Prof. Dr. Ir. Joost-Pieter Katoen

Sebastian Junges, Benjamin Kaminski, David Korzeniewski, Tim Quatmann

Übung 5

– Musterlösung –

Hinweise:

- Die Lösungen müssen bis **Donnerstag, den 17. Mai um 16:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Die Übungsblätter **müssen** in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung abgegeben werden.
- Drucken Sie ggf. digital angefertigte Lösungen aus. Abgaben z.B. per Email sind nicht zulässig.
- Namen und Matrikelnummer sowie die **Nummer der Übungsgruppe** sind auf jedes Blatt der Abgabe zu schreiben. Abgaben, die aus mehreren Blättern bestehen **müssen geheftet bzw. getackert** werden! Die **Gruppennummer muss sich auf der ersten Seite oben links** befinden.
- **Bei Nichtbeachten der obigen Hinweise müssen Sie mit erheblichen Punktabzügen rechnen!**

Aufgabe 1 (Rekursionsgleichung): (10 + 10 = 20 Punkte)

Hinweise:

- Manchmal bietet es sich an, die Rekursiongleichung zu vereinfachen.

a) Beweisen Sie per Induktion, dass $T(n) \in \mathcal{O}(n \log n)$ für

$$T(n) = \begin{cases} 1 & 1 \leq n < 16 \\ 2 \cdot T(\frac{n}{4}) + T(\frac{n}{2}) + n & n \geq 16 \end{cases}$$

b) Beweisen Sie per Induktion, dass $T(n) \in \Omega(3^{\frac{n}{3}})$ für

$$T(n) = \begin{cases} 1 & 1 \leq n \leq 3 \\ T(n-1) + T(n-2) + T(n-3) & n > 3 \end{cases}$$

Lösung:

a) Zu zeigen: Es existiert ein n_0 sodass für $n > n_0$:

$$\exists c > 0. T(n) \leq c \cdot n \log n$$

Induktionsanfang: $T(n) = 1 \leq c \cdot n \log n$ für alle $4 \leq n \leq 16$ und für alle $c > \frac{1}{4}$.

Induktionsannahme: $T(m) \leq c \cdot m \log m$ für alle $4 \leq m < n$.

Induktionsschritt (mit $n > 16$):

$$\begin{aligned} T(n) = & \quad 2 \cdot T(\frac{n}{4}) + T(\frac{n}{2}) + n \\ \leq & \quad 2 \cdot c \cdot \frac{n}{4} \cdot \log \frac{n}{4} + c \cdot \frac{n}{2} \log \frac{n}{2} + n \\ = & \quad \frac{cn}{2}(\log n - 2) + \frac{cn}{2}(\log n - 1) + n \\ \leq & \quad \frac{cn}{2} \log n + \frac{cn}{2} \log n + n \\ = & \quad cn \log n + n \end{aligned}$$

D.h. die Aussage gilt für $n_0 = 4$ und $c = 1$.

b) Wir zeigen (per Induktion) $T(n) > 0$.

- Induktionsanfang: $T(1) = T(2) = T(3) = 1 > 0$.
- Induktionsannahme: $T(m) > 0$ für alle $m < n$ für ein $n \in \mathbb{N}$.
- Induktionsschritt (für $n > 3$): $T(n+1) = T(n) + T(n-1) + T(n-2) > 0 + 0 + 0 = 0$.

Wir zeigen nun per Induktion $T(n+2) \geq T(n+1) \geq T(n)$.

- Induktionsanfang: $T(4) \geq T(3) \geq T(2) \geq T(1)$.
- Induktionsannahme: $T(n+2) \geq T(n+1) \geq T(n)$ für ein $n \in \mathbb{N}$.
- Induktionsschritt: $T(n+2+1) = T(n+2) + T(n+1) + T(n) \geq T(n+2) + 0 + 0 = T(n+2)$ und $T(n+1+1) = T(n+2) \geq T(n+1)$.

Per transitivität folgt $T(n) \geq T(m)$ für alle $m < n$.

Dann gilt:

$T(n) \geq 3 \cdot T(n-3)$.

Wir zeigen nun $T(n) \geq c \cdot 3^{\frac{n}{3}}$ für $c = \frac{1}{3}$.

- Induktionsanfang: $T(n) = 1 = \frac{1}{3} \cdot 3^1 \geq \frac{1}{3} \cdot 3^{\frac{n}{3}}$ für $n \leq 3$.
- Induktionsannahme: $T(m) \geq 3 \cdot T(m-3)$ für alle $m < n$ für ein $n \in \mathbb{N}$.
- Induktionsschritt: Nun $T(n) \geq c \cdot 3^{\frac{n}{3}}$.

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

Aufgabe 2 (Master Theorem): (8 + 8 + 8 + 8 + 8 = 40 Punkte)

Geben Sie für die folgenden Rekursionsgleichungen an, ob diese mit dem Master-Theorem aus der Vorlesung gelöst werden können. Begründen Sie ihre Antwort! Geben Sie auch die resultierende Komplexitätsklasse an, sofern das Master-Theorem angewendet werden kann.

a) $T(n) = 8 \cdot T(\frac{n}{2}) + 2^n$

b) $T(n) = 64 \cdot T(\frac{n}{4}) + (n^2 + 1) \cdot (n + 7)$

c) $T(n) = 27 \cdot T(\frac{n}{3}) + n \cdot (n^2 \log(n) + n)$

d) $T(n) = 16 \cdot T(\frac{n}{4}) + n \cdot (\log(n) + n)$

e) $T(n) = T(\frac{n}{3}) + f(n)$, wobei $f(n) = \begin{cases} 3n + 2^{3n} & \text{falls } n \in \{2^i | i \in \mathbb{N}\} \\ 3n & \text{sonst} \end{cases}$

Lösung:

a) Es sind $b = 8, c = 2$ und $f(n) = 2^n$. E wird nun wie folgt bestimmt:

$$E = \frac{\log(8)}{\log(2)} = \frac{3}{2}$$

Damit gilt $n^E = n^{\frac{3}{2}}$. Wähle $\epsilon = \frac{1}{2}, c' = 1$ und $n_0 = 4$, dann gilt

$$\forall n \geq n_0 : c' \cdot n^{E+\epsilon} = n^2 \leq 2^n = f(n)$$

und somit $f(n) \in \Omega(n^{E+\epsilon})$.
Sei $d = \frac{1}{2}$, dann gilt für genügend große n:

$$b \cdot f(\tfrac{n}{c}) = 3 \cdot 2^{\frac{n}{2}} \leq \tfrac{1}{2} 2^n = 2^{n-1} = d \cdot f(n)$$

Somit sind beide Bedingungen für den 3. Fall des Master-Theorems erfüllt. Es gilt also $T(n) \in \Theta(2^n)$.

b) Es sind $b = 64, c = 4$ und $f(n) = (n^2 + 1) \cdot (n + 7) = n^3 + 7n^2 + n + 7$. E wird nun wie folgt bestimmt:

$$E = \frac{\log(64)}{\log(4)} = \frac{6}{2} = 3$$

Damit gilt $n^E = n^3$. Wähle $c_1 = 1, c_2 = 16$ und $n_0 = 1$, dann gilt

$$\forall n \geq n_0 : c_1 \cdot n^E = n^3 \leq n^3 + 7n^2 + n + 7 = f(n) \leq 16n^3 = c_2 \cdot n^E$$

und somit $f(n) \in \Theta(n^E)$
Damit sind die Bedingungen für den 2. Fall des Master-Theorems erfüllt. Es gilt also $T(n) \in \Theta(n^3 \log n)$.

c) Es sind $b = 27, c = 3$ und $f(n) = n \cdot (n^2 \log(n) + n) = n^3 \log(n) + n^2$. E wird nun wie folgt bestimmt:

$$E = \frac{\log(27)}{\log(3)} = \frac{3}{1} = 3$$

Damit gilt $n^E = n^3$. Da $f(n)$ um mehr als einen konstanten Faktor schneller wächst als n^E gilt $f(n) \notin \mathcal{O}(n^E)$. Also kommt nur der 3. Fall in Frage.

$$\begin{aligned}
\lim_{n\to\infty} \frac{f(n)}{n^{E+\epsilon}} &= \lim_{n\to\infty} \frac{n^3\log(n)+n^2}{n^3+\epsilon} \\
&= \lim_{n\to\infty} \frac{n^3\ln(n)+n^2}{\ln(2)n^3+\epsilon} \\
&= \lim_{n\to\infty} \frac{6}{\ln(2)\cdot n\cdot\epsilon(\epsilon+1)(\epsilon+2)(\epsilon+3)n^{\epsilon-1}} && \text{vier Mal l'Hospital} \\
&= \lim_{n\to\infty} \frac{6}{\ln(2)\cdot\epsilon(\epsilon+1)(\epsilon+2)(\epsilon+3)n^{\epsilon}} \\
&= 0
\end{aligned}$$

und somit $f(n) \notin \Omega(n^{E+\epsilon})$

Das Master-Theorem ist also nicht anwendbar.

d) Es sind $b = 16, c = 4$ und $f(n) = n\cdot(\log(n)+n) = n\log(n)+n$. E wird nun wie folgt bestimmt:

$$E = \frac{\log(16)}{\log(4)} = \frac{4}{2} = 2$$

Damit gilt $n^E = n^2$. Da $f(n)$ um mehr als einen konstanten Faktor langsamer wächst als n^E, kommt nur der 1. Fall in Frage.

Wähle $\epsilon = \frac{1}{2}$.

$$\begin{aligned}
\lim_{n\to\infty} \frac{f(n)}{n^{E-\epsilon}} &= \lim_{n\to\infty} \frac{n\log(n)+n}{n^{\frac{3}{2}}} \\
&= \lim_{n\to\infty} \frac{n\ln(n)+n}{\ln(2)\cdot n^{\frac{3}{2}}} \\
&= \lim_{n\to\infty} \frac{3}{\ln(2)\cdot n\cdot 4n^{\frac{1}{2}}} && \text{zwei Mal l'Hospital} \\
&= 0
\end{aligned}$$

und somit $f(n) \in \mathcal{O}(n^{E-\epsilon})$

Damit sind die Bedingungen für den 1. Fall des Master-Theorems erfüllt. Es gilt also $T(n) \in \Theta(n^2)$.

e) Es sind $b = 1$ und $c = 3$

$$E = \frac{\log(1)}{\log(3)} = \frac{0}{1} = 0$$

Damit gilt $n^E = n^0$. Wähle $\epsilon = 1$. Dann gilt $f(n) \in \Omega(n^{E+\epsilon})$ (z.B. mit $c' = 1, n_0 = 0$) und $f(n) \notin \mathcal{O}(n^E)$, da 2^{3n} für große n um mehr als einen konstanten Faktor größer ist als n^1.

Es kommt also nur der 3. Fall des Master-Theorems in Frage. Die zweite Bedingung trifft aber nicht zu, da für alle n, sodass $\frac{n}{3}$ eine ganzzahlige Zweierpotenz ist gilt:

$$f(n) = 3n < 2^n + n = f(\tfrac{n}{3})$$

Somit ist das Master-Theorem nicht anwendbar.

Aufgabe 3 (Analyse): (5 + 4 + 6 + 5 + 5 = 25 Punkte)

Betrachten Sie folgenden Algorithmus.

```
T(Liste L der Laenge n mit natuerlichen Zahlen) -> Liste:
    if length(L) == 1:
        return L
    print "T: " + L
    i = S(L, 0, 0, -1)
    swap(L[0], L[i])
    return [L[0]] + T(L.tail)

S(Liste L, integer a, integer b, integer c) -> integer:
    print "S: " + L + " " + [a, b, c]
    if length(L) == 0:
        return b
    if L.head > c:
        return S(L.tail, a+1, a, L.head)
    else:
        return S(l.tail, a+1, b, c)
```

Hinweise:

- Die Liste L.tail beschreibt die Liste L ohne das erste Element.
- Für die Komplexität zählen Sie bitte die Anzahl Aufrufe von S.
- + zwischen Listen beschreibt das konkatenieren,

a) Bestimmen Sie die ersten 11 Ausgaben von print beim Aufruf $T([3,1,2,4])$. Was ist die Rückgabe von $T([3,1,2,4])$?

b) Beschreiben Sie in Stichpunkten die Bedeutung von S(...), und den Variablen a, b, c, und i.

c) Geben Sie eine Rekursionsgleichung für $T(n)$ und bestimmen Sie die Asymptotische Komplexität von $T(n)$ mittels der Rekursionsgleichung.

d) Geben Sie eine Liste L der Länge 6, sodass T(L) 5 Swaps ausführt.

e) Gibt es auch eine Liste L der Länge n, sodass T(L) n Swaps ausführt? Geben Sie eine kurze Begründung.

Lösung:

a) 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
T: [2, 1]

b) Selection-sort bezüglich $>$.

- S(...) lineare suche,
- a, aktueller Index
- b, Index größte Zahl
- c, größte Zahl
- i, Index größte Zahl

c) $T(n) = T(n-1) + S(n)$.
$S(n) = S(n-1) + 1$, demnach $S(n) = n$.
$T(n) = T(n-1) + n$, demnach $T(n) = \sum_{i=1}^{n} i \in \Theta(n^2)$.

d)

2	3	4	5	6	1

e) $n-1$ Swaps ist das Maximum. Nach m swaps sind sicherlich die ersten m Einträge sortiert. Nach $n-1$ Swaps sind also die ersten $n-1$ Einträge sortiert. In Iteration i wird das $i$te element nur mit Elementen $> i$ geswapt. In Iteration n kann also kein Element geswapt werden.

Aufgabe 4 (Sortieren): (7+8=15 Punkte)

a) Sortieren Sie das folgende Array mithilfe von Insertionsort. Geben Sie dazu das Array nach jeder Iteration der äußeren Schleife an.

Hinweise:

- Geben Sie *genau* die gewünschten Schritte an.

9	3	5	1	10	4	6

b) Wir definieren das UCI-Flaggen Problem. Die UCI Flagge besteht aus 5 Streifen mit den Farben mit den Farben Blau, Rot, Schwarz,Gelb, Grün. Gegeben ein Array mit gefärbten Einträgen (Blau, Rot, Schwarz, Gelb, Grün). Das Problem besteht darin, in linearer Laufzeit und konstantem Speicher, das Array so zu sortieren, wie sie in UCI Flagge auftreten. Dabei darf angenommen weden, dass Pointer in das Array nur konstantem Speicher benötigen.

Beschreiben Sie, wie die Lösung zum Dutch Flag Problem genutzt werden kann, um das UCI-Flaggen Problem zu lösen.

Hinweise:

- Sie sollen keine anderen Algorithmen zum sortieren nutzen.

- Eine einfache, elegante Lösung ergibt mehr Punkte.

Lösung:

a) Mit Insertion-sort:

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 wenden das Dutch-Flag Problem 2-Mal an: Im ersten durchlauf nehmen wir an, Blau Rot im Algorithmus entspricht, Rot, Schwarz und Gelb entsprechen Weiss im Algorithmus, und Grün entspricht Blau im Algorithmus. Anwendung vom Dutch Flag Algorithmus hat dann Blau und Grün an die richtige Stelle sortiert. Im zweiten durchlauf sortieren wir nur noch den inneren Teil, d.h. die Farben Rot, Schwarz, und Gelb, denen wir jeweils den Farben Rot, Weiss, und Blau entsprechen lassen.