DAP2 – Präsenzübung 1

Besprechung: 26.04.2017 — 28.04.2017

Abgabe:

Präsenzübungen müssen nicht zu Hause bearbeitet werden, sondern werden unter Anleitung während der Übung erarbeitet.

Präsenzaufgabe 1.1: (Erste Schritte mit der O–Notation)

Gegeben sei die Funktion

$$f(n) = \begin{cases} 2017^n & \text{falls } n \leq 20 \\ n^{2017} & \text{falls } 20 < n \leq 201 \\ 2017n & n > 201. \end{cases}$$

Bestimmen Sie eine Funktion $g(n)$, sodass $f(n) \in \Theta(g(n))$ gilt.

Welchen Einfluss haben die Konstanten 20, 201 und 2017 auf die Wahl Ihrer Funktion gehabt?

Bestimmen Sie auch eine nicht konstante Funktion $h(n)$, sodass $f(n) \in \omega(h(n))$ gilt.

Lösung:
Beh.: Eine solche gesuchte Funktion ist $g(n) = 2017n$ (und nicht etwa exponentiell oder polynomiell mit höhem Exponent).

Beweis: Sei $n > n_0 = 202$ und $c = 1$. Dann gilt $g(n) = f(n)$ und somit sowohl $f(n) \in O(g(n))$ als auch $f(n) \in \Omega(g(n))$ und somit auch $\Theta(g(n))$. Wir könnten auch $g(n) = n$ nehmen, müssten aber denn die Konstanten aus den Definitionen von $O(f(n))$ und $\Omega(f(n))$ anpassen (z.B. $c' = 2017$ und $c'' = 1$, mit $f(n) \leq c' \cdot g(n)$ und $f(n) \geq c'' \cdot g(n)$, für alle $n \geq n_0 = 202$).

Die Konstanten 20 und 201 haben keinen Einfluss auf die Wahl der Funktion $g(n)$, da sie nur eine konstante Anzahl der Werte n ausschließen. Die Konstante 2017 für den Fall $n \leq 201$ wurde oben schon diskutiert (da man $g(n) = \ell \cdot n$ für beliebiges $\ell > 0$ auswählen darf).

$f(n) \in \omega(h(n))$ bedeutet, dass die Funktion $h(n)$ asymptotisch kleiner als $f(n)$ ist. D.h. für alle $c > 0$ gibt es $n_0 \in \mathbb{N}$, sodass $f(n) > c \cdot h(n)$ für alle $n \geq n_0$ gilt. Die Funktion $h(n)$ muss nicht konstant sein, deswegen sei $h(n) = \ln n$ oder $h(n) = \sqrt{n}$. Dann gilt es $f(n) > c \cdot \ln n$ für alle $n \geq n_0 = 202$ und für alle $c > 0$.

□

Präsenzaufgabe 1.2: (Anschauung der Landau-Notation)

Geben Sie je ein Beispiel für Funktionen $a(n)$, $b(n)$, $c(n)$ und $d(n)$, sodass die Aussagen

- $a(n) \in O(1)$
- $b(n) \in o(1)$
- $c(n) \in \Omega(1)$
- $d(n) \in \omega(1)$

gelten, und beweisen Sie die Korrektheit der Aussagen für die von Ihnen ausgewählten Funktionen.

Was bedeuten diese Aussagen umgangssprachlich?

Lösung:

(a) *Beh.*: $a(n) = \sin n \in O(1)$.

Beweis: $O(1)$ bedeutet, dass es eine Konstante c und ein $n_0 > 0$ gibt, sodass für alle $n > n_0$ $c \cdot 1 \geq a(n)$. Das heißt, dass a beschränkt ist. Eine solche Funktion ist die Sinusfunktion, weil für alle $n \in \mathbb{N}$, $-1 \leq \sin n \leq 1$ ist. Wir wählen $n_0 = 1$ und $c = 1$.

(b) *Beh.*: $b(n) = \frac{1}{n} \in o(1)$.

Beweis: $o(1)$ bedeutet, dass es für jede Zahl $c > 0$ ein $n_0 > 0$ gibt, sodass für alle $n > n_0$, $b(n) < c \cdot 1$ gilt. Da c beliebig klein sein kann, muss $b(n)$ irgendwann betragsmäßig beliebig klein werden. Ein Beispiel wäre $b(n) = \frac{1}{n}$. Wir wählen für beliebiges $c > 0$ $n_0 = \lceil 1/c \rceil$, und denn für alle $n > n_0$ gilt es $n > n_0 = \lceil 1/c \rceil \geq \frac{1}{c} \Leftrightarrow \frac{1}{n} < c$.

(c) *Beh.*: $c(n) \in \Omega(1)$.

Beweis: $\Omega(1)$ bedeutet, dass es eine Konstante k und ein $n_0 > 0$ gibt, sodass für alle $n > n_0$ $k \cdot 1 \leq c(n)$ gilt. Das heißt, dass $c(n)$ von unten mit einer Konstante beschränkt ist. Eine solche Funktion ist z.B. $c(n) = n$, wobei wir $n_0 = 1$ und $c = 1$ wählen können.

(d) *Beh.*: $d(n) \in \omega(1)$.

Beweis: $\omega(1)$ bedeutet, dass es für jede Zahl c ein $n_0 > 0$ gibt, sodass für alle $n > n_0$, $c \cdot 1 < d(n)$ gilt. Irgendwann wird also jede Zahl durch d überschritten. Ein einfaches Beispiel wäre $d(n) = n$ mit $n_0 = \max\{1, c\}$.

$\square$

Präsenzaufgabe 1.3: (Asymptotische Komplexität)

Geben Sie *jeweils* eine möglichst kleine Funktion $g : \mathbb{N} \to \mathbb{R}^+$ an, für die gilt:

(a) $2^n/50 - 12n^{10} + n^9 = O(g(n))$

(b) $5n + 2\sqrt{n} \ln n = O(g(n))$

und eine möglichst große Funktion $h : \mathbb{N} \to \mathbb{R}^+$, für die gilt:

(c) $n^4 - 10n^3 + 5 = \Omega(h(n))$

(d) $n \log n^2 - 2\sqrt{n} \ln^2 n = \Omega(h(n))$

Beweisen Sie Aussagen (a) – (d) für die von Ihnen gegebenen Funktionen gemäß der Definitionen der Landau-Symbole aus der Vorlesung.

Hinweis: Bei dieser Aufgabe dürfen Sie die Aussagen nutzen, die in der Vorlesung bewiesen wurden.

Lösung:

(a) *Beh.*: $2^n/50 - 12n^{10} + n^9 = O(2^n)$.

Beweis:

Aus der Vorlesung ist bekannt: $n^9 = O(2^n) :\Leftrightarrow \exists c', n_0 > 0,\ \forall n \geq n_0 : n^9 \leq c' \cdot 2^n$. Außerdem gilt $-12n^{10} \leq 0$, für alle $n > 0$. Damit gilt:

$$\begin{aligned} & 2^n/50 - 12n^{10} + n^9 < 2^n/50 + n^9 \leq c \cdot 2^n && \mid n \geq n_0 \\ \Leftarrow\quad & 2^n/50 + n^9 \leq 2^n/50 + c' \cdot 2^n = \left(c' + \frac{1}{50}\right) \cdot 2^n \\ \Leftrightarrow\quad & c' + \frac{1}{50} \leq c \end{aligned}$$

Wir können also $c = c' + 1/50$ und das n_0 aus der Vorlesung wählen (die Existenz genügt für den Beweis), dann folgt $2^n/50 - 12n^{10} + n^9 \leq c \cdot 2^n \Rightarrow 2^n/50 - 12n^{10} + n^9 = O(2^n)$.

(b) *Beh.*: $5n + 2\sqrt{n} \ln n = O(n)$.

Beweis:

Aus der Vorlesung ist bekannt: $\ln n = O(\sqrt{n}) :\Leftrightarrow \exists c', n_0 > 0,\ \forall n \geq n_0 : \ln n \leq c' \cdot \sqrt{n}$. Damit gilt:

$$\begin{aligned} & 5n + 2\sqrt{n} \log n \leq c \cdot n && \mid n \geq n_0 \\ \Leftarrow\quad & 5n + 2\sqrt{n} \cdot c'\sqrt{n} \leq c \cdot n \\ \Leftrightarrow\quad & 5n + 2c'n \leq c \cdot n \\ \Leftrightarrow\quad & 5 + 2c' \leq c \end{aligned}$$

Wir können also $c = 2c' + 5$ und das n_0 aus der Vorlesung wählen (die Existenz genügt für den Beweis), dann folgt $5n + 2\sqrt{n} \ln n = O(n)$.

(c) *Beh.*: $n^4 - 10n^3 + 5 = \Omega(n^4)$.

Beweis: $n^4 - 10n^3 + 5 = \Omega(n^4)$ bedeutet, dass es $c > 0$ und $n_0 \in \mathbb{N}$ existieren, sodass $n^4 - 10n^3 + 5 \geq c \cdot n^4$ für alle $n \geq n_0$ gilt.

$$
\begin{aligned}
& n^4 - 10n^3 + 5 \geq c \cdot n^4 && \mid n \geq n_0 \\
\Leftrightarrow\ & n^3 \cdot (n-10) + 5 \geq c \cdot n^4 \\
\Leftarrow\ & n^3 \cdot (n-10) + 5 \geq n^3 \cdot \frac{n}{2} + 5 \geq c \cdot n^4 && \mid n \geq 20 \\
\Leftarrow\ & \frac{1}{2} \cdot n^4 + 5 \geq c \cdot n^4
\end{aligned}
$$

Für alle $n \geq n_0 = 20$ gilt, dass $n - 10 > n/2$ ist. Wir wählen $n_0 = 20$ und $c = 1/2$. Dann folgt $n^4 - 10n^3 + 5 \geq n^4/2 + 5 \geq n^4/2$, was ergibt insgesamt $n^4 - 10n^3 + 5 = \Omega(n^4)$.

Es ist wichtig zu bemerken, dass bei $\Omega(h(n))$ Funktionen wieder der "größte" Summand gesucht wird.

(d) *Beh.*: $n \log n^2 - 2\sqrt{n} \ln^2 n = \Omega(n \log n)$.

Beweis:

Wir sollen zeigen, dass es $c > 0$ existiert, sodass für alle $n \geq n_0 > 0$ es $n \log n^2 - 2\sqrt{n} \ln^2 n \geq c \cdot n \log n$ gilt. Es gilt:

$$
\begin{aligned}
& n \log n^2 - 2\sqrt{n} \ln^2 n \geq c \cdot n \log n && \mid n \geq n_0 \\
\Leftrightarrow\ & 2n \log n - 2\sqrt{n} \left(\frac{\log n}{\log e} \right)^2 \geq c \cdot n \log n && \mid\ : \sqrt{n} \cdot \log n (> 0) \\
\Leftrightarrow\ & 2\sqrt{n} - \frac{2}{\log^2 e} \log n \geq c \cdot \sqrt{n}
\end{aligned}
$$

Aus der Vorlesung ist bekannt: $\log n = o(\sqrt{n}) \Leftrightarrow \forall d > 0, \exists n_0 > 0,\ \forall n \geq n_0 : \log n \leq d \cdot \sqrt{n}$. Damit folgt, dass

$$
2\sqrt{n} - \frac{2}{\log^2 e} \log n \geq 2\sqrt{n} - \frac{2}{\log^2 e} \cdot d\sqrt{n} \geq c \cdot \sqrt{n}
$$

für alle $d > 0$ und dementsprechen ausgewählten n_0 ist. Für $c = 2 - \frac{2d}{\log^2 e}$ gilt die gewünschte Ungleichung. Um die Bedingung $c > 0$ zu erfüllen, muss $2 - \frac{2d}{\log^2 e} > 0 \Leftrightarrow 0 < d < \log^2 e$ gelten. Für die ausgewählten c und n_0 gilt somit $n \geq n_0 > 0$ $n \log n^2 - 2\sqrt{n} \ln^2 n \geq c \cdot n \log n$, was wir beweisen wollten.

$\square$

Präsenzaufgabe 1.4: (Laufzeitanalyse: Primfaktoren)

Führen Sie eine Worst-Case Laufzeitanalyse für den folgenden Algorithmus durch, der bei Eingabe einer ganzen Zahl n alle ihre Primfaktoren im Array A zurückgibt.

Primfaktor(int n):

1 $j \leftarrow 1$
2 $m \leftarrow \lfloor\sqrt{n}\rfloor$
3 $i \leftarrow 2$
4 **while** $i \leq m$ **do**
5   **if** n *teilbar durch* i **then**
6   A[j] $\leftarrow i$
7         $n \leftarrow n/i$
8         $j \leftarrow j+1$
9   **else**
10         $i \leftarrow i+1$
11 **if** $n > 1$ **then**
12   A[j] $\leftarrow n$
13 **return** A

Ordnen Sie die Laufzeit $f(n)$ bei Eingabe der Zahl n in die O-Notation ein, d.h. finden Sie eine möglichst kleine Funktion $g(n)$, sodass $f(n) \in O(g(n))$ ist.

Lösung:

Beh.: Der Algorithmus benötigt $T(n) = O(\sqrt{n})$ Rechenschritte.

Beweis: Wir gehen Zeile für Zeile durch den Pseudocode und geben jeweils an, wie viele Rechenschritte die Zeile benötigt:

1. Eine Zuweisung benötigt einen Rechenschritt.

2. 3. wie 1., jede dieser Zeilen braucht einen Rechenschritt.

4. Dieser Schleifenkopf wird $a+b+1$-mal ausgeführt. a gibt an, wie oft die Bedingung in Zeile 5 erfüllt ist, b wie oft die Bedingung in Zeile 5 nicht erfüllt ist. Die Werte von a und b werden später analysiert. Das macht $a+b+1$ Rechenschritte.

5. Diese Anweisung wird $a+b$-mal ausgeführt (als die einzige Anweisung im Schleifenrumpf). Das macht $a+b$ Rechenschritte.

6. 7. 8. Jede dieser Zeilen braucht einen Rechenschritt, und jede wird a-mal ausgeführt. Für 3 Zeilen macht es $3a$ Rechenschritte.

9. Dies ist eine Kontrollzeile und braucht keinen Rechenschritt.

10. Für jede falsche Bedingung in Zeile 5 wird diese Zeile einmal ausgeführt. Das braucht insgesamt b Rechenschritte.

11. Diese Zeile braucht einen Rechenschritt. Es kann passieren, dass höchstens eine Primzahl, die größer $m = \sqrt{n}$ ist, noch nicht gefunden wurde. Dafür brauchen wir diese Zeile.

12. Diese Zeile wird nur ausgeführt, wenn die Bedingung in Zeile 11 erfüllt ist. Im schlimmsten Fall braucht sie einen Rechenschritt.

13. Diese Zeile braucht einen Rechenschritt.

Nun summieren wir alle ermittelten Beiträge zu

$$\begin{aligned} T(n) &= 3+a+b+1+a+b+3a+b+3 \\ &= 5a+3b+7 \end{aligned}$$

Jetzt müssen wir die Werte von a und b begrenzen. Die Anzahl a der Ausführungen der Zeilen 6-8 ist durch die Verringerung von n begrenzt. Diese Zeilen werden so oft ausgeführt, wie Divisionen in Zeile 7 möglich sind. Diese Zahl ist kleiner oder gleich der Anzahl von möglichen Divisionen durch 2 (bis das Ergebnis gleich oder kleiner 1 ist), d.h. im schlimmsten Fall $\log_2 n$. Damit folgt, dass $a \leq \log_2 n$ gilt.

In schlimmsten Fall für b hat n keine Teiler außer 1 und n. Dann wird die Bedingung in Zeile 5 immer falsch und es ist $b \leq \sqrt{n} - 1$. Es gilt denn:

$$T(n) \leq 5\log_2 n + 3\sqrt{n} - 3 + 7 = 5\log_2 n + 3\sqrt{n} + 4$$

In der Vorlesung wurde gezeigt, dass $\log n \in O(\sqrt{n})$. In der Klausur dürfte man hier auch mit Worten wie "bekannt" oder "offensichtlich" argumentieren. Schließlich haben wir dass

$$T(n) \leq 5\log_2 n + 3\sqrt{n} + 4 \in O(\sqrt{n})$$

ist.

$\square$