Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. Marius Kloft
Bux, Grußien, Schäfer, Tschorsch, Völlinger

Berlin, den 20.04.2016

Übungen zur Vorlesung
Algorithmen und Datenstrukturen

Übungsblatt 1

Abgabe: Mittwoch den 4.5.2016 bis 10:55 Uhr vor der Vorlesung im Hörsaal oder bis 10:45 Uhr im Briefkasten (RUD 25, Raum 3.321). Die Übungsblätter sind in Gruppen von zwei (in Ausnahmefällen auch drei) Personen zu bearbeiten. Sie können auf diesem Übungsblatt bis zu 50 Punkte erhalten. **Die Lösungen sind auf nach Aufgaben getrennten Blättern abzugeben. Heften Sie bitte die zu einer Aufgabe gehörenden Blätter vor der Abgabe zusammen.** Vermerken Sie auf allen Abgaben Ihre Namen, Ihre Matrikelnummern, den Namen Ihrer Goya-Gruppe und an welchem Übungstermin (Wochentag, Uhrzeit, Dozent) Sie die korrigierten Abgaben abholen möchten. Beachten Sie auch die aktuellen Hinweise auf der Übungswebsite unter https://hu.berlin/algodat16.

Konventionen:

- Für ein Array A ist $|A|$ die Länge von A, also die Anzahl der Elemente in A. Die Indizierung aller Arrays auf diesem Blatt beginnt bei 1 (und endet also bei $|A|$). Bitte beginnen Sie die Indizierung der Arrays in Ihren Lösungen auch bei 1.
- Mit der Aufforderung „Analysieren Sie die Laufzeit“ ist gemeint, dass Sie eine möglichst gute obere Schranke der Zeitkomplexität angeben sollen und diese begründen sollen.
- Mit log wird der Logarithmus $\log_2$ zur Basis 2 bezeichnet.
- Die Menge der natürlichen Zahlen $\mathbb{N}$ enthält die Zahl 0.
- $\mathbb{R}_{\geq 0}$ bezeichnet die Menge der nichtnegativen reellen Zahlen.

Hinweis: Zusammenfassende Informationen zur $\mathcal{O}$-Notation finden Sie auch im $\mathcal{O}$-Tutorial auf der Übungswebsite.

Aufgabe 1 (Funktionen einordnen) — $8 \cdot 2 = 16$ Punkte

Sie dürfen voraussetzen, dass $\lim_{n\to\infty} \ln n = \infty$ und $(\ln n)' = \frac{1}{n}$ ist. Beweisen Sie die folgenden Aussagen und beachten Sie, dass es Punkte für den Rechenweg gibt:

a) $\left(e^{\pi^{17}}\right)! \in \mathcal{O}(6 \log n)$

b) $n \log n \in \omega(4 \log^2 n)$

c) $8^{2n} \notin \mathcal{O}(8^n)$

d) $2n + \sqrt{n} \in \mathcal{O}(n)$

e) $2n\sqrt{n} \in \mathcal{O}\left(\frac{n^{\frac{5}{3}}}{7}\right)$

f) $\frac{100}{n} \log^2 n \in \mathcal{O}(\log^3 n)$

g) $\sqrt{n} \in \mathcal{O}(n \log n)$

h) $\ln \ln n \in o(\sqrt{\ln n})$

Aufgabe 2 (Eigenschaften der $\mathcal{O}$-Notation) **$5 \cdot 2 = 10$ Punkte**

Seien f und g Funktionen, die von $\mathbb{R}_{\geq 0}$ nach $\mathbb{R}_{\geq 0}$ abbilden. Beweisen Sie die folgenden Aussagen:

a) $\mathcal{O}(k \cdot f) \subseteq \mathcal{O}(f)$ mit $k \in \mathbb{N}$.

b) $f \notin o(f)$.

c) Wenn $f \in \mathcal{O}(g)$ und $g \in \mathcal{O}(h)$, dann gilt $f \in \mathcal{O}(h)$.

d) Sei $p\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ mit $p(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0$ ein Polynom in n vom Grad d mit $a_i \in \mathbb{R}$ für $0 \leq i \leq d$ und $a_d > 0$, dann gilt $p(n) \in \mathcal{O}(n^d)$.

e) Seien $a, b \in \mathbb{N}$ positive Konstanten, dann gilt $a^{x+b} \in \Theta(a^x)$.

Aufgabe 3 (Pseudocodeanalyse) **$(2 + 2 + 3) + (2 + 2 + 2) = 13$ Punkte**

1. Betrachten Sie den Algorithmus **bar**, dem als Eingabe ein Array A aus n natürlichen Zahlen übergeben wird und der als Ausgabe die Zahl x liefert.

Algorithmus bar

Input: Array A der Länge $|A| = n$ mit $n \geq 2$
Output: Zahl x

```
x := 0;
for i := 1 to n do
  for j := i + 1 to n do
    if x < |A[i] − A[j]| then
      x := |A[i] − A[j]|;
    end if
  end for
end for
return x;
```

a) Was berechnet Algorithmus **bar**?

b) Analysieren Sie die Laufzeit des Algorithmus in Abhängigkeit von n.

c) Entwerfen Sie einen bezüglich der Laufzeit effizienteren Algorithmus für das Berechnungsproblem. Notieren Sie Ihren Algorithmus als Pseudocode und analysieren Sie dessen Laufzeit.
Hinweis: Die Punktevergabe erfolgt gestaffelt nach Effizienz *bezüglich der Laufzeit.* Sie dürfen unbegrenzt zusätzlichen Speicherplatz verwenden.

2. Betrachten Sie den Algorithmus **foo**, der als Eingabe eine natürliche Zahl n erhält.

Algorithmus foo

Input: Zahl n mit $n \geq 1$

```
if n = 1 then
  return 2;
end if
return 2n+foo(n − 1);
```

a) Was berechnet der Algorithmus **foo**?

b) Beweisen Sie die Korrektheit des Algorithmus **foo**.
Hinweis: Eine geeignete Beweismethode ist die vollständige Induktion.

c) Analysieren Sie die Laufzeit des Algorithmus **foo** in Abhängigkeit von n.

Aufgabe 4 (Algorithmenentwurf) **(3 + 2 + 2) + 4 = 11 Punkte**

1. Gegeben sei ein Array A der Länge $|A| = n$ mit $1 \leq A[i] \leq n$ für alle $1 \leq i \leq n$ (d.h. bei den Elementen des Arrays handelt es sich um Zahlen zwischen 1 und n).

   a) Entwerfen Sie einen Algorithmus, der als Eingabe das Array A erhält und ausgibt, ob in diesem Array Duplikate enthalten sind. Notieren Sie Ihren Algorithmus als Pseudocode.
   Hinweis: Die Punktevergabe erfolgt gestaffelt nach *Effizienz bezüglich der Laufzeit.* Sie dürfen unbegrenzt zusätzlichen Speicherplatz verwenden.

   b) Analysieren Sie die Laufzeit Ihres in Pseudocode vorgestellten Algorithmus in Abhängigkeit von n.

   c) Begründen Sie informell, warum das Problem nicht mit einer sublinearen Anzahl von Elementzugriffen gelöst werden kann (d.h. warum eine Lösung nicht in $o(n)$ liegen kann).

2. Gegeben sei ein Array A der Länge $|A| = n - 1$, in dem jede Zahl zwischen 1 und n bis auf eine genau einmal vorkommt. Ein Beispiel für A mit $n = 5$ wäre also $A = [2, 4, 1, 5]$. Entwerfen Sie einen Algorithmus, der als Eingabe das Array A erhält und als Ausgabe die in A fehlende Zahl zurückgibt. Die Laufzeit des zu entwerfenden Algorithmus soll in $\mathcal{O}(n)$ liegen. Notieren Sie Ihren Algorithmus als Pseudocode.
   Hinweis: Die Punktevergabe erfolgt gestaffelt nach *Effizienz bezüglich des Speicherplatzbedarfs* zusätzlich eingeführter Datenstrukturen. Es existiert eine Lösung mit Speicherplatzbedarf in $\mathcal{O}(1)$.