Universität des Saarlandes
FR 6.2 Informatik

Prof. Dr. Kurt Mehlhorn, Adrian Neumann **WiSe 2014/2015**

Übungen zu Ideen der Informatik

http://resources.mpi-inf.mpg.de/departments/d1/teaching/ws14/Ideen-der-Informatik/

Blatt 3 Abgabeschluss: 17.11.14

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.

a) Ich habe das Programm 15 Minuten studiert und habe keine Verständnisfragen.

b) 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 Kommilliton(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++.

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

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

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

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:

```
#include <vector>
#include <iostream>
#include <chrono>
#include <cassert>
#include <random>

using namespace std;
using namespace chrono;

int main(){ // Programmstart

  int n;  // Laenge der zu mischenden Listen

  cout << "Geben Sie bitte ein positive ganze Zahl ein: ";
  // Wir lesen die Eingabe des Nutzers und speichern sie in der Variable n
  cin >> n;

  vector<int> A(n), B(n), C(2*n);
  // Wir haben nun Speicherzellen mit den Namen A[0] bis A[n-1],
  // B[0] bis B[n-1], und C[0] bis C[2*n - 1]

  A[0] = 0;     // speichere Null in A[0] und B[0]
  B[0] = 0;

  // Bereite Zufallszahlengenerator vor fuer ganze Zahlen zwischen
  // 1 und 10
  random_device rd;
  mt19937 gen(rd());
  uniform_int_distribution<> random_number(1, 10);

  for (int i = 1; i < n; i++) {
    // A[i] ist A[i-1] plus eine zufaellige Zahl zwischen 1 und 10.
    A[i] = A[i-1] + random_number(gen);
    B[i] = B[i-1] + random_number(gen);
  }

  // Wir merken uns die aktuelle Uhrzeit in Sekunden
  system_clock::time_point start = system_clock::now();

/* Wir mischen nun A und B und speichern das Resultat in C. Dazu
```

```
benutzen wir zwei Indizes i und j. Index i indiziert das vorderste
Element in A und Index j indiziert das vorderste Element in B. Wir
vergleichen die beiden Elemente und bewegen das kleinere der beiden
Elemente (im Fall von Gleichheit das Element aus A) nach C. Die
richtige Stelle in C ist C[i + j]. Falls eine der beiden Folgen
erschoepft ist, bewegen wir das vorderste Element aus der anderen
Folge nach C.*/

  int i = 0;
  int j = 0;

  // Wiederhole solange eine der beiden Folgen nicht erschoepft ist
  while ( ( i < n  ) or ( j < n ) ) {

    if ( i < n and j < n ) {  // Beide Folgen sind nicht erschoepft

      if ( A[i] <= B[j] ) {
        // Wir uebernehmen aus der ersten Folge
        C[i + j] = A[i]; i = i + 1;
      } else {
        // Wir uebernehmen aus der zweiten Folge
        C[i + j] = B[j]; j = j + 1;
      }

    } else { // Nun ist eine der beiden Folgen erschoepft


      if ( j == n ) {
        // Die B-Folge ist erschoepft und wir uebernehmen aus A
        C[i + j] = A[i]; i = i + 1;
      } else {
        // Die A-Folge ist ersch\"opft und wir uebernehmen aus B
        C[i+j] = B[j]; j = j + 1;
      }

    }

  }

  // Wir schauen wieder auf die Uhr und geben die Differenz zur
  // gemerkten Zeit aus
  duration<double> elapsed_time = system_clock::now()-start;
  cout <<"Verstrichene Zeit: " << elapsed_time.count() << " [s]\n";

  for (int i = 1; i < n; i++) {
    assert(C[i] >= C[i-1]); // Wir ueberpruefen, ob C sortiert ist.
  }

} // Programmende
```

Sortieren war spannend ☐ okay ☐ langweilig ☐
schwierig ☐ okay ☐ einfach ☐