

# Red-black trees



# Motivasjon

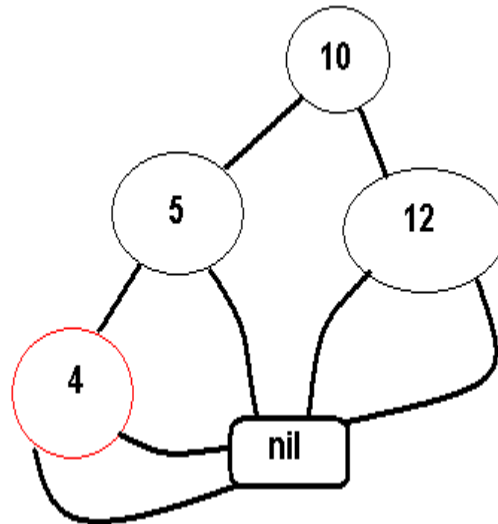
- Alle vanlige operasjoner på binærtrær har kompleksitet avhengig av høyden på treet.
- Høyden kan bli  $n$  i værste tilfelle
- Kan vi sikre at den alltid er  $O(\lg(n))$ ?

# Egenskaper ved red-black-trees

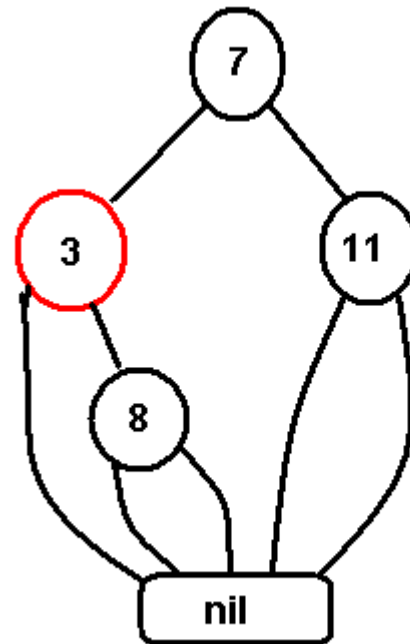
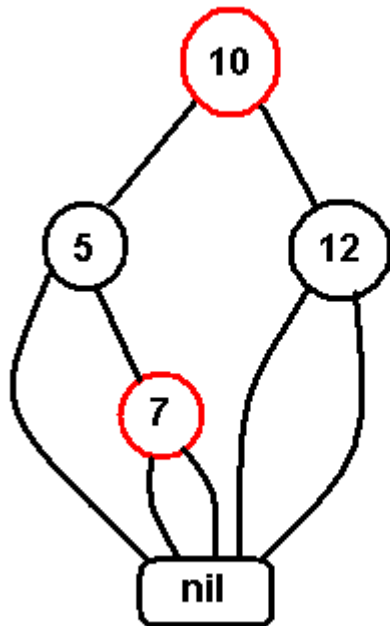
- Enhver node er enten rød eller svart.
- Roten er svart.
- Enhver løvnode(nil) er svart.
- Hvis en node er rød, så er alle dens barn svarte.
- For hver node er det slik at alle veier fra denne noden til en etterfølger som er en løvnode inneholder like mange svarte noder. (Black-height)

# Eksempel

- Er dette et lovlig red-black-tre?



## Eksempel 2



# Hva får vi ut av disse egenskapene?

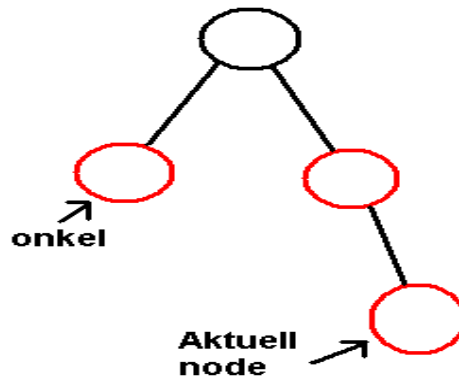
- Bevis i boken for at et red-black-tre med  $n$  noder har en maksimal høyde på  $2 \lg(n+1)$  (induksjon)
- Vi ser på det intuitivt.
  - Hvor skjevt kan vi få det?

# Hvordan opprettholde egenskapene?

- Rotasjoner og omfarging av noder.
- Vi ser på eksempel på left-rotate

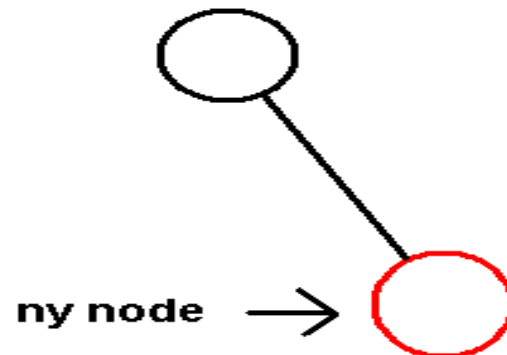
# Hvordan foregår innsetting (overfladisk)

- Noe kult? `java.util.TreeMap`
- Setter inn som i et vanlig binærtre først
- Fikser opp etterpå.
  - Dette skjer ved hjelp av rotasjonene
  - Hvordan fargene skifter oppover osv er avhengig av fargen på onkelen til den aktuelle noden



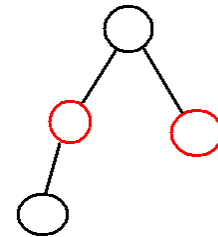
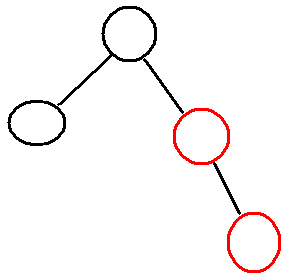
# Hva hvis?

- En node som settes inn farges rød.
- Hva skjer om faren til denne noden er svart?



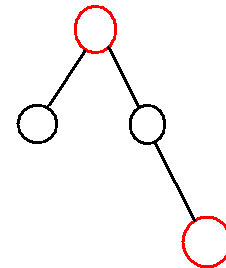
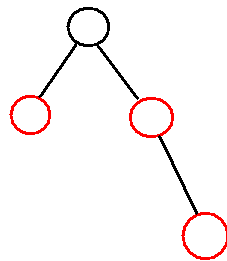
# Hva hvis? (forts)

- Hvis faren er rød har vi problemer.
- Men hva hvis onkelen er svart? Da kan vi rotere og er kvitt problemet.



# Hva hvis? (forts)

- Hvis onkelen er rød må vi forskyve problemet videre oppover.



# Eksempel

- <http://reptar.uta.edu/NOTES5311/REDBLACK/RedBlack.html>
- Anbefales å leke med denne mens man leser dette i boken.