Abschnitt 18: Effizientes Suchen in Mengen

Überblick

Motivation

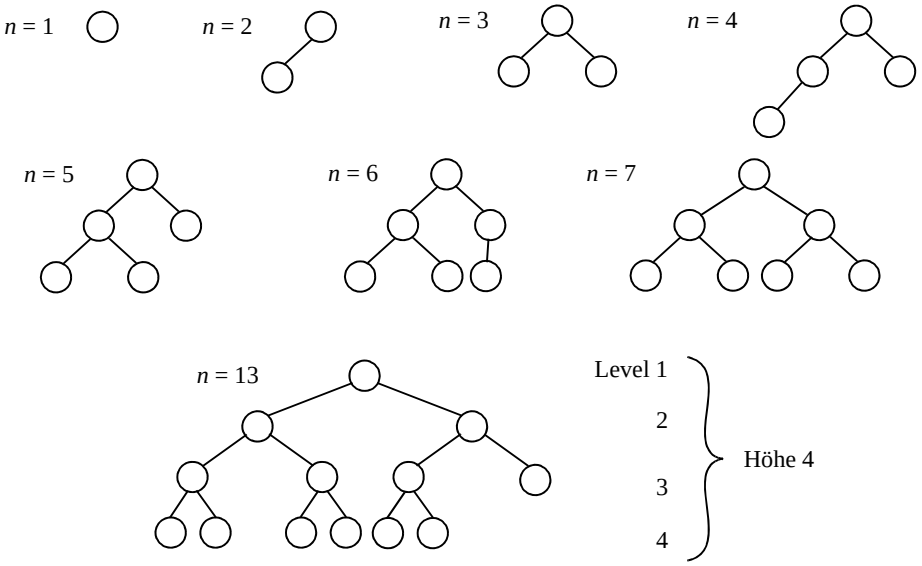
- Laufzeitanalyse der Suche (und damit auch Einfügen und Löschen) in einem einfachen binären Suchbaum ergab:
- Suche ist auf einen einzigen Pfad beginnend bei der Wurzel beschränkt.
- Maximaler Aufwand ist also $O(h)$, wobei h die Höhe des Baumes ist.
- Die Höhe h eines binären Baumes mit n Konten ist durchschnittlich $O(\log n)$, aber maximal gilt $h = n$ (wenn der Baum zu einer linearen Liste entartet).
- Daher haben die Operationen Suchen, Einfügen und Löschen im worst case eine Komplexität von $O(n)$.
- Wir bräuchten also einen binären Suchbaum, der auch im schlechtesten Fall eine logarithmische Höhe hat (und nicht zu einer linearen Liste entarten darf).

Vollständig ausgeglichene binäre Suchbäume

- Diese Bäume sind die *vollständig ausgeglichenen binären Suchbäume*.
- Vollständig ausgeglichene binäre Suchbäume besitzen unter allen binären Suchbäumen die minimale Höhe.
- Vollständig ausgeglichene binäre Suchbäume sind binäre Suchbäume, bei denen alle Levels bis auf das unterste vollständig besetzt sind.
- Der *Level* eines Knotens K ist die Anzahl der Knoten auf dem Pfad von K zur Wurzel des Baumes.
- Der *Level* l eines Baumes umfasst alle Knoten mit Level l.
- Die Höhe h eines Baumes ist der maximale Level von allen seinen Knoten.

Vollständig ausgeglichene binäre Suchbäume: Beispiele

Beispiele für vollständig ausgeglichene binäre Suchbäume



Vollständig ausgeglichene binäre Suchbäume: Höhe

- Die Höhe ergibt sich wie folgt:
- Die maximale Anzahl von Knoten in einem vollständig ausgeglichenen binären Baum der Höhe h ist:

$$n = \sum_{i=0}^{h-1} 2^i = 2^h - 1.$$

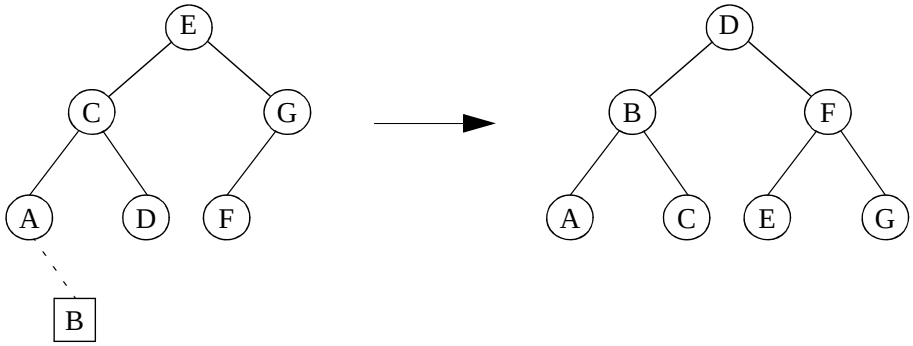
Begründung:

max. 1 Knoten auf Level 1 (Wurzel),
max. 2 Knoten auf Level 2,
max. 4 Knoten auf Level 3,
max. 8 Knoten auf Level 4,
$\vdots$
max. 2^{h-1} Knoten auf Level h.

- Umrechnung: $h = \lceil \log_2(n+1) \rceil$.
- Bemerkung: Da $\lceil \log_2(n+1) \rceil = \lfloor \log_2(n) \rfloor + 1$ gilt $h = O(\log n)$.

Vollständig ausgeglichene binäre Suchbäume: Problem

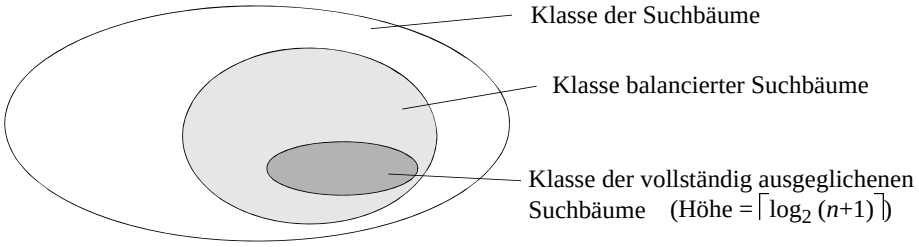
- Problem: Durch eine Einfügung muss im schlechtesten Fall der gesamte Baum reorganisiert werden.
- Beispiel: Schlüssel *B* einfügen



- Folge: Einfügezeit im schlechtesten Fall ist somit $O(n)$.

Balancierte Suchbäume

- Lösung: Auswahl einer Kompromißlösung mit den Eigenschaften:
  - Höhe des Baumes ist im schlechtesten Fall $O(\log n)$.
  - Reorganisation bleibt auf den Suchpfad zum einzufügenden bzw. zu enfernenden Knoten/Schlüssel beschränkt und ist damit im worst case in $O(\log n)$ Zeit ausführbar.
- Die Klasse von Suchbäumen, die diese Eigenschaften besitzen, heißt Klasse der *balancierten* Suchbäume.

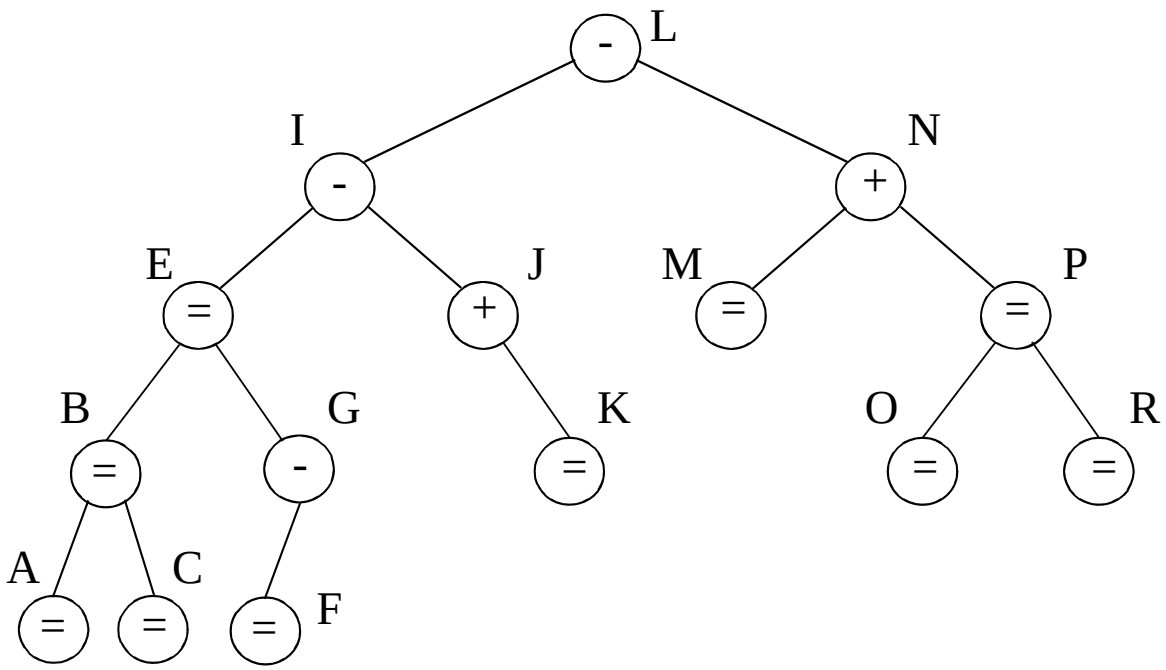


Überblick

Struktur von AVL-Bäumen

- Ein binärer Suchbaum heißt *AVL-Baum*, falls für die beiden Teilbäume lT und rT der Wurzel gilt:
  - $|h(rT) - h(lT)| \leq 1$ ($h(T)$ bezeichnet die Höhe des Baumes T),
  - rT und lT sind ihrerseits AVL-Bäume.
- Der Wert $h(rT) - h(lT)$ wird als *Balancefaktor* (BF) eines Knotens bezeichnet. Er kann in einem AVL-Baum nur die Werte -1, 0 oder 1 (dargestellt durch -, = und +) annehmen.
- AVL-Bäume sind ein Beispiel für eine Klasse balancierter Suchbäume.
- Mögliche Strukturverletzungen durch Einfügungen bzw. Entfernungen von Knoten/Schlüsseln erfordern Rebalancierungsoperationen.

AVL-Bäumen: Beispiel



Minimale Höhe von AVL-Bäumen

Lemma

Die minimale Höhe $h_{min}(n)$ eines AVL-Baumes mit n Schlüssel ist $\lceil \log_2(n+1) \rceil$.

Beweis.

Die Behauptung folgt aus der Tatsache, dass ein AVL-Baum minimaler Höhe einem vollständig ausgeglichenen binären Suchbaum entspricht. □

Maximale Höhe von AVL-Bäumen (Lemma)

Lemma

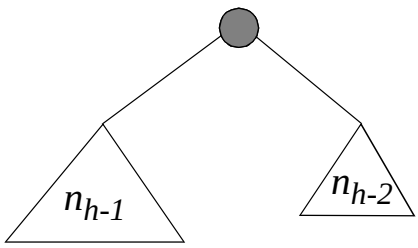
Die maximale Höhe $h_{max}(n)$ eines AVL-Baumes mit n Schlüssel ist $O(\log n)$.

Der Beweis dieses Lemmas ist etwas schwieriger zu führen und erfordert etwas Vorbereitung.

Maximale Höhe von AVL-Bäumen (Überlegung)

Die maximale Höhe wird realisiert durch einen *minimalen* AVL-Baum. Minimale AVL-Bäume sind AVL-Bäume, die für eine gegebene Höhe *h*, die minimale Anzahl von Schlüsseln abspeichern.

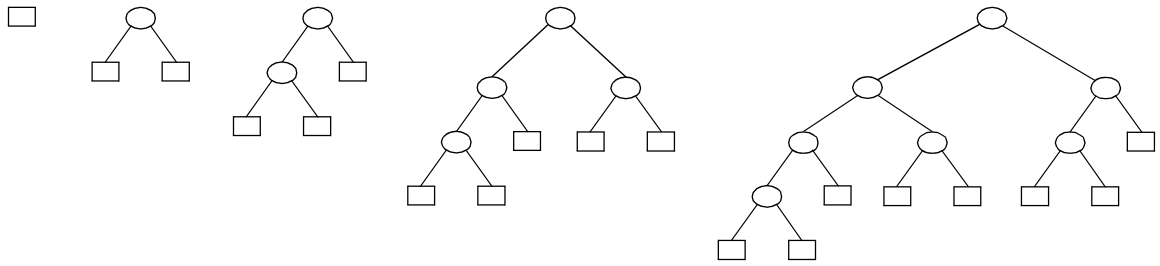
Minimale AVL-Bäume haben (bis auf Symmetrie) folgende Gestalt:



Sei n_h die minimale Anzahl von Schlüsseln eines AVL-Baumes der Höhe *h*.
Dann gilt: $n_h = n_{h-1} + n_{h-2} + 1$ für $h \geq 2$ $n_0 = 0, n_1 = 1$.

Maximale Höhe von AVL-Bäumen (Überlegung)

Die minimalen AVL-Bäume der Höhen $h = 0, 1, 2, 3, 4$ haben bis auf Symmetrie folgende Gestalt:



Die Rekursionsgleichung für n_h erinnert an die Definition der *Fibonacci*-Zahlen:

$$fib(n) = \begin{cases} n & \text{für } n \leq 1 \\ fib(n-1) + fib(n-2) & \text{für } n > 1 \end{cases}$$

Maximale Höhe von AVL-Bäumen (Hilfssatz 1)

h	0	1	2	3	4	5	6
n	0	1	2	4	7	12	20
$fib(h)$	0	1	1	2	3	5	8

Hilfssatz

Es gilt: $n_h = fib(h+2) - 1$.

Beweis.

Induktion über h

Induktionsanfang: $n_0 = 0 = 1 - 1 = fib(0+2) - 1$ $\surd$

Induktionsschluss: Induktionsvorraussetzung: $n_h = fib(h+2) - 1$

$$\begin{aligned} n_{h+1} &= n_h + n_{h-1} + 1 \\ &\overset{I.V.}{=} fib(h+2) - 1 + fib(h+1) - 1 + 1 \\ &= fib(h+2) + fib(h+1) - 1 \\ &= fib(h+3) - 1 \end{aligned}$$

□

Maximale Höhe von AVL-Bäumen (Hilfssatz 2)

Hilfssatz

Es gilt:

$$fib(n) = \frac{1}{\sqrt{5}} \cdot (\varphi_1{}^n - \varphi_2{}^n).$$

wobei

$$\varphi_1 = \frac{1+\sqrt{5}}{2} \quad \textit{und} \quad \varphi_2 = \frac{1-\sqrt{5}}{2} \approx -0,618$$

Beweis.

Induktion über n (wird hier übergangen). □

Maximale Höhe von AVL-Bäumen (Beweis)

Nun sind wir in der Lage, den Beweis des Lemmas zu führen:

Beweis

Für jede beliebige Schlüsselzahl $n \in \mathbb{N}$ gibt es ein eindeutiges $h_{max}(n)$ mit:

$$n_{h_{max}(n)} \leq n \leq n_{h_{max}(n)+1}.$$

Mit unserem ersten Hilfssatz folgt: $n + 1 \geq fib(h_{max}(n) + 2)$.

Durch Einsetzen (zweiter Hilfssatz) erhalten wir:

$$n+1 \geq \frac{1}{\sqrt{5}} \cdot (\varphi_1{}^{h_{max}(n)+2} - \underbrace{\varphi_2{}^{h_{max}(n)+2}}_{<0,62<\sqrt{5}/2}) \geq \frac{1}{\sqrt{5}} \cdot \varphi_1{}^{h_{max}(n)+2} - \frac{1}{2}$$

Und damit:

$$\frac{1}{\sqrt{5}} \cdot \varphi_1{}^{h_{max}(n)+2} \leq n + \frac{3}{2}.$$

Maximale Höhe von AVL-Bäumen (Beweis)

Durch Auflösen nach $h_{max}(n)$ ergibt sich:

$$\log_{\varphi_1}(\frac{1}{\sqrt{5}}) + h_{max}(n) + 2 \leq \log_{\varphi_1}(n + \frac{3}{2}).$$

Und für $h_{max}(n)$:

$$\begin{aligned} h_{max}(n) &\leq \log_{\varphi_1}(n + \frac{3}{2}) - (\log_{\varphi_1}(\frac{1}{\sqrt{5}}) + 2) \\ &\leq \log_{\varphi_1}(n) + c = \log_{\varphi_1}(2) \cdot \log_2(n) + c \\ &= \frac{\ln 2}{\ln_{\varphi_1}} \cdot \log_2(n) + c \text{ , da } \log_b(a) = \frac{\log_c(a)}{\log_c(b)} \\ &\approx 1,44 \cdot \log_2(n) + c \end{aligned}$$

Die Höhe eines AVL-Baumes ist somit um maximal 44% größer als die des vollständig ausgeglichenen binären Suchbaums. Also gilt die Behauptung $h_{max}(n) = O(\log n)$ (Puh!)