Datenstrukturen und Algorithmen

Heimübung 4

Eli Kogan-Wang (7251030)

David Noah Stamm (7249709)

Daniel Heins (7213874)

Tim Wolf (7269381)

8. Mai 2022

Aufgabe 1

Herleitung der geschlossenen Formel T(n):

Angenommen $n = 4^k$

$$
\begin{aligned}
T(n) &= T\left(4^k\right) \\
&= 4T\left(\frac{4^k}{4}\right) + \sqrt{4^k} \cdot c_2 \\
&= 4 \cdot T\left(4^{k-1}\right) + 2^k \cdot c_2 \\
&= 4\left(4 \cdot T\left(4^{k-2}\right) + 2^{k-1} \cdot c_2\right) + 2^k \cdot c_2 \\
&= 4^2 \qquad \underbrace{T\left(4^{k-2}\right)}_{\text{sukzessives Einsetzen}} \qquad + 2^{k+1} \cdot c_2 + 2^k \cdot c_2 \\
&= 4^k c_1 + \left(\sum_{j=0}^{k-1} 2^{k+i}\right) c_2 \\
&= 4^k c_1 + \left(2^k \cdot \sum_{j=0}^{k-1} 2^i\right) c_2 && \mid \text{geometrische Summe} \\
&= 4^k c_1 + 2^k \left(\frac{1-2^k}{1-2}\right) c_2 \\
&= 4^k c_1 + 2^k \left(2^k - 1\right) c_2 \\
&= 4^k \left(c_1 + c_2\right) - 2^k c_2
\end{aligned}
$$

Korrektheitsbeweis der Formel durch vollständige Induktion:

Beh. :

$$T(n) = T(4^k) = 4^k(c_1 + c_2) - 2^k c_2$$

I.A.: $n = 1$

$T(1) = T(4^0) = c_1 = 4^0(c_1 + c_2) - 2^0 c_2$

I.V.: Es gilt: $T(4^{k-1}) = 4^{k-1}(c_1 + c_2) - 2^{k-1} c_2$

I.S.: Zu Zeigen: $T(n) = T(4^k) = 4^k(c_1 + c_2) - 2^k c_2$

Nach Definition von $T(n)$ gilt (da $n > 1$ ist):

$$
\begin{aligned}
T(n) &= T(4^k) \\
&= 4T(\frac{4^k}{4}) + \sqrt{4^k} \cdot c_2 \\
&= 4 \cdot T(4^{k-1}) + 2^k \cdot c_2 \\
&\overset{IV}{=} 4 \cdot (4^{k-1}(c_1 + c_2) - 2^{k-1}c_2) + \sqrt{4^k} \cdot c_2 \\
&= 4^k(c_1 + c_2) - 2^{k+1}c_2 + 2^k \cdot c_2 \\
&= 4^k(c_1 + c_2) - 2^k c_2
\end{aligned}
$$

$\square$

Aufgabe 2

Aus der Rekursionsgleichung folgt: $a = 2, b = 2$ und $f(n) = n \log n$

$\Rightarrow n^{\log_b(a)} = n^{\log_2(2)} = n^1$

$f(n) = n \log n = n^{\log_n(n \log_2 n)} = n^{\log_n(n) + \log_n \log_2(n)} = n^{1 + \log_n \log_2(n)}$

Das Mastertheorem kann nur angewendet werden,

falls f(n) $\in \mathcal{O}(n^{1-\varepsilon}) \vee f(n) \in \Theta(n) \vee (f(n) \in \Omega(n^{1+\varepsilon}) \wedge a \cdot f(\frac{n}{b}) \leq c \cdot f(n))$

Es gilt trivialerweise:

$$
\begin{aligned}
& n^{1+\log_n \log_2(n)} \notin \mathcal{O}(n) \\
\Rightarrow\ & n^{1+\log_n \log_2(n)} \notin \mathcal{O}(n^{1-\varepsilon}) \\
\text{und} \quad & n^{1+\log_n \log_2(n)} \notin \Theta(n)
\end{aligned}
$$

Ist $f(n) \in \Omega(n^{1+\varepsilon})$?

$\iff \exists c, n_0 > 0$, so dass $\forall n \geq n_0 : n^{1+\log_n \log_2(n)} \geq c \cdot n^{1+\varepsilon}$

Dies für $c = 1$ und $\varepsilon = \frac{1}{10}$ erfüllt

$\Rightarrow f(n) \in \Omega(n^{1+\varepsilon})$, aber die zweite Bedingung ist nicht erfüllt

$2 \cdot \frac{n}{2} \log(\frac{n}{2}) \leq c \cdot n \cdot \log(n)$

$n \cdot \log(\frac{n}{2}) \leq c \cdot n \cdot \log(n)$

Diese Bedingung wird nur von c = 1 erfüllt, da

$\lim\limits_{n\to\infty} \dfrac{n \cdot \log(\frac{n}{2})}{n \cdot log(n)}$

$= \lim\limits_{n\to\infty} \dfrac{\log(\frac{n}{2})}{\log(n)}$

Da $\lim\limits_{n\to\infty} \log(\dfrac{n}{2}) = \infty \wedge \lim\limits_{n\to\infty} \log(n) = \infty$ gilt,

kann die Regel vom Krankenhaus angewendet werden :°)

$\lim\limits_{n\to\infty} \dfrac{\log(\frac{n}{2})}{\log(n)} = \lim\limits_{n\to\infty} \dfrac{\frac{\mathrm{d}}{\mathrm{d}n}\log(\frac{n}{2})}{\frac{\mathrm{d}}{\mathrm{d}n}log(n)} = \lim\limits_{n\to\infty} \dfrac{\frac{\mathrm{d}}{\mathrm{d}n}\log(n) - \log(2)}{\frac{\mathrm{d}}{\mathrm{d}n}log(n)}$

$\lim\limits_{n\to\infty} \dfrac{\frac{1}{n}}{\frac{1}{n}} = \lim\limits_{n\to\infty} 1 = 1$ [d.h. sie sind asympt. äquiv.]

$\Rightarrow c = 1 \lightning c < 1$

$\Rightarrow$ Die Rekursionsgleichung kann nicht mit dem allgemeinen Master-Theorem analysiert werden

Aufgabe 3

Algorithm 1 3SORT(A, i, j)

if $j \leq i + 1$ **then**
  if $A[i] > A[j]$ **then**
    $A[i] \leftrightarrow A[j]$
  return
$k \leftarrow \left\lfloor \frac{j-i+1}{3} \right\rfloor$
3Sort$(A, i, j - k)$
3Sort$(A, i + k, j)$
3Sort$(A, i, j - k)$

b) Sei $n := len(A)$. Beweis über Induktion über n:

Basisfall:

$n = 0$: Trivial

$n = 1$: Trivial

$n = 2$:

Mit $a > b$:

$A \leftarrow [a, b]; \text{3Sort}(A, 0, 1) \implies A = [b, a]$

$A \leftarrow [b, a]; \text{3Sort}(A, 0, 1) \implies A = [b, a]$

Nun Induktion:

Sei $n \in \mathbb{N}$

Angenommen $3Sort(A, i, j)$ sortiert A wenn $j - i + 1 \leq n$

Zu zeigen ist: $3Sort(A, i, j)$ sortiert A wenn $j - i + 1 = n + 1$

Wir betrachten eine Ausführung von 3Sort mit $j - i + 1 = n + 1$:

3Sort in Zeile 6 sortiert $A[i : j - k]$, da

$$((j-k) - i + 1) = \left(\left(j - \left\lfloor \frac{j-i+1}{3} \right\rfloor\right) - i + 1\right) = \left\lceil \frac{2}{3} \cdot (j - i + 1) \right\rceil \leq n$$

$A = (a|b|c)$ wobei a, b, c die drittel von A sind.

Da $A[i : j - k]$ nun sortiert ist, ist bekannt, dass das zweite $\frac{1}{3}$ von A (b) größer als das erste $\frac{1}{3}$ von A (a) ist. Da jedes Element aus a nun mindestens $\frac{1}{3} \cdot (n+1)$ Elemente über sich hat mit wissen wir, dass kein Element der obersten $\frac{1}{3}$ von A (c) im ersten $\frac{1}{3}$ von A (a) ist. Alle Elemente, die am Ende in c sein sollen sind nun in $(b|c)$.

Nach der Ausführung von 3Sort in Zeile 7 ist damit c korrekt populiert.

Alle Elemente von a, b befinden sich in $(a|b)$ und c hat keine Elemente von a und b. Nach der Ausführung von 3Sort in Zeile 8 ist damit a, b korrekt populiert. c wurde nicht geändert und ist immernoch korrekt populiert. Nun ist $A = (a|b|c)$ sortiert.

c)

$$T(n) = \begin{cases} 1 & \text{falls } n = 1 \\ 3 \cdot T(\frac{2}{3}n) + 1 & \text{falls } n \geq 2 \end{cases}$$

Nach Mastertheorem:

$a = 3; b = \frac{3}{2}; c = 1$

Da $a > b$: $T(n) \in \Theta(n^{\log \frac{3}{2}(3)})$