens nicht glatt teilbar. In diesem Fall bedeutet die 3 also, dass der Divisor 13 genau 3-mal glatt in 49 unterzubringen ist. Da 3 mal 13 jedoch 39 ergibt, bleibt ein Rest. Diesen ermitteln wir formal, indem wir die gerade errechnete Ziffer mit dem Divisor multiplizieren (= 39) und direkt unter die ersten Ziffern des Dividenden schreiben. Per Subtraktion ermitteln wir dann den Divisionsrest dieser ersten Operation und schreiben ihn hin. Mit diesem Rest muss dann die Division noch weiter betrieben werden.

498737894 : 13 = 3
39
10

Allerdings haben wir ja noch einen weiteren „Rest", nämlich die Ziffern, die wir bisher gar nicht betrachtet hatten. Sie sind in der folgenden Formel nochmals grau wiederholt, weil man sie normalerweise nicht hinschreibt, aber trotzdem mit ihnen rechnet:

498737894 : 13 = 3
39
108737894 : 13 = 3...

Jetzt geht es wirklich à la Informatik weiter: Wir verfahren mit dieser neuen Berechnung (fast) ebenso wie am Anfang mit der kompletten Division, nur dass die bereits ermittelten Anteile (in diesem Fall die 3) einfach stehen bleiben. Statt wieder zu ermitteln, wie viele Ziffern notwendig sind, damit sie größer als der Divisor werden, nehmen wir jetzt immer nur die nächste Ziffer zum Rest hinzu – in diesem Fall also

die 8. Danach wird wieder geteilt sowie der Rest und damit eine neue Rest-Berechnung ermittelt.

```
498737894 : 13 = 3
39
108737894 : 13 = 38
104
  4737894 : 13 = 38...
```

Das machen wir, bis der entstandene Divisionsrest so klein ist, dass er kleiner als der Divisor ist – die Zahl lässt sich nicht mehr weiter teilen und wir schreiben den Rest, als solchen gekennzeichnet, einfach daneben:

```
498737894 : 13 = 38364453 Rest 5
39
108737894 : 13 = 3...
104
  47737894 : 13 = 38...
  39
   83737894 : 13 = 383...
   78
    57894 : 13 = 3836...
    52
     5894 : 13 = 38364...
     52
      694 : 13 = 383644...
      65
       44 : 13 = 3836445...
       39
        5 : 13 = 38364453 Rest 5
```

Die grauen Teile schreibt man normalerweise nicht mehr auf, sie dienen hier nur der Verdeutlichung, dass im Prinzip immer wieder der gleiche Algorithmus durchgeführt wird. Für die modulo-Rechnung, wie sie in diesem Kapitel häufig verwendet wird, benötigen wir nur den Rest, nicht das eigentliche Ergebnis.

Um herauszufinden, wie die Zahlen der grünen Spalte geordnet sind, berechnen Sie einfach immer den Divisionsrest beim Teilen durch 11.

Beispiel: 945630 mod 11 = 4, daher ist 945630 in Zeile 4 einsortiert.

Es sind also wirklich alle der oben angegebenen Spalten geordnet, allerdings nur in der gelben Spalte nach einem uns Menschen offensichtlichen Kriterium. An die Ordnung der roten Spalte könnten wir uns vielleicht noch gewöhnen, aber Berechnungen anstellen, um Zahlen zu sortieren, so wie in den restlichen Spalten – das geht zu weit. So werden Computer auch immer die „menschlichen“ Sortierkriterien verwenden, wenn es darum geht, den Benutzern etwas zu präsentieren. Beispiele hierfür sind Telefonverzeichnisse, Kundendateien, eine Hand voller Spielkarten bei einem Skatprogramm usw.

Ordnung ist nicht gleich Ordnung

Wie sieht es allerdings aus, wenn die Daten ausschließlich im Computer gespeichert werden und dort auch verbleiben? Wenn jeder, der einen bestimmten Datensatz sucht, einfach den Computer danach fragt? Computer rechnen sehr schnell, haben also kein Problem mit „seltsamen" Sortierkriterien. Das heißt, prinzipiell spricht nichts dagegen, Daten im Computerspeicher nach etwas anderem als der Größe zu sortieren. Die nächste Frage ist allerdings: Spricht etwas dafür?

Um sie zu beantworten, schauen wir uns nochmals die Abbildung 11.5 an. Um einen Eintrag aus der gelben Spalte zu finden, kann man die binäre Suche anwenden. Genauso kann man auch suchen, um mit dem entsprechenden Kriterium einen Eintrag aus den anderen Spalten zu finden. Aber es geht auch schneller:

In den hinteren Spalten bestimmt das Sortierkriterium nicht nur die relative Position der Einträge untereinander (so wie „832213 kommt vor 973412"), sondern es legt die absolute Position in der Tabelle fest.

Um einen Eintrag in der roten Spalte zu finden, nehmen Sie einfach die dritte Ziffer von rechts – sie ist identisch mit der Zeilennummer. Bei der blauen Spalte ist das die doppelte Quersumme und in der grünen Spalte der Divisionsrest beim Teilen durch 11. Das erklärt auch das Rätsel, warum in dieser Spalte Zeile 6 keinen Eintrag enthält: Unter den zu speichernden Zahlen gibt es einfach keine mit Rest 6 bei der Division durch 11.

Erkennen Sie den Vorteil? Statt nach einem Eintrag suchen zu müssen, berechnet der Computer einfach dessen Position. Zum Speichern genügt ein einzelner Schreibvorgang. Um die Daten wieder abzurufen, genügt ein einziger Lesevorgang. Dieses Verfahren setzt man daher tatsächlich immer dann ein, wenn es wichtig ist, Daten sehr schnell abzuspeichern und sehr schnell wieder abzurufen.

Wie oben in den hinteren Spalten werden die Daten dabei oft in einer für den Menschen chaotisch aussehenden Weise angeordnet. Daher nennt man das Verfahren „Hashing" von englisch „Hash" – zu Deutsch „Durcheinander, Kuddelmuddel".

Loci

Die sogenannte Loci-Methode zur Steigerung der Merkfähigkeit macht sich übrigens zunutze, dass wir uns recht gut daran erinnern können, wo wir bestimmte Dinge verstauen. Möchten Sie sich etwa die Liste für den nächsten Einkauf merken, legen Sie die gewünschten Objekte in Gedanken in Ihrem Büro oder Wohnzimmer ab. Im Laden müssen Sie sich dann nur noch das Zimmer vorstellen, um sich wieder an die Einkaufsliste zu erinnern.

So weit zumindest die Theorie. Bitte probieren Sie das besser nicht gerade am Abend vor einem Feiertag aus …

Wenn Sie genau darüber nachdenken, kennen Sie Hashing auch sicherlich aus Ihrem Alltag – hier schließe ich mal von mir auf andere: Wie oft gibt es die Situation, in der man irgendein Objekt wegsortiert, ohne darüber nachzudenken. Auf diese Weise landet die Einkaufsquittung im Bücherregal oder der Lieblingskugelschreiber in der Werkzeugkiste. Überraschenderweise finde ich diese Gegenstände (meistens) sehr leicht wieder. Indem ich in Gedanken so tue, als würde ich den Kugelschreiber erneut weglegen, komme ich erneut auf die gleiche Idee „Werkzeugkiste" und werde dort fündig!

Ungleich schwieriger wird die Sache manchmal, wenn ich beim Aufräumen eine ganz bewusste Entscheidung getroffen habe: Den Pass für die Reise nächste Woche habe ich an einen Ort gelegt, wo ich ihn garantiert sicher wiederfinde – nur fällt mir zwei Tage später absolut nicht mehr ein, wo dieser Ort ist. Natürlich weiß ich noch ganz genau, wo ich den Pass gefunden hatte, bevor ich ihn am vermeintlich sicheren Ort deponierte ... hätte ich ihn nur dort gelassen!

Kommt Ihnen das irgendwie bekannt vor?

Genau dieses „intuitive" Prinzip wird beim Hashing genutzt. Aufgrund einer Formel wird der Speicherort für einen Datensatz festgelegt, und wenn wir diesen später wie-

der suchen, finden wir ihn aufgrund der gleichen Formel wieder. Bevor ich nun genauer darauf eingehe, wie das Verfahren günstig umgesetzt werden kann, möchte ich eine andere Frage aufwerfen: Wo ist der Haken? Warum benutzt man überhaupt noch eine konventionelle Sortierung?

Das können wir experimentell beantworten: Nehmen Sie 16 beliebige Sortierkarten mit Buchstaben, mischen diese und legen dann eine nach der anderen auf einem großen Schreibtisch ab. Schauen Sie sich dabei jeweils den Buchstaben an und legen die Karte verdeckt so ab, dass Sie sie wahrscheinlich wiederfinden.

Suchen Sie nun die Karte mit dem „F“ (oder eine andere aus Ihrem Satz)! Wahrscheinlich finden Sie diese recht schnell.

Jetzt machen Sie das Gleiche noch einmal, allerdings diesmal mit einer sehr kleinen Fläche: Die Karten müssen alle auf einer Postkarte Platz haben. Versuchen Sie nun die Karte mit dem „L“ zu finden! Ist das ebenfalls so einfach? Wahrscheinlich sind Sie bei diesem Experiment eher geneigt, die Karten beim Hinlegen zu sortieren.

Führen Sie den Versuch mit verschiedenen Testpersonen durch. Wahrscheinlich haben die meisten kein Problem, eine Karte auf der großen Fläche wiederzufinden, brauchen allerdings mehrere Versuche, um dies auf der kleinen Fläche zu schaffen. Genauso fällt es uns deutlich leichter, Gegenstände in einem großen Raum einfach „intuitiv“ zu platzieren und dann wiederzufinden als in einem kleinen.

Hashing benötigt Platz! Das ist im Computer nicht anders als beim realen Versuch mit den Karten. Daher benutzt man das Verfahren immer dann, wenn der schnelle Zugriff auf die Daten wichtiger ist als geringer Speicherverbrauch. Das wird im Folgenden noch klarer, wenn wir das eigentliche Verfahren besprochen haben!

Die nächste Frage ist: Warum kann das Kriterium für Hashing nicht so gewählt werden, dass letztlich auch eine sortierte Folge herauskommt? Die Antwort haben wir im Wesentlichen im Kapitel über das Sortieren bereits unter dem Stichwort „Proxmap-Sort“ gegeben: Leider haben Zahlen und andere Schlüsselbegriffe, die von Menschen erzeugt wurden, die Tendenz, sich in kleinen Bereichen zu häufen.

Wenn man zum Beispiel die literarischen Neuerscheinungen des Jahres 2017 nach ihren Standard-Buchnummern (ISBN) sortiert, werden wahrscheinlich viele kleine Zahlenbereiche dicht gefüllt sein und große Bereiche fehlen (Nummern, die Verlage bereits 2016 und früher vergeben haben, sowie solche, die noch zu vergeben sind). Das liegt daran, dass die meisten Verlage ihre Buchnummern aufsteigend lückenlos vergeben.

Ein anderes Beispiel ist die Datei einer Krankenkasse, in der die Mitglieder nach Geburtsdatum angeordnet sind – auch hier werden bestimmte Jahreszahlen besonders häufig vorkommen, weil es einfach mehr Menschen mit einem Alter von 45 gibt als solche mit 100.

Betrachten Sie nochmals Abbildung 11.5 und versuchen Sie die gegebenen Zahlen mit dem Kriterium „Zeilennummer = erste Stelle (Hunderttausender)“ korrekt einzuordnen. Das funktioniert mit den kleineren fünf Zahlen tatsächlich. Die nebenstehende Tabelle zeigt das Ergebnis. Allerdings müssten die nächsten fünf Zahlen alle in die bereits besetzte Zeile 9! Wie das aber mit Speicherstellen so ist: Sie bieten nur Platz für einen einzigen Wert.

Beim Proxmap-Sort hatten wir uns durch dynamische Anpassung der Skala für die Tabelle beholfen: In Fall des Beispiels würden nur wenige Zeilen für die Zahlen zwi-

#	Zahl
0	727
1	
2	213121
3	
4	
5	545955
6	
7	
8	832213
9	903512

#	Zahl
00 – 20	727
21 – 41	213121
42 – 62	545955
63 – 83	832213
84 – 90	903512
91 – 93	932018
94 – 95	945630
960 – 969	961347
970 – 975	973412
976 – 999	979834

schen 0 und 900000 reserviert und mehr Zeilen für Zahlen zwischen 900000 und 999999, so wie in der nächsten Tabelle. Um diese Einteilung zu finden, muss man jedoch erst eine statistische Analyse der einzusortierenden Zahlen durchführen. Das ist manchmal sehr aufwendig, oft sogar unmöglich, weil man zu Beginn noch gar nicht weiß, mit welchen Zahlen man rechnen muss.

Effektiv wird mit dieser veränderten Einteilung die Gleichverteilung über dem Speicherplatz bewirkt: Wenn man eine Anzahl von Zahlen einsortiert, werden diese ungefähr immer gleichmäßig aufgeteilt. Auf diese Weise ist es am wenigsten wahrscheinlich, dass zwei Zahlen die gleiche Speicherstelle belegen.

Wie an den hinteren Spalten von Abbildung 11.5 zu sehen war, können wir diese Gleichverteilung jedoch auch anders herstellen: indem wir das Sortierkriterium verändern. Wie, das wollen wir jetzt im Experiment herausfinden.

Sie benötigen die Sortierkarten mit den roten Zahlen und den Papierspeicher unten (für die Karten aus der Kopiervorlage im Buch) bzw. ganz rechts (für die Karten aus dem Bastelbogen) mit Speicherstellen #0 bis #9. Mischen Sie die Karten und nehmen Sie dann immer zehn Karten, um sie auf zehn Speicherpositionen zu verteilen. Dabei nutzen Sie bitte zunächst die Hunderttausender-Stelle der roten Zahlen als Kriterium. Die Zahl 544520 kommt zum Beispiel auf die Speicherstelle #5 und die Zahl 66.580 auf die Speicherstelle #0 (da die Zahl kleiner als 100000 ist und damit an der entsprechenden Stelle quasi eine 0 hat).

Führen Sie diesen Versuch 10- bis 20-mal durch und schreiben jeweils auf, wie viele Karten nicht mehr auf eine Speicherstelle gepasst haben, weil dort bereits eine Karte lag. So etwas nennt man übrigens „Kollision“, weil die gewünschte Einordnung einer Karte mit der dort bereits liegenden Karte „kollidiert“.

Wussten Sie übrigens?
Wenn 30 Menschen zufällig zusammenkommen (zum Beispiel in einer Schulklasse), dann ist es sehr wahrscheinlich, dass zwei oder mehr davon am gleichen Tag Geburtstag feiern. Genau genommen liegt die Wahrscheinlichkeit ungefähr bei 70 %. Bei 40 Personen sind das bereits ca. 90 %. Sie sehen an diesem Beispiel, dass es fast unmöglich ist, Kollisionen ganz zu vermeiden.

Bestimmen Sie die mittlere Anzahl der Kollisionen, indem Sie die Summe durch die Anzahl der Versuche teilen. Zum Beispiel bin ich bei 20 Versuchen auf 97 Kollisionen gekommen, was durchschnittlich 4,85 Kollisionen entspricht.

Jetzt machen Sie den gleichen Versuch nochmals – allerdings nehmen Sie nun die letzte Stelle, also die 1er-Stelle, um die Karten auf die Speicherstellen zu sortieren. Die Zahlen 544520 und 66580 kämen nun also beide auf die Speicherstelle 0 – was bereits zu einer Kollision führte, wenn wir diese Karten in einem gemischten Paket fänden.

Bestimmen Sie wieder die durchschnittliche Anzahl der Kollisionen. Ist dieses Ergebnis besser oder schlechter als das Ergebnis mit den Hunderter-Stellen?

Natürlich ist das mit statistischen Versuchen immer so eine Sache, aber wahrscheinlich haben Sie ein ähnliches Ergebnis wie ich erzielt: Bei 20 Versuchen waren das 75 Kollisionen, also durchschnittlich 3,75. Man kann demnach unter sonst gleichen Voraussetzungen mit diesem Kriterium ungefähr eine Karte mehr speichern.

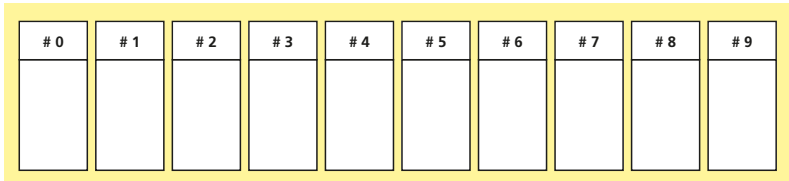


Abbildung 11.6
Hashfeld für Sortierkarten aus der Kopiervorlage im Buch

Offenbar geht es beim Hashing also hauptsächlich darum, ein gutes Sortierkriterium zu finden. In diesem Fall scheint zum Beispiel das Kriterium „1er-Stelle“ besser geeignet zu sein als „100.000er-Stelle“, weil es eine gleichmäßigere Verteilung der Karten auf die Speicherstellen ergibt.

Trotzdem sollte man selbstverständlich eine Karte auch einordnen können, wenn ihr eigentlicher Platz bereits belegt ist. Denken wir wieder an das Zimmer-Beispiel zurück: Wenn Sie einen Gegenstand in eine Schublade legen möchten, diese ist aber bereits bis zum Rand gefüllt, fällt Ihnen sicherlich noch ein „zweitbester“ Ort ein, etwa die nächste Schublade darunter. Wenn Sie beim Suchen nach dem Gegenstand dann am ersten Platz keinen Erfolg haben, ziehen Sie wahrscheinlich auch ganz intuitiv die nächste Schublade auf, um den Gegenstand dann zu finden.

Beim Hashing nennt man dies „Kollisionsbehandlung“. Eine Kollisionsbehandlung wird durchgeführt, sobald ein Datensatz nicht an seiner eigentlichen Stelle in den Speicher geschrieben werden kann, weil diese bereits belegt ist. Die einfachste Strategie dazu erinnert an unser Vorgehen beim Aufräumen im Zimmer: „Nimm einfach die nächste Schublade“, oder im Fall des Computers: „Nimm einfach die nächste freie Speicherstelle.“

Versuchen Sie nun einfach einmal einen neuen Satz aus zehn gemischten Karten auf die zehn Speicherstellen zu hashen. Bitte beachten Sie, dass man wieder von vorne anfängt, wenn man beim Suchen der nächsten freien Speicherstelle am Ende angekommen ist!

Nehmen wir an, Sie hashen die folgenden Zahlen nach der „Einerstelle“ in der Reihenfolge:

124890, 425565, 582278, 444082, 471452, 544520, 562184, 409121, 677562, 276637

Die Hashtabelle ist danach folgendermaßen gefüllt:

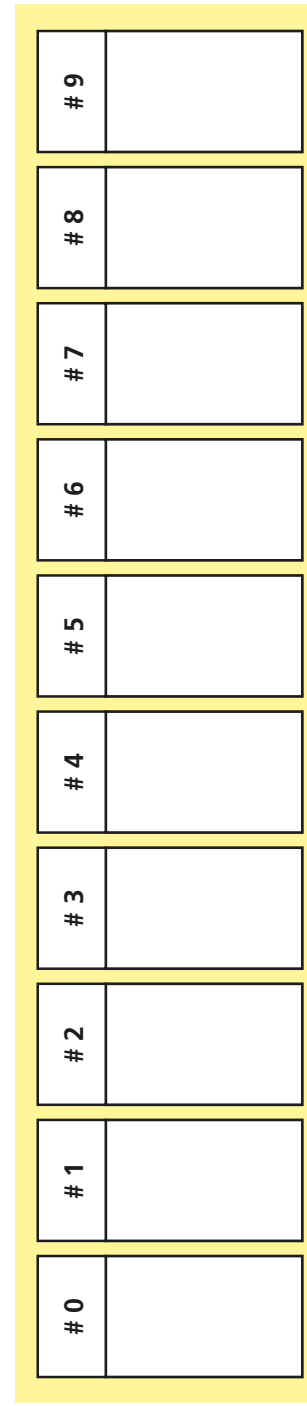
#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
124890	544520	444082	471452	562184	425565	409121	677562	582278	276637

Beim Suchen nach einer Karte darf man sich nun natürlich ebenfalls nicht darauf beschränken, an der ihr zugeteilten Speicherstelle zu schauen. Falls sich dort eine andere Karte befindet, muss man auch die nächsten Speicherpositionen durchsuchen.

Suchen wir in der Tabelle oben beispielsweise die Karte mit der roten Zahl 409121. Eigentlich würden wir sie auf Position #1 vermuten, da die letzte Stelle 1 ist. Dort befindet sich allerdings eine andere Karte. Also versuchen wir es eine Position weiter. Auch dort werden wir nicht fündig. Insgesamt müssen wir sechs Karten anschauen, bis wir die gesuchte Zahl erreichen! Mit der binären Suche in einer sortierten Folge hätten wir maximal vier Vergleiche benötigt, das Hashing ist also sogar ungünstiger!

Weiter oben hatte ich bereits beschrieben, dass das intuitive Aufräumen in einem sehr kleinen Zimmer nicht funktioniert. Hier können Sie nun das Pendant beim Hashing erkennen: Wenn der Speicherplatz zu dicht belegt ist, nimmt man besser die klassische Sortierung der Daten, zusammen mit der binären Suche. Das innovative Hashing benötigt mehr Platz!

Abbildung 11.7
Hashfeld für Sortierkarten aus dem Bastelbogen



Das impliziert sofort die Frage, wie viel Speicherplatz eigentlich notwendig ist. Als Nächstes wollen wir daher durch Probieren ermitteln, wie gut das Hashing mit verschiedenen Belegungsfaktoren abschneidet.

Bitte mischen Sie den Kartenstapel wieder. Nehmen Sie nun zehn Karten und hashen diese in die zehn Speicherstellen – genau wie oben. Diesmal führen Sie jedoch eine Strichliste: Für jede Speicherposition, die Sie ansprechen müssen (also auch belegte), machen Sie eine Strich. Falls also eine Karte erst in die vierte Speicherstelle abgelegt werden kann, weil die drei ersten Positionen belegt sind, ergibt das vier Striche.

Zur Kontrolle: Im Beispiel oben müssten beim Hashing 24 Striche gemacht werden. Pro Karte waren also durchschnittlich 24 geteilt durch 10, also 2,4 Speicherzugriffe notwendig.

Führen Sie diesen Versuch 10- bis 20-mal durch, um zufällige Ergebnisse auszuschließen. Teilen Sie dann die Anzahl der Striche durch die Anzahl der Versuche und dann nochmals durch 10. Auf diese Weise erhalten Sie die mittlere Anzahl von Speicherzugriffen, die pro Karte benötigt wird.

Das Beispiel oben scheint statistisch bereits ziemlich durchschnittlich zu sein, denn wenn man sehr viele Versuche durchführt, kommt man tatsächlich auf ungefähr 2,4 Speicherzugriffe pro Element.

Führen Sie nun den obigen Versuch mit der Strichliste noch ein paar Mal durch. Sortieren Sie nun aber nur 9, 8, 7, 6, 5, 4 und 3 Karten auf einmal in den Speicher. Natürlich müssen Sie dann die Anzahl von Strichen am Ende auch durch die Anzahl der Versuche und die Anzahl der Karten (also zum Beispiel 9) teilen, um auf die durchschnittliche Zugriffszahl pro Karte zu kommen.

Wahrscheinlich kommen Sie auf ähnliche Ergebnisse wie ich, die in folgender Tabelle zusammengefasst sind:

Anzahl Karten	3	4	5	6	7	8	9	10
Zugriffe pro Karte	1,1	1,2	1,3	1,4	1,6	1,8	2,0	2,4

Sie konnten also nachvollziehen, dass das Hashing immer besser wird, je weniger Karten man in den Speicher einsortiert. Allerdings sollte es einen guten Kompromiss zwischen Speicherausnutzung und Anzahl der Zugriffe geben, zum Beispiel eine Kartenzahl zwischen 7 und 8. Das entspricht einem Belegungsgrad von 70 % bis 80 %. Der Belegungsgrad gibt an, wie viel Speicher prozentual ausgenutzt ist.

Nun wäre ja eventuell möglich, dass unser kleiner Versuch mit zehn Karten nicht authentisch für große Hashtabellen ist. Daher habe ich für Sie das Experiment schon mit deutlich mehr Karten durchgeführt, die in einen Speicher mit 10.000 Stellen einsortiert werden. Das Ergebnis können Sie in folgender Tabelle ablesen.

Anzahl Karten	Zugriffe pro Karte	Anzahl Karten	Zugriffe pro Karte
10.000	63,1	5.000	1,5
9.500	10,1	4.500	1,4
9.000	5,5	4.000	1,3
8.500	3,8	3.500	1,3
8.000	3,0	3.000	1,2
7.500	2,5	2.500	1,2
7.000	2,2	2.000	1,1
6.500	1,9	1.500	1,1
6.000	1,8	1.000	1,1
5.500	1,6	500	1,0

Belegen wir also die 10.000 Speicherstellen vollständig, benötigen wir pro Suche durchschnittlich etwas über 63 Speicherzugriffe. Zum Vergleich: Die binäre Suche würde in einer sortierten Liste mit durchschnittlich 13 Speicherzugriffen auskommen. Allerdings wird das Hashing bereits attraktiv, wenn wir den Speicher zu 95 % füllen: Bei 9.500 Datensätzen benötigt man nur noch gut 10 Speicherzugriffe, was besser als mit der binären Suche ist.

Ein Belegungsgrad von 80 % senkt diese Zahl sogar noch auf durchschnittlich 3. In der Praxis versucht man tatsächlich, den Belegungsgrad beim Hashverfahren unter 85 % zu halten. In Programmen, bei denen es auf noch schnelleren Zugriff ankommt, kann man den Speicherplatz auch noch ausufernder nutzen. So liegt die durchschnittliche Anzahl von Zugriffen nur noch knapp über 1, wenn man nur 20 % des verfügbaren Speichers in Anspruch nimmt.

Was steckt dahinter?

Die wichtigsten Grundsätze des Hashings konnten Sie bereits selbst herausfinden: Es kommt hauptsächlich darauf an, ein Kriterium für die Verteilung der Datensätze auf die Speicherpositionen zu finden, das den Speicherplatz gleichmäßig ausnutzt.

Um hier weiter in die Tiefe zu gehen, benötigen wir noch ein paar Fachausdrücke: Den Wert oder die Zeichenfolge, nach dem bzw. der sortiert wird, nennt man auch „Schlüssel“. Im Beispiel oben war der Schlüssel einer Karte also die rote Zahl.

Aus dem Schlüssel wird dann direkt über eine Berechnung die Speicherposition bestimmt. Diese Berechnung war in unseren bisherigen Versuchen immer recht einfach – meistens wurde schlicht die erste oder letzte Ziffer genommen. Hier könnte jedoch auch eine komplizierte Formel verwendet werden. Die Berechnungsvorschrift nennt man auch „Hashfunktion“.

Wenn wir den Schlüssel mit „*s*“ abkürzen und die Hashfunktion mit „h“, ergibt sich also für die Speicherposition eines Datensatzes bzw. einer Karte:

Position = h (*s*)

Im Beispiel nach Abbildung 11.5 ist die Hashfunktion die letzte Ziffer. Diese ist identisch mit dem Divisionsrest beim Teilen durch 10. Es gilt also:

h (*s*) = *s* mod 10

Die Karte mit der 425565 kommt also auf die Position

h (425565) = 425565 mod 10 = 5

Ebenfalls im Experiment ausprobiert haben wir, dass die Hashfunktion sowohl zum Einordnen der Datensätze bzw. Karten dient als auch dazu, einen bestimmten Datensatz wieder aufzufinden.

Wenn wir also irgendwann zum Beispiel wissen möchten, welches Wort auf der Karte mit der roten Zahl 425565 steht, müssen wir die Karte finden. Dazu nutzen wir wieder die Hashfunktion, kommen auf die Speicherposition 5 und können dort das Wort „absurd" lesen.

Wie ermittelt man nun aber eine sinnvolle Hashfunktion? Ein Kriterium dafür haben wir bisher bereits implizit angewendet. Vom Wertebereich der Hashfunktion ist die Größe des Speichers abhängig, der zur Verfügung gestellt werden muss:

Die Hashfunktion „h (*s*) = letzte Ziffer von *s*" kann Zahlen zwischen 0 und 9 hervorbringen. Daher muss auch ein Speicherbereich mit zehn Positionen (#0 bis #9) reserviert werden. Für die Hashfunktion „h (*s*) = letzte drei Ziffern von *s*" müsste entsprechend ein Speicherbereich von 1000 Positionen zur Verfügung stehen, denn drei Ziffern können Werte zwischen 000 und 999 annehmen.

Natürlich müssen Sie dafür sorgen, dass der Werte- und damit der Speicherbereich groß genug ist, um alle Schlüssel aufzunehmen und zusätzlich großzügig Platz zu lassen: Nur bei einem Belegungsfaktor unter 0,8 ist das Hashing effektiv.

Wie erreicht man darüber hinaus, dass die Zahlen über den Speicherstellen möglichst gleich verteilt sind? Natürlich ist das von den jeweiligen Daten abhängig, die man erwartet.

Die folgende Tabelle enthält drei unterschiedliche Sätze von Beispielschlüsseln. Versuchen Sie für jeden der Sätze eine günstige Hashfunktion für eine Speichergröße von 100 Plätzen zu bestimmen. Kleiner Tipp: Wenn Sie darüber nachdenken, wofür die Zahlen stehen könnten, wird die Auswahl leichter.

Satz 1	Satz 2	Satz 3
19031980	3398776	8,99
21121979	3398777	1,98
09081981	3398778	5,89
06061981	3398779	7,98
14011980	3398780	4,00
29081981	6981154	9,79
11031981	6981155	2,19
17061980	6981156	4,98

Fangen wir einmal mit Satz 2 an: Hierbei könnte es sich zum Beispiel um die Buchnummern von Neuveröffentlichungen eines Verlages handeln. Die einführenden Ziffern stehen dabei eventuell für das Fachgebiet und danach werden die Zahlen fortlaufend vergeben. Es scheint vernünftig, die letzten beiden Ziffern als Hashfunktion zu nutzen. Falls dann zufällig zwei Buchreihen die gleichen Endziffern haben, kommt es allerdings zu recht vielen Kollisionen. Dagegen lässt sich aber auch nicht sonderlich viel machen, selbst mit komplizierteren Hashfunktionen nicht.

Satz 1 enthält Zahlen, die alle als Datum interpretiert werden könnten und so zum Beispiel die Geburtsdaten eines Abiturjahrgangs repräsentieren. Man sieht bereits, dass es ziemlich schwierig ist, einfach zwei Ziffern herauszuschneiden, die dann die Speicherstelle angeben: Die unterschiedlichen Jahreszahlen beschränken sich auf drei (wie das in einer Schulklasse meistens der Fall ist). Bei den Monaten hat man nur Zahlen zwischen 01 und 12, die Tage liegen zwischen 01 und 31. Würden wir aufgrund dieser Ziffern hashen, hätten wir nicht den gesamten Speicher ausgenutzt.

Am ehesten käme man noch auf eine vernünftige Verteilung, wenn man die zweite und die vierte Ziffer zu einer neuen Zahl zusammensetzt, also die 1er-Stelle des Tags und des Monats. Allerdings ist dies auch keine besonders gute Lösung: Die Ziffern 1 und 2 kommen bei der 1er-Stelle des Monats doppelt so häufig vor wie alle anderen Ziffern – sie sind im Januar und Februar, aber auch im November und Dezember enthalten.

Die gleiche Problematik zeigt sich auch bei Satz 3. Die Zahlen könnten zum Beispiel die verschiedenen möglichen Preise eines Einzelhandels darstellen. Da die Cent-Beträge meistens auf 9 oder 8 enden, kann man hier ebenfalls schlecht eine Verteilung bilden.

Allgemein gibt es offenbar ziemlich viele Fälle, in denen die Sortierschlüssel sehr stark von unserem Denken im Dezimalsystem geprägt sind: Bestimmte Ziffern stehen für bestimmte Funktionen (Jahreszahl, Mitgliedsnummer oder Ähnliches). Daher kommt es sehr oft zu Häufungen bestimmter Werte.

Günstig wäre, mit der Hashfunktion gegen das Dezimalsystem zu steuern und damit gegen unsere menschlichen Gewohnheiten. Das Gleiche gilt für das binäre System, da viele Größen im Computer hiervon sehr abhängig sind. Betrachten wir die Divisionsrestmethode: Verwendet man den Rest bei der Division durch 10, schneidet man quasi nur die letzte Stelle des Dezimalsystems ab, bei der Division durch 100 die letzten beiden Stellen usw. Genau das Gleiche gilt etwa für das binäre Zahlensystem, wenn Sie durch 2 teilen. Um bei der Divisionsrestmethode unabhängig von jeglichen Zahlensystemen zu werden, sollten Sie daher durch eine Primzahl teilen. Probieren Sie das doch gleich einmal mit den Daten von Satz 1 und Satz 3 aus (das feste Komma der Zahlen in Satz 3 können Sie dabei einfach ignorieren).

Ordnen Sie die Zahlen nach den folgenden Hashfunktionen in den Speicher:

$h(s) = s \bmod 11$

$h(s) = s \bmod 103$

s	*s* mod 11	*s* mod 103
19031980	0	52
21121979	10	78
09081981	7	59
06061981	2	19
14011980	2	44
29081981	5	34
11031981	4	63
17061980	1	30
899	8	75
198	0	95
589	6	74
798	6	77
400	4	91
979	0	52
219	10	13
498	3	86

Die Primzahlen sorgen hier für eine Entkopplung der Speicherpositionen von ihren dezimalen Abhängigkeiten. Selbst sehr ähnliche Schlüssel können komplett unterschiedliche Hashwerte aufweisen. Selbstverständlich wird automatisch die Größe des verwendeten Speichers durch die Primzahl des Divisors bestimmt.

Man tut durch das beschriebene Verfahren das Bestmögliche, um Häufungen zu vermeiden, ausschließen kann man sie jedoch keinesfalls. So wird ein guter Hash nicht nur durch seine Hashfunktion, sondern auch durch die Kollisionsbehandlung bestimmt. Im Beispiel mit dem Papierspeicher bestand diese einfach in der Regel: „Gehe so lange eine Position weiter, bis ein Speicherplatz frei wird." Um das formal auszudrücken, erweitern wir die Hashfunktion. Diese heißt nun nicht mehr h (s), sondern h0 (s). Die 0 steht dabei für den 0. Versuch, den Schlüssel im Speicher unterzubringen. (Sie erinnern sich: Informatiker fangen gerne bei 0 an zu zählen ...)

Die Funktion zur Kollisionsbehandlung wird dann mit h1 (*s*), h2 (*s*), h3 (*s*) usw. bezeichnet, wobei 1, 2, 3 für den 1., 2. und 3. Versuch stehen. Meistens möchte man eine einzige Funktion zur Kollisionsbehandlung haben und verwendet eine weitere Variable für den Versuch: h_i (*s*) steht für den *i*-ten Versuch. Auf diese Weise kann nacheinander h_0, h_1, h_2 usw. durchgeführt werden, bis man eine freie Stelle in der Hashtabelle findet.

Nehmen wir etwa die oben bereits benutzte Hashfunktion (schon in neuer Schreibweise):

$h_0(s) = s \bmod 103$

Eine Kollisionsbehandlung, die einfach immer einen Platz weiter sucht, wäre dann:

$h_i(s) = (h_0(s) + i) \bmod 103$

Zu lesen einfach: „Für den *i*-ten Versuch, *s* unterzubringen, gehe vom Speicherplatz, den die Hashfunktion ausgesucht hat, *i* Plätze nach rechts. Fange nach dem Erreichen des Endes wieder von vorne an."

Die Suche nach einer vernünftigen Kollisionsbehandlung ist noch schwerer als die Suche nach einer guten Hashfunktion. Die höchste Anforderung ist dabei, dass alle Plätze des Speichers irgendwann „gefunden", also adressiert werden. Ansonsten könnte es vorkommen, dass erst die Hälfte des Speichers gefüllt ist und ein Schlüssel trotzdem nicht eingeordnet wird, weil die schlechte Kollisionsbehandlung genau die freien Stellen gar nicht adressiert.

Mit der einfachen Kollisionsbehandlung „Gehe immer eine feste Anzahl von Speicherpositionen nach links oder rechts" erfüllt man diese Bedingung im Regelfall. Leider kann das aber zu anderen Problemen führen: Kommt es doch einmal zu Häufungen bei der Hashfunktion, werden die Kollisionen auch immer wieder auf den gleichen Speicherstellen stattfinden. Daher wäre es besser, auch die Kollisionsbehandlung vom Schlüssel abhängig zu machen. Hier ist es jedoch äußerst schwierig, eine Funktion zu finden, die zusätzlich alle Speicherplätze erreicht.

In der Praxis verwendet man tatsächlich fast ausschließlich sehr einfache Hashfunktionen nach der Divisionsrestmethode mit gleichfalls einfachen Kollisionsbehandlungen. Im Experiment haben wir ja auch gesehen, dass diese im Allgemeinen sehr gut funktionieren.

Ordnung im Chaos!

Um auf die Frage vom Anfang dieses Kapitels zurückzukommen: Was ist denn nun besser – Ordnung oder Chaos? Ich denke, dieses Kapitel hat gezeigt, dass Ordnung ein sehr gutes Mittel ist, effektiv und schnell auf Daten zuzugreifen und sie damit auch effektiv und schnell verarbeiten zu können.

Allerdings hat es ebenfalls gezeigt, dass so manche Anordnung, die auf den ersten Blick wie ein Chaos aussieht, letztlich sehr geordnet sein kann – dem Betrachter fehlt lediglich der Zugang in Form einer mehr oder weniger komplexen Hashfunktion. Das gilt meiner Meinung nach für Daten im Computerspeicher ebenso wie für den Büroschreibtisch: Oft ist der Besitzer eines scheinbar chaotischen Papiergrabs in der Lage, ein gewünschtes Dokument in Sekundenbruchteilen zwischen den Schichten hervorzuzaubern – das Hashing funktioniert.

Schwierig wird es mit diesem Verfahren, wenn viele Menschen zusammen arbeiten und auf den gleichen Datenbestand zugreifen. Meistens gelingt es in der wirklichen Welt nicht, die Hashfunktion für einen Schreibtisch auch anderen zu vermitteln – hier hilft nur traditionelles Sortieren und Ablegen. Im Computer ist das dagegen kein Problem: Es können auch mehrere Prozesse auf einen gehashten Speicherinhalt zugreifen – sie müssen lediglich die gleiche Hashfunktion benutzen.

Die Frage ist also weniger, ob Ordnung oder Chaos besser ist, sondern für jede Anwendung – im Computer wie im Alltag – muss betrachtet werden, welche Art der Ordnung sinnvoll und nützlich ist. Manchmal wirkt es dabei ordentlich, manchmal eher unordentlich – aber eigentlich kommt man immer auf eine Ordnung!

Bleibt noch die Frage, welche Anwendungen es für Hashing im Computer gibt. Im Prinzip ist es überall gut einzusetzen, wo man Daten sehr schnell ablegen und wieder

hervorholen muss. Das ist etwa bei Datenbanken der Fall – denken Sie an elektronische Wörterbücher, bei denen ein bestimmtes Wort sehr schnell gefunden werden soll. Hashing wird auch in Betriebssystemen gebraucht, um den verfügbaren Hauptspeicher zu verwalten und einem Programm, das Speicher anfordert, schnell freien Platz zuzuweisen.

Auch in Echtzeitsystemen können bestimmte Formen des Hashings verwendet werden. Echtzeitsysteme sind Anwendungen, bei denen es darauf ankommt, in einer bestimmen, definierten Zeit eine Entscheidung zu treffen. Beispiel dafür ist ein Fließband, auf dem Produkte zur Qualitätskontrolle an Sensoren vorbeilaufen. Innerhalb von Sekundenbruchteilen muss ein Computer anhand der Messdaten entscheiden, ob er das Produkt mit einem Luftstrahl vom Band befördert oder nicht. Hier können Hashverfahren helfen, die darauf getrimmt sind, eine minimale Zugriffszeit zu garantieren, indem ihr Algorithmus dafür sorgt, dass es nicht mehr als zum Beispiel zwei Kollisionen hintereinander gibt.

Zufällig

Am Ende dieses Kapitels möchte ich noch kurz auf ein Thema eingehen, das eng mit dem Hashing verwandt ist, auch wenn dies auf den ersten Blick nicht so scheint: die Erzeugung von Zufallszahlen.

Zufällige Ereignisse im Computer assoziieren wir normalerweise eher mit Computerspielen, in denen der Benutzer immer wieder mit für ihn nicht vorhersehbaren Ereignissen konfrontiert wird. Eine andere Zufallskomponente ist eher unerwünschter Natur: die Zeit, die zwischen Systemabstürzen oder Programmfehlern vergeht (trotzdem werden in England Wetten darauf abgeschlossen).

Dass man Zufallszahlen auch nutzen kann, um ganz konkrete Berechnungen anzustellen, soll das folgende Experiment zeigen. Die berühmte Zahl π kann man mit komplizierten mathematischen Reihenentwicklungen bestimmen oder man kann sie anhand ihrer geometrischen Definition ermitteln.

Abbildung 11.8 zeigt einen Kreis mit dem Radius 1. Dieser hat laut Definition einen Flächeninhalt von π, in jedem Quadranten also π/4. Ein solcher Quadrant ist nochmals sehr groß in Abbildung 11.9 dargestellt.

Dieser ist in kleine Quadrate eingeteilt, mit deren Hilfe wir einen Schätzwert für die belegte Fläche auszählen können. Der Flächeninhalt F entspricht näherungsweise der Anzahl der Kästchen mit Mittelpunkt im blauen Bereich, geteilt durch die gesamte Anzahl Kästchen.

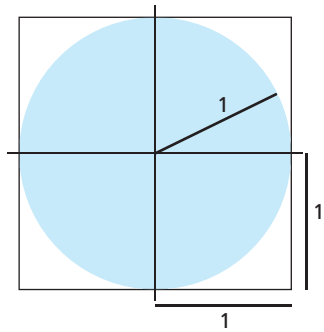


Abbildung 11.8
Der Kreis mit Radius 1 hat den Flächeninhalt π.

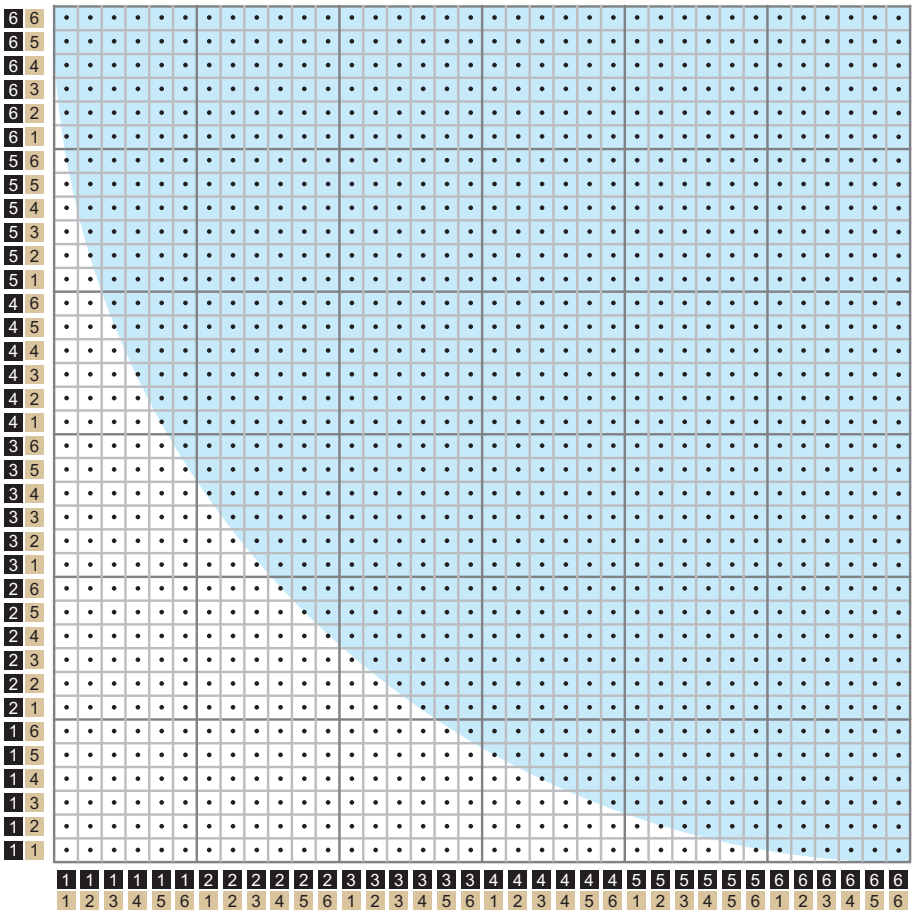


Abbildung 11.9
Ein Viertel des Kreises mit Würfelwerten in Zeilen und Spalten

Wenn wir alle Kästchen durchgehen, das sind genau $36^2 = 1296$ Stück, kommen wir auf 1018 mit Mittelpunkt im blauen Bereich (bei einigen muss man sehr genau hinschauen, wo sich der Mittelpunkt befindet). Unsere Berechnung für π ergibt also:

$$\pi = F \cdot 4 \approx \frac{1018}{1296} \cdot 4 \approx 3{,}1420$$

Das ist bereits ein ziemlich guter Näherungswert für die Zahl π – das „echte" π ist auf fünf Stellen genau 3,1416. Um diesen guten Wert zu erhalten, mussten jedoch 1296 Kästchen ausgezählt werden.

Mit deutlich weniger Zählen kommen Sie aus, wenn Sie nicht jedes Kästchen betrachten, sondern nur solche, die Sie per Würfel zufällig bestimmt haben. Hierzu benötigen Sie zwei unterschiedliche, sechsseitige Würfel, zum Beispiel einen schwarzen und einen beigen. Um ein Kästchen zu bestimmen, würfeln Sie zunächst für die Zeile und dann für die Spalte. Falls der Mittelpunkt im blauen Bereich liegt, vermerken Sie eine 1, sonst eine 0. Wenn Sie 50 Kästchen ausgezählt haben, ziehen Sie eine Zwischenbilanz.

Setzen Sie die Zahl der Einsen und die Gesamtzahl der Würfe in die Formel oben ein. Haben Sie zum Beispiel 40 Einsen erwürfelt, ergibt das für die Fläche einen Näherungswert von $\pi = F \cdot 4 \approx \frac{40}{50} \cdot 4 \approx 3{,}2$ Das ist für π auch schon recht ordentlich!

Jetzt wenden Sie sicherlich ein, dass es deutlich mehr Aufwand bedeutet, 50 Kästchen auszuwürfeln, als schnell alle Kästchen durchzuzählen. Nicht so für einen Computer – der bestimmt die nötigen Zufallszahlen im Handumdrehen. Tatsächlich wirkt sich das besonders bei Anwendungen mit sehr vielen Dimensionen aus. Beim Bestimmen des Volumens einer Kugel zeigt sich der Effekt also noch besser als bei der Kreisfläche. Noch prägnanter wird die Sache, wenn wir den Rauminhalt einer sogenannten „Hypersphäre" mit 100 Dimensionen berechnen – hier liefert Würfeln deutlich genauere Ergebnisse als das gleichmäßige Auszählen einer entsprechenden Zahl an Stichproben.

Sie fragen sich wahrscheinlich, wozu man so etwas überhaupt berechnen möchte. Zu Recht! Selbstverständlich ist das – genau wie die Bestimmung von π oben – ein rein akademisches Beispiel. Denken Sie aber etwa an die Entwicklung einer neuen Fahrzeugkarosserie, bei der 100 verschiedene Werte computergestützt fein justiert und optimiert werden müssen. Keine Maschine der Welt ist in der Lage, alle möglichen Kombinationen durchzuprobieren. Durch die Verwendung von Zufallszahlen kann man sich aber schnell dem optimalen Ergebnis annähern.

Voraussetzung für das Gelingen des Experiments oben sind Würfel, die tatsächlich zufällige Ergebnisse produzieren und nicht etwa durch Herstellungsfehler eine Zahl häufiger zeigen als eine andere. Auch sehr gut müssen die „Computer-Würfel" sein, genannt Zufallszahlengeneratoren, besser noch Pseudozufallszahlengeneratoren, denn eines können Computer im Normalfall tatsächlich nicht liefern: Zufallszahlen!

Woran liegt das?

Computer sind Maschinen, deren Hauptzweck darin besteht, Abläufe immer und immer wieder so zu wiederholen, wie sie einmal von den Programmierern vorgegeben wurden. Hier wäre es fatal, wenn der gleiche Algorithmus mit den gleichen Eingabewerten zu unterschiedlichen Zeiten unterschiedliche Ergebnisse hervorbrächte. Computer sind daher in dieser Hinsicht sehr zuverlässig!

Zufall bedarf jedoch eines anderes Konzeptes! Zufall ist sozusagen kultivierte Unzuverlässigkeit, und das widerspricht dem braven Computer. Das haben wir bereits beim Hashen gesehen, wo ja auch kein echtes Chaos zustande kommt, sondern nur eine ganz besondere Ordnung.

Aus diesem Grund simulieren Computer den Zufall nur – sie müssen alle Zufallszahlen berechnen und weil diese dann lediglich zufällig aussehen, nennt man sie Pseudozufallszahlen. Um dieses sehr lange Wort zu vermeiden, werde ich im Folgenden allerdings überwiegend einfach weiter von Zufallszahlen reden.

Im Normalfall werden solche Pseudozufallszahlen rekursiv bestimmt: Man fängt mit einer festen Zahl als „Zufallszahl 0" an und berechnet jeweils die folgende Zufallszahl anhand einer Funktion aus der letzten Zufallszahl. Sehr einfach wäre zum Beispiel eine Funktion wie:

$$Z(i) = (Z(i-1) + 5) \bmod 11$$

Die resultierende Zufallszahlenfolge mit „1" als Zufallszahl 0 wäre dann 1, 6, 0, 5, 10, 4, 9, 3, 8, ...

Auf den ersten Blick sieht das bereits zufällig aus, aber bei genauerem Hinsehen fällt dann der Summand 5 sehr deutlich auf, wenn man aufeinanderfolgende Zahlen betrachtet.

Die folgende Zahlenreihe wurde nach einem sehr einfachen und in den 1980er-Jahren sehr häufig eingesetzten Verfahren berechnet. John von Neumann hat es bereits 1949 vorgestellt. Als Startwert habe ich 42 genommen.

42 - 76 - 77 - 92 - 46 - 11 - 12 - 14 - 19 - 36 - 29 - 84 - 5 - 2

Kommen Sie auf die Berechnungsvorschrift?

Es handelt sich um den sogenannten „Mid-Square-Generator“: Man nehme die Zahl, quadriere sie (multipliziere sie mit sich selbst) und benutze die mittleren Ziffern als neue Zahl.

Beispiel: 42 · 42 = 1764. Die mittleren Ziffern sind 76, das ist die nächste Zufallszahl.

Dieser Generator ist jedoch in der Praxis nicht besonders gut geeignet. Um das experimentell zu belegen, berechnen Sie bitte die entsprechende Zufallszahlenfolge mit 24 als Startwert.

Es ergibt sich eine sehr kurze Folge 24 - 57 - 24 - 57 usw. Prinzipiell kommen also nur zwei Zahlen abwechselnd heraus. Auch mit vielen anderen Startwerten kommt man jeweils auf sehr kurze Folgen.

Für ein computergestütztes Roulette-Spiel wäre dieser Generator daher sicherlich nicht geeignet, denn es würden immer nur einige Zahlen erscheinen und andere gar nicht. Eine wichtige Anforderung an einen Zufallszahlengenerator ist also, dass er irgendwann einmal alle möglichen Zahlen hervorbringt! Auch im Beispiel der π-Berechnung von oben wäre das Ergebnis nicht akkurat, wenn bestimmte Zeilen oder Spalten ganz ausgespart würden, weil die virtuellen Würfel die entsprechenden Zahlen nie zeigten.

Das alles erinnert sehr stark an die Forderungen beim Hashing: Auch dort sollte möglichst der gesamte Speicher ausgenutzt werden, Die entsprechende Hashfunktion, die alle Adressen der Tabelle anspricht, war sehr einfach:

$$h(s) = s \bmod m$$

Diese Funktion als Zufallszahlengenerator würde direkt noch keine gute Folge ergeben, wie wir an folgendem Beispiel sehen:

$$Z(i) = Z(i-1) \bmod 40$$

Hier kommt immer wieder die Zahl heraus, die man als Startwert festlegt – probieren Sie es selbst einmal aus!

Echt zufällig

In einigen Anwendungen reichen Pseudozufallszahlen nicht aus. Denken Sie zum Beispiel an die Liste von Transaktionsnummern, die Sie von Ihrer Bank bekommen. Kommt diese aus einem Pseudozufallszahlengenerator und bekommt jemand die Funktionsweise heraus, kann er aus einer Handvoll Eingaben alle weiteren geheimen Zahlen selbst ausrechnen.
Für solche Anwendungen sollte man daher auf „echte“, also physikalische Zufallszahlengeneratoren zurückgreifen. Die schnellsten nutzen quantenphysikalische Vorgänge aus.

Wir benötigen noch etwas, das diesen Startwert verändert. Daher multiplizieren wir ihn mit einem festen Faktor und addieren dann noch etwas dazu:

$$Z(i) = (Z(i-1) \cdot 21 + 17) \bmod 40$$

Mit 1 als Startwert erhalten wir:

1 - 38 - 15 - 12 - 29 - 26 - 3 - 0 - 17 - 14 - 31 - 28 - 5 - 2 - 19 - 16 - 33 - 30 - 7 - 4 - 21 - 18 - 35 - 32 - 9 - 6 - 23 - 20 - 37 - 34 - 11 - 8 - 25 - 22 - 39 - 36 - 13 - 10 - 27 - 24 - 1

Der Generator liefert also alle 40 Möglichkeiten, bevor er von Neuem mit der gleichen Zahl anfängt. Die resultierende Zahlenfolge sieht außerdem trotz der einfachen Berechnungsvorschrift sehr zufällig aus. Es gibt einfache mathematische Regeln, um bei einem solchen sogenannten „linearen Kongruenzgenerator" die Werte so zu setzen, dass tatsächlich alle möglichen Zahlen durchlaufen werden.

John von Neumann (1903–1957) hat sich als Mathematiker sehr früh mit grundlegenden Fragen der Informatik beschäftigt. Unter anderem gilt er als Schöpfer des hier erwähnten, in der Praxis nicht besonders brauchbaren Mid-Square-Generators. Seine bedeutenden Beiträge zur Informatik sind die Formulierung der von-Neumann-Architektur, die prinzipiell noch heute den prinzipiellen Aufbau aller Computer beschreibt. Damit verbunden ist der Begriff des „von-Neumann-Flaschenhals", der beschreibt, dass alle Komponenten eines Computers Daten austauschen müssen und die dafür genutzte Schnittstelle, der sogenannte „Bus" eine kritische Engstelle – den Flaschenhals – darstellt. Einer der wichtigsten Preise für Errungenschaften in der Informatik ist die John-von-Neumann-Medaille.

Sie benötigen gar keine Zufallszahl zwischen 0 und 39, sondern nur eine zwischen 0 und 9? Gar kein Problem: Nehmen Sie einfach nur die hinterste Ziffer. Auf diese Weise entstehen aus den großen Zufallszahlen, die von den Standardgeneratoren der Computer erzeugt werden, die kleineren Zufallszahlen, die man für eine Anwendung benötigt – etwa für ein Spiel mit einem Würfel. Die meisten der „eingebauten" Generatoren arbeiten mit Zahlen zwischen 0 und 18.446.744.073.709.551.615 (über 18 Trilliarden). Hier dauert es also eine Weile, bis der Zyklus von Neuem beginnt.

Vielleicht fragen Sie nun, wie es kommt, dass im bevorzugten Skatprogramm doch nach jedem Start eine unterschiedliche Hand ausgeteilt wird, das Computer-Schachspiel meistens einen anderen Eröffnungszug vollzieht. Das passt doch nicht zusammen mit dem beschriebenen, vorbestimmten Zufall. Und Sie haben recht! Der Computer verwendet zur ersten Initialisierung seines Zufallszahlengenerators eine „echte" Zufallszahl. Meistens nimmt er hierfür die Zeit, die seit dem Einschalten des Computers vergangen ist, in Millisekunden oder sogar Mikrosekunden. Da diese Zeit maßgeblich nicht von der Technik bestimmt wird, sondern vom Bediener, ist sie mit einer starken Zufallskomponente behaftet – je nachdem, ob Sie sich vor dem Start des Spiels noch am Kopf gekratzt haben und wie lange dies gedauert hat, wird der Zufallszahlengenerator anders initialisiert.

Warum generiert der Computer danach in der Regel nur noch Pseudozufallszahlen, statt echter Zufallszahlen auf Zeit-Basis? Sobald das Spiel gestartet wurde und der Benutzer nicht eingreift, benötigt der Computer normalerweise immer die gleiche Zeitspanne für die gleichen Programmteile – Sie erinnern sich: Computer sind sehr verlässliche Maschinen. Auf diese Weise ist etwa beim Skatprogramm die Zeit zwischen dem Austeilen der ersten und der zweiten Karte sehr genau definiert. Hier liefern daher die Pseudozufallszahlengeneratoren wesentlich bessere und „scheinbar zufälligere" Ergebnisse.

Sie sehen: Selbst bei einem Thema, das eigentlich per Definition chaotisch sein sollte, nämlich Zufallszahlen, bleibt der Computer ordentlich und nachvollziehbar. Das Chaos ist auch hier – genau wie beim Hashing – lediglich ein Pseudo-Chaos...

Viel weiter möchte ich hier in die Thematik auch gar nicht einsteigen. Interessierte können unter den Stichworten „Hashing" und „Pseudozufallszahlen" oder „linearer Kongruenzgenerator" selbst recherchieren.

Resümee

Die Ausgangsfrage für dieses Kapitel „Ordnung oder Chaos“ konnte – zumindest im Sinne der Informatik – ganz eindeutig für die Ordnung beantwortet werden! Allerdings hat der Leser nun auch jedes Recht, das scheinbare Chaos auf seinem Schreibtisch einfach als Ordnung zu deklarieren – genau wie in modernen Computern scheinbar chaotisch angeordnete Daten als Hashtabelle eine der schnellsten Zugriffsmethoden darstellen und genau wie scheinbar chaotische Zufallszahlen eigentlich doch auf ganz ordentliche Weise entstehen ...

Sie konnten auch nachvollziehen, dass Informatik nicht immer nur eine Wissenschaft ist, die ihre Methoden durch konstruktive Herangehensweisen bildet. In einigen Fällen muss auch hier empirisch gearbeitet werden. Im Klartext heißt das Ausprobieren! So bleibt neben allen formalen Kriterien für die Formulierung einer guten Hashfunktion oder eines guten Pseudozufallszahlengenerators als letzte Sicherheit nur das Testen.

reits gebuchte Umsätze

onto

Geschäftsbank Giro minus, 1002499

eitraum ◉ 10 Tage

○ von 01 03 TT MM JJJJ

Umsatzauskunft aktualisieren

Aktueller Kontostand:

Dispolimit:

Summe vorgemerkter Un

reits gebuchte Umsätze

tum	Wertstellung	Art	Buchungshinweis	Betrag
03.	09.03.	Auszahlung	PGA 39824471 KRT0001/10.07 09.03 16.30 002245 53291 SCHILDA HAUPTPOST EC-CARD MIT PIN	-250,0
03.	08.03.	Lastschrift	ZAHLUNGSBELEG 483728882701 RE.KTO.NR. 983664729810 RE.NR.000049931122/27.02. Telefon AG & CO. KG	-24,7
03.	08.03.	Überweisung	4711 / 08150815 Tino Tümmler	-137,1
3.	07.03.	Gutschrift	EBay - Artikel Buch Abenteuer Informatik	38,7
3.	06.03.	Gutschrift	Gehalt März GaZ Kassel	812,2
3.	06.03.	Gutschrift	EBAY PLEXTOR WERNER UND KARIN	18,3

12. Mit Sicherheit

Die Zeiten, in denen man einer lächelnden Dame hinter zentimeterdickem Panzerglas sein Erspartes anvertraute oder etwas Geld für den nächsten Einkauf abholte, sind wohl endgültig vorbei. Der moderne Bankschalter von heute ist der heimische PC, eine Internet-Verbindung erlaubt den schnellen Zugriff auf die eigenen Finanzen.

Was sich jedoch nicht verändert hat, ist das Bestreben einiger nicht so gesetzestreuer Elemente, die Eigentumsverhältnisse zu ihren Gunsten zu manipulieren. Es hat sich inzwischen herumgesprochen, dass der klassische Bankraub mit vorgehaltener Waffe unrentabel ist. Daher versuchen es immer mehr Gauner auf elektronische Art und Weise.

Mit der Wandlung des Internets in das dominante Medium unserer Gesellschaft hat sich aber auch noch eine ganz andere „Währung" herauskristallisiert: Information. Jeder Mausklick, jede elektronische Hinterlassenschaft in Foren, aber auch jede höchst private Terminvereinbarung oder Nachricht trägt zu Ihrem Fingerabdruck im Netz bei. Mit diesem lässt sich dann gezielt Werbung schalten oder in sozialen Netzwerken ein Angebot potentieller Gleichgesinnter machen. Es soll aber auch schon vorgekommen sein, dass deutlich kritischere Einschätzungen wie Zuverlässigkeit, Kreditwürdigkeit oder Gesundheit aufgrund dieses Fingerabdrucks getroffen wurden.

Grund genug, sich darüber Gedanken zu machen, wie dieses wertvolle Gut im Internet geschützt wird und welche Möglichkeiten es gibt, diesen Schutz stärker oder schwächer auszunutzen. In diesem Kapitel können Sie alle Aspekte sogar experimentell erforschen!

Um dieses Kapitel komplett verstehen zu können, wäre es übrigens günstig, vorher das Kapitel „Paketpost" durchzuarbeiten.

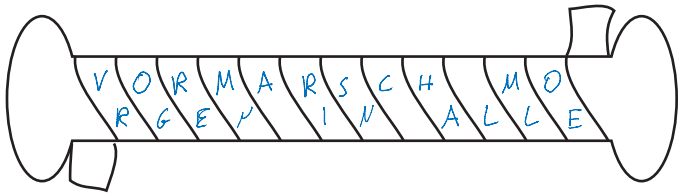
Von Griechen, Julius Cäsar und anderen

Ein Großteil der Internetnutzung bezieht sich auf den Austausch von Mitteilungen und Nachrichten. Wenn Sie in einem sozialen Netzwerk etwas schreiben oder ein Foto einstellen, ist wichtig, dass die Inhalte unversehrt beim vorgesehenen Empfänger ankommen und dass sie auch nirgendwo anders als bei diesem ankommen.

Diese Anforderung ist allerdings nicht neu – bereits einige Jahrhunderte vor unserer Zeitrechnung widmete man sich der Thematik: Besonders auf Feldzügen und bei anderen kriegerischen Auseinandersetzungen war die Gefahr immer groß, dass eine Nachricht mit militärischer Bedeutung vom Gegner abgefangen, verfälscht oder ausspioniert wurde.

Die Spartaner nutzten daher die sogenannten Skytalen, um wichtige Befehle zu verschlüsseln. Diese gab es immer paarweise, ein Exemplar war für jeden Kommunikationspartner bestimmt. Es handelt sich eigentlich nur um einfache runde Holzstäbe mit exakt identischen Durchmessern. Ein dünner Streifen Papyrus oder Pergament

Abbildung 12.1
Schema-Zeichnung einer Skytale und der zugehörige Codestreifen (links)



wird nun spiralförmig um eine Skytale gewickelt und so beschriftet, dass auf jeder Windung nur ein einzelner Buchstabe steht. Abbildung 12.1 zeigt dies – der besseren Verständlichkeit halber mit lateinischen statt griechischen Buchstaben. Am Rand sehen Sie den resultierenden Streifen mit dem Geheimtext. Er ist nur wieder lesbar, wenn man ihn um eine gleichartige Skytale wickelt.

Die Skytale war damit sozusagen eine der ersten Chiffriermaschinen der Welt. Sie erzeugt einen Vertauschungs-Code oder auch eine Transpositions-Chiffre. So werden alle Geheimbotschaften bezeichnet, die genau dieselben Zeichen enthalten wie der Klartext, allerdings in einer veränderten Reihenfolge.

Versuchen Sie den Streifen mit der Geheimbotschaft doch einmal ohne entsprechende Skytale zu lesen, indem Sie einfach immer drei Zeichen überspringen. Achtung! Leerzeichen müssen auch hier berücksichtigt werden!

Transpositions-Chiffren sind in der Regel recht einfach zu knacken und werden heute hauptsächlich noch in Rätseln zur Unterhaltung genutzt, wo dann der Leser zum Beispiel aufgefordert wird, in „ATLAS NACHSEHN“ einen geographischen Begriff zu finden (Lösung: „SACHSEN ANHALT“).

Bereits Julius Cäsar beschreibt in seinem berühmten Werk „Commentarii de bello gallico“ („Anmerkungen zum Gallischen Krieg“) ein prinzipiell anderes Verschlüsselungsverfahren: Jeder Buchstabe des Klartextes wird durch einen anderen Buchstaben ersetzt, der im Alphabet eine entsprechende Zahl von Positionen später kommt. Ist man am Ende des Alphabetes angelangt, fängt man wieder von vorne an.

Es handelt sich um einen Ersetzungscode, auch Substitutions-Chiffre genannt, da die Zeichen gegen andere ausgetauscht werden, die Position im Text jedoch gleich bleibt. Den berühmten Cäsar-Code können Sie sehr einfach verwenden, wenn Sie die Cäsar-Scheiben aus dem Anhang 12.K1 oder dem Bastelbogen aufbauen. Schneiden Sie hierfür die kleinere und die größere Scheibe mit blauen und roten Buchstaben aus. Sie können die Scheiben verbinden, indem Sie in der Mitte entweder mit einer sehr feinen Schere je ein Loch schneiden und einen Verschluss für Versandtaschen durch beide Löcher stecken. Eine sehr gute Methode ist auch, einen Reißnagel genau in die Mitte zu pieksen und diesen hinten mit einer Scheibe zu sichern, die Sie von einem gewöhnlichen Weinkorken abschneiden.

Nun können Sie mit dem grünen Pfeil einen von 26 Schlüsseln auswählen. Für jeden Buchstaben einer Nachricht schreiben Sie den entsprechenden Codebuchstaben hin, der nun in Rot direkt über dem blauen Klartextbuchstaben steht.

Versuchen Sie es: Codieren Sie die Nachricht „ABENTEUER INFORMATIK“ mit der Cäsar-Scheibe und dem Schlüssel „R“.

Die Scheibe wird so eingestellt, dass der grüne Pfeil auf dem roten R steht, wie in Abbildung 12.2 dargestellt. Danach wird Buchstabe für Buchstabe im blauen Bereich gesucht und durch den roten Buchstaben darüber ersetzt. Die Geheimbotschaft lautet also:

„RSVEKVLVI ZEWFIDRKZB“

Sie werden mir zustimmen, dass es schwierig ist, diese Botschaft zu verstehen, wenn man nicht im Besitz des korrekten Schlüssels ist. Wenn Sie sich nun mit einem Freund vertraulich austauschen wollen, vereinbaren Sie einen Schlüssel und verwenden die Cäsar-Scheibe, um die Nachrichten vor unerwünschten Augen und Ohren zu schützen.

Nehmen wir an, Sie haben den Schlüssel „N“ ausgemacht. Was schließen Sie aus der folgenden Nachricht?

„NORAQRFFRA HZ NPUG ORV ZVE“

Selbstverständlich konnten Sie Ihre Scheibe auf den Schlüssel „N“ einstellen. Nun mussten die Buchstaben im roten Ring gesucht und durch den entsprechenden blauen ersetzt werden. Es handelt sich um eine Einladung zum Abendessen.

Viele werden jetzt fragen, ob diese Kommunikation auch wirklich sicher ist. Tatsächlich beschäftigt sich ständig eine große Anzahl von Experten nicht mit der Erschaffung neuer Verschlüsselungsverfahren, sondern damit, die vorhandenen irgendwie zu brechen und auf diese Weise Nachrichten zu lesen, die für jemand anderen bestimmt waren.

Hacker
Personen, die Schutzmaßnahmen gegen das Ausspähen, Kopieren und Manipulieren von Daten in Netzwerken oder auf Datenträgern zu umgehen. Der Begriff kommt aus einer Zeit, in der die ersten Hacker versuchten, Passworte zu knacken, indem sie alle Varianten, die ihnen eingefallen waren, per Tastatur eintippten („hacken“).

Als „Täter“ fallen einem da sofort Geheimdienste, Detekteien, aber auch Betrüger, Industriespione usw. ein. Diese gehören jedoch zur absoluten Minderheit. Die meisten Hacker sitzen in den Universitäten und Entwicklungsabteilungen spezialisierter Firmen. Sie haben ganz sicher Spaß an ihrer Tätigkeit, aber trotzdem keinerlei kriminelle Absichten! Vielmehr dienen ihre Attacken dazu, die Systeme sicherer zu machen und besser vor unberechtigtem Zugriff zu schützen. Die einzige Chance, „echten“ Hackern immer einen Schritt voraus zu sein, besteht darin, die Sicherheitslücken vor ihnen zu finden!

Abbildung 12.2
Cäsar-Scheibe mit gewähltem Schlüssel R

Codebrecher

Jetzt dürfen Sie auch einmal Hacker spielen. Ihre Spitzel fangen folgende verschlüsselte Nachricht ab:

„TXC KPITG WPIIT OLTX HDTWCT SPKDC LPG STG PTAITHIT ZAJV JCS VTHRWTXI JCS LJHHITHXRW XC PAATH LDWA OJ HRWXRZTC STG YJTCVHIT PQTG LPG SJBB ZDCCIT CXRWIH QTVGTXUTC JCS ATGCTC JCS LTCC XWC SXT ATJIT HPWTC HEGPRWTC HXT BXI STB LXGS STG KPITG CDRW HTXCT APHI WPQTC LTCC CJC TILPH OJ IJC LPG HD BJHHIT TH STG PTAITHIT PAAOTXIPJHGXRWITC WXTHH XWC PQTG STG KPITG CDRW HEPTI DSTG VPG XC STG CPRWI TILPH WDATC JCS STG LTV VXCV SPQTX JTQTG STC ZXGRWWDU DSTG HDCHI TXCTC HRWPJGXVTC DGI HD PCILDGITIT TG LDWA PRW CTXC KPITG XRW VTWT CXRWI SPWXC TH VGJHTAI BXG STCC TG UJTGRWITIT HXRW“

Eine Möglichkeit wäre jetzt natürlich, nacheinander alle 26 Schlüssel des Cäsar-Codes auszuprobieren, bis einer die Nachricht decodiert, aber es gibt noch eine gezieltere Methode: Wie Sie im Kapitel über Datenkomprimierung sehen konnten, kommen einzelne Buchstaben in unserer Sprache sehr unterschiedlich häufig vor. Diesen Umstand nutzen Kryptographen auf der ganzen Welt, um den benutzten Schlüssel in einer Geheimbotschaft zu knacken.

Ich habe für Sie daher einen „Apparat“ vorbereitet, mit dem die Entschlüsselung jedes Cäsar-Codes zum Kinderspiel wird.

Schneiden Sie hierzu die großen Scheiben aus Kopiervorlage 12.K2 oder dem Bastelbogen aus und setzen diese zusammen, so wie bereits die kleinen Cäsar-Scheiben. Das Ergebnis sehen Sie zum Vergleich in Abbildung 12.3. Auf die größere Scheibe müssen Sie etwas malen. Wenn Sie möchten, können Sie diese daher mit einer Folie überziehen oder bekleben und dann mit wasserlöslichen Stiften mehrfach benutzen. Man kann auch einen entsprechenden Kreis aus Folie zum Beschriften ausschneiden und diesen zwischen den beiden Scheiben befestigen.

Das Prinzip des Codeknackens besteht darin, die Häufigkeiten einzelner Buchstaben in der Geheimbotschaft zu ermitteln und dann mit den Häufigkeiten zu vergleichen, die die Buchstaben in unserer Sprache normalerweise haben. Zunächst müssen also

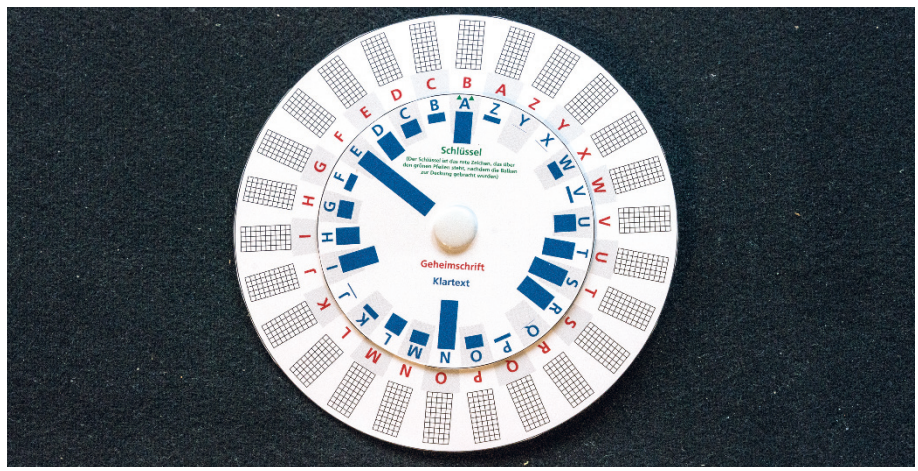


Abbildung 12.3
Codebrecherscheibe

die Häufigkeiten in der Geheimbotschaft bestimmt werden. Dazu sind die Rechtecke ganz außen auf der Scheibe da.

Gehen Sie die Geheimbotschaft oben Buchstabe für Buchstabe durch und füllen für jeden Buchstaben von innen nach außen ein kleines Kästchen aus, und zwar oberhalb des entsprechenden roten Buchstabens auf dem äußeren Ring der Scheibe. Machen Sie dies so lange, bis eines der Rechtecke komplett ausgefüllt ist (50 Kästchen).

Die so entstandenen Balken geben nun die Häufigkeit jedes Buchstabens im Geheimtext an. Sie müssen lediglich noch mit den Häufigkeiten der Buchstaben übereingebracht werden, wie sie tatsächlich in unserer Sprache vorkommen. Diese Häufigkeiten sind bereits fest vorgegeben durch die blauen Balken auf der inneren Scheibe.

Drehen Sie die Scheiben so gegeneinander, dass die Balken innen und außen ungefähr übereinstimmen! Hierfür bitte nur Augenmaß verwenden, kleine Differenzen wird es immer geben!

Die Balken stimmen dann ziemlich gut überein, wenn der grüne Pfeil auf dem „P“ steht, wie Sie auch in Abbildung 12.5 auf der nächsten Seite sehen können. Dies war offensichtlich der verwendete Schlüssel. Wenn Sie nun den Text decodieren, kommt der Anfang eines Grimm'schen Märchens zum Vorschein!

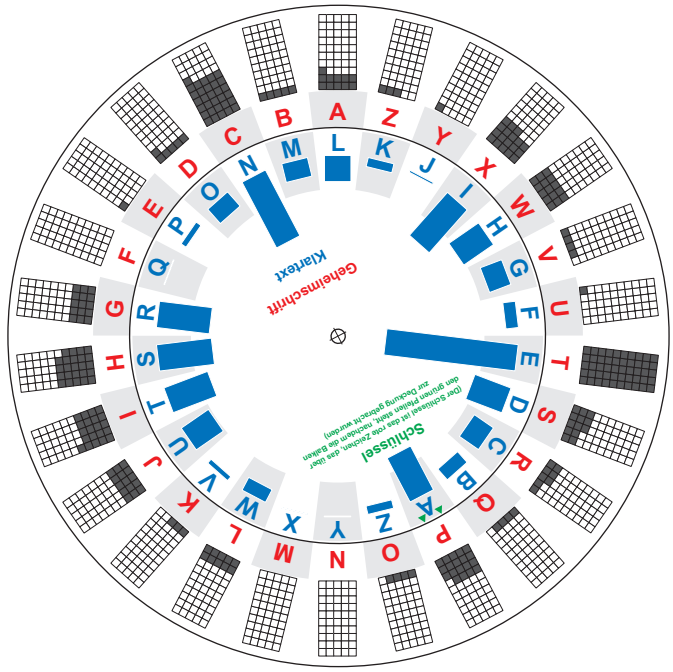
Tatsächlich beruhen sehr viele Verfahren zur Entschlüsselung fremder Geheimbotschaften auf einer statistischen Analyse. Selbstverständlich ist der Cäsar-Code sehr einfach und daher auch sehr einfach zu knacken! Sehr viel schwieriger wird es bereits, wenn die Buchstaben beliebig gegeneinander vertauscht werden dürfen, also ein Schlüssel etwa so wie in Abbildung 12.4 aussieht.

Auch hier ist allerdings meistens das „E“ anhand seiner herausragenden Häufigkeit sofort zu identifizieren. Hierbei kann man sich behelfen, indem man für häufige Buchstaben in Klartext mehrere Codes verwendet. Eine weitere, sehr erfolgreiche Methode ist, die Daten vor der Verschlüsselung mit dem Huffman'schen Verfahren aus dem Kapitel „Datenpresse“ zu komprimieren. Dadurch sind die Häufigkeiten in etwa gleich verteilt und sogar ein recht einfacher Cäsar-Code reicht aus, eine durchaus gute Verschlüsselung zu garantieren.

Abbildung 12.4
Cäsar-Scheibe mit speziellem Code

Kryptographie
Kryptographie ist die Wissenschaft der Datenverschlüsselung. Kryptographen finden einerseits neue Methoden der Codierung, andererseits werden sie auch eingesetzt, um vorhandene Verfahren zu brechen.

Abbildung 12.5
Für den geheimen Text korrekt eingestellte Codebrecher-Scheibe: Nur auf einer Position der großen Cäsar-Scheibe passen die Balken ungefähr zueinander.



Computer sind natürlich in Dingen wie Codeknacken, die oft stures Ausprobieren von verschiedenen Schlüsseln beinhalten, deutlich schneller als ihre menschlichen Pendants und ermüden auch nicht durch die stupide Tätigkeit. Andererseits kann man mit Hilfe von Computern auch deutlich komplexere Verschlüsselungen umsetzen. Stellen Sie sich zum Beispiel eine Cäsar-Codescheibe vor, bei der nicht ein einzelner Buchstabe, sondern immer die Kombination mehrerer Buchstaben einem individuellen Code zugeordnet wird (z. B. „ABC“ → „RDW“, „ABD“ → „FLQ“ usw.). Außerdem können Sie die sehr unterschiedliche Häufigkeitsverteilung minimieren, indem Sie den Text mit einem variablen Code verlustfrei komprimieren – zum Beispiel mit der Methode von Huffman aus Kapitel 6. Wenn zusätzlich die Anzahl von Buchstaben pro Codepaket groß genug gewählt wird, ist selbst ein einfacher Code wie der von Cäsar entwickelte sehr effektiv und sicher.

Es wäre jetzt allerdings müßig, für unsere Experimente auf solche „echt sicheren“ Verschlüsselungen zurückzugreifen! Stellen Sie sich vor, sie müssten eine Cäsar-Scheibe mit 950.163.143.821 Metern Durchmesser basteln – und das wäre die Größe, wenn man Blocks zu nur je zehn Buchstaben verschlüsselte ... Übrigens: 950.163.143.821 Meter entsprechen ungefähr dem 10.000sten Teil eines Lichtjahres, also etwas mehr als dem mittleren Abstand zwischen der Sonne und dem Jupiter.

Für unsere Experimente erlauben wir uns daher, nicht auf Details wie „muss wirklich wasserdicht sein“ zu achten. Vielmehr kommt es darauf an, die tatsächlichen Herausforderungen für eine sichere Verschlüsselung im Internet kennen zu lernen. Diese geht deutlich über das Finden von statistisch nicht knackbaren Verfahren hinaus! Um das jedoch mit einfachen Mitteln nachzuvollziehen, gelte für die folgenden Experimente die nachstehende Aussage als gegeben:

„Die Cäsar-Scheibe ist ein hinreichend sicheres Verschlüsselungsverfahren!“

Internet-Spione

Fassen wir noch einmal zusammen: Wenn Sie mit einem Freund sicher kommunizieren möchten, treffen Sie sich irgendwann einfach persönlich und machen einen Schlüssel aus, zum Beispiel „X“. Auf diesen Schlüssel stellt dann jeder seine Cäsar-Scheibe ein. Wenn Sie sich nun per Post Nachrichten schicken, kann der Bote

- diese Nachrichten nicht lesen,
- diese Nachrichten nicht verfälschen (denn dazu müsste er sie ja erst einmal lesen),
- nicht eine völlig andere Nachricht schicken und so tun, als wäre sie von Ihnen (denn dazu müsste er sie korrekt verschlüsseln, und das kann er ohne Schlüssel nicht).

Ich versichere Ihnen: Mit modernen Verschlüsselungsverfahren unter Zuhilfenahme von Computern kann man dies auch tatsächlich gewährleisten!

Wo liegt nun aber das Problem?

Das Szenario oben trifft schlicht und einfach auf die meisten Kommunikationen im Internet nicht zu! Stellen Sie sich vor, Sie möchten ein Buch beim neuen Online-Buchversender „Aragon“ bestellen. Dieser weiß bisher nichts von Ihnen und Sie wussten bisher nichts von ihm! Jetzt haben Sie allerdings das Buch „Abenteuer Informatik“ in seinem Angebot gesehen und wollen es sofort bestellen. Hierzu müssen Sie Ihre Kreditkartennummer übermitteln, wollen jedoch nicht, dass diese von irgendeinem Schlitzohr ausspioniert wird.

Wenn wir nach dem obigen Schema vorgehen wollten, müssten Sie sich nun erst einmal von Angesicht zu Angesicht mit einem Aragon-Vertreter treffen und einen Schlüssel übergeben. Warum?

Ist doch gar kein Problem: Sie schicken den Schlüssel einfach ebenfalls über das Internet. Diese Nachricht (mit dem Schlüssel) muss allerdings noch ohne Schutz gesendet werden, denn Sie haben ja noch keinen Schlüssel ausgetauscht. Dann kann ein Schlitzohr, das Ihre Leitung angezapft hat, zunächst den Schlüssel aufzeichnen und dann in aller Ruhe die verschlüsselten Nachrichten (mit der Kreditkartennummer) decodieren. In diesem Fall ist die Verschlüsselung vergeblich!

Tatsächlich hat man in frühen E-Banking-Varianten von der entsprechenden Bank zunächst per Einschreiben eine Diskette mit einem Schlüssel geschickt bekommen. Erst nach der Installation dieses Schlüssels konnte dann sicher mit der Bank kommuniziert werden.

Überlegen Sie, ob dieses Verfahren im Fall des Online-Buchkaufs praktikabel wäre: Sicherlich nicht! Einerseits wäre der regelmäßige Postversand von Schlüsseln an jeden Kunden sehr teuer. Noch viel wesentlicher ist jedoch: Wenn Sie im Internet etwas sehen und es kaufen möchten, dann wollen Sie sicherlich nicht erst warten, bis Sie vom Versender ein paar Tage später eine Diskette per Post erhalten, nur um dann die eigentliche Bestellung erst abzuschicken. In dieser Zeit kaufen Sie das Buch sicherlich eher im kleinen Geschäft um die Ecke, der Internet-Versender macht dann keinen Umsatz.

Es sollte also die Möglichkeit geben, von Anfang an mit einem vorher unbekannten Partner sicher zu kommunizieren.

Maria Stuart hatte bereits vorgesorgt, als sie um 1580 in einem englischen Kerker eingesperrt war: Sie hatte mit ihren Verbündeten einen Geheimcode ausgemacht! Allerdings handelte es sich um einen einfachen Substitutionscode, der einfach Buchstaben durch besondere Zeichen ersetzte. Unten sehen Sie den Namen „Stuart“ in dieser Geheimschrift. Die meisten der Botschaften wurden wohl trotz Codierung von ihren Gegnern abgefangen ...

Whitfield Diffie (Jahrgang 1944, Bild oben) und Martin Hellman (Jahrgang 1945, Bild unten) waren die Ersten, die ein asymmetrisches kryptographisches Verfahren veröffentlichten. Noch heute basiert sichere Kommunikation im Internet auf dem Diffie-Hellman-Schlüsseltausch.

Whitfield Diffie, Martin Hellman und Ralph Merkle haben sich Anfang der 1970er-Jahre mit dieser Problematik beschäftigt. Sie erkannten, dass das Hauptproblem darin besteht, dass ein und derselbe Schlüssel gleichzeitig für die Ver- und die Entschlüsselung verwendet wird.

Beispiel: Wenn Sie mit der Cäsar-Scheibe eine Nachricht schicken und dazu den Schlüssel „G“ verwenden, so stellt auch der Empfänger seine Scheibe auf „G“ ein.

Ein solches Verfahren nennt man auch „symmetrische Verschlüsselung“, denn wenn jemand den einen existierenden Schlüssel kennt, kann er damit sowohl Nachrichten codieren als auch decodieren.

Im Gegensatz dazu steht die „asymmetrische Verschlüsselung“, die zwei unabhängige Schlüssel wechselweise benutzt. Das wollen wir in einem Experiment selbst erforschen. Hierzu wird eine weitere kleine Bastelei notwendig: Trennen Sie die beiden Scheiben des ASYM-Codierers aus Kopiervorlage 12.K3 und 12.K4 oder dem Bastelbogen heraus. Danach trennen Sie noch das schmale Sichtfenster und die mit „Schlüssel“ beschriftete Aussparung aus dem oberen Teil, das die Aufschrift „ASYM-Codierscheibe“ trägt. Nun können Sie beide Scheiben genau wie die Cäsar-Scheiben zum Beispiel mit einem Reisnagel und einer Korkscheibe drehbar aufeinander befestigen, so wie in es Abbildung 12.6 zu sehen ist. Wenn Sie Schwierigkeiten mit dem Sichtfenster haben, können Sie dieses auch einfach komplett mit der Schere vom Rand der Scheibe herausschneiden. Darunter leidet etwas die Optik, aber nicht die Funktionalität.

Mit der ASYM-Codierscheibe können Sie Nachrichten ver- und entschlüsseln, die aus den Buchstaben „A“ bis „Z“ sowie den Zeichen „-“ und „.“ bestehen. Der Schlüssel besteht aus einem Buchstaben oder einer Buchstabenfolge.

Zur Verschlüsselung schreiben Sie den Schlüssel so oft wie notwendig über die Nachricht in Klartext. Nun verschlüsseln Sie jeden einzelnen Buchstaben, indem Sie die ASYM-Codierscheibe so drehen, dass der entsprechende grüne Schlüssel im grau unterlegten Fenster sichtbar wird. Lesen Sie dann den roten Geheimbuchstaben im Sichtfeld neben dem blauen Klartextbuchstaben ab! Leerzeichen bleiben unverändert und werden bei der Codierung nicht berücksichtigt.

Abbildung 12.7 zeigt ein Beispiel. Die Nachricht „ABENTEUER INFORMATIK“ soll mit dem Schlüssel „YZS“ verschlüsselt werden. „YZS“ wird hierfür entsprechend oft

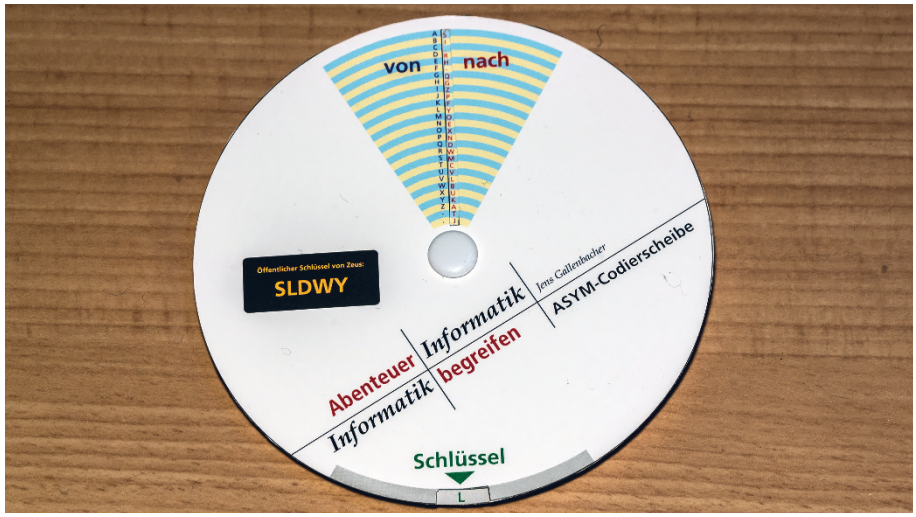


Abbildung 12.6
Die ASYM-Codierscheibe mit eingestelltem Schlüssel „L“

YZSYZSYZS YZSYZSYZSY
ABENTEUER INFORMATIK

HEAYUAW-U NFGDPTHUYA

Abbildung 12.7
Die Codierung von „ABENTEUER INFORMATIK“ mit dem Schlüssel „YZS“

über den Klartext geschrieben. In der Abbildung werden – genau wie im Codierer – Blau für den Klartext, Grün für den Schlüssel und Rot für die Geheimschrift verwendet.

Um den ersten Buchstaben „A“ zu codieren, stellen Sie den Codierer auf den ersten Schlüsselbuchstaben „Y“ ein. Lesen Sie dann den ersten Geheimbuchstaben neben dem blauen „A“ ab. Es handelt sich um ein „H“. Machen Sie nun das Gleiche für alle Klartextbuchstaben, die unter einem „Y“ stehen. Danach sind die Schlüsselbuchstaben „Z“ und „S“ dran, stellen Sie den Codierer auf die entsprechenden Schlüssel und codieren Sie die zugeordneten Buchstaben des Klartextes. Die resultierende Geheimbotschaft lautet „HEAYUAW-U NFGDPTHUYA“. Es ist sicherlich ziemlich schwierig, ohne den entsprechenden Schlüssel den Klartext herauszufinden. Dieser Schlüssel heißt allerdings nicht „YZS“, sondern „VCT“!

Probieren Sie es selbst aus! Gehen Sie genauso vor wie bei der Erstellung der Geheimschrift, nur „verschlüsseln“ Sie diesmal die Botschaft „HEAYUAW-U NFGDPTHUYA“ mit dem Schlüssel „VCT“.

Abbildung 12.8 zeigt das Ergebnis. Die neue „Geheimbotschaft“ ist die Nachricht, mit der Sie angefangen hatten – „ABENTEUER INFORMATIK“.

VCTVCTVCT VCTVCTVCTV
HEAYUAW-U NFGDPTHUYA

ABENTEUER INFORMATIK

Abbildung 12.8
Decodierung der Geheimbotschaft

Probieren Sie aus, was passiert, wenn Sie auf die Geheimbotschaft „HEAYUAW-U NFGDPTHUYA“ statt des Schlüssels „VCT“ den ursprünglichen Schlüssel „YZS“ anwenden, also das Decodieren – wie bei der Cäsar-Scheibe – mit dem gleichen Schlüssel wie das Codieren durchführen.

Abbildung 12.9 zeigt das Ergebnis. Die ASYM-Codierscheibe macht ihrem Namen alle Ehre, denn mit dem symmetrischen Ansatz kommt nur Kauderwelsch heraus. Wer also „nur“ im Besitz des gleichen Schlüssels ist, mit dem eine Nachricht codiert wurde, kann diese nicht einfach entziffern!

Abbildung 12.9
Versuch, die Geheimbotschaft mit dem gleichen Schlüssel zu entziffern

YZSYZSYZS YZSYZSYZSY
HEAYUAW-U NFGDPTHUYA

F-FZIFJLI YOMCKOFIEH

Experimentieren Sie weiter: Verwenden Sie nun zum Verschlüsseln der Nachricht den Schlüssel „VCT“ und dann zum Entschlüsseln „YZS“. Wie sieht es aus, wenn man die beiden Schlüssel „GUT“ und „DKS“ nutzt?

Das Verfahren funktioniert bei allen verwendeten Schlüsseln – die Reihenfolge der Anwendung ist also unerheblich. Auch das neue Schlüsselpaar lässt sich zum Codieren und Decodieren einer beliebigen Nachricht nutzen. Unlesbar bleibt die Botschaft, wenn man einen beliebigen Schlüssel doppelt anwendet.

Fassen wir unsere Erkenntnisse zusammen:

Die ASYM-Codierscheibe erlaubt Ihnen, einen Code zu benutzen, der zwei unterschiedliche Schlüssel hat. Durch die Anwendung beider Schlüssel hintereinander kommt wieder die ursprüngliche Nachricht hervor. Dabei ist die Reihenfolge der Anwendung unerheblich.

Eine einmal verschlüsselte Nachricht ist somit ausschließlich mit dem entsprechenden Gegenschlüssel wieder zu entziffern. Kein anderer Schlüssel, auch nicht derjenige, der zur Verschlüsselung genutzt wurde, kann den Klartext sonst noch hervorbringen!

Die Schönheit der Asymmetrie

Als Nächstes wollen wir dieses neue Verfahren dazu nutzen, Nachrichten sicher über das Internet zu senden. Zuvor jedoch noch ein Hinweis, wenn Sie gegenüber der ASYM-Codierscheibe skeptisch sind und meinen, den Code geknackt zu haben: Lesen Sie bitte den Hinweis am Ende des Kapitels. Falls das nicht der Fall ist: Nehmen Sie sich nicht den Spaß und arbeiten Sie das Kapitel einfach weiter durch.

Spielen wir jetzt in Form eines Rollenspiels ein Szenario durch, bei dem die Cäsar-Scheibe als Codierer versagt hatte. Die teilnehmenden Personen haben hier von mir Namen bekommen: Karl Kunde, Vera Verkäuferin und Spike Spitzbube. Später kommt dann noch Zoe Zertifizierung dazu. Sie stehen stellvertretend für Sie und Ihre echten Freunde, mit denen Sie das Szenario durchspielen. Das ist ein großer Spaß und ich empfehle es, um das Verfahren zu verstehen. Basteln Sie für alle Teilnehmerinnen und Teilnehmer eine Cäsar-Scheibe (als Codierer für einen symmetrischen Code) und eine ASYM-Codierscheibe. Außerdem bekommen alle außer Kunibert Kunde jeweils die Codepaare einer Farbe aus Kopiervorlage 12.K3 und 12.K4 oder den Bastelbögen ausgehändigt. Abbildung 12.10 zeigt die Ausgangsposition. Karl, Vera und Zoe haben eigene Zimmer und können sich nur mit Hilfe von Spike verständigen, der als Bote Nachrichten überbringt. Spike schreibt Nachrichten dazu prinzipiell ab und überbringt nur die Version in seiner Handschrift.

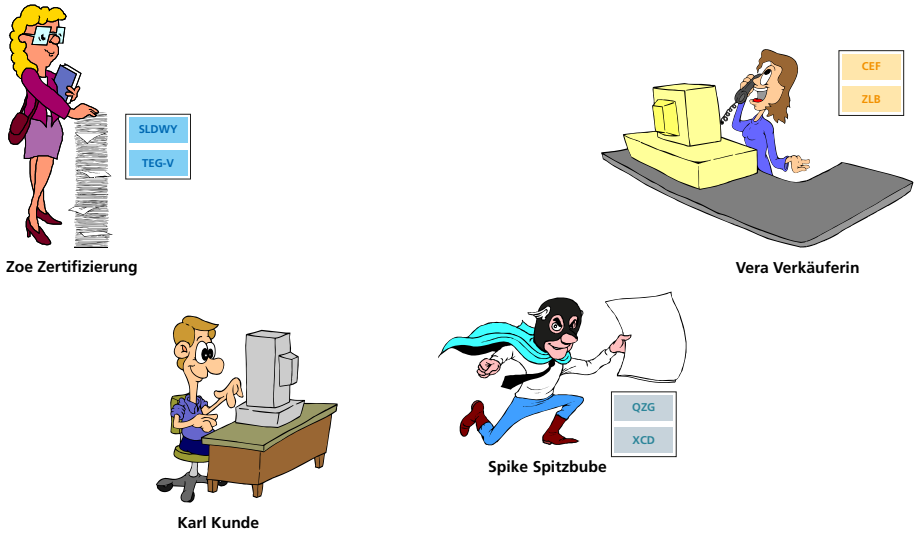


Abbildung 12.10
Ausgangslage für das Rollenspiel. Alle Personen sind als Comicfigur dargestellt.

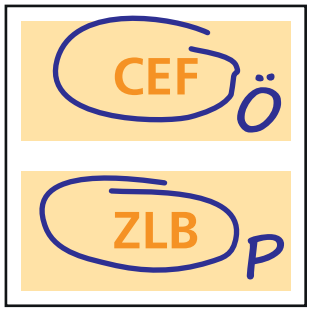
Karl sitzt sonntags gerne vor dem PC, um zu shoppen. Er entdeckt im Portal von Vera das Buch „Abenteuer Informatik“, das er natürlich sofort erwerben möchte. Für die Bezahlung möchte er seine Kreditkarte nutzen, die Vera auch gerne akzeptiert. Selbstverständlich sollen die notwendigen Daten – wir gehen hier im Spiel von der Nummer „ZWEIVIERDREIACHT“ aus – ausschließlich an Vera gehen und niemandem sonst in die Hände fallen.

Karl und Vera sitzen in unterschiedlichen Zimmern. Ein persönliches Treffen zwecks Schlüsseltausch ist ausgeschlossen. Das Verschicken geht nur über Spike. Versuchen Sie die Aufgabe alleine unter Verwendung Ihrer Cäsar-Scheiben zu lösen. Beschreiben Sie, welche Nachrichten Karl und Vera mit welchen Schlüsseln codieren, um das Ziel zu erreichen.

Wie man es auch dreht und wendet – ohne zuvor einen Schlüssel ausgetauscht zu haben, gibt es keine Möglichkeit, die Nachricht sicher zu übermitteln! Entweder Sie übermitteln gleich den Klartext oder Sie müssen vorher irgendwie den Schlüssel weitergeben, was auf das Gleiche hinausläuft, denn das geht auch nur über Spike, der ihn dann ebenfalls hat und jede Ihrer Nachrichten entschlüsseln kann.

Natürlich könnten Sie tricksen und ein Geheimnis nutzen, das bekanntermaßen von Vera und Karl geteilt wird, etwa „der Anfangsbuchstabe des Filmtitels, den wir zusammen letztes Weihnachten gesehen haben“. Allerdings ist dies für unser Rollenspiel unrealistisch, da Sie normalerweise mit einem Online-Händler kein gemeinsames Geheimnis haben.

Probieren wir, ob mit der ASYM-Codierscheibe mehr zu erreichen ist. Wir gehen dabei davon aus, dass ein zusammengehöriges Schlüsselpaar sehr schwer zu finden ist und man von einem Schlüssel des Paares nicht auf den zugehörigen anderen schließen kann. Diese Schlüsselpaare sind auf den bunten Kärtchen abgedruckt. Wie aus Abbildung 12.10 zu sehen ist, haben Vera, Zoe und auch Spike bereits ein solches Schlüsselpaar. Karl als „normaler“ Kunde im Internet hat allerdings keinen Schlüssel. Dieser müsste auf sichere Art und Weise erzeugt werden, und bisher ist dies für normale Computernutzer leider immer noch nicht üblich.



Wie geht man nun vor?

Vera nimmt eines ihrer Schlüsselpaare, zum Beispiel „CEF“ und „ZLB“. Sie markiert einen Schlüssel mit einem „Ö“ und einen anderen mit „P“. Das „Ö“ steht für „öffentlich“ – diesen Schlüssel dürfen alle haben, die ihn bekommen möchten. „P“ steht für „privat“, diesen Schlüssel hält Vera geheim! Niemand (auch nicht die engsten Freunde und auch kein Kunde) bekommt ihn. Daher tun wir jetzt auch so, als ob er nicht hier im Buch für alle zu lesen stände ...

Von alledem weiß Karl als Kunde jedoch nichts! Er möchte immer noch die Kreditkartennummer sicher an Vera übermitteln und benötigt dazu erst einmal einen Schlüssel. Er schickt daher an Vera folgende Nachricht in Klartext: „Wie lautet der Schlüssel?“ Spike liefert diese aus und nimmt die Antwort von Vera entgegen: „CEF“. Auch diese Nachricht mit dem Schlüssel geht unverschlüsselt an Karl. Abbildung 12.11 zeigt diesen Dialog. Der besseren Übersichtlichkeit halber nummerieren die Teilnehmerinnen und Teilnehmer ihre Zettel zusätzlich.

Nun hat Karl die Information über Veras Schlüssel. In Abbildung 12.12 ist dies in roter Schrift bei der Comicfigur vermerkt. Er kann die Nachricht „ZWEIVIERDREIACHT“ damit verschlüsseln und Spike die Chiffre „VRVUU.BCFMN.LTLH“ für Vera übergeben. Spike kann damit nichts anfangen, denn er ist nur im Besitz des öffentlichen Schlüssels „CEF“, und damit lässt sich die Nachricht nicht decodieren. Das sogenannte Sicherheitsziel Vertraulichkeit ist gewährleistet.

Sicherheitsziel: Vertraulichkeit

Vertraulichkeit ist dann hergestellt, wenn nur die Adressaten einer Nachricht deren Informationen erschließen können. Anders herum ausgedrückt: Eine vertrauliche Nachricht kann nicht von Dritten gelesen werden.

Vera gelingt dies allerdings mit Hilfe ihres privaten Schlüssels „ZLB“. Sie schickt Kurt die Quittung „ist angekommen!“ in Klartext zurück – hier ist ja kein Geheimnis auszukundschaften.

Auf den ersten Blick sieht es nun so aus, als ob Spike keine Chance hätte, die geheimen Botschaften zu erkunden. Ich behaupte einfach einmal, dass diese Annahme verfrüht ist. Daher dürfen Sie als Nächstes ausnahmsweise einmal Ihre volle kriminelle Energie ausleben:

Versetzen Sie sich in die Rolle von Spike. Dieser möchte unbedingt Sand ins Getriebe streuen und Vera Verkäuferin nachhaltig Schaden zufügen. Wie gelingt ihm das auf ganz einfache Art und Weise?

Sie sind jetzt sicherlich auf eine ganze Reihe diabolischer Pläne gekommen. Sprechen wir ein paar davon durch: Natürlich könnte Spike einfach alle Botschaften wegwerfen und gar nicht weiterleiten. Wenn jetzt jedoch gar keine Antworten kämen, würde Karl misstrauisch und einen anderen Boten wählen. Diese Möglichkeit wäre also nicht sehr geschickt.

Viel diabolischer ist es, wenn Spike nur die verschlüsselte Nachricht (3) nicht an Vera weiterleitet und diese wegwirft. Allerdings schreibt er selbst die Antwort „ist ange-

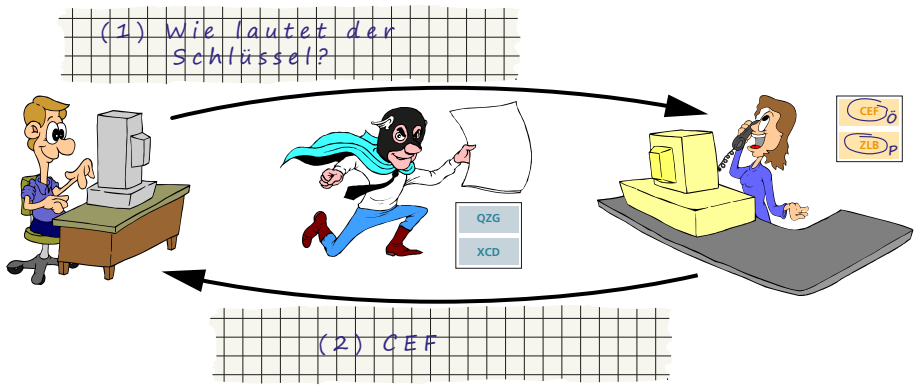


Abbildung 12.11
Zunächst wird der Schlüssel angefragt und übermittelt.

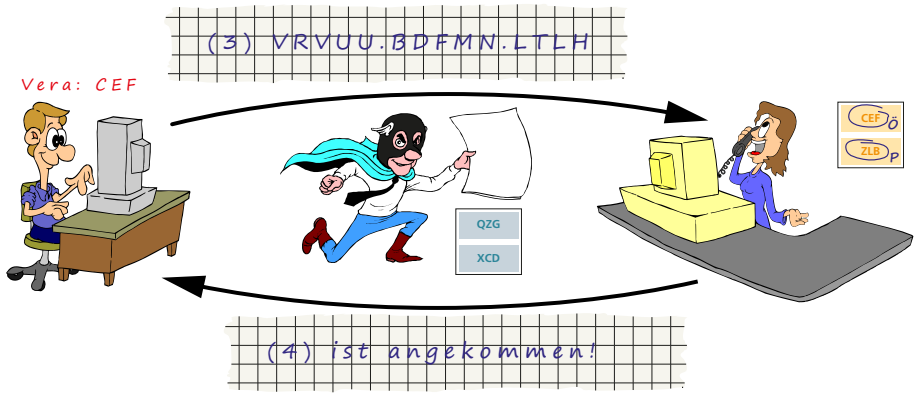


Abbildung 12.12
Danach kann die Nummer der Kreditkarte verschlüsselt übertragen werden.

kommen!“ und überbringt sie nach kurzem Warten. Auf diese Weise wird Karl nicht misstrauisch und denkt, alles sei in Ordnung, während Vera die Zahlungsdaten nie erhält und damit auch kein Kaufvertrag zustande kommt. Karl wartet vergeblich auf die Lieferung und ärgert sich über Vera. Ob er ein zweites Mal bei ihr bestellt, ist zweifelhaft. Abbildung 12.13 zeigt diese Alternative schematisch.

Überlegen Sie, wie man die Kommunikation durch eine kleine Veränderung gegen dieses Vorgehen schützen kann! Tipp: Es geht natürlich darum, die Antwort irgendwie abzusichern.

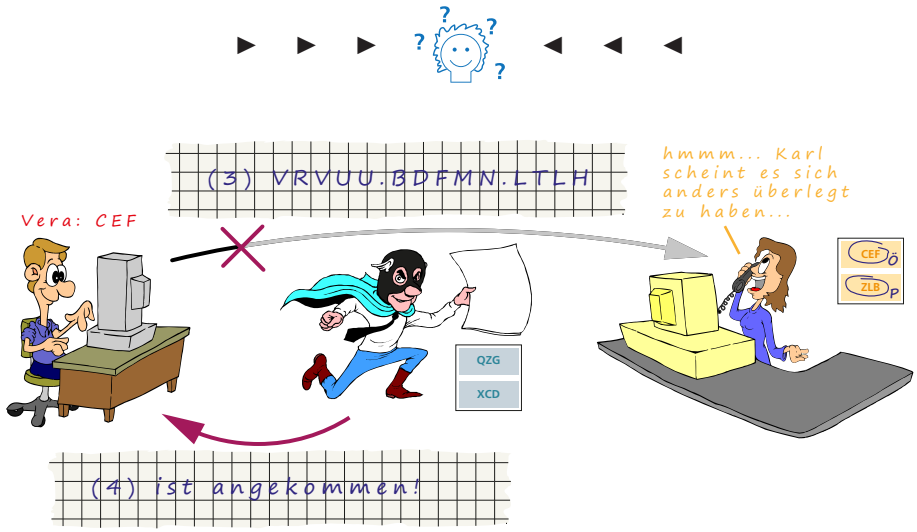


Abbildung 12.13
Alternativer Verlauf der Kommunikation. Spike fängt die verschlüsselte Nachricht ab und antwortet selbst.

Ronald Linn Rivest (oben), Adi Shamir (Mitte) und Leonard Adleman (unten) entwickelten 1977 am MIT das asymmetrische kryptographische Verfahren, das auch heute noch vom Prinzip her am weitesten verbreitet ist: Nach den Anfangsbuchstaben der Nachnamen heißt es RSA-Verfahren. Es beruht darauf, dass es recht schnell geht, zwei große Primzahlen miteinander zu multiplizieren, es aber sehr lange dauert, aus dem Produkt die Primfaktoren herauszubekommen.

Das Problem tritt auch bei der Kommunikation im Internet auf: Einerseits möchten wir für bestimmte Nachrichten die Vertraulichkeit sicherstellen. Das machen wir mit Hilfe der Verschlüsselung. Es gibt jedoch eine weitere Situation: Hier versuchen wir zu erreichen, dass eine Nachricht auch immer tatsächlich von der Person kommt, die als Absender angegeben ist. Das nennt man Authentifizierung. Genau das Problem stellt sich bei Veras Antwort „ist angekommen!“.

Nur wenn Karl sicher ist, dass die Antwort wirklich von Vera kommt, kann er sich auf die Lieferung freuen und – falls diese nicht eintrifft – mit einigem Recht Vera für das Versäumnis verantwortlich machen. Daher versuchen wir, mit der ASYM-Codierscheibe auch dieses Sicherheitsziel zu erreichen.

Stellen Sie im Rollenspiel Folgendes nach: Vera schickt ihre Antwort „IST ANGEKOMMEN“ nicht im Klartext, sondern codiert sie mit ihrem privaten Schlüssel. Überlegen Sie, was das genau bewirkt.

Spike muss nun die Nachricht „HMW QEX-FJROMF“ weiterleiten. Er kann diese jederzeit entschlüsseln, denn dies lässt sich mit dem öffentlichen Schlüssel sehr leicht erledigen. In Abbildung 12.14 ist daher auch bei Spike eingetragen, dass er natürlich ebenfalls Veras öffentlichen Schlüssel besitzt. Entweder hat er abgehört, als Vera diesen an Karl geschickt hat, oder aber er hat Vera ganz normal danach gefragt – sie gibt den öffentlichen Schlüssel ja einfach an alle heraus. Allerdings schafft Spike es nicht, selbst eine solche Nachricht zu erzeugen: Dazu bräuchte er den privaten Schlüssel, und den hält Vera geheim!

Was macht nun Karl? Sobald er eine Antwort von Vera bekommt, decodiert er diese als Erstes mit ihrem öffentlichen Schlüssel. Falls nun etwas Sinnvolles herauskommt, kann er sicher sein, dass der Absender tatsächlich Vera ist. Falls nicht, hat sich offenbar jemand an der Nachricht zu schaffen gemacht – Spike ist enttarnt und kann zum Beispiel durch einen anderen Boten ersetzt werden. Das Sicherheitsziel der Authentizität ist für die Antwort von Vera erreicht.

Sicherheitsziel: Authentizität

Authentizität ist dann hergestellt, wenn sichergestellt ist, dass Nachrichten tatsächlich von den angegebenen Absendern stammt. Anders herum ausgedrückt: Es kann niemand unbefugt unter meinem Namen eine Nachricht verschicken.

Fassen wir bis hierher nochmals die Erkenntnisse über die ASYM-Codierung zusammen: Wenn Sie ein Schlüsselpaar besitzen und einen Schlüssel als öffentlichen Schlüssel bekannt machen, dann können alle Ihnen vertrauliche Nachrichten schicken. Sie wissen allerdings nicht, ob der Absender wirklich „echt“ ist, denn der öffentliche Schlüssel kann ja auch von Spitzbuben genutzt werden.

Sie selbst können niemandem eine geheime Nachricht schicken, allerdings sind Sie in der Lage, Ihre Nachrichten zu authentifizieren: Alle können nachvollziehen, dass die von Ihnen verschickte Botschaft auch wirklich von Ihnen stammt, denn nur Ihre Botschaften geben einen Sinn, wenn man sie mit Ihrem öffentlichen Schlüssel decodiert.

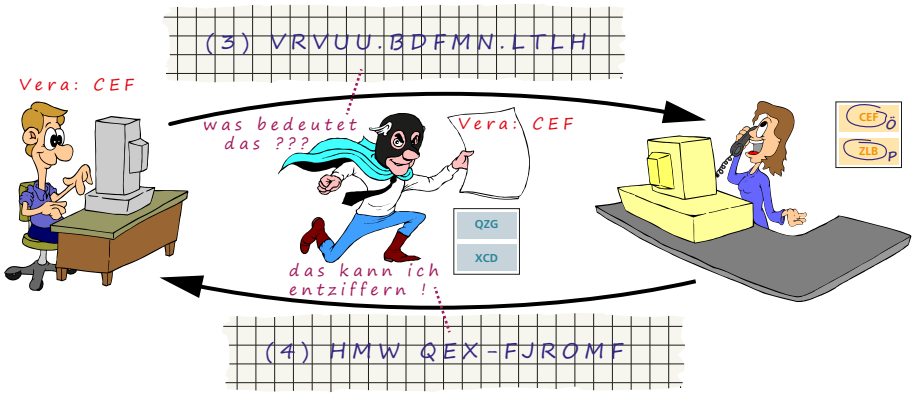


Abbildung 12.14
Vera verschlüsselt ihre Antwort mit dem privaten Schlüssel.

Trotz allem kann Spike nun noch Schaden anrichten! Das Problem liegt im anfänglichen Schlüsseltausch. Wir beginnen nochmals komplett von vorne. Karl weiß noch nichts über Vera und fragt sie: „Wie lautet der Schlüssel?“ Spike liefert nun diese Nachricht bestimmungsgemäß ab, behält allerdings die Antwort „CEF“ ein und liefert stattdessen „QZG“. Dabei handelt es sich um einen seiner eigenen öffentlichen Schlüssel.

Karl ist nicht in der Lage zu erkennen, ob die Antwort tatsächlich von Vera stammt, denn zu diesem Zeitpunkt hat er ja noch keinen Schlüssel erhalten, der die Authentizität gewährleistet. Daher codiert er seine Kreditkartennummer mit dem öffentlichen Schlüssel „QZG“ und verschickt „.NCCZQUPNF-QJV.Z“. Das kann Spike ohne Probleme mit seinem eigenen privaten Schlüssel „XCD“ entziffern. Um keinen Verdacht zu erregen, codiert er dann die Nachricht neu mit Veras öffentlichem Schlüssel „CEF“. Auf diese Weise erkennen weder Vera noch Karl, dass jemand die Botschaft abgefangen hat, und reagieren normal. Selbstverständlich muss Spike nun auch Veras Antwort wieder mit dem eigenen privaten Schlüssel umcodieren, damit Karl von der Authentizität überzeugt ist.

Spike ist nun im Besitz der Kreditkartendaten und kann eigene Bestellungen auf Karls Kosten aufgeben. Wartet er damit eine angemessene Zeit, ist es unwahrscheinlich, dass Karl die Sache mit der längst abgeschlossenen Bestellung an Vera in Zusammenhang bringt.

Spielen Sie diese Situation im Rollenspiel nach. Wechseln Sie dabei am besten in alle drei Rollen. Sie können dabei immer Ihren persönlichen Schlüssel behalten, denn Sie geben ja auf jeden Fall nur Ihren öffentlichen Schlüssel heraus.

Die Abbildungen 12.15 und 12.16 auf der nächsten Seite zeigen den Vorgang zum Vergleich nochmal als Comic. Offenbar ist der Austausch des Schlüssels hier – wie schon mit der Cäsar-Scheibe – das entscheidende Problem! Das Gleiche gilt im Internet: Die asymmetrische Verschlüsselung erlaubt eine grundsätzlich sichere Kommunikation mit einem Partner. Die Voraussetzung ist allerdings, auf sichere Art und Weise an einen Schlüssel heranzukommen.

Für unser Spiel bedeutet dies: Mit dem gegebenen Szenario gibt es für Karl keine Möglichkeit, eine vertrauliche Nachricht an Vera zu schicken. Hierfür müssen wir eine weitere Instanz hinzunehmen, wie im Folgenden beschrieben.

Abbildung 12.15
Spike schickt Karl den eigenen öffentlichen Schlüssel.

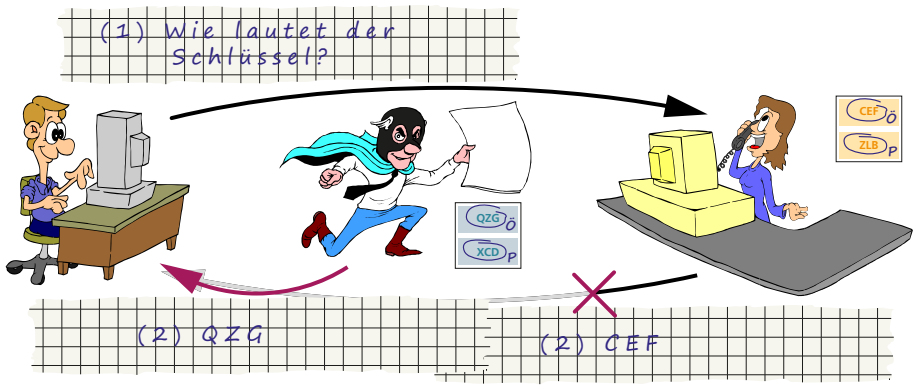
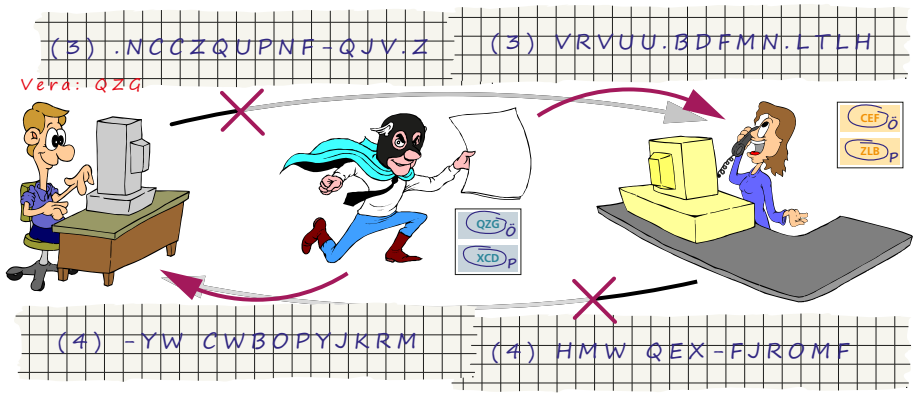
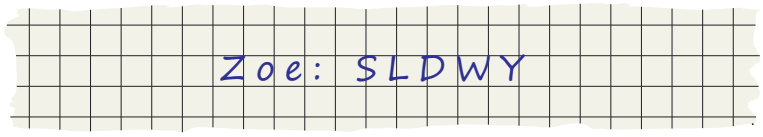


Abbildung 12.16
Spike codiert beim Überbringen die Nachrichten immer auf den „richtigen" Schlüssel um.



Nun kommt Zoe Zertifizierung ins Spiel. Sie übernimmt die Rolle der sogenannten „Zertifizierungsinstanz" oder „Certification Authority (CA)". Zu diesem Zweck drückt sie vor dem Spiel jedem Teilnehmer persönlich einen Zettel mit der Aufschrift in die Hand, die sich auch groß auf der ASYM-Codierscheibe befindet:



Außerdem bietet Sie gegen einen kleinen Obolus an, dass man sich persönlich mit ihr trifft und einen Schlüssel übergibt. Vera als Internet-Verkäuferin nimmt dies gerne in Anspruch und übergibt ihr persönlich ein Schriftstück mit ihrem öffentlichen Schlüssel „VERA-CEF". Zoe überprüft anhand offizieller Dokumente, etwa dem Personalausweis, dass es wirklich Vera ist, die ihr den Schlüssel übergibt, und archiviert den Schlüssel.

Zoe ist außerdem bereit, jedem Auskünfte zu erteilen. Insbesondere gibt sie die ihr anvertrauten öffentlichen Schlüssel gerne weiter. Abbildung 12.17 zeigt die Ausgangssituation nochmals schematisch.

Seien Sie Verschlüsselungsdetektiv! Wie könnten Karl und Vera nun ihre Nachrichten sicher austauschen, ohne Spike eine Chance zu geben? Spielen Sie Ihre Lösung gleich mit verteilten Rollen durch!

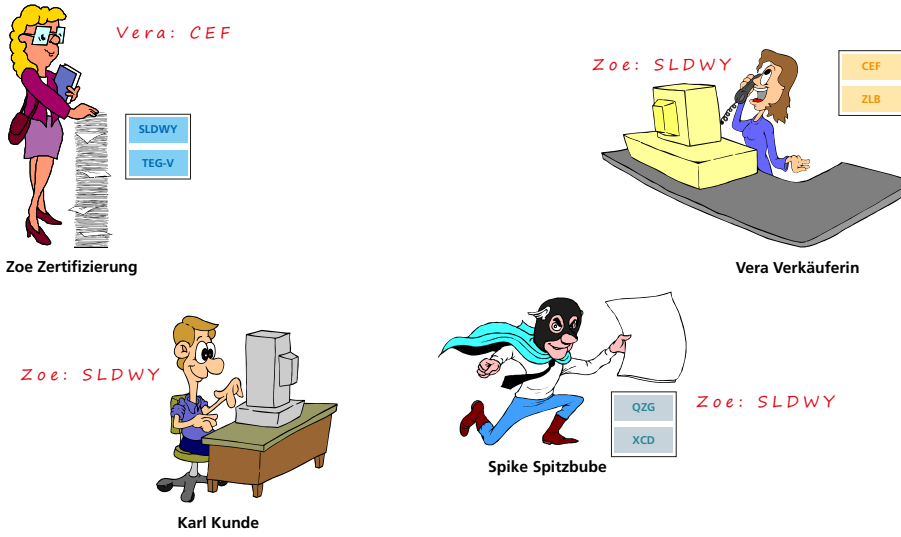


Abbildung 12.17
Ausgangssituation zum Rollenspiel mit Zoe. Alle haben Zoes öffentlichen Schlüssel direkt bekommen und kennen diesen. Zoe hat direkt Veras öffentlichen Schlüssel bekommen.

Der persönlich übergebene Zettel stellt sicher, dass alle wirklich Zoes echten öffentlichen Schlüssel bekommen haben! Auf diese Weise kann also jeder von Anfang an mit Zoe verschlüsselt Nachrichten austauschen – auf dem Hinweg vertraulich, auf dem Rückweg authentifiziert!

Das kann uns dabei helfen, die Schwäche des letzten Versuchs zu überwinden. Da hatte Spike noch ausgenutzt, dass Karl nicht feststellen konnte, ob der Schlüssel, den er von Vera bekommen hatte, auch wirklich der Schlüssel von Vera war.

In den Abbildungen 12.18 bis 12.21 zeige ich Ihnen nun, wie das funktionieren kann. Als kleine Gemeinheit ist hier allerdings immer noch eine Schwachstelle eingebaut, die Spike ausnutzen kann, wenn er sie herausfindet ...

Der besseren Verständlichkeit halber verzichte ich diesmal auf die konkrete Verschlüsselung mit der ASYM-Codierscheibe. Der entsprechend angewendete Code steht in einem roten Oval an der eigentlichen Nachricht am Pfeil zwischen den Kommunikationspartnern:

Dies bedeutet, dass die Nachricht „ZWEIVIERDREIACHT“ mit dem Schlüssel „SLDWY“ codiert wurde. Konkret steht das also für „KBRABYHAW.AZUJFD“. Der Pfeil verbindet dabei Sender und Empfänger, auch wenn jede Nachricht selbstverständlich von Spike überbracht wird – mit der Möglichkeit, diese zu manipulieren.

Der hauptsächliche Trick bei der Angelegenheit ist für Karl, nicht Vera nach ihrem Schlüssel zu fragen, denn mit ihr funktioniert zu Beginn ja noch keine verschlüsselte Kommunikation, sondern diese Information von Zoe zu erlangen, mit der Karl auch schon von Anfang an verschlüsselt Nachrichten tauschen kann.

Lesen Sie direkt in den Abbildungslegenden, was genau passiert. Spike denkt auch immer sehr laut (und lesbar), was er von den jeweiligen Nachrichten mitbekommt und was nicht.

Abbildung 12.18
Als erstes fragt Karl Zoe nach Veras öffentlichem Schlüssel. Dabei verschlüsselt er die Frage bereits mit Zoes öffentlichem Schlüssel. Spike kann diese daher nicht entziffern – auch wenn er sich natürlich vorstellen kann, was Karl von Zoe prinzipiell möchte.

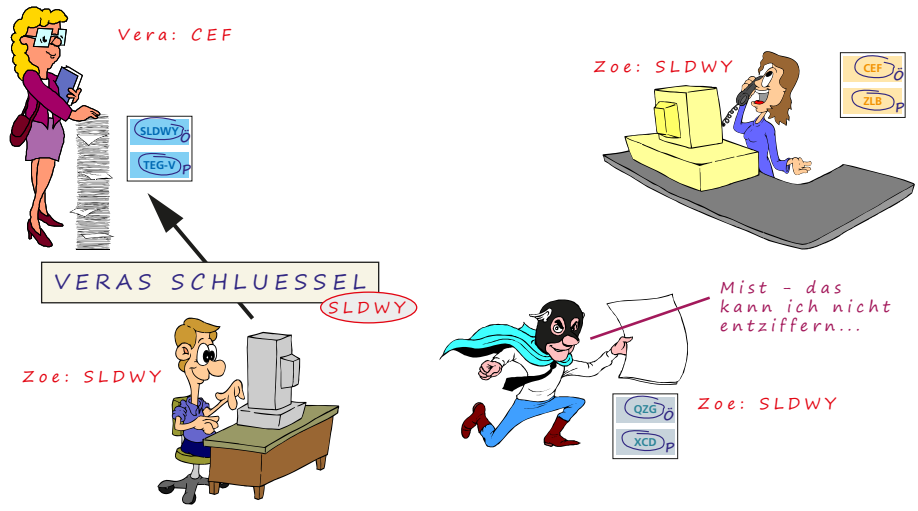


Abbildung 12.19
Zoe antwortet und schickt Veras öffentlichen Schlüssel. Sie authentifiziert die Nachricht mit ihrem privaten Schlüssel. Spike kann die Nachricht „CEF" daher ebenfalls entziffern. Da es sich aber um den öffentlichen Schlüssel handelt, kann man das tolerieren.

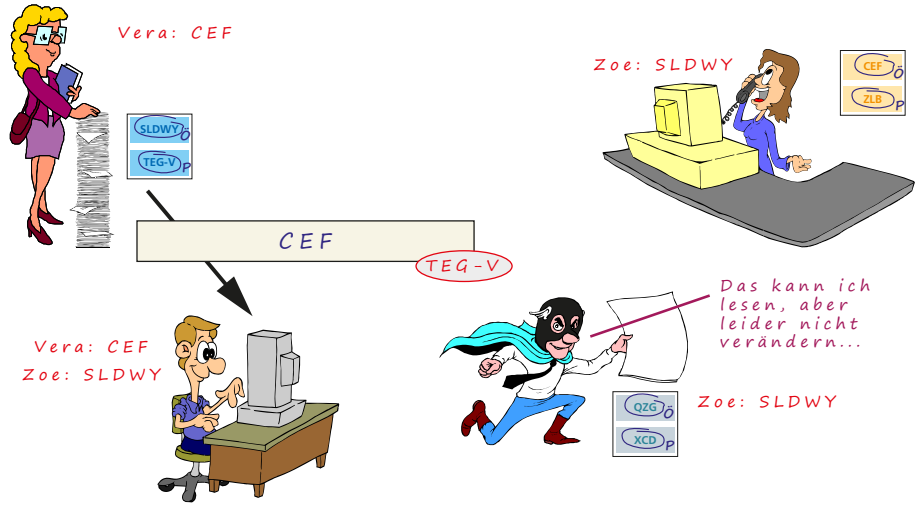
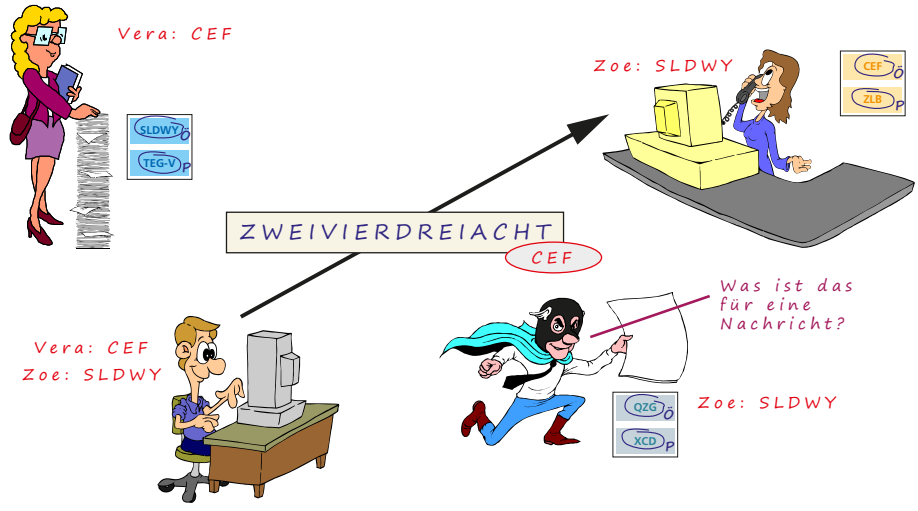


Abbildung 12.20
Nun kann Karl seine Kreditkartennummer an Zoe schicken. Er verschlüsselt sie mit dem authentifiziert empfangenen Schlüssel „CEF". Spike kann sie daher nicht lesen.



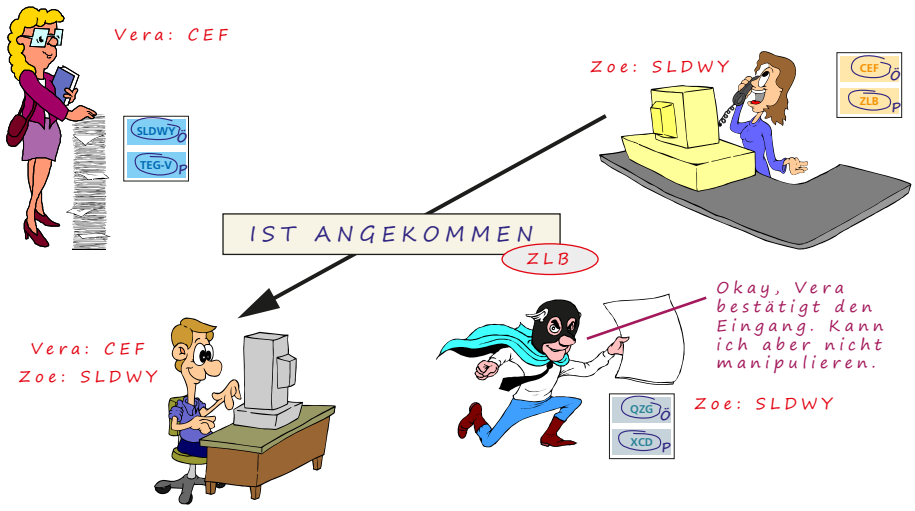


Abbildung 12.21
Vera schickt ihre Bestätigung und authentifiziert sie mit ihrem privaten Schlüssel.

Jetzt sind Sie an der Reihe: Spielen Sie Spike! Wie würden Sie vorgehen, um die Kreditkartennummer trotz aller Sicherheitsvorkehrungen auszuspionieren?

Tja – Spike ist immer noch der Überbringer aller Nachrichten. Er kann jede Nachricht abfangen und durch eine andere ersetzen. Letztlich möchte er die Kommunikation zwischen Karl und Vera abhören. Er darf daher auf keinen Fall zulassen, dass Karl Veras echten öffentlichen Schlüssel bekommt! Wenn das passiert, ist die Sache wasserdicht und er aus dem Rennen. Den Schlüssel bekommt Karl allerdings von Zoe, und genau hier greift Spike an:

Auch Spike kann sich natürlich zu Beginn mit Zoe treffen und dort ganz legal seinen öffentlichen Schlüssel hinterlegen. Nun kennt Zoe auch seinen Schlüssel und kann diesen auf Anfrage weitergeben.

Schauen Sie sich Abbildung 12.18 nochmals an. Hier fragt Karl Zoe nach dem Schlüssel von Vera. Spike weiß nicht wirklich, was in der Nachricht steht, denn Karl hat die Übertragung verschlüsselt – nur Zoe kann diese mit ihrem privaten Schlüssel entziffern. Da Zoe eine Zertifizierungsinstanz ist, kann Emil sich allerdings denken, dass es sich um die Frage nach einem Schlüssel handelt.

Er fängt daher die Nachricht ab und schickt stattdessen in Karls Namen die Frage „SPIKES SCHLÜSSEL“ an Zoe. Karl besitzt selbst keinen privaten und öffentlichen Schlüssel. Daher kann er die Nachricht zwar verschlüsseln, aber gegenüber Zoe nicht authentifizieren. Zoe kann daher nicht entscheiden, ob die Nachricht wirklich von Karl stammt oder nicht. Da die Nachricht mit dem öffentlichen Schlüssel von Zoe codiert wird, ist jeder – auch Spike – hierzu in der Lage. Abbildung 12.22 zeigt diesen Schritt.

Nun antwortet Zoe ganz wahrheitsgemäß auf die Anfrage mit Emils öffentlichem Schlüssel (Abbildung 12.23). Sie verschlüsselt diese Nachricht mit ihrem privaten Schlüssel und authentifiziert sie dadurch: Niemand anderes außer Zoe kann diese Nachricht schicken, daher vertraut ihr Karl. Wenn er nun die Kreditkartennummer an Vera übermittelt, verwendet er Spikes Schlüssel. Spike kann die Nachricht wieder

Abbildung 12.22
Spike kann zwar Karls Nachricht nicht lesen, ersetzt sie aber blind durch die Frage nach dem eigenen Schlüssel. Als Absender bleibt Kurt bestehen.

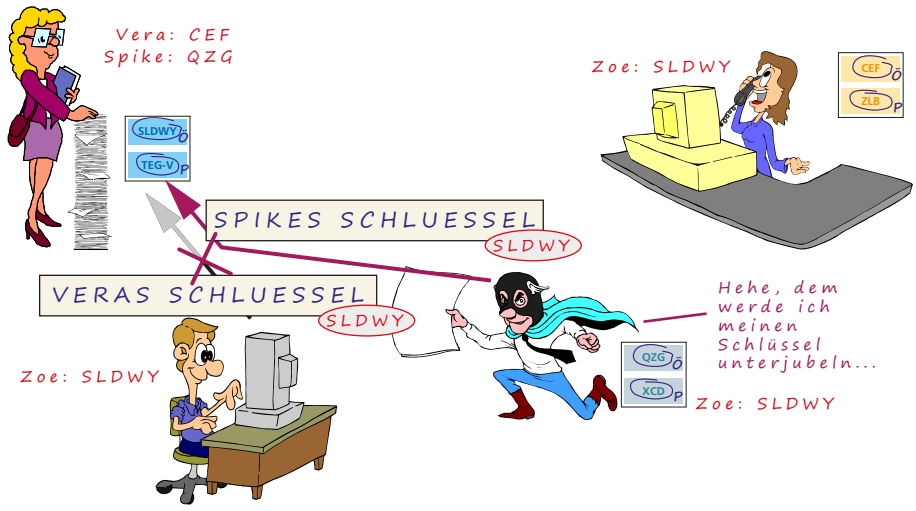
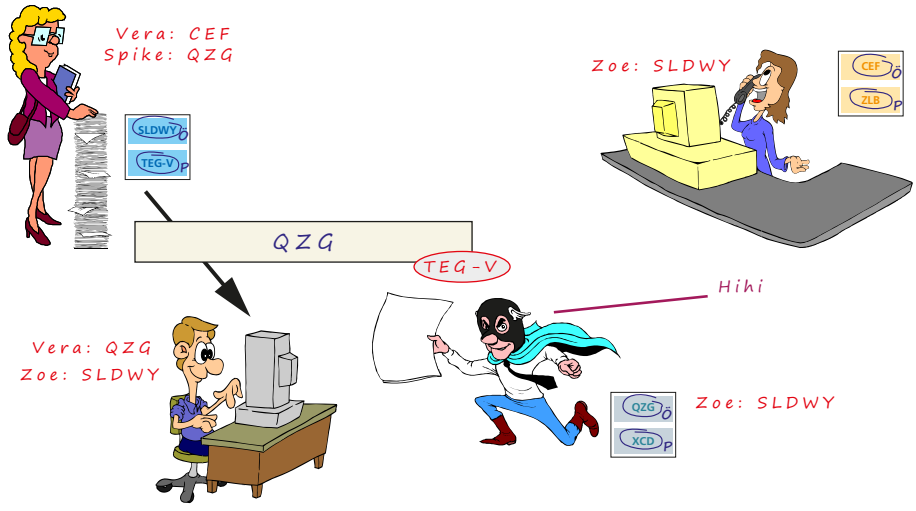


Abbildung 12.23
Zoe antwortet und schickt Spikes öffentlichen Schlüssel. Sie kann nicht ahnen, dass eigentlich Karl nach einem anderen Schlüssel gefragt hat.



abfangen und – wie bereits oben in Abbildung 12.16 dargestellt – seine Spuren verwischen.

Wieder dürfen Sie die Seiten wechseln: vom Schlitzohr zum Sicherheitsexperten. Welche zusätzliche Vorkehrung muss getroffen werden, um die Übermittlung von Veras Schlüssel wirklich sicher zu machen?

Zwei Schwachstellen treten bei der Übermittlung des Schlüssels auf:

1. Zoe kann nicht feststellen, dass die Anfrage nach einem Schlüssel echt ist (oder von einer dritten Partei wie Emil gefälscht wurde).

2. Karl kann nicht feststellen, dass er Spikes Schlüssel statt Veras bekommen hat.

An der ersten Schwachstelle können wir systembedingt nichts verändern. Da Karl – wie jeder normale Internet-Nutzer – nicht im Besitz eines eigenen Schlüssels ist,

kann er die Nachricht nicht authentifizieren. Allerdings können wir an der zweiten Schwachstelle drehen und die Übermittlung wasserdicht machen: Zoe sendet nicht nur einfach den Schlüssel, sondern in der gleichen Nachricht auch die Information, zu wem dieser Schlüssel passt, wie in Abbildung 12.24.

Auf diese Weise werden Schlüssel immer mitsamt dem korrekten Eigentümer übermittelt und abgespeichert. Kurt kann erkennen, dass offenbar mit der Kommunikation zu Zoe etwas schiefgegangen ist, denn sie hat scheinbar nicht die Antwort auf seine Frage gesendet. Auf keinen Fall kommt Kurt aber auf die Idee, seine Kreditkartennummer mit dem falschen Schlüssel zu codieren.

Eine ganz wichtige Regel bei der sicheren Kommunikation ist daher, immer alle notwendigen Informationen in die verschlüsselten Daten mit aufzunehmen und damit allen Beteiligten die Möglichkeit zu geben, diese auf Konsistenz zu überprüfen.

Was zeigt dieses Beispiel noch? Bei der Verwendung der asymmetrischen Verschlüsselung hat immer eine Partei ein Schlüsselpaar und eine andere nicht. Normalerweise haben heute nur Geschäfte, Banken oder andere Anbieter ein Schlüsselpaar, ein „normaler“ Kunde nicht. Daher spreche ich hier von „Kunde“ und „Anbieter“.

Die Kommunikation vom Kunden zum Anbieter ist immer geheim, aber nicht authentifiziert. Die Kommunikation vom Anbieter zum Kunden ist immer offen, aber authentifiziert. Diese Regel muss man jederzeit bedenken und entsprechende – vielleicht überflüssig erscheinende – Informationen zur Sicherung einfügen, wie in diesem Beispiel die Information über den Besitzer des Schlüssels.

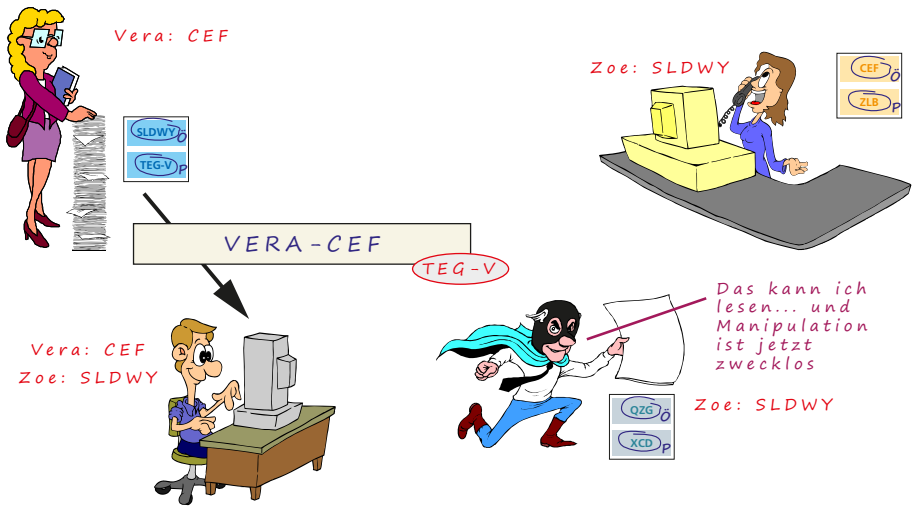


Abbildung 12.24
Zoe schickt nicht nur den puren Schlüssel, sondern einen ganzen Satz mit zugehörigen Informationen, hier insbesondere dem Namen der Eigentümerin.

Mit Zertifikat geht alles besser

Tatsächlich funktioniert es prinzipiell so wie beschrieben, wenn Sie im Internet mit dem sicheren Server eines Händlers verbunden sind und ihm Ihre Kreditkarteninformationen übermitteln: Zunächst überprüft Ihr Rechner anhand einer Zertifizierungsinstanz den korrekten öffentlichen Schlüssel des Händlers. Daraufhin kann er die Nummer sicher senden. Die notwendige Software dazu ist entweder im Internet-Browser oder im Betriebssystem integriert.

Vielleicht wenden Sie nun ein, dass Sie sich nicht erinnern können, in Realität irgendwann von irgendwem das Pendant des Zettels mit der Aufschrift „SLDWY“ zugesteckt bekommen zu haben. Mit anderen Worten: Woher kennt Ihr Computer am Anfang den öffentlichen Schlüssel der Zertifizierungsinstanz? Das ist ja wohl die Voraussetzung dafür, dass die ganzen hier vorgestellten Verfahren auch funktionieren!

Hier hat der Hersteller des Browsers bzw. Ihres Betriebssystems die Arbeit für Sie bereits übernommen. Die öffentlichen Schlüssel der wichtigsten Zertifizierungsinstanzen (Certification Authorities oder CAs) sind hier bereits fest eingebaut. Auf diese Weise ist die Verbindung dorthin auch von Anfang an sicher – es sei denn, jemand hat Ihren Computer manipuliert oder Ihnen manipulierte Software untergeschoben.

Vielleicht ist Ihnen noch ein interessantes Detail aufgefallen: Bei der asymmetrischen Verschlüsselung kann jeder heimliche Mithörer alles lesen, was vom Inhaber des Schlüssels an Sie verschickt wird. Diese Kommunikation ist lediglich authentifiziert, aber nicht vertraulich! Jetzt können Sie sagen: „Na und? Schließlich muss ich ihm ja vertrauliche Dinge schicken wie Passwort oder Kartennummern.“ Prinzipiell stimmt das, aber auch dann wäre es schon seltsam, wenn zum Beispiel Ihr Kontoauszug beim Online-Banking von allen mitgelesen werden könnte.

Nehmen wir als Beispiel aber mal die Kommunikation in einem sozialen Netzwerk: Bei den meisten Diensten verschicken die Teilnehmer ihre Texte nicht direkt untereinander – selbst wenn diese privat und nur für einen bestimmten Adressaten bestimmt sind. Die Nachrichten gehen erst an den Server des Anbieters und werden von dort an die eigentlichen Empfänger verteilt. Da die meisten privaten Anwender kein eigenes Schlüsselpaar besitzen, wäre direkt auch keine vertrauliche Nachrichtenübermittlung möglich – in dieser Hinsicht ist das Verfahren also sinnvoll. Abbildung 12.25 zeigt ein Beispiel für eine solche Kommunikation.

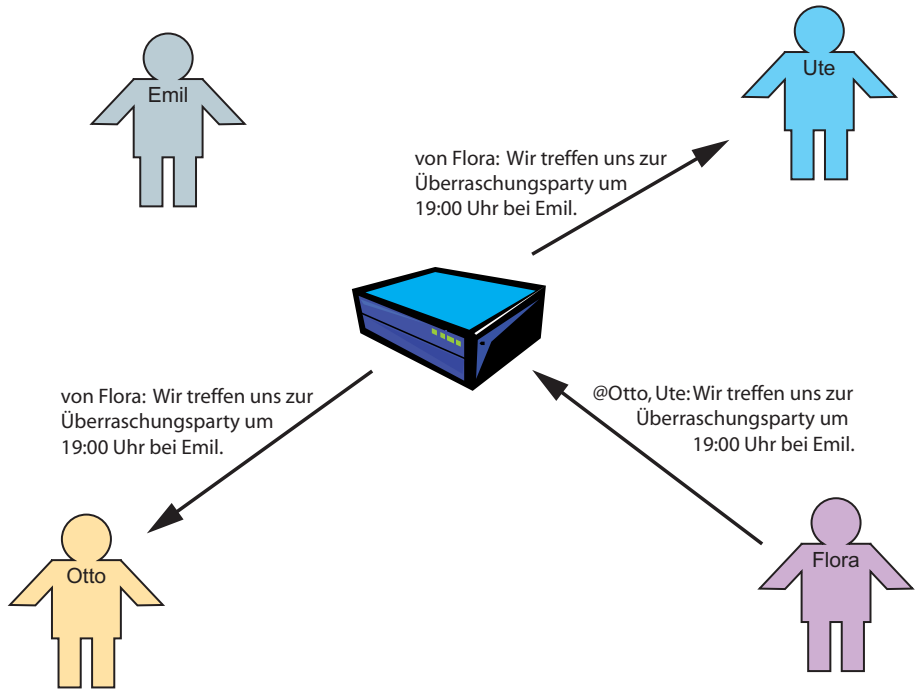


Abbildung 12.25
Flora schickt eine Nachricht, die für Ute und Otto bestimmt ist. Diese wird vom Server des sozialen Netzwerkes an die Adressaten verteilt, so dass Emil in diesem Fall ahnungslos bleibt.

Überlegen Sie, warum in diesem Szenario eine rein asymmetrische Verschlüsselung nicht hinreichend ist.

Alle Nachrichten werden zunächst an den Server geschickt. Dieser Weg ist vertraulich, also ist hier alles klar. Allerdings muss der Server die Nachrichten dann verteilen, und dieser Weg ist zwar authentifizierbar, aber nicht vertraulich. Die Nachrichten können hier also von Spitzbuben mitgelesen werden. Asymmetrische Verschlüsselung allein ist also auch noch nicht ausreichend.

Es gibt noch einen weiteren Grund dafür, nicht einfach alles asymmetrisch zu verschlüsseln, und den können Sie sicherlich selbst gut nachvollziehen, wenn Sie die Experimente dieses Kapitels gewissenhaft nachgestellt haben: Welche Codierung geht schneller – die mit der Cäsar-Scheibe oder die mit der ASYM-Codierscheibe?

Sicherlich schaffen Sie mit der Cäsar-Scheibe deutlich mehr Buchstaben pro Minute! Das Gleiche gilt für einen Computer, der ebenfalls für ein asymmetrisches Verfahren deutlich mehr Rechenleistung benötigt als für ein symmetrisches.

Insgesamt wäre es also in jeder Hinsicht praktisch, symmetrisch zu verschlüsseln – wenn wir nur irgendwie sicher den Schlüssel übergeben könnten ...

Genau dies ist aber dank des asymmetrischen Verfahrens möglich: Versetzen Sie sich nochmals in das Szenario von oben: Karl hat in Abbildung 12.24 gerade von Zoe Zertifizierung Veras öffentlichen Schlüssel „CEF“ bekommen. Er könnte ihr nun – für niemanden lesbar – seine Kreditkartennummer zukommen lassen. Statt der Kreditkartennummer ist aber auch jede andere Information vertraulich übermittelbar. Diese andere Information kann also auch der Schlüssel für die Cäsar-Scheibe sein.

Abbildung 12.26 zeigt das Vorgehen: Karl kann nicht jedes Mal den gleichen Schlüssel nutzen, denn dann könnte Vera ja diesen Schlüssel nehmen, um Karls nächste Nachricht an jemand ganz anderen abzufangen. Daher muss Karl diesen Schlüssel immer neu zufällig erzeugen. Er macht das daher in der Abbildung mit Buchstabenwürfeln und schickt ihn – zusammen mit der Bitte, die Cäsar-Scheibe zu nutzen – an Vera. Vera verschlüsselt nun ihre Bestätigung gleich mit dem neuen symmetrischen Schlüs-

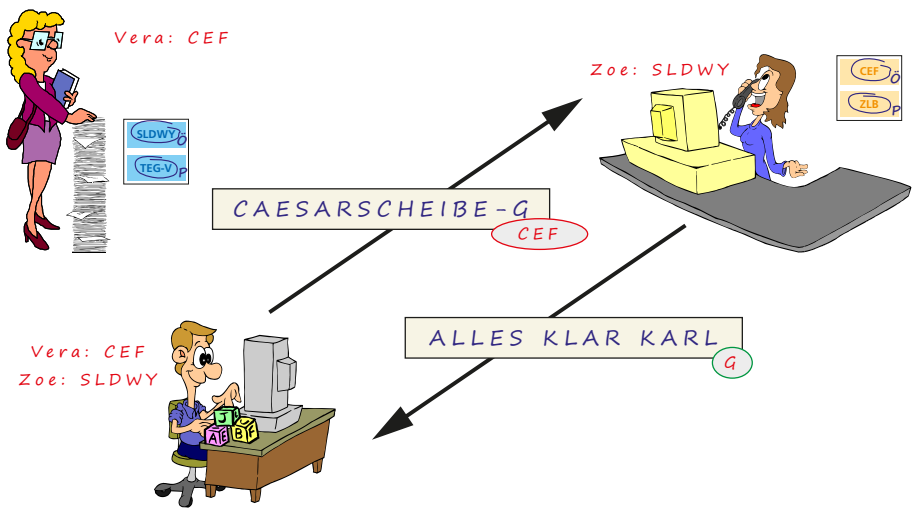


Abbildung 12.26
Zunächst „erwürfelt“ Karl einen Schlüssel für die Cäsar-Scheibe und schickt ihn Vera vertraulich. Diese benutzt ihn gleich, um die Bestätigung zu schicken.

sel und schickt sie an Karl zurück. Die Nachricht lautet also effektiv „GRRKY QRGX QGXR“. Ab diesem Moment können beide sich auf die symmetrische Verschlüsselung mit der Cäsar-Scheibe beschränken.

Spike kann die erste Nachricht nicht entziffern, denn nur Vera kann das mit ihrem privaten Schlüssel. Überlegen wir noch kurz, was passiert, wenn Spike diese Nachricht „CAESARSCHEIBE-G“ nur abfängt und durch eine eigene mit einem eigenen Code ersetzt, zum Beispiel „CAESARSCHEIBE X“. Vera hat keine Chance, den Austausch zu bemerken (denn Spike nutzt den gleichen öffentlichen Schlüssel, um Vertraulichkeit herzustellen), und möchte mit Karl nun über den Code X kommunizieren. Sie schickt also ihre Bestätigung „ALLES KLAR KARL“ mit X verschlüsselt zurück und Spike kann sie sofort entziffern.

Haben Sie bemerkt, dass die zufällige Erzeugung eines symmetrischen Schlüssels in der Kryptographie kritisch ist? Im vorangegangenen Kapitel haben wir uns mit der Erzeugung von Pseudozufallszahlen beschäftigt, weil Computer in der Regel keinen „echten“ Zufall erzeugen können. Ein schlechter Zufallszahlengenerator könnte etwa von Spike dazu genutzt werden, den Schlüssel einfach zu raten.

Damit ist er allerdings mit seinem Latein am Ende, denn Karl erwartet eine Rückantwort, die mit dem Code G verschlüsselt ist. Spike kann das nicht wissen, denn er konnte zwar die Nachricht an Vera abfangen, jedoch nicht entschlüsseln. Daher kann er Karl auch keine gefälschte Antwort schicken, die mit dem Code G verschlüsselt wurde, und seine Schurkerei fliegt auf. Ohne die korrekte Bestätigung von Vera wird sich Karl nicht auf eine Kommunikation einlassen.

Tatsächlich funktioniert auf diese Weise die sichere Kommunikation im Internet, wenn Sie sich als Online-Kunde zum Beispiel bei einer Bank oder einem vernüftigen sozialen Netzwerk, das verschlüsselte Übertragung nutzt, anmelden. Die asymmetrische Verschlüsselung wird nur zu Beginn verwendet, um einen Schlüssel für die symmetrische Verschlüsselung auszutauschen. Danach wird der so vereinbarte Schlüssel für eine Sitzung verwendet. Auf diese Weise bleiben sowohl Ihre Nachrichten an die Internetadresse (z. B. PINs, TANs, Überweisungsaufträge) als auch die Rückantwort (z. B. Kontoauszüge, Auftragslisten) ein Geheimnis!

Mit der Cäsar-Scheibe und der ASYM-Codierscheibe haben Sie übersichtliche Werkzeuge zur symmetrischen und asymmetrischen Verschlüsselung in Händen. Spielen Sie mit Freunden ruhig einmal ein paar Szenarien durch. Es macht auch wirklich Spaß, die Rolle des Schurken zu übernehmen, der versucht, die Botschaften zwischen anderen Spielern abzuhören oder irgendwie anders Kapital daraus zu ziehen.

Daran ist auch nichts Ehrenrühriges, denn in der Realität ist ein Heer ganz legaler „Hacker“ damit beschäftigt, die Sicherheitsvorkehrungen der eigenen Organisation zu umgehen. Auf diese Weise ist man besser auf die Angriffe der „echten“ Schlitzohren vorbereitet.

Was steckt dahinter?

Die durchgeführten Experimente und Rollenspiele haben die wichtigsten Prinzipien der Sicherheitstechnologie im Internet bereits veranschaulicht. Selbstverständlich sind die im letzten Abschnitt praktizierten Verschlüsselungen mit den beiden Scheiben nicht wirklich sicher! Die statistische Analyse, wie ich Sie Ihnen zu Beginn gezeigt habe, ist nur eines der vielen Werkzeuge, die heimliche Horcher zur Verfügung haben.

Es muss also ein Verschlüsselungsverfahren her, das gegen entsprechende Angriffe gefeit ist. Auch hierfür habe ich oben bereits Möglichkeiten angedeutet. Im Wesentlichen geht es darum, die statistischen Besonderheiten der Dateien zu verschleiern.

Wenn zum Beispiel ein Hacker den Datenverkehr zu einer Bank überwacht, kann er sich erst einmal selbst völlig legal bei der Bank anmelden und kennt dadurch das Schema, in dem zum Beispiel die Eingabe der persönlichen Geheimzahl erfolgt. Dieses unterscheidet sich von Kunde zu Kunde wahrscheinlich lediglich in Bezug auf die Kontonummer und die Geheimzahl selbst. Wenn er nun eine fremde, verschlüsselte Nachricht an die Bank abfängt, kann er womöglich dessen Kontonummer auch noch irgendwie herausbekommen, denn diese ist ja kein großes Geheimnis. Auf diese Weise weiß er zu 95 %, wie die unverschlüsselte Nachricht aussieht (eben bis auf die Geheimzahl). Trotzdem darf er daraus kein Kapital schlagen können: Eine winzige Änderung des Klartextes muss den verschlüsselten Text komplett umstrukturieren.

Selbst wenn ein Hacker den kompletten Klartext bereits kennt, darf er daraus nicht den Schlüssel berechnen können. Stellen Sie sich als Beispiel vor, Sie fangen eine Nachricht ab, die mit der Cäsar-Scheibe codiert wurde. Diese lautet „YRCCF“. Wenn Sie nun wissen, dass der Sender jede seiner Nachrichten mit „HALLO“ beginnt, brauchen Sie nur Ihre Scheibe so einzustellen, dass irgendeiner der Klartextbuchstaben mit dem entsprechenden Geheimtextbuchstaben zur Deckung kommt, und können den Schlüssel direkt ablesen: „R“.

Es würde den Rahmen dieses Buches sprengen, die genauen mathematischen Grundlagen für solche Verschlüsselungsverfahren allgemein verständlich darzulegen. Daher beschränke ich mich hier auf den Hinweis, dass es entsprechende Methoden gibt. DES (Data Encryption Standard, mit modernen Abwandlungen wie Triple-DES oder 3DES), AES (Advanced Encryption Standard), IDEA (International Data Encryption Algorithm) und RC4 (Ron's Cipher 4 nach dem Entwickler Ronald L. Rivest) sind nur einige der tatsächlich verwendeten symmetrischen Verschlüsselungen.

Auf die Länge kommt es an

Ganz egal, welche Verschlüsselung letztlich genutzt wird – wichtig ist die Länge des Schlüssels und damit die Anzahl unterschiedlicher Codierungsmöglichkeiten. Nehmen Sie wieder die Cäsar-Scheibe. Hier gibt es nur 26 verschiedene Schlüssel (von denen einer, „A“, nicht wirklich einen Code erzeugt). Ein Hacker könnte also einfach alle Schlüssel durchprobieren und das Verfahren auf diese Weise einfach und schnell knacken.

Stellen Sie sich nun vor, dass ein Computer das Durchprobieren der möglichen Schlüssel übernimmt und damit Millionen von Kombinationen pro Sekunde abhandelt. Die Länge des Schlüssels bestimmt normalerweise die Anzahl verschiedener Möglichkeiten.

So gibt es 248 = 281.474.976.710.656 oder ca. 281 Billionen Kombinationen einer Zahl der Länge von 48 Bit. Allerdings können die meisten Verfahren nicht einfach irgendwelche Zahlen als Schlüssel verwenden. Diese Zahlen braucht ein Hacker dann selbstverständlich nicht auszuprobieren. Daher ist die Länge des Schlüssels nur ein Anhaltspunkt für die Sicherheit. Zum Beispiel entspricht ein 56 Bit langer Schlüssel mit dem RC4-Verfahren in etwa der Sicherheitsstufe eines 640 Bit langen Schlüssels im RSA-Verfahren (auf das wir später zu sprechen kommen).

In den Anfängen hatte man für symmetrische Verfahren mit Schlüssellängen von 56 Bit gerechnet, was mit einem heutiger Computer in Sekunden geknackt werden kann. Die Erhöhung der Schlüssellänge um ein einzelnes Bit verdoppelt die Zahl möglicher

RC4	RSA	EC
48	480	96
56	640	112
64	816	128
80	1248	160
112	2432	224
128	3248	256
160	5312	320
192	7936	384
256	15424	512

Schlüssellängen, die gemäß dem von der EU finanzierten „ECRYPT II Yearly Report on Algorithms and Keysizes (2011–2012)“ als vergleichbar gelten. RC4 ist dabei stellvertretend für eine symmetrische Verschlüsselung, EC steht für „Elliptic Curve“, den neben RSA gebräuchlichen Algorithmus zum Schlüsseltausch. Der Bericht ist mit der Nummer „ICT-2007-216676“ leicht im Internet auffindbar.

Schlüssel und damit theoretisch auch den Aufwand zum Knacken des Schlüssels. Effektiv stimmt das aber nicht ganz, weil unterschiedliche Verfahren Schwächen in Bezug auf Eigenschaften der Schlüssel haben. Daher gibt es auch bei verschiedenen symmetrischen Verfahren „gute“ und „weniger gute“ Schlüssel.

Die verfügbare Rechenleistung pro Kaufkraft hat sich zwischen 1976 und 2016 „nur“ um etwa eine Milliarde, also etwa 2^{30} gesteigert (wofür es aber sehr viele sehr unterschiedliche Berechnungen gibt, das also bitte nur als ganz groben Anhaltspunkt nehmen). Dies bedeutet, dass 2016 zum Beispiel eine Billion Multiplikationen zweier Ganzzahlen etwa gleich viel kosteten wie 1976 tausend dieser Basisoperationen. Rein rechnerisch müsste man also für die gleiche Sicherheit, die 1976 ein 56 Bit langer Schlüssel bot, 30 Bit mehr verwenden, also 86 Bit.

2016 hat allerdings das Bundesamt für Sicherheit in der Informationstechnik (BSI) für entsprechende symmetrische Verfahren (wie AES) in der Richtlinie BSI TR-02102-1 bereits mindestens 128 Bit lange Schlüssel empfohlen. Für RSA gelten demnach ab 2017 sogar erst Schlüssellängen ab 3000 Bit als sicher.

Sie werden jetzt vielleicht einwenden, dass Ihre privaten Verbindungen beim Online-Banking oder für kurze Nachrichten auf einem sozialen Netzwerk immer nur kurze Zeit bestehen und daher ein kurzer Schlüssel ausreicht. Das stimmt: Während der Sitzung haben Hacker sicherlich auch bei einem „schlechten“, z. B. 128 Bit langen Schlüssel kaum eine Chance, zu intervenieren. Was hindert sie allerdings daran, die Datenpakete aufzuzeichnen und dann in aller Ruhe zu entschlüsseln, um etwa Ihren PIN-Code oder Ihre Kreditkartennummer auszuspionieren? Diese ändern sich bestimmt nicht täglich, oder? Tatsächlich werden Sie aber von verschiedenen Diensten aus genau diesem Grund aufgefordert, zumindest jährlich Ihre Passworte zu ändern.

Eine ebenfalls berechtigte Frage wäre, warum man dann nicht gleich auf Nummer sicher geht und z. B. gleich 10.000 Bit Schlüssellänge nutzt. Die Antwort können Sie sich selbst geben, wenn Sie an Ihre Experimente mit dem ASYM-Codierer denken: Längere Schlüssel bedeuten auch für den berechtigten Anwender einen deutlich höheren Aufwand. Daher werden die minimalen, als sicher eingestuften Schlüssellängen immer wieder neu aufgrund der aktuellen Bewertung der Möglichkeiten von Hackern empfohlen.

Um Ihnen übrigens die Möglichkeit zu geben, auch im Rollenspiel noch ein weiteres, asymmetrisches Verfahren auszuprobieren, habe ich Ihnen als Alternative für die doch etwas umständliche ASYM-Codierscheibe in den Kopiervorlagen 12.K5 und 12.K6 noch zwei Codetafeln zur Verfügung gestellt, die ebenfalls wechselweise zur Ver- und Entschlüsselung dienen. Sie können eine davon als öffentlichen Schlüssel nutzen und beliebig oft kopieren, um sie an Freunde zu verteilen. Die andere behalten Sie für sich. Verschlüsselt werden immer Buchstabenpaare, von denen der erste Buchstabe die Zeile, der zweite die Spalte angibt. Dort steht der Code für das Paar. Selbstverständlich können Sie auch diese Tabellen umgekehrt nutzen und jedes Paar eines verschlüsselten Textes in der „falschen“ Tabelle suchen. Das ist jedoch seht zeitraubend und dauert viel länger. Bei den „echten“ asymmetrischen Verfahren ist das genauso: Prinzipiell kann man auch hier einen Text wieder mit dem gleichen Schlüssel entziffern – allerdings dauert das bei guten Verschlüsselungsverfahren so lange, dass niemand darauf warten möchte oder kann.

Leider ist es inzwischen gar nicht mehr so selbstverständlich, auf einfache und übersichtliche Art und Weise die aktuell von einer Webseite verwendeten Schlüssellängen herauszufinden. Probieren Sie das selbst aus! Meistens finden Sie irgendwo ein

Schlüssel- oder Schlosssymbol, über das Sie weitere Informationen abrufen können. Abbildung 12.27 zeigt die Fenster unterschiedlicher Internet-Browser mit den entsprechenden Informationen.

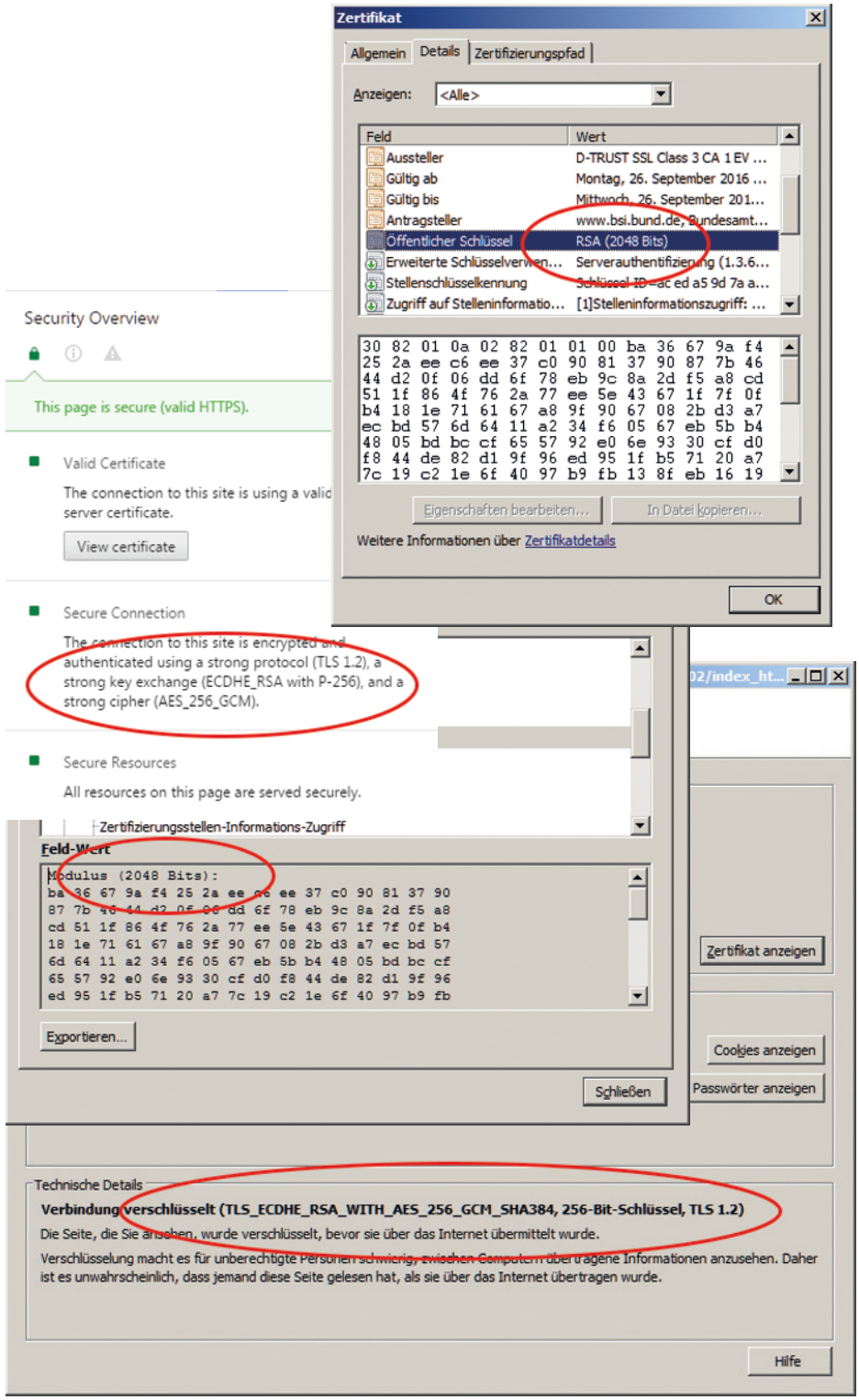


Abbildung 12.27
Hinweise auf verwendete Schlüssellängen und Verschlüsselungsverfahren von Internet-Browsern

RSA

In der tatsächlichen Praxis spielt unter den eingesetzten asymmetrischen Verfahren das mit dem Namen „RSA“ die größte Rolle, weshalb ich hier nur einen ganz kurzen Einblick in die Grundprinzipien geben möchte.

Das RSA-Verfahren ist nach den Entwicklern Ronald Rivest, Adi Shamir und Leonard Adleman benannt. Im Wesentlichen beruht es darauf, dass es ziemlich aufwendig ist, eine große Zahl in ihre Primfaktoren zu zerlegen.

Auch das können Sie experimentell nachvollziehen: Ich stelle folgende Behauptung auf: „Die Zahl 50.065.021 ist das Produkt zweier Primzahlen.“ Versuchen Sie herauszufinden, ob ich recht habe, und wenn ja, welche Primzahlen das sind.

Prinzipiell gibt es hier eigentlich nur die Möglichkeit, die Zahl durch alle möglichen Primzahlen zu dividieren, bis ein glattes Ergebnis herauskommt. Dann hat man die beiden Multiplikanden gefunden. Im Beispiel gilt $50.065.021 = 5.003 \cdot 10.007$.

Selbstverständlich muss man sehr viel größere Zahlen nehmen, damit das Verfahren auch gegen einen Computer Bestand hat.

Bei RSA werden aus zwei zufällig bestimmten, sehr großen Primzahlen nach einem mathematischen Verfahren zwei Schlüssel abgeleitet, von denen man einen als öffentlichen und den anderen als privaten Schlüssel deklariert. Nach Vernichtung der ursprünglichen Primzahlen und aller Zwischenberechnungen ist es sehr schwer, aus dem öffentlichen Schlüssel den privaten zu errechnen (ungefähr so schwer wie die Primfaktorzerlegung der großen Zahl). Genau das ist einer der wesentlichen Faktoren für die Sicherheit des Verfahrens.

Der Quantencomputer ist der Alptraum jedes Kryptographen und der Traum jedes Kryptoanalytikers. Mit seiner Hilfe kann man Zahlen in sehr kurzer Zeit in ihre Primfaktoren zerlegen und auf diese Weise das RSA-Verfahren knacken. Der Quantencomputer nutzt den quantenmechanischen Zustand der „Superposition“ aus – eine Speicherstelle hat danach nicht einen bestimmten Wert, sondern alle Werte gleichzeitig mit bestimmten Wahrscheinlichkeiten. Die Berechnung besteht darin, dass die Wahrscheinlichkeiten sich schnell zugunsten des Ergebnisses verändern. Einen Haken hat die Sache mit dem Quantencomputer allerdings: Es konnte bisher niemand einen solchen bauen, der für die Lösung echter Aufgaben gerüstet ist …

Hier liegt übrigens auch die größte Schwachstelle der ASYM-Codierscheibe aus unseren Experimenten: Sie können zu jedem Schlüssel recht einfach den „Gegenschlüssel“ erzeugen. Ein Spitzbube, der dieses Prinzip erst einmal herausgeknobelt hat, ist dann in der Lage, eine Nachricht auch nur unter Verwendung des öffentlichen Schlüssels zu entziffern. Im Gegensatz zu RSA beruht der ASYM-Code also nur auf Verschleierung des Prinzips.

Jeder Betreiber einer Internet-Seite, der seinen Kunden die sichere Datenübermittlung erlauben möchte, muss sich ein RSA-Schlüsselpaar generieren. Den privaten Schlüssel behält er für sich, den öffentlichen verbreitet er im Internet. Im Experiment haben wir festgestellt, dass dies allerdings noch nicht ausreicht, denn auf diese Weise könnte der Kunde trotzdem einen falschen öffentlichen Schlüssel untergeschoben bekommen. Irgendwie muss er die Korrektheit des Schlüssels überprüfen!

Das passiert tatsächlich, indem der Serverbetreiber seinen öffentlichen Schlüssel bei einer Certification Authority (CA) hinterlegt und ihn dort „signieren“ lässt. Was bedeutet dieses „Signieren“? Im Prinzip steckt die CA den öffentlichen Schlüssel zusammen mit dem Namen des Eigentümers, einer Gültigkeitsdauer und anderen Informationen in einen neuen Datensatz. Diesen codiert sie dann mit ihrem eigenen privaten Schlüssel und gibt ihn an jeden weiter, der ihn haben möchte: Das Ganze nennt sich „Zertifikat“ (siehe Abbildung 12.28). Nur der Vollständigkeit halber sei erwähnt, dass diese Beschreibung eine etwas vereinfachte Version des Zertifikats darstellt.

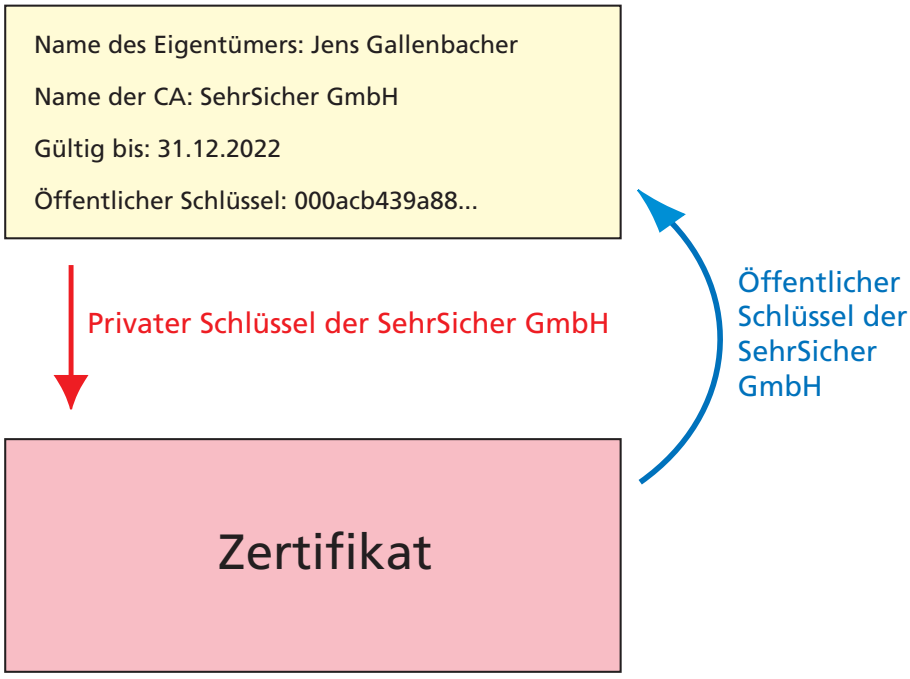


Abbildung 12.28
Zertifikatserzeugung und -benutzung

Ein Zertifikat kann von jedem gelesen werden, der im Besitz des öffentlichen Schlüssels der CA ist (also prinzipiell tatsächlich von jedem). Wenn sich das Zertifikat allerdings mit dem öffentlichen Schlüssel der CA entziffern lässt, muss es mit dem privaten Schlüssel codiert worden sein. Da wir annehmen, dass nur die CA selbst im Besitz ihres privaten Schlüssels ist, können wir dann davon ausgehen, dass das Zertifikat echt ist, und seinem Inhalt trauen.

Auf diese Weise ist egal, ob der Kunde das Zertifikat wirklich von der Certification Authority bekommt (wie in unserem Experiment) oder von irgendeiner anderen Stelle, zum Beispiel dem Anbieter selbst. Er kann die Korrektheit auf jeden Fall mit dem öffentlichen Schlüssel der CA selbst überprüfen.

Auf entsprechende Weise funktioniert übrigens auch die sogenannte „digitale Unterschrift". Die Echtheit des Zertifikats wird im obigen Beispiel sichergestellt, indem es von der CA mit ihrem privaten Schlüssel codiert wurde. Auf diese Weise kann aber jeder, der im Besitz eines privaten Schlüssels ist, eine beliebige Datenquelle – zum Beispiel eine E-Mail – unterschreiben: Er verschlüsselt diese mit seinem privaten Schlüssel. Jeder, der die Nachricht abfängt, kann sie decodieren, aber nicht manipulieren, denn er kann sie ja nicht erneut verschlüsseln. Allerdings könnte er die verschlüsselte Nachricht speichern, um sie später nochmals zu senden.

Ein Beispiel hierfür ist eine Überweisung: Sie schicken Ihrer Bank eine signierte E-Mail mit der Überweisung von 1.000 EUR an die Firma „Gauner & Co. KG". Ein Mitarbeiter dieses Syndikats spioniert die Nachricht aus und sendet sie einfach doppelt an die Bank, so dass die Überweisung nun doppelt ausgeführt würde.

Das kann jedoch leicht verhindert werden, indem der Nachricht die genaue Zeit beigefügt wird. Ist diese auf Millisekunden genau, kann die Bank erkennen, dass die zweite eingereichte Überweisung mit der ersten identisch ist, und sie zurückweisen. Ein solcher Zeitstempel ist auch tatsächlich fester Bestandteil einer digital signierten Nachricht. Der Vollständigkeit halber sollte auch an dieser Stelle erwähnt werden,

dass normalerweise nicht die gesamte Nachricht verschlüsselt wird (sie kann ohnehin von jedem gelesen werden), sondern nur eine Art Prüfsumme. Das reicht aus, um die Authentizität zu garantieren.

Wer vertraut wem?

Jetzt ist nur noch eine Erkenntnis des Experiments auszuwerten: Wo bekommt man anfangs den öffentlichen Schlüssel der entsprechenden Certification Authority her? Wie oben bereits angedeutet, besitzen heutige Browser bzw. Betriebssysteme wie Windows bereits die wichtigsten Schlüssel fest eingebaut. Wenn also das Betriebssystem von einem Originalmedium installiert wurde, kann man diesen Schlüsseln im Allgemeinen vertrauen.

Selbstverständlich gibt es auch Zertifizierungsinstanzen, die noch nicht berücksichtigt sind. Diese haben dann aber normalerweise ihrerseits den eigenen öffentlichen Schlüssel wiederum bei einer der bekannten CAs zertifizieren lassen. Benötigt der Computer einen solchen Schlüssel, besorgt er ihn, überprüft ihn und speichert ihn dann als „vertrauenswürdig“ ab. Auf diese Weise vergrößert sich kontinuierlich der Kreis vertrauenswürdiger Kommunikationspartner.

Vielleicht widersprechen Sie, wenn ich hier von „bekannten CAs“ rede, denn die wenigsten Benutzer des Internets haben bisher von „DST“ oder „Equifax“ gehört. „Deutsche Telekom“ oder vielleicht auch „Verisign“ sind da schon verbreiteter. Allerdings ist es für Benutzer auch nicht wichtig, diese Namen zu kennen – nur der Internet-Browser bzw. das Betriebssystem muss die Daten gespeichert haben, und das bezeichne ich in diesem Fall mit „bekannt“.

Vorsicht ist geboten, wenn ein Anbieter nicht das Geld ausgeben wollte, um sich bei einer der großen CAs zu registrieren, und seine eigene Zertifizierungsstelle benutzt. In diesem Fall werden Sie normalerweise vom Browser gefragt, ob Sie die Verbindung trotzdem eingehen möchten. Falls Sie das bejahen, erfolgt eine verschlüsselte Verbindung. Ohne korrektes Zertifikat einer vertrauenswürdigen Stelle können Sie jedoch nie sicher sein, dass nicht irgendein Hacker mithört oder gar die gesamte Kommunikation selbst bestreitet.

Ganz besonders kritisch ist, wenn Sie gefragt werden, ob Sie ein neues, eventuell unsicheres Zertifikat installieren möchten. Damit erweitern Sie das Netz Ihres Vertrauens. Alle dort ausgestellten Zertifikate gelten von diesem Moment an als sicher! Es ist daher ganz besonders wichtig, die Korrektheit zu überprüfen. Da das neue Zertifikat offenbar nicht von einer der bekannten CAs gegengezeichnet ist, müssen Sie sich auf eine andere Weise von der Vertrauenswürdigkeit überzeugen.

Ein wichtiges Mittel hierfür ist der sogenannte „Fingerabdruck“. Es ist so eine Art Prüfcode, der für jeden Schlüssel nahezu eindeutig ist. Ein Anbieter könnte Ihnen einen Brief mit dem Fingerabdruck seines Zertifikats schicken. Einige Zeitschriften veröffentlichen ihren Fingerabdruck im Impressum. Durch den Vergleich des schwarz auf weiß vorliegenden Fingerabdrucks mit dem des Zertifikats können Sie dann die Echtheit prüfen.

Ist also das Internet sicher?

Diese Frage muss sich jeder selbst und für jeden benutzten Dienst wie E-Banking oder soziale Netzwerke oder Bestellplattformen einzeln beantworten. Ich hoffe, Sie sind nach dem Durcharbeiten dieses Kapitels der Antwort ein Stück näher. Ich möchte aber trotzdem nochmals zusammenfassen, wem und was Sie alles vertrauen müssen, um dabei ein sicheres Gefühl zu haben.

Sie vertrauen dem Anbieter im Internet?

Damit meine ich nicht das allgemeine Vertrauen in die guten Absichten – das setze ich mal voraus. Der Anbieter muss aber auch in der Lage sein, Angreifer aus seinem Computersystem fernzuhalten. Wenn hier jemand eindringt, kann er nicht nur direkt alle Kundendaten ausspionieren, er könnte sich auch einfach den privaten Schlüssel des Anbieters aneignen, um dann zum Beispiel Überweisungen oder Bestellungen zu seinen Gunsten zu manipulieren.

Normalerweise können Sie davon ausgehen, dass Banken und größere Internet-Dienstleister sehr viel Energie und Mittel in Abwehrsysteme gegen Hacker stecken. Das ist aber auch keine Garantie, wie die Meldungen über gestohlene Kundendateien beweisen. Außerdem gilt das nicht für alle Anbieter. Wenn man also bei einem Einzelhandelsgeschäft, das über den Büro-PC auch noch im Internet etwas verkauft, ein Kundenkonto anlegt, sollte man dafür nicht das gleiche Passwort wie für das E-Banking nutzen. Hacker versuchen häufig, leicht erbeutete Passworte auch anderweitig auszuprobieren.

Sie vertrauen der Zertifizierungsinstanz?

Die Zertifizierungsinstanz ist der Dreh- und Angelpunkt, wenn es um die sichere Übermittlung der öffentlichen Schlüssel geht. Wenn dort jemand in den Computer eindringt und den privaten Schlüssel stiehlt, kann er falsche Zertifikate ausstellen – zum Beispiel eines, das auf Ihre Bank lautet, aber den Schlüssel des Hackers beinhaltet.

Auch die Zertifizierungsinstanzen gelten heute als sicher. Sie werden staatlich überwacht und bisher wurden zumindest noch keine größeren Fälle von Internetkriminalität entdeckt, die auf einen Einbruch bei einer CA zurückzuführen sind. Falls die Ausspähung eines Schlüssels auffällt, kann dieser übrigens auch gesperrt werden.

Sie vertrauen dem asymmetrischen Verfahren?

Die erste sichere Kommunikation erfolgt mit RSA oder einem auf elliptischen Kurven beruhenden Verfahren. Bisher ist niemand darauf gekommen, wie man aus dem öffentlichen Schlüssel in kürzerer Zeit den privaten Schlüssel errechnen kann. Es ist daher sehr unwahrscheinlich, dass dies gelingen kann. Gleichzeitig existiert allerdings bis heute auch kein mathematischer Beweis dafür, dass ein solches Verfahren nicht existiert!

Allerdings: Ein großer Teil der Finanz- und Geschäftswelt setzt bei vertraulicher Kommunikation momentan auf RSA. Falls sich das Verfahren irgendwann als nicht sicher herausstellen sollte, haben Sie als Kunde ein Problem – ob Sie persönlich E-Banking und andere Dienste genutzt haben oder nicht, spielt dann nur noch eine untergeordnete Rolle.

Es gibt daher meiner Meinung nach keinen Grund, dem RSA-Verfahren zu misstrauen. Allerdings müssen Sie darauf achten, dass die Verbindung mit einer ausreichend großen Schlüssellänge erfolgt.

Sie vertrauen dem symmetrischen Verfahren?

Hier gilt im Prinzip das Gleiche wie für das asymmetrische Verfahren: Bei großen Schlüssellängen kann man von der sicheren Funktion ausgehen. Einen Schlüssel mit 56 Bit Länge kann man demgegenüber mit einem normalen Rechner schon fast in Echtzeit, also noch während der Übermittlung herausrechnen.

Früher hat man über die Schutzwürdigkeit von Informationen nachgedacht, um zum Beispiel das Passwort zu einem Forum nicht so gut zu verschlüsseln wie die PIN des E-Bankings. Heute ist die Rechenleistung auf einem Niveau angekommen, dass man ohne nachzudenken alles in der maximalen Sicherheitsstufe verschlüsseln kann.

Sie vertrauen Ihrem Computer?

Wir nähern uns langsam den relevanteren Fragen: Anders als Banken besitzen Sie sicherlich für den heimischen Computer keine Expertengruppe, die Hacker fernhält.

Wenn es jemandem gelingt, auf Ihrem Computer irgendwelche schädliche Software zu installieren, kann diese zum Beispiel ohne Weiteres die vorgespeicherten öffentlichen Schlüssel der Certification Authorities verändern, ohne dass Sie es mitbekommen. Wenn dann auch noch der Internetverkehr vor der Verschlüsselung an die Adresse des Hackers umgeleitet wird, kann dieser alle Ihre Geheimnummern, Transaktionsnummern, Buchungen usw. mitlesen oder ggf. auch fälschen. Der Verlust von Daten ist also nicht das Schlimmste, was ein Computervirus oder ein sogenannter Trojaner anrichten kann! Gegebenenfalls öffnet er auch die Tür zu Ihrem Bankkonto ...

Leider bieten heutige Betriebssysteme immer noch nur unzureichenden Schutz vor solchen Einflüssen, wenn sie nicht professionell administriert werden. Daher ist es besonders für die Nutzer von Online-Diensten und Internet-Banking wichtig, Sicherheitshinweise ernst zu nehmen und ihr Betriebssystem immer auf dem neuesten Stand zu halten, aktuelle Virenabwehr zu installieren usw.

Sie vertrauen sich selbst?

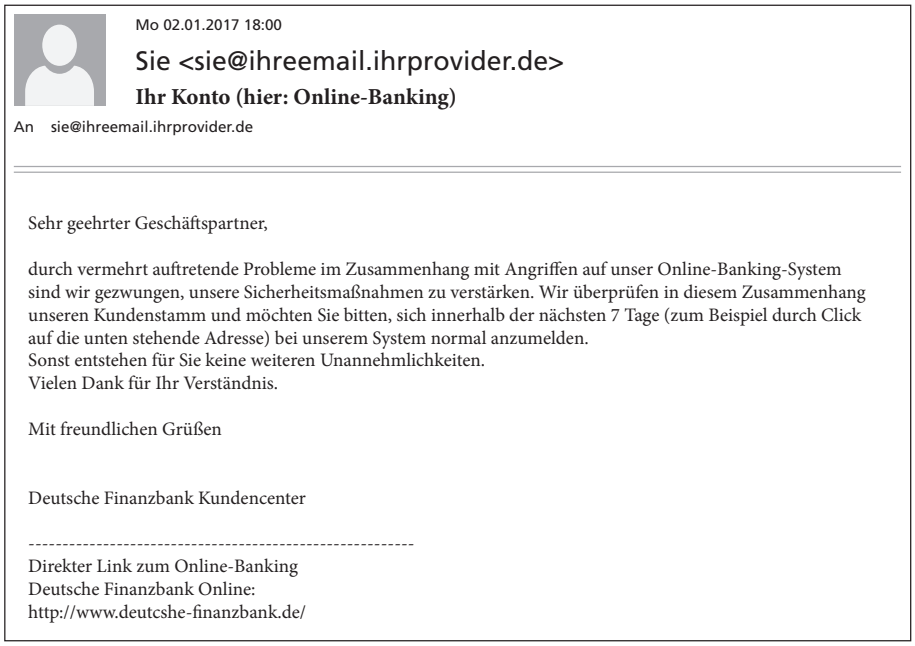
Die weitaus größte Gefahr besteht gar nicht wegen technischer Mängel oder Hackern, die sich sozusagen gewaltsam Einlass in Ihren Computer oder den der Bank verschaffen. Auch bei der Computerkriminalität hat sich etwas durchgesetzt, das am ehesten mit Trickdiebstahl vergleichbar ist.

Was tun Sie, wenn Sie eine E-Mail wie in Abbildung 12.29 bekommen (setzen Sie als Bank Ihre eigene Hausbank ein)?

Würden Sie den Instruktionen folgen? Schließlich ist ja nichts dabei, sich bei seiner Bank nur einzuloggen – die Übertragung ist doch sicher ...

Wenn Sie das tun, indem Sie Ihrem gewohnten Verweis unter „Favoriten“ folgen oder manuell die Adresse in den Browser eingeben, haben Sie da auch völlig recht. Problematisch wird es, wenn man der Bequemlichkeit halber den Verweis aus der E-Mail wählt.

Schauen Sie genau auf die Adresse: Dort ist ein kleiner Schreibfehler versteckt. Jeder kann ganz legal diesen „verdrehten“ Namen auf sich registrieren lassen, ein Zertifi-



Mo 02.01.2017 18:00

Sie <sie@ihreemail.ihrprovider.de>

Ihr Konto (hier: Online-Banking)

An sie@ihreemail.ihrprovider.de

Sehr geehrter Geschäftspartner,

durch vermehrt auftretende Probleme im Zusammenhang mit Angriffen auf unser Online-Banking-System sind wir gezwungen, unsere Sicherheitsmaßnahmen zu verstärken. Wir überprüfen in diesem Zusammenhang unseren Kundenstamm und möchten Sie bitten, sich innerhalb der nächsten 7 Tage (zum Beispiel durch Click auf die unten stehende Adresse) bei unserem System normal anzumelden.
Sonst entstehen für Sie keine weiteren Unannehmlichkeiten.
Vielen Dank für Ihr Verständnis.

Mit freundlichen Grüßen

Deutsche Finanzbank Kundencenter

Direkter Link zum Online-Banking
Deutsche Finanzbank Online:
http://www.deutcshe-finanzbank.de/

Abbildung 12.29
Mail von Ihrer Bank?

kat beantragen und damit einen sicheren WWW-Service betreiben. Kriminell wird es dann, wenn die Ähnlichkeit zum echten Namen ausgenutzt wird und die auf der falschen Adresse erreichbare Seite dem ahnungslosen Internet-Benutzer vorspiegelt, sich tatsächlich auf dem Portal der Deutschen Finanzbank zu befinden.

Wenn Sie hier Ihre Geheimzahlen usw. eingeben, kann das zwar wirklich nur der Besitzer der Webseite lesen – der ist allerdings in diesem Fall ein ausgekochtes Schlitzohr und könnte mit den Daten in Ruhe Ihr Konto plündern!

Sogenannte „Phishing-Mails“, die ähnlich wie die oben abgedruckte aussehen, überschwemmen leider immer noch die Mailboxen von Leuten, die ein Online-Konto führen, und solchen, die das nicht tun – die Betrüger habe ja (hoffentlich) keinen Zugriff auf die Kundendateien. Leider fallen tausende von Internet-Nutzern immer wieder auf solche Mails ihrer Hausbanken, Online-Händler oder anderer Anbieter herein, selbst wenn die Original-Mails oft vor Rechtschreibfehlern und grammatikalischen Grausamkeiten strotzen – so dass eigentlich schon beim Lesen klar sein müsste, dass jemand mit seriösem Geschäftsinteresse eine solche Mail nicht schreibt.

Eine andere Gefahr stellen die doch sehr häufigen Fragen des Browsers dar, ob man nicht dies oder das installieren wolle. Viele Benutzer betätigen da schon aus Gewohnheit den „OK“-Knopf. Falls es sich aber um die Nachfrage handelt, ob ein unsicheres Zertifikat installiert werden dürfe, sollte man das im Normalfall verneinen (wie bereits oben erklärt). Aber Hand aufs Herz: Sind Sie sicher, dass Sie wirklich immer alle Meldungen lesen, bevor Sie mit „OK“ bestätigen?

Die Anbieter versuchen das Ausspähen der Daten möglichst zu verhindern, indem zum Beispiel Zugangsdaten über eine zweite Verbindung verschickt werden – etwa als SMS auf das Mobiltelefon. Wenn diese dann allerdings von Ihnen in einen fremdgesteuerten Computer eingetippt werden, bringt das auch nicht viel.

Die hier aufgeführten Fälle sind nur einige Beispiele für Internet-Kriminalität, bei denen Unwissenheit und Unsicherheit der Benutzer ausgenutzt werden. Spätestens nach der Lektüre dieses Kapitels kann Ihnen das aber ja nichts mehr anhaben!

Übrigens (weil ich das oft nach einem Vortrag über das Thema gefragt werde): Ich selbst nutze Internet-Banking ...

Was man nicht weiß ...

Jetzt haben wir ausführlich darüber geredet, wie man seine Geheimnisse durch Verschlüsselung schützt. Die alten Griechen hatten da noch eine andere Methode: die sogenannte Steganographie. „Steganos" ist griechisch und heißt einfach „versteckt". Die zugrunde liegende Philosophie: Ein Geheimnis, das nicht entdeckt wird, ist sicherer als ein Geheimnis, das ein Unberechtigter nur nicht versteht.

Getreu diesem Motto wurden wichtige Nachrichten einfach vor den Blicken anderer verborgen. Eine Methode war zum Beispiel, einem Sklaven den Kopf zu rasieren und dort die Nachricht zu tätowieren. Nachdem die Haare wieder gewachsen waren, schickte man ihn zum Adressaten.

Heute geht man glücklicherweise nicht mehr ganz so martialisch vor, aber die Kunst des Versteckens ist nicht verloren gegangen, sondern wird noch fleißig in der Computertechnik verwendet. Die Datenflut nimmt immer mehr zu – im Internet gibt es Unmengen an Graphiken und Texten. Auf diese Weise lässt sich immer besser etwas verstecken, denn für jemanden, der nicht genau weiß, wo er suchen muss, wird es bei einem wachsenden Heuhaufen immer schwieriger, die Nadel zu finden.

Nachrichten kann man eigentlich überall verstecken. Um das zu demonstrieren, habe ich eine kleine Nachricht hier im Buch versteckt, und zwar in der Randspalte dieses Kapitels. Suchen Sie oben und unten im Bereich dieser Spalte jeweils einen kleinen, blauen Punkt. Dies sind sozusagen die „Anker" für Ihre Suchschablone aus dem Anhang. Wenn Sie diese fertig gebastelt haben (bitte sehr präzise schneiden), sagt sie Ihnen, wo Sie die einzelnen Buchstaben der Nachricht finden können. Legen Sie die Schablone so an, dass Sie in den Aussparungen oben und unten genau die kleinen blauen Punkte sehen. Danach suchen Sie den kleinen grünen Punkt in den Aussparungen links oder rechts. Ränder ohne Punkte bitte einfach überspringen. Wenn Sie dies für jede Seite gemacht haben, ergibt sich eine kleine Botschaft.

Haben Sie vorher die kleinen Punkte bemerkt oder sich etwas dabei gedacht? Auf folgende Weise kann man in allen Sorten von Dokumenten unbemerkt Botschaften unterbringen:

- In Texten steht in definierten Abständen ab und zu ein Leerzeichen mehr zwischen zwei Wörtern.
- Bei Bildern befinden sich an ganz bestimmten Stellen fast unsichtbar kleine Punkte.
- In Audiodateien (z. B. MP3) gibt es an leisen Stellen zusätzlich eine Millisekunde Pause.

Niemand kann diese Botschaften lesen, wenn er nicht weiß, wo er suchen muss, also das digitale Pendant Ihrer Schablone zu diesem Buch besitzt. So verschicken Sie zum Beispiel ein harmloses Urlaubsfoto an einen Freund und verstecken die eigentliche Nachricht darin.

Selbstverständlich können Sie die Stegano-Schablone auch nutzen, um mit einem blauen und einem grünen Stift geheime Nachrichten in eigenen Dokumenten zu verstecken.

Während sich dieses Verfahren tatsächlich sehr gut zum Verschicken geheimer Botschaften eignet, wird es in der Praxis doch noch häufiger für einen ganz anderen Zweck eingesetzt:

Im Zeitalter des Internets wird die Sicherung geistigen Eigentums immer schwieriger. Wie beweist ein Fotograf, dass ein Bild, welches auf einer Internetseite auftaucht, eigentlich von ihm stammt? Wie stellt ein Filmverleih fest, welcher Vorführer die Vorschau-DVD illegal im Internet verbreitet?

Indem unsichtbare individuelle Markierungen in dem Material untergebracht werden. Eine Botschaft wird versteckt. Das nennt man auch digitales Wasserzeichen, weil es wie das Wasserzeichen im Papier für den normalen Gebrauch unsichtbar bleibt. Nur wer weiß, wonach er suchen muss, kann die Botschaft lesen.

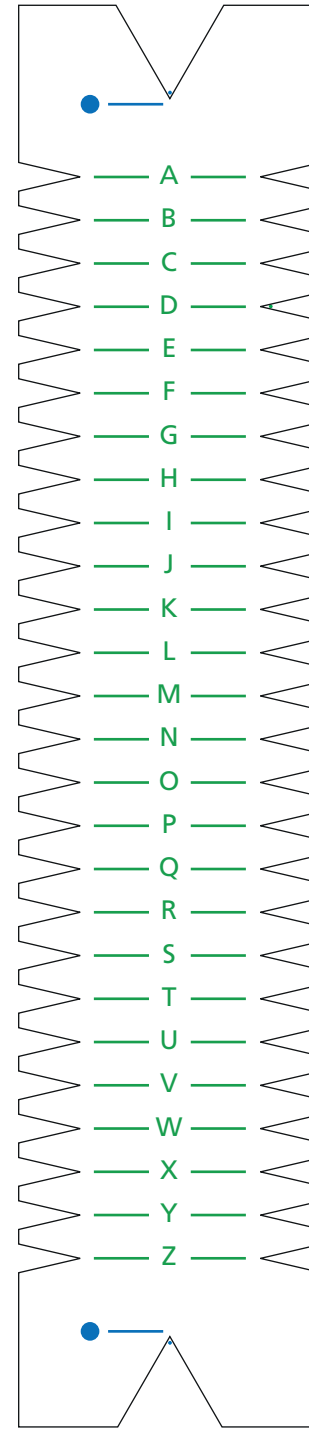
Auf diese Weise kann der Fotograf seine Urheberschaft nachweisen: Nur er kann zum Beispiel unter Aufsicht eines Notars seinen Namen aus dem Bild extrahieren, den er vorher steganographisch dort angebracht hat. Es gibt sogar Dienstleister, die man beauftragen kann, in regelmäßigen Abständen das Internet nach den eigenen, per Wasserzeichen markierten Bildern und anderen Dokumenten zu durchforsten.

Auch ein Filmverleih kann jedes verschickte Exemplar einer Vorschau-DVD personalisieren, also an versteckten Stellen Hinweise anbringen. Wenn dann eine Kopie irgendwo im Netz auftaucht, kann man diese Hinweise analysieren und weiß, wer sie in Umlauf gebracht hat.

Eine wesentliche Frage bleibt: Warum müssen diese Markierungen eigentlich versteckt werden? Eigentlich darf doch jeder lesen, wer der Urheber eines Bildes ist oder um wessen legale Filmkopie es sich bei einer DVD handelt! Das stimmt zwar – der Vorteil unsichtbarer Kennzeichnungen ist allerdings, dass man sie erst finden muss, bevor man sie entfernen kann.

Ein ganzer Forschungszweig beschäftigt sich damit, solche Wasserzeichen so prägnant anzubringen, dass sie nie rückstandsfrei entfernt werden können. So sind die Urheberrechtsinformationen von Bildern auch dann noch vorhanden, wenn man sie ausdruckt und danach mit einem Kopierer vervielfältigt.

Jemand, der sich das Material aneignen möchte und sich daher bemüht, die Wasserzeichen zu entfernen, kann dank Steganographie nie sicher sein, dabei wirklich alle erwischt zu haben.



Stegano-Schablone mit versteckt Buchstaben „D“

Resümee

In diesem Kapitel haben Sie selbst erforschen können, welche Herausforderungen bestehen, um sicher im Internet zu kommunizieren. Wesentlich ist dabei der erstmalige Austausch eines Schlüssels. Asymmetrische Verschlüsselungsverfahren ermöglichen diesen Austausch auch mit Kommunikationspartnern, die man vorher nicht kennt. Eine wichtige Erkenntnis ist, dass auch das nur funktioniert, wenn alle vorher zumindest von einer Stelle auf sichere Art und Weise einen Schlüssel erhalten haben – in diesem Fall den der Certification Authority.

Zusammenfassend habe ich die notwendigen Schritte für einen sicheren Datenaustausch in einer Graphik dargestellt, diesmal mit der Deutschen Finanzbank als Anbieter und Jennifer als Kundin. Abbildung 12.30 zeigt, wie die Bank zunächst zu ihrem

Zertifikat kommt, das die Voraussetzung für die sichere Kommunikation mit einem Kunden ist.

Abbildung 12.31 zeigt schließlich, wie ein symmetrischer Schlüssel ausgetauscht wird, damit Jennifer sicher mit ihrer Bank kommunizieren kann.

Die hier beschriebenen technischen Verfahren helfen jedoch auch nicht weiter, wenn ein Benutzer auf den Internet-Trickbetrug mit gefälschten E-Mails oder Webadressen hereinfällt. Glücklicherweise kann Ihnen das ja jetzt nicht mehr passieren ...

Als kleinen Ausblick möchte ich noch erwähnen, dass neben den beiden hier vorgestellten Sicherheitszielen der Vertraulichkeit und der Authentizität noch einige weitere existieren. So möchte man zum Beispiel elektronische Währungen ähnlich verwenden wie echtes Geld, was das Sicherheitsziel der Anonymität impliziert.

Hinweis (für Skeptiker des ASYM-Codierers):
Vielleicht haben Sie jetzt bereits erkannt, dass die ASYM-Codierscheibe genau wie der Cäsar-Code sehr schwach ist. Man kann die Verschlüsselung leicht knacken. Trotzdem weisen die beiden einfachen Verfahren wichtige Eigenschaften von symmetrischen und asymmetrische Verschlüsselungsmethoden auf und Sie können damit Ihre eigenen Versuche machen, ohne durch die aufwendigen mathematischen Berechnungen der „echten" Verfahren von den dahinter liegenden Prinzipien abgelenkt zu werden.

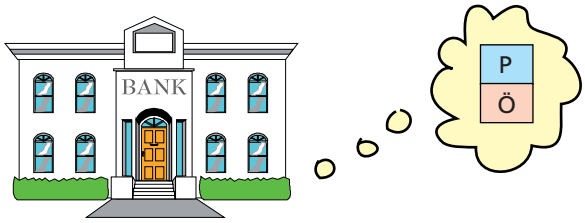
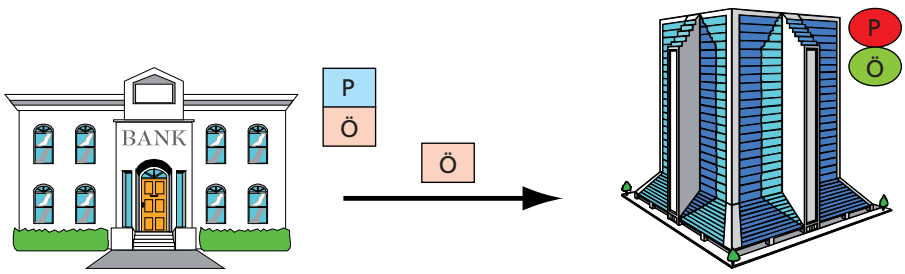
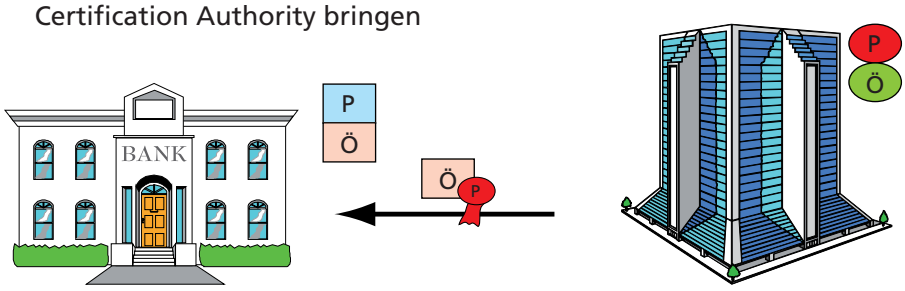


Abbildung 12.30
Die Bank erzeugt ein Schlüsselpaar und lässt es von der CA zertifizieren.

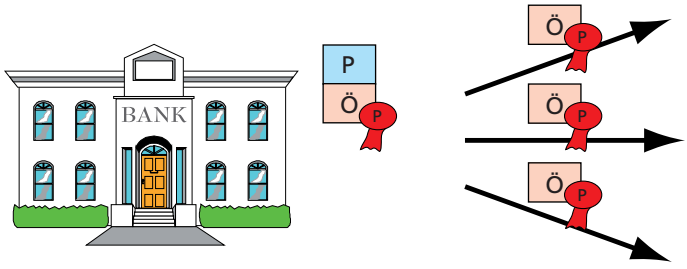
1. Ein Schlüsselpaar generieren



2. Den öffentlichen Schlüssel mit Nachweis der Identität zu einer Certification Authority bringen



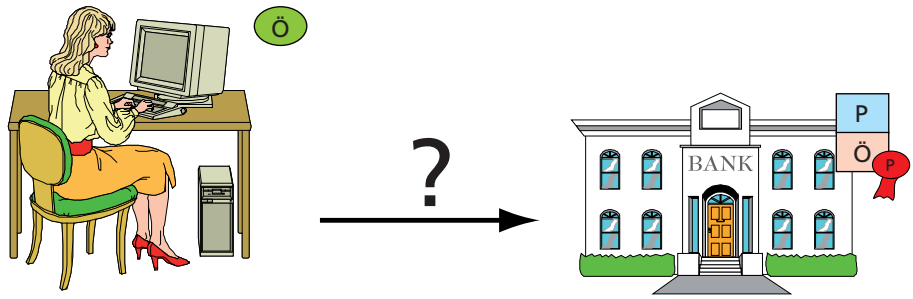
3. Die CA verschlüsselt den öffentlichen Schlüssel der Bank mit eigenem privatem Schlüssel (das ist das Zertifikat).



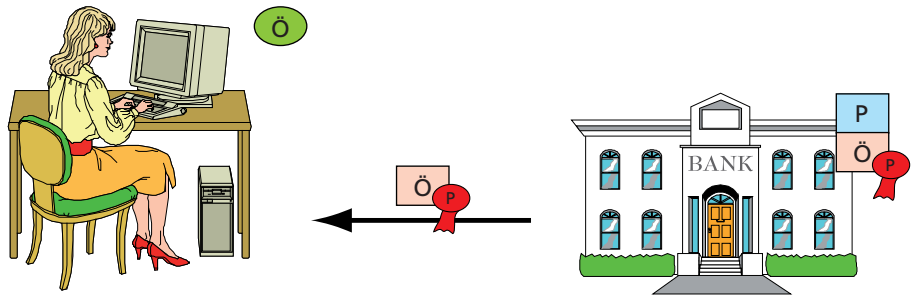
4. Die Bank kann das Zertifikat überall im Internet verbreiten.

Abbildung 12.31
Jennifer tauscht sicher Nachrichten mit der Bank aus.

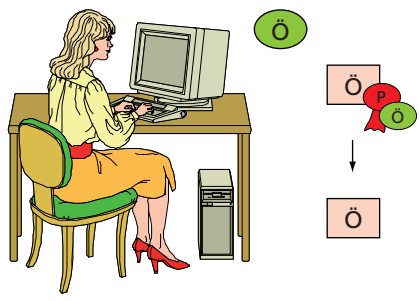
1. Jennifer ist lediglich im Besitz des öffentlichen CA-Schlüssels.



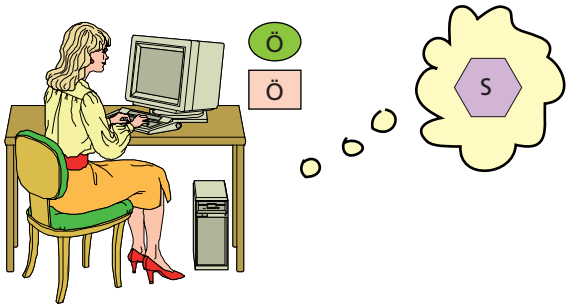
2. Jennifer fragt die Bank (oder auch eine CA) nach dem Bank-Zertifikat.



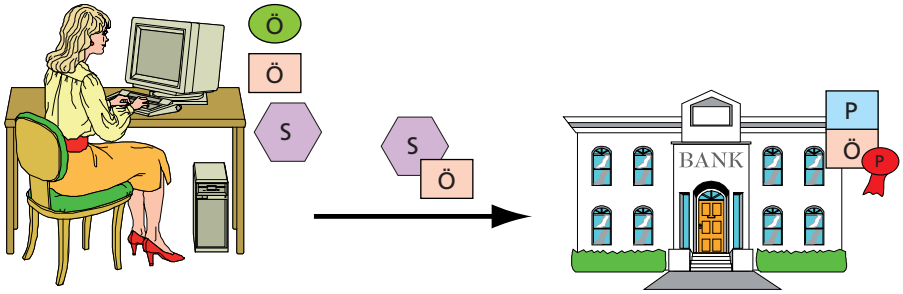
3. Die Bank schickt Jennifer ihr Zertifikat.



4. Jennifer nutzt den öffentlichen Schlüssel der CA, um das Zertifikat zu entschlüsseln und damit die Echtheit zu überprüfen.

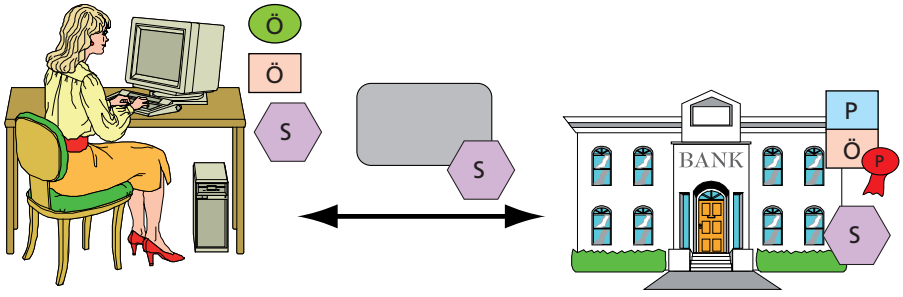


5. Jennifer erzeugt zufällig den Schlüssel für das symmetrische Verfahren.



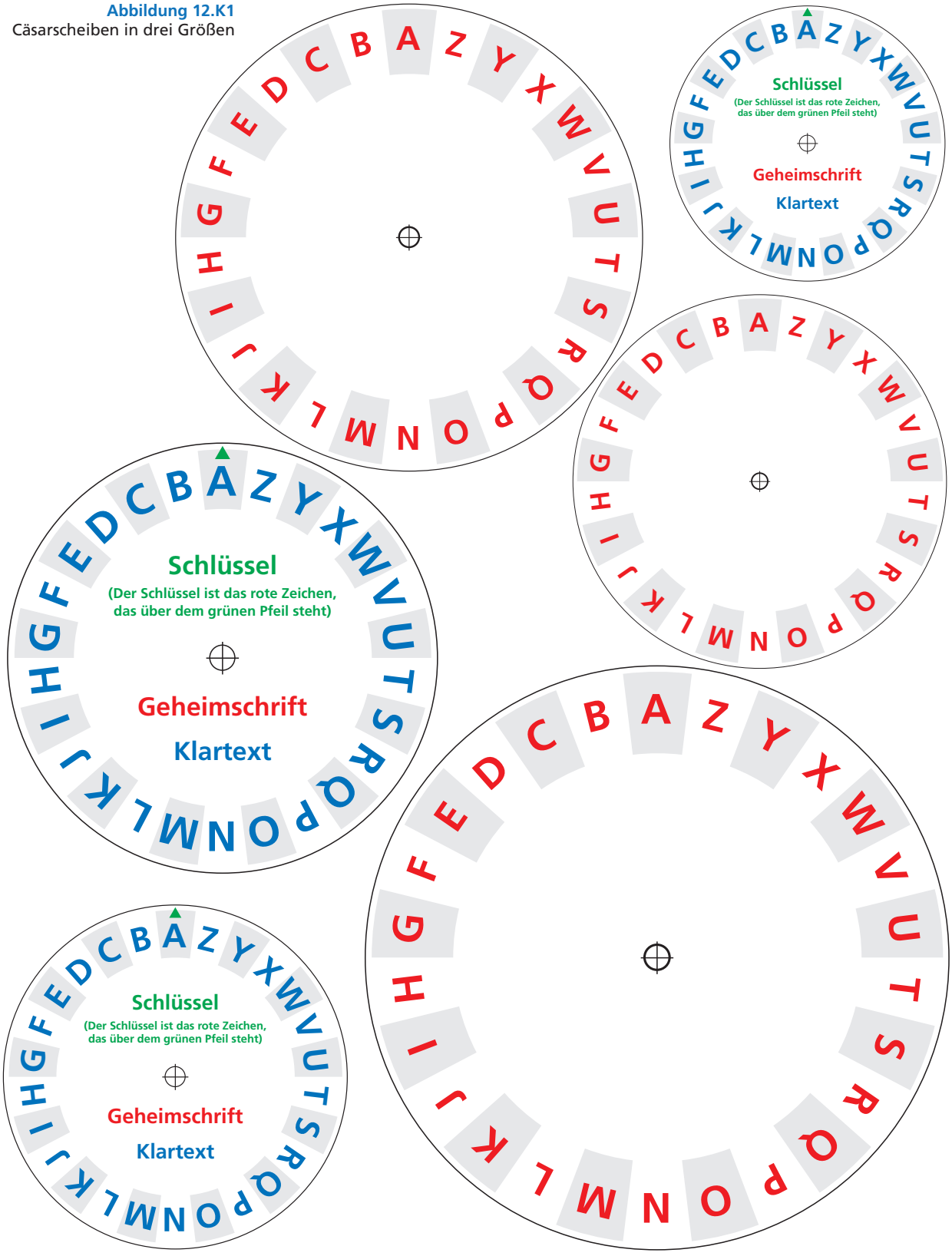
6. Jennifer verschlüsselt den symmetrischen Schlüssel mit dem öffentlichen Schlüssel der Bank und verschickt ihn.

7. Nur die Bank kann die Nachricht mit ihrem privaten Schlüssel entziffern und damit den symmetrischen Schlüssel akquirieren.



8. Da beide Parteien jetzt in Besitz des symmetrischen Schlüssels sind, können sie nun über diesen sicher Nachrichten austauschen.

Abbildung 12.K1
Cäsarscheiben in drei Größen



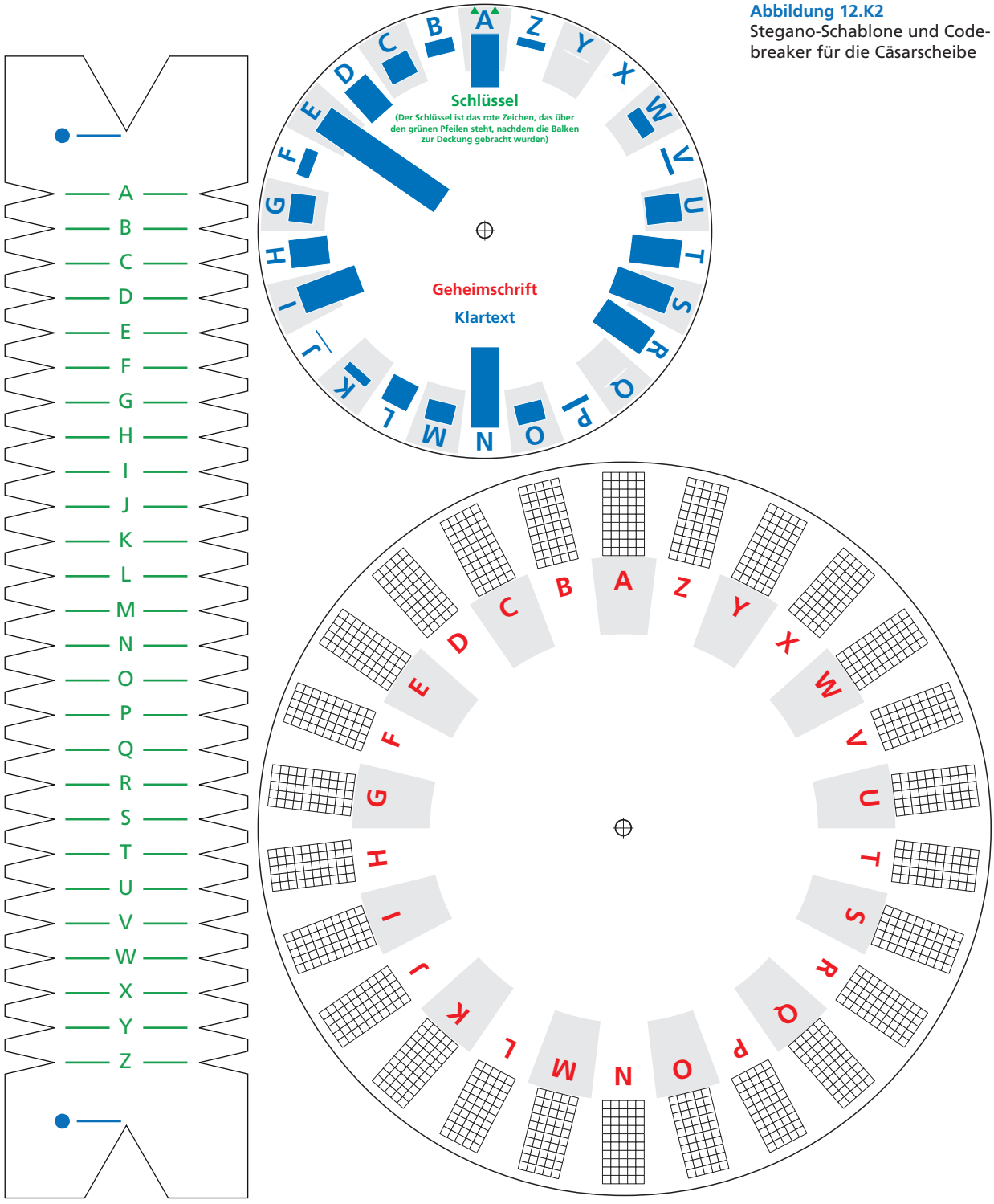
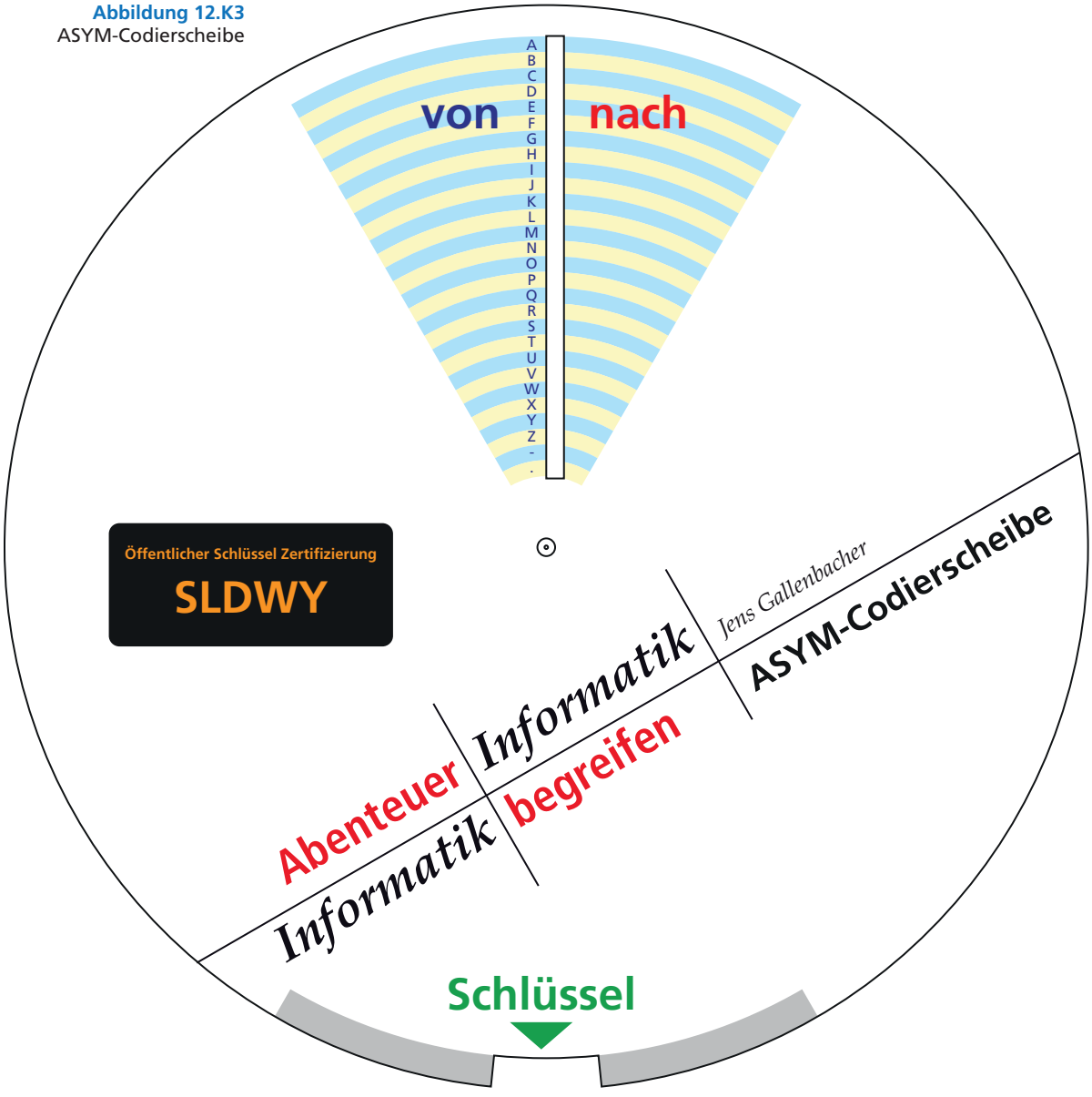


Abbildung 12.K2
Stegano-Schablone und Codebreaker für die Cäsarscheibe

Abbildung 12.K3
ASYM-Codierscheibe

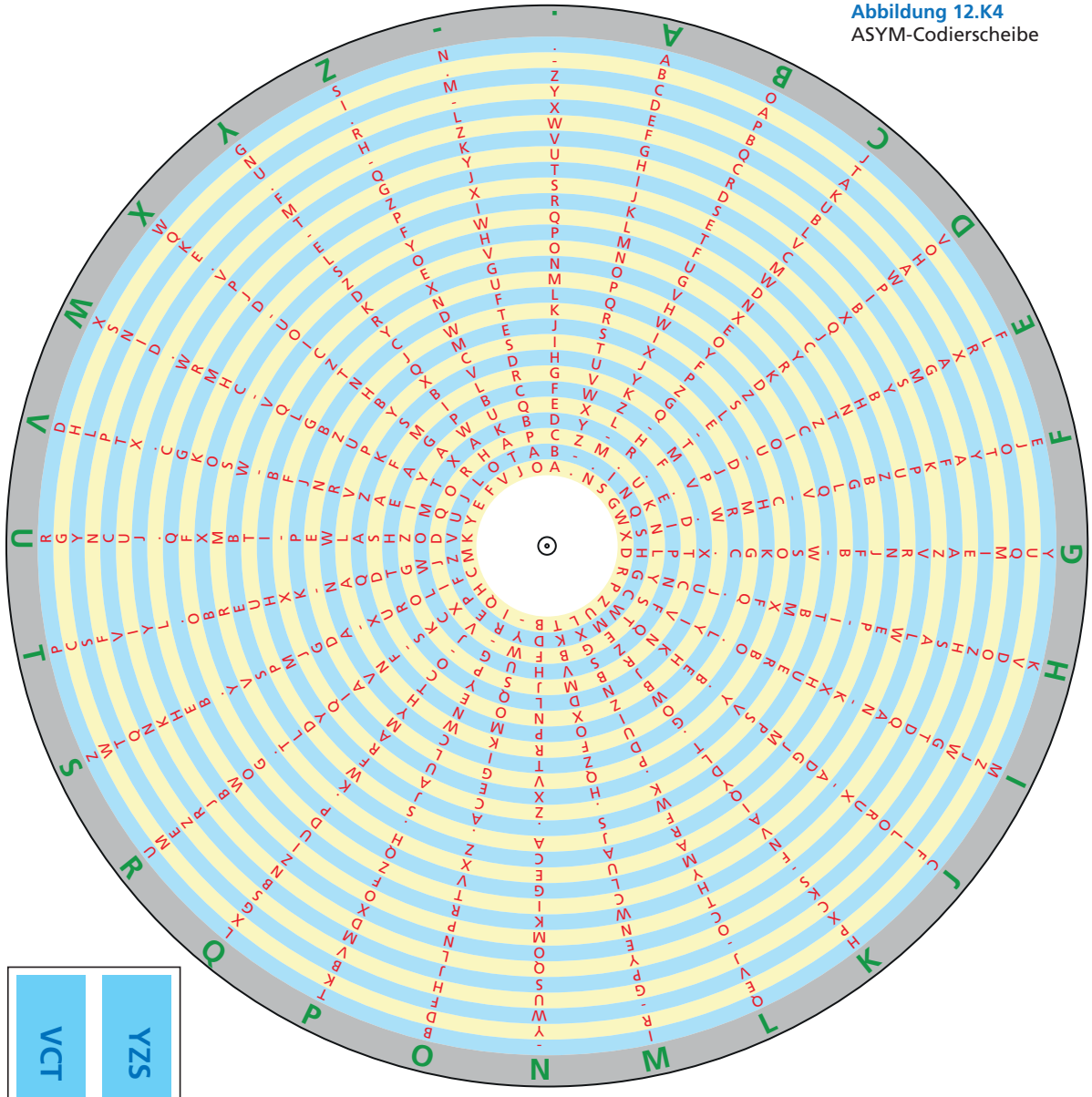


BGF	IKU	WUAD	USQYA
FDB	JUK	-KPG	KTXVP

CEF	PAF	XWZZ	DWJSZ
ZLB	APB	Q-CC	G-ITC

AZS	JGU
PCT	IDK
EFKP	PZG-S
LBUA	ACDWT

Abbildung 12.K4
ASYM-Codierscheibe



VCT	YZS
KPYL	UAVE
TEG-V	SLDWY

QZG	YH-	BDAJ	CGKAT
XCD	VRW	FGPI	ZDUPS

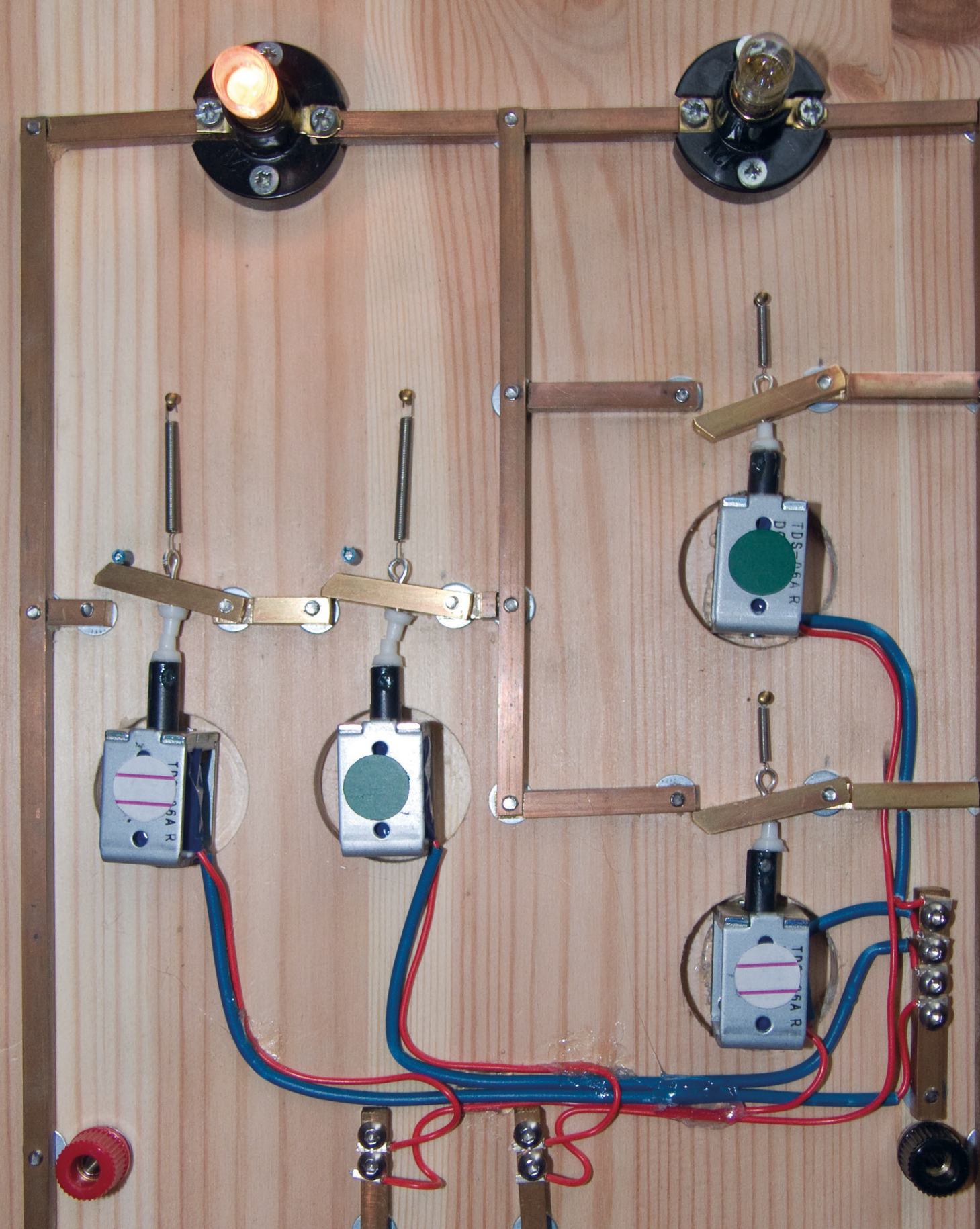
VDB	QWX	WRSS	BDAFA
YGF	X-Q	-HTT	FGPBP

1.↓ 2.→	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
_	HE	_C	MO	YS	OQ	ZF	OL	XO	CJ	SM	JL	HM	YF	QJ	_Q	SX	ZN	CU	FK	HY	VC	QP	CW	AB	NT	SC	G_
A	GK	IO	ZQ	GQ	JV	CD	ZU	LN	XH	PO	NY	DE	IP	_A	VS	KE	PJ	XJ	SP	WV	AO	CM	X_	S_	DQ	AZ	WX
B	LL	IZ	FI	NI	TL	AD	WK	IK	QW	FG	NS	ZK	GM	YX	FL	MM	IM	BQ	KJ	TF	ZD	RL	LT	UI	UO	WR	MF
C	BO	RF	QX	CH	LQ	TH	RH	DA	UB	JO	JU	AT	VH	HK	BY	BW	AL	QF	TD	UP	WN	EU	AQ	MA	GR	YN	VQ
D	PM	XW	VG	YA	_W	IH	OT	KA	TU	QY	IY	RN	V_	YP	MJ	K_	QQ	EJ	PB	GN	GI	ZY	SH	N_	_N	DI	IB
E	XE	ND	SI	EM	VU	KC	XN	WI	BV	RB	QC	EC	PW	OW	OC	KQ	Y_	VZ	NL	BE	HU	YG	CB	EZ	YE	OF	ER
F	GG	LK	YC	AV	EL	MD	__	QE	FF	YR	BK	MW	LB	_V	LW	SO	KM	CV	LM	SV	BT	SD	CX	BC	WU	HQ	DS
G	GL	EV	BU	IX	T_	H_	VJ	RT	ZP	MT	_O	UN	MI	BG	ML	FP	NX	PN	OG	NK	WW	JP	OR	TJ	LS	OV	LI
H	YL	MS	JJ	PE	CI	FR	HI	DZ	UC	OJ	GO	QT	UJ	DX	IW	EA	HF	MR	D_	QN	BP	AM	FM	YJ	GY	UY	SL
I	KI	UU	TY	OD	JA	TS	ZG	HW	XP	TG	YV	RR	UA	P_	PG	ZM	HA	HH	_F	JH	YH	HO	DP	FE	_R	PT	VL
J	GU	B_	HN	YU	KR	VP	TC	OY	QR	CZ	SS	PV	KB	C_	LU	WB	QV	NZ	US	XU	UR	WM	UG	UZ	XK	EB	WQ
K	VM	PP	TQ	XZ	ZX	YW	XY	RA	NQ	BA	SF	LO	WH	KN	CG	EG	GD	YK	XD	HZ	SQ	VD	BH	_J	EI	RG	CO
L	CP	MG	_D	BX	RD	TV	LA	IE	TX	NC	_Y	W_	WJ	QO	KX	FV	RC	XM	SY	AK	HT	WG	VB	RQ	PC	QD	HB
M	IA	RJ	AH	EN	HV	F_	WE	EF	DL	PI	EH	ST	UD	TW	NE	AE	IN	IF	QM	FW	VK	RV	CK	JX	ZE	LG	IU
N	RP	VY	OS	PA	L_	IG	QI	KW	RK	TR	XS	_X	DC	ZW	ET	LH	EQ	OB	AU	XR	EX	OX	IQ	FS	TE	MX	VN
O	A_	SN	GC	HL	UV	RM	NP	OH	NH	BJ	MH	MN	MC	NF	JG	IV	BN	IL	ED	DN	ZC	UF	JE	DV	PZ	U_	JN
P	JZ	VO	DU	TT	VX	AS	_U	FH	AX	HC	GE	DD	NM	VF	NV	RI	KG	_H	R_	ZS	MY	JQ	FD	NO	KO	CY	SW
Q	_E	KS	ZI	ON	MB	UM	BL	ME	JW	_G	KH	AG	UW	AJ	BF	KZ	KF	XA	IC	ZT	GX	XL	RE	YD	EP	RU	ZZ
R	XI	MQ	YQ	JS	LZ	KL	ZH	PY	GS	GJ	PF	LF	GF	LE	RS	UX	WO	ZO	O_	DR	DT	XB	DG	Z_	ES	LY	YY
S	FX	RO	_L	AF	TO	AW	GV	AR	KU	NW	PH	OK	YO	GW	LD	LC	FT	CT	HJ	TP	FZ	TI	KD	NJ	KT	PX	QZ
T	BB	TA	JC	VE	MZ	Q_	IR	_I	I_	VA	QA	JM	BR	IJ	SB	JK	QK	TN	AI	EO	DO	WY	DW	CE	BD	UL	IT
U	OU	IS	RZ	EW	TB	RY	UQ	SJ	UT	PU	CF	SG	ZB	WA	LJ	JR	TM	CR	FN	NB	VR	LP	FY	SR	II	_B	MU
V	AP	KK	JD	JF	QB	KV	MV	E_	WD	XT	YM	OP	DM	M_	YT	_K	CQ	EK	DH	XQ	OM	QU	_M	NR	JB	GH	FO
W	XC	OA	WC	KP	YB	ZL	HG	VI	NU	EE	OZ	DJ	J_	TZ	HR	OI	AC	GP	ID	WS	QH	PK	DF	BZ	XX	PR	BI
X	HS	ZV	SA	HD	SU	_Z	WF	CC	BM	LX	GZ	RX	HX	_S	QG	PS	VT	NN	FQ	RW	LV	AY	SE	NG	QL	MP	FU
Y	ZJ	CS	BS	TK	FC	XG	MK	GB	UE	JI	ZR	DY	CL	PD	AA	HP	_P	SZ	EY	NA	ZA	WZ	GA	FJ	YZ	DK	LR
Z	CN	FB	FA	OO	YI	PL	DB	VV	_T	QS	JY	WL	WT	UH	XV	OE	SK	KY	UK	VW	CA	PQ	WP	JT	AN	XF	GT

Abbildung 12.K5
Alternativer asymmetrischer Codierer

1. ↓ 2. →	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
_	FF	AM	UY	_A	LB	Q_	IR	QI	PQ	TG	KW	VO	SB	VV	DX	GJ	YP	_N	IX	XM	ZH	PF	FM	DD	NK	LJ	XE
A	O_	YN	_W	WP	BE	MO	SC	QK	MB	TR	QM	LS	CP	HU	ZX	AT	V_	CV	SG	PE	CK	NR	FC	SE	PH	XU	AY
B	JA	KI	T_	FW	TX	ES	QN	GM	KV	WZ	OI	FJ	QF	XH	OP	C_	HT	BQ	TL	YB	FT	GB	EH	CO	LC	CN	WW
C	JM	ZT	EV	XG	AE	TW	UJ	KN	CC	HD	_H	MV	YL	AU	Z_	KZ	L_	VP	UQ	YA	SQ	_Q	FQ	_V	FV	PY	JI
D	HR	CG	ZF	NL	PK	AK	WV	RV	VR	DY	WK	YY	MH	VL	OS	TT	IV	AX	RS	FZ	RT	PB	OW	TV	HM	YK	HG
E	VG	HO	JY	EK	OR	WI	MG	KO	MJ	KX	DQ	VQ	FD	EC	MC	TS	QX	NP	EZ	RX	NN	CU	GA	UC	NT	YR	EW
F	ME	ZB	ZA	YD	PV	IW	FH	BI	PG	BB	YW	_R	BN	HV	UR	VZ	GO	XR	HE	NW	SP	XZ	LO	MS	S_	UV	ST
G	_Z	YV	YG	OB	KP	PJ	RL	F_	VY	DT	RI	A_	G_	BL	DS	HJ	WQ	AC	CX	RH	ZZ	J_	SF	SM	QT	HX	XJ
H	GE	IP	LZ	PI	XC	__	HP	WF	IQ	HF	SR	CM	OC	_K	JB	IU	YO	FY	WN	X_	LT	ET	MD	IG	XL	_S	KS
I	TH	M_	DZ	QR	WR	LG	MQ	NE	DE	UX	TM	BG	OQ	BP	MP	AA	AL	NV	TF	UA	TZ	MZ	OO	HN	GC	DJ	BA
J	WL	ID	VX	TB	VB	OV	VC	ON	IS	YI	HB	TO	_J	TK	OZ	CI	GU	PU	UO	RC	ZW	CJ	AD	QH	MW	ZJ	P_
K	DO	DG	JL	EE	SV	AO	QP	PP	QJ	I_	BR	VA	RE	FP	KM	PX	WC	EO	JD	QA	SX	SH	VE	NG	LN	ZQ	QO
L	ND	LF	FL	SO	SN	RM	RK	MY	NO	GZ	UN	FA	B_	FR	AG	KK	UU	CD	YZ	GX	BV	JN	XT	FN	XI	RY	RD
M	VM	CW	QD	OL	FE	QG	BZ	LA	OJ	GL	DN	YF	GN	BO	OK	_B	XY	RA	HQ	HA	GI	UZ	VF	FK	NY	PT	TD
N	DW	YS	US	LI	EA	MN	OM	XW	OH	BC	SW	GS	ER	PL	XQ	PW	OF	KH	VW	BJ	_X	WH	PN	SI	GP	AJ	JQ
O	RR	WA	NQ	EN	IC	ZO	EY	GR	OG	WO	HI	SK	_F	VT	QC	ZC	VK	_D	GV	NB	DF	U_	GY	EM	NU	JG	WJ
P	IM	NC	DR	LX	YM	HC	RJ	IN	SJ	MI	AP	WU	ZE	D_	GQ	AI	KA	ZU	WY	XO	IY	UI	JK	EL	SY	RG	OX
Q	TE	TJ	VD	EJ	LY	FG	CQ	XN	WT	NF	_M	TP	XX	MR	HS	LM	_U	DP	JH	ZI	HK	VU	JP	BH	CB	DI	SZ
R	PR	KG	EI	LP	LD	QV	CA	KY	CF	PO	MA	NH	BU	OE	DK	SA	N_	LW	IK	RN	GG	QY	MU	XS	XK	UE	UB
S	AW	XB	TN	_Y	FU	XV	KJ	UK	DV	EB	UG	ZP	HZ	_I	OA	FO	AR	KT	UW	JJ	MK	XD	FS	PZ	_O	LR	YQ
T	GD	TA	UD	JF	CR	NX	BS	II	CE	SU	GW	YC	BD	UP	TQ	SD	SS	KB	NI	IE	PC	DH	LE	MM	LH	IB	WM
U	OY	IL	CH	HH	ML	YH	OU	JV	ZM	BW	HL	ZR	TY	QE	GK	BX	CS	UF	JT	JR	UH	IA	OD	QL	RO	HY	JW
V	DL	TI	LV	_T	KU	TC	PM	DB	CL	WG	GF	MT	IZ	K_	NZ	PA	JE	CZ	UT	AN	XP	ED	ZG	ZS	PD	NA	EQ
W	LK	UM	JO	WB	VH	MF	XF	LU	KL	EG	LL	BF	ZK	JU	CT	RP	ZV	JZ	BY	WS	ZL	FX	AS	GT	AZ	TU	YU
X	AV	QQ	RU	W_	KR	E_	ZY	YE	AH	R_	AQ	JX	QU	LQ	EF	_G	IH	VS	NS	NJ	VI	JS	ZN	DA	WX	KF	KC
Y	EP	DC	WD	FB	QW	EX	_L	EU	IT	ZD	HW	KQ	H_	VJ	CY	SL	DM	RB	FI	_C	VN	JC	IJ	KE	BM	RZ	YX
Z	RW	YT	UL	OT	BT	MX	_E	IF	RF	QB	Y_	BK	WE	IO	_P	RQ	GH	AB	YJ	PS	QS	AF	XA	NM	KD	DU	QZ

Abbildung 12.K6
Alternativer asymmetrischer Codierer



13. Rechnen mit Strom

Bisher haben wir sehr viele Prinzipien der Informatik erforscht. Von Computern war dabei kaum die Rede, denn in der Informatik geht es um sehr viel mehr als nur um Computerprogrammierung. Natürlich spielen Computer trotzdem eine große Rolle, und dieses Kapitel soll einen kleinen Einblick geben, wie diese grauen Kisten funktionieren, wie man also mit elektrischem Strom rechnen kann. Einen Vorgeschmack konnten Sie übrigens bereits in Kapitel 5 bekommen, wo wir mit Hilfe Jahrtausende alter Prinzipien die Multiplikation so vereinfacht haben, dass wir mit dem kleinen Einmaleins des Binärsystems auskommen. Diese Kompetenz werden wir nun einsetzen.

Falls Sie irgendwo in diesem Kapitel feststellen sollten, dass Sie mit dem Binärsystem noch nicht ganz Du auf Du sind, blättern Sie bitte einfach zurück zu Kapitel 4 und 5 und führen die dort angegebenen Experimente durch.

Die Anfänge

In Konrad Zuses ersten funktionierenden Computern verrichteten unzählige Relais ihren Dienst. Relais werden bis heute zum Schalten von Licht eingesetzt: Drückt man den Lichtschalter, fließt Strom durch eine Spule, die dadurch magnetisch wird und einen Schalter innerhalb der Spule betätigt. Über den Schalter wird dann der Stromkreis geschlossen, so dass die Lampe mit Strom versorgt wird. Im ersten Moment klingt das wie eine „Was-passiert-dann-Maschine“ aus Kindersendungen: Warum konnte man den Schalter nicht direkt an das Licht anschließen? Vorteil des Relais ist, dass man es von beliebig vielen Schaltern aus ansteuern kann und so zum Beispiel in einem Zimmer das Licht nicht nur von einer Tür aus, sondern auch bequem von allen anderen Türen ein- und ausschalten kann. (Hinweis: Das funktioniert mit einer speziellen Art von Relais, nicht mit dem einfachen, das wir im Folgenden verwenden.)

Abbildung 13.1 zeigt links schematisch einen Stromkreis für eine einfache Lampe mit Schalter. Oben ist der Pluspol, sozusagen ein Reservoir für unbegrenzt viel Strom, unten der Minuspol. Dieser kann praktisch unbegrenzt viel Strom aufnehmen. Wird der

Abbildung 13.1
Lampe mit Schalter links und geschlossener Stromkreis rechts

Schalter geschlossen, kann der Strom von plus nach minus fließen und bringt dabei die Glühbirne zum Leuchten, wie in der Abbildung 13.1 rechts.

Strom versus Wasser

Nun kann man sich dies schlecht vorstellen: Wer hat schon einmal Strom durch eine Kupferleitung fließen sehen? Genau genommen sehen wir immer nur die Auswirkung, also etwa das Licht der Glühbirne, oder fühlen, wie sich die Leitung erwärmt. Glücklicherweise können wir unserer Vorstellung aber wieder mit einer Analogie auf die Sprünge helfen. Strom verhält sich sehr ähnlich zu Wasser, und Wasser kennen wir zur Genüge aus eigener Erfahrung.

Abbildung 13.2 verdeutlicht diese Analogie: Wir gehen davon aus, dass aus dem Wasserreservoir beliebig viel Wasser gezapft werden kann (Stausee) und dieses hinter dem Wasserrad in beliebiger Menge abfließt (z. B. in einen Fluss). Wenn nun der Hahn geöffnet wird, treibt das Wasser ein Rad an, analog zur Glühlampe, die vom Strom betrieben wird.

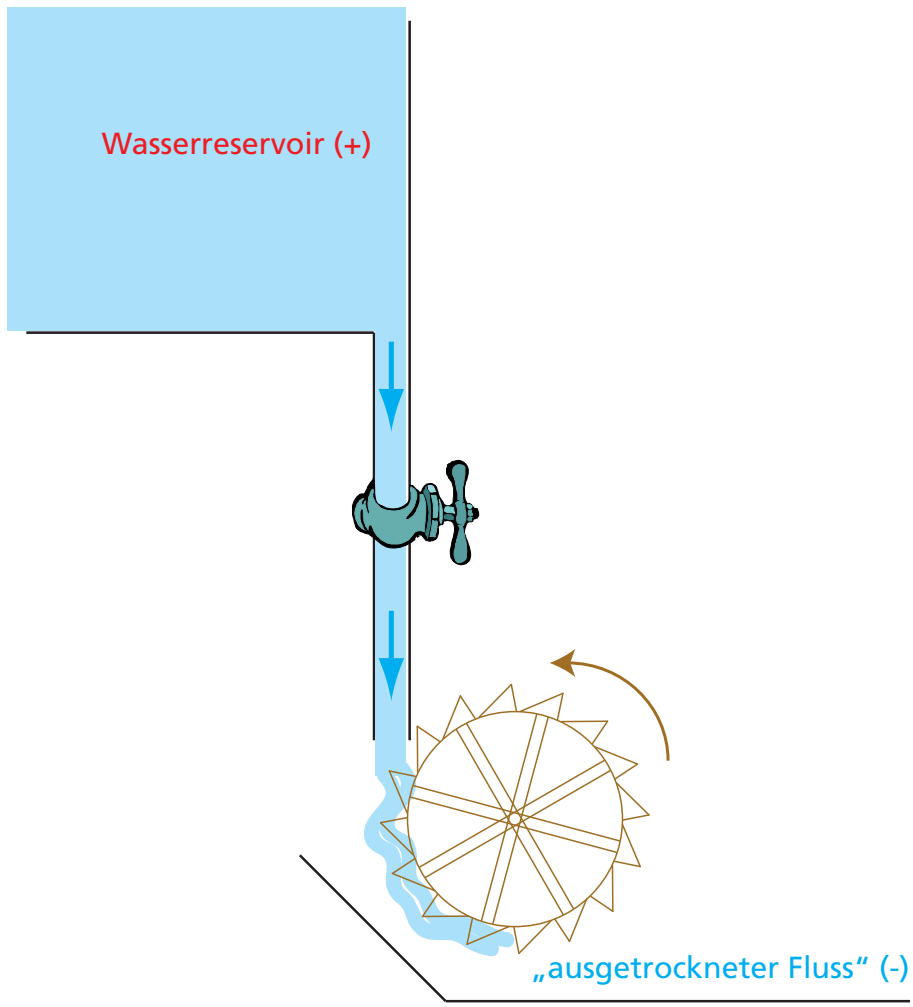


Abbildung 13.2
Wasser treibt ein Mühlrad auf dem Weg in den „ausgetrockneten Fluss“ (-) an.

Der Trick ist dabei, dass sich das Mühlrad nur bewegt, wenn es zwischen dem Wasserreservoir und dem Abfluss angeschlossen ist. Nur auf diese Weise gibt es einen Wasserfluss, der es antreibt. Genau das Gleiche gilt für elektrische Bauelemente wie eine Glühbirne oder ein Relais: Sie funktionieren (leuchten, schalten, ...) nur, wenn sie gleichzeitig an den Pluspol und den Minuspol angeschlossen werden. Die Abbildungen 12.3 bis 12.5 zeigen die Wasser-Analogie zusammen mit dem schematischen Schaltplan der entsprechenden Situation im Stromkreis. Zum besseren Verständnis habe ich dort für Sie plus immer in Rot, minus immer in Blau dargestellt. Ein Bauteil ist aktiv, wenn sowohl eine rote als auch eine blaue Leitung hineingehen.

Von plus nach minus
In den meisten Anwendungen der Elektrotechnik geht man davon aus, dass der Strom vom Pluspol zum Minuspol fließt, so wie das Wasser in unserem Modell. Physikalisch gesehen wandern negative Ladungsträger, die Elektronen, vom Minuspol zum Pluspol, wenn Strom fließt.

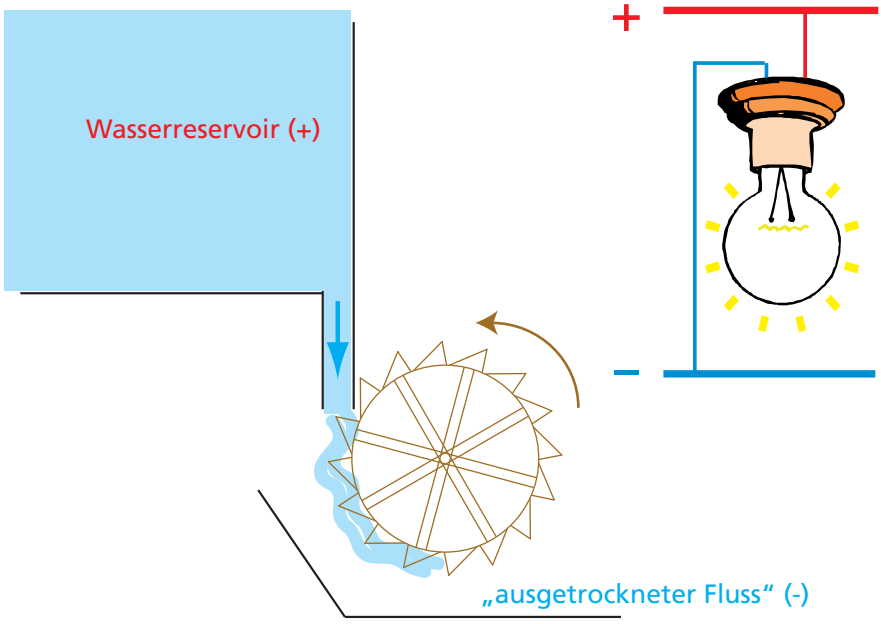


Abbildung 13.3
An beide Pole angeschlossen, brennt die Glühbirne, sobald sich das Mühlrad dreht, wenn es zwischen einem Wasserreservoir und einem Abfluss platziert ist.

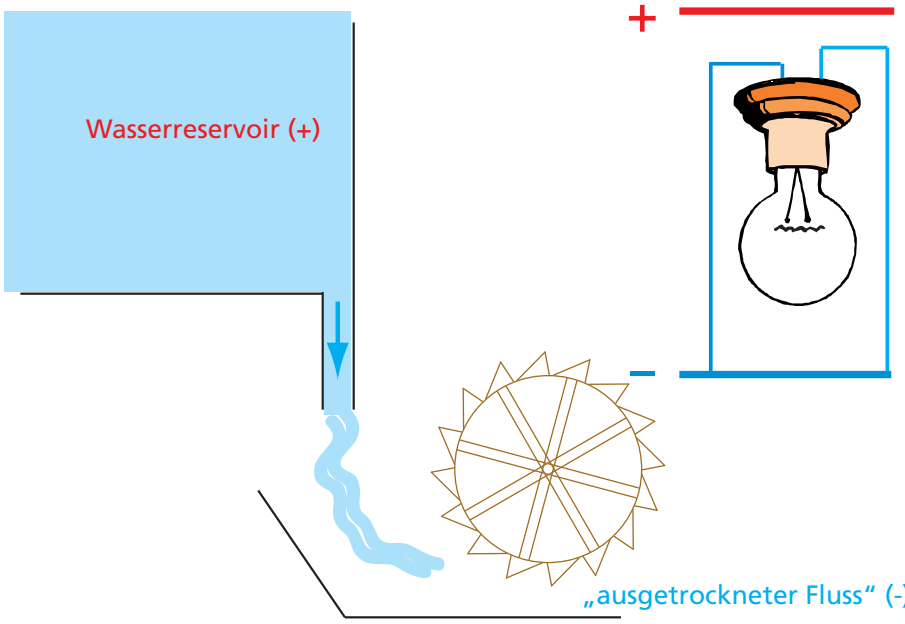


Abbildung 13.4
Hängt das Mühlrad einfach nur in der Luft ohne Wasserkontakt, bleibt es stehen. Eine Lampe, die nur an den Minuspol angeschlossen ist, brennt nicht.

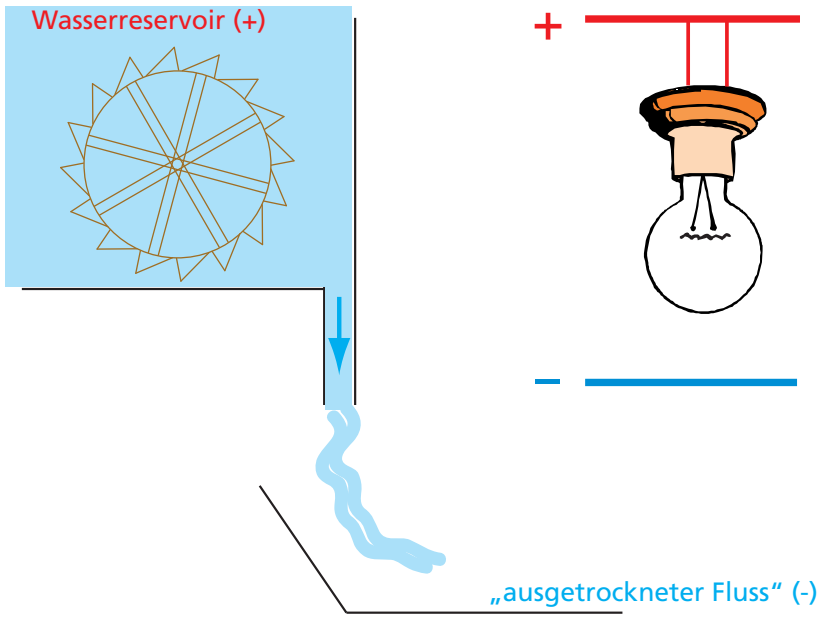
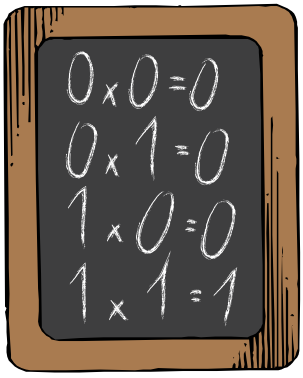


Abbildung 13.5
Aber auch ein völlig in einen Stausee eingetauchtes Mühlrad bewegt sich nicht, die Strömung ist zu gering. Ebenso brennt auch eine Lampe nicht, die lediglich an den Pluspol angeschlossen ist.



Stromkreise können wir uns nun vorstellen, aber wie rechnet man damit? Erinnern wir uns an die Multiplikation aus dem fünften Kapitel: Zumindest das kleine Einmaleins des Binärsystems brauchten wir dort als Grundlage. Wie könnte man eine Schaltung bauen, die alle Zustände der abgebildeten Schultafel realisiert?

Wir definieren dazu 0 als „Aus“ oder „Schalter aus“ und 1 als „An“ oder „Schalter an“. Dann können wir wie in Abbildung 13.6 zwei Schalter in Reihe setzen.

Spielen Sie die gesamte Tafel links in Gedanken durch: Wenn einer der Schalter bzw. Ventile geöffnet sind, kann kein Wasser bzw. Strom fließen und die Lampe bleibt aus, das Mühlrad steht still. Das entspricht den beiden oberen Zeilen. Sobald beide Schalter bzw. Ventile den Durchfluss zulassen, leuchtet die Lampe, das Mühlrad dreht sich, was wir ja als 1 interpretieren.

Sie können das auch mit einer Batterie, zwei Schaltern und einer Glühbirne oder LED nachbauen, und schon haben Sie einen einfachen Binär-Multiplizierer hergestellt – die Grundlage aller Computer. Mehr braucht es nicht!

Na ja – auch wenn das prinzipiell stimmt, ist die Handhabung unseres Multiplizierers noch etwas umständlich: Die Schalter müssen per Hand bedient werden, was dem Gedanken der Automatisierung zugegebenerweise etwas entgegenläuft. Dafür gibt es aber eine Lösung: Angenommen Sie möchten den Durchfluss des Wassers und damit die Bewegung des Mühlrades gemütlich vom Stausee aus steuern. Eine Konstruktion, die den Hahn selbst mit Wasserkraft öffnet, kann dabei helfen.

Abbildung 13.7 zeigt etwas Entsprechendes: Der Hahn unterbricht nun den Wasserfluss in einer „Nebenstrecke“, die auch deutlich dünner sein kann. Ihre einzige Aufgabe ist, durch den Wasserfluss die Sperre in der Hauptleitung aufzuheben. Sobald der Wasserfluss jedoch erneut unterbrochen wird, sorgt die Feder dafür, dass die Sperre wieder schließt. Wir haben auf diese Weise in der Wasser-Analogie ein Relais gebaut.

In gleicher Weise funktioniert ein elektrisches Relais, wie in Abbildung 13.8 dargestellt: Ein Lichtschalter sorgt dafür, dass eine Spule (dargestellt als durchgestrichenes

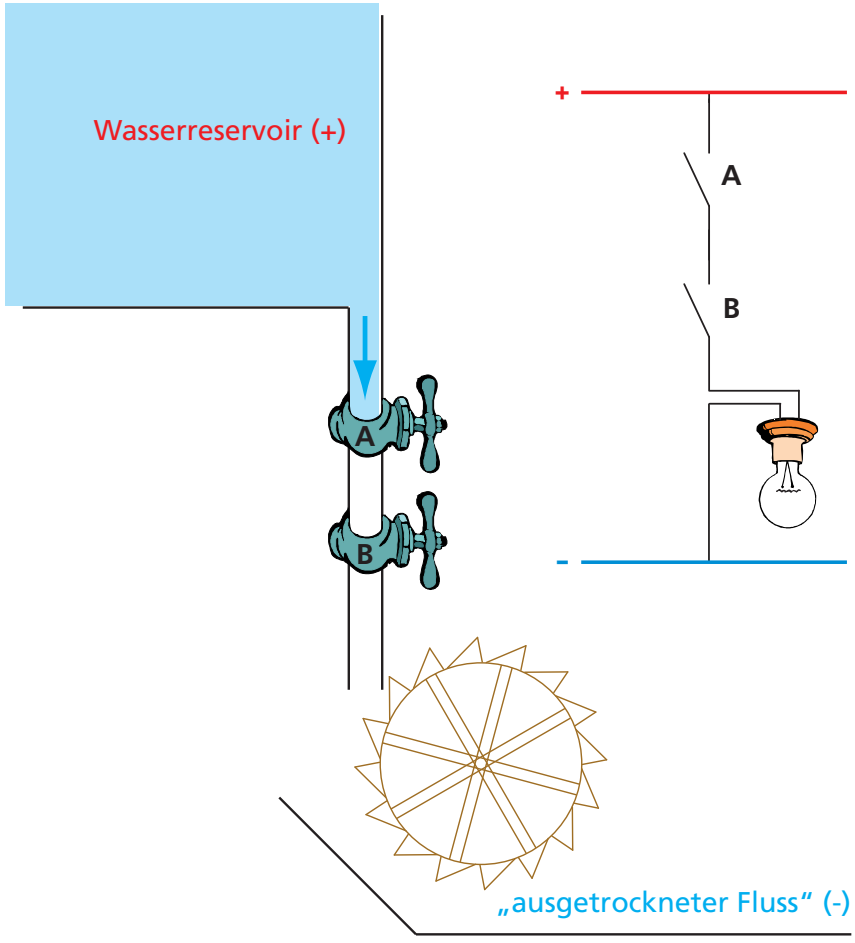


Abbildung 13.6
Einfache binäre Multiplikation mit zwei Schaltern bzw. zwei Ventilen

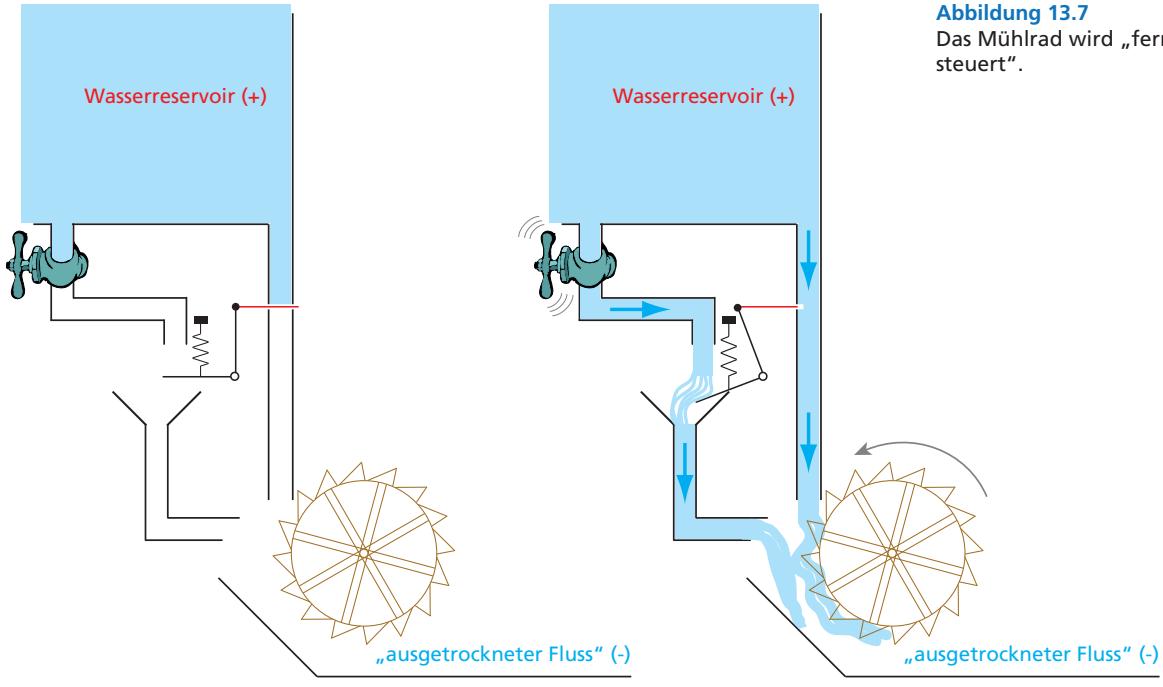
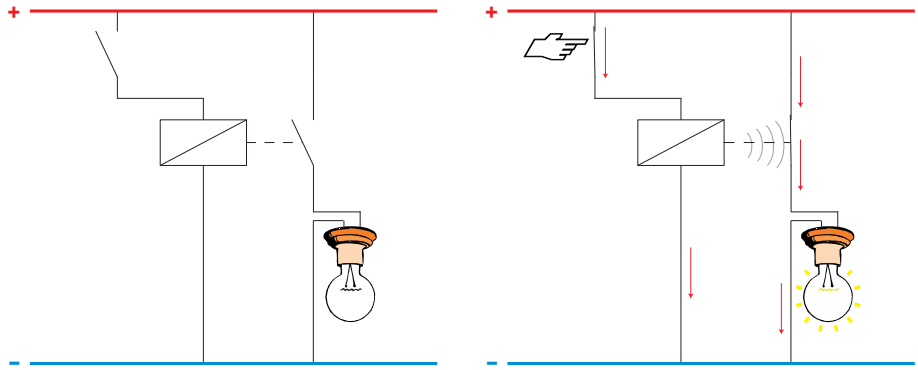


Abbildung 13.7
Das Mühlrad wird „ferngesteuert“.

Abbildung 13.8
Die Lampe brennt, wenn das Relais den Schalter schließt.



Rechteck) mit Strom versorgt wird. Sie betätigt magnetisch einen weiteren Schalter, so dass die Glühlampe leuchten kann.

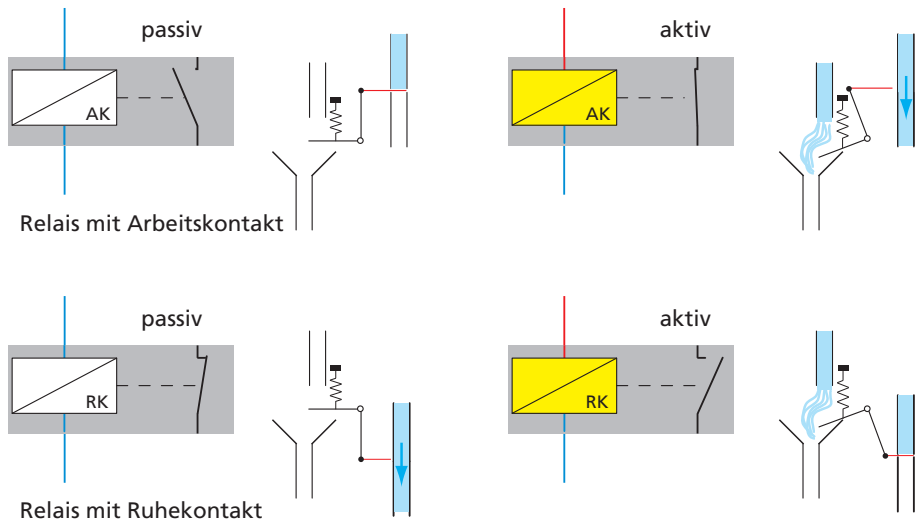
Jetzt brauchen Sie nur noch zu wissen, dass es zwei Arten von Relais gibt: Das in Abbildung 13.8 dargestellte nennt man „Relais mit Arbeitskontakt (AK)“. Wenn das Relais mit Strom versorgt wird, schließt es den Schalter. Beim „Relais mit Ruhekontakt (RK)“ ist der Schalter normalerweise geschlossen. Wenn das Relais mit Strom versorgt wird, unterbricht es den Stromkreis. Abbildung 13.9 stellt diesen Sachverhalt dar und vergleicht die Relais mit entsprechenden „Wasserschaltungen“.

Nachdem jetzt die Grundlagen geklärt sind, wollen wir für den Anfang mit den Relais eine noch einfachere Schaltung als den Multiplizierer analysieren. Nochmal – wir rechnen mit dem Binärsystem und definieren die Ziffern 0 und 1 folgendermaßen:

0 = Verbindung zum Minuspol, 1 = Verbindung zum Pluspol

Analysieren Sie die Schaltung nach Abbildung 13.10! Überlegen Sie hierzu, was passiert, wenn der Eingang mit dem Pluspol verbunden wird (= 1) bzw. wenn er mit dem Minuspol verbunden wird (= 0). Am besten benutzen Sie hierfür Buntstifte: Rot für plus, Blau für minus. Welche Stellung haben daraufhin die Schalter der Relais und welchen Zustand hat die Ausgangsleitung? Als Hilfestellung können Sie annehmen, dass zwischen Ausgang und Minuspol eine Lampe angeschlossen ist, die leuchtet, wenn eine 1 anliegt.

Abbildung 13.9
Relais mit Arbeitskontakt und mit Ruhekontakt sowie die Vergleiche mit Wasser – jeweils im passiven und aktiven Zustand



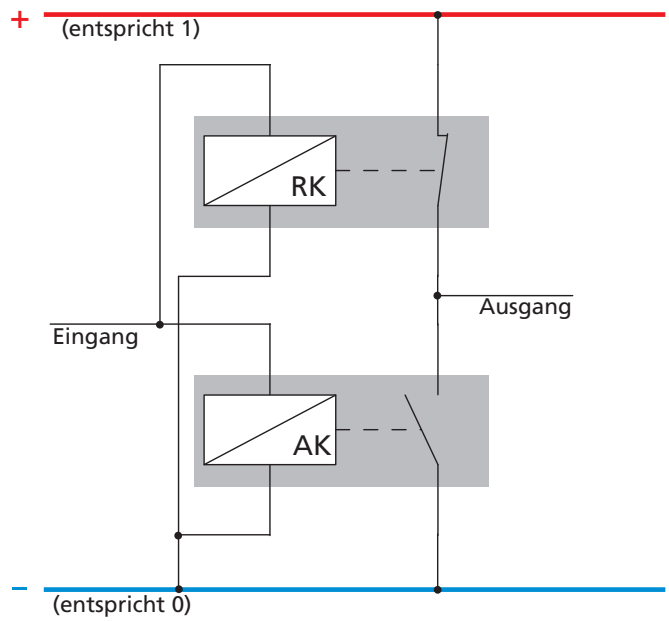


Abbildung 13.10
Unbekannte Schaltung

Übrigens: Wenn sich zwei Leitungen im Schaltplan kreuzen, muss das nicht bedeuten, dass sie eine Verbindung zueinander haben. Diese besteht nur, wenn auf der Kreuzungsstelle ein kleiner Punkt gezeichnet ist.

Sie haben es sicher herausbekommen – diese einfache Schaltung verneint stets: Liegt am Eingang kein Strom an, so bleiben die Relais passiv, und es ändert sich nichts an der Stellung der Schalter. Der Strom kann vom Pluspol zum Ausgang fließen und dort die Lampe betreiben. Abbildung 13.11 auf der nächsten Seite zeigt das. Legt man demgegenüber am Eingang Strom an, so werden die Relais aktiviert, und die Schalter verbinden den Ausgang mit dem Minuspol. Die Lampe leuchtet nicht, wie Abbildung 13.12 veranschaulicht.

Bitte erinnern Sie sich, dass zum Betreiben eines Verbrauchers – also einer Lampe oder eines Relais – immer plus und minus anliegen müssen. Daher haben die aktiven Bauteile immer eine rote und eine blaue Leitung, die zu ihnen führen. Nicht plus (symbolisiert durch das Wasser im Stausee) oder minus (symbolisiert durch den Fluss im Tal) alleine können etwas bewirken, sondern nur die Differenz beider Niveaus.

Interpretieren wir „Strom“ und „kein Strom“ als Ziffern 1 und 0, können wir das Verhalten der analysierten Schaltung in einer sogenannten „Wahrheitstafel“ aufschreiben. Das ist eine Tabelle, die alle möglichen Beschaltungen der Eingänge und deren Auswirkungen auf die Ausgänge enthält.

Eingang	Ausgang
0 (kein Strom)	1 (Strom)
1 (Strom)	0 (kein Strom)

Abbildung 13.11
Kein Strom am Eingang impliziert Strom am Ausgang.

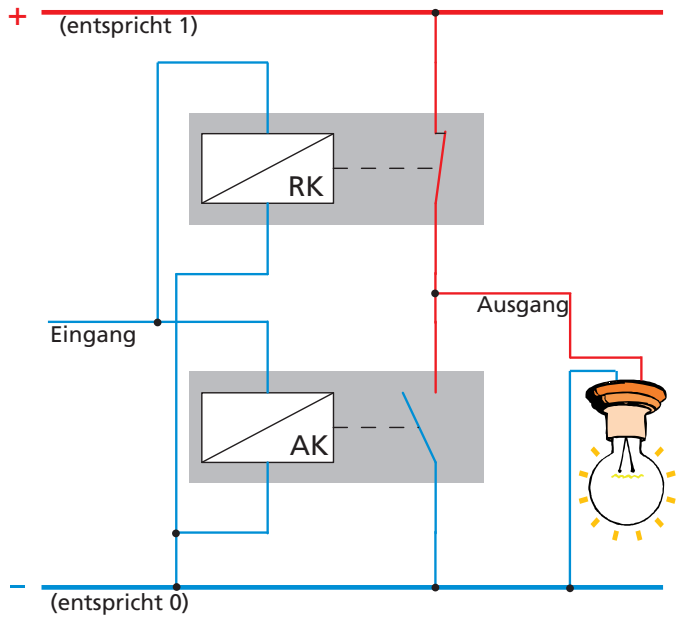
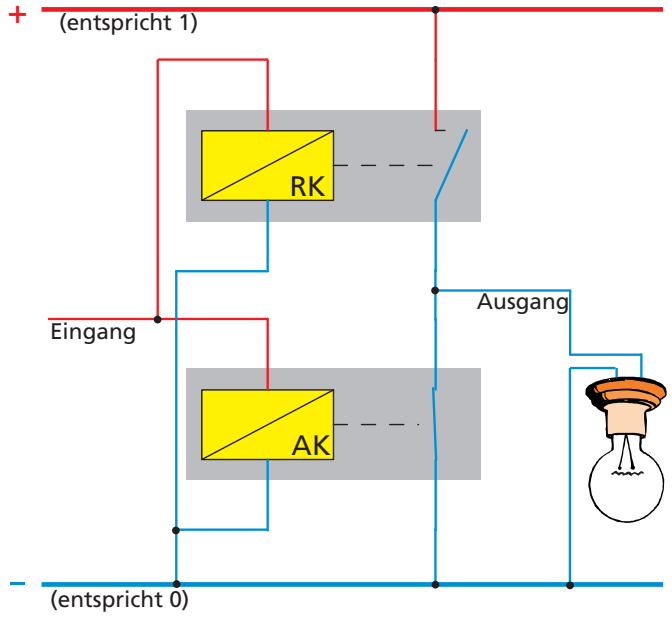


Abbildung 13.12
Liegt am Eingang Strom an, fehlt dieser am Ausgang.



Sie haben soeben erfolgreich automatisiert mit Strom gerechnet! Die „Inversion", also das Umkehren eines Wertes, ist die einfachste Rechenoperation, die ein Computer ausführen kann.

Richtig rechnen

Versuchen Sie es doch gleich mit der nächsten Schaltung! Abbildung 13.13 zeigt eine etwas kompliziertere Anordnung mit vier Relais und zwei Eingängen. Zur Steigerung der Übersichtlichkeit sind an die unteren Anschlüsse der Relais blaue horizontale Bal-

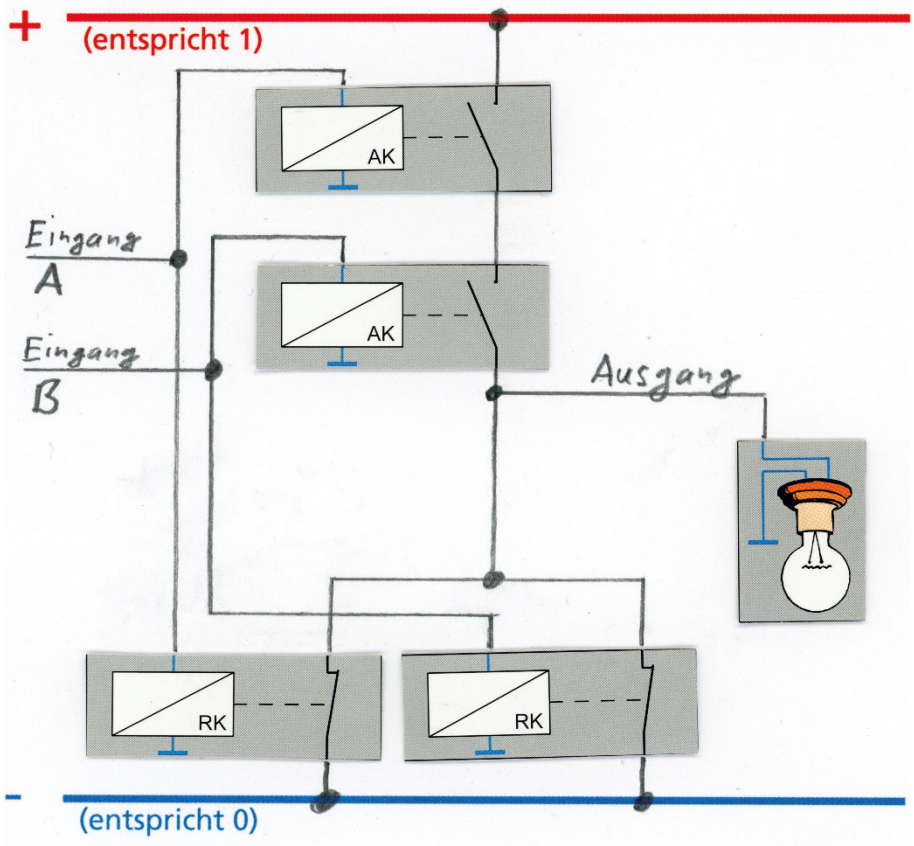


Abbildung 13.13
Unbekannte Schaltung mit vier Relais, gelegt und gezeichnet mit der Vorlage.

ken gemalt. Das ist das Symbol für „Verbindung mit dem Minuspol" – die tatsächliche Leitung braucht nicht mehr gezeichnet zu werden.

Am Ende des Kapitels finden Sie eine Kopiervorlage mit einem „leeren" Schaltplan, in dem lediglich der Plus- und Minuspol vorhanden sind sowie eine Reihe von Relaiskarten. Diese erlauben, das Verhalten von Schaltungen auf einfache Weise experimentell zu erforschen. Malen Sie mit Bleistift die Leiterbahnen und legen die Relaiskarten (zunächst mit ihrer weißen, nicht aktiven Seite) so, dass eine gültige Schaltung entsteht. Fangen Sie mit der Schaltung gemäß Abbildung 13.13 an, die bereits auf diese Weise entstanden ist!

Beachten Sie bitte, dass die Relaiskärtchen wie auch die Kärtchen für die weiteren elektrischen Bauteile wie die Lampe jeweils eine Vorder- und eine Rückseite haben. Während die Bastelbögen nur ausgeschnitten werden müssen, sind die Kärtchen aus den Kopiervorlagen noch zusammenzufalten und zu kleben, wahlweise zusätzlich zu laminieren. Alternativ können Sie die Kärtchen auch einzeln (ohne Rückseite) ausschneiden und im Plan austauschen, statt sie umzudrehen.

Zur Analyse „beschalten" Sie nun die Eingänge mit allen vier möglichen Kombinationen von „plus" und „minus". Das können Sie zum Beispiel mit blauen und roten Buntstiften machen, am besten mit solchen, die ausradierbar sind. Alternativ kann man natürlich für jede neue Beschaltung ein eigenes Blatt verwenden.

Die erste Situation „Eingang A = minus (0), Eingang B = minus (0)" sehen Sie quasi bereits in Abbildung 13.13. Finden Sie auch für die weiteren Situationen heraus, was am Ausgang ankommt, ob dieser also mit plus oder mit minus verbunden ist. Hierzu

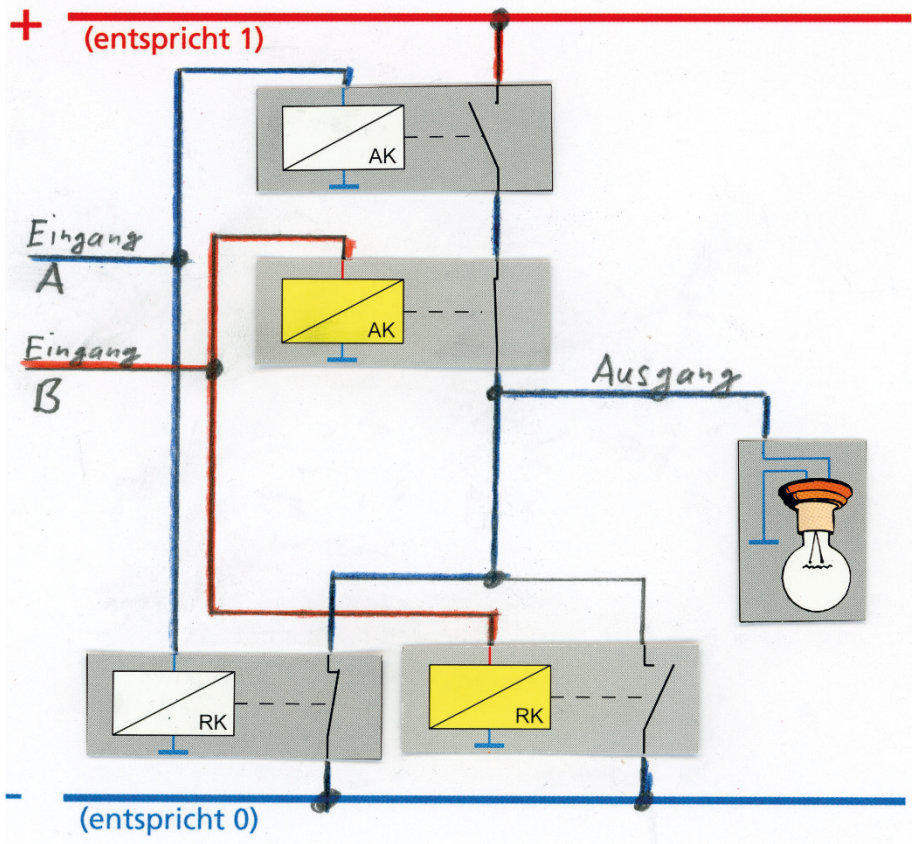
müssen Sie gegebenenfalls die Relaiskärtchen gemäß ihrer tatsächlichen Beschaltung (rot = plus, blau = minus) herumdrehen. Der Ausgang sollte dann anhand der Schalterpositionen eindeutig entweder mit plus oder mit minus verbunden sein. Auch die Lampe können Sie dann auf die entsprechende Position drehen und auf diese Weise das Ergebnis ablesen. Tragen Sie es in die Tabelle ein.

Eingang A	Eingang B	Ausgang
0	0	
0	1	
1	0	
1	1	

Analysieren Sie die Schaltung nach Abbildung 13.13.

In Abbildung 13.14 ist die Analyse des Falls „A = 0, B = 1“ dargestellt: Zwei der Relais schalten (roter Eingang) und die Kärtchen sind daher umgedreht. Der Ausgang ist immer noch mit minus (0) verbunden, die Lampe leuchtet nicht!

Abbildung 13.14
Beschaltung mit A = 0 und B = 1



Auf diese Weise können Sie die gesamte Wahrheitstafel ableiten.

Eingang A	Eingang B	Ausgang
0	0	0
0	1	0
1	0	0
1	1	1

Können Sie den Inhalt der Tafel einer Ihnen bekannten Rechenvorschrift zuordnen? Richtig! Es handelt sich um das bereits bekannte kleine Einmaleins des Binärsystems.

Versuchen Sie nun verschiedene Verschaltungen von Relais auszuprobieren. Wie funktioniert eine Multiplikation mit drei Eingängen? Wie könnte eine Addition aussehen? Gibt es nur zulässige Schaltungen oder sind manche Spielarten tabu?

Vielleicht sind Sie auf Anordnungen gestoßen, bei denen nicht ganz klar war, ob am Ausgang nun plus oder minus anliegt. Als Beispiel sehen Sie in Abbildung 13.15 die einfachste Variante.

Wenn am Eingang das Signal 0 ankommt, die Relais also nicht schalten, ist der Ausgang weder mit plus noch mit minus verbunden. Wenn man wiederum eine Lampe anschließt, leuchtet diese nicht – die gleiche Situation, als sei der Ausgang mit minus verbunden. Prinzipiell also kein Problem, wenn man dieses Verhalten der Schaltung intendiert. Tatsächlich sind solche „undefinierten" Ausgänge erst ungünstig, wenn wir die Relais durch Transistoren ersetzen, wie sie in modernen Chips verwendet werden. Lesen Sie hierzu mehr unter der Rubrik „Was steckt dahinter?".

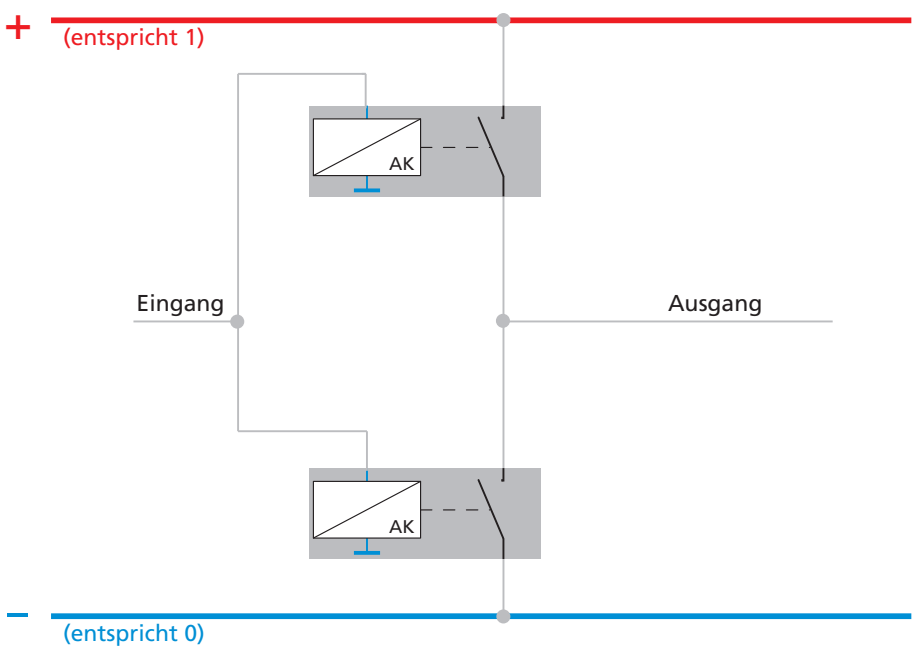


Abbildung 13.15
Ungünstige Verschaltung

Jetzt schließen Sie jedoch den Eingang an den Pluspol an. Beide Relais schalten und wir kommen wiederum zu einer undefinierten Situation: Der Ausgang ist nun sowohl mit plus als auch mit minus verbunden. Das wäre an sich noch nicht so schlimm, aber dadurch sind gleichzeitig der Pluspol und der Minuspol auch direkt miteinander verbunden, und das stellt einen klassischen Kurzschluss dar: Strom kann ungehindert durch die beiden Schalter von plus nach minus fließen.

Stellen Sie sich einen Staudamm vor, bei dem durch einen kleinen Riss Wasser unkontrolliert direkt durch die Staumauer fließen kann. Der Riss wird sich durch die immense Gewalt des Wassers sehr schnell ausweiten, was die Wassermenge steigert und schließlich zum Dammbruch führt. Genau das Gleiche gilt hier für das elektrische Äquivalent: Der Strom fließt direkt von plus nach minus. Dabei wird Wärme produziert, die im günstigsten Fall eine Leitung zum Durchbrennen bringt und die Schaltung zerstört, so dass der Kurzschluss unterbrochen wird. Im ungünstigsten Fall setzt die Schaltung vorher noch umliegende Bauteile in Brand und richtet größeren Schaden an.

Daher sollten Sie bei Ihren Schaltungsentwürfen unbedingt darauf achten, dass jede Leitung jederzeit genau mit einem der Pole plus oder minus und keinesfalls mit beiden, verbunden ist. Das muss für alle Kombinationen von Signalen gelten, die an die Eingangsleitungen angelegt werden.

Machen wir nach diesem kurzen Intermezzo weiter mit der initialen Frage, wie man mit Strom rechnet: Die Multiplikation haben wir erledigt, aber für die Berechnungen wie in den Kapiteln 4 und 5 müssen wir nicht nur multiplizieren, sondern auch addieren. Hier noch einmal die Ergebnisse der binären Addition zweier einstelliger Zahlen:

0 + 0 = 00, 0 + 1 = 01, 1 + 0 = 01, 1 + 1 = 10

Wir benötigen also nicht nur zwei Eingänge, sondern auch zwei Ausgangsleitungen: eine für die erste Ziffer, eine für die zweite. Die Wahrheitstafel der Schaltung muss folgendermaßen aussehen:

Eingang A	Eingang B	Ausgang X	Ausgang Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Einen Tipp habe ich noch: Man kann die Relais auch in mehreren Stufen verschalten. Das bedeutet, der Ausgang eines Relais betätigt nicht direkt die Lampe, sondern ein weiteres Relais, das ggf. wieder ein Relais schaltet usw.

Die nächste Aufgabe ist knifflig – lassen Sie sich nicht entmutigen. Spielen Sie einfach mit den verschiedenen Relais – es gibt mehrere korrekte Lösungen. Die hier später ausgeführte Lösung benötigt zehn Relais: fünf mit Arbeitskontakt und fünf mit Ruhekontakt.

Bauen Sie also nun eine Schaltung, die eine Addition ausführt, so wie in der Wahrheitstafel dargestellt.

Abbildung 13.16 zeigt eine der möglichen Verschaltungen. Sie kommt mit zehn Relais aus. Vollziehen Sie ihre Funktion mit den Relaiskärtchen nach und überprüfen, ob wirklich die Wahrheitstafel von oben herauskommt! Es handelt sich um einen sogenannten Halbaddierer. Lesen Sie weiter, um zu erfahren, was hinter dieser etwas seltsamen Bezeichnung steckt.

Zur weiteren Vereinfachung gehen wir ab jetzt immer davon aus, dass die Eingänge links, die Ausgänge rechts liegen.

Aus diesem einfachen Halbaddierer kann man bereits eine wesentliche Erkenntnis ableiten: Die Schaltungen werden sehr schnell sehr unübersichtlich und komplex. Versuchen Sie einen ganz einfachen Addierer für drei binäre Ziffern zu konstruieren (einen sogenannten Volladdierer), machen Sie sich aber nichts daraus, wenn dies nur schwerlich gelingt!

Dieses Experiment sollte nur dazu dienen, die Komplexität zu begreifen. Es wird Zeit, dem Titel des Buches Rechnung zu tragen. Während dieses Kapitel bisher eher Elektrotechnik vermittelte, stehen wir nun vor einer typischen Anwendung der Informatik: Eine Aufgabenstellung ist zu komplex und unübersichtlich, um sie direkt zu lösen. Kommt Ihnen das bekannt vor?

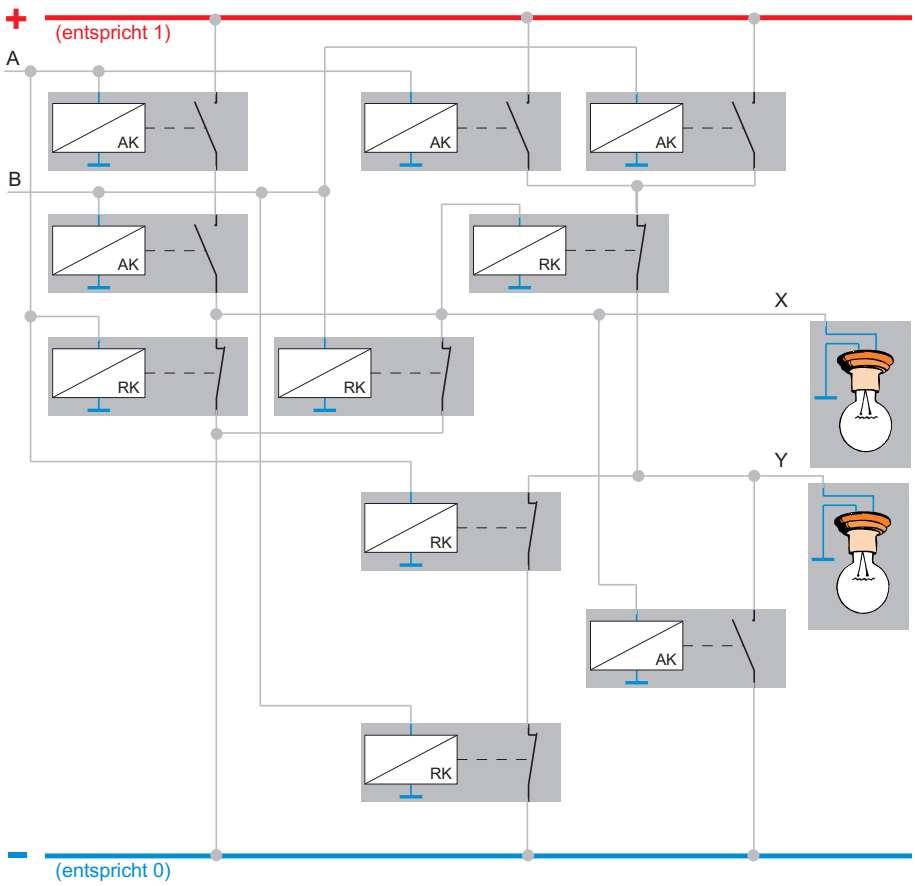


Abbildung 13.16
Additionswerk

Wir haben hier bisher weitgehend zwei Lösungsstrategien verfolgt. Die eine teilte das Problem in kleinere Probleme, die einfacher zu lösen waren. Die andere definierte kleinste Problemstellungen, aus deren Lösung wiederum größere Problemlösungen zusammengesetzt wurden. In jedem Fall wurden brauchbare Ergebnisse nicht verworfen, sondern immer wieder verwendet.

Genau das könnte uns auch in der aktuellen Situation helfen: Mit den einfachen Relais haben wir bisher mehrere brauchbare Schaltungen entwickelt, zum Beispiel den einfachen Multiplikator und den einfachen Addierer (Halbaddierer). Vielleicht können wir komplexere Schaltungen leichter erstellen, indem wir sie auf Basis der bereits geschaffenen Ergebnisse konstruieren. Die Komplexität dieser Konstruktion wird deutlich geringer, wenn wir nur noch über die Funktion der verwendeten Komponenten, nicht aber über die Funktionsweise nachdenken müssen. Das zugrunde liegende Prinzip ist die Abstraktion.

Zu diesem Zweck erschaffen wir neue Schaltsymbole, die nur noch die Funktion symbolisieren und daher deutlich übersichtlicher sind: Abbildung 10.17 zeigt das für den Halbaddierer. Statt komplexer Schaltungen steht nun nur noch das Additionssymbol sowie „HA“ für Halbaddierer darin. Links sind die Eingangsleitungen gezeichnet, rechts die Ausgangsleitungen. Im Bastelbogen bzw. der Kopiervorlage finden Sie Kärtchen zum Ausschneiden, mit denen Sie das Verhalten des Halbaddierers nachstellen können. Aufgrund der zwei Eingangsleitungen benötigen Sie nun allerdings zwei beidseitig bedruckte Kärtchen, um alle Eingangszustände zu simulieren. Der besseren Übersichtlichkeit halber sind die ein- und ausgehenden Leitungen außer durch die Farben Blau und Rot zusätzlich mit den entsprechenden binären Ziffern 0 und 1 gekennzeichnet.

Selbstverständlich müsste ein solches Bauteil auch noch einen elektrischen Anschluss zum Plus- und Minuspol besitzen. Das Prinzip der Abstraktion erlaubt uns jedoch, auch diese Leitungen im Symbol wegzulassen, weil sie für den aktuellen Denkprozess – aus einfachen logischen Bauteilen ein komplexeres zu gestalten – nicht nötig sind.

Nun erscheint die Aufgabe, drei Binärstellen zu addieren, vielleicht nicht mehr ganz so schwierig. Man könnte ja zunächst zwei Stellen addieren, dann zum Ergebnis eine weitere. Da wir oben ein Bauteil für diese Addition entwickelt haben, wäre es einen Versuch wert, zwei dieser Elemente hintereinander zu schalten. Abbildung 13.18 zeigt dies.

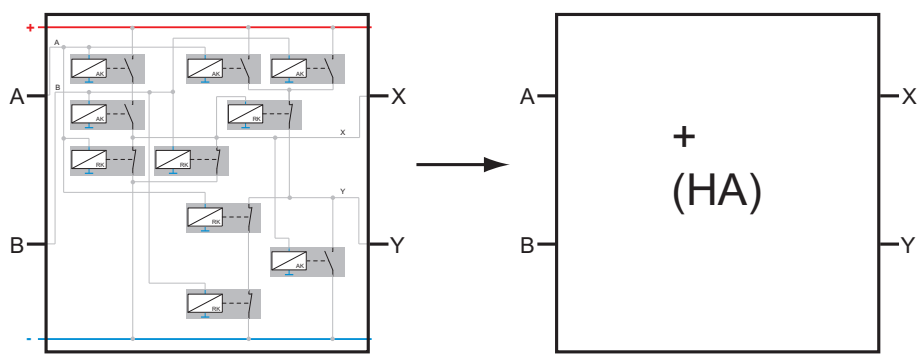


Abbildung 13.17
Der Halbaddierer bekommt ein eigenes Schaltsymbol.

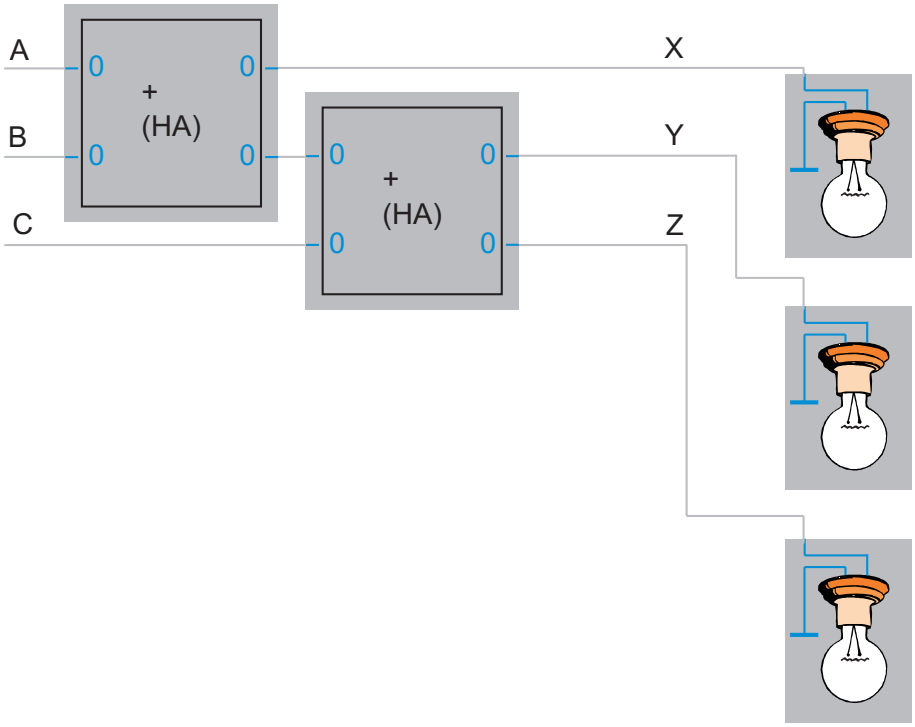


Abbildung 13.18
Verschaltung zweier Halbaddierer

Analysieren Sie die Schaltung mit Hilfe der Wahrheitstafel! Tragen Sie auch ein, welches Ergebnis Sie bei der Addition dreier Binärziffern in einem finalen Baustein eigentlich erwarten würden.

Eingang A	Eingang B	Eingang C	Ausgang X	Ausgang Y	Ausgang Z	erwartet
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Sie sind sicher auf das korrekte Ergebnis gekommen:

Eingang A	Eingang B	Eingang C	Ausgang X	Ausgang Y	Ausgang Z	erwartet
0	0	0	0	0	0	00
0	0	1	0	0	1	01
0	1	0	0	0	1	01
0	1	1	0	1	0	10
1	0	0	0	0	1	01
1	0	1	0	1	0	10
1	1	0	1	0	0	10
1	1	1	1	0	1	11

Wenn Sie nun die Ausgänge mit dem erwarteten Ergebnis vergleichen, fällt auf, dass Ausgang Z bereits genau das angibt, was wir als letzte Stelle des Additionsergebnisses wünschen. In der Tabelle habe ich das violett hervorgehoben. Die vordere Stelle des Ergebnisses ist – grün hervorgehoben – die Addition von Ausgang X und Y, wobei ein Übertrag nie vorkommt und wir ihn deshalb ignorieren können.

Was liegt also näher, als einen weiteren Halbaddierer einzusetzen, um die Schaltung „perfekt“ zu machen. Abbildung 13.19 zeigt dies.

Nun folgen wir wieder dem Prinzip der Informatik und packen dieses brauchbare neue Bauteil in ein eigenes Schaltsymbol, um es zu verwenden, ohne uns über den konkreten Bauplan Gedanken machen zu müssen.

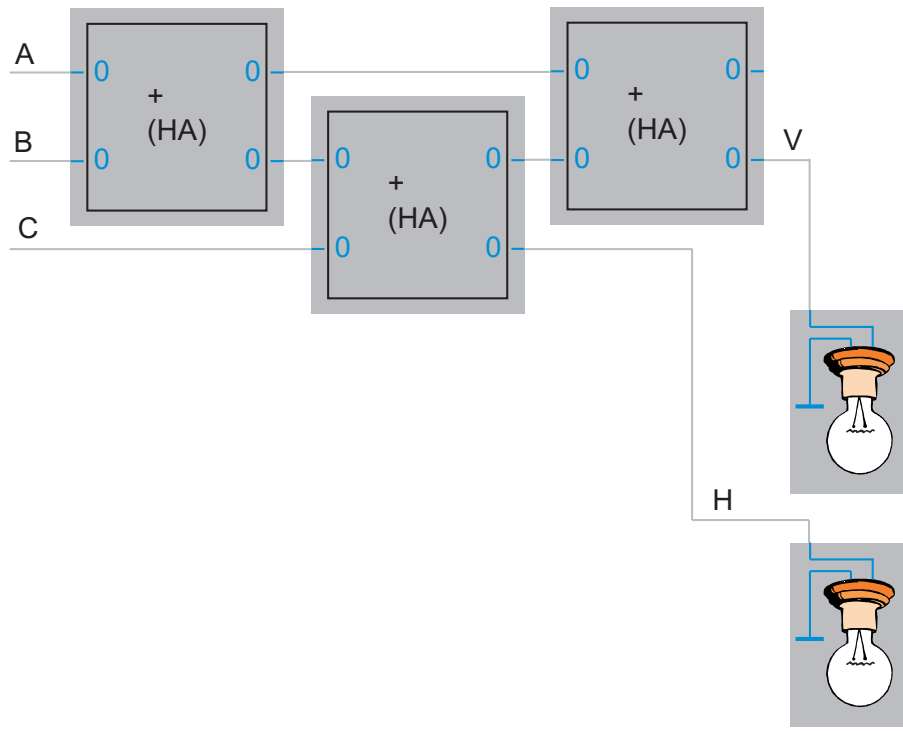


Abbildung 13.19
Schaltung für einen Volladdierer

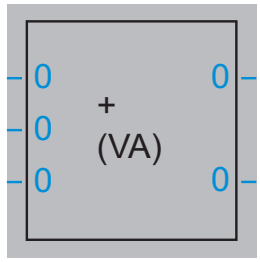


Abbildung 13.20
Der Volladdierer als Papier-Bauteil

Um das Geheimnis der Bezeichnung „Volladdierer“ zu lüften, bauen Sie bitte mit den Kärtchen die Schaltung von Abbildung 13.21 nach.

Nehmen Sie dann an, dass A und B jeweils eine vierstellige Binärzahl sei. Die Ziffern stehen in der Reihenfolge A3 A2 A1 A0 bzw. B3 B2 B1 B0.

Versuchen Sie nun zwei Zahlen in die Schaltung einzugeben, zum Beispiel 5 und 13. Auch wenn Sie aus den Kapiteln 4 und 5 noch Experten im binären Zählen und Rechnen sind, habe ich zur direkten Anschauung nochmal eine Tabelle mit Binärzahlen neben die Abbildung 13.21 gesetzt. Suchen Sie sich Zahlen heraus und legen Sie entsprechende Zustände an Ihre Schaltung. Kleine Hilfe: Dezimal 5 entspricht binär 0101. Damit müsste für Eingang A als A3 = 0, A2 = 1, A1 = 0, A0 = 1 geschaltet werden. Dezimal 13 entspricht binär 1101, daher schalten wir B3 = 1, B2 = 1, B1 = 0, B0 = 1.

Lesen Sie nun in gleicher Weise den Ausgang E (E für Ergebnis) ab und wandeln Sie die fünfstellige binäre Zahl anhand der Tabelle zurück in eine Dezimalzahl. Haben Sie eine Vermutung, was als Ergebnis herauskommt?

Überprüfen Sie Ihre Vermutung, indem Sie für A und B weitere Zahlen annehmen.

Das Ergebnis der Verschaltung von oben ist als Beispiel in Abbildung 13.22 dargestellt. Ausgang E stellt die Binärzahl 10010 dar, was laut Tabelle einer dezimalen 18 entspricht.

Es liegt nahe auszuprobieren, ob die Kaskade der eingesetzten Addier-Bausteine auch insgesamt eine Addition durchführt. Tatsächlich ergibt 5 + 13 = 18. Auch mit weiteren Werten bestätigt sich die Vermutung A + B = E.

Man kann also beliebig lange Binärzahlen addieren, wenn man entsprechend viele Volladdierer wie in der Schaltung nach Abbildung 13.31 zusammenfügt. Dafür reichen nicht Halbaddierer (man darf nur einen einzigen bei Stelle 0 verwenden), daher die Unterscheidung in „Voll-“ und „Halbaddierer“.

Beim schriftlichen Addieren zweier (Dezimal-)Zahlen betrachten wir die Ziffern einzeln von hinten nach vorne. Es kommt ein Ergebnis und – eventuell – ein Übertrag heraus, der dann ab der zweiten Stelle auch bei der Addition berücksichtigt werden muss.

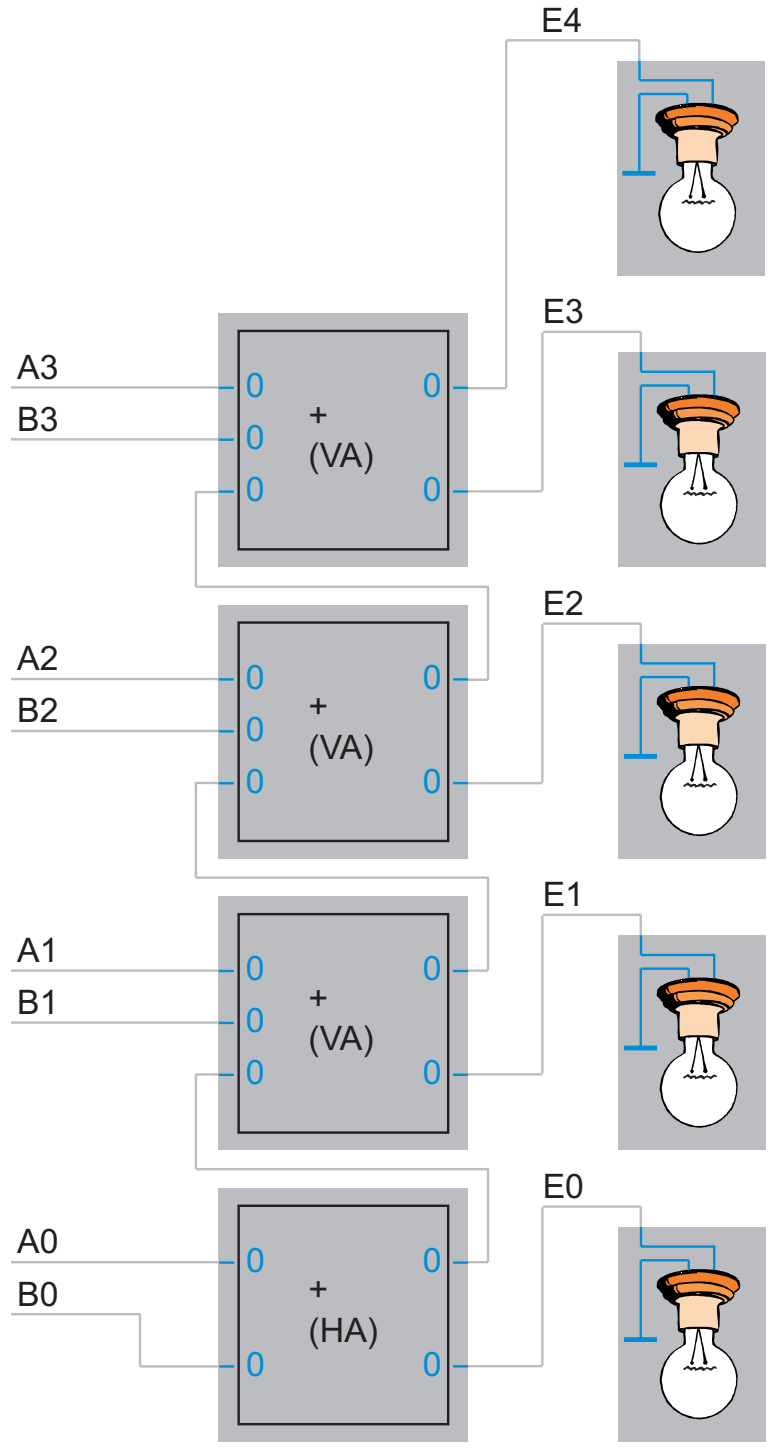
In gleicher Weise funktioniert das schriftliche Addieren im Binärsystem. Auch hier fallen Überträge an. Da pro Stelle also eigentlich drei Ziffern addiert werden (zwei plus Übertrag), hat der Volladdierer drei Eingänge.

Schriftliches Addieren
Hier finden Sie eine Gedächtnisstütze für die schriftliche Addition. Ausführlicher können Sie das in Kapitel 4 nachlesen.

```
   7 8 9
+  1 6 4
     1 1
-------
=  9 5 3
```

```
   1 0 1
+  0 1 1
     1 1
-------
= 1 0 0 0
```

Abbildung 13.21
Schaltung aus Voll- und Halbaddierern



Dezimal	Binär
0	00000
1	00001
2	00010
3	00011
4	00100
5	00101
6	00110
7	00111
8	01000
9	01001
10	01010
11	01011
12	01100
13	01101
14	01110
15	01111
16	10000
17	10001
18	10010
19	10011
20	10100
21	10101
22	10110
23	10111
24	11000
25	11001
26	11010
27	11011
28	11100
29	11101
30	11110
31	11111

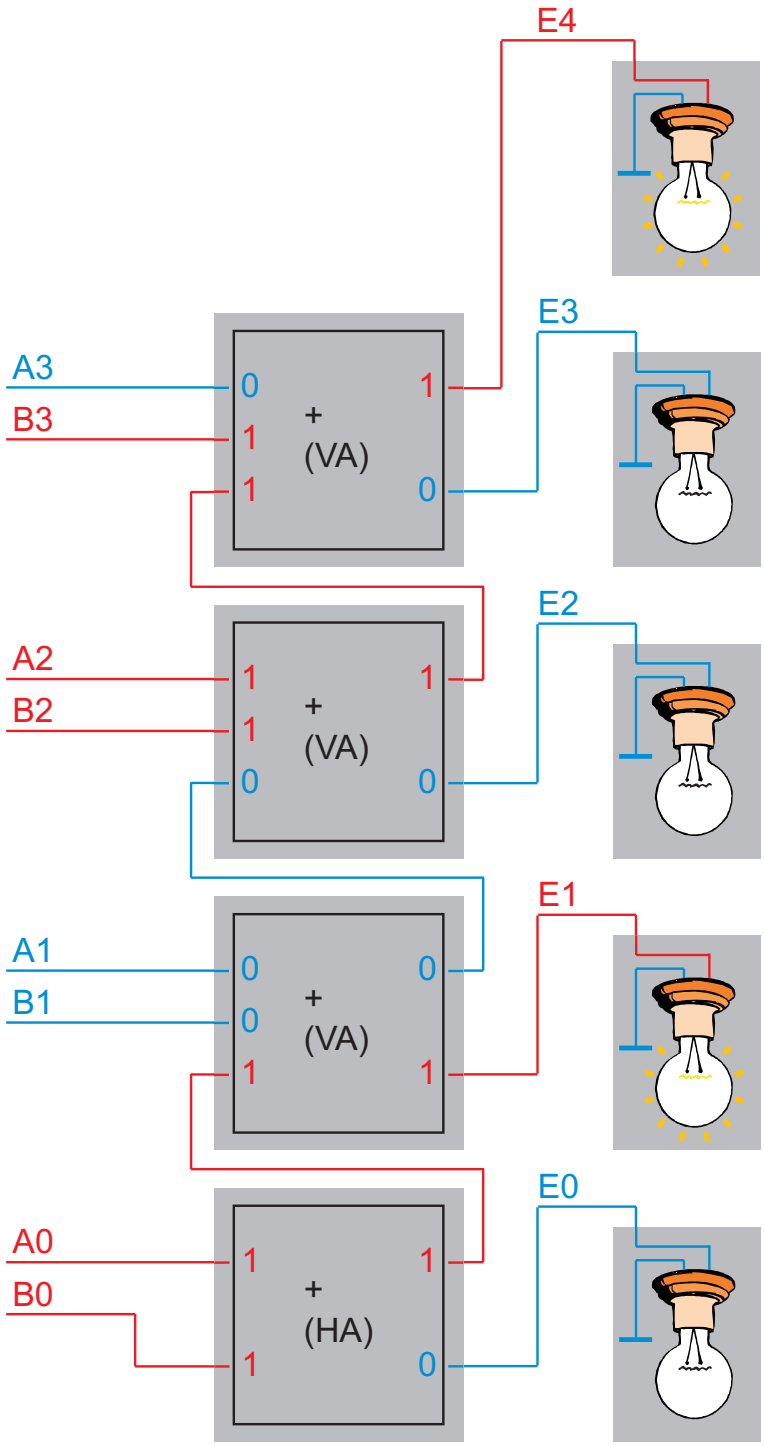


Abbildung 13.22
Schaltung mit A = 5 und B = 13

Zwischenresümee

Sie wissen nun, wie man mit Strom einfache Berechnungen im Binärsystem ausführen kann, indem man Relais verschaltet. Was jedoch noch viel wichtiger ist: Sie haben eine Vorstellung davon bekommen, wie man auch sehr komplexe, umfangreiche Schaltungen in den Griff bekommen kann: Wiederum konnten wir das wichtigste Prinzip der Informatik erfolgreich anwenden. Einfache Probleme werden gelöst und als Bausteine genutzt, um größere Probleme anzugehen. Die dabei entstehenden Lösungen dienen wiederum als Grundlage für die Lösung noch komplizierterer Probleme. Das funktioniert bei Algorithmen und abstrakten mathematischen Aufgabenstellungen genauso gut wie bei konkreten elektronischen Bauteilen, die zur Lösung einer Rechenaufgabe verschaltet werden sollen.

Was steckt dahinter?

Wahrscheinlich fragen Sie sich jetzt, warum Ihr Computer verhältnismäßig ruhig arbeitet. Die ganzen Relais darin müssten doch ziemlich klappern ...

Tatsächlich erkannte auch Zuse selbst recht früh, dass Relais entscheidende Nachteile mit sich brachten: Bei diesem mechanischen Bauteil ist die Schaltzeit nicht beliebig kurz realisierbar, denn ein Metallplättchen muss jedes Mal bewegt werden. Selbst schnellste Relais bringen es lediglich auf 1.000 Schaltvorgänge pro Sekunde. Das entspricht einer Taktfrequenz, die von aktuellen Mikroprozessoren millionenfach überschritten wird.

Noch relevanter ist jedoch der Energieverbrauch. Relais benötigen Strom, um zu arbeiten: Ein Elektromagnet betätigt den mechanischen Schalter. Während das Relais aktiv ist, muss die ganze Zeit Strom fließen, was Energie kostet. Das können Sie sich wieder anhand der Wasser-Analogie klarmachen. Betrachten Sie Abbildung 13.9: Während das Ventil betätigt wird, muss die ganze Zeit Wasser über den Hebel laufen, um die Feder zu strecken. Folge ist ein hoher Wasserverbrauch, der im Stromkreis für einen hohen Energieverbrauch steht. Bei den Relais kommt noch hinzu, dass die elektrische Energie in Wärme umgesetzt wird. Ein Computer aus Relais erwärmt sich daher und benötigt für den Betrieb zusätzlich Klimatisierung, was noch mehr Energie kostet.

Der MOS-Transistor
MOS ist die Abkürzung für „Metal Oxide Semiconductor", also Metalloxid-Halbleiter. Dies bezieht sich auf seinen Aufbau, bei dem eine Schicht Siliziumdioxid und eine aufgedampfte Metallschicht zentrale Rollen spielen. Wichtig ist, dass das Verhalten von MOS-Transistoren in digitalen Schaltungen recht einfach zu verstehen ist und dass MOS-Transistoren die am meisten verbreiteten Halbleiter sind.

Etwas besser sieht die Sache bei Röhren aus, hier wird statt eines Stück Metalls ein Elektronenstrahl umgelenkt. Daher waren Computer auf dieser Basis schneller als solche mit Relais, etwa die ENIAC. Über die Entwicklung der Technik können Sie sich in Kapitel 5 informieren.

Heute werden in der Computertechnik weder Relais noch Röhren verwendet, sondern Transistoren. Hier im Buch stelle ich die weit verbreiteten MOS-Transistoren vor. Diese funktionieren recht ähnlich zu unseren Relais: Auch sie schließen oder öffnen eine Art elektronischen Schalter. Allerdings benötigen sie hierfür keinen Strom, sondern eine Spannung. In unserer Wasser-Analogie entspricht die Spannung dem Wasserdruck. Abbildung 13.23 zeigt ein Modell dafür.

Die Betätigung des Ventils erfordert hier also keinen kontinuierlichen Fluss von Wasser, sondern lediglich das Anlegen des Wasserdrucks, um das Ventil zu schließen, und

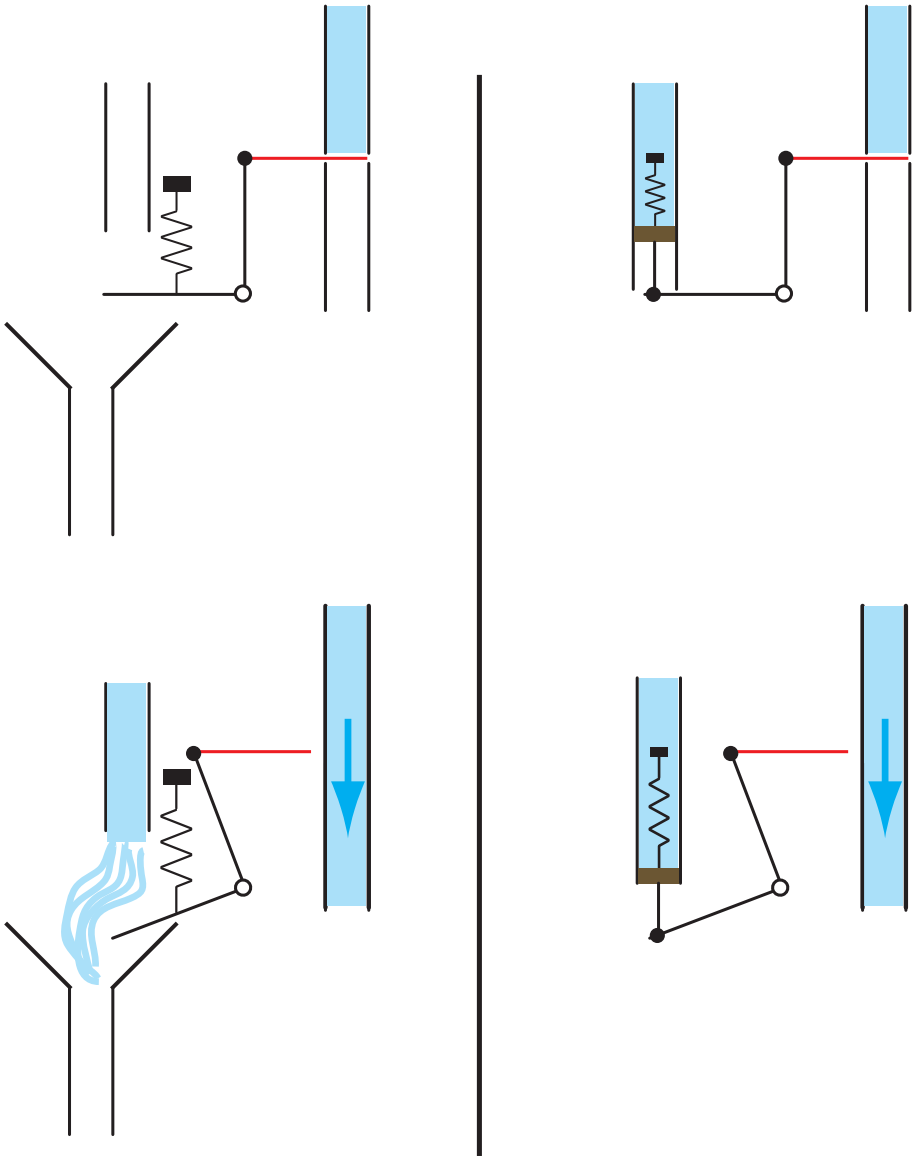
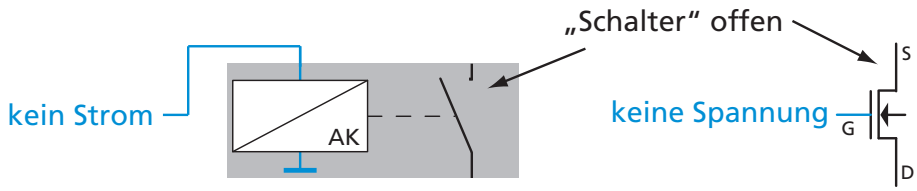


Abbildung 13.23
Schalten mit Wasserfluss (links) vs. Schalten mit Wasserdruck (rechts)

das Ablassen einer kleinen Menge Wasser, so dass der Druck wieder abnimmt, um das Ventil wieder zu öffnen.

Prinzipiell ist das die Funktionsweise eines MOS-Transistors. Wenn am Schalteingang (genannt „Gate“) eine Spannung angelegt wird, verändert sich die Stellung des elektronischen Schalters zwischen den beiden weiteren Anschlüssen des Transistors (genannt „Source“ und „Drain“). Die elektrische Spannung entspricht dem Wasserdruck in unserer Analogie. Der Pluspol entspricht großem Wasserdruck (so wie er durch den Stausee erzeugt wird), also einer Spannung. Der Minuspol entspricht dem Fehlen von Wasserdruck, also keiner Spannung.

Abbildung 13.24
Der NMOS-Transistor entspricht einem Relais mit Arbeitskontakt.



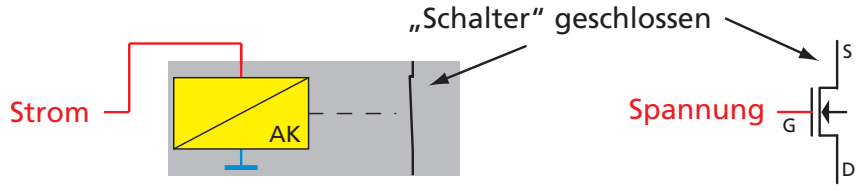
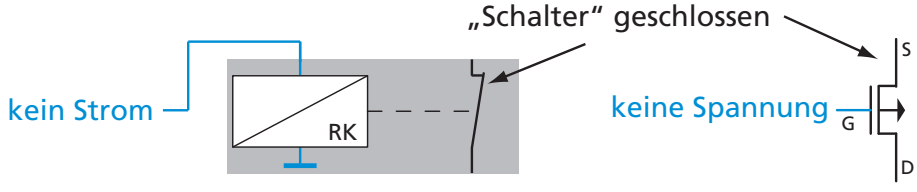
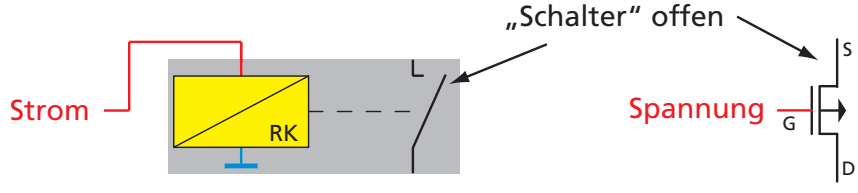


Abbildung 13.25
Der PMOS-Transistor entspricht einem Relais mit Ruhekontakt.





Die Abbildungen 13.24 und 13.25 zeigen die Schaltsymbole für zwei Arten von Transistoren: NMOS und PMOS sowie die Relais-Entsprechungen. Wie beim Relais mit Arbeitskontakt ist der interne Schalter des NMOS-Transistors im Ruhezustand geöffnet. Erst wenn eine Spannung am Gate angelegt wird, schließt der Schalter, und zwischen Source und Drain kann Strom fließen. Der PMOS-Transistor entspricht einem Relais mit Ruhekontakt.

Wegen dieser Entsprechungen können wir unsere Relaisschaltungen ohne Probleme in Transistorschaltungen umwandeln. Abbildung 13.26 zeigt den Inverter. Wenn am Eingang keine Spannung anliegt, ist der Ausgang über den oberen PMOS-Transistor mit dem Pluspol verbunden, was einer Spannung entspricht. Liegt am Eingang eine Spannung an, schalten beide Transistoren, der Ausgang ist mit dem Minuspol verbunden, hat also keine Spannung. Der Inverter funktioniert!

Vielleicht haben Sie sich beim Arbeiten mit den Relais auch schon folgende Fragen gestellt: Warum benötigt man überhaupt die unteren Relais, die den Ausgang gegebenenfalls mit dem Minuspol verbinden? Reicht es nicht aus, wenn die Lampe keinen Strom bekommt, was ja auch schon der Fall ist, wenn sie mit keinem der Pole verbunden ist?

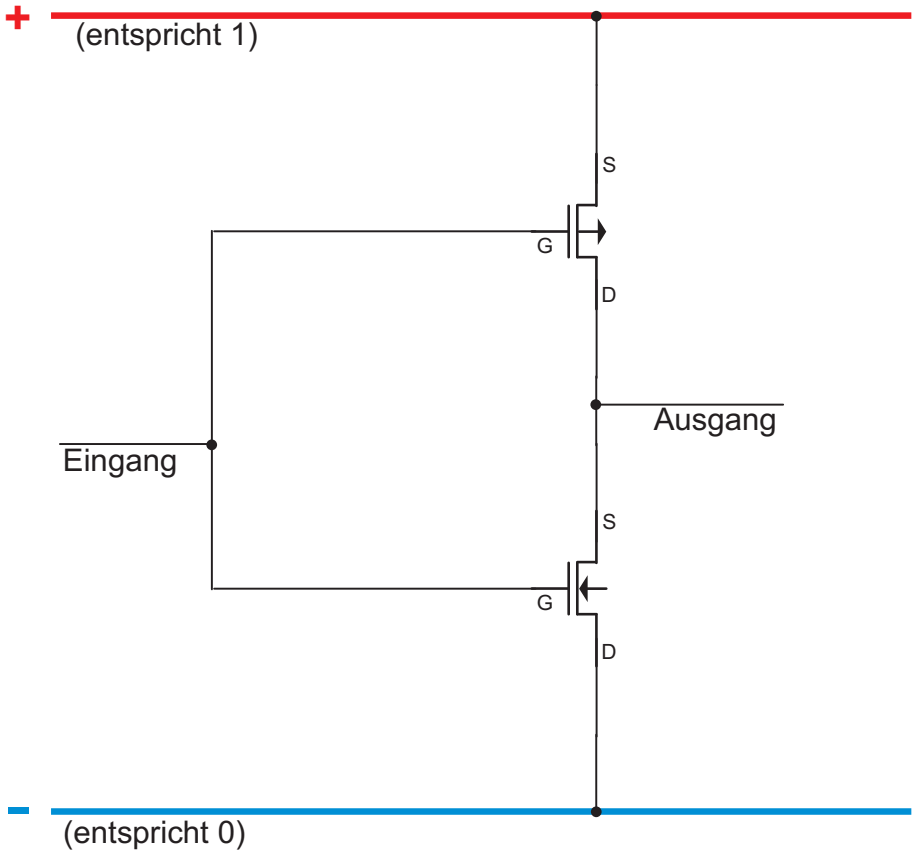


Abbildung 13.26
Inverter mit MOS-Transistoren

Wenn man die Wasser-Analogie zurate zieht, kommt man ebenfalls zu diesem Schluss: Damit das Mühlrad sich nicht bewegt, reicht es, wenn man es nicht mit Wasser versorgt – man braucht es nicht zusätzlich an eine leere Wasserleitung anzuschließen. Und Sie haben recht: Eigentlich war tatsächlich die Hälfte der Relais in der Schaltung überflüssig.

Wie sieht das aber bei der Verwendung von Transistoren aus? Jetzt ist plötzlich nicht mehr Strom relevant, sondern Spannung – in der Wasser-Analogie also Wasserdruck. Was passiert damit, wenn man ihn auf einer Leitung aufbaut und dann den Hahn zudreht? Richtig: Er bleibt erhalten. Das kennen Sie vielleicht vom Gartenschlauch. Wenn Sie zunächst vorne die Sprühdüse zudrehen und erst danach den Wasserhahn, bleibt der Druck eine ganze Zeit erhalten und wenn Sie den Schlauch an der Kupplung abziehen, spritzen Sie sich nass.

Genau das Gleiche passiert mit der Spannung auf einer elektrischen Leitung: Wird zunächst die Verbindung zum Pluspol geschaltet und daraufhin diese Verbindung unterbrochen, bleibt die Spannung erhalten, bis sie abfließen kann. Daher muss bei Schaltungen mit MOS-Transistoren der Ausgang unbedingt entweder zum Pluspol oder zum Minuspol verbunden werden. Ansonsten könnte er undefiniert sein, je nachdem, was für einen Zustand er vorher hatte. Vielleicht haben Sie schon die Abkürzung „CMOS“ gesehen, die für „Complementary Metal Oxide Semiconductor“ steht. Complementary, zu Deutsch „gegensätzlich“, steht genau für die oberen und unteren Schaltungsteile, die gegensätzlich schalten und damit den Ausgang definiert mit plus oder mit minus verbinden.

Elektrosmog

Beim Betrieb jedes elektronischen Gerätes entstehen – gewollt oder ungewollt – Radiowellen, die wiederum andere elektronische Schaltungen beeinflussen können. Daher ist es heute noch deutlich wichtiger als früher, Schaltkreise unempfindlich gegen solche Einflüsse zu machen. So kann ich mich an Folgendes erinnern: Als Kind bekam ich einen Elektronikbaukasten geschenkt mit Bastelanleitungen für diverse Anwendungen. Ein Mikrofonverstärker, ein Geräuschgenerator und vieles mehr waren dabei. Der Haken dabei war: Egal, welche Anleitung ich nachgebaut hatte, es kam immer ein Empfänger für den damals sehr starken Rundfunksender AFN heraus. Sogar bei einer Lichtschranke quakte leise Musik aus einem der Bauteile, das eigentlich gar nicht als Lautsprecher gedacht war ... Jahre später hat mich das dann motiviert, die Anleitungen nochmals durchzusehen, und ich fand heraus, dass die Ausgänge oft in einem undefinierten Zustand hinterlassen wurden. Die Entwickler des Baukastens hatten offenbar an einem Ort ohne starken Radiosender gewirkt.

Eine Leitung, die nicht eindeutig zugeordnet ist, wird darüber hinaus sehr stark vom sogenannten „Elektrosmog“ beeinflusst, also von den elektrischen Signalen unzähliger Handys, Radiosender und anderer Störquellen um uns herum. Diese verändern den Zustand der Leitung quasi zufällig. Aus diesem Grund war es mir wichtig, auch schon bei den Relaisschaltungen jeden Ausgang definiert entweder auf 0 oder auf 1 zu schalten. Auf diese Weise können Sie nun jeden Ihrer bisher erprobten Schaltpläne auch mit einer Transistorschaltung realisieren.

Zwei wesentliche Vorteile von Transistoren gegenüber Relais habe ich bereits erwähnt: Geschwindigkeit und geringer Stromverbrauch. Ein weiterer Vorteil wird in Kapitel 5 geschildert: So wie es mit steigender Anzahl von Bauteilen immer schwieriger wird, beim Schaltplan den Überblick zu behalten, gilt das auch für die tatsächliche Verdrahtung. Für die Erstellung von komplexeren Schaltungen haben wir einen Ausweg gefunden: Abstraktion ermöglicht das Zusammenfassen brauchbarer Schaltungen zu Modulen, die wir dann einfach einsetzen, ohne deren Aufbau weiter im Kopf haben zu müssen.

Im Fall von Transistoren kann man genau den gleichen Vorgang auch in der tatsächlichen Schaltungstechnik nachvollziehen: Transistoren und die Verbindungen zwischen ihnen können fotochemisch in einem Arbeitsschritt auf einem Stück Silizium hergestellt werden – dem Kern eines integrierten Schaltkreises oder kurz „Chip“. So wurden über die Zeit immer komplexere Funktionen direkt als Bauteil verfügbar, und man konnte mit den Chips Schaltungen entwerfen, die immer besser, schneller und umfangreicher waren.

Ein Vergleich: Atlas, der größte Computer, der jemals aus einzelnen Transistoren erbaut wurde, enthielt 80.000 dieser Bauteile. Die ersten einfachen Chips fassten 8 bis 20 Transistoren in einem integrierten Schaltkreis zusammen. Der erste echte Mikroprozessor, der Intel 4004, enthielt 1971 bereits 2.300 Transistoren. Im legendären IBM-PC wurde 1979 ein Intel 8088 eingesetzt, der bereits 29.000 Transistoren vereinigte. Die leistungsfähigsten Prozessoren enthalten heute deutlich über 1 Milliarde Transistoren, also ungefähr 20.000-mal so viel wie der gesamte Atlas-Computer.

Rechnen mit Strom und Computer

Ich habe Ihnen nun gezeigt, wie man mit Strom rechnet. Sie haben Schaltungen ausprobiert, die in der Lage sind, einfache Additionen und Multiplikationen auszuführen. Abbildung 13.27 zeigt schematisch eine solche Schaltung, auch Schaltnetz genannt: Setzt man die Zahlen für eine gewünschte Berechnung in elektrische Signale um, so kann man sie in die Eingänge der Schaltung füttern und an den Ausgängen das Ergebnis ablesen. Genau genommen berechnet die Schaltung eine mathematische Funktion über alle Eingänge, weshalb f (Eingang) am Kästchen steht.

Nun möchte ich Sie ein letztes Mal zum Nachdenken anregen: Ist ein solches Schaltnetz bereits ausreichend, um einen einfachen Computer zu bauen?

Vergleichen Sie das Verhalten von Schaltnetzen mit einem billigen Taschenrechner. Beide – Schaltnetz und Taschenrechner – sind in der Lage, einfache mathematische Berechnungen anzustellen. Gibt es trotzdem einen Unterschied?

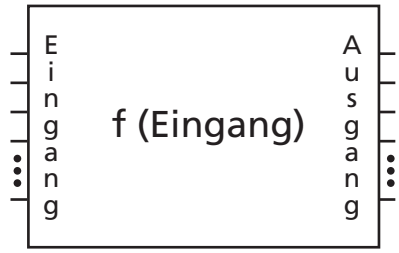


Abbildung 13.27
Schema eines Schaltnetzes als Funktion auf den Eingängen

Machen Sie mit dem Taschenrechner ein einfaches Experiment: Schalten Sie ihn ein. Auf der Anzeige erscheint „0“. Nun tippen Sie die Taste 4. Auf der Anzeige erscheint „4“. Tippen Sie erneut 4 und auf der Anzeige sehen Sie „44“.

Vielleicht fällt es Ihnen nicht sofort auf, weil das die normale Funktionsweise eines Taschenrechners ist. Trotzdem ist dieses Verhalten bemerkenswert: Sie haben zweimal hintereinander die gleiche Eingabe getätigt, der Taschenrechner hat aber nicht zweimal das Gleiche ausgegeben. Nach dem ersten Mal war die Ausgabe „4“, nach dem zweiten Mal war sie „44“.

Das kann keines unserer Schaltnetze leisten! Bei gleicher Beschaltung der Eingänge kann man (nach einer gewissen Zeit) immer das gleiche Ergebnis vom Ausgang ablesen!

Beim Taschenrechner – wie beim Computer – hängt das Ergebnis jedoch nicht nur vom Zustand der Eingänge ab, sondern auch von einer Art „innerem Zustand“, einem Speicher. Das ist der wesentliche Unterschied zu unseren einfachen Schaltnetzen.

Vielleicht können Sie aus dem Taschenrechner-Beispiel auch schließen, was dem Schaltnetz fehlt, um eine Art Speicher zu erhalten? Das Ergebnis des Taschenrechners ist nicht nur abhängig von der aktuellen Eingabe, sondern auch von dem, was während dieser Eingabe bereits auf der Anzeige zu sehen ist. Abbildung 13.28 zeigt das.

Die Idee liegt daher nahe, die Ausgänge eines Schaltnetzes nicht nur als Ausgänge zu benutzen, sondern sie gleichzeitig nochmals als Eingabe in dieses einzuspeisen, so wie in Abbildung 13.29. Man nennt das auch „Rückkopplung“.

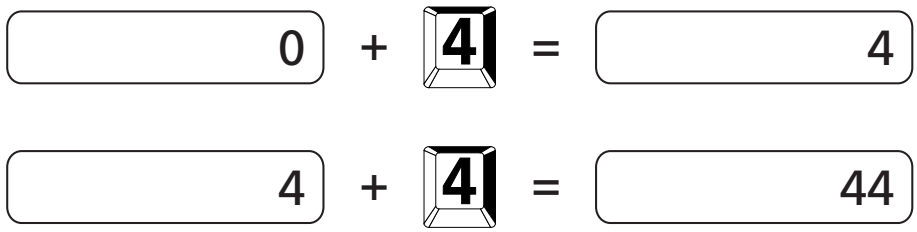


Abbildung 13.28
Die Ausgabe des Taschenrechners ist von der Eingabe und dem Inhalt der Anzeige abhängig.

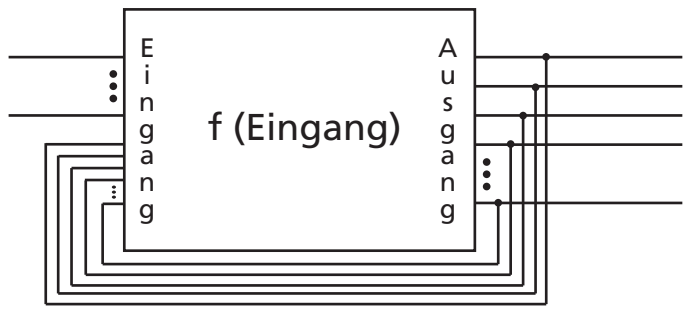


Abbildung 13.29
Die Ausgabe wird als Teil der Eingabe weiterverwendet.

Nun kann unser Schaltnetz genauso funktionieren wie ein Taschenrechner. Das Resultat heißt freilich in der Fachsprache nun nicht mehr Schaltnetz, sondern Schaltwerk oder einfach Automat. Auch hier gibt es wieder viele Varianten mit unterschiedlichen Möglichkeiten zu erforschen, aber das ist ein anderes Thema ...

Resümee

In diesem Kapitel haben Sie Methoden der Informatik auf elektrotechnische Schaltungen angewendet, um mit ihnen einfache Berechnungen durchzuführen. Sie haben einmal mehr Kompliziertes vereinfacht, um es besser lösen zu können. Diesmal haben Sie erfahren, dass man die gleiche Technik sogar anwenden kann, um das Verdrahtungsproblem zu lösen: Funktionen werden auf Chips verlagert, die Transistoren und Verdrahtung gleich „an Bord" haben. Und zum Schluss haben Sie noch erfahren, was unseren Schaltungen noch fehlt, um als echte Computer zu fungieren: die Rückkopplung.

→

Abbildung 13.K1
Kopiervorlage für die Papier-Schaltpläne

+ (entspricht 1)

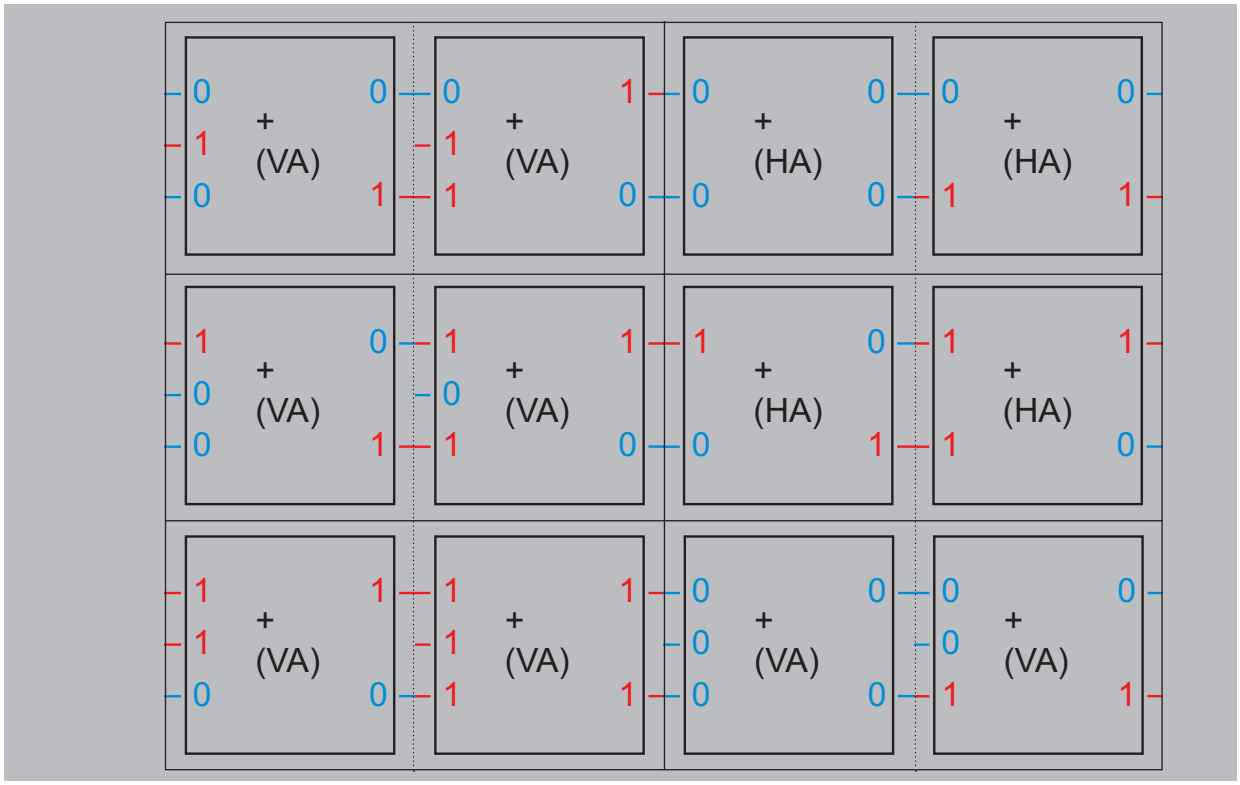
– (entspricht 0)

Abbildung 13.K2
Mini-Version der Vorlage für Papier-Schaltpläne

+ (entspricht 1)

Abbildung 13.K3
Volladdierer und Halbaddierer als Legeplättchen. Bitte mehrfach kopieren!

− (entspricht 0)



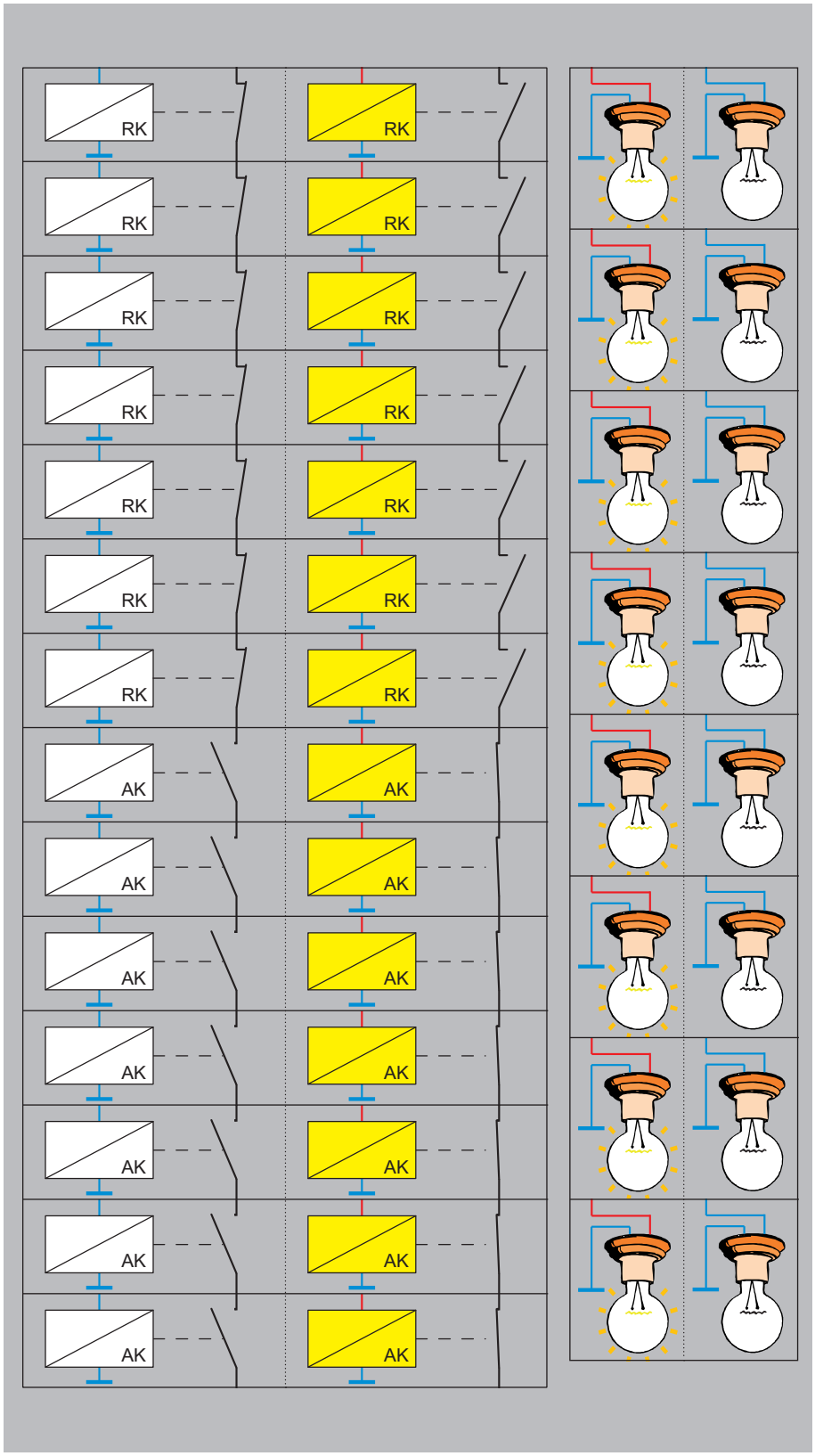
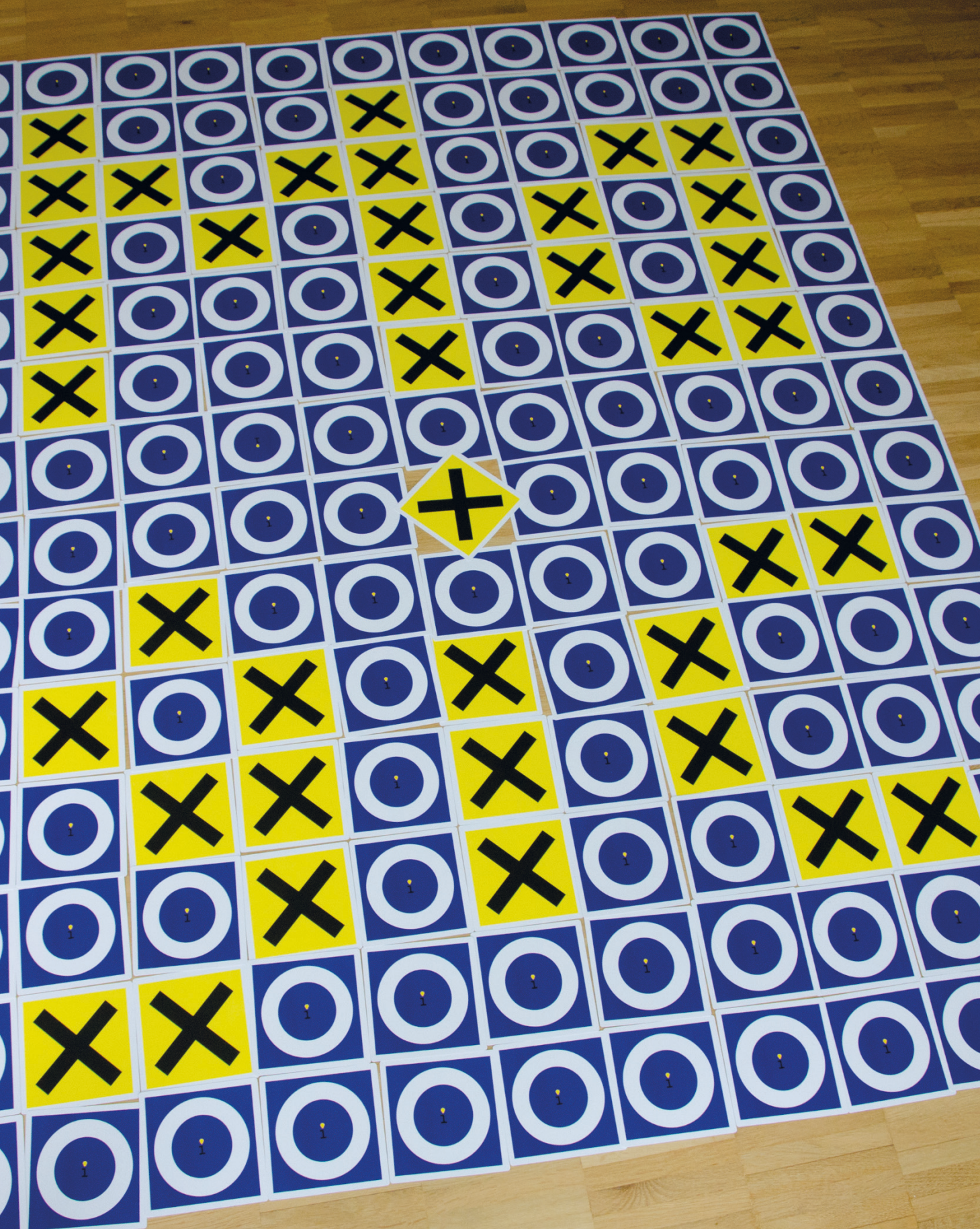


Abbildung 13.K4
Relais mit Arbeitskontakt und solche mit Ruhekontakt sowie Lampe als Legeplättchen. Bitte bei Bedarf mehrfach kopieren!



14. InformaGik

Einführung

Zum Einstieg in dieses Kapitel würde ich Ihnen gerne ein kleines Zauberkunststück vorführen. Ein Buch ist als Zauberkünstler freilich nicht besonders geeignet und daher benötigen wir einen Zauberkünstler in spe, der sich in die Geheimnisse einweihen lässt. Bevor Sie weiterlesen, sollten Sie sich daher entscheiden, ob Sie sich den Trick von jemand anderem vorführen lassen möchten. In diesem Fall geben Sie dem designierten Zauberkünstler nun das Buch und bitten ihn, die Vorführanleitung am Ende dieses Kapitels zu lesen. Der Trick ist recht einfach und kann auch von sonst in der Magie ungeübten Menschen durchgeführt werden.

Nachdem Sie selbst den Aha-Effekt erlebt haben, können Sie die Anleitung lesen und selbst Zauberer spielen. Bitte denken Sie daran, im Anschluss immer zusammen mit Ihrem Publikum zu überlegen, wo in Ihrem Alltag eine Anwendung zu finden ist, die eine ganz ähnliche „InformaGik“ verwendet.

Information

Erinnern Sie sich? Bereits in den Kapiteln 6 und 7 haben wir uns mit Information und deren Codierung beschäftigt. Dort konnten wir die Häufigkeit einzelner Informationseinheiten (Buchstaben) ausnutzen, um diese so sparsam wie möglich durch Folgen von 0 und 1 auszudrücken. Auch in diesem Kapitel geht es um Information: Ein Feld aus X und O enthält 25 unabhängige Informationseinheiten. In diesem Fall sind sie zufällig von Freiwilligen als Muster gelegt worden, aber es könnte sich durchaus auch um einen kleinen Text in binärer Codierung handeln.

Diese Informationen werden vom Zauberkünstler ergänzt – allerdings nur scheinbar zufällig: Er folgt einer festen Regel, so dass danach in jeder Zeile und Spalte immer eine gerade Anzahl von X liegt. Das Legen wird auf diese Weise eindeutig, man hat keinerlei Wahlfreiheit.

Was bedeutet das für den Informationsgehalt des Spielfelds? Die Zahl der Karten ist größer geworden. Ist damit auch der Informationsgehalt von 25 auf 36 „Einheiten“ gestiegen? Denken Sie bei der Beantwortung an Ihre Erfahrungen aus Kapitel 6: Einen Text können wir durch geschickte Codierung nach dem Verfahren von Huffman deutlich im Umfang reduzieren, ihn aber jederzeit wieder in die ursprüngliche Form bringen. Der Informationsgehalt ändert sich durch diese Codierung nicht.

Können Sie aus dem 6 × 6-Feld aus X und O ebenfalls Informationen wegnehmen, so dass dieses trotzdem wieder komplett rekonstruierbar ist?

Selbstverständlich lassen sich die zusätzlich gelegte Zeile und Spalte wieder entfernen, denn diese können wir jederzeit nach der Regel neu legen, nach der sie ursprünglich entstanden sind. Wenn wir genau darüber nachdenken, können wir sogar eine beliebige Zeile und eine beliebige Spalte aus dem Feld entfernen! Mit der Regel, Zeilen und Spalten auf eine gerade Zahl von X zu ergänzen, sind wir auch jetzt noch in der Lage, das Muster wiederherzustellen. Versuchen Sie es in Abbildung 14.1.

Wenn man aber etwas weglassen kann und danach immer noch keine Information verloren ist, scheint das Weggelassene ganz offensichtlich keine Information zu enthalten. Wozu soll es also dienen, so etwas hinzuzufügen?

In unserem Zaubertrick ermöglichten uns die zusätzlichen Karten die Identifikation eines Fehlers, den in diesem Fall jemand absichtlich durch Umdrehen einer Karte erzeugt hat. Fehler passieren aber in unserer Umgebung auch unbeabsichtigt:

- Beim Abschreiben von Zahlenkolonnen verwechseln Sie eine Ziffer oder vertauschen zwei benachbarte Stellen.
- Durch Nebengeräusche im Telefon notiert jemand Ihren Namen falsch.
- Ein Autokennzeichen ist verschmutzt und so wird aus KU für Kulmbach leicht KO für Koblenz.

Während solche Fehler durch höhere Sorgfalt reduziert werden könnten, tun wir uns besonders in Systemen, die sehr große Datenmengen auf engstem Raum speichern, damit sehr schwer: Die sogenannten Pits auf einer Blu-Ray-Disc haben einen Durchmesser von 0,15 µm (1 µm = 0,001 mm), die Spuren einen Abstand von 0,32 µm. Auf eine Fläche von einem Millimeter mal einem Millimeter kommen demnach pro Lage etwa 20 Millionen Informationseinheiten oder 2,5 MByte.

Bei einer zweilagigen Scheibe macht also selbst ein winziges Staubkorn mit 10 µm Durchmesser (auf einem Quadratmillimeter haben 10.000 dieser Teilchen an der Grenze zum Feinstaub Platz) bereits etwa 4.200 Pits unlesbar, was immerhin ungefähr

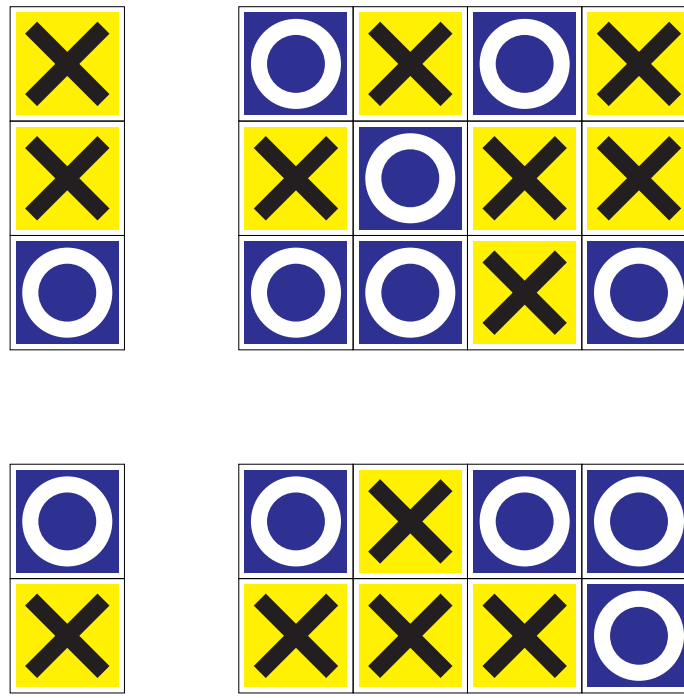


Abbildung 14.1
In dem 6 × 6-Muster fehlt die zweite Spalte und die vierte Zeile. Können Sie diese wiederherstellen?

dem Informationsgehalt des Texts auf einer Buchseite entspricht. Da kaum jemand im Wohnzimmer einen staubfreien Reinstraum unterhält, sind solche Partikel unvermeidbar. Es ist daher notwendig, den Staub zu akzeptieren: Die Fehler müssen irgendwie erkannt und korrigiert werden.

Fehlerkorrektur

Beim Vorführen des Zaubertricks vom Anfang des Kapitels haben wir uns daher implizit mit einer weiteren Bedeutung des Begriffs „Datensicherheit“ beschäftigt: der Sicherung von Daten gegen unbeabsichtigtes Verfälschen. Zusätzliche – fachsprachlich „redundante“ – Daten erhöhen den Informationsgehalt eines Datensatzes nicht, aber sie ermöglichen, dass eine gewisse Zahl von Fehlern erkannt und unter Umständen auch korrigiert werden kann.

Das wollen wir gleich an unserem Zauberquadrat ausprobieren: Wenn Sie von einer korrekten 6 × 6-Anordnung ein Kärtchen umdrehen, können Sie sofort erkennen, welches das war, und es wieder korrigieren.

Nun drehen wir zwei Kärtchen um. Versuchen Sie herauszufinden, ob Fehler in Abbildung 11.2 vorliegen, und wenn ja, diese zu korrigieren – jeweils ohne die Abbildung mit dem „Original“ vom Ende des Kapitels zu vergleichen.

Redundanz
In der römischen Antike sprach man hauptsächlich von „redundare“, wenn Flüsse über die Ufer getreten waren. In der Technik hat sich der Begriff heute für etwas etabliert, was über das absolut Notwendige hinaus vorhanden ist. In Flugzeugen sorgen etwa redundante Systeme dafür, dass ein abstürzender Computer nicht zum Absturz der Maschine führt. Redundante Daten sorgen für Fehlertoleranz.

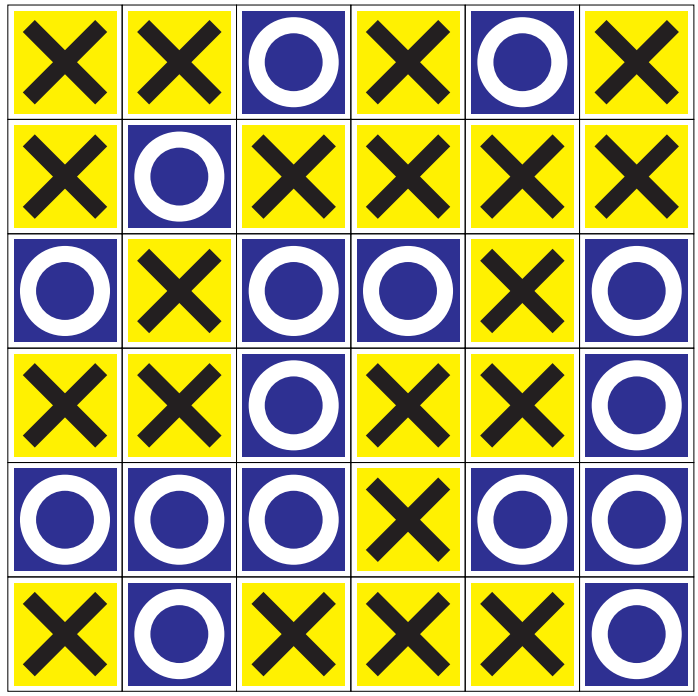


Abbildung 14.2
Finden Sie Fehler im Muster?

Wir stellen fest, dass die zweite und die fünfte Zeile sowie die zweite und die vierte Spalte nicht der Regel entsprechen. Allerdings könnten wir nicht mit Gewissheit sagen, welche Fehler zu dieser Konstellation geführt haben: Es wäre möglich, dass entweder die in Abbildung 14.3 grün markierten oder die rot markierten Karten umgedreht wurden.

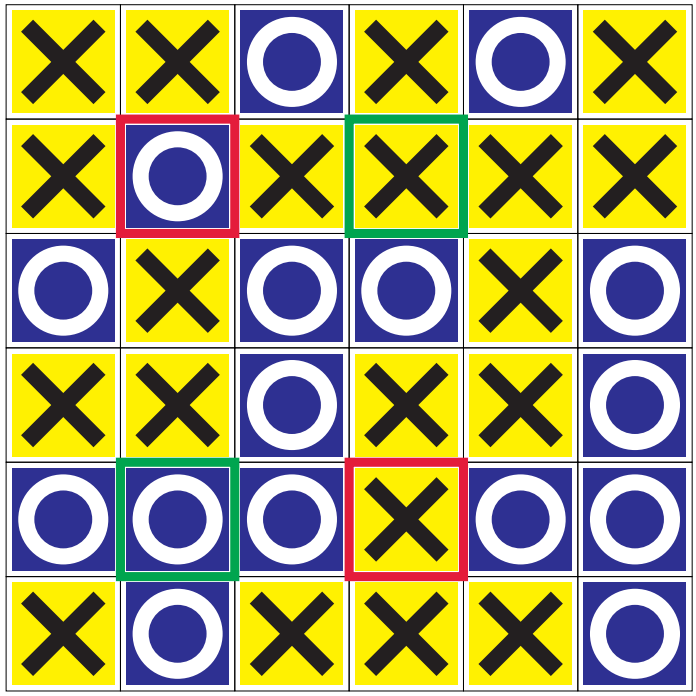


Abbildung 14.3
Entweder die grün oder die rot markierten Karten sind herumgedreht worden.

Versuchten Sie, den Fehler zu korrigieren, hätten sie mit gleicher Wahrscheinlichkeit den Fehler beseitigt oder verschlimmert, indem dann nicht zwei, sondern vier Kärtchen falsch lägen. In diesem Fall würde man sich also eher dafür entscheiden, gar nicht zu korrigieren. Ein Computersystem könnte zum Beispiel einfach auf den Fehler aufmerksam machen oder – falls es sich etwa um ein Datenpaket aus dem Internet handelt – die Daten erneut anfordern.

Noch schlechter sieht die Sache aus, wenn wir drei Kärtchen umdrehen, die drei Ecken eines Rechtecks beschreiben. In Abbildung 14.4 sind das die drei rot markierten Kärtchen. Folgten wir dem Fehlerkorrekturschema aus dem Zaubertrick, würden wir die rot markierte Karte umdrehen und damit die Fehlerzahl auf vier erhöhen, statt sie zu senken.

Spielen Sie eine Weile selbst mit den Karten und untersuchen dabei, wie viele Fehler zu einem erkennbaren Regelverstoß führen und in welchen Fällen Sie die Fehler mit Gewissheit korrigieren können.

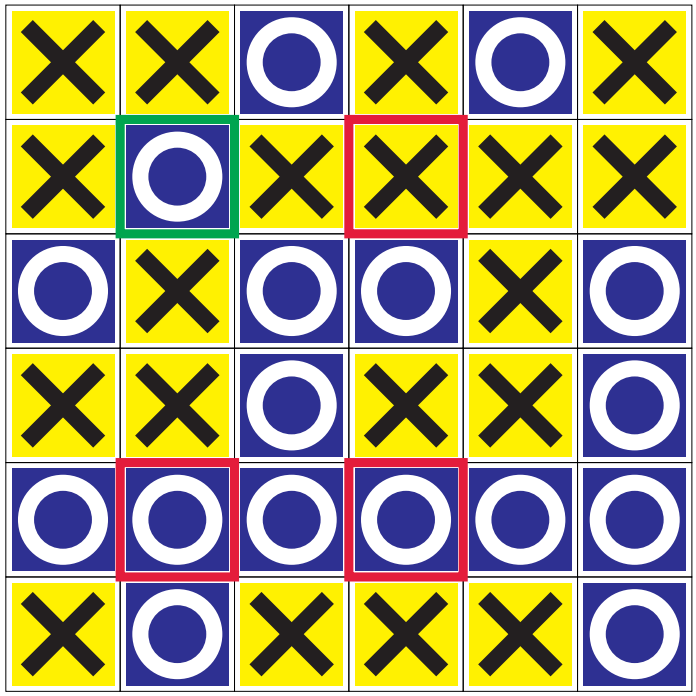


Abbildung 14.4
Statt die drei Fehler zu korrigieren, würde ein Korrekturschema die grün umrandete Karte umdrehen.

Vielleicht kommen Sie auf eine Antwort wie: „Ein Fehler lässt sich erkennen und korrigieren, zwei Fehler lassen sich erkennen und drei Fehler lassen sich ebenfalls erkennen."

Das stimmt allerdings nur annähernd. Mit dem letzten Beispiel habe ich demonstriert, wie die Fehlerkorrektur die Situation sogar verschlimmert, wenn aus drei Fehlern vier werden. Andererseits kann man den Fehler doch erkennen, aber nicht korrigieren ...

Richtig wäre daher eine Antwort wie :

- Man kann sich entscheiden, ob man Fehler korrigiert oder nicht.
- Korrigiert man Fehler, kann man einen Fehler korrigieren und zwei Fehler noch erkennen.
- Verzichtet man auf die Korrektur, kann man bis zu drei Fehler erkennen.

Genau das gilt prinzipiell für alle Verfahren, die mit Hilfe zusätzlicher Daten Informationen toleranter gegenüber Fehlern machen: Man muss sich jeweils entscheiden, ob man Fehler nur erkennen oder auch korrigieren möchte. Die Korrektur funktioniert nur mit einer kleineren Anzahl von Fehlern und man muss bei mehr Fehlern eine falsche Korrektur in Kauf nehmen.

Die Zahl korrigierbarer bzw. erkennbarer Fehler lässt sich durch mehr zusätzliche Informationen steigern. So könnte man auf die Idee kommen, zwei Zeilen und Spalten zu ergänzen und etwa die Zahl der X für die 1., 3., 5. und 7. Zeile bzw. Spalte und für die anderen getrennt zu berechnen, wie in Abbildung 14.5 gezeigt.

Schleichen sich nun zwei Fehler ein, können Sie diese meistens erkennen und korrigieren. Abbildung 14.6 zeigt ein auf diese Weise entstandenes Quadrat (mit einer anderen Information als oben), in dem zwei Fehler versteckt sind. Finden Sie diese?

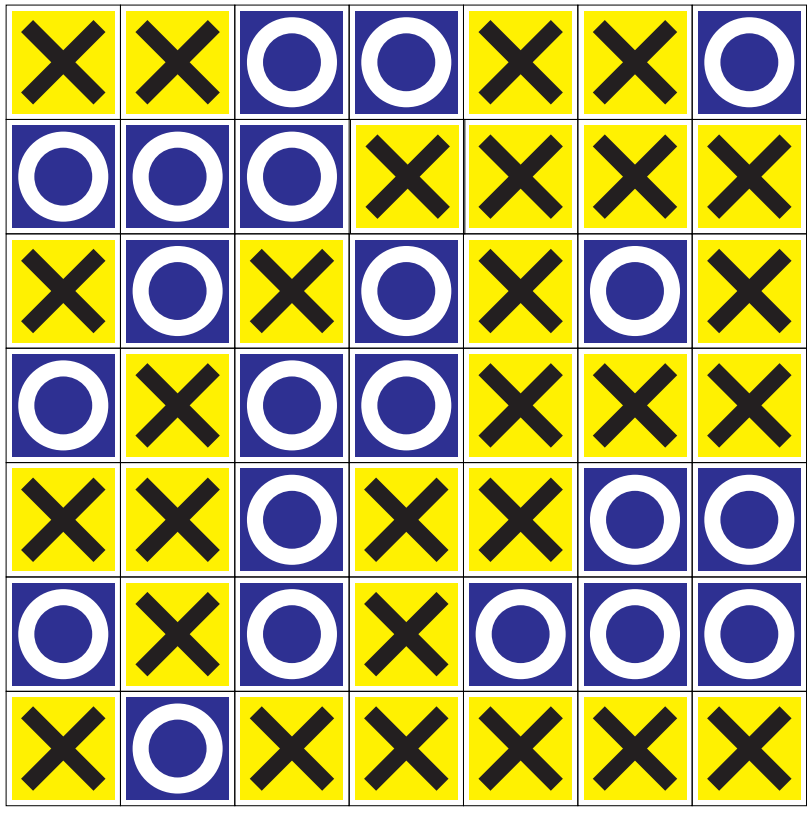


Abbildung 14.5
Mehr Fehler können mit mehr zusätzlichen Daten korrigiert werden.

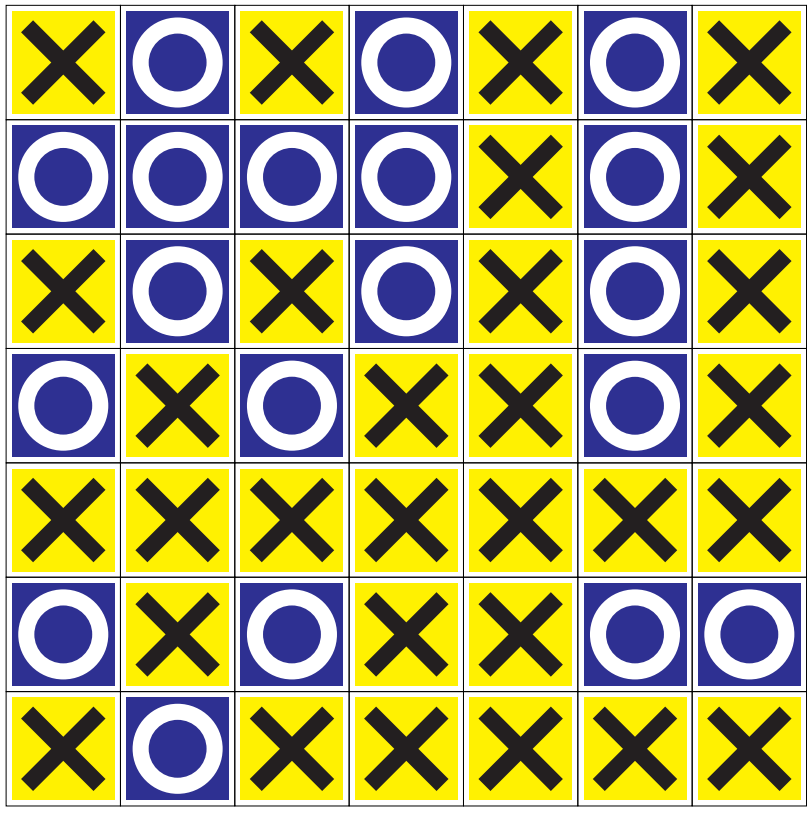


Abbildung 14.6
Finden Sie die beiden Fehler!

Wieder halten wir nach Fehlern in den Zeilen und in den Spalten getrennt Ausschau. In Abbildung 14.7 habe ich die potentiell fehlerhaften Felder mit roten Quadraten (für fehlerhafte Zeilen) und grünen Kreisen (für fehlerhafte Spalten) markiert. Dort, wo ein Feld sowohl für einen Spalten- als auch für einen Zeilenfehler verantwortlich ist (und daher mit Quadrat und Kreis markiert ist), ist der Fehler am wahrscheinlichsten – wir sollten daher genau diese Felder korrigieren.

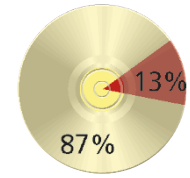
Auf diese Weise lassen sich nun auch zwei Fehler erkennen. Überlegen Sie, ob das Verfahren für alle möglichen Fehlerfälle funktioniert.

Leider gilt das nicht immer! Durch die Trennung in gerade und ungerade Zeilen bzw. Spalten haben wir lediglich unser Quadrat mit der Information sozusagen in zwei kleinere Teilquadrate aufgeteilt – so als würden Sie bei einem Schachbrett hintereinander zuerst nur die schwarzen und danach die weißen Felder betrachten. Auf diese Weise wären zwei Fehler dann korrigierbar, wenn sie sich auf unterschiedlichen Feldfarben befänden – zwei Fehler, die beide auf den „schwarzen" Feldern wären, könnten wir immer noch nicht einordnen. Unser Muster ist noch etwas komplizierter als das Schachbrett, Sie können das aber trotzdem ausprobieren, indem Sie zwei Karten umdrehen, die auf einer horizontalen, vertikalen und diagonalen Gerade liegen und zwischen denen sich noch genau eine weitere Karte befindet.

Auf jeden Fall ist unser neues Verfahren aber gewappnet gegen einen häufigen, menschlichen Fehler: das Vertauschen zweier nebeneinanderliegender Karten.

Fehlerkorrektur von DVD oder Blu-Ray

Bei DVDs oder Blu-Ray-Discs werden von der gesamten möglichen Speicherkapazität nur etwa 87 % für „echte Daten" ausgenutzt, der Rest wird für redundante Daten investiert, die eine umfangreiche Fehlerkorrektur ermöglichen. Diese Korrektur ist darüber hinaus auf die Besonderheit ausgerichtet, dass Fehler wahrscheinlich lokal sehr konzentriert auftreten (durch ein Staubkorn oder eine Beschädigung).



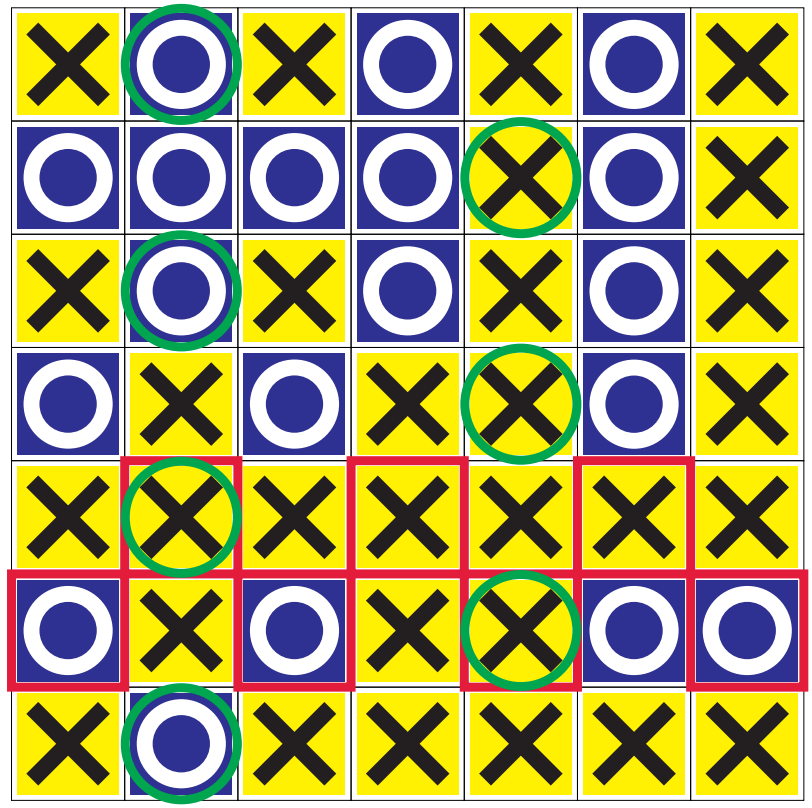


Abbildung 14.7
Potentielle Fehlerquellen

Diese befinden sich sowohl horizontal als auch vertikal immer in unterschiedlichen „Bereichen“ (geraden oder ungeraden Zeilen oder Spalten).

Auch bei den wirklich eingesetzten Fehlerkorrekturverfahren lassen sich bestimmte Fehlerarten besser korrigieren als andere, und auch dort liegt ein besonderes Augenmerk auf dem Vertauschen von Stellen, den beliebten „Zahlendrehern“. Das können Sie anhand Ihrer Kreditkarte oder Ihres Personalausweises leicht überprüfen: Die Nummern werden öfters auch einmal per Telefon weitergegeben oder abgetippt (z. B. wenn Sie online etwas damit bestellen). Dabei passiert es leicht, dass zwei Ziffern in die falsche Reihenfolge geraten. Aus diesem Grund gibt es auch hier die sogenannte „Prüfziffer“, die so ergänzt wird, dass die gesamte Zahlenkolonne eine festgelegte Regel einhält, die durch Vertauschen oder Vertippen gestört wird.

Bei der Kreditkarte wird einfach abwechselnd jede Ziffer mit 2 und mit 1 multipliziert. Von den Ergebnissen addiert man wiederum die einzelnen Ziffern. Diese Summe muss ein Vielfaches von 10 sein.

Probieren wir das für unsere Beispiel-Kreditkarte in Abbildung 14.8 aus. Die Rechnung lautet:

Die Nummern verraten noch mehr:
Wussten Sie, dass man aus der Nummer auch ermitteln kann, um welche Art Kreditkarte es sich handelt? So beginnt zum Beispiel die Nummer einer MasterCard immer mit 51 bis 55, eine Visa Karte mit 4 und eine BahnCard mit 70.

6	4	0	0	0	5	1	2	3	4	5	6	7	8	9	5
× 2	× 1	× 2	× 1	× 2	× 1	× 2	× 1	× 2	× 1	× 2	× 1	× 2	× 1	× 2	× 1
12	4	0	0	0	5	2	2	6	4	10	6	14	8	18	5
1 + 2 + 4 + 0 + 0 + 0 + 5 + 2 + 2 + 6 + 4 + 1 + 0 + 6 + 1 + 4 + 8 + 1 + 8 + 5 = 60															

60 ist ein Vielfaches von 10, daher hat diese Kreditkarte eine gültige Nummer. Das ist natürlich keine Garantie für die Echtheit (und die hier verwendete Kombination ist offiziell für Testzwecke vorgesehen)!

Probieren Sie das doch gleich einmal mit Ihren Kreditkarten aus. Wie verändert sich die Summe, wenn Sie zwei Ziffern vertauschen?

Abbildung 14.8
Kreditkarte

Durch das abwechselnde Multiplizieren mit 1 und 2 fallen Zahlendreher in den allermeisten Fällen auf. Gegenüber bestimmten Vertauschungen ist das Verfahren allerdings blind. Erkennen Sie, welche das sind?

Zahlendreher bei den Ziffern 0 und 9 fallen nicht auf, da beide Ziffern auf jeder Position den gleichen Teil zur Prüfsumme beitragen: 0 ergibt multipliziert mit 1 und 2 ohnehin die gleiche Zahl. $9 \times 1 = 9$ und $9 \times 2 = 18$. Da aber die Ziffern einzeln in die Prüfsumme eingehen, ergibt sich $1 + 8 = 9$.

Nicht besser ist das Verfahren beim EU-Personalausweis oder Reisepass: Der Ausweis einer berühmten Persönlichkeit in Abbildung 14.9 dient uns als Beispiel. Hier ist explizit festgelegt, dass immer die letzte Ziffer eines Blocks (Ausweisnummer, Geburtsdatum, Ablaufdatum) eine Prüfziffer darstellt, die folgendermaßen berechnet wird:

Die Ziffern werden abwechselnd nacheinander mit 7, 3 und 1 multipliziert. Die Prüfziffer ist die letzte Ziffer der Summe. Buchstaben (wie das „M“) spielen dabei keine Rolle. Für die Ausweisnummer gilt daher die Rechnung:

5	0	0	4	5	6	7	8	9
× 7	× 3	× 1	× 7	× 3	× 1	× 7	× 3	× 1
35	0	0	28	15	6	49	24	9
Summe: 166								

Die Prüfziffer ist also 6. Am Ende wird dann auf die gleiche Weise noch eine Gesamtprüfziffer über alle Ziffern (einschließlich der drei Einzelprüfziffern) gebildet.

Wollen Sie das Verfahren mit den Daten auf dem Beispielausweis ausprobieren? Oder Sie überprüfen gleich den eigenen Ausweis! Welche Zahlendreher fallen hier nicht auf?

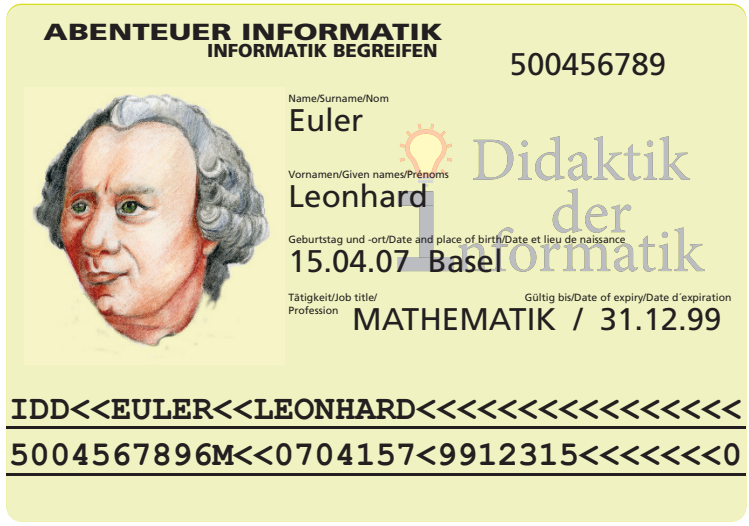


Abbildung 14.9
Beispiel-Ausweis

Zum Beispiel sind 0 und 5 vertauschbar (da die 5 auf jeder Position wieder eine 5 zur Prüfsumme beiträgt), genauso aber etwa 1 und 6, da in jeder möglichen Position beim Vertauschen die gleichen Anteile herauskommen.

Beheben könnte man das Problem durch Kombination der Verfahren: Für die einzelnen Prüfziffern bei Ausweisnummer, Geburtsdatum und Verfallsdatum würde das bisherige Verfahren angewendet, für die Gesamtprüfziffer aber das Verfahren der Kreditkarte. Da beide Verfahren unterschiedliche Anfälligkeiten bezüglich Vertauschungen haben, würde eine solche in jedem Fall auffallen. Offenbar hätte man bei der Einführung besser Informatiker zurate gezogen ...

Was steckt dahinter?

Sie haben es bereits bemerkt: Alles dreht sich in diesem Kapitel wieder um Codierung: die Darstellung einer Informationseinheit durch eine Folge von Symbolen – in der Informatik meistens aus dem sehr eingeschränkten Alphabet „0“ und „1“.

Man kann nun jede mögliche Folge von Symbolen (dem Codewort) einer bestimmten Informationseinheit zuordnen. Auf diese Weise können wir etwa ein Alphabet aus acht unterschiedlichen Zeichen durch je 3 Bit codieren:

A	B	C	D	E	F	G	H
000	001	010	011	100	101	110	111

Die Folge 011000010111 lässt sich so in „DACH“ übersetzen. Jedes einzelne Bit ist dabei relevant – wird es falsch übertragen, so kommt eine andere Nachricht heraus. 011001010111 steht zum Beispiel für „DBCH“.

Das kennen wir auch in unserer Sprache, sichtbar gemacht oft von unseren jüngsten Erdenbewohnern, die neue Begriffe mit dem ihnen verfügbaren Sprachschatz erschließen. So merkt sich der eine oder andere, dass ein Waldzwerg Lieferant für Eisenplatten ist, weil Papa ’mal so etwas vom Walzwerk geholt hat. Andere wundern sich, warum Apfelschalen einerseits fest sind, es sie aber andererseits auch als Getränk gibt („Apfelschorle“) ...

Richard Wesley Hamming (1915–1998) war als Mathematiker zunächst mit Berechnungen für das Militär beschäftigt und nutzte dabei bereits Computer. Später arbeitete er u. a. zusammen mit Claude Shannon an den Bell Labs an Methoden zur sicheren Informationsübermittlung. Den Turing Award erhielt er unter anderem für seine fehlerkorrigierenden Codes.

Andere Begriffe würde man nie verwechseln: „Stoppschild“ oder „Boot“ oder „Haus“ – es gibt keine anderen Wörter, die ähnlich klingen, aber eine unterschiedliche Bedeutung haben. Richard Hamming hat diese Erkenntnis auf Codes in der Informatik übertragen: Ein Code ist gegen Fehler sicherer, wenn die verwendeten „korrekten“ Symbolfolgen möglichst unterschiedlich sind.

Versuchen wir unser Glück mit folgender Codetabelle:

A	B	C	D	E	F	G	H
000000	000111	011001	011110	101010	101101	110011	110100

Für das Wort „DACH“ benötigen wir nun doppelt so viele Bits wie vorher:

011110000000011001110100

Wenn sich nun ein einzelner Fehler einschleicht, ergibt sich vielleicht:

011110001000011001110100

Versucht nun jemand die Nachricht zu lesen, kann er das „D“ vorne und „CH“ hinten identifizieren. Das zweite Zeichen findet er aber nicht in der Zeichentabelle: Das Codewort 001000 gibt es nicht. Nun kann man raten, welches Zeichen wohl gemeint war. Dazu vergleichen wir die empfangene Zeichenfolge mit allen möglichen korrekten Zeichenfolgen:

A	B	C	D	E	F	G	H
000000 001000	000111 001000	011001 001000	011110 001000	101010 001000	101101 001000	110011 001000	110100 001000

Dabei gibt es einen eindeutigen Sieger – das A. Das entsprechende Codewort unterscheidet sich nur an einer Stelle von der empfangenen Nachricht, und wenn wir davon ausgehen, dass es deutlich wahrscheinlicher ist, dass nur ein Bit verfälscht wurde als mehrere Bits, können wir die Nachricht korrekt zu „DACH“ korrigieren.

In der verwendeten Codetabelle unterscheiden sich zwei beliebige Codeworte immer an mindestens drei Stellen. Man spricht hier von einer Hamming-Distanz von 3. Auf diese Weise gibt es bei einem einzelnen Bitfehler immer ein einzelnes Codewort, das die höchste Übereinstimmung mit der empfangenen Nachricht hat.

Die kleinste Hamming-Distanz, die zwei beliebige Codeworte einer Codetabelle haben, bestimmt die Hamming-Distanz des gesamten Codes.

Codes mit einer höheren Hamming-Distanz sind also noch robuster gegenüber Fehlern, da die korrekten Codeworte so unterschiedlich sind, dass ggf. auch mehrere Fehler erkannt und korrigiert werden können.

Versuchen Sie doch einmal, die Hamming-Distanz der folgenden Codetabelle auszuknobeln. Wie viele Fehler können korrigiert werden?

A	B	C	D	E	F	G	H
00000000000	00000111111	00111000111	00111111000	11001001011	11001110100	11110001100	11110110011

Die Codes haben untereinander mindestens die Hamming-Distanz 6. In einer auf diese Weise codierten Nachricht „dürfen“ pro Zeichen 2 Bits falsch übertragen werden. Drei Fehler können immer noch erkannt, aber eventuell nicht mehr korrekt korrigiert werden: Empfängt man zum Beispiel die Bitfolge 11010001111, kann hier mit gleicher Wahrscheinlichkeit ein „E“ oder ein „G“ gesendet worden sein – zu beiden hat die empfangene Nachricht die gleiche Hamming-Distanz von 3.

Wenn sich mehr als 3 Fehler einschleichen, würde man dann falsch korrigieren. Werden zum Beispiel die letzten 4 Bits für „B“ falsch übertragen und es kommt 00000110000 an, würde man am ehesten ein „A“ annehmen.

So wie bei der binären Magie kann man sich aber auch hier entscheiden: entweder

- 2 Fehler erkennen und korrigieren und 3 Fehler noch erkennen

oder gleich

- 5 Fehler nur erkennen, dafür aber keinerlei Korrekturversuche unternehmen.

Fehlerkorrektur von AudioCDs

Das Prinzip, dass die Fehlerkorrektur so gut sein muss, wie die Anwendung es erfordert, kann man auch bei Audio-CDs erkennen: Dort ist (zumindest im Musikteil) keinerlei Korrektur vorgesehen, denn im Zweifel wurde von den Entwicklern eine kurze Unterbrechung oder ein winziges Knistern als akzeptabel angesehen. Daher kann es beim Einlesen oder „Grabben" einer Audio-CD auch zu unterschiedlichen Ergebnissen kommen – auch wenn die Information eigentlich digital gespeichert ist.

Sie sehen, dass unser Experiment vom Anfang des Kapitels durchaus einen guten Einblick in die echte Magie der Informatik ermöglichte: Indem wir die Anzahl der X in einer Zeile oder Spalte gerade hielten, haben wir dafür gesorgt, dass alle gültigen Zeilen und Spalten die Hamming-Distanz 2 besitzen. Für ganze Quadrate gilt (wegen der Kombination aus Zeilen und Spalten) sogar die Hamming-Distanz von 3, wodurch ein einzelner Fehler korrigierbar ist.

Die Wahl der richtigen Hamming-Distanz und damit Fehlertoleranz eines Codes kann individuell je nach erwarteter Fehlerwahrscheinlichkeit und der Fatalität eines Fehlers angepasst werden:

Bei der Übertragung von Sprache über das Haustelefon ins Nachbarzimmer ist kaum ein Fehler zu erwarten, und wenn er doch auftritt, knackt die Leitung allenfalls – man wird daher auf eine Korrektur wahrscheinlich komplett verzichten.

Die Signalisierung zu einer mehrere Lichtminuten entfernten Raumsonde muss dagegen mit immensen Störquellen kämpfen. Außerdem kann ein falsches Signal unter Umständen dazu führen, dass die Antenne falsch ausgerichtet wird und die Sonde damit verloren geht. Hier wird man ein Maximum an Sicherungen einfügen.

Resümee

In Kapitel 6 haben wir Informationen so codiert, dass sie möglichst kompakt übertragen oder gespeichert werden konnten. Wir haben zu diesem Zweck die Redundanz vermindert.

In diesem Kapitel haben wir erforscht, dass Redundanz andererseits sinnvoll und manchmal sogar notwendig für eine sichere Übermittlung und Speicherung von Daten ist. Dazu ist allerdings die „natürliche" Redundanz meistens ungeeignet – da sie nicht konsequent umgesetzt ist, wie man an sehr ähnlichen Wörtern in unserem Sprachschatz erkennen kann.

Als Informatiker versucht man daher am besten, zunächst die Redundanz zu eliminieren (z. B. mit dem Huffman-Verfahren) und danach wieder gezielt hinzuzufügen, um die maximal mögliche Sicherheit für die zur Verfügung stehende Datenmenge zu erreichen.

Zauberanleitung: So führen Sie das Kunststück vor

Sie benötigen 36 der Karten mit X auf einer und O auf der anderen Seite. Diese können Sie aus Kopiervorlage 14.K1 oder aus den Bastelbögen ausschneiden. In der Version aus der Kopiervorlage müssen die Rückseiten noch durch Falten hergestellt und die Karten zusammengeklebt werden, während die Bastelbögen bereits passende Vorder- und Rückseiten haben.

Kommen wir nun aber zum eigentlichen Zaubertrick: Suchen Sie Freiwillige unter Ihren Zuschauern.

Geben Sie diesen 25 Karten und fordern Sie sie auf, ein möglichst kompliziertes Quadrat aus 5×5 Karten zu legen, etwa auf einem Tisch. Achten Sie aber darauf, dass auf der Unterlage noch Platz für eine sechste Zeile und Spalte bleibt. Abbildung 14.10 zeigt ein Beispiel für das, was Ihre Freiwilligen gelegt haben könnten.

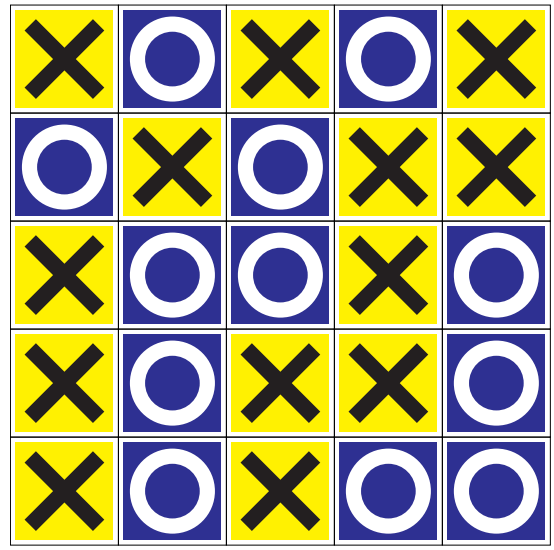


Abbildung 14.10
Gelegtes Muster

Sagen Sie dann etwas wie: „Es sind ja noch Karten übrig – ich werde die Sache noch etwas schwerer machen, indem ich zufällig noch eine sechste Zeile und Spalte dazulege."

Legen Sie dann auch tatsächlich eine 6. Zeile und Spalte. Dabei gehen Sie aber nur scheinbar zufällig vor: Sie achten darauf, dass die Anzahl von „X" in jeder Zeile und in jeder Spalte geradzahlig ist (0, 2, 4 oder 6). Wenn also in einer Zeile zum Beispiel 3 X und 2 O liegen, ergänzen Sie ein X. Wenn 2 X und 3 O liegen, ergänzen Sie ein O. Abbildung 14.11 zeigt ein Beispiel für die zusätzlich von Ihnen angefügte linke und untere Spalte. Zur Verdeutlichung gibt die roten Ziffer die Anzahl der X in der jeweiligen Zeile bzw. Spalte an. Alle sind gerade.

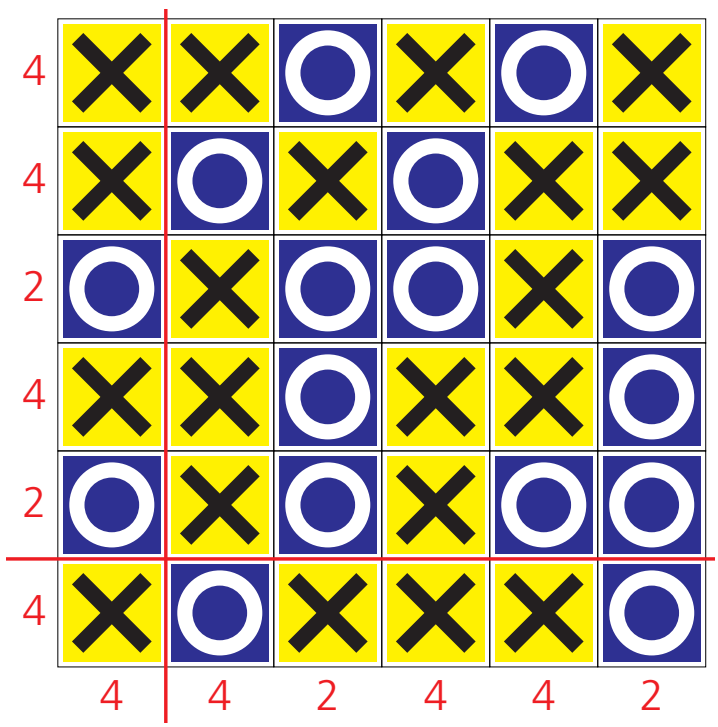


Abbildung 14.11
Muster mit „zufällig" angelegter linker und unterer Reihe

Wenden Sie sich nun ab und fordern einen Freiwilligen auf, eine Spielkarte möglichst unauffällig umzudrehen.

Danach identifizieren Sie diese augenblicklich: Es gibt nun genau eine Zeile und eine Spalte mit einer ungeraden Anzahl von X. Die gesuchte Karte ist diejenige im Schnittpunkt. Im Beispiel befindet sie sich in der zweiten Zeile und dritten Spalte, wie in Abbildung 14.12 gezeigt.

Wenn Sie die Rolle des Magiers noch etwas intensiver auskosten möchten, schauen Sie vor der Verkündung der korrekten Antwort Ihrer freiwilligen Person tief in die Augen und behaupten, dass es sich in Wahrheit um einen Gedankenlesetrick handle. Spielen Sie dann ruhig noch etwas mit Aussagen wie: „Ich merke, dass Sie versuchen, extra nicht an die umgedrehte Spielkarte zu denken, aber das gelingt Ihnen nicht – ich spüre ganz deutlich, dass Sie besonders intensiv an Zeile zwei nicht denken ... Wenn ich nun mit dem Finger über die Spalten streiche – ha, bei der dritten Spalte zucken sie regelmäßig, jetzt habe ich die Karte herausgefunden ...“

Auf diese Weise wird die Vorführung noch verblüffender und Ihre Gäste sind umso mehr davon fasziniert, wenn herauskommt, dass Sie nur elementare Informatik angewendet haben ...

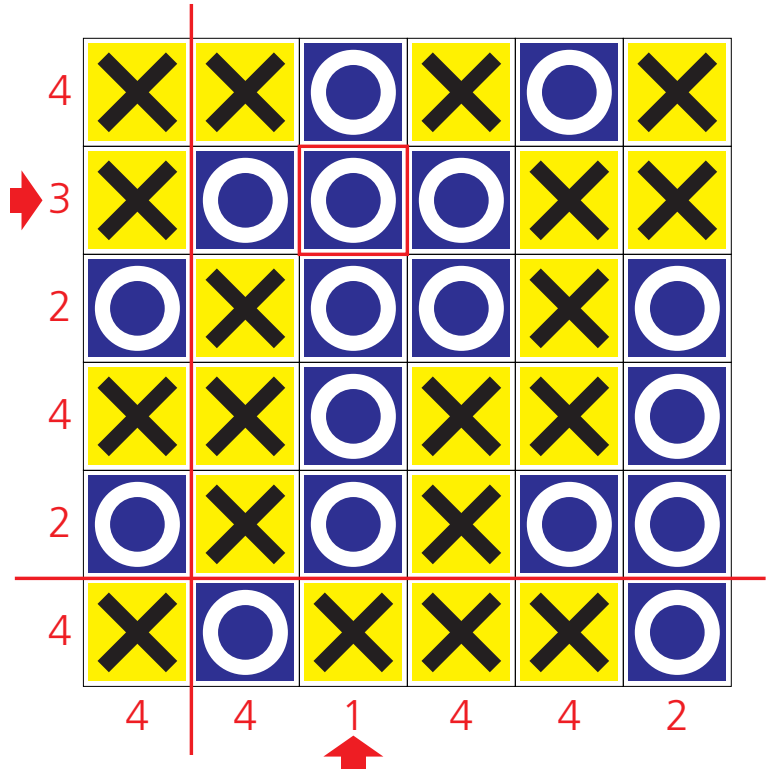
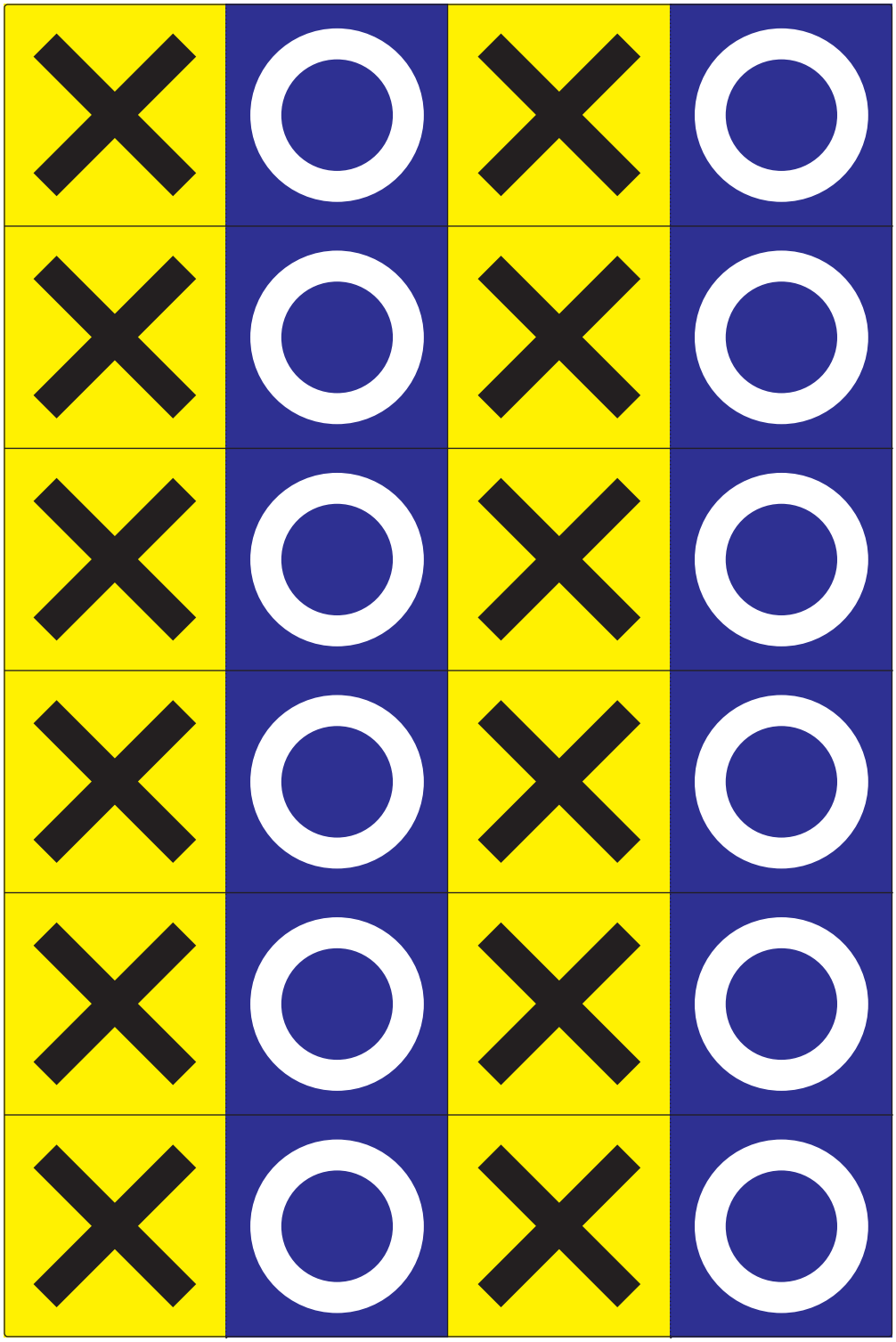
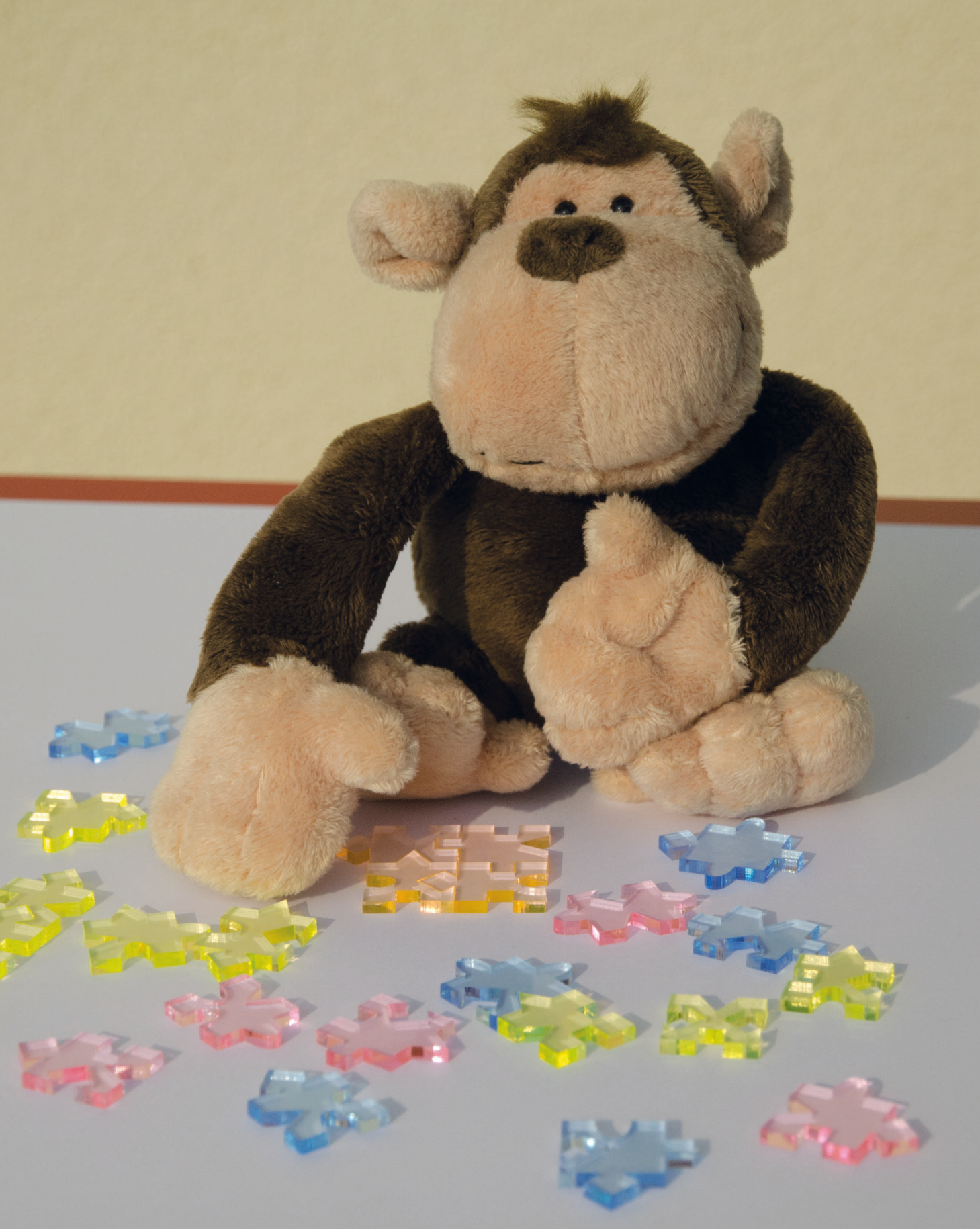


Abbildung 14.12
Die zweite Zeile und die dritte Spalte haben eine ungerade Zahl an X. Offenbar hat dort jemand etwas verändert.

Abbildung 14.K1
Kopiervorlage für XO-Karten. Wird mindestens 3-mal benötigt!





15. Allmächtiger Computer!?

Einführung

Edsger W. Dijkstra ist uns bereits im ersten Kapitel begegnet. Neben brillanten algorithmischen Ideen hatte er auch eine sehr moderne Sicht auf die Wissenschaft Informatik. So prägte er den Satz: „Informatik hat ungefähr so viel mit Computern zu tun wie Astronomie mit Teleskopen.“ Diese Analogie spiegelt natürlich einerseits ein Leitthema dieses Buches wider: Informatik funktioniert auch ohne Computer. Andererseits zeigt sie auch die Grenzen des Werkzeugs Computer auf: Selbst mit dem bestmöglichen Teleskop können Astronomen nicht alles herausbekommen, was sie über die Himmelskörper wissen möchten.

Für den Computer gilt allerdings die weit verbreitete Meinung, das Gerät könne prinzipiell alles, die Limitierung liege allenfalls beim Programmierer. Diesen Widerspruch wollen wir hier aufgreifen und die Frage klären, ob Computer allmächtig sind, wenn sie nur richtig programmiert werden.

Das Affenpuzzle

Auch ein so schwieriges Thema sollte Spaß machen, und daher beginnen wir unsere Forschung mit einem kleinen Spielchen: Schneiden Sie die Spielsteine mit den Affen hinten im Buch aus und beginnen Sie zu puzzeln. Die Steine sind mit unterschiedlichen Rückseiten markiert. Das einfachste Puzzle besteht lediglich aus den vier Teilen mit gelben und dunkelroten Punkten. Sie dürfen die Puzzlesteine beliebig drehen. Ziel ist ein Quadrat, das zwei Puzzlesteine hoch und breit ist und bei dem die Affen im mittleren Bereich sowohl in der Farbe zusammenpassen als auch jeweils vollständig aus Ober- und Unterteil bestehen.

Abbildung 15.1 zeigt ein korrekt gelöstes Puzzle (allerdings mit anderen Spielsteinen, um Ihnen den Spaß nicht zu nehmen). Achtung: Bitte schauen Sie vor und nach dem Puzzeln auf die Uhr und schreiben Sie sich die für die Lösung benötigte Zeit auf!

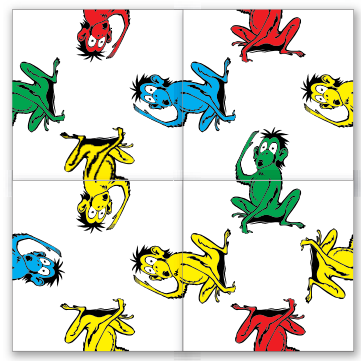


Abbildung 15.1
Gelöstes 2 × 2-Puzzle

Sie haben ganz sicher nicht allzu lange gebraucht, daher wollen wir Sie ein klein wenig mehr herausfordern: Nehmen Sie jetzt zusätzlich die fünf Karten mit gelben und grünen Punkten.

Wenn Sie nun die neun Puzzleteile (also gut doppelt so viele wie im 2 × 2-Fall) wiederum korrekt zusammenfügen möchten – wie lange wird das wohl dauern? Doppelt so lange? Oder vier Mal? Sechzehn Mal?

Nachdem Sie eine Zeit geraten haben, fangen Sie an zu puzzeln und notieren wieder die tatsächlich benötigte Zeit. Kleiner Tipp: Das soeben fertiggestellte 2 × 2-Puzzle ist nicht Bestandteil der 3 × 3-Version.

Sicherlich haben Sie deutlich länger als die 16-fache Zeit benötigt – falls nicht, lassen Sie auch Freunde oder Familienmitglieder auf Zeit puzzeln, um einen zufälligen „Treffer" auszuschließen.

Es ist bemerkenswert, wie viel schwerer ein solches Puzzle wird, wenn man die Zahl der Teile vergrößert. Wenn Sie bereit sind, führen wir dieses Experiment noch weiter: Nehmen Sie nun zusätzlich die sieben Puzzleteile mit gelben und blauen Punkten und versuchen Sie, aus den insgesamt 16 Feldern ein korrektes 4 × 4-Quadrat zu legen. Bitte notieren Sie sich wiederum die Zeit, auch wenn Sie ggf. das Puzzeln unterbrechen.

Auch wenn die Zahl der Teile nun noch nicht einmal verdoppelt wurde, benötigen Sie wahrscheinlich noch einmal deutlich länger. Nehmen Sie noch die Spielsteine mit den gelben und hellroten Punkten hinzu, können Sie zur Krönung auch noch ein 5 × 5-Affenpuzzle lösen. Das schaffen nur die allerwenigsten Menschen ohne Computerunterstützung!

Eine Million Dollar Preisgeld ...
... bekommt derjenige, der für unser Affenpuzzle oder ein beliebiges anderes nicht praktisch berechenbares Problem (Fachbegriff: Problem aus der Klasse NPC oder „nicht deterministisch polynomiell berechenbar") ein Verfahren findet, mit dem es doch praktisch berechenbar wäre. Alternativ dürfen Sie auch mathematisch beweisen, dass es kein solches Verfahren geben kann. Damit hätten Sie nämlich eines der sieben sogenannten „Millennium Problems" gelöst. Das Clay Mathematics Institute in Cambridge hat auf die Lösung ein Preisgeld von 1.000.000 US-Dollar ausgesetzt. Inzwischen gibt es über 70 registrierte Versuche, das Geld einzustreichen, es wurde aber bisher kein einziger der eingereichten und veröffentlichten Beweise als stichhaltig akzeptiert. Das Rennen ist also noch offen ...

Die Zahlen auf den Rückseiten geben Ihnen übrigens keinen Hinweis, aber damit können Sie sich ggf. in Foren oder per Mail leichter über Lösungsansätze verständigen.

Sie überlegen vielleicht, ob man nicht durch geschickteres Vorgehen Zeit sparen könnte. Auch in den vorangegangenen Kapiteln waren einige Probleme mit reinem Ausprobieren kaum lösbar, aber unter Anwendung eines pfiffigen Algorithmus schon.

Auch in diesem Fall könnte es einen solchen guten Lösungsalgorithmus geben – nur hat ihn bis heute niemand gefunden! Das Affenpuzzle gehört zur Gruppe der nicht praktisch berechenbaren Probleme, die ich bereits im ersten Kapitel erwähnt hatte. Da sich schon die klügsten Köpfe der Menschheit vergeblich an einer guten Lösung versucht haben, also einer solchen, die mehr macht als nur geschickt auszuprobieren, geht man davon aus, dass es tatsächlich keine solche Lösung gibt. Allerdings konnte die Unlösbarkeit bisher auch noch nicht bewiesen werden.

In der Informatik gibt es immer wieder Probleme, die „nicht skalieren". Das bedeutet, zur Lösung einer doppelt so großen Aufgabe benötigt man weit mehr als doppelt so viel Zeit. So etwas haben wir zum Beispiel im zweiten Kapitel für das Sortieren von Karten herausgefunden, genauso wie im ersten Kapitel für die Berechnung des kürzesten Wegs von Frankfurt nach München. In vielen Anwendungen muss die Aufgabengröße zum Quadrat genommen werden, um die zur Lösung benötigte Zeit abzuschätzen. Trotzdem sind diese Aufgaben immer noch praktisch lösbar.

Einige Aufgaben sind aber so schwer, dass sie mit Verdopplung der Aufgabengröße gleich unermesslich viel mehr Lösungszeit beanspruchen. Das gilt sowohl für das Lö-

sen von Hand als auch für die Verwendung eines Computers: Das Hinzufügen eines einzigen Puzzlesteins zu einer lösbaren Aufgabe kann dann bedeuten, dass das Problem gar nicht mehr lösbar ist, weil man eine absurd lange Zeit mit der Lösung beschäftigt wäre! Hierbei spricht man dann von nicht praktisch lösbaren Problemen.

Auch andere Disziplinen kennen Probleme, die nicht skalieren: Planen Sie eine Brücke über einen Graben von einem Meter Breite, legen Sie einfach eine Platte aus Metall oder Stein darüber – fertig. Eine Brücke über eine 10 Meter breite Kluft kann aber nicht einfach mit zehn solcher Platten gebaut werden. Hier ist eine statisch genau berechnete Konstruktion erforderlich mit deutlich höherem als dem zehnfachen Aufwand.

Wenn Sie nun per Brücke mehr als 100 Meter überwinden möchten, ist der erforderliche Aufwand wiederum um ein Vielfaches höher. Die größte bisher erreichte Spannweite einer Brücke misst ca. 2.000 Meter. Jeder weitere Meter hat enormen Planungsaufwand und immense Material- und Baukosten zur Folge. Natürlich steigert sich der Aufwand beim Brückenbau immer noch nicht so stark wie bei den hier beschriebenen Problemen der Informatik, insbesondere nicht exponentiell. Allerdings verbilligt sich das Material dort auch nicht in dem Maße wie bei Computern ...

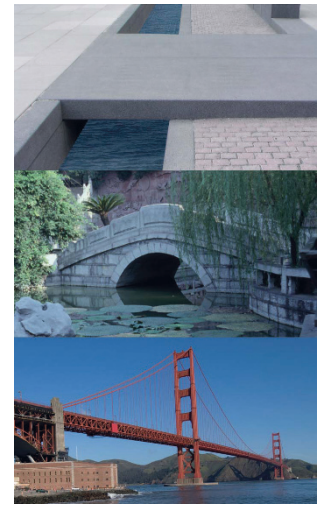


Abbildung 15.2
Verschiedene Brückenkonstruktionen

Was steckt dahinter? Wie schwer ist das Affenpuzzle?

Bevor wir wieder mit einfacheren Experimenten fortfahren, wollen wir ergründen, wie schwer das Affenpuzzle wirklich ist. Diese Analyse gehört in der Informatik zum normalen Geschäft.

Gehen wir erst einmal davon aus, dass wir einfach stur sämtliche Möglichkeiten durchgehen, die Affenpuzzleteile zu legen, bis endlich alle zueinanderpassen. Die n Puzzleteile können in $n!$ Permutationen gelegt werden. Jedes Teil passt dann in einer von vier möglichen Drehrichtungen. Insgesamt gibt es also $n! \cdot 4^n$ Möglichkeiten.

Wenn wir davon ausgehen, dass es sich bei den Puzzles um schwierige Exemplare handelt, also solche, für die es nur vier Lösungen gibt (das ist das Minimum, da ein korrektes Puzzle auch um je 90° gedreht noch korrekt ist), müssen wir im Durchschnitt ein Achtel der Anordnungen durchprobieren, bis wir eine korrekte gefunden haben. Für das Probieren einer Lösung müssen jeweils n Karten gelegt werden. Daher kommen wir auf einen Gesamtaufwand von $\frac{n! \cdot 4^n \cdot n}{8}$ Kartenbewegungen.

Anhand der Tabelle auf der nächsten Seite können wir uns das Ausmaß dieser Formel klarmachen. Um das gewaltige Wachstum mit Größen zu verbinden, die wir uns gerade noch vorstellen können, habe ich in den ersten vier Zeilen noch hinzugefügt, welche (fiktive) Zeit für das Legen der Karten benötigt würde, wenn alle 7,4 Milliarden Menschen der Erde kollektiv mit dem Legen je einer Karte pro Sekunde an der Lösung des Problems arbeiten würden. Danach steht die Zeit, die benötigt würde, wenn alle 3,2 Milliarden Internetnutzer jeweils einen PC zur Verfügung stellten, der in der Lage sei, 100.000.000 Karten pro Sekunde durchzuprobieren.

Nach diesen Zahlen müsste bereits das 4 × 4-Puzzle als nicht lösbar gelten, wenn selbst die gesammelte weltweite Rechenleistung noch länger dafür benötigt, als wir warten möchten ...

Das Alter des Universums ...
... wird hier übrigens mit 13,81 Milliarden Jahren angenommen.

l	n	Aufwand (Anzahl Kartenbewegungen, ggf. Zeit)
2	4	3.072 Menschen: <1s Computer: <1a
3	9	107.017.666.560 Menschen: 14s Computer: <1s
4	16	44.931.349.155.019.751.424.000 Menschen: 770.143 Jahre Computer: 6,5 Tage
5	25	54.575.218.571.258.534.055.702.019.493.068.800.000.000 Menschen: 16.934.152.967.523 Universumsalter Computer: 391.602 Universumsalter
6	36	7.905.099.682.277 .119.717.342.033.983.757.604.896.115.292.972.951.955.046.400.000.000
7	49	1.180.729.832.265.571.546.676.070.502.407.423.580.546.073 .948.103.340.049.125.825.596.717.261.114.299.834.695.680.000.000.000
8	64	345.419.084.924.394.172.191.185.257 .773.432.828.262.854.268.878.975.649.499.687.764.330.512.824.305.201 .652.497.882.047.272.271.334.994.814.759.547.699.200.000.000.000.000
9	81	343.136.621.431.002.337 .786.709.970.485.642.214.722.923.593.384.804.322.752.915.173.234.645 .573.655.033.362.806.997.582.256.301.213.162.919.165.359.539.644.104 .862.215.977.336.452.651.709.655.091.773.440.000.000.000.000.000.000
10	100	187.461.807.654.480 .988.260.853.429.579.919.223.038.131.359.269.917.465.885.032.981.485 .445.345.582.505.986.614.579.612.840.862.698.631.134.479.971.447.239 .541.967.209.937.374.074.125.047.269.586.955.790.037.269.029.982.932 .844.944.630.432.642.493.401.006.080.000.000.000.000.000.000.000.000

Dass hier noch etwas mit unseren Annahmen nicht stimmen kann, wird schon allein daran ersichtlich, dass Sie bei den kleineren Puzzles für die Lösung von Hand deutlich weniger Zeit benötigt haben, als veranschlagt wird, etwa für 3×3, das durchaus in einer ruhigen Stunde machbar ist.

Überlegen wir daher also, wie man geschickter vorgehen könnte. Als Informatiker sind wir in der Lage, recht einfach formal zu beweisen, dass das Affenpuzzle zu den nicht praktisch lösbaren Problemen gehört, und daher wissen wir, dass wahrscheinlich jedes Verfahren etwas mit Ausprobieren zu tun hat. Allerdings kann man geschickt ausprobieren.

Eine Technik dafür nennt sich „Branch and Bound“. „Branch“ (englisch „Verzweigung“) bezieht sich auf das Ausprobieren: Wie in einem Baum, in dem man aus allen Blättern das mit einer bestimmten Herbstfärbung finden muss, verzweigt man zu allen potentiellen Lösungen, um die korrekte zu finden. „Bound“ ist die Schranke: Man versucht möglichst frühzeitig zu erkennen, dass man beim Ausprobieren einen definitiv falschen Zweig erreicht hat und umkehren muss.

Wie könnten wir das beim Affenpuzzle umsetzen? Schauen wir uns dafür doch einfach beim Puzzeln zu! Wir legen nicht alle Puzzleteile vollständig hin, sondern erst eines, dann ein weiteres passendes usw. Falls es kein passendes Teil mehr gibt, wissen wir, dass wir auf dem Holzweg sind, und versuchen, die bereits gelegten Teile zu verändern. Das passiert oft auch schon bei sehr wenigen gelegten Teilen.

Beim Konstruieren der Ausstellung „Abenteuer Informatik“ wurde ich von Gero Scholz daher auf folgende Lösung hingewiesen: Zuerst wird das Teil oben links gelegt, hierfür gibt es alle Möglichkeiten (Anzahl der Teile mal vier Ausrichtungen). Danach legen wir das nächste Puzzleteil rechts davon. Dieses muss ja an das vorhandene Affensymbol angelegt werden. Nur eines von acht Symbolen passt und daher schränkt sich die Zahl der Züge auf etwa 1/8 der noch vorhandenen Kombinationen ein.

Nun könnten wir wieder ein weiteres Teil rechts anlegen, stattdessen puzzeln wir vom Startteil nach unten weiter. Auch hier gilt, dass nur etwa 1/8 der noch vorhandenen Teile passt.

Abbildung 15.3 zeigt die Situation. Legen wir als Nächstes die rot markierte Position, werden die dort möglichen Teile durch das Symbol nach links und das Symbol nach oben jeweils auf 1/8 begrenzt, insgesamt also auf 1/64. Beim 4 × 4-Puzzle besteht daher schon eine hohe Wahrscheinlichkeit, dass bereits jetzt kein Teil mehr passt und wir den Versuch schon nach drei gelegten Teilen als Sackgasse identifizieren können.

In der Abbildung muss auf die Position eine Karte kommen, die einen grünen Affenkopf und im Uhrzeigersinn daneben einen roten Affenhintern aufweist. Davon gibt es im Puzzle lediglich ein Exemplar, das in einer bestimmten Ausrichtung gelegt werden muss. Statt also die restlichen 22 Karten in je vier Positionen auszuprobieren, beschränken wir uns auf das Legen dieses einen Teils.

In Kapitel 8 haben Sie sogar bereits eine Technik kennen gelernt, mit der man prinzipiell die Puzzleteile im Computer so ablegt, dass man mit einem einzigen Zugriff identifizieren kann, ob ein passendes Teil existiert oder nicht: Hashing.

Auch im weiteren Verlauf sorgt man nun dafür, recht früh immer Situationen zu erzeugen, in denen mit hoher Wahrscheinlichkeit eine Sackgasse erkannt wird.

Mit „Branch and Bound“ sieht unsere Tabelle auf der nächsten Seite schon deutlich besser aus (die Zahlen entstehen aus einer komplizierteren statistischen Berechnung und keiner einfachen Formel).

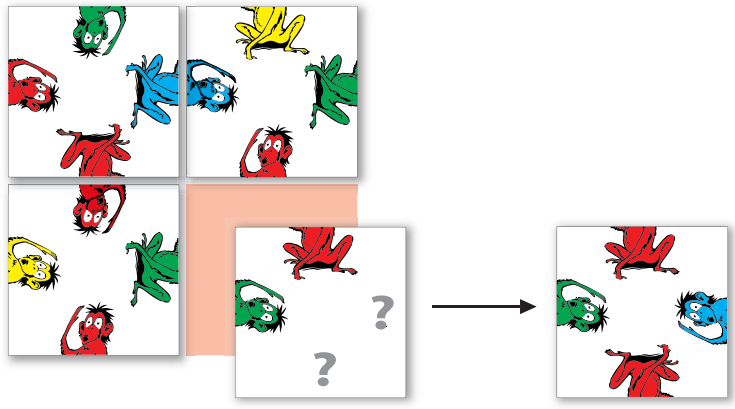


Abbildung 15.3
Es gibt nur eine passende Karte für die rot markierte Position.

l	n	Aufwand (Anzahl Kartenbewegungen, ggf. Zeit)
2	4	129 Menschen: <1s Computer: <1a
3	9	6.773 Menschen: <1s Computer: <1s
4	16	1.965.501 Menschen: <1s Computer: <1s
5	25	11.385.972.942 Menschen: 1s Computer: <1s
6	36	4.482.592.451.432.549 Menschen: 7 Tage Computer: <1s
7	49	319.714.732.333.844.702.623.086 Menschen: 1.370.011 Jahre Computer: 11,6 Tage
8	64	8.846.020.083.295.533.725.639.427.309.614.807 Menschen: 2.744.832 Universumsalter Computer: 876.579.552 Jahre
9	81	175.951.318.252.867.500.823.848.449.208.798.428.774.205.194.798 Menschen: 54.595.961.612.889.107.459 Universumsalter Computer: 1.262.531.612.298 Universumsalter
10	100	4.265.326.330.573 .335.142.293.854.240.283.428.951.958.671.066.972.613.909.968.375.819

Der Zeitaufwand für das manuelle Lösen des 2 × 2-Puzzles berechnet sich übrigens zu etwa einer Minute, wenn wir uns zutrauen, zwei Karten pro Sekunde zu legen, für das 3 × 3-Puzzle bräuchte man alleine rechnerisch 56 Minuten. Das sind meiner Erfahrung nach realistische Einschätzungen. Offenbar haben Puzzle-Experten sogar noch eine bessere Strategie, denn ich habe bereits mehrfach beobachtet, wie von Besuchern der Ausstellung das 4 × 4-Puzzle gelöst wurde – in deutlich unter elf Tagen, die das nach unserer Berechnung in Anspruch nähme. Daher haben Sie auch eine gute Chance, alleine oder mit Freunden das 5 × 5-Puzzle zu schaffen.

Mit einem Supercomputer könnten wir sogar noch bis 6 × 6 gehen, bereits bei 7 × 7 streikt jedoch auch dieser: Unser Affenpuzzle bleibt auch angesichts der „Branch and Bound"-Strategie ein nicht praktisch lösbares Problem.

Das Problem des Fliesenlegers

Noch sind wir mit dem Affenpuzzle nicht fertig. Einem Hersteller von Fliesen gefällt das Dekor so gut, dass er die „blaue Serie“ herstellt: drei Fliesen mit den typischen Affen an den Rändern und blauem Hintergrund wie in Abbildung 15.4. Ein Fliesenleger könnte sich nun die Frage stellen, ob er mit diesen drei Sorten prinzipiell alle möglichen Flächen ausfüllen könnte, wenn er nur genügend davon bestellte.

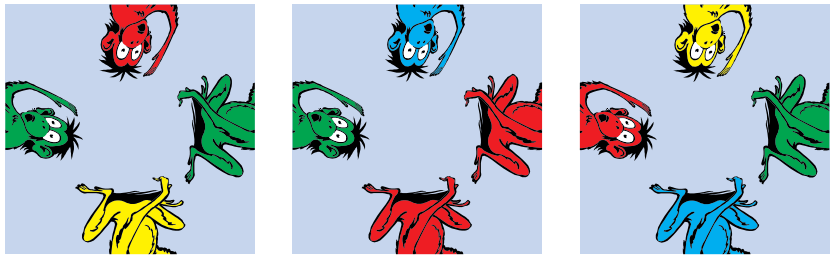


Abbildung 15.4
Die blaue Serie

Probieren Sie es aus: Versuchen Sie eine möglichst große Fläche mit den ausgeschnittenen Puzzleteilen aus dem Bastelbogen bzw. der Kopiervorlage zu legen. Können Sie außerdem irgendwie herausfinden, ob man mit mehr Teilen immer größere Flächen ausfüllen könnte?

Zur Antwort reicht schon eine 3 × 3 Felder große Fläche. Abbildung 12.6 auf der nächsten Seite zeigt sie.

Was ist so bemerkenswert an dieser Anordnung? Die obere Kante passt genau an die untere und die linke an die rechte Kante. Auf diese Weise können Sie beliebig viele dieser 3 × 3-Blöcke aneinanderlegen und so in alle Richtungen beliebig große Flächen ausfüllen.

Vom großen Erfolg der blauen Serie angestachelt, legt der Fliesenhersteller die „rosa Serie“ nach Abbildung 15.5 auf. Versuchen Sie auch hier herauszufinden, wie flexibel man damit beim Auslegen ist.

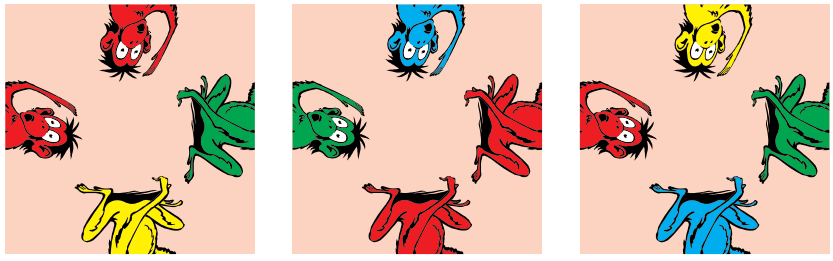
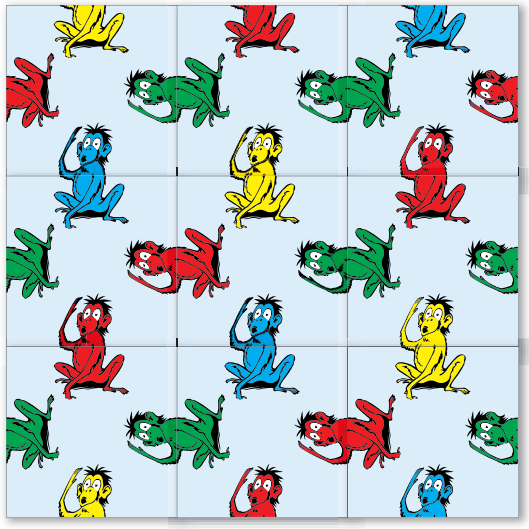


Abbildung 15.5
Die rosa Serie

Abbildung 15.6
3 × 3-„Multikachel“



Sie sehen, dass es nur jeweils ein gelbes und ein blaues Affenvorder- und Hinterteil gibt. Daher ist es ziemlich klar, dass die Affen in einer Reihe wie in Abbildung 15.7 gelegt werden müssen – es gibt keine andere Möglichkeit.

Da nur ein grünes Hinterteil zur Verfügung steht, legen wir es rechts an die beiden grünen Köpfe. Abbildung 15.8 zeigt dies. Obwohl wir nur Puzzleteile gelegt haben, die wir aufgrund der zur Verfügung stehenden Teile legen mussten, passen nun zwei Teile

Abbildung 15.7
Die rosa Affen können nur in dieser Weise gelegt werden.

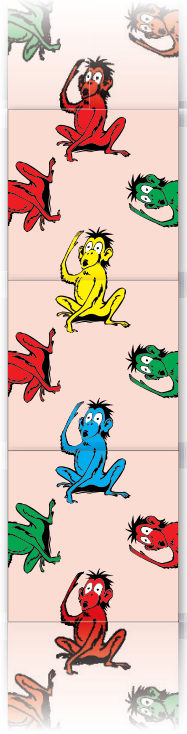
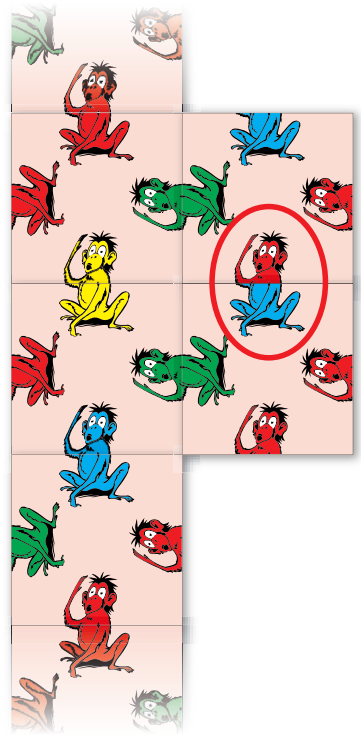


Abbildung 15.8
Widerspruch mit den rosa Affen



nicht zueinander. Für die rosa Serie können wir daher mit Bestimmtheit sagen, dass man damit auf keinen Fall beliebig große Flächen ausfüllen kann.

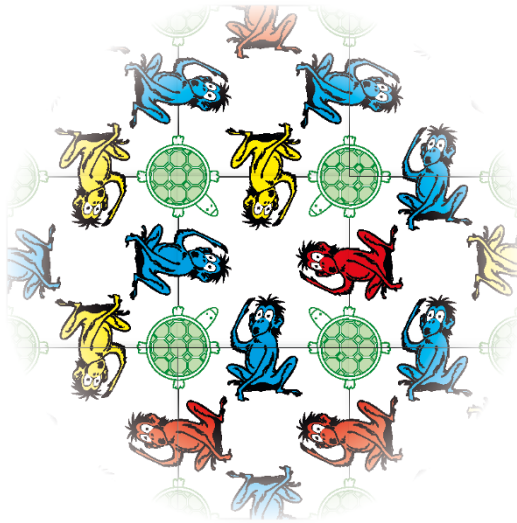
Nun könnte man die gewonnenen Erfahrungen ja als Grundlage für ein Verfahren nehmen, das für beliebige Sätze von Puzzleteilen bestimmt, ob man damit beliebig große Flächen auslegen kann:

```
Versuche, die Kacheln regelkonform aneinanderzulegen.

Falls dabei ein Rechteck entsteht, bei dem die obere und
untere sowie die linke und rechte Kante aneinander passen
          → Antwort JA.

Falls trotz Ausreizung aller Möglichkeiten eine Karte
nicht passt
          → Antwort NEIN.
```

Es scheint so, als ob man mit genügend Geduld immer auf eine der beiden Alternativen stößt: Ein Programm, das auf diese Weise vorgeht, würde ggf. sehr lange brauchen, aber irgendwann entweder das Ergebnis „JA“ oder „NEIN“ hervorbringen ... oder doch nicht?

Um das zu ergründen, habe ich eine weitere Sorte von Affenpuzzleteilen für Sie vorbereitet: mit Schildkröten in den Ecken. Die zusätzliche Regel beim Legen ist nun, dass jede Schildkröte genau einen Kopf haben muss. Einen korrekten Ansatz sehen Sie in Abbildung 15.9.

Abbildung 15.9
Ausschnitt aus einem Affenpuzzle mit Schildkröten



Versuchen Sie nach dem soeben bestimmten Verfahren vorzugehen, um zu entscheiden, ob die Teile tauglich für beliebig große Flächen sind.

Nach einigem Puzzeln haben Sie es sicherlich geschafft, alle Teile irgendwie zu verwenden und trotzdem noch nicht auf einen Widerspruch zu stoßen. Ein zueinander passendes Rechteck haben Sie aber sicherlich trotzdem nicht gefunden.

Wenn Sie Lust haben: Kopieren Sie noch viel mehr dieser Puzzleteile und erzeugen Sie auf diese Weise ein riesiges Affenpuzzle. Ich garantiere Ihnen: Selbst wenn Sie von nun an jede freie Minute in die Erstellung des Affenpuzzles stecken – mit dem gerade ausgeknobelten Verfahren kommen Sie weder auf die Antwort „JA“ noch auf „NEIN“.

Wenn Sie mir das nicht glauben, sehen Sie sich Abbildung 15.11 an. Sie zeigt auf der kompletten nächste Doppelseite einen Ausschnitt aus einem solchen Puzzle. Versuchen Sie, in diesem Puzzle ein entsprechendes Rechteck zu finden. Sie können dafür mit den Händen einen Ausschnitt definieren, wie in Abbildung 15.10.

Vielleicht glauben Sie es mir auch einfach: Mit den acht neuen Puzzleteilen können Sie prinzipiell unendlich weiterpuzzeln, werden aber definitiv nie Rechtecke finden, die

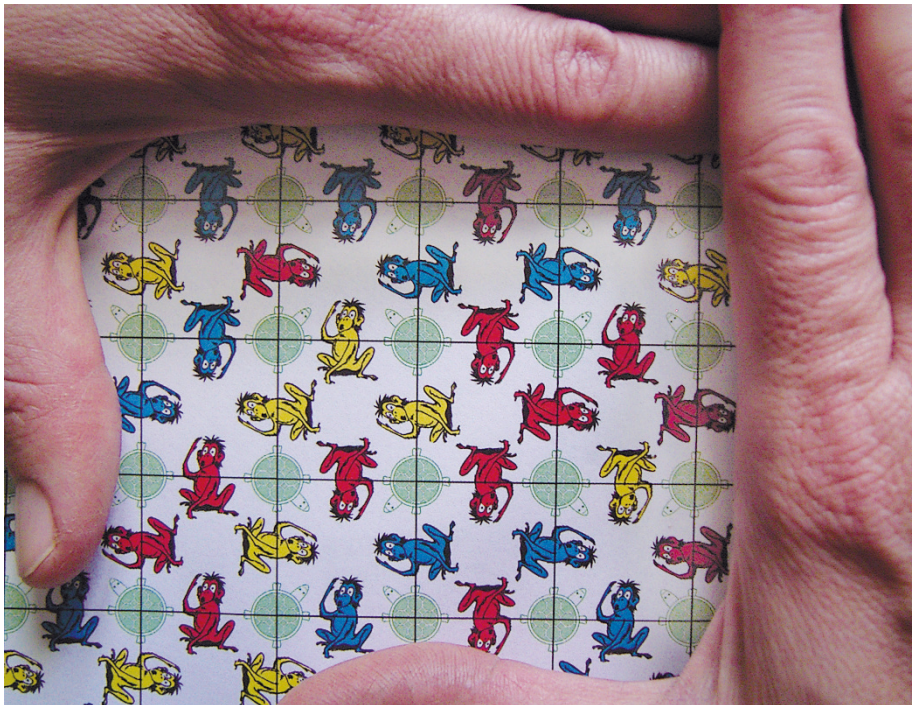


Abbildung 15.10
Suche nach einem Rechteck, das sich periodisch fortsetzt.

sich periodisch wiederholen. Ein Programm, das nach dem oben aufgestellten Verfahren vorgeht, würde nie fertig werden und nie auch nur ein so einfach klingendes Ergebnis wie „JA“ oder „NEIN“ hervorbringen.

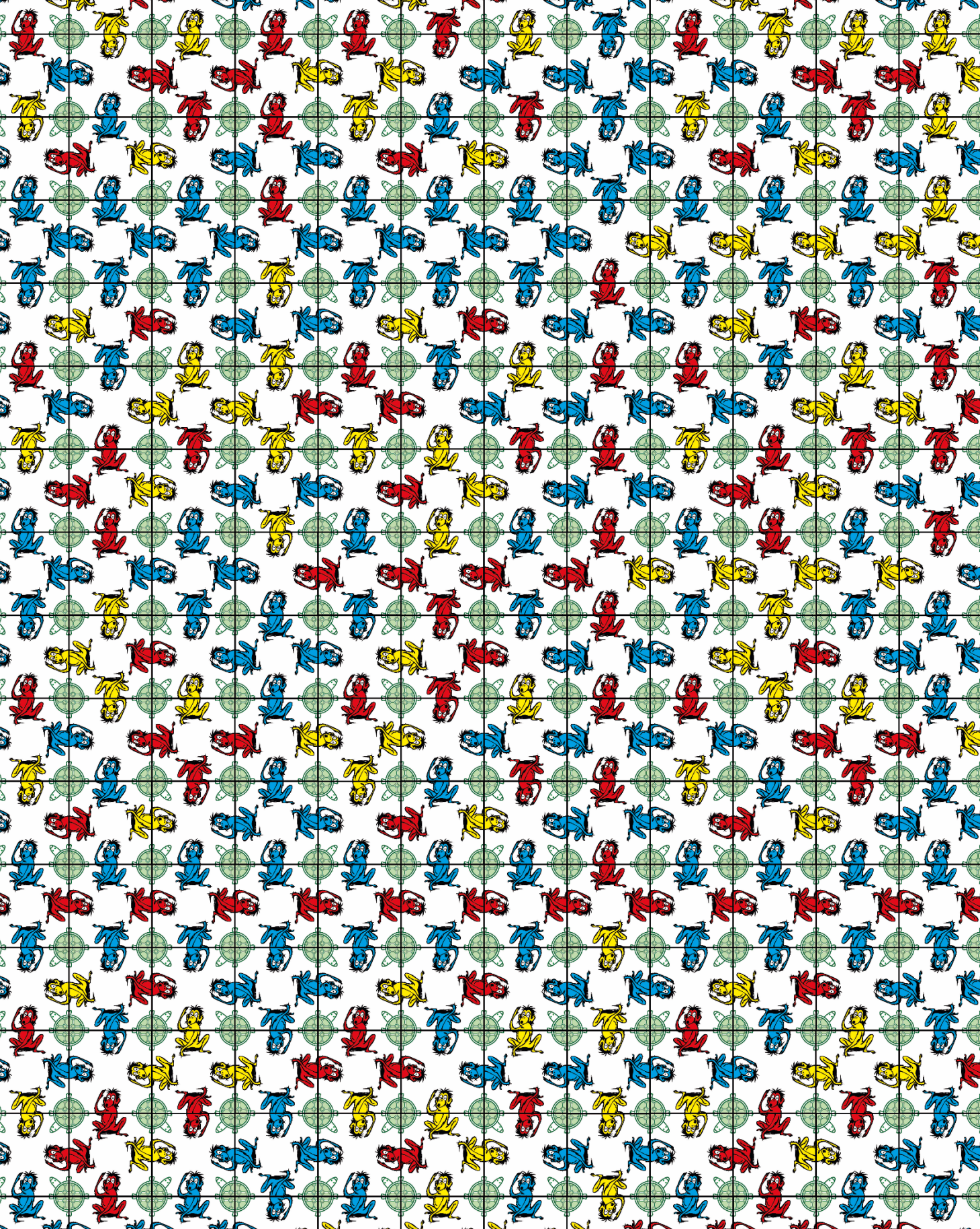
Effektiv hat sich mit dem speziellen Affenpuzzle-Satz (bzw. einem geometrischen Pendant) der Mathematiker Raphael M. Robinson auseinandergesetzt und bewiesen, dass es sich um eine unendliche, nichtperiodische Kachelung handelt. Schlauberger können nun einwenden, dass man doch einem Computer auch den Beweis des Herrn Robinson beibringen könnte, so dass der das Muster erkennt und auch bei den Affenpuzzlesteinen mit Schildkröte mit „JA“ antwortet.

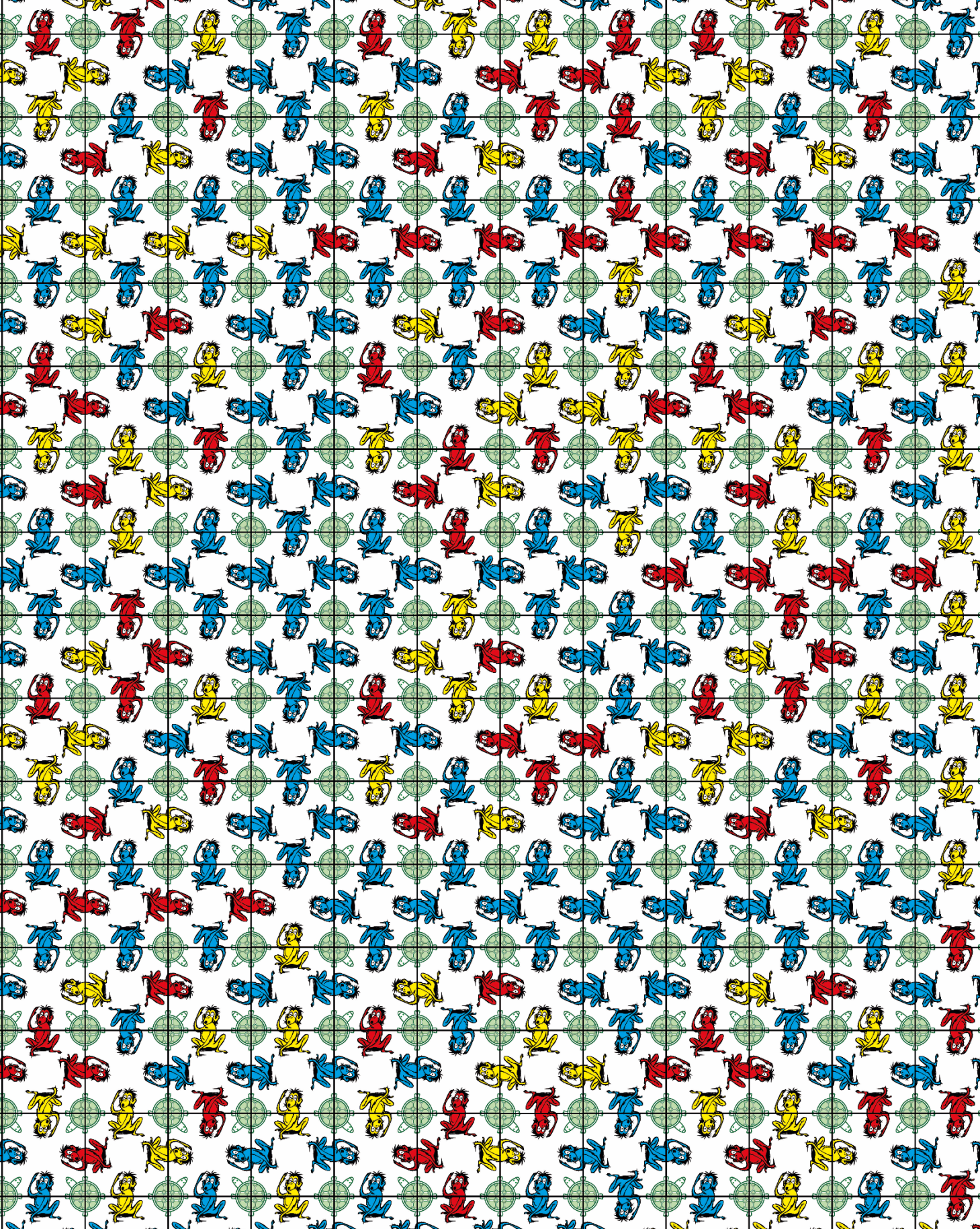
Tatsächlich ist das völlig richtig! Allerdings gibt es quasi unendlich viele Kombinationsmöglichkeiten für Affenpuzzle-Fliesensets, die wieder andere Wiederholungsmuster aufweisen als das nach Robinson, und es ist unmöglich, diese alle vorauszusehen und dem Computer beizubringen. Nur ein Mensch kann sich mit seiner Kreativität immer wieder auf neue Sets einstellen und jeweils definitiv entscheiden, ob man damit beliebig große Flächen ausfüllen kann.

Die hier geschilderte Problemstellung kann also umfassend von keinem Computer der Welt gelöst werden und es wird auch nie einen Computer geben, der sie lösen kann! Ist es nicht beruhigend zu wissen, dass manche Dinge doch dem Menschen vorbehalten sind?

Die Terminierung
Computerprogramme sollen natürlich im Normalfall irgendwann einmal mit ihrer Berechnung fertig werden und mit einem Ergebnis aufwarten. In der Fachsprache heißt das Terminierung.
Ihnen bekannte Formen von nicht terminierenden Programmen sind die berühmten „Abstürze“, bei denen ein Computer „gar nichts mehr macht“. Effektiv macht der Computer selbstverständlich doch noch etwas, aber er führt eventuell immer wieder den gleichen Code in einer Endlosschleife aus und reagiert nicht mehr auf Ihre Eingaben – das Programm bzw. Betriebssystem terminiert nie.
Ein wichtiger Aspekt der theoretischen Informatik ist die Betrachtung, ob ein Programm immer (also für jede mögliche Eingabe) terminiert.

Abbildung 15.11
Riesiges Affenpuzzle
(nächste Doppelseite)





Was steckt dahinter? Nichtberechenbare Probleme

Computer können nicht alles: Das haben wir mit dem Affenpuzzle anschaulich gemacht. Die Frage des Fliesenlegers nach der Kachelung beliebig großer Flächen ist in der theoretischen Informatik als Domino-Problem bekannt. Es kann von Computern nicht gelöst werden. Besser gesagt: Es gibt immer Eingaben (in diesem Fall Sätze von Fliesen), für die der Computer keine Entscheidung „JA“ oder „NEIN“ treffen kann. Man sagt dann: „Das Problem ist nicht entscheidbar.“

Beweisen wollen wir diese Unentscheidbarkeit allerdings für eine andere Fragestellung: das sogenannte Halteproblem. Wie oben bereits in der Randspalte erwähnt, ist es bei Software sehr wichtig, dass diese irgendwann anhält bzw. terminiert. Es wäre nun schön, wenn man die Frage „das Programm hält irgendwann“ oder „das Programm hält nicht immer“ automatisch beantworten lassen könnte.

Wir nehmen daher an, unter Aufbietung aller unserer Talente schreiben wir ein Programm HÄLTS?, dem man ein anderes Programm zum Testen übergibt. HÄLTS? ermittelt daraufhin, ob dieses übergebene Programm immer anhält, und antwortet „JA“, wenn das der Fall ist, und sonst „NEIN“.

Zur Veranschaulichung verwenden wir zwei einfache Programme:

```
PROGRAMM A
Zeile 10: FALLS X=U GEHE ZU Zeile 10
Zeile 20: ENDE
```

```
PROGRAMM B
Zeile 10: FALLS X=X GEHE ZU Zeile 10
Zeile 20: ENDE
```

Sie sehen, dass Programm A testet, ob X und U identisch sind. Nur falls das zutrifft, springt die bedingte Ausführung wieder zur Zeile 10. Da sich der Computer kein X für ein U vormachen lässt, geht er im Programm einfach weiter und führt Zeile 20 aus, die das Programm beendet. Ein Aufruf von HÄLTS? mit diesem Programm führt also zu:

```
HÄLTS? (PROGRAMM A) = JA
```

Anders sieht die Sache beim zweiten Programm aus. Hier wird in Zeile 10 wieder zu Zeile 10 verzweigt, falls X identisch mit X ist (was natürlich immer stimmt). Damit verliert sich das Programm in einer Endlosschleife.

```
HÄLTS? (PROGRAMM B) = NEIN
```

Der Trick ist nun, unser eigenes Superprogramm in diese bedingte Ausführung einzubauen:

```
PROGRAMM C
Zeile 10: FALLS HÄLTS? (PROGRAMM C) = JA GEHE ZU Zeile 10
Zeile 20: ENDE
```

Wir haben ja bereits festgestellt, dass in Zeile 10 die entscheidende Bedingung steckt: Ist sie wahr, verliert sich das Programm in einer Endlosschleife, ist sie falsch, terminiert es.

Andere Möglichkeiten gibt es auf die einfache Frage „Hält das Programm an?“ nicht. Da die Bedingung bei Programm C etwas knifflig ist, probieren wir die beiden möglichen Thesen im Detail durch:

These 1: PROGRAMM C hält an.

- Folge: Die Bedingung „HÄLTS? (PROGRAMM C) = JA“ ist immer wahr. Damit geht das Programm C bei Zeile 10 in eine Endlosschleife.
- Konsequenz: Das Programm C hält nicht an, was der These 1 widerspricht, die daher falsch sein muss.

These 2: PROGRAMM C hält nicht an.

- Folge: Die Bedingung „HÄLTS? (PROGRAMM C) = JA“ ist immer falsch. Damit überspringt das Programm C die Zeile 10 und hält bei Zeile 20 an.
- Konsequenz: Das Programm C hält an, was der These 2 widerspricht, die daher falsch sein muss.

Beide Thesen sind falsch, weitere kann es nicht geben. Es muss also irgendeine andere Voraussetzung falsch sein. Die einzige Möglichkeit dafür ist die Voraussetzung, dass wir überhaupt in der Lage sind, ein Programm HÄLTS? zu entwickeln.

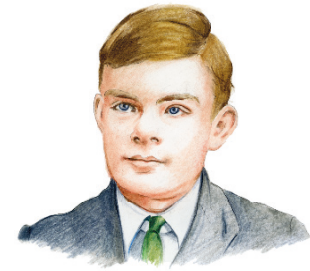
Die einzig mögliche Folgerung aus unseren Schlüssen ist daher, dass es solch ein Programm HÄLTS? nicht gibt und auch nie geben kann.

Herzlichen Glückwunsch! Sie haben nun eine ganz wichtige Erkenntnis der theoretischen Informatik nachvollzogen: Ein Computer kann für beliebige Software noch nicht einmal entscheiden, ob diese immer anhält oder nicht.

Noch viel schwerer ist zum Beispiel der Nachweis, ob ein bestimmter Teil eines Computerprogramms das macht, was es soll. So gibt es bei kritischen Softwaresystemen, etwa in Flugzeugen oder Atomkraftwerken, die Pflicht, sie zu validieren, was so viel bedeutet wie ihre Fehlerfreiheit zu beweisen.

Da ein Computer noch nicht einmal potentielle Endlosschleifen herausfinden kann, ist er mit der Validierung noch viel mehr überfordert. Hier müssen immer Menschen – meistens Informatiker oder Mathematiker – heran.

Um Missverständnisse auszuschließen: Auch ein Mensch kann für diese Aufgabenstellungen kein allgemeines Lösungsverfahren angeben – das geht nicht! Und wenn es ginge, könnte man die Durchführung auch einem Computer übertragen. Der Mensch ist aber in der Lage, für die meisten speziellen Eingaben (also z. B. Programme beim



Alan Turing (1912–1954) ist einer der wichtigsten Väter der modernen Informatik. Er hat sich nicht nur mit Computern und deren Bau beschäftigt, sondern insbesondere immer die Möglichkeiten und die Grenzen der neuen Technik theoretisch analysiert. Dazu schuf er ein Modell für Maschinen, das heute als „Turing-Maschine“ bekannt ist. Er bewies, dass sein Modellcomputer quasi stellvertretend für alle realen Geräte eingesetzt werden kann, um bestimmte Eigenschaften zu analysieren. So bewies er dann auch anhand seiner Turing-Maschine, dass die Lösung des Halteproblems von Computern nicht berechnet werden kann.

Halteproblem) durch Kreativität herauszufinden und zu beweisen, ob diese anhalten oder nicht.

Es soll nicht verschwiegen werden, dass einige – fast immer konstruierte – Aufgabenstellungen bisher auch von Menschen nicht gelöst werden konnten. Für einen Teil davon wurde sogar bewiesen, dass sie überhaupt nicht lösbar sind.

Resümee

Wir wissen nun, dass der Computer nicht allmächtig ist. Einige Problemlösungen bleiben dem Menschen vorbehalten, weil Computer einfach zu lange benötigen würden, und andere Probleme bedürfen der menschlichen Lösung, weil Computer sie aus Prinzip nicht lösen können.

Was bringt uns diese Erkenntnis außer der tröstlichen Botschaft, dass der Informatiker sich nicht selbst überflüssig macht? Natürlich: Wenn wir wissen, dass ein Problem mit dem Computer nicht lösbar ist, versuchen wir das auch erst gar nicht. Wir könnten dafür versuchen, das Problem etwas umzuformulieren, so dass es einerseits mit dem Computer lösbar wird und andererseits uns die Lösung trotzdem weiterhilft.

So könnte man beim Affenpuzzle über Jokerkarten nachdenken, die universell einsetzbar sind. Bereits zwei Jokerkarten machen auch ein 6×6-Puzzle in Sekunden lösbar.

An der Front der automatischen Programmbeweise gibt es teilautomatische Beweiser: Immerhin ist der Computer ein sehr schnelles und konsequentes Werkzeug. Ein teilautomatischer Beweiser kann daher über lange Strecken Software auf Fehler hin analysieren, muss allerdings ab und zu ein Teilproblem an den Bediener weitergeben. Auf diese Weise schafft man den gesamten Beweis sozusagen im Zusammenspiel von Intelligenz auf der einen und schierer Rechenleistung auf der anderen Seite.

Dijkstra hatte also recht ...

Abbildung 15.K1
Affenpuzzle

Abbildung 15.K2
Affenpuzzle

Abbildung 15.K3
Affenpuzzle und zwei Bananenjoker

Abbildung 15.K4
Affenpuzzle-Kacheln
3-mal kopieren

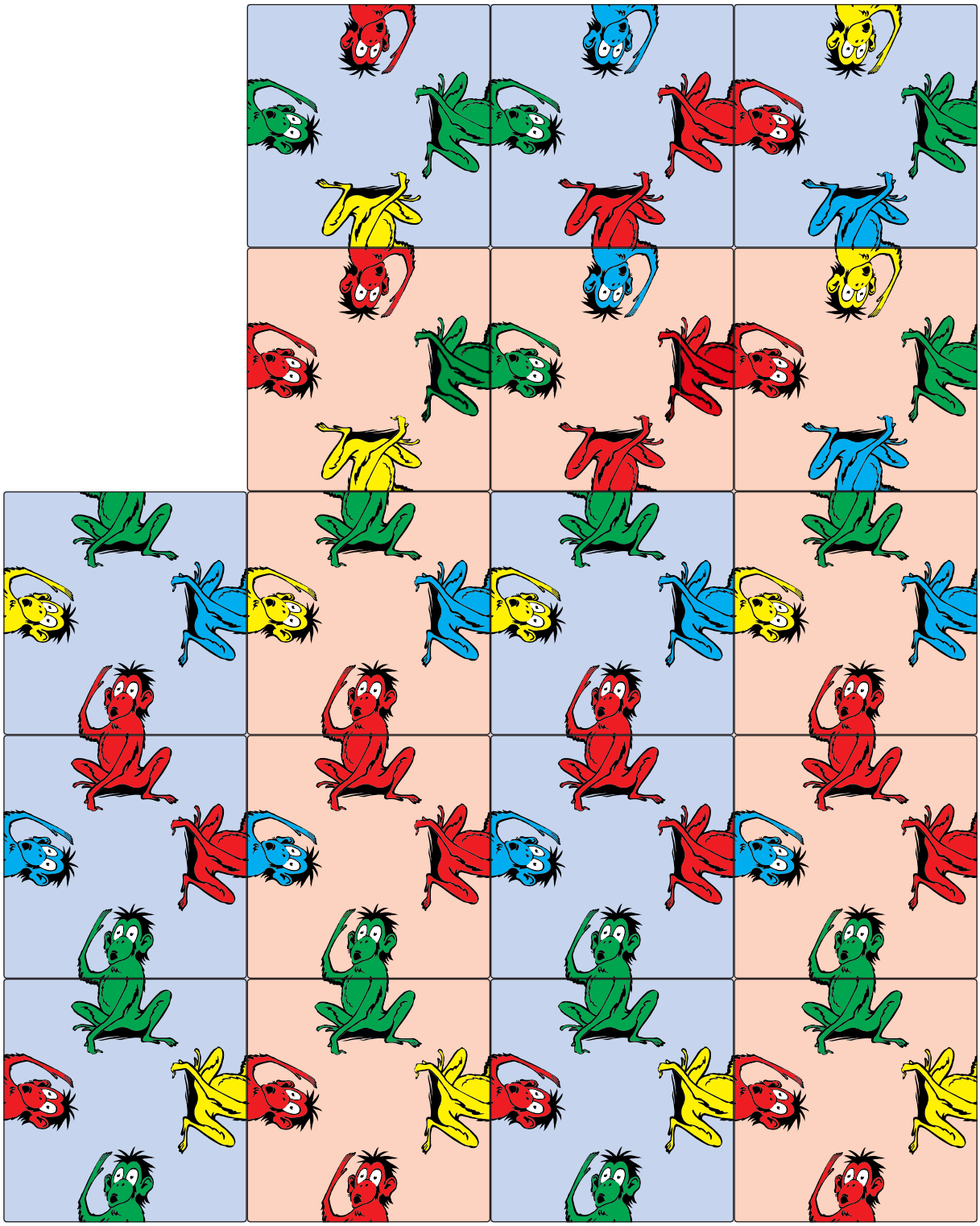
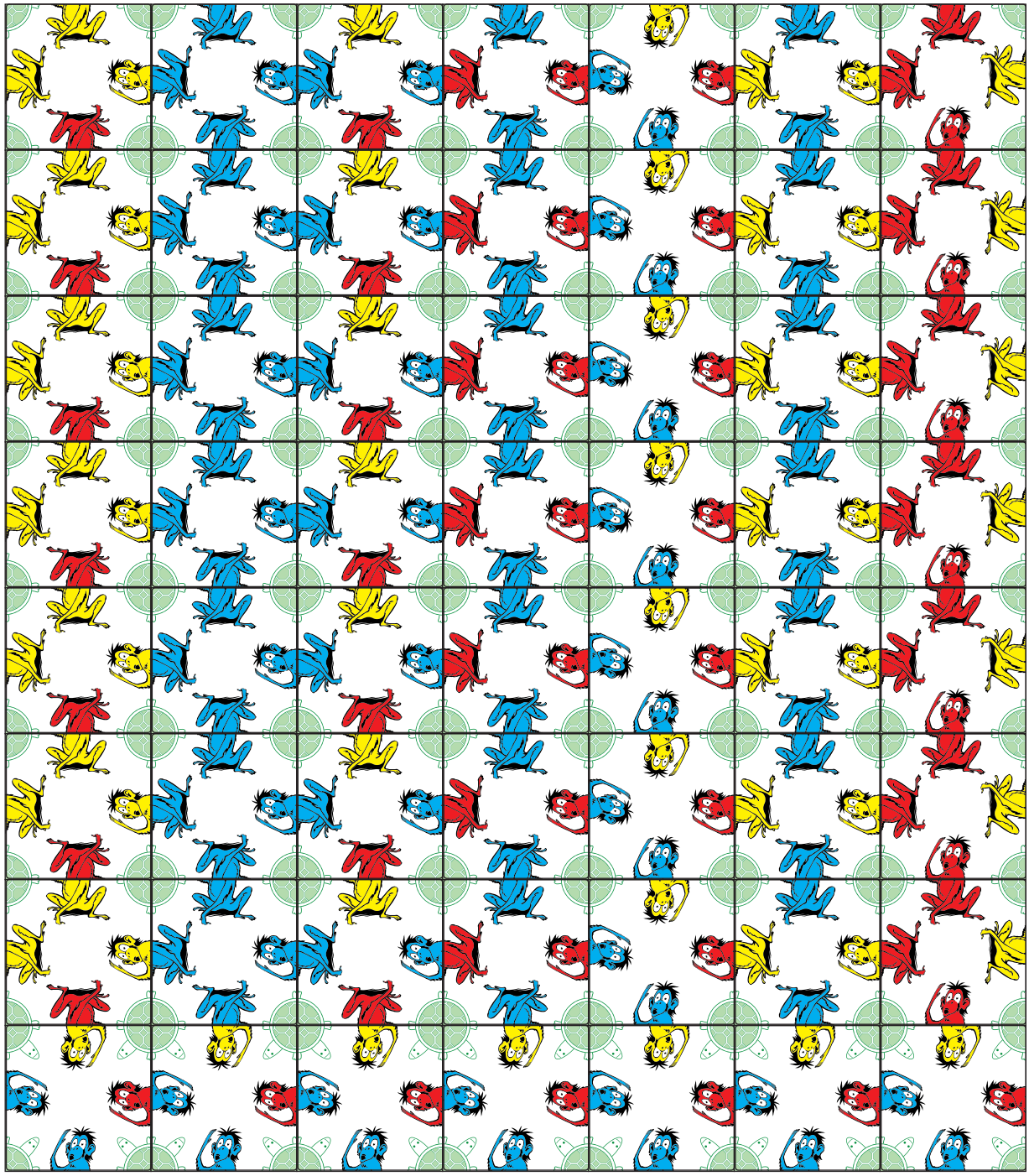


Abbildung 15.K5
Affenpuzzle mit Schildkröten.
Ggf. mehrfach kopieren!



16. Spielchen gefällig?

Der Satz „Gott würfelt nicht“ ist eines der bekanntesten Zitate von Albert Einstein. Der Physiker Leonard Mlodinow konterte mit „Wenn Gott würfelt“. Insgesamt sind viele Wissenschaftler momentan der Meinung, dass der Zufall in unserem Universum eine bedeutende Rolle spielt. Die Frage, ob es nun echten Zufall gibt oder ob uns nur Ereignisse überraschen, weil wir die Wirkprinzipien noch nicht vollständig verstanden haben, wird wohl noch eine Weile offenbleiben.

Auf jeden Fall müssen wir im Leben sehr häufig Entscheidungen treffen. Manchmal können wir uns dabei auf solide Grundlagen stützen, die wir sorgfältig gegeneinander abwägen können. Diese komfortable Situation ist auch meistens der Ausgangspunkt informatischer Betrachtungen: In den vorangegangenen Kapiteln haben wir uns immer mit Aufgaben beschäftigt, für deren optimale Lösung prinzipiell alle Fakten vorhanden waren. Besonders in den letzten Kapiteln ist uns klargeworden, dass wir manchmal nicht in der Lage sind, die Zeit dafür aufzubringen, um die wirklich beste Lösung zu berechnen, und dass manche Probleme sogar nicht berechenbar sind. Das lag aber nie an fehlenden Informationen.

Oft im Leben fehlt jedoch eine solide Grundlage für Entscheidungen: Fakten fehlen, weil diese schlicht nicht erhoben wurden oder weil sie überhaupt erst in der Zukunft erhoben werden können. Trotzdem müssen wir hier und jetzt eine Entscheidung treffen. Als Mensch haben wir die Gabe, das mehr oder weniger gut „aus dem Bauch heraus“ zu tun. Ein Beispiel im Alltag ist das Tanken zu einem günstigen Preis, wenn wir nicht wissen, ob das Benzin vielleicht am nächsten Tag noch weniger kostet – oder sich plötzlich stark verteuert. Oder wir kaufen eine BahnCard, ohne zu wissen, ob wir im nächsten Jahr tatsächlich so viel reisen werden, dass diese sich lohnt. Sehr viele Kaufentscheidungen beruhen auf einer unsicheren Datenlage.

Weitere Beispiele abseits des Konsums sind:

- „Nehme ich die Badehose mit oder lieber den dicken Pullover?“
- „Erhöhe ich oder passe ich beim Poker?“
- „Wann fahre ich los, um pünktlich anzukommen?“

Zunehmend sollen solche Entscheidungen auch automatisiert getroffen werden. So könnte in einem Betrieb zum Beispiel ein Computerprogramm bekannte Dienstreiseverläufe auswerten, um die Frage nach der BahnCard mit statistischen Methoden zu beantworten.

Da die Entscheidung quasi im laufenden Betrieb getroffen werden muss, während noch nicht alle Fakten vorliegen, spricht man hier von Online-Problemen. Bevor wir aber weiter auf die ernsten Anwendungen eingehen, wollen wir uns der Thematik in gewohnter Weise nähern: mit einem Spielchen.

Ökonomisches Schwimmen

Im Naturbadesee Ihrer Wahl gibt es mehrere Möglichkeiten, den Eintritt zu entrichten:

- Tageskarte für 1 EUR
- Saisonkarte für 30 EUR

Sie wissen, dass Sie als großer Schwimmfan an jedem Tag schwimmen gehen möchten, an dem das Wetter einigermaßen geeignet ist. Allerdings können Sie für die Saison absolut nicht voraussehen, wie viele schöne Tage es geben wird. Es gab bereits Vorjahre mit nur einem geeigneten Tag, aber auch solche, an denen Sie die 200 Tage der Saison komplett ausnutzen konnten.

An jedem schönen Tag stehen Sie nun am Kassenhäuschen vor der Entscheidung, entweder 30 EUR zu bezahlen und damit für den Rest der Saison den Eintritt zu entrichten oder es bei 1 EUR zu belassen, um dann am nächsten schönen Tag neu zu entscheiden. Die bereits gekauften Tageskarten werden natürlich auf die Saisonkarte nicht angerechnet. Bisher haben Sie das immer direkt „aus dem Bauch heraus“ entschieden.

Für die kommende Saison wollen Sie gewappnet sein und entwickeln im Voraus eine Strategie, nach der Sie sich dann strikt richten. Gehen Sie davon aus, dass Sie das Wetter überhaupt nicht voraussehen können!

Zunächst ist zu überlegen, wie eine solche Strategie überhaupt prinzipiell aussieht. Da Sie Faktoren wie das Wetter der nächsten Tage nicht kennen, bleibt eigentlich nur etwas übrig wie: „Ich kaufe t-mal eine Tageskarte, am ersten schönen Tag danach eine Saisonkarte.“

Wenn Sie erwägen, sofort eine Saisonkarte zu kaufen, ist $t = 0$, wenn Sie komplett nur Tageskarten kaufen möchten ist $t = 200$. Es liegt auf der Hand, dass eine Entscheidung wie $t = 190$ auf jeden Fall wirtschaftlich unsinnig ist, da dann maximal noch zehn gute Tage übrig sind, die man günstiger weiter mit Tageskarten bestreitet.

Als Nächstes haben Sie sich wahrscheinlich einige Qualitätskriterien überlegt, die für eine Entscheidung in die eine oder andere Richtung sprechen. So würde das Kriterium „Ich habe keine Lust, mich täglich an die Schlange am Kassenhäuschen anzustellen“ stark für die Entscheidung $t = 0$ sprechen.

Wiltrud Wissend

Eventuell haben Sie aber – der Überschrift entsprechend – auch rein ökonomische Kriterien angesetzt. Hierbei schwimmt man auch gedanklich, wenn keinerlei statistische Daten über das Wetter existieren, die eine bestimmte Zahl geeigneter Tage plausibel machen. Ganz automatisch sucht man nach einem Vergleichsmaßstab wie „Meine Lösung ist nicht schlecher als ...“. Das versuchen wir hier ebenfalls.

Ich stelle Ihnen daher Frau Wiltrud Wissend vor. Sie kann dank ihrer Kristallkugel perfekt in die Zukunft sehen und gibt daher keinen Cent zu viel für ihre Schwimmbadkarten aus. Besser kann ihre – auf Vermutungen beruhende – Strategie also gar nicht sein. Trotzdem ist es sinnvoll, sich mit Wiltrud zu messen in der Form: „Meine Strategie gibt höchstens dreimal so viel Geld aus wie Wiltrud.“

Vergleichen Sie Ihre Strategie mit der Wiltruds in allen möglichen Szenarien, also für ganz wenige und auch für ganz viele geeignete Tage. Überlegen Sie, ob Sie noch eine bessere Strategie finden können und wievielmal mehr Geld als Wiltrud sie schlimmstenfalls damit zahlen.

Ich möchte mit Ihnen drei Strategien durchsprechen:

- Von Anfang an eine Saisonkarte kaufen ($t = 0$)
- Nie eine Saisonkarte kaufen ($t = 200$)
- Am 30. geeigneten Tag eine Saisonkarte kaufen ($t = 29$)

Eine Saisonkarte wird immer lohnender, je mehr geeignete Tage es gibt. Wenn Sie gleich am ersten Tag eine Saisonkarte erwerben, wäre daher der schlechteste Fall, dass danach nur noch schlechtes Wetter folgt.

Wiltrud hätte das natürlich gewusst und sich lediglich ein einzelnes Tagesticket geleistet. Mit 30 EUR für die Saisonkarte hätten Sie daher im schlechtesten Fall das 30-fache ausgegeben.

Nicht viel besser sieht die Sache aus, wenn Sie sich nie zur Saisonkarte entschließen. Am schlechtesten schneidet diese Strategie offensichtlich ab, wenn nur schönes Wetter herrscht und sich daher die Saisonkarte, die Wiltrud selbstverständlich dann gleich am ersten Tag erwerben wird, maximal lohnt. Wiltrud bezahlt daher 30 EUR, Sie bezahlen 200 EUR, was knapp dem 7-fachen entspricht.

Betrachten wir den letzten Fall, müssen wir zwei mögliche Szenarien unterscheiden:

Es gibt maximal 29 geeignete Tage:

In diesem Fall kaufen wir – genau wie Wiltrud – nur Tageskarten, geben also exakt gleich viel Geld aus.

Es gibt mindestens 30 geeignete Tage:

In diesem Fall kauft Wiltrud sicher von Anfang an eine Saisonkarte, gibt also 30 EUR aus. Wir erwerben diese erst am 30. Tag und geben daher insgesamt 59 EUR aus, was nicht ganz dem doppelten Betrag entspricht.

Sie können versuchen, noch andere Strategien durchzuspielen, um festzustellen, dass „$t = 29$“ in Bezug auf die Optimierung des schlechtestmöglichen Falls gegenüber Wiltrud die günstigste Strategie ist.

Tatsächlich ist es üblich, die bei Online-Algorithmen verwendeten Strategien daran zu messen, wievielmal schlechter sie schlimmstenfalls gegenüber einer allwissenden Strategie sind. Den resultierenden Faktor nennt man Kompetitivität. Unsere Strategie „$t = 29$“ ist 2-kompetitiv, weil sie höchstens zwei Mal so viel Geld ausgibt wie unbedingt nötig. Abbildung 16.1 zeigt, wie sich die Kompetitivität k mit der Wahl der Strategie für den Kauf der Saisonkarte verändert. Das Minimum (das man ja erreichen möchte, um wenig Geld auszugeben) liegt bei $t = 29$ und $k \approx 1{,}97$.

Ski-rental
In der Literatur wird das hier beschriebene Problem meistens als „Ski-rental“ beschrieben, weil in der ersten Veröffentlichung der Problemstellung die als Beispiel gewählten Ski entweder immer wieder von Neuem geliehen oder endgültig gekauft werden konnten.

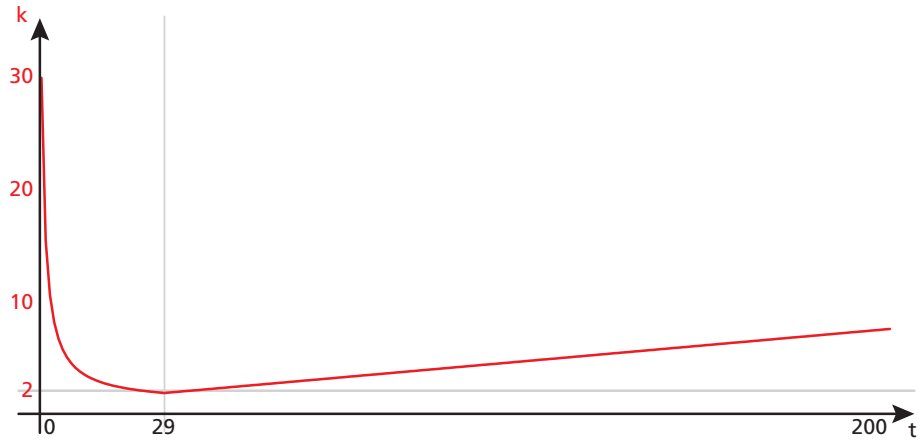


Abbildung 16.1
Kompetitivität der verschiedenen Strategien $t =$

Online-Rucksäcke

Kompetitivität wird als Wort einerseits oft negativ empfunden, in Gesellschafts- oder auch Computerspielen sorgt der Wettbewerb aber andererseits auch für den Spaß: Am Ende werden zum Beispiel erreichte Punkte oder eingenommenes Spielgeld verglichen. „Abenteuer Informatik“ ist selbstverständlich dem Spaß verpflichtet und daher spielen wir in den folgenden Abschnitten viel. Am meisten Spaß macht das mit einer Mitspielerin bzw. einem Mitspieler. Sie können natürlich die Rollen notfalls auch abwechselnd alleine spielen, um die Beispiele nachzuvollziehen, ich empfehle aber unbedingt das „echte“ Spiel.

Arne Arglos

Hier im Buch werden immer Wiltrud Wissend und Arne Arglos gegeneinander antreten. Die Rollen sind dabei fest verteilt:

- Arne möchte für eine gegebene Aufgabe einen möglichst guten Online-Algorithmus entwickeln. Er schreibt diesen daher so präzise wie möglich stichpunktartig auf. Wenn dann die eigentlichen Spielrunden laufen, hält sich Arne mit seinen Spielzügen möglichst exakt an seinen selbst definierten Algorithmus.
- Wiltrud ist mit den Schicksalsgöttinnen im Bunde und kann daher nicht nur in jeder Situation völlig neu über ihre eigenen Spielzüge entscheiden, sondern sie kann in der Regel auch die Zufallsereignisse des Spielszenarios frei beeinflussen. Mit anderen Worten: Wenn gewürfelt wird, nimmt Wiltrud den Würfel und setzt ihn so auf den Tisch, wie sie das für geeignet hält.

Beispiel
Arne ruft bei seiner Wette Faktor 3 auf.
In der ersten Runde erreicht Arne 10 Punkte, Wiltrud 25. Damit hat Wiltrud 2,5-mal mehr Punkte und ist sogar schwächer als von Arne vorausgesagt. Sie verliert die Wette.
In der nächsten Runde muss Arne dann zum Beispiel 2,4 ansagen, weil er unter dem bleiben muss, was er in der aktuellen Runde bereits geschafft hat.

Unfair? Ja, natürlich! Es ist klar, dass Wiltrud immer das bessere Ergebnis hat. Daher können Punkte oder Spielgeld auch nicht die alleinige Bedingung für Sieg oder Niederlage sein! Vielmehr wettet Arne gegen Wiltrud: Er ruft auf, wievielmal besser Wiltrud sein darf als er. Schafft er es, hat er dem Schicksal ein Schnippchen geschlagen. Werden mehrere Runden gespielt, muss Arne seine Wette nach einem Sieg immer verschärfen. Sein Aufruf muss unter dem Faktor liegen, den er in der aktuellen Runde erreicht hat.

Eine interessante Spielvariante ergibt sich auch, wenn Sie zu mehreren sind. Dann können sich mehrere „Arnes“ gegenseitig unterbieten und derjenige mit dem kleinsten Gebot muss dann gegen Wiltrud antreten und zeigen, dass er den Faktor, den er sich zugetraut hat, auch im Spiel umsetzen kann. Alternativ können auch alle mit ihrem gebotenen Faktor gegen Wiltrud spielen und derjenige gewinnt, der den niedrigsten auch erfüllt.

Noch Fragen? Ich gehe davon aus! Wir wollen trotzdem endlich mit dem Spielen anfangen, es wird sich dann sicher vieles klären. Sie erinnern sich an die Schatzsuche aus Kapitel 3? Dort sollten optimal Kisten einer bestimmten Größe gepackt werden. Die Schatzjäger hatten dabei allerdings den vollen Überblick über die zur Verfügung stehenden Objekte. Auf diese Bequemlichkeit müssen Sie nun verzichten.

Stellen Sie sich vor, Arne wühlt sich durch einen Schatzhaufen und muss jedes Mal, wenn er einen Goldbrocken findet, sofort entscheiden, ob er diesen in seinen Rucksack steckt und damit sichert oder ob er ihn wieder weglegt. Auf weggelegte Stücke stürzen sich sofort die anderen Schatzsucher und verschwinden damit auf Nimmerwiedersehen. Aufgrund seiner nicht sehr standfesten Position kann Arne auch nicht seinen Rucksack ausleeren oder gar einen Goldbrocken austauschen: Einmal im Rucksack ist das Gold fest „ausgewählt".

Arnes Rucksack hält allerdings nur ein bestimmtes Gewicht aus, er darf also nicht überladen werden. Glücklicherweise kennt Arne dieses Maximalgewicht und kann auch von jedem Stück sehr genau das Gewicht abschätzen. Arnes Ziel ist selbstverständlich, möglichst viel Gold einzusammeln.

Wiltrud spielt keine konkurrierende Schatzsucherin, sondern das Schicksal. Sie legt vor jeder Spielrunde fest, wie viel Gewicht Arnes Rucksack tragen kann, und sagt das auch an. Danach bietet sie ihm immer wieder Goldbrocken an, die er entweder sofort nehmen und in seinen Rucksack stecken darf, falls damit das Maximalgewicht noch nicht überschritten wird, oder eben nicht. Wiltrud entscheidet danach jeweils für ihren mit Arnes identischen Rucksack, ob sie einen identischen Goldbrocken einpackt.

Wiltrud darf jederzeit entscheiden, dass die Schatzkammer erschöpft ist, Arne also nichts Weiteres findet. Damit ist die Runde beendet und es wird ausgewertet, wie viel jeder Rucksack wert ist. Wiltrud möchte natürlich als Schicksal gerne mit ihrem prall gefüllten Rucksack prahlen, der für Arne nicht erreichbar ist. Durch andere Entscheidungen hätte er aber natürlich seinen Rucksack auch so füllen können, denn Wiltrud hatte ja die gleichen Goldbrocken zur Verfügung.

Sie können zum Spielen die Vorlage in 16.K1 benutzen. Oben wird das Maximalgewicht des Rucksacks eingetragen. Bei den Goldbrocken, die Wiltrud anbietet, kann sie aus dem Vollen schöpfen und notiert beliebige Gewichtseinheiten in der Mitte auf dem Zettel. Arne macht dann links davon ein Kreuz, wenn er den Brocken haben möchte, Wiltrud danach rechts davon, ob sie ihn nimmt. Der Bogen nach Abbildung 16.2 zeigt ein Beispiel. Wiltrud erreicht 100 Gramm, Arne nur 92 Gramm, der Faktor liegt daher bei etwa 1,09. Hat Arne für seine Wette 1,1 angesagt, gewinnt er diese Runde.

Spielen Sie! Versuchen Sie mit verteilten Rollen einerseits eine gute Strategie zu finden, angebotene Gegenstände zu nehmen oder abzulehnen, um dem Schicksal ein Schnippchen zu schlagen. Schlüpfen Sie dann aber auch in die Rolle des Schicksals, um unseren armen Schatzsucher zur Verzweiflung zu treiben und seine Ausbeute kontinuierlich schlechter werden zu lassen ... Wer kann in diesem Spiel gewinnen?

Abbildung 16.2
Spielbogen

Maximalgewicht  100g

Arne	Gewicht Goldbrocken	Wiltrud
X	19g	X
X	5g	X
-	55g	X
X	68g	-
--	21g	X
92g	Ende!	100g

Wer hatte bei Ihnen die Nase vorne? Arne oder Wiltrud? Falls es nicht Wiltrud war, sollten Sie noch einmal nachdenken, denn ich verrate jetzt, dass dieses Spiel äußerst unfair zu Wiltruds Gunsten ist: Es existiert eine bombensichere Gewinnstrategie. Egal, welchen Faktor Arne ansagt – Wiltrud kann ihn schlagen.

Wir nehmen hier an, Arne habe Faktor f angesagt.

Wiltrud legt die Rucksackgröße in dieser Runde dann fest bei einem Wert zum Beispiel in Gramm, der f um ein Gramm übersteigt, also $f + 1$. Danach bietet sie Arne einen winzigen Goldbrocken von einem Gramm Gewicht an. Arne kann diesen nun nehmen oder ablehnen.

- Falls Arne den Brocken akzeptiert, nimmt Wiltrud ihn selbst nicht. Danach bietet sie nur noch Goldbrocken mit einem Gewicht von genau $f + 1$ Gramm an, von denen sie sich selbstverständlich selbst einen sichert. Arne würde damit seinen Rucksack wegen des winzigen Goldstückchens vom Anfang überfüllen und kann sie daher nicht nehmen. Wiltrud gewinnt mit $(f + 1) : 1$, also Faktor $f + 1$, was das Gebot von f übertrifft. Abbildung 16.3 zeigt einen Spielbogen, der diese Variante repräsentiert.
- Falls Arne den Brocken ablehnt, nimmt Wiltrud ihn selbst und beendet danach die Runde. Arne ist damit komplett leer ausgegangen und Wiltrud gewinnt $1 : 0$. Der Faktor ist also unendlich, was f ebenfalls übertrifft. Selbstverständlich können Sie einwenden, dass Wiltrud mit einem Gramm mehr nun wirklich nicht so viel besser ist. Nach unserem Wettschema gewinnt sie damit trotzdem immer, und darauf kommt es an! Den Spielbogen für diese Variante sehen Sie in Abbildung 16.4.

Bieten Sie ruhig versuchsweise Freunden die Rolle von Wiltrud an und testen, ob sie auf die hier gezeigte Strategie kommen. Es ist eine interessante Erfahrung, weil auch die Balance in vielen Gesellschaftsspielen durch entsprechende Handlungen gestört werden kann.

Abbildung 16.3
Spielbogen, falls Arne den 1 Gramm schweren Goldbrocken einpackt

Maximalgewicht  100g

Arne	Gewicht Goldbrocken	Wiltrud
X	1g	-
--	100g	X
1g	Ende!	100g

Abbildung 16.4
Spielbogen, falls Arne den 1 Gramm schweren Goldbrocken verschmäht

Maximalgewicht  100g

Arne	Gewicht Goldbrocken	Wiltrud
-	1g	X
0g	Ende!	1g

Wie gut ist gut?

Warum habe ich Ihnen ein solch offensichtlich unsinniges Spiel überhaupt präsentiert? Eine Möglichkeit zur Bewertung der Qualität von Strategien für Online-Probleme ist die Kompetetivität. Eine Strategie ist k-kompetetiv, wenn sie höchstens k-mal schlechter als eine „allwissende" Strategie ist. Mit anderen Worten: In unserem spielerischen Szenario darf Wiltrud maximal k-mal so viele Punkte, Beträge bzw. andere Bewertungseinheiten kassieren wie Arne.

Im Fall der Online-Rucksäcke konnte kein solches k gefunden werden – im Gegenteil: Mit Hilfe von Wiltruds Strategie, mit der sie jede Wette gewinnen kann, die Arne anbietet, haben wir nachgewiesen, dass es auch kein k geben kann. Das ist anders als beim Schwimmbad-Problem, denn dort konnten wir eine Strategie für $k = 2$ aufstellen.

Im Spiel für dieses Szenario ist Arne der Schwimmbadbesucher, der immer wieder die Entscheidung zwischen Tages- und Saisonkarte trifft. Wiltrud spielt Schicksal, das immer wieder von Neuem entscheidet, ob es noch weitere schöne Tage gibt oder nicht. Weiter oben haben wir begründet, warum Arne mit der Strategie $t = 29$ die Wette 2,0 immer gewinnt.

Mit Hilfe unserer Spiele sind wir also einem momentan gebräuchlichen Kriterium zur Bewertung von Strategien für die Lösung von Online-Problemen auf die Spur gekommen. Versuchen wir das doch gleich noch mit einem neuen Spiel.

Schlüsseldienst

Arne betreibt hier einen Schlüsseldienst. Während er selbst die Zentrale besetzt hält und Notrufe entgegennimmt, stehen zwei Service-Mobile mit versierten Türöffnungsexperten bereit, die er zu den betreffenden Adressen schicken kann. Benzin ist teuer, und so möchte Arndt die Kosten möglichst gering halten, indem er versucht, die gesamte Fahrtstrecke nicht zu groß werden zu lassen. Es kommt darauf an, jeweils das „richtige" Service-Mobil zum Notruf zu schicken.

Die Situation simulieren wir mit einem Schach- oder Damebrett. Vorlage 16.K3 hilft, wenn Sie gerade keines zur Hand haben. Die Service-Mobile darf Arne dabei auf beliebige Startpositionen auf dem Brett stellen. Wiltrud benutzt ein eigenes Brett, das Arne nicht sehen kann (während natürlich Wiltrud umgekehrt jeden Schritt von Arne genau beobachtet). Sie muss allerdings ihre Service-Mobile auf die gleichen Startpositionen stellen. Nutzen Sie dafür die Spielchips aus der Vorlage oder aber irgendwelche Spielsteine, zum Beispiel die Schach-Läufer und -Türme.

In jeder Runde sagt nun Wiltrud an, für welches Feld ein Notruf eingeht. Arne darf nun alle seine Service-Mobile beliebig nach oben, unten, links und rechts ziehen. Jeder Zug kostet dabei pro Feld einen Benzinpunkt. Die Züge werden notiert, die Kosten dafür ebenfalls.

Auch Wiltrud zieht verdeckt ihre Service-Mobile und notiert Züge und Kosten. Danach sagt sie die Position des nächsten Notrufs an und läutet damit eine neue Runde ein. Wiltrud bestimmt auch willkürlich, wann das Spiel endet und ihre Benzinkosten mit denen von Arne verglichen werden. Selbstverständlich sollte Wiltrud dabei besser abschneiden, weil sie die Position der Notrufe frei bestimmen kann – zum Beispiel dort, wo bereits eines ihrer Service-Mobile in der Nähe steht.

Spielen Sie das Beispiel nach Abbildung 16.5 nach, um die Regeln einzuüben. Es ist auf der Vorlage nach Abbildung 16.K2 entstanden, die Sie auch für eigene Experimente nutzen können.

Der Übersicht halber werden in den Beispielen hier Wiltruds Züge immer direkt auf dem gleichen Bogen geführt wie Arnes. Da Arne diese aber erst nach seinem allerletzten Zug sehen darf, kann Wiltrud entweder zunächst ein anderes Blatt zum Notieren nutzen und die Züge dann übertragen oder Sie kopieren den Bogen einfach zweimal und jeder füllt die entsprechenden Spalten aus.

Im Spiel werden zunächst nur zwei Service-Mobile Ⓢ und Ⓣ benutzt. Wiltrud setzt den ersten Notruf auf F2 und Arne nutzt selbstverständlich das näher stehende Servic-Mobil Ⓣ, um diesen zu bedienen. Wiltrud stürzt sich allerdings in Unkosten und zieht Ⓢ, das zwei Felder weiter weg steht. Die Bilanz sieht also nach dem ersten Zug noch recht gut für Arne aus.

Als nächsten Notruf sucht Wiltrud nun allerdings C1 heraus, die ursprüngliche Position des Service-Mobils Ⓣ. Arne muss es wieder dorthin bewegen, während Wiltrud ihr Fahrzeug bereits dort stehen hat, also für sich Kosten von null eintragen kann. Dieses Spiel treibt Wiltrud nun weiter und wählt als nächsten Notruf wieder F2, wo ihr Fahrzeug Ⓢ stehen geblieben ist, Arne seines aber wegbewegt hat.

Beim nächsten C1 riecht Arne den Braten und ändert seine Strategie: Statt nur Ⓣ zu bewegen, zieht er zusätzlich sein Service-Mobil Ⓢ zumindest ein Stückchen in die Richtung des von ihm vermuteten nächsten Notrufs. Das widerspricht nicht den Regeln – man kann auch mehrere Fahrzeuge ziehen. Der nächste Notruf wird von Wiltrud auch prompt auf F2 gesetzt und Arne kann nun mit geringeren Kosten sein

Abbildung 16.5
Proberunde von Schlüsseldienst

Startfelder

C1 H6

Notruf	Arne		Wiltrud	
	Zug	*Kosten*	*Zug*	*Kosten*
F2	T-F2	4	S-F2	6
C1	T-C1	4	T-C1	0
F2	T-F2	4	S-F2	0
C1	T-C1 S-G5	6	T-C1	0
F2	S-F2	4	S-F2	0
Ende		Summe: 22		Summe: 6

Fahrzeug Ⓢ schicken (freilich hat er insgesamt mit der Vorleistung aus der letzten Runde auch sechs Benzinpunkte verbraucht).

Daraufhin beschließt Wiltrud, dass ihr Vorsprung ausreichend sei, und beendet das Spiel. Sie könnte natürlich einfach weitermachen wie bereits gezeigt, um die Punktedifferenz weiter zu vergrößern.

Spielen Sie nun Schlüsseldienst! Versuchen Sie zunächst, die optimalen Strategien für Arne und Wiltrud mit zwei Service-Mobilen herauszufinden, danach können Sie die Anzahl erhöhen.

Es ist gar nicht so einfach! Arne muss ständig aufpassen, Wiltruds Vorsprung nicht zu groß werden zu lassen, während Wiltrud auch immer wieder neu nachdenken muss, um die Notrufe so zu setzen, dass sie auch über mehrere Züge hinweg einen Vorteil behält.

Es ist trotzdem klar, dass Wiltrud mit weniger Benzinpunkten aus dem Spiel geht. Die Frage ist allerdings, ob Arne eine Wette gewinnen kann. Im Beispiel nach Abbildung 16.5 hätte er eine Wette mit Ansage „vier“ auf jeden Fall gewonnen, denn er hat nur knapp viermal so viele Benzinpunkte ausgegeben. Wie war das in Ihren Spielrunden? Denken Sie auch allgemeiner darüber nach!

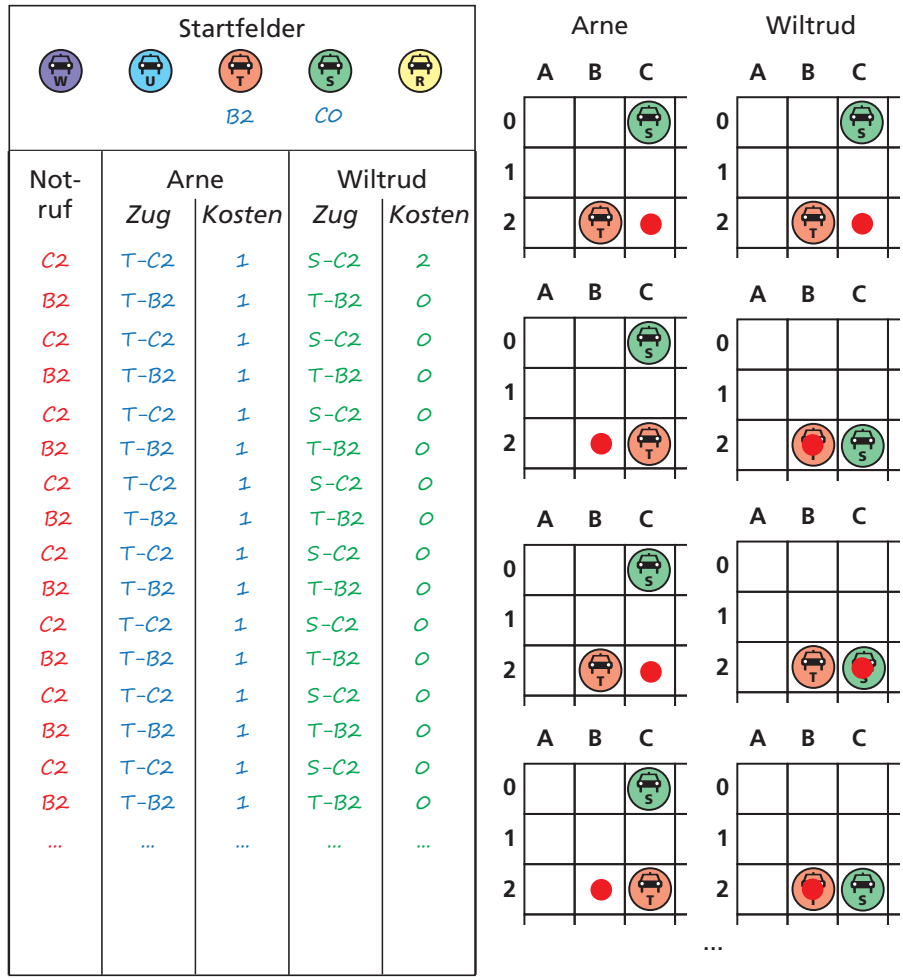
Kann Wiltrud – wie im Beispiel der Online-Rucksäcke – auch jede Wette gegen Arne gewinnen?

Arne hat immer das Problem des Informationsdefizits. Da er die Position der Service-Mobile von Wiltrud nicht kennt, kann er auch nicht vorausschauend einen eigenen Zug machen, der den Schaden minimiert.

Andererseits ist auch Wiltrud in ihren Aktionen durch das Spielbrett limitiert. Die maximale Zugzahl zu einem beliebigen Ziel ist 18 (zum Beispiel von A0 nach J9). Sie kann es zwar schaffen, einige Runden mit Benzinkosten von 0 zu bestehen, indem sie den Notruf genau auf der Position eines ihrer Service-Mobile ansagt, irgendwann wird Arne das aber durchschauen und seine beiden Fahrzeuge ebenfalls auf diese Positionen bringen. Dann muss Wiltrud „echte“ Züge machen und kann daher den Multiplikationsfaktor zu Arne nicht beliebig in die Höhe treiben.

Wenn wir die ursprüngliche Idee des Online-Problems zugrunde legen, hat Arne allerdings auch noch zu viele Freiheiten: Eigentlich wollen wir ja ausprobieren, wie gut Arnes Rolle von einem Computer übernommen werden könnte. Computer gehen aber nach einer festen Strategie, einem Algorithmus vor, und das fordern wir jetzt in Form verschärfter Regeln auch von Arne: Er denkt sich vorher ein Zugverfahren aus, das er dann strikt befolgt.

Auf diese Weise können Sie nun sowohl Arne als auch Wiltrud ganz alleine spielen, indem Sie zunächst in Arnes Rolle ein Verfahren festlegen. Während des Spiels hat Arne dann ja ohnehin keine Freiheiten mehr und Sie können sich auf die Rolle von Wiltrud konzentrieren. Mehr Spaß macht es freilich weiterhin zu zweit oder sogar zu mehreren.



Spielen Sie nun nach den neuen Regeln. Welche möglichen Strategien für Arne sind gut, weil Wiltrud sie nicht konterkarieren kann? Gibt es Strategien, die kompetitiv sind, für die Arne also die Wette „Ich bin höchstens um Faktor *k* schlechter als Wiltrud“ gewinnt?

▶ ▶ ▶ ◀ ◀ ◀

Vielleicht sind Sie als Erstes auf die sogenannte „Greedy-Strategie“ gekommen. Greedy heißt gierig und bedeutet in diesem Fall, dass immer der für die momentane Situation günstigste Zug gewählt wird, ohne Rücksicht darauf, ob sich das eventuell für die Zukunft ungünstig auswirkt.

Konkret ist die Vorschrift für Arne, dass immer das Service-Mobil zum aktuellen Notruf gezogen wird, mit dem das mit den kleinsten Benzinkosten möglich ist.

Diese Vorschrift ist nicht kompetitiv. Wiltrud kann genau das tun, was bereits im Beispiel zu sehen war. Abbildung 16.6 zeigt, wie sie nach der Investition einer bestimmten Zahl eigener Benzinpunkte jeweils selbst immer Nullrunden provozieren kann, während Arne kontinuierlich Benzin ausgeben muss. Auf diese Weise kann Wiltrud

Abbildung 16.6
Wiltrud kann die Greedy-Strategie sehr einfach konterkarieren.

Notruf	Arne Zug	Arne Kosten	Wiltrud Zug	Wiltrud Kosten
C2	T-C2	1	S-C2	2
B2	T-B2	1	T-B2	0
C2	T-C2	1	S-C2	0
B2	T-B2	1	T-B2	0
C2	T-C2	1	S-C2	0
B2	T-B2	1	T-B2	0
C2	T-C2	1	S-C2	0
B2	T-B2	1	T-B2	0
C2	T-C2	1	S-C2	0
B2	T-B2	1	T-B2	0
C2	T-C2	1	S-C2	0
B2	T-B2	1	T-B2	0
C2	T-C2	1	S-C2	0
B2	T-B2	1	T-B2	0
C2	T-C2	1	S-C2	0
B2	T-B2	1	T-B2	0
...	...	...	...	...

das Spiel so lange treiben, bis Arne die Wette verloren hat – unabhängig von seiner Ansage.

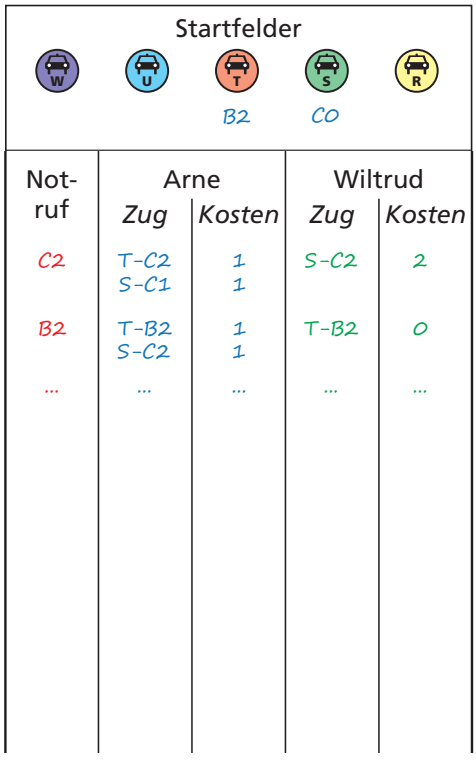
Es gibt noch etliche Verfeinerungen der Greedy-Strategie. So könnte man etwa festlegen, dass jedes Service-Mobil einen Bereich des Spielplanes zugewiesen bekommt, den es nicht verlässt. Spielen Sie ruhig ein wenig damit herum – ich gehe davon aus, dass immer ein sehr einfacher „Trick" wie der in Abbildung 16.6 dargestellte die Kosten für Arne in unbegrenzte Höhe treibt, während Wiltrud mit konstantem Aufwand aus dem Spiel geht.

Im Beispiel nach Abbildung 16.5 handelt Arne auch intelligenter: Er kann irgendwann ahnen, dass Wiltrud die Notrufe immer zwischen zwei Feldern hin- und herspringen lässt, die sie mit ihren eigenen Service-Mobilen besetzt hält. Daraufhin zieht er ebenfalls auf diese beiden Felder, was Wiltrud zwingt, zumindest die Positionen zu wechseln, um ihr Spiel von Neuem zu beginnen.

Versuchen wir es daher mit folgender erweiterter Greedy-Strategie:

- Merke dir die Position *P* des dem Notruf nächsten Service-Mobils.
- Ziehe dieses Service-Mobil auf den Notruf mit *b* Benzinpunkten.
- Ziehe das andere Service-Mobil *b* Schritte in Richtung *P*. Wenn die Wahl zwischen einem Schritt horizontal und vertikal besteht, dann ziehe in die Richtung, die „weiter" vom Ziel entfernt ist. Gibt es einen Gleichstand zwischen horizontal und vertikal („diagonale Position"), ziehe in die Richtung, die im letzten Schritt nicht gezogen wurde.

Abbildung 16.7 zeigt die Wirkung der neuen Strategie in unserem kleinen Beispiel. Die Strategie kostet pro Zug 2*b* Benzinpunkte, dafür nähert sich aber das zweite Service-Mobil immer weiter dem potentiellen Standort von Wiltruds Service-Mobil.



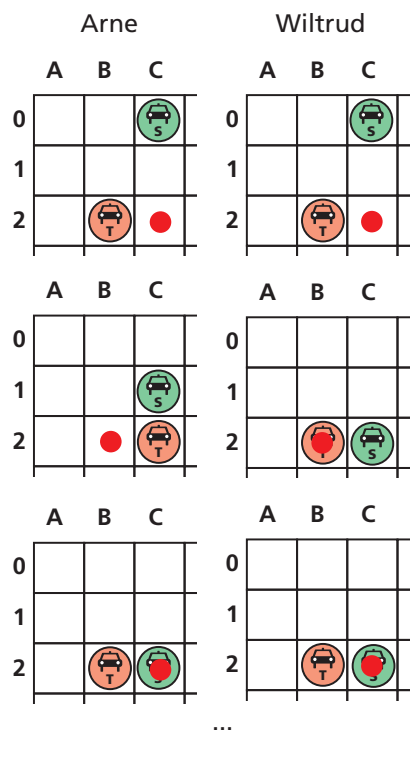


Abbildung 16.7
Wenn Arne die Greedy-Strategie so verändert, dass er zusätzlich immer das andere Service-Mobil in die Richtung zieht, in der vorher das andere Service-Mobil stand, entkommt er der „Falle".

Bereits nach zwei Runden und lediglich doppelt so viel ausgegebenen Benzinpunkten hat Arne seine Service-Mobile wieder auf den gleichen Positionen wie Wiltrud.

Betrachten wir die Kompetitivität dieser Strategie: Wenn Wiltrud versucht, Arne in die „Greedy-Falle“ zu locken, benötigt er – wie bereits weiter oben eruiert – maximal 18 Schritte, um das normalerweise dabei passive Service-Mobil quer über das Spielfeld nachzuziehen. Da er gleichzeitig auch die Notrufe mit seinem aktiven Service-Mobil bedienen muss, verbraucht er dafür nochmal die gleiche Menge an Benzinpunkten, insgesamt also 36. Danach ist er definitiv wieder auf den gleichen Positionen wie Wiltruds Service-Mobile und beide würden keine Benzinpunkte mehr in Anspruch nehmen, wenn Wiltrud den Notruf nicht so setzt, dass sie auch selbst Benzinpunkte verbrauchen muss.

Arne kann daher eine entsprechende Wette eingehen – bei einem Faktor von 36 würde er immer gewinnen, effektiv ist er sogar deutlich besser, wie wir auch am Beispiel in Abbildung 16.6 sehen. Für Arnes letzte Strategie steht der sogenannte Double-Coverage Algorithmus Pate, der allerdings nicht auf ein in Quadrate eingeteiltes Spielfeld limitiert ist und daher auch leicht anders funktioniert. Dieser hat einen Faktor der Kompetitivität von zwei.

Einfach würfeln

Arne muss seine Entscheidungen treffen, ohne genau zu wissen, was Wiltrud vorhat. Genau genommen muss er seine Strategie ja sogar bereits festlegen, bevor er Wiltrud überhaupt in die Augen schauen kann. Danach ist er aufgrund der Strategie festgelegt. Wie sahen denn Ihre Strategien für Arne aus? Es waren sicherlich auch einige recht komplizierte dabei.

Arne könnte sich das Leben nun einfacher machen. Was würden Sie tun, wenn Sie als absoluter Anfänger gegen einen Poker-Profi spielen müssten? Sie wissen, dass er Ihnen jeden Bluff an der Nasenspitze ansieht und entsprechend handelt. Eine kluge Vorgehensweise ist daher, die eigenen Karten gar nicht anzuschauen und das komplett Spiel vom Zufall leiten zu lassen, alle eigenen Entscheidungen also auszuwürfeln. Der Profi-Gegner ist natürlich immer noch im Vorteil, weil er seine Karten ansieht und nicht „blind“ spielt. Sein Gewinn wird allerdings nicht so hoch sein wie üblich.

Genau diese Strategie könnte Arne auch im Spiel „Schlüsseldienst“ verfolgen: Er zieht immer mit genau einem seiner Service-Mobile auf den Notruf. Welches er nutzt, bestimmt er mit einer Münze oder einem Würfel.

Probieren Sie es aus! Spielen Sie gegen Wiltrud mit einer beliebigen, ausgeklügelten Strategie und gleichzeitig auf einem weiteren Spielfeld, indem Sie Ihre Züge zufällig bestimmen. Welche Strategie ist auf Dauer günstiger?

Wahrscheinlich haben Sie festgestellt, dass Würfeln zumindest auf Dauer nicht schlechter abschneidet als ein beliebig klug ausgedachtes Verfahren. Das funktioniert nicht bei allen Aufgabenstellungen und auch nicht bei allen Maßstäben für Qualität, wird aber in der Informatik inzwischen recht häufig eingesetzt, weil es oft erstaunlich gute Ergebnisse hervorbringt.

Randomisierter Algorithmus

Algorithmus, der Entscheidungen nicht ausschließlich aufgrund vorgegebener oder gemessener Eingabewerte trifft, sondern sie auch zufällig bestimmt. Oft sind randomisierte Algorithmen im Vergleich zu solchen ohne virtuellen Würfel deutlich einfacher aufgebaut und daher sowohl schneller implementierbar als auch schneller in der Laufzeit.

Was steckt dahinter?

Die vorgestellten Spielchen sind ja ganz nett, aber was haben sie mit Informatik zu tun? Sicherlich könnte man sie auch als Computerspiel umsetzen und hätte mit Wiltrud eine adäquate Gegnerin. Das würde aber ein menschlicher Spieler wiederum als ziemlich ungerecht ansehen und wünscht sich einen Computer, der „ehrlich" spielt, also mit den gleichen Voraussetzungen was die Information über spielwichtige Parameter angeht. In sogenannten Spielen mit perfekter Information, also zum Beispiel Schach, hätte Wiltrud auch keinen Vorteil gegenüber Arne, denn das gesamte Spielfeld ist jederzeit komplett sichtbar und der Zufall – den Wiltrud ja beeinflussen kann – spielt beim Schach keine Rolle.

Einige „ernsthafte" Aufgabenstellungen, die mit Online-Algorithmen gelöst werden können, habe ich bereits in der Einleitung aufgeführt. Hierzu gehören etwa der automatisierte Kauf einer Firmen-BahnCard, die vorausschauende Herstellung von Produkten, bevor sie überhaupt bestellt werden, oder automatisierte Empfehlungen für den Kauf bzw. Verkauf von Aktien in Fonds.

Vielleicht verwundert es, wenn ich nun berichte, dass Online-Algorithmen auch im ganz normalen Betrieb von Computern im Einsatz sind – zum Beispiel in Ihrem heimischen PC oder Arbeitsplatzrechner. In einem anderen Kapitel hatte ich erwähnt, dass Computer sehr verlässliche Maschinen sind. Eigentlich sollte doch dann alles klar sein, Entscheidungen also aufgrund von Fakten und nicht Vermutungen getroffen werden, oder?

Die Aussage über Computer stimmt. Allerdings wird der Computer von Menschen bedient, und deren Verhalten ist von vielen Faktoren beeinflusst, die dem Computer nicht bekannt sind. So verändert sich etwa die Bedienung, wenn am Arbeitsplatz das Telefon klingelt. Insbesondere betrifft diese Unfähigkeit, genaue Voraussagen zu machen, aber Ressourcen wie Speicherplatz oder Rechenleistung. Das möchte ich am Beispiel des sogenannten „paging" erläutern.

Moderne Computer verwalten in der Regel einen größeren Hauptspeicher, als sie tatsächlich besitzen, also etwa 32 GByte, wenn effektiv nur Speicherbausteine für insgesamt 16GB auf der Platine verbaut sind. Auf diese Weise bekommen Programme, die sehr speicherintensiv sind, den benötigten Speicher zugewiesen. Dazu gehören zum Beispiel Bildverarbeitungsprogramme, die zum Beispiel etliche Bilder moderner Digitalkameras miteinander verknüpfen und diese dafür im Speicher halten.

Um dieses „Versprechen" des hohen Speicherplatzes zu halten, müssen die Betriebssysteme tricksen und lagern einen Teil der Daten, die sich eigentlich im Hauptspeicher befinden sollten, auf die Festplatte aus. Das nennt man dann „swapping" oder „paging". Sobald ein Programm auf auch nur ein Bit der ausgelagerten Daten zugreift,

müssen diese zunächst von der Festplatte in den Hauptspeicher geladen werden, was Zeit kostet. Damit das einigermaßen effizient vonstattengeht, wird immer gleich ein ganzer Speicherblock bzw. eine Speicherseite in den Hauptspeicher geladen, daher der Begriff „paging".

Vielleicht ist Ihnen der Haken beim „Laden einer Speicherseite" aufgefallen? Dafür müsste ja Platz im Hauptspeicher sein. Wenn es den aber gäbe, hätten wir ursprünglich auch keine Daten auf die Festplatte auslagern müssen. Bevor eine Seite von der Festplatte in den Hauptspeicher geladen werden kann, muss also eine andere Seite vom Hauptspeicher auf die Festplatte ausgelagert werden, um Platz zu schaffen.

Welche nimmt man dafür?

Eine Idee wäre, die speicherintensiven Programme genau zu analysieren, um festzustellen, wann sie auf welchen Speicherbereich zugreifen werden, und den Speicher entsprechend zu verwalten. Im letzten Kapitel mussten wir allerdings feststellen, wie schwer es allein schon ist, ein Programm dahingehend zu testen, ob es irgendwann anhält. Ein Computer kann diese geforderte Analyse daher nicht leisten!

Selbstverständlich könnten die Programmierer speicherintensiver Programme die Verwaltung selbst übernehmen, da sie ja ihre Programme kennen. Das wird auch tatsächlich an der einen oder anderen Stelle versucht. Allerdings kennen die Programmierer die Situation im Computer des Anwenders nicht, also etwa die Größe des Hauptspeichers und welche anderen Programme noch gleichzeitig laufen. Daher ist auch das schwierig.

In den meisten Fällen muss daher das Betriebssystem abschätzen, welche Speicherseite aus dem Hauptspeicher einer on der Festplatte benötigten weichen muss. Bei einer Fehlentscheidung muss diese eventuell im nächsten Moment wieder von dort geladen werden, was sehr viel Zeit kostet.

Erinnert Sie das an etwas? Auch beim Schlüsseldienst musste sich Arne dafür entscheiden, ein bestimmtes Service-Mobil zu bewegen, auch wenn vielleicht im nächsten Zug genau dort der folgende Notruf gemeldet wird.

In der Informatik haben „Worst-case"-Analysen eine große Tradition. Man nimmt also den schlimmsten Fall an und schätzt ab, wie gut dann ein System immer noch reagiert, um verschiedene Strategien miteinander zu vergleichen. Für eine Online-Strategie gibt es nichts Schlimmeres, als wenn jemand diese genau kennt, analysiert und dann das Schicksal genau so steuert, dass die Strategie maximale „Hätte ich in dieser Situation nur anders entschieden"-Gefühle hervorbringt. Im Spiel übernimmt Wiltrud diese Rolle. Unsere Erfahrungen mit den Szenarien helfen uns also dabei, Online-Algorithmen zu entwickeln, die im schlimmsten Fall im Vergleich zu einem „wissenden" Algorithmus nicht zu schlecht abschneiden.

Wie schlecht ist schlecht?

Eventuell waren Sie ja bereits bei den Online-Rucksäcken skeptisch, ob Wiltruds große Entscheidungsfreiheiten für das Ziel gerechtfertigt sind, eine möglichst gute Spielstrategie zu finden. Im folgenden Abschnitt wollen wir diese Skepsis an einem weiteren, allseits bekannten Spiel vertiefen.

Blackjack oder auch „Einundzwanzig“ oder „Siebzehn und vier“ spielt man mit französischen Spielkarten, also den Farben Kreuz ♣, Pik ♠, Herz ♥ und Karo ♦ sowie den Werten As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Bube, Dame, König.

Es werden etliche vollständige Spiele ineinandergemischt, so dass man beim Ziehen tatsächlich nahezu jede Karte mit derselben Wahrscheinlichkeit aufdeckt. Wir gehen hier daher von absolut gleichen Wahrscheinlichkeiten aus. Die Farben spielen keine Rolle, daher hat jeder Wert eine Wahrscheinlichkeit von $\frac{1}{13}$.

Jede Karte „zählt“ so viel, wie ihr Wert ist. Die sogenannten Bilder, also Bube, Dame und König, zählen jeweils 10 Punkte. Das As wird normalerweise nach Wunsch als 1 oder 11 gewertet. Um es uns mit den Berechnungen etwas leichter zu machen, werden wir immer 11 Punkte annehmen.

Beim Blackjack tritt immer der „Spieler“ gegen die Bank bzw. den „Croupier“ an. Er darf zunächst nacheinander beliebig viele Karten aufdecken. Sein Ziel ist, mit der aufgedeckten Gesamtpunktzahl der 21 möglichst nahe zu kommen bzw. sie genau zu erreichen. Auf keinen Fall darf er mehr Punkte erzielen, denn nach diesem sogenannten „bust“ hat der Croupier sofort gewonnen.

Danach ist der Croupier an der Reihe und deckt beliebig viele Karten auf. Hat der Spieler am Ende mehr Punkte als der Croupier (aber kein „bust“) oder hat der Croupier ein „bust“, der Spieler aber nicht, gewinnt der Spieler, sonst der Croupier.

Die Bank hat selbstverständlich ein Interesse daran, dass die Croupiers verlässlich spielen und nicht ins „Zocken“ geraten. Daher haben Croupiers eine sehr feste Anweisung:

„Hast Du eine Gesamtpunktzahl bis w - 1, decke eine weitere Karte auf, ab Gesamtpunktzahl w bleibst Du stehen.“

Das kann man durchaus als Online-Strategie sehen, denn der Croupier muss seine Entscheidung treffen, ohne die Werte der als Nächstes aufgedeckten Karten zu kennen. Insofern wollen wir nun als Repräsentanten einer Spielbank agieren und mittels Spielen zwischen Arne und Wiltrud bestimmen, welcher Wert von w angemessen ist. Als Maßstab kennen wir die Kompetitivität.

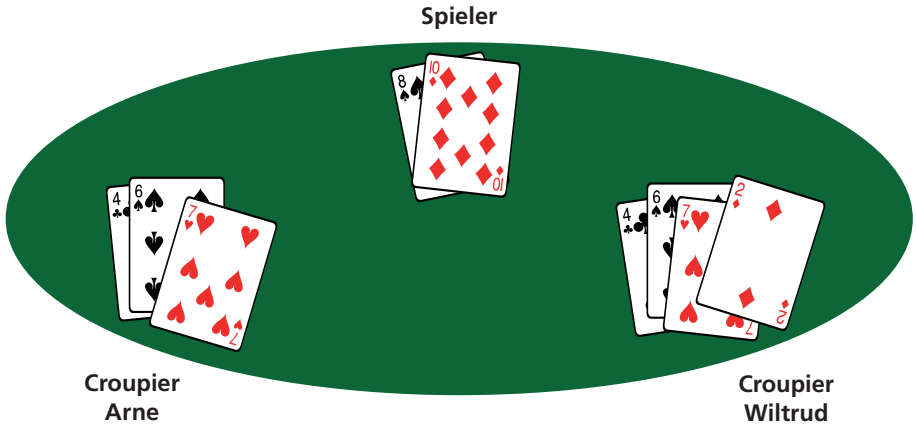
Überlegen Sie, welche Rolle nach diesen Voraussetzungen Arne, welche Wiltrud übernimmt.

Wir wollen, dass der Croupier quasi automatisiert nach Vorschrift spielt. Diese Rolle kommt in unserem Modell Arne zu. Wiltrud darf sowohl den Spieler übernehmen als auch wieder das Schicksal, also alle Kartenwerte bestimmen. Außerdem spielt sie natürlich auch für einen „alternativen“ Croupier, der identische Karten zu denen Arnes bekommt, aber keine vorgegebene Strategie hat – die braucht sie ja auch nicht, denn sie kennt ja alle nachfolgenden Kartenwerte.

Lassen Sie nach diesen Regeln Arne gegen Wiltrud antreten. Versuchen Sie einen für Arne vergleichsweise günstigen Wert von w herauszubekommen.

Wie Sie es drehen und wenden: Wiltrud wird immer die Nase vorne haben. Falls w kleiner 20 ist, gibt sie dem „Spieler“ $w + 1$ Punkte, denn sie kann ja als Schicksal die Werte der aufgedeckten Karten bestimmen. Danach deckt sie für den „Croupier“ genau w Punkte auf. Arne darf nun aufgrund seiner Vorgabe keine weitere Karte nehmen, während Wiltruds alternativer Croupier natürlich noch eine weitere aufdeckt und zufälligerweise mit einer 2 gegenüber dem Spieler gewinnt, während Arnes Croupier seinen Einsatz verloren hat. Mit negativem Gewinn kann Arne daher nie eine wie auch immer bezifferte Wette gegen Wiltrud gewinnen. Abbildung 16.8 zeigt das.

Abbildung 16.8
Arnes Croupier spielt nach der Strategie $w = 17$ und wird von Wiltrud immer geschlagen.

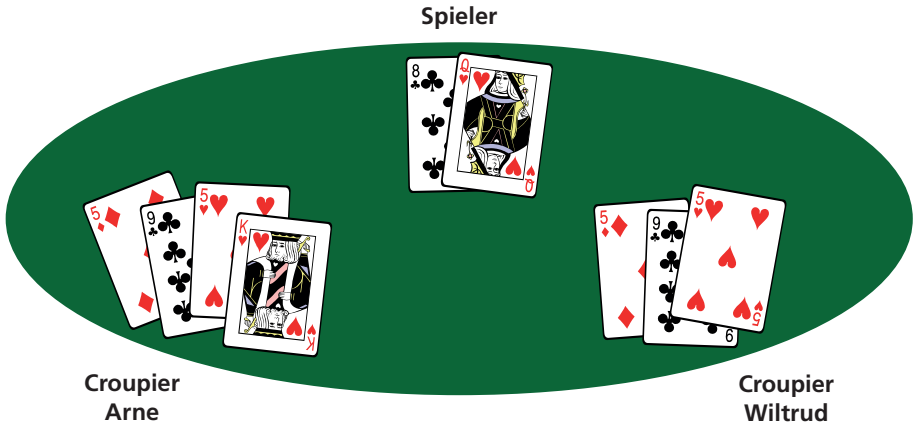


Falls das w in Arnes festgelegter Strategie mindestens 20 beträgt, kann das Wiltrud auf andere Weise konterkarieren. Sie gibt dem Spieler zum Beispiel 18 Punkte, deckt für ihren und Arnes Croupier danach 19 Punkte auf. Ihr Croupier ist zufrieden und hört auf, während Arnes ja – strikt seiner Vorgabe folgend – noch eine Karte aufdeckt und prompt mit einem König über 21 Punkte hat und damit verliert, wie in Abbildung 16.9.

Die Erkenntnis aus dem Experiment muss also aufgrund des festgelegten Qualitätsfaktors „Kompetitivität“ sein, dass es keine vernünftige Online-Strategie für Croupiers in Spielkasinos gibt.

Das geht nun allerdings ein wenig an der Realität vorbei, denn wir wissen auch, dass tausende von Spielkasinos sehr erfolgreich mit der Strategie $w = 17$ fahren. Offenbar stimmt hier also noch etwas mit dem von uns definierten Maß für Qualität nicht.

Abbildung 16.9
Arnes Croupier spielt nach der Strategie $w = 20$ oder $w = 21$ und wird von Wiltrud immer geschlagen.



Verlässliche Würfel

Auch wenn wir bestimmte Ereignisse nicht konkret voraussagen können, so gibt es doch meistens sehr genaue statistische Erkenntnisse, die uns Wahrscheinlichkeiten für das Eintreten der verschiedenen Möglichkeiten liefern.

Eventuell ist Ihnen daher die Vorgehensweise auch bereits beim Schwimmbad-Problem am Anfang des Kapitels etwas seltsam vorgekommen. Der gesunde Menschenverstand rät uns, entweder von Anfang an eine Saisonkarte zu kaufen oder sich auf Dauer mit Einzeltickets zu begnügen. Das Schöne dabei ist: Der gesunde Menschenverstand hat höchstwahrscheinlich recht.

Die „unwissenden" Annahmen in unserem Schwimmbad-Beispiel waren sehr unrealistisch. Natürlich kann es prinzipiell vorkommen, dass in einem Jahr das Wetter mal komplett verrückt spielt und entweder nur einen einzelnen Sonnentag beschert oder auch durchgehend trocken bleibt. Die Wahrscheinlichkeit dafür ist aber ziemlich gering, wenn Sie sich in unseren Breiten befinden. Meistens wird es eine mittlere, recht gut abschätzbare Zahl an Sonnentagen geben. Legt man diese zugrunde, kann man die Kaufentscheidung recht gut treffen.

Erweitern wir unser Modell des Schwimmbad-Problems auch nur um eine Wahrscheinlichkeit *Pt*, mit der an jedem Tag „erwürfelt" wird, ob er der letzte schöne Tag ist. Das bildet die Realität mit unterschiedlichem *Pt* für unterschiedlich sonnige Regionen bereit ziemlich gut ab. Die rechnerisch günstigste Strategie verändert sich dadurch komplett:

Um im Mittel am wenigsten Geld auszugeben, sollte man

- für $Pt < \frac{1}{30}$ sofort eine Saisonkarte kaufen,
- für $Pt > \frac{1}{30}$ sich mit Tagestickets begnügen und
- für $Pt = \frac{1}{30}$ ist es egal.

Sie sehen, dass unabhängig von der tatsächlichen Sonnigkeit auch rechnerisch die beiden Möglichkeiten herauskommen, die uns das Bauchgefühl von Anfang an signalisiert hat.

Eine wichtige Erkenntnis ist also, dass wir das Modell der Wirklichkeit, anhand dessen wir automatisiert Entscheidungen treffen lassen möchten, auch mit den uns zur Verfügung stehenden Informationen über die Wirklichkeit anreichern müssen. Der Zufall ist zwar allgegenwärtig, aber er ist doch meistens statistisch erfassbar und insofern recht verlässlich. Wenn wir das ignorieren, bekommen wir eine interessante akademische Knobelei, aber eben doch keine sinnvolle Entscheidungshilfe für die Realität.

Lassen Sie uns versuchen, Blackjack etwas umfassender zu modellieren. Zunächst hatten wir bereits zu Beginn festgelegt, dass wir von einer gleich bleibenden Wahrscheinlichkeit von $\frac{1}{13}$ für jeden Kartenwert ausgehen. Kartenzählen wäre in unserem Modell also sinnlos.

Kartenzählen

Kartenzählen beruht darauf, dass in „echten" Spielkasinos mit „echten" Kartenspielen gearbeitet wird und daher die Wahrscheinlichkeiten für das Aufdecken bestimmter Werte nicht ganz gleich sind. Meistens werden sechs Kartenstapel ineinandergemischt. Wenn man das Spiel also sehr bewusst verfolgt und bereits 36-mal eine 9 aufgedeckt wurde, ist es unmöglich, dass noch eine 9 kommt, bis ein neuer Kartenstapel angebrochen wird. Spielt man auf Basis dieser Erkenntnisse, kann man die Gewinnaussichten erheblich erhöhen. Spielkasinos verbieten daher in der Regel das Kartenzählen, auch wenn es ziemlich schwer ist, das nachzuweisen.

Als Erstes können wir bestimmen, mit welchen Wahrscheinlichkeiten der Croupier eine bestimmte Gesamtpunktzahl e in einer Runde aufdeckt, wenn er mit Strategie w spielt. Dazu addieren Sie einfach die Wahrscheinlichkeiten sämtlicher Möglichkeiten, auf die entsprechende Punktzahl zu kommen. Zum Beispiel können Sie den Wert 4 erreichen durch 1 ,1, 1, 1 oder 1, 1, 2 oder 1, 2, 1 oder 2, 1, 1 oder 1, 3 oder 3, 1 oder 2, 2 oder 4.

	$w = 14$	$w = 15$	$w = 16$	$w = 17$	$w = 18$	$w = 19$	$w = 20$
p(e=14)	0,1165						
p(e=15)	0,1206	0,1206					
p(e=16)	0,1157	0,1247	0,1247				
p(e=17)	0,1105	0,1194	0,1287	0,1287			
p(e=18)	0,1048	0,1138	0,1231	0,1327	0,1327		
p(e=19)	0,0988	0,1078	0,1170	0,1266	0,1365	0,1365	
p(e=20)	0,1456	0,1546	0,1638	0,1734	0,1833	0,1935	0,1935
p(e=21)	0,0854	0,0944	0,1036	0,1132	0,1231	0,1333	0,1438
p(e=22)	0,0507	0,0596	0,0689	0,0785	0,0884	0,0986	0,1091
p(e=23)	0,0427	0,0517	0,0609	0,0705	0,0804	0,0906	0,1011
p(e=24)	0,0087	0,0445	0,0538	0,0634	0,0733	0,0835	0,0940
p(e=25)		0,0090	0,0461	0,0557	0,0656	0,0758	0,0863
p(e=26)			0,0093	0,0476	0,0575	0,0677	0,0762
p(e=27)				0,0096	0,0492	0,0594	0,0699
p(e=28)					0,0099	0,0507	0,0612
p(e=29)						0,0102	0,0522
p(e=30)							0,0105
p($e \leq 21$)	0,8979	0,8352	0,7610	0,6747	0,5757	0,4634	0,3374

Was bringt Ihnen diese Tabelle, um eine vernünftige Strategie zu ermitteln? Gar nichts, denn der einzige Hinweis ist, dass die Wahrscheinlichkeit, immerhin nicht „bust" zu gehen, für w = 14 am höchsten ausfällt. Der Gegenspieler findet in dieser Statistik noch gar keine Berücksichtigung.

Kleine Knobelaufgabe: Erkennen Sie, warum es da eine kleine „Anomalie" bei der Wahrscheinlichkeit für 20 Punkte gibt? Sowohl 19 als auch 21 Punkte sind in allen Spalten deutlich unwahrscheinlicher!

Genau genommen ist die Wahrscheinlichkeit, eine 10 aufzudecken, viermal so hoch wie die aller anderen Werte, denn es gibt vier Karten mit Wert zehn: 10, Bube, Dame und König. Daher ist auch die Wahrscheinlichkeit für Vielfache von zehn entsprechend höher.

Jetzt dürfen Sie sich aber richtig anstrengen: Ihre Aufgabe lautet, trotz der korrekten Berücksichtigung von Wahrscheinlichkeiten ein Szenario für Blackjack als Online-Problem zu formulieren. Ziel ist immer noch, für den Croupier eine fest vom Kasino vorgegebene Strategie festzulegen, indem wir ihm die Zahl w geben, ab der er keine weitere Karte aufdeckt. Beschreiben Sie, wie Arne diese Strategie stellvertretend ausführt und wie Wiltrud sie auf die Probe stellen kann, ohne wie bisher die Werte der als nächste aufgedeckten Karten aktiv zu bestimmen.

Überlegen wir: Was wäre im Blackjack ein „optimaler Gegenspieler", der aber nicht aktiv den Wert aufgedeckter Karten bestimmen kann? Selbstverständlich einer, der immerhin den Wert und die Position aller Karten im Stapel kennt.

Ein optimaler Spieler würde daher nie „bust" gehen, sondern immer so lange Karten ziehen, bis die nächste ihn über 21 Punkte triebe. Es kann also sein, dass auch der optimale Spieler sich mit 11 Punkten begnügen muss, wenn er weiß, dass die nächste Karte ein As wäre (das nach unseren vereinfachten Regeln ja immer 11 Punkte zählt). Wenn Sie auf die gleiche Weise wie schon für den „normalen Spieler" die Möglichkeiten, auf bestimmte Summen zu kommen, und die Wahrscheinlichkeiten dafür kombinieren, erhalten Sie folgende Wahrscheinlichkeiten für Spielergebnisse des optimalen Spielers:

	optimaler Spieler
p(e=11)	0,0103
p(e=12)	0,0395
p(e=13)	0,0522
p(e=14)	0,0627
p(e=15)	0,0742
p(e=16)	0,0863
p(e=17)	0,0990
p(e=18)	0,1123
p(e=19)	0,1260
p(e=20)	0,1935
p(e=21)	0,1438

Wiltrud übernimmt sozusagen den Part der optimalen Spielerin. Kombinieren Sie beide Tabellen, um Arnes Strategie zu ermitteln.

Die Wahrscheinlichkeit dafür, dass Arne gegen die optimal spielende Wiltrud gewinnt, ist jeweils die Wahrscheinlichkeit, dass er eine bestimmte Punktzahl erhält, kombiniert mit der Wahrscheinlichkeit, dass Wiltrud eine niedrigere Punktzahl hat. Das macht beispielsweise für $w = 19$ bei einer Rechnung auf vier Stellen nach dem Komma genau mit *ea* als Punktergebnis von Arne und *ew* als Punktergebnis von Wiltrud:

$$
\begin{aligned}
&p(\text{Sieg Arnes bei } w = 19) = \\
&p(ea = 19) \cdot p(ew < 19) + \\
&p(ea = 20) \cdot \mathrm{p}(\mathrm{ew} < 20) + \\
&p(ea = 21) \cdot \mathrm{p}(\mathrm{ew} < 21) = \\
&0{,}1365 \cdot (0{,}0103 + 0{,}0395 + 0{,}0522 + 0{,}0627 + 0{,}0742 + 0{,}0863 + 0{,}0990 + 0{,}1123) + \\
&0{,}1935 \cdot (0{,}0103 + 0{,}0395 + 0{,}0522 + 0{,}0627 + 0{,}0742 + 0{,}0863 + 0{,}0990 + 0{,}1123 + 0{,}1260) + \\
&0{,}1333 \cdot (0{,}0103 + 0{,}0395 + 0{,}0522 + 0{,}0627 + 0{,}0742 + 0{,}0863 + 0{,}0990 + 0{,}1123 + 0{,}1260 + 0{,}1935) = \\
&0{,}3155
\end{aligned}
$$

Die Tabelle gegenüber zeigt die Wahrscheinlichkeiten für Arnes Sieg – und damit den des Croupiers – für unterschiedliche Strategien in der letzten Zeile. Wir lernen daraus, dass das Kasino gegen einen optimalen Spieler auf Dauer immer verliert. So weit ist es keine Überraschung.

Wir sehen aus der Tabelle allerdings auch, dass das Kasino gegen einen optimalen Spieler am wenigsten Verlust mit der Strategie $w = 16$ macht, denn damit beträgt die Gewinnwahrscheinlichkeit immerhin 0,3840.

Interessanterweise lassen alle mir bekannten Kasinos der Welt ihre Croupiers mit der Strategie $w = 17$ antreten. Haben die ihre Hausaufgaben nicht gemacht? Sollten wir sie aufklären, dass unsere Lösung des Online-Problems ergeben hat, wie sie ihren Gewinn noch verbessern könnten?

Kaum! Unser letztes Szenario hat zwar bereits realistischere Ergebnisse als der erste Ansatz ergeben, aber es handelt sich immer noch um die Modellierung eines sogenannten Worst-Case-Szenarios. Es ist eine Abschätzung dafür, wie viel das Kasino in jedem Spiel verliert, wenn es tatsächlich jemand schafften sollte, die Karten zu markieren. Warren Beatty ist im 1966er Film „Kaleidoscope“ zu diesem Zweck extra in eine Druckerei eingebrochen und hat die Druckplatten für die Rückseiten manipuliert. Ziemlich viel Aufwand dafür, dass das Kasino trotzdem noch mehr als jedes dritte Spiel gewinnt ...

Ein Kasino muss natürlich davon ausgehen, dass alles mit rechten Dingen zugeht. Ein realistischer Spieler wird daher sicher auf keinen Fall bei 11 stehen bleiben und übernimmt sich auch manchmal, wenn er dann eine hohe Karte aufdeckt. Außerdem ist die Strategie der Spieler nicht unabhängig von der festgelegten Strategie der Croupiers: Muss der Croupier bei 16 noch eine Karte nehmen, veranlasst das die Spieler, ebenfalls ein höheres Ergebnis zu wagen mit dem Effekt, dass sie häufiger „bust“ gehen und verlieren.

Wie im ersten Kapitel, wo wir den Algorithmus zum Finden des kürzesten Weges in seiner Laufzeit nochmals verbessern konnten, indem wir das Modell unserer Landkarte erweiterten, gilt also auch bei den Online-Problemen, dass die Ergebnisse ganz entscheidend von unserem Modell der Realität abhängen. Das Modell erfasst sehr genau mathematisch darstellbare Größen wir Wahrscheinlichkeiten, wie hier die Kartenhäufigkeiten. Es muss aber durchaus auch ein Bild der Menschen zeichnen, die mit dem System interagieren, was lediglich im letzten Absatz angedeutet wird.

	$w = 14$	$w = 15$	$w = 16$	$w = 17$	$w = 18$	$w = 19$	$w = 20$	o. Spl.
$p(e=11)$								0,0103
$p(e=12)$								0,0395
$p(e=13)$								0,0522
$p(e=14)$	0,1165							0,0627
$p(e=15)$	0,1206	0,1206						0,0742
$p(e=16)$	0,1157	0,1247	0,1247					0,0863
$p(e=17)$	0,1105	0,1194	0,1287	0,1287				0,0990
$p(e=18)$	0,1048	0,1138	0,1231	0,1327	0,1327			0,1123
$p(e=19)$	0,0988	0,1078	0,1170	0,1266	0,1365	0,1365		0,1260
$p(e=20)$	0,1456	0,1546	0,1638	0,1734	0,1833	0,1935	0,1935	0,1935
$p(e=21)$	0,0854	0,0944	0,1036	0,1132	0,1231	0,1333	0,1438	0,1438
$p(e=22)$	0,0507	0,0596	0,0689	0,0785	0,0884	0,0986	0,1091	
$p(e=23)$	0,0427	0,0517	0,0609	0,0705	0,0804	0,0906	0,1011	
$p(e=24)$	0,0087	0,0445	0,0538	0,0634	0,0733	0,0835	0,0940	
$p(e=25)$		0,0090	0,0461	0,0557	0,0656	0,0758	0,0863	
$p(e=26)$			0,0093	0,0476	0,0575	0,0677	0,0762	
$p(e=27)$				0,0096	0,0492	0,0594	0,0699	
$p(e=28)$					0,0099	0,0507	0,0612	
$p(e=29)$						0,0102	0,0522	
$p(e=30)$							0,0105	
p(Sieg)	0,3625	0,3779	0,3840	0,3780	0,3565	0,3157	0,2514	

Diese Tabelle zeigt die Gewinnwahrscheinlichkeiten für Wiltrud als optimale, wissende Spielerin (o. Spl.) und Arne als Croupier, der bei einer vorgegebenen Punktzahl w keine weitere Karte aufdecken darf.

Der Gewinn aus dem Spiel

So spielen Modelle von Internet-Nutzern, Kunden in Einzelhandelsläden, Bahnreisenden, Autofahrern usw. auch die entscheidende Rolle, wenn aus unserem Spiel echte Anwendungen werden. Spielziel ist dann immer noch die Optimierung des Gewinns, der aber sehr unterschiedlich definiert sein kann:

- Schaltung von Werbung, die den Internet-Nutzer auch interessiert und mehr zum Kauf anregt.
- Erzielung höherer Umsätze durch gezielte Bonus-Angebote von Kundenkarteninhabern.
- Erhöhung der Pünktlichkeit im Nahverkehr, indem Einsteigezeiten mit eingeplant werden können.

- Bremsbereitschaft autonomer Autos auch unter Berücksichtigung des Verhaltens von Fußgängern, Radfahrern und anderen Autofahrern, die nicht alle Informationen haben.

Worst-Case-Szenarios sind bei manchen dieser Anwendungen unbedingt nötig, denn ein autonomes Auto soll in jeder erdenklichen Situation sicher zum Stehen kommen und dabei niemanden verletzen. In anderen Anwendungen sind sie lediglich für die Abschätzung der Folgen nach unten gedacht, etwa wenn es um Marketing geht. Hier kann man beim Einsatz einer Strategie viel risikobereiter sein und wählt daher eventuell eine, die im schlimmsten Fall einen Totalverlust beschert, aber im Mittel dafür deutlich mehr Gewinn verspricht.

Industrie 4.0 ist ein Schlagwort unserer Zeit. Es verspricht unter anderem, den heute oft extrem komplexen Pfad zu einem verkaufsfertigen Produkt zu „entwirren“: Hier sind teilweise unüberschaubar viele Zulieferer und Dienstleister beteiligt. Die Einzelteile eines Fernsehers kommen häufig aus der ganzen Welt. Transportverzögerungen, Qualitätsprobleme, Lieferengpässe aufgrund von Fehlentscheidungen, Konkurse, etc. – das alles sind recht schwer zu überblickende Unwägbarkeiten, die bisher zum Beispiel durch große Lagerflächen in den Einzelbetrieben kompensiert wurden. Verzögerte sich eine Lieferung, konnte man einfach vorrätige Teile entnehmen. Lagerfläche ist aber teuer und oft auch ökologisch ungünstig, wenn sie geheizt, gekühlt oder sonst irgendwie mit Energie versorgt werden muss.

An vielen Stellen versucht man daher, den kompletten Weg vom Marketing, der Bestellung über die komplexe Produktion bis hin zur Auslieferung als gigantisches Online-Problem zu modellieren und dann auch zu berechnen. Man kann sich vorstellen, dass selbst hier Arne und Wiltrud miteinander spielen. Ich schreibe ganz bewusst miteinander, weil sie ja tatsächlich an einem Strang ziehen und auch die destruktive Intelligenz Wiltruds am Ende dazu führt, das gesamte System sicherer und stabiler zu machen.

Wenn Sie aber hier im Kapitel bereits in Bezug auf die akademischeren Aufgabenstellungen mit sehr verlässlichen Prognosen den Überblick verloren haben, können Sie abschätzen, welche gigantische Aufgabe die Modellierung für „echte“ Herausforderungen ist.

Resümee

Bei den Online-Problemen handelt es sich um eine ganz eigene Betrachtung von Aufgabenstellungen, für deren Lösung man nicht alle eigentlich nötigen Informationen besitzt, sondern für die man etwas „möglichst gut“ lösen muss. Große Relevanz kommt daher der Abschätzung von „gut“ im Kontext der Aufgabenstellung zu, also der Definition von Qualitätskriterien.

Insgesamt erkennt man an den Online-Problemen besonders gut die Relevanz einer guten Modellierung, ohne die auch eine kreative und in anderen Fällen durchaus erfolgreiche Lösungsstrategie versagen muss.

Maximalgewicht 

Arne	Gewicht Goldbrocken	Wiltrud

Abbildung 16.K1
Kopiervorlage für den Spielplan zu den Online-Rucksäcken.

Abbildung 16.K2
Kopiervorlage für den Spielplan zu „Schlüsseldienst“

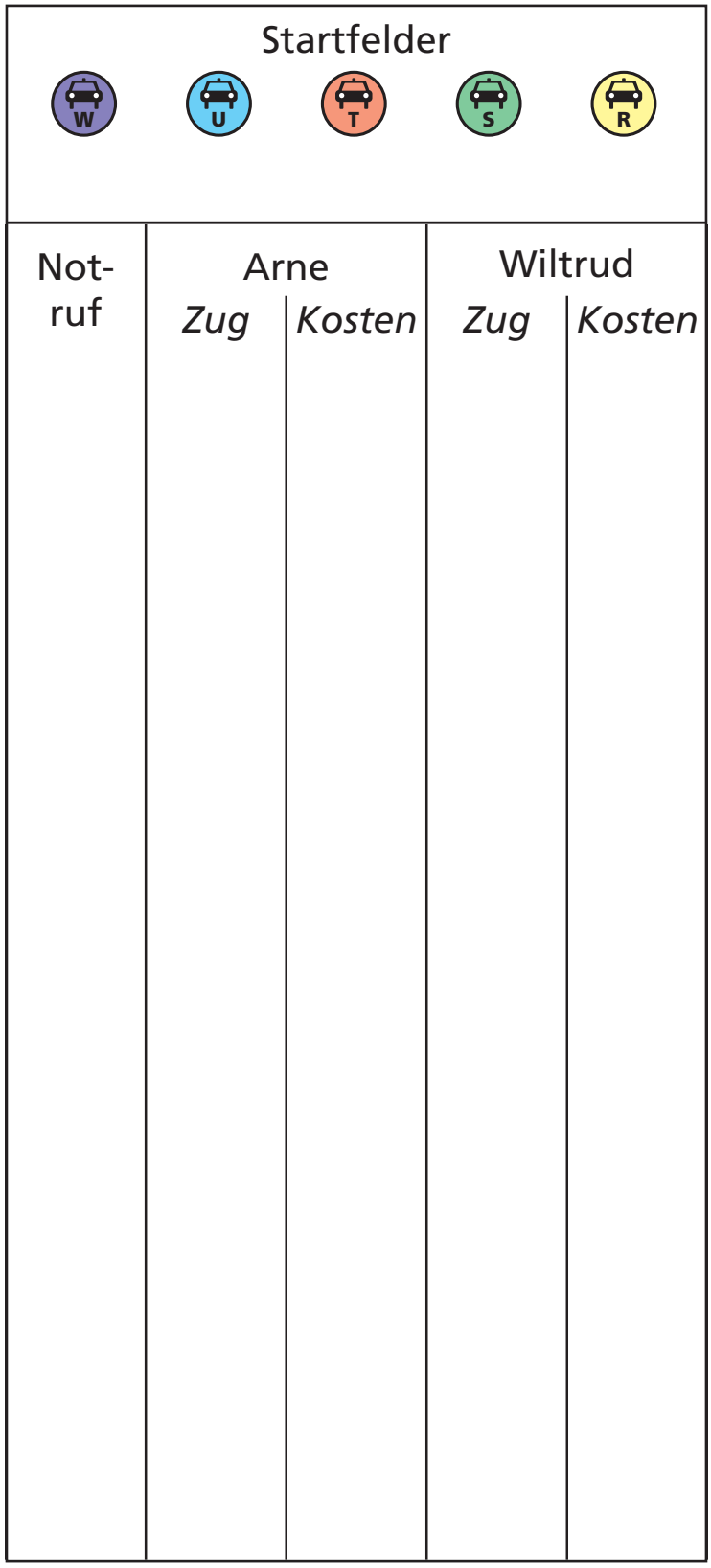


Abbildung 16.K3
Spielfeld für „Schlüsseldienst“ mit den Spielsteine zum Ausschneiden bzw. um eigene „Pöppel“ daraufzustellen. Sie benötigen das 2-mal.

	A	B	C	D	E	F	G	H	I	J
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

Marpurg 1742 in 8. nebst andern lesen. Eine ausführliche Nachricht vom Schulwesen in Europa ertheilet Friedrich Lucä in seinem Europäischen Helicon.

Schule, heisset im Gartenbau ein Ort, da junge Obstbäume aus Kernen gezogen, oder Wildlinge versetzet, die zu gehöriger Zeit gepfropfet werden, und so lange darinnen stehen bleiben, bis sie genugsam erwachsen, und anders wohin an eine bleibliche Stelle versetzet werden können. Wie dergleichen Schulen anzulegen, siehe **Baumschule**, im III Bande, p. 767.

Schule, auf der Reitschule die Lectionen, worauf so wohl die Scholaren, als die Pferde abgerichtet werden. Ein Reiter, der seine Schule gelernet, sitzet wohl zu Pferde, mit einer freyen ungezwungenen Positur, führet die Faust und die Schenckel anständig, und weiß sein Pferd gehörig zu regieren. Ein Pferd, das auf die Schule abgerichtet, hat seinen Schritt, Trab und Gallop wohl gelernet, nimmt die Hülffe wohl auf, fürchtet die Strafe, und macht alle übrigen Lectiones, Capriolen, Courbetten, Passaden, und also mehr mit so guter Art, daß es zu Lust und Ernst, bey Aufzü-

aus der Arithmetick, Geometrie, Baukunst, Mechanick, Hydraulick und Hydrometrie so viel Unterricht erhielten, als zu gründlichem Verstand ihres Mühlbaues erfordert wird, wenn sie nehmlich von allem demjenigen, was sie nach diesem durch Ubung lernen, gnugsamen Grund verstehen wollen. Es wird sich aber von rechter Einrichtung der Handwercks-Schulen alsdenn erst reden lassen, wenn man von allen Künsten und Handwercken tüchtige Beschreibungen haben wird, wenn sie in Form der Wissenschafften werden gebracht worden seyn.

Schule, (Heyden-) siehe **Schule der Heyden.**

Schule, (höhere) siehe **Schule.**

Schule, (hohe) siehe *Academia*, im I p. 238. u. f. wie auch **Schule**, ingleichen **Stär.**

Glossar

Erfahrungsgemäß wird ein Index für ein „Lesebuch“ wie dieses eher spärlich genutzt. Daher sind hier nur die wichtigsten Begriffe alphabetisch mit einer kurzen Erläuterung aufgeführt und mit dem Namen des Kapitels, das den jeweiligen Begriff einführt.

Abstraktion (Sag mir wohin ...)

In zur Verfügung stehender Information stecken sowohl relevante als auch unwesentliche Anteile. Durch Abstraktion reduzieren Sie die Information auf das für die aktuelle Problemlösung Wesentliche: Dadurch können Sie sich besser auf Ihre Aufgabe konzentrieren.

Algorithmus (Sag mir wohin ...)

Ein Algorithmus ist eine Handlungsvorschrift zur Lösung eines Problems bzw. einer Kategorie von Problemen. Diese Handlungsvorschriften lassen sich im Allgemeinen in ein Computerprogramm umsetzen. Hierfür müssen sie hinreichend genau formuliert sein.

A-Stern-Algorithmus (Sag mir wohin ...)

Verfahren, das den Dijkstra-Algorithmus an einer sehr kleinen Stelle verändert und zusätzlich die Information der minimalen Strecke zwischen zwei Punkten einbezieht, um für geographische Aufgabenstellungen bzw. vergleichbare Probleme die Laufzeit deutlich zu optimieren.

Asymmetrische Verschlüsselung (Mit Sicherheit)

Kryptographisches Verfahren, bei dem eine Nachricht, die mit einem Schlüssel verschlüsselt wurde, nur mit einem passenden anderen Schlüssel wieder decodiert werden kann. Nur durch asymmetrische Verfahren ist es möglich, im Internet sicher zu kommunizieren, ohne vorher mit dem konkreten Gesprächspartner auf anderem Wege einen Schlüssel ausgetauscht zu haben.

Aufwandsabschätzung (Ordnung muss sein!)

Mit der Aufwandsabschätzung wird in der Informatik oft die Qualität von Algorithmen bestimmt. Diese Abschätzung gibt an, wie stark die notwendige Rechenzeit in Bezug zur Problemgröße anwächst. Konstante Faktoren werden hier nicht berücksichtigt. Auf diese Weise kann für kleine Problemgrößen die tatsächliche Rechenzeit eines „schlechteren“ Algorithmus unter der des besseren liegen.

Authentizität (Mit Sicherheit)

Das Sicherheitsziel Authentizität ist dann hergestellt, wenn sichergestellt ist, dass eine Nachricht tatsächlich von den angegebenen Absendern stammt. Anders herum ausgedrückt: Es kann niemand unbefugt unter meinem Namen eine Nachricht verschicken.

Backtracking (Alles im Fluss)

Wichtige Entwurfsstrategie der Informatik: „Versuche so lange Teillösungen zu errechnen und zu einer Gesamtlösung zu vereinigen, bis diese entweder gefunden ist oder klar wird, dass diese auf dem aktuellen Weg nicht gefunden werden kann. Gehe dann gezielt einige Schritte zurück und versuche andere Teillösungen. Mache das, bis Du mit der gefundenen Lösung zufrieden bist." Backtracking kann sehr effizient gestaltet werden, wenn man Wege, die nicht zum Ziel führen, möglichst schnell erkennt und frühzeitig abbricht. Diese Vorgehensweise ist dann auch unter „Branch and Bound" bekannt. Im Vergleich mit anderen Strategien beanspruchen solche mit Backtracking jedoch in der Regel deutlich mehr Rechenzeit.

Binärzahlen (Der Trick mit dem Binären)

Eine Zahlendarstellung mit zwei Ziffern. Diese werden üblicherweise durch „0" und „1" repräsentiert. Binäre Darstellungen sind im Computerbereich üblich, weil sie sich direkt durch eine Elektronik verarbeiten lassen, die zwei Zustände („eingeschaltet" und „ausgeschaltet") versteht.

BIT, bit (Von Kamelen und dem Nadelöhr)

Kleinste Informationseinheit. Die Abkürzung wird je nach Zweck als „BInary DigiT" oder „basic indissoluble information unit" übersetzt, steht natürlich auch treffenderweise englisch für „ein bisschen".

Breitensuche (Alles im Fluss)

Standardverfahren, um in einer Datenstruktur bestimmte Elemente aufzufinden und dabei zunächst vom Startpunkt aus in der Nachbarschaft zu suchen. Englisch „Breadth First Search, BFS".

Bubblesort (Ordnung muss sein!)

Sehr einfaches Verfahren zum Sortieren vergleichbarer Elemente, bei dem immer nur zwei direkt nebeneinanderliegende Elemente einer Liste verglichen werden. Es ist für Menschen sehr intuitiv und wird daher auch in der Praxis häufiger eingesetzt als gedacht, wenn es um sehr kleine Aufgabenstellungen geht, bei denen der Zeitbedarf keine besondere Rolle spielt.

Burrows-Wheeler-Transformation (Von Kamelen und dem Nadelöhr)

Verfahren, um insbesondere Texte so umzuformen, dass sie sich viel leichter codieren lassen, um verlustfrei mit weniger Speicherplatz auszukommen.

Codierung (Von Kamelen und dem Nadelöhr)

Verfahren, die Symbole einer Nachricht in eine andere Form zu wandeln, ohne den Informationsgehalt der Nachricht einzuschränken. Die Codierung impliziert, dass eine entsprechende Decodierung möglich ist, um die Nachricht wieder in den Originalzustand zu versetzen. Codierung bedeutet in der Informationstechnik meistens die Umwandlung von der für den Menschen verständlichen Form in eine für Maschinen verarbeitbare und über Netzwerke kommunizierbare Version.

Divide et impera (Ordnung muss sein!)

Sehr häufig hat man im Leben Aufgaben zu lösen, die zu unüberschaubar und groß sind, um sie in einem Ansatz zu lösen. Vielmehr teilen wir das Gesamtproblem auf in mehrere handhabbare Stücke, die wir lösen. Die Teillösungen werden danach nur noch zusammengefasst. Dieses Prinzip wird auch in der Informatik sehr stark verwendet: Ein Programm zerlegt die gestellte Aufgabe zunächst in mehrere kleinere Einheiten, genannt Teilprobleme (divide = teile) und weist danach andere Programme an, diese zu lösen (impera = herrsche, befehlige). Dabei ist sehr wichtig, dass die Teilprobleme unabhängig voneinander gelöst werden können, denn sonst müssten die Programme miteinander kommunizieren, unter Umständen auf Lösungen voneinander warten, was den Aufwand wiederum sehr erhöht. Andere Bezeichnungen sind „Teile und herrsche“ oder „divide and conquer“.

Dijkstra-Algorithmus (Sag mir wohin ...)

Verfahren, das in einem Graphen den kürzesten Weg zwischen einem Startknoten und allen anderen Knoten findet. Der Graph repräsentiert dabei oft eine Landkarte.

Dynamische Programmierung (Ich packe meinen Koffer und ...)

Strategie zur Lösung komplexer Probleme, besonders zur Lösung von Optimierungsaufgaben. Man löst zunächst viele Teilprobleme und verwendet diese Bausteine, um die nächstgrößeren Probleme zu lösen usw., bis das Gesamtproblem gelöst ist.

Elementaroperationen (Ordnung muss sein!)

Operationen bzw. Arbeitsschritte, die wir bei der Bestimmung der Laufzeit eines Algorithmus für wichtig bzw. zeitkritisch erachten. Welche Operationen hier ausgewählt werden, ist nicht global definiert, sondern vom jeweiligen Anwendungsfall abhängig. Oft handelt es sich um Speichervorgänge (Lesen und Schreiben im Speicher).

Entscheidungsbaum (Erkennungsdienst)

Struktur, die ein Verfahren als Folge von Entscheidungen darstellt. Werden alle überhaupt möglichen Entscheidungen aufgeführt, erscheint das als eine Art Baum und ist dann als vollständiger Algorithmus anzusehen.

Gleichformung (Sag mir wohin ...)

Versuchen Sie, die verschiedenen Facetten eines Problems auf die gleichen Grundelemente zurückzuführen. Dadurch wird einerseits das Problem übersichtlicher und andererseits benötigt man weniger Lösungsansätze: Für gleichförmige Teilprobleme kann der gleiche Lösungsansatz verwendet werden.

Graph (Sag mir wohin ...)

Ein in der Informatik sehr beliebtes Modell der Wirklichkeit, das Sachverhalte durch Knoten visualisiert, die durch Kanten verbunden sind. Meistens werden die Knoten als Kreise, die Kanten als Linien oder Pfeile zwischen den Knoten dargestellt. Knoten und Kanten können markiert sein. Die Markierung erscheint visuell oft als Beschriftung.

Greedy-Prinzip (Alles im Fluss)

Folgt ein Verfahren der Informatik dem Greedy-Prinzip, geht es über einen einmal erreichten Stand nicht mehr zurück. Das Greedy-Prinzip ist damit quasi das Gegenstück zum Backtracking. Algorithmen, die dem Greedy-Prinzip folgen, benötigen in der Regel eine kurze Laufzeit. Oft ist es daher günstiger, nur eine Näherungslösung mit Greedy zu berechnen, als das Optimum per Backtracking.

Hashing (Ordnung im Chaos)

Verfahren, Daten im Computer so zu organisieren, dass man mit dessen Hilfe schnell darauf zugreifen kann. Dabei wird eine für den Computer günstige Ordnung auf Basis mathematischer Funktionen gewählt, die für den Menschen oft chaotisch wirkt.

Huffman-Codierung (Von Kamelen und dem Nadelöhr)

Verfahren, die unterschiedlichen Zeichen eines Textes auf Basis ihrer Häufigkeit und damit ihres Shannon'schen Informationsgehaltes durch unterschiedlich lange Binärcodes zu repräsentieren.

Komprimierung (Von Kamelen und dem Nadelöhr)

Bei der Komprimierung wird eine Datenmenge reduziert, indem zwischen relevanten und irrelevanten Informationen unterschieden wird und lediglich die relevanten Informationen beibehalten werden. Diese Unterscheidung wird oft im Medienbereich anhand physiologischer Studien vorgenommen, nach denen bestimmte Informationsteile nicht oder nur von wenigen Menschen wahrgenommen werden können.

Lauflängencodierung (Von Kamelen und dem Nadelöhr)

Bei der Lauflängencodierung werden nicht einzelne Bildpunkte gespeichert bzw. übermittelt, sondern die Anzahl gleicher Bildpunkte hintereinander. Englisch lautet der entsprechende Fachausdruck „run-length". In manchen Bildverarbeitungsprogrammen bekommen Sie beim Speichern die Möglichkeit, „RLC" auszuwählen, was ausgeschrieben „run-length coding" bzw. Lauflängencodierung bedeutet.

Modellbildung (Sag mir wohin ...)

Ein informatisches Modell beschreibt genau die Teile einer Problemstellung übersichtlich, die zur Lösung des Problems nötig sind. Das gilt insbesondere auch für die Modellierung der Problemlösung als Algorithmus oder als Datenstruktur. Modellbildung findet auf vielen unterschiedlichen Ebenen statt: Selbst die Implementierung in einer gängigen Programmiersprache stellt eine Modellierung dar, da sie bestimmte Konzepte und Denkweisen impliziert.

OSI-Modell (Paketpost)

Abkürzung für „Open Systems Interconnection Reference Model". Es bildet die gängige Modellvorstellung für Datenkommunikation zwischen Computern, wie sie heute hauptsächlich im Internet genutzt wird.

Paradigma (Paketpost)

Das Paradigma ist ein (Denk-)Muster oder Vorbild, anhand dessen ein technisches System oder ein abstraktes Konzept konstruiert wird. Dadurch wird es für die Anwender leichter, sich im System zurechtzufinden.

Präfix (Von Kamelen und dem Nadelöhr)

Ein Präfix ist ein Wort oder eine Zeichenfolge, das bzw. die mit dem Anfang einer anderen Zeichenfolge identisch ist. In der Codierung versucht man Präfixe zu vermeiden, da man mit Präfix codierte Nachrichten nicht mehr direkt decodieren kann.

Problem (Sag mir wohin ...)

In der Informatik kann man das mit „herausfordernde Aufgabenstellung" übersetzen. Der Begriff Problem ist also vor allem nicht negativ besetzt.

Problemgröße (Ordnung muss sein!)

Ein wichtiger Vorgang bei der Lösung von Aufgaben aus der Informationstechnik ist das Bestimmen der sogenannten Problemgröße. Sie stellt ein Maß dar, wie schwierig bzw. umfangreich die Aufgabe bzw. das Problem ist. Daher hängt von der Problemgröße entscheidend der zu leistende Aufwand ab.

Proxmap-Sort (Ordnung muss sein!)

Sortierverfahren, das nicht auf dem Vergleich der Elemente basiert, sondern für jedes Element schätzt, wo es in der sortierten Folge hinkommt. Das macht man als Mensch ganz implizit, wenn man Spielkarten auf einem Tisch sortiert und das As ganz automatisch an den Anfang, den König ans Ende legt.

Pseudozufallszahlen (Ordnung im Chaos)

Computer sind verlässliche Maschinen und es ist ohne spezielle Hardware kaum möglich, zu würfeln. Daher muss der Zufall durch mathematische Verfahren simuliert werden. Da dies dann nicht wirklich zufällig ist, sondern nur den Anschein erweckt, nennt man das Resultat Pseudozufallszahl.

Online-Problem (Spielchen gefällig?)

Aufgabenstellung, bei der der Computer nicht alle nötigen Informationen besitzt, um eine optimale Lösung zu berechnen. Daher sind hier Erfahrung und statistische Wahrscheinlichkeit mitzudenken.

Redundanz (InformaGik)

Eigentlich „überflüssige" Daten, die zur Repräsentation der reinen Information eigentlich nicht nötig wären, aber meistens die Fehlertoleranz von Speichern oder Übertragungsstrecken erhöhen.

Rucksackproblem (Ich packe meinen Koffer und ...)

Aufgabenstellung, bei der Elemente in einen vorgegebenen begrenzten Raum optimal angeordnet werden müssen. Meistens wird das mit einem Rucksack anschaulich gemacht. Tatsächlich handelt es sich oft um endliche Ressourcen des Computers wie Rechenzeit oder Speicherplatz, die optimal genutzt werden sollen.

Selection-Sort (Ordnung muss sein!)

Sehr einfaches Verfahren zum Sortieren vergleichbarer Elemente. Wird in der Praxis kaum eingesetzt, ist aber gut, um die Prinzipien zu verstehen.

Steganographie (Mit Sicherheit)

Verfahren, das Botschaften in anderen Daten versteckt. Statt eines Schlüssels braucht man zum Lesen also die Information, wo man suchen muss.

Symmetrische Verschlüsselung (Mit Sicherheit)

Kryptographisches Verfahren, bei dem eine Nachricht, die mit einem Schlüssel verschlüsselt wurde, mit demselben Schlüssel wieder decodiert werden kann. Um sicher zu kommunizieren, muss mit dem konkreten Gesprächspartner auf sicherem Wege (zum Beispiel durch persönliche Treffen) ein Schlüssel ausgetauscht werden.

Tournament-Sort (Ordnung muss sein!)

Rein akademisches Verfahren, um Elemente zu sortieren. Ist aber sehr gut geeignet, um ein Verständnis für die Verbesserung des Aufwands zum Sortieren zu erlangen und dann etwa das tatsächlich eingesetzte Heapsort zu verstehen.

Tiefensuche (Alles im Fluss)

Standardverfahren, um in einer Datenstruktur bestimmte Elemente aufzufinden und dabei einem Pfad auch in die Tiefe der Struktur zu folgen, ohne die direkte Nachbarschaft zu priorisieren. Engllisch „Depth First Search, DFS".

Verlustfreie Komprimierung (Von Kamelen und dem Nadelöhr)

Dies ist ein populäres Synonym für eine Codierung, bei der die Datenmenge im durchschnittlichen Fall reduziert wird. Da es sich um eine Codierung handelt, ist gewährleistet, dass die Originaldaten aus dem Code wiederhergestellt werden können.

Vertraulichkeit (Mit Sicherheit)

Das Sicherheitsziel Vertraulichkeit ist dann hergestellt, wenn nur die Adressaten einer Nachricht deren Informationen erschließen können. Anders herum ausgedrückt: Eine vertrauliche Nachricht kann nicht von Dritten gelesen werden.

Volladdierer (Rechnen mit Strom)

Logisches Bauteil eines Computers, mit dessen Hilfe eine Addition von drei Zahlen vorgenommen wird, die jeweils 1 Bit lang sind. Aus vielen dieser Volladdierer, kann man ganz einfach Addierwerke für beliebig lange Binärzahlen zusammensetzen.

Vollständige Induktion (Ich packe meinen Koffer und ...)

Die vollständige Induktion ist ein zweistufiges mathematisches Beweisverfahren. Es funktioniert insbesondere für Aussagen in Bezug auf ganze Zahlen. Man beweist, dass die Aussage für eine bestimmte kleine Zahl a gilt. Ferner beweise man: Falls die Aussage für die Zahl $n - 1$ gilt, dann gilt sie auch für die Zahl n. Damit hat man dann bewiesen, dass die Aussage für alle ganzen Zahlen größer oder gleich a gilt.

Zertifikat (Mit Sicherheit)

Bestätigung eines öffentlichen Schlüssels, die normalerweise von einer vertrauenswürdigen Stelle ausgestellt wird, der Certification Authority.


T B 100 50 25
1 : 4.5 F = 10,5 cm.
Juwella – Anastigmat
∞
2
3
4,5
5,6
8
11
16
22
32
5
7

Bildnachweis

Alle hier nicht explizit ausgewiesenen Abbildungen und Photos einschließlich der Kapiteleingangsbilder wurden von Jens Gallenbacher erstellt. Teilweise wurden hierfür IMSI MasterClips (www.imsi.de) verwendet.

Alle im Buch befindlichen gezeichneten Portraits von Personen wurden von Cornelia Kandler angefertigt, einschließlich der fiktiven Personen für das Spiel „Erkennungsdienst“. Alle Verwertungsrechte für diese Illustrationen liegen bei Jens Gallenbacher.

Weitere Bilder:

Seite 3, Marginalienspalte, „Das blaue Pferd“ von Franz Marc, 1911, aus The Yorck Project, 10.000 Meisterwerke der Malerei, DVD-ROM, 2002

Seite 134, Abbildung 5.1, Abakus im Technoseum Mannheim, photographiert mit freundlicher Genehmigung

Seite 135, Abbildung 5.2, Nachbau der Schickard'schen Rechenmaschine im Technoseum Mannheim, photographiert mit freundlicher Genehmigung

Seite 136, Abbildung 5.3, Automatischer Webstuhl mit Lochkarten im Technoseum Mannheim, photographiert mit freundlicher Genehmigung

Seite 143, Abbildung 5.8, ENIAC, U.S. Army Photo from K. Kempf, „Historical Monograph: Electronic Computers Within the Ordnance Corps“

Seite 143, Abbildung 5.9, ENIAC, U.S. Army Photo from the archives of the ARL Technical Library

Seite 143, Abbildung 5.10, UNIVAC, Werbeanzeige, ca. 1956

Seite 146, Abbildung 5.11 unten, Apple II aus der Alfred-Delp-Schule Dieburg, photographiert mit freundlicher Genehmigung

Seite 147, Abbildung 5.12, Erste Computermaus, Courtesy of SRI International, Menlo Park, CA.

Bastelbögen

Bitte besuchen Sie die Webseite

www.abenteuer-informatik.de

um sich über die Bezugsmöglichkeit der Bastelbögen für „Abenteuer Informatik“ zu informieren.