Gruppe **14**
Datenstrukturen und Algorithmen

Phil Pützstück, 377247
Benedikt Gerlach, 376944
Sebastian Hackenberg, 377550

Hausaufgabe 5

Aufgabe 1

a)

Wir zeigen, dass für $n \in \mathbb{N}$ mit $n \geq 2$ stets $T(n) \in \mathcal{O}(n)$.

Sei $n \in [2, 15]_{\mathbb{N}}$. Dann ist

$$T(n) = 1 \leq n \log_2 n$$

Also existiert ein $c = 1 > 0$ sodass $T(n) \leq c \cdot n \log_2 n$.

Sei nun ein $n \in \mathbb{N}$ gegeben, sodass für alle $n' \in \mathbb{N}$ mit $n' < n$ ein $c > 0$ existiert für das $T(n) \leq c \cdot n \log_2 n$ ist (IV). Es folgt:

$$T(n) = 2 \cdot T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n \quad \overset{\text{IV}}{\leq} \quad 2c_1 \left(\frac{n}{4} \log_2 \frac{n}{4}\right) + c_2 \left(\frac{n}{2} \log_2 \frac{n}{2}\right) + n$$

$$= c_1 \frac{n}{2}(\log_2 n - 2) + c_2 \frac{n}{2}(\log_2 n - 1) + n \quad \leq \quad c\frac{n}{2}(\log_2 n - 2 + \log_2 n - 1) + n$$

$$= c\frac{n}{2}(2 \log_2 n - 3) + n = cn \log_2 n - \frac{n}{2} \leq cn \log_2 n$$

wobei c_1, c_2 die Konstanten gemäß von (IV) sind und $c := \max\{c_1, c_2\}$. Folglich gilt nach dem Prinzip der vollständigen Induktion, dass es zu jedem $n \in \mathbb{N}, n \geq 2$ ein $c > 0$ gibt, sodass $T(n) \leq c \cdot n \log_2 n$ ist. Folglich ist $T(n) \in \mathcal{O}(n \log_2 n)$. □

b)

Sei $n \in [1, 3]_{\mathbb{N}}$. Dann ist

$$T(n) = 1 \geq \frac{1}{3} \cdot 3^{\frac{n}{3}}$$

Also existiert ein $c = \frac{1}{3} > 0$ sodass $T(n) \geq c \cdot 3^{\frac{n}{3}}$.

Sei nun ein $n \in \mathbb{N}$ gegeben, sodass für alle $n' \in \mathbb{N}$ mit $n' < n$ ein $c > 0$ existiert für das $T(n) \geq c \cdot 3^{\frac{n}{3}}$ ist (IV). Es folgt:

$$T(n) = T(n-1) + T(n-2) + T(n-3) \quad \overset{\text{IV}}{\geq} \quad c_1 3^{\frac{n-1}{3}} + c_2 3^{\frac{n-2}{3}} + c_3 3^{\frac{n-3}{3}}$$

$$3c \cdot 3^{\frac{n-3}{3}} = c \cdot 3^{\frac{3}{n}}$$

wobei c_1, c_2, c_3 die Konstanten gemäß von (IV) sind und $c := \min\{c_1, c_2, c_3\}$. Folglich gilt nach dem Prinzip der vollständigen Induktion, dass es zu jedem $n \in \mathbb{N}$ ein $c > 0$ gibt, sodass $T(n) \geq c \cdot 3^{\frac{n}{3}}$ ist. Foglich ist $T(n) \in \Omega(3^{\frac{n}{3}})$. □

Aufgabe 2

a) Es ist $b = 8, c = 2, E = 3$. Wir wählen $\varepsilon = 1 > 0$. Offensichtlich gilt (siehe z.B. HA02)

$$f(n) = 2^n \in \Omega(n^{E+\varepsilon}) = \Omega(n^4)$$

Ferner gilt

$$8f\left(\frac{n}{c}\right) = 8 \cdot 2^{\frac{n}{2}} = 2^{\frac{n}{2}+3} \leq \frac{1}{2}2^n = 2^{n-1}$$

für $n \geq 8$, da dann auch stets $\frac{n}{2} + 3 \leq n - 1$ gilt. Also existiert auch ein $d = \frac{1}{2} < 1$ und ein $n_0 = 8$ sodass

$$\forall n \geq n_0 : bf(\frac{n}{c}) \leq df(n)$$

Nach dem Mastertheorem folgt, dass $T(n) \in \Theta(f(n)) = \Theta(2^n)$.

b)

Es ist $b = 64, c = 4, E = 3, f(n) = n^3 + 7n^2 + n + 7$. Nach Vorlesung (oder auch HA02) ist n^3 der dominierende Faktor in $f(n)$. Es folgt $f(n) \in \Theta(n^3) = \Theta(n^E)$ und damit $T(n) \in \Theta(n^3 \log n)$

c) nicht möglich **FEHLT NOCH**

d) Es ist $b = 16, c = 4, E = 2, f(n) = n \log n + n^2$. Es gilt (auch nach Vorlesung) stets $n \log n \leq n^2$, womit n^2 der dominierende Faktor in $f(n)$ ist und sofort $f(n) \in \Theta(n^2) = \Theta(n^E)$ folgt. nach dem Mastertheorem gilt dann $T(n) \in \Theta(n^2 \log n)$.

e) Es ist $b = 1, c = 3, E = 0$. Nach Konstruktion von f gilt stets $f(n) \geq 3n > 1 = n^E > n^{E-\varepsilon}$ für $n \in \mathbb{N}$ und $\varepsilon > 0$. Also $f(n) \notin \Theta(n^E) = \Theta(1)$ sowie $f(n) \notin \mathcal{O}(n^{E-\varepsilon})$ für alle $\varepsilon > 0$.
Damit können weder Fall 2 noch Fall 1 erfüllt werden.

Nach Vorlesung (und HA02) gilt jedoch $\sqrt{n} \in \mathcal{O}(n)$ sowie $n \in \Omega(\sqrt{n})$. Ferner ist ja $\sqrt{n} := n^{\frac{1}{2}}$. Wir wählen also $\varepsilon = \frac{1}{2}$. Dann ist gilt stets

$$f(n) \geq 3n \geq \sqrt{n} = n^{E+\varepsilon}$$

für $n \in \mathbb{N}$. Foglich ist $f(n) \in \Omega(n^{E+\varepsilon})$. Wir unterscheiden nun 2 Fälle:

Fall 1: $n \notin \{2^i \mid i \in \mathbb{N}\}$. Dann ist $f(n) = 3n$ und es gilt stets

$$f(\frac{n}{3}) = n \leq \frac{3}{2}n = \frac{1}{2}f(n)$$

Fall 2: $n \in \{2^i \mid i \in \mathbb{N}\}$. Dann ist $f(n) = 3n + 2^{3n}$ und es gilt stets

$$f(\frac{n}{3}) = n + 2^n \leq \frac{3}{2}n + 2^{3n} \overset{n \geq 1}{\leq} \frac{3}{2}n + 2^{3n-1} = \frac{1}{2}f(n)$$

Folglich gibt es in beiden Fällen ein $d = \frac{1}{2} < 1$ sodass insgesamt für alle $n \in \mathbb{N}$ gilt:

$$bf(\frac{n}{c}) \leq df(n)$$

Nach dem Mastertheorem folgt nun $T(n) \in \Theta(f(n))$.

Aufgabe 3

a)

Der Rückgabewert des Aufrufs `T([3,1,2,4])` ist `[4,3,2,1]`.
Die 11 ersten `print`-Statements lauten wie folgt:

```
T: [3,1,2,4]
S: [3,1,2,4] [0,0,-1]
S: [1,2,4] [1,0,3]
S: [2,4] [2,0,3]
S: [4] [3,0,3]
S: [] [4,3,4]
T: [1,2,3]
S: [1,2,3] [0,0,-1]
S: [2,3] [1,0,1]
S: [3] [2,1,2]
S: [] [3,2,3]
```

b)

`S` gibt den Index des Maximums der Eingabeliste `L` zurück.
Dieser wird der Variable `i` dann in der Funktion `T` zugewiesen.

`c` stellt die bis jetzt größte gesehen Zahl dar; Die Funktion achtet nur auf den Kopf der Liste und rekursiert dann weiter, ist dieser Kopf größer als `c`, so ist `c` im rekursiven Aufruf eben dieser Kopf, also das momentane Maximum.

`a` stellt den Index von `L.head` in der ursprünglich übergebenen Liste `L` aus dem ersten Aufruf von `S` dar. Entsprechend wird `a` bei jedem rekursiven Aufruf inkrementiert, da wir beim rekursiven Aufrufe nur `L.tail` übergeben und damit über `L.head` in dem rekursiven Aufurf das nächste Element der Liste betrachten.

`b` ist dabei der Index von `c` in der urpsrünglich übergebenen Liste `L` aus dem ersten Aufruf von `S`, also der Index des momentanten Maximums `c`. Entsprechend wird `b` auf `a` gesetzt sobald auch `c` geändert wird, also ein neues Maximum gefunden wurde.

c)

Wenn wir die Aufrufe von S zählen ergeben sich die Rekursiongleichungen

$$T(n) = \begin{cases} 0 & n = 1 \\ S(n) + T(n-1) & \text{sonst} \end{cases} \qquad S(n) = \begin{cases} 0 & n = 0 \\ 1 + S(n-1) & \text{sonst} \end{cases}$$

Es ist offensichtlich $S(n) = n$:
Sei $n = 0$, dann ist $S(n) = 0 = n$. Sei $n \in \mathbb{N}_0$ mit $S(n) = n$ gegeben (IV).
Es folgt $S(n+1) = 1 + S(n) \overset{\text{IV}}{=} n + 1$. Folglich gilt nach Prinzip der vollständigen Induktion dass $S(n) = n$ für alle $n \in \mathbb{N}_0$. Es folgt also nun

$$T(n) = \begin{cases} 0 & n = 1 \\ n + T(n-1) & \text{sonst} \end{cases}$$

Damit gilt auch offensichtlich $T(n) = \sum_{i=1}^{n} i - 1 = \sum_{i=2}^{n} i$:
Sei $n = 1$, dann ist $T(n) = 0 \stackrel{\text{def}}{=} \sum_{i=2}^{1} i$. Sei $n \in \mathbb{N}$ mit $T(n) = \sum_{i=2}^{n} i$ gegeben (IV).
Es folgt

$$T(n+1) = n + 1 + T(n) \stackrel{\text{IV}}{=} n + 1 + \sum_{i=2}^{n} i = \sum_{i=2}^{n+1} i$$

Folglich gilt nach dem Prinzip der vollständigen Induktion dass $T(n) = \sum_{i=2}^{n} i$ für alle $n \in \mathbb{N}$.
Weiter gilt dann

$$T(n) = \sum_{i=2}^{n} i = \sum_{i=1}^{n} i - 1 = \frac{n(n+1)}{2} - 1 \in \Theta(n^2)$$

denn für $n_0 = 2$ gilt für alle $n \in \mathbb{N}$ mit $n \geq n_0$ dass

$$\frac{n(n+1)}{2} - 1 \leq n^2 \qquad \text{sowie} \qquad \frac{n(n+1)}{2} - 1 \stackrel{n \geq 2}{\geq} \frac{n^2}{2}$$

Also existieren auch $c_1 = 1$ und $c_2 = \frac{1}{2}$ sodass stets gilt

$$\forall n \in \mathbb{N}, n \geq n_0 : c_2 n^2 \leq T(n) \leq c_1 n^2$$

Folglich ist $T(n) \in \Theta(n^2)$. □

Aufgabe 4

a) Die Iterationen lauten wie folgt:

```
[9,3,5,1,10,4,6]
[3,9,5,1,10,4,6]
[3,5,9,1,10,4,6]
[1,3,5,9,10,4,6]
[1,3,5,9,10,4,6]
[1,3,4,5,9,10,6]
[1,3,4,5,6,9,10]
```

b) Wir benennen den eigentlichen "roten" Pointer als `left` und den für die "blaue Region" `right`. Wir gehen nun wie folgt vor:

Zuerst lassen wir den Dutch-Flag Algorithmus auf dem ganzen Array laufen, sodass er das Array in 3 Bereiche unterteilt:
Im ersten Bereich (`0` bis `left`) sollen alle blauen Einträge sein, Im mittleren Bereich (`left+1` bis `right-1`) sollen zum Ende dann rote, schwarze, und gelbe Einträge in beliebiger Reihenfolge auftreten im rechten Bereich (`right` bis `E.length-1`) sollen dann alle grünen Einträge sein. Der Algorithmus unterscheidet hier also noch nicht zwischen roten, schwarzen und gelben Einträgen.

Damit sind die blauen und grünen Einträge schon an ihrer vorhergesehenen Position. Als nächstes lassen wir den Algorithmus ein zweites mal laufen, diesmal auf den noch unsortierten mittleren Bereich (`left+1` bis `right-1`) eingeschränkt. Wir belassen also die `left` und `right` pointer bei ihren Werten anstatt sie auf 0 bzw, `E.length` zu setzen. Der Pointer `u` wird dann auf `left+1` gesetzt. Dann wird der Algorithmus normal fortgeführt, sodass er die roten, schwarzen und gelben Einträge in den jeweils neuen linken, mittleren und rechten Bereich einsortiert.

Zum Ende ist dann auch der ursprünglich mittlere Teil des Arrays korrekt sortiert. Dies lässt sich in diesem Schema beliebig oft fortführen in dem man immer erst die beiden äußersten Bereiche sortiert und dann rekursiv den mittleren, bis man beim standard Dutch-Flag Problem für den Basecase ankommt.