

Kartet er fra Prims
artikkell.

Forelesning 10

Korteste vei fra én til alle

Repetisjon

- › Korrekthet, MST
- › Kjøretid, Prim

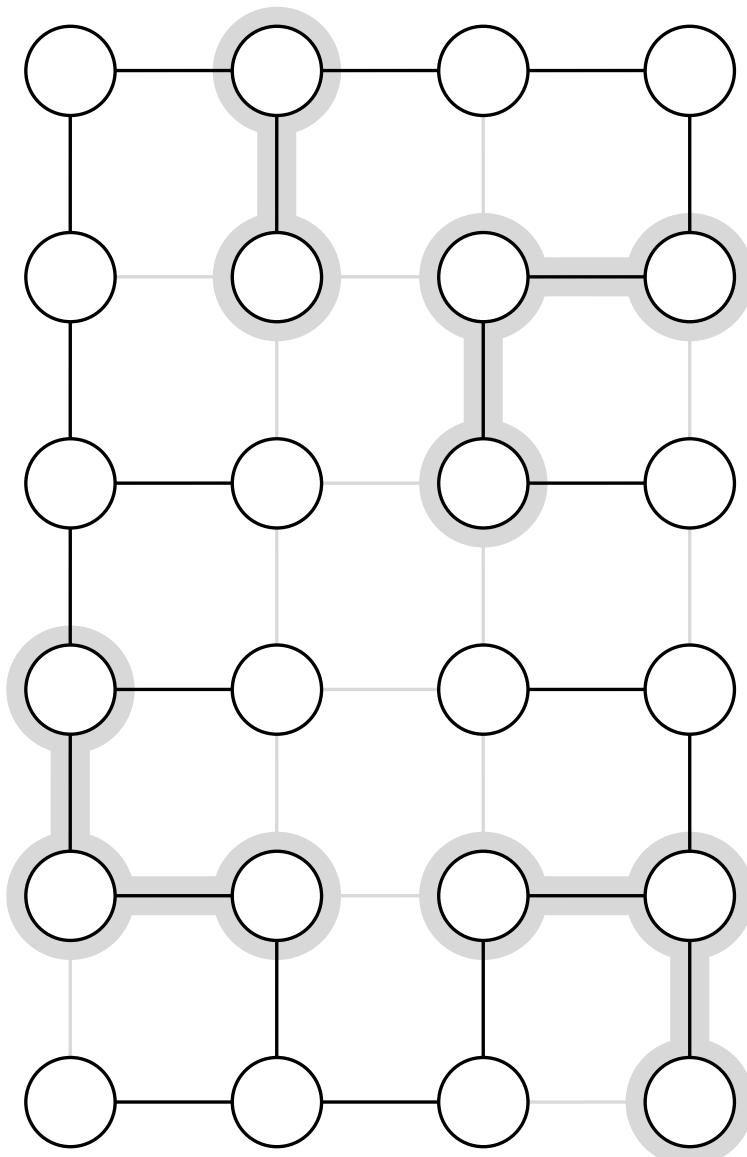
Korteste vei, én til alle

- › Om korteste veier
- › Kantslakking
- › Bellman-Ford
- › Dag-Shortest-Path
- › Dijkstra

MST → Korrekthet

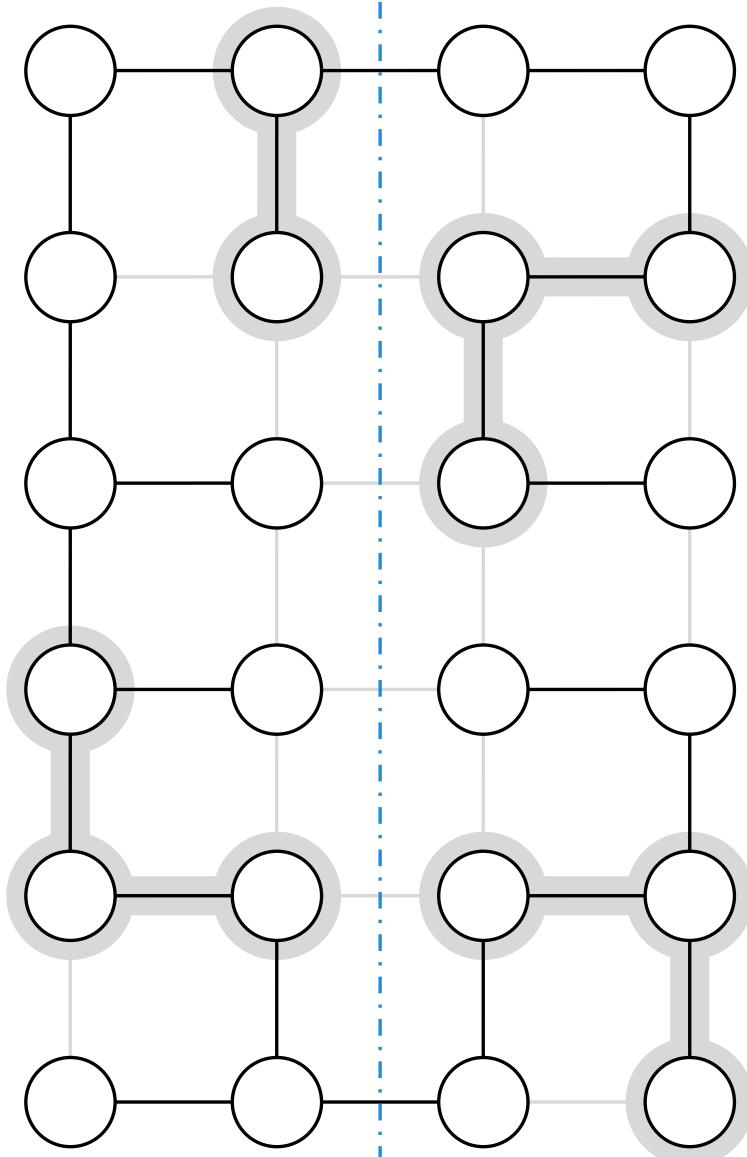
Kantmengden A er del av et MST.

MST > korrekthet



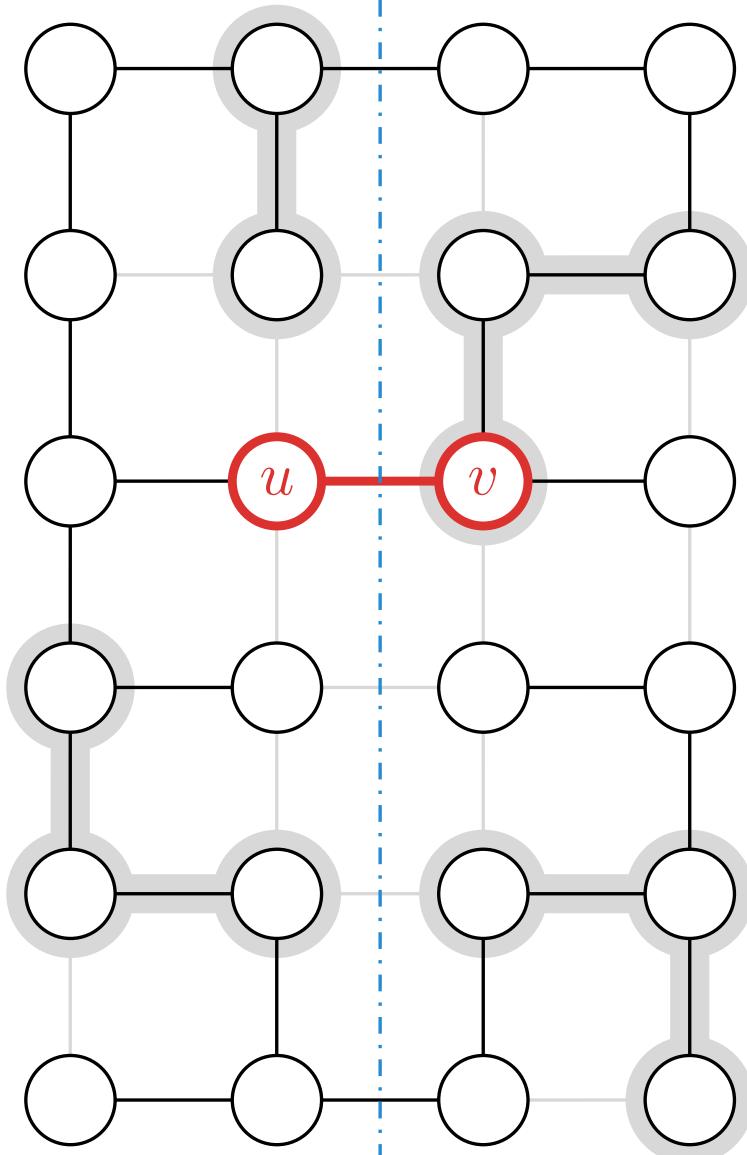
Kantmengden A er del av et MST.
Snittet $(S, V - T)$ respekterer A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.
Snittet $(S, V - T)$ respekterer A .
 (u, v) er minimal over snittet.

MST \rightarrow korrekthet



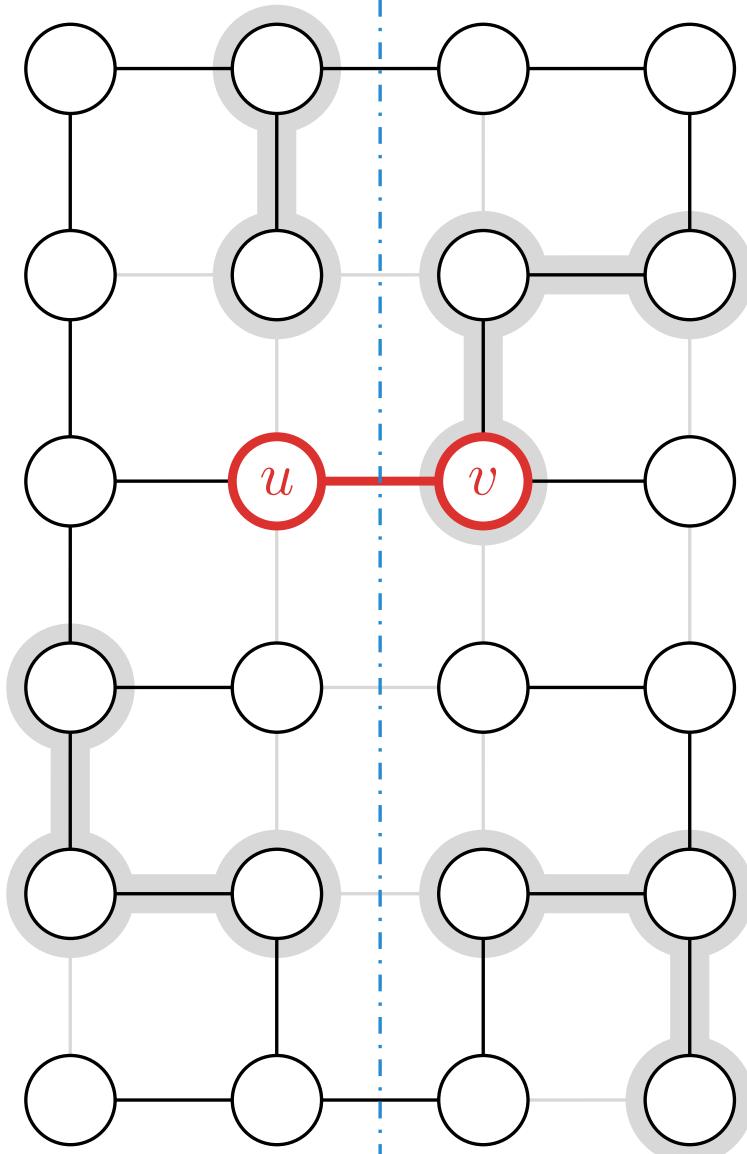
Kantmengden A er del av et MST.

Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

(u, v) er trygg for A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.

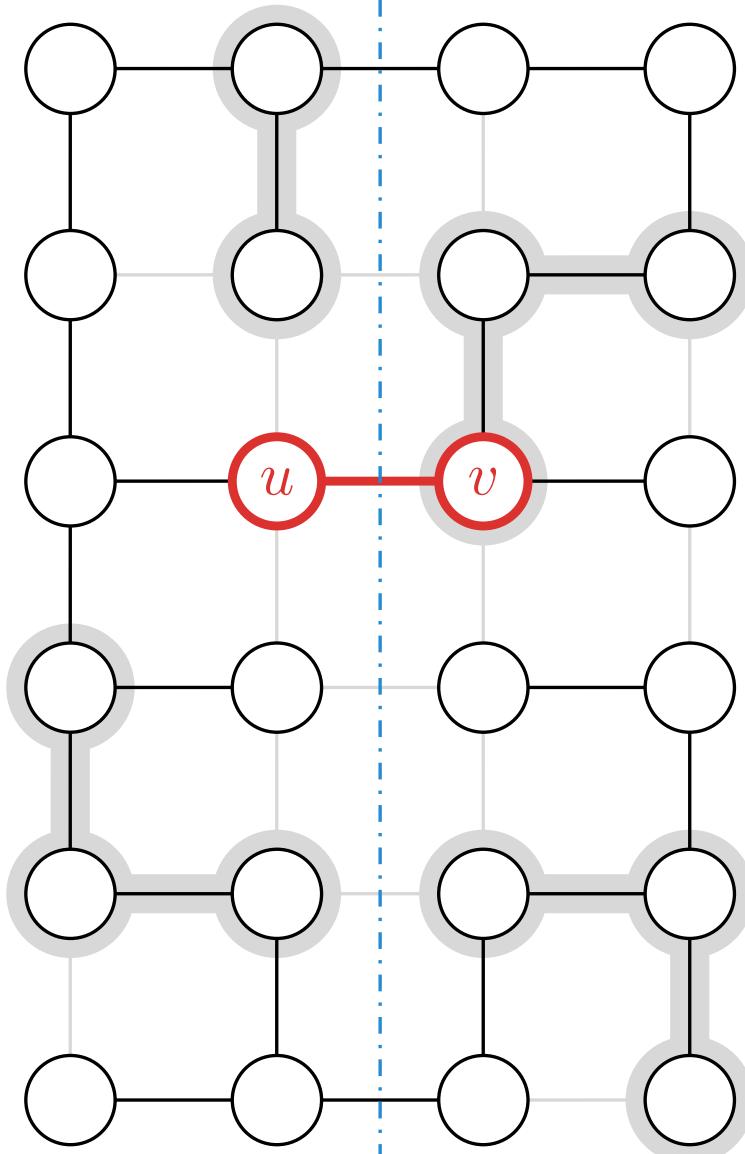
Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

$A + (u, v)$ er del av et MST.

Det vil si, (u, v) er trygg for A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.

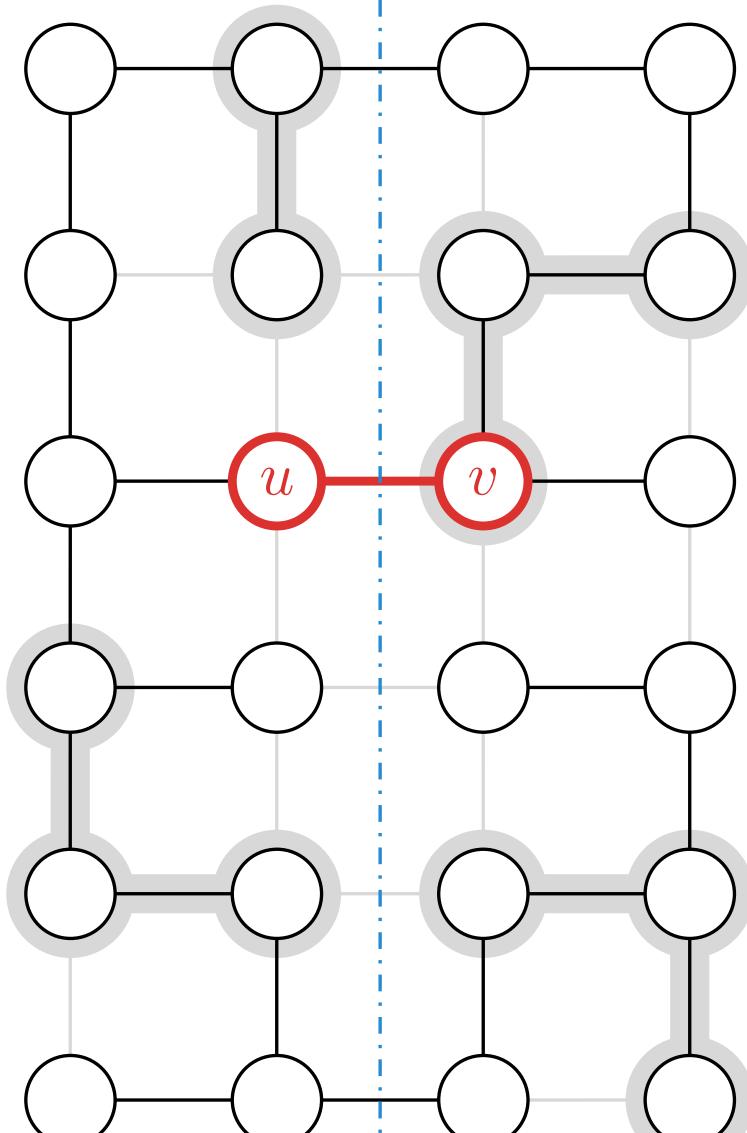
Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

For et MST $T \supseteq A$ finnes et like bra spennetre $T' \supseteq A + (u, v)$.

Det vil si, (u, v) er trygg for A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.

Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

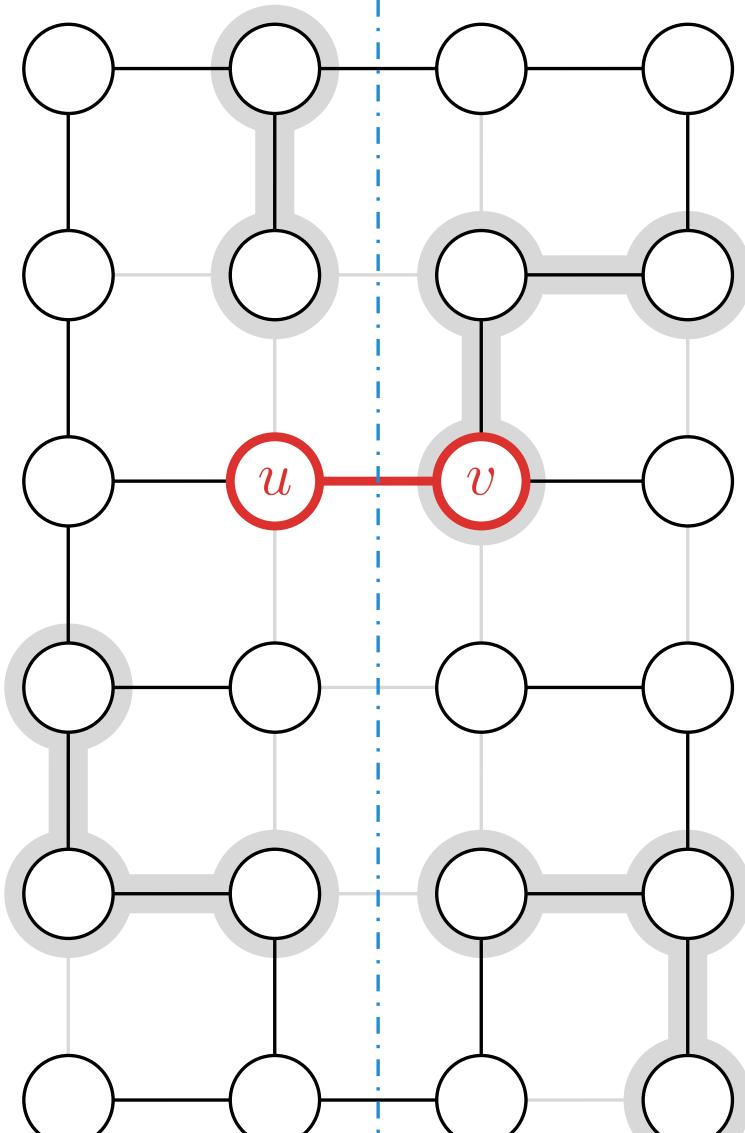
La T være et MST der $A \subseteq T$.

› Det finnes et spennetre $T' \subseteq A + (u, v)$ med $w(T') \leq w(T)$.

Altså, for et MST $T \supseteq A$ finnes et like bra spennetre $T' \supseteq A + (u, v)$.

Det vil si, (u, v) er trygg for A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.

Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

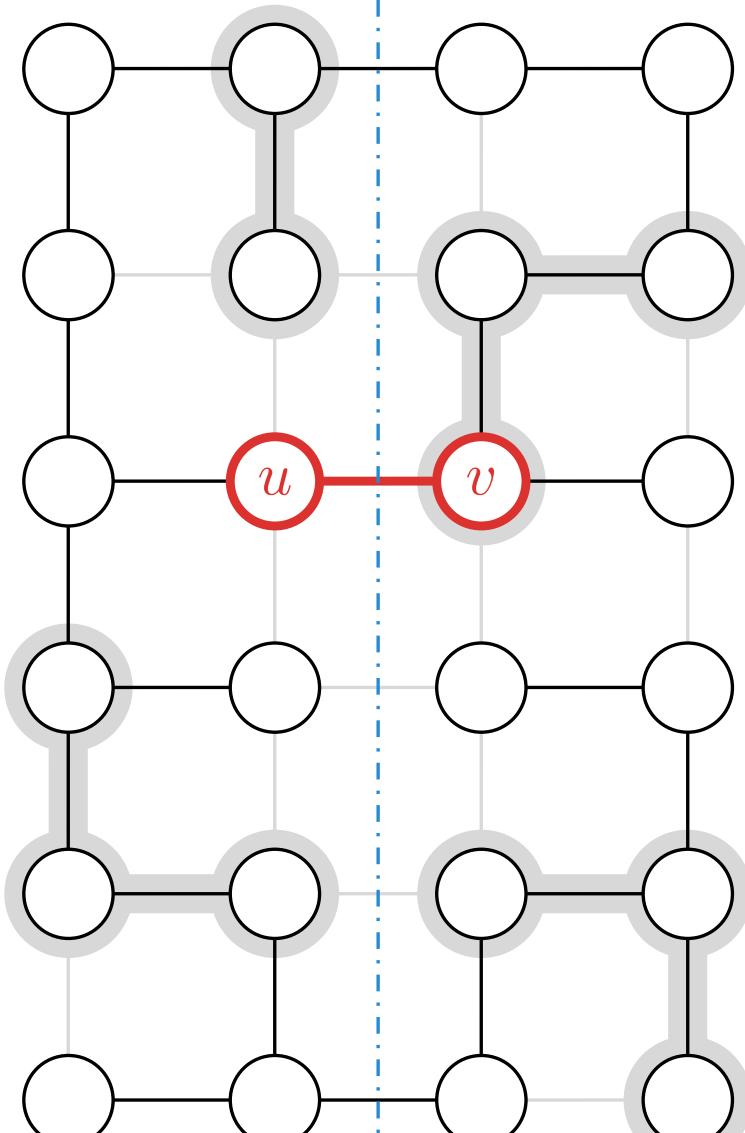
La T være et MST der $A \subseteq T$.

- › Tilfelle 1: $(u, v) \in T$. ($T' = T$)
- › Tilfelle 2: $(u, v) \notin T$.
 - › Det finnes et spennetre $T' \subseteq A + (u, v)$ med $w(T') \leq w(T)$.

Altså, for et MST $T \supseteq A$ finnes et like bra spennetre $T' \supseteq A + (u, v)$.

Det vil si, (u, v) er trygg for A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.

Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

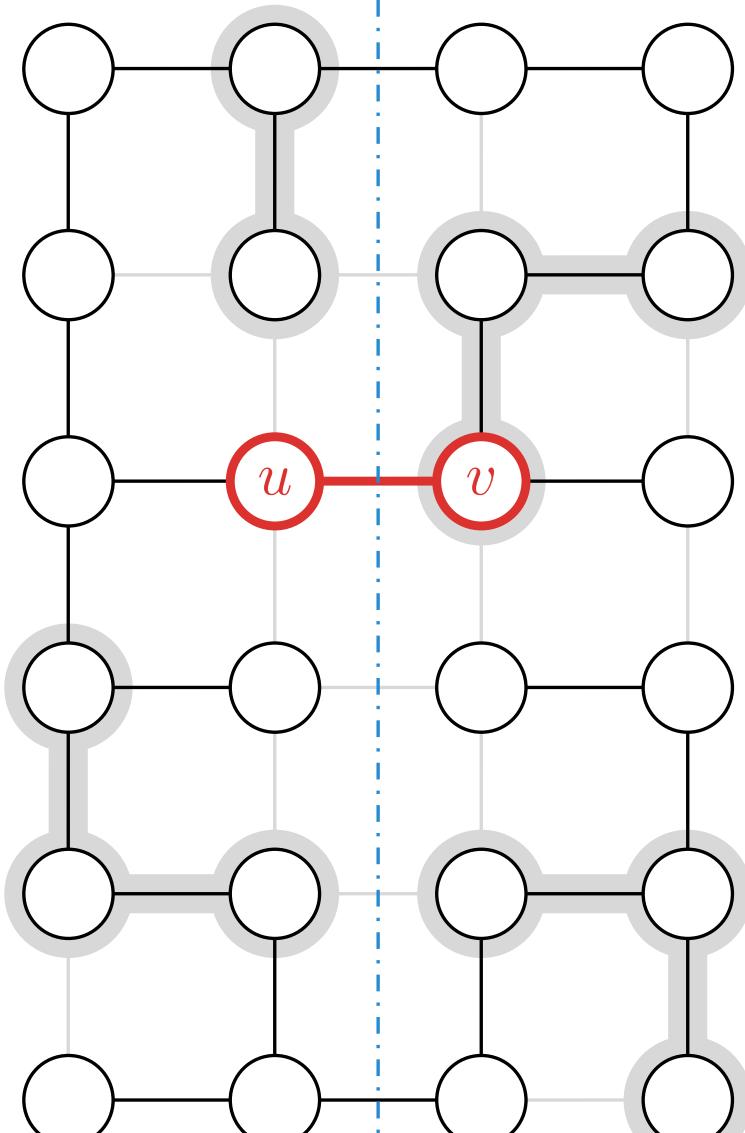
La T være et MST der $A \subseteq T$.

- › Tilfelle 1: $(u, v) \in T$. ($T' = T$)
- › Tilfelle 2: $(u, v) \notin T$.
 - › Det finnes en $(x, y) \in T$ der ...
 - › $w(u, v) \leq w(x, y)$.
 - › $T - (x, y) + (u, v)$ er et spennetre.

Altså, for et MST $T \supseteq A$ finnes et like bra spennetre $T' \supseteq A + (u, v)$.

Det vil si, (u, v) er trygg for A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.

Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

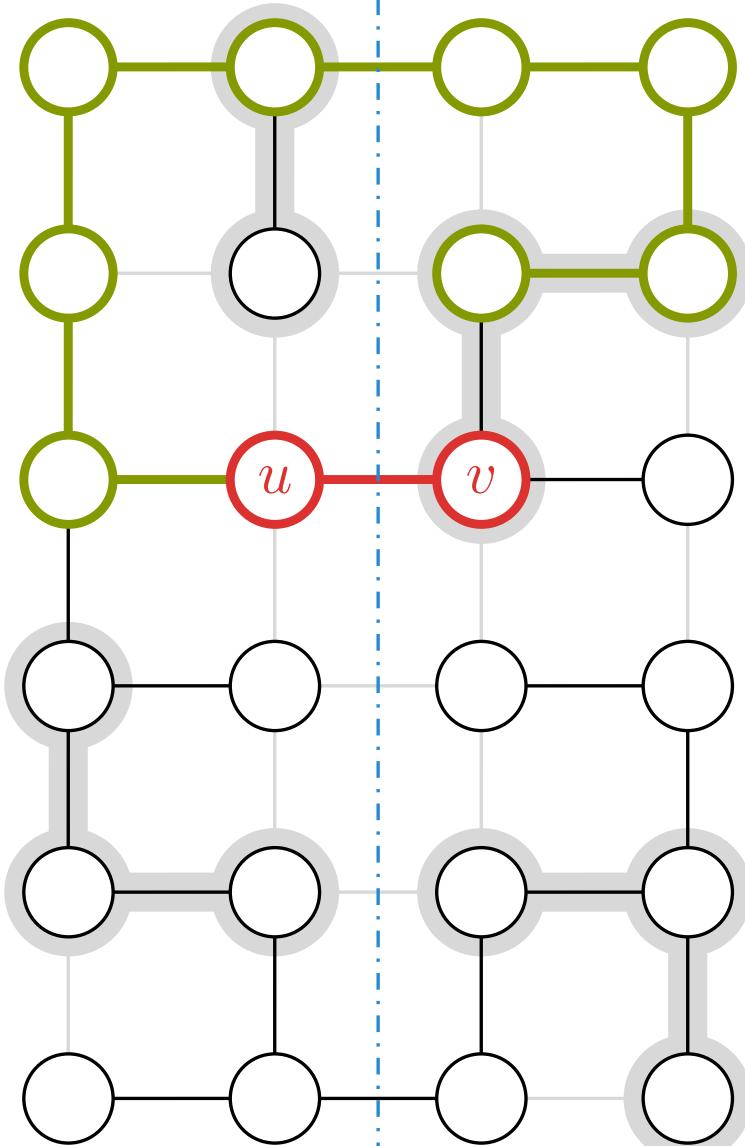
La T være et MST der $A \subseteq T$.

- › Tilfelle 1: $(u, v) \in T$. ($T' = T$)
- › Tilfelle 2: $(u, v) \notin T$.
 - › La p være stien fra u til v i T .
 - › Det finnes en $(x, y) \in p$ der ...
 - › $w(u, v) \leq w(x, y)$.
 - › $T - (x, y) + (u, v)$ er et spennetre.

Altså, for et MST $T \supseteq A$ finnes et like bra spennetre $T' \supseteq A + (u, v)$.

Det vil si, (u, v) er trygg for A .

MST \rightarrow korrekthet



Kantmengden A er del av et MST.

Snittet $(S, V - T)$ respekterer A .

(u, v) er minimal over snittet.

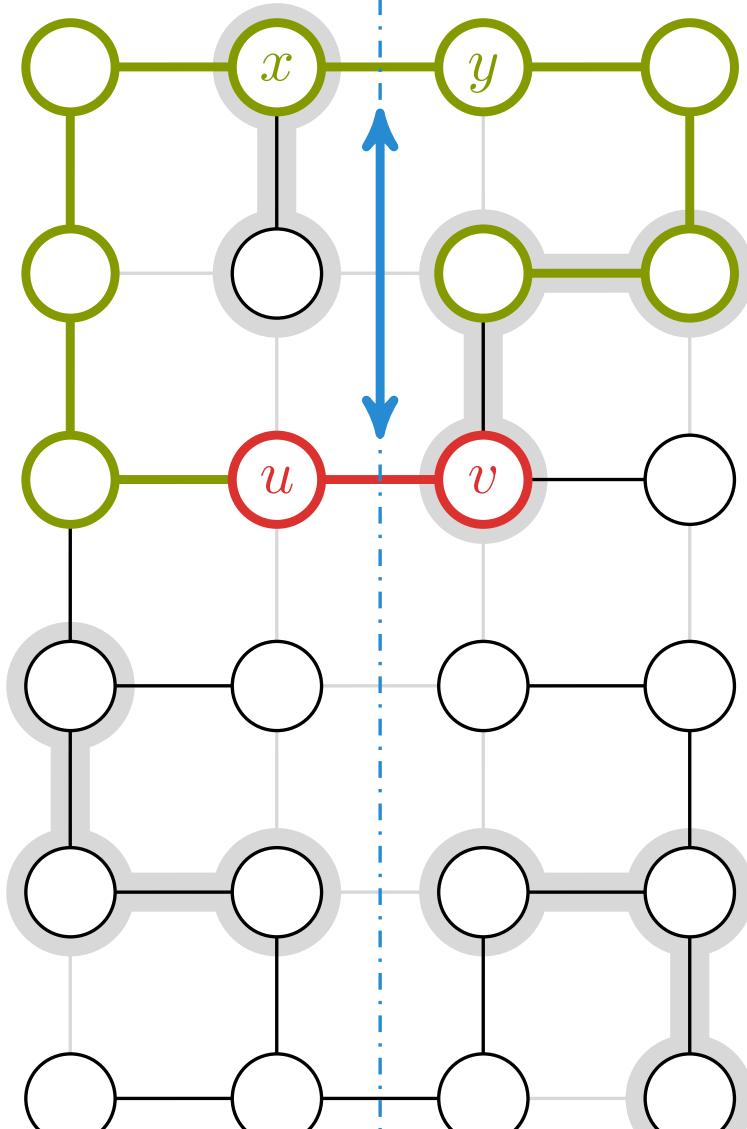
La T være et MST der $A \subseteq T$.

- › Tilfelle 1: $(u, v) \in T$. ($T' = T$)
- › Tilfelle 2: $(u, v) \notin T$.
 - › La p være stien fra u til v i T .
 - › La $(x, y) \in p$ krysse snittet.
 - › $w(u, v) \leq w(x, y)$.
 - › $T - (x, y) + (u, v)$ er et spennetre.

Altså, for et MST $T \supseteq A$ finnes et like bra spennetre $T' \supseteq A + (u, v)$.

Det vil si, (u, v) er trygg for A .

MST \rightarrow korrekthet



MST → Prim

Shortest Connection Networks And Some Generalizations

By R. C. PRIM

(Manuscript received May 8, 1957)

The basic problem considered is that of interconnecting a given set of terminals with a shortest possible network of direct links. Simple and practical procedures are given for solving this problem both graphically and computationally. It develops that these procedures also provide solutions for a much broader class of problems, containing other examples of practical interest.

1. INTRODUCTION

A problem of inherent interest in the planning of large-scale communication and transportation networks also arises in connection with the rate structure for Bell System leased-line services. The problem is to connect a set of (point) terminals, connect them by a network of lines having the smallest possible total length, and have the network "connected," that is, between every two terminals there is a path.

Operasjon	Antall	Kjøretid
BUILD-MIN-HEAP	1	$O(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY	E	$O(\lg V)$

Totalt: $O(E \lg V)$

$O(1)$ amortisert for Fib.-haug

Dette gjelder om vi bruker en binærhaug

Operasjon	Antall	Kjøretid
BUILD-MIN-HEAP	1	$O(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY	E	$O(\lg V)$

Totalt: $O(E \lg V)$

$O(1)$ amortisert for Fib.-haug

Kan forbedres til $O(E + V \lg V)$ med Fibonacci-haug

Korteste vei, én til alle

Korteste vei ›

Problemet

Input: En rettet graf $G = (V, E)$, vekt-funksjon $w : E \rightarrow \mathbb{R}$ og node $s \in V$.

Output: For hver node $v \in V$, en sti $p = \langle v_0, v_1, \dots, v_k \rangle$ med $v_0 = s$ og $v_k = v$, som har minimal vektsum

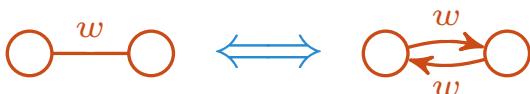
$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i).$$

Vi kaller også w lengde; $\delta(s, v) = w(p)$ er avstanden fra s til v

Input: En rettet graf $G = (V, E)$, vekt-funksjon $w : E \rightarrow \mathbb{R}$ og node $s \in V$.

Output: For hver node $v \in V$, en sti $p = \langle v_0, v_1, \dots, v_k \rangle$ med $v_0 = s$ og $v_k = v$, som har minimal vektsum

$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i).$$

Urettede grafer: 

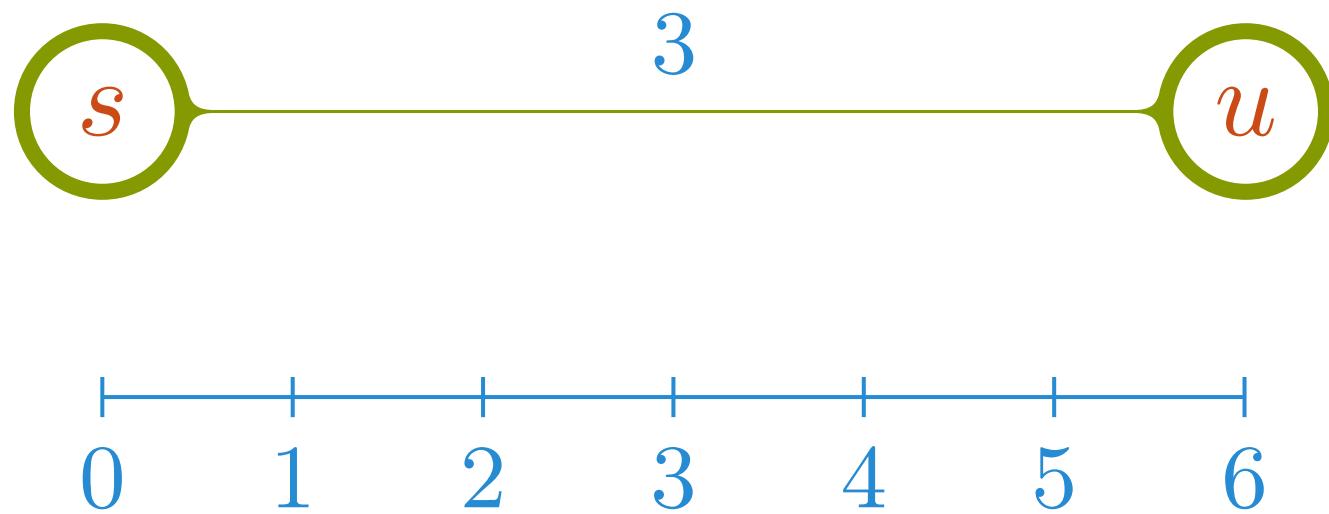
- › En enkel sti er en sti uten sykler
- › En kortest vei vil aldri inneholde en positiv sykel
- › Om vi ikke kan nå noen negative sykler så er «korteste sti» det samme som «korteste enkle sti»
- › Om en sti til v har en negativ sykel, så finnes det alltid en kortere sti – ingen er kortest!
- › Det vil likevel finnes en kortest **enkel** sti til v , men vi kjenner ingen generelle algoritmer for å finne den

Korteste vei ›

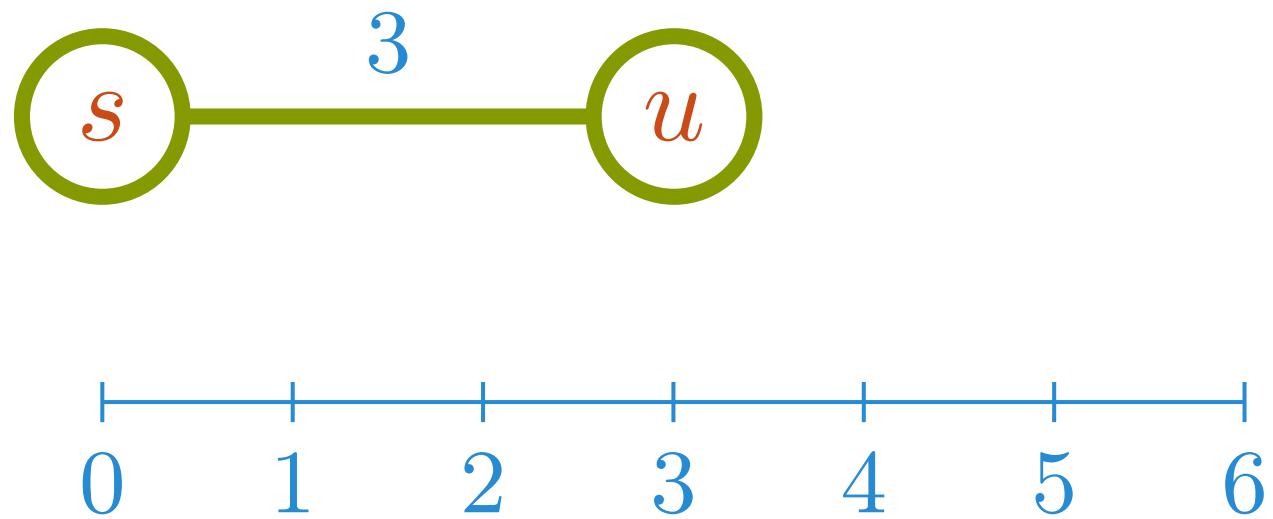
Kantslakking

Se fotnote på side 648 i
boka for mer om navnet.

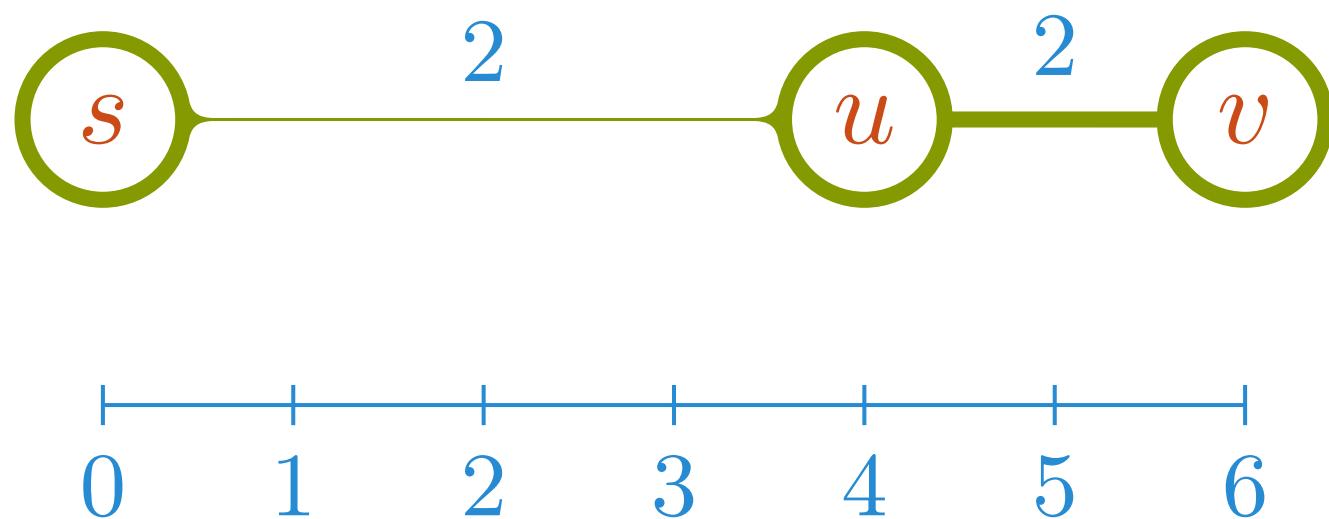
korteste vei → slakking



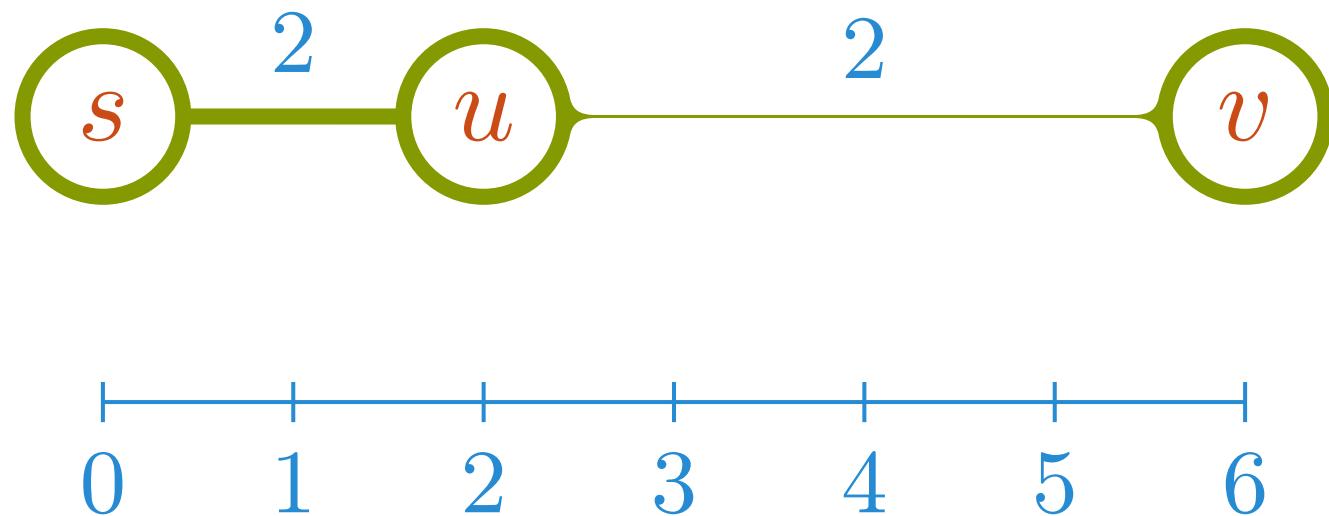
korteste vei \rightarrow slakking



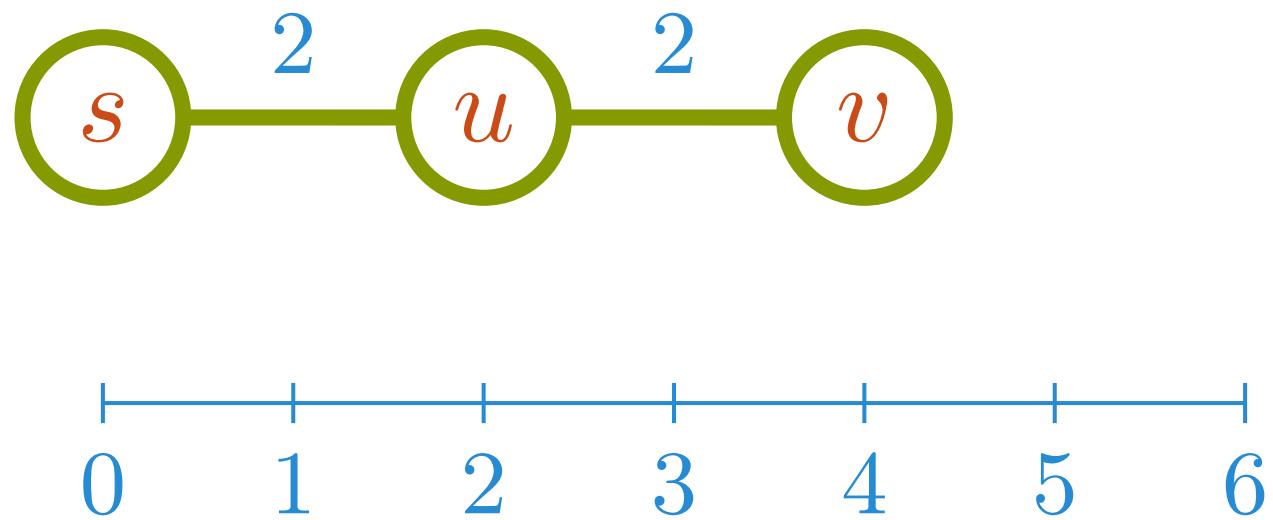
korteste vei \rightarrow slakking



korteste vei → slakking



korteste vei \rightarrow slakking



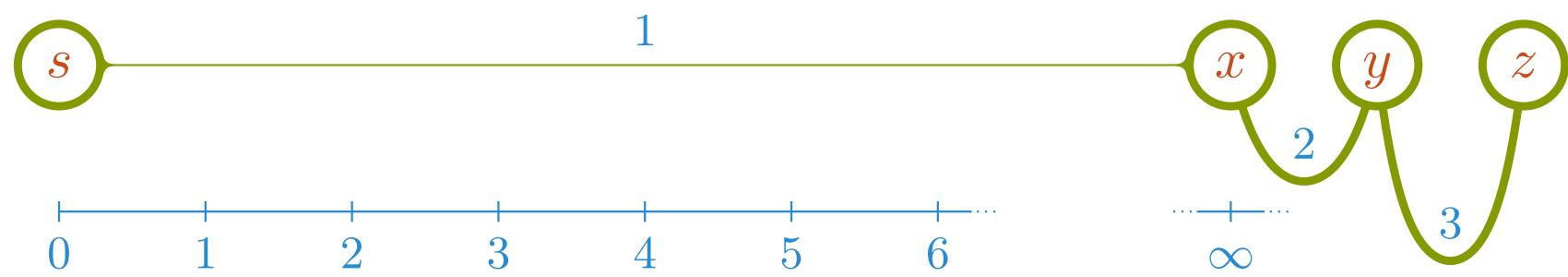
$$a \neq -\infty$$

$$a + \infty = \infty + a = \infty$$

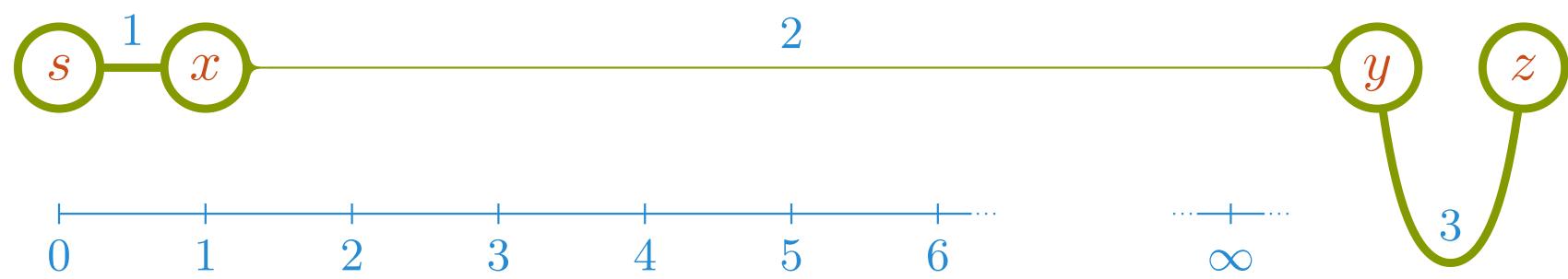
$$a + (-\infty) = (-\infty) + a = -\infty$$

Vi bruker den utvidede tallinja $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$

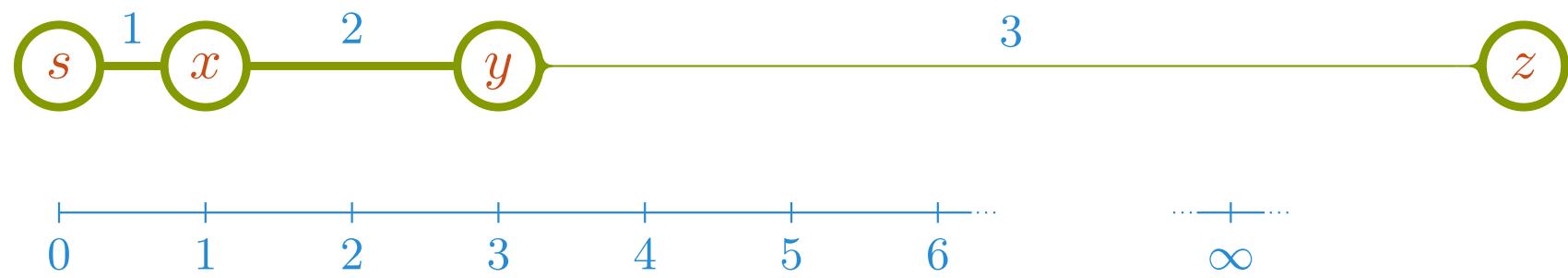
korteste vei > slakking



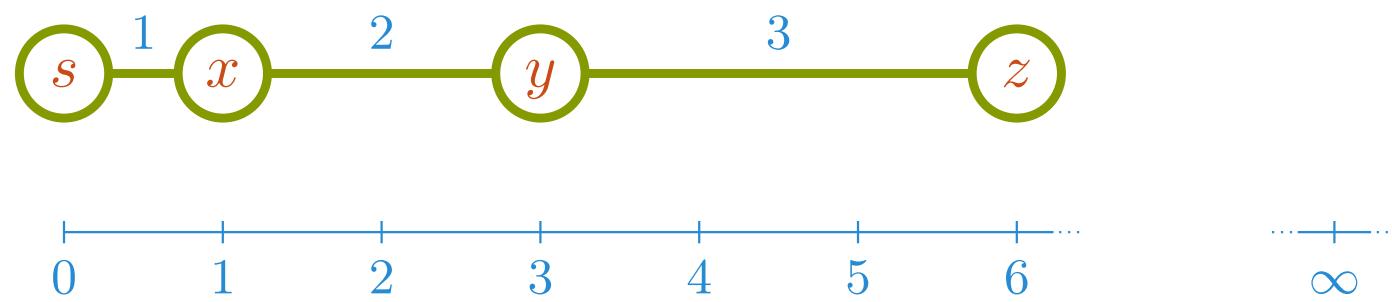
korteste vei \rightarrow slakking



korteste vei → slakking



korteste vei \rightarrow slakking



Sti-slakkings-egenskapen

Om p er en kortest vei fra s til v og vi slakker kantene til p i rekkefølge, så vil v få riktig avstandsestimat. Det gjelder uavhengig av om andre slakninger forekommer, selv om de kommer innimellom.

«The path-relaxation property»; flere slektninger på s. 650

INITIALIZE-SINGLE-SOURCE(G, s)

- 1 **for** each vertex $v \in G.V$
- 2 $v.d = \infty$
- 3 $v.\pi = \text{NIL}$
- 4 $s.d = 0$

```
RELAX( $u, v, w$ )
1  if  $v.d > u.d + w(u, v)$ 
2       $v.d = u.d + w(u, v)$ 
3       $v.\pi = u$ 
```

- › Vi starter med å initialisere grafen
- › Vi vil så slakke alle kantene langs én av de korteste veiene til hver node
- › Vi vil slakke så få kanter som mulig

Korteste vei ›
Bellman-Ford

- › Om vi slakker alle kantene vil vi automatisk måtte ha slakket første kant i hver kortest sti
- › Om vi slakker alle kantene k ganger vil vi automatisk måtte ha slakket de k første
- › En kortest sti uten negative sykler kan maksimalt ha $V-1$ kanter
- › Slakk alle kantene $V-1$ ganger!
- › Prøv å slakke alle én gang til; en forbedring da betyr at vi har en negativ sykel, og at vi dermed gir opp

```
BELLMAN-FORD( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3    for each edge  $(u, v) \in G.E$ 
4      RELAX( $u, v, w$ )
5    for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7        return FALSE
8  return TRUE
```

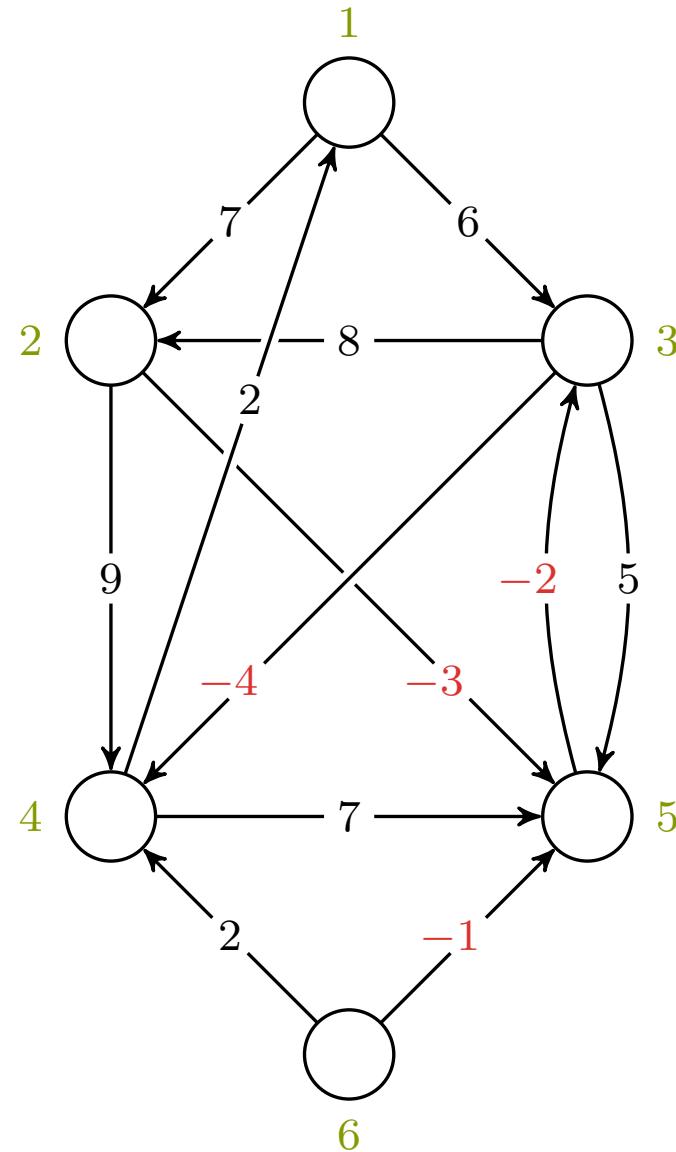
BELLMAN-FORD(G, w, s)

```

1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE

```

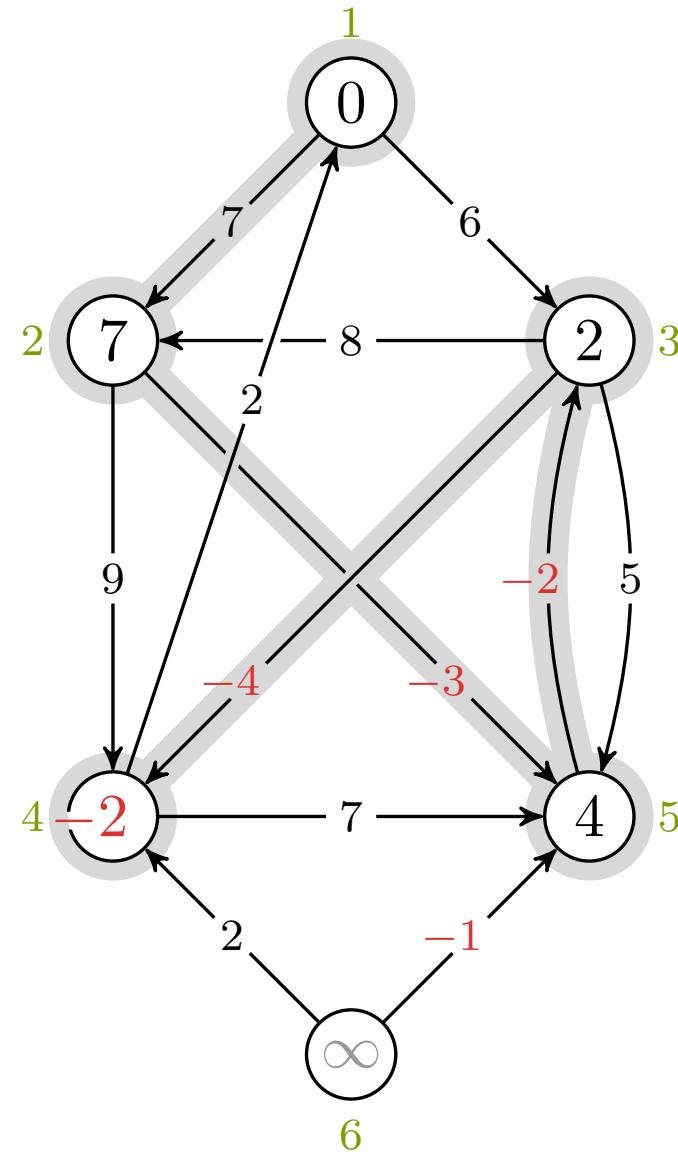
$i, u, v = -, -, -$



korteste vei > bellman-ford

```
BELLMAN-FORD( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE
→ TRUE
```

$i, u, v = -, -, -$



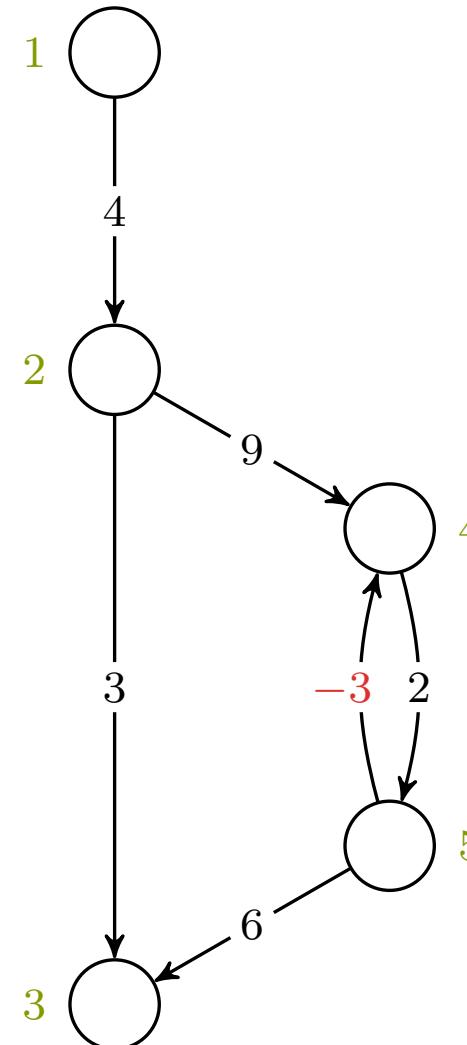
BELLMAN-FORD(G, w, s)

```

1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3      for each edge  $(u, v) \in G.E$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE

```

$i, u, v = -, -, -$



Korteste vei ›
Bellman-Ford › **Kjøretid**

Se også oppg. 24-1

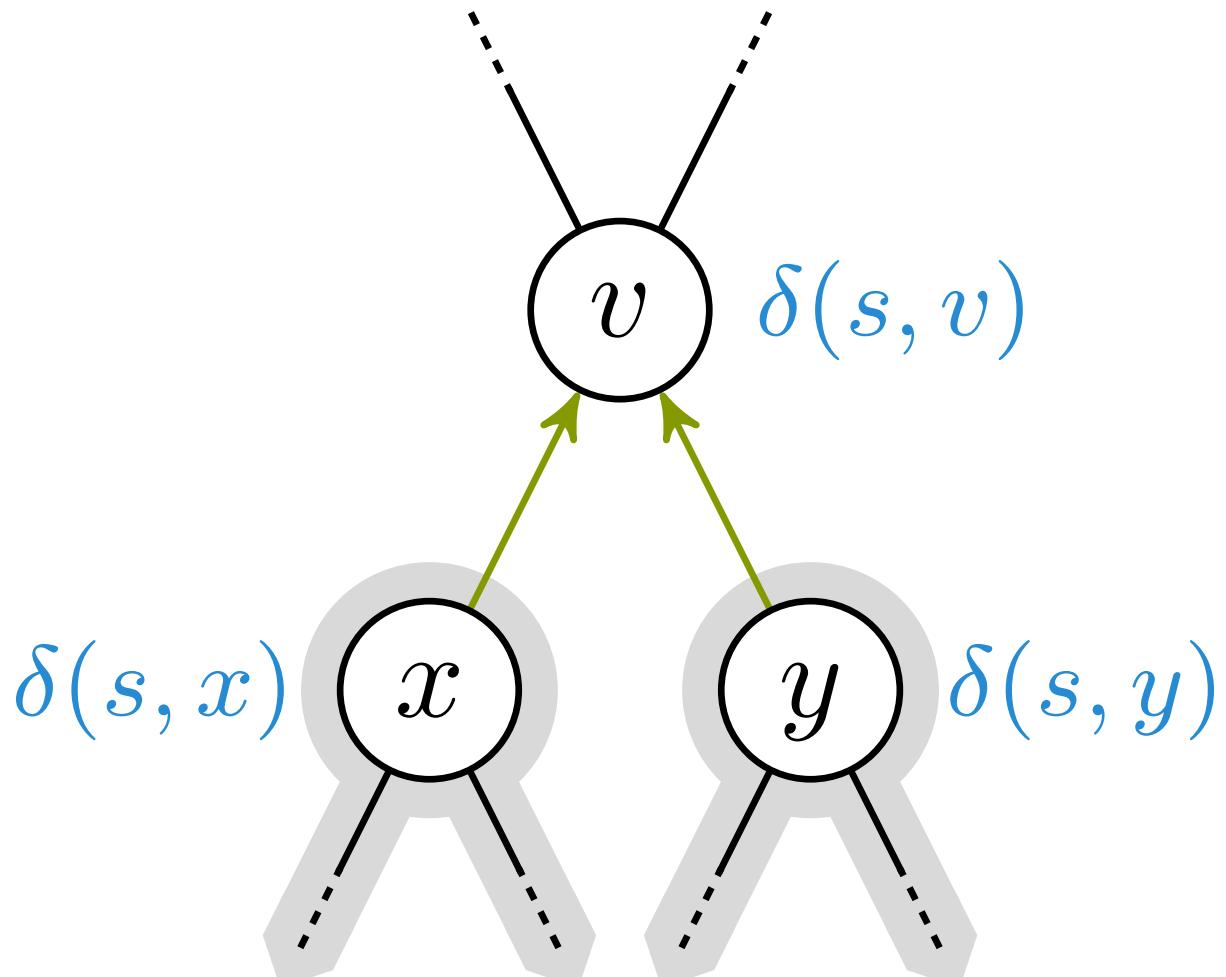
Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
RELAX	$V - 1$	$\Theta(1)$
RELAX	$O(V)$	$\Theta(1)$

Totalt: $O(VE)$

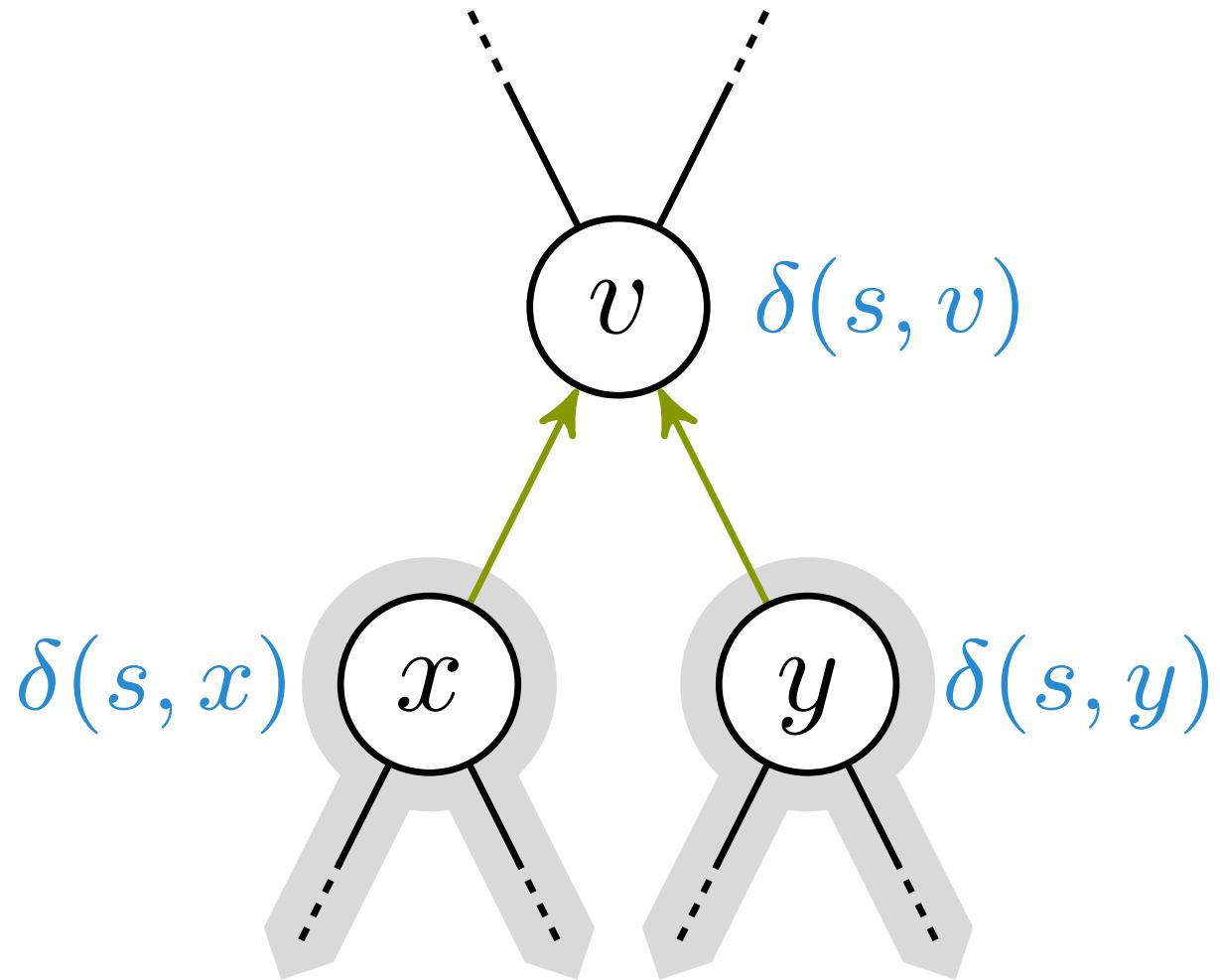
Stopp når ingenting endres? Fortsatt $O(VE)$

Korteste vei →

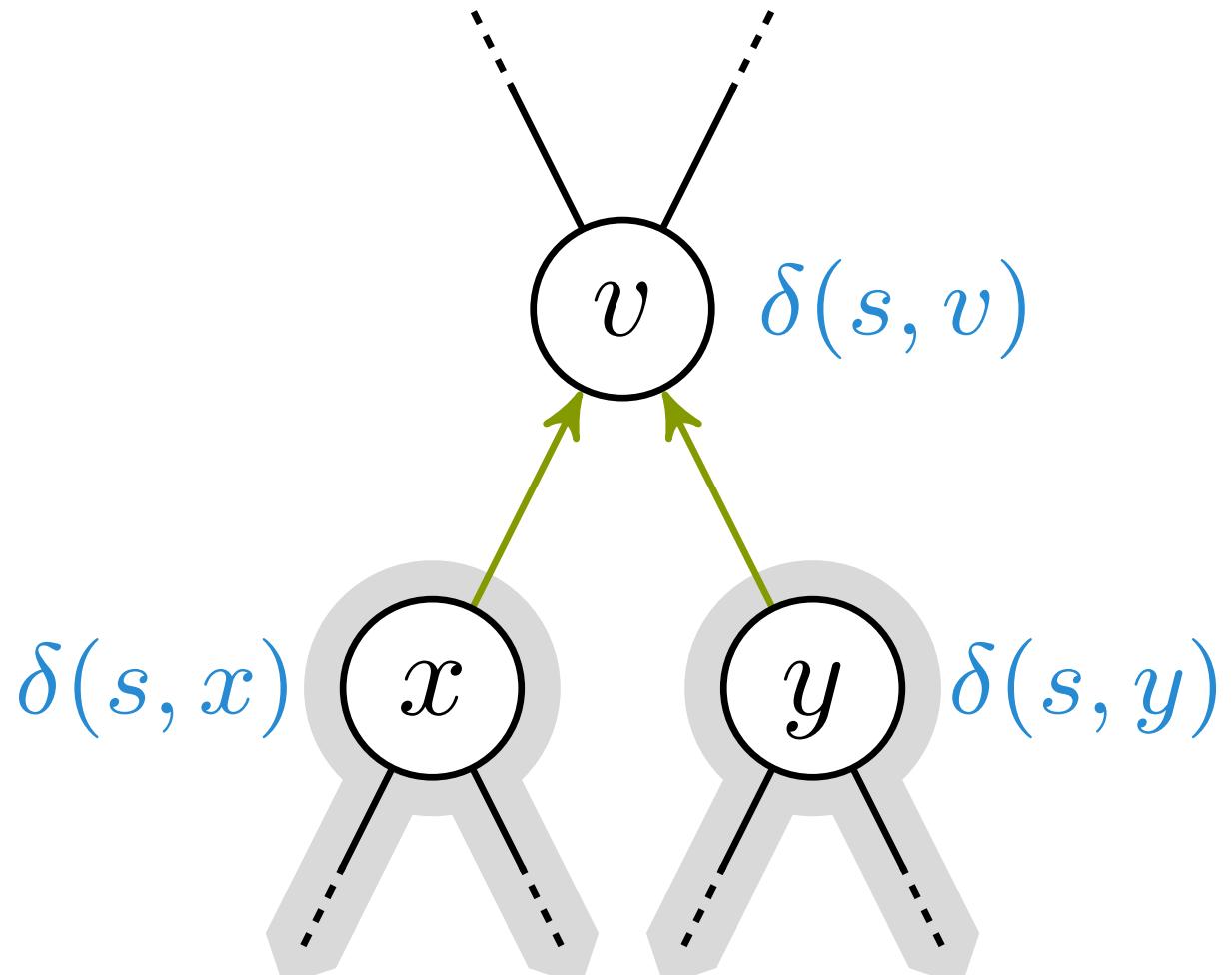
DAG-Shortest-Path



Vi vet: $\delta(s, v) \leq \delta(s, u) + w(u, v)$ for alle $u \in V$

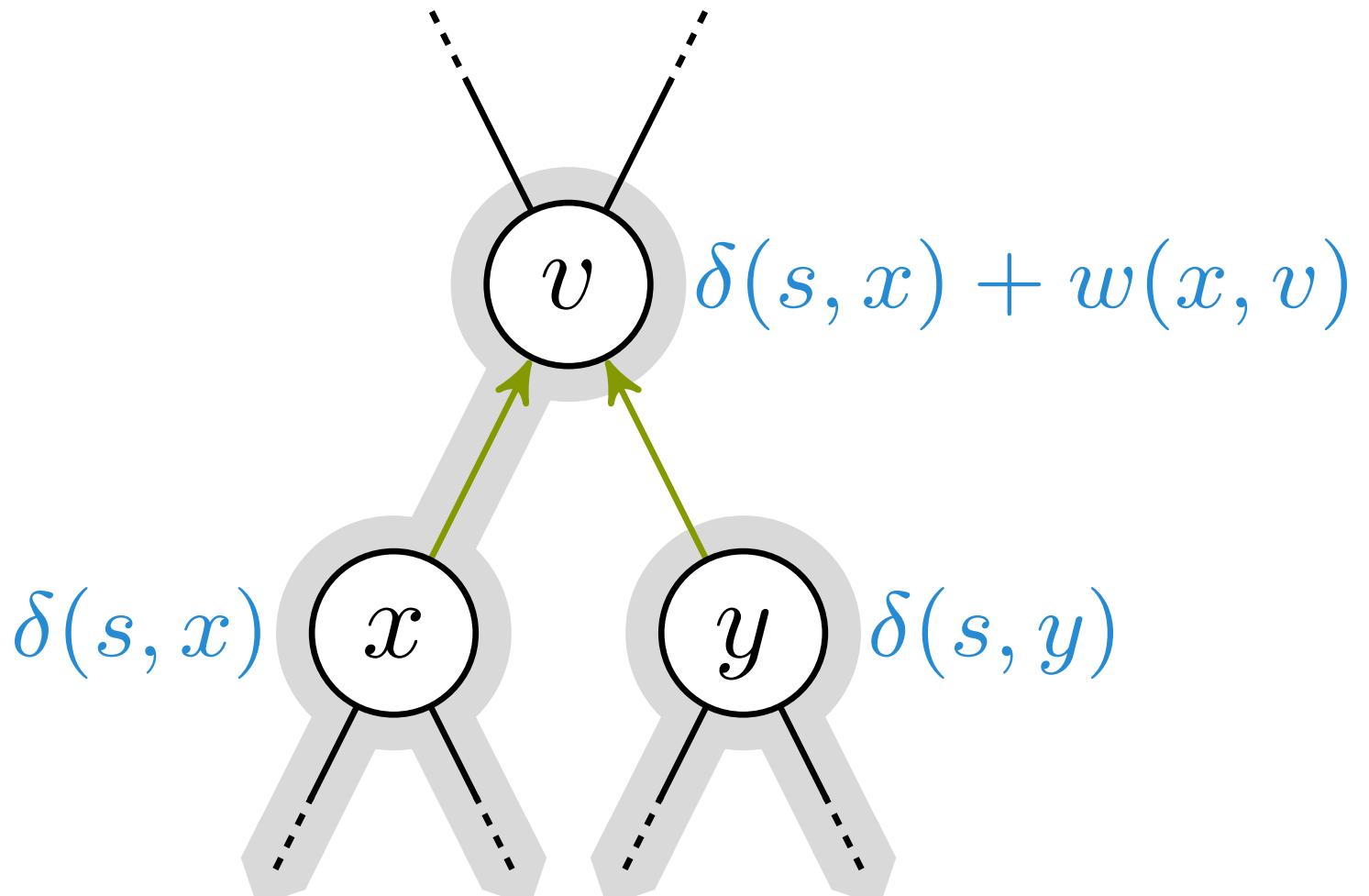


Mer presist: $\delta(s, v) = \min_u \delta(s, u) + w(u, v)$ for alle $(u, v) \in E$

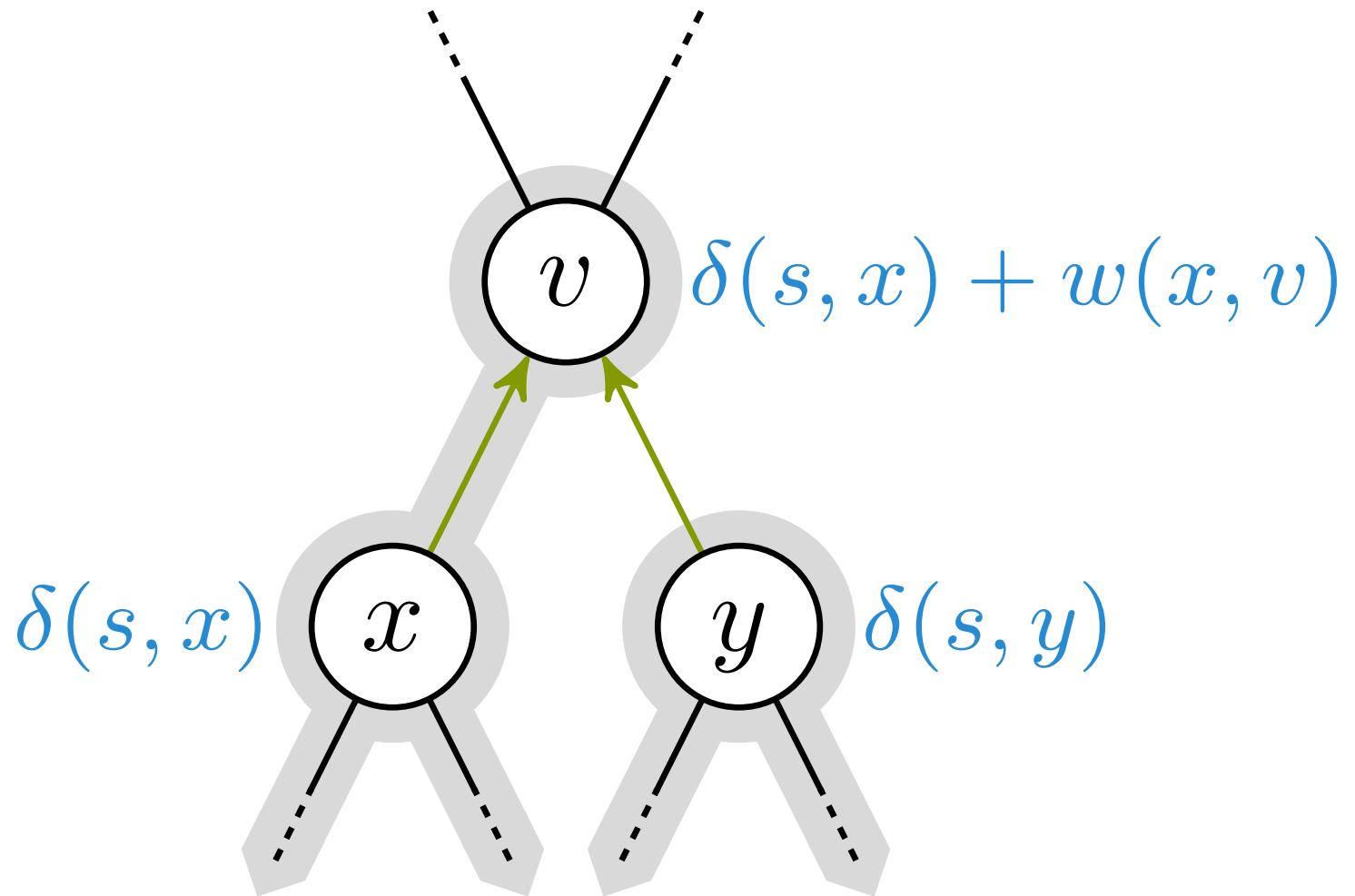


$$\text{Anta } \delta(s, x) + w(x, v) \leq \delta(s, y) + w(y, v)$$

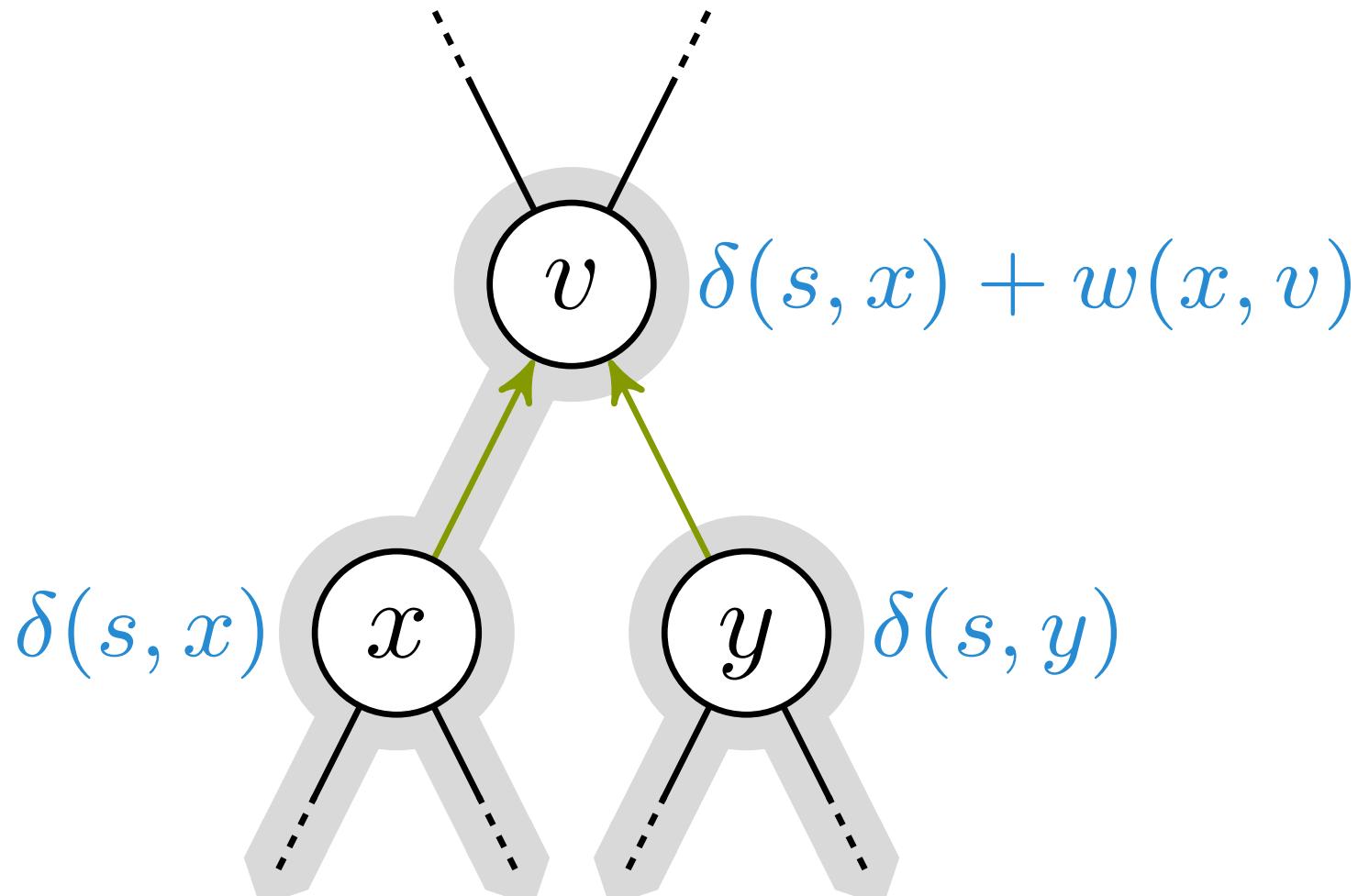
korteste vei \rightarrow slakking



$$\text{Anta } \delta(s, x) + w(x, v) \leq \delta(s, y) + w(y, v)$$



Følger av $\text{RELAX}(x, v)$ og $\text{RELAX}(y, v)$ om x og y er ferdige



Rekursiv dekomponering: $\delta(s, x)$ og $\delta(s, y)$ er delproblemer av $\delta(s, v)$

- › God mental modell for dynamisk programmering; erkeeksempel
- › Delproblemer er avstander fra s til inn-naboer; velg den som gir deg best resultat
- › Bottom-up: Kantslakking av inn-kanter i topologisk sortert rekkefølge (såkalt **pulling**)
- › Gir samme svar: Kantslakking av ut-kanter i topologisk sortert rekkefølge (såkalt **reaching**)

- › Alternativt perspektiv: Sti-slakkings-egenskapen
- › Enhver sti – og dermed enhver kortest sti – går fra venstre til høyre i topologisk sortert rekkefølge
- › Ved å slakke ut-kanter i denne rekkefølgen vil enhver kortest sti få sine kanter slakket i rekkefølge

DAG-SHORTEST-PATH(G, w, s)

- 1 topologically sort the vertices of G
- 2 INITIALIZE-SINGLE-SOURCE(G, s)
- 3 **for** each vertex u , in topsort order
- 4 **for** each vertex $v \in G.Adj[u]$
- 5 RELAX(u, v, w)

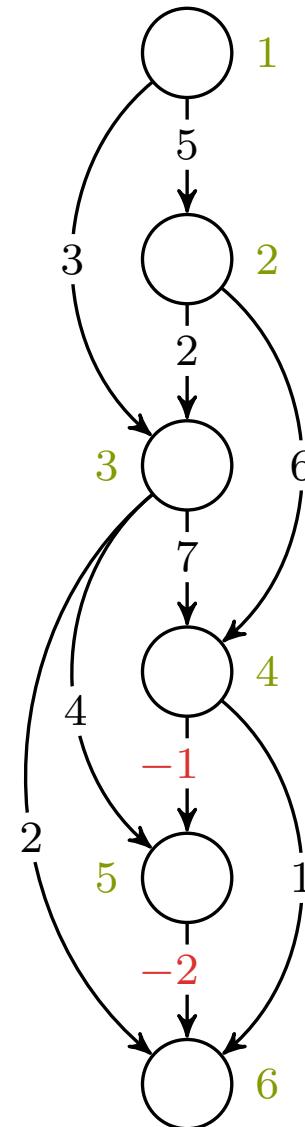
```

DAG-SHORTEST-PATH( $G, w, s$ )
1  topologically sort the vertices of  $G$ 
2  INITIALIZE-SINGLE-SOURCE( $G, s$ )
3  for each vertex  $u$ , in topsort order
   for each vertex  $v \in G.Adj[u]$ 
      RELAX( $u, v, w$ )

```

$u, v = -, -$

korteste vei \rightarrow DAG-SP



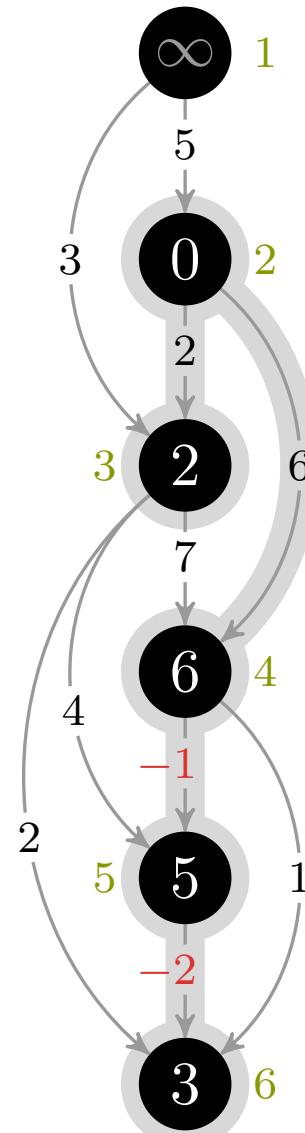
```

DAG-SHORTEST-PATH( $G, w, s$ )
1  topologically sort the vertices of  $G$ 
2  INITIALIZE-SINGLE-SOURCE( $G, s$ )
3  for each vertex  $u$ , in topsort order
4      for each vertex  $v \in G.Adj[u]$ 
5          RELAX( $u, v, w$ )

```

$$u, v = -, -$$

korteste vei → DAG-SP

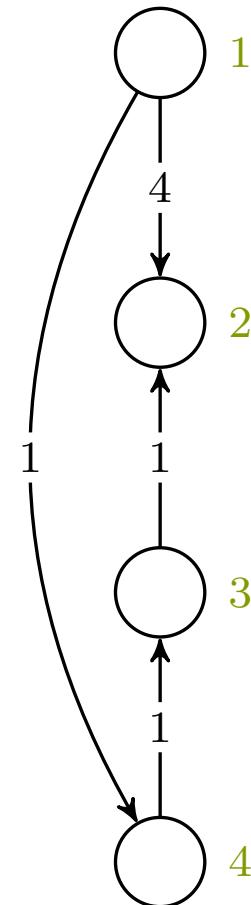


```

DAG-SHORTEST-PATH( $G, w, s$ )
1  topologically sort the vertices of  $G$ 
2  INITIALIZE-SINGLE-SOURCE( $G, s$ )
3  for each vertex  $u$ , in topsort order
4    for each vertex  $v \in G.Adj[u]$ 
5      RELAX( $u, v, w$ )

```

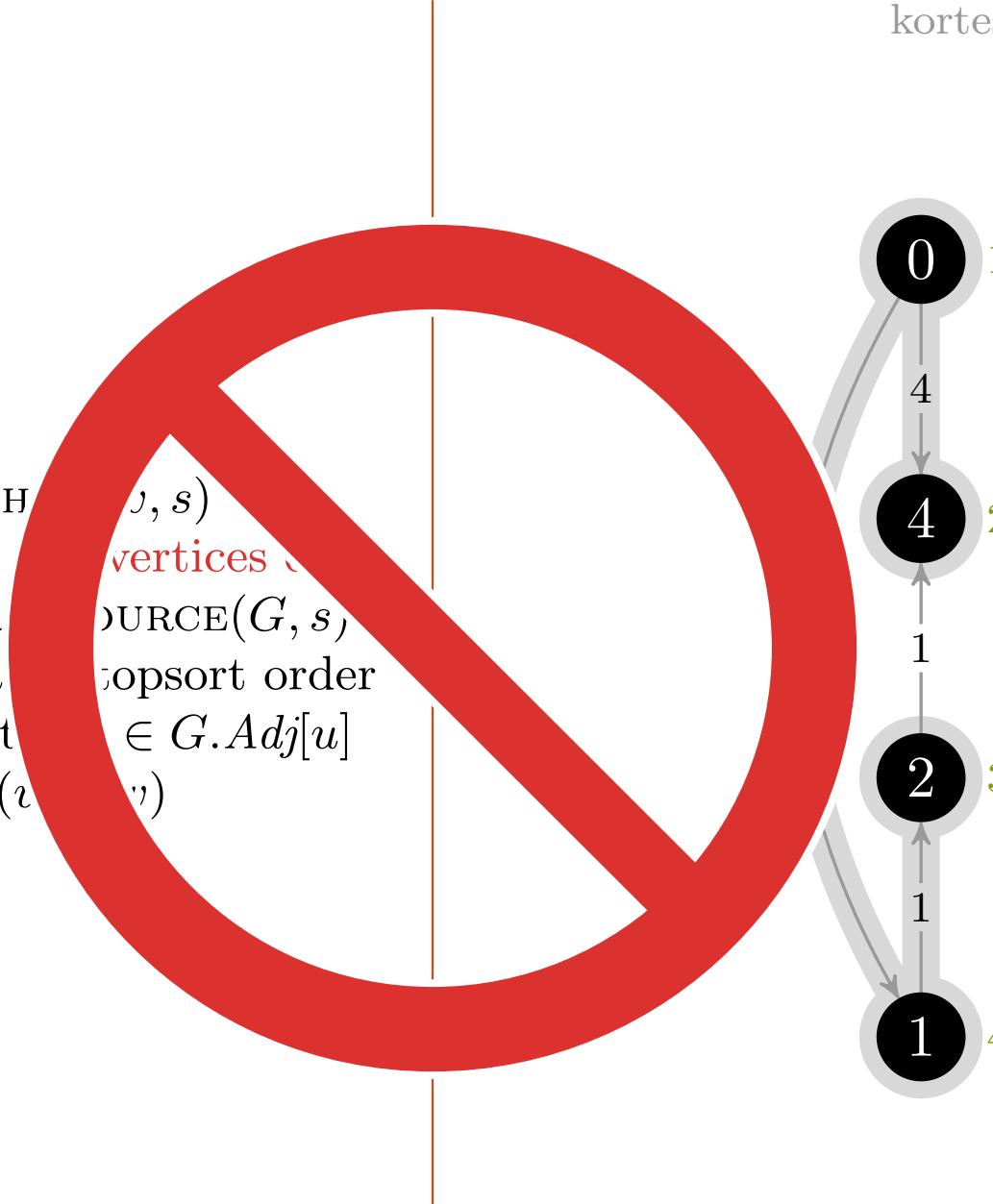
$u, v = -, -$



DAG-SHORTEST-PATH(G, s)

- 1 **topologically** sort vertices $\langle v_1, v_2, \dots, v_n \rangle$
- 2 INITIALIZE-SINGLE-SOURCE(G, s)
- 3 **for** each vertex $u \in V$ **do** dopsort order
- 4 **for** each vertex $v \in G.\text{Adj}[u]$ **do**
- 5 RELAX(u, v)

$u, v = -, -$



Sykler forbudt!

Korteste vei ›
DAG-SP › **Kjøretid**

Operasjon	Antall	Kjøretid
Topologisk sortering	1	$\Theta(V + E)$
Initialisering	1	$\Theta(V)$
RELAX	E	$\Theta(1)$

Totalt: $\Theta(V + E)$

Korteste vei ›

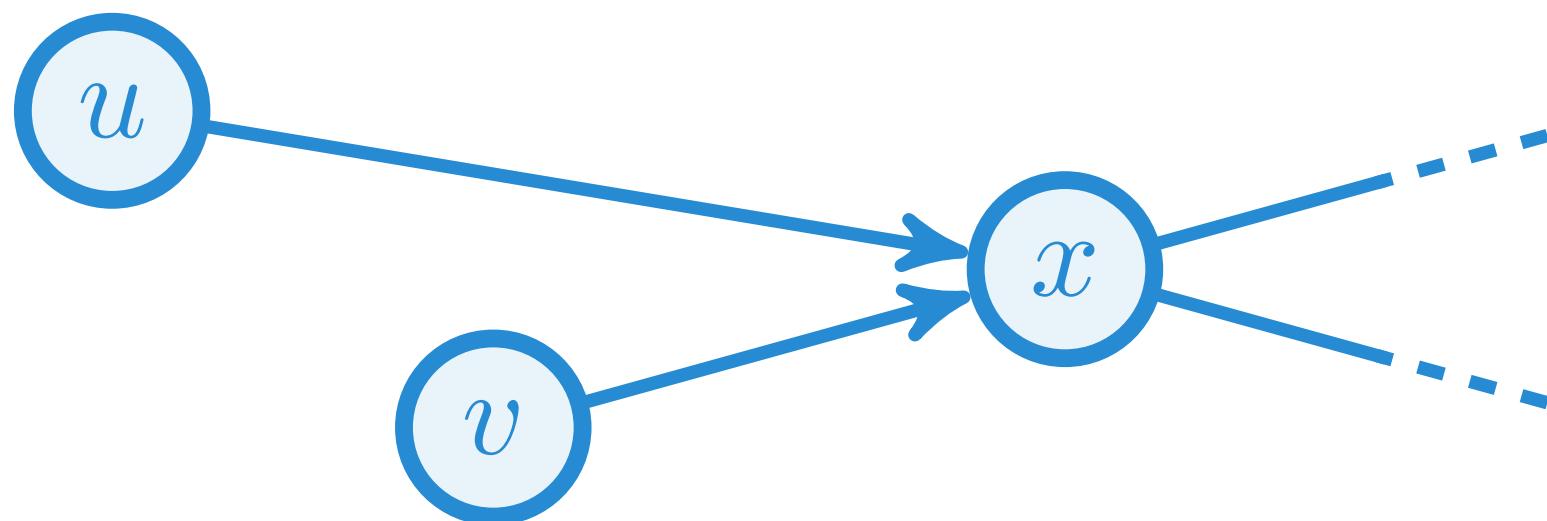
Dijkstras algoritme

- › Om vi har sykler, kan vi ikke få til topologisk sortering
- › Alternativ: Besøke nodene i stigende avstandsrekkefølge
- › Alle korteste stier får da fortsatt sine kanter slakket i riktig rekkefølge
- › Men vi kjenner jo ikke avstandsrekkefølgen!

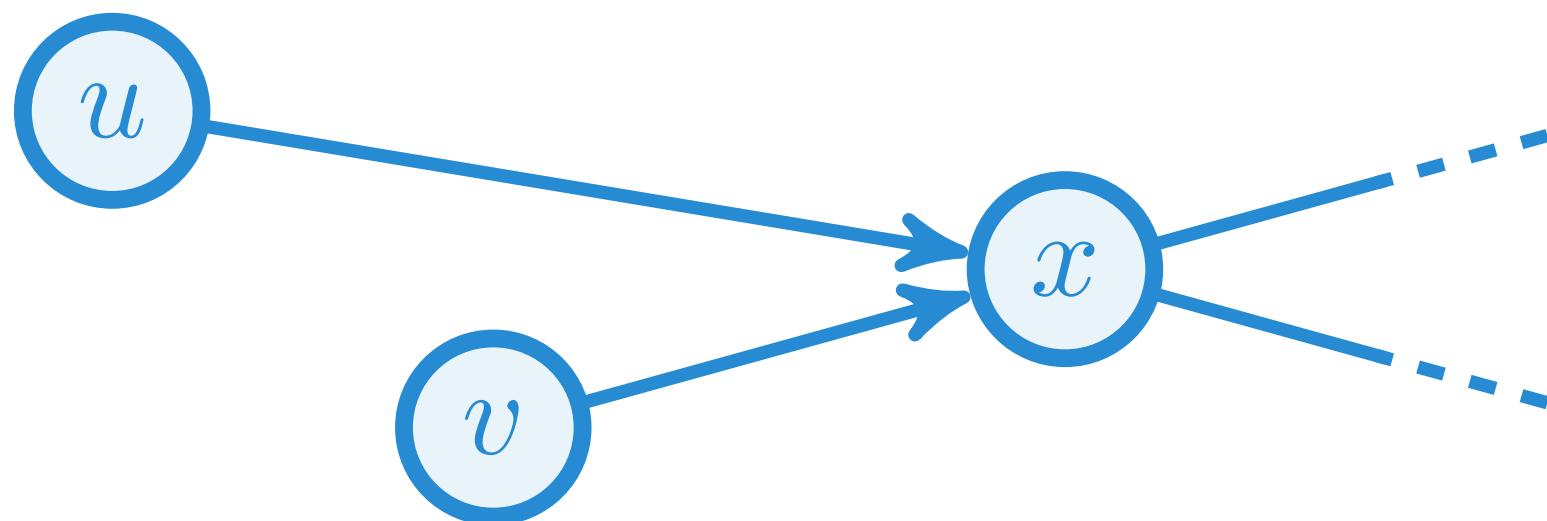
Fakkelstafett



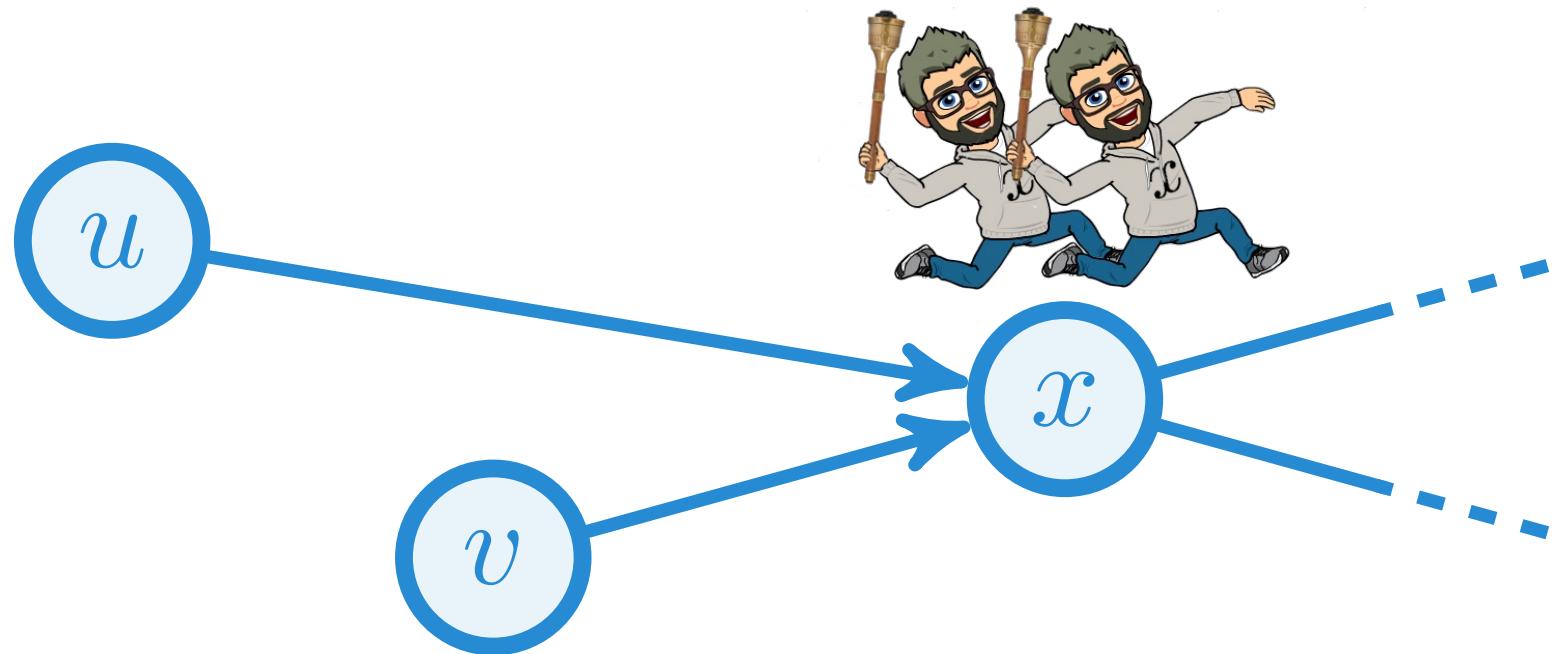
... med
forgreninger



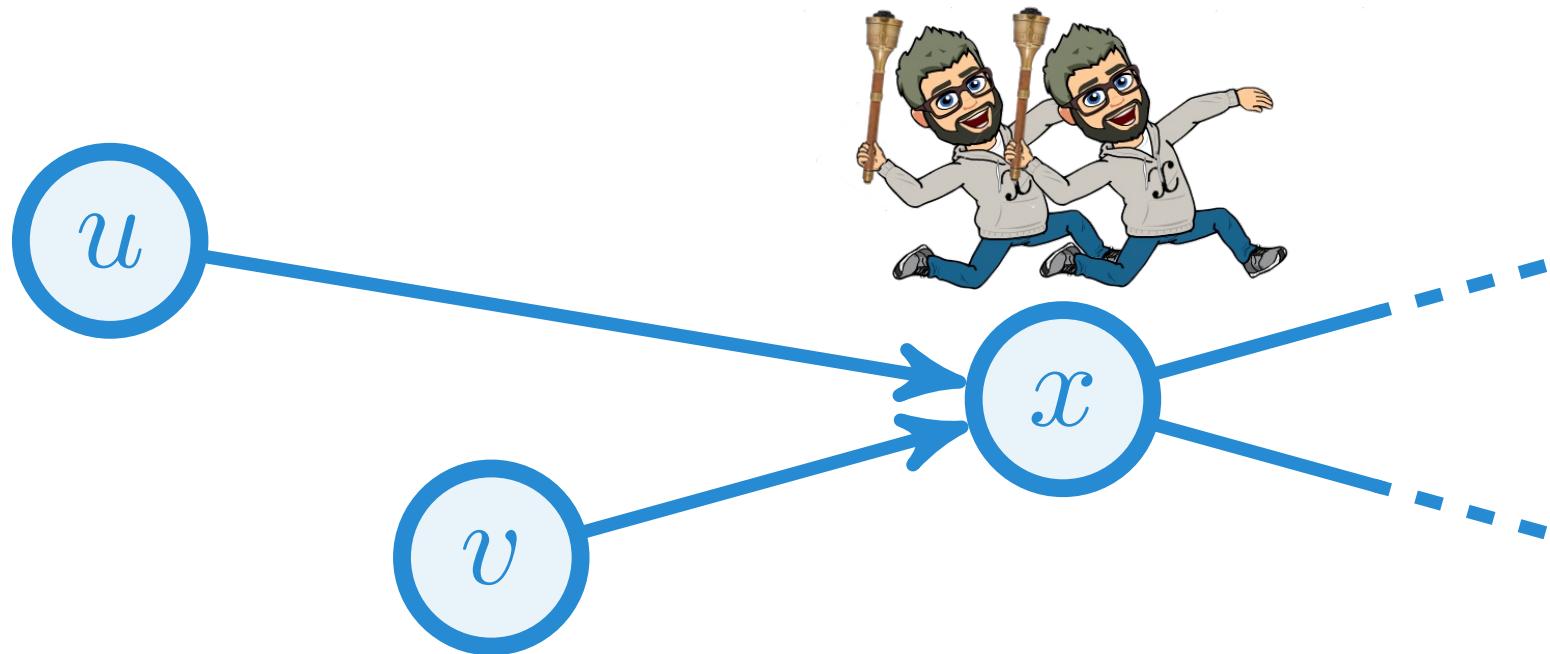
Kanter og noder er etapper og vekslingssoner i en fakkelstafett ...



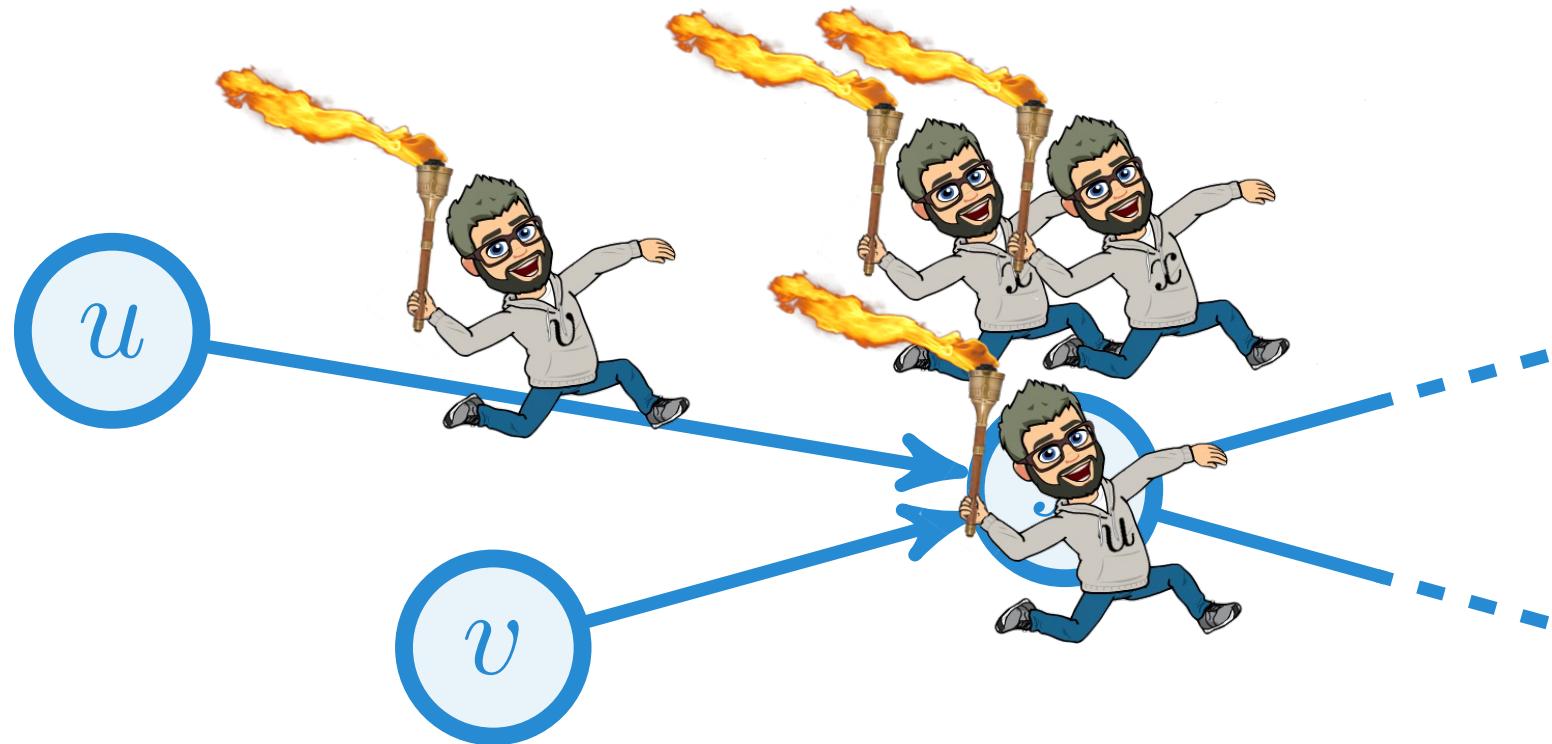
... der vi har forgreninger og parallele etapper



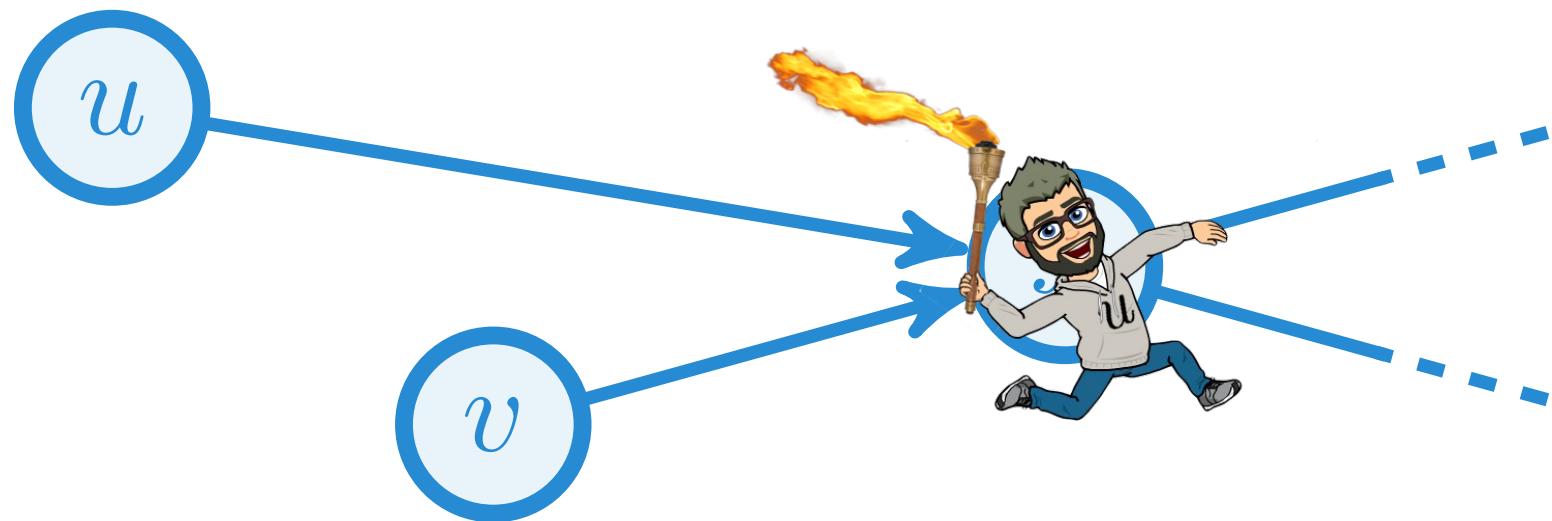
En vekslingssone x med k ut-etapper har k løpere merket x



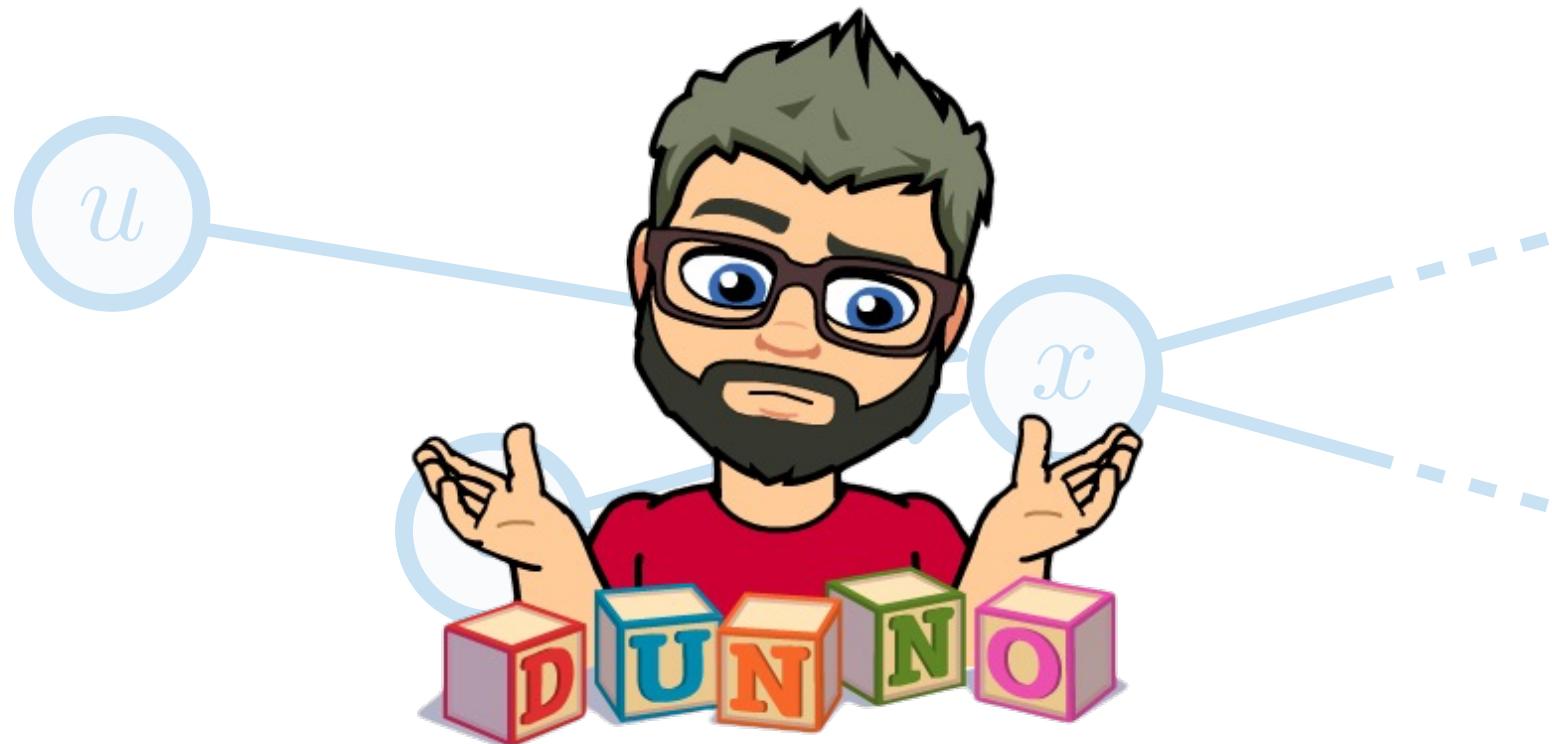
Løperne i s starter ved tid 0; alle løper like fort



Første ankomsttid til vekslingssone x gir oss avstanden!



Hvilken løper som ankommer x først gir oss ruta frem til x !



Hvordan kan vi simulere dette effektivt?

- › **Avstandsestimat: tidligste ankomsttid funnet**
- › **Kan forbedres, men bare via andre vekslingssoner**
- › **Anta at vi ikke har noen negative kanter (... tidsreise?!)**
- › **Da kan ingen få lavere ankomsttid enn den laveste så langt**
- › **Denne ankomsttiden må dermed være riktig!**
- › **Dette gjelder som en invariant, for ubesøkte noder**
- › **Simulering: Bruk en min-prioritetskø med avstandsestimat**

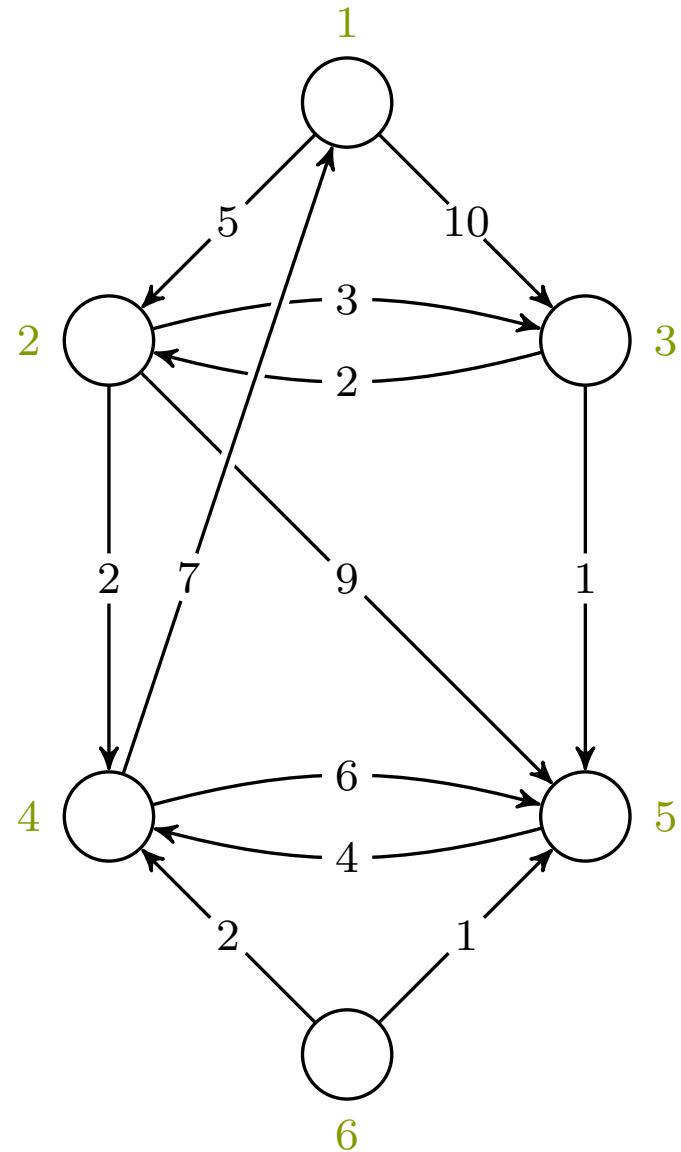
```
DIJKSTRA( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5     $u = \text{EXTRACT-MIN}(Q)$ 
6     $S = S \cup \{u\}$ 
7    for each vertex  $v \in G.Adj[u]$ 
8      RELAX( $u, v, w$ )
```

DIJKSTRA(G, w, s)

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

$u, v = -, -$

korteste vei \rightarrow dijkstra



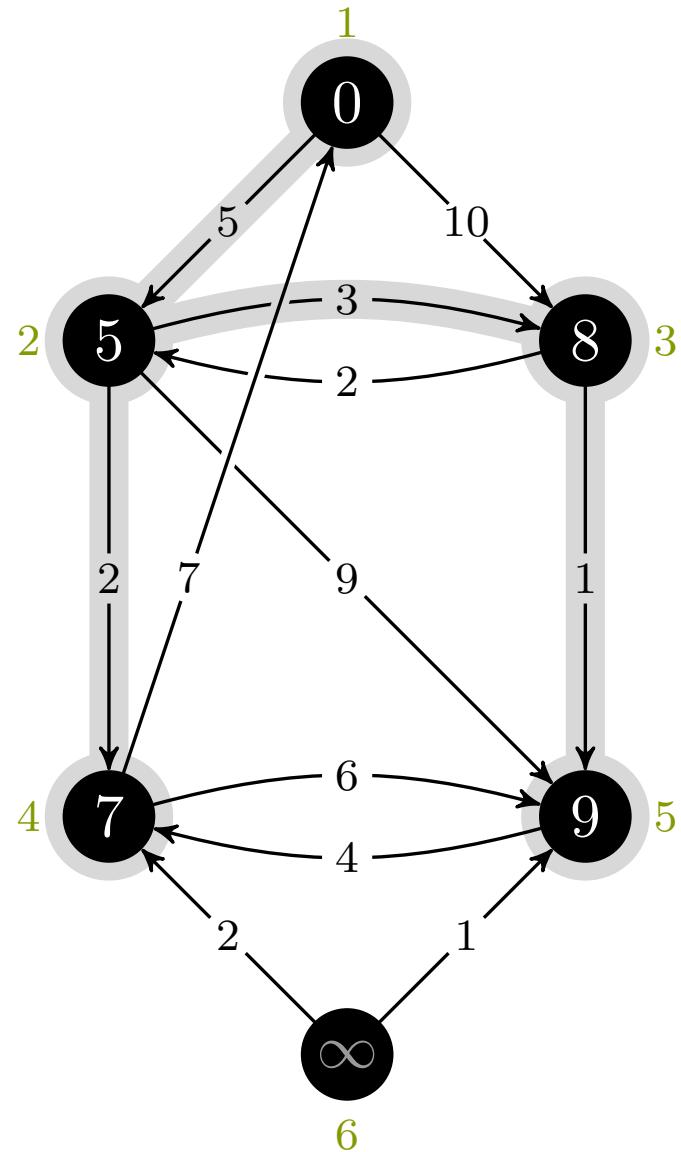
```

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

```

$u, v = -, -$

korteste vei \rightarrow dijkstra

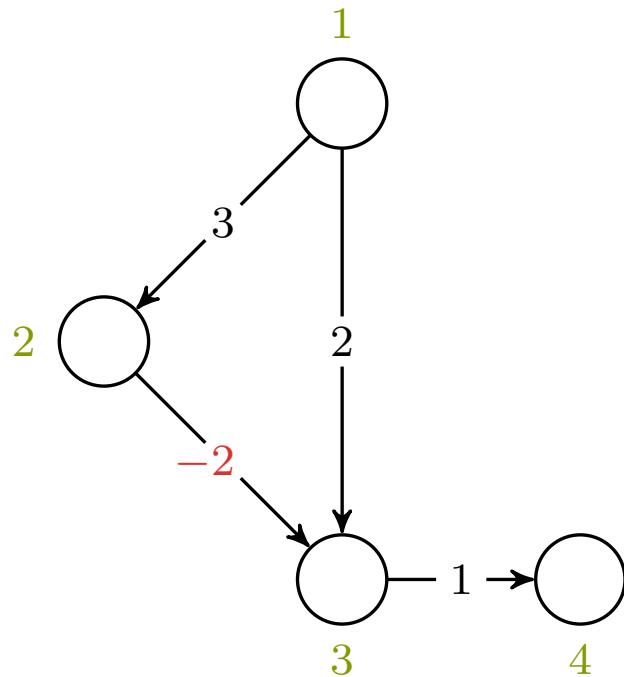


korteste vei \rightarrow dijkstra

DIJKSTRA(G, w, s)

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

$u, v = -, -$



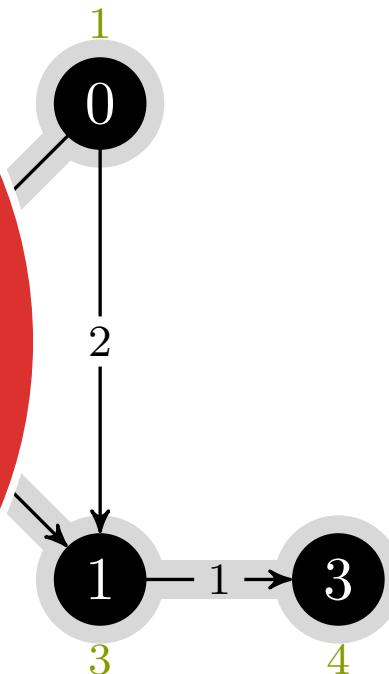
korteste vei → dijkstra

DIJKSTRA(G, w, s)

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

$u, v = \text{---}$

Negative kanter forbudt!



Dette eksemplet kunne vi ha løst korrekt - og mer effektivt! - med DAG-Shortest-Path.

Korteste vei ›

Dijkstra › **Kjøretid**

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	$O(\lg V)$

Totalt: $O(V \lg V + E \lg V)$

*Nødvendig i RELAX

Boka bruker $V \times \text{INSERT}$; fortsatt $O(E \lg V)$

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	$O(\lg V)$

Totalt: $O(V \lg V + E \lg V)$

*Nødvendig i RELAX

Fibonacci-heaps: EXTRACT-MIN er $O(1)$; vi får $O(V \lg V + E)$

