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

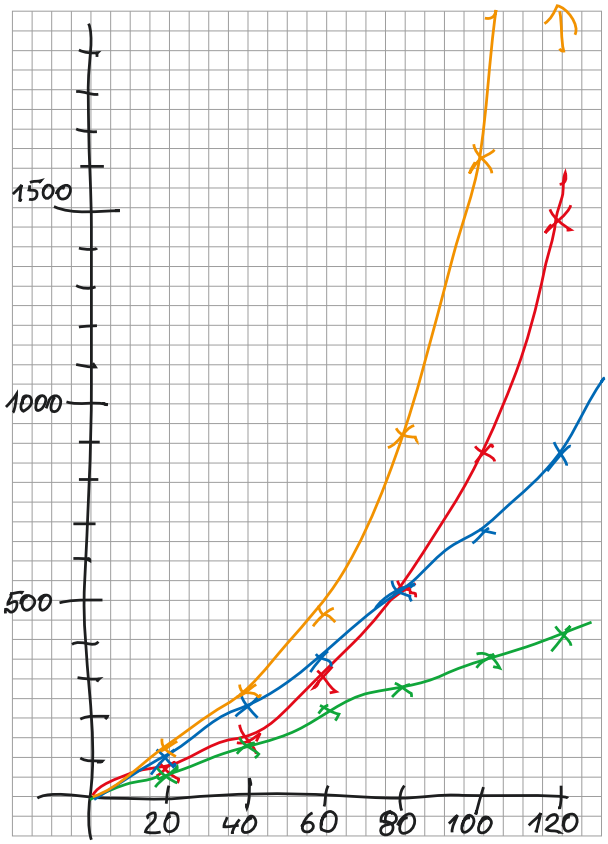


Abbildung 2.1
Graph Anzahl der Karten vs. Zeit in Sekunden.

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.

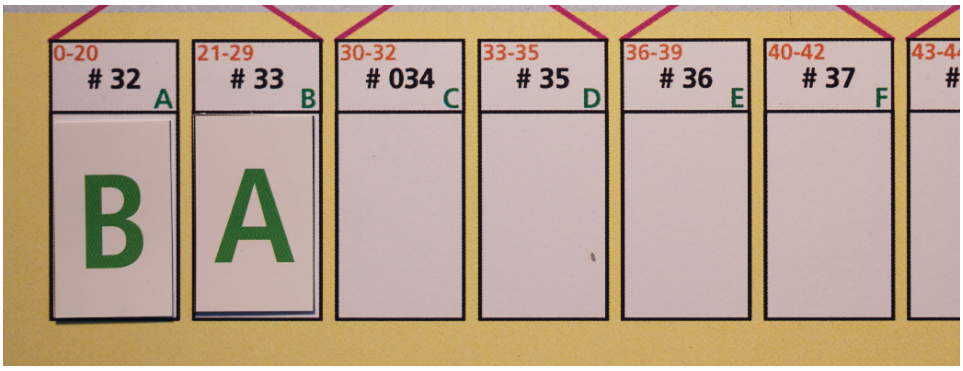
Der größte Unterschied zwischen Mensch und Maschine besteht beim Sortieren in der Speicherung: Der menschliche Kartensortierer besitzt quasi überall „Speicher“ – auf der Hand, als Stapel auf dem Tisch, ausgelegt auf dem Tisch usw. Beim Computer ist der Speicher streng in Positionen organisiert und jede 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 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.

Abbildung 2.2
Papierspeicher mit unsortierten Karten



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.

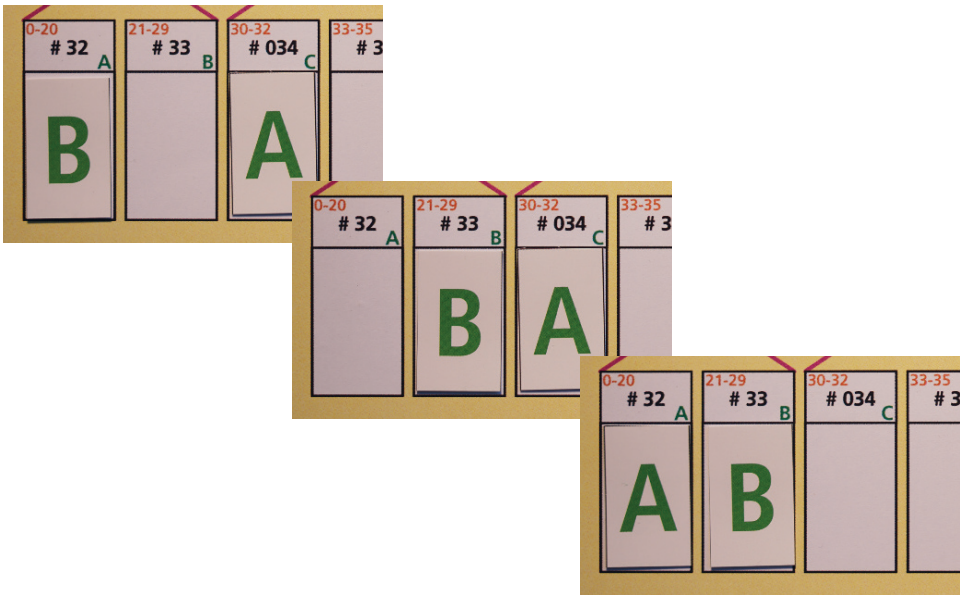


Abbildung 2.3
Kartentausch nur mit Hilfe einer zusätzlichen Speicherstelle

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.

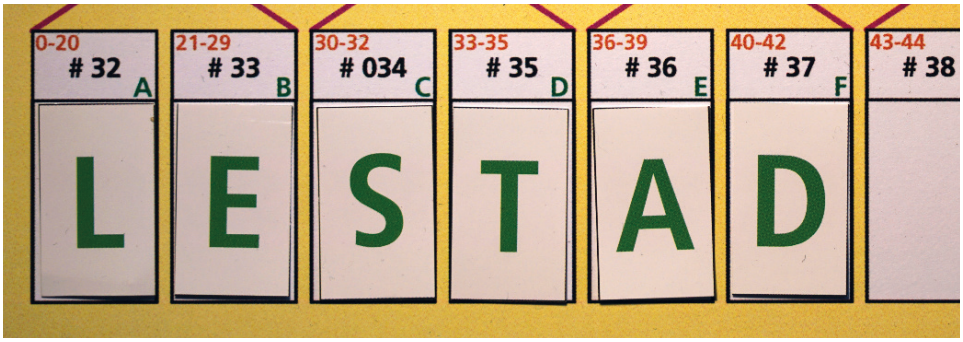


Abbildung 2.4
Unsortierte Karten im Speicher

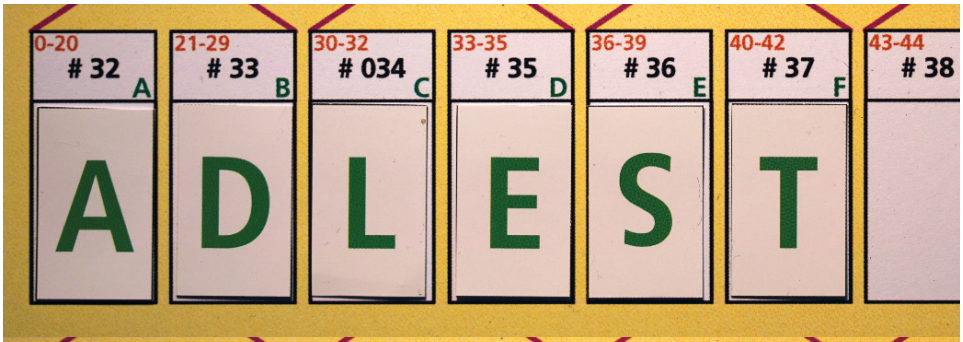


Abbildung 2.5
Nach den ersten Sortierschritten

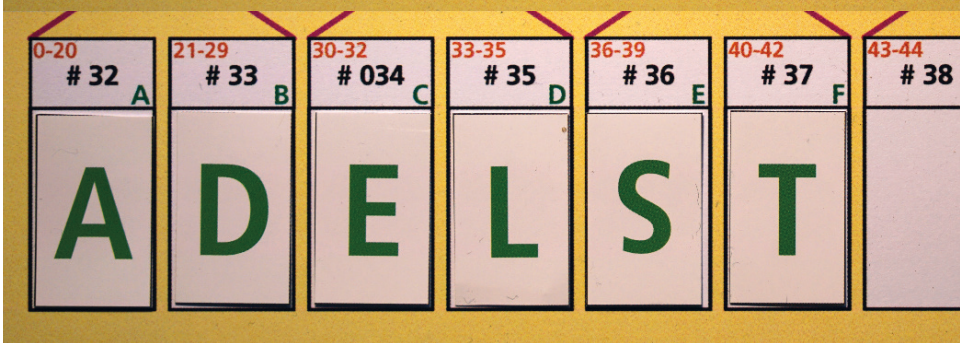


Abbildung 2.6
Die sortierte Folge

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.

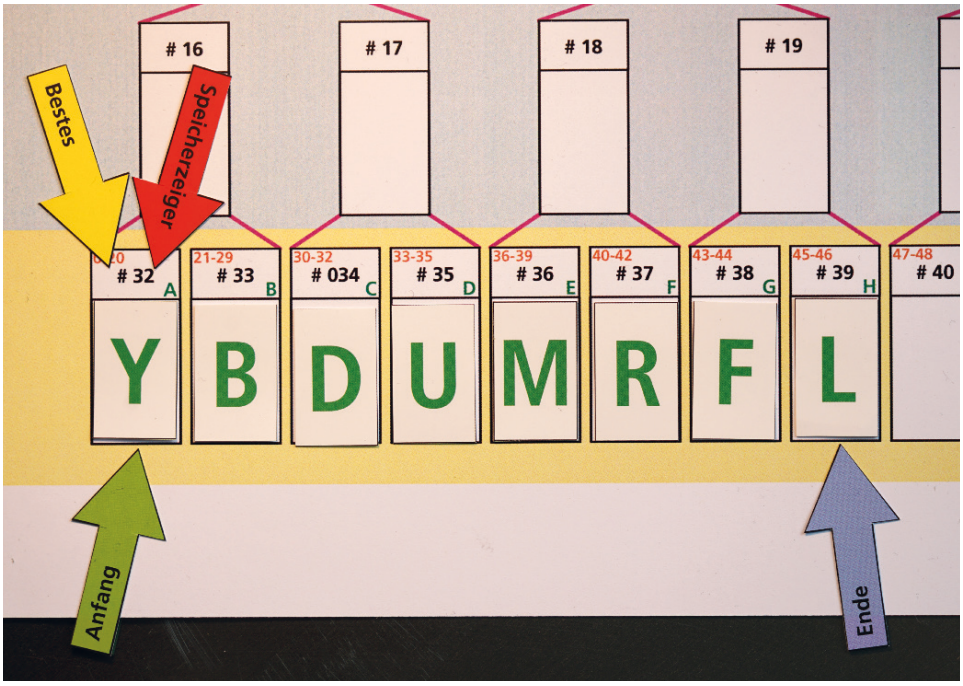


Abbildung 2.7
Sortierfolge mit Zeigern

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.

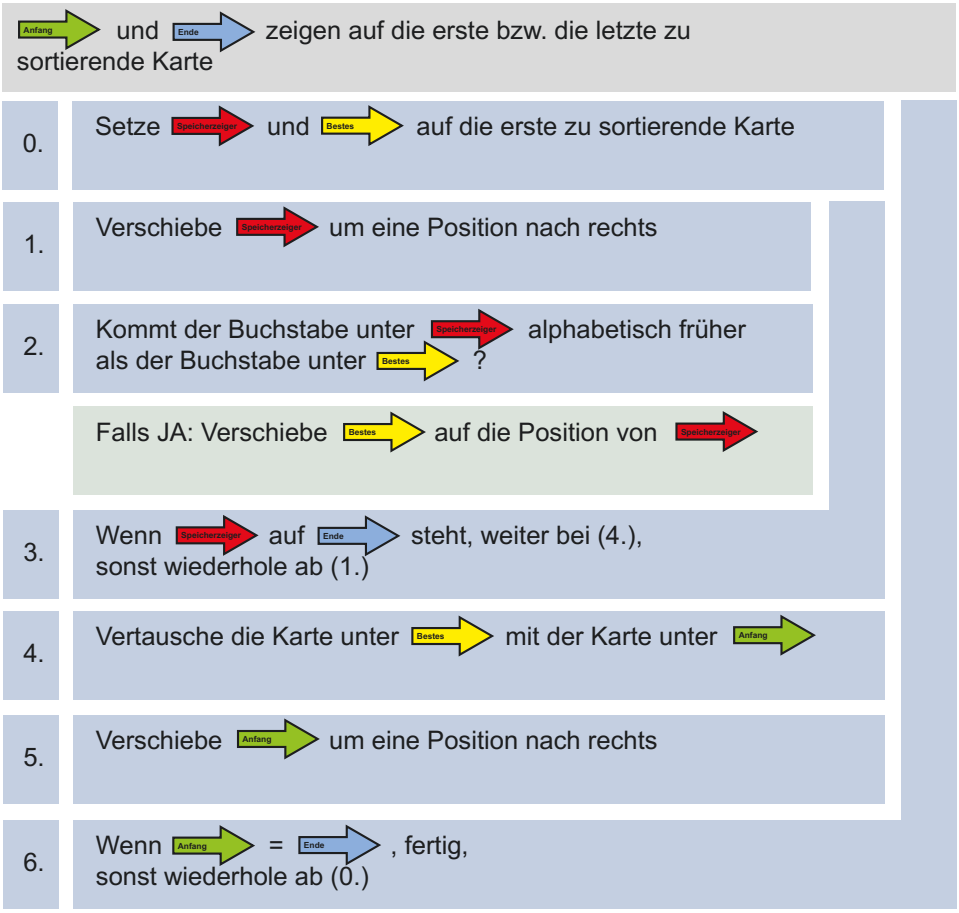


Abbildung 2.8
Die hier verwendete Notation für Algorithmen ist eine vereinfachte Form der von Isaac Nassi und Ben Shneiderman entwickelten modularen Darstellungsart. Im deutschen Sprachraum ist sie unter den Bezeichnungen „Nassi-Shneiderman-Diagramm" oder „Struktogramm" bekannt.

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.

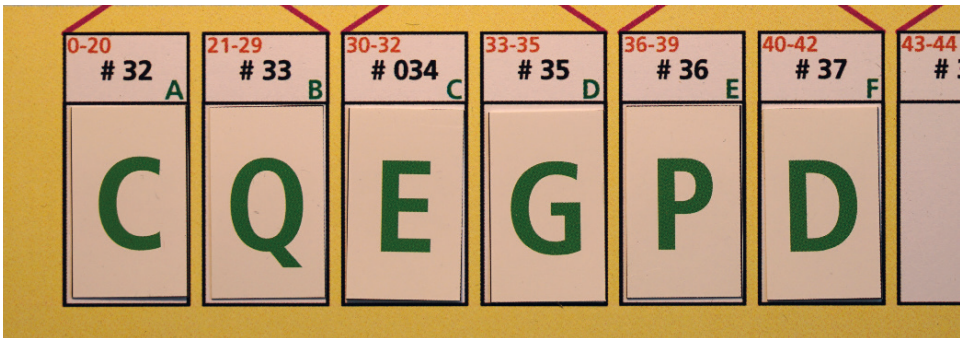


Abbildung 2.9
Unsortierte Folge

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

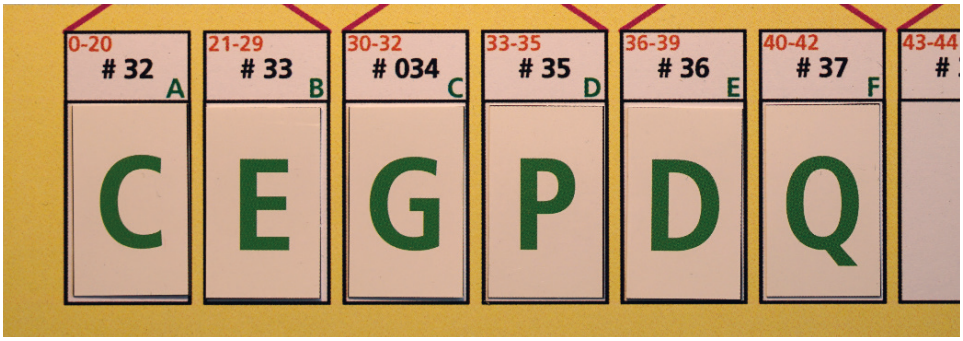


Abbildung 2.10
Folge nach einem Durchlauf des Verfahrens

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.

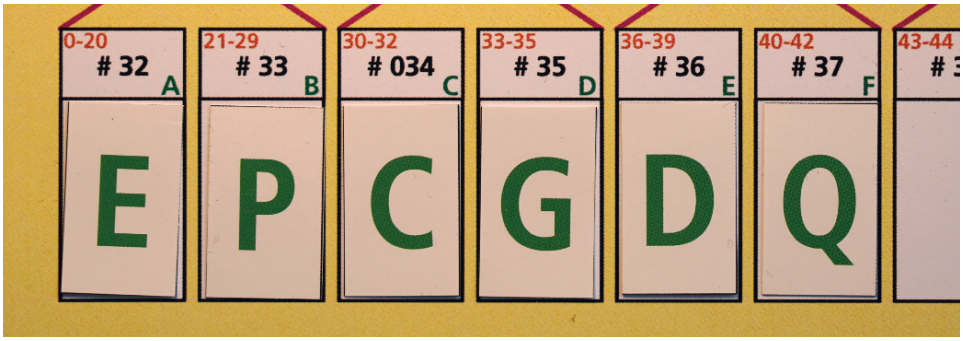


Abbildung 2.11
Unsortierte Folge

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.

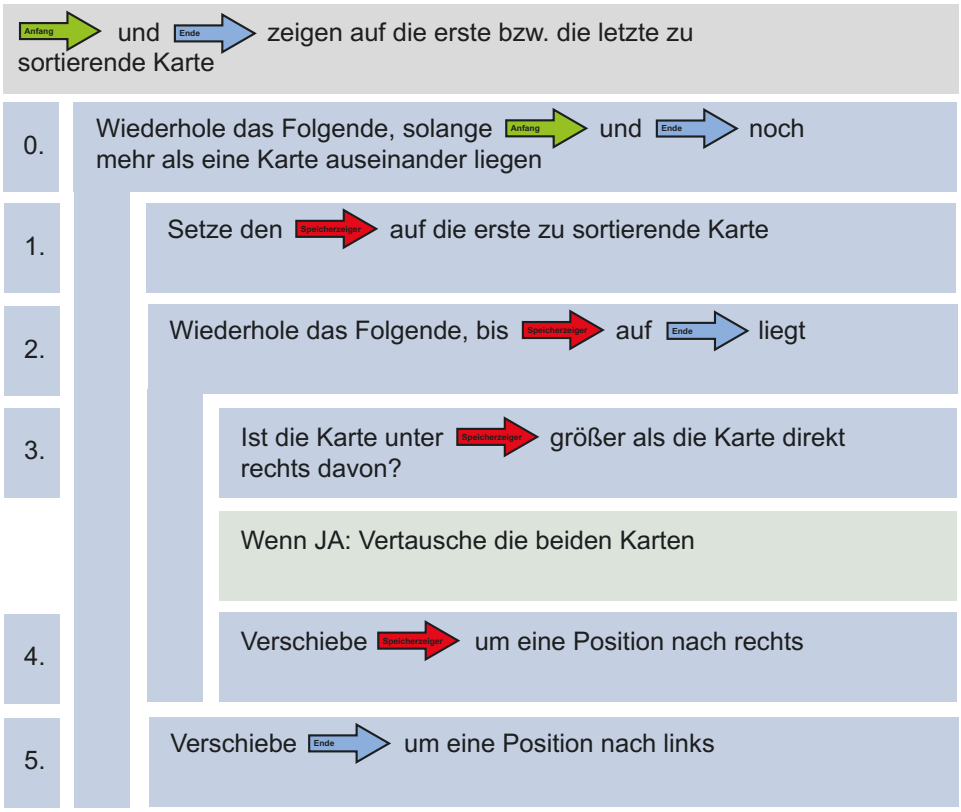


Abbildung 2.12
Ein mögliches Struktogramm für Bubblesort

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.

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.

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).

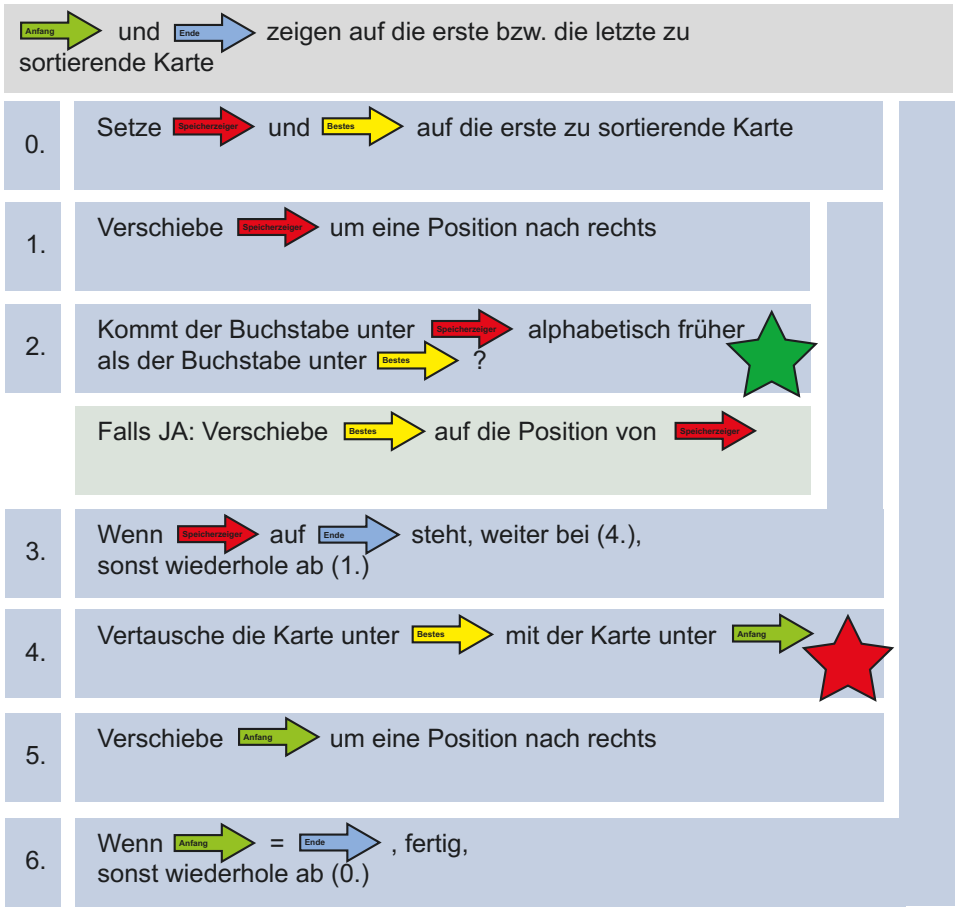
Preisfrage: Wie viele unserer Elementaroperationen benötigt das weiter oben beschriebene Verfahren zum Tausch zweier Karten im Papierspeicher?

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.

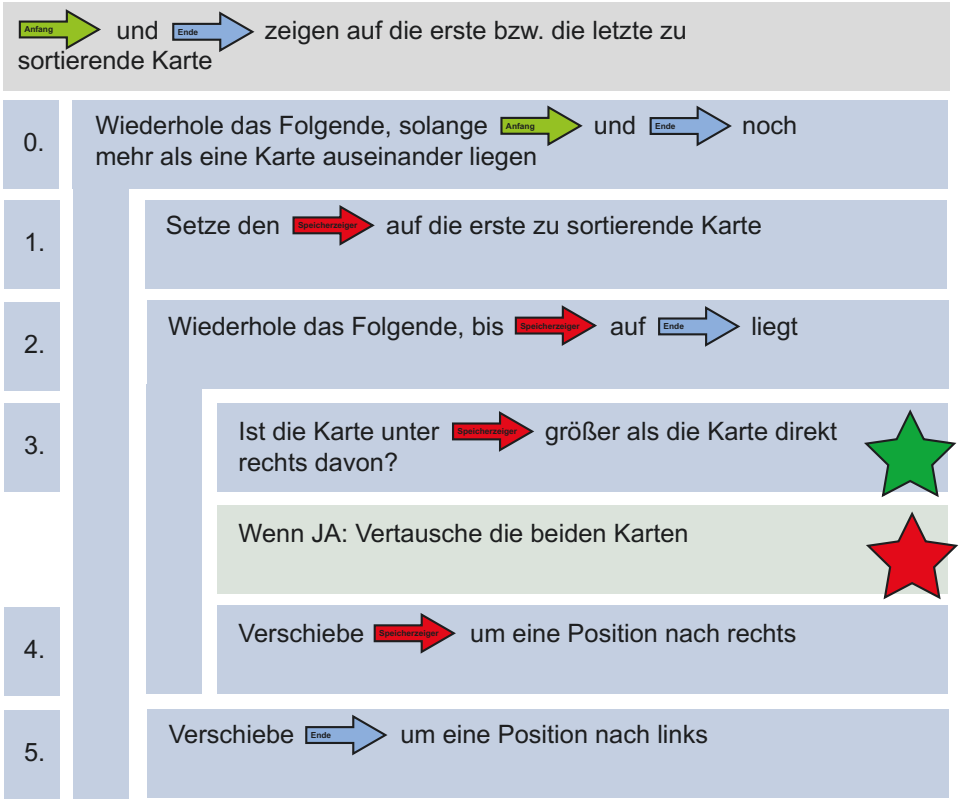


Abbildung 2.14
Bubblesort mit markierten Elementaroperationen

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 (=Problem-größe)	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

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

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

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:

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

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

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.