TU Ilmenau, Fakultät IA
FG Telematik/Rechnernetze
Prof. Dr.-Ing. G. Schäfer
http://www.tu-ilmenau.de/telematik/pua

TECHNISCHE UNIVERSITÄT
ILMENAU

Programmierung und Algorithmen WS 23/24
Übungsblatt 7

Die Lösungen der Aufgaben sind bis zum 10.12.23, 23:59 Uhr abzugeben.
Die Besprechung der Aufgaben erfolgt in KW 50.

Aufgabe 1 ($\mathcal{O}$-Notation) — 6 Punkte

Entscheiden Sie für jede der folgenden Aussagen, ob sie zutrifft oder nicht. Begründen Sie Ihre Antwort bitte kurz (z.B. durch Anwendung der Definitionen). Ohne Begründung kann Ihre Antwort nicht gewertet werden!

(a) $2^{42} = \mathcal{O}(13)$

(b) $\frac{1}{2}n \cdot \log n = \Omega(n)$

(c) $10 \cdot n^7 + 9 \cdot n^4 + 1337 \cdot n = \Theta(n^7)$

(d) $\frac{n^2}{n \cdot \log n} = \mathcal{O}(n)$

(e) $2^{\log_2 2n} = \Theta(n)$

(f) $\log n = \Omega(n^{\frac{1}{4}})$

Aufgabe 2 (Vollständige Induktion) — 2 + 2 + 3 Punkte

Beweisen Sie per vollständiger Induktion die folgenden Behauptungen.

(a) Für jede natürliche Zahl $n \geq 1$ gilt: $\sum_{i=0}^{n-1} 2^i = 2^n - 1$

(b) Sei F_i die i-te Fibonacci-Zahl nach der Definition aus der Vorlesung ($F_0 = F_1 = 1$). Dann gilt für alle $n \in \mathbb{N}_0$: $\sum_{i=0}^{n} F_i = F_{n+2} - 1$

(c) Für alle $n \in \mathbb{N}$ gilt: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Aufgabe 3 (Korrektheit applikativer Algorithmen) **1 + 4 Punkte**

`Gegeben` sei der folgende applikative Algorithmus, mit den Eingaben `x, y` $\in \mathbb{Z}$ und der Ausgabe `f(x,y)` $\in \mathbb{Q}$:

```
f(x,y) = if (y = 0) then 1
         else if (y < 0) then (1.0)/f(x,-y)
         else if (y > 0) then x · f(x,y-1) fi fi fi
```

(a) Welche Funktion wird durch den Algorithmus berechnet?

(b) Beweisen Sie Ihre Behauptung aus (a).

Hinweis: Für einen Teil des Beweises ist es sinnvoll, vollständige Induktion zu verwenden.

Aufgabe 4 (Aufwandsschätzung in imperativen Programmen) **1 + 4 + 1 Punkte**

Gegeben seien zwei Felder `a` und `b` aus ganzen Zahlen, welche bereits *aufsteigend sortiert* sind. Ferner wird angenommen, dass *insgesamt* keine Zahl aus `a` und `b` doppelt vorkommt. Gesucht ist ein Algorithmus, welcher diese beiden Felder zu einem einzigen *aufsteigend sortierten Feld* zusammenfügt.

Die folgende Java-Methode ist eine Möglichkeit, dieses Problem zu lösen:

```
1  public static int[] dumbMerge(int [] a, int[] b) {
2    int x = a.length;
3    int s = a.length;
4    int last = Integer.MIN_VALUE;
5    int[] c = new int[a.length + b.length];
6
7    for(int i = 0; i < c.length; ++i) {
8      int min = Integer.MAX_VALUE;
9      for(int j = 0; j < c.length; ++j) {
10       if(j < s) { x = a[j]; } else { x = b[j - s]; }
11       if((x < min) && (last < x)) { min = x; }
12     }
13     c[i] = min;
14     last = min;
15   }
16   return c;
17 }
```

(a) Schätzen Sie den von der Methode `dumbMerge` verursachten Aufwand an Rechenoperationen ab. Verwenden Sie dazu asymptotische Notation und gehen Sie davon aus, dass die Felder `a` und `b` jeweils die Länge l_a bzw. l_b haben.

(b) Schreiben Sie eine alternative Java-Methode, welche die Aufgabe mit einem Aufwand von $O(l_a + l_b)$ löst.

(c) Begründen Sie kurz, warum keine Lösung mit asymptotisch geringerem Aufwand existieren kann.