10. AVL Bäume

Balancierte Bäume [Ottman/Widmayer, Kap. 5.2-5.2.1, Cormen et al, Kap. Problem 13-3]

Ziel

Suchen, Einfügen und Entfernen eines Schlüssels in Baum mit n Schlüsseln, welche in zufälliger Reihenfolge eingefügt wurden im Mittel in $\mathcal{O}(\log_2 n)$ Schritten.

Schlechtester Fall jedoch: $\Theta(n)$ (degenerierter Baum).

Ziel: Verhinderung der Degenerierung. Künstliches, bei jeder Update-Operation erfolgtes Balancieren eines Baumes

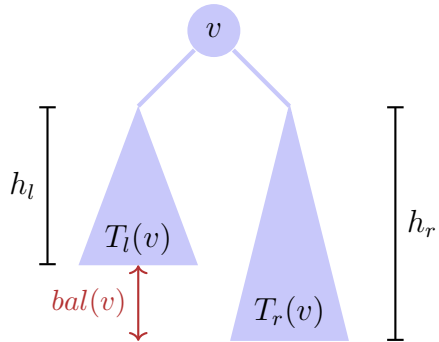
Balancierung: garantiere, dass ein Baum mit n Knoten stets eine Höhe von $\mathcal{O}(\log n)$ hat.

Adelson-Venskii und Landis (1962): AVL-Bäume

Balance eines Knotens

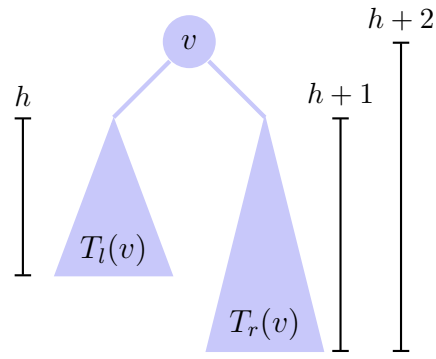
Die **Balance** eines Knotens v ist definiert als die Höhendifferenz seiner beiden Teilbäume $T_l(v)$ und $T_r(v)$

$$\text{bal}(v) := h(T_r(v)) - h(T_l(v))$$

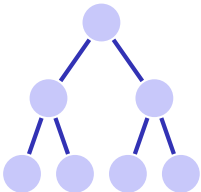


AVL Bedingung

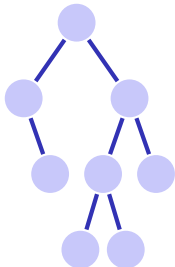
AVL Bedingung: für jeden Knoten v eines Baumes gilt $\mathrm{bal}(v) \in \{-1, 0, 1\}$



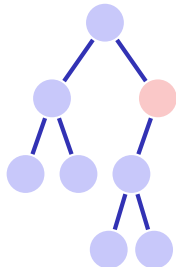
(Gegen-)Beispiele



AVL Baum der Höhe 2



AVL Baum der Höhe 3

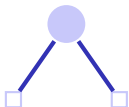


Kein AVL Baum

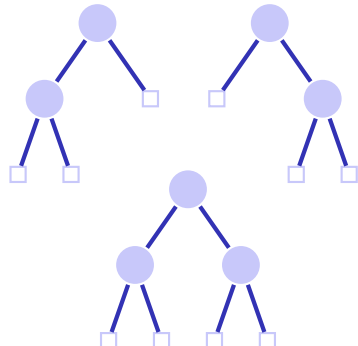
Anzahl Blätter

- 1. Beobachtung: Ein Suchbaum mit n Schlüsseln hat genau $n + 1$ Blätter. Einfaches Induktionsargument.
  - Der Suchbaum mit $n = 0$ Schlüsseln hat $m = 1$ Blätter
  - Wird ein Schlüssel (Knoten) hinzugefügt ($n \to n + 1$), so ersetzt er ein Blatt und fügt zwei Blätter hinzu ($m \to m - 1 + 2 = m + 1$).
- 2. Beobachtung: untere Grenze für Anzahl Blätter eines Suchbaums zu gegebener Höhe erlaubt Abschätzung der maximalen Höhe eines Suchbaums zu gegebener Anzahl Schlüssel.

Untere Grenze Blätter



AVL Baum der Höhe 1 hat
$N(1) := 2$ Blätter



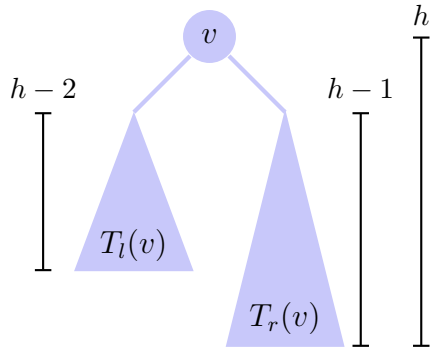
AVL Baum der Höhe 2 hat
mindestens $N(2) := 3$ Blätter

Untere Grenze Blätter für $h > 2$

- Höhe eines Teilbaums $\geq h - 1$.
- Höhe des anderen Teilbaums $\geq h - 2$.

Minimale Anzahl Blätter $N(h)$ ist

$$N(h) = N(h-1) + N(h-2)$$



Insgesamt gilt $N(h) = F_{h+2}$ mit **Fibonacci-Zahlen** $F_0 := 0$, $F_1 := 1$, $F_n := F_{n-1} + F_{n-2}$ für $n > 1$.

Fibonacci Zahlen, geschlossene Form

Es gilt

$$F_i = \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i)$$

mit den Wurzeln $\phi, \hat{\phi}$ der Gleichung vom goldenen Schnitt $x^2 - x - 1 = 0$:

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$$
$$\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0.618$$

Fibonacci Zahlen, Induktiver Beweis

$F_i \stackrel{!}{=} \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i) \qquad [*]$ $\qquad \left(\phi = \frac{1+\sqrt{5}}{2}, \hat{\phi} = \frac{1-\sqrt{5}}{2}\right).$

1. Klar für $i = 0, i = 1$.
2. Sei $i > 2$ und Behauptung $[*]$ wahr für alle F_j, $j < i$.

$$F_i \stackrel{def}{=} F_{i-1} + F_{i-2} \stackrel{[*]}{=} \frac{1}{\sqrt{5}}(\phi^{i-1} - \hat{\phi}^{i-1}) + \frac{1}{\sqrt{5}}(\phi^{i-2} - \hat{\phi}^{i-2})$$

$$= \frac{1}{\sqrt{5}}(\phi^{i-1} + \phi^{i-2}) - \frac{1}{\sqrt{5}}(\hat{\phi}^{i-1} + \hat{\phi}^{i-2}) = \frac{1}{\sqrt{5}}\phi^{i-2}(\phi + 1) - \frac{1}{\sqrt{5}}\hat{\phi}^{i-2}(\hat{\phi} + 1)$$

($\phi, \hat{\phi}$ erfüllen $x + 1 = x^2$)

$$= \frac{1}{\sqrt{5}}\phi^{i-2}(\phi^2) - \frac{1}{\sqrt{5}}\hat{\phi}^{i-2}(\hat{\phi}^2) = \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i).$$

Baumhöhe

Da $|\hat{\phi}| < 1$, gilt insgesamt

$$N(h) \in \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^h\right) \subseteq \Omega(1.618^h)$$

und somit

$$\begin{aligned} N(h) &\geq c \cdot 1.618^h \\ \Rightarrow \quad h &\leq 1.44 \log_2 n + c'. \end{aligned}$$

Ein AVL Baum ist asymptotisch nicht mehr als 44% höher als ein perfekt balancierter Baum.[5]

[5] Ein perfekt balancierter Baum hat Höhe $\lceil \log_2 n + 1 \rceil$

Einfügen

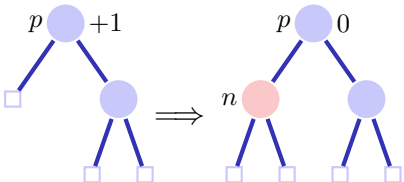
Balancieren

- Speichern der Balance für jeden Knoten
- Baum rebalancieren bei jeder Update-Operation

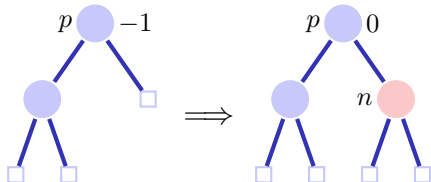
Neuer Knoten n wird eingefügt:

- Zuerst einfügen wie bei Suchbaum.
- Prüfe die Balance-Bedingung für alle Knoten aufsteigend von n zur Wurzel.

Balance am Einfügeort



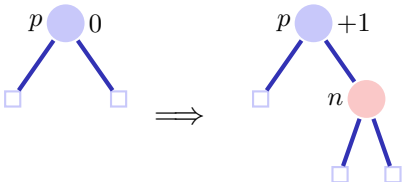
Fall 1: $\mathrm{bal}(p) = +1$



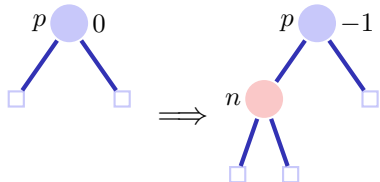
Fall 2: $\mathrm{bal}(p) = -1$

Fertig in beiden Fällen, denn der Teilbaum ist nicht gewachsen.

Balance am Einfügeort



Fall 3.1: $\mathrm{bal}(p) = 0$ rechts



Fall 3.2: $\mathrm{bal}(p) = 0$, links

In beiden Fällen noch nicht fertig. Aufruf von **`upin(p)`**.

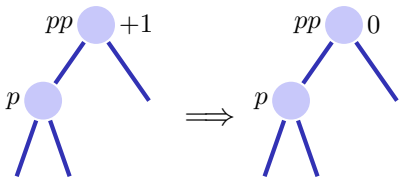
upin(p) - Invariante

Beim Aufruf von **`upin(p)`** gilt, dass

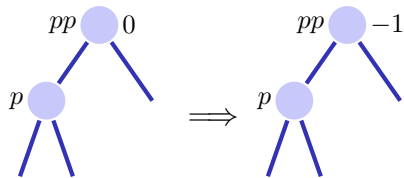
- der Teilbaum ab p gewachsen ist und
- $\mathrm{bal}(p) \in \{-1, +1\}$

upin(p)

Annahme: p ist linker Sohn von pp[6]



Fall 1: $\mathrm{bal}(pp) = +1$, fertig.



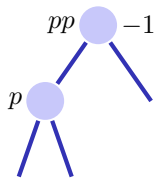
Fall 2: $\mathrm{bal}(pp) = 0$, **`upin(pp)`**

In beiden Fällen gilt nach der Operation die AVL-Bedingung für den Teilbaum ab pp

[6] Ist p rechter Sohn: symmetrische Fälle unter Vertauschung von $+1$ und -1

upin(p)

Annahme: p ist linker Sohn von pp



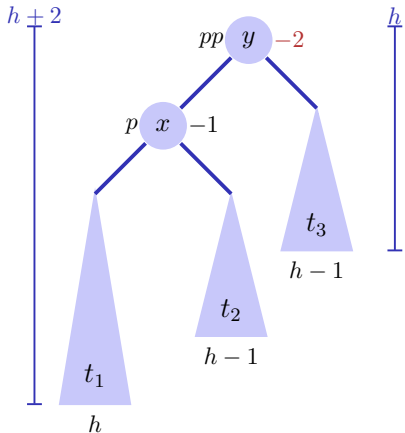
Fall 3: $\mathrm{bal}(pp) = -1$,

Dieser Fall ist problematisch: das Hinzufügen von n im Teilbaum ab pp hat die AVL-Bedingung verletzt. Rebalancieren!

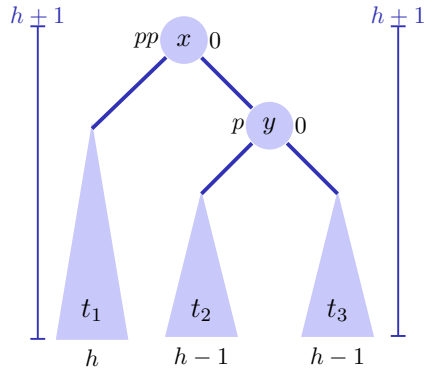
Zwei Fälle $\mathrm{bal}(p) = -1$, $\mathrm{bal}(p) = +1$

Rotationen

Fall 1.1 $\mathrm{bal}(p) = -1$. [7]



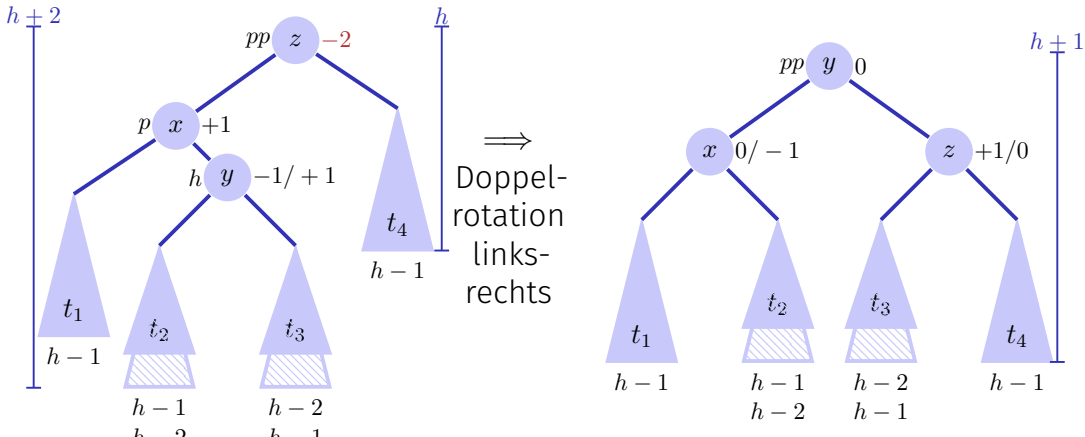
$\Longrightarrow$ Rotation nach rechts



[7] p rechter Sohn $\Rightarrow \mathrm{bal}(pp) = \mathrm{bal}(p) = +1$, Linksrotation

Rotationen

Fall 1.2 $\mathrm{bal}(p) = +1$. [8]



[8] p rechter Sohn $\Rightarrow \mathrm{bal}(pp) = +1$, $\mathrm{bal}(p) = -1$, Doppelrotation rechts links

Analyse

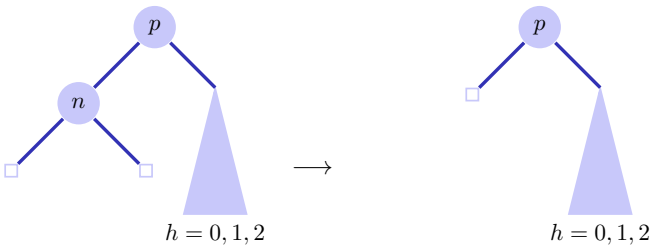
- Höhe des Baumes: $\mathcal{O}(\log n)$.
- Einfügen wie beim binären Suchbaum.
- Balancieren durch Rekursion vom Knoten zur Wurzel. Maximale Pfadlänge $\mathcal{O}(\log n)$.

Das Einfügen im AVL-Baum hat Laufzeitkosten von $\mathcal{O}(\log n)$.

Löschen

Fall 1: Knoten n hat zwei Blätter als Kinder Sei p Elternknoten von n. $\Rightarrow$ Anderer Teilbaum hat Höhe $h' = 0$, 1 oder 2

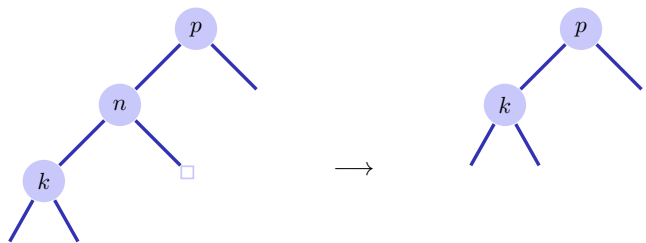
- $h' = 1$: $\mathrm{bal}(p)$ anpassen.
- $h' = 0$: $\mathrm{bal}(p)$ anpassen. Aufruf **`upout(p)`**.
- $h' = 2$: Rebalancieren des Teilbaumes. Aufruf **`upout(p)`**.



Löschen

Fall 2: Knoten n hat einen inneren Knoten k als Kind

- Ersetze n durch k. **`upout(k)`**



Löschen

Fall 3: Knoten n hat zwei inneren Knoten als Kinder

- Ersetze n durch symmetrischen Nachfolger. **`upout(k)`**
- Löschen des symmetrischen Nachfolgers wie in Fall 1 oder 2.

upout(p)

Sei pp der Elternknoten von p

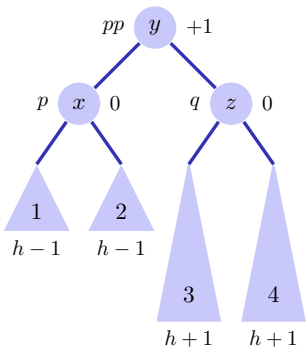
(a) p linkes Kind von pp

1. $\mathrm{bal}(pp) = -1 \;\Rightarrow\; \mathrm{bal}(pp) \leftarrow 0$. **upout(pp)**
2. $\mathrm{bal}(pp) = 0 \;\Rightarrow\; \mathrm{bal}(pp) \leftarrow +1$.
3. $\mathrm{bal}(pp) = +1 \Rightarrow$ nächste Folien.

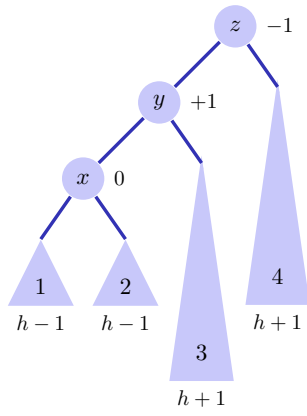
(b) p rechtes Kind von pp: Symmetrische Fälle unter Vertauschung von $+1$ und -1.

upout(p)

Fall (a).3: $\mathrm{bal}(pp) = +1$. Sei q Bruder von p
(a).3.1: $\mathrm{bal}(q) = 0$.[9]



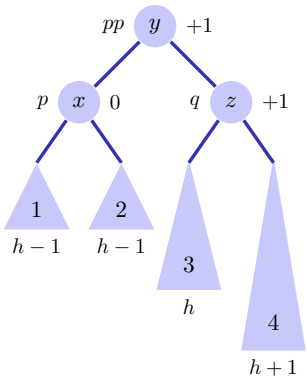
$\Longrightarrow$
Linksrotation
(y)



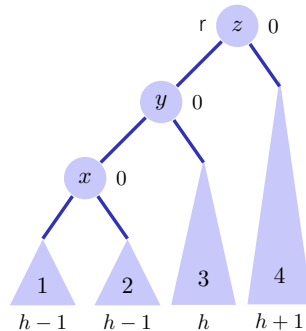
[9] (b).3.1: $\mathrm{bal}(pp) = -1$, $\mathrm{bal}(q) = -1$, Rechtsrotation.

upout(p)

Fall (a).3: $\mathrm{bal}(pp) = +1$. (a).3.2: $\mathrm{bal}(q) = +1$.[10]



$\Longrightarrow$
Linksrotation
(y)

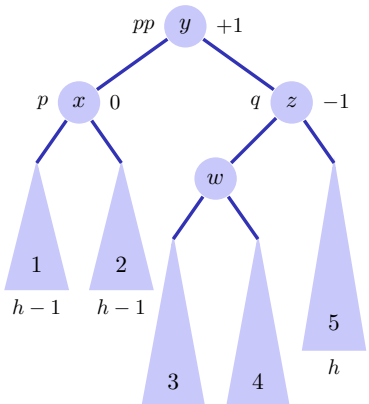


plus **upout(r)**.

[10] **(b).3.2:** $\mathrm{bal}(pp) = -1$, $\mathrm{bal}(q) = +1$, **Rechtsrotation+upout**

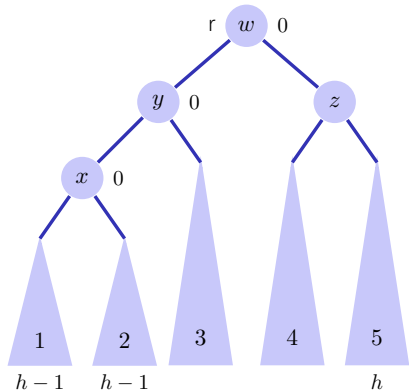
upout(p)

Fall (a).3: $\mathrm{bal}(pp) = +1$. (a).3.3: $\mathrm{bal}(q) = -1$.[11]



$\Longrightarrow$

Doppelrotation rechts (z) links (y)



plus **upout(r)**.

[11](b).3.3: $\mathrm{bal}(pp) = -1$, $\mathrm{bal}(q) = -1$, Links-Rechts-Rotation + upout

Zusammenfassung

- AVL-Bäume haben asymptotische Laufzeit von $\mathcal{O}(\log n)$ (schlechtester Fall) für das Suchen, Einfügen und Löschen von Schlüsseln
- Einfügen und Löschen ist verhältnismässig aufwändig und für kleine Probleme relativ langsam.