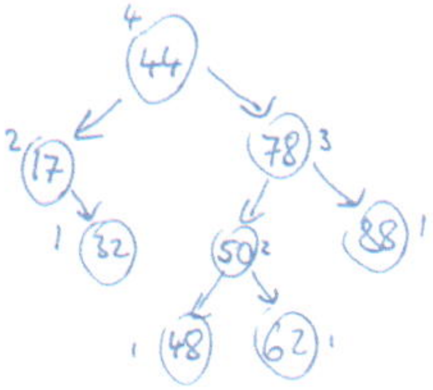
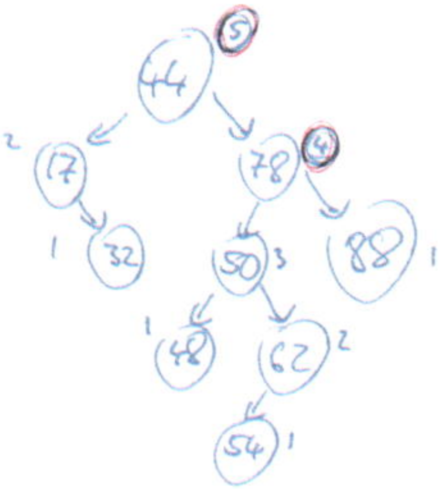
17.01.2012

Idee 5: Beim Einfügen oder Löschen ändert sich die Eigenschaft nur lokal und nur ein bisschen → nimm begrenzte lokale Änderungen zur Reparatur vor!

Beispiel:



Füge 54 ein!

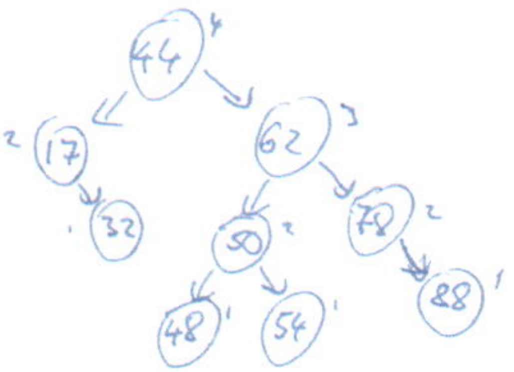


Was ist zu tun?

- Teilbaum der 78 ist nicht höhenbalanciert
- Höhe sollte höchstens 3 sein, damit auch der ganze Baum unter 44 höhenbalanciert ist

→ ~~Verbessel.~~ Betrachte 78, Kind 50, Enkelkind 62
(Kritischer Pfad unter 78)

Neuer Baum:



- höhenbalanciert!
- nur lokale Umsortierung von 78, 62, 50

Vorher: 78 oben, darunter 50, 62
Jetzt: 62 oben, darunter 78, 50

↳ Rotation!

Mehr Details und Fallbetrachtung folgen...

Genauer: Betrachte Einfügen eines Knotens v

(Im Beispiel: 54)

Wenn Baum weiter höhenbalanciert → Ok!

Wenn Baum nicht mehr höhenbalanciert

→ Vorfahre von v hat Gewicht dazubekommen, was zu Unbalanciertheit geführt hat

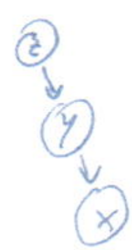
(Im Beispiel sind das 44 und 78)

Sei z der niedrigste unbalancierte Vorfahre von v (Im Beispiel: 78) → Mindestens Höhe 3

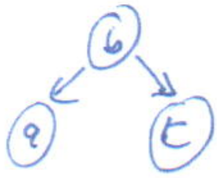
Sei y das Kind von z, das Vorfahre von v ist (im Beispiel: 50); dann muss y zwei höher sein als das andere Kind von z.

Sei x das Kind von y, das im Teilbaum von v liegt. (im Beispiel: 62)

Also: Großvater z
Vater y
Kind x
(eventuell auch x = v möglich)



Gemeinsames Muster:

$a < b < c$ wird zu  !

(Vgl. Fälle !)

Also:

__Algorithmus 4.8__ (Umstrukturierung im Binärbaum)

Restructure (x)

Eingabe: Knoten x eines binären Suchbaumes T, Unterknoten y, Großvaterknoten z

Ausgabe: Binärer Suchbaum T nach Umstrukturierung mit x, y, z.

1 Sei (a, b, c) die Größensortierung der Knoten x, y, z; seien (T_0, T_1, T_2, T_3) die Größensortierung der vier Teilbäume unter x, y, z, die nicht Wurzeln x, y, z haben.

2 Ersetze den Teilbaum mit Wurzel z durch einen neuen Teilbaum mit Wurzel b.

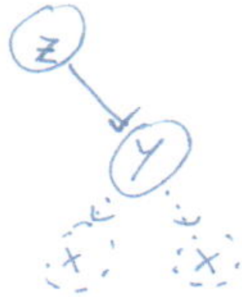
3 Setze a als linkes Kind von b, T_0 und T_1 als linken und rechten Teilbaum unter a.

Wie können x, y, z zueinander stehen?

Beobachtung: Wenn $z < y$,

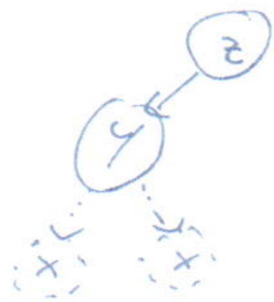
dann auch $z < x$

(Suchbaumeigenschaft!)



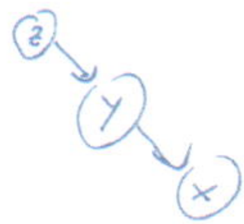
Wenn $z > y$,

dann auch $z > x$

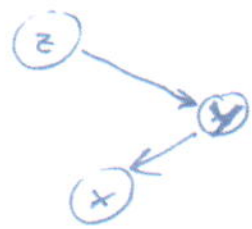


Es bleiben die Möglichkeiten:

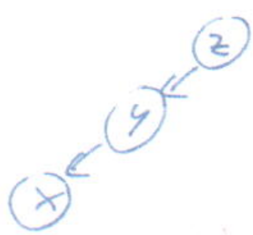
(I) $z < y < x$



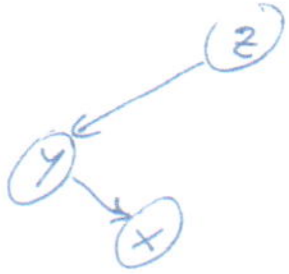
(II) $z < x < y$



(III) $x < y < z$

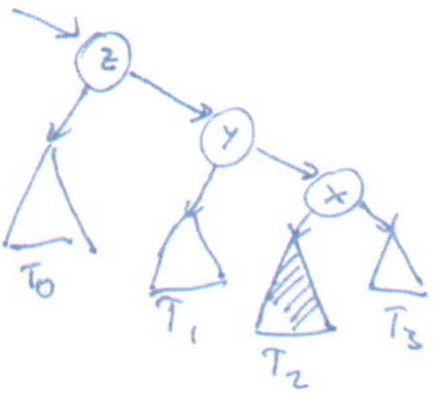


(IV) $y < x < z$

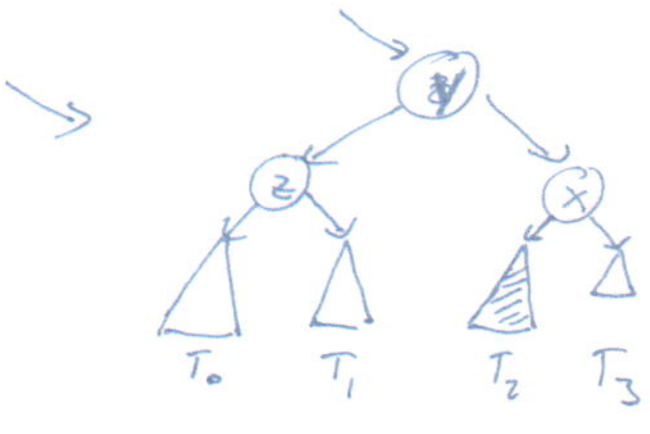


Jetzt betrachten wir folgende Reparaturoperationen:

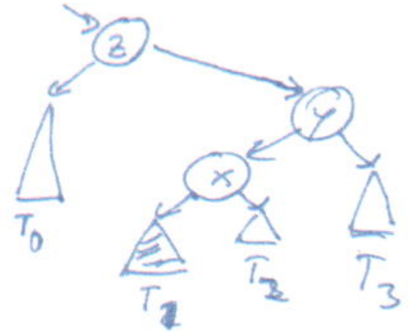
(I)



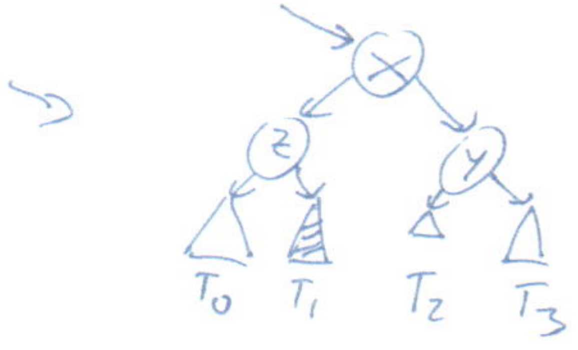
T_2 enthält v, geht zwei Level tiefer als T_0, ein Level tiefer als T_1, T_3



(II)

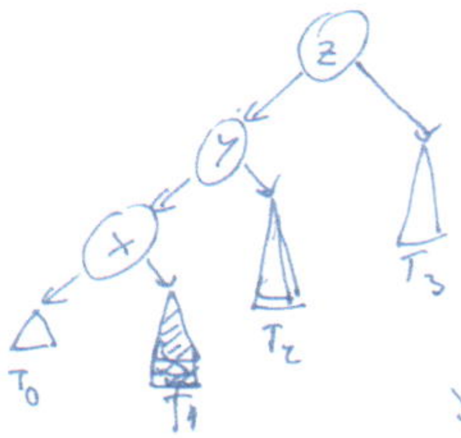


(T_1 zwei Level tiefer als T_0!)

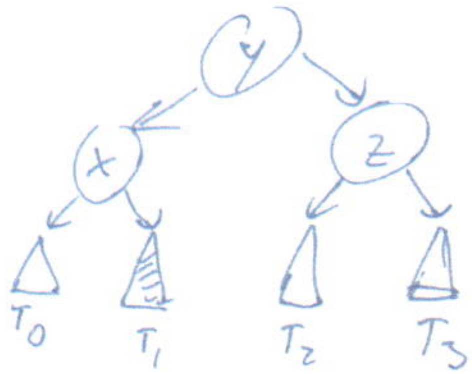


96c

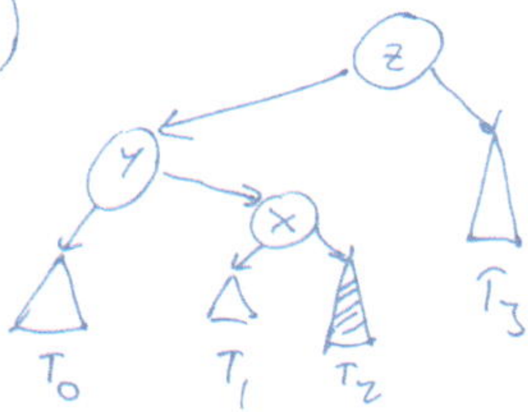
III



(T_1 zwei Level tiefer als T_3, eins tiefer als T_0, T_2)



IV



(T_2 zwei Level tiefer als T_3, eins tiefer als T_0, T_1)