Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Jan Küntzer

Programmierung II, SS 2004
Musterlösung zur 10. Übung

Aufgabe 1: Rekurrenz, Substitutionsmethode (10 Punkte)

i) $T(n) = 3T(\lfloor \frac{n}{2} \rfloor) + n^2$

Es liegt wohl in $\mathcal{O}(n^2)$, was wir nun per Induktion beweisen:

Behauptung: $T(n) \leq 4n^2$

Beweis. n=1: $T(1) = 1 \leq 4$

Induktionsschritt:

$$\begin{aligned} T(n) &= 3T(\lfloor \frac{n}{2} \rfloor) + n^2 \\ &\leq 3 \cdot 4(\frac{n}{2})^2 + n^2 \\ &= 4n^2 \end{aligned}$$

■

ii) $T(n) = 3T(\lfloor \frac{n}{8} \rfloor) + T(\lfloor \frac{n}{4} \rfloor) + 1$

Es liegt wohl in $\mathcal{O}(n)$. Versuchen wir aber induktiv zu zeigen, dass $T(n) \leq cn$, so stossen wir auf ein Problem, da sich unsere Induktionsannahme als zu schwach herausstellen wird, um die Annahme zu beweisen. Wir zeigen statt dessen die (stärkere) Annahme: $T(n) \leq cn - b$ für ein $b > 0$.

Beweis. n=1: $T(1) = 1 \leq c - b \qquad c \geq 1 + b$

Induktionsschritt:

$$\begin{aligned} T(n) &= 3c\lfloor \frac{n}{8} \rfloor - 3b + c\lfloor \frac{n}{4} \rfloor - b + 1 \\ &\leq \frac{5}{8}cn - 4b + 1 \\ &\leq cn - 4b + 1 \\ &\leq cn - b \text{ für } b \geq \frac{1}{3} \end{aligned}$$

■

iii) $T(n) = 2T(\lfloor \frac{n}{4} \rfloor) + 8T(\lfloor \frac{n}{16} \rfloor) + n$

Es liegt wohl in $\mathcal{O}(n \log(n))$. Versuchen wir induktiv zu zeigen, dass $T(n) \leq n \log(n)$:

Beweis. n=1: $T(1) = 1$

Induktionsschritt:

$$\begin{aligned} T(n) &\leq 2((\log(n) - \log(4)) \cdot \frac{n}{4}) + 8((\log(n) - \log(16))\frac{n}{16}) + n \\ &= \frac{n}{2}(\log(n) - 2) + \frac{n}{2}(\log(n) - 4) + n \\ &= n \log(n) - 2n \\ &\leq n \log(n) \end{aligned}$$

■

Aufgabe 2: Rekurrenz, Mastertheorem (10 Punkte)
Das Mastertheorem ermöglicht es, sehr einfach die Lösung von Rekurrenzen der Form $T(n) = aT(\frac{n}{b}) + f(n)$ anzugeben. Dabei wird überprüft, ob die Funktion f oder der Teil $aT(\frac{n}{b})$ die Kosten dominieren werden. Allerdings gibt es zwei Definitionslücken, in denen das Theorem keine Aussage machen kann, nämlich dann, wenn $f(n)$ zwar kleiner (größer) als der von der Rekursion herrührende Term ist, aber nicht *polynomiell* kleiner (größer).

i) $T(n) = 49\sqrt{7}T(\frac{n}{7}) + 7\sqrt{n^5} + 49\sqrt{7}$

Hier gilt $f(n) := 7n^{\frac{5}{2}} - 49\sqrt{7}$ und $n^{\log_7(49\sqrt{7})} = n^{\log_7(\sqrt{7^5})} = n^{\frac{5}{2}}$. Offenbar gilt $f(n) \in \Theta(n^{\frac{5}{2}})$, und damit nach dem Fall 2 des Mastertheorems: $T(n) \in \Theta(n^{\frac{5}{2}} \log_7(n))$

ii) $T(n) = 27T(\frac{n}{3}) + \frac{n^3}{\log_9^3(n)}$

Für die Anwendung des Mastertheorems müssen wir zunächst $f(n) := \frac{n^3}{\log_9^3(n)}$ und $n^{\log_3(27)} = n^3$ miteinander vergleichen. Offenbar gilt $f(n) \in \mathcal{O}(n^3)$! Aber um den ersten Fall des Mastertheorems anwenden zu können, müsste $f(n)$ sogar polynomiell kleiner sein als n^3, also müsste ein $\epsilon > 0$ existieren, so dass $f(n) \in \mathcal{O}(n^{3-\epsilon})$. Da aber der Logarithmus *langsamer* wächst als $jedePotenzn^{\epsilon}$, können wir kein $\epsilon > 0$ finden mit $\frac{n^3}{\log_9^3(n)} \in \mathcal{O}(\frac{n^3}{n^{\epsilon}})$, und damit können wir mit dem Mastertheorem hier keine Aussage treffen!

iii) $T(n) = 9T(\frac{n}{3}) + 3\log(n)n^2$

Hier gilt $f(n) := 3\log(n)n^2$ und $n^{\log_3(9)} = n^2$. Damit gilt natürlich auch $f(n) \in \Omega(n^2)$, aber können wir ein $\epsilon > 0$ finden, so dass $f(n) \in \Omega(n^{2+\epsilon})$? Nein, denn der Logarithmus wächst langsamer als jede Potenz, wie man sich leicht klarmachen kann.

Und damit können wir das Mastertheorem hier nicht anwenden!

iv) $T(n) = 15625T(\frac{n}{5}) + \frac{1}{37}n^7$

Hier gilt $f(n) := \frac{1}{37}n^7$ und $n^{\log_5(15625)} = n^6$. Offenbar gilt damit auch $f(n) \in \Omega(n^6)$, und da f(n) schneller wächst als n^6, können wir ein beliebiges $0 < \epsilon < 1$ wählen, so dass auch $f(n) \in \Omega(n^{6+\epsilon})$. Nun bleibt für den 3. Fall des Mastertheorems noch zu zeigen, dass für irgendein $c < 1$ und genügend großes n auch $15625f(\frac{n}{5}^7) \leq cn^7$ gilt. Dazu betrachten wir den Limes $\lim_{n\to\infty} \frac{15625(\frac{n}{5}^7)}{n^7} = (\frac{1}{5})^7 \cdot 15625 = \frac{1}{5}$, und damit ist die Behauptung gezeigt. Also gilt nach dem Fall 3 des Mastertheorems: $T(n) \in \Theta(n^7)$

v) $T(n) = 1296T(\frac{n}{6}) + n^3\sqrt{n}\log(n)$

Hier gilt $f(n) := n^3\sqrt{n}\log(n) = n^{\frac{7}{2}}\log(n)$ und $n^{\log_6(1296)} = n^4$. Damit ist offenbar $f(n) \in \mathcal{O}(n^4)$, und nun müssen wir für den 1.Fall des Mastertheorems noch ein $\epsilon > 0$ finden, so dass auch $f(n) = n^{\frac{7}{2}}\log(n) \in \mathcal{O}(n4 - \epsilon)$ liegt.

Da $\log(n)$ langsamer wächst als $n^{\frac{1}{4}}$, ist offenbar $f(n) \in \Omega(n^{\frac{15}{4}})$, und mit $\epsilon = \frac{1}{4}$ folgt $f(n) \in \mathcal{O}(n^{4-\epsilon})$. Damit folgt nach dem Fall 1 des Mastertheorems: $T(n) \in \Theta(n^4)$

Aufgabe 3: Rot-Schwarz-Baum (10 Punkte)

i) Siehe Abb. (a).

ii) Der angegebene Baum war kein Rot-Schwarz-Baum. Wer dies bewiesen hat, musste damit keinen Knoten löschen. Wenn man dennoch den Knoten löscht, erhält man den Baum in Abb. (b).

Aufgabe 4: Binäre Suchbäume (10 Punkte)

i) Da die in der Vorlesung vorgestellte Implementierung in diesem Fall eine lineare Kette, also einen maximal unbalancierten Baum erzeugt, und jeweils unten an den letzten Knoten angehängt wird, muss vor jedem *einfügen* die gesamte bisherige Kette abgelaufen werden. Also läuft die i-te *einfüge*-Operateion in $\mathcal{O}(i)$. Damit haben wir als Gesamtkosten für die Sequenz: $\sum_{i=1}^{n} ci = \frac{1}{2}n(n+1) \in \mathcal{O}(n^2)$.

ii) In diesem Fall erzeugen wir offenbar einen balancierten binären Baum! Im i-ten *einfüge*-Aufruf kann dieser dann maximal die Höhe $\log_2(i)$ besitzen, und wir erhalten als Gesamtkosten der Sequenz: $\sum_{i=1}^{n} \log_2(i) = \log_2(\prod_{i=1}^{n} i) = \log_2(n!)$

iii) In diesem Fall erhalten wir einen Baum mit genau einem Knoten. Da wir z.B. vorne an die Liste in $\mathcal{O}(1)$ anhängen können, erhalten wir eine Gesamtlaufzeit von $\mathcal{O}(n)$.

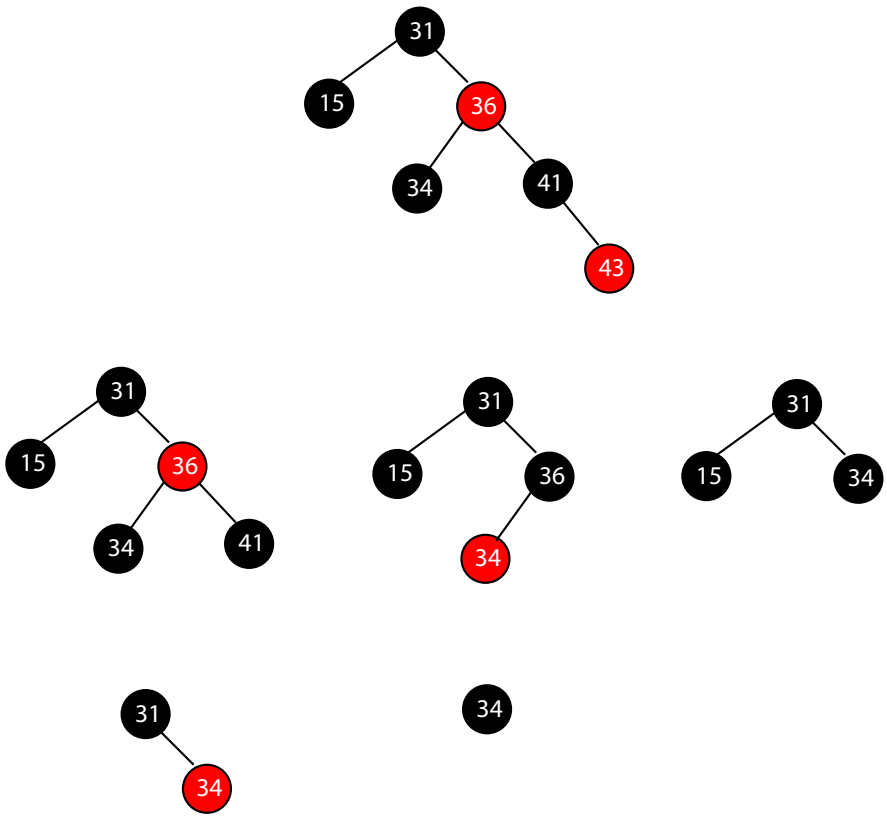

Abbildung 1: Der obere Baum entsteht durch das Einfügen der Knoten. Die einzelnen Teilbäume, die beim Löschen entstehen, sind darunter angeführt.

iv) Im worst-case erhalten wir genau das Verhalten, das wir aus dem Algorithmus der Vorlesung kennen, d.h. wir erzeugen in $\mathcal{O}(n^2)$ eine lineare Kette.

Wie sieht es nun im Durchschnitt aus? Ganz informell gilt: der Baum wächst an allen Enden etwa gleich schnell. Daher erwarten wir einen balancierten Baum und damit das Verhalten wie im 2. Fall $\Rightarrow \in \mathcal{O}(n \log_2(n)$.

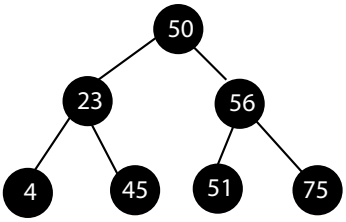

Abbildung 2: Der Rot-Schwarz-Baum nach dem Löschen des Knotens.