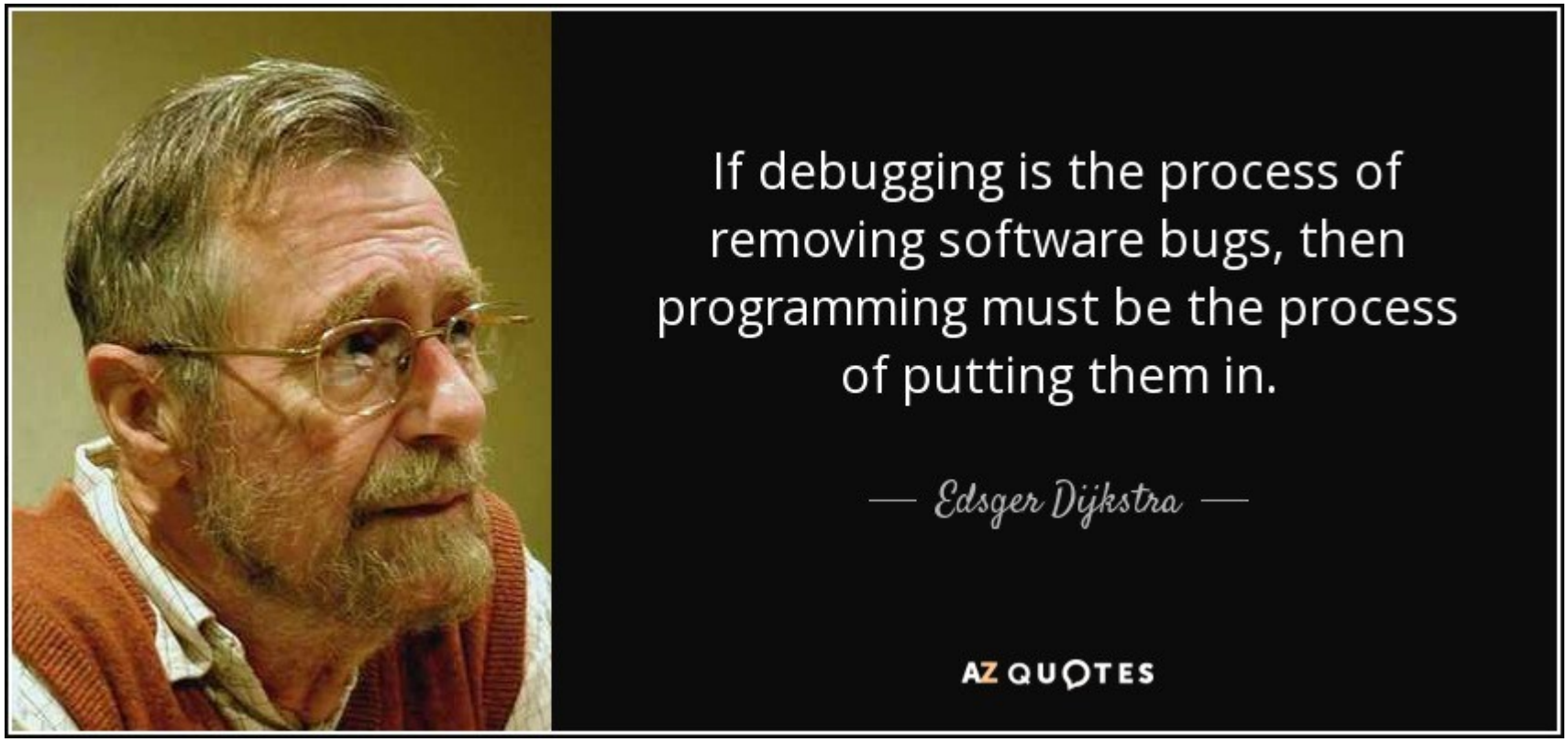
Grafalgoritmer: Korteste vei



Korteste-vei problemer for vektede grafer *

- "Single Source Shortest Path Problem"
  - Finn *lengden* av korteste vei fra *én* bestemt node til *alle* andre noder i grafen
- "All-Pairs Shortest Path Problem"
  - Finn *lengden* av korteste vei fra *alle* noder til *alle* andre noder i grafen

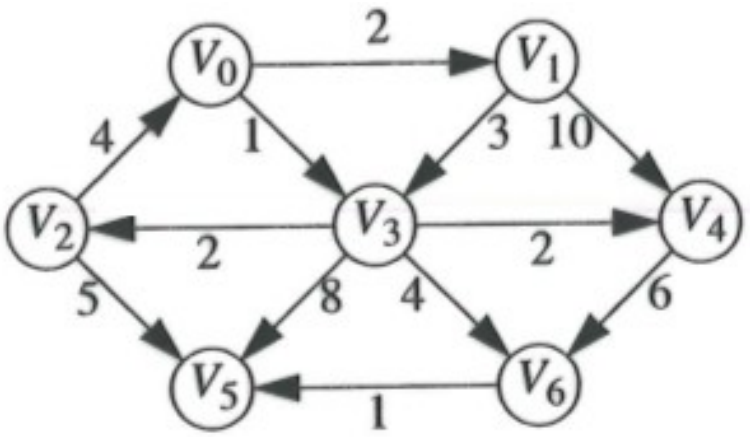
*: For uvektede grafer finnes korteste veier enkelt med bredde-først søk

Anvendelser av korteste-vei algoritmer

- Ruteplanlegging
- GPS
- Google Maps
- Travelling Salesman's Problem
- Flåtestyring
- Ulike typer optimaliseringer
- Og veldig mye mer...

Vektede grafer for korteste-vei problemer

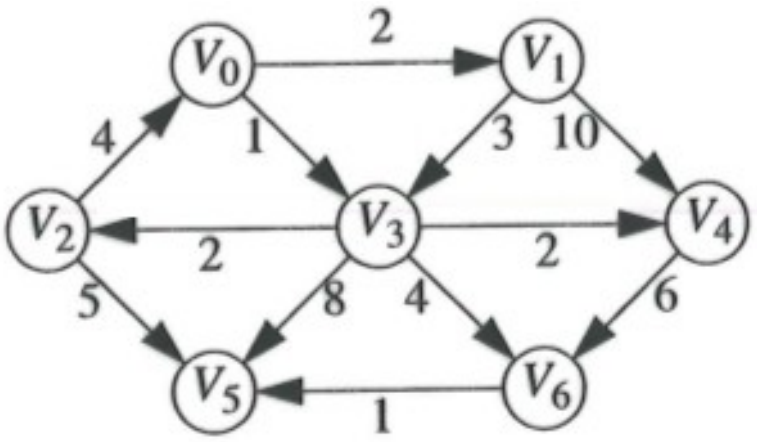
- Representerer grafen som en nabomatrise med kantlengder
- Hvis det ikke går en kant mellom to noder, legger vi inn en "uendelig stor" kantlengde



	0	1	2	3	4	5	6
0	0	2	∞	1	∞	∞	∞
1	∞	0	∞	3	10	∞	∞
2	4	∞	0	∞	∞	5	∞
3	∞	∞	2	0	2	8	4
4	∞	∞	∞	∞	0	∞	6
5	∞	∞	∞	∞	∞	0	∞
6	∞	∞	∞	∞	∞	1	0

All-Pairs Shortest Path

- Finn lengden av korteste vei mellom alle par av noder
- Kan løses ved å transformere nabomatrisen til en løsnings-matrise med korteste veilengder



	0	1	2	3	4	5	6
0	0	2	∞	1	∞	∞	∞
1	∞	0	∞	3	10	∞	∞
2	4	∞	0	∞	∞	5	∞
3	∞	∞	2	0	2	8	4
4	∞	∞	∞	∞	0	∞	6
5	∞	∞	∞	∞	∞	0	∞
6	∞	∞	∞	∞	∞	1	0

	0	1	2	3	4	5	6
0	0	2	3	1	3	6	5
1	9	0	5	3	5	8	7
2	4	6	0	5	7	5	9
3	6	8	2	0	2	5	4
4	∞	∞	∞	∞	0	7	6
5	∞	∞	∞	∞	∞	0	∞
6	∞	∞	∞	∞	∞	1	0

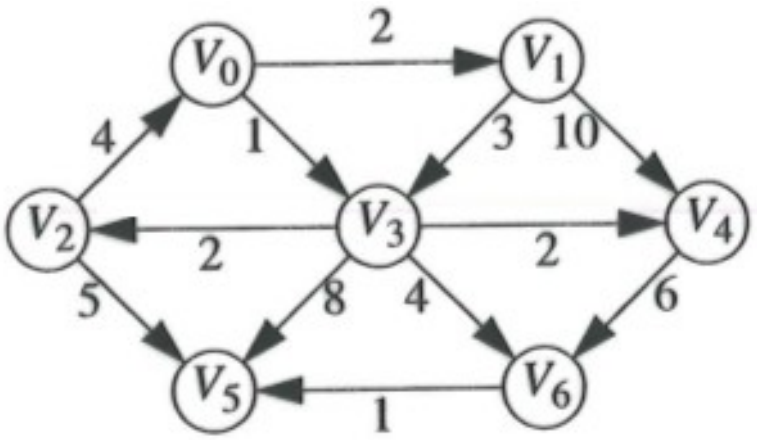
Floyds algoritme *

- Starter med nabomatrisen, bygger opp løsnings-matrisen steg for steg
- Samme prinsipp som i Warshall-algoritmen:
  - 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*
  - Sjekker alle mulige veier, tar hele tiden vare på den *korteste* veien som til nå er funnet mellom to noder
  - I hvert steg øker antall noder langs veiene med 1
- Etter *n* steg vil garantert lengden til alle de korteste veiene i hele grafen ligge lagret i løsningsmatrisen

*: Floyd, Robert W.: "Algorithm 97: Shortest Path", *Communications of the ACM*, 1962

Floyd: Stegvis løsning

- Steg 1:
  - Finn alle korteste veier med maks. 3 noder
- Steg 2:
  - Finn alle korteste veier med maks 4 noder
- Steg *n* – 2:
  - Alle korteste veier funnet



	0	1	2	3	4	5	6
0	0	2	∞	1	∞	∞	∞
1	∞	0	∞	3	10	∞	∞
2	4	∞	0	∞	∞	5	∞
3	∞	∞	2	0	2	8	4
4	∞	∞	∞	∞	0	∞	6
5	∞	∞	∞	∞	∞	0	∞
6	∞	∞	∞	∞	∞	1	0

	0	1	2	3	4	5	6
0	0	2	3	1	3	6	5
1	9	0	5	3	5	8	7
2	4	6	0	5	7	5	9
3	6	8	2	0	2	5	4
4	∞	∞	∞	∞	0	7	6
5	∞	∞	∞	∞	∞	0	∞
6	∞	∞	∞	∞	∞	1	0

Animasjon av Floyd's algoritme

- 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

- Floyd-Warshall All-Pairs Shortest Path

Floyds algoritme: Effektivitet og implementasjon

- Programmeres enkelt med tre løkker, på samme måte som Warshall-algoritmen:
  - Ytre løkke går $n - 2$ ganger, i hvert steg finnes alle de korteste veiene med 3, 4, 5, … noder
  - To indre løkker som går gjennom hele nabomatrisen og kobler node *i* og *j* hvis begge har en vei til node *k*
- Floyd er alltid $O(n^3)$ for graf lagret i nabomatrise
- Enkel implementasjon: `floyd.java`

Single Source Shortest Path

- Finn lengden av korteste vei fra én bestemt node til alle andre noder i grafen
- Kan løses med Dijkstras algoritme *
- Standard algoritme i mange systemer, f.eks. GPS-enheter, Google Maps, flybillettbestilling, ruting av av data på internett...

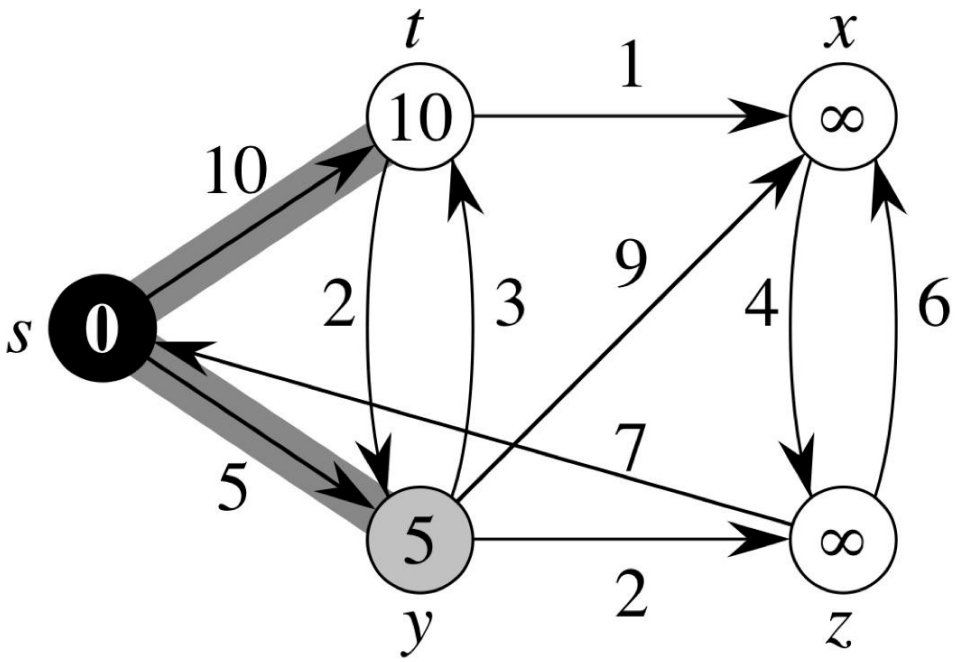
*: Dijkstra, Edsger. W.: "A note on two problems in connexion with graphs", *Numerische Mathematik 1*, 1959

Dijkstras algoritme

- Anvendes på en vektet graf med *n* noder
- Starter i en node *S*, finner korteste avstand fra *S* til alle andre noder
- Bruker maksimalt *n* – 1 steg, i hvert steg finner vi den korteste avstanden fra *S* til én ny node i grafen
- Ligner mye på et bredde-først søk
- Basert på følgende prinsipp:
  - Den korteste veien fra *S* til en ny node *må* gå langs en av de korteste veiene til en av den nye nodens naboer

Dijkstra: Ekstra datastruktur

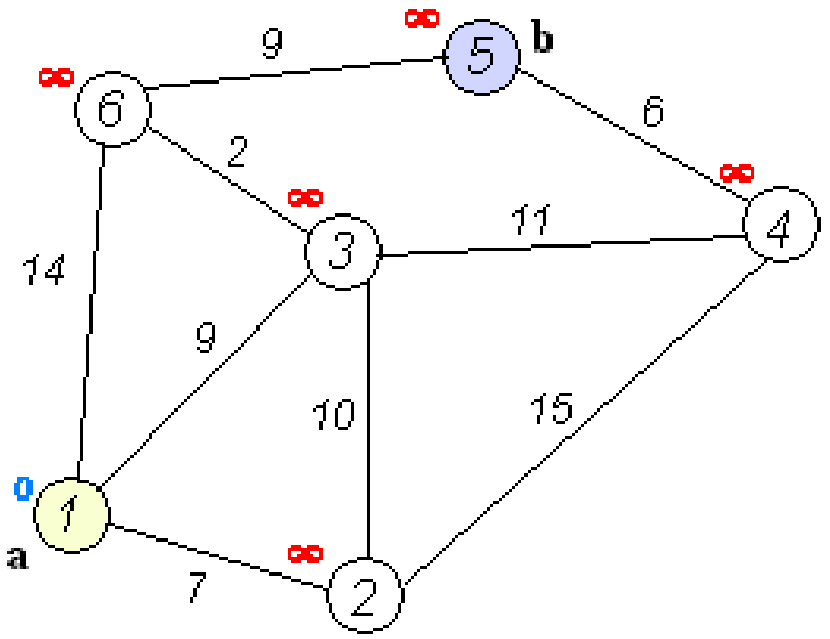
- En boolsk array `funnet` av lengde *n:*
  - Alle elementer er intitielt lik `false`
  - Brukes til å merke nodene som vi *allerede* har funnet korteste vei til
- En array `avstand` med *n* veilengder:
  - Lagrer den *hittil* korteste veien som vi til nå har sett fra startnoden *S* til alle andre noder i grafen
  - Alle elementer er lik “uendelig” til å begynne med, unntatt for startnoden der korteste vei settes lik 0
  - Løsningen ligger i arrayen `avstand` når algoritmen er ferdig



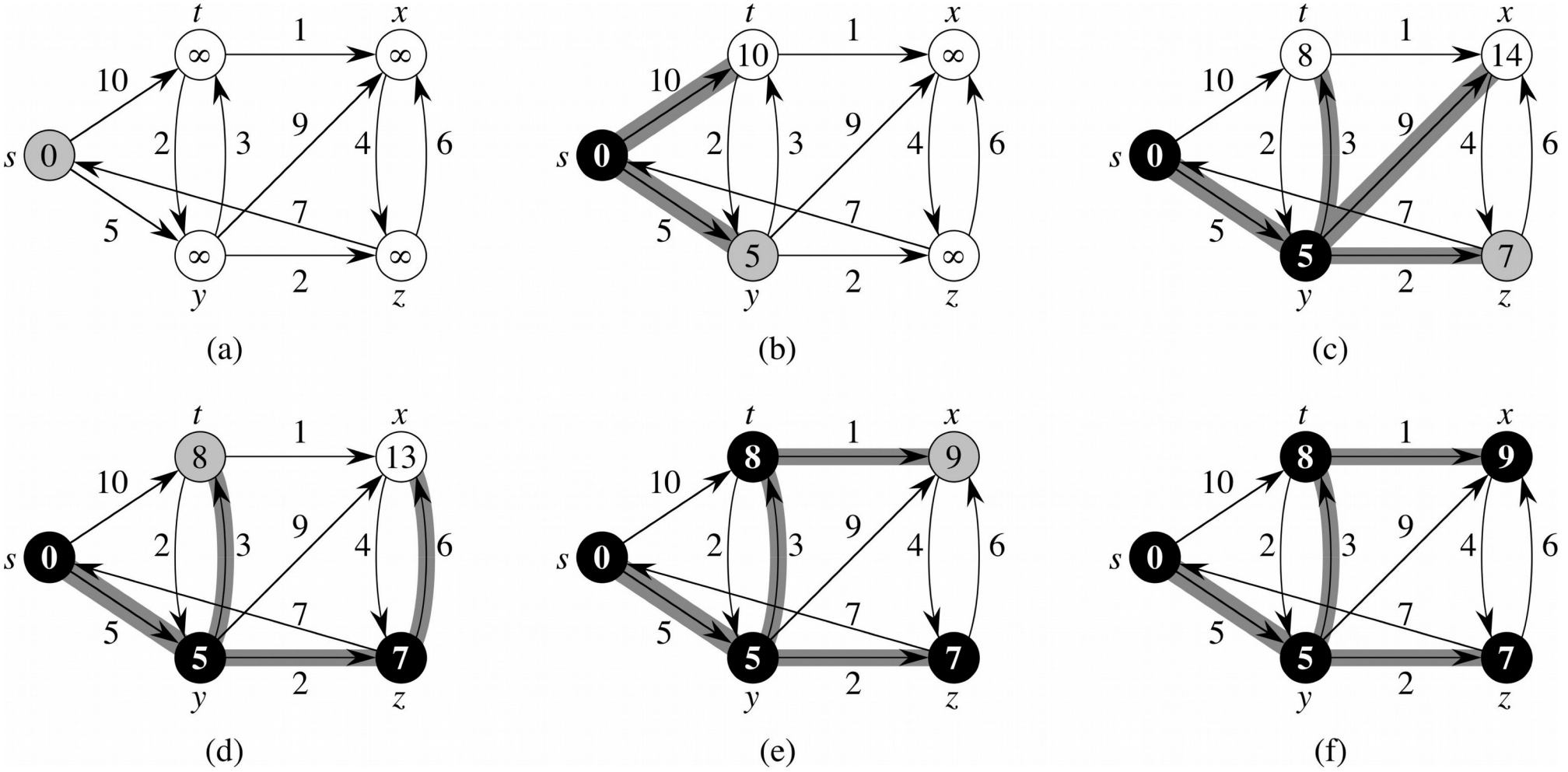

```
                s=0    y=1     t=2     x=3     z=4
funnet   =   { true, true, false, false, false }
avstand  =   {  0,     5,      10,     ∞,      ∞    }
```

Dijkstras algoritme: Animasjon

(finner korteste vei fra **a** til **b)**



Dijkstras algoritme: Eksempel

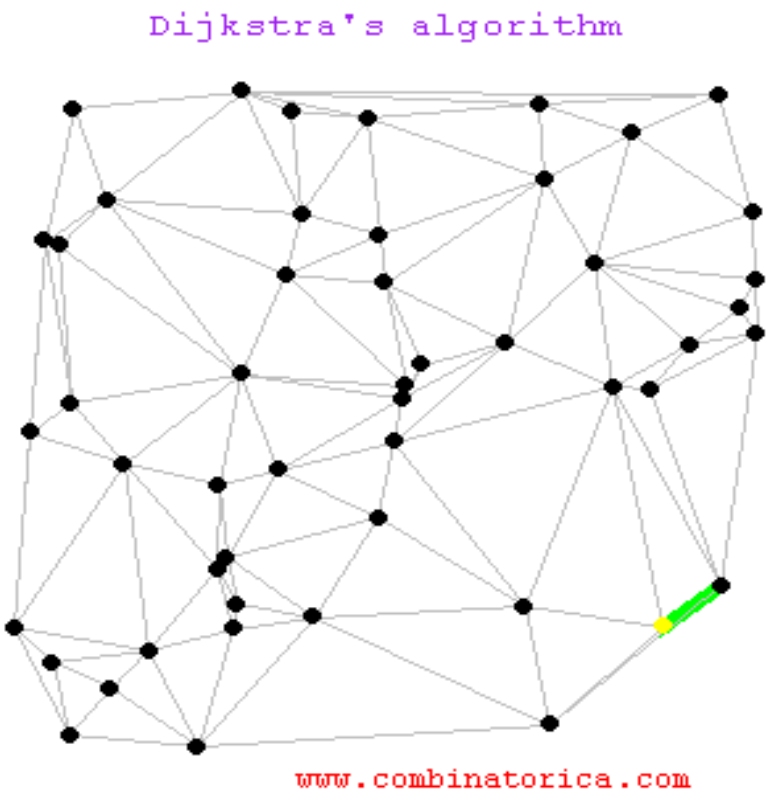


Dijkstra: Algoritme med start i S

```
funnet[i]  = false, i = 0, 1, 2,... , n – 1
avstand[i] = ∞,     i = 0, 1, 2,... , n – 1
avstand[S] = 0, denne = S, antall = 0

while (antall < n)
{
  denne = (noden som har minst verdi av avstand og
           der funnet[denne] == false)
  for (alle naboer i til denne, der funnet[i] == false)
  {
     l = avstand[denne] + kantLengde[denne][i]
     if (l < avstand[i])
        avstand[i] = l
  }
  funnet[denne] = true
  antall++
}
```

Dijkstras algoritme: Animasjon



Kantene i figuren er tegnet “like lange som” vekten/kantlengden

Mer animasjon av Dijkstra

- 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

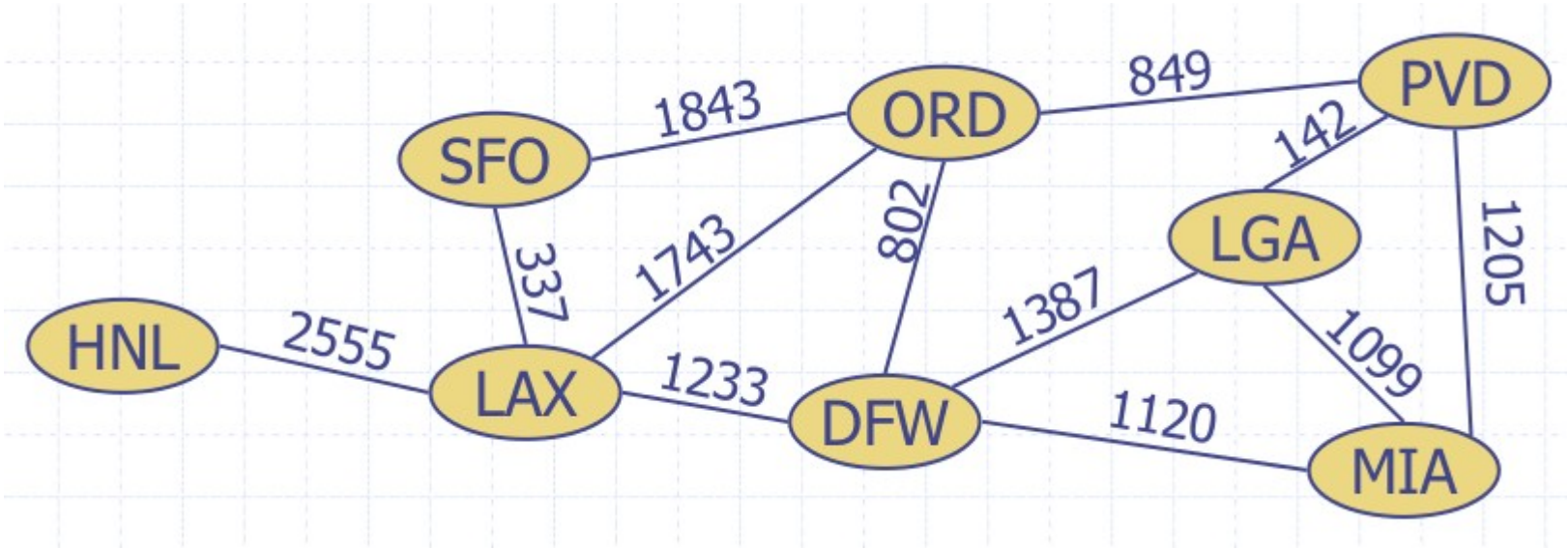
- Dijkstra Shortest Path

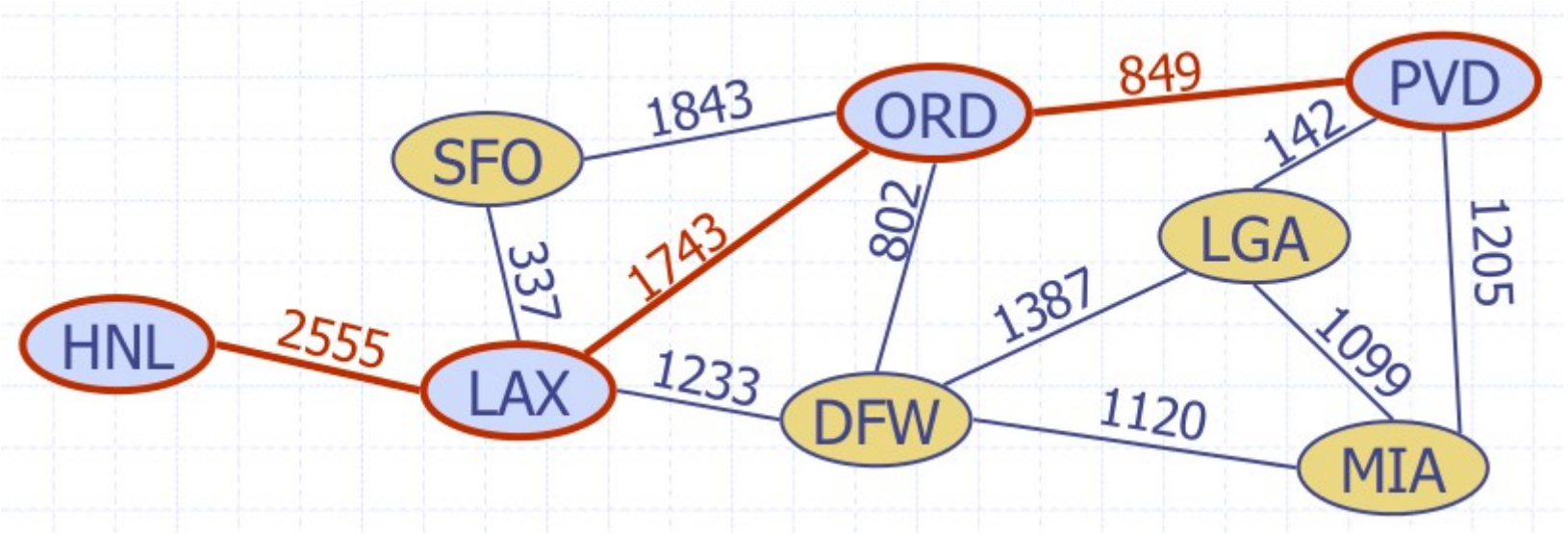
Dijkstra: Effektivitet og implementasjon

- Er $O(n^2)$ i "originalutgaven", for en graf med n noder
- Blir $O(n \log n)$ med smartere koding (øvingsoppgave):
  - Bruk en *min-heap* for å lagre avstandene
  - Søking etter noden med minst avstand blir da $O(\log n)$
  - Er litt fiklete fordi vi må *justere* heapen hver gang avstander oppdateres i grafen
- Dijkstra kan også løse all-pairs shortest path:
  - Start algoritmen én gang fra hver node i grafen
- Enkel $O(n^2)$ implementasjon: `dijkstra.java`

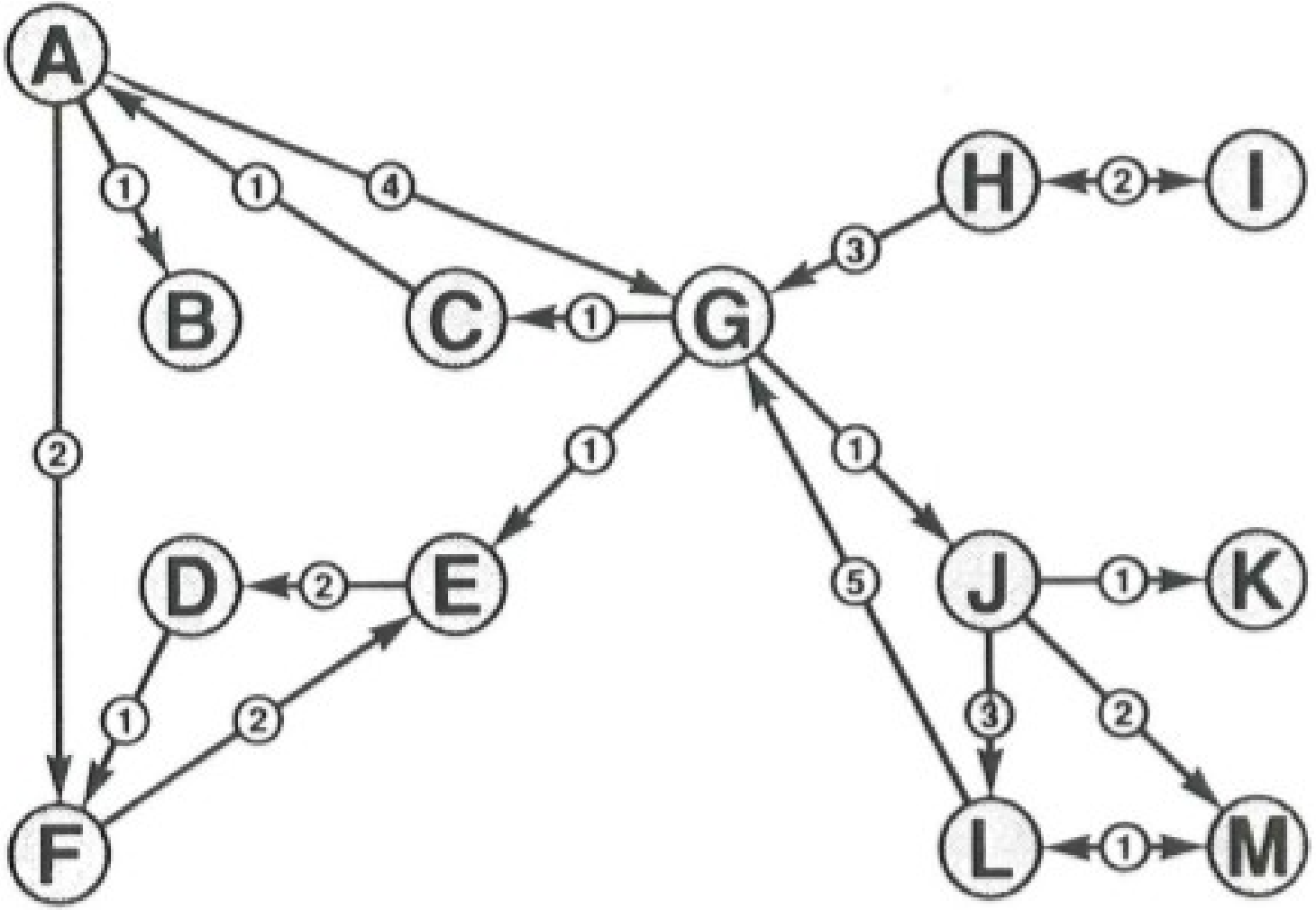
Dijkstra: Lagring av korteste veier

- Et problem med denne versjonen av Dijkstra er at det *bare* er lengden på de korteste veiene til hver node som tas vare på
- Ofte vil vi også ha behov for å vite *hvilke* noder den korteste veien går innom (se eksempel neste side)
- Øvingsoppgave: Programmering av en versjon av Dijkstra som *både* lagrer de korteste avstandene *og* veiene


SFO
ORD
PVD
HNL
LAX
DFW
LGA
MIA
1843
849
142
337
1743
802
1205
2555
1387
1099
1233
1120


SFO
ORD
PVD
HNL
LAX
DFW
LGA
MIA
1843
849
142
337
1743
802
1205
2555
1387
1099
1233
1120

Vektet, rettet graf for testing



Fil: vgraf_13.txt