

Norman May

B6, 29, Raum C0.05

68131 Mannheim

Telefon: (0621) 181-2517

Email: norman@pi3.informatik.uni-mannheim.de

Matthias Brantner

B6, 29, Raum C0.05

68131 Mannheim

Telefon: (0621) 181-2517

Email: msb@pi3.informatik.uni-mannheim.de

Algorithmen und Datenstrukturen  
Wintersemester 2004/054. Übungsblatt  
18. November 2004

Aufgabe 1

5 Punkte

Sei  $a$  eine Konstante mit  $a \geq 1$ . Lösen Sie folgende Rekurrenz mit Hilfe der Iterationsmethode. Geben Sie die Lösung mittels der  $\Theta$ -Notation an.

$$T(n) = T(n - a) + T(a) + n$$

Hinweis: Die Iterationsmethode wird in zwei Schritten durchgeführt. Zunächst wird für die Rekurrenz durch wiederholtes Expandieren eine Summenformel bestimmt. Anschließend wird für diese eine asymptotische Abschätzung ermittelt. Gehen Sie davon aus, daß  $T(c) = \Theta(1)$  für eine kleine Konstante  $c > 0$ .

Aufgabe 2

5 Punkte

Lösen Sie folgende Rekurrenz mittels der Substitutionsmethode. Zeigen Sie, daß für

$$T(n) = 3T(n/3 + 5) + n/2$$

gilt:  $T(n) = \Theta(n \lg n)$ .

Hinweis: Bei der Substitutionsmethode werden asymptotische Schranken für die gegebene Rekurrenz geraten. Deren Korrektheit wird mittels vollständiger Induktion nachgewiesen. Gehen Sie davon aus, daß  $T(1) = \Theta(1)$ .

Aufgabe 3

10 Punkte

Lösen ( $\Theta$ -Notation) Sie folgende Rekurrenzen.

Aufgabe 3 a)

2 Punkte

$$T(n) = 2T(n/2) + n^3$$

Aufgabe 3 b)

2 Punkte

$$T(n) = T(n - 1) + \lg n$$

Aufgabe 3 c)

2 Punkte

$$T(n) = 16T(n/4) + n^2$$

Aufgabe 3 d)

2 Punkte

$$T(n) = 7T(n/2) + n^2$$

Aufgabe 3 e)

2 Punkte

$$T(n) = T(n - 1) + \frac{1}{n}$$

---

Aufgabe 4

14 Punkte

---

Bei Datenbeständen, die größer sind als der Hauptspeicher, wird das externe Sortieren angewendet. Die Sortierung des gesamten Datenbestands erfolgt in 2 Schritten:

1. *Partitionierung* des gesamten Daten und Sortierung je einer Partition im Hauptspeicher. Die sortierten Partitionen (*Runs*) werden i.a. auf den Hintergrundspeicher geschrieben.
2. *Mischen* der Runs zum sortierten Ergebnis.

Implementieren Sie externes Sortieren, wobei ein Parameter die Größe des Hauptspeichers in Anzahl der Elemente angibt.

Aufgabe 4 a)

6 Punkte

Implementieren Sie Heapsort. Der Sortieralgorithmus bekommt ein Array bzw. eine Liste als Eingabe und liefert die sortierte Datenstruktur als Ergebnis.

Aufgabe 4 b)

2 Punkte

Gegeben Sei folgender Sortieralgorithmus:

AnotherSort(array A)

```
for (j = A.length; j > 2; --j)
  for (i = 1; i < j; ++i)
    if (A[i] > A[i+1])
      swap(A[i], A[i+1])
```

Leiten Sie die Komplexität des Algorithmus in  $\Theta$ -Notation her. Vergleichen Sie die Komplexität mit der des Heapsort-Algorithmus. Welche Algorithmus ist besser?

Aufgabe 4 c)

6 Punkte

Implementieren Sie die Partitionierung und den Mischvorgang. Gehen Sie davon aus, dass nur ein Mischvorgang nötig ist. Orientieren Sie sich an folgender Java-Schnittstelle:

```
public class ExternalSort {  
  
    /* constructor */  
    public ExternalSort(int memSize) { ... }  
  
    /**  
     * partition the input and sort each partition  
     * @param filename the input file to read from  
     */  
    public void partition (String filename ) { ... }  
  
    /**  
     * merge the partitions into the sorted result  
     * @param filename the output file for the sorted result  
     */  
    public void merge(String filename ) { ... }  
    ...  
}
```

Hinweis: Lesen von Daten aus einer Datei kann z.B. mit folgendem Codefragment durchgeführt werden. Sie können annehmen, dass nur atomare Werte eines vorher bekannten Typs (z.B. int) gelesen und geschrieben werden.

```
DataInputStream input =  
    new DataInputStream(new FileInputStream(filename));  
try {  
    while (true) {  
        int val = input.readInt();  
    }  
} catch (EOFException e) {  
} finally {  
    input.close();  
}
```

Schreiben von Daten erfolgt analog (ohne Ausnahmebehandlung):

```
DataOutputStream output =  
    new DataOutputStream(new FileOutputStream(filename));  
for (int i = 0; i < size; ++i) {  
    int val = 5;  
    output.writeInt(val);  
}  
output.close();
```

Wenn Sie mit der Ein- und Ausgabe in Dateien nicht zurecht kommen, verwenden Sie Arrays oder Listen sowohl für die Ein- und Ausgabe als auch für die sortierten Runs.

Wenn Sie keinen der Sortieralgorithmen implementiert haben, können sie z.B. in Java `java.util.Collections.sort(List)` oder `java.util.Arrays.sort(Array)` verwenden oder die unsortierten Runs mischen (aber unter der Annahme, daß sie sortiert sind).