

# Øving 6: Trær

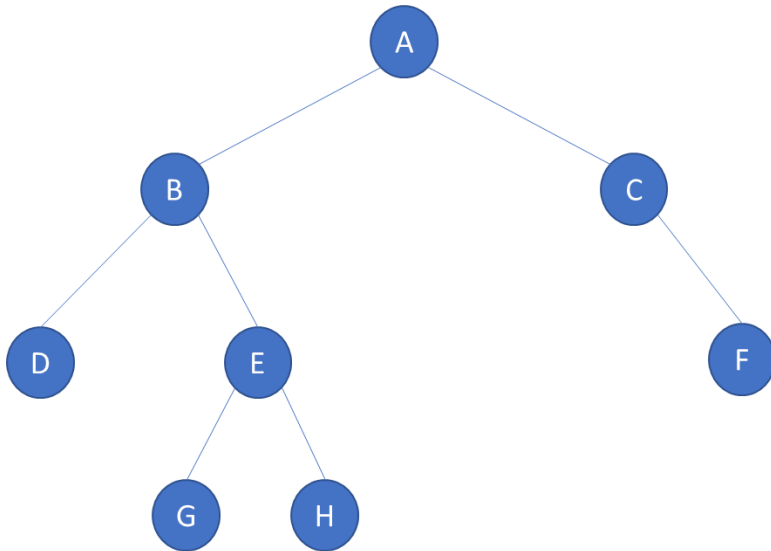
## Læringsmål

Dere skal forstå hvordan trestrukturer virker og lære hvordan å bruke dem.

## Godkjenning

Denne øvingen kan leveres i grupper på to studenter. Øvingen skal normalt sett godkjennes av studentassistent under øvingstimene.

## Oppgave 1: Forståelse av trestrukturer



Gitt treet over

a) Hvilken node er rotnoden?

A

b) Hvilke noder er bladnoder?

D, G, H, F

c) Fyll inn tabellen:

Node	Barn	Søsken	Dybde
A	B C	B C	0
B	D E	D E	1
C	F		1
D			2
E	G H	G H	2
F			2
G			3
H			3

d) Skriv nodene i treet i

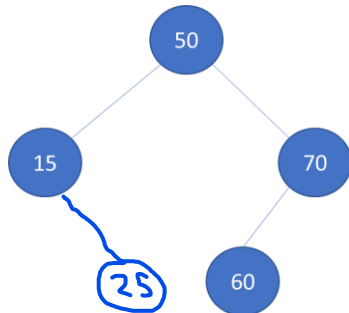
a. Preorder rekkefølge

A B D E G H C F

- b. Inorder rekkefølge  
c. Postorder rekkefølge

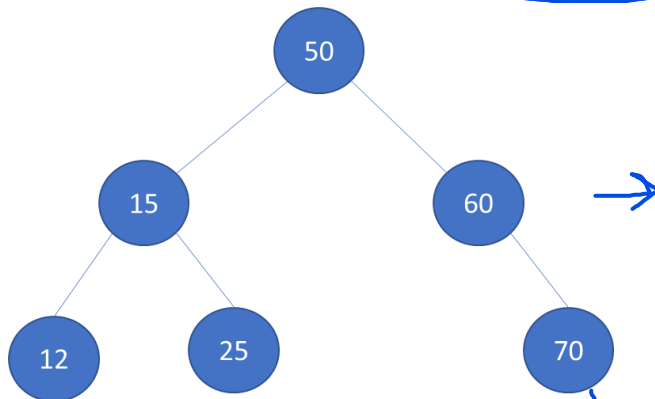
D B G E H A C F  
D G H E B F C A

## Oppgave 2: AVL trær



← Nice ←

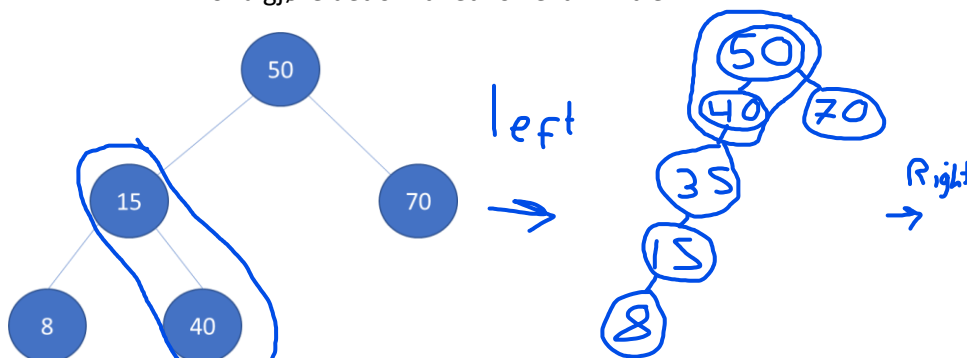
- a) Gitt AVL-treet over, som du kan anta er lovlig. Sett inn tallet 25.  
a. Tegn AVL treet  
b. Er AVL treet korrekt etter denne innsetningen? Hvis ikke, vis hvert steg i prosessen for å gjøre det om til et korrekt AVL tre



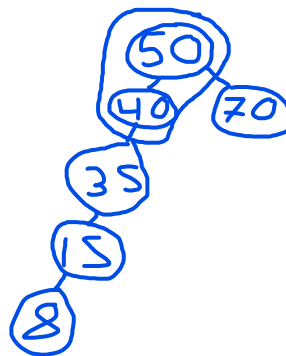
Balance  $\leq 1$

trenger ingen  
justering

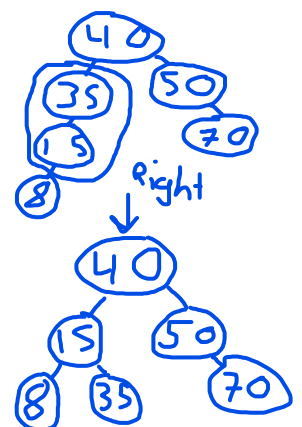
- b) Gitt AVL-treet over, som du kan anta er lovlig. Sett inn tallet 85.  
a. Tegn AVL treet  
b. Er AVL treet korrekt etter denne innsetningen? Hvis ikke, vis hvert steg i prosessen for å gjøre det om til et korrekt AVL tre



left



Right



- c) Gitt AVL-treet over, som du kan anta er lovlig. Sett inn tallet 35.  
a. Tegn AVL treet  
b. Er AVL treet korrekt etter denne innsetningen? Hvis ikke, vis hvert steg i prosessen for å gjøre det om til et korrekt AVL tre

35

### Oppgave 3: Bruk av trestrukturer

- a) Med utgangspunkt i den utdelte koden for et binærtre (`binaertre.py`), skriv en rekursiv metode som beregner antall bladnoder i treet.
- b) Lag en preorder iterator for den utdelte koden for et binærtre (`binaertre.py`). Du kan basere deg på postorder iteratoren som allerede ligger i den utdelte koden, men trenger ikke det. Merk at preorder iteratoren kan gjøres betydelig enklere enn postorderiteratoren er.
- c) Gitt fila `vindmaalinger_redusert_mer.txt` samt utdelt kode for å lese inn denne fila (`vindmaaling.py`). Bruk den utdelte AVL-tre implementasjonen (`avl_tre.py`) for å lage et map over vindmålinger for å gjøre følgende spørringer, samt implementere spørringene:
  - a. Finn hvor sterk vinden var midt på dagen (klokka 12:00) den 4. september 2010. Det holder å returnere nærmeste vindmåling etter det oppgitte tidspunktet.
  - b. Finn hvor sterk vinden var på sitt kraftigste den 7. desember 2011. Du kan gjøre dette ved å spørre om alle vindmålinger denne datoen (fra 7. desember 2011 klokka 00:00 til 8. desember 2011 klokka 0:00), og så finne maksimum gjennom å gå gjennom denne lista.
  - c. **Frivillig:** Utvid koden slik at bruker kan oppgi tidspunktene for oppgave a og b. Merk at fila bare inneholder målinger fra årene 2009 til 2012. Du kan for eksempel bruke datoen 2011-12-25 (tidspunktet Dagmar herjet på nordvestlandet og Trøndelag)
- d) **Frivillig:** En operasjon man av og til ønsker å gjøre på et AVL tre er å finne k-ende minste element, hvor k er en parameter til funksjonen. For eksempel «Finn det tredje minste elementet» eller «finn medianen». Medianen er elementet som er i midten av lista, så k er da lik halvparten av lengden til lista. En måte å gjøre dette på er å inkludere en ekstra egenskap i noden, `størrelse`. `Størrelse` er antall noder i subtreet som har rot i denne noden.
  - a. Vedlikehold «`størrelse`» egenskapen under innsetninger, inkludert rotasjoner
  - b. Lag metoden `finn_k_ende(k)` som finner k-ende minste verdi. Algoritmen for dette er som følgende:
    - i. Hvis `k-1` er mindre enn `størrelsen` til venstre barn, er svaret i venstre subtre. Kall metoden rekursivt på venstre barn.
    - ii. Hvis `k-1` er lik `størrelsen` på venstre barn, ligger verdien du leiter etter i den noden, og du kan returnere den.
    - iii. Hvis `k-1` er større enn `størrelsen` til venstre barn, er svaret i høyre subtre. Kall metoden rekursivt på høyre barn men med `k` redusert med `størrelsen` til venstre subtre +1 (den siste +1 representerer noden du står i nå)
  - c. Vedlikehold «`størrelse`» egenskapen under sletting
- e) **Frivillig:** Lag en levelorder iterator for basis binærtreet (`binaertre.py`)
- f) **Frivillig:** Filsystemet på en typisk datamaskin er formet som et tre hvor du har en eller flere rot-mapper («C:\» på en windows-maskin, «/» på en mac eller unix maskin), hvor hver mappe kan inneholde en kombinasjon av vanlige filer (bladnoder) og andre mapper (interne noder). Skriv et Python program som beregner `størrelsen` til alle filene som ligger under en oppgitt mappe, inkludert filer som ligger i undermappene. Merk: For å skaffe informasjon om data slik som `størrelsen` til filer, må dere importere `pathlib`, som er et mer avansert bibliotek for å behandle filer enn det som ligger inne standard.