Johannes Blömer
u.v.a.m.

Paderborn, den 17. Mai

Musterlösungen zur Vorlesung

Datenstrukturen und Algorithmen

SS 2004

Blatt 3

AUFGABE 1:
Wir schreiben zunächst alle Ausdrücke zur Basis 2. Wir erhalten

$$(3/2)^n = 2^{\log(3/2)\cdot n} \quad n^3 = 2^{\log(3)\cdot\log(n)} \quad (\log(n))^{\log(n)} = 2^{\log\log(n)\cdot\log(n)} \quad 4^{\log(n)} = 2^{2\cdot\log(n)}.$$

Um die ursprünglichen Funktionen gemäß ihres Wachstums zu ordnen, genügt es die Funktionen in den Exponenten zu ordenen. Zunächst gilt offensichtlich

$$2\cdot\log(n) \leq 3\cdot\log(n) \quad \text{für alle } n \geq 1.$$

Weiter gilt

$$\log\log(2^8) = 3.$$

Da die Funktion $\log\log(n)$ streng monoton wächst, gilt daher

$$3\cdot\log(n) \leq \log\log(n)\cdot\log(n) \text{ für alle } n \geq 2^8.$$

Schließlich gilt noch $\log(3/2) \geq 1/2$ und damit

$$\log(3/2)\cdot n \geq \log\log(n)\log(n) \quad \text{für } n = 2^4.$$

Dann aber gilt

$$\log(3/2)\cdot n \geq \log\log(n)\log(n) \quad \text{für alle } n \geq 2^4.$$

Damit können wir unsere ursprünglichen Funktionen wie folgt ordnen:

$$4^{\log(n)} = \mathcal{O}(n^3) = \mathcal{O}((\log(n))^{\log(n)}) = \mathcal{O}((3/2)^n).$$

AUFGABE 2:
Wir betrachten zunächst die Rekursion $T(n) = 7T(n/2)+n^2$. Wir nehmen zur Vereinfachung immer nur natürliche Zahlen n, die eine Zweierpotenz sind. Durch sukzessives Einsetzen der Rekursionsgleichung für $\frac{n}{2^i}$ für $i = 1,\ldots\log(n)$ in sich selbst und Setzung von $T(1) = c$ für

eine Konstante c erhalten wir

$$
\begin{aligned}
T(n) &= 7T(n/2) + n^2 \\
&= 7\left(7T(\frac{n}{4}) + \frac{n^2}{4}\right) + n^2 \\
&= 49T(\frac{n}{4}) + \frac{7}{4}n^2 + n^2 \\
&= 49\left(7T(\frac{n}{8}) + (\frac{n}{4})^2\right) + \frac{7}{4}n^2 + n^2 \\
&= 7^3T(\frac{n}{8}) + \frac{49}{16}n^2 + \frac{7}{4}n^2 + n^2 \\
&\vdots \quad \vdots \\
&= 7^iT(\frac{n}{2^i}) + (\frac{7}{4})^{i-1}n^2 + \cdots + \frac{7}{4}n^2 + n^2 \\
&= 7^iT(\frac{n}{2^i}) + n^2\sum_{j=0}^{i-1}(\frac{7}{4})^j \\
&\vdots \quad \vdots \\
&= 7^{\log(n)}T(1) + n^2\sum_{j=0}^{\log(n)-1}(\frac{7}{4})^j \\
&= c7^{\log(n)} + n^2\frac{(7/4)^{\log(n)} - 1}{7/4 - 1} \\
&= cn^{\log(7)} + \frac{4}{3}n^2 \cdot n^{\log(7)-\log(4)} - \frac{4}{3}n^2 \\
&= cn^{\log(7)} + \frac{4}{3}n^{\log(7)} - \frac{4}{3}n^2 \qquad = \mathcal{O}(n^{\log(7)}).
\end{aligned}
$$

Bei der Rekursionsgleichung $T(n) = aT(n/4) + n^2$ gehen wir analog vor und erhalten unter der Annahme, dass n von der Form $n = 4^k$ ist

$$
\begin{aligned}
T(n) &= aT(n/4) + n^2 \\
&= a\left(aT(\frac{n}{16}) + (\frac{n}{4})^2\right) + n^2 \\
&= a^2T(\frac{n}{16}) + \frac{a}{16}n^2 + n^2 \\
&= a^2\left(aT(\frac{n}{4^3}) + (\frac{n}{16})^2\right) + \frac{a}{16}n^2 + n^2 \\
&= a^3T(\frac{n}{4^3}) + \frac{a^2}{16^2}n^2 + \frac{a}{16}n^2 + n^2 \\
&\vdots \quad \vdots \\
&= a^iT(\frac{n}{4^i}) + (\frac{a}{16})^{i-1}n^2 + \cdots + \frac{a}{16}n^2 + n^2 \\
&= a^iT(\frac{n}{4^i}) + n^2\sum_{j=0}^{i-1}(\frac{a}{16})^j \\
&\vdots \quad \vdots \\
&= a^{\log_4(n)}T(1) + n^2\sum_{j=0}^{\log(n)-1}(\frac{a}{16})^j \\
&= ca^{\log_4(n)} + n^2\frac{(a/16)^{\log_4(n)} - 1}{a/16 - 1} \\
&= cn^{\frac{1}{2}\log(a)} + \frac{1}{a/16-1}n^2 \cdot n^{\frac{1}{2}(\log(a)-\log(16))} - \frac{1}{a/16-1}n^2 \\
&= \left(c + \frac{1}{a/16-1}\right)n^{\frac{1}{2}\log(a)} - \frac{1}{a/16-1}n^2.
\end{aligned}
$$

Dabei haben wir vom Übergang von der viert- zur drittletzten Zeile angenommen, dass $a \neq 16$. Auf den Fall $a = 16$ kommen wir später zurück.
Nehmen wir nun zunächst $a < 16$ an. Dann gilt $\frac{1}{2}\log(16) < 2$ und $T(n) = \mathcal{O}(n^2)$. Dann ist Algorithmus B immer schneller als Algorithmus A.
Gilt $a > 16$, so gilt $T(n) = \mathcal{O}(n^{\frac{1}{2}\log(a)})$. Damit nun B asymptotisch schneller als A ist, muss gelten

$$\frac{1}{2}\log(a) < \log(7) \text{ oder } a < 49.$$

Schliesslich betrachten wir noch den Fall $a = 16$. Dann betrachten wir die viertletzte Zeile in

unseren Umformungen oben und erhalten

$$\begin{aligned} T(n) &= 16^{\log_4(n)}T(1) + n^2 \sum_{j=0}^{\log(n)-1} (\frac{16}{16})^j \\ &= c16^{\log_4(n)} + n^2 \sum_{j=0}^{\log(n)-1} 1 \\ &= cn^{\frac{1}{2}\log(16)} + n^2\log(n) \qquad\qquad = \mathcal{O}(n^2\log(n)). \end{aligned}$$

In diesem Fall ist B schneller als A.

AUFGABE 3:
Das Array ist kein max-Heap. Es gilt $A[4] = 6 < A[9] = 7$, aber Knoten 9 ist rechtes Kind von Knoten 4. Vertauschen wir $A[4]$ und $A[9]$, so erhalten wir den max-Heap $[23, 17, 14, 7, 13, 10, 1, 5, 6, 12]$.

AUFGABE 4:
Der Algorithmus MIN-HEAPIFY arbeitet vollkommen analog zum Algorithmus MAX-HEAPIFY. Wir vertauschen nur den Vergleichsoperator $>$ durch den Vergleich $<$. Die Korrektheit von MIN-HEAPIFY ergibt sich dann genau wir die Korrektheit von MAX-HEAPIFY. Hier ist der Algorithmus in Pseudocode.

MIN-HEAPIFY(A, i)

```
 1 l ← LEFT(i)
 2 r ← RIGHT(i)
 3 if l ≤ heap-size[A] und A[l] < A[i]
 4    then smallest ← l
 5    else smallest ← i
 6 if r ≤ heap-size[A] und A[r] < A[smallest]
 7    then smallest ← r
 8 if smallest ≠ i
 9    then A[i] ↔ A[smallest]
10         MIN-HEAPIFY(A, smallest)
```

Nun zur Laufzeit von MIN-HEAPIFY. Wie bei MAX-HEAPIFY benötigen die Zeilen 1-9 jeweils nur konstante Zeit. Die Laufzeit bestimmt sich damit im wesentlichen durch den rekursiven Aufruf in Zeile 10. Hat nun Knoten i Höhe h, so erfolgt der rekursive Aufruf in Zeile 10 mit einem Knoten der Höhe $h-1$. Wir analysieren nun die Laufzeit $T(h)$ zunächst in Abhängkeit der Höhe h. Es gilt

$$T(h) = \begin{cases} c & \text{falls } h = 0. \\ \\ T(h-1) + c & \text{falls } h > 0 \end{cases}$$

Dabei ist die Konstante c so gewählt, dass sie sowohl den Aufwand der Zeilen 1-9 als auch die Gesamtlaufzeit bei Knoten i der Höhe 0 abschätzt. Wie in der Vorlesung bei MAX-HEAPIFY

gesehen, führt diese Rekursion auf $T(h) = \mathcal{O}(h)$. Wir wissen aber auch, dass ein (Teil-)Array mit n Elementen höchstens Höhe $\lfloor \log(n) \rfloor$ besitzt. Damit können wir die Laufzeit $T(n)$ von MIN-HEAPIFY in Abhängigkeit von der Größe des Heaps mit Wurzel i abschätzen durch $T(n) = \mathcal{O}(\log(n))$.