

Übungen zu Algorithmen

Wintersemester 2003/2004

Blatt 7

Aufgabe 7.1 (20 Punkte)

Betrachten Sie die Java-Klasse `/home/ainf/Uebung/Blatt7/Aufg7_1.java`.

1. Welches Problem löst die Methode `main` dieser Klasse?
2. Zeigen Sie die partielle Korrektheit mit Hilfe von Zusicherungen, die Sie als Kommentare in den Programmtext einfügen (siehe Programmtext).
3. Zeigen Sie die totale Korrektheit durch den Nachweis der Terminierung.
4. Geben Sie die asymptotische Laufzeit in der O-Notation an.

Aufgabe 7.2 (20 Punkte)

Implementieren Sie in Java folgende Sortieralgorithmen als Methoden, die ein Integer-Array sortieren (analog zu `/home/ainf/Vorlesung/BubbleSort.java`):

- a) Verbesserter Bubblesort, der sich die Position des letzten Austausches eines Durchlaufs merkt und bei den folgenden Durchläufen deshalb bereits dort aufhört zu vergleichen.
- b) *Shakersort*, der die Idee in a) bei abwechselndem Vorwärts- und Rückwärtslaufen anwendet.

Testen Sie Ihre Methoden.

Aufgabe 7.3 (20 Punkte)

Gegeben sei die Zahlenfolge 7, 3, 9, 2, 1, 6, 5, 7, die aufsteigend sortiert werden soll. Stellen Sie die Arbeitsweise der folgenden Sortieralgorithmen durch Zwischenergebnisse geeignet dar.

- a) Selectionsort b) Bubblesort c) Shakersort d) Mergesort

Aufgabe 7.4 (30 Punkte)

Schreiben Sie `MergeSort.java` mit den Methoden

```
public static void sort_iterativ(int[] a) { // hinterher ist a sortiert
public static int[] sort_rekursiv(int[] a) { // liefert sortiertes Array zurueck
```

die ein Array `a` mit 2^n Elementen nach der Methode von Mergesort **iterativ** bzw. **rekursiv** sortieren. Verwenden Sie dazu die Methode `Merge.merge` aus der Klasse `/home/ainf/Vorlesung/Merge.java`

Hinweis: Um die Methoden der Klasse `Merge` aus der Vorlesung verwenden zu können brauchen sie diese nicht in Ihr aktuelles Arbeitsverzeichnis zu kopieren, da der Pfad zu den Klassen (`/home/ainf/Vorlesung/`) bereits in Ihrem CLASSPATH eingetragen ist.

Schreiben Sie eine Klasse `Aufg7_4.java`, die `int`-Arrays der Länge 2^n einliest und mit den beiden Methoden sortiert.

Aufgabe 7.5 (10 Punkte)

In den bisherigen Aufgaben wurden alle Tauschoperationen mit einer Hilfsvariablen als Zwischenstation durchgeführt.

Unter welchen Bedingungen kann man den Tausch zweier Variablen ohne Verwendung einer dritten Hilfsvariable abwickeln und wie läuft so ein Tausch ab? Ist die Variante mit den drei Variablen in Bezug auf Laufzeit und auf Speicherbedarf effizienter als die mit den zwei Variablen?

Aufgabe 7.6 (0 Punkte)

UNIX-Kommandos, die nach dem UNIX-Prompt eingegeben werden, werden zunächst von einem Programm namens *Shell* (Muschel) interpretiert, bevor sie ausgeführt werden. Wir verwenden die sogenannte *bash* (GNU Bourne-Again SHell). Diese verfügt über eigene Kommandos, mit denen zuvor eingegebene UNIX-Kommandos wiederholt und editiert werden können. Die *bash* ist so voreingestellt, daß sie das Editieren der Kommandozeilen insbesondere mit Hilfe der Pfeiltasten ermöglicht. Probieren Sie es aus:

Kommando	Erklärung
<code>history, h</code>	Listet die zuvor eingegebenen Kommandos.
<code>!34</code>	Wiederholt das 34. Kommando in der history.
<code>!!</code>	Wiederholt das letzte Kommando.
<code>!his</code>	Wiederholt das letzte Kommando, das mit <code>his</code> beginnt.
<code>!-2</code>	Wiederholt das vorletzte Kommando.
<code>vi Auf Tab</code>	Ergänzt den Dateinamen <code>Auf . . .</code> eindeutig, falls möglich, wenn nicht gibt ein doppeltes Tab Alternativen an.
<code>Ctrl a</code>	Bewegt den Cursor an den Anfang der Kommandozeile.
<code>Ctrl e</code>	Bewegt den Cursor an das Ende der Kommandozeile.
<code>Ctrl u</code>	Löscht die Kommandozeile bis zur aktuellen Cursorposition.
<code>↑, ↓</code>	Holen die vorherigen Kommandos auf die Kommandozeile.
<code>←, →</code>	Bewegen den Cursor in der Kommandozeile.

Beachten Sie die Möglichkeit, die *Wildcards* $*^1$ und $?^2$ von der Shell interpretieren zu lassen:

Kommando	Erklärung
<code>rm *.class</code>	Löscht alle Dateien, deren Name auf <code>.class</code> endet.
<code>more ?.txt</code>	Zeigt alle Dateien, deren Name vorne ein Zeichen hat und auf <code>.txt</code> endet.
<code>more Aufg?.java</code>	Zeigt Dateien an, wie z.B.: <code>Aufg1.java</code> , <code>Aufg_.java</code> , <code>Aufg. .java</code> .
<code>ls *.*</code>	Listet alle Dateien und Verzeichnisse, deren Name einen <code>.</code> enthält.
<code>rm -i *</code>	Löscht alle Dateien nach Abfrage.
<code>vi Aufg*.java</code>	Lädt alle Dateien in den <code>vi</code> , deren Name mit <code>Aufg</code> beginnt und auf <code>.java</code> endet.

¹* bedeutet: beliebig viele Zeichen, insbesondere auch keins.

²? bedeutet: genau ein Zeichen.