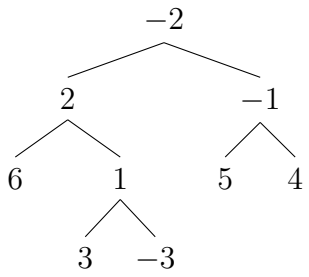
DAP2 – Präsenzübung 9

Besprechung: 21.06.2017 — 23.06.2017

Präsenzaufgabe 9.1: Teile und Herrsche im Baum

In dieser Aufgabe betrachten wir Binärbäume, deren Schlüsseleinträge ganze Zahlen sind. Für zwei Knoten a und b eines Binärbaumes T bezeichne $PairSum\,(a,b)$ die Summe aller Schlüsseleinträge der Knoten in T, die auf dem kürzesten Weg von a nach b liegen. Ist z.B. in dem folgenden Binärbaum a der Knoten mit Schlüssel 5 und b der Knoten mit Schlüssel 6, so ist $PairSum\,(a,b) = 10$, da auf dem kürzesten Weg zwischen diesen beiden Knoten die Schlüssel $5, -1, -2, 2, 6$ gelesen werden.



Sei nun $MaxPairSum\,(T)$ definiert als

$$MaxPairSum\,(T) = \max\{PairSum\,(a,b) \mid a,b \text{ Knoten von } T\}\,.$$

Für den oben angegebenen Binärbaum T wäre somit $MaxPairSum\,(T) = 6+2+1+3 = 12$.

Wir suchen nun einen *Teile-und-Herrsche*-Algorithmus, der bei Eingabe eines Binärbaums T den Wert $MaxPairSum\,(T)$ berechnet.

a) Entwerfen Sie einen solchen Teile-und-Herrsche-Algorithmus und beschreiben Sie ihn mit eigenen Worten. Geben Sie eine Implementierung Ihres Algorithmus in Pseudocode an.

b) Analysieren Sie die Laufzeit Ihres Algorithmus. Stellen Sie hierzu eine Rekursionsgleichung für die Laufzeit Ihres Algorithmus (in Abhängigkeit von der Höhe des Baumes) auf und lösen Sie diese.

c) Zeigen Sie die Korrektheit Ihres Algorithmus.

Lösung:

a) Wir bezeichnen im folgenden mit $T(w)$ den Teilbaum von T mit Wurzel w. Um $MaxPairSum\,(T(w))$ des Baumes $T(w)$ zu berechnen, sind folgende Fälle für die maximale Summe zu betrachten:

i) sie wird im linken Teilbaum erzielt;

ii) sie wird im rechten Teilbaum erzielt;

iii) sie wird durch einen Pfad erzielt, der die Wurzel enthält.

Im Fall iii) kann der Pfad in der Wurzel beginnen und in einem der beiden Teilbäume enden, im linken Teilbaum beginnen und im rechten enden, oder lediglich aus der Wurzel bestehen. Deswegen werden wir für einen Baum das Paar $(\ell(w), m(w))$ berechnen. Der Wert $\ell(w)$ bezeichnet die maximale Summe eines Pfades, der in der Wurzel w beginnt, und $m(w)$ bezeichnet die maximale erzielbare Summe im Baum $T(w)$.

Die Rekursion wird abgebrochen, wenn der Baum nur einen Knoten w enthält. In diesem Fall wird das Paar $(\ell(w), m(w)) = (\text{wert}(w), \text{wert}(w))$ zurückgegeben. Da $PairSum(a, b)$ über dem kurzesten Weg zwischen zwei Knoten definiert wurde, gilt im Fall $a = b$ genau $PairSum(a, a) = \text{wert}(a)$.

Wenn der Baum mit Wurzel w nicht nur einen Knoten enthält, wird $\ell(w)$ berechnet durch

$$\ell(w) = \max\{\text{wert}(w), \text{wert}(w) + \ell(\text{left}(w)), \text{wert}(w) + \ell(\text{right}(w))\} \ .$$

Dabei fallen die Möglichkeiten aus, die die Ergebnisse von linken bzw. rechten Teilbaum verwenden, wenn es keinen linken bzw. rechten Teilbaum vorhanden ist. Dieser Fall wird durch den Wert $-\infty$ bezeichnet.

Der Wert von $m(w)$ wird wie folgt berechnet:

$$m(w) = \max\{\ell(w), m(\text{left}(w)), m(\text{right}(w)), \text{wert}(w) + \ell(\text{left}(w)) + \ell(\text{right}(w))\}$$

Diese vier Möglichkeiten beschreiben die Fälle (in gegebener Reihenfolge) dass die maximale Summe a) in der Wurzel anfängt; b) im linken Teilbaum erzielt wird, c) in rechten Teilbaum erzielt wird; d) die Teilpfads in beiden linken und rechten Teilbaum enthält.

Das ergibt den folgenden Pseudocode:

```
MPS(Baum T, Wurzel w):
1  if left[w] ≠ nil then
2      (ℓ_A, m_A) ← MPS(T, left[w]))
3  else
4      (ℓ_A, m_A) ← (−∞, −∞)                          /* unmögliche Ergebnisse */
5  if right[w] ≠ nil then
6      (ℓ_B, m_B) ← MPS(T, right[w])
7  else
8      (ℓ_B, m_B) ← (−∞, −∞)
          /* Rekursive Ergebnisse wurden im linken/rechten Unterbaum berechnet */
9  ℓ(w) ← wert(w)
10 if ℓ_A > 0 then
11     ℓ(w) ← wert(w) + ℓ_A
12 if ℓ_B > 0 then
13     ℓ(w) ← max{ℓ(w), wert(w) + ℓ_B}
         /* Maximum für ℓ(w) wurde berechnet. Leere Unterbäume wurden ignoriert. */
14 m(w) ← max{ℓ(w), m_A, m_B, wert(w) + ℓ_A + ℓ_B}
15 return (ℓ(w), m(w))
```

Für den Startaufruf geben wir eine Hilfsfunktion an:

```
MPSStart(Baum T):
1  (ℓ(w), m(w)) ← MPS(T, w)
2  return m(w)
```

b) Die Laufzeit unseres Algorithmus kann mit folgender Rekursionsgleichung beschrieben werden:

$$T(n) = \begin{cases} c & \text{falls } n = 0 \vee n = 1 \\ c + T(n') + T(n - n' - 1) & \text{falls } n > 1 \end{cases}$$

Im letzten Fall der Rekursionsgleichung nutzen wir aus, dass die Anzahl der Knoten im betrachteten Teilbaum gleich 1 plus der Summe der Knotenzahl in den Teilbäumen ist. Wir zeigen mit Induktion, dass $T(n) \leq 2cn + c$ für $n \geq 0$:

I.A. $n = 0$: $T(0) = c \leq c$ und $n = 1$: $T(1) = c \leq 2c + c$

I.V. Für beliebiges aber festes n gelte die Aussage für alle n' mit $0 \leq n' < n$

I.S. Wir beweisen es jetzt für den Baum mit n Knoten.

$$\begin{aligned} T(n) &= c + T(n') + T(n - n' - 1) \\ &\leq c + 2cn' + c + 2c(n - n' - 1) + c && \text{nach I.V.} \\ &= 2cn + c \end{aligned}$$

Damit ist unsere Laufzeitschranke nach dem Konzept der Induktion bewiesen.

c) Wir beweisen die folgende Behauptung per Induktion über die Baumhöhe:

Beh: Der Algorithmus $MPS(T, w)$ berechnet die $MaxPairSum(T)$ für den Baum T der Höhe h und mit der Wurzel w.

IA. Wenn $h = 0$ ist, hat der Baum nur einen Knoten und linken bzw. rechten Teilbaum sind leer. Dann werden ℓ_a, ℓ_B, m_A und m_B alle zum Wert $-\infty$ als nicht existierende Teillösungen gesetzt (Zeilen 1-8). In der Zeile 9 wird $\ell(w)$ zum wert(w) gesetzt. Die Zeilen 10-13 werden übersprungen. In der Zeile 14 wird $m(w) = \ell(w) =$ wert(w) gesetzt. Das ist korrekt und wird in der Zeile 15 zurückgegeben.

IV. Es gelte die Behauptung für alle Bäume der Höhe $0 \leq h' < h$, wobei h beliebig aber fest ausgewählt wurde.

IS. Sei T ein Binärbaum der Höhe h. Der linke bzw. rechte Teilbaum von T haben je eine Höhe, die echt kleiner ist als h. Deswegen werden nach der I.V. die maximalen Werte von $\ell(\text{left}(w))$, $m(\text{left}(w))$, $\ell(\text{right}(w))$ und $m(\text{right}(w))$ korrekt berechnet, und in ℓ_A, m_A, ℓ_B und m_B gespeichert.

Der maximale Wert von $\ell(w)$ kann entweder nur die Wurzel w enthalten, oder einen Pfad von w zu einem Knoten des linken (über left(w)) oder des rechten (über right(w)) Teilbaumes, sofern existent. Da die Werte ℓ_A und ℓ_B nach der I.V. maximal sind und alle mögliche Fälle betrachtet, muss auch $\ell(w)$ die maximale Summe eines Pfades in T(w) sein, die in der Wurzel w anfängt.

Der maximale Wert von $m(w)$ kann durch die folgenden Fälle für einen entsprechenden Pfad erzielt werden:

- Der Pfad fängt in w an: Für $\ell(w)$ ist bewiesen, dass es maximal ist.
- Er ist vollständig im linken bzw. rechten Teilbaum: Die Werte m_A und m_B sind nach I.V. maximal für die jeweiligen Teilbäume
- Er erstreckt sich über die Wurzel w vom linken in den rechten Teilbaum: Im letzten Fall setzt sich der Wert diesen Pfades zusammen aus den Werten des jeweils optimalen Teilpfades von left(w) und right(w), der in der Wurzel des jeweiligen Baumes beginnt, und dem Wert der Wurzel w. Nach I.V. werden die ersten beiden Werte korrekt berechnet und in den Variablen ℓ_A und ℓ_B gespeichert.

Somit gibt der Algoritmus die maximalen Werte $\ell(w)$ und $m(w)$ für den Baum T der Höhe h korrekt.

Der Algorithmus berechnet also $MaxPairSum(T)$ korrekt für alle binäre Bäume T, wenn er auf der Wurzel w zuerst aufgerufen wird.

Präsenzaufgabe 9.2: (AVL-Bäume)

Beschreiben Sie eine Familie von AVL-Bäumen, in der eine einzige Löschoperation eine Folge von $\Theta(\log n)$ Rotationen nach sich ziehen kann.

Lösung:

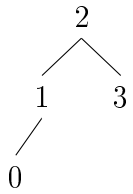
Wir definieren induktiv die Fibonacci-Bäume $T_n,\ n \in \mathbb{N}$ der Ordnung n. Es kommt hier nur auf deren Gestalt an.

- Der Fibonacci-Baum T_0 der Ordnung 0 ist der leere Baum.
- Der Fibonacci-Baum T_1 der Ordnung 1 ist ein Binärbaum mit genau 1 Knoten.

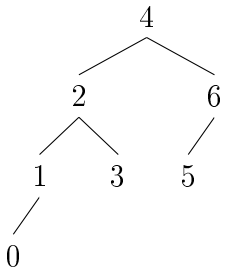
- Der Fibonacci-Baum T_n der Ordnung n ist ein Binärbaum, dessen Wurzel T_{n-1} als linken Unterbaum und T_{n-2} als rechten Unterbaum hat.

Wir geben einige Beispiele von Fibonacci-Bäumen an. Als Schlüsseleinträge verwenden wir die Nummerierungen, die sich bei einem Inorder-Durchlauf ergeben.

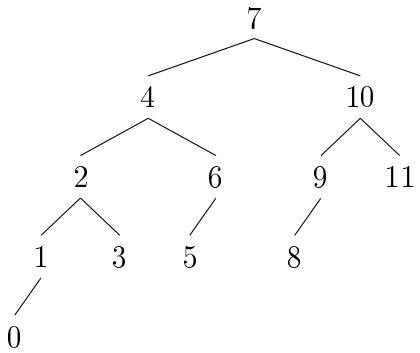
- T_3:



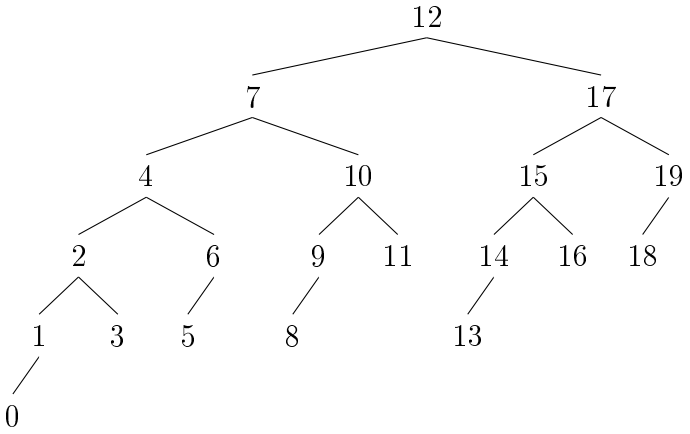
- T_4:



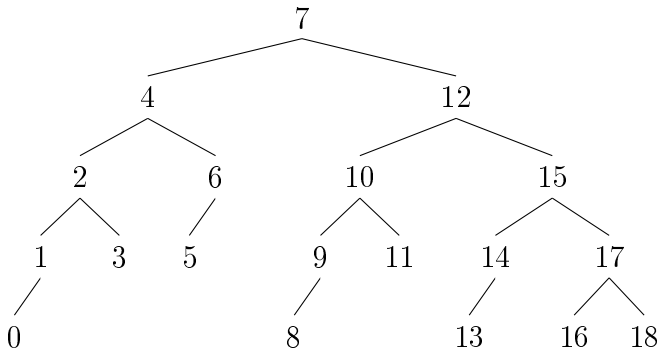
- T_5:



- T_6:



Wir betrachten den Effekt, wenn wir den größten Schlüssel löschen.



Man überlegt sich leicht, dass das Löschen des größten Schlüssels im Fibonaccibaum der Ordnung n eine Folge von Rechtsrotationen nach sich zieht, die alle Knoten auf dem Pfad zur Wurzel betreffen. Da dieser Pfad eine Länge von $O(\log n)$ besitzt, bedeutet dies somit $\Theta(\log n)$ viele Rotationen.