Technische Universität München
Fakultät für Informatik, I-14
Prof. Harald Räcke
Robin Münk

SoSe 2025
Übungsblatt 2
2. Mai 2025

Algorithmen und Datenstrukturen

Aufgabe 1 Asymptotische Notation I

Vergegenwärtigen Sie sich zunächst die Definitionen von $O(n)$ und $\Theta(n)$, wie sie in der Vorlesung vorgestellt wurden. Zeigen Sie dann folgende Behauptungen mit einem formalen Beweis wie in der Vorlesung (Grenzwertabschätzung oder Mengendarstellung):

a) $7n^4 \in O(n^5)$

b) $n^2/2 - 2n \in \Theta(n^2)$

c) $\log n \in o(n)$

d) $2^{2n} \notin O(2^n)$

Beispiellösung:

Die Definitionen der für uns relevanten Landau-Symbole waren:

$$
\begin{aligned}
O(g) &:= \{f : \mathbb{R} \to \mathbb{R} \;:\; \text{es existieren } c > 0 \text{ und } n_0 \in \mathbb{N}, \text{ so dass für alle } n \geq n_0 \text{ gilt:}\\
&\qquad f(n) \leq c \cdot g(n)\}\\
\Omega(g) &:= \{f : \mathbb{R} \to \mathbb{R} \;:\; \text{es existieren } c > 0 \text{ und } n_0 \in \mathbb{N}, \text{ so dass für alle } n \geq n_0 \text{ gilt:}\\
&\qquad f(n) \geq c \cdot g(n)\}\\
\Theta(g) &:= \{f : \mathbb{R} \to \mathbb{R} \;:\; \text{es existieren } c_1, c_2 > 0 \text{ und } n_0 \in \mathbb{N}, \text{ so dass für alle } n \geq n_0 \text{ gilt:}\\
&\qquad c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}\\
o(g) &:= \{f : \mathbb{R} \to \mathbb{R} \;:\; \text{für alle } c > 0 \text{ existiert } n_0 \in \mathbb{N}, \text{ so dass für alle } n \geq n_0 \text{ gilt:}\\
&\qquad f(n) \leq c \cdot g(n)\}\\
\omega(g) &:= \{f : \mathbb{R} \to \mathbb{R} \;:\; \text{für alle } c > 0 \text{ existiert } n_0 \in \mathbb{N}, \text{ so dass für alle } n \geq n_0 \text{ gilt:}\\
&\qquad f(n) \geq c \cdot g(n)\}
\end{aligned}
$$

Wenn der Grenzwert existiert kann man auch die folgende Definition benutzen:

$$f \in O(g) \iff 0 \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \ .$$

$$f \in \Omega(g) \iff 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty.$$

$$f \in \Theta(g) \iff 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty.$$

$$f \in o(g) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

$$f \in \omega(g) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty.$$

Damit werden die gegebenen Behauptungen gezeigt:

a) $7n^4 \in O(n^5)$

Wir betrachten die Ungleichung aus der Definition von $O(n)$:

$$0 \leq 7n^4 \leq c \cdot n^5$$

Für $n \geq 1$ kann ohne Umkehr der Ungleichung durch n^4 dividiert werden:

$$0 \leq 7 \leq c \cdot n$$

Wählen wir nun $c = 7$ und $n_0 = 1$, so gilt die Ungleichung für alle $n \geq n_0$. Somit ist die Behauptung gezeigt. $\square$

Alternativ lässt sich auch die Grenzwertnotation anwenden:

$$\lim_{n\to\infty} \frac{7n^4}{n^5} = \lim_{n\to\infty} \frac{7}{n} = 0$$

Dann gilt sogar $f \in o(g)$.

b) $n^2/2 - 2n \in \Theta(n^2)$

Auch hier betrachten wir die Ungleichung aus der Definition von $\Theta(n^2)$, teilen aber auf in eine linke und eine rechte Seite.

Zunächst also die Abschätzung der unteren Schranke:

$$c_1 \cdot n^2 \leq \frac{n^2}{2} - 2n$$

Auch hier können wir für $n \geq 1$ durch n^2 ohne Umkehr der Ungleichung dividieren:

$$c_1 \leq \frac{1}{2} - \frac{2}{n}$$

Diese Ungleichung gilt etwa für $c_1 = \frac{1}{4}$ und alle $n \geq n_0 = 8$.

Ähnlich kann man die obere Schranke abschätzen:

$$\frac{n^2}{2} - 2n \leq c_2 \cdot n^2$$

Wieder teilen wir – wie oben – durch n^2 und erhalten:

$$\frac{1}{2} - \frac{2}{n} \leq c_2$$

Diese Ungleichung gilt dann etwa für $c_2 = \frac{1}{2}$ bei gleich gewähltem $n_0 = 8$ wie oben.

Insgesamt haben wir nun die beiden Schranken, $c_1 = \frac{1}{4}$ und $c_2 = \frac{1}{2}$, und eine Grenze, $n_0 = 8$, identifiziert, so dass die komplette Ungleichung

$$0 \leq c_1 \cdot n^2 \leq \frac{n^2}{2} - 2n \leq c_2 \cdot n^2$$

für alle $n \geq n_0$ gilt. Also gilt die Behauptung. $\square$

Alternativ lässt sich auch die Grenzwertnotation anwenden:

$$\lim_{n\to\infty} \frac{n^2/2 - 2n}{n^2} = \lim_{n\to\infty} \frac{1}{2} - \frac{2}{n} = \frac{1}{2}$$

Da $0 < \frac{1}{2} < \infty$, gilt $n^2/2 - 2n \in \Theta(n^2)$.

c) $\log n \in o(n)$

Wir benutzen die Grenzwertnotation:

$$\lim_{n\to\infty} \frac{\log n}{n} \overset{\text{l'Hospital}}{=} \lim_{n\to\infty} \frac{1/n}{1} = \lim_{n\to\infty} \frac{1}{n} = 0$$

Hier benutzen wir die Regel von l'Hospital. Diese Regel ist bei Aufgaben dieser Art häufig sehr nützlich. Da $\lim_{n\to\infty} \frac{\log n}{n} = 0$, gilt die Behauptung. □

d) $2^{2n} \notin O(2^n)$

Angenommen, dass diese Behauptung *nicht* zutrifft, dann existieren auch Konstanten c und $n_0 > 0$, so dass die Ungleichung

$$0 \le 2^{2n} \le c \cdot 2^n$$

für alle $n \ge n_0$ gilt. Wir zerlegen 2^{2n} und erhalten

$$2^{2n} = 2^n \cdot 2^n \le c \cdot 2^n \Leftrightarrow 2^n \le c\ .$$

Die Folge 2^n ist aber unbeschränkt, und es existiert keine Konstante c, die für beliebige Werte von n die Forderung $c \ge 2^n$ erfüllt. Dies ist ein Widerspruch zur Annahme, und die die zu zeigende Behauptung gilt somit. □

Alternativ lässt sich auch die Grenzwertnotation anwenden:

$$\lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} \frac{4^n}{2^n} = \lim_{n\to\infty} (\frac{4}{2})^n = 2^n = \infty$$

Da $\lim_{n\to\infty} \frac{2^{2n}}{2^n} = \infty$, gilt $2^{2n} \in \omega(2^n)$ und dementsprechend $2^{2n} \notin O(2^n)$.

Aufgabe 2 Asymptotische Notation II

Treffen die folgenden Behauptungen zu? Beweisen sie deren Richtigkeit, oder widerlegen Sie sie, entweder mittels Beweis oder durch Angabe eines Gegenbeispiels!

a) Sei die Komplexität einer Funktion f bestimmt als $O(n)$. Dann ist die n-malige Ausführung $O(n^2)$.

b) Falls $f(n) = \Theta(g(n))$, dann folgt $2^{f(n)} \in \Theta(2^{g(n)})$.

c) $n^n = O(2^n)$

d) Falls $g = O(f)$ und $h = O(f)$, dann gilt auch $g = O(h)$

Beispiellösung:

a) *Sei die Komplexität einer Funktion f bestimmt als $O(n)$. Dann ist die n-malige Ausführung $O(n^2)$.*

Sei $T_f(n)$ die Laufzeitfunktion von f. Aus der Annahme folgt, dass es c und n_0 gibt, so dass

$$0 \le T_f(n) \le c \cdot n$$

für alle $n \ge n_0$. Bei n- Ausführung muss einfach multipliziert werden:

$$0 \le n \cdot T_f(n) \le n \cdot c \cdot n = c \cdot n^2$$

Das ist direkt die Forderung aus der Definition von $O(n^2)$, die Aussage ist **korrekt**!

b) *Falls* $f(n) = \Theta(g(n))$*, dann folgt* $2^{f(n)} = \Theta(2^{g(n)})$.

Wähle $f(n) = n$ und $g(n) = 2 \cdot n$, dann gilt offensichtlich $f(n) = \Theta(g(n))$:

$$0 \leq c_1 \cdot 2 \cdot n \leq n \leq c_2 \cdot 2 \cdot n \; ,$$

etwa mit $c_1 = c_2 = \frac{1}{2}$ und $n_0 = 1$. Nehmen wir nun an, dass die Aussage korrekt ist, und wir erhalten aus $2^{f(n)} = \Theta(2^{g(n)})$ die Ungleichung

$$0 \leq c_1 \cdot 2^{2 \cdot n} \leq 2^n \leq c_2 \cdot 2^{2 \cdot n} \; .$$

Vereinfachen wir diese Formel nun durch Teilung durch 2^n, was wegen $n > 0$ ohne Umkehr der Vorzeichen problemlos möglich ist:

$$0 \leq c_1 \cdot 2^n \leq 1 \leq c_2 \cdot 2^n \; .$$

Dies ist aber ein Widerspruch, da wir zwar ein c_2, nicht aber ein c_1 finden können, da die Folge 2^n unbeschränkt ist.

Damit ist die komplette Aussage **falsch**, da mit der getroffenen Auswahl für f und g ein Gegenbeispiel gefunden wurde.

c) $n^n = O(2^n)$

Nehmen wir wieder an, dass die Aussage wahr ist. Dann gilt aus der Definition von O die Ungleichung

$$0 \leq n^n \leq c \cdot 2^n \; ,$$

was sich (wieder wegen $2^n > 0$ für $n > 0$) umformen lässt zu

$$\left(\frac{n}{2}\right)^n \leq c \; .$$

Auch diese Folge ist unbeschränkt und wächst beliebig für große Werte von n. Damit kann keine Konstante c im Sinn einer oberen Schranke existieren.

Damit ist die Aussage als **falsch** widerlegt!

d) *Falls* $g = O(f)$ *und* $h = O(f)$*, dann gilt auch* $g = O(h)$

Die Aussage ist **falsch**. Es gilt zum Beispiel $n^2 = O(n^3)$ und $n = O(n^3)$, aber nicht $n^2 = O(n)$.

Aufgabe 3 (P) Addition

Gegeben sind eine Basis b und zwei `vector<int> x, y`, die zwei positive base-b Integer repräsentieren. Ziffer i ist dabei an Stelle `i` mit Wertigkeit b^i, geordnet von der höchst- zur niedrigstwertigen Ziffer von Anfang bis Ende der Liste, die höchstwertige Ziffer steht also am Anfang der Liste. Es gibt keine führenden Nullen.

Bsp.: [42, 53, 28], [37], 55 → [42, 54, 10]

a) Implementieren Sie eine C++ Funktion, um die repräsentierten Zahlen zu addieren und die Summe als `vector<int>` zurückzugeben. Beachten Sie, dass die Listen der Zahlen unterschiedlich lang sein können. Nutzen Sie gerne auch das Testing Framework[1].

Als optionale Hilfestellung finden Sie in der Datei `blatt2hint.cpp` ein grobes Gerüst einer Lösung, welches Sie verwenden können, wenn Sie nicht weiter kommen.

Hinweis: Vielleicht hilft es Ihnen, die Zahlen zu Begin so zu tauschen, dass z.B. `y` die Zahl mit mehr Ziffern ist.

[1] Weitere Informationen in *Setup Programmieraufgaben* in Moodle

b) Seien n_x, n_y jeweils die Anzahl der Ziffern der Zahlen x, y geschrieben zur Basis b. Geben Sie abhängig von n_x, n_y eine Abschätzung an die worst-case Laufzeit Ihrer Implementierung in asymptotischer Notation an.

Beispiellösung:

a) Wie im Hinweis angegeben, tauschen wir zu Beginn ggf. die Zahlen aus, sodass `y` die Zahl mit mehr Ziffern ist. Dann iterieren wir über $i = 0, \ldots, n_y - 1$: Die Indizes in den Listen der Zahlen `x, y` sind dann jeweils $n_x - i - 1$ und $n_y - i - 1$. Wir addieren die Ziffern und merken uns ggf. einen Übertrag in einer *carry* Variable. Sollte `x` keine Ziffern mehr haben und das carry ist 0, sind wir fertig. Wenn wir am Ende von `y` angekommen sind, aber das carry nicht 0 ist, fügen wir es am Anfang der Liste ein.

b) Im worst-case müssen wir die gesamte Liste der längeren Zahl durchlaufen und zum Schluss vorne in der Liste eine 1 einfügen. Es ergibt sich die folgende Abschätzung

$$O(1) + \underbrace{\max\{n_x, n_y\}}_{\text{Iterationen}} \cdot \underbrace{O(1)}_{\text{Schleifen-Inhalt}} + \underbrace{O(\max\{n_x, n_y\})}_{\text{Vorne Einfügen in } \texttt{vector}} = O(\max\{n_x, n_y\})$$