Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Friedrich Gretz, Benjamin Kaminski, Thomas Ströder

Tutoraufgabe 1 (Asymptotische Komplexität):

Ordnen Sie die folgenden Funktionen nach ihrer asymptotischer Komplexität in aufsteigender Reihenfolge:

$$n!, \quad n^2, \quad n\log n, \quad n^n, \quad \sqrt{n}, \quad n, \quad n^3, \quad 2^n, \quad 2^{\log n}, \quad \log n, \quad n\sqrt{n} \ .$$

Lösung:

$$\log n \leq \sqrt{n} \leq n \overset{*}{=} 2^{\log n} \leq n\log n \leq n\sqrt{n} \leq n^2 \leq n^3 \leq 2^n \leq n! \leq n^n \ .$$

für alle n größer als ein gewisses n_0.
$(*)$ Die Aussage "$n = 2^{\log n}$" gilt nur für den Logarithmus zur Basis 2.

Tutoraufgabe 2 (Ungleichungen):

Zeigen Sie die folgenden Aussagen für beliebige $n \in \mathbb{N}^{>0}$:

a) $\sum_{i=1}^{n}(4 \cdot i + 3) \ \leq \ 4(n^2 + n)$ **b)** $\sum_{i=1}^{n} \log_2 i \ \leq \ \log_2 n^n$ **c)** $\log_2 n \ \leq \ 3 \cdot \log_4 n$

Lösung:

a)

$$\begin{array}{rcl} \sum_{i=1}^{n}(4i+3) & = & \sum_{i=1}^{n} 4i \ + \ \sum_{i=1}^{n} 3 \\ & = & 4 \cdot \sum_{i=1}^{n} i + 3n \\ & = & 4 \cdot (\frac{n(n+1)}{2}) + 3n \\ & < & 4n^2 + 4n \\ & = & 4(n^2+n) \end{array}$$

b) Diese Aufgabe lösen wir mit Hilfe des Logarithmengesetzes: $\log_a x + \log_a y = log_a(x \cdot y)$.

1. Möglichkeit:

$$\sum_{i=1}^{n} \log_2 i = log_2 \prod_{i=1}^{n} i = \log_2 n! \leq \log_2 n^n$$

2. Möglichkeit

$$\sum_{i=1}^{n} \log_2 i \leq \sum_{i=1}^{n} \log_2 n = n \cdot \log_2 n = \log_2 n^n$$

c) Für diese Aufgabe nutzen wir ein weiteres Logarithmengesetz, den Basistausch: $\log_a x = \frac{\log_b x}{\log_b a}$

$$\log_2 n = \frac{\log_4 n}{\log_4 2} = 2 \cdot \log_4 n \leq 3 \cdot \log_4 n$$

Tutoraufgabe 3 (O-Notation):

Zeigen oder widerlegen Sie die folgenden Aussagen:

a) $g(n) = 2n^3 + 142n^2 + 462 \in \Theta(n^3)$
b) $2^{n+1} \in \Theta(2^n)$
c) $\log n \in \mathcal{O}(\sqrt{n})$
d) $max(f(n),\ g(n)) \in \Theta(f(n) + g(n))$
e) $g(n) + f(n) \in \mathcal{O}(g(f(n)))$

Lösung:

(a) Die Aussage gilt: $g(n) = 2n^3 + 142n^2 + 462 \in \Theta(n^3)$

$$\begin{aligned} & g(n) \in \Theta(n^3) \\ \Leftrightarrow\ & \exists c_1, c_2 \in \mathbb{R}, n_0 \in \mathbb{N} : c_1 \cdot n^3 \leq g(n) \leq c_2 \cdot n^3\ \forall n > n_0 \quad | \text{ nach Definition} \end{aligned}$$

Diese Aussage ist für $n_0 = 1, c_1 = \frac{1}{c_2}$ und $c_2 = 2 + 142 + 462 = 606$ erfüllt.

q.e.d

(b) Die Aussage gilt: $2^{n+1} \in \Theta(2^n)$

$$\begin{aligned} & 2^{n+1} \in \Theta(2^n) \\ \Leftrightarrow\ & \exists c_1, c_2 \in \mathbb{R}, n_0 \in \mathbb{N} : c_1 \cdot 2^n \leq 2^{n+1} \leq c_2 \cdot 2^n\ \forall n > n_0 \quad | \text{ nach Definition} \\ \Leftrightarrow\ & \exists c_1, c_2 \in \mathbb{R}, n_0 \in \mathbb{N} : c_1 \cdot 2^n \leq 2 \cdot 2^n \leq c_2 \cdot 2^n\ \forall n > n_0 \\ \Leftarrow\ & 2 \cdot 2^n \leq 2 \cdot 2^n \leq 2 \cdot 2^n\ \forall n > 1 \quad | \ c_1 = 2, c_2 = 2, n_0 = 1 \end{aligned}$$

q.e.d

Alternative Lösung:

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

Hieraus folgt dass $2^{n+1} \in \Theta(2^n)$.

(c) Die Aussage gilt. Beweise:
Wir müssen zeigen dass $\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} \geq 0$ und $\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} < \infty$.

$$\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} \overset{\text{L'Hôspital}}{=} \lim_{n\to\infty} \frac{\frac{1}{\ln(2)n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{2\sqrt{n}}{n\ln(2)} = 0$$

q.e.d

(d) Die Aussage gilt.
Es ist zu zeigen, dass Folgendes gilt:

$$\exists c_1, c_2 \in \mathbb{R}_{\geq 0},\ n_0 \in \mathbb{N} :\ c_1 \cdot (f(n) + g(n)) \leq max(f(n), g(n)) \leq c_2 \cdot (f(n) + g(n)) \quad \forall n \geq n_0$$

für $c_1, c_2 \in \mathbb{R}_{\geq 0}$ und $n \geq n_0$ für ein hinreichend großes n_0 mit $n_0 \in \mathbb{N}$.
Wir beginnen mit $max(f(n), g(n)) \leq c_2 \cdot (f(n) + g(n))$:

Da $f(n)$ und $g(n)$ positive Funktionen sind gilt, $(f(n) + g(n)) \geq max(f(n), g(n))$. Aus dieser Abschätzung erhalten wir dann unmittelbar:

$$\exists c_2 \in \mathbb{R}_{\geq 0},\ n_0 \in \mathbb{N} :\ max(f(n), g(n)) \leq c_2 \cdot (f(n) + g(n)) \quad \forall n \geq n_0$$

für ein beliebiges $c_2 \geq 1$ und $n \geq n_0$.

Um den 2. Teil zu zeigen schätzen wir den Ausdruck $f(n) + g(n)$ nach oben hin ab und erhalten

$$\begin{array}{rrcl} & f(n) + g(n) & \leq & 2 \cdot max(f(n), g(n)) \\ \Leftrightarrow & \frac{1}{2} \cdot f(n) + g(n) & \leq & max(f(n), g(n)) \end{array}$$

Hieraus ergibt sich unmittelbar:

$$\exists c_1 \in \mathbb{R}_{\geq 0},\ n_0 \in \mathbb{N} :\ c_1 \cdot (f(n) + g(n)) \leq max(f(n), g(n))$$

für $c_1 = \frac{1}{2}$

q.e.d

(e) Die Aussage gilt nicht, wie wir an dem folgenden einfachen Gegenbeispiel sehen.
Für $g(n) = 1$ und $f(n) = n^2$ gilt $n^2 \notin \mathcal{O}(1)$, denn es gibt kein $c \in \mathbb{R}_{\geq 0}$, sodass für alle $n \in \mathbb{N}$ gilt:

$$n^2 \leq c \cdot 1$$

Offensichtlich gilt diese Gleichung für kein $c \in \mathbb{R}_{\geq 0}$, wenn wir $n = c + 1$ wählen.

q.e.d

Tutoraufgabe 4 (Programmanalyse):

Gegeben sei der folgende Algorithmus zur Suche von Duplikaten in einem Array:

```
bool duplicate(int E[], int n) {
  for(int i = 0; i < n; i++)
    for(int j = i+1; j < n; j++)
      if(E[i] == E[j])
        return true;

  return false;
}
```

Er erhält ein Array E mit n Einträgen, für das er überprüft, ob zwei Einträge des Arrays denselben Wert besitzen.

Für die Average-Case Analyse gehen wir von folgenden Voraussetzungen aus:

- Mit einer Wahrscheinlichkeit von 0.3 gibt es im Array ein Duplikat.
- Es gibt immer maximal ein Duplikat.
- Wenn ein Wert doppelt enthalten ist, so steht dieser Wert an der ersten Position im Array.
- Das zweite Auftauchen des Wertes ist an jeder (anderen) Position gleich wahrscheinlich.

Bestimmen Sie in Abhängigkeit von n ...

a) die exakte Anzahl der Vergleiche von Einträgen des Arrays im Best-Case.

b) die exakte Anzahl der Vergleiche von Einträgen des Arrays im Worst-Case.

c) die exakte Anzahl der Vergleiche von Einträgen des Arrays im Average-Case.

Lösung:

Wie in der Aufgabe gefordert zählen wir in dieser Aufgabe nur Vergleiche mit Einträgen Im Array, nicht aber der Vergleich, der in der der Schleifenbedingung stattfindet.

a) Best-case Analyse $B(n)$
Die kleinste Anzahl von Vergleichen wird erreicht, wenn die ersten beiden Positionen den gleichen Wert haben. Dann ist lediglich ein Vergleich nötig. Eine Ausnahme bildet der Fall, dass das Array leer ist oder nur ein einziges Element enthält ($n \leq 1$), dann findet kein einziger Vergleich statt. Somit ergibt sich:

$$B(n) = \begin{cases} 0 & ,falls\ n \leq 1 \\ 1 & ,sonst \end{cases}$$

b) Worst-case Analyse $W(n)$
Die meisten Vergleiche ergeben sich, wenn kein Duplikat im Array vorhanden sind. In diesem Fall wird die äussere Schleife vollständig durchlaufen (n-Mal) und auch die innere Schleife wird jeweils vollständig durchlaufen. Hierbei wird die innere Schleife beim i-ten Durchlauf der äusseren Schleife $(n-1-i)$-fach durchlaufen. In jedem Durchlauf der inneren Schleife findet ein Vergleich statt. Somit ergibt sich eine Gesamtanzahl von Vergleichen im Worst-case von:

$$W(n) = \sum_{i=0}^{n-1}(n-1-i)\cdot 1 = \sum_{i=0}^{n-1}(n-1) - \sum_{i=0}^{n-1} i = n\cdot(n-1) - \frac{(n-1)\cdot n}{2} = \frac{n\cdot(n-1)}{2}$$

c) Average-case Analyse $A(n)$
Die Average-case Vergleichsanzahl ergibt sich wie folgt aus der Anzahl der Vergleiche $A_{Duplikat}(n)$ für den Fall, dass ein Duplikat enthalten ist, sowie $A_{\neg Duplikat}(n)$ für den Fall, dass kein Duplikat enthalten ist:

$$A_{\neg Duplikat}(n)\cdot 0,7 + A_{Duplikat}(n)\cdot 0,3$$

Wie bereits in Aufgabenteil b) bestimmt, gilt $A_{\neg Duplikat}(n) = W(n) = \frac{n\cdot(n-1)}{2}$. Für den Fall, dass ein Wert x an zwei Positionen steht, steht x unter anderem an der ersten Position im Array. Das zweite Auftreten ist an jeder weiteren Position gleich wahrscheinlich, d.h. mit einer Wahrscheinlichkeit von $\frac{1}{n-1}$ steht der Wert x an Position $1 \leq i < n$. Um ein Duplikat an den Stellen 0 und i zu entdecken, wird die äußere Schleife einmal ausgeführt, die innere wird i-mal durchlaufen. Es ergibt sich die folgende Anzahl von Vergleichen für $A_{Duplikat}$:

$$\begin{aligned} A_{Duplikat}(n) &= \sum_{i=1}^{n-1} Pr\{E[i] = x \mid \exists j.\ 1 \leq j < n \wedge E[0] = E[j]\} \cdot t(E[0] = E[i]) \\ &= \sum_{i=1}^{n-1} \left(\frac{1}{n-1}\cdot i\right) \\ &= \frac{1}{n-1}\cdot\sum_{i=1}^{n-1} i \\ &= \frac{1}{n-1}\cdot\frac{(n-1)n}{2} \\ &= \frac{n}{2} \end{aligned}$$

Somit ergibt sich eine Average-case Vergleichszahl von:

$$A(n) = \frac{n(n-1)}{2} \cdot 0,7 + \frac{n}{2} \cdot 0,3$$

Aufgabe 5 (Asymptotische Komplexität): (5 · 3 Punkte)

Begründen oder widerlegen Sie die folgenden Aussagen:

a) $n^2 \in O(n^2 - 20n - 250)$.

b) $3n^3 + 12n^2 - 34n + 100 \in \mathcal{O}(n^3)$.

c) Aus $f(n) \in \Omega(g(n))$ und $f(n) \in \mathcal{O}(h(n))$ folgt $g \in \Theta(h(n))$.

d) Aus $f(n) \in \Theta(n)$ folgt $2^{f(n)} \in \mathcal{O}(2^n)$.

e) $\log_a(n) \in \Theta(\log_b(n))$ für zwei beliebige Konstanten $a, b > 1$.

Lösung:

(a) Im folgenden ist $p(n) = n^2 - 20n - 250$. Wir wählen einfach (zum Beispiel) $c = 2$. Jetzt müssen wir zeigen, dass es ein n_0 gibt, sodass $n^2 \leq cp(n)$ für alle $n > n_0$. Äquivalent dazu gilt

$$0 \leq cp(n) - n^2 = n^2 - 40n - 500$$

für alle $n > n_0$. Mithilfe der pq-Formel finden wir $n^2 - 40n - 500 = (n + 10)(n - 50)$. Das gesuchte n_0 ist also 50.

(b) Wir wählen $c = 3 + 12 + 100 = 115$ und $n_0 = 1$. Offensichtlich gilt

$$3n^3 + 12n^2 - 34n + 100 \leq 3n^3 + 12n^2 + 100 \leq 3n^3 + 12n^3 + 100n^3 = 115n^3$$

für $n \geq 1$ und die Aussage ist gezeigt.

(c) Stimmt nicht, wähle z.B. $f(n) = n$, $g(n) = 1$, $h(n) = n^2$. Es gilt $n \in \Omega(1)$ und $n \in \mathcal{O}(n^2)$ aber $1 \notin \Theta(n^2)$.

(d) Stimmt nicht, wähle z.B. $f(n) = 2n$. Dann ist aber $4^n \notin \mathcal{O}(2^n)$. ($\lim_{n\to\infty} \frac{2^n}{4^n} = \lim_{n\to\infty} (\frac{2}{4})^n = 0$).

(e) Die Aussage gilt. Zu zeigen ist:

$$\exists c_1, c_2 \in \mathbb{R}^{>0}, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N} \text{ mit } n \geq n_0 : c_1 \cdot \log_b(n) \leq \log_a(n) \leq c_2 \cdot \log_b(n)$$

Aus den Logarithmengesetzen folgt $\log_a(n) = \frac{\log_b(n)}{\log_b(a)}$. Wähle also $c_1 = c_2 = \frac{1}{\log_b(a)}$. Dies ist möglich, da $a, b > 1$ gilt und damit auch $\frac{1}{\log_b(a)} > 0$. Damit erhalten wir

$$c_1 \cdot \log_b(n) = \log_a(n) = c_2 \cdot \log_b(n)$$

und damit ist die Aussage bewiesen.

Aufgabe 6 (Beweis): (3+6 Punkte)

Wir wollen beweisen, dass $n!$ und n^n nicht dieselbe Komplexität haben, also $n! \notin \Theta(n^n)$. Führen sie dazu folgende Schritte aus.

a) Zeigen Sie zunächst, dass $n! \in o(n^n)$.
Hinweis: es gilt allgemein $f(n) \in o(g(n))$, wenn $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

b) Zeigen Sie nun, dass für eine beliebige Funktion g gilt: $o(g(n)) \cap \Theta(g(n)) = \emptyset$.
Hinweis: es bietet sich ein Widerspruchsbeweis an.
Folgern Sie schließlich die oben gewünschte Aussage.

Lösung:

a) Wir stellen fest, dass

$$\frac{n!}{n^n} = \frac{n}{n} \cdot \frac{n-1}{n} \cdots \frac{1}{n} \leq \frac{1}{n} \ .$$

Die Ungleichung gilt, da jeder Faktor nicht größer als 1 ist. Daher gilt für alle n

$$0 \leq \frac{n!}{n^n} \leq \frac{1}{n} \ .$$

Das "Sandwichlemma" aus Afl[1] erlaubt die Schlussfolgerung:

$$\text{Aus} \quad \lim_{n\to\infty} 0 = 0 = \lim_{n\to\infty} \frac{1}{n} \quad \text{folgt} \quad \lim_{n\to\infty} \frac{n!}{n^n} = 0 \ .$$

b) Per Definition gilt

$$\Theta(g(n)) = \{f(n) \mid \text{es gibt positive Konstanten } c_1, c_2, n_0 \text{ sodass für alle } n \geq n_0 \text{ gilt } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

und

$$o(g(n)) = \{f(n) \mid \text{für alle } c > 0 \text{ existiert ein } n_0 \text{ sodass für alle } n \geq n_0 \text{ gilt } 0 \leq f(n) < cg(n)\} \ .$$

Zum Zwecke des Widerspruchs nehmen wir an es gäbe eine Funktion f^*, die im Schnitt der beiden Mengen liegt. Dann gibt es auch Konstanten c_1^* und n_0^*, sodass

$$c_1^* g(n) \leq f^*(n) \quad \text{für } n \geq n_0^* \ .$$

Gleichzeitig muss es für c_1^* ein $\tilde{n_0}$ geben, sodass

$$f^*(n) < c_1^* g(n) \quad \text{für } n \geq \tilde{n_0} \ .$$

Somit gibt es ein n', das größer als n_0^* und $\tilde{n_0}$ ist, für das gilt

$$c_1^* g(n') \leq f^*(n') < c_1^* g(n') \ .$$

Das ist ein Widerspruch! Damit wissen wir, dass $n! \in o(n^n)$ und da $o(n^n)$ und $\Theta(n^n)$ keine gemeinsamen Funktionen besitzen folgt die gewünschte Aussage $n! \notin \Theta(n^n)$.

[1] Der Einschnürungssatz, Einschließungssatz, Dreifolgensatz oder Sandwichsatz (u. a.: Schachtelungssatz, Quetschkriterium resp. Satz von den zwei Polizisten; englisch sandwich theorem) ist in der Analysis ein Satz über den Grenzwert einer Funktion. Gemäß dem Einschnürungssatz strebt eine Funktion, die von oben und unten durch zwei gegen denselben Wert strebenden Funktionen "eingezwängt" wird, auch gegen diesen Wert. [http://de.wikipedia.org/wiki/Einschn%C3%BCrungssatz]

Aufgabe 7 (Programmanalyse): (2+2+3 Punkte)

Gegeben sei ein Algorithmus zur Suche eines Modus (einer am häufigsten vorkommenden Zahl) in einem Array:

```
int findModus(int [] E) {
  int i = 0;
  int [] counter = {0,0,0,0}; // Zaehlt Vorkommen der Zahlen 0 bis 3
  boolean flag = false;
  int haelfte = (int)Math.ceil(E.length/2.0); // Bei ungerader Laenge
                                              // des Arrays runden wir auf
  while(!flag && i < E.length) {
    int number = E[i];
    counter[number]++;
    if(counter[number] >= haelfte) {
      flag = true;
    }
    i++;
  }

  return getMaxCount(counter); // Extrahiert einen Modus aus dem Zaehler in
                               // konstanter Zeit
}
```

Er erhält ein Array E von Zahlen **zwischen 0 und 3** und findet heraus welche dieser vier Zahlen am häufigsten in E vorkommt. Bei Betrachtung der Laufzeit vernachlässigen wir die Initialisierung der Variablen sowie den Aufruf von `getMaxCount(counter)`, da diese Operationen unabhängig von der Eingabe E immer konstante Laufzeit haben. Wir interessieren uns lediglich für die **Anzahl der Schleifendurchläufe**, von denen jeder einzelne Durchlauf eine Zeiteinheit kostet. Sei n die Länge des Arrays E. Bestimmen Sie in Abhängigkeit von n...

a) die Anzahl der Schleifendurchläufe im Best-case.

b) die Anzahl der Schleifendurchläufe im Worst-case.

c) die Anzahl der Schleifendurchläufe im Average-case. Hierzu nehmen wir an, dass in E eine 1 vorkommt und alle anderen Einträge gleich 3 sind. Die 1 kann jedoch an jeder der n Positionen gleich wahrscheinlich auftreten.

Lösung:

a) Im besten Fall kommt in den ersten $\lceil \frac{n}{2} \rceil$ Positionen die gleiche Zahl vor und die Schleife bricht nach ebenso vielen Iterationen ab, da der Modus unabhängig von den restlichen Einträgen bereits feststeht.

b) Im schlechtesten Fall muss das gesamte Array durchlaufen werden. Es gibt dann also genau n Iterationen.

c) Der Modus ist offensichtlich 3, die Frage ist lediglich nach wie vielen Schritten wir das feststellen werden. Es gibt zwei Fälle, entweder kommt die 1 in der zweiten Hälfte des Arrays vor und wir haben die Best-case Laufzeit oder die 1 kommt in der ersten Hälfte vor und wir brauchen eine Iteration mehr. Genauer gilt für gerade n

$$A(n) = \frac{1}{2} \cdot \frac{n}{2} + \frac{1}{2} \cdot \left(\frac{n}{2} + 1\right) = \frac{n+1}{2} \ .$$

Falls n ungerade ist erhalten wir analog

$$\begin{aligned} A(n) &= \frac{1}{n} \cdot \frac{n+1}{2} \cdot \frac{n+3}{2} + \frac{1}{n} \cdot \frac{n-1}{2} \cdot \frac{n+1}{2} \\ &= \frac{n^2+4n+3}{4n} + \frac{n^2-1}{4n} = \frac{2n^2+4n+2}{4n} = \frac{n}{2} + 1 + \frac{1}{2n} \ . \end{aligned}$$