

AVL-Bäume

(Algorithmen und Datenstrukturen I)

Prof. Dr. Oliver Braun

Letzte Änderung: 18.03.2018 18:16

Balancierte Bäume

- ▶ in einem zufällig erzeugten Binärbaum haben die Algorithmen *Suchen*, *Einfügen* und *Löschen*
 - ▶ im Mittel eine Laufzeit von $O(\log_2 N)$
 - ▶ aber im schlechtesten Fall eine lineare Laufzeit
 - ▶ entarteter Baum
- ▶ das Degenerieren des Baumes soll durch eine zusätzliche Bedingung an die Struktur der Bäume verhindert werden

- ▶ ein binärer Suchbaum ist **AVL-ausgeglichen** oder **höhenbalanciert**
 - ▶ kurz: **ein AVL-Baum**
- ▶ wenn für jeden Knoten p des Baumes gilt:
 - ▶ die Höhe des linken Teilbaumes unterscheidet sich höchstens um 1 von der Höhe des rechten Teilbaumes

- ▶ exakt wie in natürlichem Baum

Einfügen

- ▶ zunächst wie in natürlichem Baum
 - ▶ ersetze Blatt durch inneren Knoten mit Schlüssel
- ▶ aber: der entstehende Suchbaum ist eventuell nicht mehr AVL-ausgeglichen
- ▶ dazu läuft man den Pfad von der Einfügestelle zur Wurzel entlang
 - ▶ und überprüft für jeden Knoten ob die Höhendifferenz höchstens 1 ist
- ▶ ist das nicht der Fall kann man eine **Rotation** oder **Doppelrotation** ausführen
 - ▶ erhält Sortierung der Schlüssel
 - ▶ aber verändert die Höhendifferenz

Balancefaktor

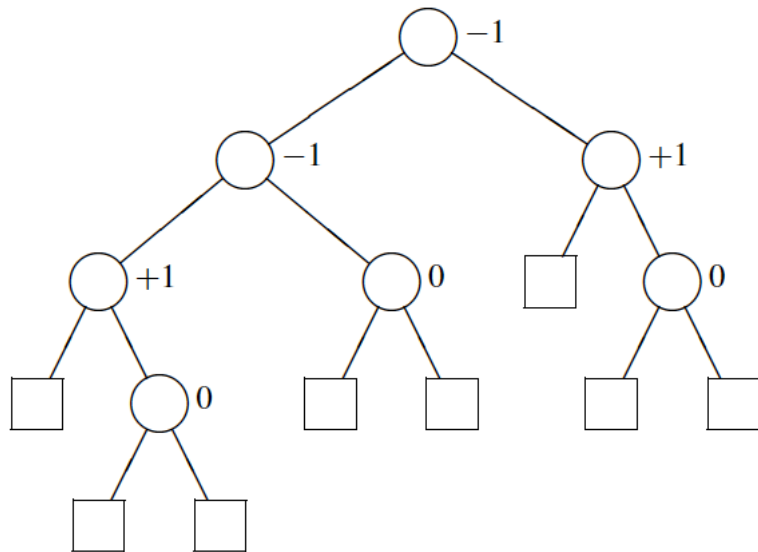
- ▶ zur Prüfung der Höhenbedingung reicht es den Balancefaktor zusätzlich in jedem inneren Knoten zu speichern
- ▶ es ist nicht notwendig die Höhe der Teilbäume zu speichern
- ▶ der **Balancefaktor** $bal(p)$ ist definiert durch

$$bal(p) = \text{Höhe des rechten Teilbaumes von } p \\ - \text{Höhe des linken Teilbaumes von } p.$$

- ▶ damit gilt für jeden inneren Knoten p :

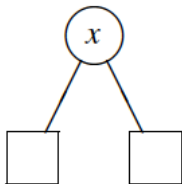
$$bal(p) \in \{-1, 0, 1\}$$

Beispiel



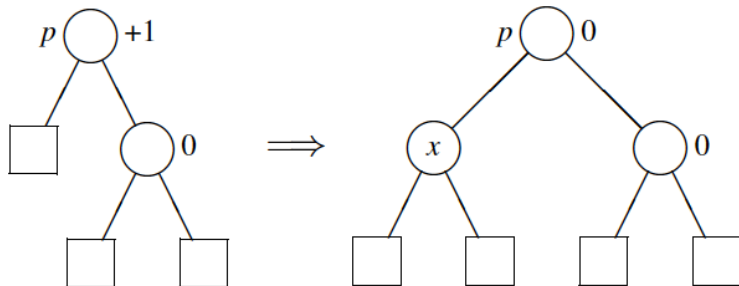
Einfügen in leeren Baum

- ▶ Einfügen des Schlüssel x in den leeren Baum



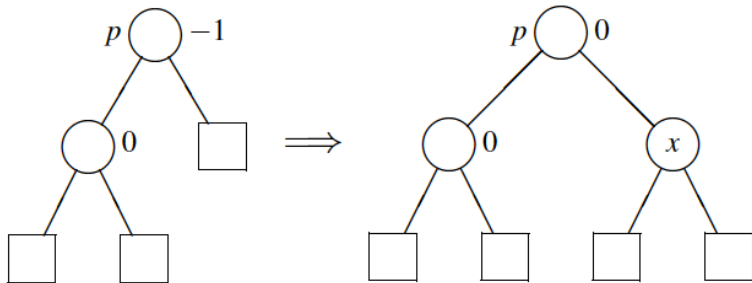
Einfügen mit Vorgänger (1/2)

- ▶ sei p der Vorgänger des Blattes an dem die Suche endet
- ▶ drei Fälle sind möglich
- ▶ Fall 1: $bal(p) = +1$



Einfügen mit Vorgänger (2/2)

- ▶ Fall 2: $bal(p) = -1$



- ▶ Fall 3: $bal(p) = 0$

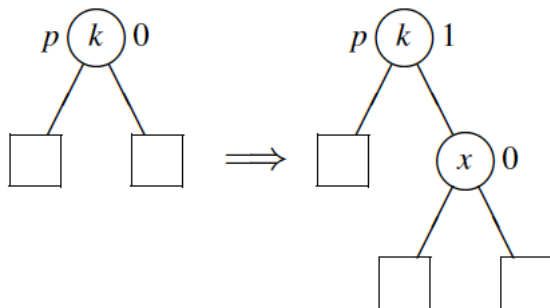
- ▶ nur in diesem Fall ändert sich die Höhe des Baumes mit der Wurzel p
- ▶ p hat nach dem Einfügen einen Balancefaktor 1 oder -1

Die Prozedur $upin(p)$

- ▶ da sich der Balancefaktor von p geändert hat, haben sich eventuell auch die Balancefaktoren der Knoten auf dem Pfad von p zur Wurzel geändert
 - ▶ aber nur diese
- ▶ dazu wird eine Prozedur $upin(p)$ definiert, die den Pfad zur Wurzel zurück läuft und die Balancefaktoren anpasst

$$bal(p) = 0 \quad (1/2)$$

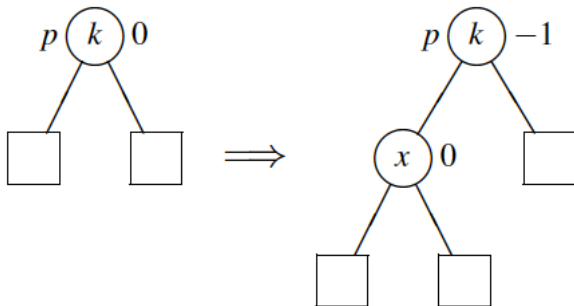
- ▶ Fall 3.1: $bal(p) = 0$ und einzufügender Schlüssel $x >$ Schlüssel k von p



- ▶ anschließend Aufruf $upin(p)$

$bal(p) = 0$ (2/2)

- ▶ Fall 3.2: $bal(p) = 0$ und einzufügender Schlüssel $x <$ Schlüssel k von p



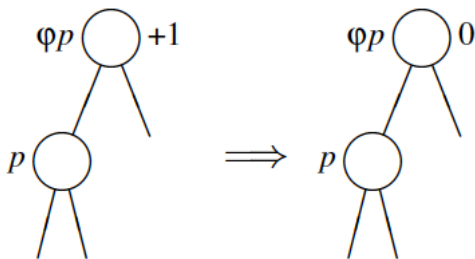
- ▶ anschließend Aufruf $upin(p)$

upin(p)

- ▶ wenn *upin(p)* aufgerufen wird, ist $bal(p) \in \{-1, 1\}$ und die Höhe des Teilbaumes mit der Wurzel p ist um 1 gewachsen
- ▶ diese Invariante muss vor jedem rekursiven Aufruf von *upin* gelten
- ▶ *upin(p)* bricht ab, wenn p die Wurzel ist

Fall 1: p ist linker Nachfolger von φp (1/7)

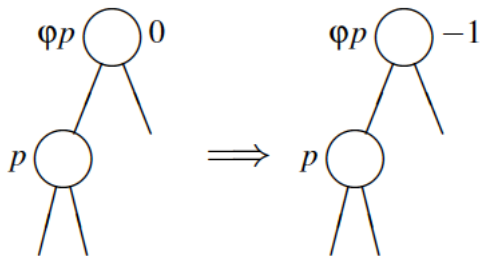
- ▶ Fall 1.1: $bal(\varphi p) = 1$



- ▶ fertig

Fall 1: p ist linker Nachfolger von φp (2/7)

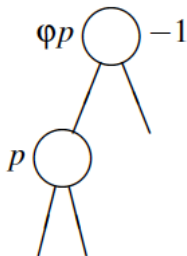
- ▶ Fall 1.2: $bal(\varphi p) = 0$



- ▶ $upin(\varphi p)$

Fall 1: p ist linker Nachfolger von φp (3/7)

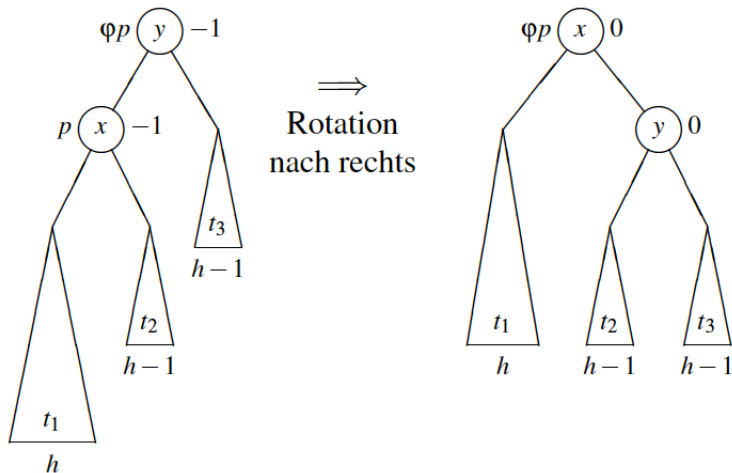
- ▶ Fall 1.3: $bal(\varphi p) = -1$



- ▶ der linke Teilbaum von φp war schon höher als der rechte Teilbaum
- ▶ da der Baum von p um 1 gewachsen ist, ist die AVL-Ausgeglichenheit nun bei φp verletzt

Fall 1: p ist linker Nachfolger von φp (4/7)

- Fall 1.3.1 $bal(p) = -1$, anschließend fertig

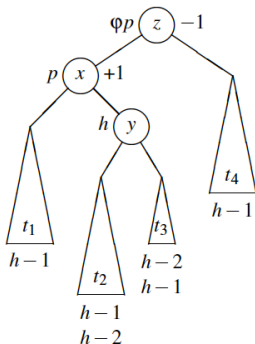


Fall 1: p ist linker Nachfolger von φp (5/7)

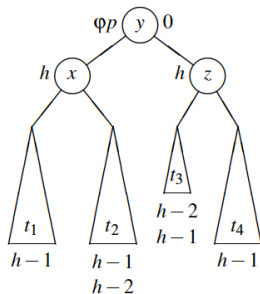
- ▶ nach Voraussetzung:
 - ▶ Baum von p um 1 gewachsen
 - ▶ linker Teilbaum von p um 1 höher als rechter
- ▶ nach Rotation ist Teilbaum mit Wurzel φp nicht gewachsen

Fall 1: p ist linker Nachfolger von φp (6/7)

- ▶ Fall 1.3.2 $bal(p) = +1$, anschließend fertig



\implies
Doppel-
rotation
links-rechts



Fall 1: p ist linker Nachfolger von φp (7/7)

- ▶ für t_2 und t_3 gilt:
 - ▶ entweder beide leer
 - ▶ oder einer Höhe $h - 1$ und der andere Höhe $h - 2$
 - ▶ können nicht die gleiche Höhe haben,
 - ▶ denn rechter Teilbaum von p ist um 1 gewachsen
 - ▶ und der rechte Teilbaum ist um 1 höher als der linke

Fall 2: p ist rechter Nachfolger von φp (1/1)

- ▶ analog zu Fall 1
- ▶ wenn nötig Rotation nach links
- ▶ oder Doppelrotation rechts links

Zusammenfassung Einfügen

- ▶ im schlimmsten Fall wird *upin* für alle Knoten auf dem Suchpfad bis zur Wurzel aufgerufen
- ▶ in jedem Fall wird aber nur eine Rotation oder Doppelrotation durchgeführt
- ▶ damit ist klar, dass Einfügen eines neuen Schlüssels in einen AVL-Bau mit N Schlüsseln in $O(\log N)$ Schritten ausführbar ist

Entfernen eines Schlüssels

- ▶ zunächst Suche des zu entfernenden Schlüssels
- ▶ nicht gefunden \Rightarrow fertig
- ▶ sonst 3 Fälle
 1. beide Nachfolger Blätter
 2. ein Nachfolger innerer Knoten und einer Blatt
 3. beide Nachfolger innere Knoten

Fall 1: beide Nachfolger Blätter (1/2)

- ▶ beide Nachfolger des Knotens mit dem zu entfernenden Schlüssel sind Blätter
- ▶ ersetzen des Knotens durch ein Blatt
- ▶ sei p der Vorgänger des neuen Blattes
- ▶ dann muss der andere Teilbaum q von p die Höhe 0, 1 oder 2 haben

Fall 1: beide Nachfolger Blätter (2/2)

- ▶ q hat Höhe 1: jetzt $bal(p) \in \{-1, 1\}$
- ▶ q hat Höhe 0: jetzt $bal(p) = 0$ und Höhe von p hat sich um 1 verringert
 - ▶ Aufruf einer Prozedur *upout* auf dem Suchpfad zur Wurzel
- ▶ q hat Höhe 2:
 - ▶ zuerst Rotation oder Doppelrotation um Baum mit Wurzel p wieder auszugleichen
 - ▶ wenn die neue Wurzel dieses Teilbaumes neuen Balancefaktor 0 hat, ist die Höhe um 1 gesunken
 - ▶ dann Aufruf von *upout*

Fall 2: ein Nachfolger innerer Knoten und einer Blatt

- ▶ wenn p nur einen inneren Knoten q und ein Blatt als Nachfolger hat
- ▶ müssen beide Nachfolger von q Blätter sein
- ▶ dann wird der Schlüssel von p durch den Schlüssel von q ersetzt und q durch ein Blatt
- ▶ da die Höhe des Teilbaumes mit Wurzel p von 2 auf 1 gesunken ist
 - ▶ Aufruf von *upout*

Fall 3: beide Nachfolger von p innere Knoten

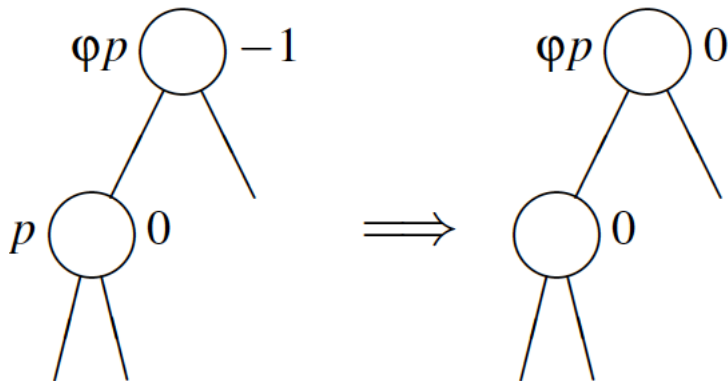
- ▶ wie bei natürlichen Suchbäumen
 - ▶ Ersetzen des Schlüssels von p durch den Schlüssel des symmetrischen Vorgängers oder Nachfolgers
 - ▶ und Entfernen des symmetrischen Vorgängers oder Nachfolgers
- ▶ d.h. es wird ein Knoten entfernt, der entsprechend Fall 1 oder Fall 2 zu behandeln ist

upout(p)

- ▶ wenn *upout(p)* aufgerufen wird, gilt:
 - ▶ $bal(p) = 0$ und
 - ▶ die Höhe des Teilbaumes mit Wurzel p ist um 1 gesunken

Fall 1: p ist linker Nachfolger von φp (1/7)

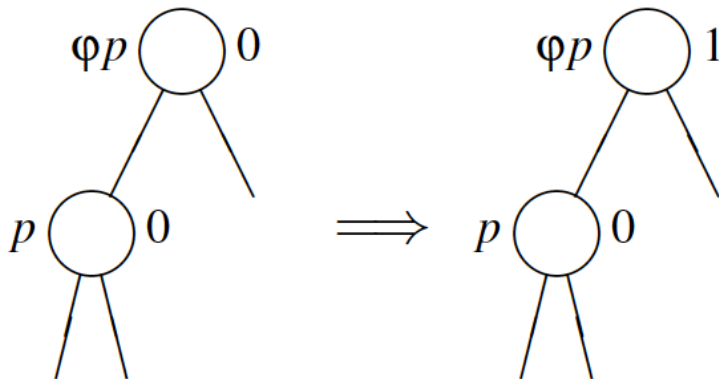
- ▶ Fall 1.1: $bal(\varphi p) = -1$



- ▶ Aufruf von $upout(\varphi p)$

Fall 1: p ist linker Nachfolger von φp (2/7)

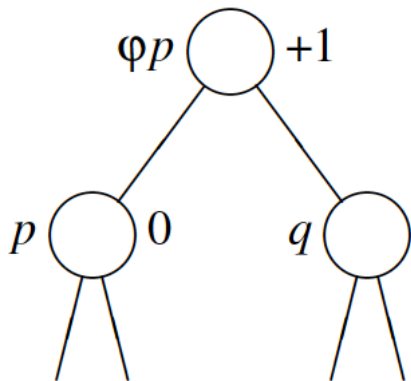
- ▶ Fall 1.2: $bal(\varphi p) = 0$



- ▶ fertig

Fall 1: p ist linker Nachfolger von φp (3/7)

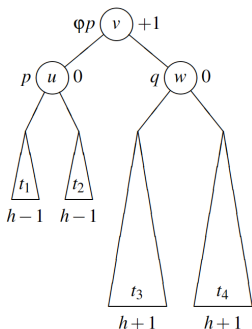
- ▶ Fall 1.3: $bal(\varphi p) = 1$



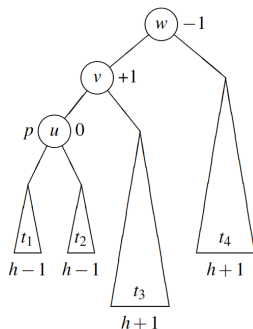
- ▶ der rechte Teilbaum war also schon höher als der linke, der noch um 1 niedriger wurde

Fall 1: p ist linker Nachfolger von φp (4/7)

► Fall 1.3.1: $bal(q) = 0$



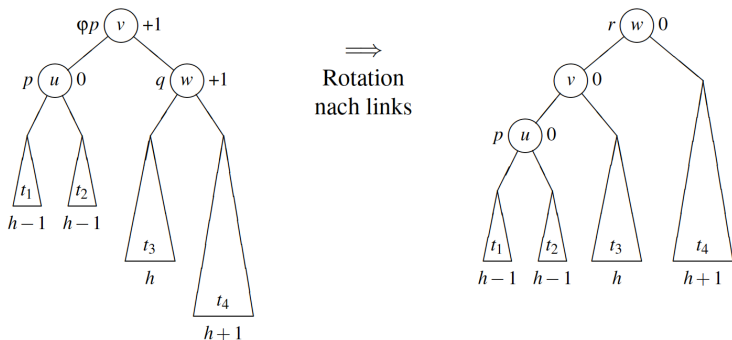
⇒
Rotation
nach links



► fertig

Fall 1: p ist linker Nachfolger von φp (5/7)

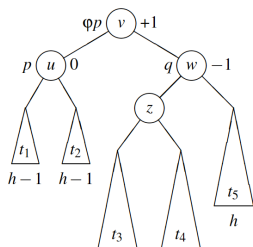
- ▶ Fall 1.3.2: $bal(q) = 1$



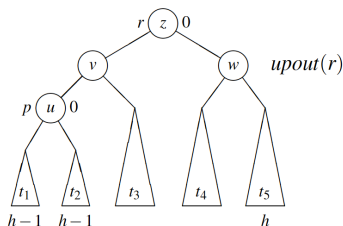
- ▶ $upout(r)$

Fall 1: p ist linker Nachfolger von φp (6/7)

► Fall 1.3.3 $bal(q) = -1$



\implies
Doppel-
rotation
rechts-links



Fall 1: p ist linker Nachfolger von φp (7/7)

- ▶ Teilbaum mit Wurzel q war bereits vor dem Entfernen um 1 höher als Teilbaum mit Wurzel p
- ▶ d.h. Teilbaum mit Wurzel q muss die Höhe $h + 2$ haben
- ▶ wegen $bal(q) = -1$ hat der linke Teilbaum von q die Höhe $h + 1$ und der rechte die Höhe h
- ▶ die Teilbäume von z haben entweder beide die Höhe h oder höchstens einer die Höhe $h - 1$
- ▶ die Höhe des Teilbaumes mit der Wurzel r hat um 1 abgenommen

Fall 2: p ist rechter Nachfolger von φp

- ▶ ist völlig symmetrisch zu Fall 1

Zusammenfassung Entfernen

- ▶ anders als *upin*, kann es sein, dass *upout* nach einer Rotation oder Doppelrotation erneut aufgerufen wird
- ▶ aber Aufwand für Rotation ist konstant und Höhe ist beschränkt
 - ▶ Entfernen eines Schlüssels in einem AVL-Baum mit N Schlüsseln in $O(\log N)$ Schritten ausführbar

Literaturhinweis

Die Graphen in diesem Kapitel sind aus dem Buch *Algorithmen und Datenstrukturen* von Ottmann & Widmayer.