

Föreläsning 5

Sökträd: AVL-träd, (2,3)-träd

TDDC91, TDDE22, 725G97: DALG

Utskriftsversion av föreläsning i *Datastrukturer och algoritmer* 18 september 2018

Magnus Nielsen, IDA, Linköpings universitet

5.1

Innehåll

Introduktion

`find`, `insert` och `remove` i ett binärt sökträd tar $O(h)$ tid, där h är höjden av trädet.

Håll sökträdet balanserat!!

5.2

Sökträd är coolt!

Sökträd får till och med vara med på TV!

- CSI
- Criminal Minds
- Missing (CAN)

5.3

Riktigt coolt!

ACT FOUR

FADE IN:

48 INT. FBI HQ - NIGHT

48

Antonio is at THE COMPUTER as Jess explains herself to Nicole and Pollock. The CONFERENCE TABLE is covered with OPEN REFERENCE BOOKS, TOURIST GUIDES, MAPS and REAMS OF PRINTOUTS.

JESS

It was the red door again.

POLLOCK

I thought the red door was the storage container.

JESS

But it wasn't red anymore. It was black.

ANTONIO

So red turning to black means... what?

POLLOCK

Budget deficits? Red ink, black ink?

NICOLE

Yes. I'm sure that's what it is. But maybe we should come up with a couple other options, just in case.

Antonio refers to his COMPUTER SCREEN, which is filled with mathematical equations.

ANTONIO

It could be an algorithm from a binary search tree. A **red-black tree** tracks every simple path from a node to a descendant leaf with the same number of black nodes.

JESS

Does that help you with girls?

5.4

1 AVL-träd

AVL-träd

- Självbalanserande BST/höjdbalanserat BST
- AVL = Adelson-Velskii och Landis, 1962
- Idén: Håll reda på balansinformation i varje nod
- **AVL-egenskapen**För varje intern nod v i T skiljer sig höjden av barnen till v med högst 1 ... eller alternativt... För varje intern nod v i T gäller att $balans(v) \in \{-1, 0, 1\}$, där

$$balans(v) = \text{height}(\text{leftChild}(v)) - \text{height}(\text{rightChild}(v))$$

5.5

Maximal höjd av AVL-träd

Proposition 1. Höjden av ett AVL-träd som lagrar n poster är $O(\log n)$.

Vilket får som följd att...

Proposition 2. Vi kan göra **find**, **insert** och **remove** i ett AVL-träd i tid $O(\log n)$ medan vi bevarar AVL-egenskapen.

5.6

Borttagning i ett AVL-träd

- **find** och **remove** som i ett vanligt binärt sökträd
- Uppdatera balansinformationen på väg tillbaka upp till roten
- Om för obalanserat: Strukturera om... men...
 - När vi återställer balansen på ett ställe kan det uppstå obalans på ett annat
 - Måste upprepa balanseringen (eller kontroll av balansen) till dess vi når roten
 - Högst $O(\log n)$ ombalanseringar

5.7

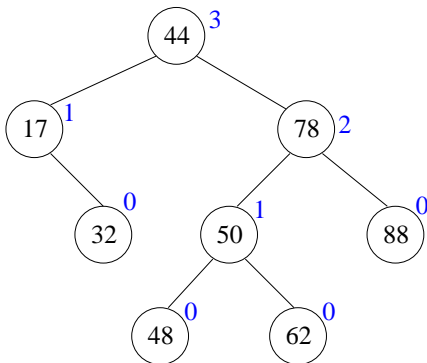
Rotationer

- Balansering måste göras både vid insert och remove
- Fyra möjliga rotationer:
 - "Höger" eller "vänster"
 - Enkel- eller dubbelrotation

5.8

Exempel: ett AVL-träd

5.9

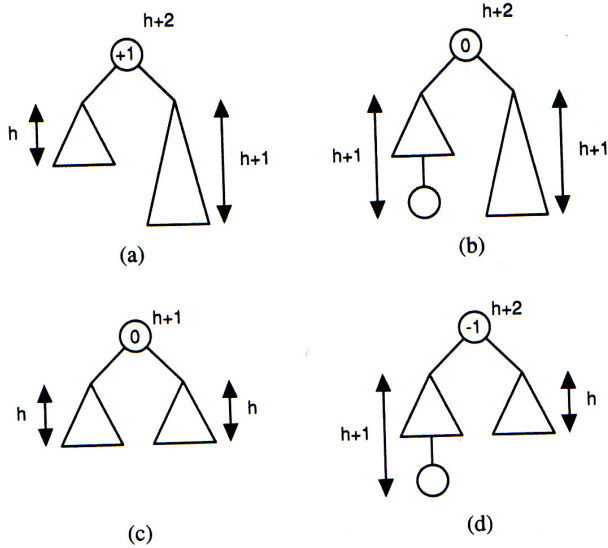


Insättning i ett AVL-träd

- Den nya noden gör att trädhöjden förändras och att trädet måste höjdbalanseras.
 - Man kan hålla reda på delträdens höjd på olika sätt:
 - * Lagra höjden explicit i varje nod
 - * Lagra balansfaktorn för noden
- Förändringen brukar beskrivas som en höger- eller vänsterrotation av ett delträd.
- Det räcker med en rotation för att få trädet i balans igen.

5.10

Insättning i AVL-träd (enkla fall)



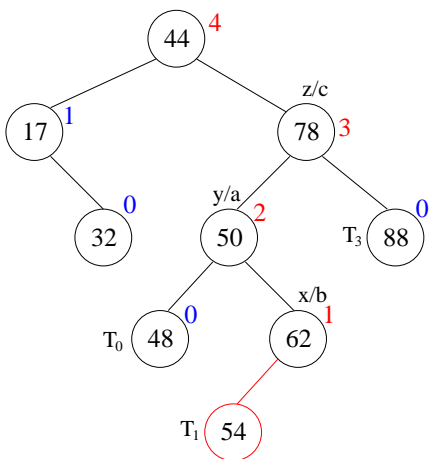
5.11

Insättningsalgoritm

- Starta från den nya noden och leta uppåt tills man hittar en nod x s.a. dess "grandparent" z är obalanserad. Markera x 's förälder med y .
- Gör en rekonstruering av trädets så här:
 - Döp om x, y, z till a, b, c baserat på deras inorder-ordning.
 - Låt T_0, T_1, T_2, T_3 vara en uppräknig i inorder av delträden till x, y och z . (Inget av delträden får ha x, y eller z som rot.)
 - z byts mot b , dess barn är nu a och c .
 - T_0 och T_1 är barn till a och T_2 och T_3 är barn till c .

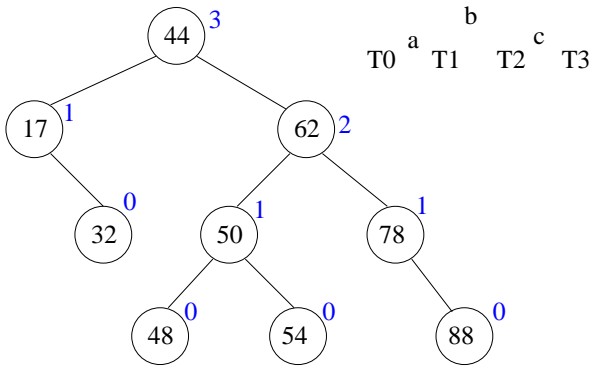
5.12

Exempel insättning i ett AVL-träd

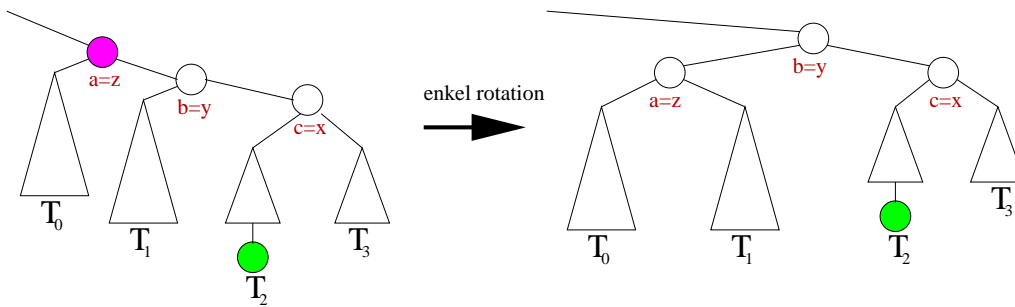


5.13

Exempel: insättning i ett AVL-träd

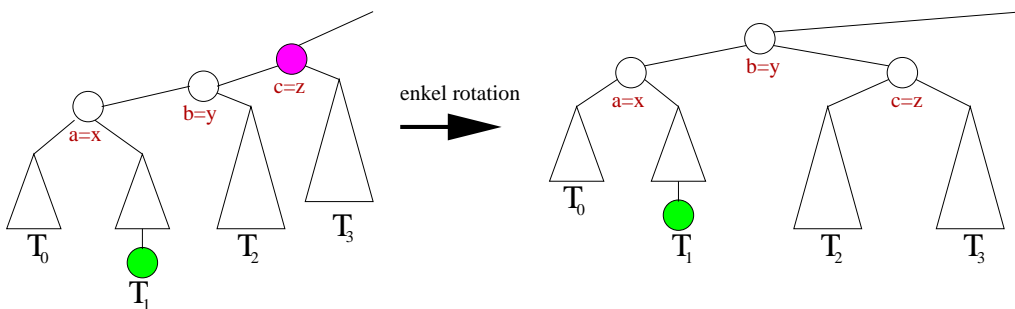


Fyra olika rotationer



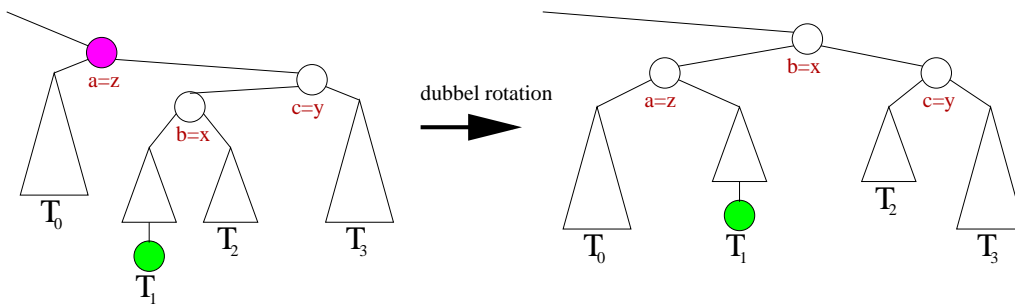
Om $b = y$ kallas det en enkel rotation. "Rotera upp y över z "

Fyra olika rotationer



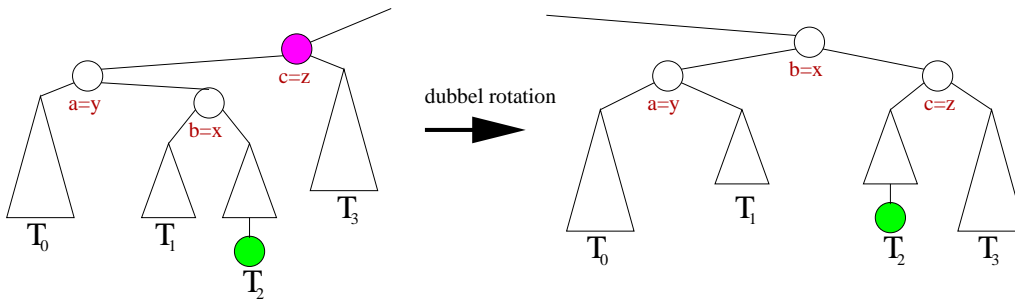
Om $b = y$ kallas det en enkel rotation. "Rotera upp y över z "

Fyra olika rotationer



Om $b = x$ kallas det en dubbel rotation. "Rotera upp x över y och sedan över z "

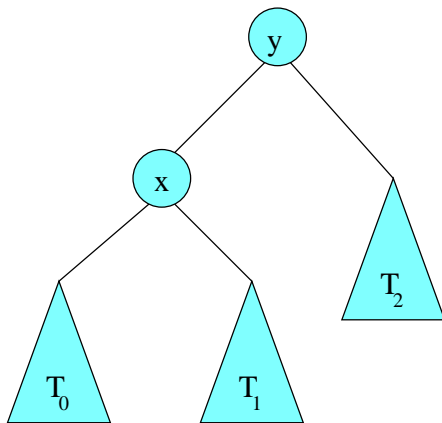
Fyra olika rotationer



Om $b = x$ kallas det en dubbel rotation. "Rotera upp x över y och sedan över z "

5.18

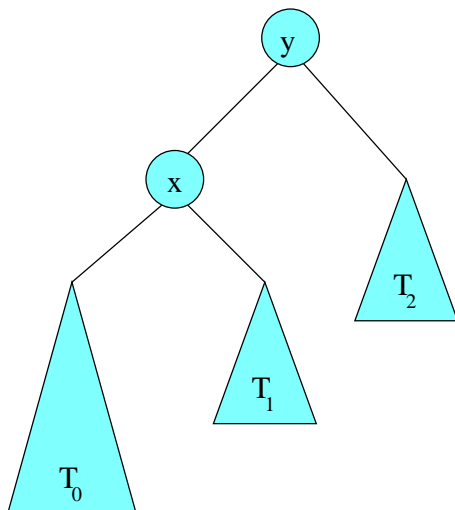
Ett annat sätt att beskriva det på



Antag att vi har balans...

5.19

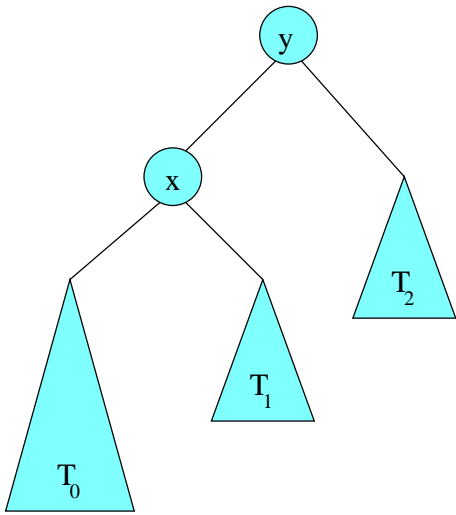
Ett annat sätt att beskriva det på



... och sedan stoppar in något som sabbar den

5.20

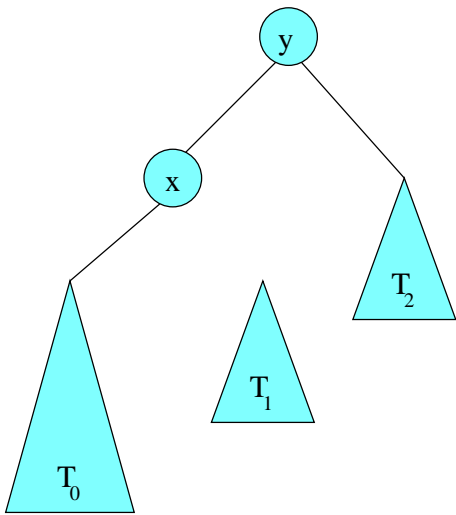
Ett annat sätt att beskriva det på



Gör en enkel rotation

5.21

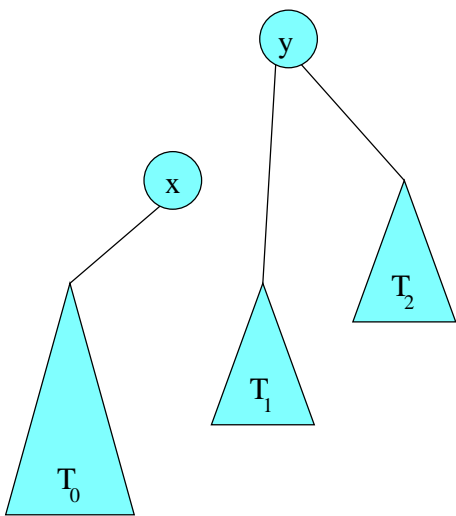
Ett annat sätt att beskriva det på



Gör en enkel rotation

5.22

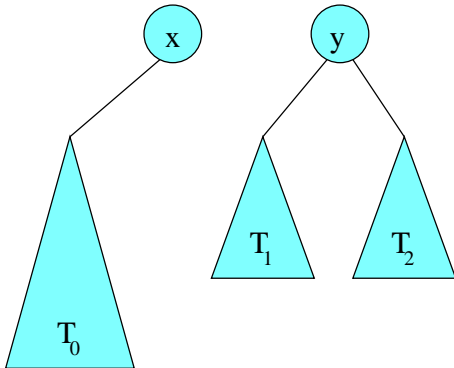
Ett annat sätt att beskriva det på



Gör en enkel rotation

5.23

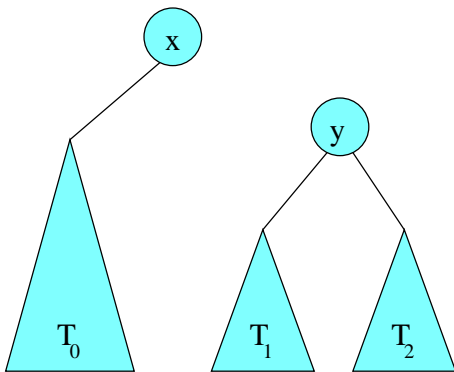
Ett annat sätt att beskriva det på



Gör en enkel rotation

5.24

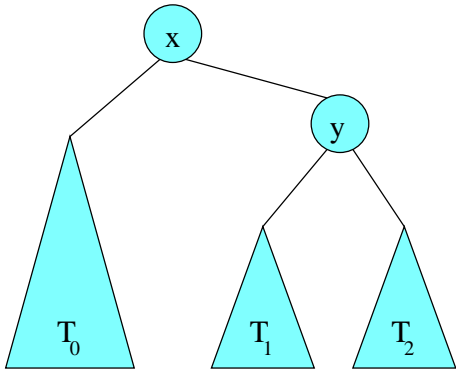
Ett annat sätt att beskriva det på



Gör en enkel rotation

5.25

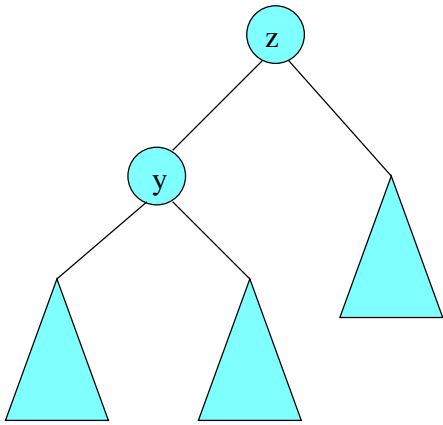
Ett annat sätt att beskriva det på



Klart!

5.26

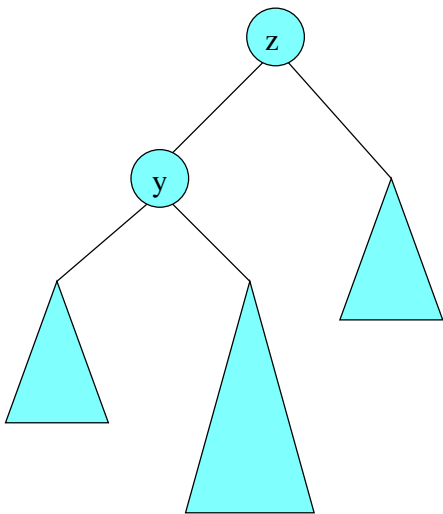
Ett annat sätt att beskriva det på



Ett nytt exempel...

5.27

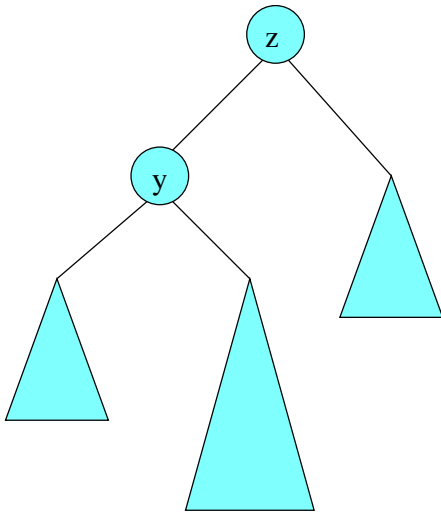
Ett annat sätt att beskriva det på



... den här gången stoppar vi in något på ett annat ställe

5.28

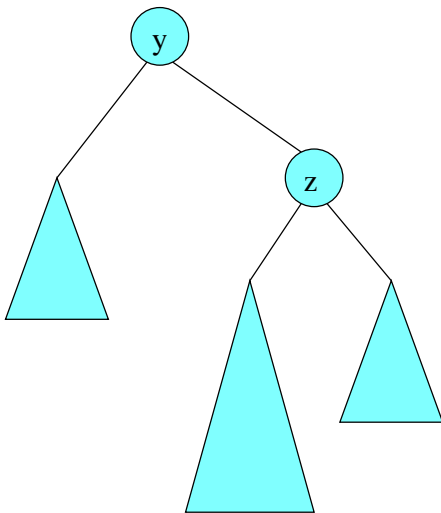
Ett annat sätt att beskriva det på



Prova en enkel rotation igen...

5.29

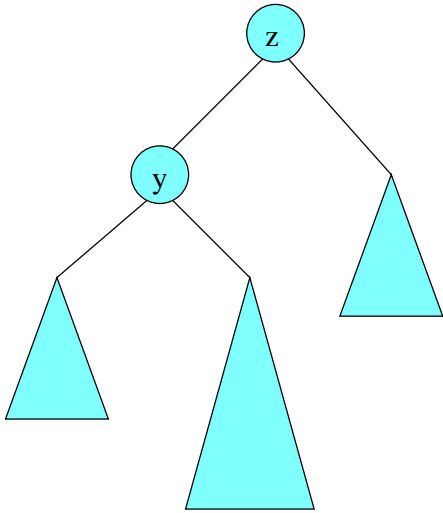
Ett annat sätt att beskriva det på



... hmm, vi har inte fått balans

5.30

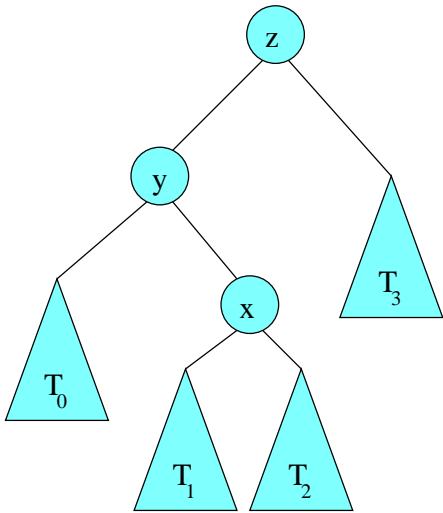
Ett annat sätt att beskriva det på



Börja om från början... och titta på strukturen i y

5.31

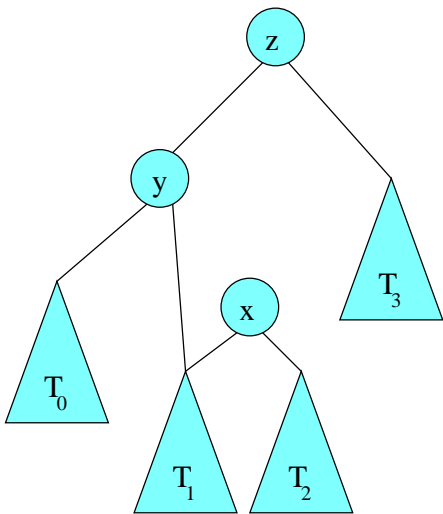
Ett annat sätt att beskriva det på



Vi får lov att göra en dubbel rotation

5.32

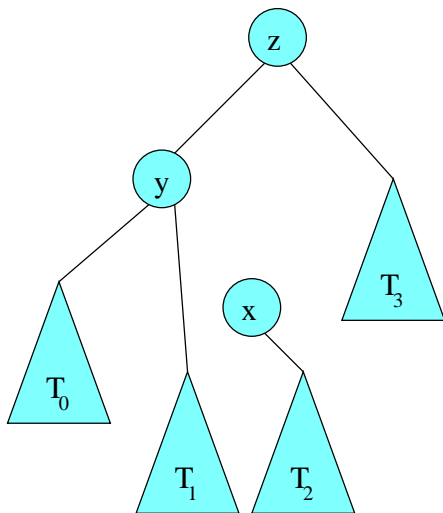
Ett annat sätt att beskriva det på



Vi får lov att göra en dubbel rotation

5.33

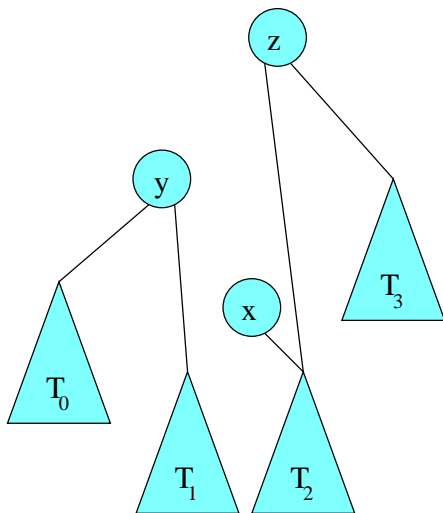
Ett annat sätt att beskriva det på



Vi får lov att göra en dubbel rotation

5.34

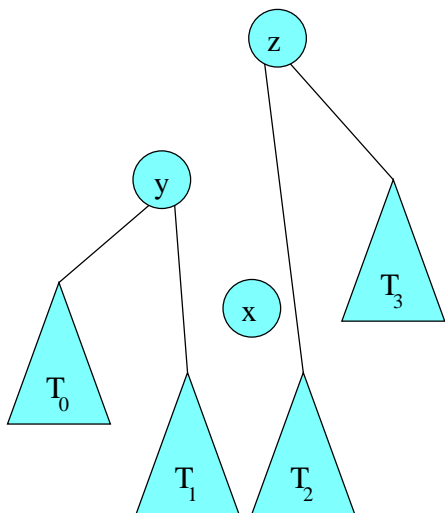
Ett annat sätt att beskriva det på



Vi får lov att göra en dubbel rotation

5.35

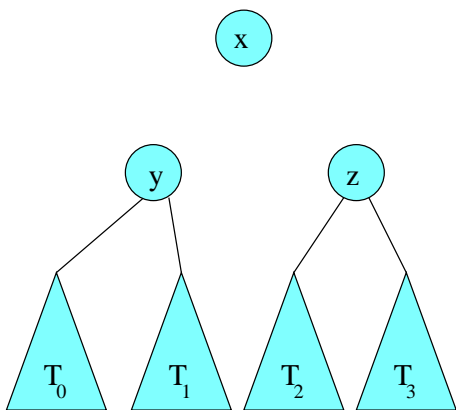
Ett annat sätt att beskriva det på



Vi får lov att göra en dubbel rotation

5.36

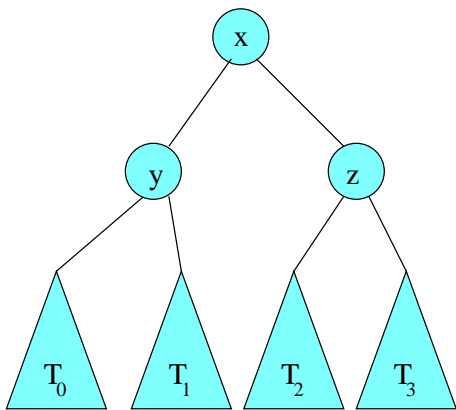
Ett annat sätt att beskriva det på



Vi får lov att göra en dubbel rotation

5.37

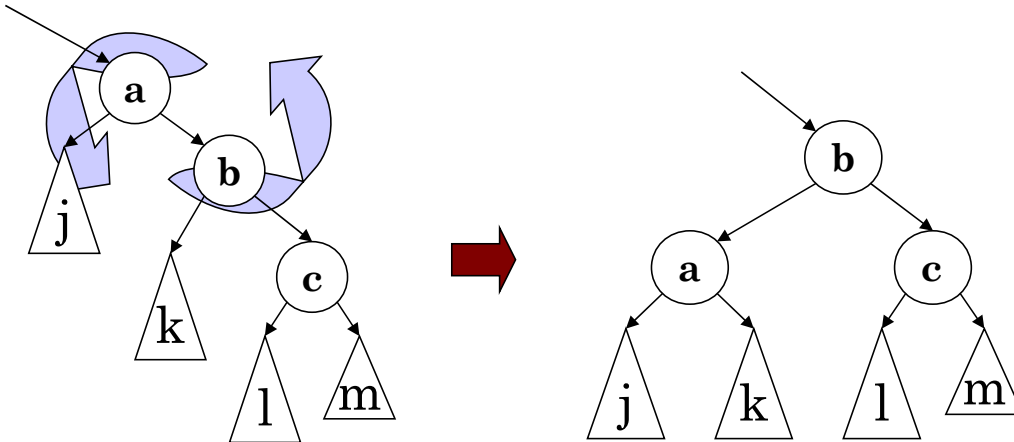
Ett annat sätt att beskriva det på



Trenodsrekonstruering = rotationer...

Vissa författare använder vänster- och högerrotationer: Enkel vänsterrotation:

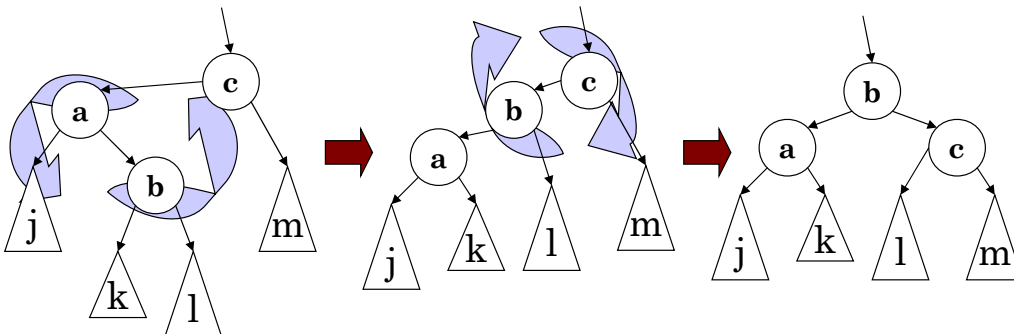
- vänstra delen av delträdet (a och j) sänks ner
- vi har "roterat (upp) b över a "



Dubbla rotationer...

Två rotationer behövs när noderna som ska balanseras om är placerade i ett sicksackmönster.

- Roterar upp b över a
- Roterar upp b över c



Borttagning i ett AVL-träd

- **find** och **remove** som i ett vanligt binärt sökträd
- Uppdatera balansinformationen på väg tillbaka upp till roten
- Om för obalanserat: Strukturera om ... men...
 - När vi återställer balansen på ett ställe kan det uppstå obalans på ett annat
 - Måste upprepa balanseringen (eller kontroll av balansen) till dess vi når roten
 - Högst $O(\log n)$ ombalanseringar

2 (2,3)-träd

Ny approach: släpp på något av kraven

- AVL-träd: *binärt* träd, accepterar viss (liten) obalans...
- Kom ihåg: **Fullt binärt träd**: icke-tomt; graden är antingen 0 eller 2 för varje nod **Perfekt binärt träd**: fullt, alla löv har samma djup
- Kan vi bygga och underhålla ett perfekt träd (om vi struntar i "binärt")? Då skulle vi alltid känna till söktiden i värsta fall exakt!

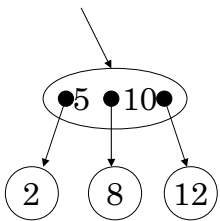
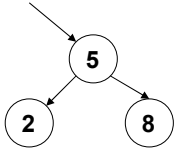
(2,3)-träd

Förut:

- Ett ”pivotelement”
- Om större letar vi till höger
- Om mindre letar vi till vänster

Nu:

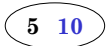
- Tillåt flera (nämligen 1-2) ”pivotelement”
- Antalet barn till en intern nod är antalet pivotelement + 1 (dvs 2-3)



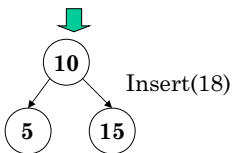
Insättning i ett (a,b)-träd med a = 2 och b = 3



Insert(10)

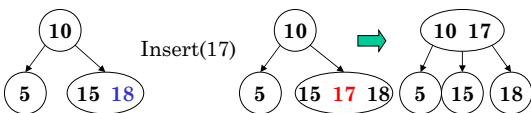


Insert(15)



- Så länge det finns plats i barnet vi hittar, lägg till elementet i det barnet...

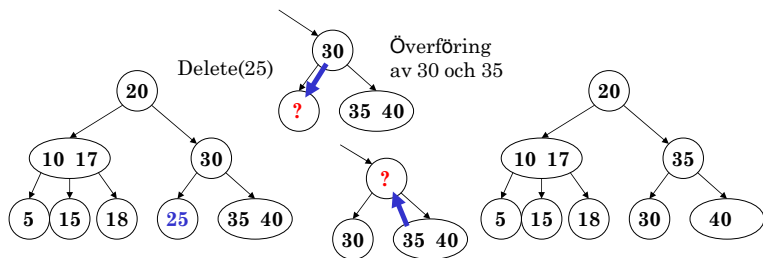
- Om fullt, dela upp och tryck det utvalda pivotelementet uppåt. . . . detta kan hända upprepade gånger



Borttagning i (2,3)-träd

Tre fall:

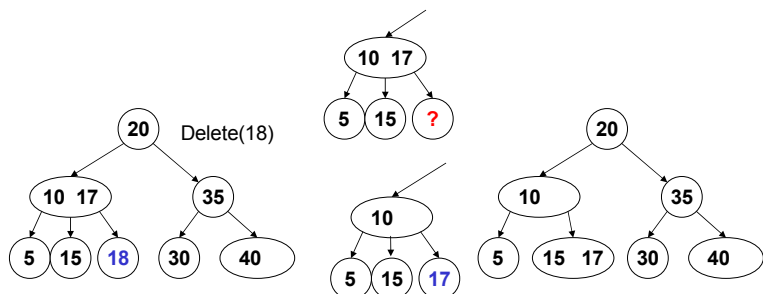
- Inga villkor bryts genom borttagning
- Ett löv tas bort (blir tomt) För då över någon annan nyckel till det lövet, ... ok om vi har syskon med 2+ element



5.45

Borttagning i (2,3)-träd

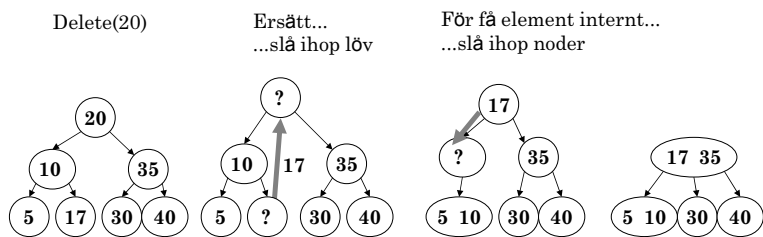
- Om ett löv tas bort (blir tomt)
- För då över någon annan nyckel till det lövet, eller
- Slå ihop det med en granne



5.46

Borttagning i (2,3)-träd

- En intern nod blir tom Roten: ersätt med föregångare eller efterföljare i inorder Reparera sedan inkonsistenser med lämpliga ihopslagningar och överföringar. . .



5.47