Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 01

Algorithmen I – Sommersemester 2022

Abgabe im ILIAS bis 27.04.2022, 14:00 Uhr

Bitte beschrifte deine Abgabe gut sichtbar mit deinem Namen und deiner Matrikelnummer. Achte insbesondere bei handschriftlichen Abgaben auf Lesbarkeit und genügend Platz für Korrektur-Anmerkungen. Die Abgabe erfolgt über das Übungsmodul in der Gruppe Deines Tutoriums im ILIAS. Gib deine Ausarbeitungen im PDF-Format ab.

Aufgabe 1 - O-Kalkül (2 Punkte)

Ordne die folgenden Funktionen so an, dass $f \in O(g)$ genau dann, wenn f links von g eingeordnet ist. Du musst Deine Antwort nicht begründen.

$$\log(\sqrt[3]{n}), \quad 2^{\sqrt{n}}, \quad n!, \quad n^{\log\log(n)}, \quad \sqrt[3]{\log(n)}, \quad \sqrt{4^{\log(n)}}$$

Lösung 1

$$\sqrt[3]{\log n}, \qquad \log(\sqrt[3]{n}), \qquad \sqrt{4^{\log(n)}}, \qquad n^{\log\log n}, \qquad 2^{\sqrt{n}}, \qquad n!$$

Lösungsansatz (war nicht verlangt):

- $\sqrt[3]{\log(n)}$ wird links von $\log(\sqrt[3]{n})$ eingeordnet, denn

$$\log(\sqrt[3]{n}) = \log(n^{1/3}) = \frac{\log n}{3} \in \Theta(\log(n))$$

- $\sqrt{4^{\log(n)}}$ wird rechts von $\log(\sqrt[3]{n})$ eingeordnet, denn

$$\sqrt{4^{\log(n)}} = (4^{\log(n)})^{1/2} = 4^{1/2 \cdot \log(n)} = 2^{\log(n)}$$

- Da $O(\log n) \subset O(\sqrt{n})$ wird $2^{\sqrt{n}}$ rechts von $\sqrt{4^{\log(n)}} = 2^{\log n}$ eingeordnet

- $2^{\sqrt{n}}$ muss rechts von $n^{\log\log(n)}$ stehen, denn wir wissen, dass $2^{\sqrt{n}}$ bereits asymptotisch schneller wächst als $n^{\log(n)}$ und dass für $m \geq 1 : \log\log(m) < \log(m)$
- Wir wissen, dass $n!$ asymptotisch schneller wächst als $2^{\sqrt{n}}$, also muss $n!$ ganz rechts eingeordnet werden

Aufgabe 2 - Landau-Notation (4 Punkte)

Berechne n_0 und c aus der formalen Definition des jeweiligen Landau-Symbols, um zu zeigen, dass

1. $2^{100}n \in O(n^2)$ (1 Punkt)
2. $2^{100}n \in O(2^n)$ (1 Punkt)
3. $\sqrt[3]{n^2} \in \Omega(\log^8(n))$ (2 Punkte)

Lösung 2

1. $n_0 = 2^{100}, c = 1$

   Zeige: $\forall n > n_0 : 2^{100}n \leq n^2$.

   Es gilt $n_0^2 = 2^{100}n_0$ und für alle $n > 2^{100}$: $n^2 - 2^{100}n > n^2 - n \cdot n = n^2 - n^2 = 0$.

2. $n_0 = 1, c = 2^{100}$

   Zeige: $\forall n > n_0 : 2^{100} \cdot n \leq 2^{100+n}$.

   Es gilt für alle $n > 1$, dass $2^{100}2^n - 2^{100}n = 2^{100}(2^n - n) > 0$, da $2^n > n$ für alle $n > 1$.

3. $n_0 = 1, c = \left(\frac{1}{12}\right)^8$

   Zeige: $\forall n > n_0 : n > n_0 : \sqrt[3]{n^2} \geq c \cdot \log^8 n$.

   Es gilt $\sqrt[3]{n^2} = n^{2/3} = n^{8/12} = (\sqrt[12]{n})^8$ und

   $$c \cdot \log^8(n) = \left(\frac{1}{12}\right)^8 \cdot \log^8(n) = \left(\frac{1}{12}\log(n)\right)^8 = \log^8(\sqrt[12]{n})$$

   Da für alle $m \geq 1$ gilt, dass $m \geq \log(m)$ und somit auch $m^8 \geq \log^8(m)$ folgt $\left(\sqrt[12]{n}\right)^8 \geq \log^8(\sqrt[12]{n})$.

Aufgabe 3 - Logarithmengesetze (4 Punkte)

Für die folgenden Aufgaben dürfen (nur) die folgenden Äquivalenzen verwendet werden:

$$x^y \cdot x^z = x^{y+z} \qquad (x^y)^z = x^{y \cdot z} \qquad x^{\log_x(y)} = \log_x(x^y) = y$$

$$\log_x(y) = \frac{\log y}{\log x} \qquad x \log_y(z) = \log_y(z^x) \qquad \log_x(y \cdot z) = \log_x y + \log_x z$$

1. Zeige: $\log_b(a) = 1/\log_a(b)$ (1 Punkt)
2. Zeige: $a^{1/\log_n(b)} = n^{\log_b(a)}$ (1.5 Punkte)
3. Vereinfache den Ausdruck $2^{\sum_{i=1}^{n} k \log(i)}$ (1.5 Punkte)

Lösung 3

1. Es gilt

$$\begin{aligned}\log_b(a) &= \frac{\log a}{\log b} \\ &= \frac{1}{\frac{\log b}{\log a}} \\ &= \frac{1}{\log_a(b)}\end{aligned}$$

2. Es gilt

$$\begin{aligned}a^{1/\log_n(b)} &\overset{1.}{=} a^{\log_b(n)} \\ &= \left(b^{\log_b(a)}\right)^{\log_b(n)} \\ &= b^{\log_b(a)\cdot\log_b(n)} \\ &= \left(b^{\log_b(n)}\right)^{\log_b(a)} \\ &= n^{\log_b(a)}\end{aligned}$$

3. Es gilt

$$\begin{aligned}2^{\sum_{i=1}^{n} k \log(i)} &= 2^{\sum_{i=1}^{n} \log(i^k)} \\ &= 2^{\log\left(\prod_{i=1}^{n} i^k\right)} \\ &= 2^{\log\left((n!)^k\right)} \\ &= (n!)^k\end{aligned}$$

Aufgabe 4 - Palindromzahlen (7 Punkte)

Palindromzahlen sind natürliche Zahlen, deren Darstellung von vorne nach hinten sowie von hinten nach vorne gelesen gleich ist. Die Zahlen 1331, 74247 oder 1 sind zum Beispiel Palindromzahlen. Wir beschränken uns in dieser Aufgabe auf Darstellungen im Dezimalsystem.

Um zu überprüfen, ob eine gegebene Zahl $x \in \mathbb{N}$ eine Palindromzahl ist, können die Ziffern von x paarweise verglichen werden: Zunächst werden die beiden äußersten Ziffern überprüft. Sind sie ungleich, ist x keine Palindromzahl. Ansonsten werden die nächstinneren Ziffern verglichen. Dieser Vorgang wiederholt sich so lange, bis die Mitte der Darstellung erreicht ist. Waren alle verglichenen Ziffern gleich, ist x eine Palindromzahl. Im Folgenden soll die Beschreibung des Algorithmus verfeinert und in Pseudocode umgesetzt werden.

1. Zunächst ist es hilfreich, die Anzahl Ziffern bestimmen zu können, die die Dezimaldarstellung einer beliebigen Zahl $x \in \mathbb{N}$ enthält. Beschreibe einen Algorithmus, der dies leistet. Nenne und begründe dabei die Anzahl benötigter arithmetischer Operationen. (1.5 Punkte)

2. Gegeben sei eine Zahl $x \in \mathbb{N}_+$ mit $n \in \mathbb{N}_+$ Ziffern. Wir nummerieren die Ziffern bei 0 beginnend aufsteigend von rechts nach links. Die höchstwertige Ziffer bezeichnen wir also als die $(n-1)$-ste Ziffer. Beschreibe, wie die i-te Ziffer von x bestimmt werden kann. Du darfst davon ausgehen, dass $0 \leq i < n$ gilt. Nenne und begründe dabei die Anzahl benötigter arithmetischer Operationen. (1.5 Punkte)

3. Formuliere nun den Algorithmus mit der folgenden Signatur in Pseudocode, der überprüft, ob seine Eingabe eine Palindromzahl ist. (3 Punkte)

   isPalindrome(x: $\mathbb{N}$): Bool

   Dabei stehen dir die Hilfsmethoden zu Verfügung, die in Teilaufgabe 1 und 2 entwickelt wurden:

   numberOfDigits(x: $\mathbb{N}$): $\mathbb{N}$

   getDigit(x: $\mathbb{N}$, i: $\mathbb{N}$): $\mathbb{N}$

4. Gib eine möglichst genaue Abschätzung der Anzahl arithmetischer Operationen im O-Kalkül an, die dein Algorithmus benötigt. Begründen deine Antwort. (1 Punkt)

Lösung 4

1. Gegeben sei $x \in \mathbb{N}$. Um die Anzahl $n \in \mathbb{N}_+$ an Ziffern in der Dezimaldarstellung von x zu bestimmen, zählen wir, wie oft wir x durch 10 teilen können, bis das Ergebnis kleiner als 1 ist. Dazu müssen n Divisionen und nach jeder Division ein Vergleich durchgeführt werden. Damit benötigt dieses Verfahren Zeit in $O(n) = O(\log x)$.

2. Um die $i-1$ Ziffern rechts der gesuchten Ziffer zu "entfernen", berechnen wir zunächst $x' = x \texttt{ div } 10^i$. Die i-te Ziffer von x lässt sich nun als $x' \texttt{ mod } 10$ bestimmen. Da beide Operationen in konstanter Zeit ablaufen, liegt der Zeitbedarf insgesamt in $O(1)$.

3. Mögliche Lösung:

```
1:  isPalindrome(x: ℕ): Bool
2:      leftIndex : ℕ = numberOfDigits(x)−1
3:      rightIndex : ℕ = 0
4:      leftDigit, rightDigit: ℕ
5:      while leftIndex > rightIndex do
6:          leftDigit ≔ getDigit(x, leftIndex)
7:          rightDigit ≔ getDigit(x, rightIndex)
8:          if leftDigit ≠ rightDigit then
9:              return false
10:         end
11:         leftIndex−−
12:         rightIndex++
13:     end
14:     return true
```

4. Sei $x \in \mathbb{N}$ die Eingabe mit $n \in \mathbb{N}_+$ Ziffern. Da die Laufzeit von numberOfDigits in $O(n)$ liegt und die Zeilen 3 und 4 nur Anweisungen mit konstantem Zeitbedarf beinhalten, liegt der Zeitbedarf vor dem Eintritt in die While-Schleife in $O(n)$. Die im Schleifenrumpf enthaltenen Anweisungen, inklusive der Aufrufe von getDigit, benötigen alle konstanten Zeitaufwand. In jedem Schleifendurchlauf verringert sich die Differenz von leftIndex und rightIndex um 2. Da diese Differenz vor dem ersten Schleifendurchlauf bei $n - 1$ liegt, liegt auch der Zeitbedarf für die Zeilen 5–13 in $O(n)$. Insgesamt liegt die Laufzeit von isPalindrome also in $O(n) = O(\log x)$.