Lehrstuhl für Informatik I
Algorithmen, Komplexität und wissensbasierte Systeme
Universität Würzburg

Würzburg, den 24. Oktober 2017

Prof. Dr. Alexander Wolff
Fabian Lipp, M. Sc.
Andre Löffler, M. Sc.
Johannes Blum, B. Sc.

1. Übungsblatt zur Vorlesung Algorithmen und Datenstrukturen (Winter 2017/18)

PABS **Aufgabe 1 – MaxFinder**

In dieser Aufgabe werden Sie in Java einen Divide&Conquer-Algorithmus implementieren, der in einem Eingabefeld das größte Element nach vorne verschiebt.

Erstellen Sie die Klasse `public class MaxFinder` im Paket `algorithms`.

a) Implementieren Sie zunächst die folgende Methode:

- `private static void conquerMax(int[] a, int ℓ, int m, int r)`.

Werfen Sie eine `IllegalArgumentException`, wenn $0 \leq \ell \leq m < r < a.\texttt{length}$ nicht gilt. Nach Aufruf der Methode soll a die gleichen Zahlen wie vor dem Aufruf enthalten, wobei in $a[\ell]$ das Maximum von $\{a[\ell], a[m+1]\}$ stehen soll. Die Laufzeit soll in $O(1)$ sein. **2 Punkte**

b) Implementieren Sie weiter folgende Methode:

- `public static void moveMaxToFront(int[] a, int ℓ, int r)`

Werfen Sie ggf. eine `IllegalArgumentException`. Nach Aufruf von `moveMaxToFront(a,ℓ,r)` soll a die gleichen Zahlen wie vor dem Aufruf enthalten, wobei die größte Zahl von $a[\ell..r]$ an der Stelle $a[\ell]$ stehen soll.

Ihre Implementierung soll nach dem Divide&Conquer-Prinzip aufgebaut sein: Wenn $r > \ell$, teilen Sie das Problem in zwei Teilprobleme $a[\ell..m]$ und $a[m+1..r]$ für ein geeignetes m und lösen Sie diese rekursiv. Anschließend lösen Sie das Problem für $a[\ell..r]$ mithilfe der Lösungen für die Teilprobleme. Benutzen Sie hierfür Ihre Methode `conquerMax`. **3 Punkte**

Aufgabe 2 – O-Notation

Sei $f\colon \mathbb{N} \to \mathbb{R}$ eine Funktion. Beweisen oder widerlegen Sie folgende Behauptungen. Arbeiten Sie mit der Definition aus der Vorlesung, nicht mit Grenzwertbetrachtungen.

a) Für $f\colon n \mapsto (n+1)^5$ gilt $f \in \Omega(n^6)$. **2 Punkte**

b) Für $f\colon n \mapsto \log_2(n^2)$ gilt $f \in \Theta(\log_{10} n)$. **3 Punkte**

Aufgabe 3 – Vereinigung

Geben Sie in gut kommentiertem Pseudocode einen Algorithmus an, der als Eingabe zwei aufsteigend sortierte Felder A und B erhält. Die Ausgabe soll ein Feld C sein, das jede Zahl aus A und B genau einmal enthält. Die Laufzeit soll $O(n)$ sein, wobei $n = A.length + B.length$. **4 Punkte**

Tipp: In A und B können Zahlen mehrfach vorkommen! Anfangs ist also die Länge von C unbekannt, verwenden Sie ein Hilfsfeld der Länge n.

Aufgabe 4 – EasySort

Sei A ein Array der Länge n und sei folgender Algorithmus in Pseudocode gegeben.

```
EasySort(int[ ] A)
for i = 1 to A.length − 1 do
    min = A[i]
    indexMin = i
    for j = i + 1 to A.length do
        if A[j] < min then
            min = A[j]
            indexMin = j
    A[indexMin] = A[i]
    A[i] = min
```

a) Beweisen Sie, dass EasySort das Eingabefeld A aufsteigend sortiert.
Verwenden Sie dazu folgende Schleifeninvariante: **3 Punkte**

Bei der i-ten Ausführung des äußeren for-Schleifenkopfes gilt, dass

(i) *die $(i-1)$ kleinsten Elemente von A aufsteigend geordnet in den ersten $(i-1)$ Zellen des Feldes stehen und*

(ii) *in den übrigen Zellen die $(n-i+1)$ größten Zahlen von A stehen.*

b) Wie viele Vergleiche benötigt EasySort in Abhängigkeit von n? **2 Punkte**

c) Welchen Sortieralgorithmus würden Sie vorziehen, InsertionSort oder EasySort? Begründen Sie Ihre Antwort. **1 Punkt**

Bitte werfen Sie Ihre Lösungen bis **Dienstag, 7. November 2017, 10:10 Uhr** in den Vorlesungs-Briefkasten im Informatik-Gebäude. Geben Sie stets die Namen und Übungsgruppen aller BearbeiterInnen sowie die Übungsgruppe, in der das Blatt zurückgegeben werden soll, an.

Die Lösungen zu den mit PABS gekennzeichneten Aufgaben, geben Sie bitte nur über das PABS-System ab. Geben Sie Ihre Namen hier als Kommentare in den Quelltextdateien an.