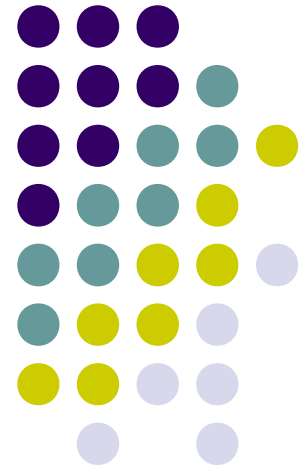
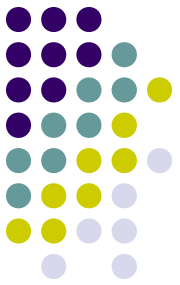


Øvingsforelesning 7

Dijkstras algoritme

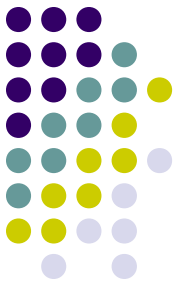
Foiler: Fredrik Ludvigsen &
Håkon Jacobsen
Foreleser: Håkon Jacobsen





Korteste sti - hvorfor?

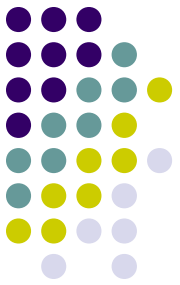
- Eksempel på bruk
 - Routing i internett
 - GPS-systemer
 - Bilde-krymping
 - Simulering av kombinasjonskostnader
 - + mye, mye mer



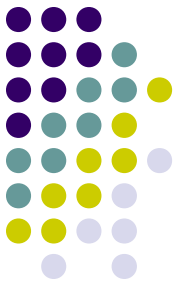
Korteste sti med dijkstra – hvordan?

- Dijkstras algoritme følger en tankegang som kan ligne på BFS.
- Mer korrekt er egentlig Dijkstra en dynamisk programmeringsalgoritme.
 - Men dette er kanskje ikke så lett å få øye på
 - Presenteres normalt som en grådig algoritme

Dijkstras algoritme i én setning

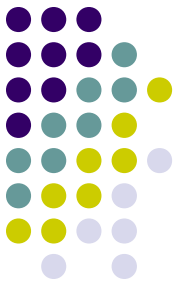


Velg noder med (minste) økende avstand fra utgangsnoden, helt til målet er nådd.



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```



Dijkstras algoritme

INITIALIZE-SINGLE-SOURCE(G, s)

$S = \emptyset$

$Q = G.V$

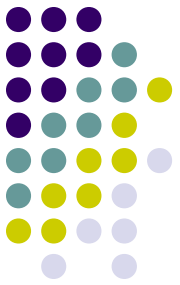
while $Q \neq \emptyset$

do $u = \text{EXTRACT-MIN}(Q)$

$S = S \cup \{u\}$

for each vertex $v \in G.\text{Adj}[u]$

do RELAX(u, v, w)



Dijkstras algoritme

sett alle estimater til ∞
sett startnodens estimat til 0

S er en tom liste

Q er en prioritetskø

legg alle noder inn i Q

så lenge Q ikke er tom:

 sett u til den "korteste" noden i Q

 fjern u fra Q

 legg u til de kjente nodene

for hver nabo v av u :

hvis u kan tilby en kortere sti til v :

 oppdater v sitt estimat

 sett u som v sin forgjenger

INITIALIZE-SINGLE-SOURCE(G, s)

$S = \emptyset$

$Q = G.V$

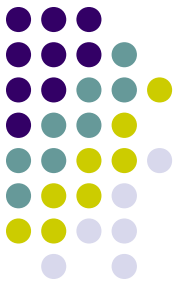
while $Q \neq \emptyset$

do $u = \text{EXTRACT-MIN}(Q)$

$S = S \cup \{u\}$

for each vertex $v \in G.\text{Adj}[u]$

do $\text{RELAX}(u, v, w)$



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
```

```
S =  $\emptyset$ 
```

```
Q =  $G.V$ 
```

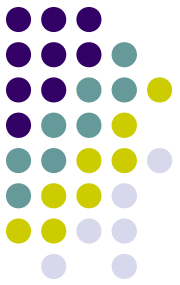
```
while Q  $\neq \emptyset$ 
```

```
    do u = EXTRACT-MIN(Q)
```

```
    S = S  $\cup$  {u}
```

```
    for each vertex v  $\in$  G.Adj[u]
```

```
        do RELAX(u, v, w)
```



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
```

```
 $S = \emptyset$ 
```

```
 $Q = G.V$ 
```

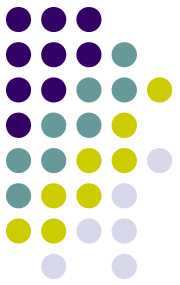
```
while  $Q \neq \emptyset$ 
```

```
    do  $u = \text{EXTRACT-MIN}(Q)$ 
```

```
     $S = S \cup \{u\}$ 
```

```
    for each vertex  $v \in G.\text{Adj}[u]$ 
```

```
        do  $\text{RELAX}(u, v, w)$ 
```



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
```

```
 $S = \emptyset$ 
```

```
 $Q = G.V$ 
```

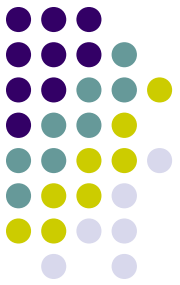
```
while  $Q \neq \emptyset$ 
```

```
    do  $u = \text{EXTRACT-MIN}(Q)$ 
```

```
     $S = S \cup \{u\}$ 
```

```
    for each vertex  $v \in G.\text{Adj}[u]$ 
```

```
        do  $\text{RELAX}(u, v, w)$ 
```



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
```

```
 $S = \emptyset$ 
```

```
 $Q = G.V$ 
```

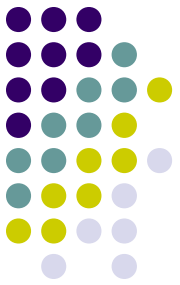
```
while  $Q \neq \emptyset$ 
```

```
    do  $u = \text{EXTRACT-MIN}(Q)$ 
```

```
     $S = S \cup \{u\}$ 
```

```
    for each vertex  $v \in G.\text{Adj}[u]$ 
```

```
        do RELAX( $u, v, w$ )
```



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
```

```
 $S = \emptyset$ 
```

```
 $Q = G.V$ 
```

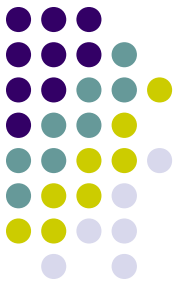
```
while  $Q \neq \emptyset$ 
```

```
    do  $u = \text{EXTRACT-MIN}(Q)$ 
```

```
     $S = S \cup \{u\}$ 
```

```
    for each vertex  $v \in G.\text{Adj}[u]$ 
```

```
        do  $\text{RELAX}(u, v, w)$ 
```



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
```

```
S =  $\emptyset$ 
```

```
Q = G.V
```

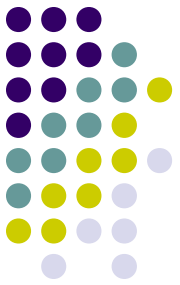
```
while Q  $\neq \emptyset$ 
```

```
    do u = EXTRACT-MIN(Q)
```

```
    S = S  $\cup$  {u}
```

```
    for each vertex v  $\in$  G.Adj[u]
```

```
        do RELAX(u, v, w)
```



Dijkstras algoritme

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
```

```
 $S = \emptyset$ 
```

```
 $Q = G.V$ 
```

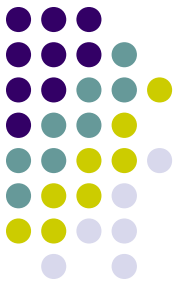
```
while  $Q \neq \emptyset$ 
```

```
    do  $u = \text{EXTRACT-MIN}(Q)$ 
```

```
     $S = S \cup \{u\}$ 
```

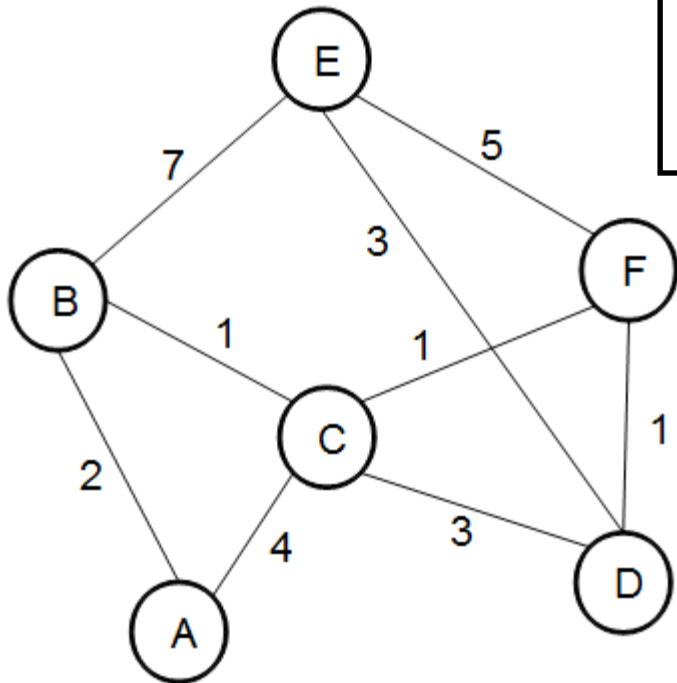
```
    for each vertex  $v \in G.\text{Adj}[u]$ 
```

```
        do RELAX( $u, v, w$ )
```



Dijkstras algoritme – eksempel, andre gang

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )  
 $S = \emptyset$   
 $Q = G.V$   
while  $Q \neq \emptyset$   
  do  $u = \text{EXTRACT-MIN}(Q)$   
   $S = S \cup \{u\}$   
  for each vertex  $v \in G.\text{Adj}[u]$   
    do RELAX( $u, v, w$ )
```

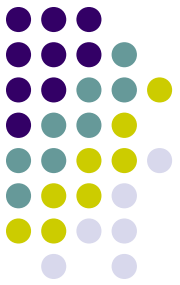


10/05/10

		Q
A	0	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	

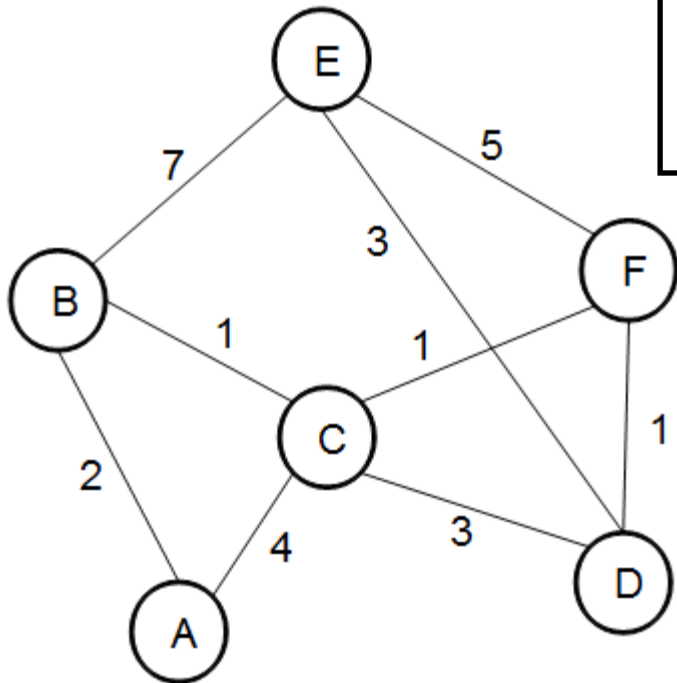
		S

15

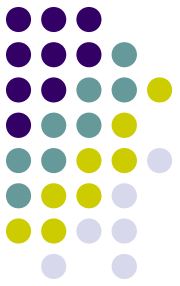


Dijkstras algoritme – eksempel, andre gang

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )  
 $S = \emptyset$   
 $Q = G.V$   
while  $Q \neq \emptyset$   
  do  $u = \text{EXTRACT-MIN}(Q)$   
   $S = S \cup \{u\}$   
  for each vertex  $v \in G.\text{Adj}[u]$   
    do RELAX( $u, v, w$ )
```



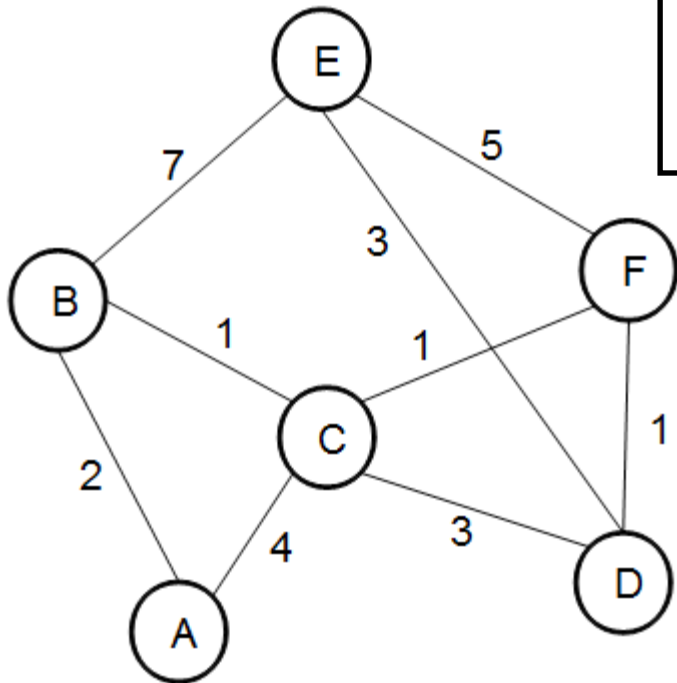
Q			S		
			A	0	*
B	∞				
C	∞				
D	∞				
E	∞				
F	∞				



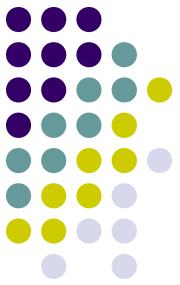
Dijkstras algoritme – eksempel, andre gang

```

INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
  
```

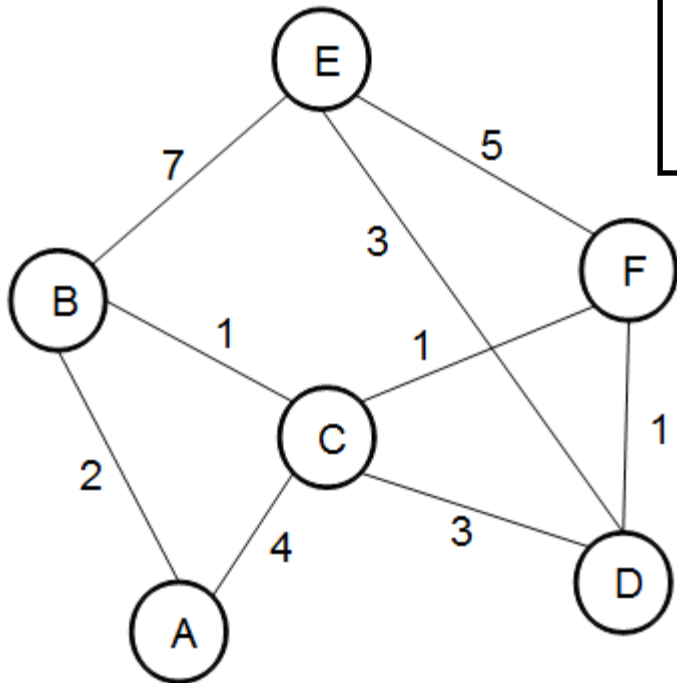


Q			S		
			A	0	*
B	2	A			
C	4	A			
D	∞				
E	∞				
F	∞				

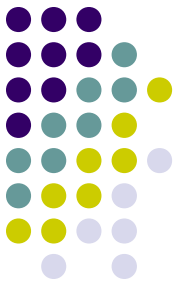


Dijkstras algoritme – eksempel, andre gang

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )  
 $S = \emptyset$   
 $Q = G.V$   
while  $Q \neq \emptyset$   
  do  $u = \text{EXTRACT-MIN}(Q)$   
   $S = S \cup \{u\}$   
  for each vertex  $v \in G.\text{Adj}[u]$   
    do RELAX( $u, v, w$ )
```

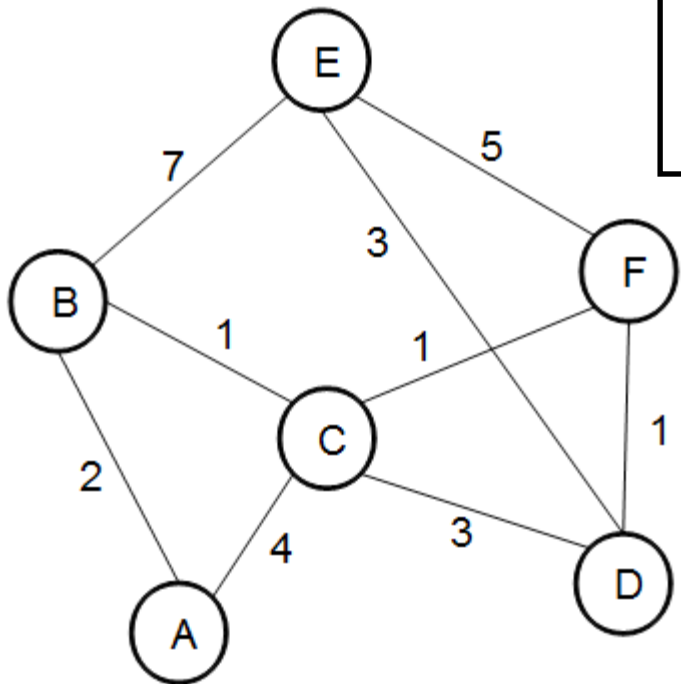


Q			S		
			A	0	*
			B	2	A
C	4	A			
D	∞				
E	∞				
F	∞				



Dijkstras algoritme – eksempel, andre gang

```
INITIALIZE-SINGLE-SOURCE( $G, s$ )  
 $S = \emptyset$   
 $Q = G.V$   
while  $Q \neq \emptyset$   
  do  $u = \text{EXTRACT-MIN}(Q)$   
   $S = S \cup \{u\}$   
  for each vertex  $v \in G.\text{Adj}[u]$   
    do RELAX( $u, v, w$ )
```



Q			S		
			A	0	*
			B	2	A
C	3	B			
D	∞				
E	9	B			
F	∞				

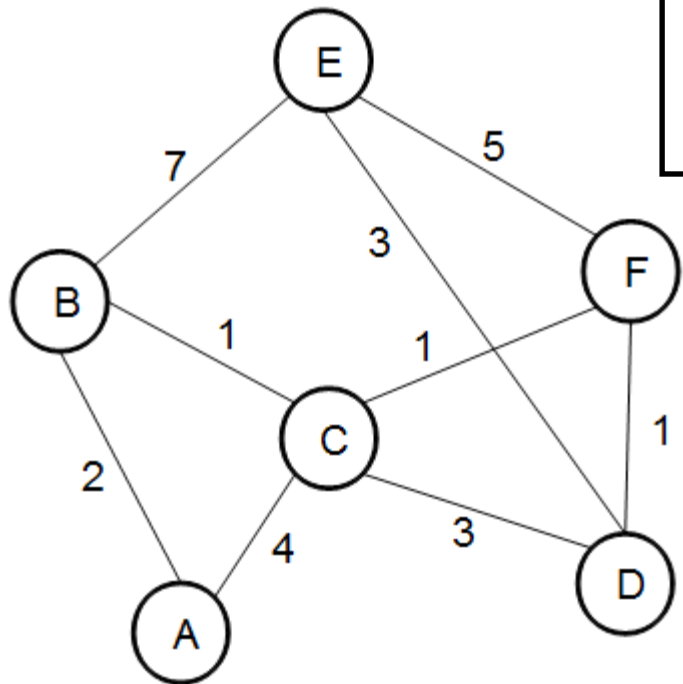




Dijkstras algoritme – eksempel, andre gang

```

INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
  
```



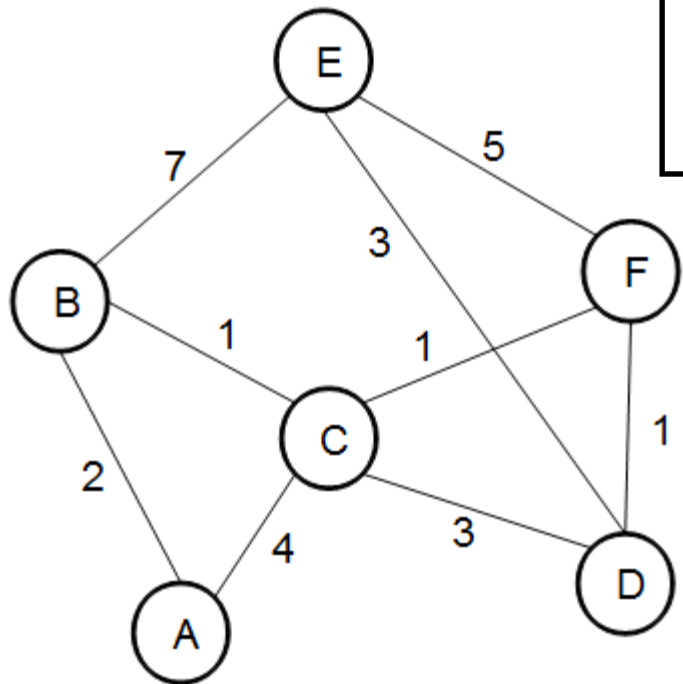
	Q	S
		A 0 *
		B 2 A
		C 3 B
D	∞	
E	9	B
F	∞	



Dijkstras algoritme – eksempel, andre gang

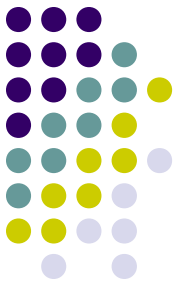
```

INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
  
```



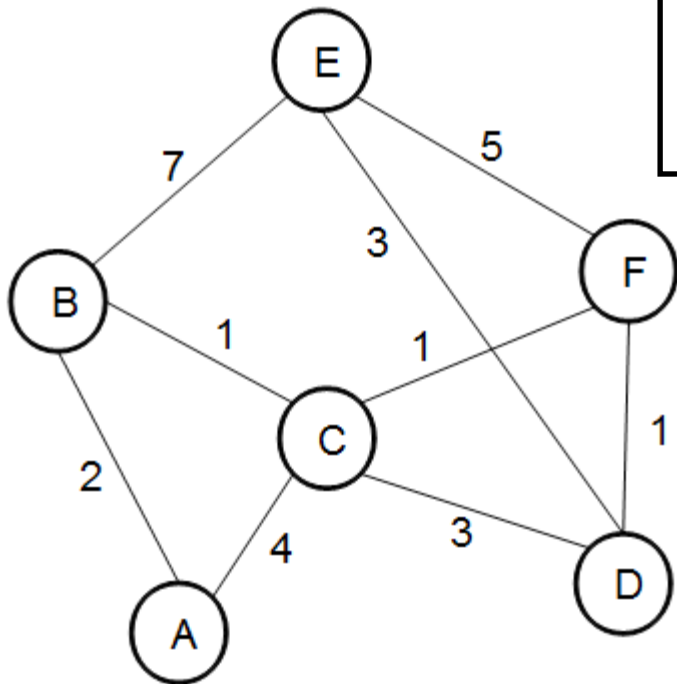
Q			S		
			A	0	*
			B	2	A
			C	3	B
D	6	C			
E	9	B			
F	4	C			

Dijkstras algoritme – eksempel, andre gang

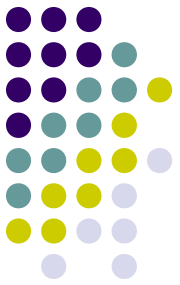


```

INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
    
```



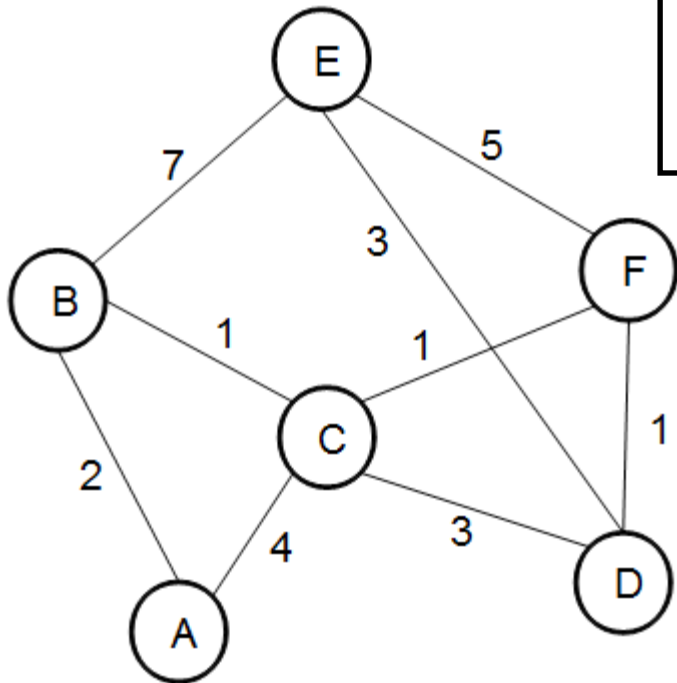
Q			S		
			A	0	*
			B	2	A
			C	3	B
			F	4	C
D	6	C			
E	9	B			



Dijkstras algoritme – eksempel, andre gang

```

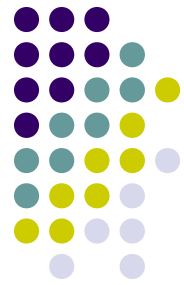
INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
  
```



Q			S		
			A	0	*
			B	2	A
			C	3	B
			F	4	C
D	5	F			
E	9	B			

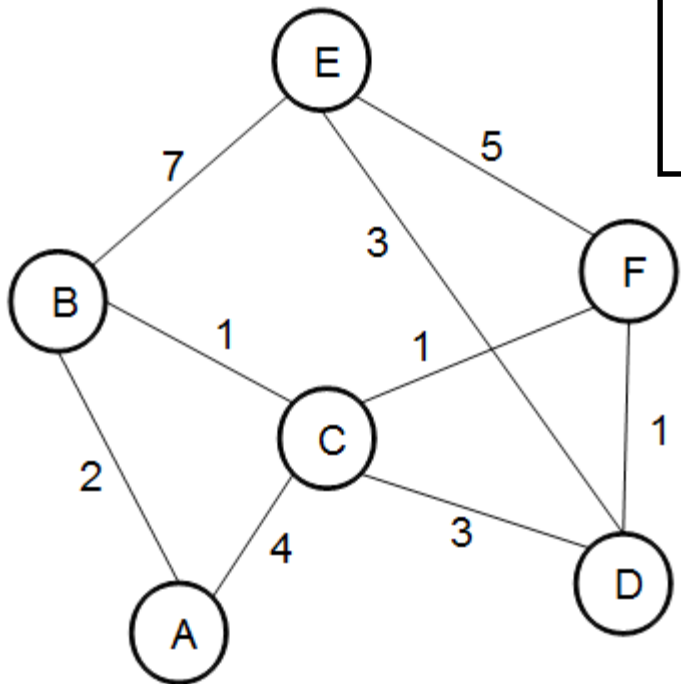


Dijkstras algoritme – eksempel, andre gang



```

INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
  
```



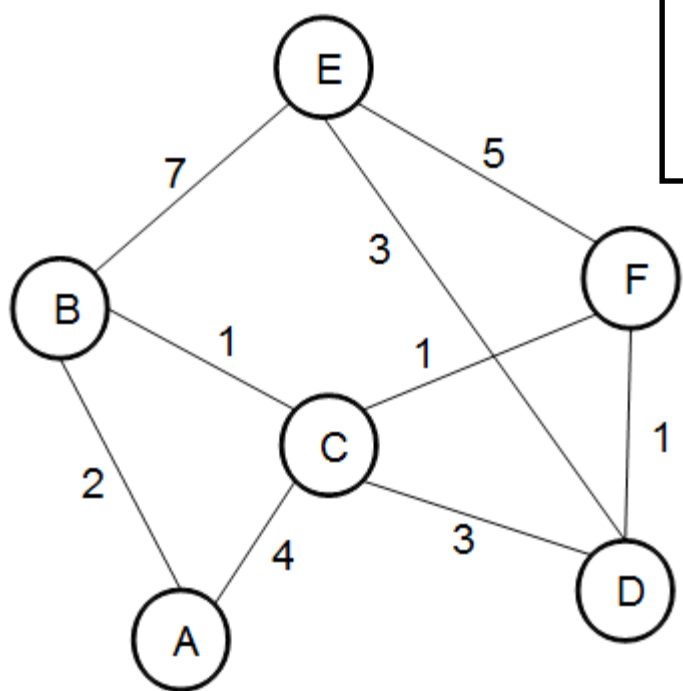
Q			S		
			A	0	*
			B	2	A
			C	3	B
			F	4	C
			D	5	F
E	9	B			



Dijkstras algoritme – eksempel, andre gang

```

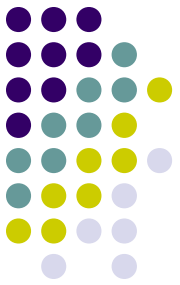
INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
  
```



Q			S		
			A	0	*
			B	2	A
			C	3	B
			F	4	C
			D	5	F
E	8	D			

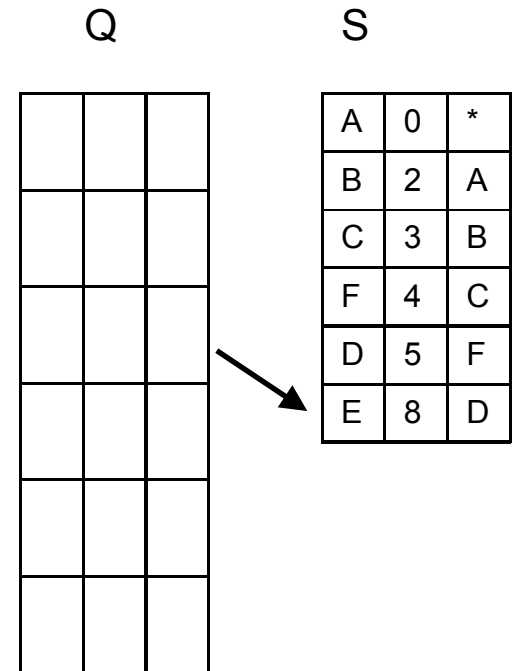
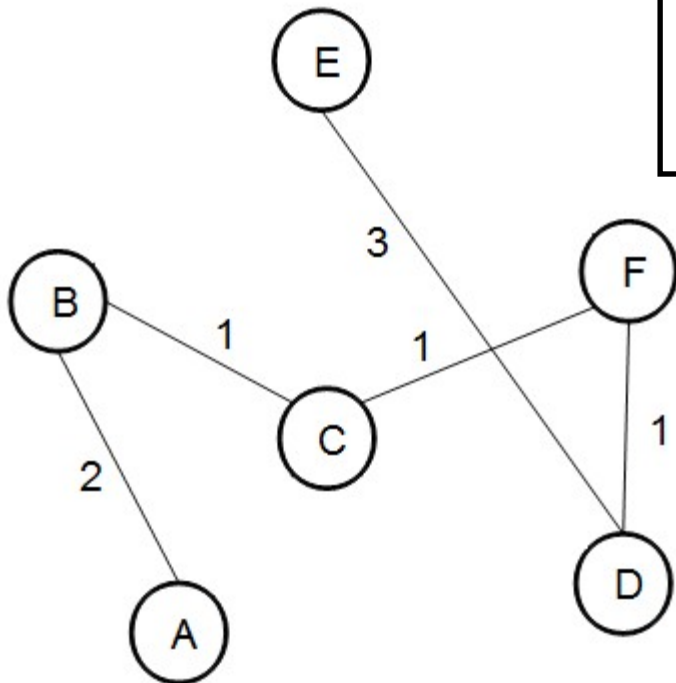


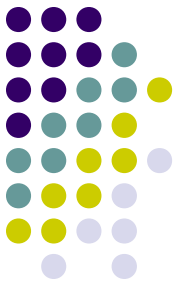
Dijkstras algoritme – eksempel, andre gang



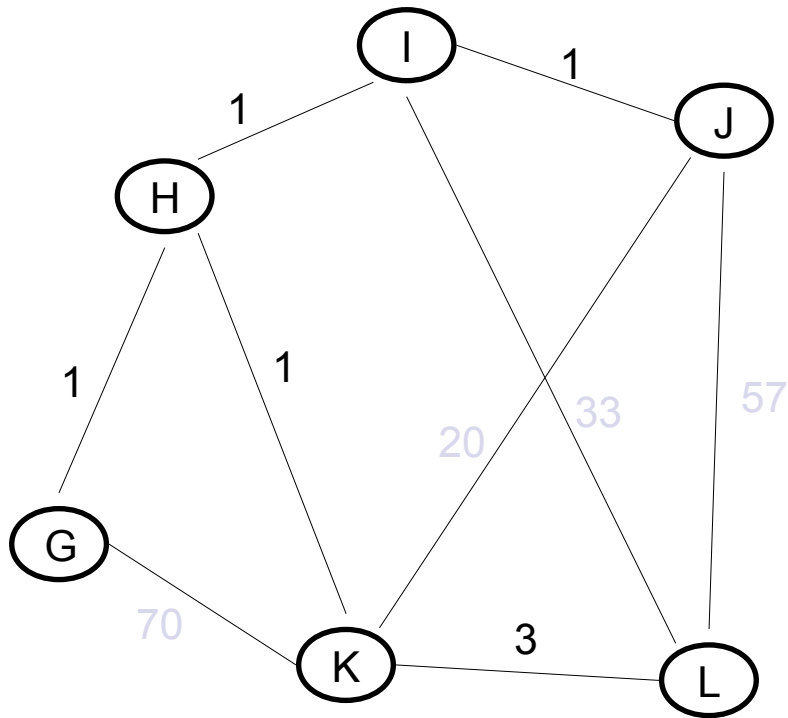
```

INITIALIZE-SINGLE-SOURCE( $G, s$ )
 $S = \emptyset$ 
 $Q = G.V$ 
while  $Q \neq \emptyset$ 
  do  $u = \text{EXTRACT-MIN}(Q)$ 
   $S = S \cup \{u\}$ 
  for each vertex  $v \in G.\text{Adj}[u]$ 
    do RELAX( $u, v, w$ )
    
```

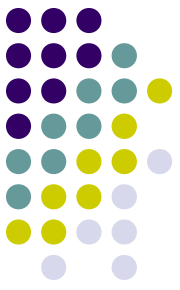




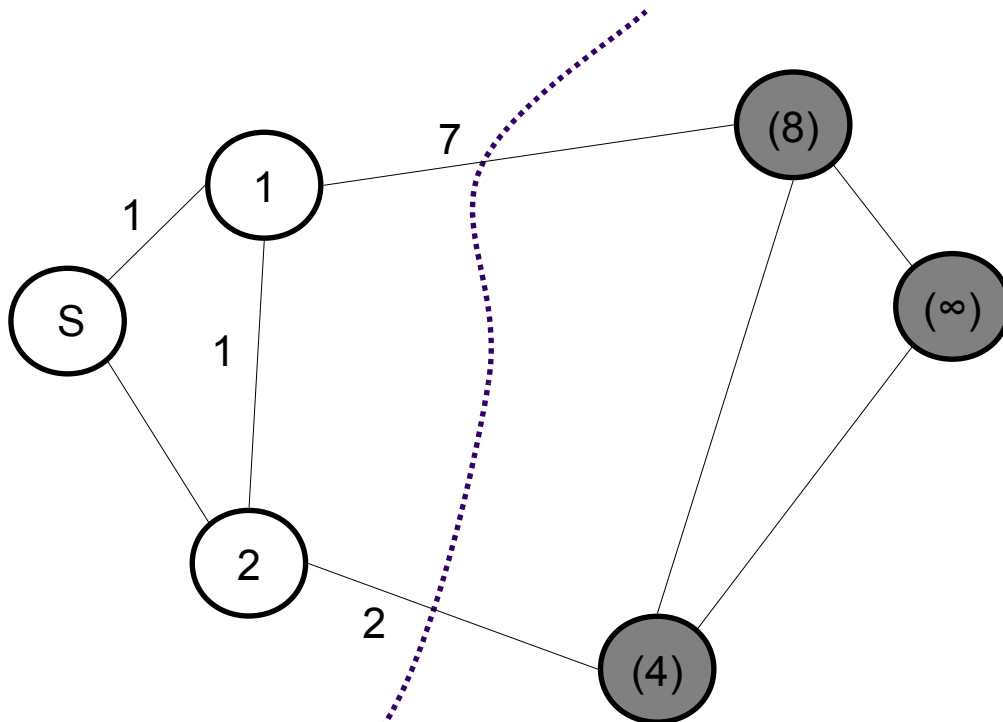
Dijkstras algoritme



- I forrige eksempel lå alle nodene etter hverandre på den korteste stien
- Dijkstra finner korteste vei "en-til-alle", så resultatet er et tre.



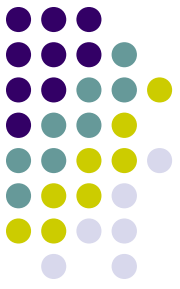
Dijkstras algoritme - korrekthet



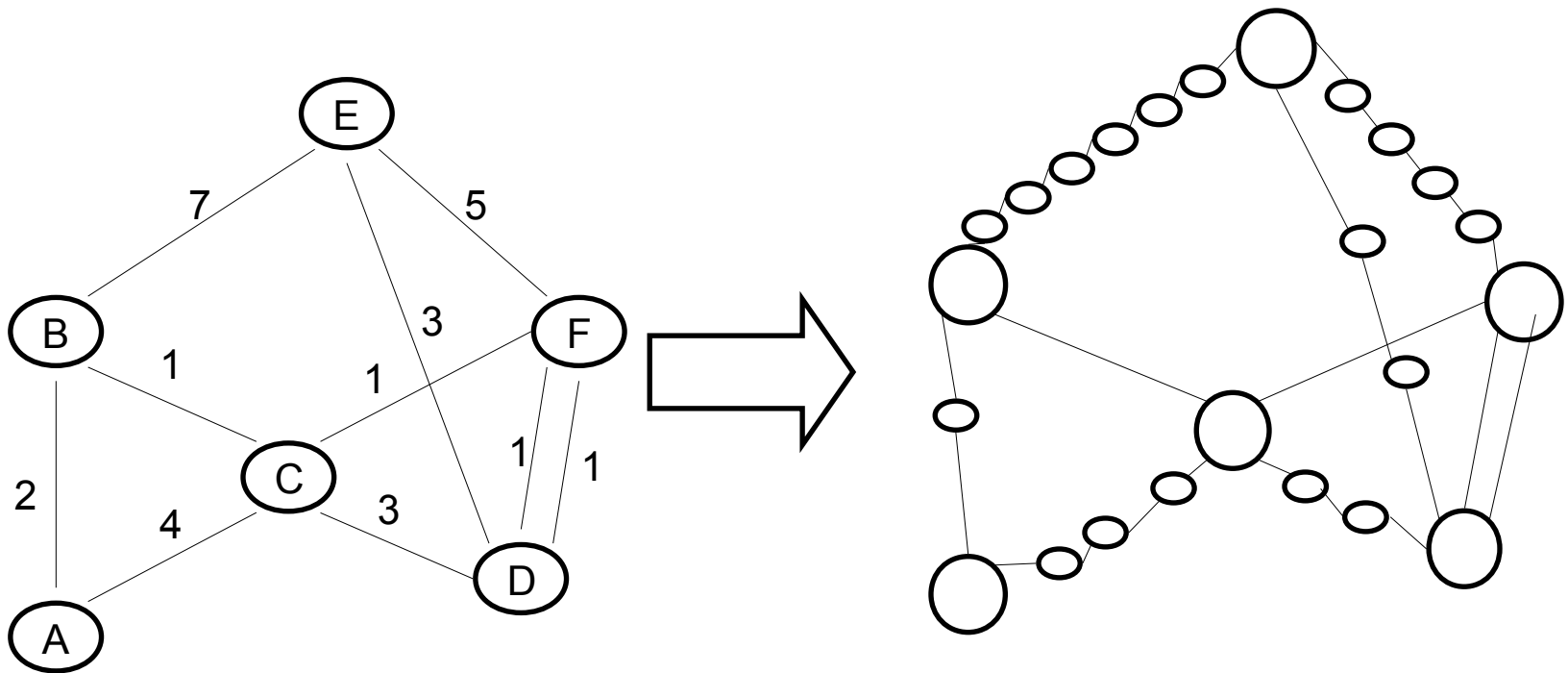
Tankegangen her likner tankegangen i prim's algoritme, men nå velges nye noder ut ifra avstand til startnoden, ikke bare avstand til en hvilken som helst del av treeet.

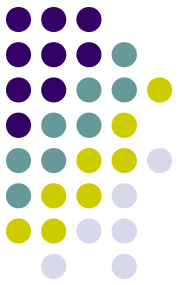
Hver gang vi velger en ny node u vet vi at avstanden til den er riktig! Vi vet at avstanden til alle de andre nodene i S er riktig, og vi kan ikke finne en kortere vei til u via bare disse. Så en kortere vei til u må gå via noder i Q . MEN: Alle disse har avstand større enn eller lik avstanden til u , og vi antar at det ikke fins negative kanter, så da kan ikke en vei gjennom Q være kortere likevel :)

Hvis det finnes negative kanter fungerer verken algoritmen eller beviset.



Dijkstras algoritme – alternativ tankegang, bruk av BFS

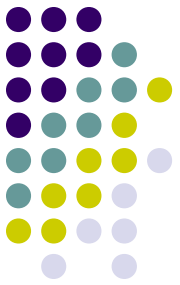




Dijkstras algoritme – valg av prioritetskø

```
1 INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  $S = \emptyset$ 
3  $Q = G.V$ 
4 while  $Q \neq \emptyset$ 
5     do  $u = \text{EXTRACT-MIN}(Q)$ 
6      $S = S \cup \{u\}$ 
7     for each vertex  $v \in G.\text{Adj}[u]$ 
8         do  $\text{RELAX}(u, v, w)$ 
```

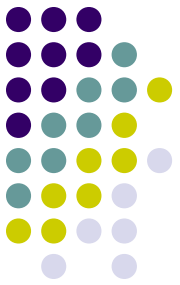
- Konstruksjon av Q med $|V|$ elementer (3)
- **while**-løkka kjøres totalt $|V|$ ganger
 - Ett uttak fra Q (5)
- **for**-løkka kjøres totalt $|E|$ ganger (7)
 - I verste fall én oppdatering i Q for hver kant (8)



Dijkstras algoritme – valg av prioritetskø

- Vi skal gjøre følgende:
- 1: Konstuere prioritetskø
 - 2: $|V|$ antall extract-min fra køen
 - 3: $|E|$ antall oppdateringer i køen (til lavere verdi)

	Sortert Array	Usortert Array	(Min) Heap	Fibonacci Heap
konstruer	$O(n \log n)$	$O(n)$	$O(n)$	$O(n)$
extract-min	$O(1)$	$O(n)$	$O(\log n)$	$O(\log n)^*$
decreasekey	$O(n)$	$O(1)$	$O(\log n)$	$O(1)^*$
totalt	$ V \log V + V + E * V $	$ V + V ^2 + E $	$ V + V \log V + E \log V $	$ V + V \log V + E $
forenklet	$ E * V $	$ V ^2$	$ E \log V $	$ V \log V + E $



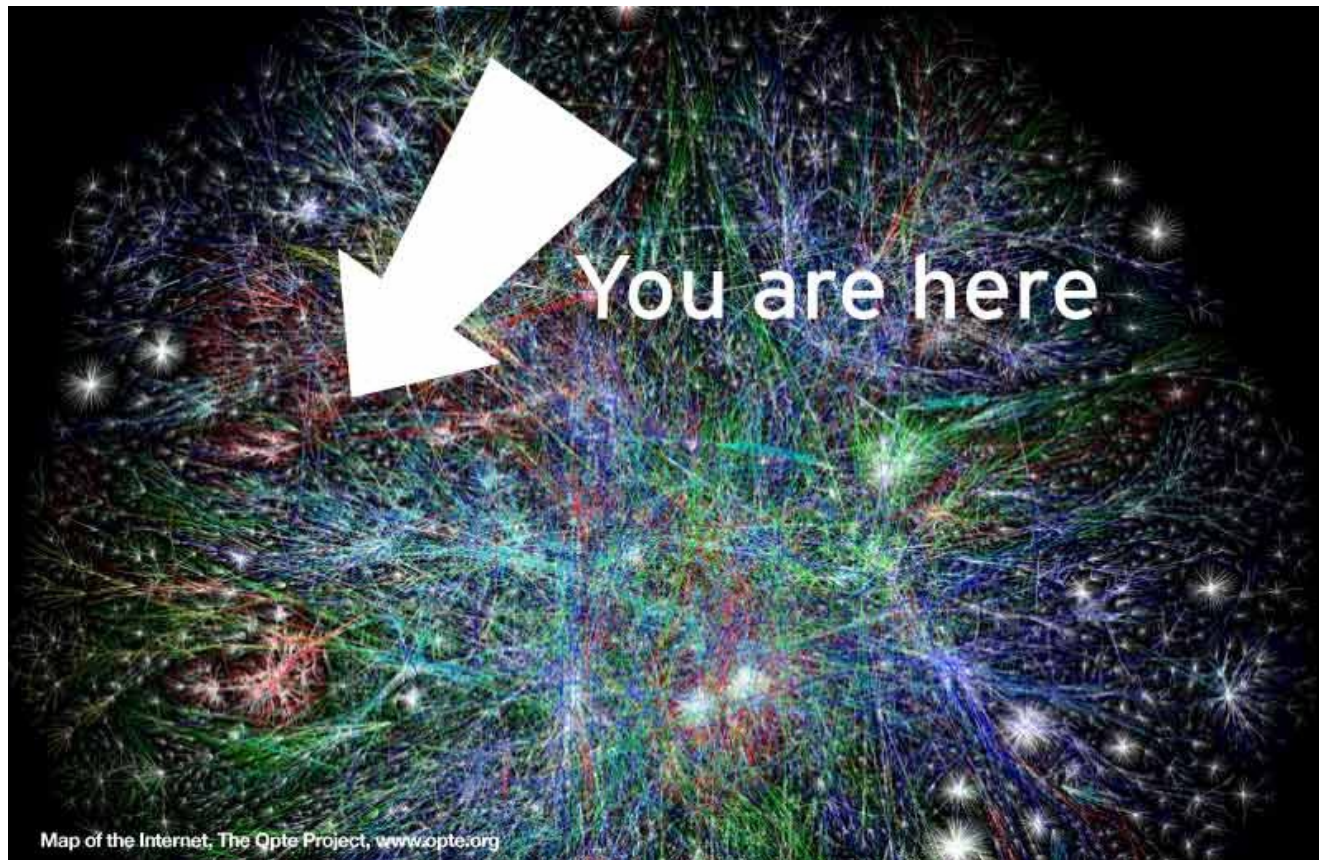
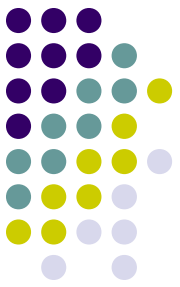
Dijkstras algoritme – valg av prioritetskø

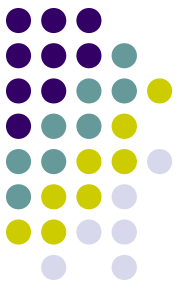
Når vi sammenligner kjøretidene til dijkstras algoritme ved bruk av usortert array og min-heap

$$|V|^2 \quad |E| \log |V|$$

ser vi at dersom $|E|$ nærmer seg $|V|^2$ vil det lønne seg å bruke usortert array.

Dijkstras algoritme i praksis - routing i internett



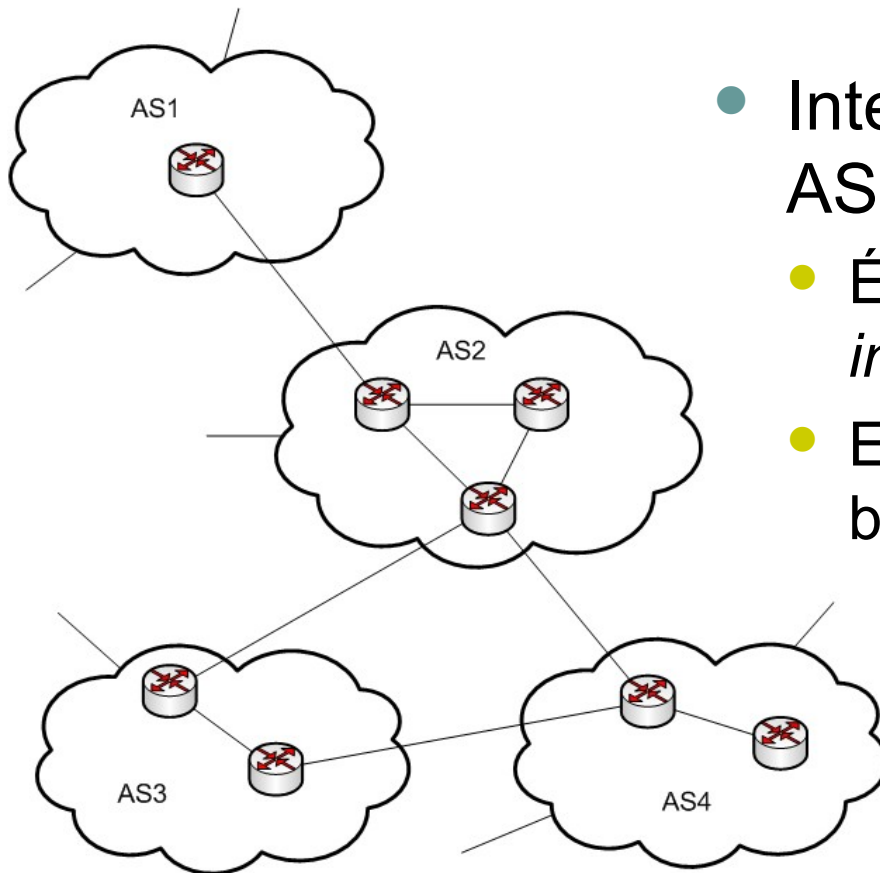


Routing i internett

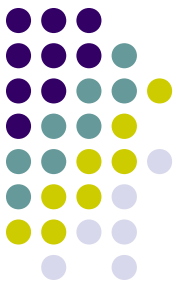
- Basert på IP-adresser
- Routere finner riktig vei for pakkene
 - Basert på tabeller i routerne
 - Tabellene generert av routing-algoritmer
- IPv4 benytter 32-bit til adressering
 - Dvs ~4 mrd enheter kan adresseres unikt
 - Umulig for én router å vite beste vei til alle andre



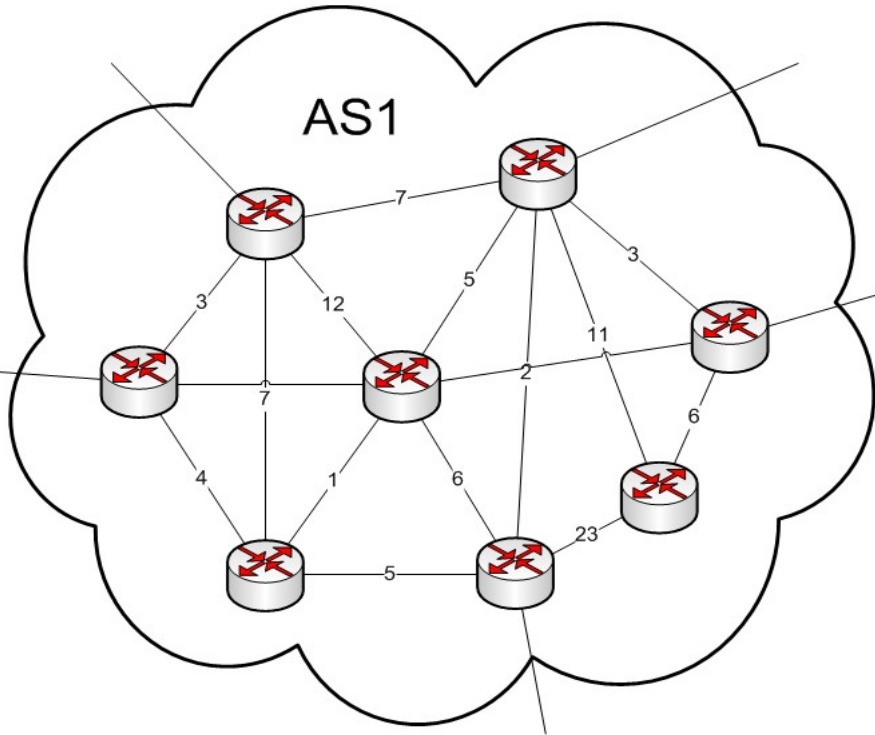
Routing i internett – Autonome systemer (AS)



- Internett deles opp i mange AS
 - Én routing-algoritme benyttes *internt* i et AS
 - En (annen) routing-algoritme benyttes *mellom* AS

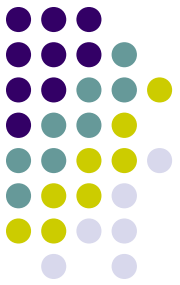


Routing i internett – Open Shortest Path First (OSPF)

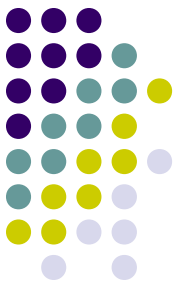


- En av de mest utbredte routing-protokollene innad i et AS
- Hver router averterer sine linkkostnader (dvs båndbredde) til alle andre routere i AS-et
- Etter en stund har alle routere laget seg et kart over nettverket
- Og så...

Routing i internett – Open Shortest Path First (OSPF)

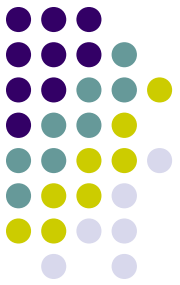


Dijkstra!!

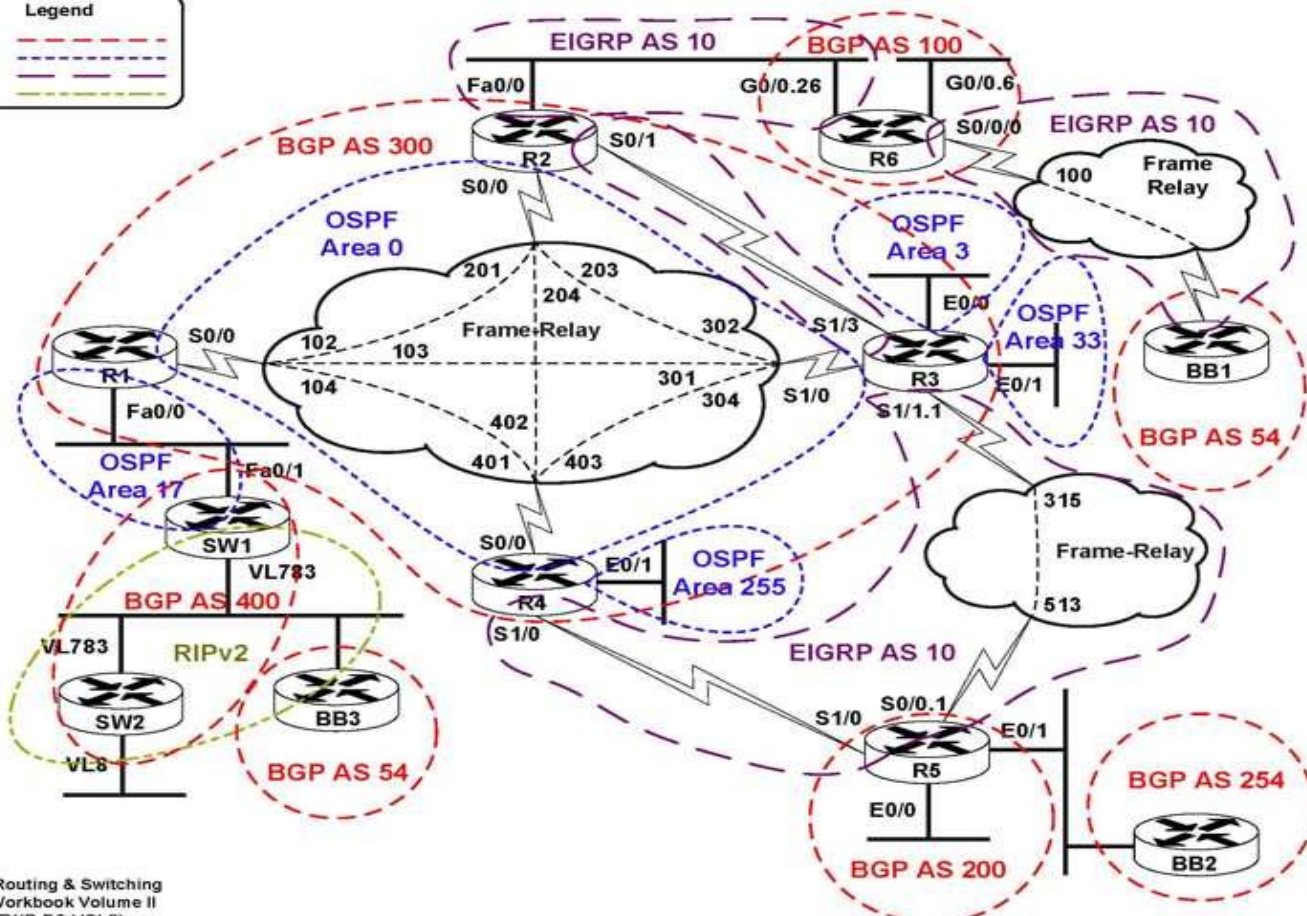
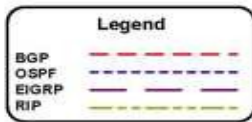


Routing i internett – Open Shortest Path First (OSPF)

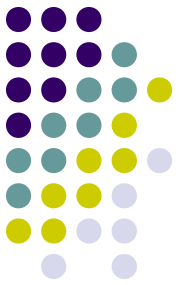
- Hver router kjører Dijkstra med seg selv som startnode
- Siden alle baserer seg på det samme nettverkskartet vil OSPF konvergere mot den beste veien for alle routerne
- Enkelt?



Routing i internett – Open Shortest Path First (OSPF)



1



Oppsummering

- Dijkstra velger alltid noden med minst avstandsestimat fra startnoden
- Hver kant blir kjørt Relax på nøyaktig én gang
- Når en node blir valgt har den fått sin endelige avstand
- Kjøretiden dominert av datastrukturen vi benytter