

AD Übungsblatt 5

Simon Thelen

4. November 2018

Aufgabe 2

1

z.z.: Ein Heap mit n Elementen hat Höhe $\lfloor \log n \rfloor$

Beweis. Beweis durch Induktion

IA $\lfloor \log 1 \rfloor = 0$

IV Ein Heap mit n Elementen hat Höhe $\lfloor \log n \rfloor$

IS $n \mapsto n + 1$

Fall 1 $n + 1 = 2^i, i \in \mathbb{N}$

$$\lfloor \log n + 1 \rfloor = \lfloor \log 2^i \rfloor = \log 2^i = i = \lfloor \log n \rfloor + 1$$

Stimmt, da bei einem vollen Heap gilt: $n = 2^i - 1$, was bedeutet, dass ein Baum mit $n + 1$ ein höher ist.

Fall 2 $n + 1 = 2^i - k, k \in \mathbb{N}, 1 \leq k < 2^{i-1}$

$$n + 1 \leq 2^i - 1 \Rightarrow \lfloor \log(n + 1) \rfloor \leq i - 1$$

$$n + 1 > 2^i - 2^{i-1} = 2^{i-1} \Rightarrow \lfloor \log(n + 1) \rfloor \geq i - 1$$

$$n \leq 2^i - 2 \Rightarrow \lfloor \log n \rfloor \leq i - 1$$

$$n > 2^i - 2^{i-1} - 1 = 2^{i-1} - 1 \Rightarrow n \geq 2^{i-1} \Rightarrow \lfloor \log n \rfloor \geq i - 1$$

$$\Rightarrow \lfloor \log(n + 1) \rfloor = \lfloor \log n \rfloor = i - 1$$

Stimmt, da der Heap bei n noch nicht voll war. Das heißt, die Höhe bleibt bei $n + 1$ gleich.

□

2

z.z.: Ein Heap mit n Elementen hat höchstens $\lceil \frac{n}{2^{h+1}} \rceil$ viele Knoten der Höhe h .

Beweis. Gegeben sei ein voller Heap mit n Knoten und einer Höhe von h_0 . Dann gibt es immer 1 Element der Höhe h_0 , 2 der Höhe $h_0 - 1$, 4 mit $h_0 - 2$, ...

Das bedeutet, $N(n, h)$, die Anzahl an Knoten mit Höhe h dieses Heaps, beträgt

$$\begin{aligned} N(n, h) &= 2^{h_0-h} = 2^{\lfloor \log n \rfloor - h} = \frac{2^{\lfloor \log n \rfloor}}{2^h} \\ &= \frac{\lfloor n \rfloor}{2^h} \leq \left\lceil \frac{n}{2^h} \right\rceil \leq \left\lceil \frac{n}{2^{h-1}} \right\rceil. \end{aligned}$$

Für einen nicht vollen Heap mit n Knoten der Höhe h_0 gilt:

$$N(n, h) = \begin{cases} N(n-1, 0) & h = 0 \wedge n \text{ ist gerade} \\ N(n-1, 0) + 1 & h = 0 \wedge n \text{ ist ungerade} \\ N(n-1, 0) + 1 & h > 0 \wedge n \text{ ist gerade} \\ N(n-1, 0) & h > 0 \wedge n \text{ ist ungerade} \end{cases}$$

Ein voller Heap kann also nie weniger Knoten einer bestimmten Höhe haben, als ein nicht voller Heap der gleichen Höhe. Da die obige Formel $\lceil \frac{n}{2^{h-1}} \rceil$ durch das Aufrunden eine Obergrenze darstellt, ist sie allgemein für jeden Heap gültig. \square

3

$$\text{z.z.: } \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

Beweis.

$$\begin{aligned} \sum_{k=0}^{\infty} x^k &= \frac{1}{1-x} \\ \left(\sum_{k=0}^{\infty} x^k \right)' &= \left(\frac{1}{1-x} \right)' \\ \sum_{k=0}^{\infty} kx^{k-1} &= \frac{1}{(1-x)^2} \quad | \cdot x \\ \sum_{k=0}^{\infty} kx^k &= \frac{x}{(1-x)^2} \end{aligned}$$

\square

Vertauschbarkeit von Heapify bei BuildHeap

Die Heapify-Aufrufe können nicht einfach vertauscht werden. Ein Gegenbeispiel ist das Array $[1, 2, 3, 4]$. Wird hier Heapify wie im BuildHeap zuerst auf das zweite und dann auf das erste Element aufgerufen, ergibt sich der Heap $[4, 2, 3, 1]$. Dreht man die Reihenfolge jedoch um, lautet das Ergebnis $[3, 4, 1, 2]$, was kein korrekter Heap ist.

Aufgabe 3

Variante 1

Beweis. Beweis durch Induktion

IA für $n = 2$: $M \cdot N = O$ per Definition (normale Matrixmultiplikation für $\mathbb{R}^{2 \times 2}$)

IV Berechnung der Multiplikation ist korrekt für n

IS $n \mapsto 2n$

$$\begin{aligned} & \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \cdot \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{pmatrix} = \\ & \begin{pmatrix} \begin{pmatrix} M_{11,11} & M_{11,12} \\ M_{11,21} & M_{11,22} \end{pmatrix} & \cdots \\ \vdots & \ddots \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} N_{11,11} & N_{11,12} \\ N_{11,21} & N_{11,22} \end{pmatrix} & \cdots \\ \vdots & \ddots \end{pmatrix} = \\ & \begin{pmatrix} \begin{pmatrix} M_{11,11} & M_{11,12} \\ M_{11,21} & M_{11,22} \end{pmatrix} \cdot \begin{pmatrix} N_{11,11} & N_{11,12} \\ N_{11,21} & N_{11,22} \end{pmatrix} + \begin{pmatrix} M_{12,11} & M_{12,12} \\ M_{12,21} & M_{12,22} \end{pmatrix} \cdot \begin{pmatrix} N_{21,11} & N_{21,12} \\ N_{21,21} & N_{21,22} \end{pmatrix} & \cdots \\ \vdots & \ddots \end{pmatrix} = \\ & \begin{pmatrix} M_{11,11} \cdot N_{11,11} + M_{11,12} \cdot N_{11,21} + M_{12,11} \cdot N_{21,11} + M_{12,12} \cdot N_{21,21} & M_{11,11} \cdot N_{11,12} + M_{11,12} \cdot N_{11,22} + M_{12,11} \cdot N_{21,12} + M_{12,12} \cdot N_{21,22} \\ M_{11,21} \cdot N_{11,11} + M_{11,22} \cdot N_{11,21} + M_{12,21} \cdot N_{21,11} + M_{12,22} \cdot N_{21,21} & M_{11,21} \cdot N_{11,12} + M_{11,22} \cdot N_{11,22} + M_{12,21} \cdot N_{21,12} + M_{12,22} \cdot N_{21,22} \\ \vdots & \ddots \end{pmatrix} = \\ & = \begin{pmatrix} O_{11,11} & O_{11,12} & \cdots \\ O_{11,21} & O_{11,22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \end{aligned}$$

Das gleiche gilt analog auch für O_{12} , O_{21} und O_{22} .

□

Laufzeitverhalten

$$T(n) = \begin{cases} 1 & n = 2 \\ 8T(n/2) + \Theta(n^2) & n > 2 \end{cases}$$

Da $f(n) = \Theta(n^2) = O(n^{\log_2(8)})$, gilt nach dem 1. Fall der Mastermethode $T(n) = \Theta(n^{\log_2(8)}) = \Theta(n^3)$, was keine Verbesserung zur „Standardmethode“ darstellt.

Variante 2

Beweis. Es gilt

$$\begin{aligned} O_{11} &= (M_{11} + M_{22}) \cdot (N_{11} + N_{22}) + M_{22} \cdot (N_{21} - N_{11}) \\ &\quad - (M_{11} + M_{12}) \cdot N_{22} + (M_{12} - M_{22}) \cdot (N_{21} + N_{22}) \\ &= M_{11} \cdot N_{11} + M_{11} \cdot N_{22} + M_{22} \cdot N_{11} + M_{22} \cdot N_{22} + M_{22} \cdot N_{21} - M_{22} \cdot N_{11} \\ &\quad - M_{11} \cdot N_{22} - M_{12} \cdot N_{22} + M_{12} \cdot N_{21} + M_{12} \cdot N_{22} - M_{22} \cdot N_{21} - M_{22} \cdot N_{22} \\ &= M_{11} \cdot N_{11} + M_{12} \cdot N_{21}, \end{aligned}$$

was dem gleichen Wert für O_{11} wie bei Variante 1 entspricht. Durch Nachrechnen lässt sich zeigen, dass analog auch bei O_{12} , O_{21} und O_{22} die gleichen Ergebnisse wie bei Variante 1 herauskommen. Unter der Annahme, dass Variante 1 korrekt ist, muss damit auch Variante 2 korrekt sein. □

Laufzeitverhalten

$$T(n) = \begin{cases} 1 & n = 2 \\ 7T(n/2) + \Theta(n^2) & n > 2 \end{cases}$$

Da $f(n) = \Theta(n^2) = O(n^{\log_2 7})$, gilt nach dem 1. Fall der Mastermethode $T(n) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.81})$, was asymptotisch gesehen eine Verbesserung der Laufzeit ist.