

Algorithmen und Datenstrukturen

Sommersemester 2020

Stefan Näher

Universität Trier

naeher@uni-trier.de

Vorlesung 12

3. Juni 2020

AVL-Bäume: Balance

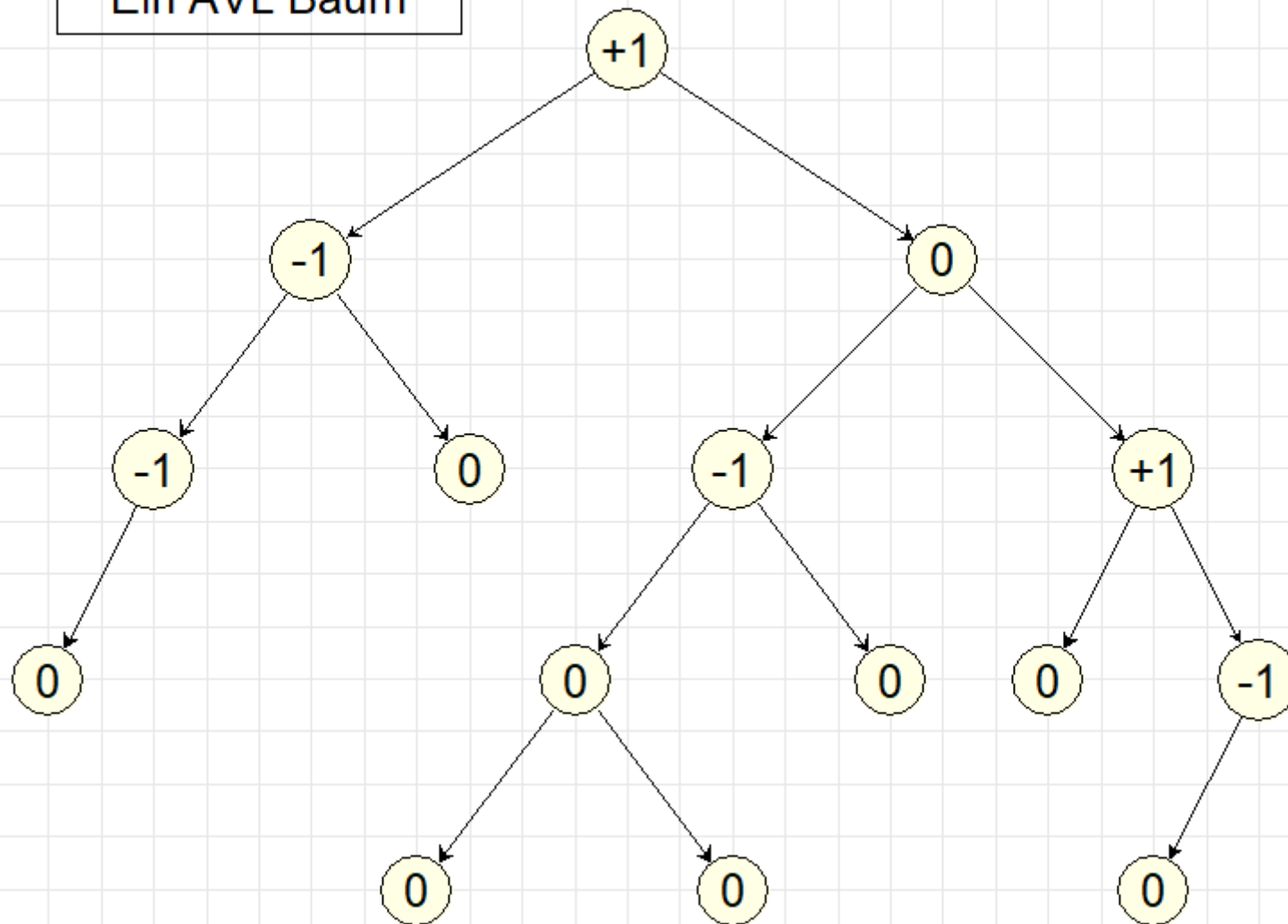
Jedem Knoten können wir als **Balancewert** die Differenz der Höhen des rechten und linken Teilbaumes zugeordnet.

$$\delta(v) = \text{höhe}(T_r(v)) - \text{höhe}(T_\ell(v))$$

Ein binärer Suchbaum T heißt AVL-Baum, wenn für alle Knoten v gilt:

$$\delta(v) \in \{-1, 0, +1\}$$

Ein AVL Baum



AVL-Bäume: Rebalancierung nach Einfügen

Nach dem Einfügen eines neuen Blattes v , wird der Suchpfad von v aus rückwärts durchlaufen, um nach Knoten zu schauen, die aus der Balance geraten sind.

Wird solch ein Knoten x mit einem Balancewert von 2 oder (-2) gefunden, so muss rebalanciert werden.

Dies geschieht durch **Rotationen** bzw. **Doppelrotationen**.

Wir orientieren uns hier an der Darstellung aus dem Buch “Datenstrukturen und Algorithmen” von Ralf Hartmut Güting.

AVL-Bäume: Einfügen

Wir behandeln hier den Fall **a**, dass $\delta(x) = +2$ d.h. der rechte Teilbaum von x ist zu tief geworden (der Fall -2 ist symmetrisch).

Sei y das rechte Kind von x .

Wir unterscheiden 2 Unterfälle **a1** und **a2** und werden sehen, dass nach einem INSERT **eine einzige** Rotation bzw. Doppelrotation ausreicht, um den AVL-Baum zu rebalancieren.

Fall a1 Der **rechte** Teilbaum von y ist zu groß (dort Einfügestelle).

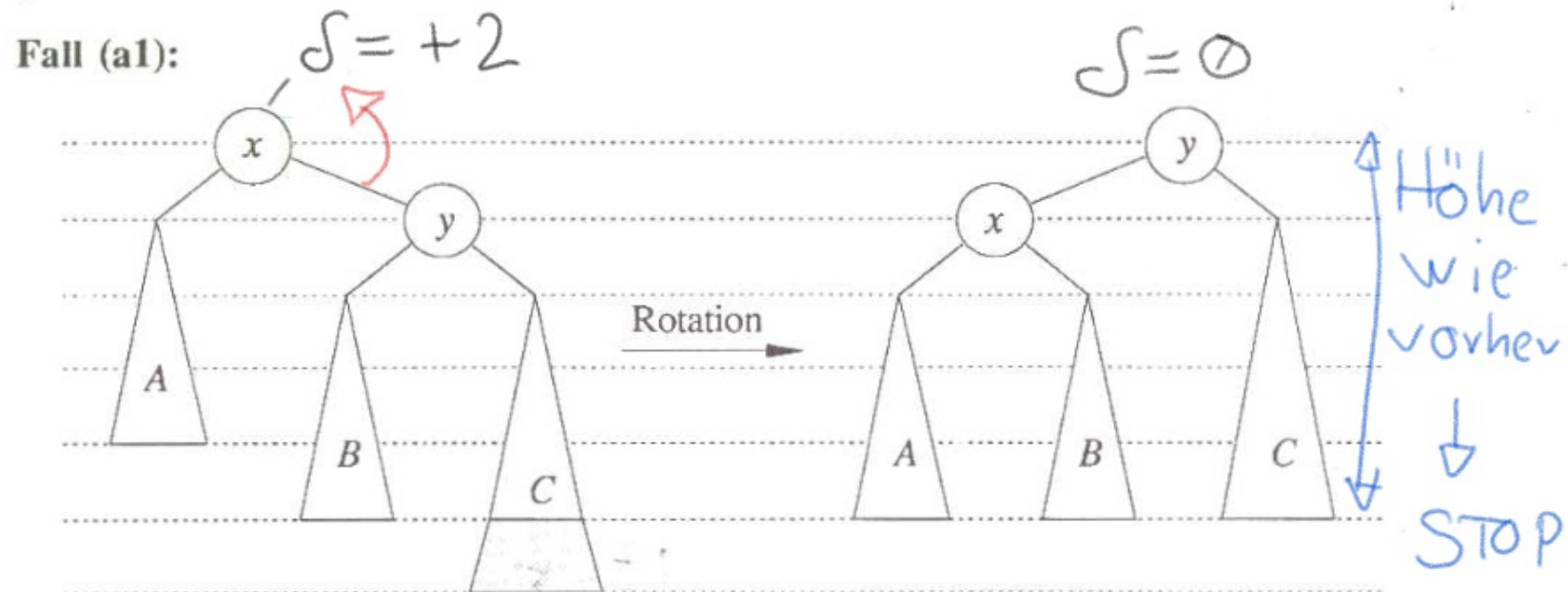


Abbildung 4.9: Anschließend Zustand ok, Höhe um 1 vermindert

Fall a2 Der **linke** Teilbaum von y ist zu groß (dort Einfügestelle).

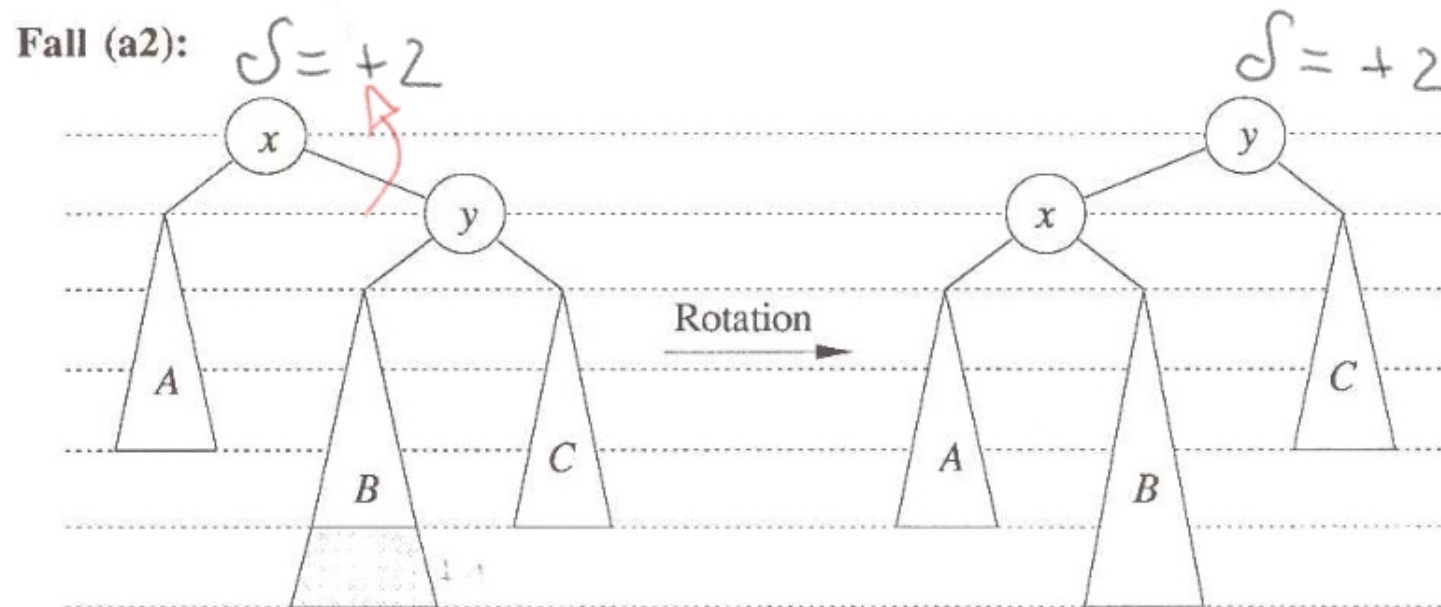


Abbildung 4.10: Anschließend Zustand nicht ok, Rotation reicht nicht

In einem solchen Fall kann eine Operation wie oben die Balance offensichtlich nicht wiederherstellen. Wir sehen uns Teilbaum B im Fall (a2) genauer an:

Fall a2.1 Der **linke** Teilbaum von y ist zu groß (dort Einfügestelle).

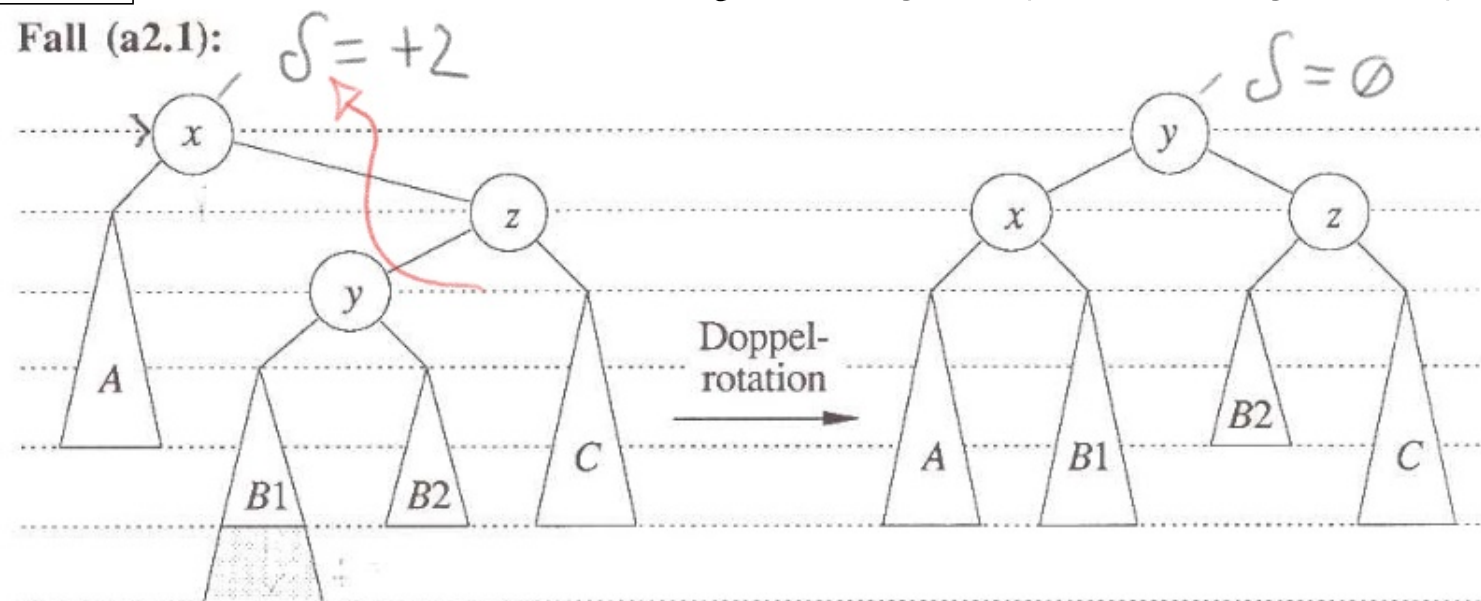


Abbildung 4.11: Anschließend Zustand ok, Höhe unverändert $\rightarrow$ STOP

Fall a2.2 Der **linke** Teilbaum von y ist zu groß (dort Einfügestelle).

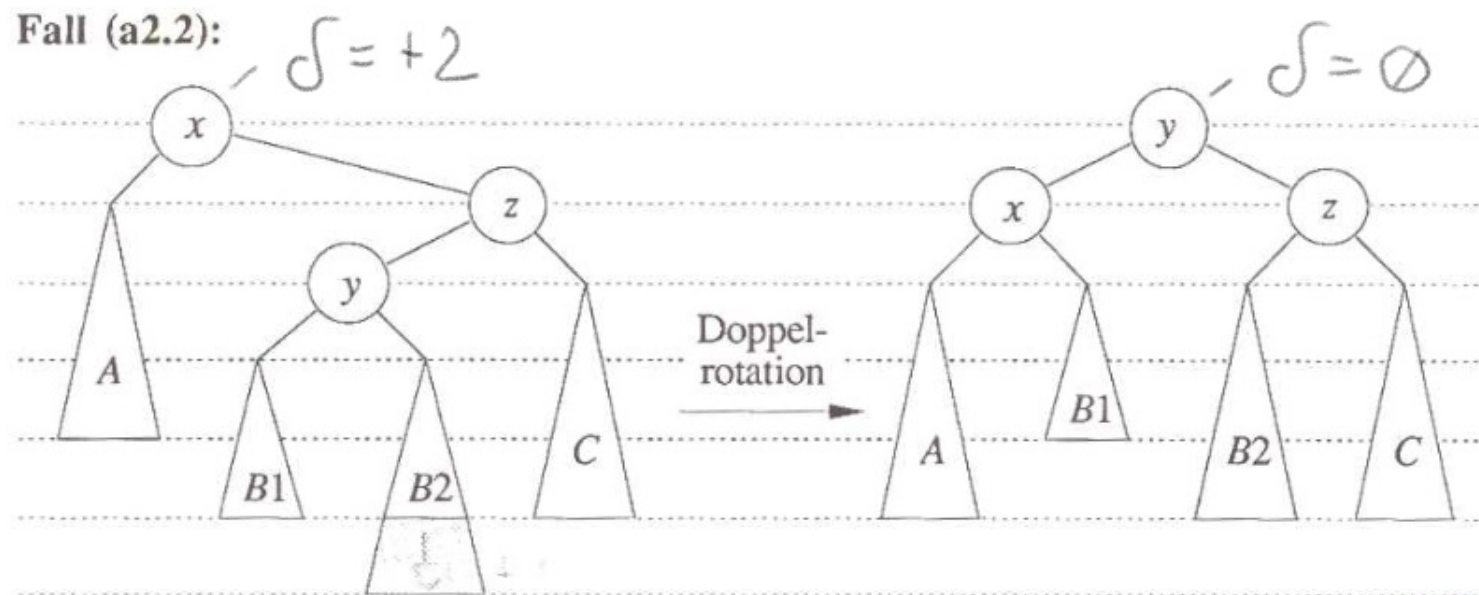


Abbildung 4.12: Anschließend Zustand ok, Höhe unverändert $\rightarrow$ STOP

AVL-Bäume: Löschen

Wir betrachten jetzt den Fall, dass ein Knoten wegen einer DELETE - Operation aus der Balance geraten ist.

Sei also $\delta(x) = +2$, weil der linke Unterbaum durch ein DELETE nun zu flach geworden ist (der Fall $\delta(x) = -2$ ist wieder symmetrisch).

Sei wieder y das rechte Kind von x . Wir unterscheiden drei Fälle

b1 der rechte Unterbaum von y ist tiefer als der linke

b2 die beiden Unterbäume von y sind gleich tief

b3 der linke Unterbaum von y ist tiefer als der rechte

der letzte Fall (b3) muss in 3 weiteren Unterfällen genauer untersucht werden.

Fall b1 Der **rechte** Teilbaum von y ist tiefer.

Fall **b1**

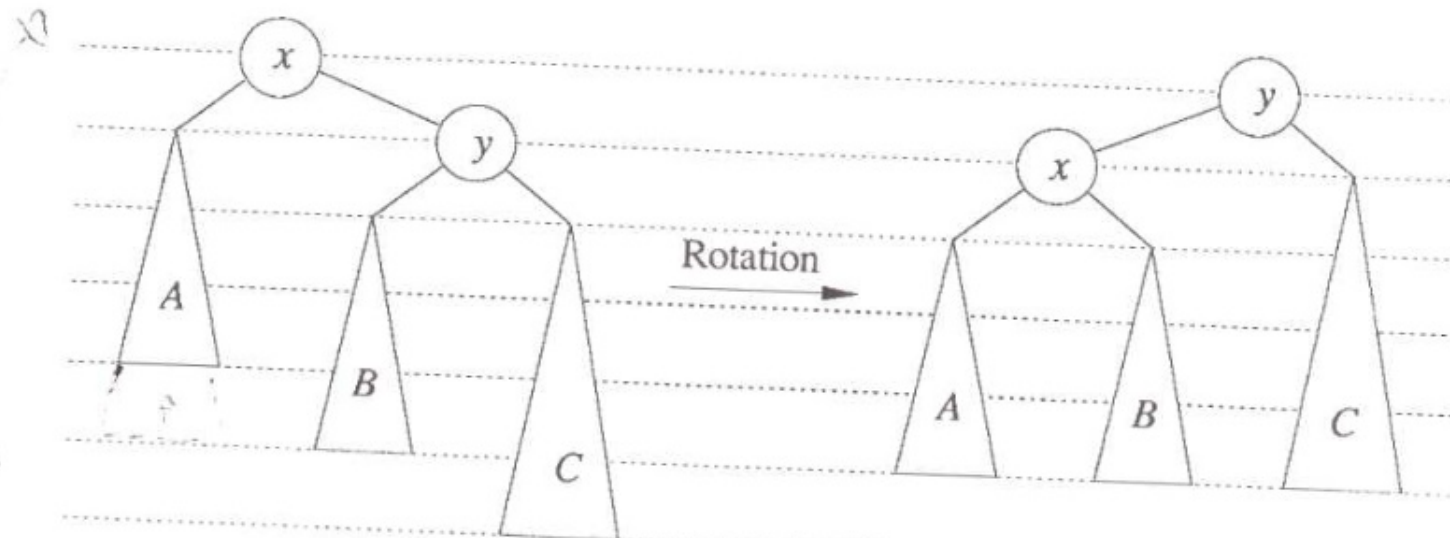


Abbildung 4.14: Anschließend Zustand ok, Höhe um 1 vermindert

-D CONTINUE!

Fall b2 Die Unterbäume von y sind gleich tief.

Fall b2

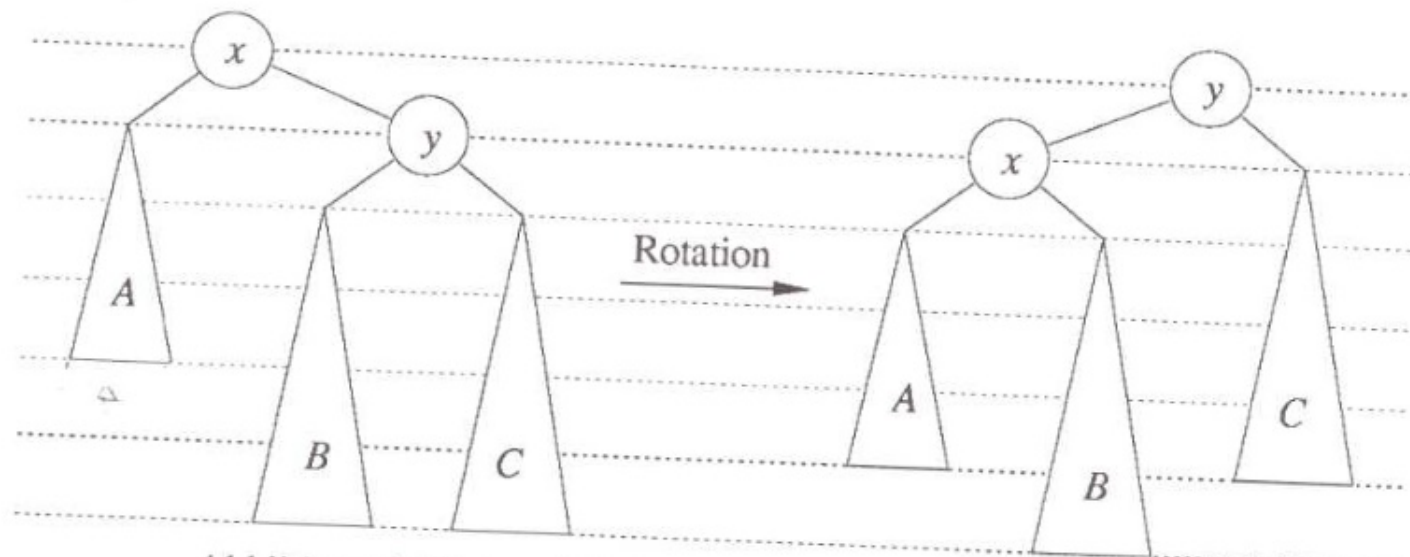
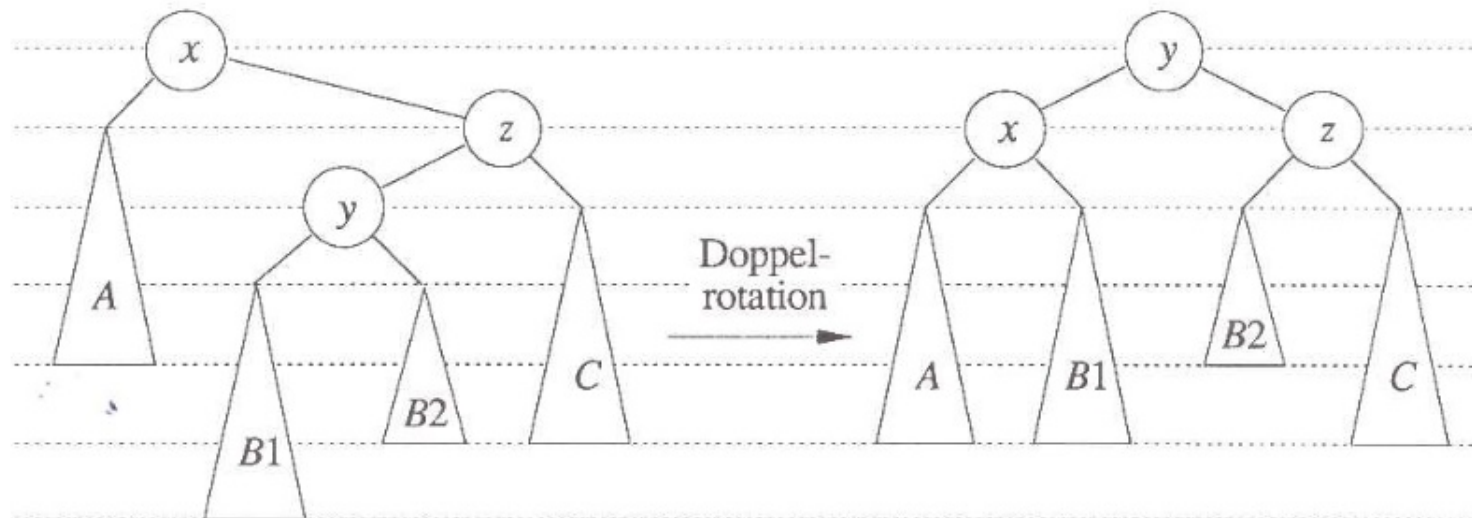


Abbildung 4.15: Anschließend Zustand ok, Höhe unverändert

Stop

Fall b3.1 Der **linke** Teilbaum ist tiefer und.

~~Fall (1-1-1)~~
Fall b3.1.

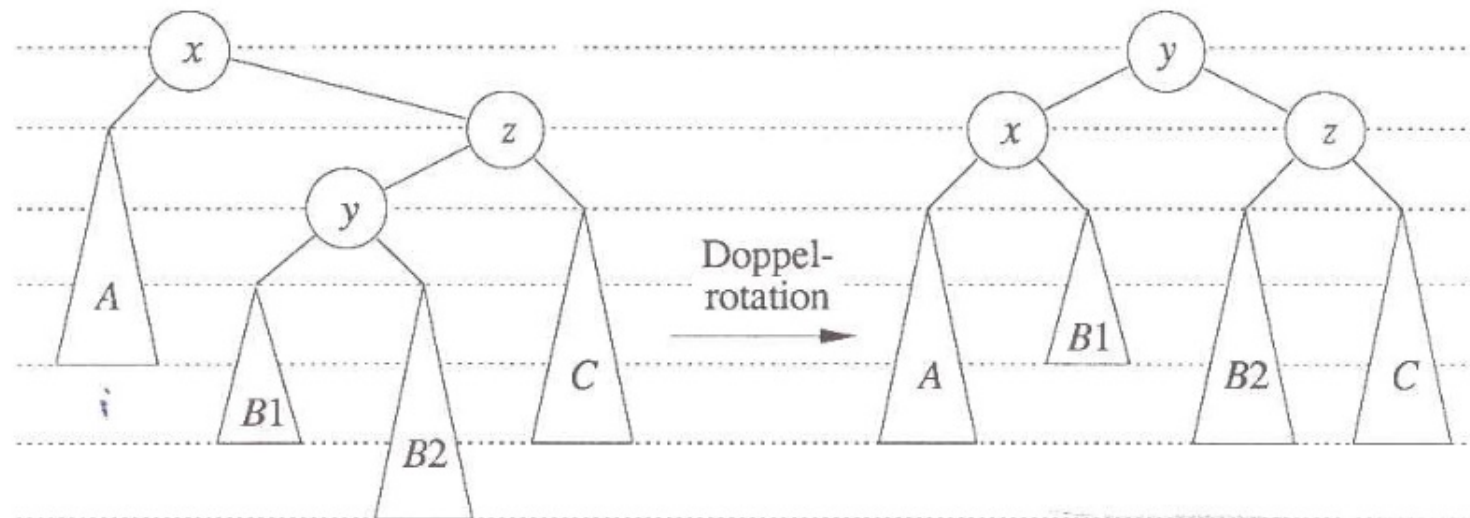


~~Abbildung 4.16~~ Abbildung 4.16: Anschließend Zustand ok, Höhe um 1 vermindert *→ continue*

Fall b3.2 Der **linke** Teilbaum ist tiefer und.

~~b3.2 Abbildung 4.16: Anschließend Zustand ok, Höhe um 1 vermindert~~

Fall ~~b3.2~~: **b3.2**



~~Abbildung 4.17: Anschließend Zustand ok, Höhe um 1 vermindert~~ **ce-hack**

Fall b3.3 Der **linke** Teilbaum ist tiefer und.

Fall (b3.3): b3.3

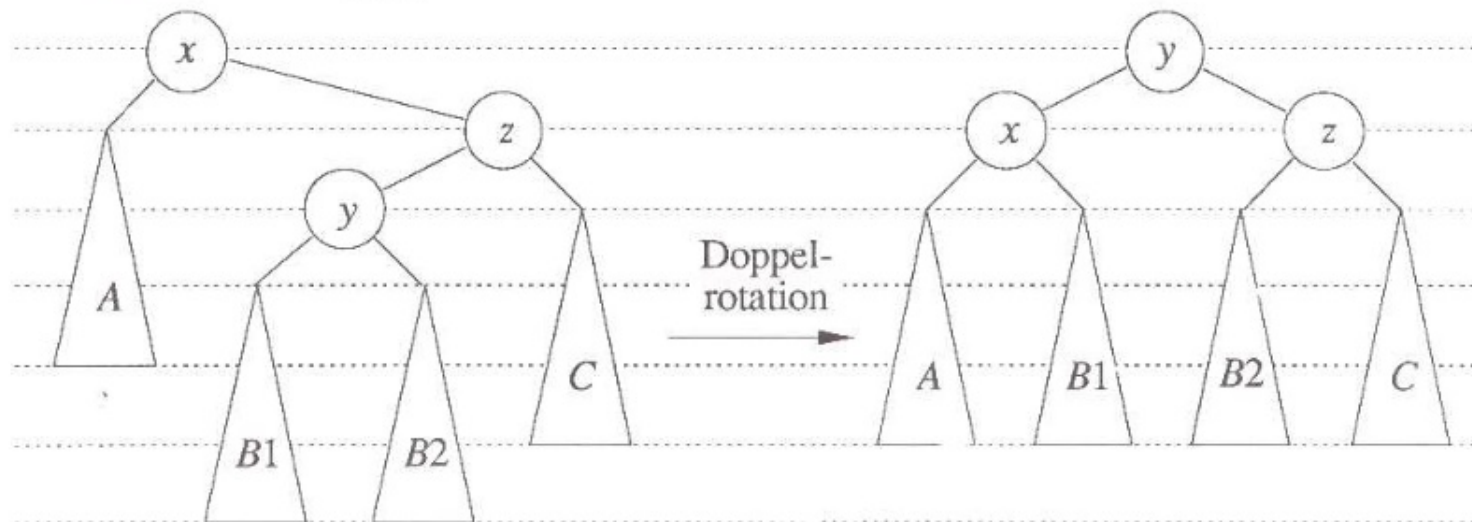


Abbildung 4.18: Anschließend Zustand x , Höhe um 1 vermindert

Maximale Höhe eines AVL-Baums

Wir möchten die maximale Höhe eines AVL-Baumes mit n Knoten nach oben abschätzen.

Dazu beweisen wir umgekehrt eine untere Schranke für die minimale Zahl von Knoten in einem AVL-Baum mit einer gegebenen Höhe h .

Sei $N(h)$ die **minimale Anzahl** von Knoten in einem AVL-Baum der Höhe h .

Dann gilt:

$$N(0) = 1$$

$$N(1) = 2 \quad \text{und}$$

$$N(h) \geq 1 + N(h-1) + N(h-2) \quad \text{für } h > 1$$

Das erinnert an die Folge der **Fibonacci-Zahlen**

$$F_0 = 0, F_1 = 1 \quad \text{und}$$

$$F_i = F_{i-1} + F_{i-2} \quad \text{für } i > 1$$

Wertetabelle für $N(i)$ und F_i

i	0	1	2	3	4	5	6	7	8	9	10	11	12	...
F_i	0	1	1	2	3	5	8	13	21	34	55	89	144	...
$N(i)$	1	2	4	7	12	20	33	54	88	143	232	376	...	

Lemma 1: $N(i) = F_{i+3} - 1$

Beweis (durch Induktion)

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

$$= 1 + (F_{i+2} - 1) + (F_{i+1} - 1) \quad \text{I.A.}$$

$$= F_{i+3} - 1$$

Lemma 2: $F_{i+2} \geq \Phi^i$ mit $\Phi = \frac{1+\sqrt{5}}{2}$

Beweis (durch Induktion)

Es gilt: $\Phi^2 = \Phi + 1$ **Warum ?**

$$\begin{aligned}
\Phi^2 &= \left(\frac{1 + \sqrt{5}}{2} \right)^2 \\
&= \frac{1 + 2\sqrt{5} + 5}{4} \\
&= \frac{6 + 2\sqrt{5}}{4} \\
&= \frac{3 + \sqrt{5}}{2} \\
&= 1 + \frac{1 + \sqrt{5}}{2} \\
&= 1 + \Phi
\end{aligned}$$

$$F_{i+2} = F_{i+1} + F_i$$

$$\geq \Phi^{i-1} + \Phi^{i-2} \quad \text{nach Induktionsannahme}$$

$$= \Phi^{i-2} \cdot (\Phi + 1)$$

$$= \Phi^{i-2} \cdot \Phi^2 \quad (\text{da } \Phi + 1 = \Phi^2)$$

$$= \Phi^i$$

Abschätzung der Höhe

Sei T nun ein AVL-Baum mit n Knoten und $h = \text{Höhe}(T)$.
Dann gilt

$$\begin{aligned} n &\geq N(h) \quad \text{da } N(h) \text{ Mindestanzahl von Knoten in } T \\ &\geq F_{h+3} - 1 \quad (\text{Lemma 1}) \\ &\geq F_{h+2} \\ &\geq \left(\frac{1 + \sqrt{5}}{2} \right)^h \quad (\text{Lemma 2}) \\ &\geq 1.618^h \end{aligned}$$

oder

$$h \leq \log_{1.618} n \leq 1.4405 \cdot \log_2 n = \mathcal{O}(\log n)$$

Zusammenfassung AVL-Bäume

AVL-Bäume sind **höhenbalancierte binäre Suchbäume**

Ein AVL-Baum T mit n Knoten

- hat eine Höhe von maximal $1.44 \cdot \log n$
- unterstützt die Wörterbuchoperationen LOOKUP, INSERT, DELETE in Zeit $\mathcal{O}(\text{Höhe}(T)) = \mathcal{O}(\log n)$.
- benötigt Speicherplatz $\mathcal{O}(n)$.