1 Forelesning 10

1.1 Plan for forelesningen

- Induktive datastrukturer
- AVL trær
- Syntakstrær

1.2 Induktive data strukturer

En datastruktur er induktiv hvis den bruker seg selv som argument til minst en av konstruktørene.

```
data binTree a = Empty
    | Branch (BinTree a) a (BinTree a)
```

ELementene i en induktiv datastruktur kan settes sammen ved hjelp av konstrukt;rene til store strukturer (trær)

1.3 Rekursjon på induktive datastrukturer

For å definere en funksjon som tar en indktiv struktur som argumen ved rekursjon

- Gjøre mønster matching på funksjonen:
  - Lag et tilfelle for hver konstruktør
  - Argumentene til konstruktøren blir variabler
- Definer funksjonen i hvert tilfelle
- Kall funksjonen rekursivt på de variblene som igjen er elementer av den induktive datastrukturen

Eksempel. La oss beregne høyden av et binært tre

```
height :: BinTree a -> Integer
height Empty = 0
height (Branch lefts _ rights)
    = 1 + max (height lefts) (height rights)
```

Følger denne funksjonen oppskriften fra forrige slide? ◇

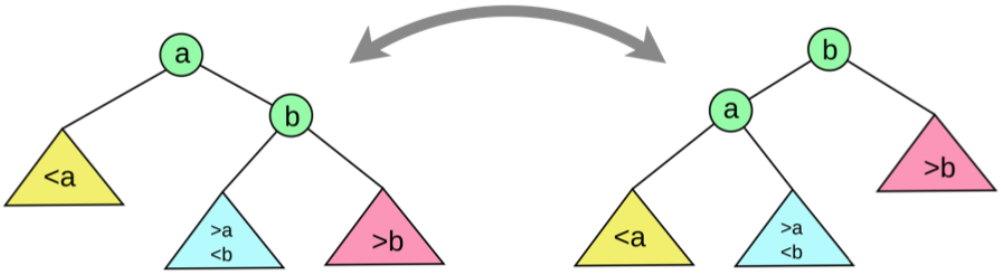


Figure 1: Illustrasjon av rotasjonsoperasjonen

1.4 Rotasjon av trær

Et tre kan roteres mot høyre, eller mot venstre, og en rotasjon kan endre høyden til treet.

Eksempel. Rotasjon av trær

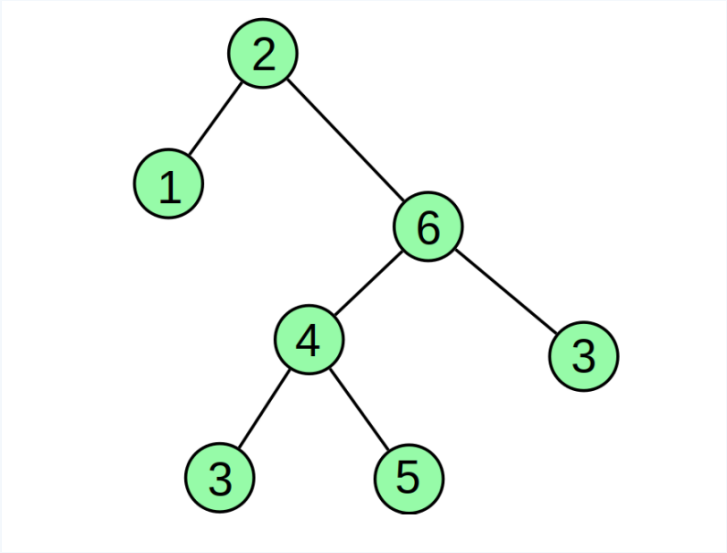
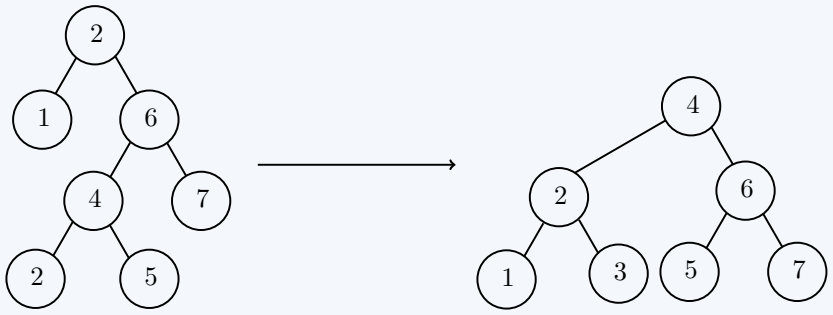


Figure 2: Bruk rotasjonsoperasjoner til å balansere treet.



◇

1.5 AVL Trær

- Introdusert av Adelson-Velsky og Landig i 1962
- Består av et binært tre med ekstra balanse informasjon
- Definerer et API p3l binære trær som roterer hver gang en ubalanse oppstår

1.6 AVL Trær: Implementasjon

AVL trær har en balanseinvariant

- Ingen subtrær kan ha en høydeforskjell på mer enn 1.
- En hver forgening er dekorert med høydeforskjellen på de to subtrærne. Denne kalles **balansefaktoren**.

Oppgave. Som en øvelse før vi gjør AVL trær, så implementerer vi sorterte lister, dekorert med om de er av odde eller partalls lengde.

1.7 AVL Datatypen

```
data BalanceFactor = LeftHeavy
                   | Balanced
                   | RightHeavy
                   deriving (Eq, Show)
data AVLTree a = Empty
               | Branch BalanceFactor
                        (AVLTree a)
                        a
                        (AVLTree a)
```