Algorithmen & Datenstrukturen 1

Sommersemester 2022

Prof. Dr. Martin Hoefer
Tim Koglin, Marco Schmalhofer

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Institut für Informatik
Algorithmen und Komplexität

Übungsblatt 2

Ausgabe: 26.04.2022
Abgabe: 03.05.2022, **08:00**

Aufgabe 2.1 *Laufzeitanalyse* (3 × 3 Punkte)

Wir betrachten den folgenden Algorithmus für eine ganzzahlige Eingabe $n \geq 0$:

```
x = 0;
for (i = 1; i <= n; i++) {
    for (j = 1; j <= f(i); j++) {
        x = x + 1;
    }
}
```

Bestimmen Sie für jede der folgenden Funktionen $f(i)$ den Wert der Variable x nach Ende des Algorithmus exakt.

a) $f(i) = 2n - i$

b) $f(i) = c \cdot \sqrt{n}$ für ein $c \in \mathbb{N}$.

c) $f(i) = 2^{\log_4(n^2)}/(3^i)$. Hier dürfen Sie annehmen, dass $n = 3^k$ für ein $k \in \mathbb{N}$.

Aufgabe 2.2 *Mastertheorem* (3 × 3 Punkte)

Verwenden Sie das Mastertheorem, um das Wachstum folgender rekursiver Funktionen $T(n)$ asymptotisch exakt zu bestimmen. Es kann angenommen werden, dass $T(1)$ konstant und n eine Potenz von b ist.

a) $T(n) = 8 \cdot T(\frac{n}{2}) + n^3$

b) $T(n) = 125 \cdot T(\frac{n}{5}) + \sqrt{4n}\left(\frac{n}{12} + n\sqrt{n}\right) + 2^{101}$

c) $T(n) = 3 \cdot T\left(\frac{n}{3}\right) + \left(3^{\frac{11}{5}\log_9 n}\right)\log_2 n$

Aufgabe 2.3 *Pseudocode* (3 + 4 + 4 Punkte)

Gegeben ist ein Array $A = (A[1], ..., A[n])$ der Länge $n \geq 2$ von ganzen Zahlen.

a) Beschreiben Sie kurz, welche Eigenschaft das Array A haben muss, damit der folgende Pseudocode `"Ja"` ausgibt. Analysieren Sie außerdem seine Laufzeit asymptotisch exakt.

```
int x = A[1];
int y = A[2];
for (int i = 3; i <= n; i += 2) {
    x = x * A[i];
}
for (int i = 4; i <= n; i += 2) {
    y = y * A[i];
}
if (y >= x) print "Ja";
else print "Nein";
```

b) Bestimmen Sie die asymptotische *best-* und *worst-case* Laufzeit des folgenden Pseudocodes. Geben Sie jeweils ein konkretes Eingabearray A der Länge $n \geq 2$ an, durch welches das entsprechende Verhalten erzeugt wird.

```
for (int i = 1; i <= n; i++) {
    for (int j = i + 1; j <= n; j++) {
        if (A[i]*A[j] == 2022){
            print "Ja";
            EXIT; // beende das gesamte Programm sofort
        }
    }
}
print "Nein";
```

c) Wir bezeichnen ein Array $(A[i], ..., A[i + k - 1])$ mit $i, k \in \{1, ..., n\}$ und $i + k - 1 \leq n$ als ein *Teilarray* von A mit Länge k.

Beschreiben Sie einen Algorithmus in Pseudocode, der ausgibt, ob es *genau ein* Teilarray von A gibt, dessen Elemente sich auf die Zahl 2022 aufaddieren. Geben Sie eine kurze Begründung für die Korrektheit Ihres Algorithmus und analysieren Sie seine *worst-case* Laufzeit.

Hinweis: Die Laufzeit Ihres Algorithmus muss nicht optimal sein.

Aufgabe 2.4 *Das Ratespiel - Omas Revanche* (2 + 2 + 3 + 4 Punkte)

Noch immer ist der kleine Theo bei seinen Großeltern zu Besuch und der Regen lässt nicht nach. Dank seiner cleveren Strategie hat er beim Ratespiel immerhin fünf Bonbons von seiner Oma als Preis bekommen. "Doppelt oder nichts", sagt seine Oma und fordert ihn zu einem weiteren Spiel heraus: Opa soll sich zwei ganze Zahlen $b > 0$ und $k \geq 0$ ausdenken. Wenn sie es schafft, die Potenz b^{2^k} mit *strikt* weniger Multiplikationen zu berechnen als Theo, verliert er alle Bonbons, andernfalls bekommt er fünf weitere Bonbons dazu. Theo glaubt, dass es keine Alternative dazu gibt, die gesuchte Potenz naiv durch 2^k-fache Multiplikation von b mit sich selbst zu berechnen, d.h. $b^{2^k} = b \cdot b \cdot ... \cdot b$, und willigt deshalb siegesgewiss ein.

a) Geben Sie an, wie Theos Oma jede Potenz b^{2^k} mit genau k Multiplikationen berechnen kann.

Theo ist beeindruckt von Omas Trick (und trauert daher auch nicht um seine eben verlorenen Bonbons). Zusammen machen sie sich Gedanken, wie dieses Verfahren für beliebige Potenzen b^n verallgemeinert werden kann.

b) Geben Sie zunächst an, wie mit dem Verfahren von Theos Oma jede Potenz b^{2^k+1} mit genau $k+1$ Multiplikationen berechnet werden kann.

c) Beschreiben Sie nun auf Basis der Methoden in a) und b) ein *rekursives* Verfahren `hoch(b,n)` in Pseudocode, dass jede beliebige Potenz b^n für $b \in \mathbb{N}_{>0}$ und $n \in \mathbb{N}$ berechnet. Das Verfahren soll für feste $b \in \mathbb{N}_{>0}$ eine *worst-case* Laufzeit von $\mathcal{O}(\log n)$ haben.

d) Zeigen Sie, dass das Verfahren die in c) gegebene Laufzeitschranke einhält.

Hinweis: Sie können zunächst eine Rekursionsgleichung $T(n) = ...$ aufstellen und dann durch vollständige Induktion eine obere Schranke auf $T(n)$ zeigen.