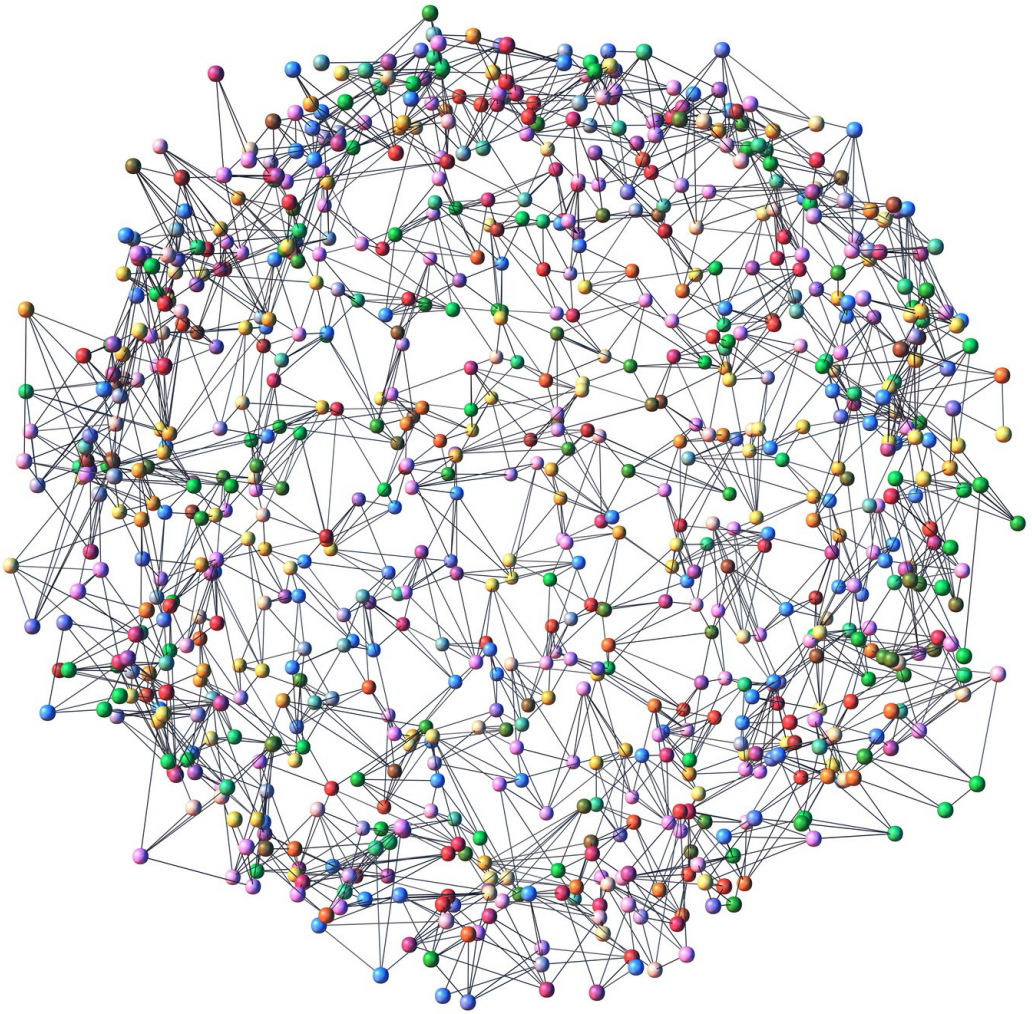
Grafalgoritmer: Traversering



Vi skal se på grafalgoritmer for:

- Traversering:
  - Oppsøk alle nodene i grafen en og bare en gang, på en eller annen systematisk måte
- Nåbarhet:
  - Finnes det en vei fra en node til en annen node?
- Korteste vei:
  - Hvor lang er den korteste veien mellom par av noder i en vektet graf?

Traversering av grafer

- Traversering (kalles også “søking i grafen”):
  - Gå innom/søke opp alle noder i grafen en og bare en gang
- Problemer ved graftraversering:
  - Det er ofte ingen “rot” eller naturlig startpunkt i en graf
  - Grafen kan være cyklisk slik at en node oppsøkes flere ganger
  - Grafen kan være usammenhengende eller svakt sammenhengede, slik at ikke alle noder kan nås fra en gitt node

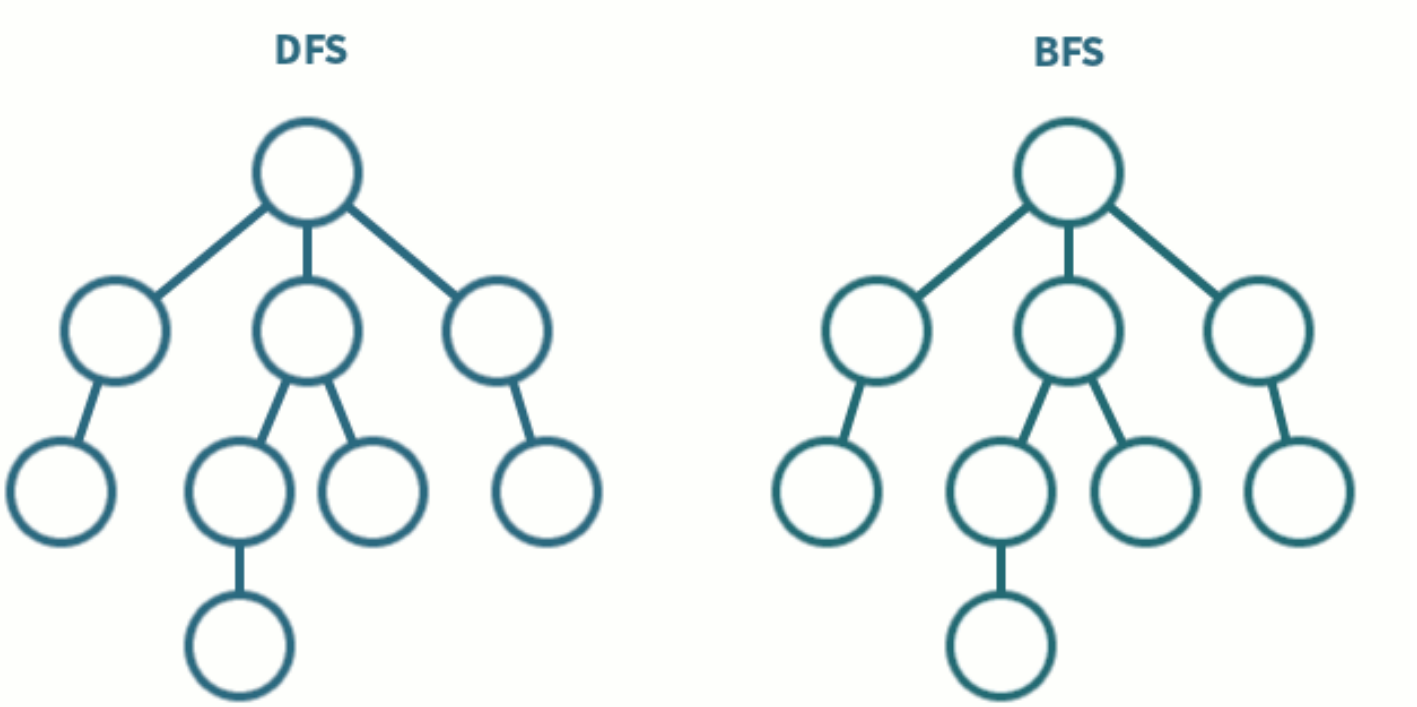
DFS og BFS

- To standard rekkefølger for traversering av grafer:
  - DFS: Dybde-først søk (Depth-First Search)
  - BFS: Bredde-først søk (Breadth-First Search)
- Felles for begge traverseringsmetoder:
  - Starter med en første node som markeres oppsøkt
  - Går deretter videre til sist oppsøkte nodes uoppsøkte naboer, på en eller annen systematisk måte
  - Traversering med nabomatrise er alltid $O(n^2)$

Forskjellen på DFS og BFS

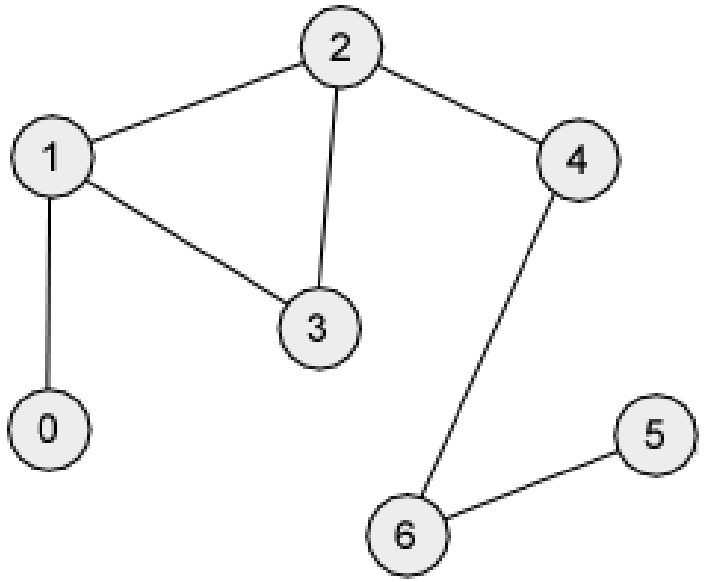
- Dybde-først:
  - Går rekursivt direkte videre til første uoppsøkte node
  - Vil gå “i dybden” langs veier med uoppsøkte noder
- Bredde-først:
  - Lagrer uoppsøkte noder i en (FIFO) kø
  - De uoppsøkte nodene nærmest startnoden oppsøkes først
  - Søkingen brer seg “i ringer” fra utgangspunktet

Animasjon av DFS og BFS



Traversering: Eksempel

- Urettet graf, starter i node 2
- Uoppsøkte naboer oppsøkes i stigende rekkefølge

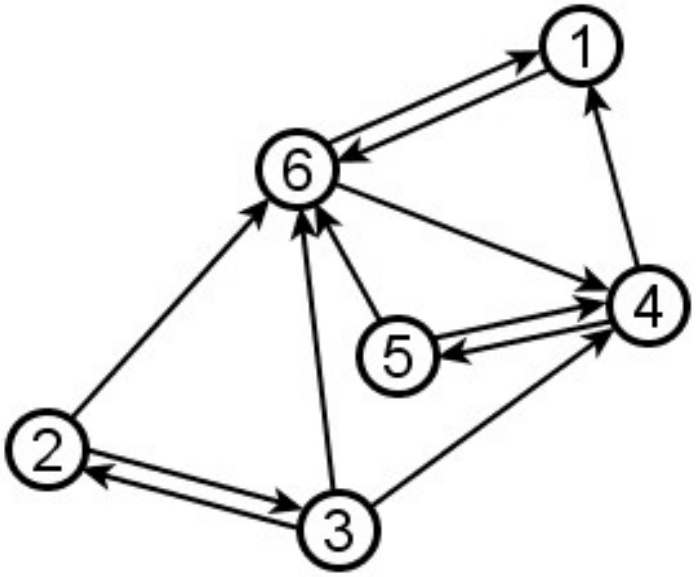


DFS: 2 1 0 3 4 6 5

BFS: 2 1 3 4 0 6 5

Traversering: Eksempel

- Rettet graf, starter i node 2
- Uoppsøkte naboer oppsøkes i stigende rekkefølge

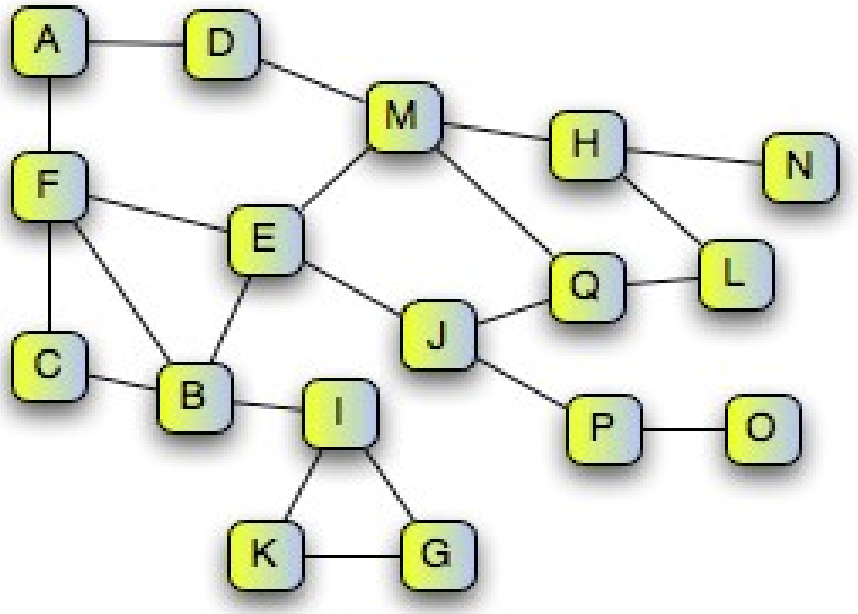


DFS: 2 3 4 1 6 5

BFS: 2 3 6 4 1 5

Traversering: Eksempel

- Urettet graf, starter i node A
- Uoppsøkte naboer oppsøkes i alfabetisk rekkefølge



DFS: A D M E B C F I G K J P O Q L H N

BFS: A D F M B C E H Q I J L N G K P O

Mer animasjon av DFS og BFS

- Med muligheter for å variere:
  - Rettet/urettet graf
  - Størrelsen på eksempelgrafen
  - Fremvisning av grafen:
    - Med noder som sirkler og kanter som linjer/piler
    - Som nabolister
    - Som nabomatrise

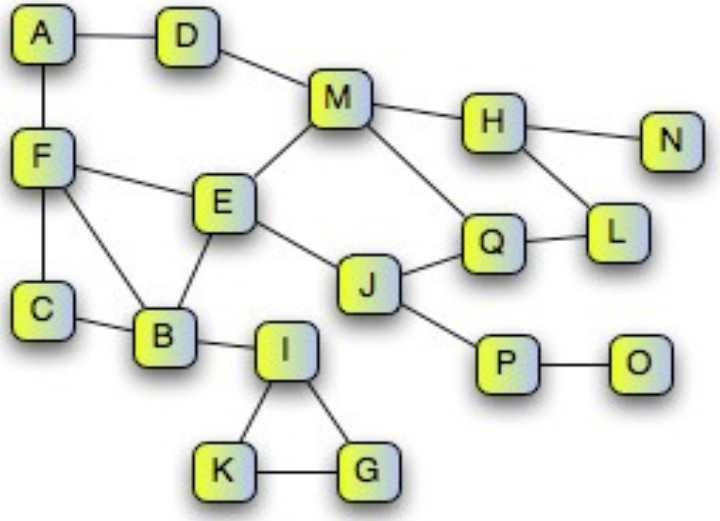
- Dybde-først søk
- Bredde-først søk

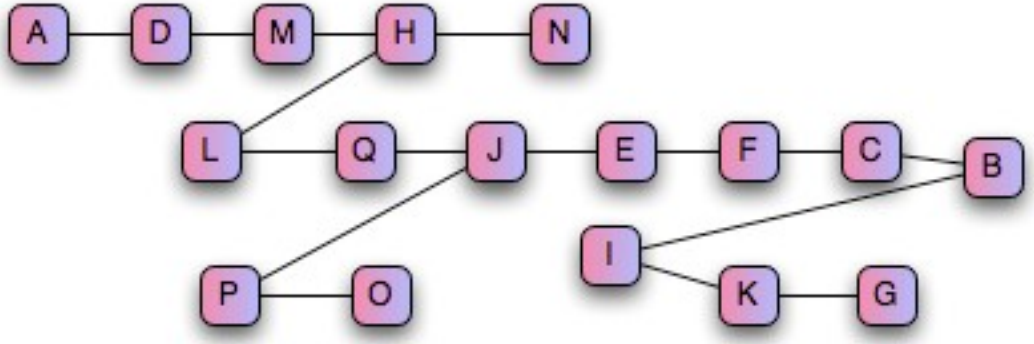
Traversering og spenn-trær *

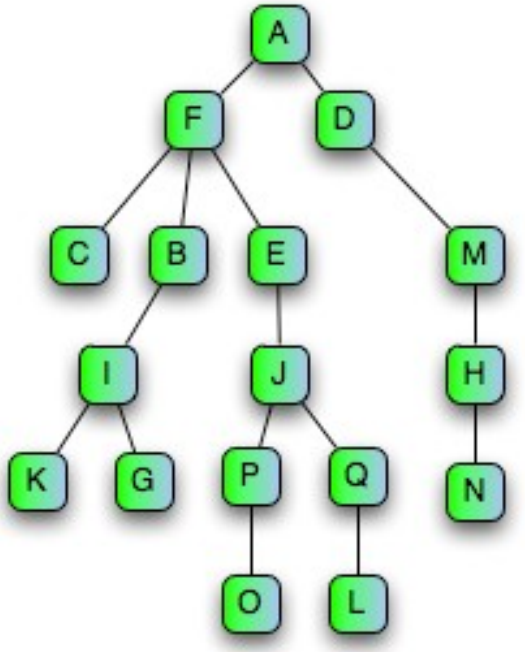
- Spenn-tre:
  - Et *tre* som består av alle nodene i en graf og et *utvalg* av kantene, slik at alle nodene er forbundet
- I læreboka:
  - Algoritme for å finne spenn-treet med minst total veilengde (minimum spanning tree) i en vektet graf
- Kantene som følges i en traversering, vil alltid utgjøre et spenn-tre:
  - Dybde-først: Spenn-treet blir høyt og ubalansert
  - Bredde-først: Lavere, mer balansert spenn-tre

* Engelsk: “spanning tree”

Spenntrær for DFS og BFS







Dybde-først søk: Algoritme

- DFS implementeres naturlig rekursivt:

```
DFS(node x)
{
   oppsøk(x)
   for hver node y som er nabo med x
      hvis y ikke er oppsøkt
         DFS(y)
}
```

- Læreboka implementerer grafer som en generisk ADT med en ikke-rekursiv DFS-iterator som bruker en stack

Bredde-først søk: Algoritme

- BFS implementeres iterativt med bruk av en kø:

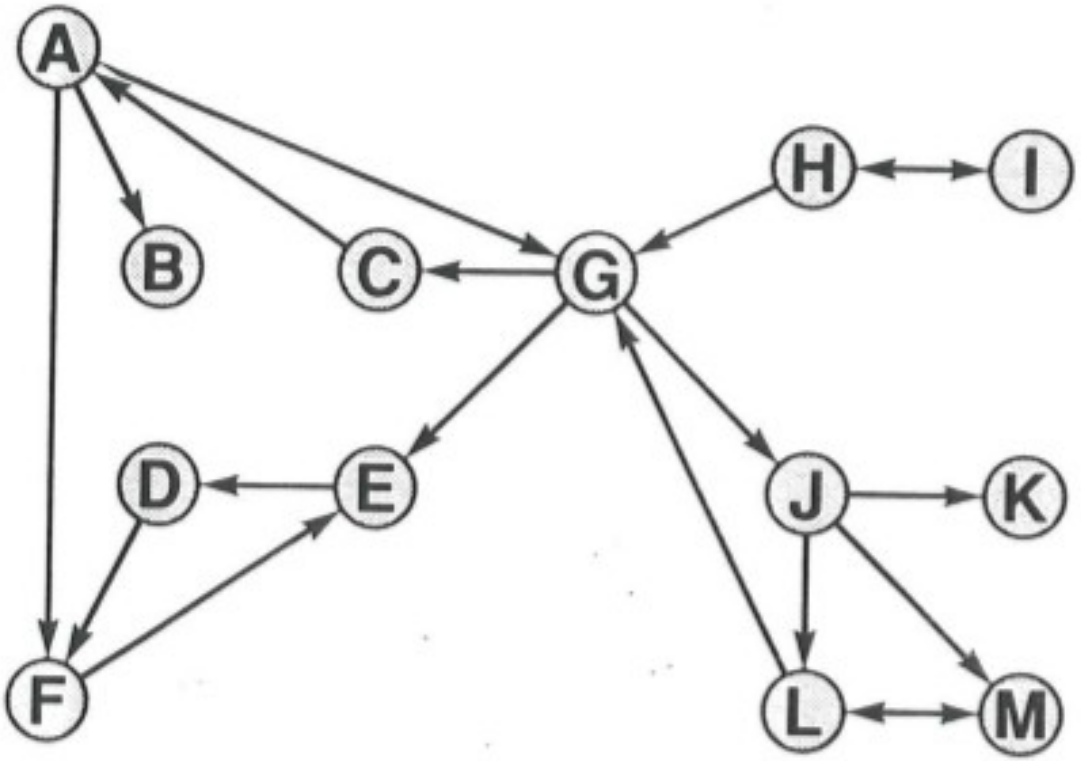
```
BFS(node start)
{
  enqueue(start)
  så lenge kø ikke tom
  {
     x = dequeue()
     oppsøk(x)
      for hver node y som er nabo med x
        hvis y ikke er oppsøkt eller ligger i kø
          enqueue(y)
   }
}
```

- Læreboka implementerer en generisk BFS-iterator

Enkel implementasjon av DFS og BFS

- Utvider klassen `enkelGraf.java` med metoder for dybde-først og bredde-først traversering
- DFS:
  - Rekursiv
  - Bruker boolsk array til å merke noder som er oppsøkt
- BFS:
  - Bruker Javas innebygde kømodul
  - Boolsk array merker noder som er oppsøkt *eller* i kø
- Java-kode: `traverserGraf.java`

Uvektet, rettet graf for testing av DFS og BFS



Fil: `graf_13.txt`