Algorithmen und Datenstrukturen

Übungsgruppe 14

Utz Pöhlmann
4poehlma@informatik.uni-hamburg.de
6663579

Louis Kobras
4kobras@informatik.uni-hamburg.de
6658699

Paul Testa
paul.testa@gmx.de
6251548

8. November 2015

Punkte für den Hausaufgabenteil:

2.1	2.2	2.3	2.4	Σ

1 Zettel vom 14.-16. Oktober – Abgabe: N/A

1.1 Präsenzaufgabe 1.1

Wiederholen Sie die O-Notation und die verwandten Notationen. Wie sind die einzelnen Mengen definiert? Was bedeutet es, wenn $f \in O(g)$ gilt, was wenn $f \in \Theta(g)$ gilt und so weiter?

$$\begin{array}{lll l}
O(g(n)): & f(n) \in O(g(n)) & \Leftrightarrow \exists c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n >= n_0: & \|f(n)\| <= c \cdot \|g(n)\| \\
o(g(n)): & f(n) \in o(g(n)) & \Leftrightarrow \forall c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n >= n_0: & \|f(n)\| <= c \cdot \|g(n)\| \\
\Omega(g(n)): & f(n) \in \Omega(g(n)) & \Leftrightarrow \exists c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n >= n_0: & \|f(n)\| >= c \cdot \|g(n)\| \\
\omega(g(n)): & f(n) \in \omega(g(n)) & \Leftrightarrow \forall c \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n >= n_0: & \|f(n)\| >= c \cdot \|g(n)\| \\
\Theta(g(n)): & f(n) \in \Theta(g(n)) & \Leftrightarrow \exists c_1, c_2 \in \mathbb{R}^+ \exists n_0 \in \mathbb{N} \forall n >= n_0: & c_1 \cdot \|g(n)\| <= \|f(n)\| <= c_2 \cdot \|g(n)\|
\end{array}$$

1.2 Präsenzaufgabe 1.2

Beweisen Sie:

- $n^2 + 3n - 5 \in O(n^2)$
- $n^2 - 2n \in \Theta(n^2)$
- $n! \in O((n+1)!)$

Gilt im letzten Fall auch $n! \in o((n+1)!)$?

$$\begin{aligned}
f(n) \in O(g(n)) &\Leftrightarrow lim_{n\to\infty} \tfrac{f(n)}{g(n)} < \infty \\
f(n) &= n^2 + 3n - 5 \\
g(n) &= n^2 \\
\tfrac{f(n)}{g(n)} &= \tfrac{n^2+3n-5}{n^2} \\
\\
lim_{n\to\infty} \tfrac{n^2+3n-5}{n^2} &= lim_{n\to\infty} 1 + \tfrac{3}{n} - \tfrac{5}{n^2} \\
&= 1 + \tfrac{3}{\infty} - \tfrac{5}{\infty^2} \\
&= 1 + 0 + 0 \\
&= 1 < \infty \Rightarrow f(n) \in O(g(n))
\end{aligned}$$

$\square$

$$\begin{aligned}
c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N} \forall n >= n_0: \quad & c_1 \cdot n^2 <= n^2 - 2n <= c_2 \cdot n^2 \\
\Leftrightarrow \quad & c_1 <= 1 - \tfrac{1}{n} <= c_2
\end{aligned}$$

Dies ist erfüllbar ab $n_0 >= 2$, da für $n = 1$ im mittleren Ausdruck 0 herauskommt und c_1 größer als 0, aber kleiner als der mittlere Ausdruck sein muss. Ist $n >= 2$, so kommt im mittleren Ausdruck $0,5$ heraus, für c_1 lässt sich ein beliebiger Wert aus $]0; 0.5[$ wählen, sei es an dieser Stelle $\frac{1}{4}$. Als Obergrenze für c_2 lässt sich jeder Wert größer oder gleich 1 wählen, da der mittlere Ausdruck nicht größer als 1 werden kann und somit die Bedingung des "kleiner gleichßofort erfüllt ist.
Somit wird als Ergebnis für die Belegung gewählt: $c_1 = \frac{1}{4}; c_2 = 1; n_0 = 2$. Mit dieser Belegung gilt $n^2 - 2n \in \Theta(n^2)$

$\square$

$$
\begin{aligned}
f(n) \in O(g(n)) & \Leftrightarrow lim_{n\to\infty} \tfrac{f(n)}{g(n)} < \infty \\
f(n) & = n! \\
g(n) & = (n+1)! = (n+1) \cdot n! \\
\\
lim_{n\to\infty} \tfrac{n!}{(n+1)\cdot n!} & = lim_{n\to\infty} \tfrac{1}{n+1} \\
& = \tfrac{1}{\infty} \\
& = 0 < \infty \Rightarrow f(n) \in O(g(n))
\end{aligned}
$$

Da die Bedingung für $o(g(n))$ ist, dass der Quotient nicht nur kleiner unendlich, sondern gleich null ist, was hier wie oben gezeigt gegeben ist, gilt auch $n! \in o((n+1)!)$.

□

1.3 Präsenzaufgabe 1.3

Beweisen oder widerlegen Sie:

1. $f(n), g(n) \in O(h(n)) \Rightarrow f(n) + g(n) \in O(h(n))$
2. $f(n), g(n) \in O(h(n)) \Rightarrow f(n) \cdot g(n) \in O(h(n))$

$$\exists c_1 \in \mathbb{R}^+ \exists n_{0_1} \in \mathbb{N} \forall n >= n_{0_1} : \|f(n)\| <= c_1 \cdot \|h(n)\|$$

$$\exists c21 \in \mathbb{R}^+ \exists n_{0_2} \in \mathbb{N} \forall n >= n_{0_2} : \|g(n)\| <= c_2 \cdot \|h(n)\|$$

$$n_0 = max(n_{0_1}, n_{0_2})$$

$$\|f(n) + g(n)\| <= c_1 \cdot \|h(n)\| + c_2 \cdot \|h(n)\| <= (c_1 + c_2) \cdot \|h(n)\|$$

Seien $f(n)$ und $g(n)$ Polynome zweiten Grades sowie $h(n)$ ein Polynom dritten Grades. Dann sind sowohl $f(n)$ als auch $g(n)$ durch die *limes*-Bedingung in $O(h(n))$. Das Produkt zweier Polynome zweiten Grades ist allerdings ein Polynom vierten Grades, sodass gilt:

$$\lim_{n\to\infty} \frac{n^2 \cdot n^2}{n^3} = \lim_{n\to\infty} \frac{n^4}{n^3} = \lim_{n\to\infty} n = \infty$$

Damit ist das Produkt der Polynome nicht mehr in $O(h(n))$, da die *limes*-Bedingung, nach der der Quotient der Polynome für n gegen Unendlich kleiner als Unendlich sein zu hat, nicht erfüllt ist. Damit ist (2) widerlegt.

□

2 Zettel vom 15.10. – Abgabe: 26.10.

2.1 Übungsaufgabe 2.1

[| 2]

Begründen Sie formal, warum folgende Größenabschätzungen gelten bzw. nicht gelten:

1. $3n^3 - 6n + 20 \in O(n^3)$
2. $n^2 \cdot \log n \in O(n^3) \cap \Omega(n^2)$

2.1.1

$$3n^3 - 6n + 20 \in O(n^3) \Leftrightarrow lim_{n\to\infty} \frac{3n^3 - 6n + 20}{n^3} < \infty$$

$$lim_{n\to\infty} \frac{3n^3 - 6n + 20}{n^3} = lim_{n\to\infty} \frac{3n^3}{n^3} - \frac{6n}{n^3} + \frac{20}{n^3} = lim_{n\to\infty} 3 - \frac{6}{n^2} + \frac{20}{n^3} = 3 - 0 + 0 < \infty$$

$$\Rightarrow 3n^3 - 6n + 20 \in O(n^3) \qquad \square$$

2.1.2

$$n^2 \cdot \log n \in O(n^3) \cap \Omega(n^2) \Leftrightarrow lim_{n\to\infty} \frac{n^2 \cdot \log n}{n^3} < \infty \wedge lim_{n\to\infty} \frac{n^2 \cdot \log n}{n^2} > 0$$

$$\frac{n^2 \cdot \log n}{n^2} = \frac{1 \cdot \log n}{1} = \log n > 0 \ \forall n > 1 \Rightarrow n^2 \cdot \log n \in \Omega(n^2)$$

$$lim_{n\to\infty} \frac{n^2 \cdot \log n}{n^3} = lim_{n\to\infty} \frac{\log n}{n} \overset{\text{l'H}}{=} lim_{n\to\infty} \frac{1}{n} \cdot \frac{1}{1} = lim_{n\to\infty} \frac{1}{n} = \frac{1}{\infty} = 0 \Rightarrow n^2 \cdot \log n \in O(n^3)$$

$$\Rightarrow n^2 \cdot \log n \in O(n^3) \cap \Omega(n^2) \qquad \square$$

2.2 Übungsaufgabe 2.2

[| 4]

Ordnen Sie die folgenden Funktionen nach ihrem Wachstumsgrad in aufsteigender Reihenfolge, d.h. folgt eine Funktion $g(n)$ einer Funktion $f(n)$, so soll $f(n) \in O(g(n))$ gelten.

$$n, \log n, n^2, n^{\frac{1}{2}}, \sqrt{n}^3, 2^n, \ln n, 1000$$

Mit log ist hier der Logarithmus zur Basis 2, mit ln der natürliche Logarithmus (Basis e) gemeint. Begründen Sie stets Ihre Aussage. Zwei Funktionen $f(n)$ und $g(n)$ befinden sich ferner in der selben Äquivalenzklasse, wenn $f(n) \in \Theta(g(n))$ gilt. Geben Sie an, welche Funktionen sich in derselben Äquivalenzklasse befinden und begründen Sie auch hier ihre Aussage.

Die bearbeitete Menge wird i.F. als M_F bezeichnet. Die Menge, die gerade alle Elemente von M_F in aufsteigend sortierter Reihenfolge enthält, wird als M'_F bezeichnet.
M_F wird mit InsertionSort in M'_F hineinsortiert.
Sei $e \in M_F$. Für e wird das Element 1000 gewählt. Da $|M'_F|$ leer ist, muss 1000 nicht weiter geprüft werden.

$$M'_F = \{1000\}$$

e wird nun über M_F iteriert, bis $M'_F = Sorted(M_F)$.

$e = n$

$$f(n) = n$$
$$g(n) = 1000 \quad \lim_{n\to\infty} \frac{n}{1000} = \infty \quad \Rightarrow n \notin O(1000)$$

$$f(n) = 1000$$
$$g(n) = n \quad \lim_{n\to\infty} \frac{1000}{n} = 0 \quad \Rightarrow 1000 \in O(n) \quad \Rightarrow n \text{ folgt } 1000$$

$$M'_F = \{1000, n\}$$

$e = \log n$

$$f(n) = log(n)$$
$$g(n) = n \quad \lim_{n\to\infty} \frac{log(n)}{n} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\frac{1}{ln(2)\cdot n}}{1} = \lim_{n\to\infty} \frac{1}{ln(2)\cdot n} = 0 \quad \Rightarrow log(n) \in O(n) \quad \Rightarrow n \text{ folgt } log(n)$$

$$f(n) = log(n)$$
$$g(n) = 1000 \quad \lim_{n\to\infty} \frac{log(n)}{1000} = \infty \quad \Rightarrow log(n) \notin O(1000)$$

$$M'_F = \{1000, \log n, n\}$$

$e = 4$

$$f(n) = 4$$
$$g(n) = n \quad \lim_{n\to\infty} \frac{4}{n} = 0 \quad \Rightarrow 4 \in O(n) \quad \Rightarrow n \text{ folgt } 4$$

$$f(n) = 4$$
$$g(n) = log(n) \quad \lim_{n\to\infty} \frac{4}{log(n)} = 0 \quad \Rightarrow 4 \in O(log(n)) \quad \Rightarrow log(n) \text{ folgt } 4$$

$$f(n) = 4$$
$$g(n) = 1000 \quad \lim_{n\to\infty} \frac{4}{1000} = 0,004 \quad \Rightarrow 4 \in \Theta(1000) \quad \Rightarrow 4 \text{ und } 1000 \text{ befinden sich in der selben Ä.-klasse}$$

$$M'_F = \{4, 1000, \log n, n\}$$

$e = n^2$

$$f(n) = n^2$$
$$g(n) = n \quad \lim_{n\to\infty} \frac{n^2}{n} = \infty \quad \Rightarrow n^2 \notin O(n)$$

$$f(n) = n$$
$$g(n) = n^2 \quad \lim_{n\to\infty} \frac{n}{n^2} = 0 \quad \Rightarrow n \in O(n^2) \quad \Rightarrow n^2 \text{ folgt } n$$

$$M'_F = \{4, 1000, \log n, n, n^2\}$$

$e = n^{\frac{1}{2}}$

$$f(n) = n^{\frac{1}{2}}$$
$$g(n) = n^2 \qquad \lim_{n\to\infty} \frac{n^{\frac{1}{2}}}{n^2} = \lim_{n\to\infty} \frac{1}{n^{\frac{3}{2}}} \qquad = 0 \qquad \Rightarrow n^{\frac{1}{2}} \in O(n^2) \qquad \Rightarrow n^2 \text{ folgt } n^{\frac{1}{2}}$$

$$f(n) = n^{\frac{1}{2}}$$
$$g(n) = n \qquad \lim_{n\to\infty} \frac{n^{\frac{1}{2}}}{n} = \lim_{n\to\infty} \frac{1}{n^{\frac{1}{2}}} \qquad = 0 \qquad \Rightarrow n^{\frac{1}{2}} \in O(n) \qquad \Rightarrow n \text{ folgt } n^{\frac{1}{2}}$$

$$f(n) = n^{\frac{1}{2}}$$
$$g(n) = log(n) \qquad \lim_{n\to\infty} \frac{n^{\frac{1}{2}}}{log(n)} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\frac{1}{2}\cdot n^{-\frac{1}{2}}}{\frac{1}{ln(2)\cdot n}} = \lim_{n\to\infty} \frac{ln(2)\cdot n^{\frac{1}{2}}}{2} \qquad = \infty \qquad \Rightarrow n^{\frac{1}{2}} \notin O(log(n))$$

$$M'_F = \{4, 1000, \log n, n^{\frac{1}{2}}, n, n^2\}$$

$e = \sqrt{n}^3$

$$f(n) = \sqrt{n}^3$$
$$g(n) = n^2 \qquad \lim_{n\to\infty} \frac{\sqrt{n}^3}{n^2} = \lim_{n\to\infty} \frac{1}{n^{\frac{1}{2}}} \qquad = 0 \qquad \Rightarrow \sqrt{n}^3 \in O(n^2) \qquad \Rightarrow n^2 \text{ folgt } \sqrt{n}^3$$

$$f(n) = \sqrt{n}^3$$
$$g(n) = n \qquad \lim_{n\to\infty} \frac{\sqrt{n}^3}{n} = \lim_{n\to\infty} n^{\frac{1}{2}} \qquad = \infty \qquad \Rightarrow \sqrt{n}^3 \notin O(n)$$

$$M'_F = \{4, 1000, \log n, n^{\frac{1}{2}}, n, \sqrt{n}^3, n^2\}$$

$e = 2^n$

$$f(n) = 2^n$$
$$g(n) = n^2 \qquad \lim_{n\to\infty} \frac{2^n}{n^2} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{ln(2)\cdot 2^n}{2\cdot n} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{ln^2(2)\cdot 2^n}{2} \qquad = \infty \qquad \Rightarrow 2^n \notin O(n^2)$$

$$f(n) = n^2$$
$$g(n) = 2^n \qquad \lim_{n\to\infty} \frac{n^2}{2^n} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{2\cdot n}{ln(2)\cdot 2^n} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{2}{ln^2(2)\cdot 2^n} \qquad = 0 \qquad \Rightarrow n^2 \in O(2^n) \qquad \Rightarrow n^2 \text{ folgt } 2^n$$

$$M'_F = \{4, 1000, \log n, n^{\frac{1}{2}}, n, \sqrt{n}^3, n^2, 2^n\}$$

$e = \ln n$

$$f(n) = ln(n) \qquad g(n) = 2^n \qquad \lim_{n\to\infty} \frac{ln(n)}{2^n} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{ln(2)\cdot 2^n} = \lim_{n\to\infty} \frac{1}{n\cdot ln(n)\cdot 2^n} = 0 \quad \Rightarrow ln(n) \in O(2^n) \quad \Rightarrow 2^n \text{ folgt } ln(n)$$

$$f(n) = ln(n) \qquad g(n) = n^2 \qquad \lim_{n\to\infty} \frac{ln(n)}{n^2} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{2\cdot n} = \lim_{n\to\infty} \frac{1}{2\cdot n^2} = 0 \quad \Rightarrow ln(n) \in O(n^2) \quad \Rightarrow n^2 \text{ folgt } ln(n)$$

$$f(n) = ln(n) \qquad g(n) = \sqrt{n}^3 \qquad \lim_{n\to\infty} \frac{ln(n)}{\sqrt{n}^3} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\sqrt[3]{n}}{2\cdot n} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{1}{6\cdot n^{\frac{2}{3}}} = 0 \quad \Rightarrow ln(n) \in O(\sqrt{n}^3) \quad \Rightarrow \sqrt{n}^3 \text{ folgt } ln(n)$$

$$f(n) = ln(n) \qquad g(n) = n \qquad \lim_{n\to\infty} \frac{ln(n)}{n} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{1} = \lim_{n\to\infty} \frac{1}{n} = 0 \quad \Rightarrow ln(n) \in O(n) \quad \Rightarrow n \text{ folgt } ln(n)$$

$$f(n) = ln(n) \qquad g(n) = n^{\frac{1}{2}} \qquad \lim_{n\to\infty} \frac{ln(n)}{n^{\frac{1}{2}}} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\cdot n^{\frac{1}{2}}}} = \lim_{n\to\infty} \frac{2}{n^{\frac{1}{2}}} = 0 \quad \Rightarrow ln(n) \in O(n^{\frac{1}{2}}) \quad \Rightarrow n^{\frac{1}{2}} \text{ folgt } ln(n)$$

$$f(n) = ln(n) \qquad g(n) = log(n) \qquad \lim_{n\to\infty} \frac{ln(n)}{log(n)} \overset{\text{L'Hospital}}{=} \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{ln(2)\cdot n}} = ln(2) \quad \Rightarrow ln(n) \in \Theta(log(n)) \quad \Rightarrow ln(n) \text{ und } log(n)$$

befinden sich in der selben Ä.-klasse

$$M'_F = \{4, 1000, \ln n, \log n, n^{\frac{1}{2}}, n, \sqrt{n}^3, n^2, 2^n\}$$

In der selben Äquivalenzklasse befinden sich zum einen 4 und 1000 und zum anderen $log(n)$ und $ln(n)$. Die restlichen Werte sind jeweils alleine in ihrer Äquivalenzklasse.

2.3 Übungsaufgabe 2.3

[| 2]

Beweisen oder widerlegen Sie:

$$f(n), g(n) \in O(h(n)) \Rightarrow f(n) \cdot g(n) \in O((h(n))^2)$$

Für diesen Beweis wird der Beweis des dritten Satzes der Summen- und Produkteigenschaften der O-Notation[1] zu Hilfe genommen:

Beweis. Sei $f \in O(h_1)$ und $g \in O(h_2)$, dann gibt es ein c, n_0, so dass $f(n) \leq c \cdot h_1(n) \forall n \geq n_0$ und ebenso c', n'_0, so dass $g(n') \leq c' \cdot h_2(n') \forall n' \geq n'_0$. Daraus folgt $f(n'') \cdot g(n'') \leq c \cdot c' \cdot h_1(n'') \cdot h_2(n'') \forall n'' \geq max(n_0, n'_0)$, also $f \cdot g \in O(h_1 \cdot h_2)$. □

Setzt man nun $h_1, h_2 = h$ folgt daraus für den letzten Ausdruck des Beweises $f(n) \cdot g(n) \in O(h(n) \cdot h(n)) \Rightarrow f(n) \cdot g(n) \in O((h(n))^2)$.

[1]vgl. Vorlesung, Foliensatz 1 (14.10.), S.33

2.4 Übungsaufgabe 2.4

[| 8]

Seien

1. $$T(n) := \begin{cases} 0, & \text{für } n = 0 \\ 3 \cdot T(n-1) + 2, & \text{sonst} \end{cases}$$

2. $$S(n) := \begin{cases} c, & \text{für } n = 1 \\ 16 \cdot S(\frac{n}{4}) + n^2, & \text{sonst} \end{cases}$$

Rekurrenzgleichungen (c ist dabei eine Konstante).

Bestimmen Sie wie in der Vorlesung jeweils die Größenordnung der Funktion $T : \mathbb{N} \to \mathbb{N}$ einmals mittels der (a) Substitutionsmethode und einmal mittes des (b) Mastertheorems. Ihre Ergebnisse sollten zumindest hinsichtlich der O-Notation gleich sein, so dass Sie etwaige Rechenfehler entdecken können! Führen Sie bei (a) auch den Induktionsbeweis, der in der Vorlesung übersprungen wurde!

1. a)

$$\begin{array}{rcll} T(n) & = & 3 \cdot T(n-1) + 2 & \\ & = & 3 * (3 * T(n-2) + 2) + 2 & = 3^2 * T(n-2) + 3^2 - 1 \\ & = & 3^2 * (3 * T(n-3) + 2) + 8 & = 3^3 * T(n-3) + 3^3 - 1 \\ & = & ... & \\ & = & 3^k * T(n-k) + 3^k - 1 & \end{array}$$

Wir kommen auf eine sinnvolle Verallgemeinerung der Formel.
Beweis der Formel durch vollständige Induktion:

Induktionsanfang: $T(0)$ gilt nach Definiton.
Induktionsschritt: Sei $n \in \mathbb{N}$ (s. Aufgabenstellung). Wir nehmen an, dass $T(n)$ gilt (Induktionsannahme) und zeigen $T(n+1)$. Es gilt

$$\begin{array}{rcl} T(n) & = & 3 * T(n-1) + 2 \\ T(n+1) & = & 3 * T(n+1-1) + 2 \\ & = & 3 * T(n) + 2 \end{array}$$

$$\begin{array}{rcl} T(n) & = & 3^k * T(n-k) + 3^k - 1 \\ T(n+1) & = & 3^k * T(n+1-k) + 3^k - 1 \end{array}$$

Das zeigt $T(n+1)$.
Damit sind der Induktionsanfang und der Induktionsschritt bewiesen. Es folgt, dass $T(n)$ für alle $n \in \mathbb{N}$ gilt.

Da die Rekursion bei $T(0) = 0$, also $n - k = 0$ abbricht, wird mit $k = n$ weiter gerechnet.

$$\begin{array}{rcl} T(n) & = & 3^k * T(n-k) + 3^k - 1 \\ & = & 3^n * T(n-n) + 3^n - 1 \\ & = & 3^n * T(0) + 3^n - 1 \\ & = & 3^n * 0 + 3^n - 1 \\ & = & 3^n - 1 \in \Theta(3^n) \end{array}$$

b) Das Mastertheorem ist auf Aufgabe 1. nicht anwendbar, da die Form

$$T(n) := \begin{cases} c, & \text{falls } n = 1 \\ a \cdot T(\frac{n}{b}) + f(n), & \text{falls } n > 1 \end{cases}$$

bei

$$T(n) := \begin{cases} 0, & \text{für } n = 0 \\ 3 \cdot T(n-1) + 2, & \text{sonst} \end{cases}$$

nicht eingehalten wurde.

2. a)

$$\begin{aligned} S(n) &= 16 \cdot S(\tfrac{n}{4}) + n^2 \\ &= 16 \cdot S(\tfrac{16 \cdot S(\frac{n}{4}) + n^2}{4}) + n^2 \\ &= 16 \cdot S(\tfrac{16 \cdot S(\frac{16 \cdot S(\frac{n}{4}) + n^2}{4}) + n^2}{4}) + n^2 \\ &= 16 \cdot S(\tfrac{16 \cdot S(\frac{16 \cdot S(\frac{16 \cdot S(\frac{n}{4}) + n^2}{4}) + n^2}{4}) + n^2}{4}) + n^2 \\ &= 16 \cdot S(\tfrac{16 \cdot S(\frac{16 \cdot S(\frac{16 \cdot S(\frac{16 \cdot S(\frac{n}{4}) + n^2}{4}) + n^2}{4}) + n^2}{4}) + n^2}{4}) + n^2 \end{aligned}$$

Keine sinnvolle Vereinfachung erkennbar. => Substitutionsmethode nicht anwendbar.

b) Die Form

$$S(n) := \begin{cases} c, & \text{falls } n = 1 \\ a \cdot T(\frac{n}{b}) + f(n), & \text{falls } n > 1 \end{cases}$$

ist bei

$$S(n) := \begin{cases} c, & \text{für } n = 1 \\ 16 \cdot S(\frac{n}{4}) + n^2, & \text{sonst} \end{cases}$$

eingehalten. Das Mastertheorem ist daher anwendbar.

I. $S(n) \in \Theta(n^{log_b(a)})$, falls $f(n) \in O(n^{log_b(a)-\epsilon})$ für ein $\epsilon > 0$.

$$\begin{aligned} f(n) &\in O(n^{log_b(a)-\epsilon}) \\ n^2 &\in O(n^{log_4(16)-\epsilon}) \\ n^2 &\in O(n^{2-\epsilon}) \end{aligned}$$

Hierfür kann kein ϵ gefunden werden. Daher gilt diese Aussage nicht

II. $S(n) \in \Theta(n^{log_b(a)} \cdot log_2(n))$, falls $f(n) \in \Theta(n^{log_b(a)})$.

$$\begin{aligned} f(n) &\in O(n^{log_b(a)}) \\ n^2 &\in O(n^{log_4(16)}) \\ n^2 &\in O(n^2) \end{aligned}$$

Dies stimmt, daher gilt diese Aussage.

III. $S(n) \in \Theta(f(n))$, falls $f(n) \in \Omega(n^{log_b(a)+\epsilon})$ für ein $\epsilon > 0$ **und** $a \cdot f(\frac{n}{b}) \leq \delta \cdot f(n)$ für ein $\delta < 1$ und große n.

$$
\begin{aligned}
f(n) &\in \Omega(n^{log_b(a)+\epsilon}) \\
n^2 &\in \Omega(n^{log_4(16)+\epsilon}) \\
n^2 &\in \Omega(n^{2+\epsilon})
\end{aligned}
$$

Dies stimmt für alle $\epsilon \geq 0$, also auch für mindestens ein $\epsilon > 0$.

$a \cdot f(\frac{n}{b}) \leq \delta \cdot f(n)$	\einsetzen
$16 \cdot (\frac{n}{4})^2 \leq \delta \cdot n^2$	\\$\sqrt{()}$
$4 \cdot \frac{n}{4} \leq \sqrt{\delta} \cdot n$	
$n \leq \sqrt{\delta} \cdot n$	\:n (n ist immer positiv, da $n \in \mathbb{N}$, s. Aufgabenstellung)
$1 \leq \sqrt{\delta}$	\\$()^2$
$1 \leq \delta$	

Damit ist $\delta \geq 1$ und nicht, wie benötigt, $\delta < 1$. Daher gilt diese Aussage nicht.

Da nur II. gilt, gilt $S(n) \in \Theta(n^{log_b(a)} \cdot log_2(n))$, also $S(n) \in \Theta(n^2 \cdot log_2(n))$.

3 Zettel vom 29. 10. – Abgabe: 09. 11.

3.1 Übungsaufgabe 3.1

[| 3]

Gegeben seien die folgenden Code-Fragmente. Geben Sie eine möglichst dichte asymptotische obere Schranke für die Laufzeit der einzelnen Code-Fragmente jeweils in Abhängigkeit von n an. Begründen Sie Ihre Behauptung. (Es geht hier nicht um die Bedeutung des Codes, nur um die Laufzeit. An der Stelle von $sum = sum+j$ könnte sinnvoller(er) Code stehen. Die Einrückung gibt den Skopus der Schleifenkonstrukte an.)

```
Algo1()
1  for i = 0 to n
2      for j = n downto 1
3          sum = sum + j
4      for j = 1 to n
5          sum = sum + j
```

```
Algo2()
1  i = 1
2  while i < 2 · n
3      for j = 1 to i
4          sum = sum + j
5      i = i + 2
```

```
Algo3()
1  i = 1
2  while i · i < n
3      i = i + 1
4      j = n
5      while j > 1
6          sum = sum + j
7          j = j/2
```

1. Der Algorithmus läuft in $O(n^2)$.
   Bei beiden inneren **for**-Blöcke laufen jeweils $n-1$ mal durch, sodass der innere Block insgesamt $2\cdot(n-1) = 2n-2$ mal ausgeführt wird. Die äußere **for**-Schleife läuft gerade n mal; also wird der Block der Länge $2n-2$ n mal ausgeführt. Dies führt zu einer Laufzeit von $n\cdot(2n-2) = 2n^2-2n \in O(n^2)$.

2. Der Algorithmus läuft in $O(n)$.
   Zeile 1 läuft in c, Zeile 2 und 3 sind zusammen $2n$, Zeile 4 und Zeile 5 sind jeweils wieder konstant. Das addiert gibt $2*n+c$ und das liegt in $O(2n)$ und genauer in $O(n)$, also in Linearzeit.

3. Der Algorithmus läuft in $O(\log(n))$.
   Zeile 1 läuft mit konstantem Zeitaufwand und auch nur einmal. Zeile 2 ist durch das i^2 logarithmisch. Zeile 3 und Zeile 4 sind wieder konstant. Zeile 5 - 7 laufen in $\log(n)$. Addiert ergibt das $2\log(n)+c$, also $O(\log(n))$.

3.2 Übungsaufgabe 3.2

[| 2]

$$A(n) := \begin{cases} 5, & \text{falls } n < 4 \\ A(\frac{n}{2}) + A(\frac{n}{4}) + 2n + 4, & \text{sonst} \end{cases}$$

Die 5 im ersten Teil der Rekurrenzgleichung erklärt sich durch die ersten beiden Zeilen im Pseudocode.

Der Aufruf in Zeile 8 dauert $\frac{n}{2}$, der in Zeile 9 dauert $\frac{n}{4}$. Dadurch erklärt sich das $A(\frac{n}{2}) + A(\frac{n}{4})$ im zweiten Teil der Rekurrenzgleichung.

Zeile 5 und 6 laufen in n ab.
Zeile 11 und 12 laufen auch in n ab.
Zeile 4, 7, 10 und 13 laufen jeweils mit konstantem Zeitaufwand.
Deshalb ist das, was hinter den $A(x)$-Aufrufen steht, $n + n + c$, wobei c in diesem Fall den Wert 4 hat, da wir vier Zeilen mit konstantem Zeitaufwand im Pseudocode haben.

3.3 Übungsaufgabe 3.3

[| 3]

a) Die Form

$$S(n) := \begin{cases} c, & \text{falls } n = 1 \\ a \cdot T(\frac{n}{b}) + f(n), & \text{falls } n > 1 \end{cases}$$

ist bei

$$T_1(n) := \begin{cases} c_1, & \text{für } n = 1 \\ 8 \cdot T_1(\frac{n}{2}) + d_1 \cdot n^3, & \text{sonst} \end{cases}$$

eingehalten. Das Mastertheorem ist daher anwendbar.

I. $T_1(n) \in \Theta(n^{log_b(a)})$, falls $f(n) \in O(n^{log_b(a)-\epsilon})$ für ein $\epsilon > 0$.

$$\begin{aligned} f(n) &\in O(n^{log_b(a)-\epsilon}) \\ d_1 \cdot n^3 &\in O(n^{log_2(8)-\epsilon}) \\ d_1 \cdot n^3 &\in O(n^{3-\epsilon}) \end{aligned}$$

Hierfür kann kein ϵ gefunden werden. Daher gilt diese Aussage nicht.

II. $T_1(n) \in \Theta(n^{log_b(a)} \cdot log_2(n))$, falls $f(n) \in \Theta(n^{log_b(a)})$.

$$\begin{aligned} f(n) &\in O(n^{log_b(a)}) \\ d_1 \cdot n^3 &\in O(n^{log_2(8)}) \\ d_1 \cdot n^3 &\in O(n^3) \end{aligned}$$

Dies stimmt, daher gilt diese Aussage.

III. $T_1(n) \in \Theta(f(n))$, falls $f(n) \in \Omega(n^{log_b(a)+\epsilon})$ für ein $\epsilon > 0$ **und** $a \cdot f(\frac{n}{b}) \leq \delta \cdot f(n)$ für ein $\delta < 1$ und große n.

$$\begin{aligned} f(n) &\in \Omega(n^{log_b(a)+\epsilon}) \\ d_1 \cdot n^3 &\in \Omega(n^{log_2(8)+\epsilon}) \\ d_1 \cdot n^3 &\in \Omega(n^{3+\epsilon}) \end{aligned}$$

Dies stimmt für alle $\epsilon \geq 0$, also auch für mindestens ein $\epsilon > 0$.

$a \cdot f(\frac{n}{b}) \leq \delta \cdot f(n)$	\einsetzen
$8 \cdot d_1 \cdot (\frac{n}{2})^3 \leq \delta \cdot d_1 \cdot n^3$	\\$\sqrt[3]{()}$
$\sqrt[3]{d_1} \cdot 2 \cdot \frac{n}{2} \leq \sqrt[3]{\delta} \cdot \sqrt[3]{d_1} \cdot n$	\\: $\sqrt[3]{d_1}$
$n \leq \sqrt[3]{\delta} \cdot n$	\\:n (n ist immer positiv, da $n \in \mathbb{N}$, s. Aufgabenstellung)
$1 \leq \sqrt[3]{\delta}$	\\$()^3$
$1 \leq \delta$	

Damit ist $\delta \geq 1$ und nicht, wie benötigt, $\delta < 1$. Daher gilt diese Aussage nicht.

Da nur II. gilt, gilt $T_1(n) \in \Theta(n^{log_b(a)} \cdot log_2(n))$, also $T_1(n) \in \Theta(n^3 \cdot log_2(n))$.

b) Die Form

$$S(n) := \begin{cases} c, & \text{falls } n = 1 \\ a \cdot T(\frac{n}{b}) + f(n), & \text{falls } n > 1 \end{cases}$$

ist bei

$$T_2(n) := \begin{cases} c_2, & \text{für } n = 1 \\ 5 \cdot T_2(\frac{n}{4}) + d_2 \cdot n^2, & \text{sonst} \end{cases}$$

eingehalten. Das Mastertheorem ist daher anwendbar.

I. $T_2(n) \in \Theta(n^{log_b(a)})$, falls $f(n) \in O(n^{log_b(a)-\epsilon})$ für ein $\epsilon > 0$.

$$\begin{array}{ll} f(n) & \in O(n^{log_b(a)-\epsilon}) \\ d_2 \cdot n^2 & \in O(n^{log_4(5)-\epsilon}) \\ d_2 \cdot n^2 & \in O(n^{1.160964047443681-\epsilon}) \end{array}$$

Hierfür kann kein ϵ gefunden werden. Daher gilt diese Aussage nicht.

II. $T_2(n) \in \Theta(n^{log_b(a)} \cdot log_2(n))$, falls $f(n) \in \Theta(n^{log_b(a)})$.

$$\begin{array}{ll} f(n) & \in O(n^{log_b(a)}) \\ d_2 \cdot n^2 & \in O(n^{log_4(5)}) \\ d_2 \cdot n^2 & \in O(n^{1.160964047443681}) \end{array}$$

Dies stimmt nicht, daher gilt diese Aussage nicht.

III. $T_2(n) \in \Theta(f(n))$, falls $f(n) \in \Omega(n^{log_b(a)+\epsilon})$ für ein $\epsilon > 0$ **und** $a \cdot f(\frac{n}{b}) \leq \delta \cdot f(n)$ für ein $\delta < 1$ und große n.

$$\begin{array}{ll} f(n) & \in \Omega(n^{log_b(a)+\epsilon}) \\ d_2 \cdot n^2 & \in \Omega(n^{log_4(5)+\epsilon}) \\ d_2 \cdot n^2 & \in \Omega(n^{1.160964047443681+\epsilon}) \end{array}$$

Dies stimmt für alle $\epsilon \geq 2 - log_4(5) \approx 0.839035952556319$, also auch für mindestens ein $\epsilon > 0$.

$a \cdot f(\frac{n}{b}) \leq \delta \cdot f(n)$	\einsetzen
$4 \cdot d_2 \cdot (\frac{n}{5})^2 \leq \delta \cdot d_2 \cdot n^2$	\\$\sqrt{()}$
$\sqrt{d_2} \cdot 2 \cdot \frac{n}{5} \leq \sqrt{\delta} \cdot \sqrt{d_2} \cdot n$	\\: $\sqrt{d_2}$
$\frac{2}{5} \cdot n \leq \sqrt{\delta} \cdot n$	\\:n (n ist immer positiv, da $n \in \mathbb{N}$, s. Aufgabenstellung)
$\frac{2}{5} \leq \sqrt{\delta}$	\\$()^2$
$\frac{4}{25} \leq \delta$	
$0.16 \leq \delta$	

Damit gilt $\delta \geq 0.16$. Somit wurde, wie benötigt, mindestens ein $\delta < 1$ gefunden ($0.16 \leq \delta < 1$). Daher gilt diese Aussage.

Da nur III. gilt, gilt $T_2(n) \in \Theta(f(n))$, also $T_2(n) \in \Theta(d_2 \cdot n^2)$, also $T_2 \in \Theta(n^2)$.

c) Die Form

$$S(n) := \begin{cases} c, & \text{falls } n = 1 \\ a \cdot T(\frac{n}{b}) + f(n), & \text{falls } n > 1 \end{cases}$$

ist bei

$$T_3(n) := \begin{cases} c_3, & \text{für } n = 1 \\ 6 \cdot T_3(\frac{n}{3}) + d_3 \cdot n \cdot log(n), & \text{sonst} \end{cases}$$

eingehalten. Das Mastertheorem ist daher anwendbar.

I. $T_3(n) \in \Theta(n^{log_b(a)})$, falls $f(n) \in O(n^{log_b(a)-\epsilon})$ für ein $\epsilon > 0$.

$$\begin{aligned} f(n) &\in O(n^{log_b(a)-\epsilon}) \\ d_3 \cdot n \cdot log(n) &\in O(n^{log_3(6)-\epsilon}) \\ d_3 \cdot n \cdot log(n) &\in O(n^{1.6309297535714573-\epsilon}) \end{aligned}$$

Es gilt: $d_3 \cdot n \cdot log(n) < n^{log_3(6)}$ für alle $n > 0$, für $d_3 = 1$.
Da aber nur $d_3 \cdot n \cdot log(n) \leq n^{log_3(6)}$ für alle $n > 1$ (s. Definition T_3) gefordert ist, kann hierfür mindestens ein ϵ gefunden werden.
Für größere d_3 gilt allerdings, dass die n nicht mehr beliebig größer 1 sein dürfen, sondern eine obere Schranke bekommen.
Somit hängt die Lösung stark von d_3 ab und das Mastertheorem ist ungeeignet für die Lösung dieser Aufgabe.

3.4 Übungsaufgabe 3.4

[| 8]

1.

MERGE(A, B)

```
1  cur_A = 0
2  cur_B = 0
3  C = newlist
4  while cur_A < A.length && cur_B < B.length
5      if cur_A < A.length
6          a_i = A[cur_A]
7      if cur_B < B.length
8          b_j = B[cur_B]
9      if a_i < b_j
10         C.append a_i
11         cur_A = cur_A + 1
12     else
13         C.append b_j
14         cur_B = cur_B + 1
15 if cur_B = B.length
16     for i = cur_A to A.length
17         C.append A[i]
18 else
19     for j = cur_B to B.length
20         C.append B[j]
21 return C
```

Korrektheit von MERGE(A, B):
<u>Schleifeninvariante:</u>
Zu Beginn jeder Iteration sind die Listen A, B und C in sich sortiert.

Initialisierung:
Vor der ersten Iteration der **while**-Schleife sind die Listen A und B geordnet, weil sie geordnet „angeliefert“ werden. Die Liste C ist geordnet, weil sie zu diesem Zeitpunkt noch leer ist.

Fortsetzung:
Bei jeder Iteration wird jeweils das kleinste Element aus beiden Listen der Liste C hinzugefügt. Die Listen A und B verlieren kein Element, also sind sie immer noch sortiert. Die Liste C bekommt immer ein neues Element, welches größer oder gleich dem letzten ist. Somit bleibt C auch bei jeder Iteration sortiert.

Terminierung:
Die **while**-Schleife terminiert stets, da bei jeder Iteration mindestens einer der beiden Zeiger um eine Stelle in Richtung Listenende verrückt wird. Damit kommt ein Zeiger - sofern die Listen endlich sind - irgendwann am Ende an.

Neue Initialisierung:
Die **for**-Schleife in Zeile 16 oder 19 verletzt auch nicht die Invariante. Zu Beginn sind alle drei Listen sortiert, aber B - oder A - wurde schon vollständig an C angefügt. Somit sind alle Elemente der verbleibenden Liste A - oder B - größer oder gleich dem letzten - und größten - Element in C.

Neue Fortsetzung:
Da alle Elemente der verbleibenden Liste A - oder B - größer oder gleich jedem Element in C sind, können wir einfach das nächste - und damit kleinste - verbleibende Element aus A - oder B - an C

anfügen. Bei diesem Schritt bleibt A - oder B - sortiert, weil sie nicht verändert wird, und C auch, da sie nur ein weiteres größeres Element angesetzt bekommt.

Neue Terminierung:
Die **for**-Schleife terminiert auch, da der Zähler bei jeder Iteration um 1 erhöht wird. Solange die Liste A - oder B - also nicht unendlich ist, erreicht der Zähler irgendwann ihr Ende.

Wenn die Schleifen abgebrochen sind, gilt, dass die Listen A, B und C sortiert sind. Die Ausgabeliste C also auch. Dies wollten wir zeigen, damit ist dieser Algorithmus korrekt.

Korrektheit von MERGESORT(A, L, R):
Induktion:
Induktion über die Größe der Eingabeliste:

Induktionsanfang:
Hat die Liste die Länge 1, so ist $l = r$. Die Ursprungsliste wird nicht verändert, womit MERGESORT korrekt ist.

Induktionsannahme:
Ist MERGESORT für die Länge $n \in \mathbb{N}$ wahr, so ist es auch für die Länge $n + 1$ wahr.

Induktionsschritt:
Durch den rekursiven Aufruf, wird MERGESORT(n) für jedes $n > 1$ zu zwei Aufrufen von MERGESORT$(\frac{n}{2})$. Somit wird es bei jedem Rekursionsschritt halb so groß und kommt damit irgendwann bei x Aufrufen von MERGESORT(1) an. Dieses gilt nach Induktionsanfang.

2.
Idee: