

## Algorithmen und Datenstrukturen

### Aufgabe 1 Validation (Beispiellösung)

- a)  $-2^7, 0, 2^7-1$
- b) Fall 1:  $a = 0, b$  beliebig und  $b = 0, a$  beliebig  
Fall 2:  $a < 0, b \neq 0$   
Fall 3:  $a > 0, b \neq 0; a \geq 2$  für doppelten Durchlauf der Schleife
- c) Nein. Nur Sortierungs-Eigenschaft und Stabilität ist testbar.

### Aufgabe 2 Wachstumsraten (Beispiellösung)

schwarz:  $f(n) = 1$                       grün:  $f(n) = n$                       pink:  $f(n) = n^2$   
blau:  $f(n) = \log(n)$                       rot:  $f(n) = n \log(n)$                       cyan:  $f(n) = 2^n$

### Aufgabe 3 Rechnen mit Landau-Symbolen (Beispiellösung)

a)  $7n^4 = O(n^5)$

$$\begin{aligned} \stackrel{\text{Def.}}{\Rightarrow} 0 \leq 7n^4 \leq c \cdot n^5 & \quad | : n^4 \quad (> 0 \quad \forall n \geq 1) \\ \Rightarrow 0 \leq 7 \leq c \cdot n & \\ \Rightarrow \text{Behauptung mit } c = 7 \text{ und } \forall n \geq n_0 = 1 & \end{aligned}$$

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

Untere Schranke:

$$\begin{aligned} \stackrel{\text{Def.}}{\Rightarrow} c_1 \cdot n^2 \leq \frac{n^2}{2} - 2n & \quad | : n^2 \quad (> 0 \quad \forall n \geq 1) \\ \Rightarrow c_1 \leq \frac{1}{2} - \frac{2}{n} & \\ \Rightarrow \text{Gilt mit } c_1 = \frac{1}{4} \text{ und } \forall n \geq n_0 = 8 & \end{aligned}$$

Obere Schranke:

$$\begin{aligned} \stackrel{\text{Def.}}{\Rightarrow} \frac{n^2}{2} - 2n \leq c_2 \cdot n^2 & \quad | : n^2 \quad (> 0 \quad \forall n \geq 1) \\ \Rightarrow \frac{1}{2} - \frac{2}{n} \leq c_2 & \\ \Rightarrow \text{Gilt mit } c_2 = \frac{1}{2} \text{ und } \forall n \geq n_0 = 8 & \end{aligned}$$

Insgesamt folgt die Behauptung mit  $c_1 = \frac{1}{4}, c_2 = \frac{1}{2}$  und  $n_0 = 8$ .

c)  $2^{n+1} = O(2^n)$

$$\begin{aligned} &\stackrel{\text{Def.}}{\Rightarrow} 0 \leq 2^{n+1} \leq c \cdot 2^n \\ &\stackrel{2^{n+1}=2 \cdot 2^n}{\Rightarrow} \text{Behauptung mit } c = 2 \text{ und } \forall n \geq n_0 = 1 \end{aligned}$$

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

Annahme: Behauptung trifft *nicht* zu (also  $2^{2n} = O(2^n)$ )

$\Rightarrow \exists$  Konstanten  $c$  und  $n_0 > 0$ , so dass  $0 \leq 2^{2n} \leq c \cdot 2^n \quad \forall n \geq n_0$

$\Rightarrow$  Zerlege  $2^{2n} = 2^n \cdot 2^n \quad (\leq c \cdot 2^n)$

$\Rightarrow 2^n \leq c$

Widerspruch! (Folge  $2^n$  ist unbeschränkt.) Also gilt die ursprüngliche Annahme.

#### Aufgabe 4 Rechnen mit Landau-Symbolen (Beispiellösung)

a) Die Behauptung ist **falsch**. Konstante Faktoren der tatsächlichen Laufzeitfunktion entfallen bei Landau-Symbolen der Einfachheit halber. Daher wird das doppelt so große Beispiel tatsächlich etwa doppelt so lange rechnen, wie das einfacher Größe.

b) Sei  $T_f(n)$  die Laufzeitfunktion von  $f$ .

$\Rightarrow \exists c, n_0 : 0 \leq T_f(n) \leq c \cdot n \quad \forall n \geq n_0$

Bei  $n$ -maliger Ausführung:

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

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

c) Wähle  $f(n) = n$  und  $g(n) = 2 \cdot n$ .

$\Rightarrow f(n) = \Theta(g(n))$ :

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

etwa mit  $c_1 = c_2 = \frac{1}{2}$  und  $n_0 = 1$ .

Annahme: Aussage ist wahr.

$\Rightarrow 2^{f(n)} = \Theta(2^{g(n)})$ :

$$\begin{aligned} &0 \leq c_1 \cdot 2^{2 \cdot n} \leq 2^n \leq c_2 \cdot 2^{2 \cdot n} \quad | : 2^n \quad (> 0 \quad \forall n \geq 1) \\ \Rightarrow &0 \leq c_1 \cdot 2^n \leq 1 \leq c_2 \cdot 2^n \end{aligned}$$

Widerspruch (Gegenbeispiel)! (Folge  $2^n$  ist unbeschränkt.) Also ist Behauptung **falsch**.

d) Annahme: Aussage ist wahr.

$\Rightarrow n^n = O(2^n)$ :

$$\begin{aligned} &0 \leq n^n \leq c \cdot 2^n \quad | : 2^n \quad (> 0 \quad \forall n \geq 1) \\ \Rightarrow &\left(\frac{n}{2}\right)^n \leq c \end{aligned}$$

Widerspruch! (Folge  $\left(\frac{n}{2}\right)^n$  ist unbeschränkt.) Also ist Behauptung **falsch**.

e) Annahme: Aussage ist wahr.

$$\Rightarrow n = O(n^3):$$

$$\begin{aligned} 0 \leq n \leq c \cdot n^3 & \quad | : n^3 \quad (> 0 \quad \forall n \geq 1) \\ \Rightarrow \frac{1}{n^2} \leq c & \end{aligned}$$

Da  $\frac{1}{n^2} \rightarrow 0$ , findet sich leicht etwa  $c = 1$  und  $n_0 = 10$ . Die Behauptung ist **wahr**.

### Aufgabe 5 Komplexität von Selection Sort (Beispiellösung)

a) **Swap** besteht aus 3 Befehlen, also Aufwand 3 laut RAM-Modell, also  $O(1)$ . (Beweis mit  $c \geq 3$ .)

```

b) 1   i = j;
    2   k = j + 1;
    3   while (k < n) {
    4       if (A[k] < A[i]) {
    5           i = k;
    6       }
    7       k = k + 1;
    8   }
    
```

Aufrufhäufigkeiten mit  $t_k$  Ausführungen von Zeile 5 (wegen erfolgreichen Tests in Zeile 4):

Zeile	Kosten	Häufigkeit
1	$c_1$	1
2	$c_2$	1
3	$c_3$	$n - j$
4	$c_4$	$n - j - 1$
5	$c_5$	$t_k$
6	$c_6$	$n - j - 1$

Insgesamt:

$$\begin{aligned} T(n) &= 1 + 1 + (n - j) + (n - j - 1) + t_k + (n - j - 1) \\ &= 3 \cdot n - 3 \cdot j + t_k \end{aligned}$$

*Best Case:* Zeile 5 nie ausgeführt ( $t_k = 0$ ), also immer  $A[k] \geq A[i]$ , also  $A$  bereits aufsteigend sortiert.

*Worst Case:* Zeile 5 in jeder Iteration ausgeführt ( $t_k = n - j - 1$ ), also immer  $A[k] < A[i]$  immer, also  $A$  umgekehrt (absteigend) sortiert.

*O-Notation* (als Hinweis für *Average Case*, ignoriert Startindex  $j$ ):  $T(n) = O(n)$ . (Beweis: Im aller-schlechtesten Fall gilt  $j = 0$  und  $A$  ist absteigend sortiert. Dann  $t_k = n$ , damit  $T(n) \leq 4 \cdot n$ .)

c) Die **for**-Schleife wird  $n - 1$  mal durchlaufen: Darin jedes Mal zunächst die  $O(n)$ -Funktion **IndexOfMin**, dann – nach einem Vergleich – nötigenfalls die  $O(1)$ -Funktion **Swap**. Dazu kommt  $O(1)$ -Aufwand zur Initialisierung der Schleife.

$$\Rightarrow O(n^2)$$

**Aufgabe 6 Komplexitätsschätzung (Beispiellösung)**

a)  $O(n)$  (linear)

b)  $O(n^3)$  (kubisch)