Leitprogramm Bubblesort

Dr. Rainer Hauser

Inhalt

1 Übersicht

Sortieren – also die Tätigkeit, Listen von Elementen in eine gewünschte Reihenfolge zu bringen – gehört zu den Aufgaben, die ein Rechner sehr häufig ausführen muss. Wenn Sie die Adressen Ihrer Freunde in einem Spreadsheet gespeichert haben, möchten Sie diese vielleicht erst alphabetisch nach Vornamen, anschliessend nach Nachnamen und zum Schluss nach dem Wohnort sortieren. Im Spreadsheet ist das ganz einfach. Was aber passiert hinter den Kulissen?

Sie werden im Folgenden einen Sortier-Algorithmus – er heisst Bubblesort – kennen lernen, und Sie werden auch lernen, welche Aussagen man über seine Effizienz machen kann. Das, was Sie dadurch gelernt haben, können Sie erst an einer Übung überprüfen und zum Schluss beim Tutor testen, ob Sie diese Unterrichtseinheit verstanden haben. Und jetzt: Viel Vergnügen!

2 Input-Block I: Der Sortieralgorithmus Bubblesort

Für die folgenden Aufgaben brauchen Sie Papierschnipsel mit den unten stehenden Zahlen. Schreiben Sie diese Zahlen also auf ein Notizpapier und zerschneiden Sie es. Die folgende Anordnung benutzen Sie in den Aufgaben 1a bis 1d jeweils als **Ausgangssituation**:

51 13 9 44 18 93 25

> **Aufgabe 1a:**
> Sortieren Sie die Papierschnitzel ausgehend von der obigen Reihenfolge der Grösse nach aufsteigend, – also so, dass die kleinste Zahl links und die grösste rechts zu liegen kommt.

Können Sie Ihr Vorgehen so beschreiben, dass jemand es Schritt für Schritt wiederholen kann? Vermutlich nicht, denn Sie haben wohl die Papierschnipsel auf dem Tisch herum geschoben, diejenigen, die sie gerade nicht brauchen konnten, zur Seite gelegt, und die grossen Zahlen rechts platziert, um sie zur Hand zu haben. Sie haben also sehr schnell und etwas chaotisch gearbeitet, sodass Sie jetzt nicht mehr genau rekonstruieren können, was Sie gemacht haben, oder Ihr Vorgehen zumindest nicht so genau aufschreiben können, dass ein Mensch oder gar ein Rechner es wiederholen kann.

> **Aufgabe 1b:**
> Bringen Sie die Papierschnipsel wieder in die Ausgangssituation und sortieren Sie sie zum zweiten Mal der Grösse nach aufsteigend. Lösen Sie diese Aufgabe diesmal aber, indem Sie als einzige Operation auf den Papierschnipsel erlauben, zwei benachbarte Schnipsel zu vertauschen (siehe Abbildung 1).

Möglicherweise haben Sie die Zahl 51, die ursprünglich ganz links lag, in mehreren nacheinander ausgeführten Schritten mit ihren jeweiligen Nachbarn vertauscht, bis sie auf der rechten Seite links vor der Zahl 93 zu liegen kam. Andernfalls bekommen Sie gleich Gelegenheit, dies zu tun und zu beobachten, was dabei geschieht.

Beispiel:

Situation bei ①: 44 (linke Zahl), 18 (rechte Zahl)

Situation bei ②: 18 (linke Zahl), 44 (rechte Zahl)

Abbildung 1:
Der Vergleich eines Paars benachbarter Zahlen als Ausschnitt eines Flussdiagrams

Aufgabe 1c:
Bringen Sie die Papierschnitzel nochmals in die Ausgangssituation und sortieren Sie sie wieder durch Vertauschen von Nachbarn, aber wählen Sie diesmal die Nachbarn systematisch von links nach rechts, indem Sie erst, falls nötig, die erste mit der zweiten Zahl vertauschen, dann die zweite mit der dritten, die dritte mit der vierten und so weiter, bis sie beim letzten Paar ganz rechts angekommen sind (siehe Abbildung 2).
Was stellen Sie fest? Wann entscheiden Sie sich, benachbarte Zahlen zu vertauschen, und wann behalten Sie die bestehende Reihenfolge bei?

Die Zahlen sind nach dem ersten Durchgang von links nach rechts noch nicht in der gewünschten Reihenfolge. Beginnen Sie also wieder links und führen Sie einen zweiten, dritten und so weiter Durchgang durch, bis die Zahlen so sortiert sind wie verlangt.
Was stellen Sie fest? Können Sie etwas darüber sagen, was bei einem einzelnen Durchgang passiert? Wann können Sie aufhören und brauchen keinen weiteren Durchgang mehr?

Wenn Sie die Aufgaben bis hierher gelöst haben, sind Sie schon ganz nahe bei dem, was der Bubblesort tut, um eine Liste von Elementen zu sortieren. Bevor wir aber zur Beschreibung dieses Algorithmus kommen, schauen wir uns eine Variation davon an.

Aufgabe 1d:
Wiederholen Sie, was Sie in der letzten Aufgabe gemacht haben, gehen Sie diesmal aber von rechts nach links mit dem Vertauschen der Nachbarn.
Was ändert sich?

Damit sind wir in der Lage, den Bubblesort-Algorithmus in Worten zu beschreiben. Sein Name – "bubble" ist Englisch und heisst „Blase" – kommt daher, dass die grossen Zahlen nach rechts (oder die kleinen Zahlen nach links) wandern, so wie Blasen im Wasser aufsteigen.

Definition:
Der Bubblesort-Algorithmus sortiert eine Liste von Zahlen aufsteigend, indem er so lange von links nach rechts durch die Liste geht und benachbarte Zahlen vertauscht (falls die linke Zahl grösser als die rechte ist), bis ein ganzer Durchgang durch die Liste zu keiner Änderung mehr führt.

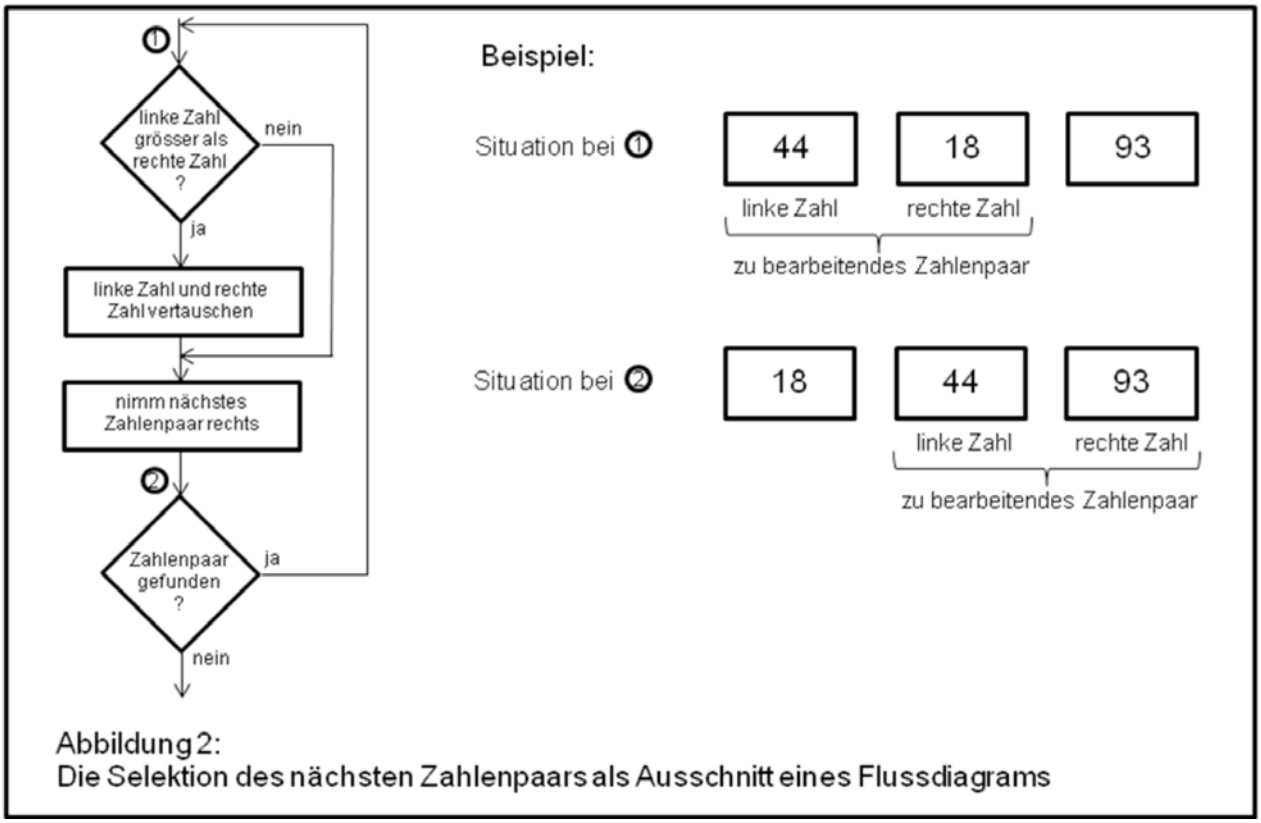

Abbildung 2:
Die Selektion des nächsten Zahlenpaars als Ausschnitt eines Flussdiagrams

Diese Definition ist noch nicht so allgemein, wie man sie gerne hätte. Mit der folgenden Aufgabe finden Sie zwei Richtungen, in der sie verallgemeinert werden kann.

Aufgabe 1e:
Was müssen Sie an diesem Algorithmus ändern, um eine Liste von Zahlen absteigend – also so, dass die grösste Zahl links und die kleinste rechts zu stehen kommt – zu sortieren? Und was müssen Sie ändern, um statt Zahlen der Grösse nach Namen alphabetisch zu sortieren?

Sie haben im ersten Teil dieses Leitprogramms den Bubblesort-Algorithmus für das Sortieren einer Liste von Elementen kennen gelernt (siehe Abbildung 3). Kontrollieren Sie nun, ob Sie alles korrekt verstanden haben.

Wissenssicherung 1:
Sortieren Sie die folgende Liste mit dem Bubblesort-Algorithmus und schreiben Sie dabei sämtliche Zwischenschritte auf:

97 15 33 28 25 11 73

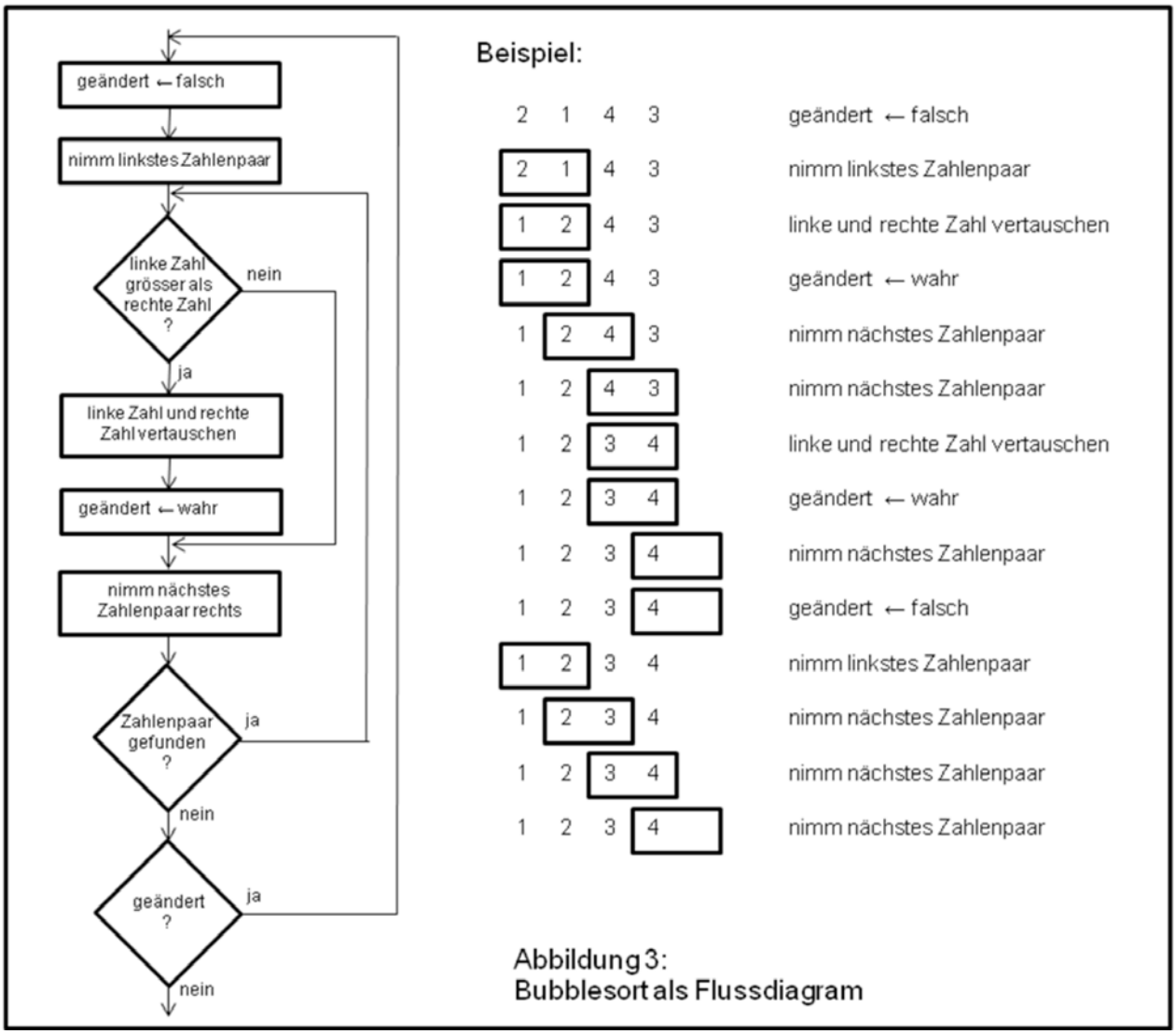

Abbildung 3:
Bubblesort als Flussdiagram

3 Input-Block II: Die Effizienz von Bubblesort

Nachdem Sie den Bubblesort-Algorithmus kennen gelernt haben, fragen wir uns, welche Aussagen wir darüber machen können, wie lange ein Rechner mit dem Sortieren einer Liste beschäftigt ist. Das scheint auf den ersten Blick unmöglich, weil wir gar nichts wissen über die Geschwindigkeit, mit der ein Rechner arbeitet. Das ist richtig, wir werden aber sehen, dass wir trotzdem sehr nützliche Aussagen machen können.

Aufgabe 2a:
Überlegen Sie sich, wie viele Durchgänge von links nach rechts maximal nötig sind, um eine Liste mit *n* Elementen zu sortieren?
Tipp: Lesen Sie nochmals die Bemerkung zur Lösung des Arbeitsauftrags für die Wissenssicherung 1.

Wie Sie vermutlich gemerkt haben, genügen *n* Durchgänge, denn mit jedem Durchgang landet mindestens eine der Zahlen an seinem definitiven Platz. (Weil nach *n-1* Durchgängen alle bis auf höchstens eine Zahl nicht mehr mit ihren Nachbarn auf der linken Seite vertauscht werden müssen, die letzte Zahl aber keinen Nachbarn mehr hat, mit dem sie vertauscht werden könnte, genügen sogar *n-1* Durchgänge. Das ist wie beim Schreiben von Briefen. Es braucht mindestens zwei Briefe und zwei Umschläge, um einen Brief in den falschen Umschlag zu stecken. Hier braucht es auch mindestens zwei Zahlen, die am falschen Platz liegen.)

Fakultative Knobelaufgabe:
Kann man die Tatsache, dass bei jedem Durchgang mindestens eine Zahl auf der rechten Seite an ihren endgültigen Platz verschoben wird, verwenden, um den Bubblesort-Algorithmus zu verbessern, dass er Paare, von denen schon im Vorneherein klar ist, dass sie nicht vertauscht werden müssen, erst gar nicht anschaut? (Man nennt Schritte, die die Effizienz eines Algorithmus verbessern und den Algorithmus somit schneller machen, Optimierungen.)

Um den Aufwand abzuschätzen, den ein Rechner zum Sortieren von *n* Zahlen benötigt, begnügen wir uns damit zu wissen, dass der Rechner für den Vergleich von zwei benachbarten Zahlen eine bestimmte messbare Zeit und für das Vertauschen von zwei benachbarten Zahlen ebenfalls eine bestimmte messbare Zeit braucht, und dass diese Zeiten mehr oder weniger unabhängig von den Zahlen sind, die wir vergleichen oder vertauschen. Wenn er also für den Vergleich von einer Anzahl Zahlenpaaren eine bestimmte Zeit braucht, können wir annehmen, dass er für den Vergleich von doppelt so vielen Zahlenpaaren auch etwa doppelt so lange braucht.

Aufgabe 2b:
Wie viele Vergleichsoperationen benötigt der Rechner für das Sortieren einer Liste von *n* Zahlen mit dem Bubblesort maximal?

In einer Liste mit *n* Zahlen gibt es *n-1* Paare von benachbarten Zahlen, die bei einem Durchgang verglichen werden müssen, und aus der Aufgabe 2a wissen wir, dass wir höchstens *n* Durchgänge benötigen. Das heisst, *n(n-1)* Vergleichsoperationen sind maximal nötig. (Falls Sie die fakultative Knobelaufgabe gelöst haben, wissen Sie, dass bei einer optimierten Version des Bubblesort-Algorithmus die Hälfte dieser Operationen genügen.)

Weil es bei den Aufwandabschätzungen nicht darum geht, die benötigte Zeit auf ein paar Prozent genau zu bestimmen, sondern nur die Grössenordnung festzustellen, begnügen wir uns mit der Abschätzung n^2 Vergleichsoperationen für eine Liste mit *n* Elementen.

Aufgabe 2c:
Wie viele Vergleichsoperationen benötigt der Rechner für das Sortieren einer Liste von *n* Zahlen mit dem Bubblesort minimal? Was lässt sich in diesem Fall über die Liste sagen?

Wenn die Liste bereits sortiert ist, vergleicht der Algorithmus alle *n-1* benachbarten Zahlenpaare einmal und stellt fest, dass es nichts zu tun gibt. Damit ist er fertig, und der Rechner braucht in diesem Fall keine Zahlen zu vertauschen. Das ist der beste und somit schnellste Fall und benötigt *n-1* Vergleiche und 0 Vertauschungen.

Umgekehrt, falls eine Liste absteigend sortiert ist und aufsteigend sortiert werden soll, wird bei jedem Vergleich festgestellt, dass das Zahlenpaar vertauscht werden muss. Dies ist der schlechteste und somit langsamste Fall und benötigt *n(n-1)/2* Vergleiche und gleich viele Vertauschungen.

Jetzt wollen wir noch kurz abschätzen, wie viele Zahlen der Rechner im Normalfall, also im Fall, dass die Liste nicht schon sortiert ist, vertauschen muss. Das lässt sich nicht genau angeben, weil wir ja nichts über die ursprüngliche Reihenfolge wissen. Es lässt sich aber sicher sagen, dass der Rechner nicht mehr Vertauschungsoperationen als Vergleichsoperationen ausführen muss. Wenn wir annehmen, dass in der Hälfte der Fälle eine Vergleichsoperation feststellt, dass die rechte Zahl grösser ist als die linke, und in der anderen Hälfte, dass die linke Zahl grösser ist als die rechte, so können wir daraus ableiten, dass der Rechner etwa halb so viele Vertauschoperationen wie Vergleichsoperationen ausführt.

Trotz diesen tiefsinnigen Überlegungen machen wir es uns einfach und gehen davon aus, dass Vergleichen und Vertauschen etwa gleich aufwändig ist und der Aufwand des Bubblesort mit n^2 Operationen abgeschätzt werden darf. (Was eine Operation genau ist, wissen wir nicht, und es braucht uns hier auch nicht zu kümmern.)

Aufgabe 2d:
Wir nehmen an, dass das Telefonbuch der Schweiz etwa 5'000'000 Einträge hat. Wie lange braucht ein Rechner zum Sortieren dieser Telefonbucheinträge, wenn er zum Sortieren von 1000 Einträgen eine Hundertstelsekunde braucht?

Falls Sie auf 25'000'000 – (5'000'000 / 1000)^2 – Hundertstelsekunden also fast 3 Tage gekommen sind, haben Sie sich nicht verrechnet. Heutige Computer können zwar 1000 Einträge schneller als in einer Hundertstelsekunde sortieren, aber der quadratisch mit der Anzahl Element anwachsende Aufwand bleibt natürlich bestehen.

Wissenssicherung 2:
Wenn das Sortieren der 5'000'000 Einträge des Telefonbuches der Schweiz zwei Sekunden dauert, wie lange braucht derselbe Computer für das Sortieren des Telefonbuches von Deutschland mit 60'000'000 Einträgen?

4 Zusammenfassung

- Sie haben in dieser Unterrichtseinheit einen Algorithmus namens Bubblesort kennen gelernt, der Listen mit n Elementen aufsteigend oder absteigend sortieren kann.
- Sie haben auch seinen Aufwand abschätzen gelernt und dabei als besten und schlechtesten Fall die bereits richtig sortierte und die umgekehrt sortierte Liste gefunden.
- Dabei haben Sie am Bespiel der Telefonbücher der Schweiz und Deutschlands ein Gefühl dafür bekommen, was die quadratische Abhängigkeit des Aufwandes bedeutet.

5 Lernkontrolle

Damit sind Sie am Ende dieser Unterrichtseinheit angekommen. Lösen Sie die folgenden Aufgaben zur Lernkontrolle. Wenn Sie die korrekte Lösung gefunden haben, sind Sie bereit für den abschliessenden Test der Unterrichtseinheit beim Tutor oder bei der Lehrperson. Wenn beim Lösen Fragen auftauchen, die Sie nicht selber beantworten können, so gehen Sie nochmals zum betreffenden Input-Block zurück und lesen Sie die relevanten Stellen ein weiteres Mal durch. Bringt auch das keine Klärung, wenden Sie sich an den Tutor, der Ihnen weiterhelfen wird.

Lernkontrollaufgabe 1:
Sortieren Sie die folgende Liste mit dem Bubblesort-Algorithmus und schreiben Sie sämtliche Zwischenschritte auf:

88 13 25 93 52 48 19 5 83

Erklären Sie, weshalb dieser Algorithmus Bubblesort heisst.

Lernkontrollaufgabe 2:
Begründen Sie, weshalb der Bubblesort-Algorithmus auf einer Liste mit *n* Elementen *n(n-1)/2* Vertauschungen braucht, wenn die Liste ursprünglich umgekehrt sortiert ist als gewünscht.
Tipp: Probieren Sie es mit den Papierschnipsel aus der ersten Aufgabe aus.

6 Lösungen

Lösung zur Wissenssicherung 1

97	15	33	28	25	11	73
15	**97**	33	28	25	11	73
15	**33**	**97**	28	25	11	73
15	33	**28**	**97**	25	11	73
15	33	28	**25**	**97**	11	73
15	33	28	25	**11**	**97**	73
15	33	28	25	11	**73**	**97**
15	**28**	**33**	25	11	73	97
15	28	**25**	**33**	11	73	97
15	28	25	**11**	**33**	73	97
15	**25**	**28**	11	33	73	97
15	25	**11**	**28**	33	73	97
15	**11**	**25**	28	33	73	97
11	**15**	25	28	33	73	97

Bemerkung:
Beachten Sie, dass bei einem ganzen Durchgang von links nach rechts mindestens eine Zahl an ihren endgültigen Platz verschoben wird.

Lösung zur fakultativen Knobelaufgabe

Weil nach dem ersten Durchgang die grösste Zahl ganz rechts liegt, muss man im nächsten Durchgang die letzten beiden Zahlen nicht mehr vergleichen. Aus demselben Grund muss man nach dem zweiten Durchgang die letzten beiden und die vorletzten beiden Zahlen nicht mehr vergleichen. Man kann sich also nach jedem Durchgang einen Vergleichsschritt einsparen.

Lösung zur Wissenssicherung 2

Der Computer braucht ungefähr 288 Sekunden, also knapp 5 Minuten.

Bemerkung:
Nur die Grössenordnung und nicht das genaue Resultat ist wichtig. Falls Sie mit Ihrem Computer eine Liste mit 5 Millionen Elementen in 2 Sekunden sortiert haben und anschliessend eine Liste mit 60 Millionen Elementen sortieren, sollten Sie sich nicht wundern, wenn der Computer nach 3 Minuten noch nicht fertig ist. Wenn er jedoch nach einer Stunde immer noch am Arbeiten ist, können Sie davon ausgehen, dass etwas nicht stimmt.

Lösung zur Lernkontrollaufgabe 1

88	13	25	93	52	48	19	5	83
13	**88**	25	93	52	48	19	5	83
13	**25**	**88**	93	52	48	19	5	83
13	25	88	**52**	**93**	48	19	5	83
13	25	88	52	**48**	**93**	19	5	83
13	25	88	52	48	**19**	**93**	5	83
13	25	88	52	48	19	**5**	**93**	83
13	25	88	52	48	19	5	**83**	**93**
13	25	**52**	**88**	48	19	5	83	93
13	25	52	**48**	**88**	19	5	83	93
13	25	52	48	**19**	**88**	5	83	93
13	25	52	48	19	**5**	**88**	83	93
13	25	52	48	19	5	**83**	**88**	93
13	25	**48**	**52**	19	5	83	88	93
13	25	48	**19**	**52**	5	83	88	93
13	25	48	19	**5**	**52**	83	88	93
13	25	**19**	**48**	5	52	83	88	93
13	25	19	**5**	**48**	52	83	88	93
13	**19**	**25**	5	48	52	83	88	93
13	19	**5**	**25**	48	52	83	88	93
13	**5**	**19**	25	48	52	83	88	93
5	**13**	19	25	48	52	83	88	93

Die Zahlen steigen wie Blasen im Wasser auf und gehen nach rechts und links, wie man an den Zahlen 93, 88 und 5 in diesem Beispiel sehr schön sieht.

Lösung zur Lernkontrollaufgabe 2

Die grösste Zahl steht anfänglich ganz links und wird im ersten Durchgang in *n-1* Vertauschungen ans rechte Ende der Liste gebracht. Die übrigen Elemente der Liste wandern einen Platz nach links. Jetzt liegt die zweitgrösste Zahl ganz links und wird im zweiten Durchgang in *n-2* Vertauschungen ans rechte Ende der Liste links der grössten Zahl transportiert. So geht das weiter, bis beim letzten Durchgang die beiden verbleibenden Zahlen links vertauscht werden:

$$(n-1) + (n-2) + \ldots + 1 = n(n-1)/2$$