



Prof. Dr. Kurt Mehlhorn  
Michael Dirnberger

WiSe 2015/16

## Übungen zu Ideen der Informatik

<http://www.mpi-inf.mpg.de/departments/algorithms-complexity/teaching/winter15/ideen/>

Blatt 3

Abgabeschluss: 16.11.2015

**Aufgabe 1 (10 Punkte)** Nehmen Sie an, Sie wollen eine Menge mit einer Million Zahlen sortieren. Sie wissen, dass die Zahlen aus  $\{1, 2, 3, 4, 5\}$  kommen. Sie wissen aber nicht, wie häufig die Zahlen sind. Es könnten eine Million Einsen sein.

Überlegen Sie sich ein Sortierverfahren für diese Art Mengen, das weniger aufwändig ist als *Sortieren durch Mischen*. Skizzieren Sie, wie der Arbeitsaufwand mit der Grösse der Eingabemenge wächst. Wie verhält sich bei Ihrem Verfahren der Arbeitsaufwand, wenn man doppelt, oder vier mal so viele Zahlen sortieren möchte?

**Aufgabe 2 (10 Punkte)** Sortieren Sie mit Hilfe der folgenden Anweisungen die Menge  $\{0, 7, 5, 1, 2, 3, 6, 4\}$  und geben Sie alle Schritte an.

```
Sei L eine leere Liste
für jedes Element e aus der Eingabe
  Sei i gleich 0
  solange i < Länge(L) und L[i] <= e
    erhöhe i um 1
  füge e an die i-te Position in L ein
```

Hier ist  $L[i]$  das  $i$ -te Element in  $L$  und wir fügen etwas an die  $i$ -te Position ein, indem wir alle Elemente an Positionen größer oder gleich  $i$  um eins nach rechts verschieben.

Wie verhält sich die Anzahl der Vergleiche, wenn sie doppelt, oder vier mal so viele Zahlen sortieren möchten?

**Aufgabe 3 (10 Punkte)** *Kurt's Kleine Knobelei*: Das folgende Programm realisiert das Mischen zweier sortierter Folgen  $A$  und  $B$  in eine sortierte Folge  $C$ . Lesen Sie das Programm sorgfältig und formulieren Sie Verständnisfragen.

Für folgende Antworten bekommen sie die volle Punktzahl.

- Ich habe das Programm 15 Minuten studiert und habe keine Verständnisfragen.
- Ich habe das Programm 15 Minuten studiert und habe Verständnisfragen formuliert.

Falls sie die erste Antwort geben, wird Ihr Tutor Sie bitten die Verständnisfragen ihrer Kommiliton(inn)en zu beantworten.

Die folgenden Konstrukte haben Sie bereits in Form von Pseudocode kennen gelernt. Diesmal formulieren wir ein Programm mit Hilfe der Programmiersprache C++.

```
1 while ( i < n or j < n) {
2   // zu wiederholende Anweisungen
3 }
```

Liest sich als: solange  $i < n$  oder  $j < n$  ist, führe die Anweisungen zwischen den geschweiften Klammern aus.

```
1 for ( int i = 1; i < n; i++) {
2   // zu wiederholende Anweisungen
3 }
```

liest sich als:  $i$  hat anfänglich den Wert 1 und wird in jeder Wiederholung um 1 erhöht. Der Vorgang bricht ab, sobald  $i < n$  falsch ist. Es folgt das C++ Programm:

```
1 #include <vector>
2 #include <iostream>
3 #include <chrono>
4 #include <cassert>
5 #include <random>
6
7 using namespace std;
8 using namespace chrono;
9
10 int main(){ // Programmstart
11
12   int n; // Laenge der zu mischenden Listen
13
14   cout << "Geben Sie bitte ein positive ganze Zahl ein: ";
15   // Wir lesen die Eingabe des Nutzers und speichern sie in der Variable n
16   cin >> n;
17
18   vector<int> A(n), B(n), C(2*n);
19   // Wir haben nun Speicherzellen mit den Namen A[0] bis A[n-1],
20   // B[0] bis B[n-1], und C[0] bis C[2*n - 1]
21
22   A[0] = 0; // speichere Null in A[0] und B[0]
23   B[0] = 0;
24
25   // Bereite Zufallszahlengenerator vor fuer ganze Zahlen zwischen
26   // 1 und 10
27   random_device rd;
28   mt19937 gen(rd());
29   uniform_int_distribution<> random_number(1, 10);
30
31   for (int i = 1; i < n; i++) {
32     // A[i] ist A[i-1] plus eine zufaellige Zahl zwischen 1 und 10.
33     A[i] = A[i-1] + random_number(gen);
34     B[i] = B[i-1] + random_number(gen);
35   }
36
37   // Wir merken uns die aktuelle Uhrzeit in Sekunden
38   system_clock::time_point start = system_clock::now();
39
40   /* Wir mischen nun A und B und speichern das Resultat in C. Dazu
```

```

41 benutzen wir zwei Indizes i und j. Index i indiziert das vorderste
42 Element in A und Index j indiziert das vorderste Element in B. Wir
43 vergleichen die beiden Elemente und bewegen das kleinere der beiden
44 Elemente (im Fall von Gleichheit das Element aus A) nach C. Die
45 richtige Stelle in C ist C[i + j]. Falls eine der beiden Folgen
46 erschöpft ist, bewegen wir das vorderste Element aus der anderen
47 Folge nach C.*/
48
49 int i = 0;
50 int j = 0;
51
52 // Wiederhole solange eine der beiden Folgen nicht erschöpft ist
53 while ( ( i < n ) or ( j < n ) ) {
54
55     if ( i < n and j < n ) { // Beide Folgen sind nicht erschöpft
56
57         if ( A[i] <= B[j] ) {
58             // Wir uebernehmen aus der ersten Folge
59             C[i + j] = A[i]; i = i + 1;
60         } else {
61             // Wir uebernehmen aus der zweiten Folge
62             C[i + j] = B[j]; j = j + 1;
63         }
64
65     } else { // Nun ist eine der beiden Folgen erschöpft
66
67
68         if ( j == n ) {
69             // Die B-Folge ist erschöpft und wir uebernehmen aus A
70             C[i + j] = A[i]; i = i + 1;
71         } else {
72             // Die A-Folge ist erschöpft und wir uebernehmen aus B
73             C[i+j] = B[j]; j = j + 1;
74         }
75
76     }
77
78 }
79
80 // Wir schauen wieder auf die Uhr und geben die Differenz zur
81 // gemerkten Zeit aus
82 duration<double> elapsed_time = system_clock::now()-start;
83 cout <<"Verstrichene Zeit: " << elapsed_time.count() << " [s]\n";
84
85 for (int i = 1; i < n; i++) {
86     assert(C[i] >= C[i-1]); // Wir ueberpruefen, ob C sortiert ist.
87 }
88
89 } // Programmende

```

Suchen und Sortieren war spannend  okay  langweilig   
 schwierig  okay  einfach