Prof. Dr. Ir. Joost-Pieter Katoen

Sebastian Junges, Benjamin Kaminski, David Korzeniewski, Tim Quatmann

Übung 2

Hinweise:

- Die Lösungen müssen bis **Donnerstag, den 26. April um 16:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Die Übungsblätter sollen vorzugsweise in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung bearbeitet werden. Ab dem nächsten Übungsblatt **müssen** die Übungsblätter in Dreiergruppen abgegeben werden.
- Drucken Sie ggf. digital angefertigte Lösungen aus. Abgaben z.B. per Email sind nicht zulässig.
- Namen und Matrikelnummer sowie die **Nummer der Übungsgruppe** sind auf jedes Blatt der Abgabe zu schreiben. Abgaben, die aus mehreren Blättern bestehen, **müssen geheftet bzw. getackert** werden! Die **Gruppennummer muss sich auf der ersten Seite oben links** befinden.
- **Bei Nichtbeachten der obigen Hinweise müssen Sie mit erheblichen Punktabzügen rechnen!**

Aufgabe 1 (Funktionen sortieren): (6 + 14 = 20 Punkte)

Wir definieren die Quasiordnung $\sqsubseteq$ auf Funktionen als

$$f \sqsubseteq g \qquad \text{genau dann wenn} \qquad f \in \mathcal{O}(g)\ .$$

a) Beweisen Sie, dass $\sqsubseteq$ eine Quasiordnung ist, das heißt dass $\sqsubseteq$ reflexiv und transitiv ist.

b) Sortieren Sie die Funktionen

$$n!, \quad 2^{9000}, \quad 4, \quad 0, \quad \log(n), \quad \sum_{i=0}^{n} \frac{14i^2}{1+i}, \quad \frac{n^3}{2},$$
$$n^n, \quad n^3, \quad n^2, \quad n \cdot \log(n), \quad 2^n, \quad n^2 \cdot \log(n), \quad n \cdot \sqrt{n}$$

in aufsteigender Reihenfolge bezüglich der Quasiordnung $\sqsubseteq$. Schreiben Sie also

$$f \sqsubseteq g \sqsubseteq h \sqsubseteq \ldots \qquad \text{falls} \qquad f \in \mathcal{O}(g) \quad \text{und} \quad g \in \mathcal{O}(h) \quad \text{und} \quad \ldots\ .$$

Aufgabe 2 ($\mathcal{O}$-Notation konkret): (5 · 5 = 25 Punkte)

Beweisen oder widerlegen Sie folgenden Aussagen:

a) $\frac{1}{4}n^3 - 7n + 17 \in \mathcal{O}(n^3)$

b) $n^4 \in \mathcal{O}(2n^4 + 3n^2 + 42)$

c) $\log(n) \in \mathcal{O}(n)$

d) $\forall \epsilon > 0\colon\ \log(n) \in \mathcal{O}\left(n^\epsilon\right)$

e) $a^n \in \Theta(b^n)$, für zwei beliebige Konstanten $a, b > 1$

Aufgabe 3 ($\mathcal{O}$-Notation abstrakt): (8 + 8 = 16 Punkte)

a) Zeigen oder widerlegen Sie:

$$o(g(n)) \cap \Theta(g(n)) = \emptyset$$

b) Zeigen oder widerlegen Sie:

$$f(n) \in \Omega(g(n)) \quad \text{und} \quad f(n) \in \mathcal{O}(h(n)) \qquad \text{impliziert} \qquad g \in \Theta(h(n))$$

Aufgabe 4 (Laufzeitanalyse): (5 + 5 + 15 + 10 + 4 = 39 Punkte)

Gegeben sei ein Algorithmus, der für ein Array von Booleans überprüft, ob alle Einträge wahr sind:

```
int allTrue(bool [] E) {
    if (E.length < 1) {
        return -1;
    }
    int m = E.length;
    int i = 0;
    while (i < E.length) {
        if (E[i] == true) {
            m = m - 1;
            if (m == 0) {
                return 1;
            }
        }
        i = i + 1;
    }
    return 0;
}
```

Bei Betrachtung der Laufzeit wird angenommen, dass Vergleiche (z.B. x < y oder b == 0) jeweils eine Zeiteinheit benötigen. Die Laufzeit aller weiteren Operationen wird vernachlässigt.
Sei n die Länge des Arrays E.

a) Bestimmen Sie in Abhängigkeit von n die Best-case Laufzeit $B(n)$.

b) Bestimmen Sie in Abhängigkeit von n die Worst-case Laufzeit $W(n)$.

c) Bestimmen Sie in Abhängigkeit von n die Average-case Laufzeit $A(n)$. Hierzu nehmen wir eine uniforme Verteilung der möglichen Eingaben D_n an, d.h. jede beliebige Eingabe $E \in D_n$ tritt mit Wahrscheinlichkeit $1/|D_n|$ auf.

Hinweise:

- Ihre Lösung darf Summenzeichen $\sum$ enthalten.

d) Geben Sie einen äquivalenten Algorithmus an, dessen Average-case Laufzeit ab einer gewissen Eingabelänge kleiner ist. Hierzu nehmen wir wieder eine uniforme Verteilung der möglichen Eingaben an. Begründen Sie Ihre Antwort kurz. Sie müssen nicht die Average-case Laufzeit ihres Algorithmus berechnen.

Hinweise:

- Wir nennen zwei Algorithmen äquivalent, wenn sie mit der gleichen Eingabe die gleiche Ausgabe produzieren.

e) Gibt es einen äquivalenten Algorithmus, dessen Worst-case Laufzeit in $o(n)$ liegt? Begründen Sie Ihre Antwort.