Datenstrukturen und Algorithmen

Abgabe: 03.05.2017

Adrian C. Hinrichs Matr.Nr. 367129
Georg C. Dorndorf Matr.Nr. 366511

Aufgabe 3

Aufgabe 3.a

Zu zeigen: $n^2 + n \in \mathcal{O}(n^2 - n)$
Nach Definition gilt dies genau dann wenn gilt:

$$\begin{aligned}
\exists n_0 \in \mathbb{N}, c_1 \in \mathbb{R}_{>0} : \forall n \geq n_0 : n^2 + n &< c_1 \cdot (n^2 - n) && (1)\\
\Leftrightarrow n^2 + n &< c_1 \cdot n^2 - c_1 \cdot n && (2)\\
\Leftrightarrow n &< (c_1 - 1) \cdot n^2 - c_1 \cdot n && (3)\\
\Leftrightarrow (c_1 + 1) \cdot n &< (c_1 - 1) \cdot n^2 && (4)\\
\overset{n>0}{\Longleftrightarrow} (c_1 + 1) &< (c_1 - 1) \cdot n && (5)\\
& && (6)
\end{aligned}$$

Dies gilt offensichtlich für alle $n > 1$, wenn $c_1 \geq 2$. Also gilt nach Definition die Behauptung mit $n_0 = 2$ und $c_1 \geq 2$.

QED

Aufgabe 3.b

Zu zeigen: $(n + 1)(n^2 + 3) \in \Theta(n^3)$

$$(n + 1)(n^2 + 3) = n^3 + 3n^2 + 3 \quad (7)$$

Damit die Behauptung erfüllt ist, müssen nach Vorlesung $n_0 \in \mathbb{N}$, $c_1, C_2 \in \mathbb{R}_{>0}$ existieren, so dass für alle $n \geq 0$ gilt:

$$c_1 \cdot n^3 \leq n^3 + n^2 + 3m + 3 \leq c_2 \cdot n^3 \quad (8)$$

Sei $c_1 = 1$, so gilt:

$$c_1 \cdot n^3 = n^3 \leq n^3 + 3n + n^2 \text{ für alle } n \in \mathbb{N} \quad (9)$$

Noch zu bestimmen: $c_2 \in \mathbb{R}, n_0 \in \mathbb{N}$. Es muss für alle $n > n_0$ gelten:

$$\begin{aligned}
n^3 + n^2 + 3 &\leq c_2 \cdot n^3 && (10)\\
\overset{n^3>0}{\Longleftrightarrow} n^2 + 3n + 3 &\leq (c_2 - 1)n^3 && (11)
\end{aligned}$$

da für alle $n \in \mathbb{N}_0$ gilt: $n^2 + 3n + 3 > 0$, muss auch $(c_2 - 1)n^3 > 0$ für alle $n \in \mathbb{N}_0$ gelten, also muss $c_2 > 1$ sein. Wir können nach Vorlesung annehmen, dass $n_0 > 1$ gilt, daher gilt für alle $n \geq n_0$ die Äquivalenz

$$\Leftrightarrow \frac{n^2 + 3n + 3}{(c_2 - 1)n^3} \leq 1 \tag{12}$$

Durch scharfes Hinsehen ergibt sich ein kandidat für c_2, sei also $c_2 = 4$:

$$\Rightarrow \frac{n^2 + 3n + 3}{3n^3} = \frac{1}{3n} + \frac{1}{n^2} + \frac{1}{n^3} \leq 1 \tag{13}$$

Da $\frac{1}{3n} + \frac{1}{n^2} + \frac{1}{n^3}$ streng monoton fallend ist, lässt sich durch scharfes Hinsehen festellen, dass die Ungleichung offensichtlich für alle $n \geq 2$ erfüllt ist (da $\frac{1}{3 \cdot 2} + \frac{1}{4} = \frac{13}{24} < 1$). Also existieren c_1, c_2 und n_0 mit $c_1 = 1, c_2 = 4$ und $n_0 = 2$ derart dass die Gleichung 8 erfüllt ist.
Nach VL ist also die Behaupting $(n+1)(n^2+3) \in \Theta(n^3)$ bewiesen.

QED

Aufgabe 3.c

Zu zeigen: $(\log n)^2 \in \mathcal{O}(n)$
Nach Definition ist dies erfüllt , genau dann wenn:

$$\exists c \in \mathbb{R}_{\geq 0}, c \neq \infty : \limsup_{n \to \infty} \frac{(\log n)^2}{n} = c \tag{14}$$

$$\limsup_{n \to \infty} \frac{(\log n)^2}{n} = \limsup_{n \to \infty} \frac{(\log n)}{n} \; cdot \log n \tag{15}$$

$$\overset{perDef.}{=} 0 \tag{16}$$

Also gilt die Behauptung.

QED

Aufgabe 3.d

Zu wiederlegen: $f(n) + g(n) \in \Theta(\min(f(n), g(n)))$
Seien zwei Funktionen $f(n) = n$ und $g(n) = (e^\pi - \pi)^n$[1] gegeben. Nun gilt für alle $n \in \mathbb{N}$:

$$\min(f(n), g(n)) = \min(n, (e^\pi - \pi)^n) = n = f(n) \tag{17}$$

Also gilt für alle $n \in \mathbb{N}$

$$\Theta(\min(f(n), g(n)) = \Theta(f(n)) \not\ni (e^\pi - \pi)^n = g(n) \tag{18}$$

$$\Rightarrow \Theta(\min(f(n), g(n)) = \Theta(f(n)) \not\ni (e^\pi - \pi)^n + n = f(n) + g(n) \tag{19}$$

Somit ist die Behauptung wiederlegt.

QED

[1] Vgl. `https://xkcd.com/217/`

Aufgabe 3.e

Zu zeigen: $((f(n) \in \Omega(g(n))) \wedge (g(n) \in \Omega(h(n)))) \Rightarrow (f(n) \in \Omega(h(n)))$

$$f(n) \in \Omega(g(n)) \Rightarrow \exists n_0 \in \mathbb{N}, c_1 > 0 : f(n) \geq g(n) * c_1 \tag{20}$$

$$\overset{c_1>0}{\Longleftrightarrow} \frac{f(n)}{c_1} \geq g(n) \tag{21}$$

$$g(n) \in \Omega(h(n)) \Rightarrow \exists n_0' \in \mathbb{N}, c_1' > 0 : g(n) \geq h(n) * c_1' \tag{22}$$

Aus (21) & (22) folgt:

$$\frac{f(n)}{c_1} \geq g(n) \geq h(n) \cdot c_1' \forall n \geq \max(n_0, n_0') \tag{23}$$

$$\Rightarrow \frac{f(n)}{c_1} \geq h(n) \cdot c_1' \forall n \geq \max(n_0, n_0') \tag{24}$$

$$\overset{c_1>0}{\Longleftrightarrow} f(n) \geq h(n) \cdot c_1 c_1' \forall n \geq \max(n_0, n_0') \tag{25}$$

Dies ist nichts anderes als die Definition der Klasse Ω.

QED

Aufgabe 4

$k \in \mathbb{N}$ konstant, Funktionsschar $f_0(n), f_1(n), \ldots, f_k(n) \in \mathcal{O}(g(n))$
Zu zeigen: $\sum_{i \in [0,k]} f_i(n) \in \mathcal{O}(g(n))$ Es gilt:

$$\forall i \in [0,k] : f_i(n) = g(n) \implies \tag{26}$$

$$\sum_{i \in [0,k]} f_i(n) \overset{\text{per def.}}{\in} \mathcal{O}(\sum_{i \in [0,k]} g(n)) = \mathcal{O}(k \cdot g(n)) \tag{27}$$

$$\overset{\text{per def.}}{=} \mathcal{O}(g(n)) \tag{28}$$

QED

Aufgabe 5

Aufgabe 5.i

Der Algorithmus funktioniert indem er linear das Array nach allen Zykeln durchsucht ($\rightarrow 8$:) und immer wenn er einen neuen gefunden hat diesen zunächst ganz durchläuft. Dabei zählt er die Elemente im Zykel und prüft am Ende ob der zuletzt gefundene Zykel länger als der bisher längste ist. Falls dies der Fall ist wird der index und die länge aktualisiert ($\rightarrow 10$:). Danach kehrt der Algorithmus zu seiner Suche zurück($\rightarrow 15$:).

```
    a ← input
    index ← 0
    length ← 0
    length′ ← 0
 5: i ← 0
    globalcount ← 0

    while globalcount < a.length do
      if a[i] == -1 then
10:     if length′ > length then
          index ← i
          length ← length′
        end if
        length′ ← 0
15:     i ← ++globalcount
      else
        i′ ← i
        i ← a[i]
        a[i′] ← −1
20:     length′++
      end if
    end while
    return index
```

Aufgabe 5.ii

Der Algorithmus hat eine lineare Worst-Case Laufzeit $W(n) \in \Theta(n)$ abhängig von der Eingabelänge n. Dies lässt sich damit begründen, dass die While-Schleife jedes Element genau einmal überprüft und durch die Sprünge des Index i jedes Element der Zykel genau ein weiteres mal besucht wird. Es ergeben sich also genau $W(n) = 2n$ Vergleiche der Elemente des Arrays. Nach der Definition von Θ gilt $W(n) \in \Theta(n)$ da der Konstante lineare Faktor irrelevant ist.

QED