Algorithmen & Datenstrukturen - Aufgaben zum 09. November 2015 (Blatt 03)

Tobias Knöppler (6523815), Nico Tress (6378086)

09.11.2015

3.1

$ALGO1()$: Da die beiden inneren Schleifen für jeden Durchlauf der äußeren je n mal durchlaufen werden (also 2n zusammen) und die äußere Schleife n+1 mal durchlaufen wird. daraus folgt $ALGO1() \in O((n+1) * 2n) = O(2n^2 + 2n)$.

$ALGO2()$: Da die äußere Schleife bis 2n läuft, i aber jeweils um 2 inkrementiert wird, wird die while-Schleife n mal durchlaufen. Die innere Schleife wird jeweils i mal ausgeführt, also insgesamt
$1 + 3 + 5 + ... + 2n - 3 + 2n - 1 = \sum_{i=1}^{n} 2i - 1$
$= 2n - 1 + 2n - 3 + ... + 4 - 1 + 2 - 1$
$= 2n + 2n + ... + 4 - 1 - 3 + 2 - 1 - 1$
$= \begin{cases} (n-1) \cdot 2n & \text{für gerade n} \\ 2n + (n-2) \cdot 2(n-1) & \text{für ungerade n} \end{cases}$
$= \begin{cases} 2n^2 - 2n & \text{für gerade n} \\ 2n + 2n^2 - 8n + 4 & \text{für ungerade n} \end{cases}$
$= \begin{cases} 2n^2 - 2n & \text{für gerade n} \\ 2n^2 - 6n + 4 & \text{für ungerade n} \end{cases}$
mal...
Damit gilt $ALGO2() \in O(2n^2 - 6n + 4) \in O(2n^2)$.

$ALGO3()$: Die äußere Schleife wird genau $\sqrt{n}$ mal ausgeführt. Die innere Schleife wird jeweils sooft durchlaufen, wie n durch 2 geteilt werden kann, ohne dass eine Zahl kleiner oder gleich 1 resultiert. Dies entspricht dem Exponenten der zu n nächstgrößeren 2er-Potenz. Daraus folgt für eine Zweierpotenz a mit $a > n, (a - n)$ so klein, wie möglich: $a = 2^x => ln(a) = x \cdot ln(2) => x = \frac{ln(a)}{ln(2)} = \frac{10}{7} \cdot ln(a) => ALGO3() \in O(\sqrt{n} \cdot \lceil \frac{10}{7} \cdot ln(n) \rceil)$

3.2

Dies ist eine Rekurrenzgleichung für den Algorithmus FUNC(A):
$T(x) := \begin{cases} 1, & wenn\ n < 4 \\ 5 + 3n + 2 \cdot T(\frac{n}{4}) & sonst \end{cases}$

Sie ergibt sich daraus, dass bei einer Eingabegröße (Länge von A) $<$ 4 lediglich 5 zurückgegeben

wird; die Komplexität ist also 1.
Ansonsten gilt (für $n =$ A.länge):

- 5 'einfache' Anweisungen werden unabhängig von der Eingabe ausgeführt (Zeilen 4, 7, 10 und der Zuweisungsanteil in den Zeilen 8 und 9).
- Die beiden for-Schleifen werden beide je A.länge mal durchlaufen; die erste enthält eine; die zweite 2 Anweisungen, daraus ergibt sich eine Komplexität von $3 \cdot n$.
- FUNC ruft sich selbst zweimal mit je einem Viertel des Arrays auf, daraus ergibt sich eine Komplexität von $2 \cdot T(n/4)$

3.3

1.

$T_2(n) := \begin{cases} c_1, & \text{für n=1} \\ 8 \cdot T_1\left(\frac{n}{2}\right) + d_1 \cdot n^3 & \text{sonst} \end{cases}$

daraus folgt nach dem Mastertheorem:

$$\begin{aligned} a &= 8 \\ b &= 2 \\ f(n) &= d_1 \cdot n^3 \\ \rightarrow log_b(a) &= log_2(8) = 3 \\ \Rightarrow T_2(n) &\in \Theta(n^3 \cdot log_2(n)), \text{ da } f(n) = d_1 \cdot n^3 \in \Theta(n^3) \end{aligned}$$

2.

$T_2(n) := \begin{cases} c_2, & \text{für n=1} \\ 5 \cdot T_2\left(\frac{n}{4}\right) + d_2 \cdot n^2 & \text{sonst} \end{cases}$

$$\begin{aligned} a &= 5 \\ b &= 4 \\ f(n) &= d_2 \cdot n^2 \\ \rightarrow log_b(a) &= log_4(5) = \frac{log(5)}{log(4} \approx 1,161 \\ \Rightarrow T_2(n) &\in \Theta(f(n)) \rightarrow T_2(n) \in \Theta(d_2 \cdot n^2), \text{ da } f(n) \in \Omega(n^{log_4(5)+\epsilon}) \text{ z.B. für } \epsilon = 1 \\ \text{wegen } &\lim_{n\to\infty}\left(\frac{d_2 \cdot n^2}{n^{1,161+1}}\right) \\ &= \lim_{n\to\infty}\left(\frac{d_2}{n^{0,161}}\right) = 0 \end{aligned}$$

3.

$$T_3(n) := \begin{cases} c_3, & \text{für n=1} \\ 6 \cdot T_3\left(\frac{n}{3}\right) + d_3 \cdot n \cdot log(n) & \text{sonst} \end{cases}$$

$$\begin{aligned}
a &= 6 \\
b &= 3 \\
f(n) &= d_3 \cdot n \cdot log(n) \\
\rightarrow log_b(a) &= log_3(6) = \frac{log(6)}{log(3} \approx 1,631
\end{aligned}$$

$\Rightarrow T_3(n) \in \Theta(f(n)) \rightarrow T_2(n) \in \Theta(d_2 \cdot n^2)$, da $f(n) \in \Omega(n^{log_4(5)+\epsilon})$ z.B. für $\epsilon = 1$

$$\begin{aligned}
\text{wegen } &\lim_{n\to\infty}\left(\frac{d_3 \cdot n \cdot log(n)}{n^{1,631}}\right) \\
&= \lim_{n\to\infty}\left(\frac{d_3 \cdot log(n)}{n^{0,631}}\right) \\
&\overset{H}{=} \lim_{n\to\infty}\left(\frac{d_3}{n} \cdot \frac{n^{-0,369}}{0,631}\right) \\
&= \lim_{n\to\infty}\left(\frac{d_3 \cdot n^{0,369}}{0,631 \cdot n}\right) \\
&= \lim_{n\to\infty}\left(\frac{d_3 \cdot n^{0,369}}{0,631 \cdot n}\right) \\
&= \lim_{n\to\infty}\left(\frac{d_3}{0,631 \cdot n^{0,631}}\right) \\
&= 0
\end{aligned}$$

3.4

1.

```
merge(A, B):
  O = empty list
  while A is not empty and B is not empty:
    if A[0] < B[0]:
      append A[0] to O
      remove A[0]
    else
      append B[0] to O
      remove B[0]
    if A is empty
      return concat(O, B)
```

2.

```
numberOfConflicts(A)
  pendMax = 0
  numConflicts = 0
  for i=1 to A.length
```