Theoretische Aufgaben

Aufgabe 1

Die Definition der O-Notation $O(g(n)) = \{f(n) :$ es existieren positive Konstanten c und $n_0,$ sodass $0 \leq f(n) \leq f(n) \leq c \cdot g(n)$ für alle $n \geq n_0\}$ Die $O(n)$-Notation kann auch als obere symptotische Schranke verstanden werden. Sie kann verwendet werden, bis auf einen konstanten Faktor die obere Schranke einer Funktion anzugeben.
Bezogen auf die Aufgabe lässt sich sagen, dass die Aussage deshalb keinen Sinn macht, weil die $O(n)$-Notation eine obere Schranke festlegt, die Formulierung von *mindestens* jedoch dem widerspricht.

Aufgabe 2

Gemäss der Definition $\Theta(g) = \{f :$ es existieren positive Konstanten c_1, c_2 und $n_0,$ sodass
$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ für alle $n \geq n_0\}$
ist die $\Theta(n)$-Notation dadurch definiert, dass es ab einem Wert n_0 eine obere Schranke $c_1 \cdot g(n) = O(n)$ gibt, und ebenso eine untere Schranke $\Omega(g(n))$. Die Laufzeit eines Algorithmus muss deshalb dann genau $\Theta(g(n))$ sein, wenn $O(g(n))$ und $\Omega(g(n))$ auch zutreffen.

Aufgabe 3

Zeige $a^{log_b n} = n^{log_b a}$:

$$a^{log_b n} = n^{log_b a} \ // \ log_n \tag{1}$$
$$log_n(a^{log_b n}) = log_n(n^{log_b a}) \tag{2}$$
$$log_b(n) \cdot log_n(a) = log_b(a) \cdot log_n(n) \tag{3}$$
$$log_b(n) \cdot \frac{log_b(a)}{log_b(n)} = log_b(a) \tag{4}$$
$$log_b(a) = log_b(a) \tag{5}$$

Aufgabe 4

Die Gleichung $\Theta(log_a n) = \Theta(log_b n)$ stimmt nur, wenn die beiden $\Theta()$-Funktionen um einen konstanten Faktor danebenliegen.
Es gilt deshalb zu zeigen: $\Theta(log_a n) = c \cdot \Theta(log_b n)$.

$$\Theta(log_a n) = \Theta(\frac{log_a n}{log_a b}) \tag{6}$$
$$\Rightarrow log_a b \cdot \Theta(log_a n) = \Theta(log_a n) \tag{7}$$

Da a und b konstant sind, muss auch $log_a b$ konstant sein. Damit muss auch $\Theta(log_a n) = \Theta(log_b n)$ gelten.

Aufgabe 5

Zeigen Sie, dass die Rekursionsgleichung $T(n) = 2 \cdot T([n/4] + 12) + 3n$ die Lösung $O(n \cdot log(n))$ hat.

Die Induktionsannahme wird in die Gleichung eingesetzt:

$$T(n) = 2 \cdot T(\frac{n}{4} + 12) + 3n \leq 2 \cdot \frac{n}{4} \cdot log\frac{n}{4} + 3n \tag{8}$$

$$= \frac{n}{2} \cdot c \cdot log\frac{n}{4} + 3n \tag{9}$$

$$= \frac{n \cdot c \cdot log_4 n}{2} - \frac{n \cdot c \cdot log_4 4}{2} + 3n \tag{10}$$

$$\underbrace{c \cdot n \cdot log_4(n)}_{Induktionsannahme} \underbrace{-\frac{c \cdot n \cdot log_4(n)}{2} - \frac{c \cdot n}{2} + 3n}_{Residuum} \tag{11}$$

Ungleichung gilt, falls $c \geq 1$.

Aufgabe 6

Strukturbaum zu $T(n) = T(n/3) + T(2n/3) + cn$:

$c \cdot n$ $= c \cdot n$

c$\cdot n \cdot \frac{1}{3}$ c$\cdot n \cdot \frac{2}{3}$ $= c \cdot n$

$c \cdot n \cdot \frac{1}{3}^2$ $c \cdot n \cdot \frac{1}{3} \cdot \frac{2}{3}$ $c \cdot n \cdot \frac{2}{3} \cdot \frac{1}{3}$ $c \cdot n \cdot \frac{2}{3}^2$ $= c \cdot n$

Höhe des Baumes: $log(n)$ (da der Algorithmus stoppt, wenn $\frac{1}{3}^n = 1$).

$T(n) = \sum_{i=0}^{h} cn$, wobei die Höhe $h = log(n)$

$\Rightarrow T(n) = c \cdot n \cdot log(n)$. Dies ist in $\Omega(n \cdot log(n))$.

Aufgabe 7

$$T(n) = T(n/2) + \Theta(1) \tag{12}$$

$$a = 1, b = 2, f(n) = 1 \tag{13}$$

$$\text{Fall 2: } f(n) = \Theta(n^{log_b a}) \tag{14}$$

$$\Rightarrow 1 = \Theta(n^0) \tag{15}$$

$$T(n) = \Theta(lg(n)) = \Theta(log(n)) \tag{16}$$

Aufgabe 8

Zeitkomplexität T(n)	Problemgrösse lösbar in 10s	Problemgrösse lösbar in 1000s
$5n$	200	20000
$3n^5$	3	8
$2n^{1.1}$	284	18697
$5log_2(7n)$	$2.3 \cdot 10^{59}$	$3.9 \cdot 10^{6020}$
2^{4n}	2	4

Aufgabe 9

Äussere Schleife: $log(n)$. Innere Schleife: n.

Funktion: $\Theta(n \cdot log(n))$. Laufzeit in der Summenformel: $T(n) = \sum_{i=0}^{n} 2^{i-1} = \frac{2^{n+1}-1}{2}$