Algorithmen und Datenstrukturen 1

9. Vorlesung

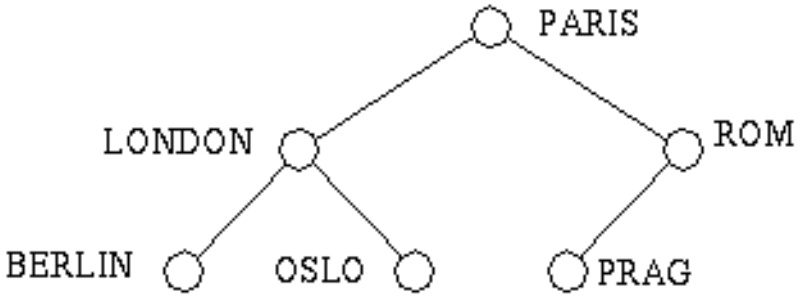
Martin Middendorf / Peter F. Stadler
Universität Leipzig
Institut für Informatik

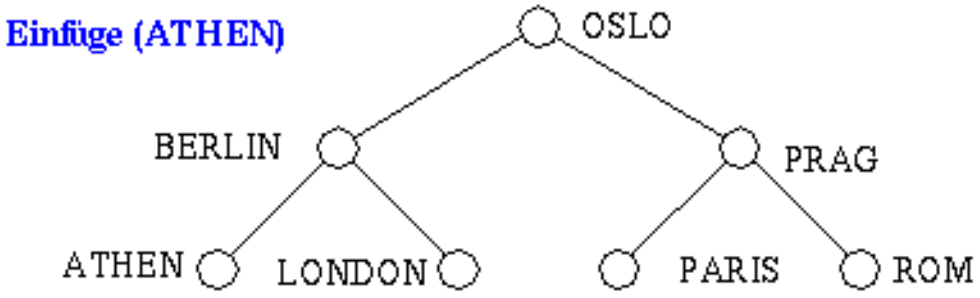
middendorf@informatik.uni-leipzig.de
studla@bioinf.uni-leipzig.de

Balancierte Binärbäume

Der ausgeglichene binäre Suchbaum verursacht für alle Grundoperationen die geringsten Kosten

Perfekte Balancierung zu jeder Zeit ist jedoch sehr teuer.





- In welchem Maße sollen Strukturabweichungen bei Einfügungen und Löschungen toleriert werden ?

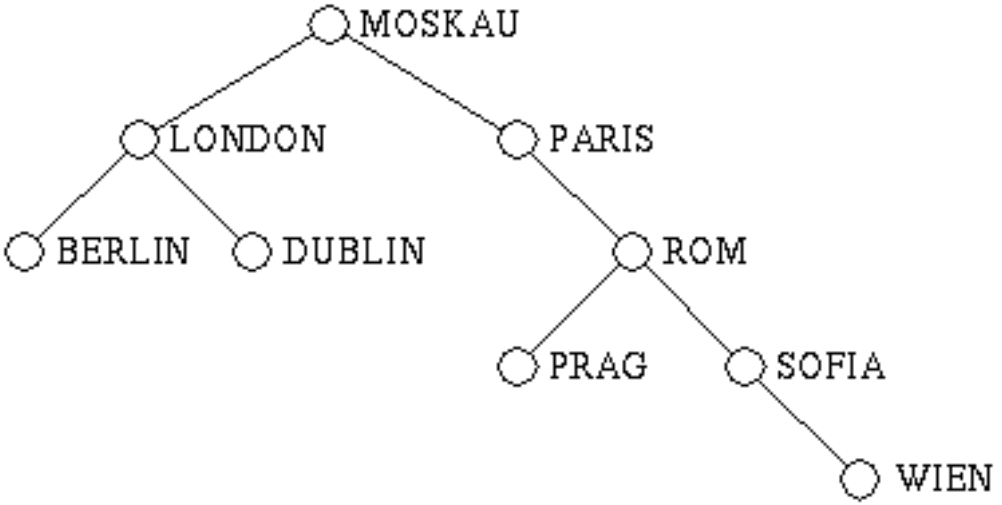
k-balancierter Binärbaum

Def.: Seien $B_l(x)$ und $B_r(x)$ die linken und rechten Unterbäume eines Knotens x. Weiterhin sei h(B) die Höhe eines Baumes B. Ein k-balancierter Binärbaum ist entweder leer oder es ist ein Baum, bei dem für jeden Knoten x gilt:

$$|h(B_l(x)) - h(B_r(x))| \leq k$$

k läßt sich als Maß für die zulässige Entartung im Vergleich zur ausgeglichenen Baumstruktur auffassen.

Prinzip:



$$\left|h(B_l(SOFIA)) - h(B_r(SOFIA))\right| =$$

$$\left|h(B_l(ROM)) - h(B_r(ROM))\right| =$$

$$\left|h(B_l(PARIS)) - h(B_r(PARIS))\right| =$$

$$\left|h(B_l(MOSKAU)) - h(B_r(MOSKAU))\right| =$$

AVL-Baum

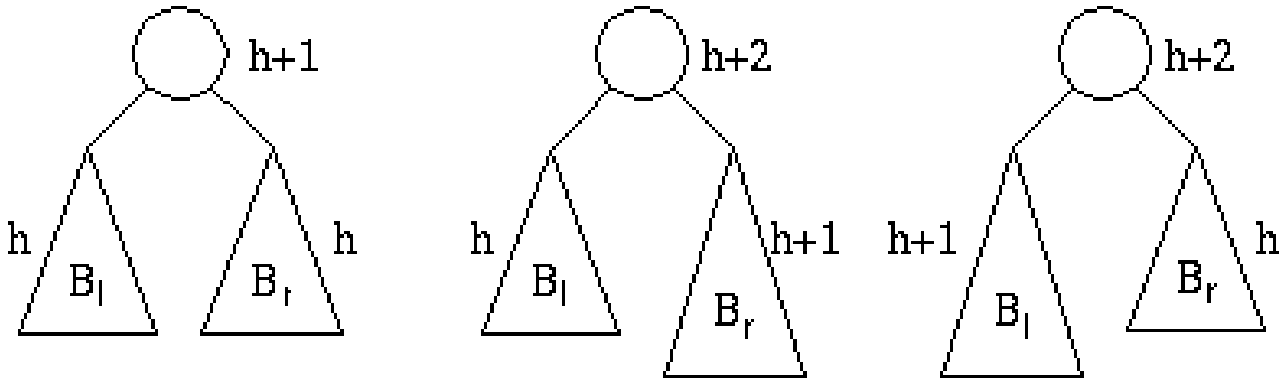
benannt nach russischen Mathematikern: Adelson-Velski und Landis

Definition: Ein 1-balancierter Binärbaum heißt AVL-Baum

-> Balancierungskriterium: $|h(B_l(x)) - h(B_r(x))| <= 1$

Konstruktionsprinzip:

B_l und B_r seien AVL-Bäume der Höhe h und h+1. Dann sind die nachfolgend dargestellten Bäume auch AVL-Bäume:



Suchoperationen wie für allgemeine binäre Suchbäume

AVL-Baum: Wartungsalgorithmen

Wann und wo ist das AVL-Kriterium beim Einfügen verletzt?

- Es kann sich nur die Höhe von solchen Unterbäumen verändert haben, deren Wurzeln auf dem Suchpfad von der Wurzel des Baumes zum neu eingefügten Blatt liegen
- Reorganisationsoperationen lassen sich lokal begrenzen; es sind höchstens h Knoten betroffen

Def.: Der *Balancierungsfaktor* BF(x) eines Knotens x ergibt sich zu

$$BF(x) = h(B_l(x)) - h(B_r(x)).$$

Einfügen in AVL-Bäumen

Sobald ein BF(x) durch eine Einfügung verletzt wird, muss eine Rebalancierung des Baumes durch sog. Rotationen durchgeführt werden.

- Ausgangspunkt der Rotation ist der naheste Vater des neu eingefügten Knotens mit BF = ±2.
- Dieser Knoten dient zur Bestimmung des Rotationstyps. Er wird durch die von diesem Knoten ausgehende Kantenfolge auf dem Pfad zum neu eingefügten Knoten festgelegt.

Rotationstypen

Es treten vier verschiedene Rotationstypen auf. Der neu einzufügende Knoten sei X. Weiter sei Y der bezüglich der Rotation kritische Knoten - der naheste Vater von X mit BF = ±2. Dann bedeutet:

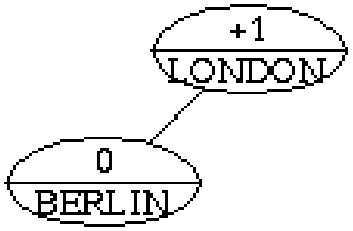
- RR: X wird im rechten Unterbaum des rechten Unterbaums von Y eingefügt (Linksrotation)
- LL: X wird im linken Unterbaum des linken Unterbaums von Y eingefügt (Rechtsrotation)
- RL: X wird im linken Unterbaum des rechten Unterbaums von Y eingefügt (Doppelrotation)
- LR: X wird im rechten Unterbaum des linken Unterbaums von Y eingefügt (Doppelrotation)

Die Typen LL und RR sowie LR und RL sind symmetrisch zueinander.

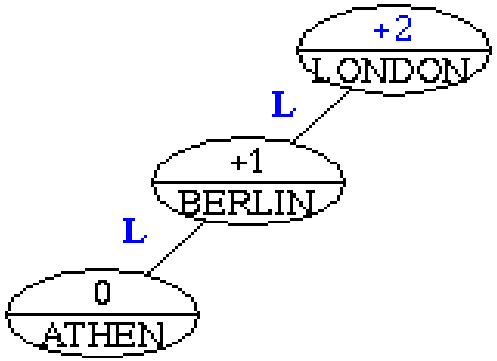
neuer Schlüssel | *nach Einfügung* | *nach Rebalancierung*

LONDON

BERLIN

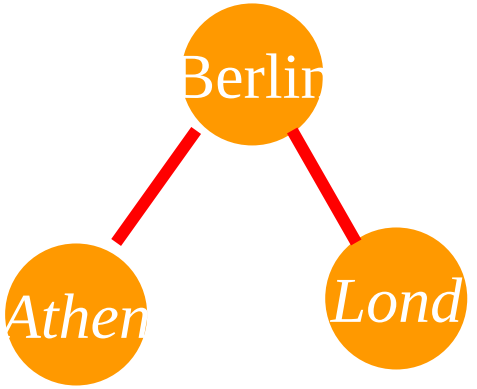


ATHEN

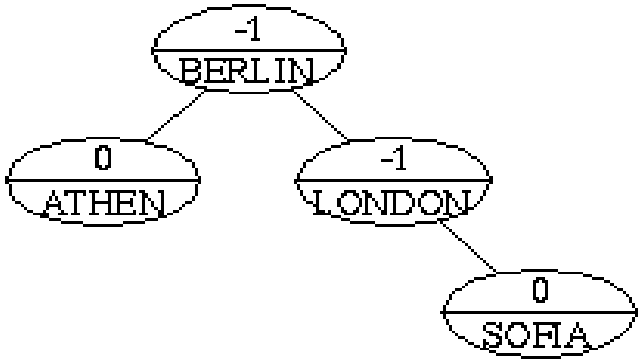


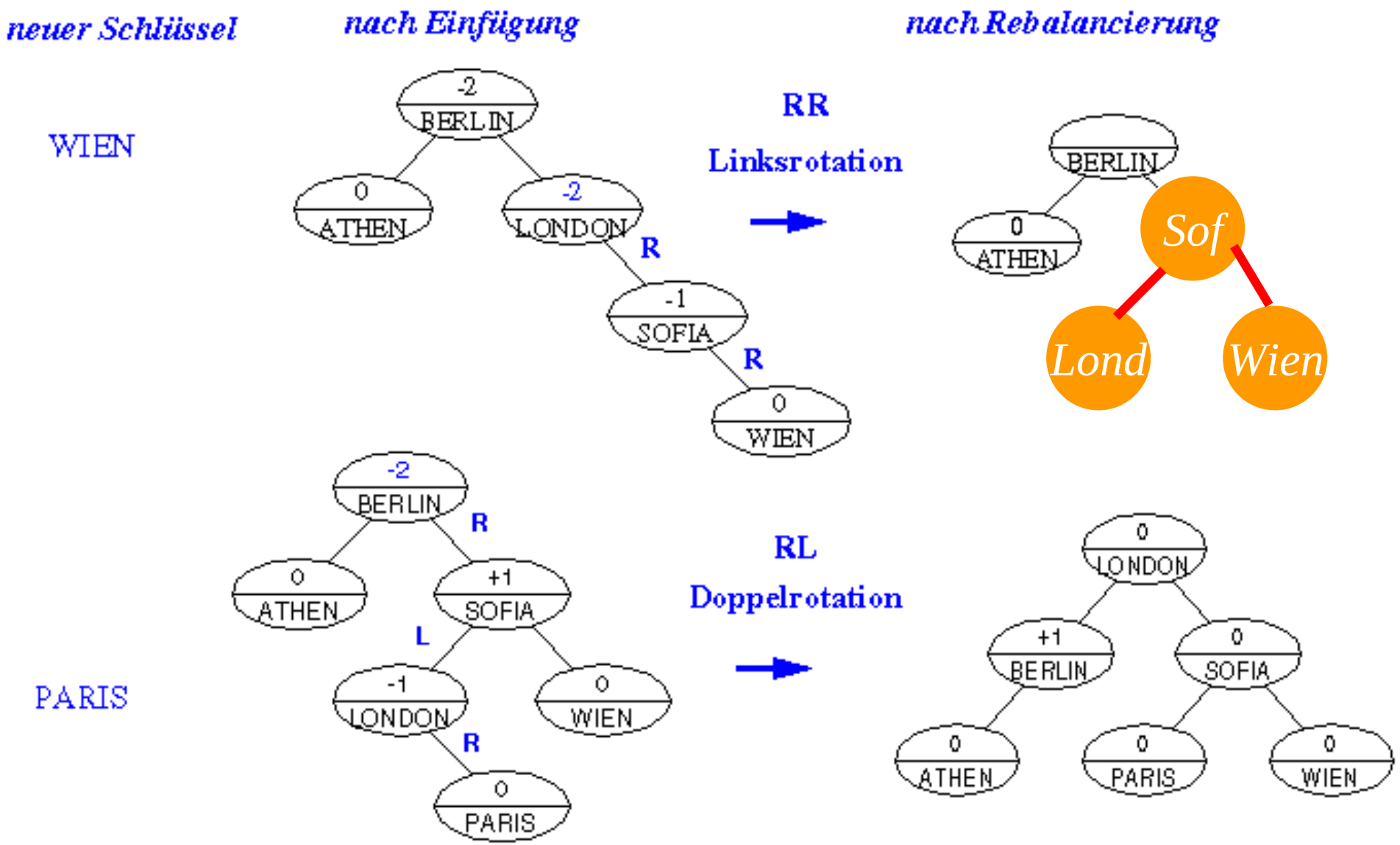
LL

Rechtsrotation



SOFIA

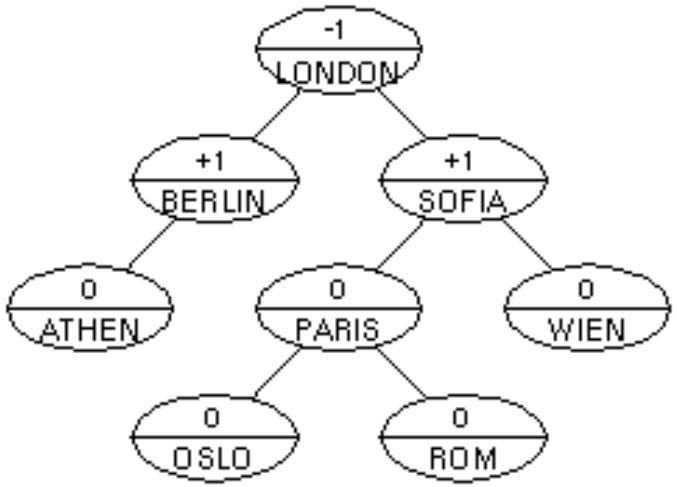



neuer Schlüssel
nach Einfügung
nach Rebalancierung
WIEN
-2
BERLIN
0
ATHEN
-2
LONDON
R
-1
SOFIA
R
0
WIEN
RR
Linksrotation
BERLIN
0
ATHEN
Sof
Lond
Wien
PARIS
-2
BERLIN
R
0
ATHEN
+1
SOFIA
L
-1
LONDON
0
WIEN
R
0
PARIS
RL
Doppelrotation
0
LONDON
+1
BERLIN
0
SOFIA
0
ATHEN
0
PARIS
0
WIEN

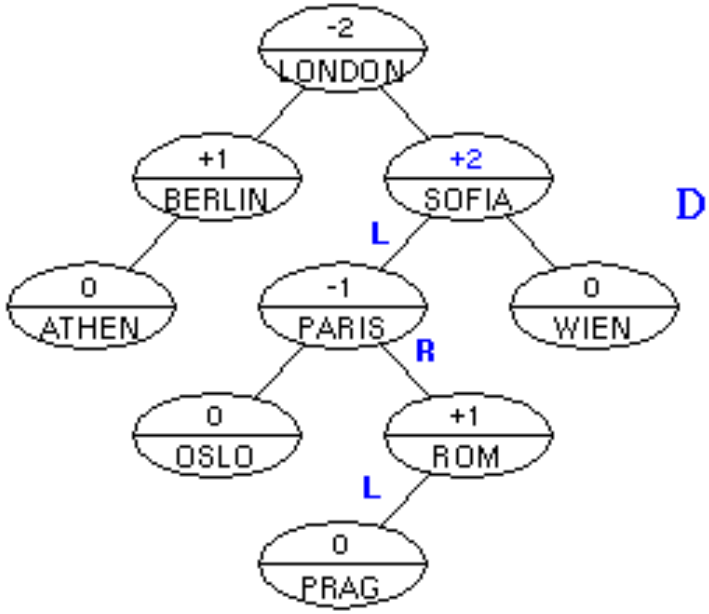
neuer Schlüssel *nach Einfügung* *nach Rebalancierung*

OSLO

ROM

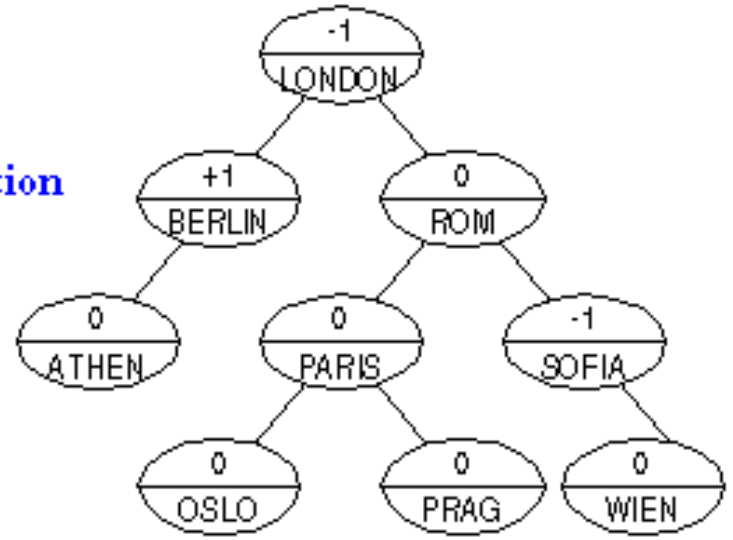


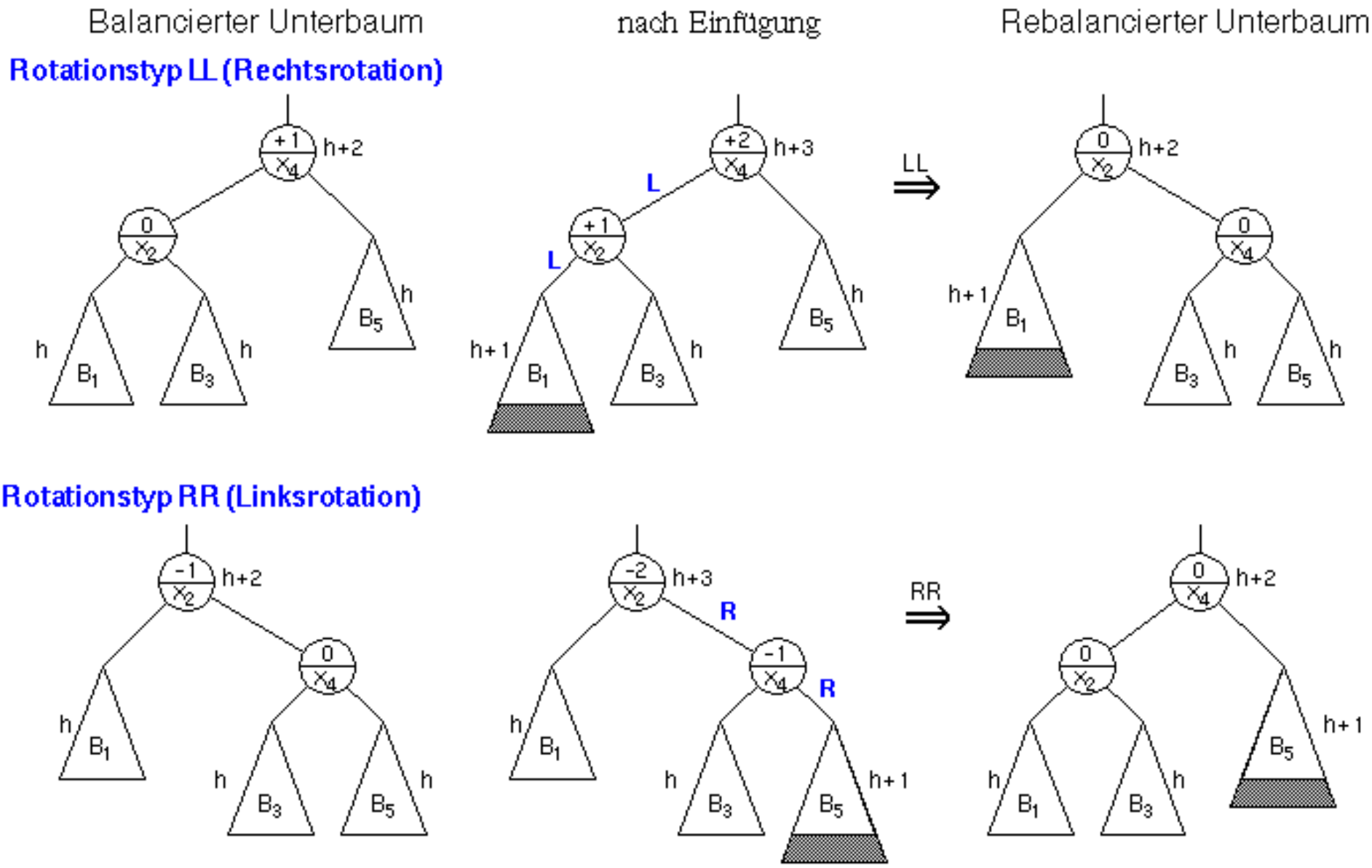
PRAG



LR

Doppelrotation

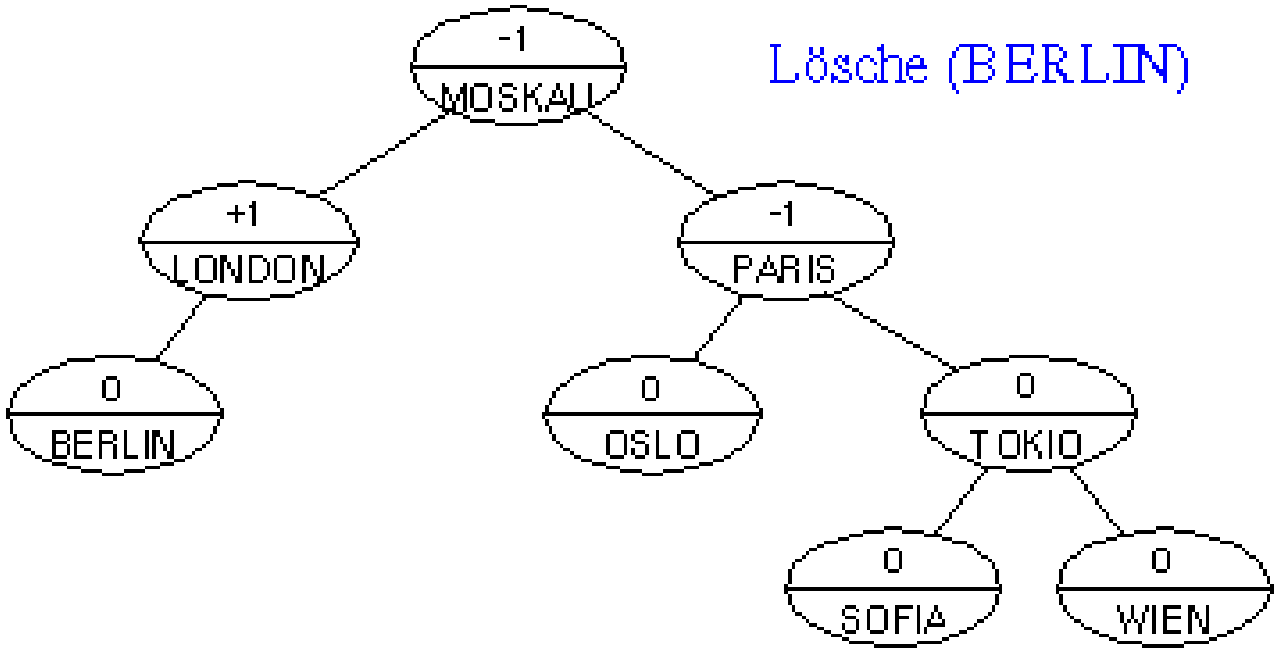



Balancierter Unterbaum
nach Einfügung
Rebalancierter Unterbaum
Rotationstyp LL (Rechtsrotation)
+1
X4
h+2
0
X2
h
B1
B3
h
B5
h
+2
h+3
L
L
h+1
LL
0
h+2
h+1
0
h
Rotationstyp RR (Linksrotation)
-1
-2
R
R
RR
h+1

Löschen in AVL-Bäumen I

Löschen eines Blattes bzw. eines Randknotens (max. 1 Sohn)

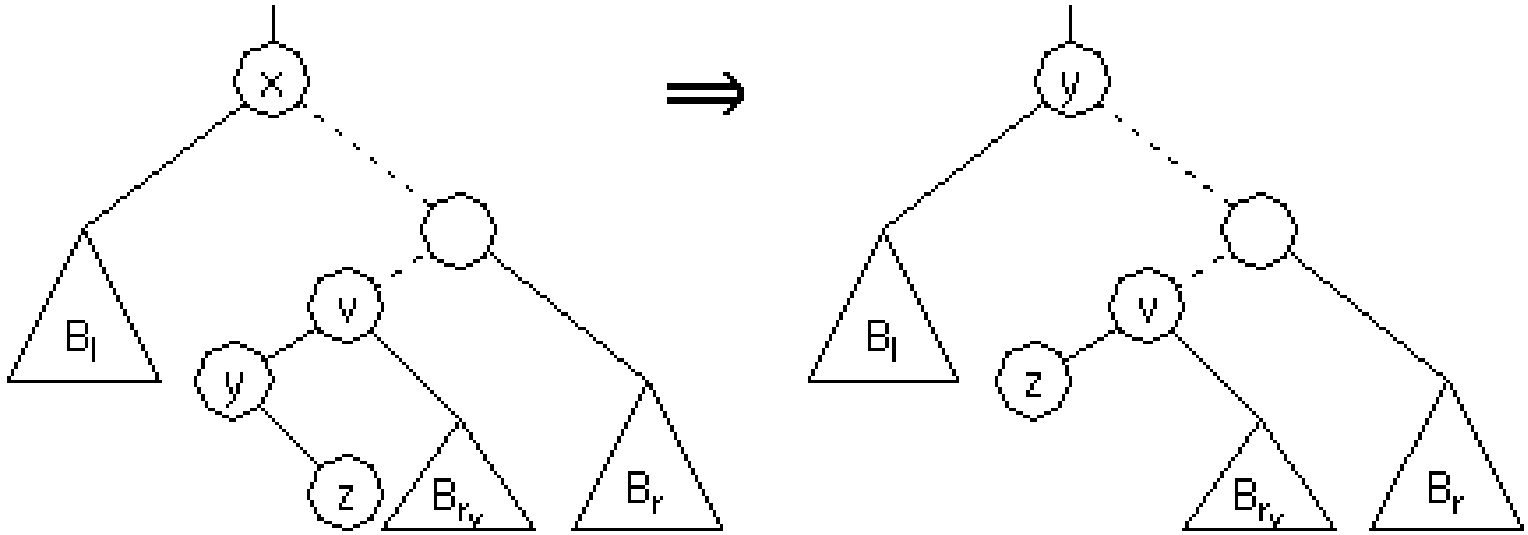
- Höhenreduzierung ändert Balancierungsfaktoren der Vaterknoten- Rebalancierung für UB der Vorgängerknoten mit BF = +/- 2
- ggf. fortgesetzte Rebalancierung (nur möglich für Knoten mit BF = +/- 2 auf dem Weg vom zu löschenden Element zur Wurzel)



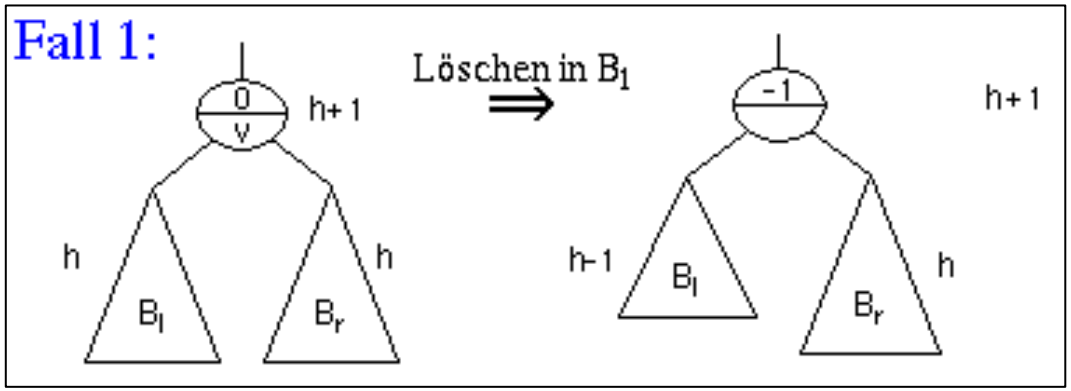
Löschen in AVL-Bäumen II

Löschen eines Knotens (Schlüssel x) mit 2 Söhnen kann auf Löschen für Blatt/Randknoten zurückgeführt werden

- x wird ersetzt durch kleinsten Schlüssel y im rechten Unterbaumbaum von x (oder größten Schlüssel im linken Unterbaum)
- führt zur Änderung des Balancierungsfaktors für v (Höhe des linken Unterbaums von v hat sich um 1 reduziert)

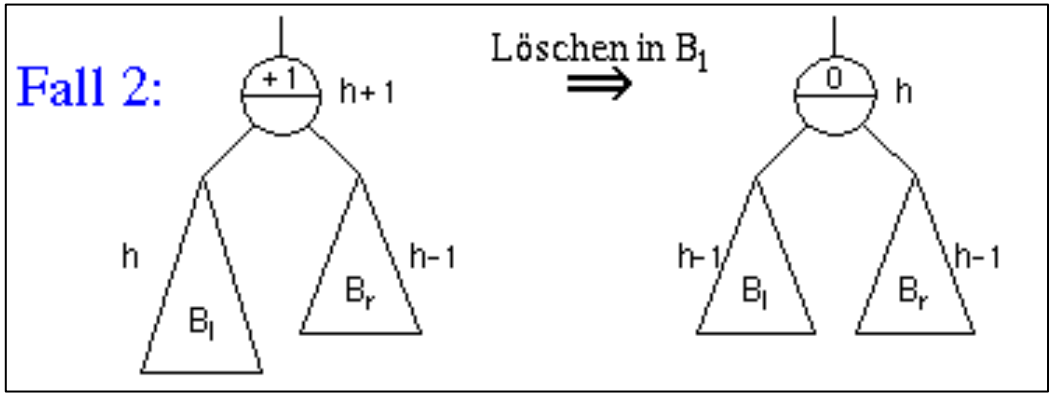


Löschen in AVL-Bäumen III



Bis auf Symmetrie treten nur 3 Fälle auf:

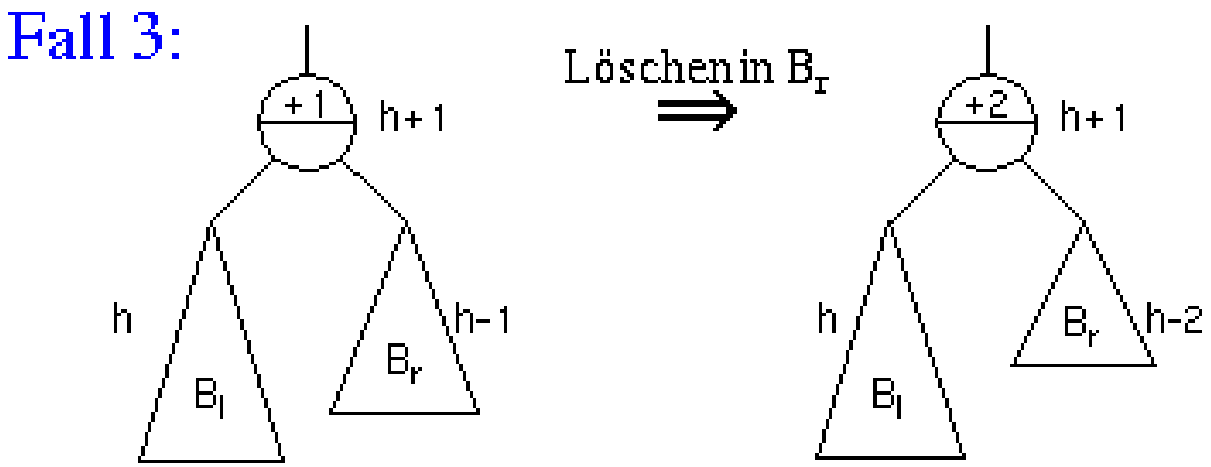
In diesem Fall pflanzt sich Höhenerniedrigung nicht fort, da in der Wurzel das AVL-Kriterium erfüllt bleibt.
-> kein Rebalancieren erforderlich.



Die Höhenerniedrigung von B_l pflanzt
sich hier zur Wurzel hin fort.

Sie kann auf diesem Pfad eine Rebalancierung auslösen.

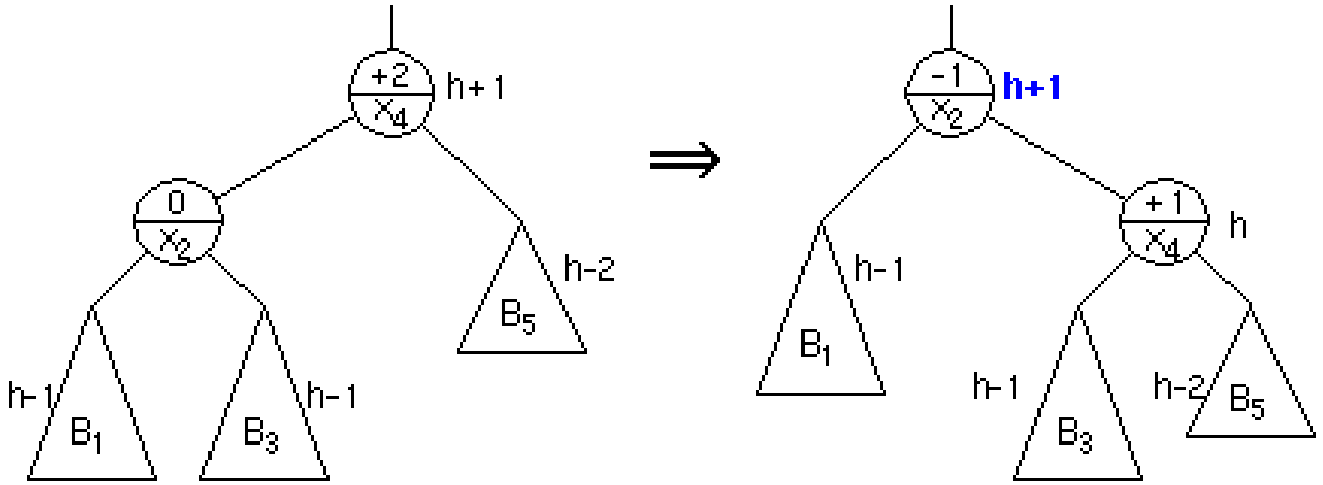
Löschen in AVL-Bäumen IV



Für die Behandlung dieser Situation ist der linke Unterbaum B_l in größerem Detail zu betrachten. Dabei ergeben sich die folgenden 3 Unterfälle

Löschen in AVL-Bäumen V

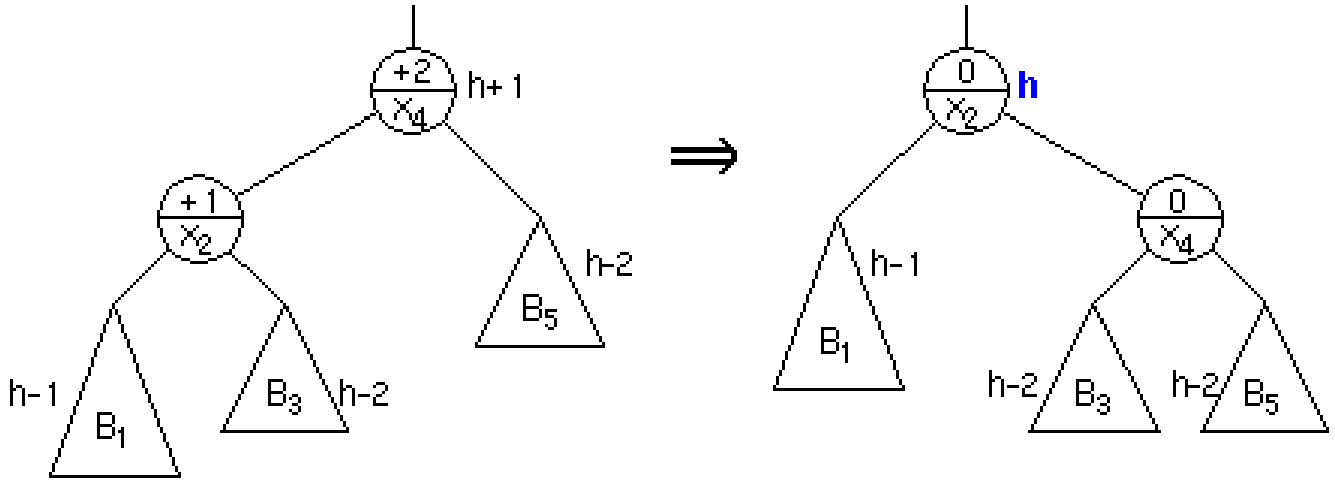
Fall 3a:



- Rechtsrotation führt zur Erfüllung des AVL-Kriteriums
- Unterbaum behält ursprüngliche Höhe
- keine weiteren Rebalancierungen erforderlich

Löschen in AVL-Bäumen VI

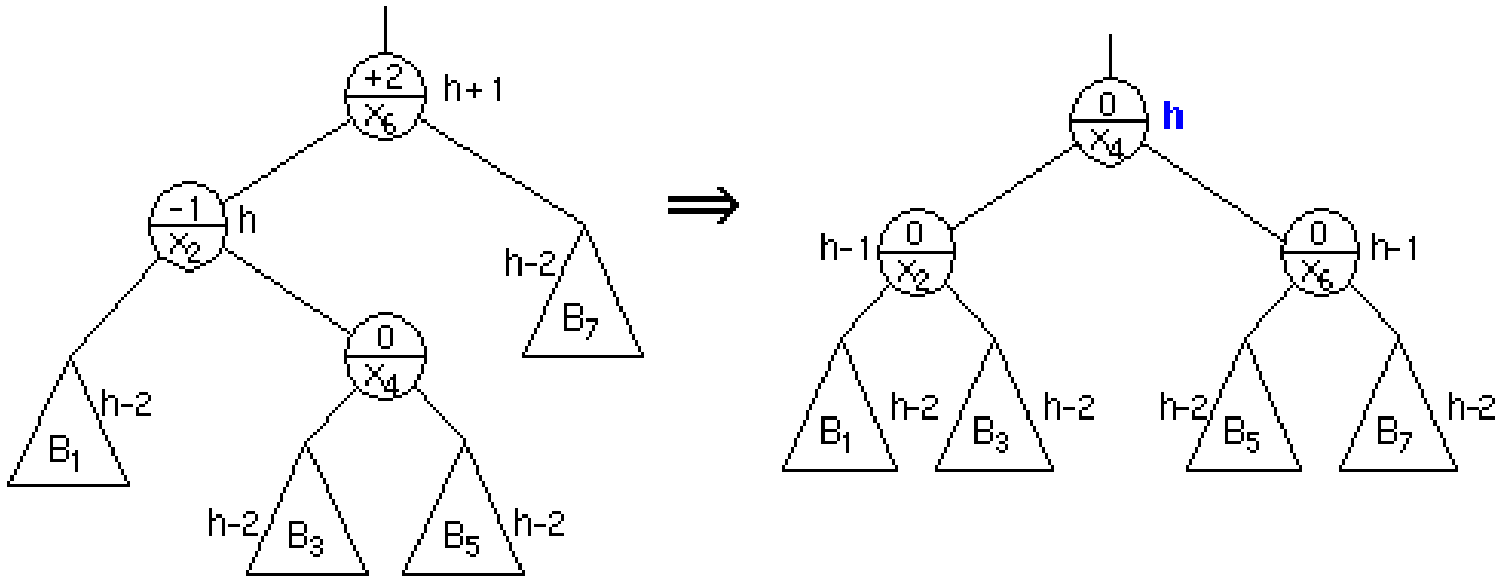
Fall 3b:



- Rechtsrotation reduziert Höhe des gesamten UB von h+1 nach h
- Höhenreduzierung pflanzt sich auf dem Pfad zur Wurzel hin fort und kann zu weiteren Rebalancierungen führen.

Löschen in AVL-Bäumen VII

Fall 3c:



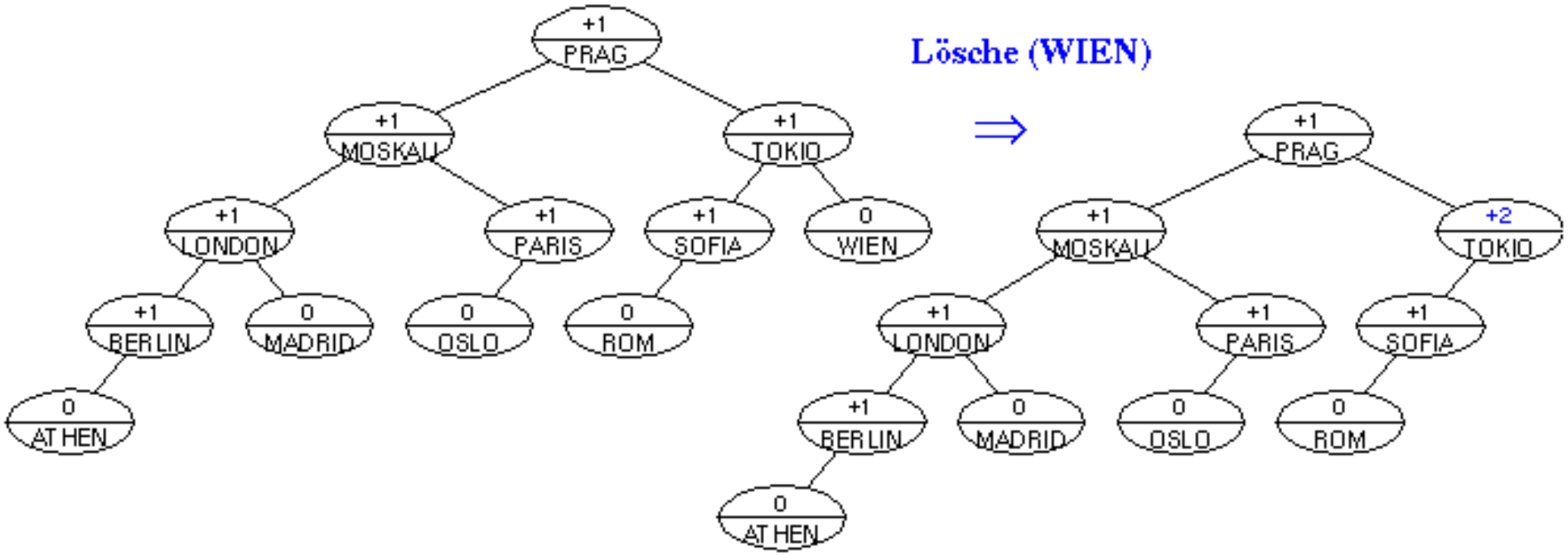
- Doppelrotation (-> Höhenerniedrigung)
- ggf. fortgesetzte Rebalancierungen

Löschen in AVL-Bäumen VIII

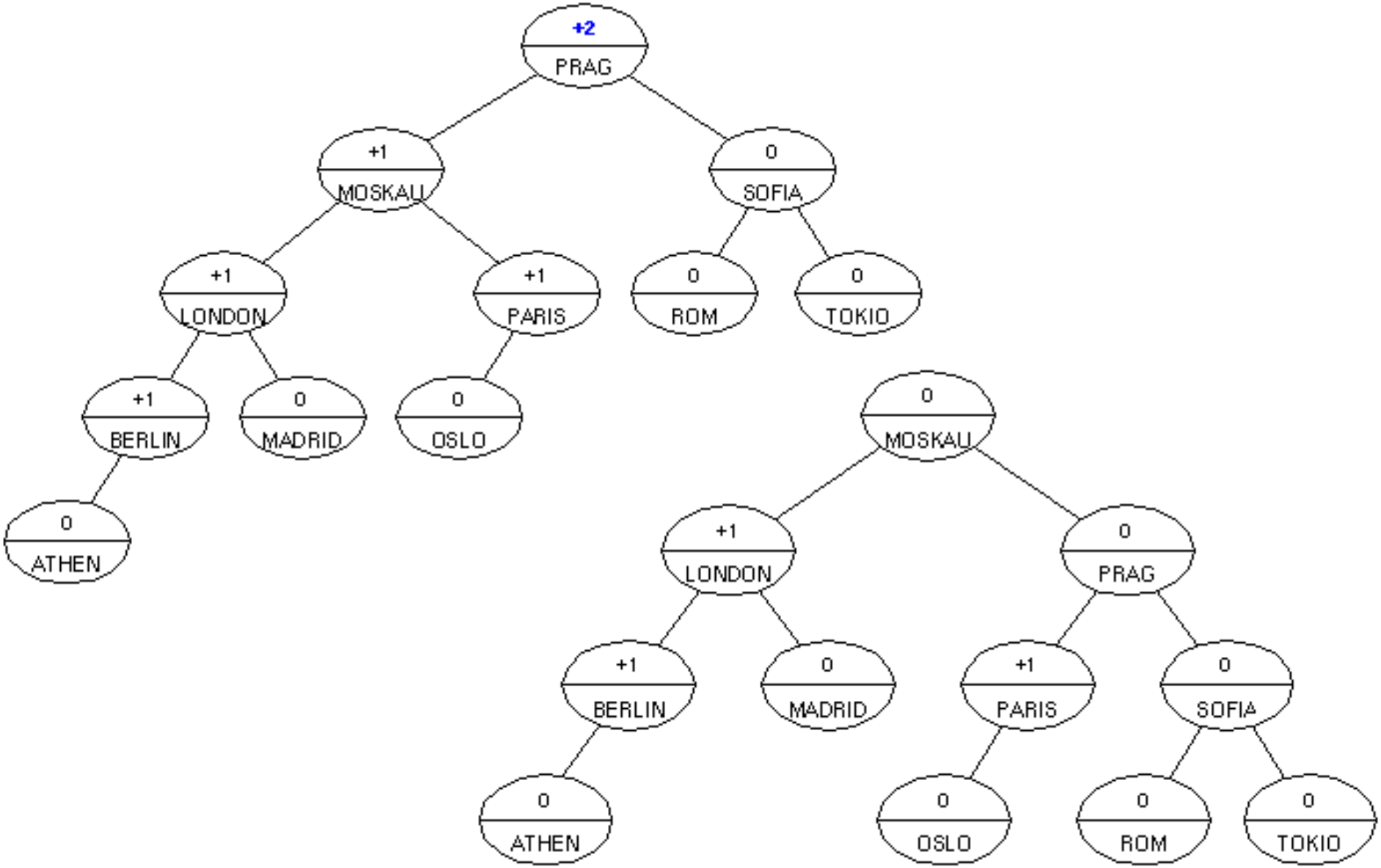
Rebalancierungsschritte beim Löschen:

1. Suche im Löschpfad nähesten Vater mit BF = +/- 2 .
2. Führe ggf. Rotation im gegenüberliegenden Unterbaum dieses Vaters aus.
   Im Gegensatz zum Einfügevorgang kann hier eine Rotation wiederum eine Rebalancierung auf dem Pfad zur Wurzel auslösen, da sie in gewissen Fällen auf eine Höhenerniedrigung des transformierten Unterbaums führt. Die Anzahl der Rebalancierungsschritte ist jedoch durch die Höhe h des Baums begrenzt

Beispiel-Löschvorgang I



Beispiel-Löschvorgang II



Höhe von AVL-Bäumen

Balancierte Bäume wurden als Kompromiss zwischen ausgeglichenen und natürlichen Suchbäumen eingeführt, wobei logarithmischer Suchaufwand im schlechtesten Fall gefordert wurde.

Für die Höhe h_b eines AVL-Baumes mit n Knoten gilt:

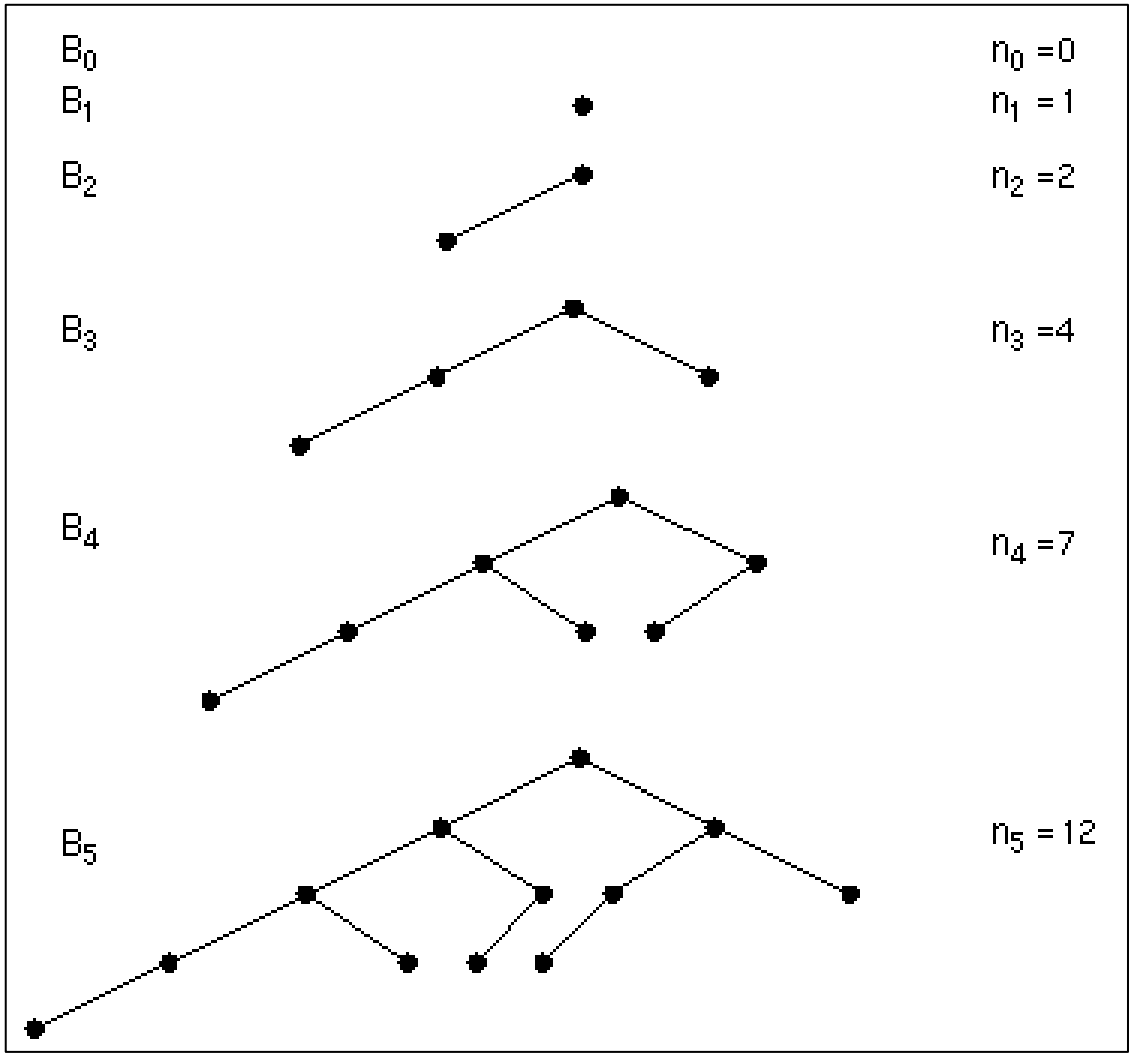
$$\lfloor \log_2(n) \rfloor + 1 < h_b < 1{,}44 \cdot \log_2(n+1)$$

Die obere Schranke lässt sich durch sog. Fibonacci-Bäume, eine Unterklasse der AVL-Bäume, herleiten.

Definition für Fibonacci-Bäume

Konstruktionsvorschrift

- Der leere Baum ist ein Fibonacci-Baum der Höhe 0
- Ein einzelner Knoten ist ein Fibonacci-Baum der Höhe 1
- Sind B_{h-1} und B_{h-2} Fibonacci-Bäume der Höhe h-1 und h-2, so ist $B_h = (B_{h-1},x,B_{h-2})$ ein Fibonacci-Baum der Höhe h
- Keine anderen Bäume sind Fibonacci-Bäume



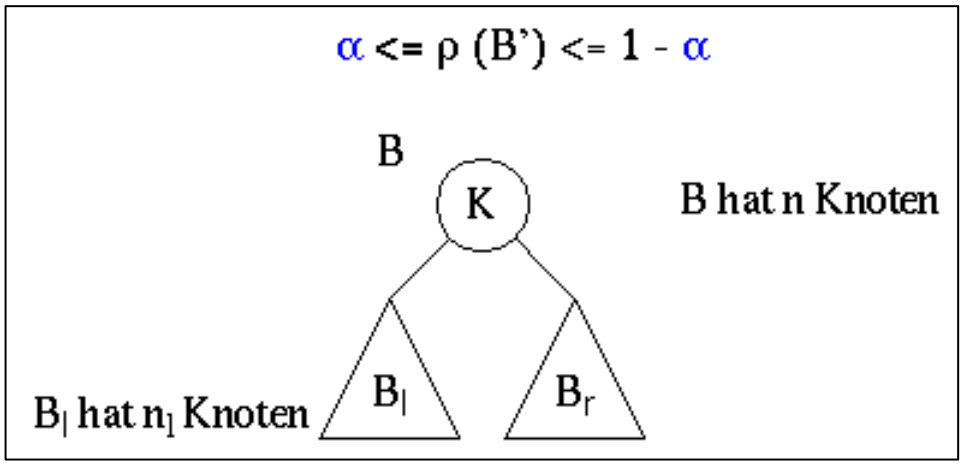
Gewichtsbalancierte Suchbäume I

Gewichtsbalancierte oder BB-Bäume (bounded balance)

Zulässige Abweichung der Struktur vom ausgeglichenen Binärbaum wird als Differenz zwischen der Anzahl der Knoten im rechten und linken Unterbaum festgelegt

Def.: Sei B ein binärer Suchbaum mit linkem Unterbaum B_l und sei n (n_l) die Anzahl der Knoten in B (B_l)

- $p(B) = (n_l+1)/(n+1)$ heißt die Wurzelbalance von B.
- Ein Baum B heißt *gewichtsbalanciert* (BB(α)) oder von *beschränkter Balance* α, wenn für jeden Unterbaum B' von B gilt:



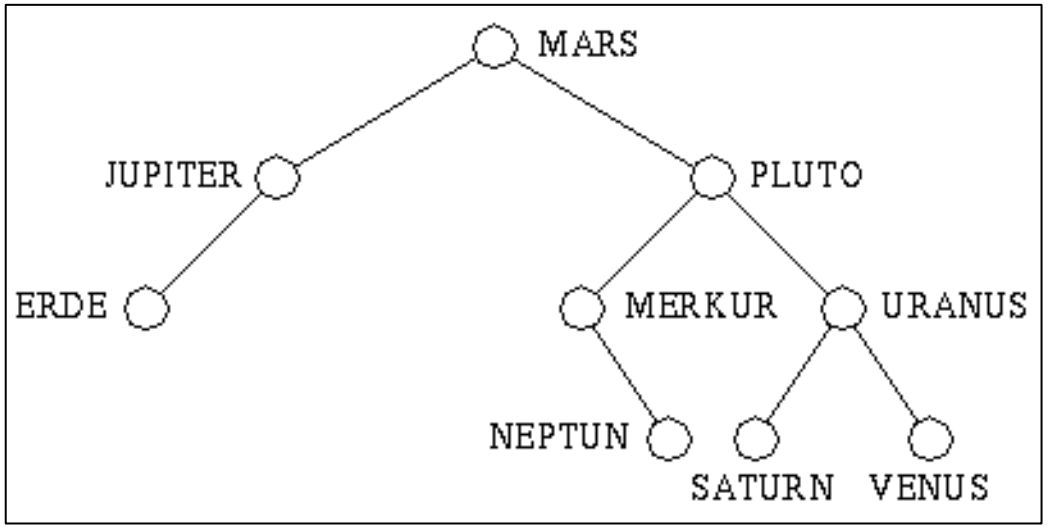
Gewichtsbalancierte Suchbäume II

Parameter als Freiheitsgrad im Baum

- $\alpha = 1/2$: Balancierungskriterium akzeptiert nur vollständige Binärbäume
- $\alpha < 1/2$: Strukturbeschränkung wird zunehmend gelockert

Frage: Welche Auswirkungen hat die Lockerung des Balancierungskriteriums auf die Kosten?

Beispiel: Gewichtsbalancierter Baum in BB(α) für $\alpha = 3 / 10$



Rebalancierung

- ist gewährleistet durch eine Wahl von $\alpha <= 1 - \sqrt{2}/2$
- Einsatz derselben Rotationstypen wie beim AVL-Baum

Kosten für Suche und Aktualisierung: O ($\log_2 n$)

Positionssuche mit balancierten Bäumen I

Balancierte Suchbäume

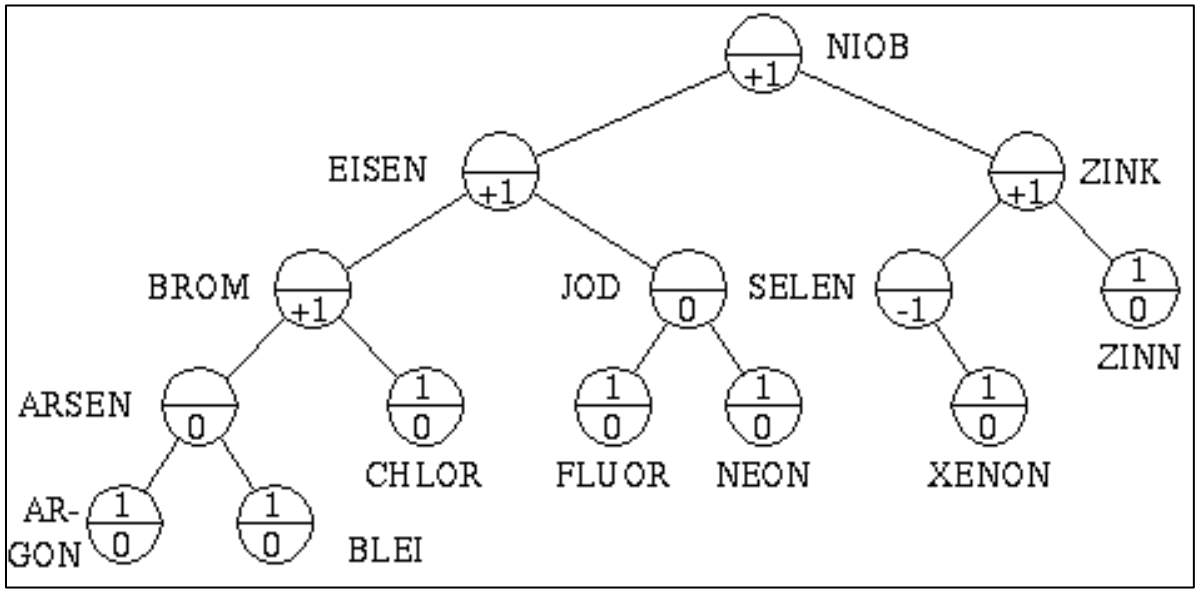
- sind linearen Listen in fast allen Grundoperationen überlegen
- Lösung des Auswahlproblems bzw. Positionssuche (Suche nach k-tem Element der Sortierreihenfolge) kann jedoch noch verbessert werden

Def.: Der *Rang* eines Knotens ist die um 1 erhöhte Anzahl der Knoten seines linken Unterbaums

- Blattknoten haben Rang 1

Verbesserung bei Positionssuche durch Aufnahme des Rangs in jedem Knoten

Beispiel für AVL-Baum:



Positionssuche mit balancierten Bäumen II

Rangzahlen erlauben Bestimmung eines direkten Suchpfads im Baum für Positionssuche nach dem k-ten Element

- Position p := k; beginne Suche am Wurzelknoten
- Wenn Rang r eines Knotens = p gilt: Element gefunden
- falls r > p: Suche im linken UB des Knotens weiter
- falls r < p: Ersetze p := p - r und setze Suche im rechten UB fort.

Die Wartungsoperationen sind etwas komplexer

Änderung im linken Unterbaum erfordert Ranganpassung aller betroffenen Väter bis zur Wurzel