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

TECHNISCHE UNIVERSITÄT
ILMENAU

# Programmierung und Algorithmen WS 25/26

# Übungsblatt 7

---

**Die Lösungen der Aufgaben sind bis zum 14.12.25, 23:59 Uhr abzugeben.**

**Die Besprechung der Aufgaben erfolgt in KW 51.**

---

**Aufgabe 1** ($\Omega$ und $\Theta$-Notation) **4 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 und Angabe der entsprechenden Konstanten). Ohne Begründung kann Ihre Antwort nicht gewertet werden! Nehmen Sie (falls nötig) an, dass Logarithmus-Funktionen die Basis 2 haben.

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

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

(c) $2^{\log 2n} = \Theta(n)$

(d) $\log n = \Omega\left(n^{\frac{1}{4}}\right)$

**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.

**Bitte wenden!**

**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 Aufwand $\mathcal{O}(l_a + l_b)$ löst.

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

**Aufgabe 5** (Post'sches Korrespondenzproblem in Java) **1 + 2 + 4 Punkte**

Das Post'sche Korrespondenzproblem (siehe auch Kapitel 6, Folie 17) ist im Allgemeinen nicht entscheidbar. Schränkt man die Suche allerdings auf Korrespondenzen mit einer maximalen Länge $k$ ein, wird das Problem lösbar.

Entwerfen Sie ein Java-Programm, welches das Post'sche Korrespondenzproblem mit der Beschränkung $k = 16$ löst. Die Eingaben $\alpha, \beta$ des Problems sollen dabei durch zwei String-Arrays (`String[]`) repräsentiert werden (welche Sie der Einfachheit halber z. B. fest in Ihr Programm kodieren dürfen).

(a) Ihr Programm soll zunächst überprüfen, ob $\alpha$ und $\beta$ die gleiche Länge haben.

(b) Sei $A = \{0, 1, ...9, a, b, ..., z\}$. Es soll überprüft werden, ob alle Worte aus $\alpha$ und $\beta$ über dem Alphabet $A$ (Dezimalziffern und Kleinbuchstaben) gebildet wurden und mindestens ein Zeichen lang sind.

(c) Sind die ersten beiden Bedingungen erfüllt, so soll nach einer Korrespondenz gesucht werden. Sofern eine solche existiert, soll sowohl die Korrespondenz (Folge von Indizes aus $\alpha$ bzw. $\beta$) als auch das resultierende zusammengesetzte Wort ausgegeben werden.

**Hinweis:** Zur Lösung des Problems eignet sich der rekursive Ansatz des *Backtrackings*.