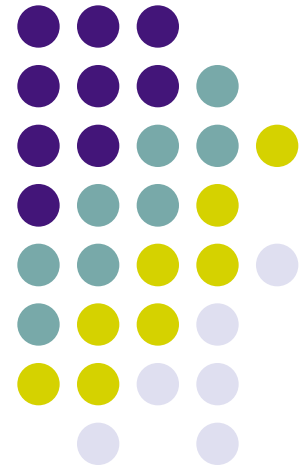


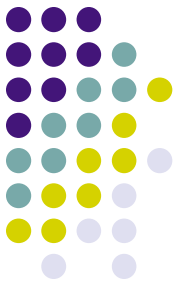
Øvingsforelesning 7

Korteste vei (én til alle)

Forelest av Martin Gammelsæter

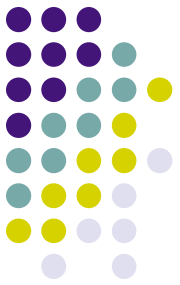
Foiler av Fredrik Ludvigsen, Håkon Jacobsen og
Martin Gammelsæter





I dag

- Neste øving
- Hvorfor korteste sti?
- Dijkstra
- Bellman-Ford
- DAG-Shortest Path
- Forrige øving



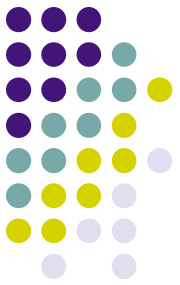
Neste øving

- Pipesortering

- Input: Usortert liste + intervall
- Output: Intervall i listen som er minst like stort som de oppgitte grensene

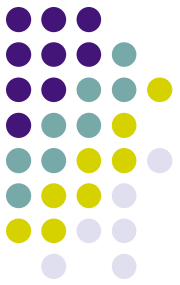
- Tips

- Se på type input og velg en god sorteringsmetode
- Bruk binærsøk to ganger for å finne nedre og øvre
- For å toppe lista: Testsettene inneholder mange forskjellige typer input, ha flere søke- og sorteringsmetoder og bruk de som er optimale for inputen.



Korteste sti - hvorfor?

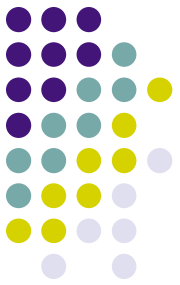
- Eksempel på bruk
 - GPS-systemer
 - Bilde-krymping
 - Routing-protokoller i IP-nettverk (lan, internett etc.)
 - Simulering av kombinasjonskostnader
 - + mye, mye mer



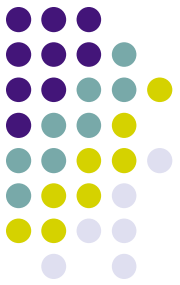
Korteste sti med Dijkstra – hvordan?

- Dijkstras algoritme følger en tankegang som ligner på BFS.
- BFS finner faktisk korteste sti i grafer, hvis man kun ser på antall kanter traversert.
(Hvis du tenker at hver kant har lengde 1)

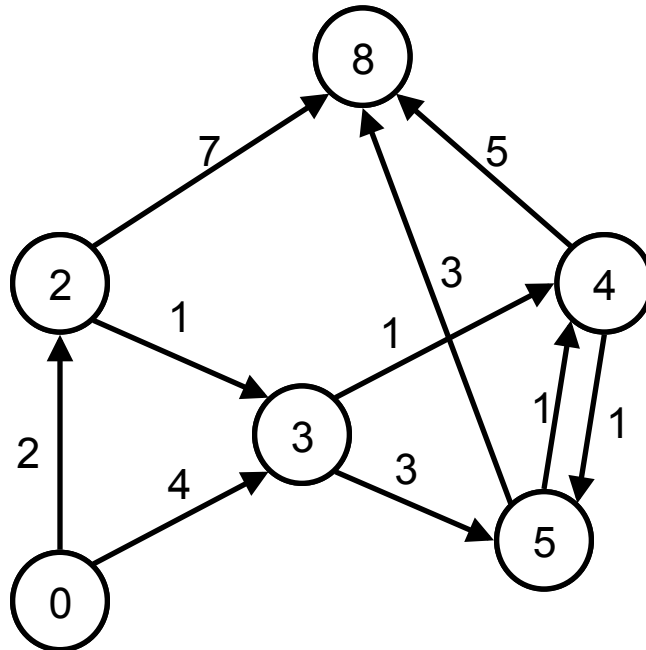
Dijkstras algoritme i én setning

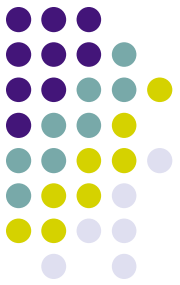


Velg noder med (minste) økende avstand fra utgangsnoden, helt til målet er nådd.



Dijkstras algoritme – eksempel, første gang





Dijkstras algoritme i praksis

INITIALIZE-SINGLE-SOURCE(G, s)

$S \leftarrow \emptyset$

$Q \leftarrow V[G]$

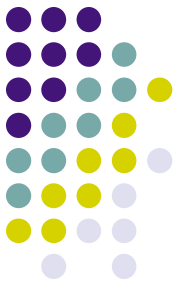
while $Q \neq \emptyset$

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

$S \leftarrow S \cup \{u\}$

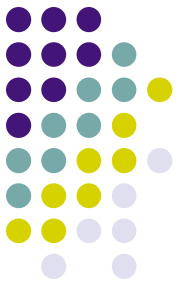
for each vertex $v \in \text{Adj}[u]$

do RELAX(u, v, w)



Dijkstras algoritme i praksis

```
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 i praksis

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)

$S \leftarrow \emptyset$

$Q \leftarrow V[G]$

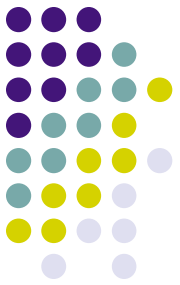
while $Q \neq \emptyset$

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

$S \leftarrow S \cup \{u\}$

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

do RELAX(u, v, w)



Dijkstras algoritme i praksis

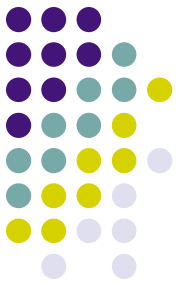
```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere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

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

```
S  $\leftarrow \emptyset$ 
Q  $\leftarrow V[G]$ 

while Q  $\neq \emptyset$ 
    do u  $\leftarrow$  EXTRACT-MIN(Q)

    S  $\leftarrow S \cup \{u\}$ 
    for each vertex v  $\in$  Adj[u]
        do RELAX(u, v, w)
```



Dijkstras algoritme i praksis

```
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( $G, s$ )
```

```
 $S \leftarrow \emptyset$ 
```

```
 $Q \leftarrow V[G]$ 
```

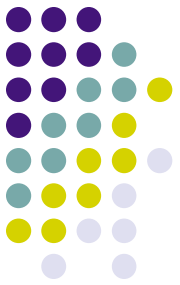
```
while  $Q \neq \emptyset$ 
```

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

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

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

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



Dijkstras algoritme i praksis

```
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( $G, s$ )
```

```
 $S \leftarrow \emptyset$ 
```

```
 $Q \leftarrow V[G]$ 
```

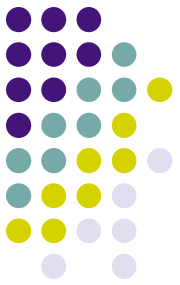
```
while  $Q \neq \emptyset$ 
```

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

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

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

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



Dijkstras algoritme i praksis

```
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( $G, s$ )
```

```
 $S \leftarrow \emptyset$ 
```

```
 $Q \leftarrow V[G]$ 
```

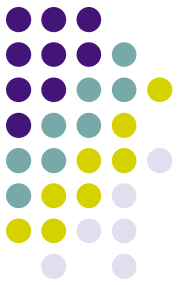
```
while  $Q \neq \emptyset$ 
```

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

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

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

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



Dijkstras algoritme i praksis

```
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( $G, s$ )
```

```
 $S \leftarrow \emptyset$ 
```

```
 $Q \leftarrow V[G]$ 
```

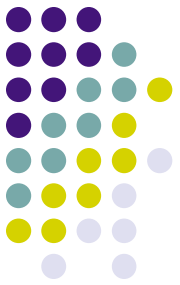
```
while  $Q \neq \emptyset$ 
```

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

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

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

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



Dijkstras algoritme i praksis

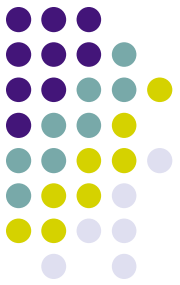
```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
S er en tom liste
Q er en prioritetskø
legg alle noder inn i Q
så lenge Q ikke er tom:
    sett u til den "korteste" noden i Q
    fjern u fra Q
    legg u til de kjente nodene
    for hver nabo v av u:
        hvis u kan tilby en kortere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

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

```
S  $\leftarrow$   $\emptyset$ 
Q  $\leftarrow$  V[G]

while Q  $\neq$   $\emptyset$ 
    do u  $\leftarrow$  EXTRACT-MIN(Q)

    S  $\leftarrow$  S  $\cup$  {u}
    for each vertex v  $\in$  Adj[u]
        do RELAX(u, v, w)
```



Dijkstras algoritme i praksis

```
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( $G, s$ )
```

```
 $S \leftarrow \emptyset$ 
```

```
 $Q \leftarrow V[G]$ 
```

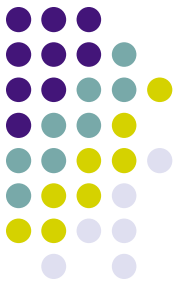
```
while  $Q \neq \emptyset$ 
```

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

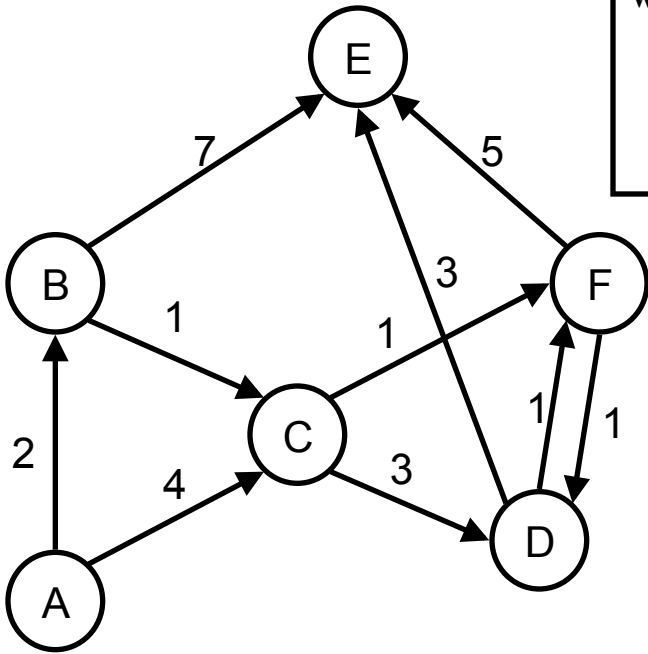
```
     $S \leftarrow S \cup \{u\}$ 
```

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

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



Dijkstras algoritme – eksempel, andre gang

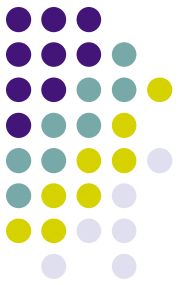


```
INITIALIZE-SINGLE-SOURCE( $G, s$ )  
 $S \leftarrow \emptyset$   
 $Q \leftarrow V[G]$   
while  $Q \neq \emptyset$   
  do  $u \leftarrow \text{EXTRACT-MIN}(Q)$   
   $S \leftarrow S \cup \{u\}$   
  for each vertex  $v \in \text{Adj}[u]$   
    do RELAX( $u, v, w$ )
```

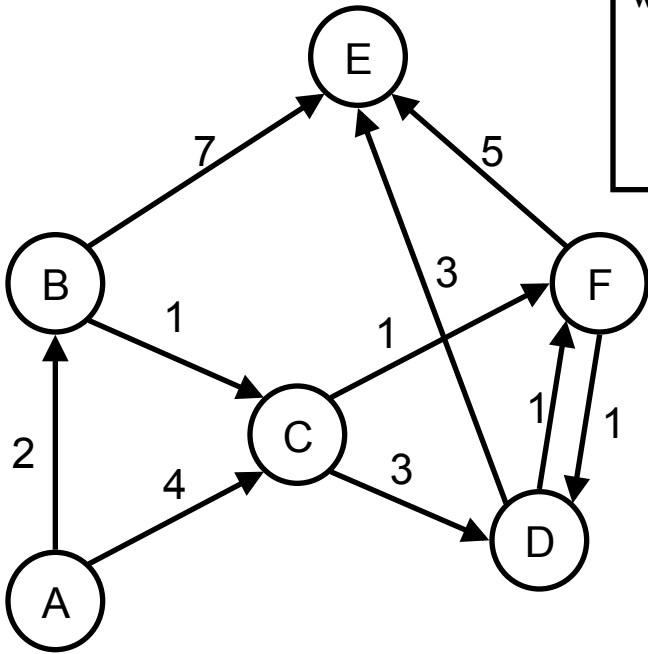
Q

A	0	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	

S

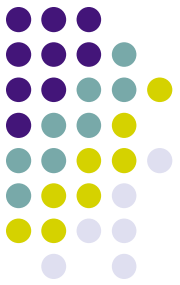


Dijkstras algoritme – eksempel, andre gang



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

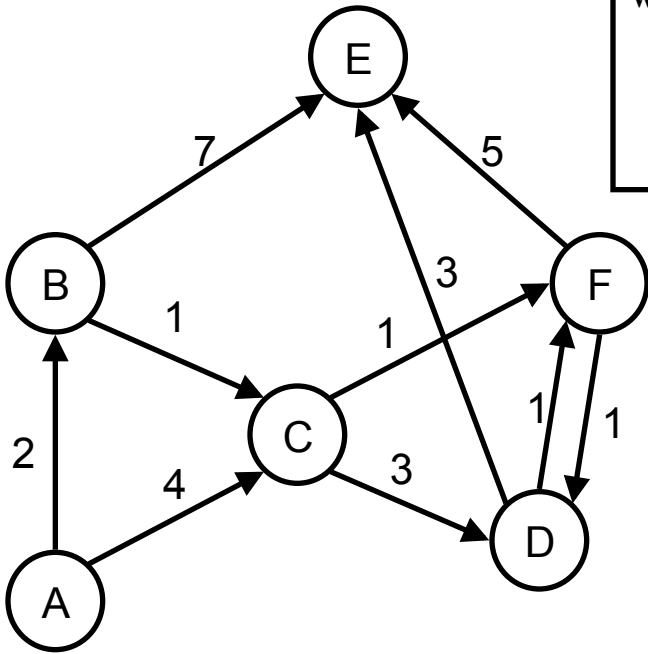
	Q		S
			A 0 *
B	∞		
C	∞		
D	∞		
E	∞		
F	∞		



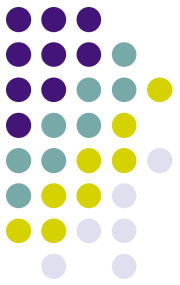
Dijkstras algoritme – eksempel, andre gang

```

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



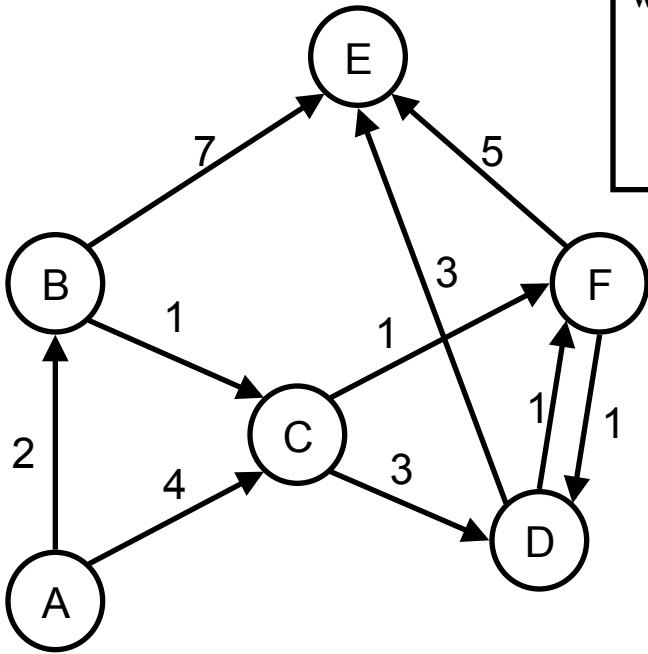
Q			S		
			A	0	*
B	2	A			
C	4	A			
D	∞				
E	∞				
F	∞				



Dijkstras algoritme – eksempel, andre gang

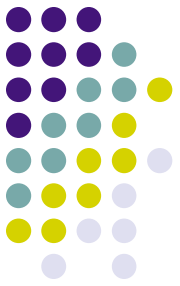
```

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



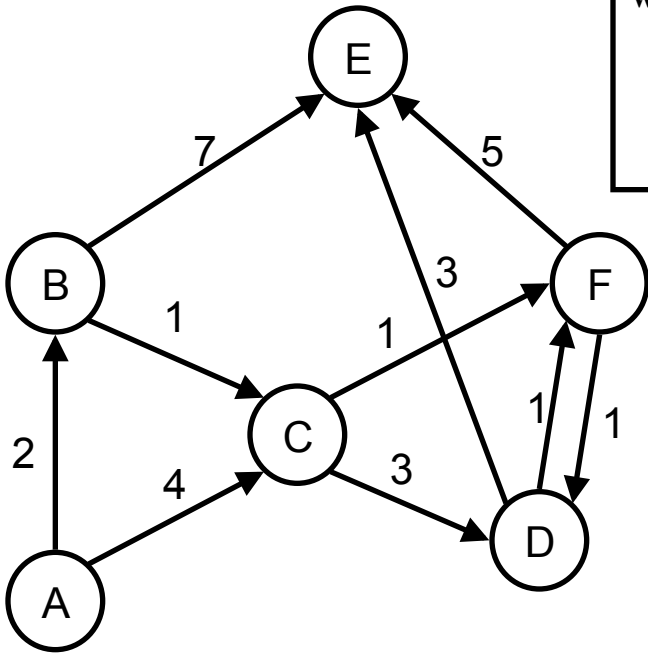
	Q	S
		A 0 *
		B 2 A
C	4 A	
D	∞	
E	∞	
F	∞	

Dijkstras algoritme – eksempel, andre gang



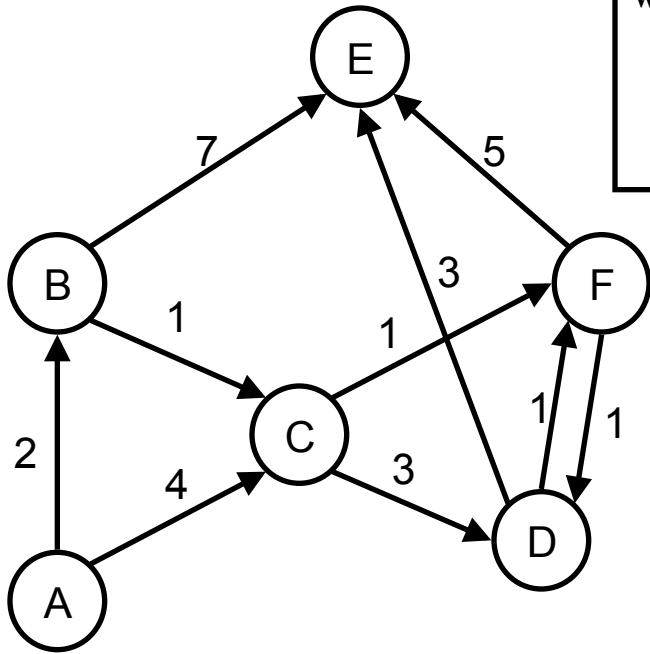
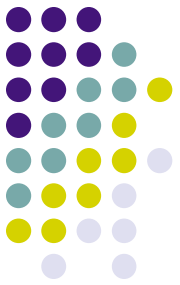
```

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

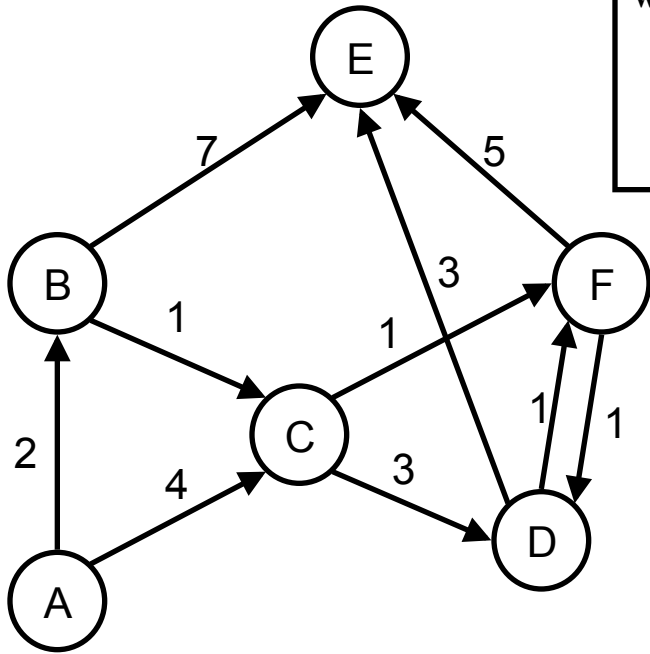
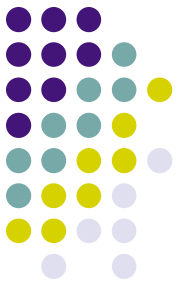


```

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

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

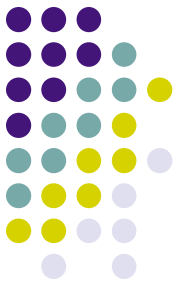
Dijkstras algoritme – eksempel, andre gang



```

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

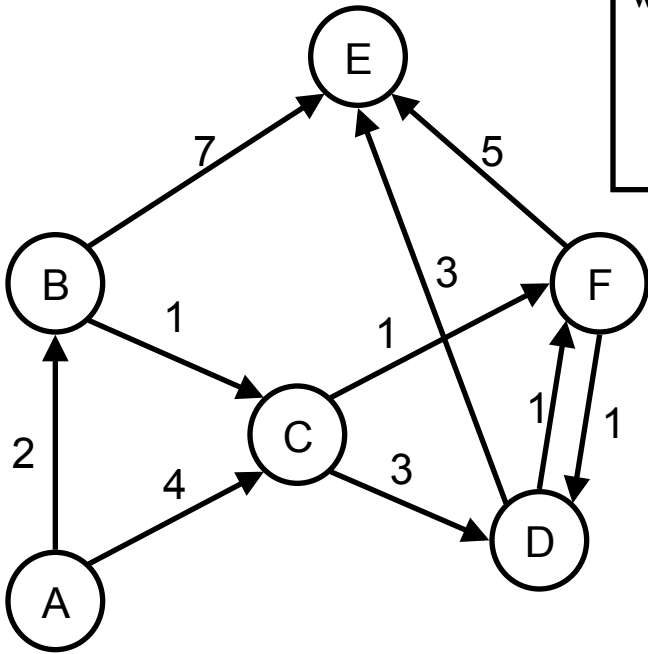
Q			S		
			A	0	*
			B	2	A
			C	3	B
D	6	C			
E	9	B			
F	4	C			



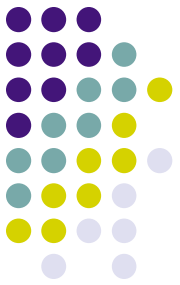
Dijkstras algoritme – eksempel, andre gang

```

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



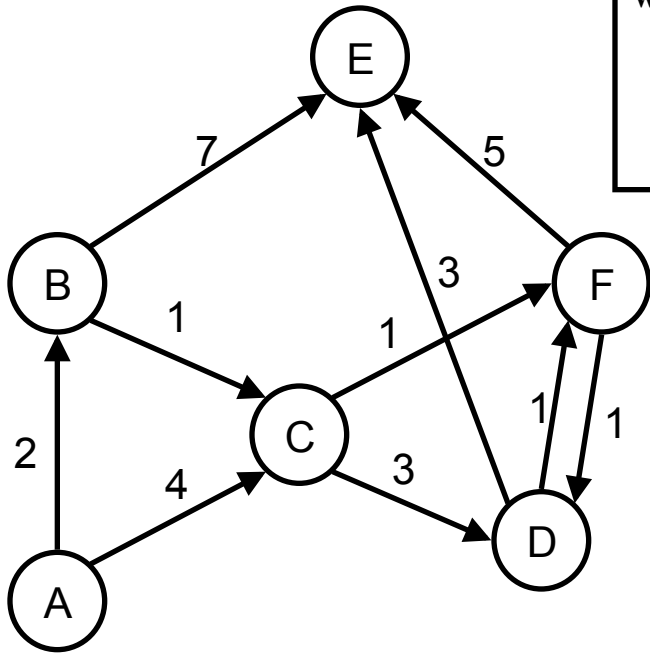
Q			S		
			A	0	*
			B	2	A
			C	3	B
D	6	C	F	4	C
E	9	B			



Dijkstras algoritme – eksempel, andre gang

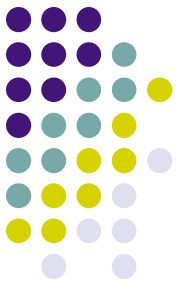
```

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



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

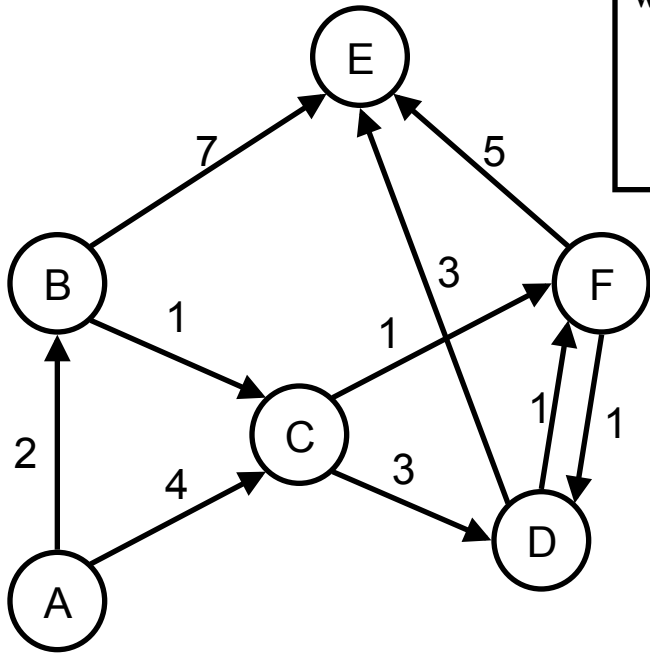




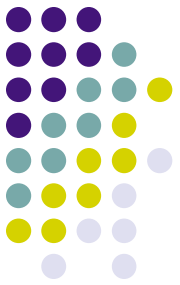
Dijkstras algoritme – eksempel, andre gang

```

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



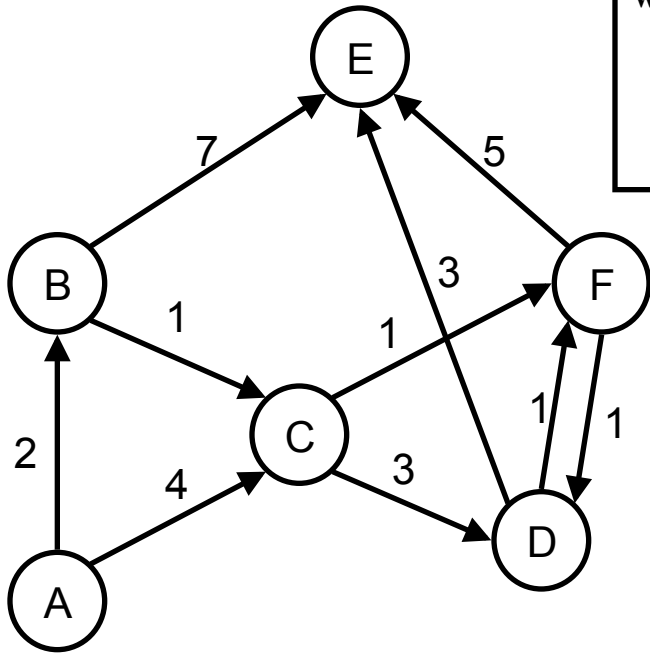
Q			S		
			A	0	*
			B	2	A
			C	3	B
			F	4	C
E	9	B	D	5	F



Dijkstras algoritme – eksempel, andre gang

```

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



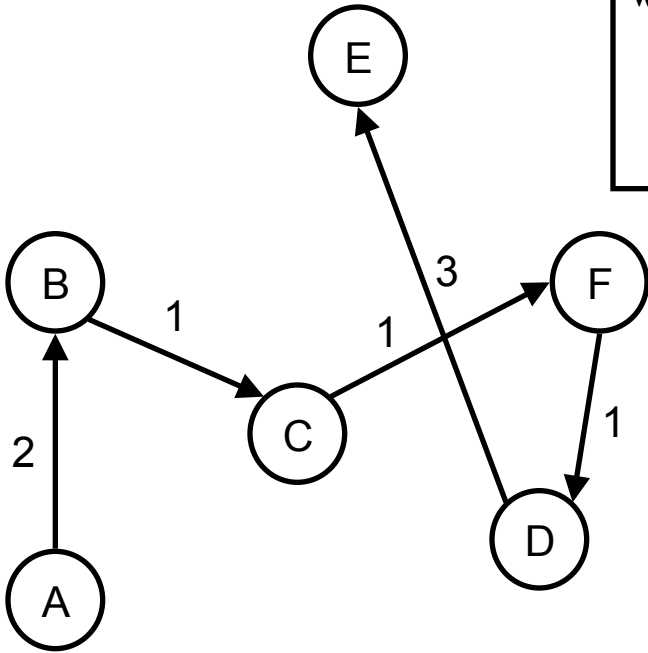
Q			S		
			A	0	*
			B	2	A
			C	3	B
			F	4	C
E	8	D	D	5	F



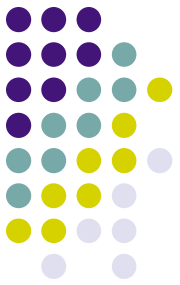


Dijkstras algoritme – eksempel, andre gang

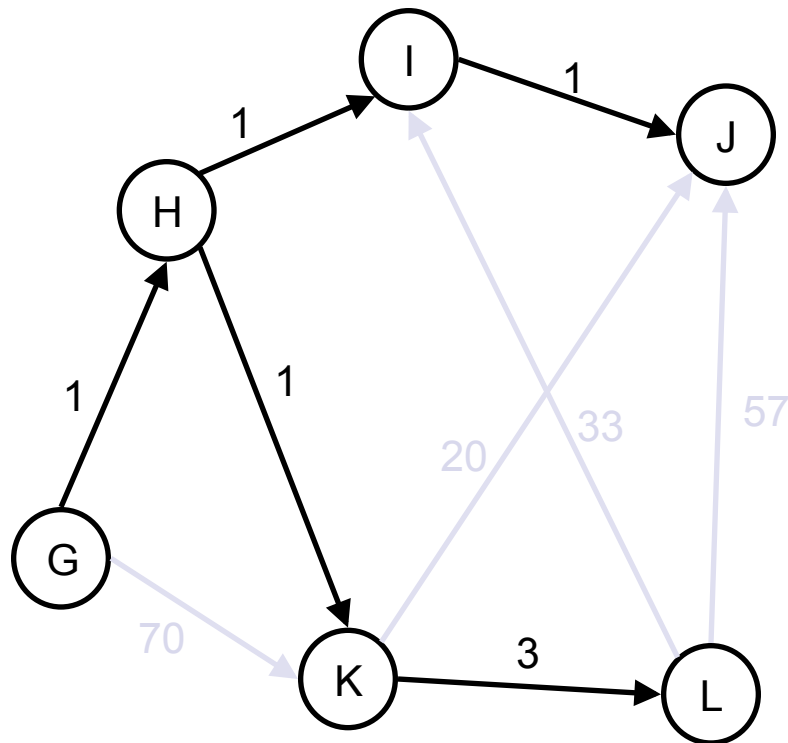
```
INITIALIZE-SINGLE-SOURCE( $G, s$ )  
 $S \leftarrow \emptyset$   
 $Q \leftarrow V[G]$   
while  $Q \neq \emptyset$   
  do  $u \leftarrow \text{EXTRACT-MIN}(Q)$   
   $S \leftarrow S \cup \{u\}$   
  for each vertex  $v \in \text{Adj}[u]$   
    do RELAX( $u, v, w$ )
```



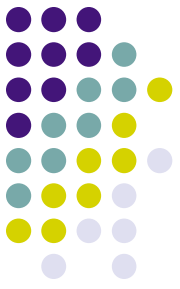
Q			S		
			A	0	*
			B	2	A
			C	3	B
			F	4	C
			D	5	F
			E	8	D



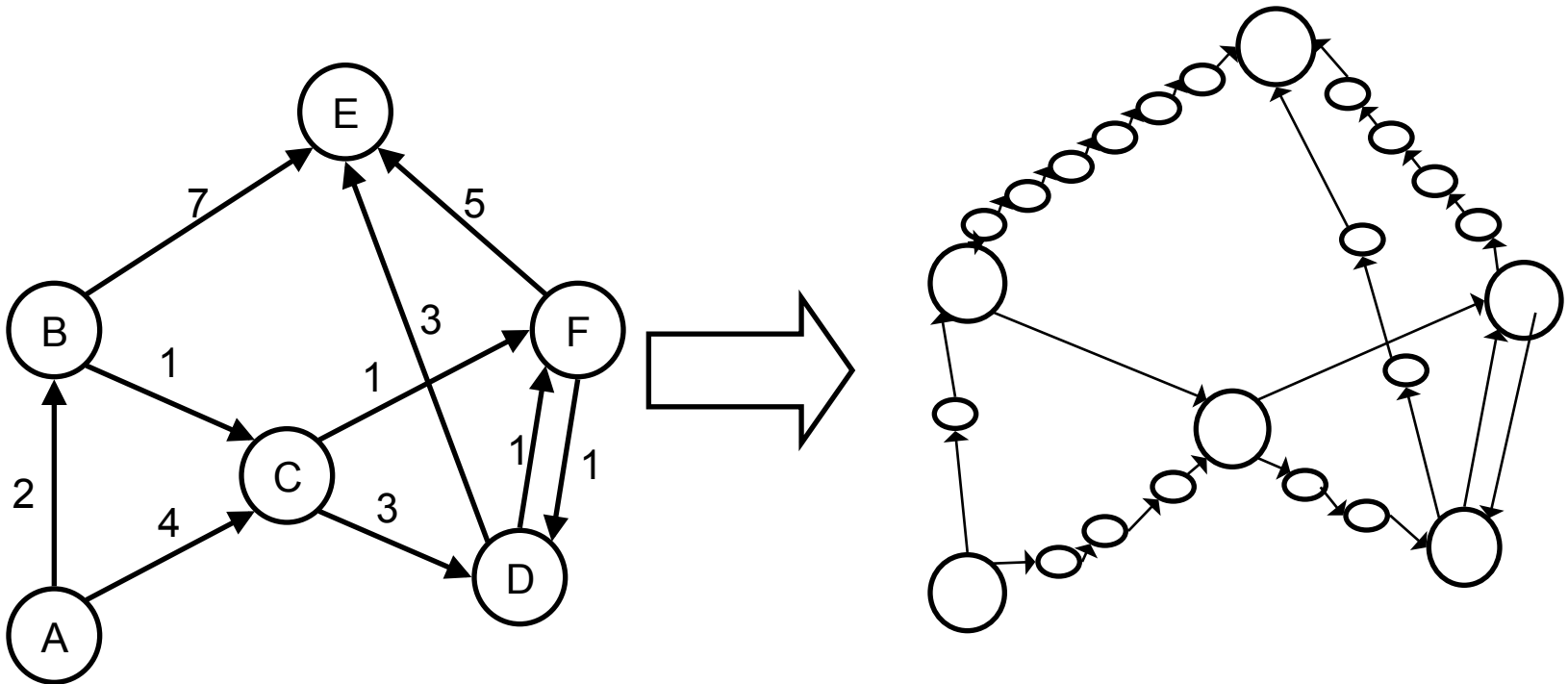
Dijkstras algoritme

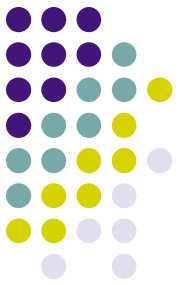


- I forrige eksempel lå alle nodene etter hverandre på den korteste stien
- Dijkstra finner korteste vei "en-til-alle", så resultatet er et tre.



Dijkstras algoritme – alternativ tankegang, bruk av BFS

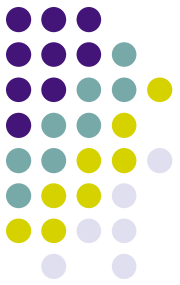




Dijkstras algoritme – valg av prioritetskø

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

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

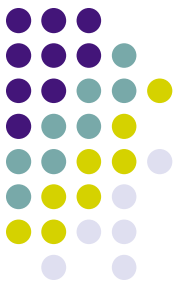


Dijkstras algoritme – valg av prioritetskø

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

	Sortert Array	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 $

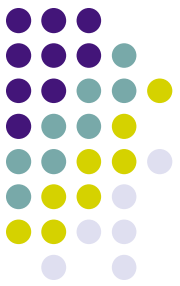


Dijkstras algoritme – valg av prioritetskø

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

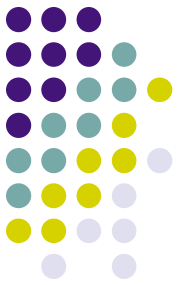
$$|V|^2 \quad |E| \log |V|$$

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



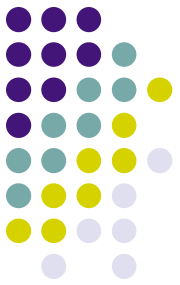
Oppsummering Dijkstra

- Effektiv, ofte brukt i den virkelige verden
- $O(E \lg V)$
- Takler ikke negative kanter
- Ligner veldig på BFS, men bruker prioritetskø istedet for FIFO-kø



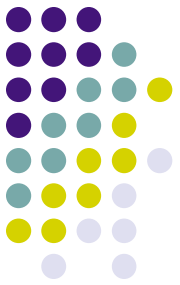
Bellman-Ford

- Ligner veldig på Dijkstra i grunnleggende struktur
- Relaxer alle kanter, $|V| - 1$ ganger
- Takler negative kanter, kan detektere negative sykler
- Veldig mye redundant arbeid



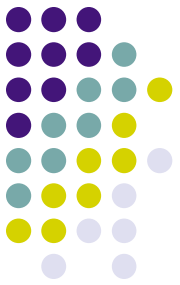
Bellman-Ford steg 1

```
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
Sett alle forløpere til null
legg alle noder inn i  $Q$ 
for i fra 1 til  $|V| - 1$ :
    for hver node u:
        for hver nabo v:
            hvis u kan tilby en kortere sti til v:
                oppdater v sitt estimat
                sett u som v sin forgjenger
```

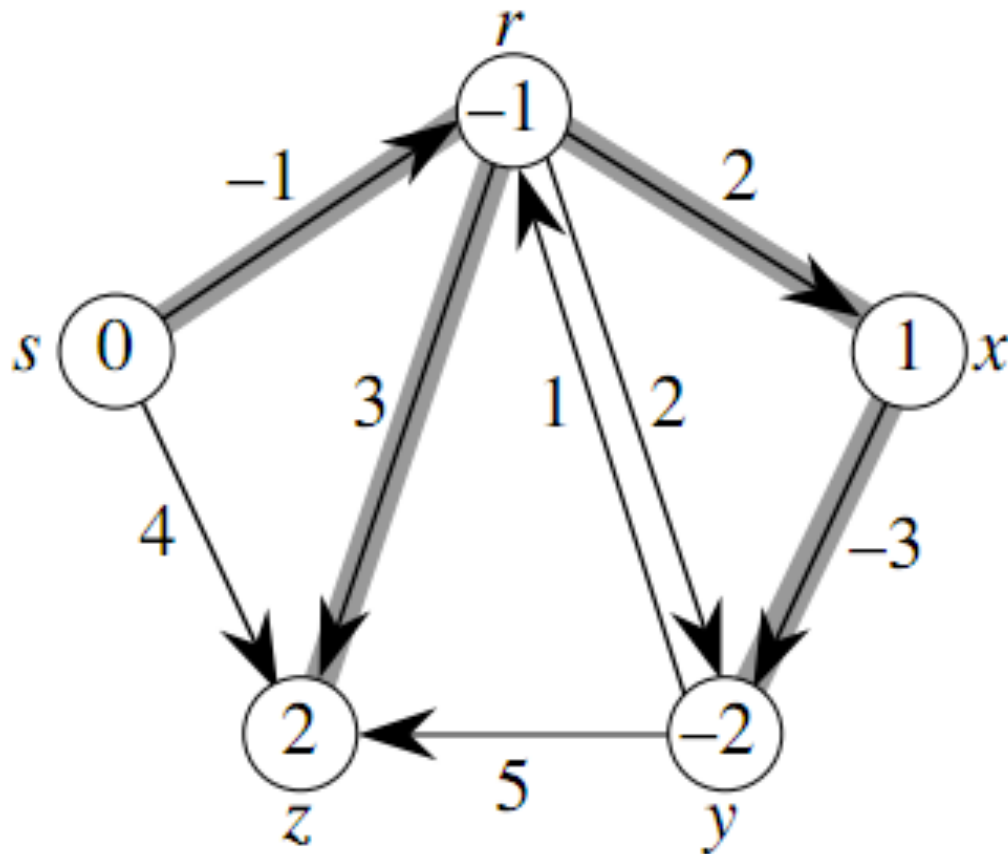


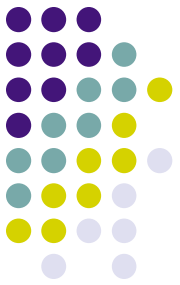
Bellman-Ford steg 2

```
for hver node u:  
    for hver nabo v:  
        hvis u kan tilby en kortere sti til v:  
            error "Grafen inneholder negative sykle(r)"
```



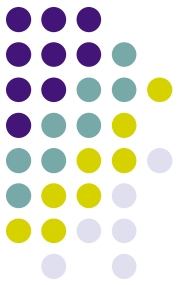
Bellman-Ford eksempel





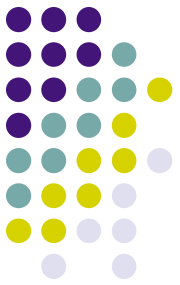
Oppsummering Bellman-Ford

- Kjøretid: $O(EV)$
- Takler negative kanter
- Kan oppdage negative sykler



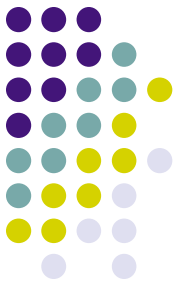
DAG-Shortest Path

- Sorterer først nodene topologisk, før den relaxer kantene.
- Tillater (åpenbart) ikke sykler
- Tillater negative kanter

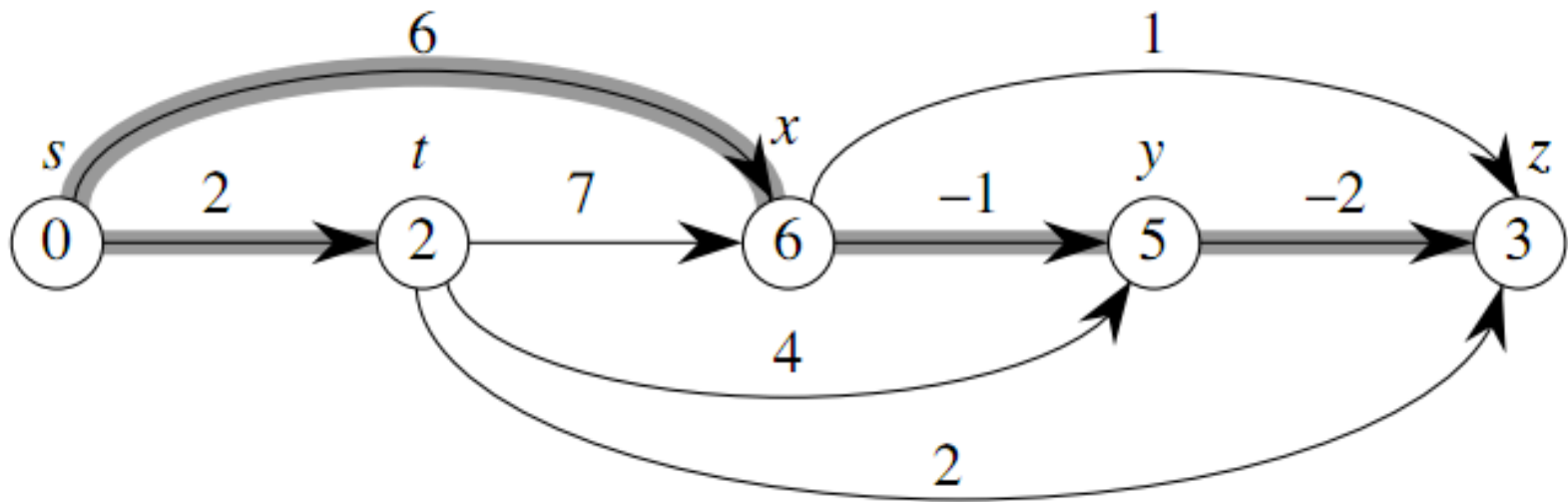


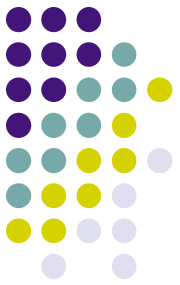
DAG-SP pseudokode

```
sorter grafen topologisk
sett alle estimer til  $\infty$ 
sett startnodens estimat til 0
sett alle forløpere til null
for hver node u:
    for hver nabo:
        hvis u kan tilby en kortere sti til v:
            oppdater v sitt estimat
            sett u som v sin forgjenger
```



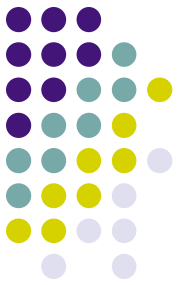
DAG-SP eksempel





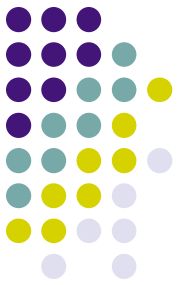
Oppsummering DAG-SP

- Mye samme tankegang som de to første
- Fungerer bare på DAGer selvfølgelig
- Utnytter topologisk sort så vi bare trenger å iterere over alle nodene en gang
- Kjøretid: $O(V + E)$



Oppsummering korteste vei (én til alle)

- Brukes til veldig mye i den virkelige verden
- Om man har en DAG
 - DAG-Shortest Path (Kjøretid: $O(V + E)$)
- Om man har en graf med bare positive kantvekker
 - Dijkstra (Kjøretid: $O(E \lg V)$)
- Ellers
 - Bellman-Ford (Kjøretid: $O(VE)$)



Denne ukens øving

- LF
 - Utfør Prims algoritme, finn max etterpå
- Tweak
 - Bruk heapq som er en prioritetskø basert på en heap