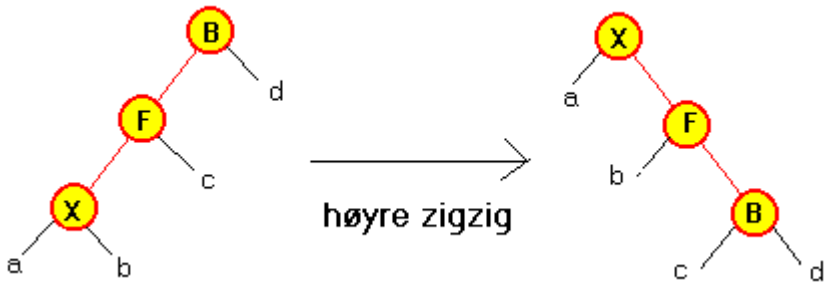
Algoritmer og datastrukturer
Kapittel 9 - Delkapittel 9.4

9.4 Splay-trær

9.4.1 Splay-rotasjoner

Et splay-tre er et sortert binætre der treet **restruktureres** på en bestemt måte etter hver operasjon (d.v. innsetting, søking eller sletting). Idéen er at restruktureringene skal sørge for at verdier som etterspørres ofte ligger høyt oppe (nær roten) i treet og at verdier som etterspørres sjelden ligger langt nede (langt fra roten) i treet.

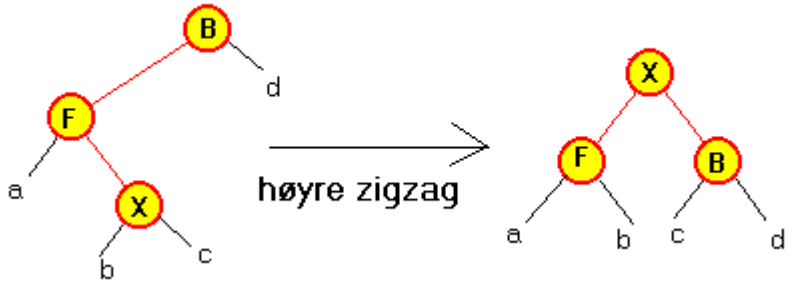
En restrukturering gjennomføres som en serie **rotasjoner** og kalles en "splay". Det er flg. seks typer rotasjoner som er aktuelle:



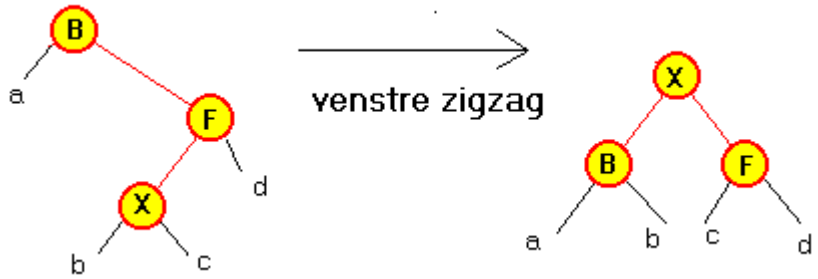

```
Node<T> høyreZigZig(Node<T> p) // to enkle høyrerotasjoner
{
  Node<T> q = p.venstre;
  p.venstre = q.høyre;
  q.høyre = p;

  p = q.venstre;
  q.venstre = p.høyre;
  p.høyre = q;

  return p;
}
```


```
Node<T> venstreZigZig(Node<T> p) // to enkle venstrerotasjoner
{
  Node<T> q = p.høyre;
  p.høyre = q.venstre;
```

```
  q.venstre = p;

  p = q.høyre;
  q.høyre = p.venstre;
  p.venstre = q;

  return p;
}
```




```
Node<T> høyreZigZag(Node<T> p) // dobbel høyrerotasjon
{
  Node<T> q = p.venstre;
  Node<T> r = q.høyre;

  q.høyre = r.venstre;
  r.venstre = q;
  p.venstre = r.høyre;
  r.høyre = p;
  return r;
}
```




```
Node<T> venstreZigZag(Node<T> p) // dobbel venstrerotasjon
{
  Node<T> q = p.høyre;
  Node<T> r = q.venstre;

  q.venstre = r.høyre;
  r.høyre = q;
  p.høyre = r.venstre;
  r.venstre = p;
  return r;
}
```


```
Node<T> høyreZig(Node<T> p) // enkel høyrerotasjon
{
  Node<T> q = p.venstre;

  p.venstre = q.høyre;
  q.høyre = p;
  return q;
}
```


```
Node<T> venstreZig(Node<T> p) // enkel venstrerotasjon
{
  Node<T> q = p.høyre;

  p.høyre = q.venstre;
  q.venstre = p;
  return q;
}
```

Det å utføre en **splay** på en node ***X*** betyr at ***X*** skal roteres oppover i treet og bli en ny rot. En gitt node ***X*** vil ha en forelder ***F*** og en besteforelder ***B***. Det utføres så en zigzig eller en zigzag avhengig av mønsteret for ***X***, ***F*** og ***B***. Dermed har ***X*** kommet to nivåer oppover (nærmere roten) i treet, og en ny zigzig eller zigzag kan utføres på ***X***. Til slutt vil ***X*** enten bli ny rot eller et barn til den gamle roten. I siste tilfellet avsluttes det med en zig slik at ***X*** blir ny rot.

9.4.2 Når skal det utføres en splay?

Innsetting Det er **ikke** tillatt med duplikater i et splay-tre. En ny verdi settes inn på vanlig sortert plass i treet, og deretter splayes noden ***X*** med den nye verdien. Hvis dette var treets første verdi, så skal det ikke gjøres noe mer. Hvis vi forsøker å sette inn en verdi som allerede finnes i treet, så skal noden ***X*** som inneholder denne verdien splayes.

Søking Hvis den søkte verdien ligger i noden ***X*** så skal ***X*** splayes. Hvis den søkte verdien ikke finnes i treet, så splayes den siste noden ***X*** vi passerer på veien nedover under søkingen.

Sletting Hvis verdien som skal slettes ikke finnes i treet, så splayes den siste noden ***X*** vi passerer på veien nedover under søkingen etter verdien som skal slettes. Hvis verdien som skal slettes finnes i treet, så skal en node slettes og da ikke nødvendigvis den noden der verdien ligger. I noen tilfeller erstatter vi verdien med verdien fra en annen node og sletter denne andre noden. Men **splay-reglen** er hele tiden at det er **foreldrenoden** til **noden som slettes**, som skal splayes.

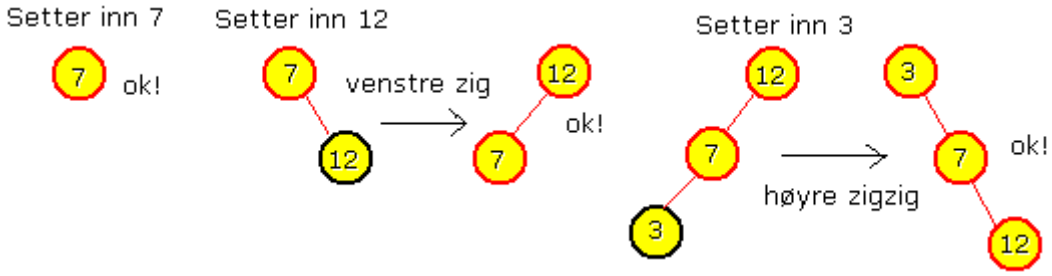
La ***Y*** være noden som inneholder den verdien vi skal slette fra treet. Vi deler det opp i to ulike situasjoner:

> **1. Y har ikke et venstre barn:** La ***X*** være forelder til ***Y***. Vi sletter så noden ***Y*** ved at høyre barn til ***Y*** erstatter ***Y*** og så splayes ***X***. Hvis ***Y*** ikke har et høyre barn (d.v.s. ***Y*** er en bladnode) så sletter vi bare ***Y***, men fortsatt skal vi splaye ***X***. Hvis ***X*** (eller eventuelt ***Y***) er rotnoden så skal det ikke gjøres noen splaying.
>
> **2. Y har et venstre barn:** La ***Z*** være den noden som kommer rett foran ***Y*** i sorteringen. Det er den som ligger lengst til høyre i det venstre subtreet til ***Y***. La ***X*** være forelderen til ***Z***. (Obs. ***X*** og ***Y*** blir samme node hvis det venstre barnet til ***Y*** ikke har et høyre barn). Verdien i ***Y*** erstattes med verdien i ***Z***, ***Z*** slettes og ***X*** splayes.

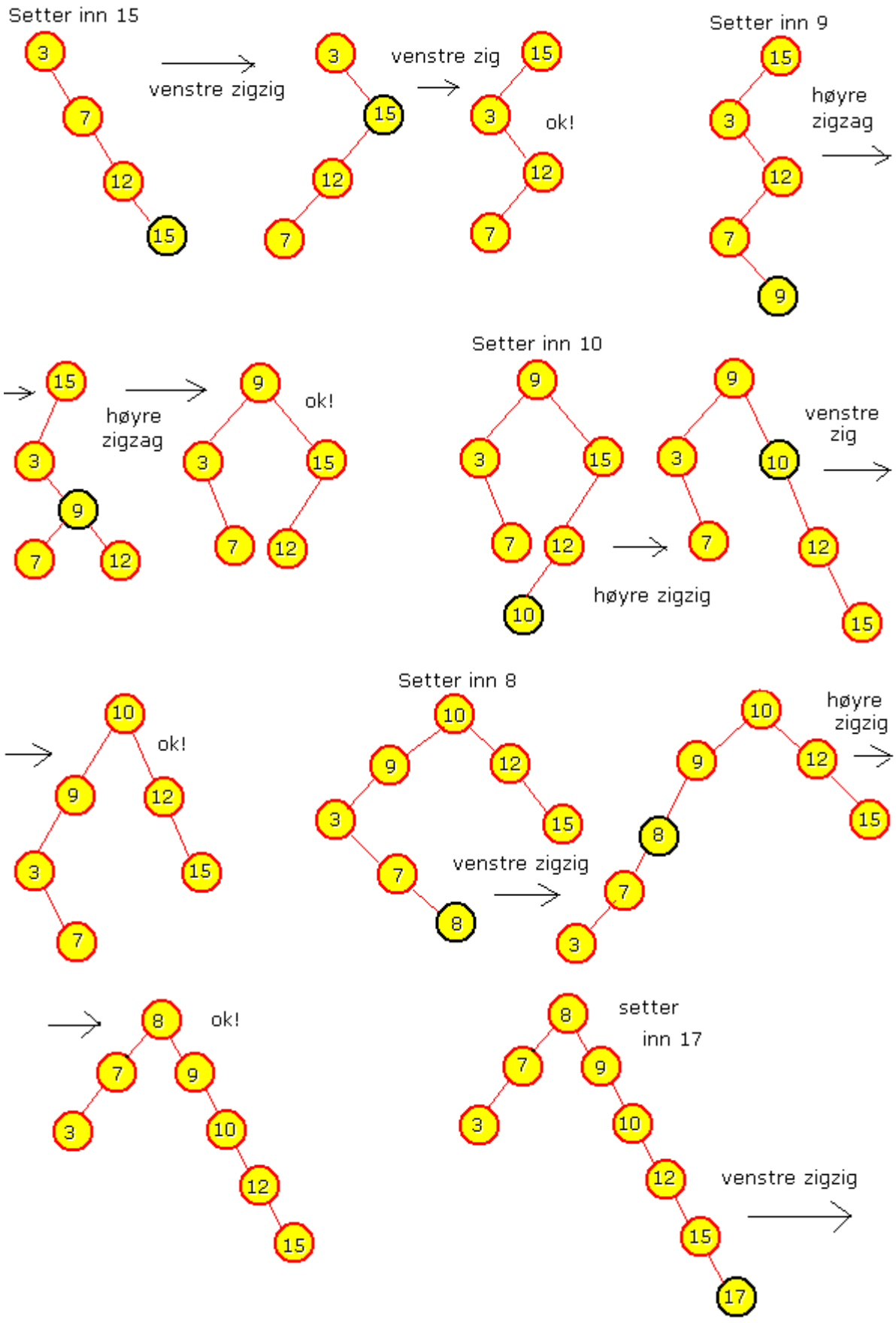
9.4.3 Et eksempel

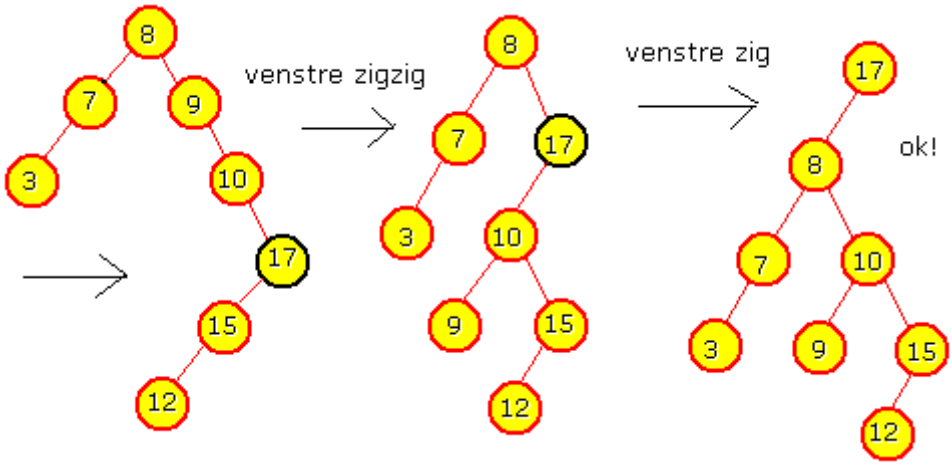
Verdiene 7, 12, 3, 15, 9, 10, 8, 17, 12 skal fortløpende settes inn i et splay-tre. På tegningene nedenfor er hver av verdiene satt inn og alle rotasjonene (zig-er, zigzig-er og zigzag-er) som trengs i hver splay er tegnet inn. Det er laget i form av en "tegneserie" der en pil viser til neste "rute" i tegneserien. En ny verdi settes inn på rett sortert plass. Det er markert ved at den nye noden har svart ring. Deretter splayes noden til topps. Den svarte ringen beholdes inntil noden har blitt rotnode - da blir ringen rød og på tegningen er det markert med ok!

Vi starter med å sette inn 7, 12 og 3:

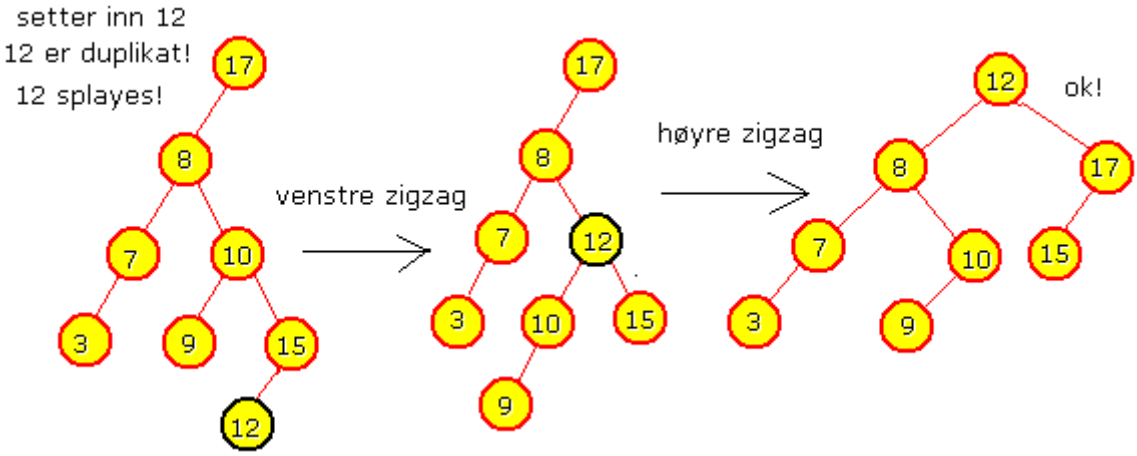


Fra og med verdien 15 blir det litt mer arbeid ved innsettingen:

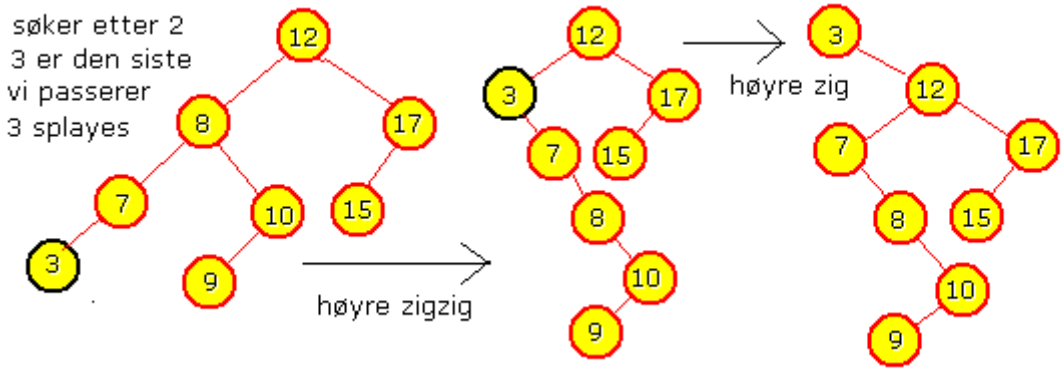

Setter inn 15
venstre zigzig
venstre zig
ok!
Setter inn 9
høyre
zigzag
høyre
zigzag
ok!
Setter inn 10
venstre
zig
høyre zigzig
ok!
Setter inn 8
høyre
zigzig
venstre zigzig
ok!
ok!
setter
inn 17
venstre zigzig



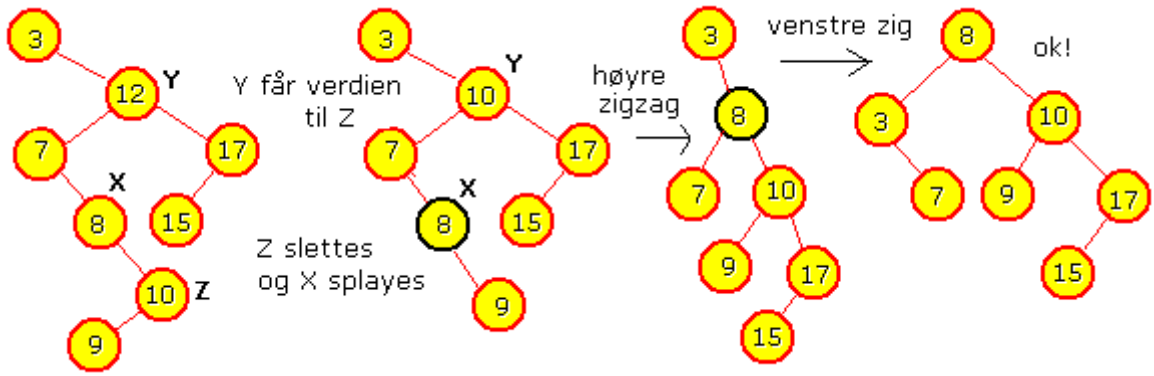
Til slutt skal vi sette inn 12. Da ser vi at 12 finnes fra før. Dermed skal noden som inneholder 12 splayes:



Vi skal søke etter verdien 2, men den er ikke i treet. Da skal den siste noden vi passerer på veien nedover under letingen etter 2 splayes. D.v.s. at her skal 3 splayes:



Vi skal slette verdien 12. I sletting er det to hovedtilfeller - 1) noden som skal slettes **mangler** et venstre barn og 2) noden **har** et venstre barn. I dette tilfellet ser vi at noden som inneholder 12 har et venstre barn. La noden som inneholder 12 hete **Y**. Den noden som kommer rett foran **Y** i sorteringen kaller vi **Z** - her blir det noden som inneholder 10. Foreldren til **Z** kaller vi **X** - her blir det noden med verdien 8. Reglen sier nå at verdien i **Y** erstattes med verdien i **Z**, at **Z** slettes og at **X** splayes!



En animasjon av splay-trær Denne animasjonen laget av Vladimir Yanovsky, Israel Inst. of Techn. kan brukes til å teste ens forståelse av operasjonene i et splay-tre der "bottom up" teknikken brukes.