Aufgaben für Algorithmische Mathematik 1

Aufgabe 1.)

(a) Sei $f = O(g) \Leftrightarrow \exists C>0\ \exists \varepsilon>0\ \forall n>n_0 : |f(n)| \le \overset{\text{groß } C}{C} \cdot |g(n)|$

$$\overset{(*)}{\Leftrightarrow} \exists c>0\ \exists \varepsilon>0\ \forall n>n_0 : \overset{\text{klein } c}{c} \cdot |f(n)| \le |g(n)|$$

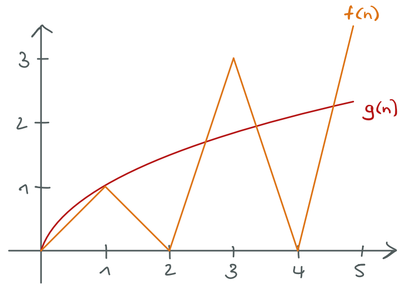
$$\Leftrightarrow g = \Omega(f)$$

(*) „$\Rightarrow$": $\frac{\cdot}{C}$, erlaubt weil $C \neq 0$ und Vergleichsoperator dreht sich nicht, weil $C>0$: $c := \frac{1}{C}$

„$\Leftarrow$": Vergleichsoperator dreht sich nicht, weil $c>0$

(b) Sei $f, g : \mathbb{N} \to \mathbb{R}_{\ge 0}$ mit

$$f: \begin{cases} n, & n \text{ ungerade} \\ 0, & n \text{ gerade} \end{cases} \qquad g(n) = \sqrt{n}$$



„$\not\Rightarrow$": Angenommen es wäre $f = O(g)$

$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall n \ge n_0 : |f(n)| \le C \cdot |g(n)|$$
$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall n \ge n_0 : |f(n)| \le C \cdot \sqrt{n}$$
$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall 2n \ge n_0 : |f(2n)| \le C \cdot \sqrt{2n}$$
$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall 2n \ge n_0 : |2n| \le C \cdot \sqrt{2n}$$
$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall 2n \ge n_0 : |4n^2| \le C^2\, |2n| \quad ↯$$

„$\not\Leftarrow$": Angenommen es gelte $f = \Omega(g)$

$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall n \ge n_0 : c \cdot |g(n)| \le |f(n)|$$
$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall 2n+1 \ge n_0 : c \cdot |g(2n+1)| \le |f(2n+1)|$$
$$\Leftrightarrow \exists c>0\ \exists \varepsilon>0\ \forall 2n+1 \ge n_0 : \underbrace{c}_{>0} \cdot |\underbrace{\sqrt{2n+1}}_{>0}| \le |0| \quad ↯$$

Somit gilt nicht für alle $f, g : \mathbb{N} \to \mathbb{R}_{\ge 0}$: $f = O(g)$ oder $f = \Omega(g)$

Aufgabe 2.)

(a) Zz.: $\log(n!) = \Theta(n \log(n))$

$$\Leftrightarrow \log(n!) \in O(n \log(n)) \quad \wedge \quad \log(n!) \in \Omega(n \log(n))$$

$$n! \leq n^n \Leftrightarrow \log(n!) \leq \log(n^n) \Leftrightarrow \log(n!) \leq n \cdot \log(n)$$

$$\Rightarrow \log(n!) \leq 1 \cdot n \cdot \log(n) \overset{c=1}{\Rightarrow} \log(n!) \in O(n \cdot \log(n))$$

$$n! = \frac{n!}{(n-\frac{n}{2})!} \cdot (n-\tfrac{n}{2})! \geq \frac{n!}{(n-\frac{n}{2})!} \geq \left(\frac{n}{2}\right)^{\left(\frac{n}{2}\right)}$$

$$\Leftrightarrow \log(n!) \underset{c=\frac{1}{2}}{\geq} \log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right) = \frac{1}{2} \cdot n \cdot \log\left(\frac{n}{2}\right) = \frac{1}{2} n \left(\log(n) - \log(2)\right)$$

$$\Rightarrow \log(n!) \in \Omega(n \cdot \log(n))$$

(b) Zz.: $(\log(n))^k = O(n^{\frac{1}{l}})$ für alle $k, l \in \mathbb{N}$:

$$(\log(n))^k = n^{\frac{1}{l}} \cdot \frac{(\log(n))^k}{n^{\frac{1}{l}}}$$

Für ein bel. aber festes $k, l \in \mathbb{Z}$ gelte:

$$\lim_{n\to\infty} \frac{(\log(n))^k}{n^{\frac{1}{l}}} = \lim_{n\to\infty} \left(\frac{\log(n)}{n^{\frac{1}{l \cdot k}}}\right)^k = \left(\lim_{n\to\infty} \frac{\log(n)}{n^{\frac{1}{l \cdot k}}}\right)^k \overset{(1)}{=} 0^k = 0$$

Für $r > 0$ gilt:

$$\lim_{n\to\infty} \frac{\log(n)}{n^r} \overset{L'H}{=} \lim_{n\to\infty} \frac{1/n}{r \cdot n^{r-1}} = \lim_{n\to\infty} \frac{1}{r \cdot n^r} = 0 \qquad (1)$$

$$\Rightarrow (\log(n))^k = o(n^{\frac{1}{l}}) \overset{o(f) \subseteq O(f)}{\Rightarrow} (\log(n))^k = O(n^{\frac{1}{l}})$$

Aufgabe 3.)

- $S = \{s_1, \dots, s_n\}$, $s_i \in \mathbb{N}$ für alle $i = 1, \dots, n$
- $\sigma : \{1, \dots, n\} \to \{1, \dots, n\} : \forall_{1 \le i \le j \le n} \; s_{\sigma(i)} \le s_{\sigma(j)}$

(a)

IA: $i = 1$ $\quad \{s_{\sigma(1)}\}$ ist als 1-Elementige Liste trivialerweise sortiert

und aus σ ergibt sich $s_{\sigma(1)} \le s_{\sigma(j)}$ für $j \in \{2, \dots, n\}$

IV: Für ein bel. aber festes $i \in \{1, \dots, n-1\}$ gelte:

$\{s_{\sigma(1)}, \dots, s_{\sigma(i)}\}$ ist sortiert und für jedes $s \in \{s_{\sigma(i+1)}, \dots, s_{\sigma(n)}\} : s \ge \{s_{\sigma(1)}, \dots, s_{\sigma(i)}\}$

IS: $i \rightsquigarrow i+1$:

5: $\quad j := i+2$
6: $\quad$ if $s_{\sigma(i+2)} < s_{\sigma(i+1)}$
7-9: $\quad$ Tausche $\sigma(i+2)$ und $\sigma(i+1)$
10: $\quad s_{\sigma(i+1)} < s_{\sigma(i+2)}$

$$\overset{IV}{\Rightarrow} \left(s_{\sigma(1)} \le \dots \le s_{\sigma(i)}\right) \overset{\text{Def. }\sigma}{\le} s_{\sigma(i+1)} \overset{\text{Algo.}}{\le} s_{\sigma(i+2)} \Rightarrow \{s_{\sigma(1)}, \dots, s_{\sigma(i+1)}\} \text{ sortiert}$$

Aus Def. $\sigma \Rightarrow \quad s_{\sigma(i+1)} \le s_{\sigma(i+2)}, \dots, s_{\sigma(n)}$

$\Rightarrow$ jedes Element aus $\{s_{(i+2)}, \dots, s_{\sigma(n)}\}$ ist größer, als ein beliebiges Element aus $\{s_{\sigma(1)}, \dots, s_{\sigma(i+1)}\}$

$\square$

(b)

```
 1: for i ← 1 to n do
 2:     σ(i) ← i
 3: end for
 4: for i ← 1 to n − 1 do
 5:     for j ← i + 1 to n do
 6:         if s_σ(j) < s_σ(i) then
 7:             tmp ← σ(i)
 8:             σ(i) ← σ(j)
 9:             σ(j) ← tmp
10:         end if
11:     end for
12: end for
13: return σ
```

(Zeilen 1–3:) nach n-Durchläufen ist $i > n$, also bricht die Scheife ab

(Zeilen 5–10:) nach $n-(i+1)$-Durchläufen ist $j = i+1 > n$, also bricht die Schleife ab

(Zeilen 4–12:) nach $(n-1)$-Durchläufen ist $i > n-1$, also bricht die Schleife ab

$\Rightarrow$ Der Algorithmus terminiert.

(c)

```
for i ← 1 to n do                     O(n)
    σ(i) ← i                          · O(1)
end for
for i ← 1 to n − 1 do                 + O(n-1)
    for j ← i + 1 to n do             · O(n-(i+1))
        if s_σ(j) < s_σ(i) then       · [O(1)
            tmp ← σ(i)                + O(1)
            σ(i) ← σ(j)               + O(1)
            σ(j) ← tmp                + O(1)]
        end if
    end for
end for
return σ
```

$$\Rightarrow T(n) = \underbrace{O(n)\cdot O(1)}_{\in O(n)} + \underbrace{O(n-1)}_{\in O(n)} \cdot \left[\underbrace{O(n-(i+1))}_{\in O(n)} \cdot \underbrace{(O(1)+O(1)+O(1))}_{\in O(1)}\right]$$

$$\Rightarrow T(n) = O(n) + \underbrace{O(n)\cdot\left[(\underbrace{O(n)\cdot O(1)}_{\in O(n)})\right]}_{\in O(n^2)}$$

$$\Rightarrow T(n) = \underbrace{O(n) + O(n^2)}_{\in O(n^2)}$$

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

Der Algorithmus liegt in der Zeitkomplexität $O(n^2)$