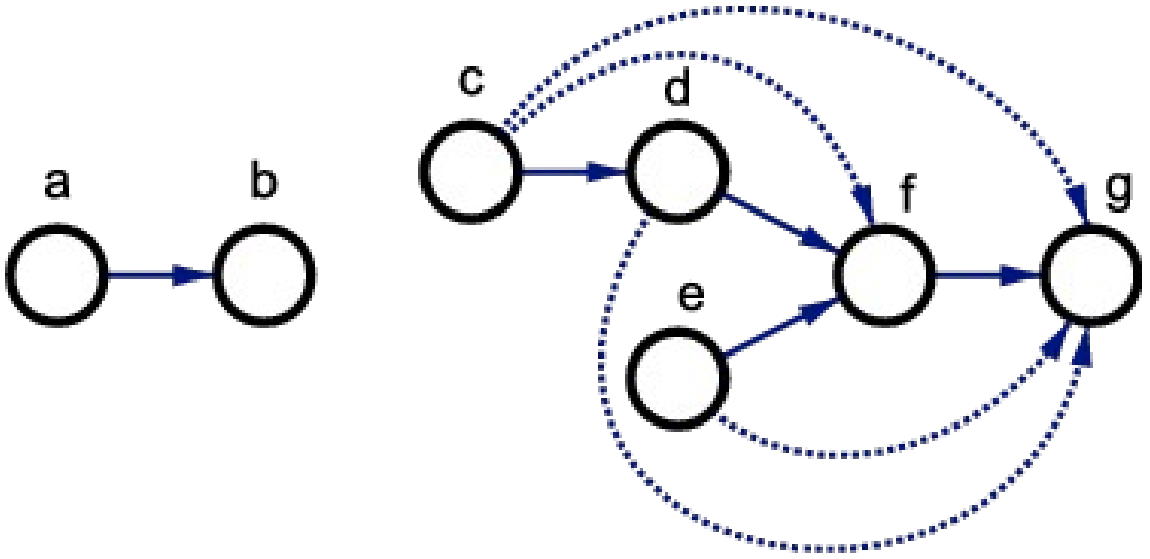
Grafalgoritmer:
Nåbarhet / Reachability



Nåbarhet *

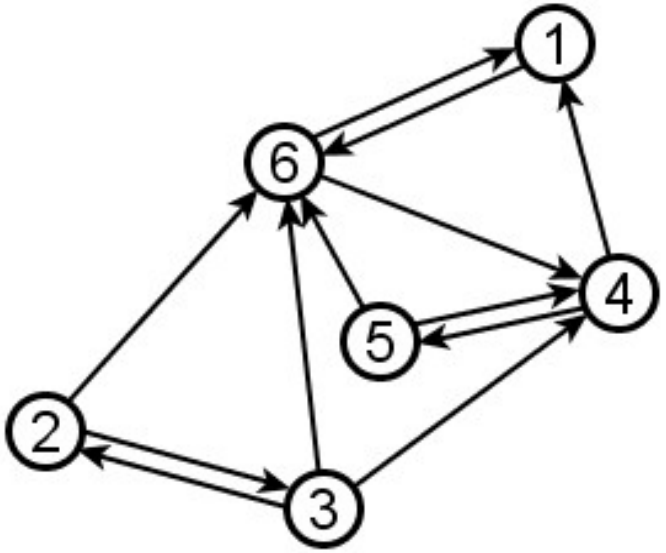
- En node er *nåbar* fra en annen node hvis det finnes en vei i grafen mellom de to nodene
- En vanlig problemstilling for grafer er å finne ut hvilke noder som er innbyrdes nåbare – “kan vi reise fra A til B?”
- Bestemmelse av nåbarhet kan også brukes til å avgjøre om en graf er (sterkt eller svakt) sammenhengende

*: Engelsk: “Reachability”

To standardproblemer for nåbarhet i grafer

- SSRP – "Single Source Reachability Problem":
  - Finn alle noder som er nåbare fra *en bestemt* node
- APRP – "All-Pairs Reachability Problem"
  - Finn alle noder som kan nås fra *enhver node* i grafen

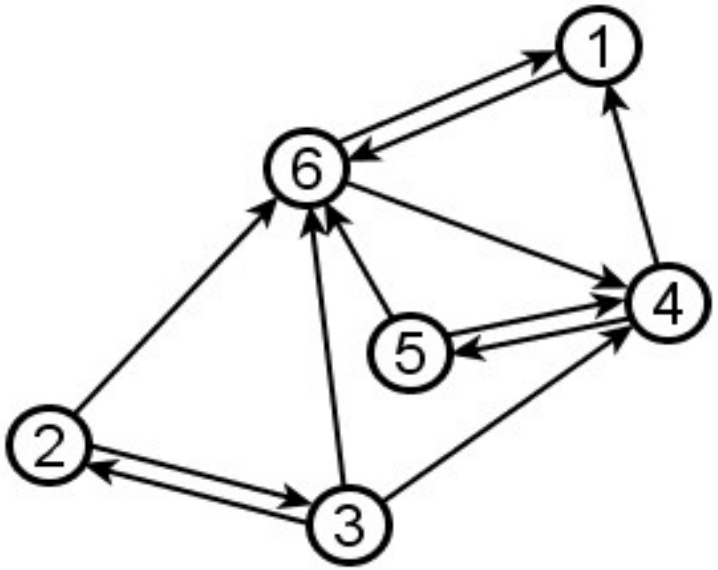
Single Source Reachability Problem



Alle noder nåbare fra 1: 4, 5, 6

- Enkel $O(n^2)$ løsning på SSRP:
  - Gjør en dybde-først eller bredde-først traversering fra startnoden
  - Alle noder som oppsøkes i traverseringen er nåbare

All-Pairs Reachability Problem



Alle noder nåbare fra:

1: 4, 5, 6

2: 1, 3, 4, 5, 6

3: 1, 2, 4, 5, 6

4: 1, 5, 6

5: 1, 4, 6

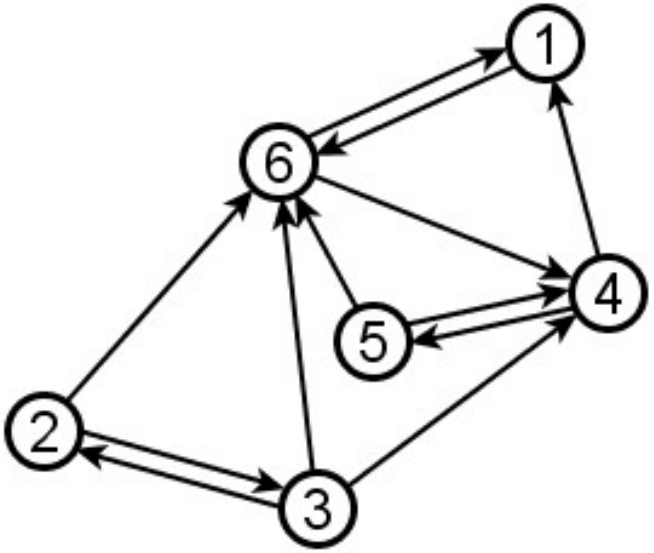
6: 1, 4, 5

Enkel løsning av APRP

- Traversér grafen *n* ganger (med DFS eller BFS), én gang med start i hver node i grafen
- Denne løsningsmetoden er alltid $O(n^3)$ hvis grafen er lagret med nabomatrise
- Kan også brukes for å teste *konnektivitet*:
  - Hvis *alle* traverseringene er innom *alle n* noder, er grafen (sterkt) sammenhengende
- Implementasjon: `traverserGraf.java`

Løsningsmatrise for APRP

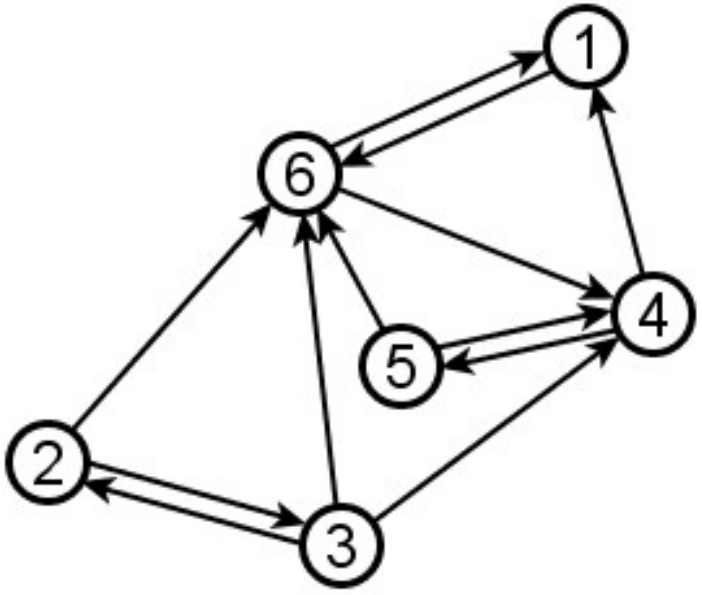
Løsningen på APRP kan representeres som en boolsk matrise/2-D tabell *A*, der *A*[*i*] [*j*] er *true* hvis og bare hvis det går en vei fra node *i* til node *j*



	1	2	3	4	5	6
1	T	F	F	T	T	T
2	T	T	T	T	T	T
3	T	T	T	T	T	T
4	T	F	F	T	T	T
5	T	F	F	T	T	T
6	T	F	F	T	T	T

Løsningsmatrise og nabomatrise

Løsningsmatrisen for APRP kan betraktes som en *utbygging* av nabomatrisen for grafen



	1	2	3	4	5	6
1	T	F	F	F	F	T
2	F	T	T	F	F	T
3	F	T	T	T	F	T
4	T	F	F	T	T	F
5	F	F	F	T	T	T
6	T	F	F	T	F	T

Nabomatrise

	1	2	3	4	5	6
1	T	F	F	T	T	T
2	T	T	T	T	T	T
3	T	T	T	T	T	T
4	T	F	F	T	T	T
5	T	F	F	T	T	T
6	T	F	F	T	T	T

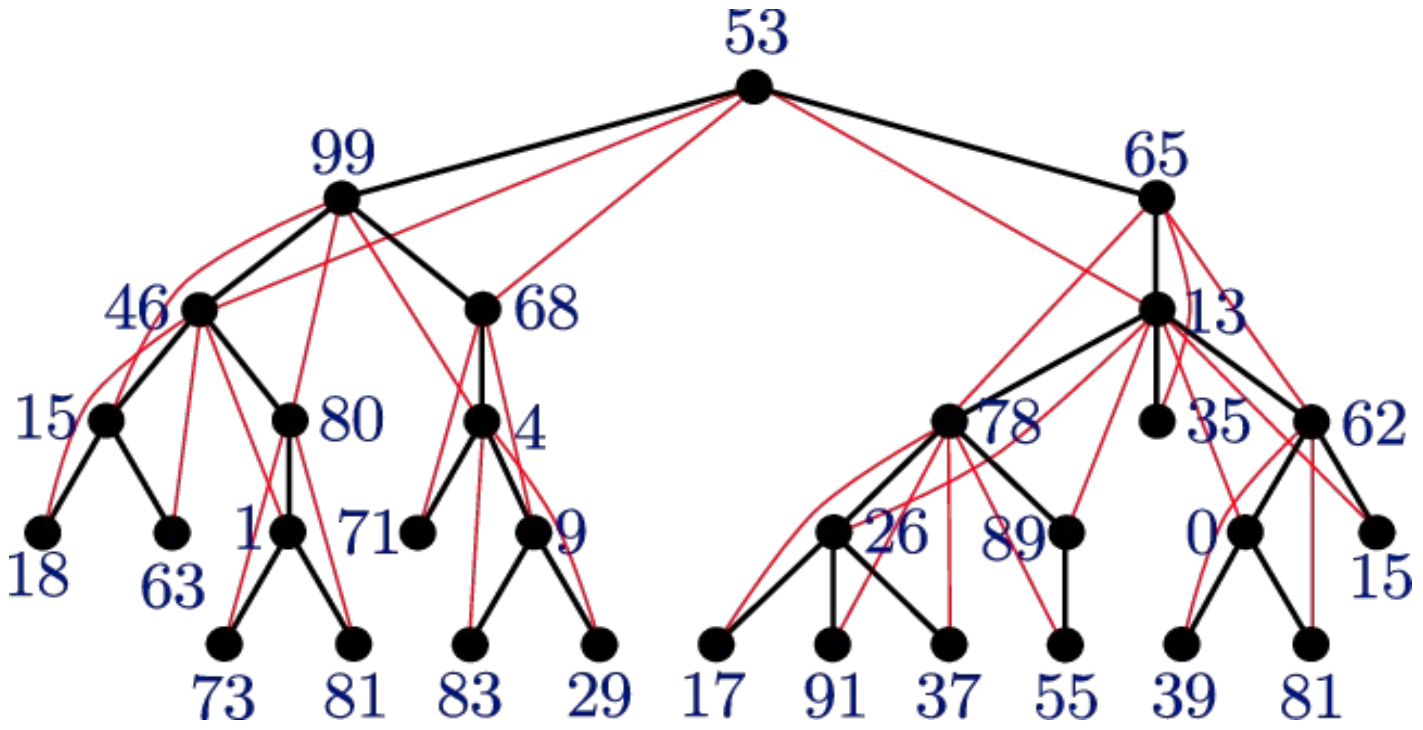
Løsningsmatrise

Warshall-algoritmen*

- Løser APRP ved å transformere nabomatrisen til løsningsmatrisen
- Bruker følgende prinsipp:
  - Det finnes en vei fra node *i* til node *j*, hvis det finnes en vei fra node *i* til node *k*, og fra node *k* til node *j*
- Algoritmen bygger stegvis ut nabomatrisen med alle mulige veier mellom alle par av noder
- Etter *n* steg vil garantert alle veier i hele grafen ligge lagret i løsningsmatrisen

*: Warshall, Stephen: "A theorem on Boolean matrices", *Journal of the ACM*, 1962

- Sort: Kantene i grafen
- Rødt: Forbindelser av lengde 2 funnet etter første steg i Warshall-algoritmen

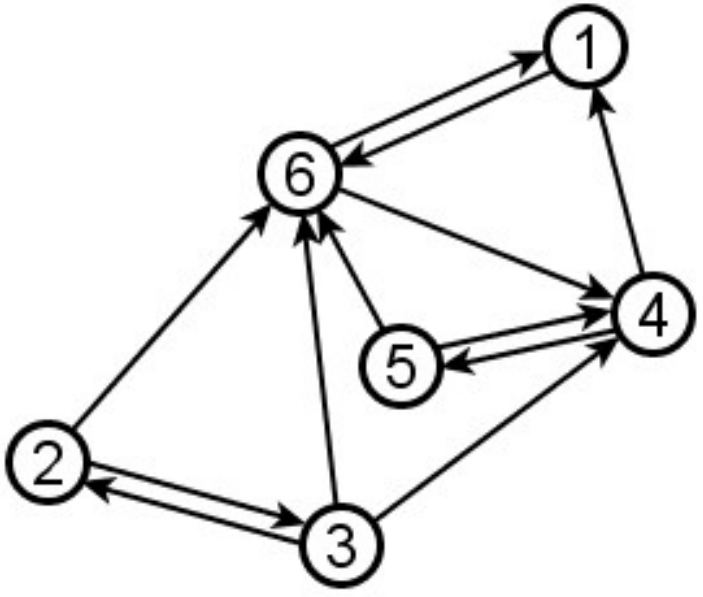


Warshall: Stegvis løsning

- Steg 1: Finn alle veier av lengde 2
- Steg 2: Finn alle veier av lengde 3

.
.
.

- Steg *n*: Alle veier funnet



	1	2	3	4	5	6
1	T	F	F	F	F	T
2	F	T	T	F	F	T
3	F	T	T	T	F	T
4	T	F	F	T	T	F
5	F	F	F	T	T	T
6	T	F	F	T	F	T

Nabomatrise

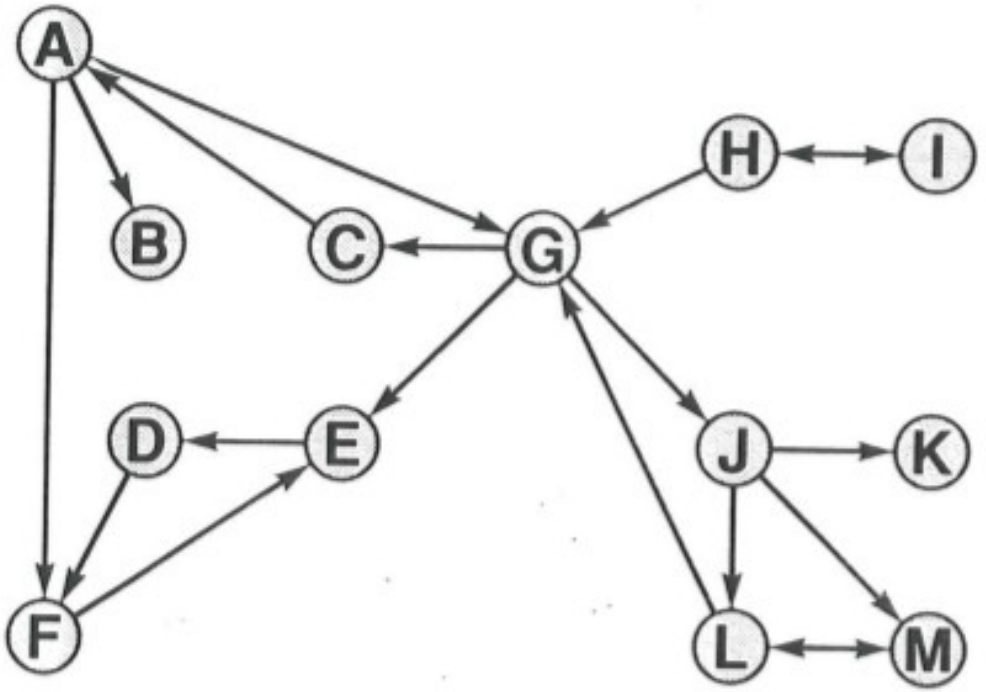
	1	2	3	4	5	6
1	T	F	F	T	T	T
2	T	T	T	T	T	T
3	T	T	T	T	T	T
4	T	F	F	T	T	T
5	T	F	F	T	T	T
6	T	F	F	T	T	T

Løsningsmatrise etter to steg

Warshall-algoritmen: Effektivitet og implementasjon

- Programmeres enkelt med tre løkker:
  - Ytre løkke går *n* ganger, i hvert steg finnes alle veier av lengde 1, 2, 3, ...
  - To indre løkker som går gjennom hele nabomatrisen og kobler node *i* og *j* hvis begge har en vei til node *k*
- Warshall er alltid $O(n^3)$ for graf lagret i nabomatrise
- Enkel implementasjon: `warshall.java`

Uvektet, rettet graf for testing av APRP-metoder



Fil: `graf_13.txt`