Forelesning 9

Her har vi en graf med vekter på kantene, og ønsker å bare beholde akkurat de kantene vi må for å koble sammen alle nodene, med en så lav vektsum som mulig. Erke-eksempel på grådighet: Velg én og én kant, alltid den billigste lovlige.

Pensum

- ☐ Kap. 21. Data structures for disjoint sets: Innledning, 21.1 og 21.3
- ☐ Kap. 23. Minimum spanning trees

Læringsmål

- [I_1] Forstå *skog*-implementasjonen av *disjunkte mengder*
- [I_2] Vite hva *spenntrær* og *minimale spenntrær* er
- [I_3] Forstå Generic-MST
- [I_4] Forstå hvorfor *lette kanter* er *trygge kanter*
- [I_5] Forstå MST-Kruskal
- [I_6] Forstå MST-Prim

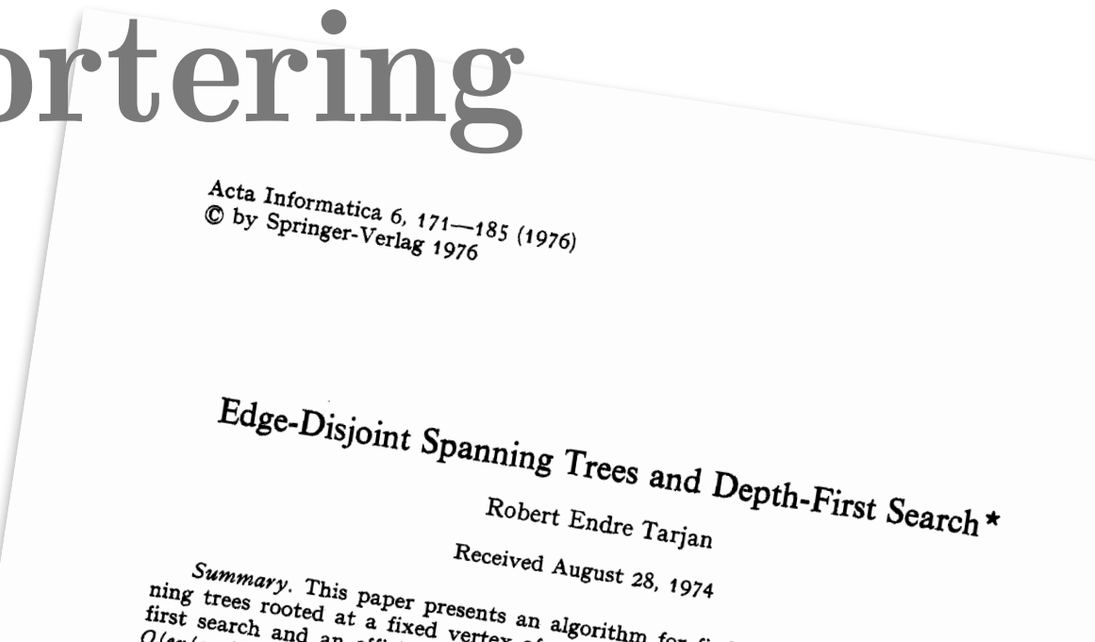
mlh@ntnu.no

Forelesningen filmes

0:4

Topologisk sortering

Det finnes flere algoritmer for dette – men varianten som bruker DFS ble trolig først beskrevet av Tarjan.



Acta Informatica 6, 171—185 (1976)
© by Springer-Verlag 1976

Edge-Disjoint Spanning Trees and Depth-First Search*

Robert Endre Tarjan

Received August 28, 1974

Summary. This paper presents an algorithm for finding trees rooted at a fixed vertex of
first search and an

Kjøretid, traversering

Algoritme	WC	BC
Dybde-først-søk	$\Theta(V + E)$	
Bredde-først-søk	$\Theta(V + E)$	

Alle noder farges grå og svarte; alle kanter undersøkes

Algoritme	WC	BC
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	

DFS starter fra alle noder etter tur, så topsort skal funke!

Algoritme	WC	BC
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	

(Generelt går det også fint å kjøre DFS-Visit fra bare én node)

Algoritme	WC	BC
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	$\Theta(V)$

Grunnen til at best-case ikke er O(1) er at vi må initialisere alle nodene. Dersom vi hadde brukt f.eks. en hashtabell til å holde denne informasjonen, og kun lagt den inn etter behov, så kunne vi ha fått O(1) – men det er altså ikke slik boka gjør det.

BFS starter ikke fra alle i vår versjon (men kunne ha gjort det)

Algoritme	WC	BC
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	$\Theta(V)$

F.eks. Edmonds-Karp trenger stier fra én bestemt node

Algoritme	**WC**	**BC**
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	$\Theta(V)$
_____-først-søk	$\Omega(V + E)$	

Samme logikk for andre prioriteringer, men kan ha dyrere kø!

Algoritme	**WC**	**BC**
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	$\Theta(V)$
_____-først-søk	$\Omega(V + E)$	

Binærhaug (Prim, Dijkstra): $\Theta(V \lg V + E \lg E) =$

Algoritme	**WC**	**BC**
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	$\Theta(V)$
_____-først-søk	$\Omega(V + E)$	

Binærhaug (Prim, Dijkstra): $\Theta(V \lg V + E \lg E) = \Theta(E \lg V)$

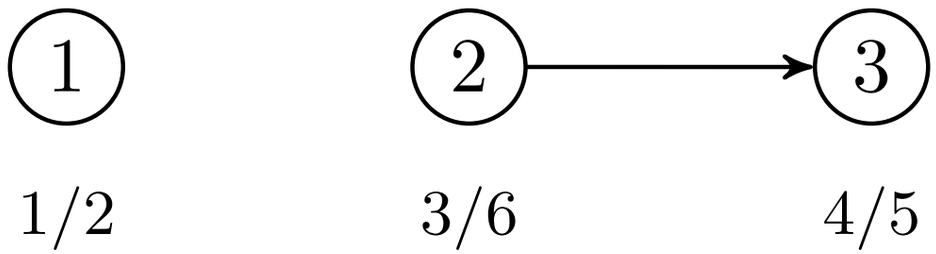
Algoritme	**WC**	**BC**
Dybde-først-søk	$\Theta(V + E)$	$\Theta(V + E)$
Bredde-først-søk	$\Theta(V + E)$	$\Theta(V)$
_____-først-søk	$\Omega(V + E)$	$\Theta(V)$

Bortsett fra i DFS starter vi vanligvis kun ett sted

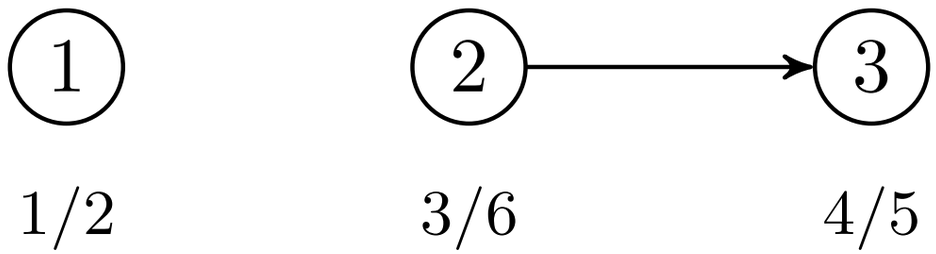
Topologisk sortering

- **Gir nodene en rekkefølge**
- **Foreldre før barn**
- **Evt.: Alle kommer etter avhengigheter**
- **Det er egentlig det vi gjør med delproblemgrafen i dynamisk programmering**
- **Krever at grafen er en DAG!**

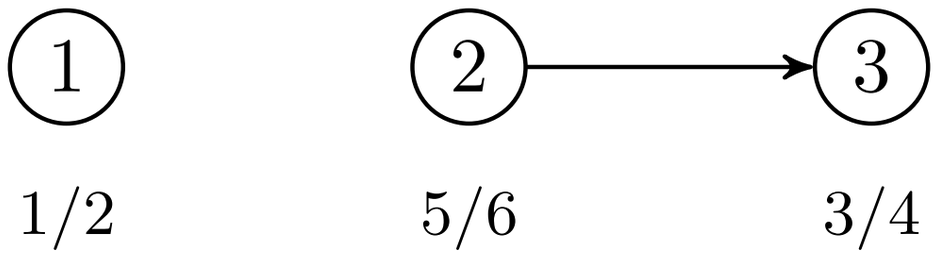
Her må 2 komme før 3; ellers fritt frem



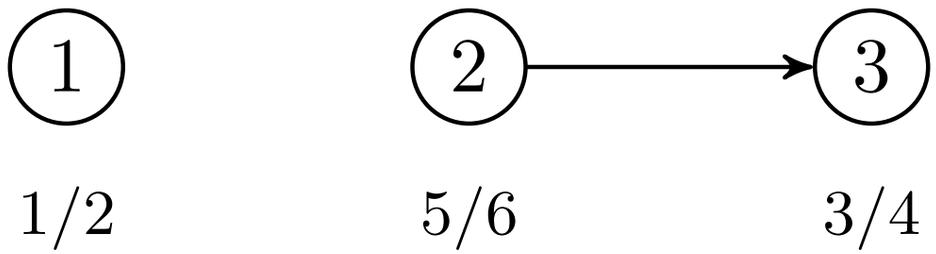
Mulige *discover-/finish-times*



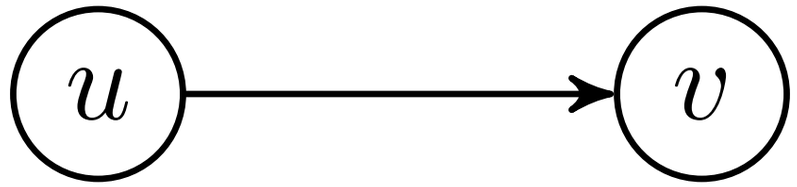
Her oppdages 2 før 3, men det er ikke garantert!



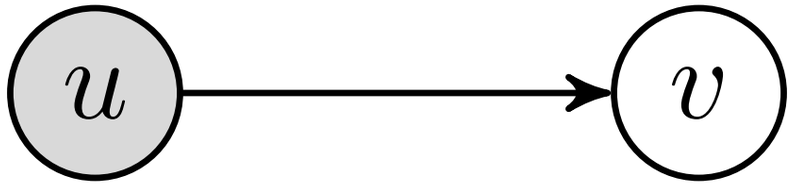
Her kalles DFS-Visit(G, 3) før DFS-Visit(G, 2)



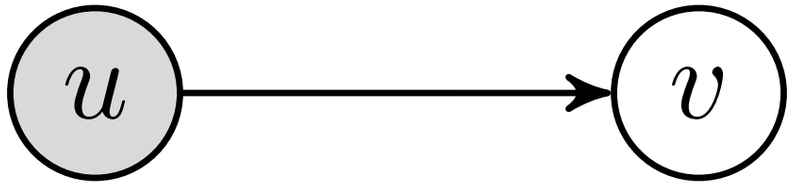
Merk at 2 fortsatt har høyere finish-time enn 3!



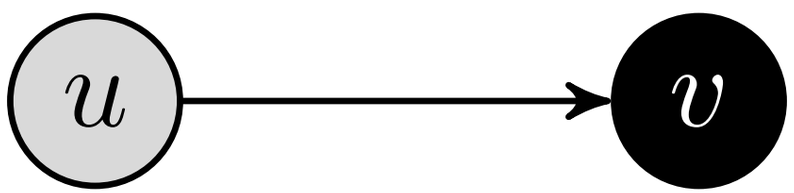
La oss si vi undersøker kanten (u, v) under DFS



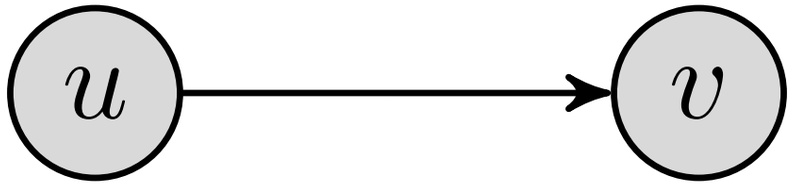
La oss si vi undersøker kanten (u, v) under DFS



Hvis v er hvit, så utforskers den rekursivt: $u.f > v.f$



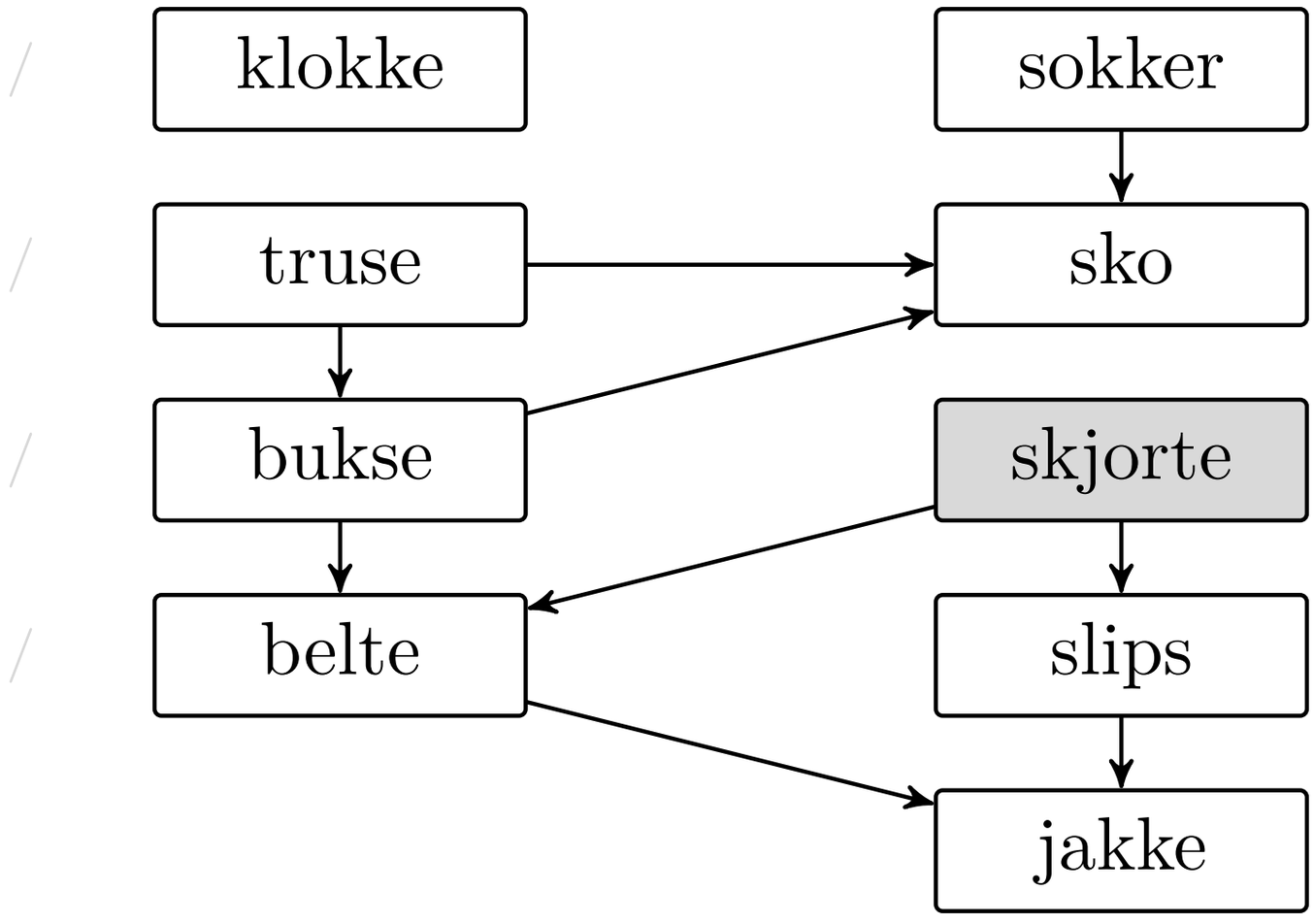
Hvis v er svart, så er den alt ferdig: $u.f > v.f$

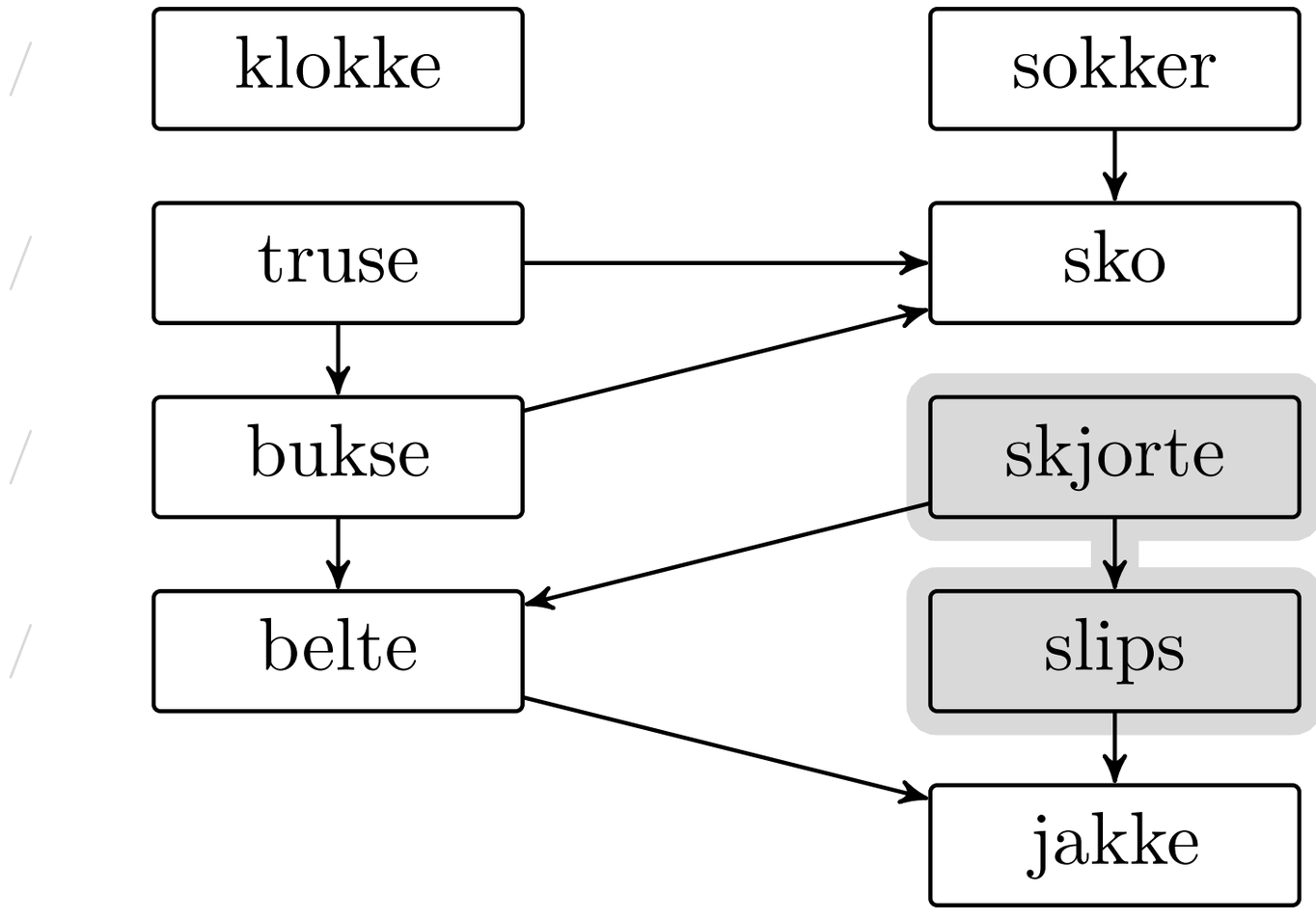


Hvis v er grå, så har vi en sykel – og det er umulig!

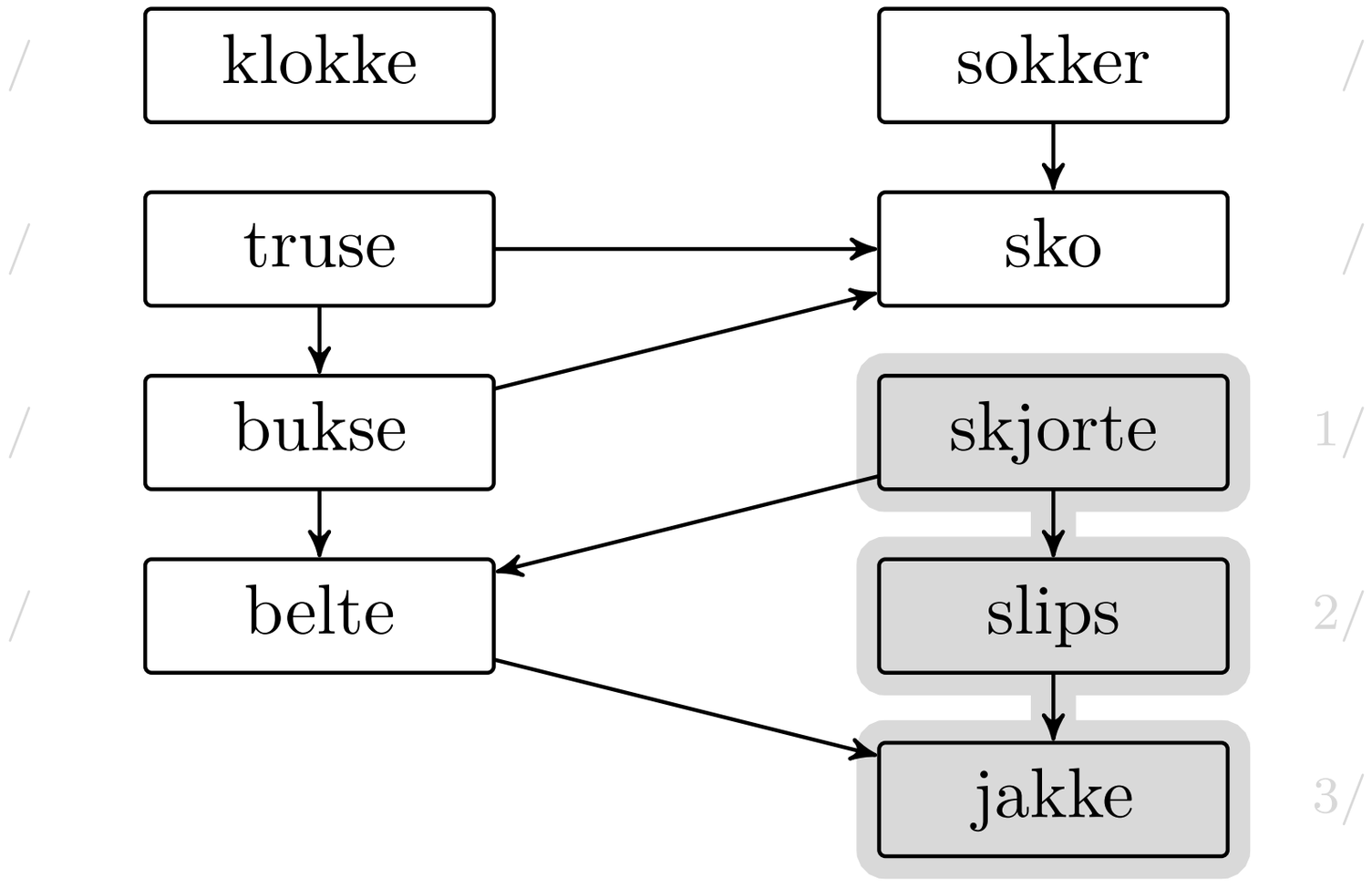
Altså ...

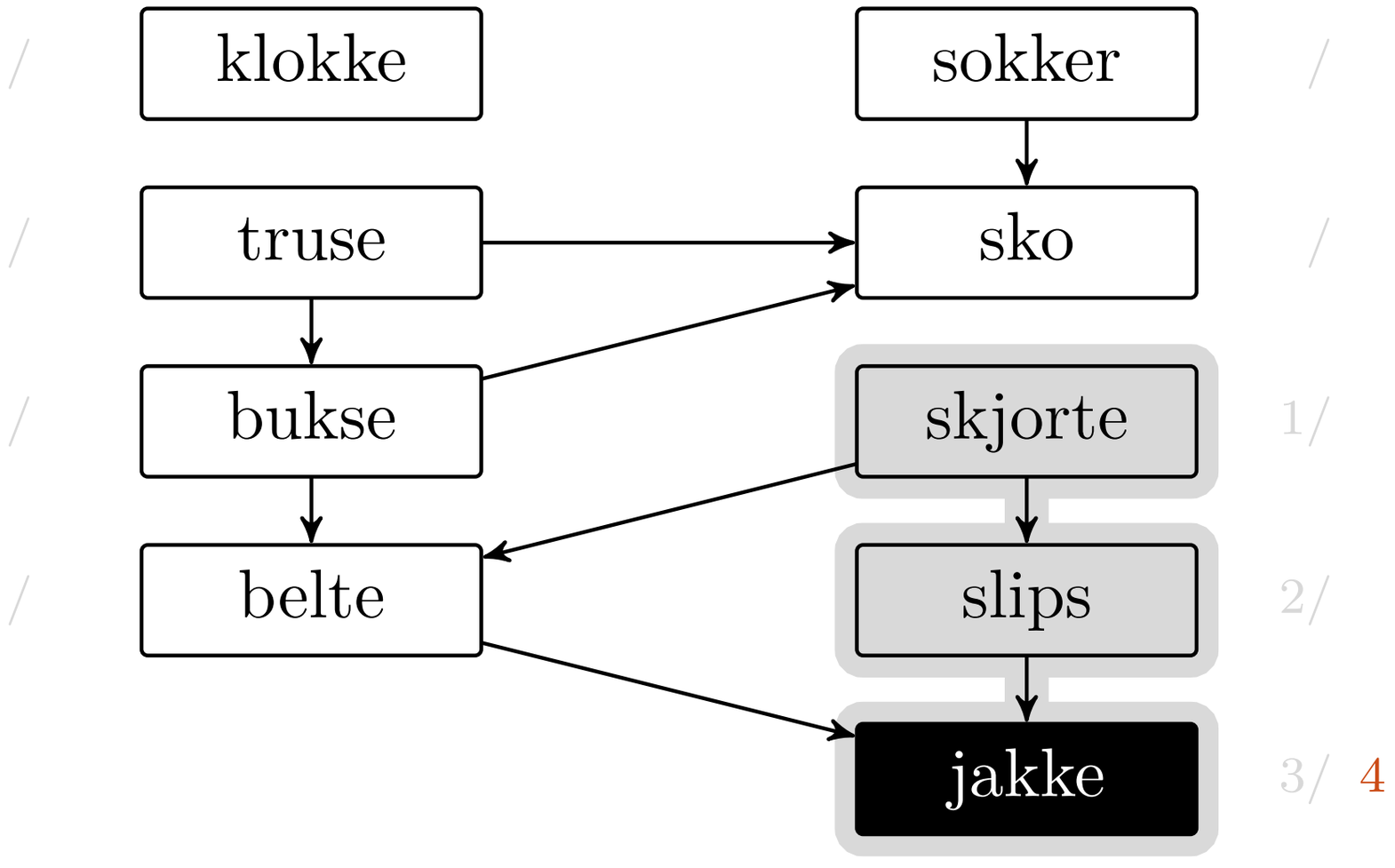
- Stigende discover-tid: Ikke trygt
- Synkende finish-tid: Trygt
- Dvs.: Det gir en topologisk sortering



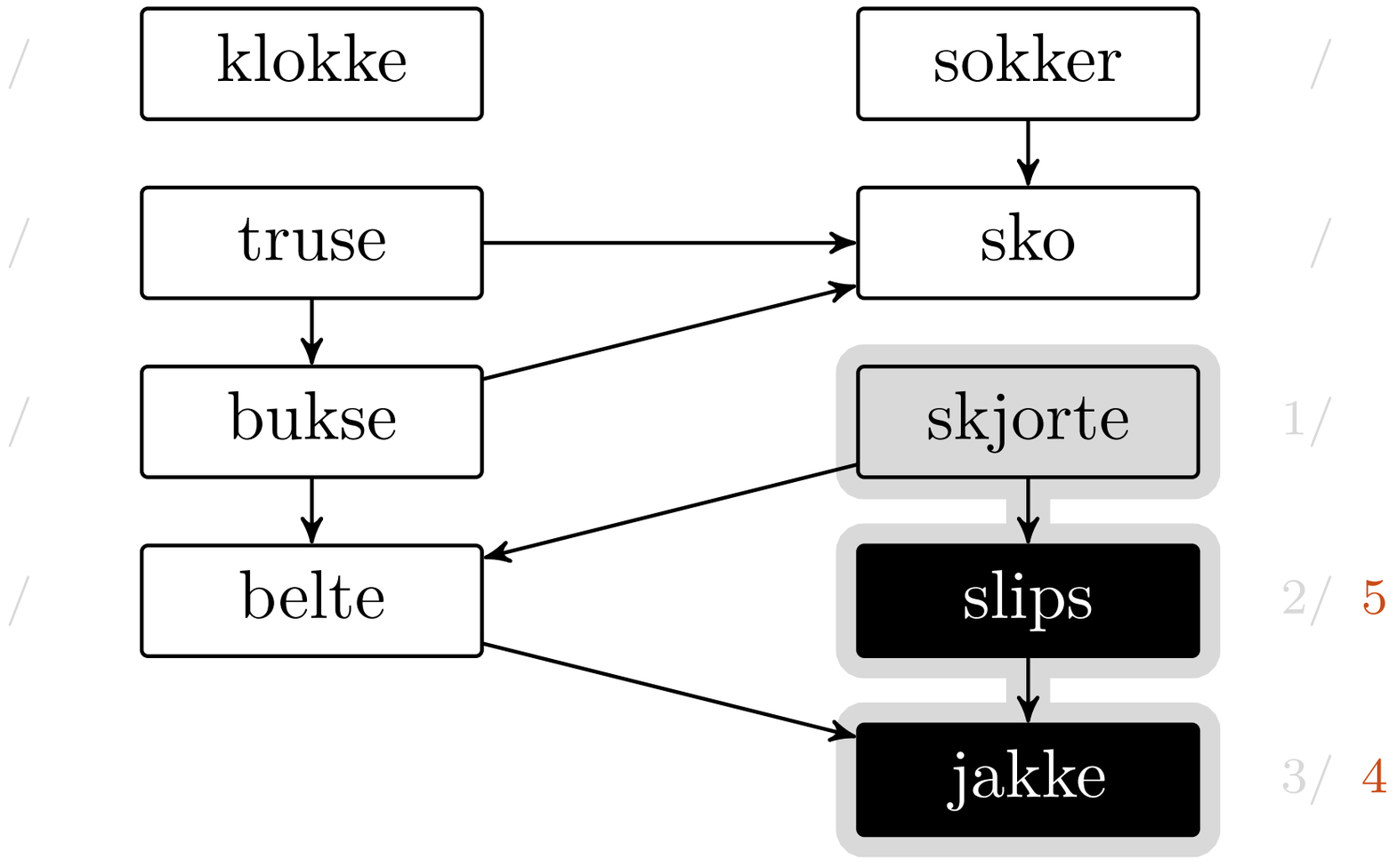


mlh@ntnu.no


klokke
sokker
truse
sko
bukse
skjorte
1/
belte
slips
2/
jakke
3/

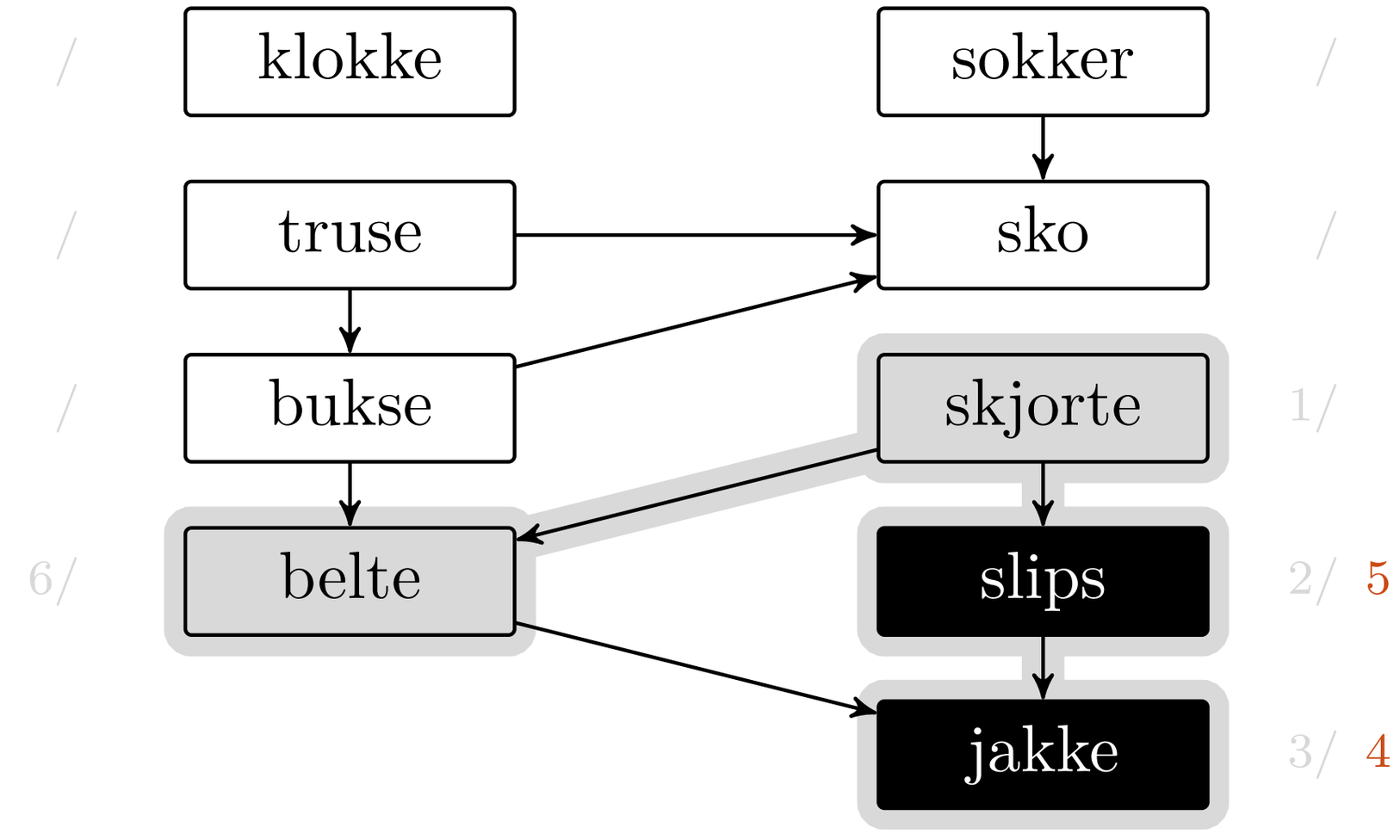


1/ 2/ 3/ 4

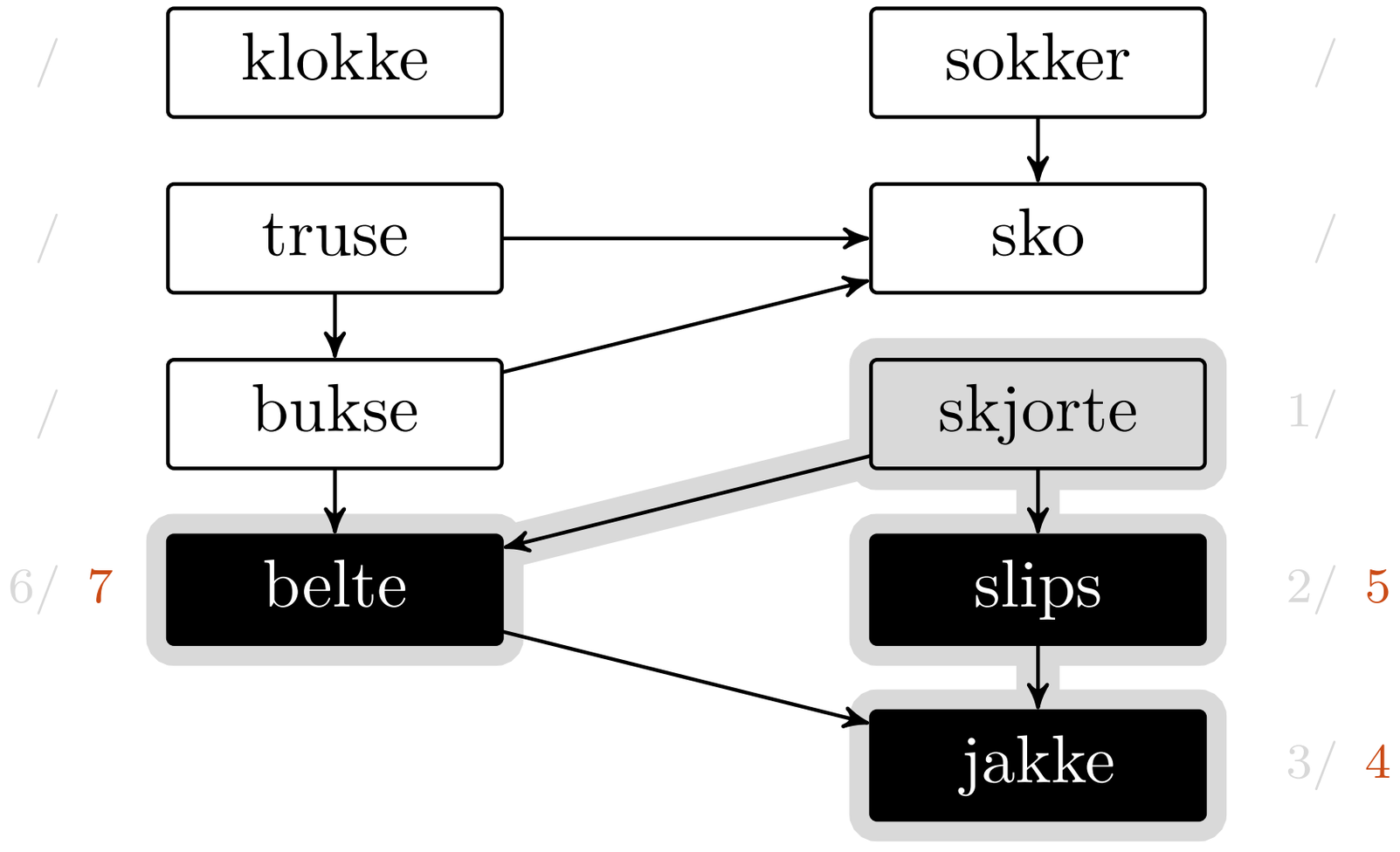


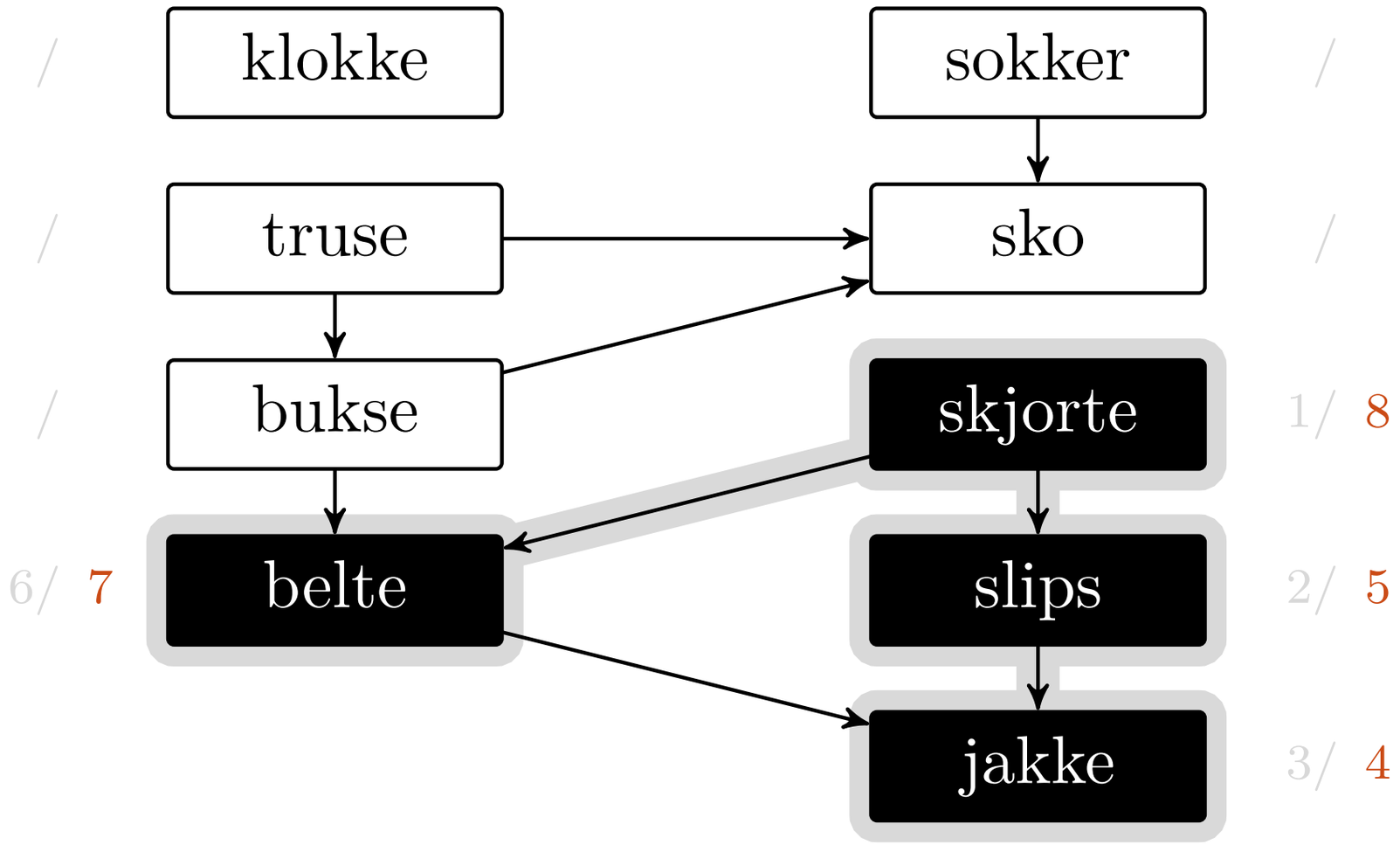
/ 1/ / 2/ 5 / 3/ 4

mlh@ntnu.no



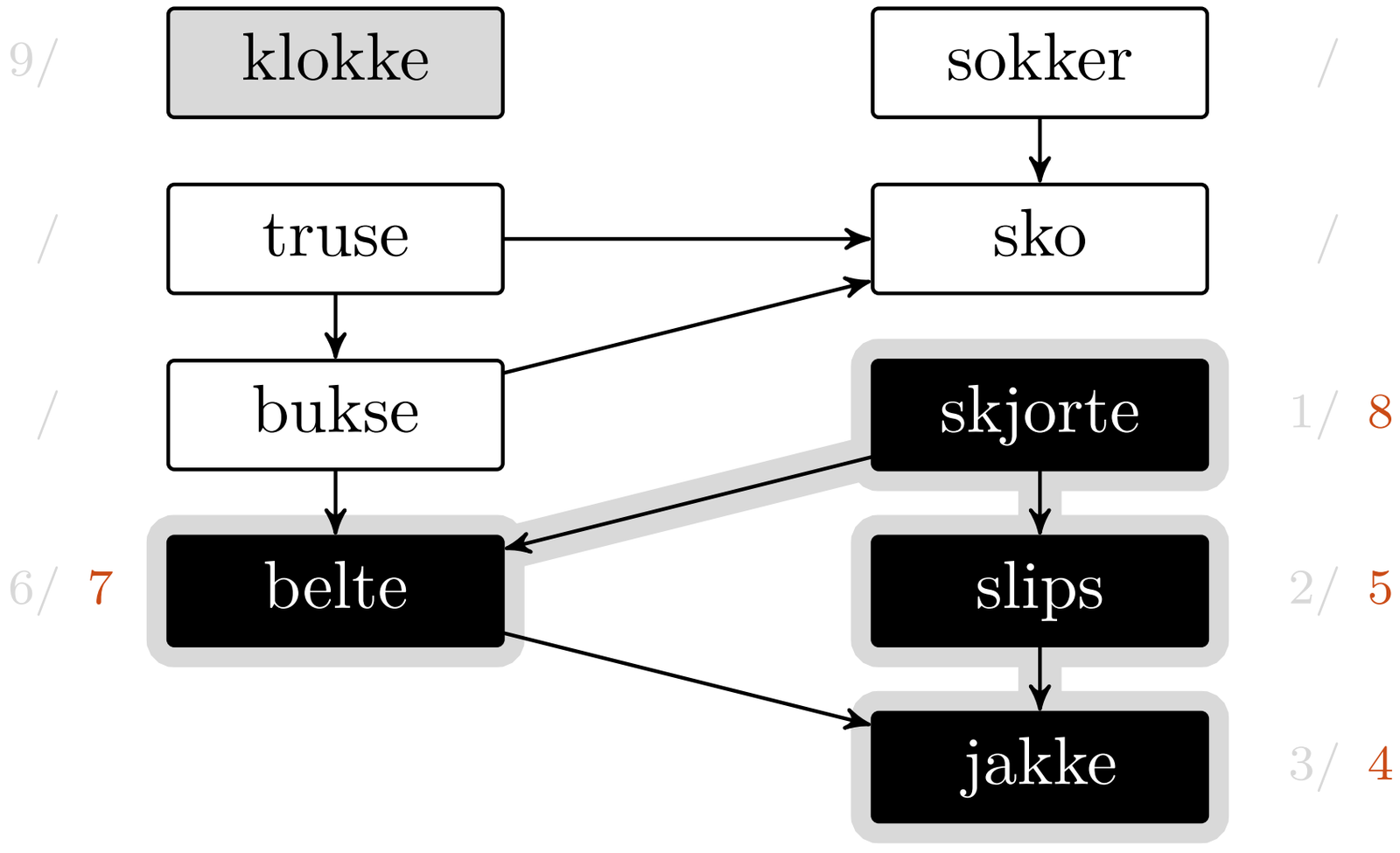
mlh@ntnu.no

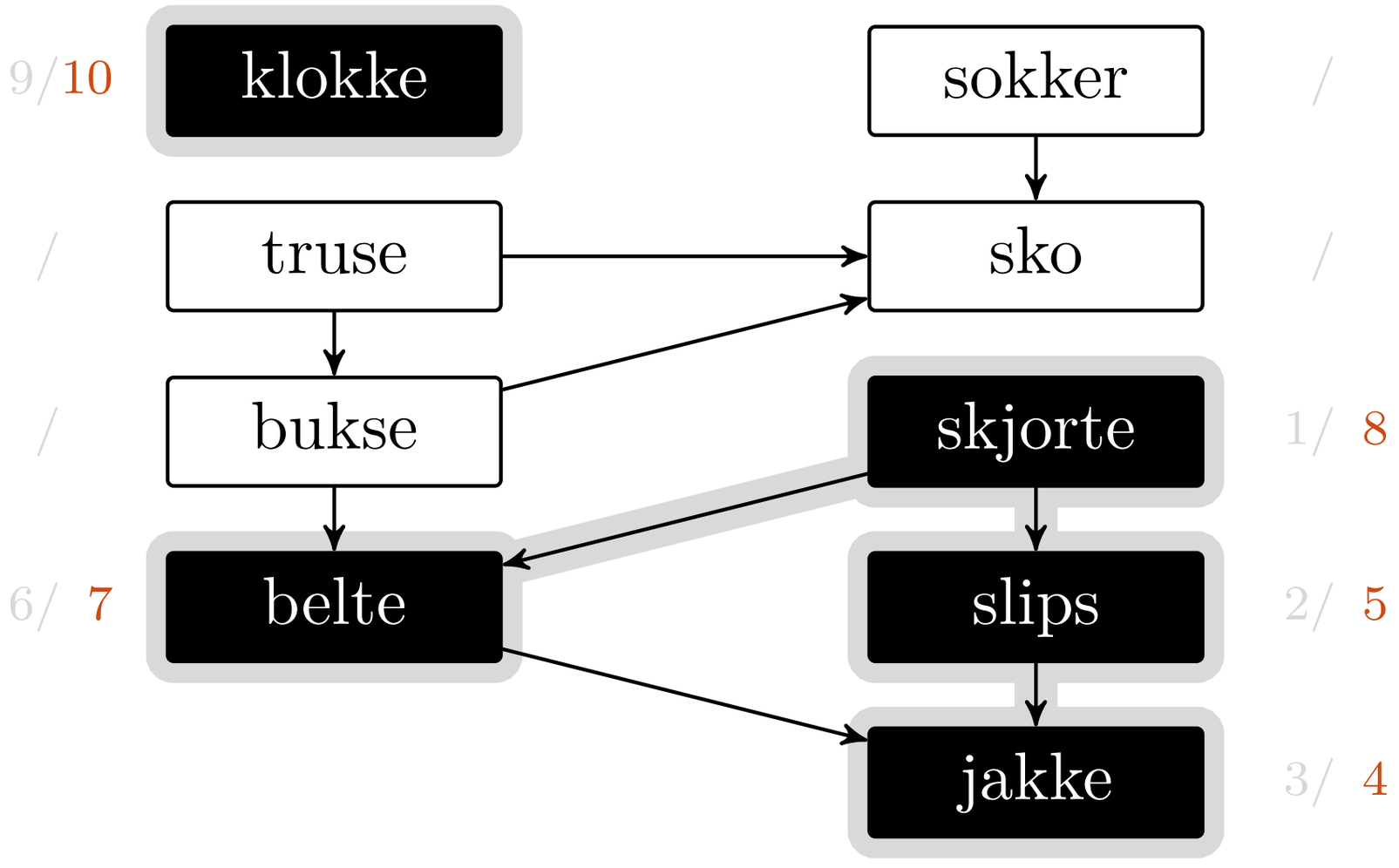




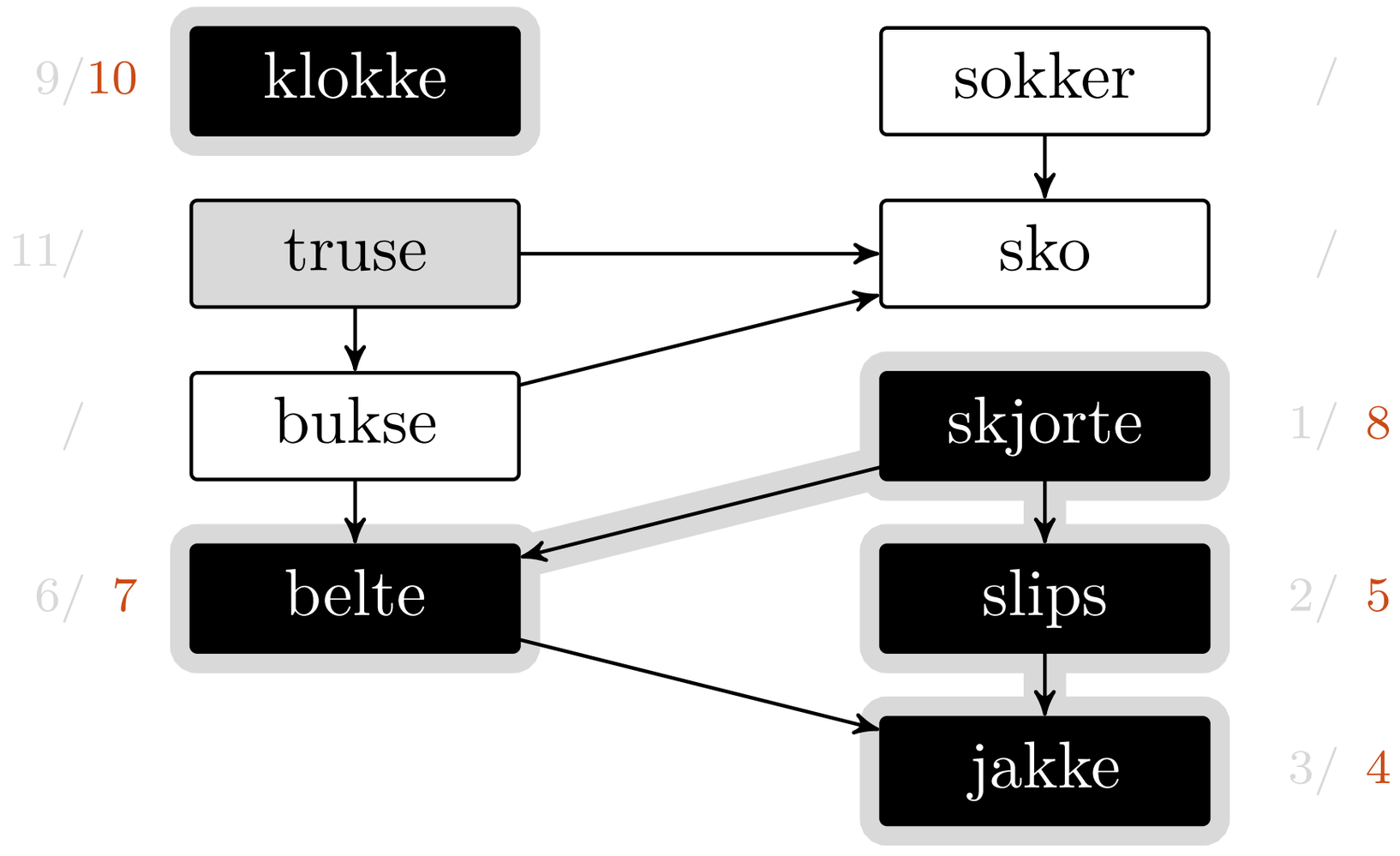
1/ 8
6/ 7
2/ 5
3/ 4

mlh@ntnu.no

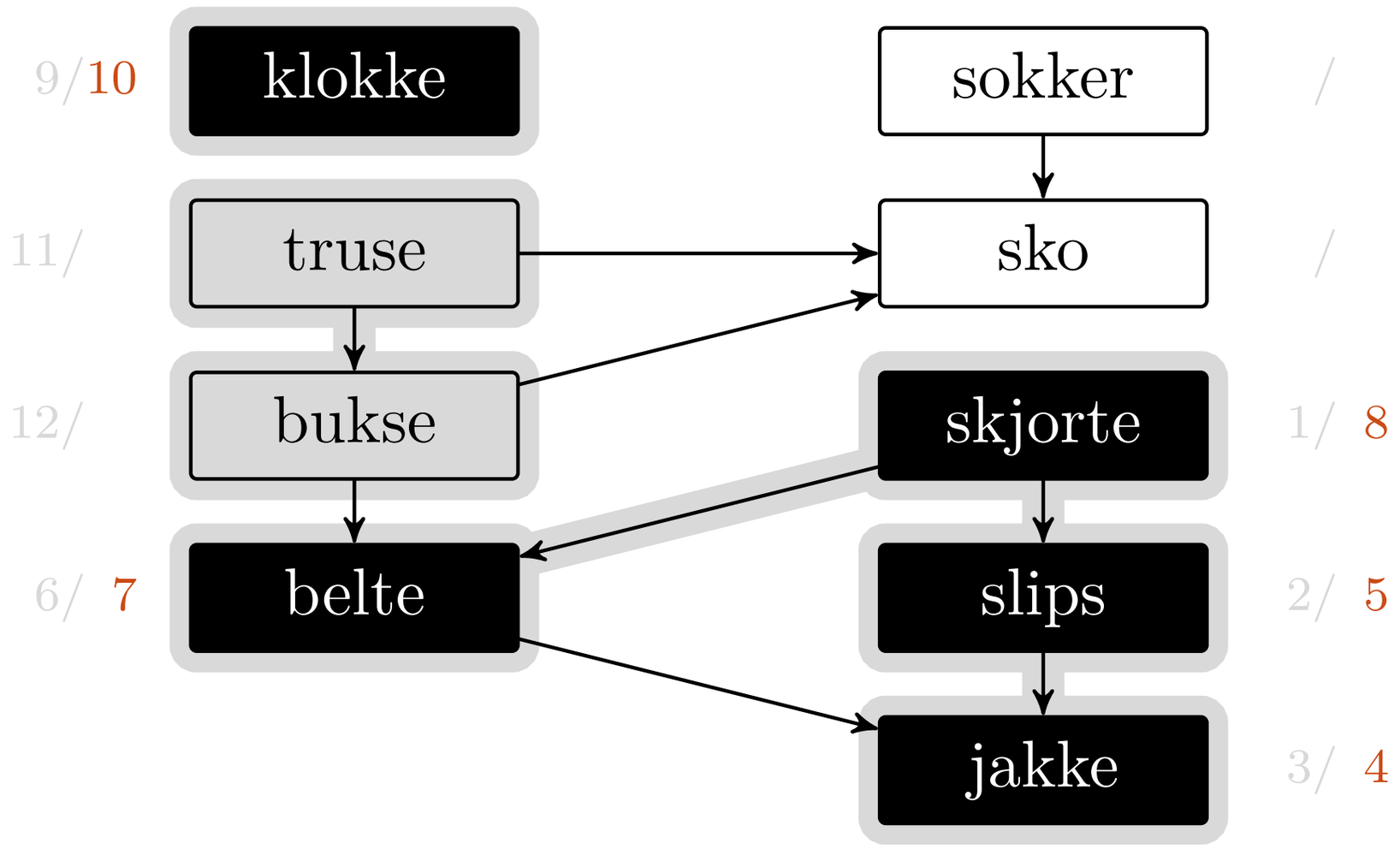



9/10
klokke
sokker
/
/
truse
sko
/
/
bukse
skjorte
1/ 8
6/ 7
belte
slips
2/ 5
jakke
3/ 4

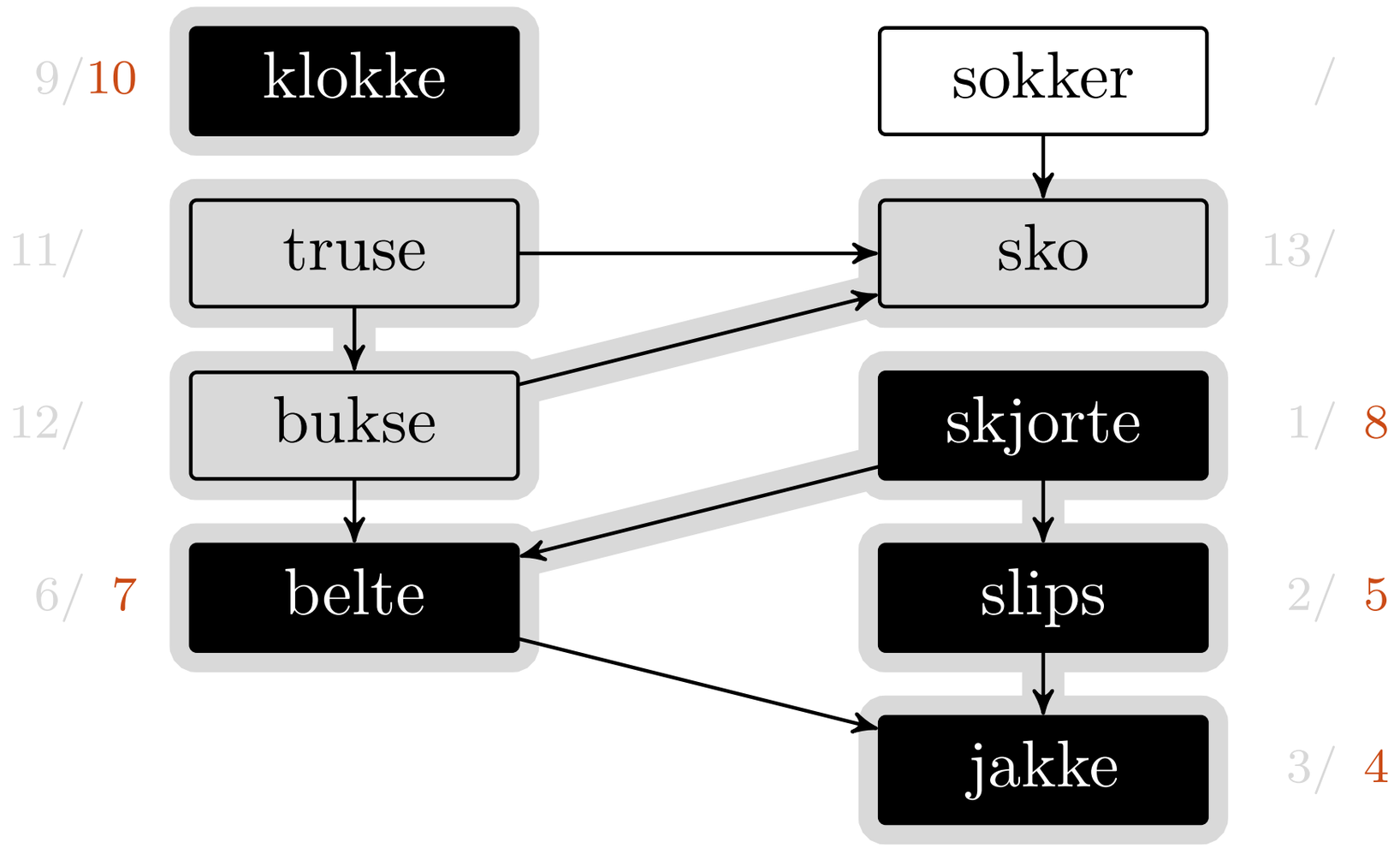
mlh@ntnu.no

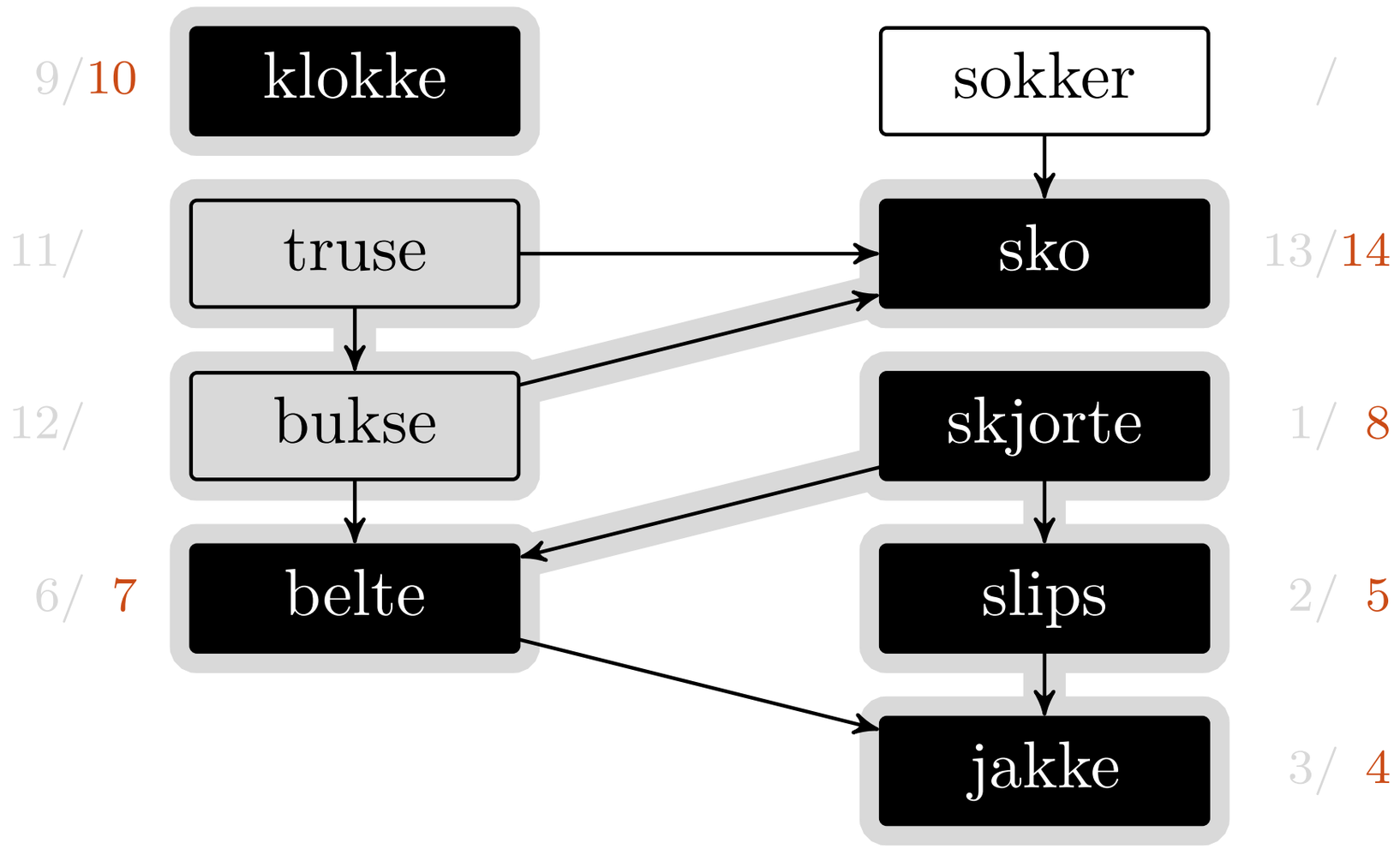

9/10
klokke
sokker
/
11/
truse
sko
/
/
bukse
1/ 8
skjorte
6/ 7
belte
slips
2/ 5
jakke
3/ 4

mlh@ntnu.no

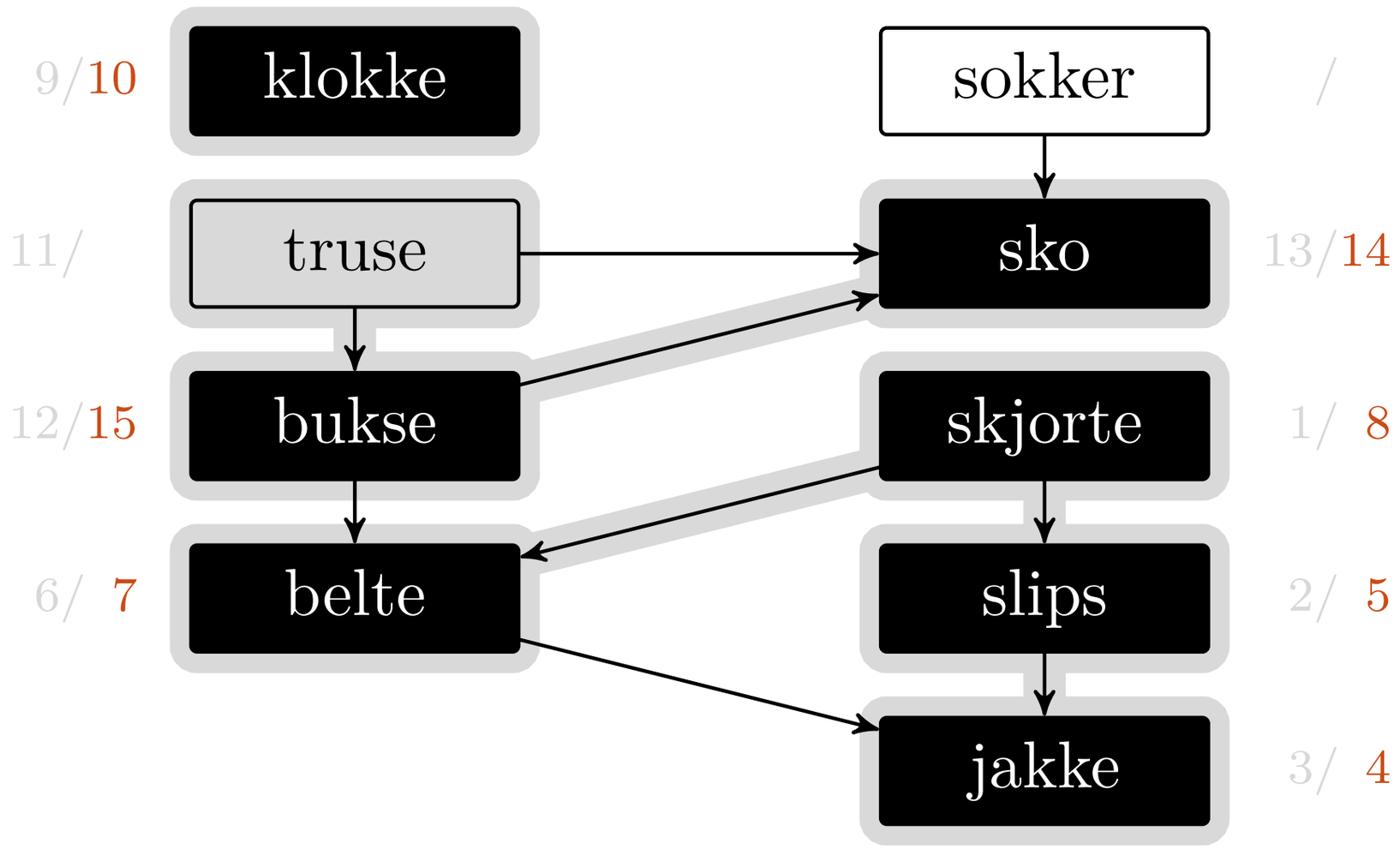

9/10
klokke
sokker
/
11/
truse
sko
/
12/
bukse
skjorte
1/ 8
6/ 7
belte
slips
2/ 5
jakke
3/ 4

mlh@ntnu.no

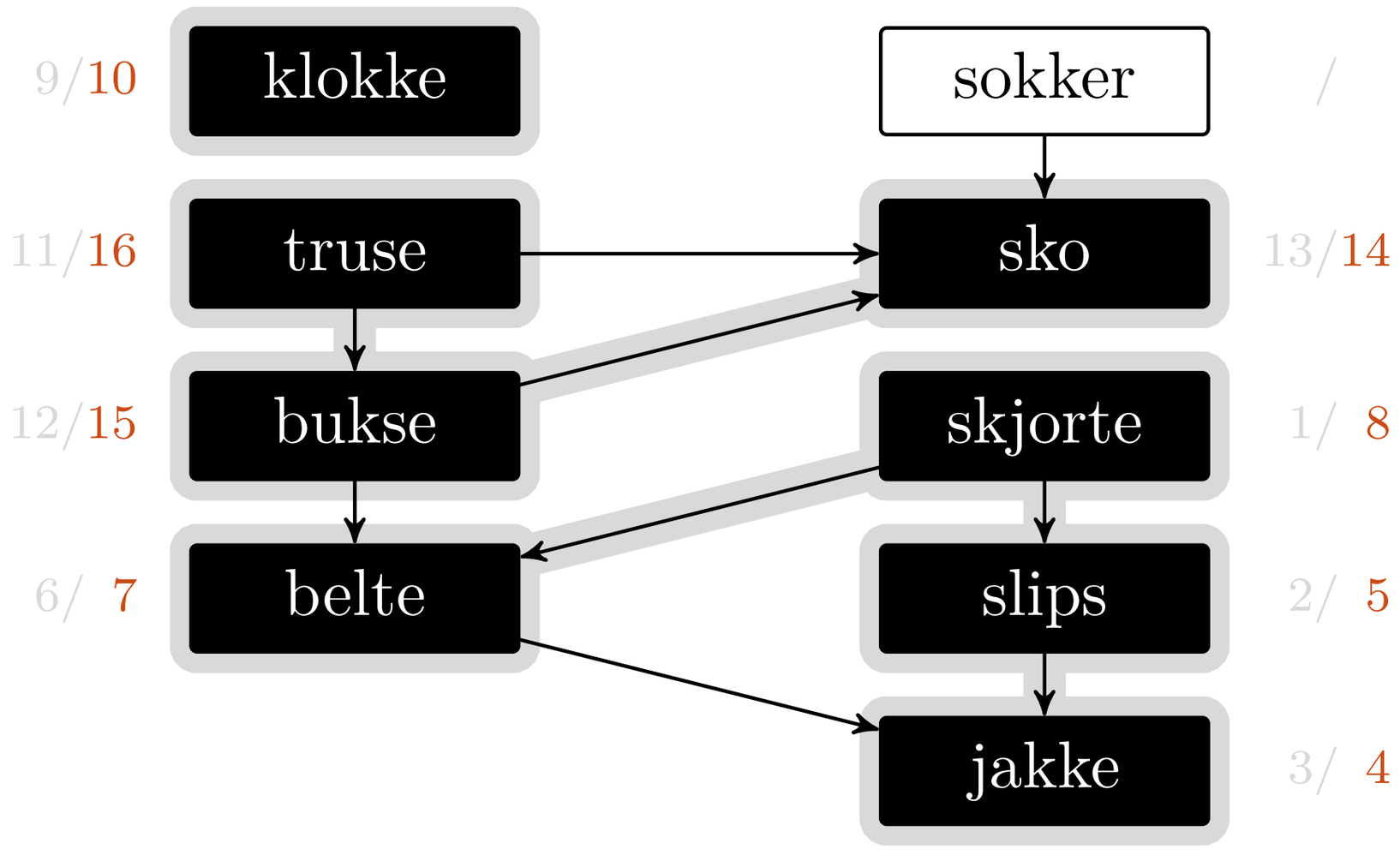

9/10
klokke
sokker
/
11/
truse
sko
13/
12/
bukse
skjorte
1/ 8
6/ 7
belte
slips
2/ 5
jakke
3/ 4


9/10
klokke
sokker
/
11/
truse
sko
13/14
12/
bukse
skjorte
1/ 8
6/ 7
belte
slips
2/ 5
jakke
3/ 4

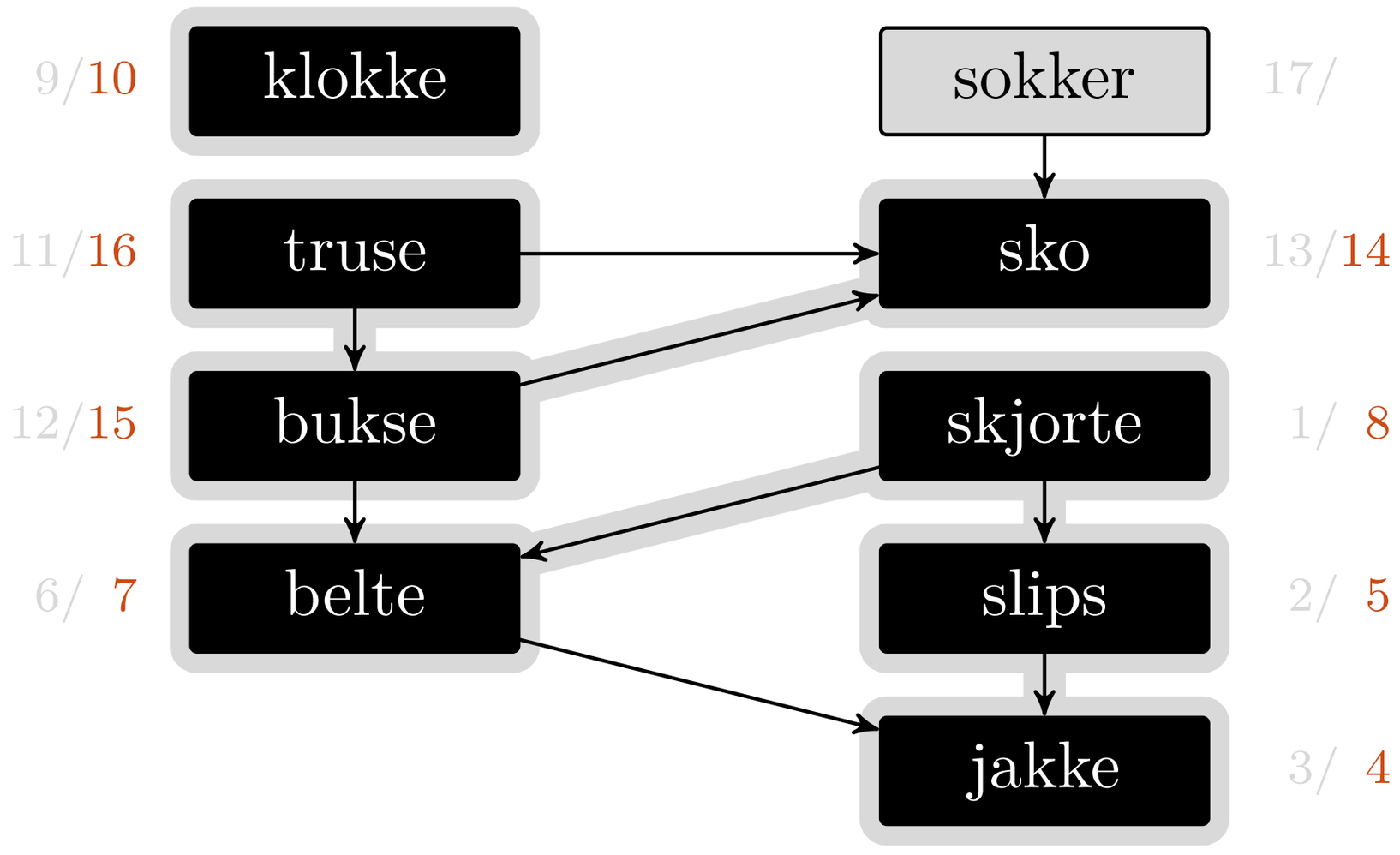
mlh@ntnu.no


9/10
klokke
sokker
/
11/
truse
sko
13/14
12/15
bukse
skjorte
1/ 8
6/ 7
belte
slips
2/ 5
jakke
3/ 4

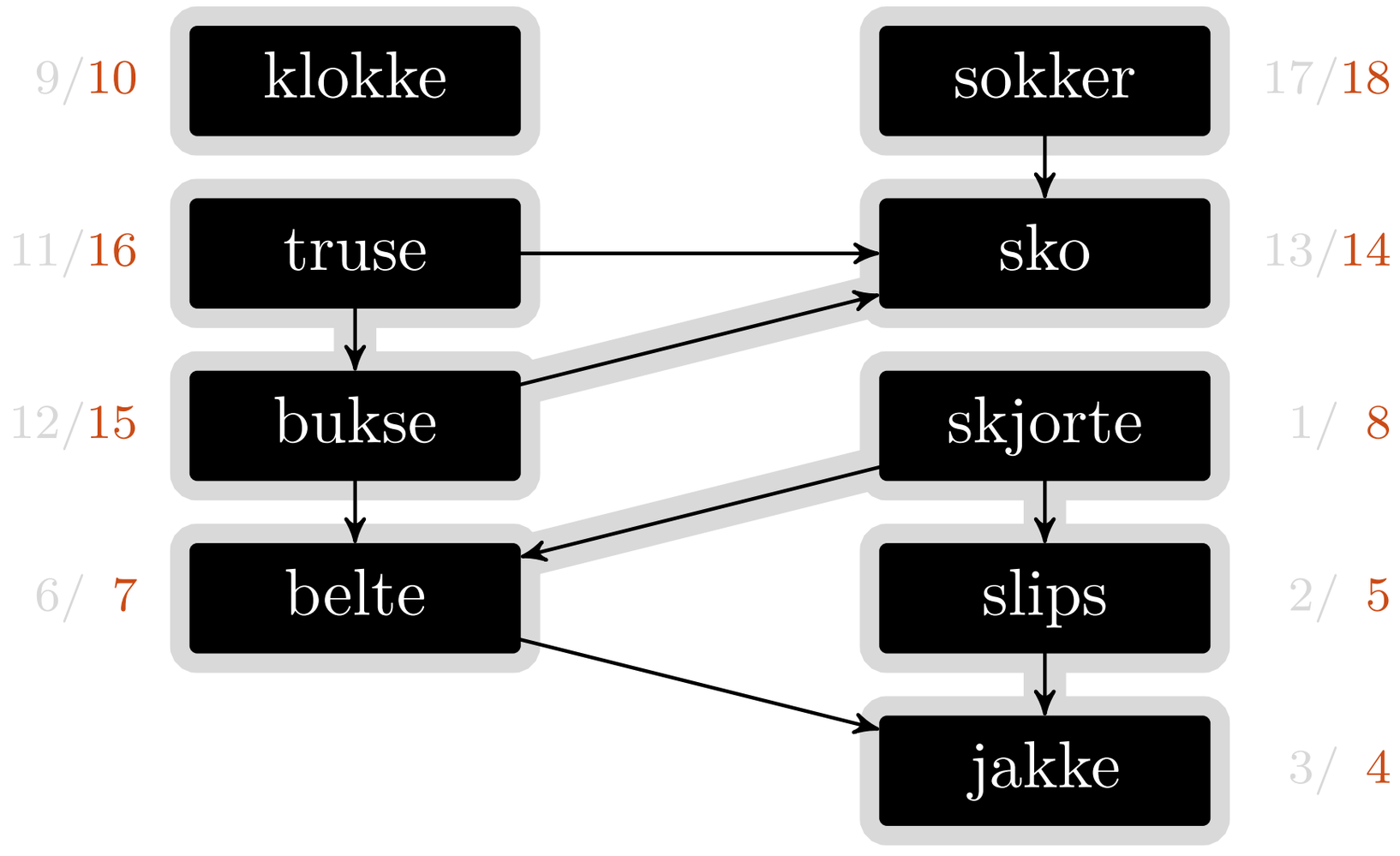
mlh@ntnu.no


9/10
klokke
sokker
/
11/16
truse
sko
13/14
12/15
bukse
skjorte
1/ 8
6/ 7
belte
slips
2/ 5
jakke
3/ 4

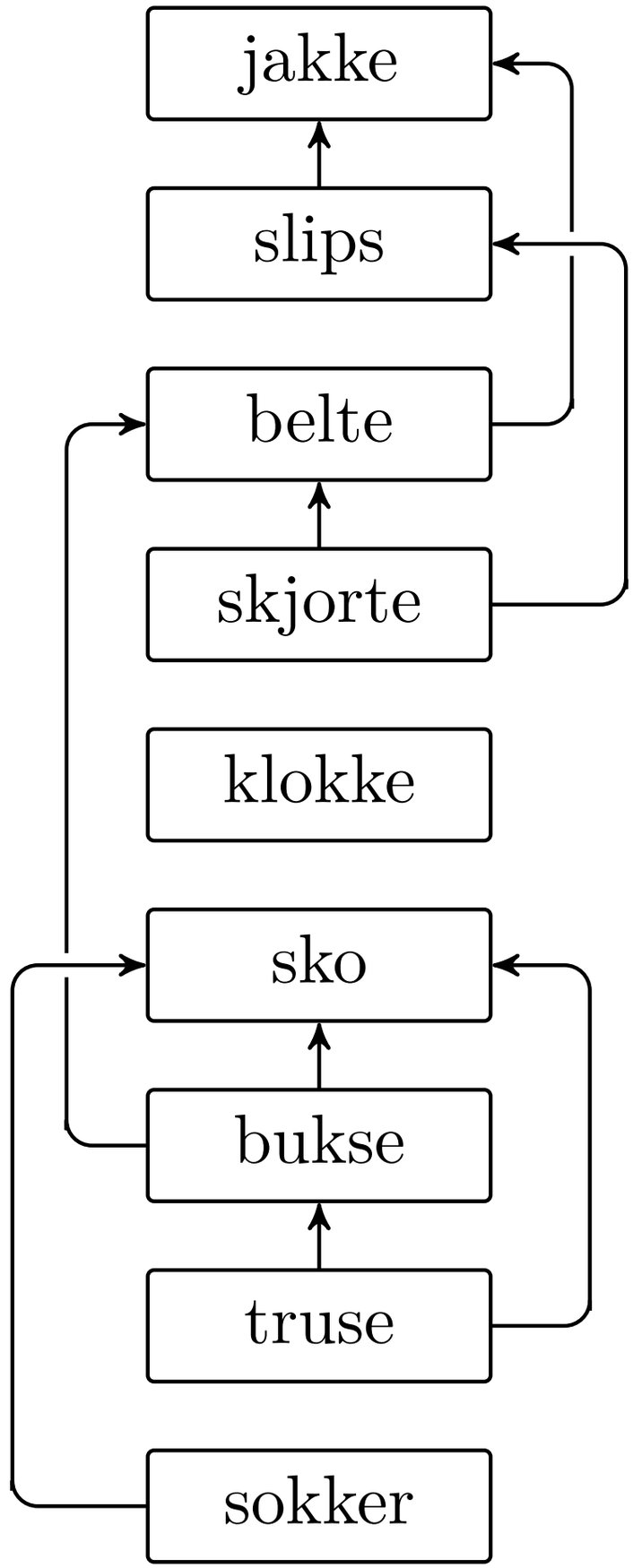
mlh@ntnu.no


9/10
klokke
sokker
17/
11/16
truse
sko
13/14
12/15
bukse
skjorte
1/ 8
6/ 7
belte
slips
2/ 5
jakke
3/ 4

mlh@ntnu.no


9/10
klokke
17/18
sokker
11/16
truse
13/14
sko
12/15
bukse
1/ 8
skjorte
6/ 7
belte
2/ 5
slips
3/ 4
jakke

3/ 4 2/ 5 6/ 7 1/ 8 9/10 13/14 12/15 11/16 17/18



- **I DP med memoisering: Vi utfører implisitt DFS på delproblemene**
- **Vi får automatisk en topologisk sortering: Problemer løses etter delproblemer**
- **Det samme som å sortere etter synkende finish-tid**

Tenk på selv: Hva er sammenhengen mellom pakkesystemer (som automatisk installerer programpakker og avhengigheter) og topologisk sortering ved dybde-først-søk?

Kartet er fra Prims artikkel.

Forelesning 9

Minimale spenntrær

Fig. 1 — Example of a shortest connection network.

1. **Disjunkte mengder**

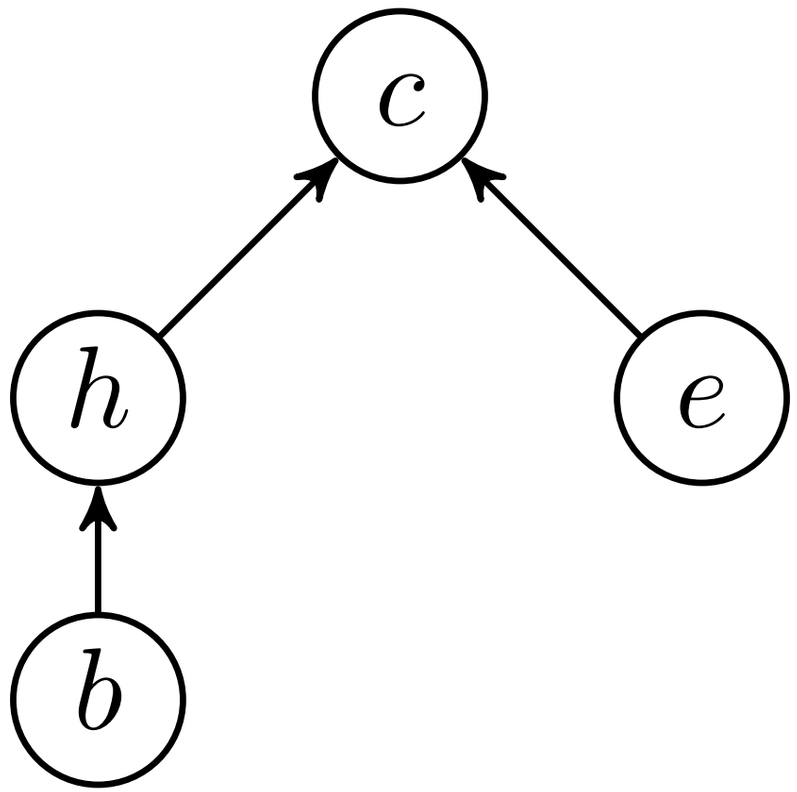
2. **Generisk MST**

3. **Kruskals algoritme**

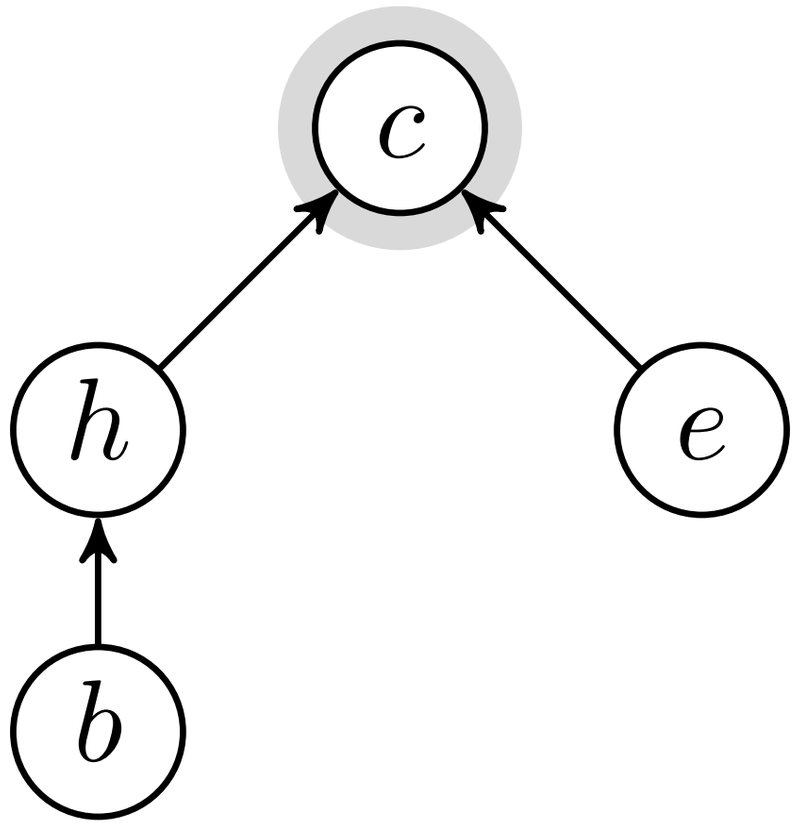
4. **Prims algoritme**

1.4

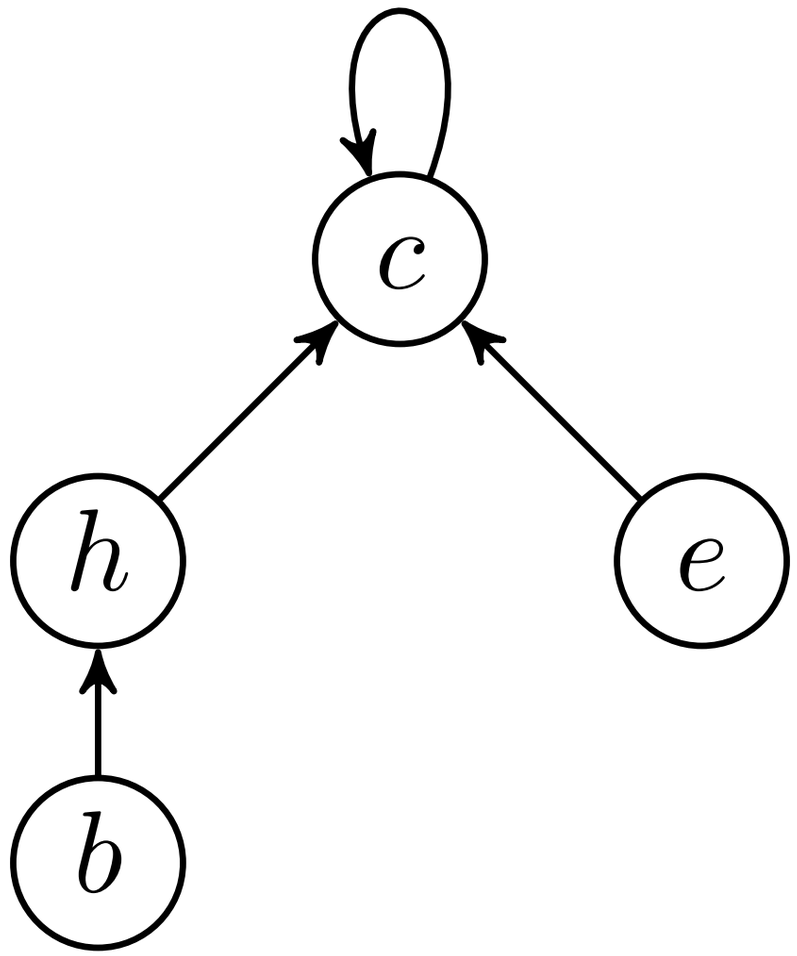
Disjunkte mengder



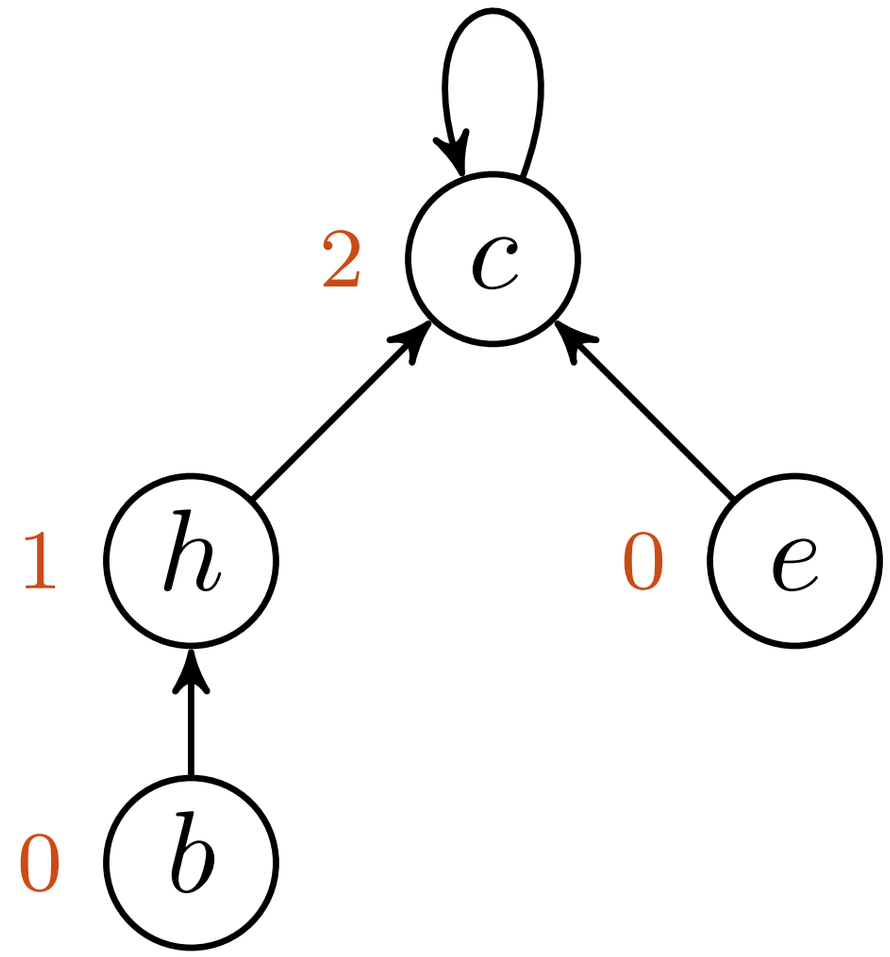
Mengder representeres som trær vha. foreldrepekere $v.p$



Rota representerer mengden; FIND-SET(v) gir peker til rota



Self-loop: Fjerner spesialtilfelle fra Find-Set



For *union by rank*-heuristikk: Rang er øvre grense for nodehøyde

Make-Set(x)

Initialisering: Noden representerer mengden $\{x\}$

MAKE-SET(x)

1 $x.p = x$

Mengden representeres av et tre; dette er foreldrenoden til x

MAKE-SET(x)

1 $x.p = x$
2 $x.rank = 0$

En slags «høyde»; største avstand til en løvnode

$\textsc{Union}(x, y)$

Kombinér to mengder/trær

```
UNION(x, y)
1  LINK(FIND-SET(x), FIND-SET(y))
```

Finn røttene/«representantene» og koble dem sammen

$\textsc{Link}(x, y)$

To røtter; én henges under den andre

```
LINK(x, y)
1  if x.rank > y.rank
```

Utjevning: Den med høyest rang blir rot

```
LINK(x, y)
1  if x.rank > y.rank
2        y.p = x
```

Den nærmest «bunnen» blir barn av den andre

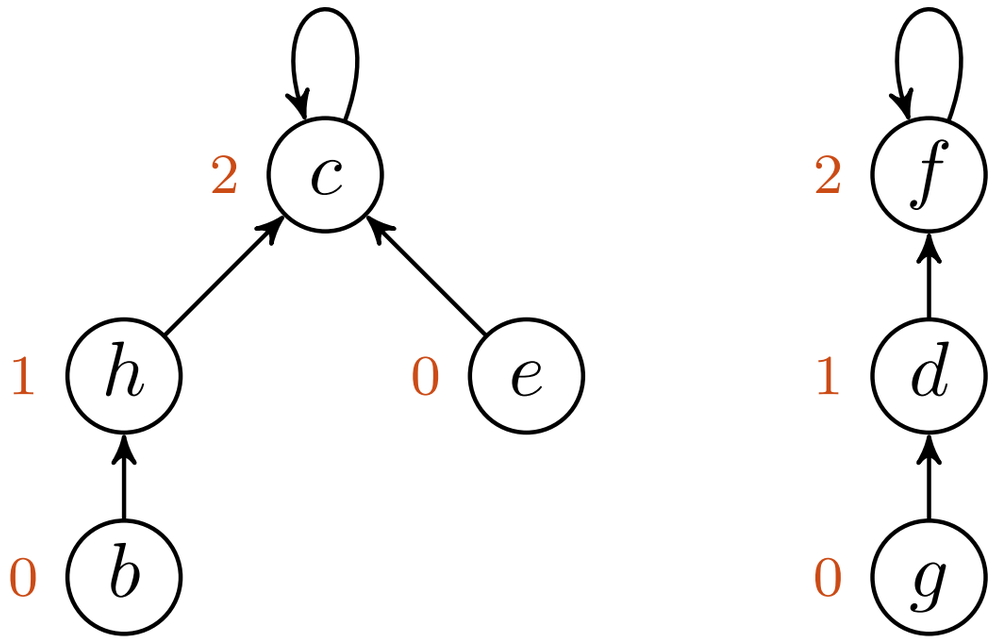
```
LINK(x, y)
1  if x.rank > y.rank
2        y.p = x
3  else x.p = y
```

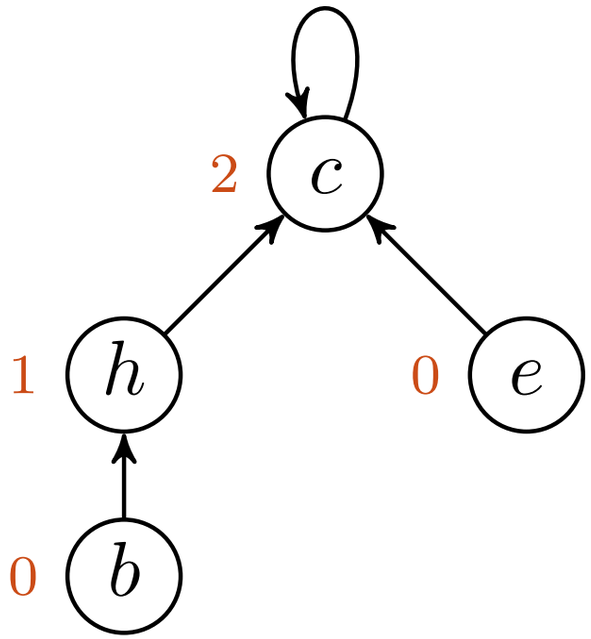
```
LINK(x, y)
1  if x.rank > y.rank
2        y.p = x
3  else x.p = y
4        if x.rank == y.rank
```

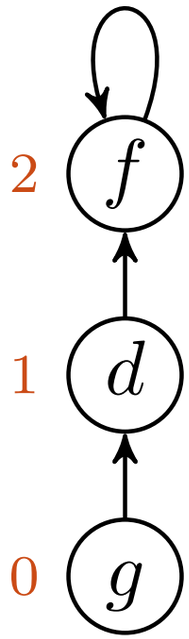
Barn strengt mindre rang? Alt i orden. Ellers . . .

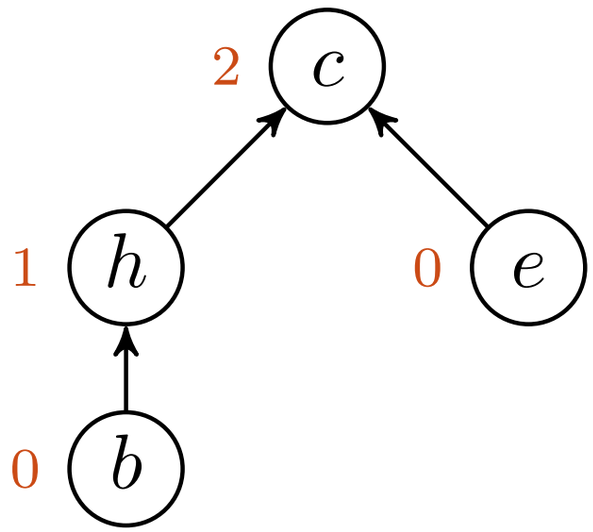
```
LINK(x, y)
1  if x.rank > y.rank
2        y.p = x
3  else x.p = y
4       if x.rank == y.rank
5            y.rank = y.rank + 1
```

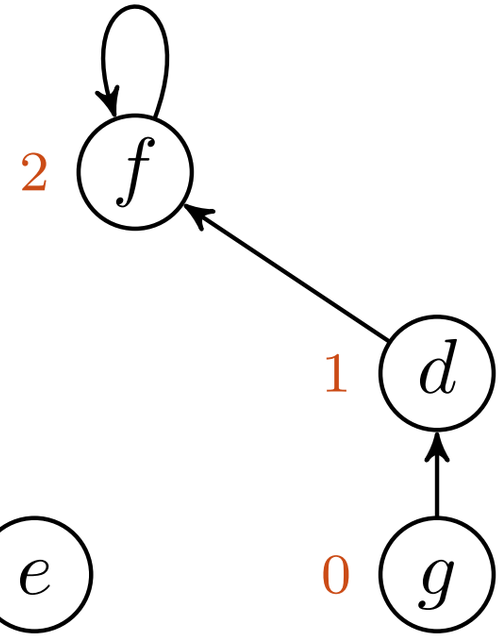
Vedlikehold betydningen av rang: Avstand til bunnen

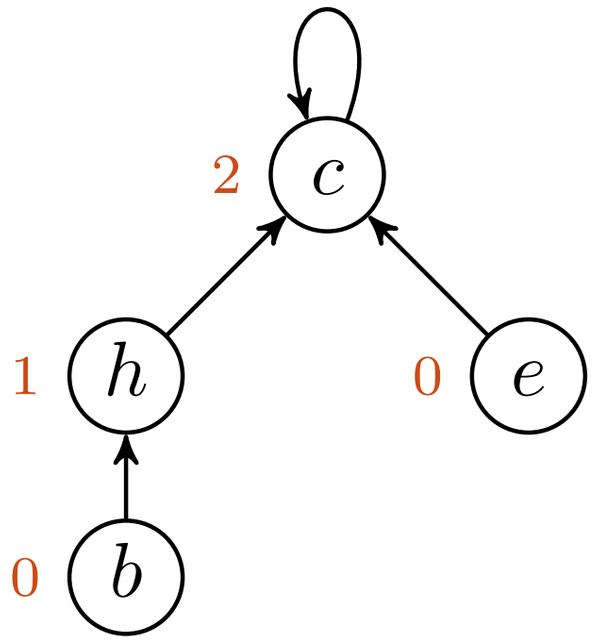


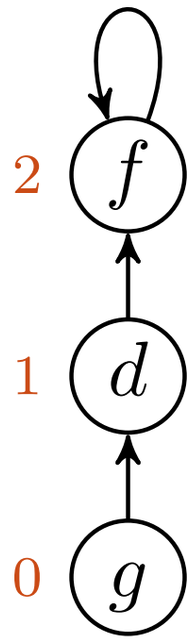


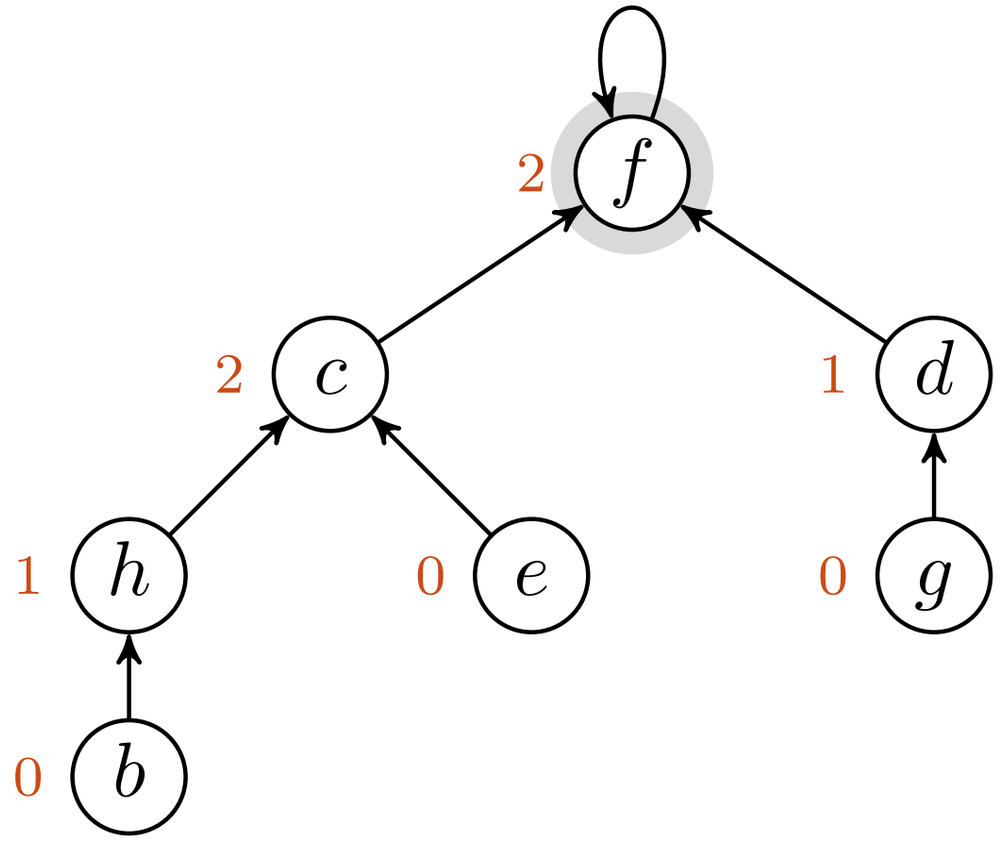


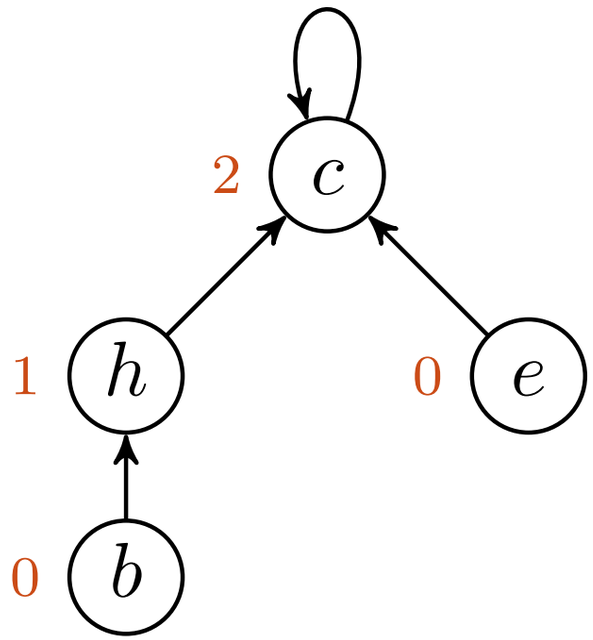


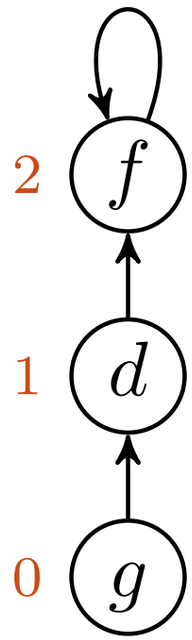


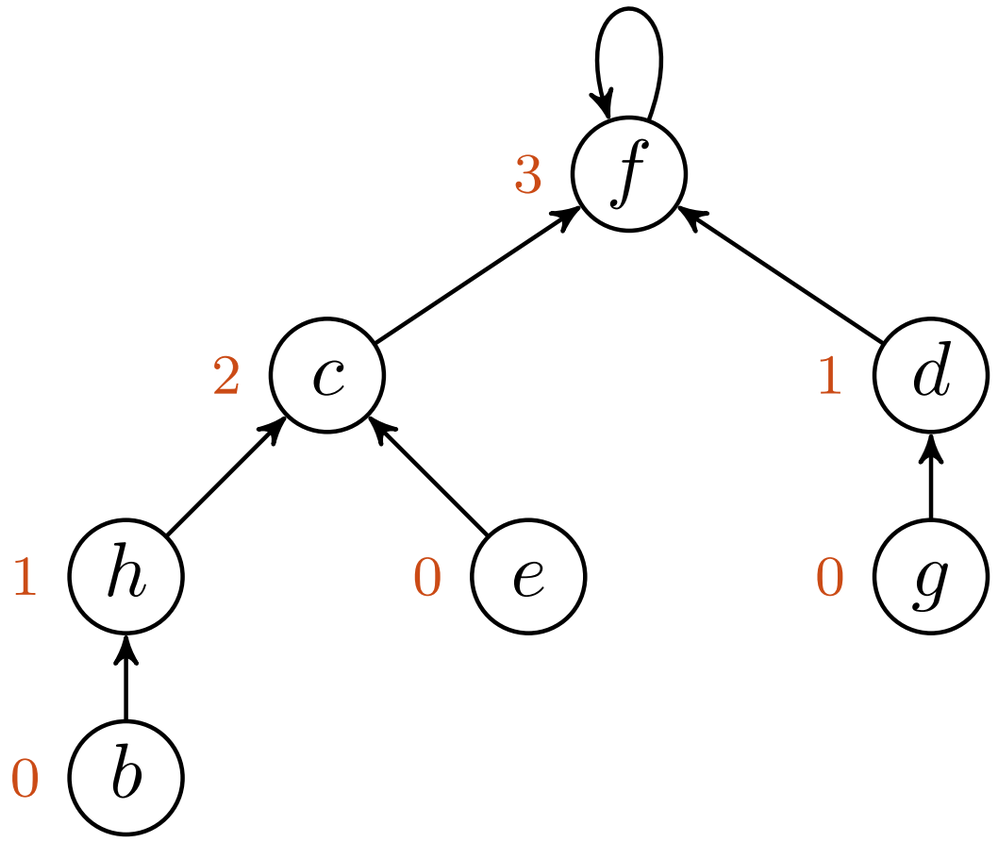












Find-Set(x)

Finn «representanten»/rota til mengden/treet som inneholder x

```
FIND-SET(x)
1  if x ≠ x.p
```

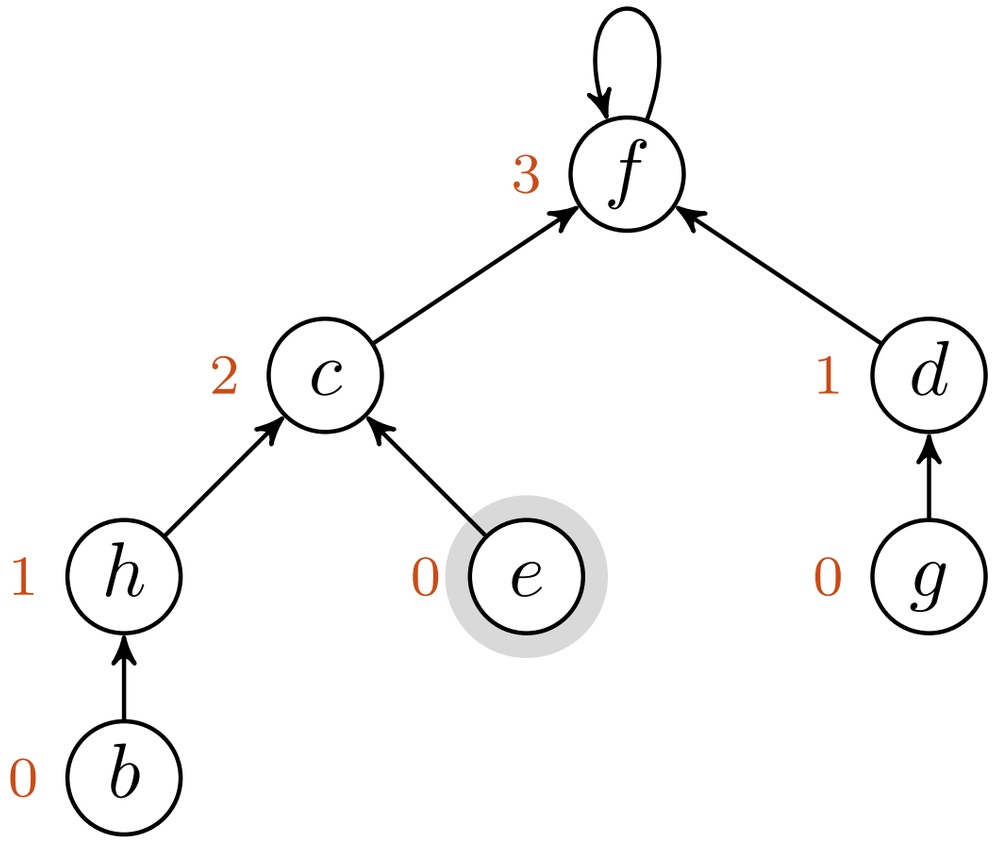
Rota har seg selv som representant. Ikke der ennå . . .

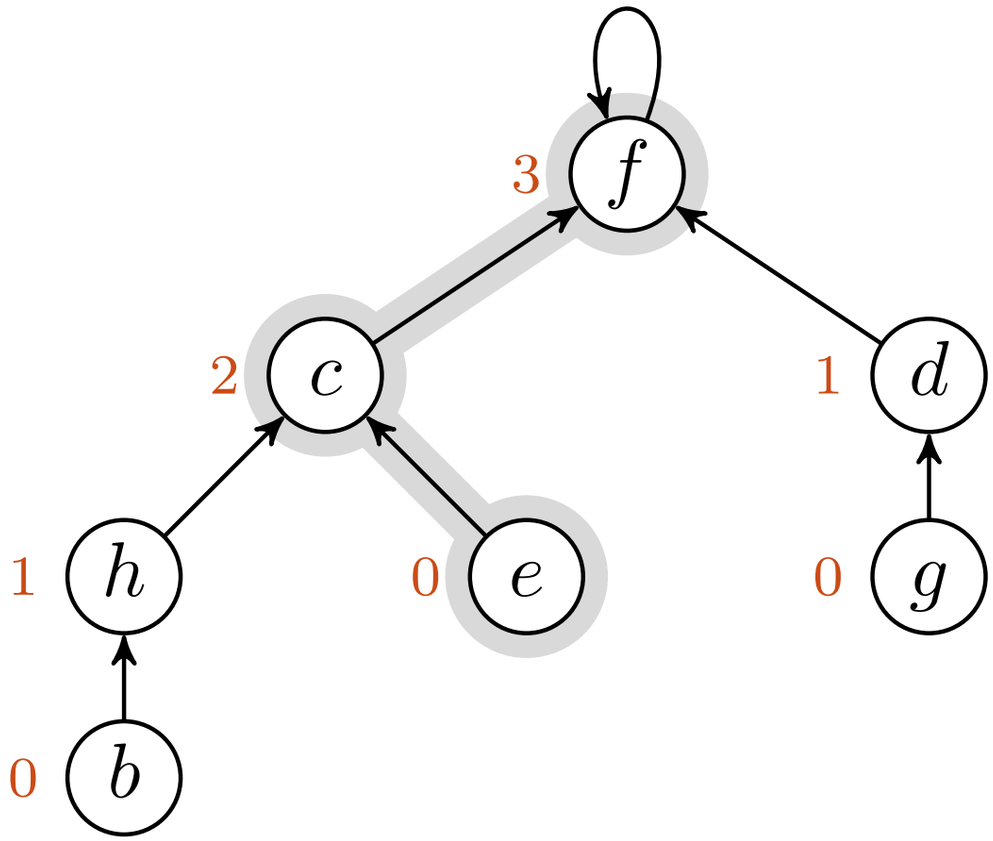
```
FIND-SET(x)
1  if x ≠ x.p
2      x.p = FIND-SET(x.p)
```

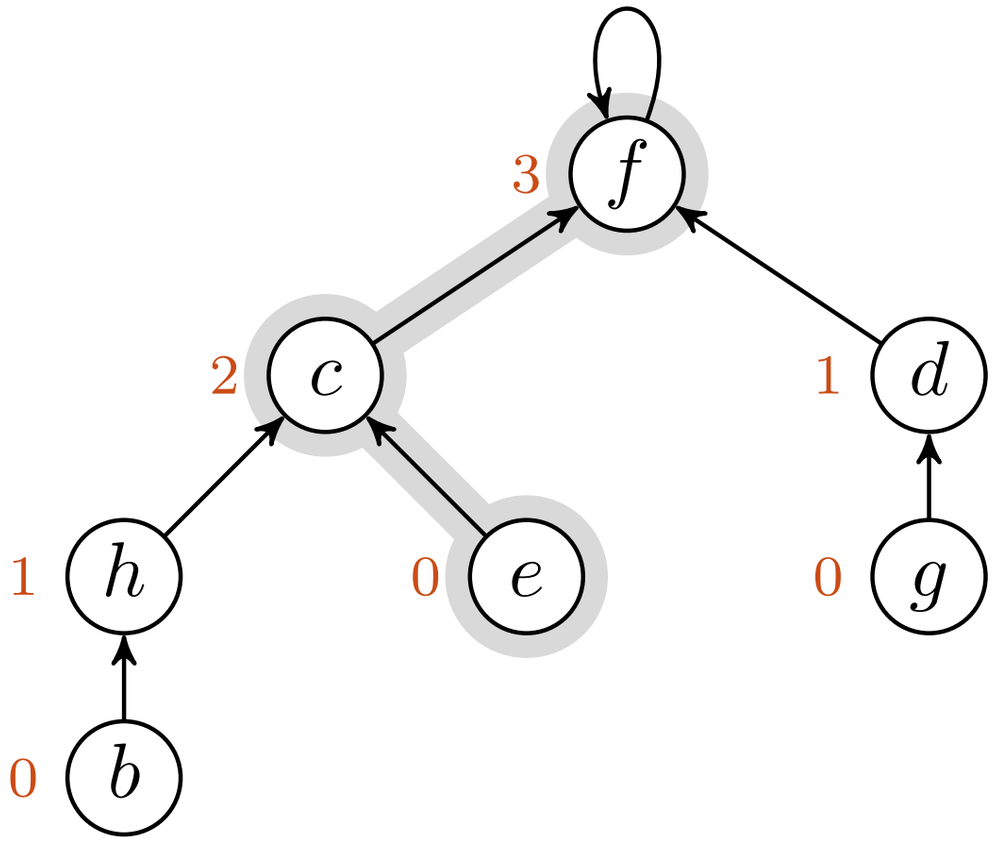
. . . så vi leter videre (rekursivt)

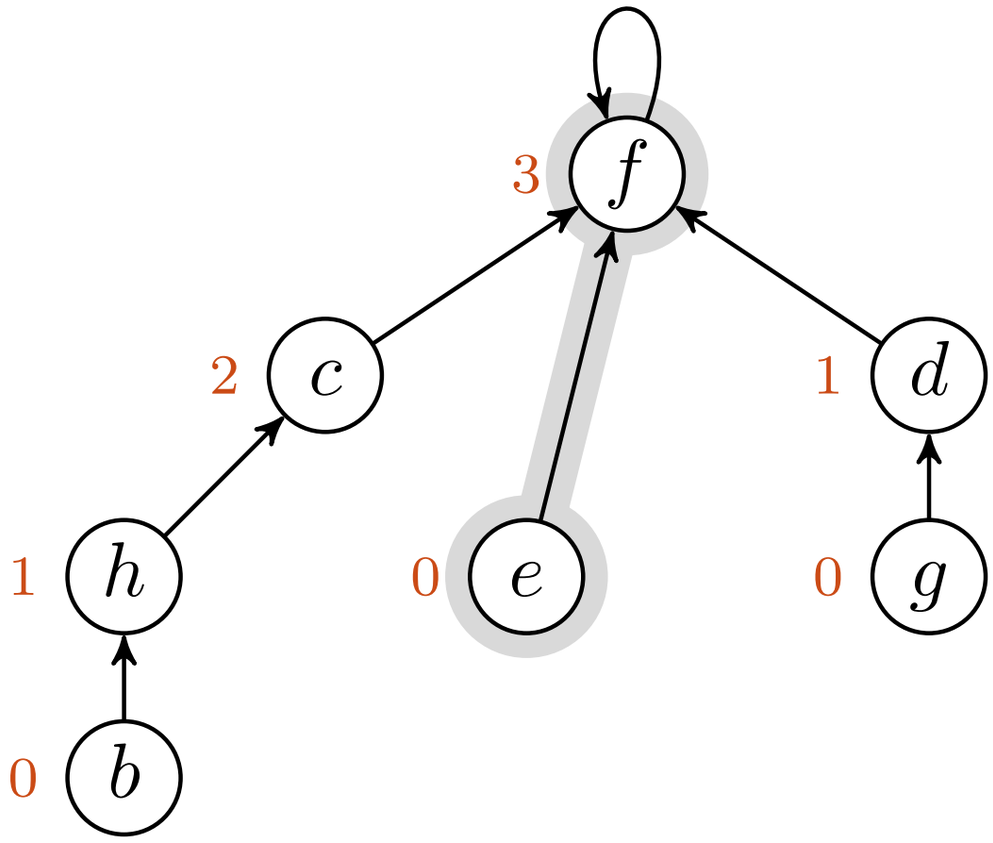
```
FIND-SET(x)
1  if x ≠ x.p
2      x.p = FIND-SET(x.p)
3  return x.p
```

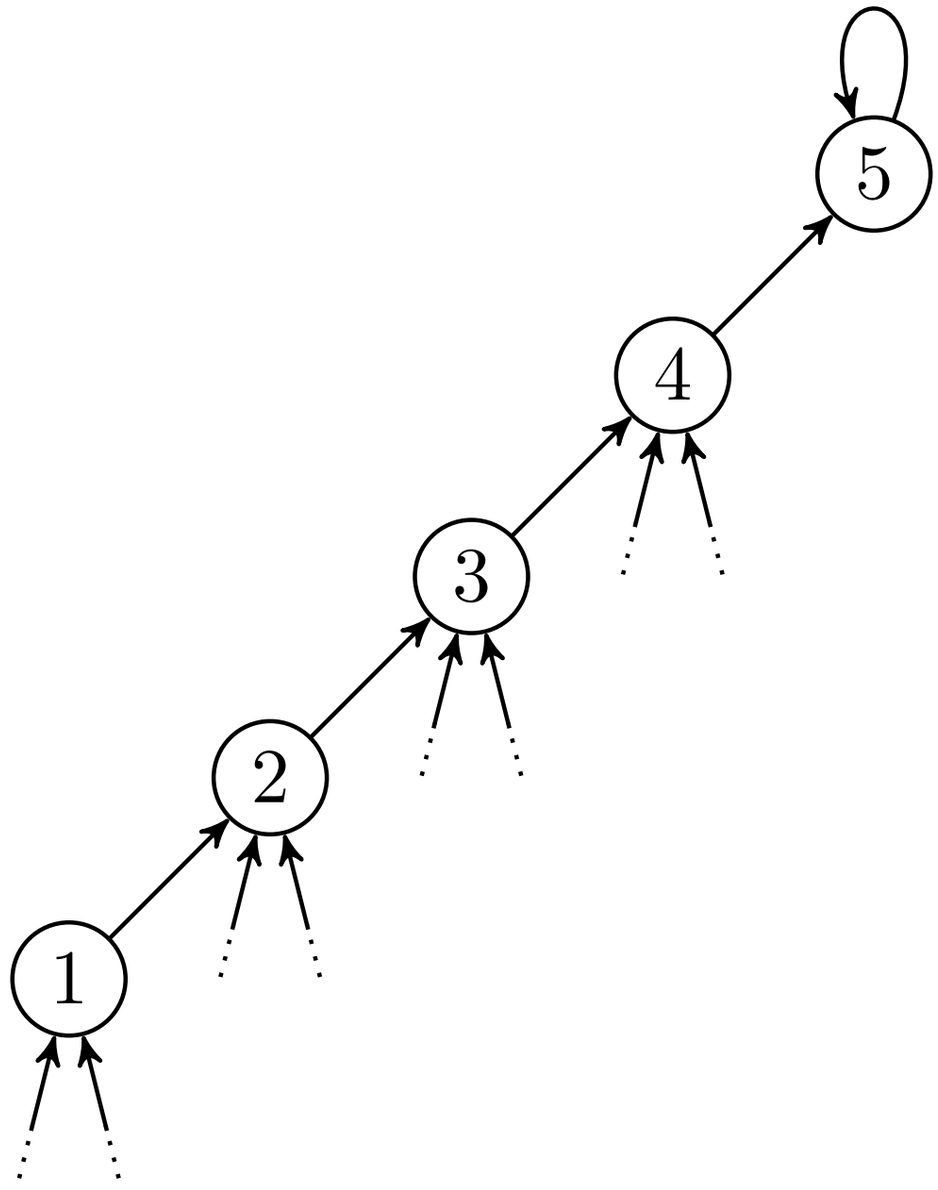
Nå har x fått rota som forelder; snarvei til neste gang!



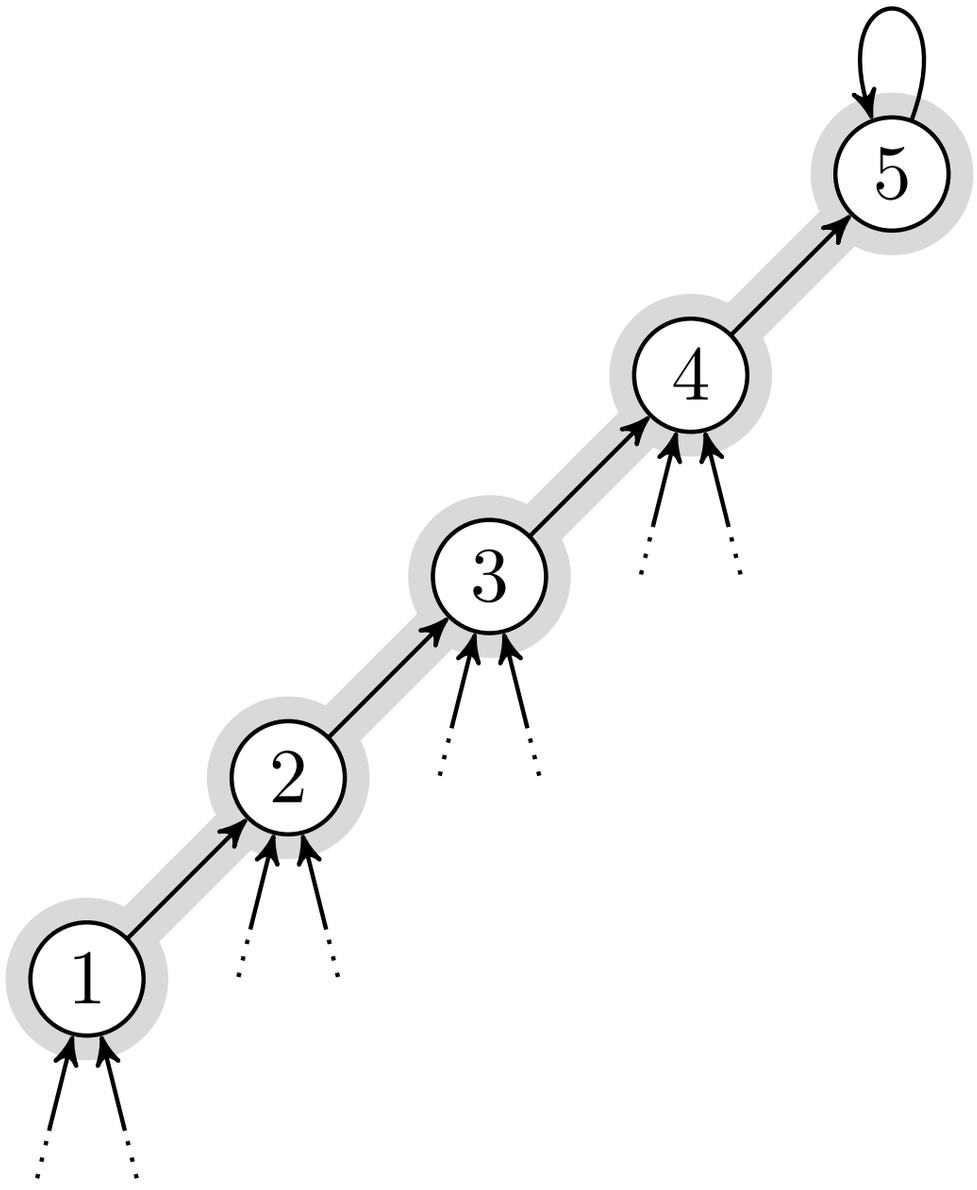




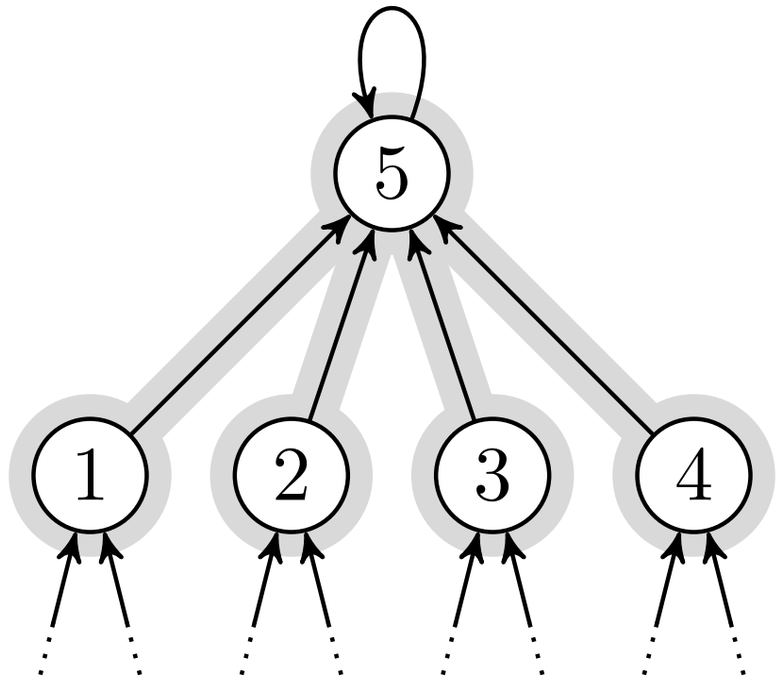




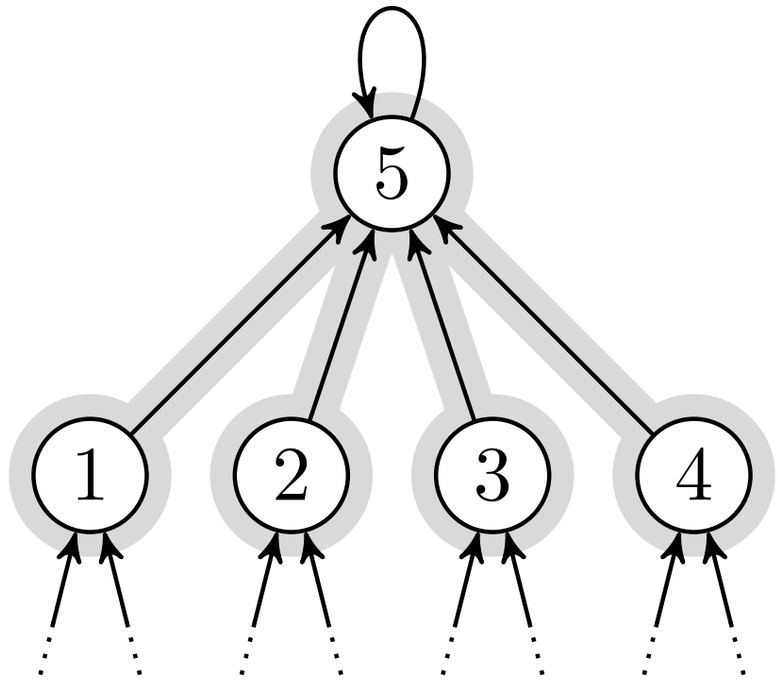
Før FIND-SET(1) komprimerer stien $1 \to \cdots \to 5$



FIND-SET er rekursiv og «destruktiv»; alle på stien får $p = 5$



FIND-SET er rekursiv og «destruktiv»; alle på stien får $p = 5$



Oppdaterer ikke rang; det er derfor det bare er et overestimat

m operasjoner: $\mathrm{O}(m \cdot \alpha(n))$

m operasjoner: $\mathrm{O}(m \cdot \alpha(n))$

$$\alpha(n) = \begin{cases} 0 & \text{hvis } 0 \leq n \leq 2\,, \\ 1 & \text{hvis } n = 3\,, \\ 2 & \text{hvis } 4 \leq n \leq 7\,, \\ 3 & \text{hvis } 8 \leq n \leq 2047\,, \\ \end{cases}$$

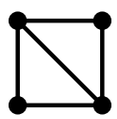
m operasjoner: $\mathrm{O}(m \cdot \alpha(n))$

$$\alpha(n) = \begin{cases} 0 & \text{hvis } 0 \leq n \leq 2\,, \\ 1 & \text{hvis } n = 3\,, \\ 2 & \text{hvis } 4 \leq n \leq 7\,, \\ 3 & \text{hvis } 8 \leq n \leq 2047\,, \\ 4 & \text{hvis } 2048 \leq n \leq 16^{512}\,. \end{cases}$$

2.4

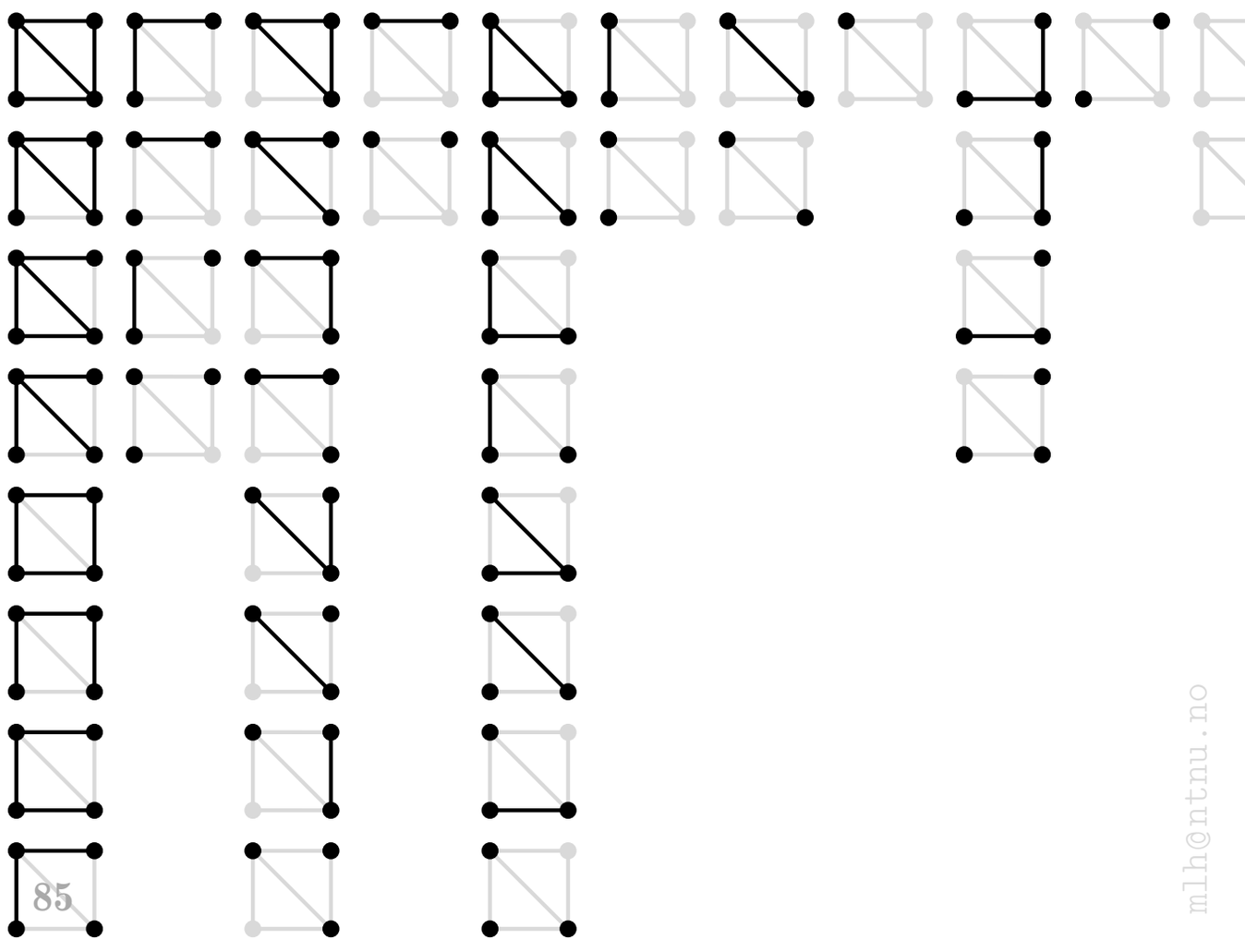
Generisk MST

En graf



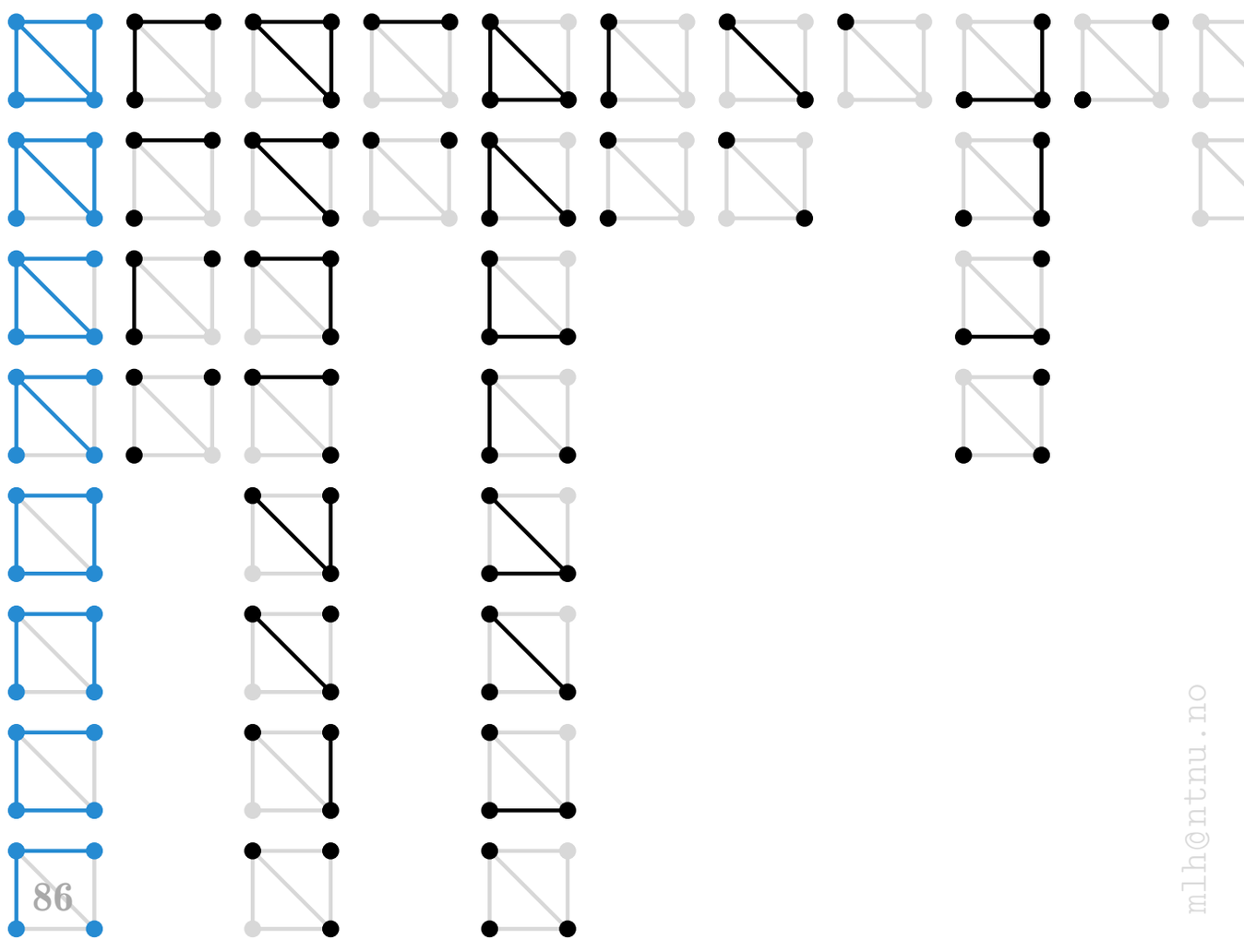
Delgrafer

Delgrafer er grafer som består av delmengder av nodene og kantene til den opprinnelige grafen. (En graf er en delgraf av seg selv – men ikke en såkalt *ekte* delgraf.)



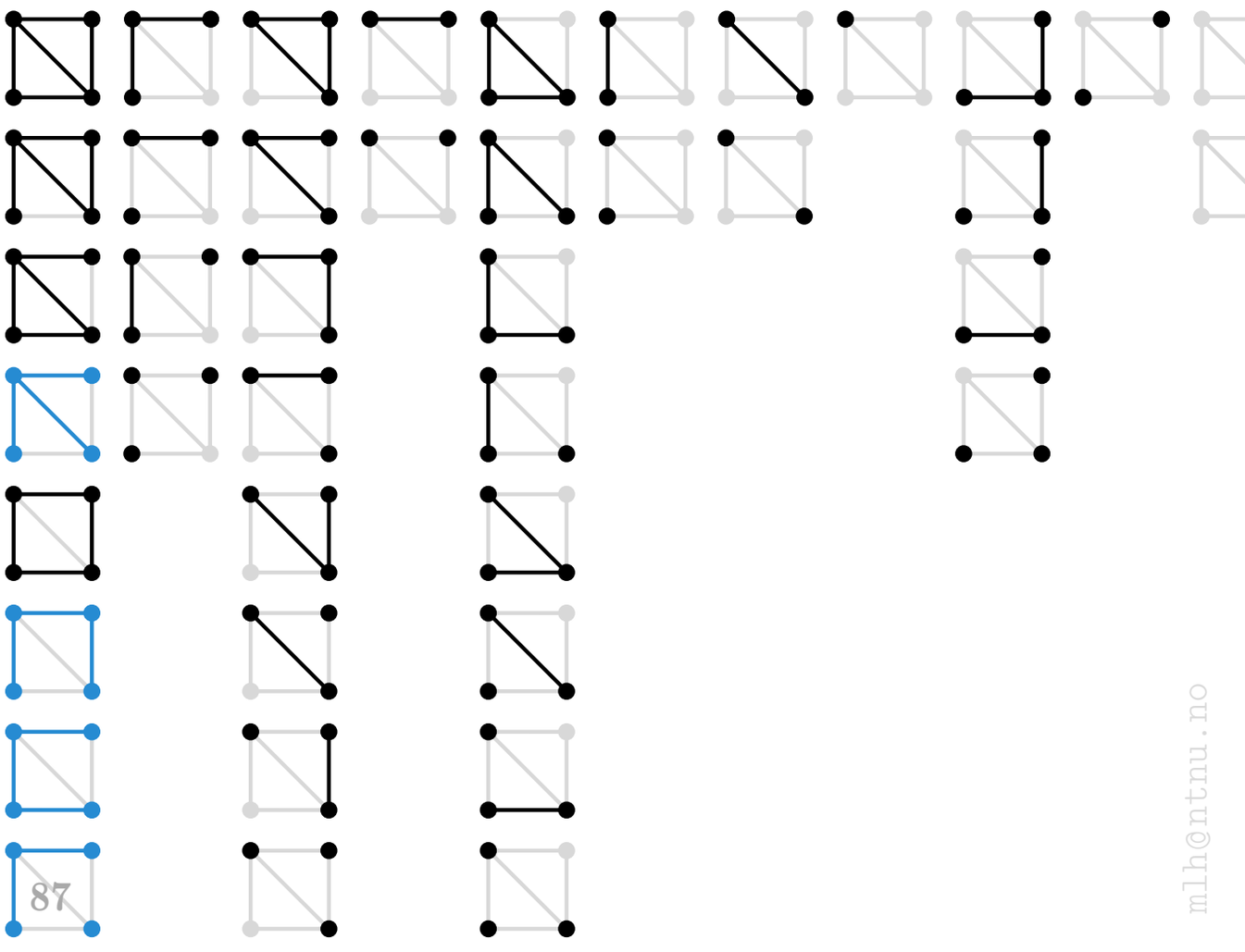
«Spenngrafer»

En «dekkende delgraf» (spanning subgraph) eller «spenngraf» er en delgraf med det samme nodesettet som originalgrafen.



«Spennskoger»

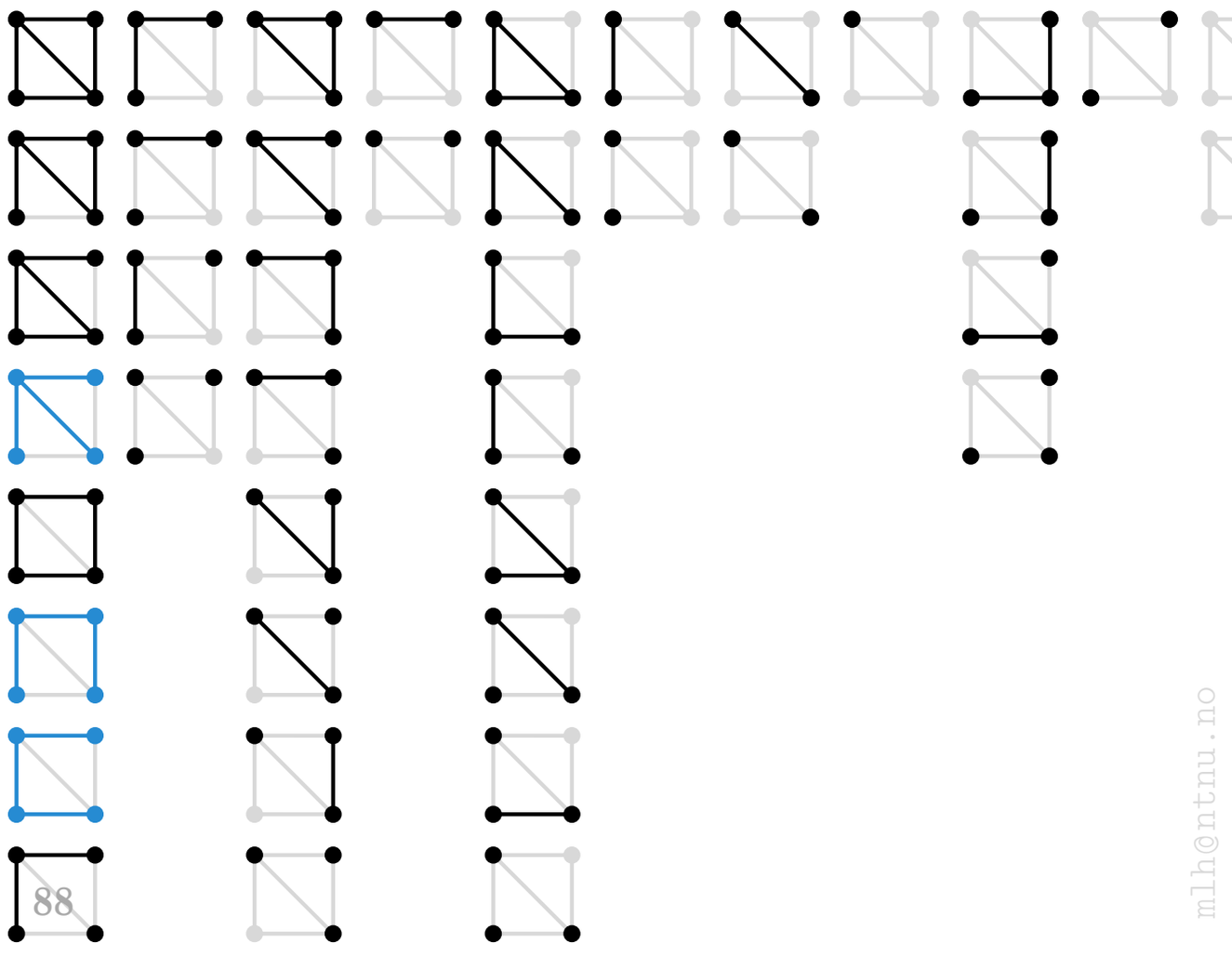
En «dekkende skog» (spanning forest) eler «spenngskog» er en asyklisk spenngraf.

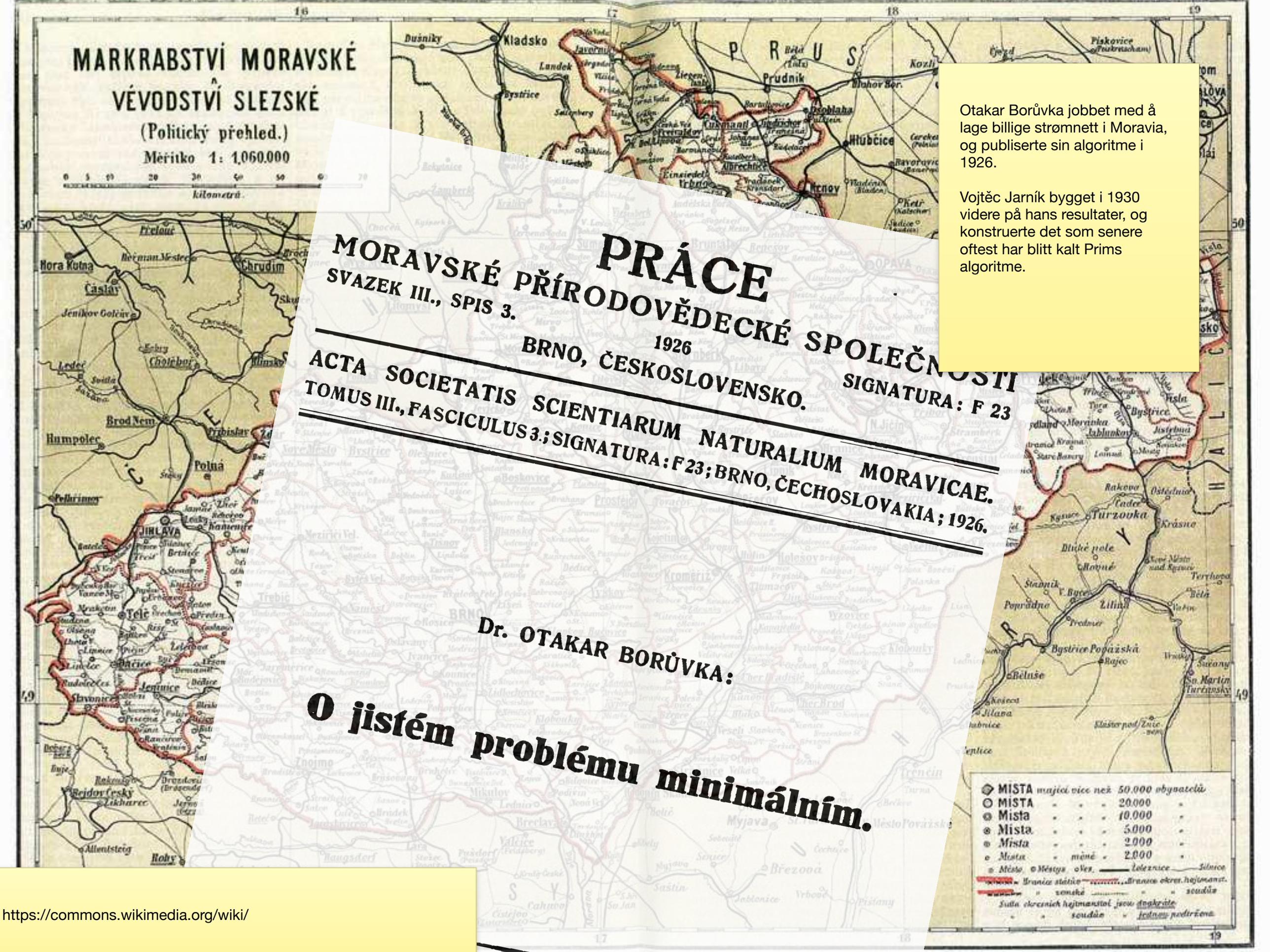


Spenntrær

Et spenntre (spanning tree) er en sammenhengende spennskog.

(Spenntre er et vanlig begrep på norsk. Spenngraf og spennskog er det ikke.)





PRÁCE
MORAVSKÉ PŘÍRODOVĚDECKÉ SPOLEČNOSTI

SVAZEK III., SPIS 3.

1926

BRNO, ČESKOSLOVENSKO.

SIGNATURA: F 23

ACTA SOCIETATIS SCIENTIARUM NATURALIUM MORAVICAE.

TOMUS III., FASCICULUS 3.; SIGNATURA: F 23; BRNO, ČECHOSLOVAKIA; 1926.

Dr. OTAKAR BORŮVKA:

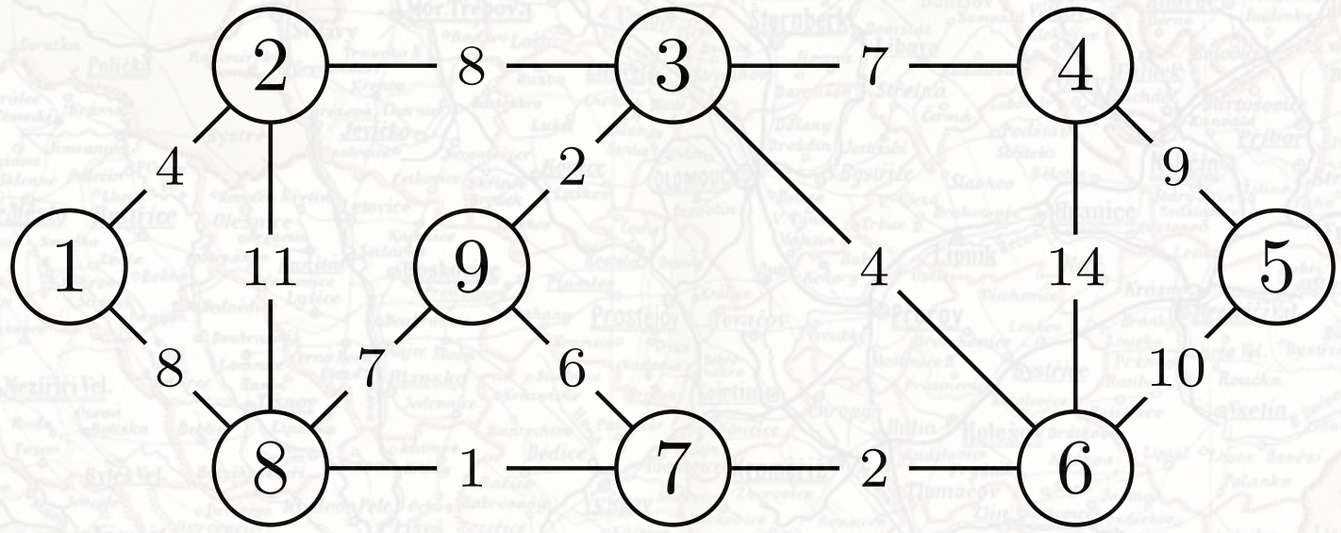
O jistém problému minimálním.

Otakar Borůvka jobbet med å lage billige strømnett i Moravia, og publiserte sin algoritme i 1926.

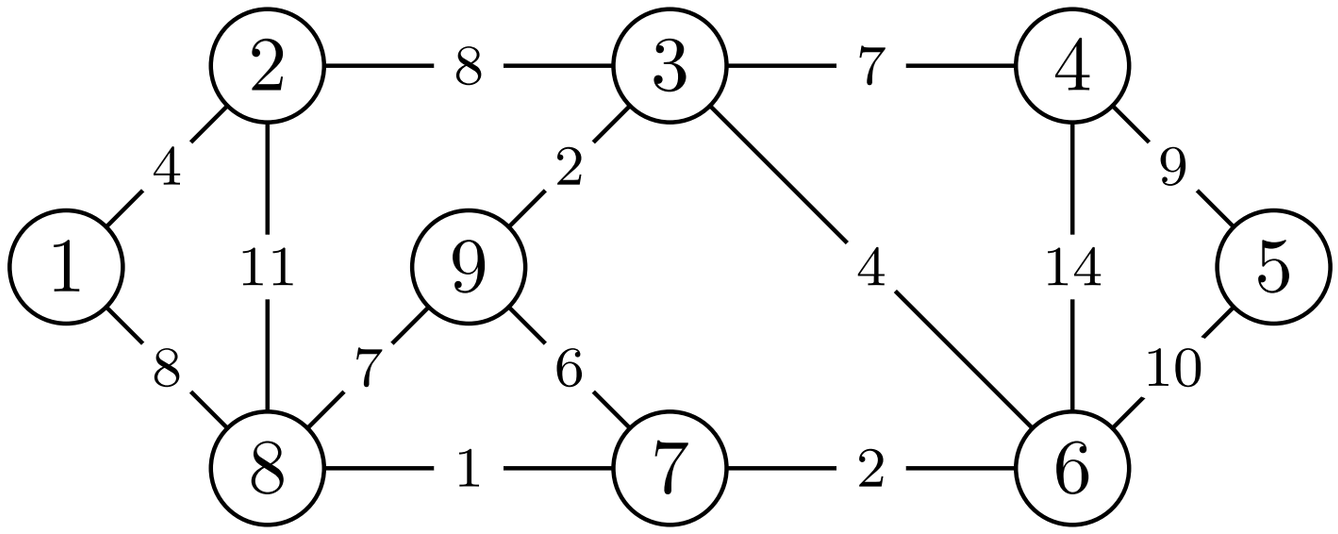
Vojtěc Jarník bygget i 1930 videre på hans resultater, og konstruerte det som senere oftest har blitt kalt Prims algoritme.

https://commons.wikimedia.org/wiki/

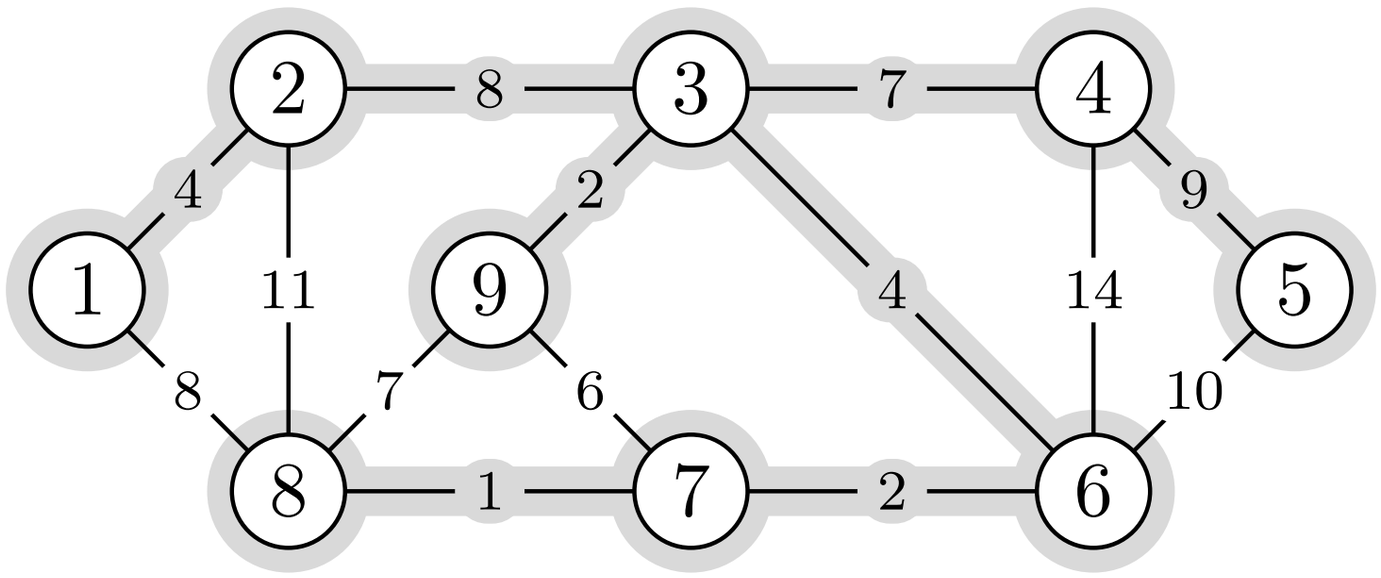
Vi innfører nå vekter på kantene. Disse omtales også som lengder eller kostnader.



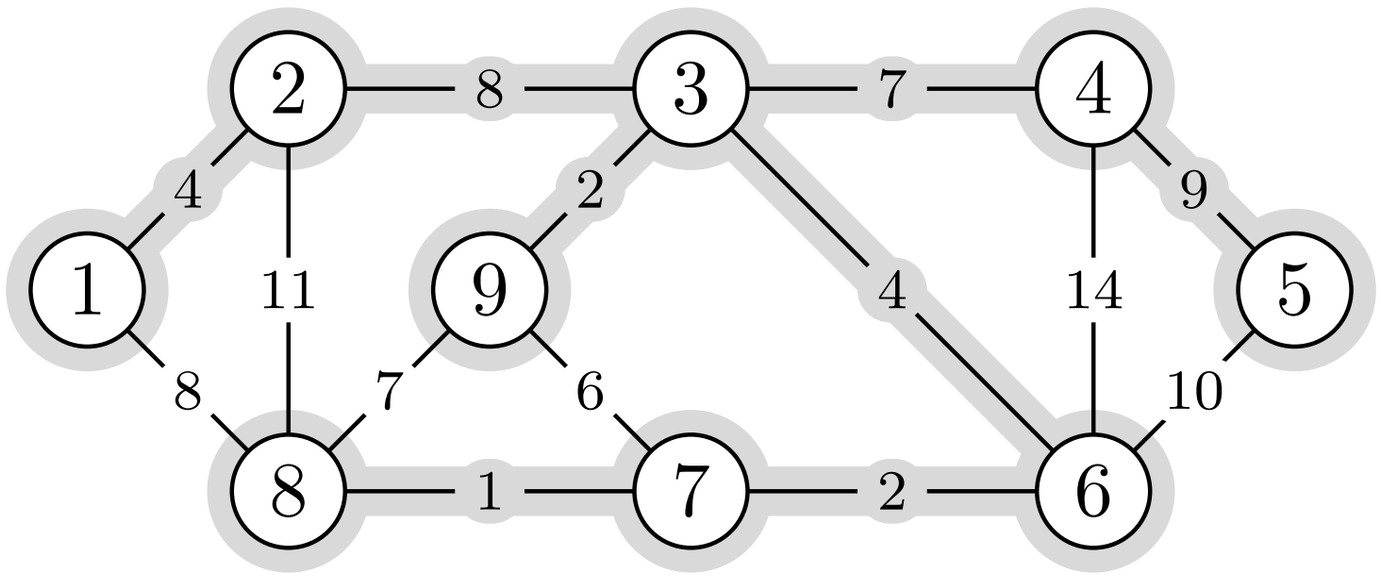
Vi vil knytte sammen nodene billigst mulig



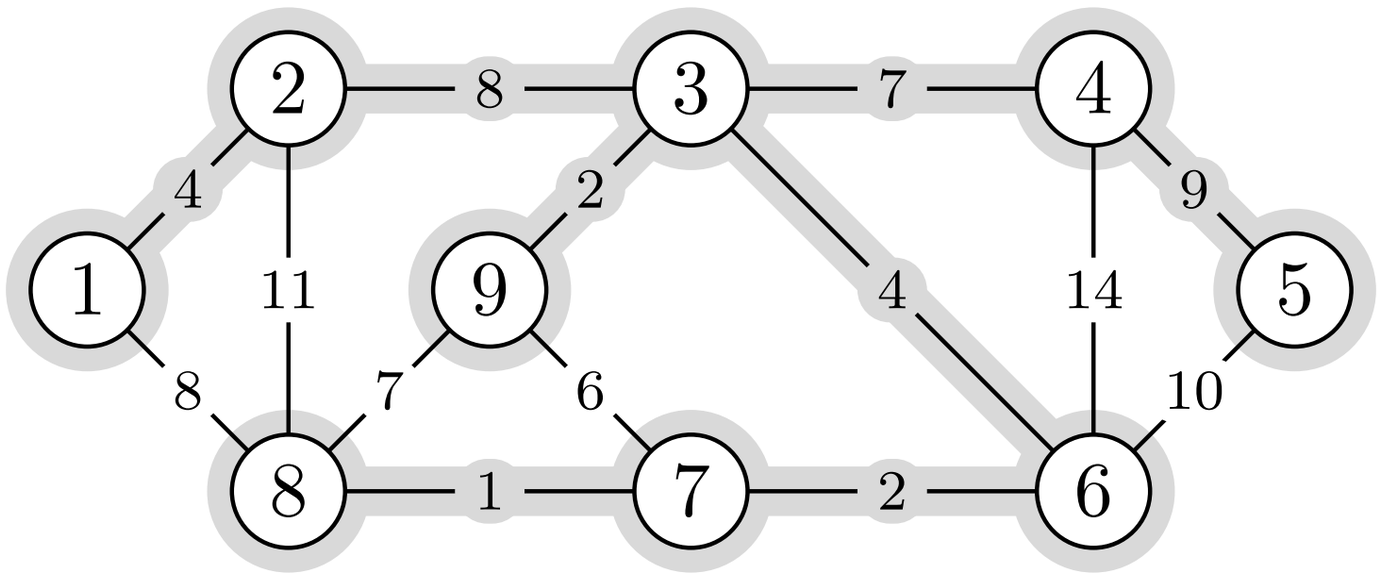
Delmengde $\mathrm{T} \subseteq \mathrm{E}$ som spenner over V og minimerer $\sum_{e \in \mathrm{T}} w(e)$



Delmengde $\mathrm{T} \subseteq \mathrm{E}$ som spenner over V og minimerer $\sum_{e \in \mathrm{T}} w(e)$



Hvis w er positiv vil T alltid bli asyklisk



Generelt: Tillater negativ w men krever asyklisk T

Input: En urettet graf $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ og en vektfunksjon $w : \mathrm{E} \to \mathbb{R}$.

Input: En urettet graf $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ og en vektfunksjon $w : \mathrm{E} \to \mathbb{R}$.

Output: En asyklisk delmengde $\mathrm{T} \subseteq \mathrm{E}$ som kobler sammen nodene i V og som minimerer vektsummen

$$w(\mathrm{T}) = \sum_{(u,v) \in \mathrm{T}} w(u, v) \, .$$

Input: En urettet graf $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ og en vektfunksjon $w : \mathrm{E} \to \mathbb{R}$.

Output: En asyklisk delmengde $\mathrm{T} \subseteq \mathrm{E}$ som kobler sammen nodene i V og som minimerer vektsummen

$$w(\mathrm{T}) = \sum_{(u,v) \in \mathrm{T}} w(u, v) \, .$$

T er altså et spenntre for G

- Vi utvider en kantmengde (partiell løsning) gradvis
- Invariant: Kantmengden utgjør en del av et minimalt spenntre
- En «trygg kant» er en kant som bevarer invarianten

GENERIC-MST(G, w)

Grådig fremgangsmåte for å finne minimale spenntrær

GENERIC-MST(G, w)
1 A = $\emptyset$

Kantene vi har valgt; skal utgjøre spenntre til slutt

GENERIC-MST(G, w)
1 A = $\emptyset$
2 **while** A does not form a spanning tree

Terminerer om G er sammenhengende og vi ikke lager sykler

GENERIC-MST(G, w)
1 A = $\emptyset$
2 **while** A does not form a spanning tree
3 find an edge (u, v) that is safe for A

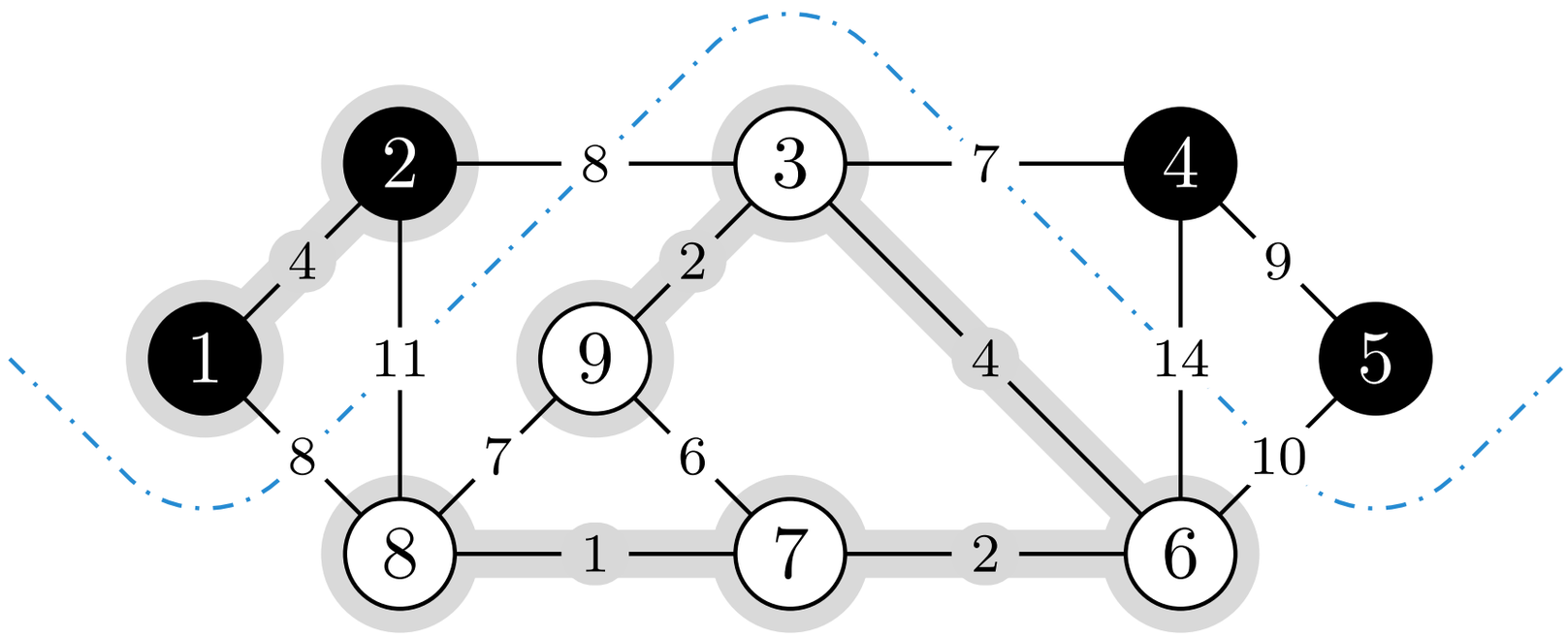
Det grådige valget. (Men . . . hva er trygt?)

GENERIC-MST(G, w)
1 A = $\emptyset$
2 **while** A does not form a spanning tree
3 find an edge (u, v) that is safe for A
4 A = A $\cup \{(u, v)\}$

GENERIC-MST(G, w)
1 A = $\emptyset$
2 **while** A does not form a spanning tree
3 find an edge (u, v) that is safe for A
4 A = A $\cup \{(u, v)\}$
5 **return** A

Et snitt er bare en bipartisjon av nodene i grafen. Et snitt respekterer A dersom ingen av kantene i A krysser snittet (dvs., går mellom de to partisjonene).

S

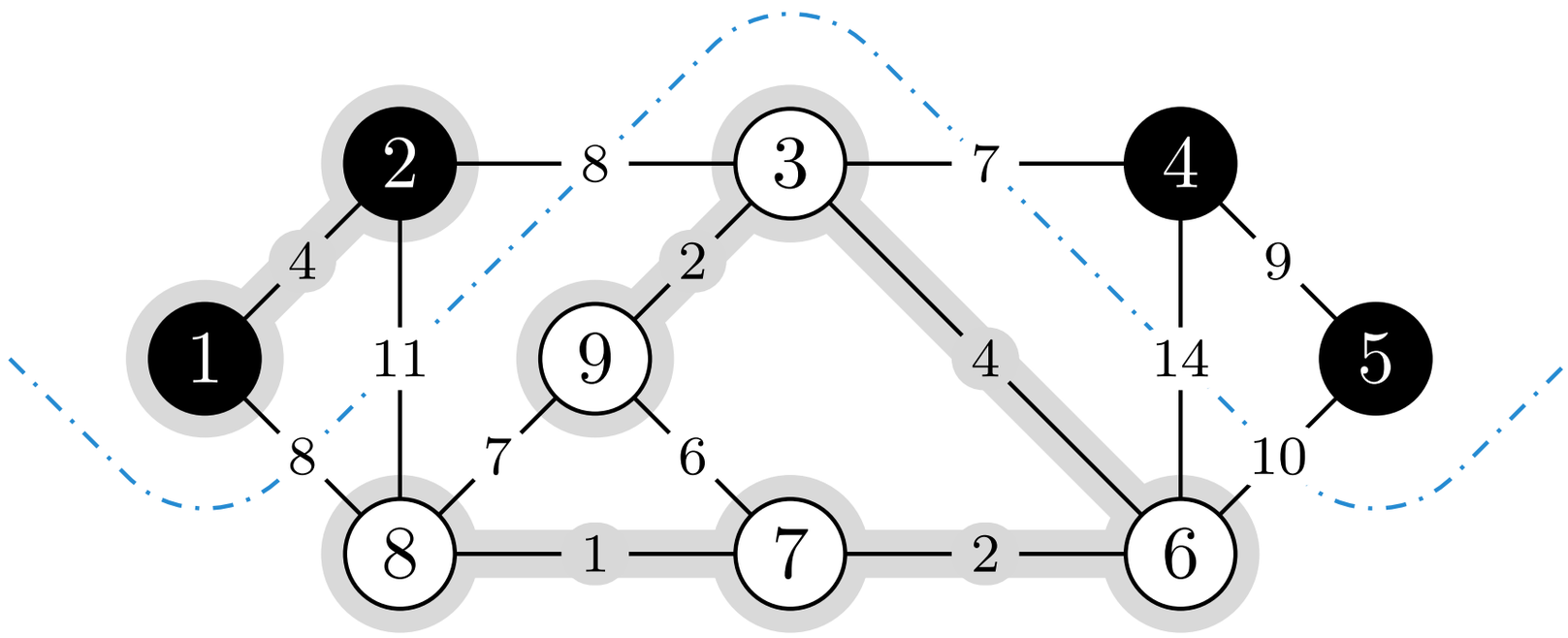


V – S

Et snitt (S, V – S) som respekterer kantmengden A (uthevet)

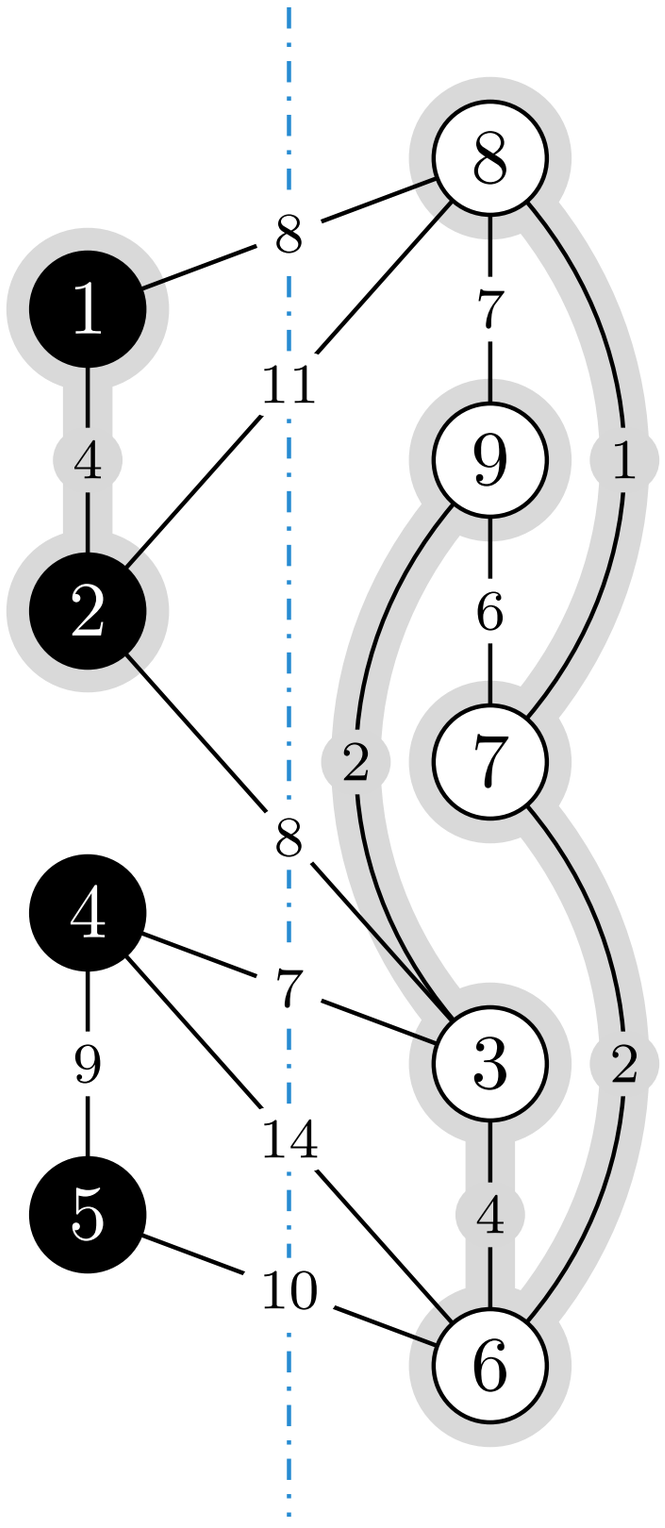
En «lett» kant over et snitt er en med minimal vekt blant kantene som går over snittet.

S



V − S

Kanten $(4, 3)$ er en (unik) lett kant over snittet



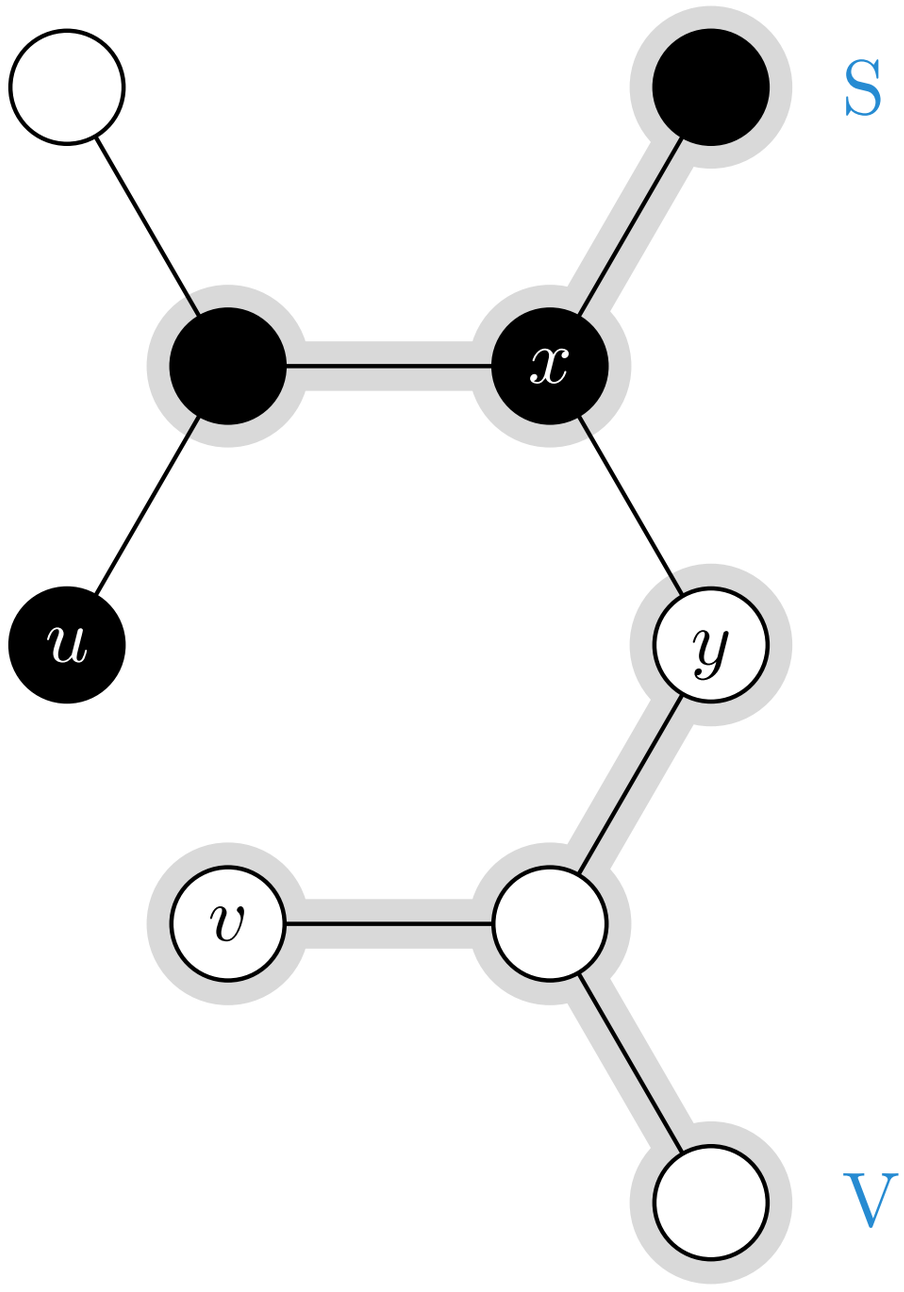
Samme snitt, tegnet på en annen måte. Vi vurderer her altså kanter som går på tvers av midtlinjen.

S

V – S

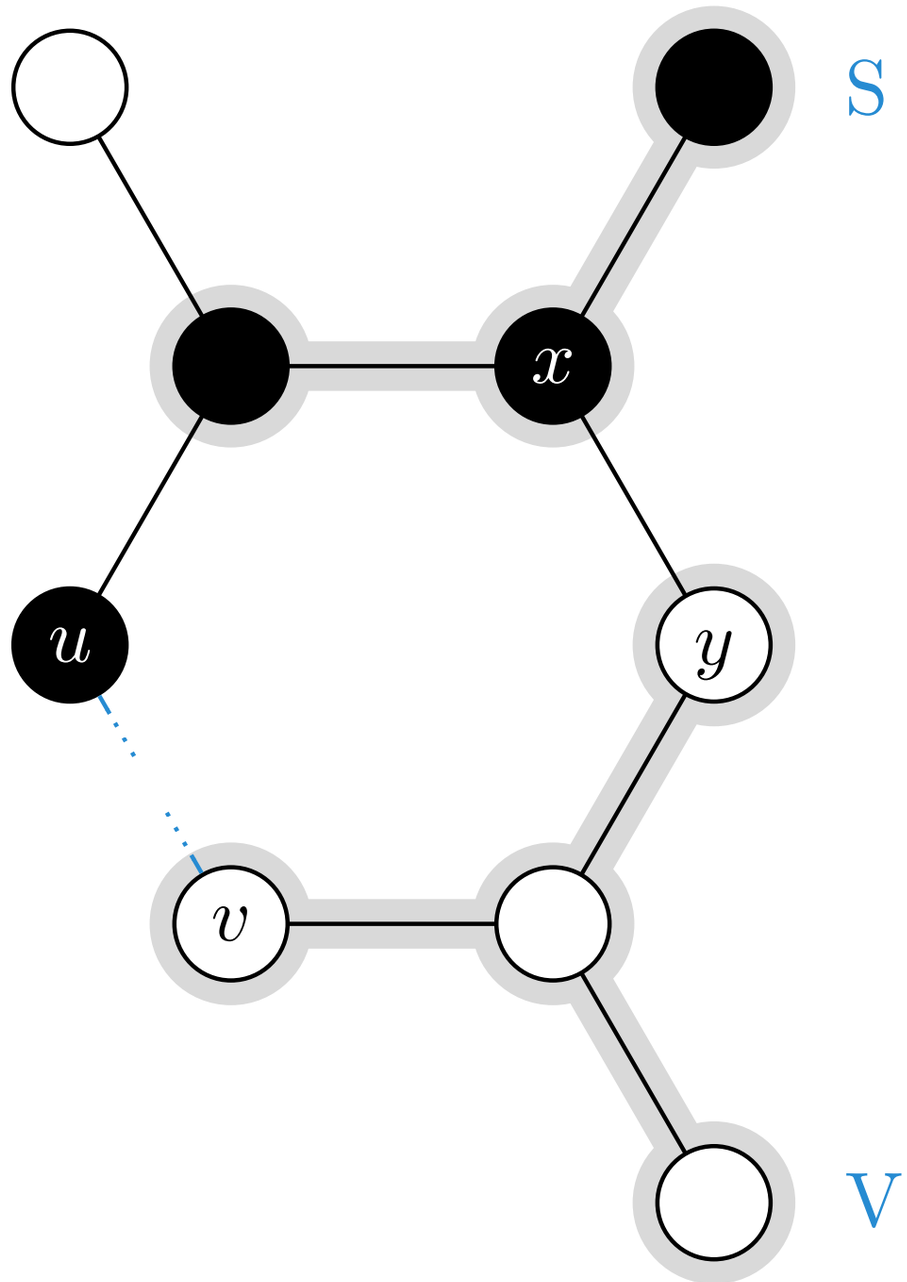
- «Exchange argument», som før
- Ta en optimal (eller vilkårlig) løsning som ikke har valgt grådig
- Vis at vi kan endre til det grådige valget uten å få en dårligere løsning

T er den delen av grafen som er tegnet opp her, altså de nodene og kantene vi ser. (Resten av grafen er skjult.)

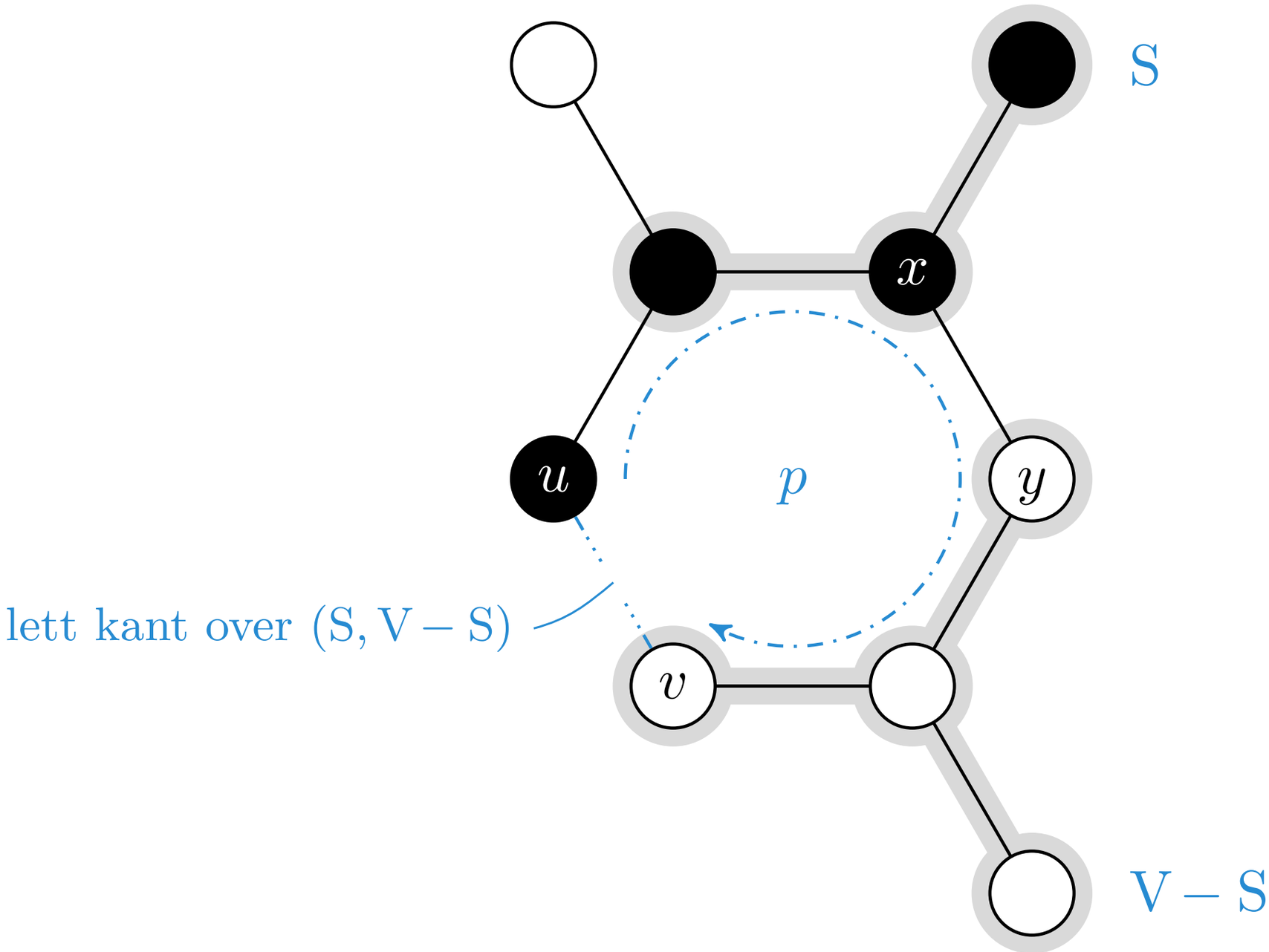


Mengden S er de svarte nodene, så snittet går mellom de svarte og hvite nodene. Mengden A er de uthevede kantene (som altså respekterer snittet; ingen kanter går mellom en svart og en hvit node.)

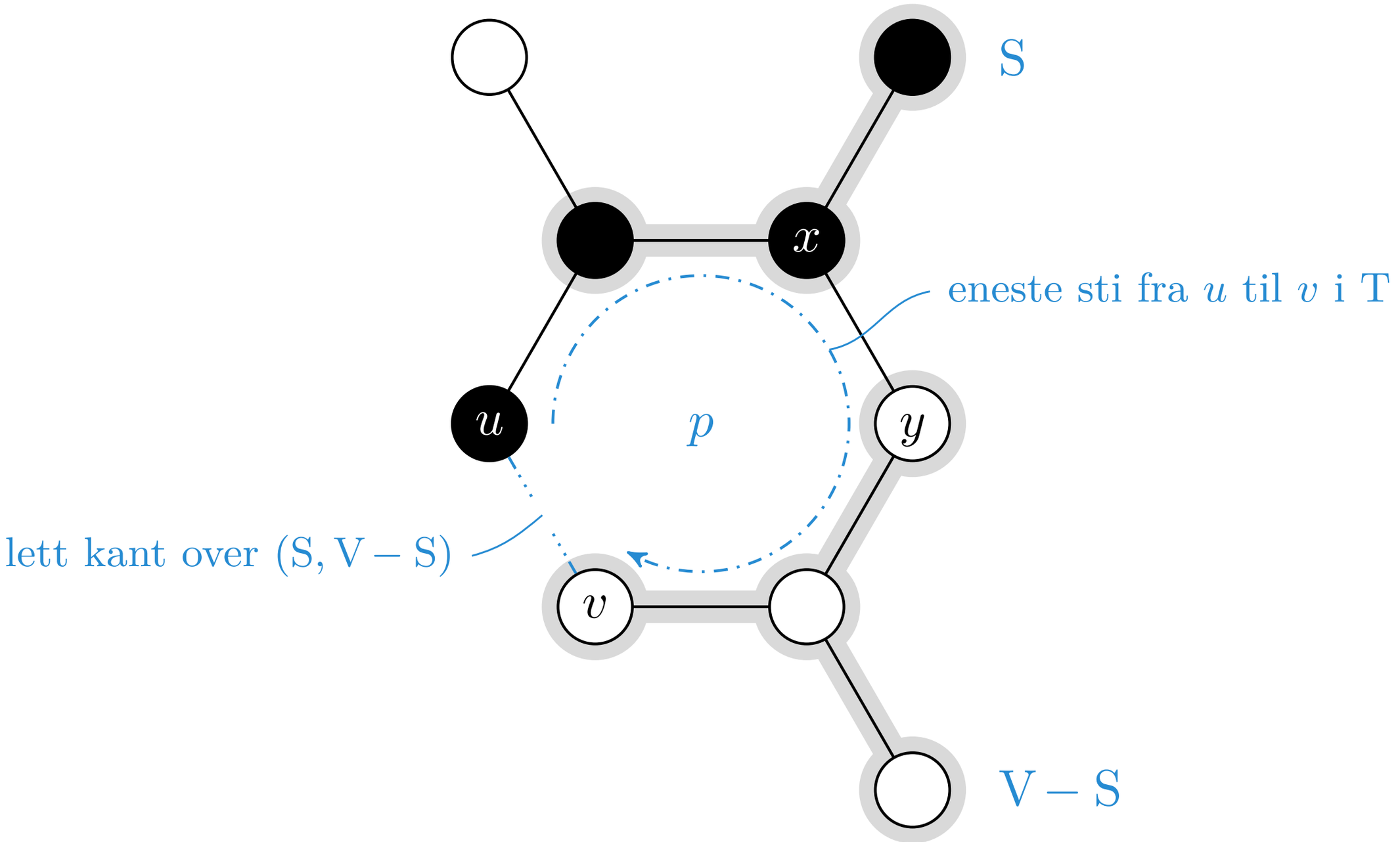
Spenntreet T inneholder A og snittet (S, V – S) respekterer A



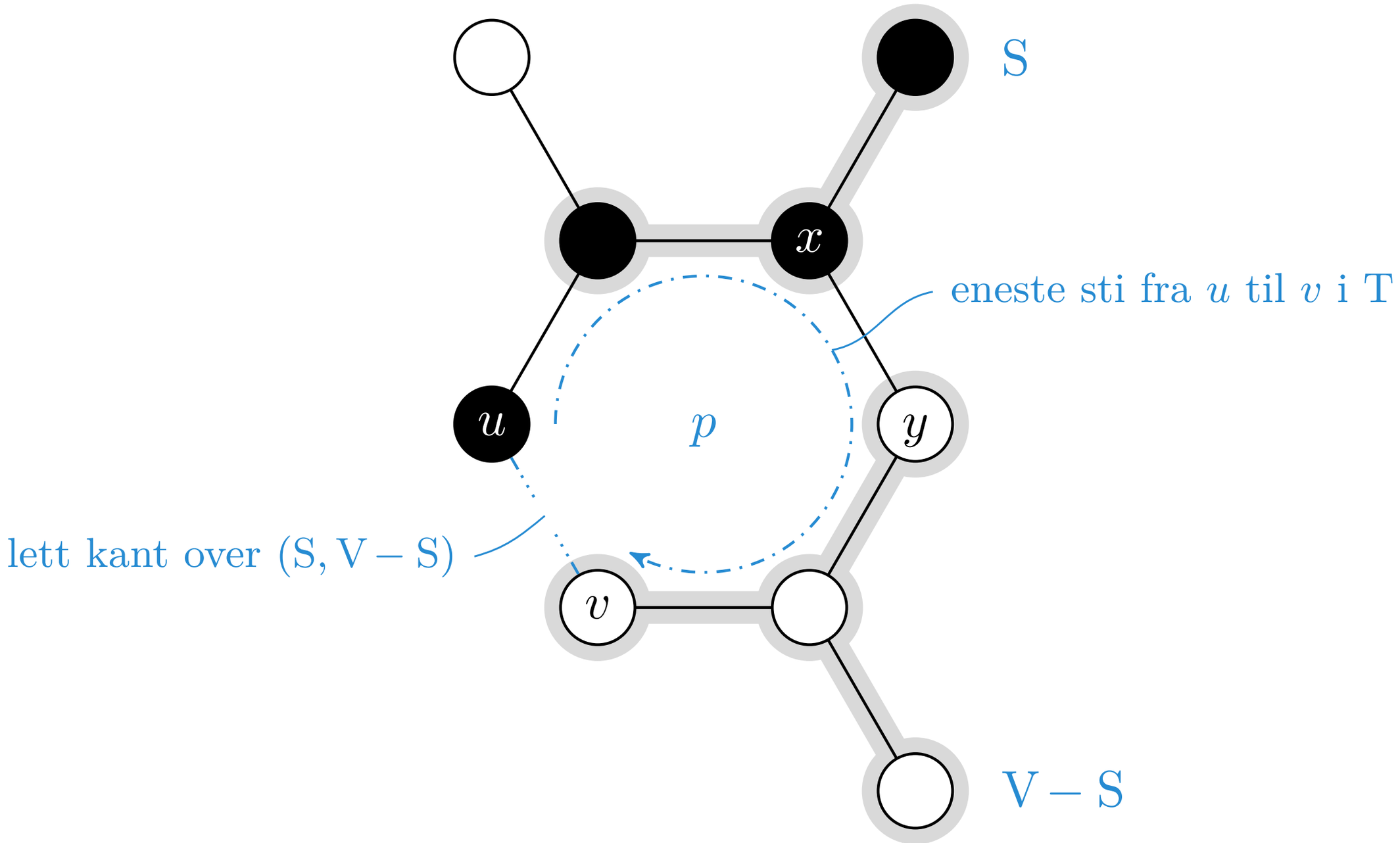
Kanten (u, v) er en lett kant over snittet $(\mathrm{S}, \mathrm{V} - \mathrm{S})$



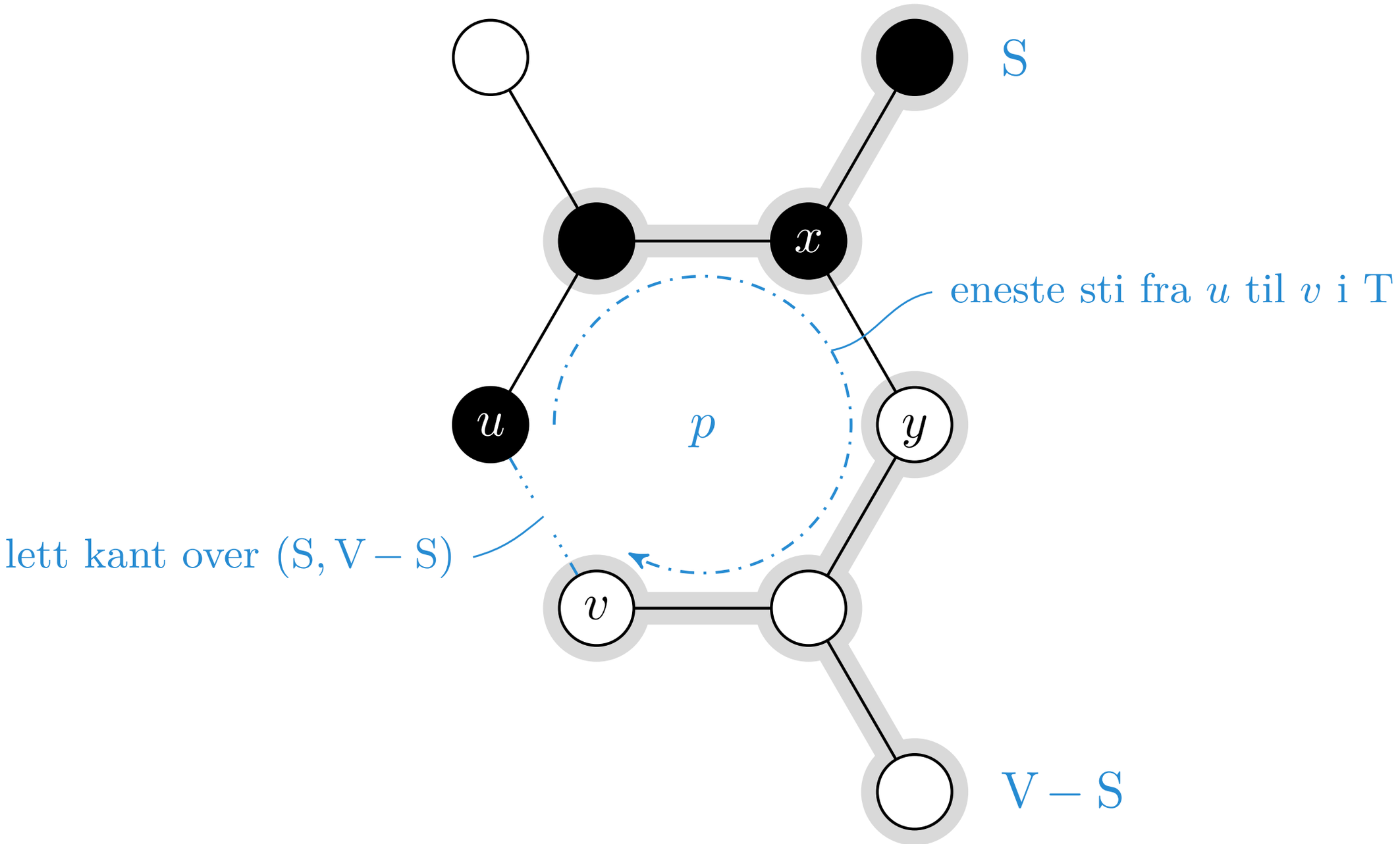
Et spenntre vil ha nøyaktig én sti fra u til v



T har én kant (x, y) over snittet, og $w(u, v) \leq w(x, y)$



Hvis (u, v) erstatter (x, y) i T′, får vi $w(\mathrm{T}') \leq w(\mathrm{T})$



T var vilkårlig og T′ minst like bra, så (u, v) er trygg for A

Noe å tenke på: De hypotetiske stien hadde her én kant over snittet. Hva om den går i sikk-sakk, og har flere? Holder argumentet fortsatt?

- En lett kant over et snitt som respekterer løsningen vår er trygg
- Vi kan løse problemet grådig!
- Men ... hvilke snitt skal vi velge?

Greedy-choice: Det finnes en optimal løsning som inneholder det grådige valget.

Optimal substruktur: Om vi foretar det grådige valget, så må resten løses optimalt (og er av samme type).

Kruskal: En kant med minimal vekt blant de gjenværende er trygg så lenge den ikke danner sykler.

Kruskal: En kant med minimal vekt blant de gjenværende er trygg så lenge den ikke danner sykler.

Prim: Bygger ett tre gradvis; en lett kant over snittet rundt treet er alltid trygg.

Prims algoritme ble egentlig først oppfunnet av Jarník

Kruskal: En kant med minimal vekt blant de gjenværende er trygg så lenge den ikke danner sykler.

Prim: Bygger ett tre gradvis; en lett kant over snittet rundt treet er alltid trygg.

Borůvka: En slags blanding. Kobler hvert tre til det nærmeste av de andre.

Kruskal: En kant med minimal vekt blant de gjenværende er trygg så lenge den ikke danner sykler.

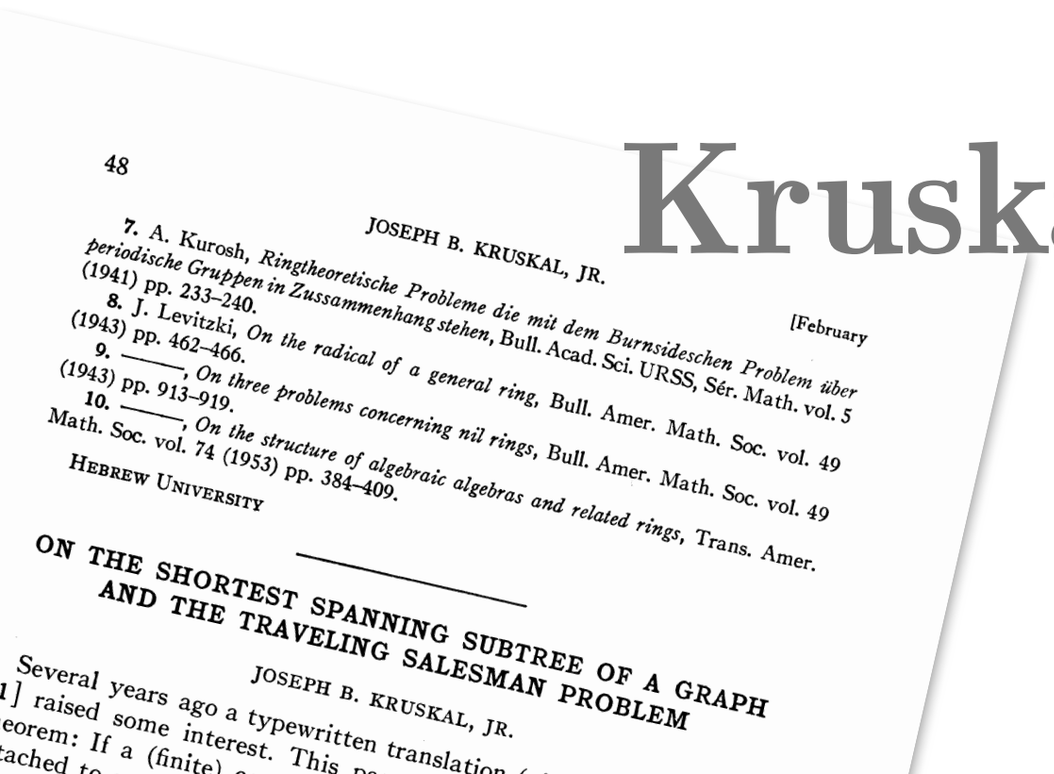
Prim: Bygger ett tre gradvis; en lett kant over snittet rundt treet er alltid trygg.

(**Borůvka:** En slags blanding. Kobler hvert tre til det nærmeste av de andre.)

Borůvkas algoritme er ikke sentral; ikke beskrevet i læreboka

3.4

Kruskals algoritme



48 JOSEPH B. KRUSKAL, JR. [February

7. A. Kurosh, *Ringtheoretische Probleme die mit dem Burnsideschen Problem über periodische Gruppen in Zussammenhang stehen*, Bull. Acad. Sci. URSS, Sér. Math. vol. 5 (1941) pp. 233–240.
8. J. Levitzki, *On the radical of a general ring*, Bull. Amer. Math. Soc. vol. 49 (1943) pp. 462–466.
9. ———, *On three problems concerning nil rings*, Bull. Amer. Math. Soc. vol. 49 (1943) pp. 913–919.
10. ———, *On the structure of algebraic algebras and related rings*, Trans. Amer. Math. Soc. vol. 74 (1953) pp. 384–409.

HEBREW UNIVERSITY

ON THE SHORTEST SPANNING SUBTREE OF A GRAPH AND THE TRAVELING SALESMAN PROBLEM

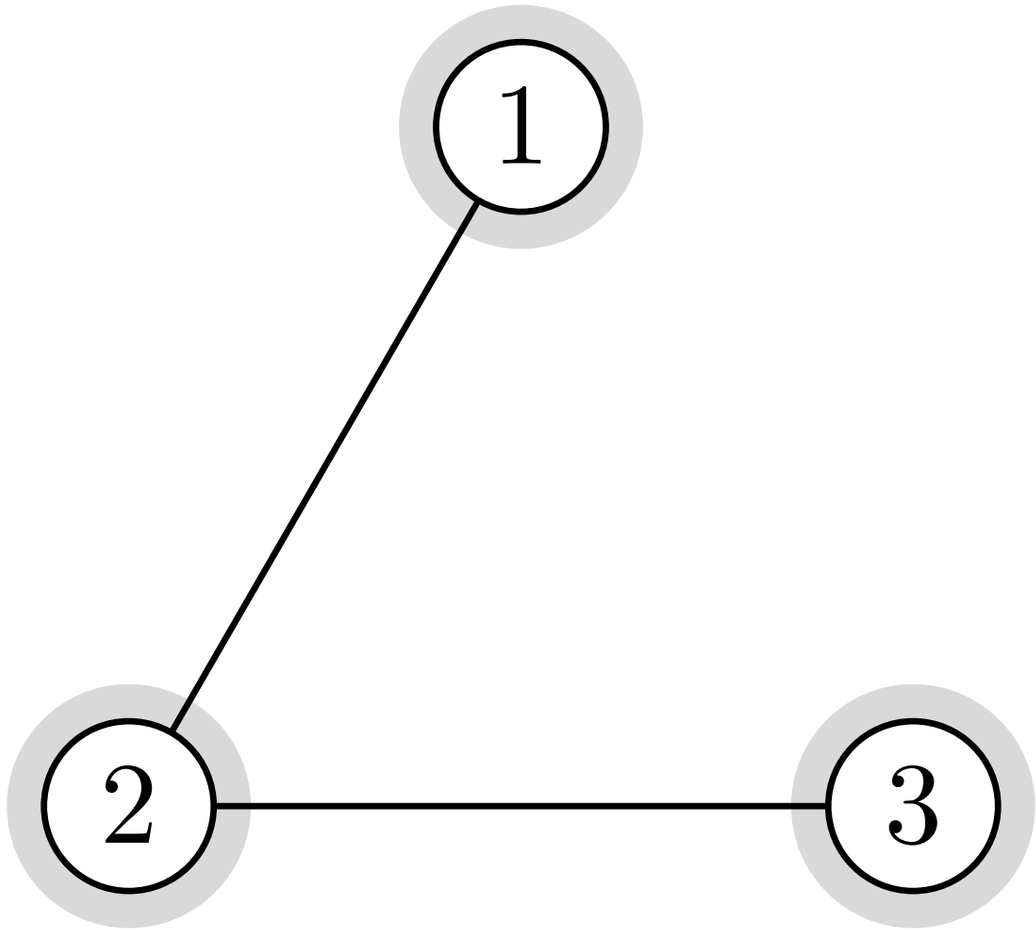
JOSEPH B. KRUSKAL, JR.

Several years ago a typewritten translation (
1] raised some interest. This pa
eorem: If a (finite)
tached to

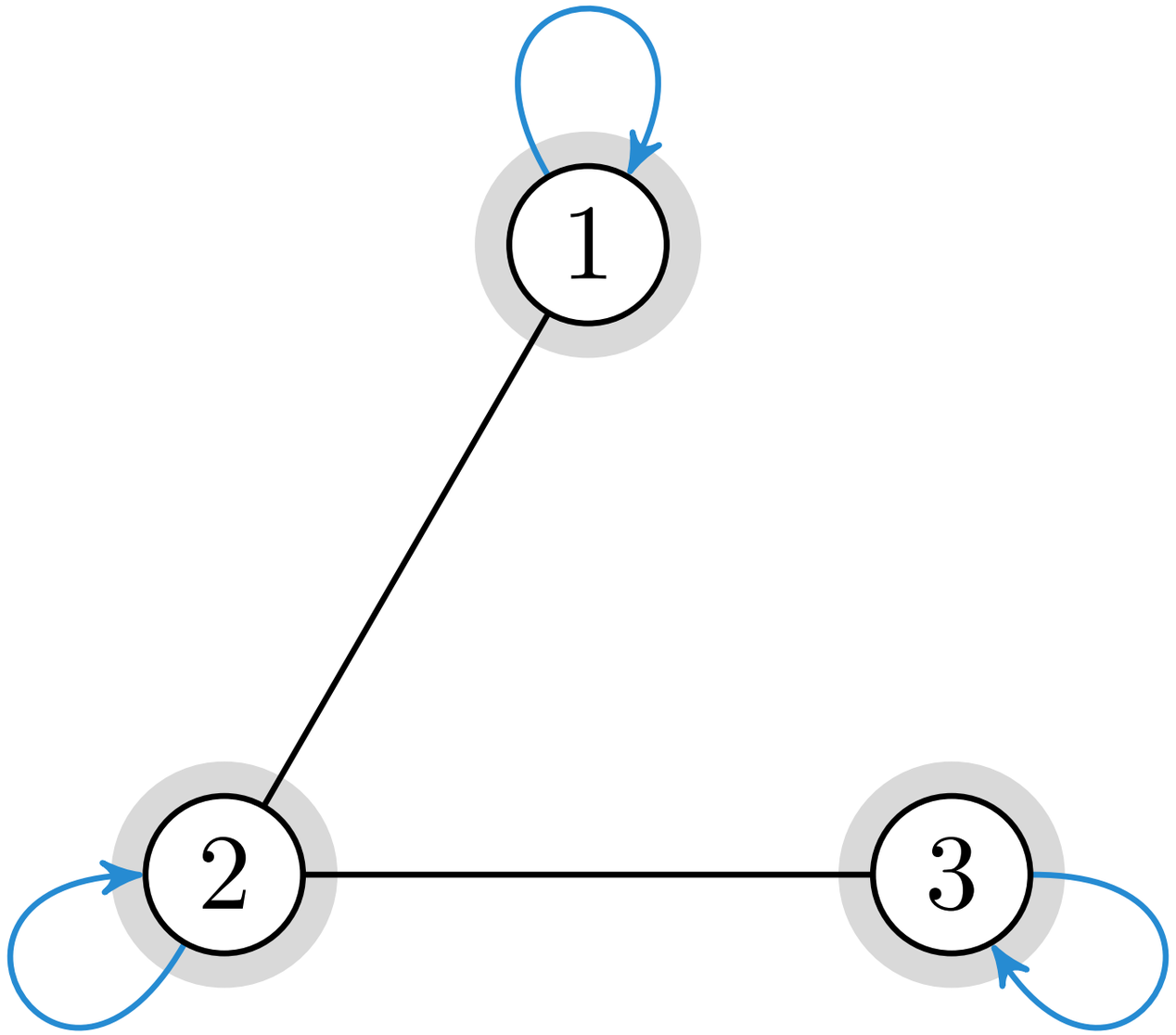
Fra 1956

Han har også et par andre varianter i artikkelen – f.eks. å starte med hele grafen, og hele tiden fjerne den lengste kanten, så lenge resultatet er sammenhengende.

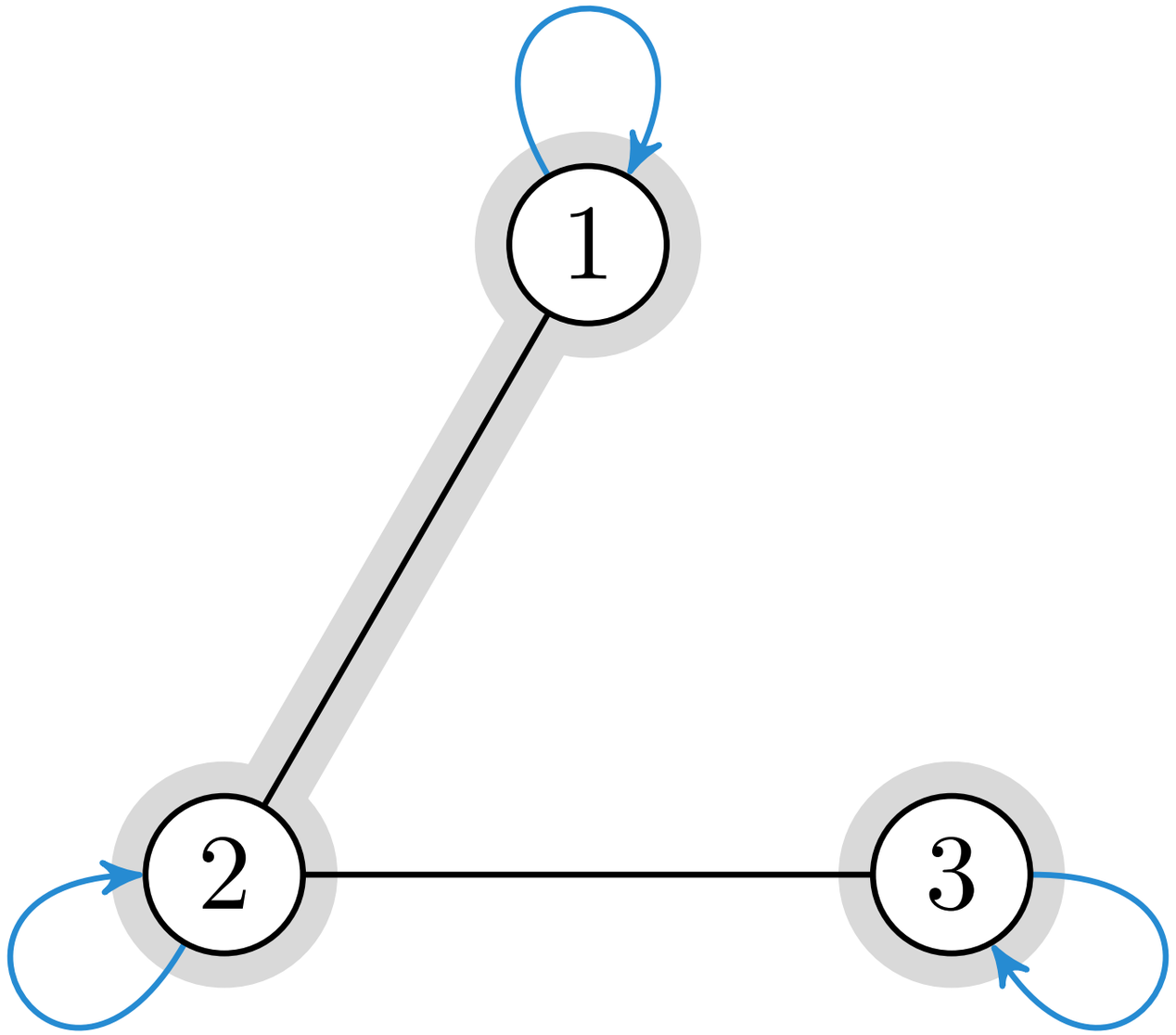
To skoger på én gang



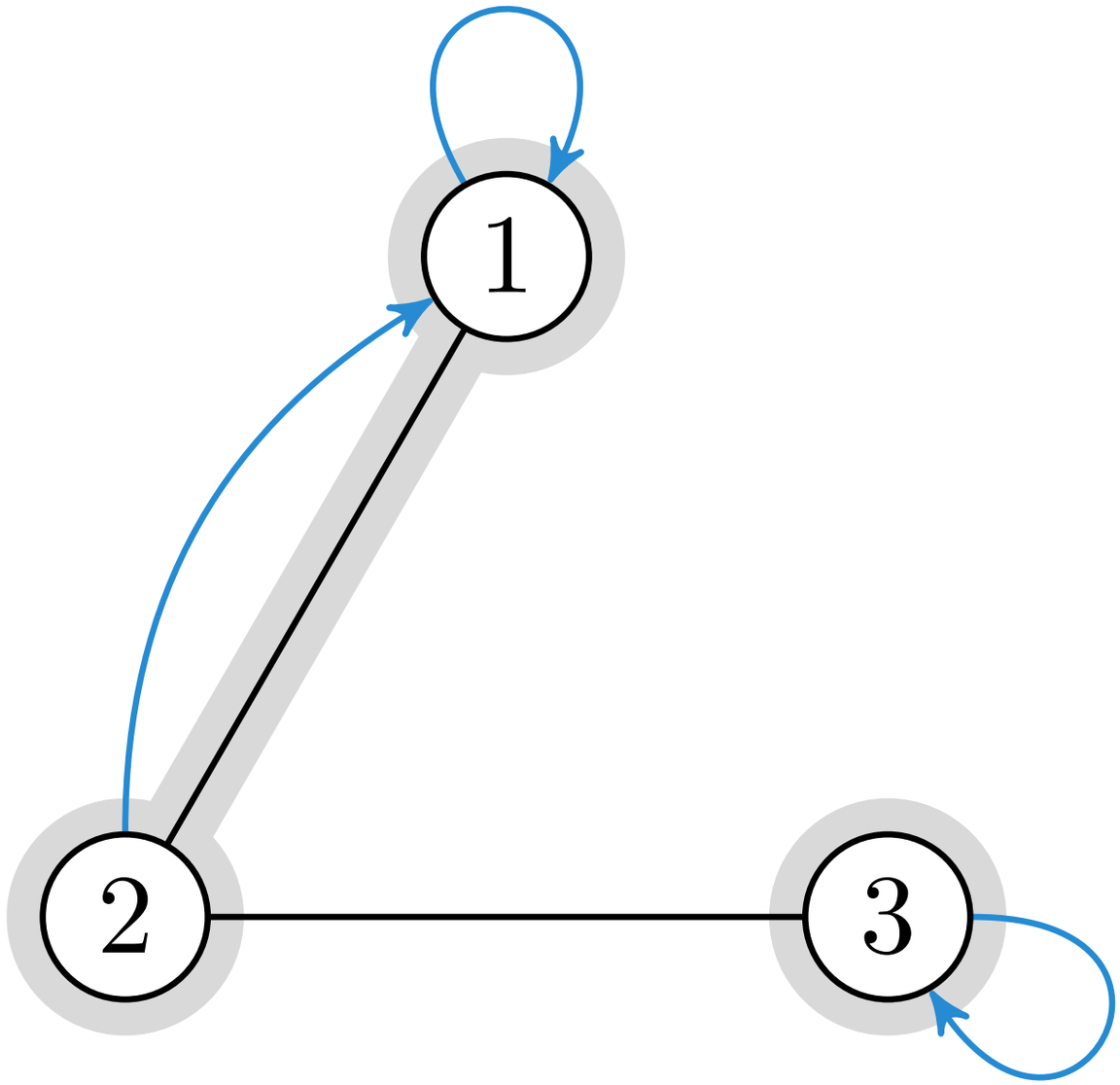
Til å begynne med er hver node en komponent i en partiell løsning



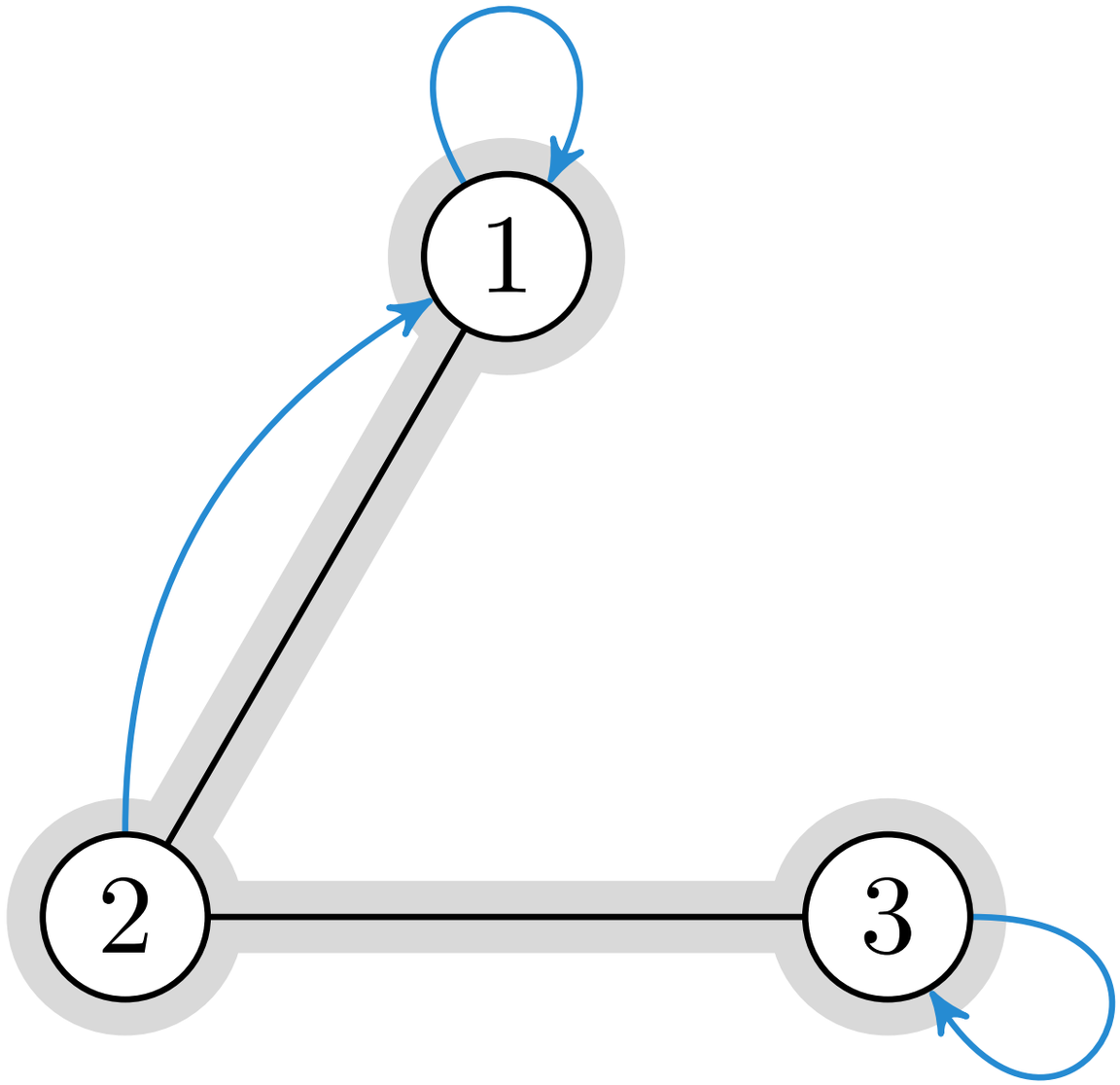
Komponenter: Disjunkte nodemengder. Starter med $v.p = v$

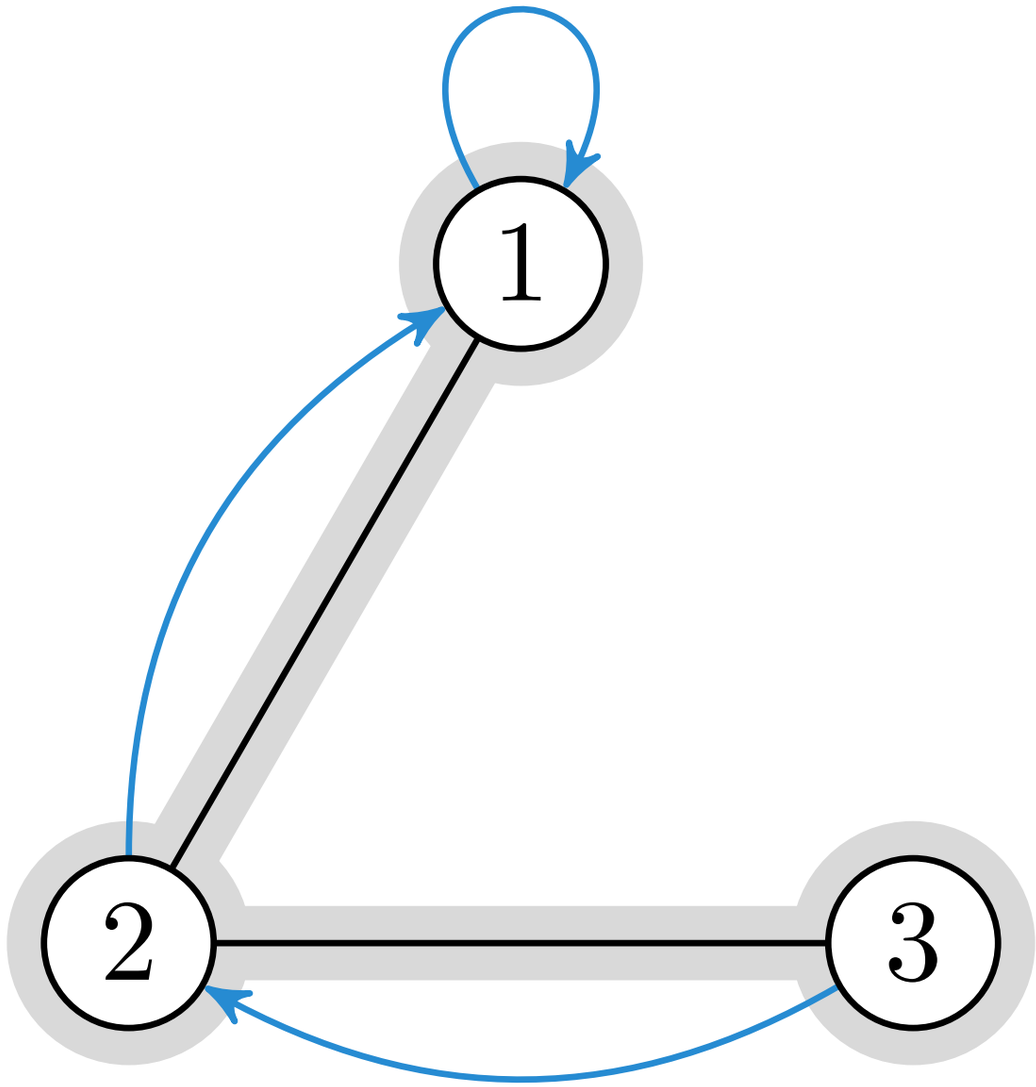


Hver kant kobler sammen to komponenter til et større tre

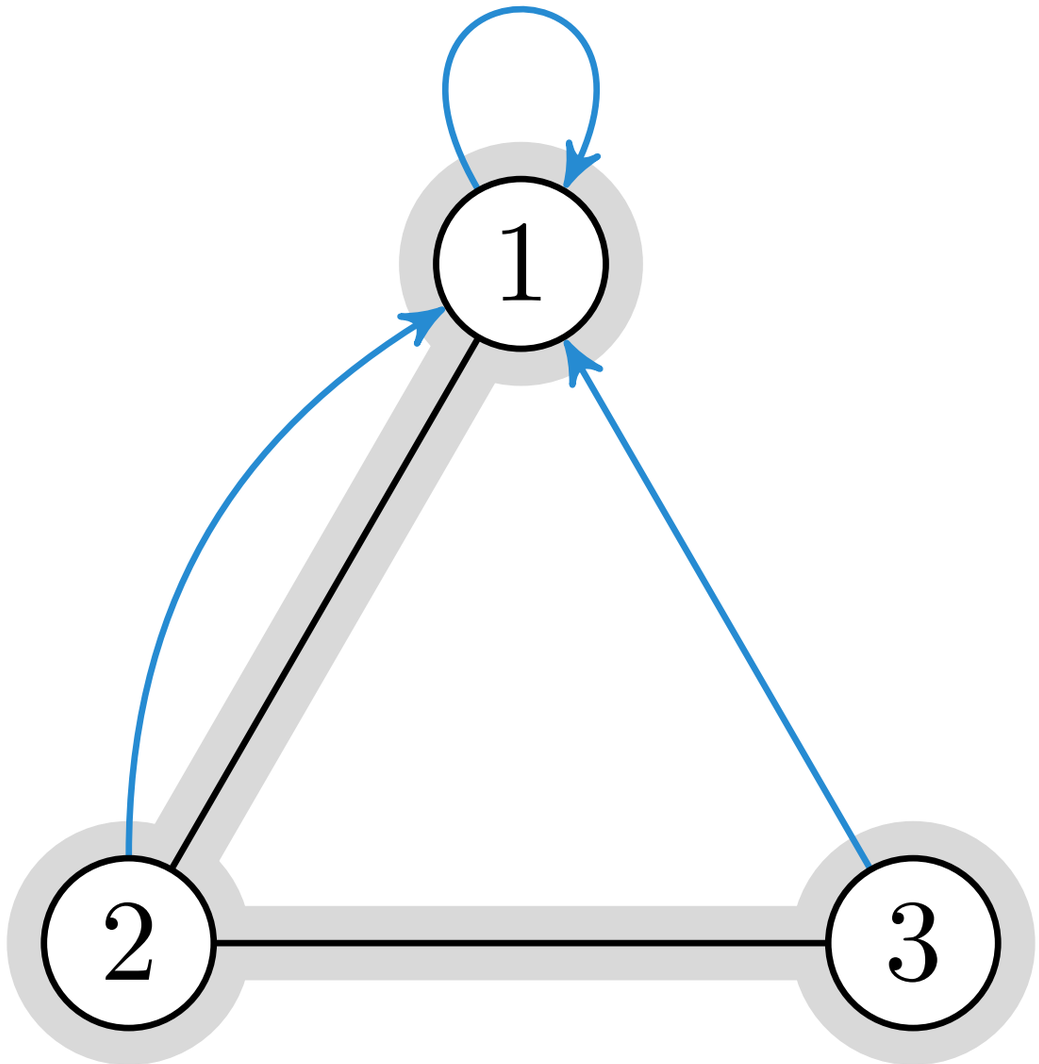


Vi kobler sammen komponentene ved å oppdatere $v.p$





Tilsammen så *kan* *v.p*-pekerne utgjøre MST-et vårt . . .



Ofte mer effektivt om de ikke gjør det!

- Én skog er fragmenter av et MST
- Den andre skogen: Disjoint-set forest
  - Samme noder og komponenter
  - Rettede kanter/pekere som spiller en helt annen rolle
- Vi behandler denne siste skogen som en «black box» i algoritmen

MST-Kruskal(G, w)

G graf
w vekting

Finn minimalt spenntre ved å velge minste lovlige kant

MST-Kruskal(G, w)
1 A = $\emptyset$

G graf
w vekting
A kantutvalg

Kantene som skal utgjøre et minimalt spenntre til slutt

mlh@ntnu.no

MST-Kruskal(G, w)
1 A = $\emptyset$
2 **for** each vertex $v \in$ G.V

G graf
w vekting
A kantutvalg

Initialisering for mengdeoperasjonene våre

MST-Kruskal(G, w)
1 A = $\emptyset$
2 **for** each vertex $v \in$ G.V
3 Make-Set(v)

G graf
w vekting
A kantutvalg

Vi starter med |V| individuelle partielle spenntrær

```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3        Make-Set(v)
4  sort G.E by w
```

G graf
w vekting
A kantutvalg

Preprosessering: Vil hele tiden velge minste kant

```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
```

G graf
w vekting
A kantutvalg

Altså i stigende rekkefølge, etter vekt

```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
```

G graf
w vekting
A kantutvalg

Mellom to ulike komponenter av den partielle løsningen?

```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
```

G graf
w vekting
A kantutvalg

Da dannes ingen sykler, og vi kan velge kanten

```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
```

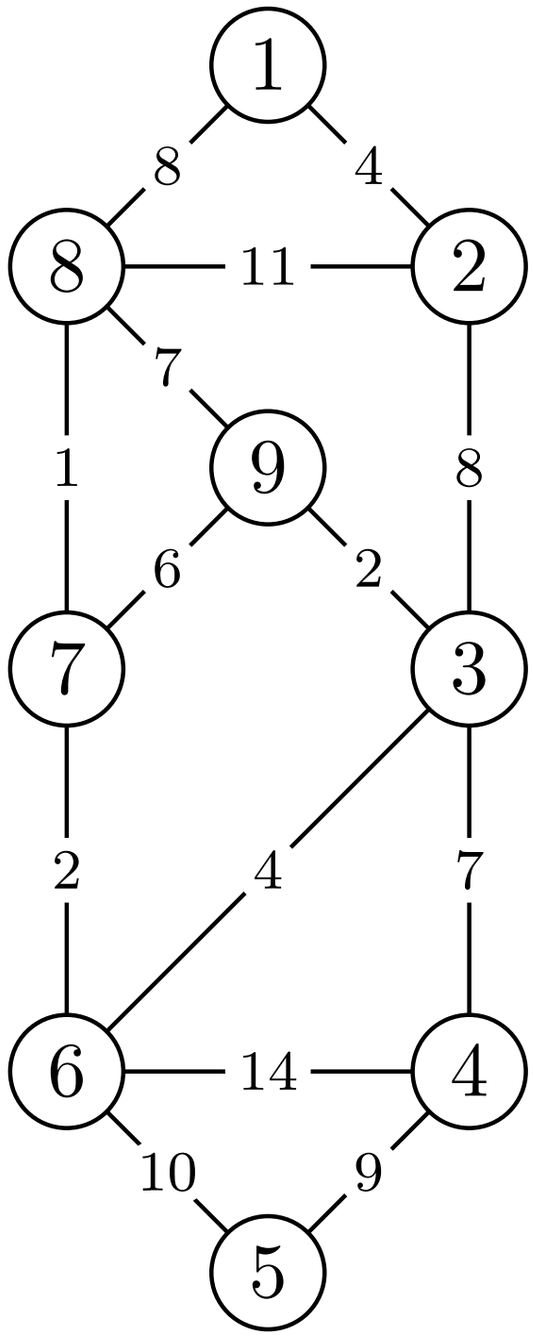
G graf
w vekting
A kantutvalg

Vi kobler sammen to partielle spenntrær til ett

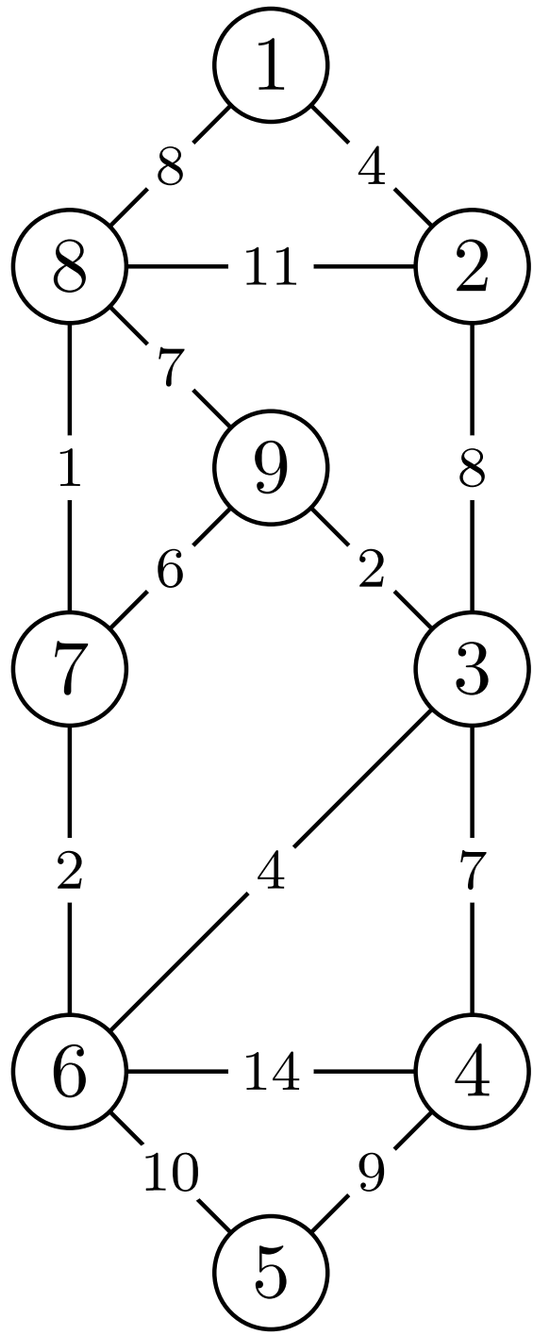
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```

G graf
w vekting
A kantutvalg

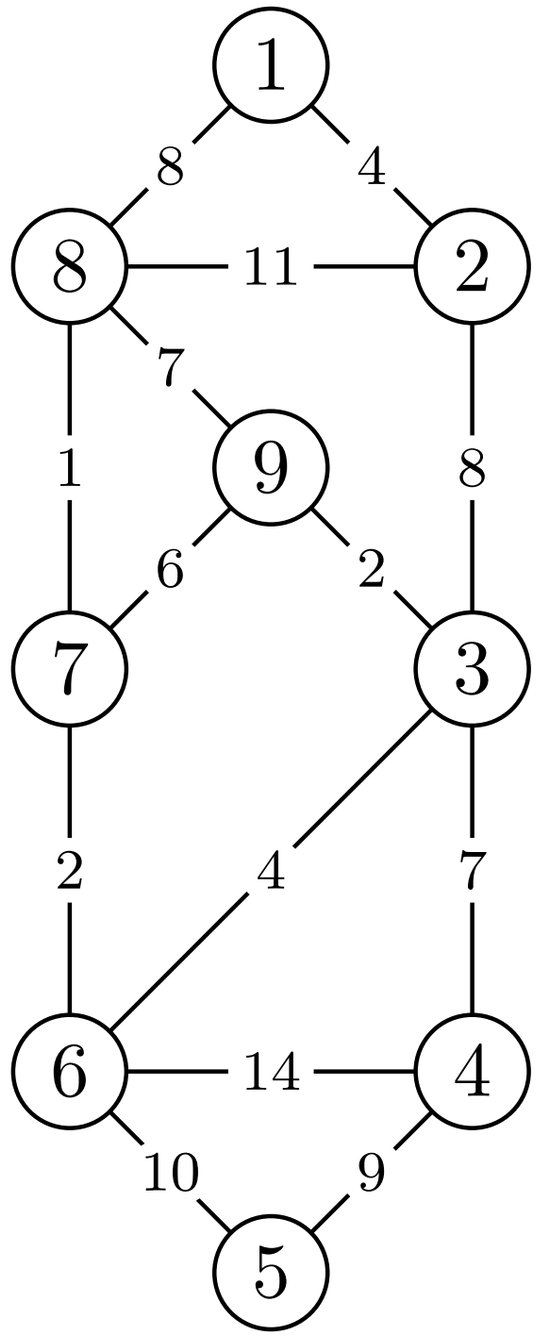
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



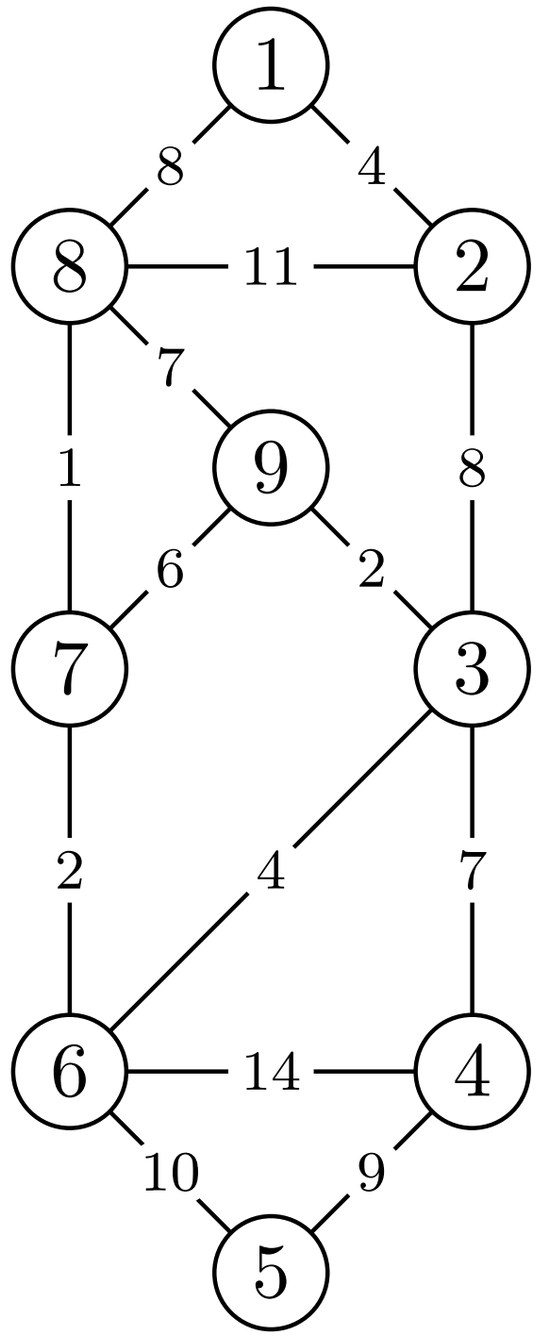
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



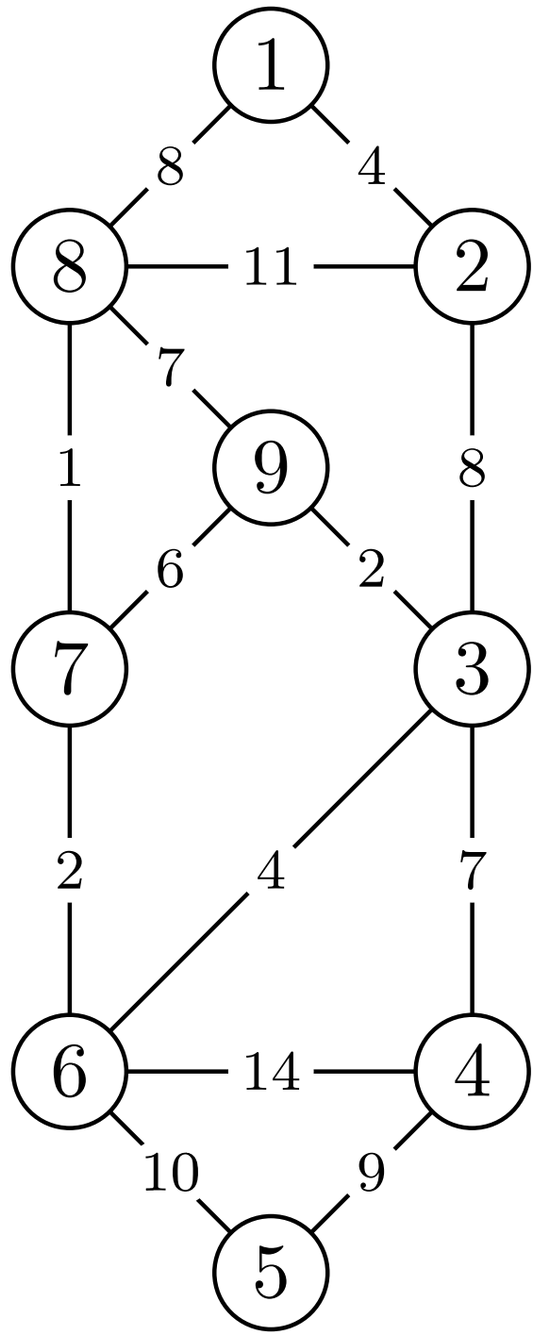
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



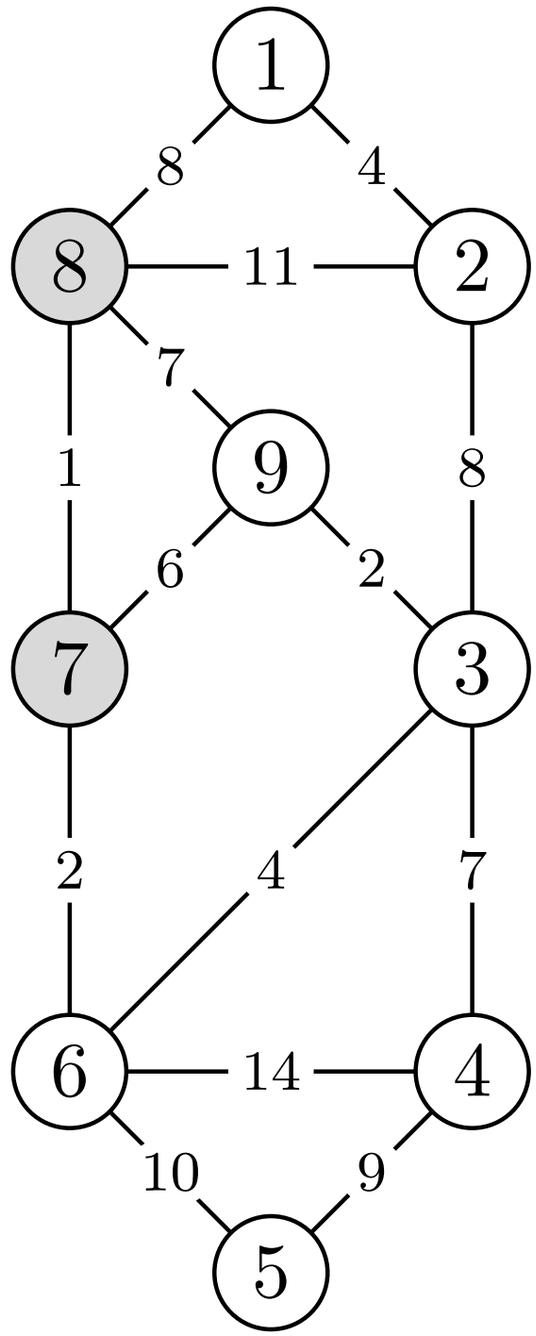
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



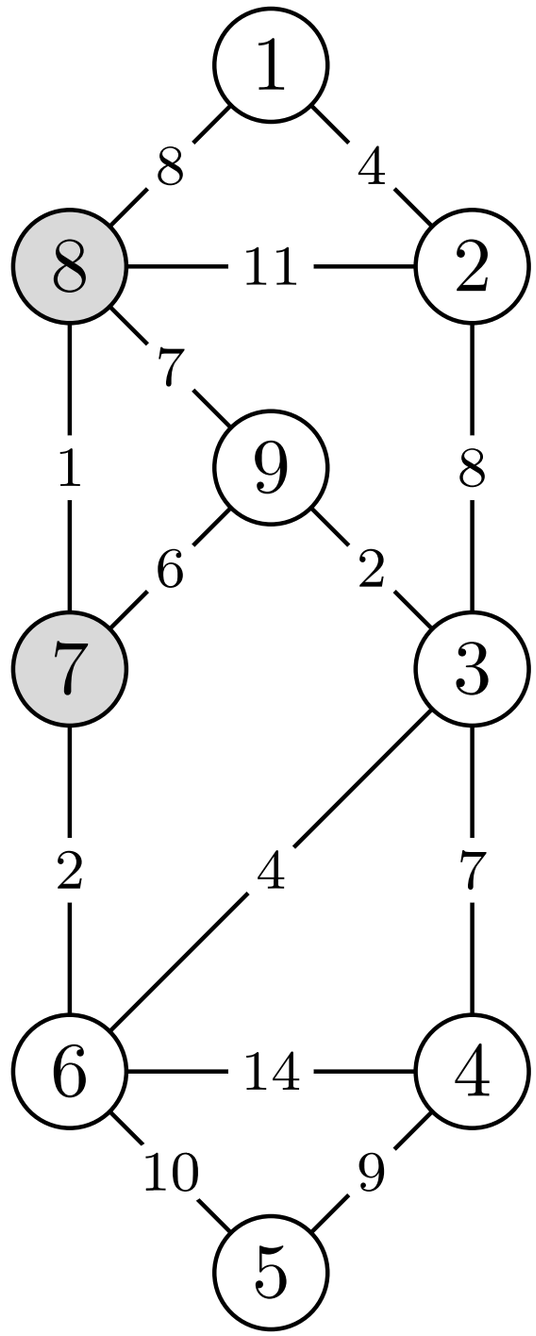
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



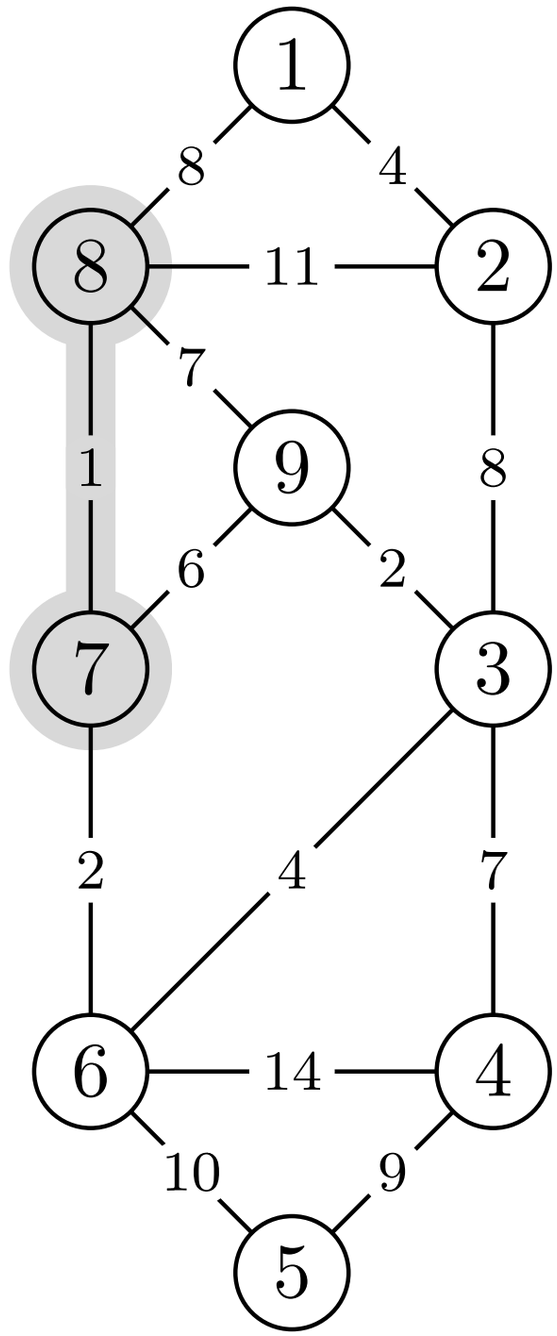
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



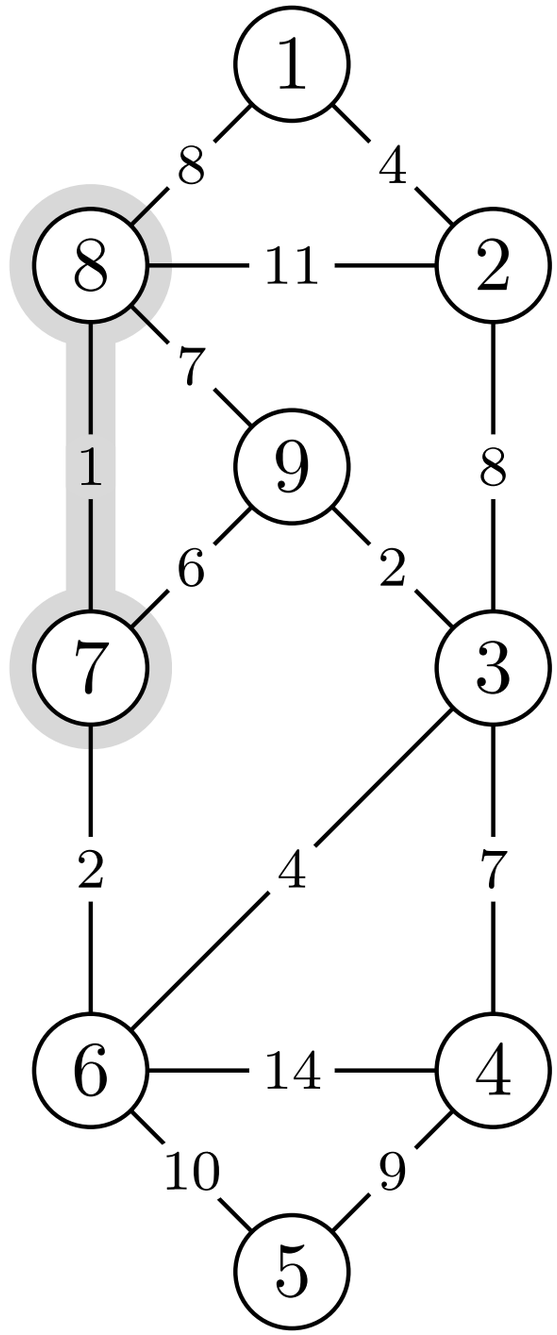
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



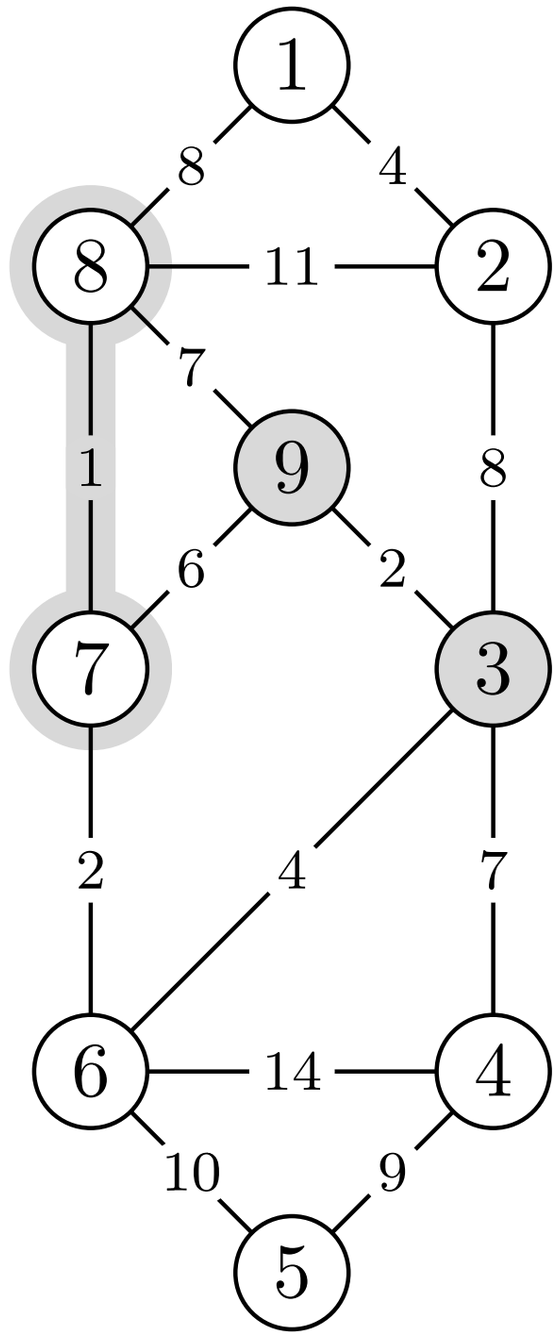
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



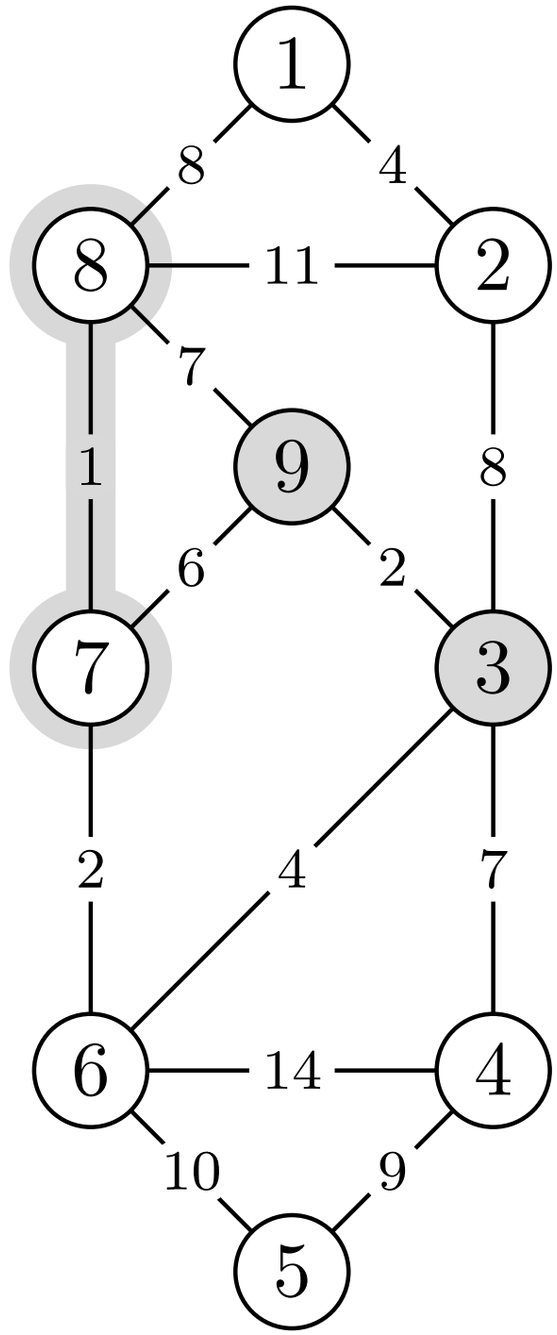
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



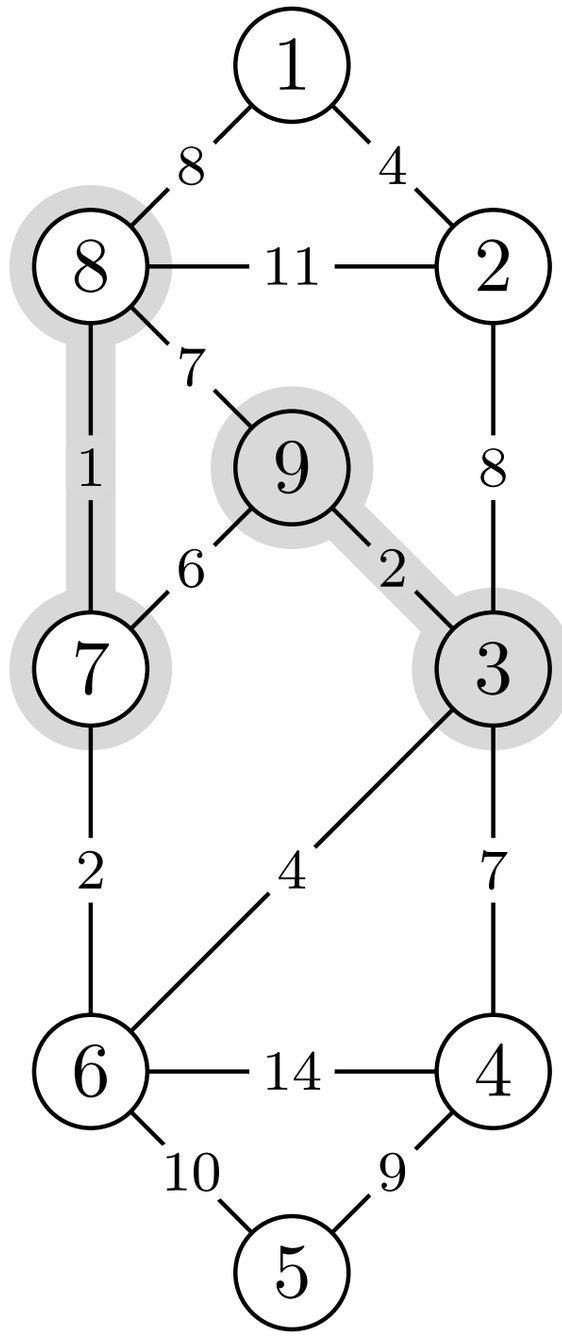
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



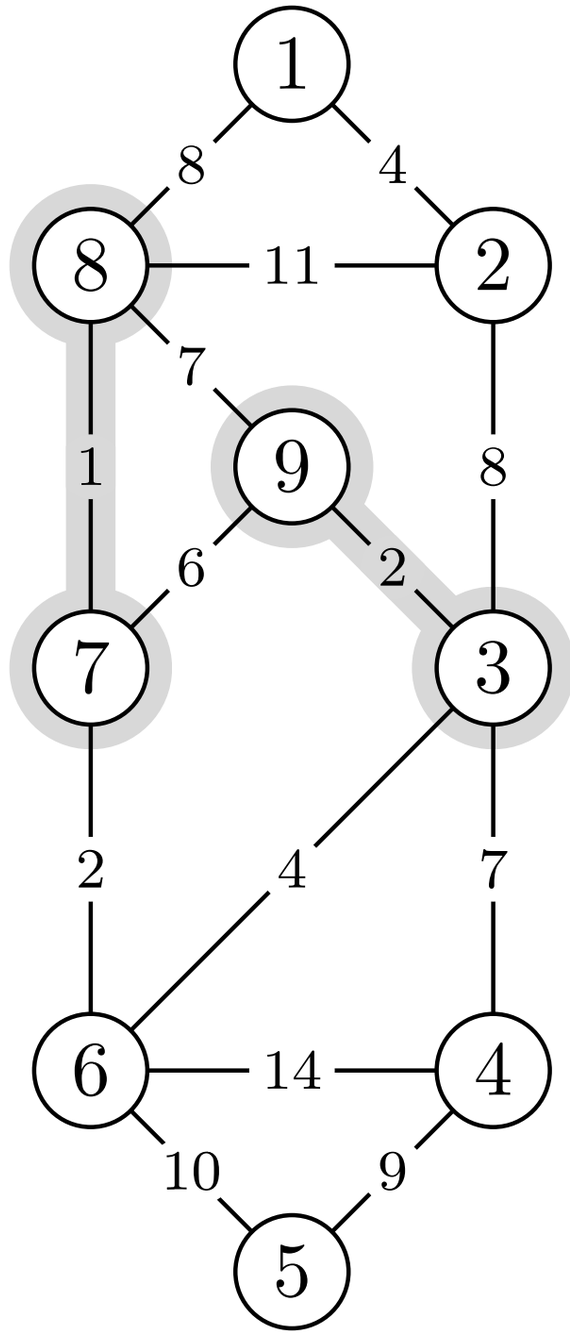
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



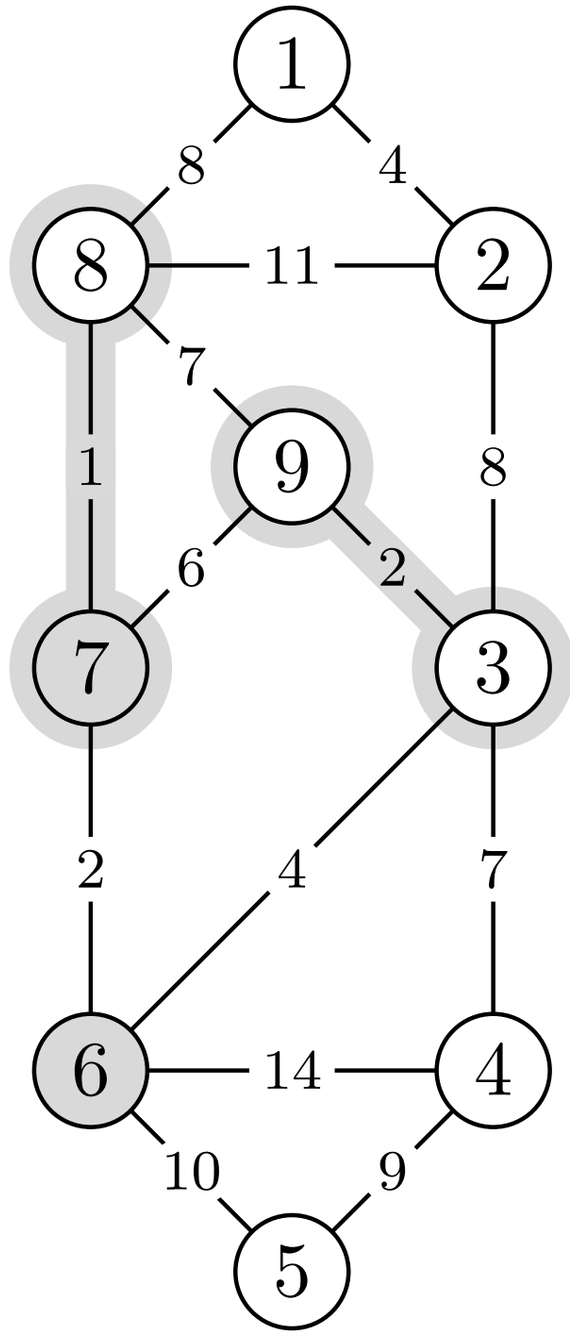
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



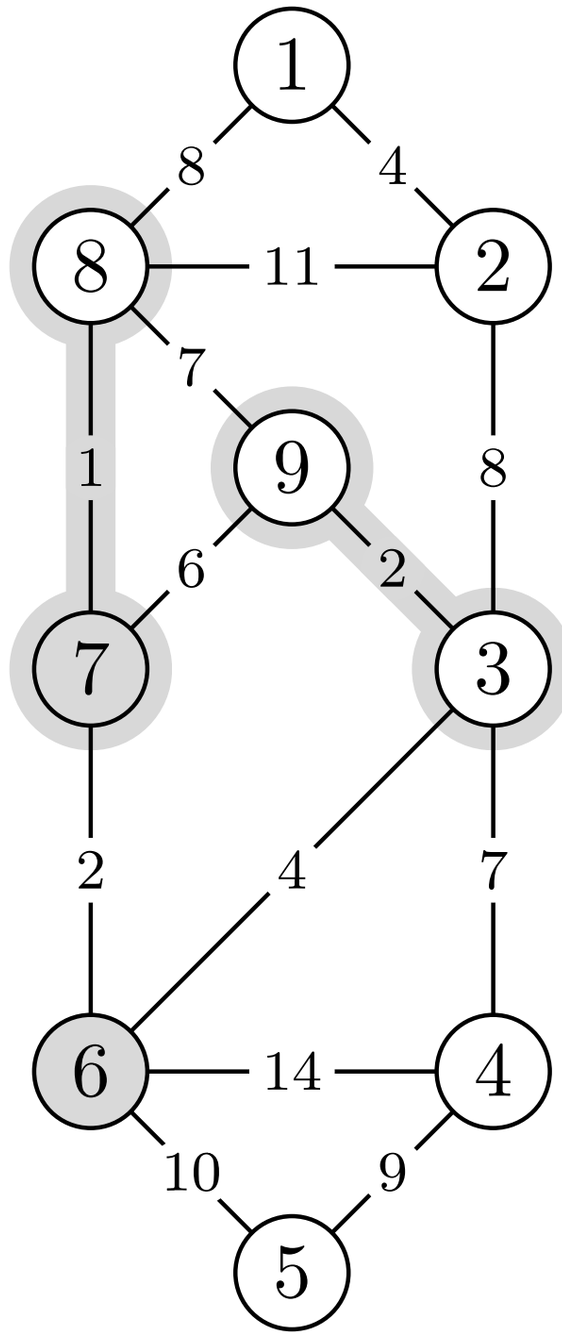
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



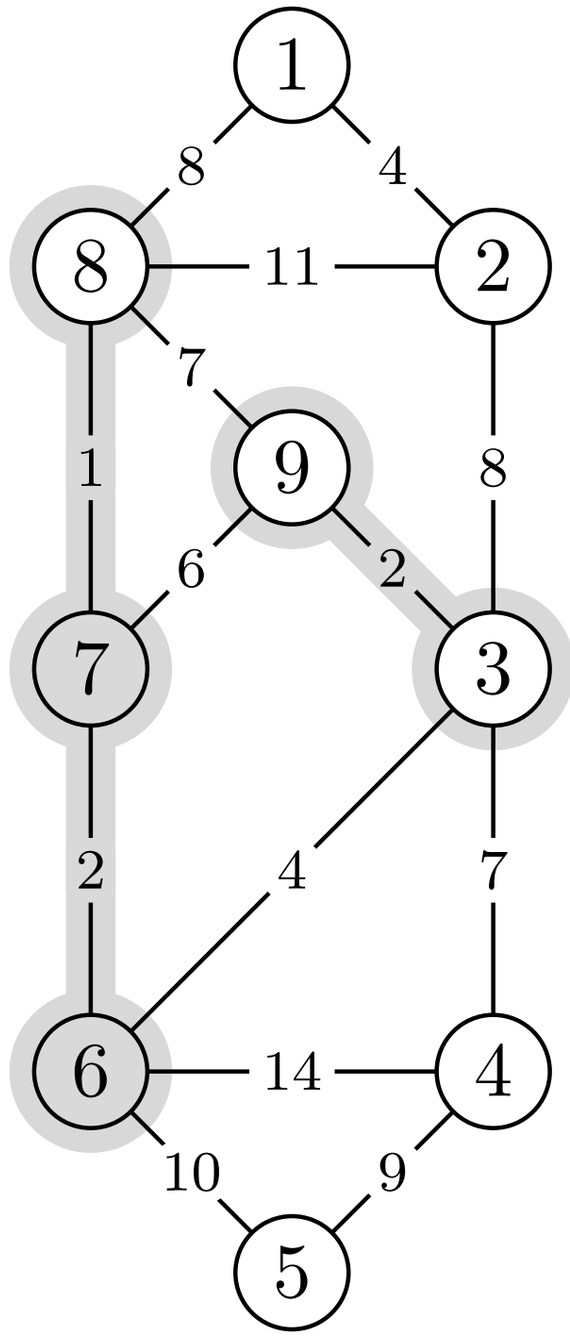
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



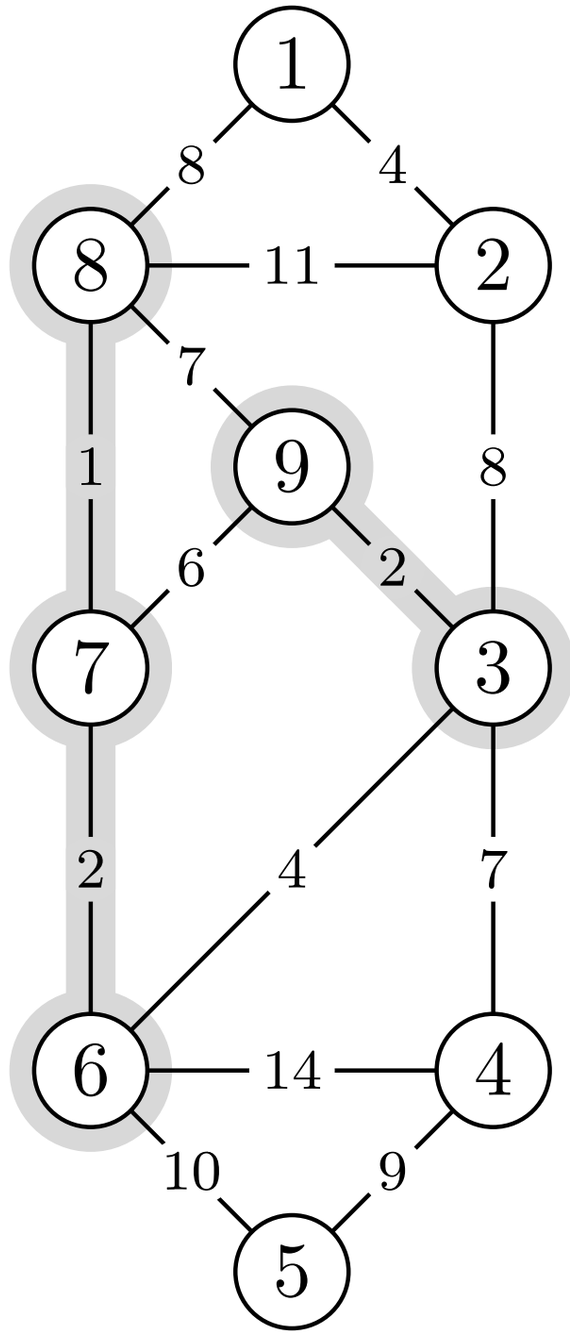
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



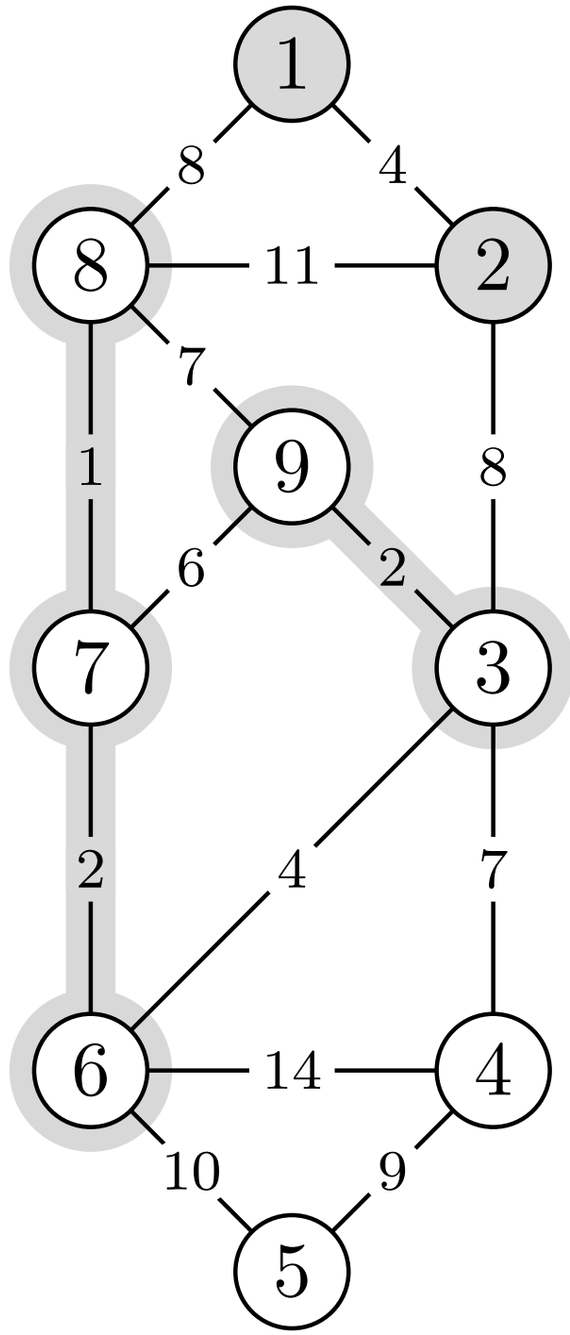
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



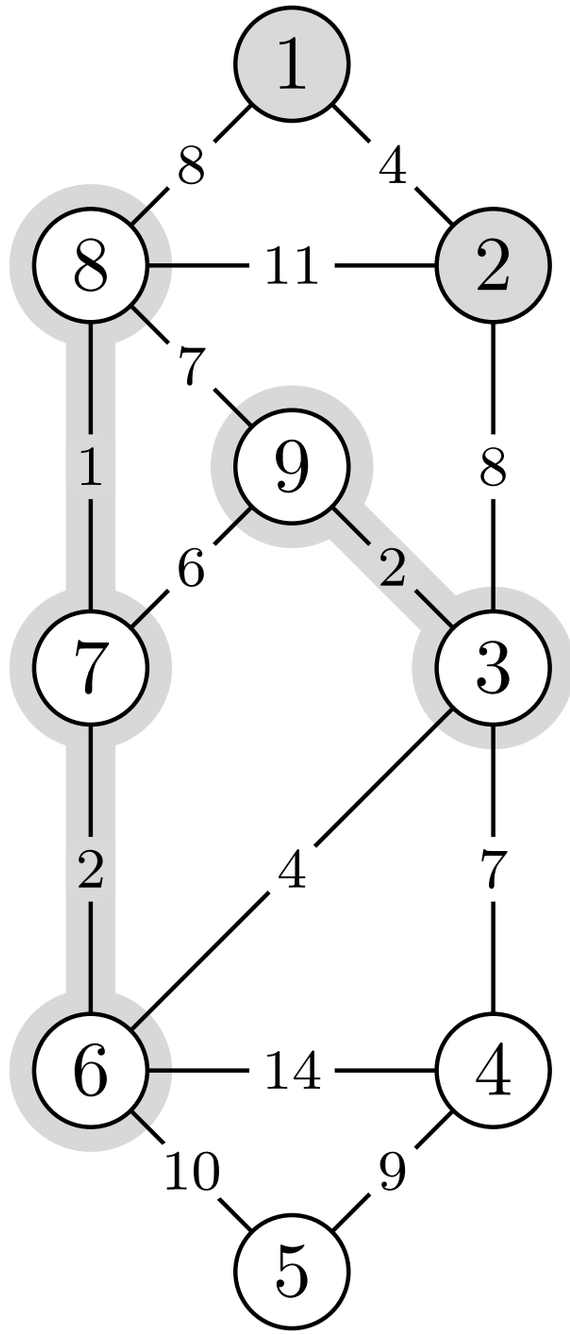
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



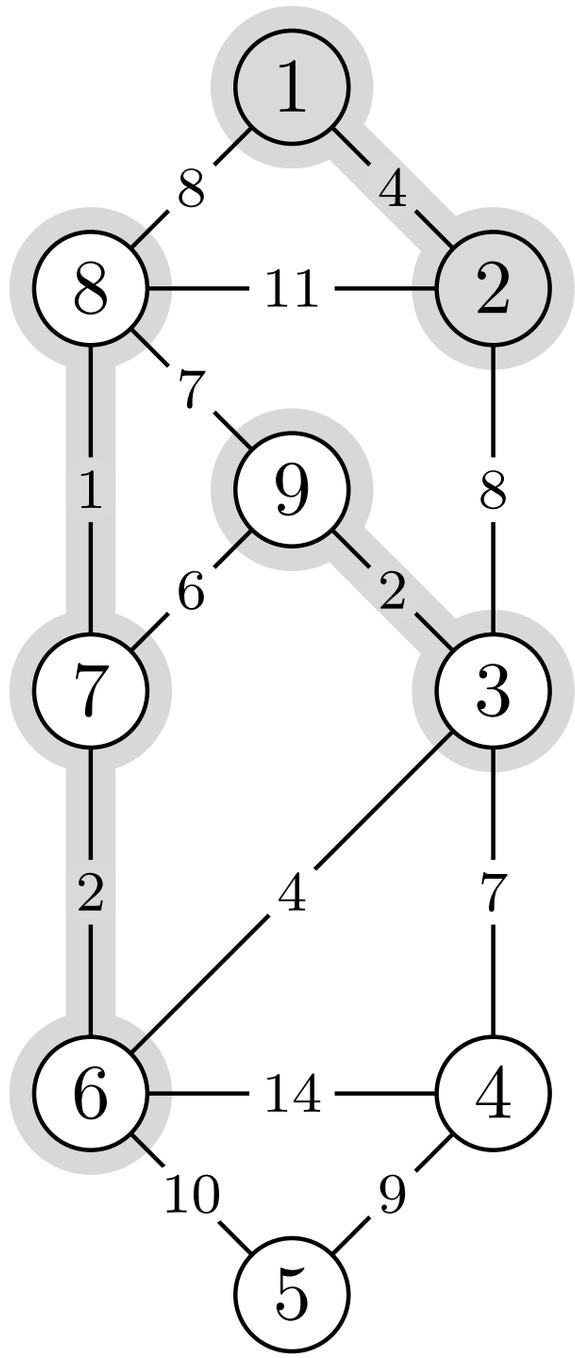
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



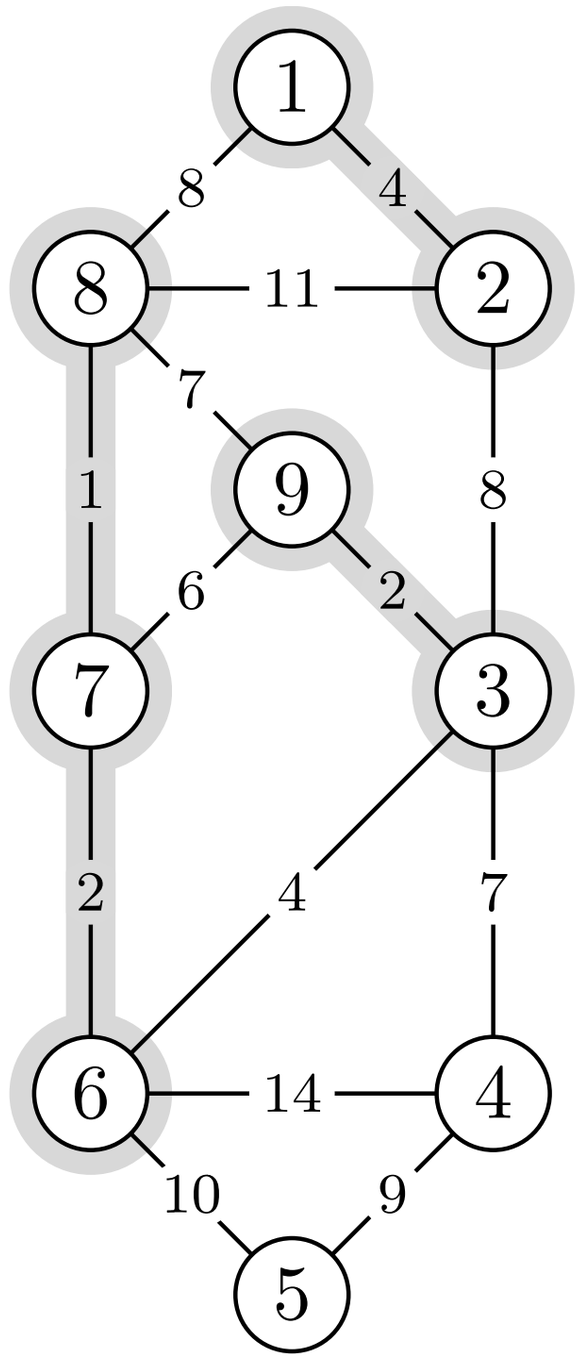
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



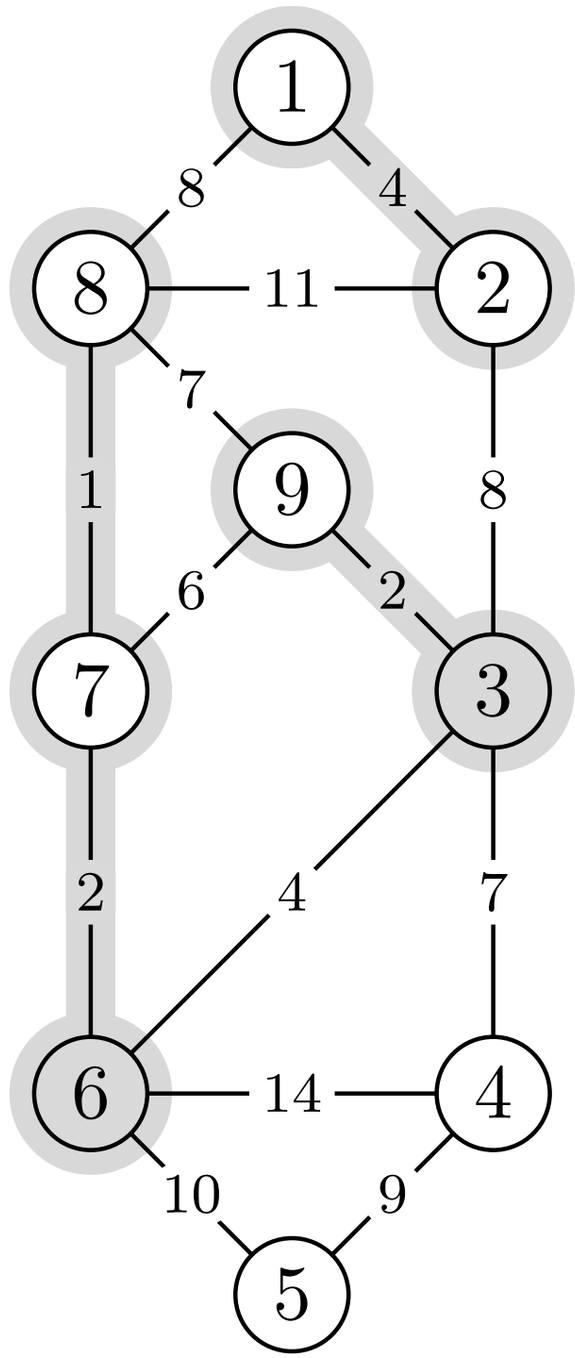
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



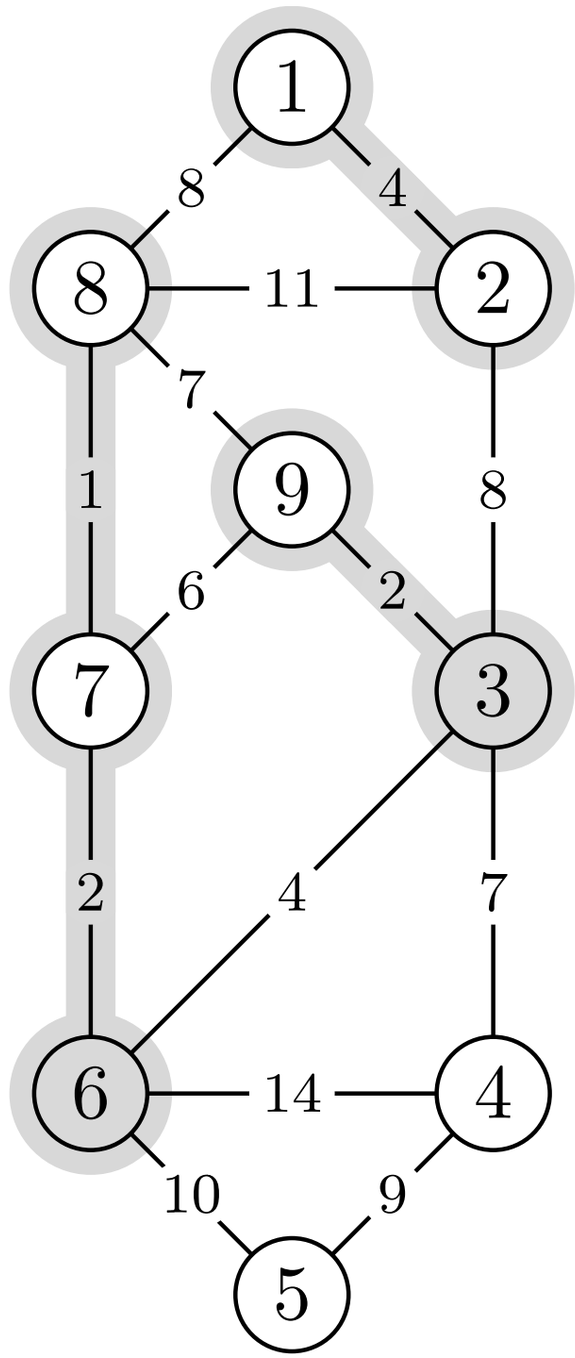
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



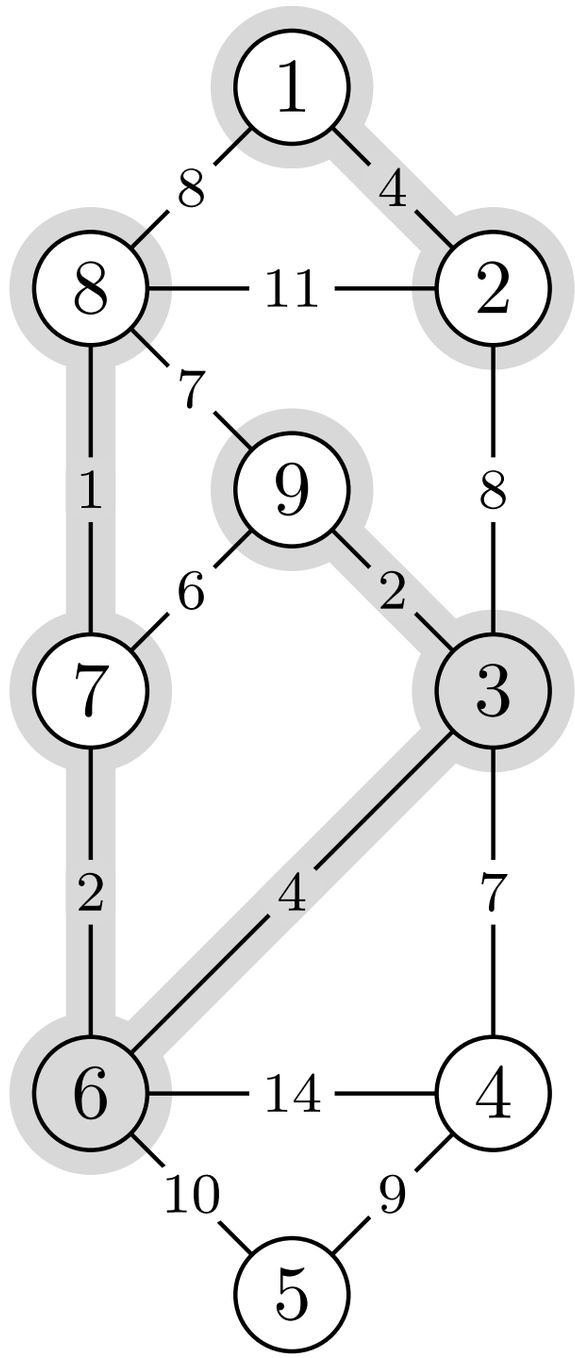
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



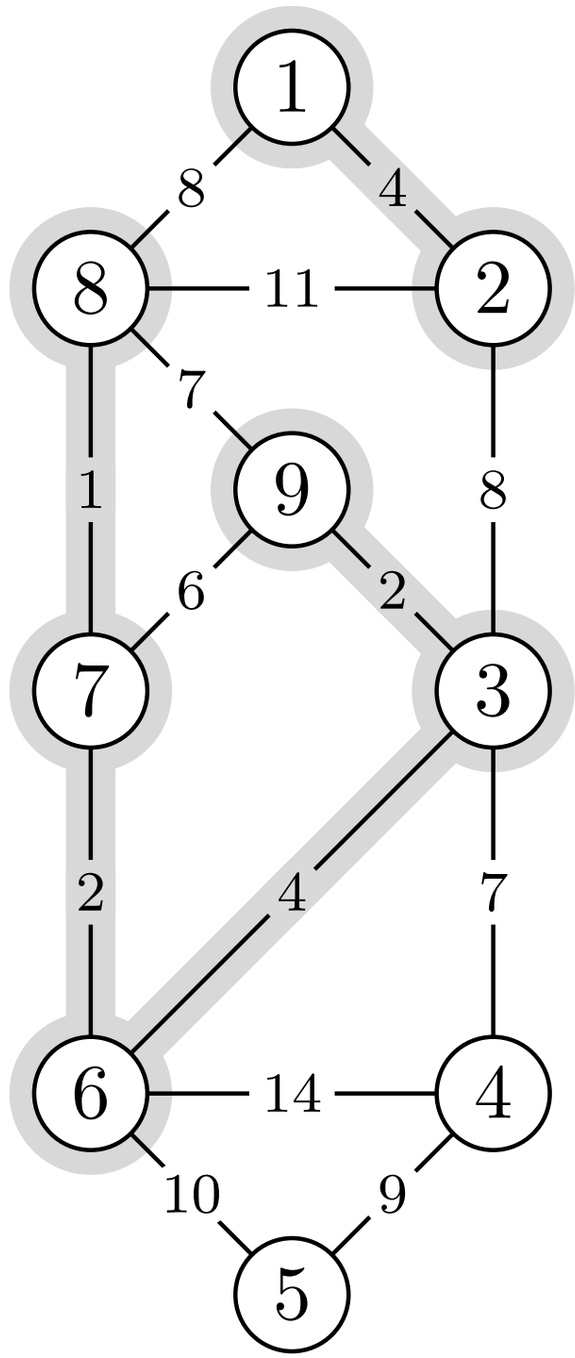
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



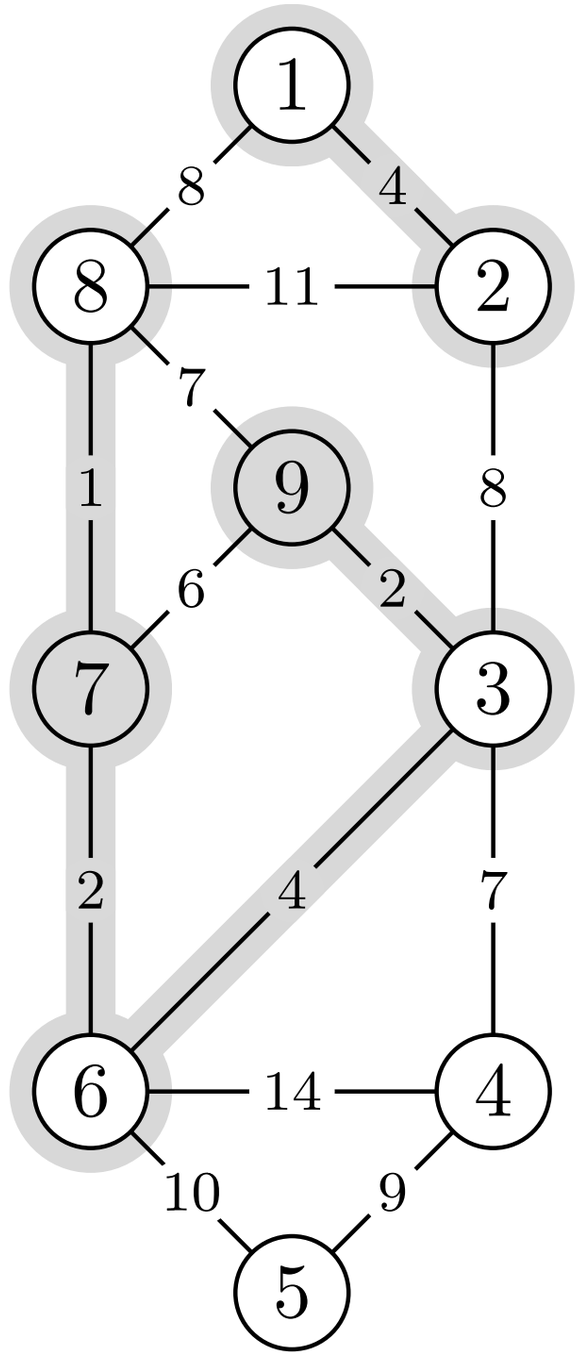
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



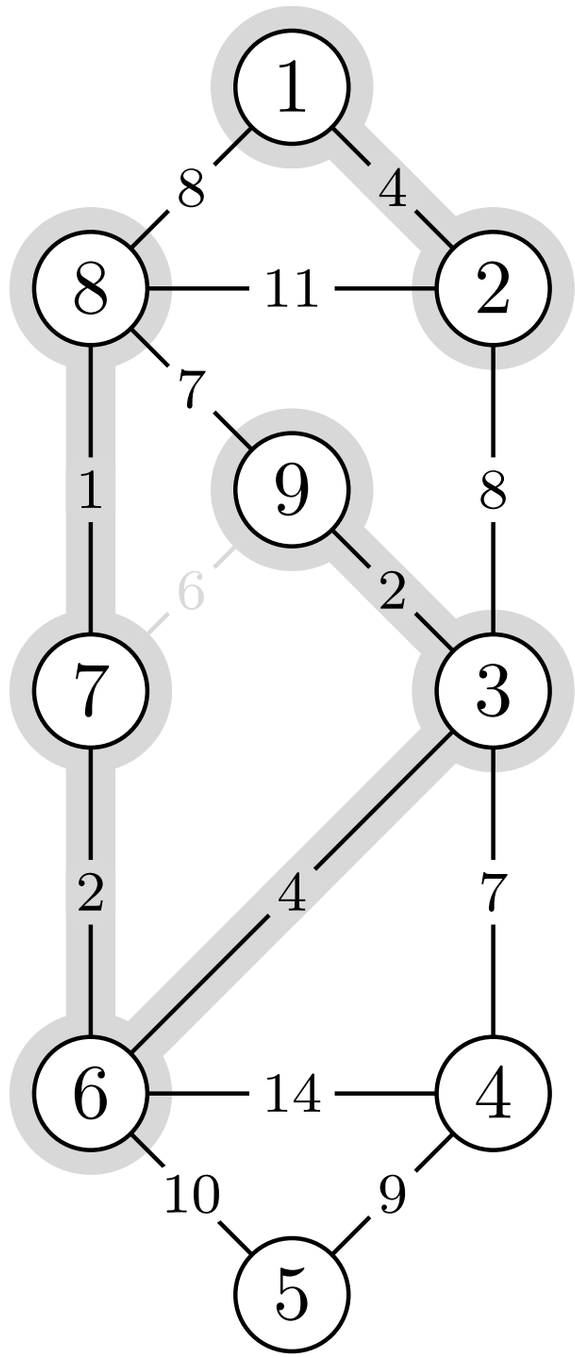
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



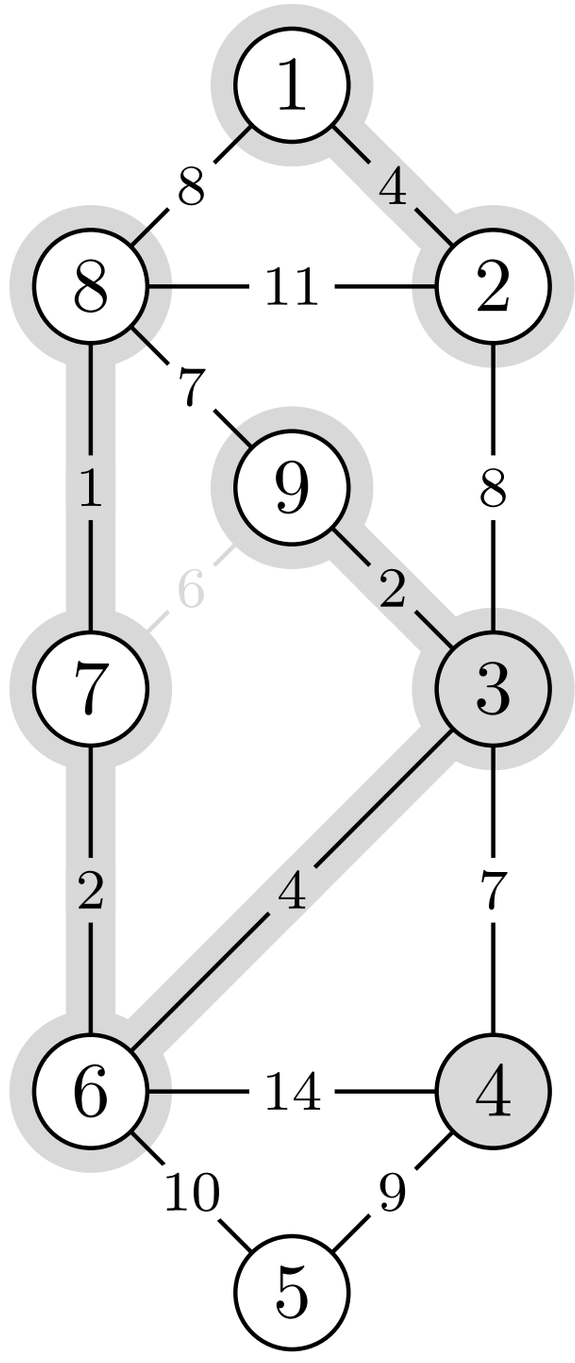
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



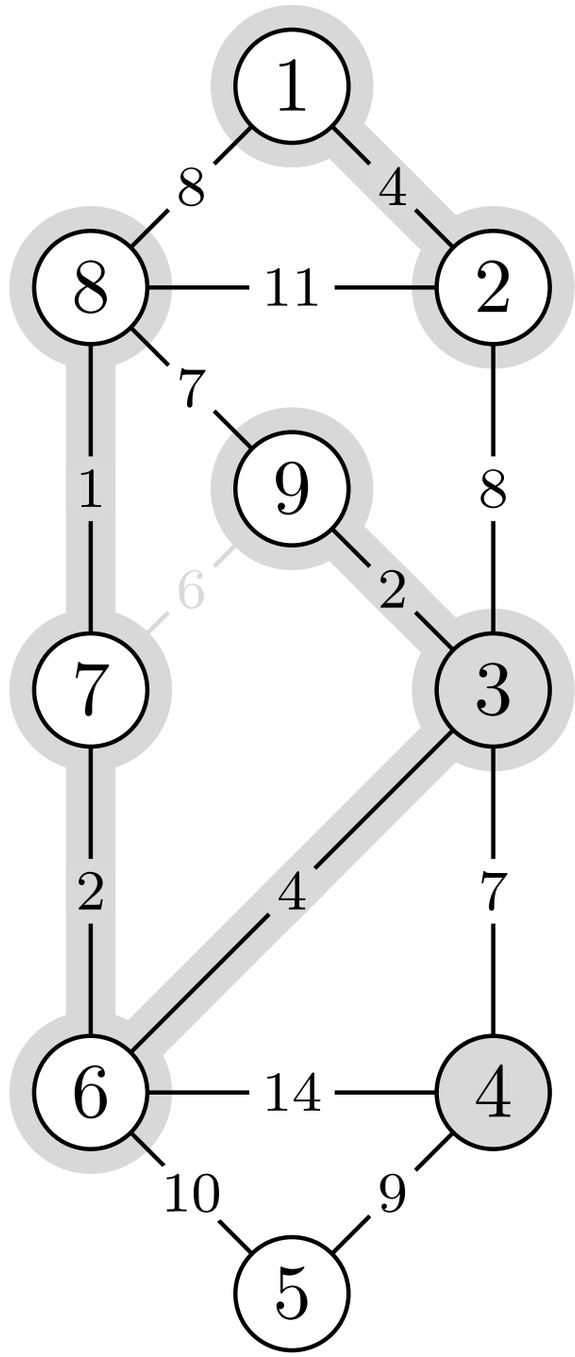
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



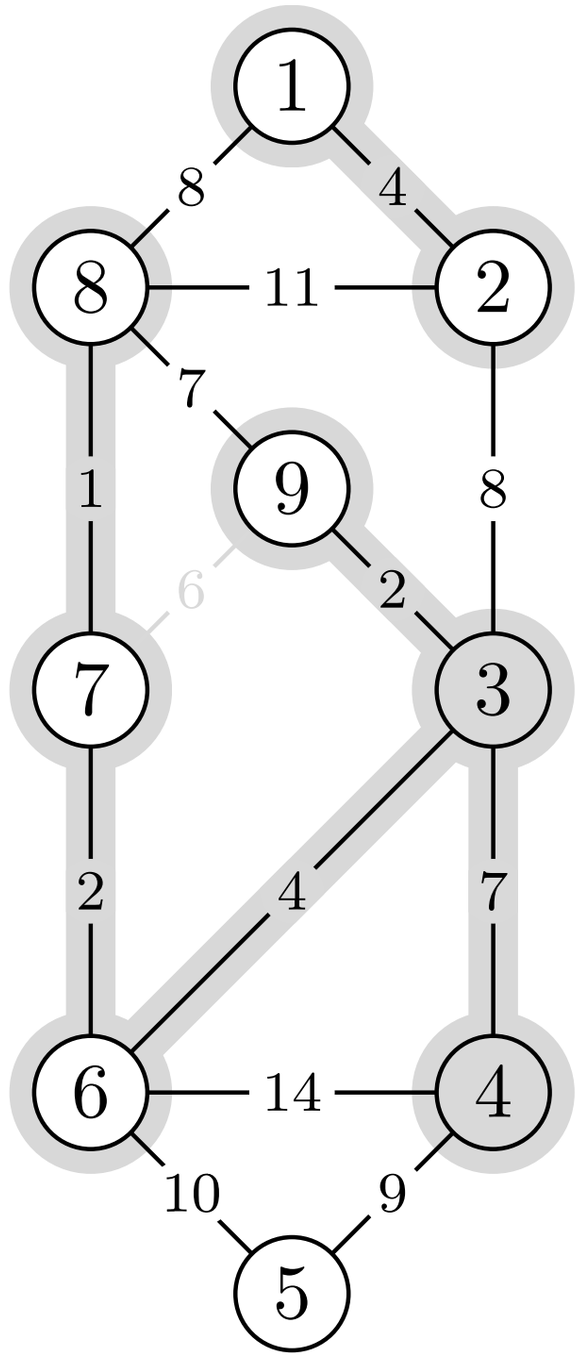
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



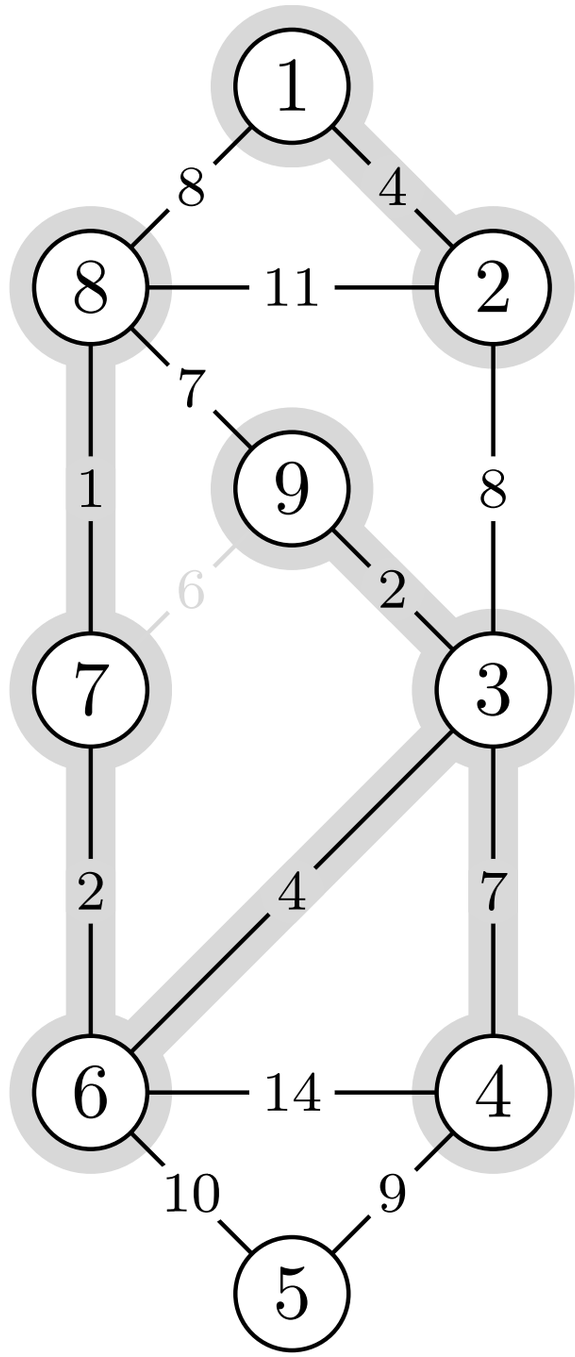
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



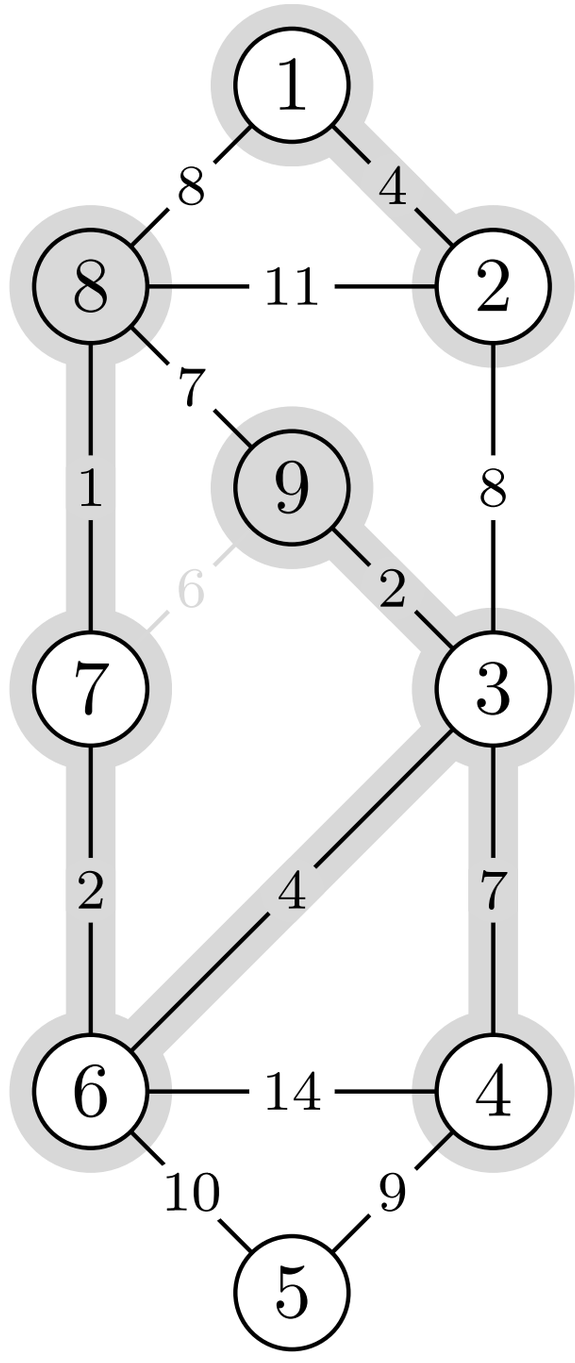
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3        Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6        if Find-Set(u) ≠ Find-Set(v)
7              A = A ∪ {(u, v)}
8              Union(u, v)
9  return A
```



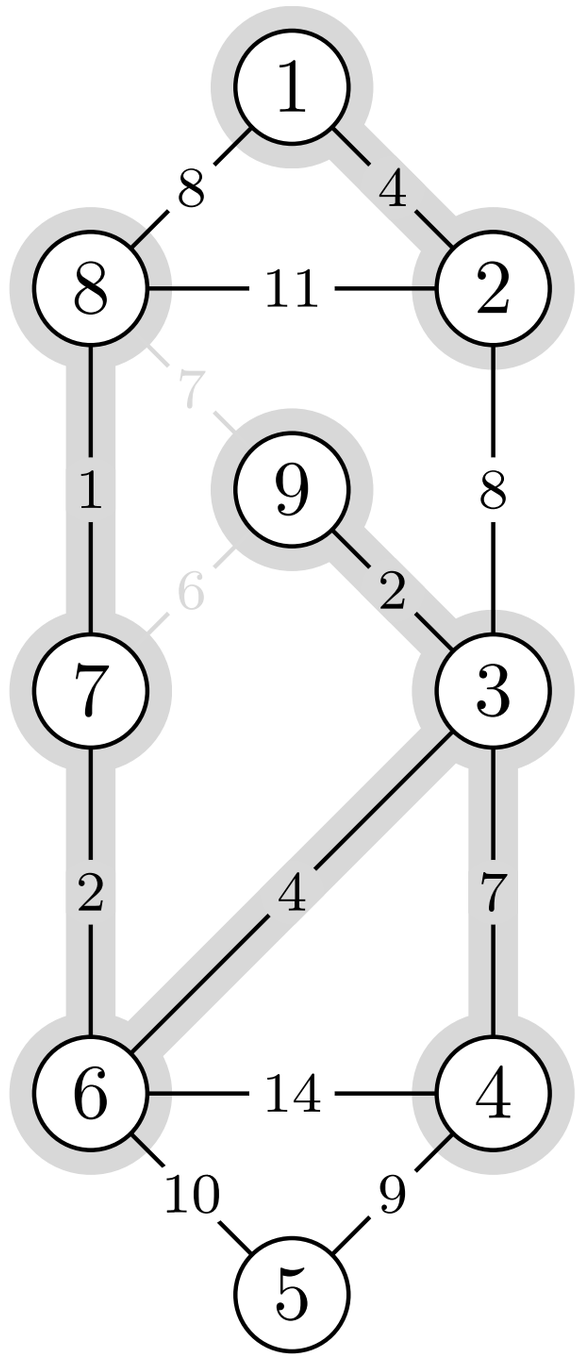
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



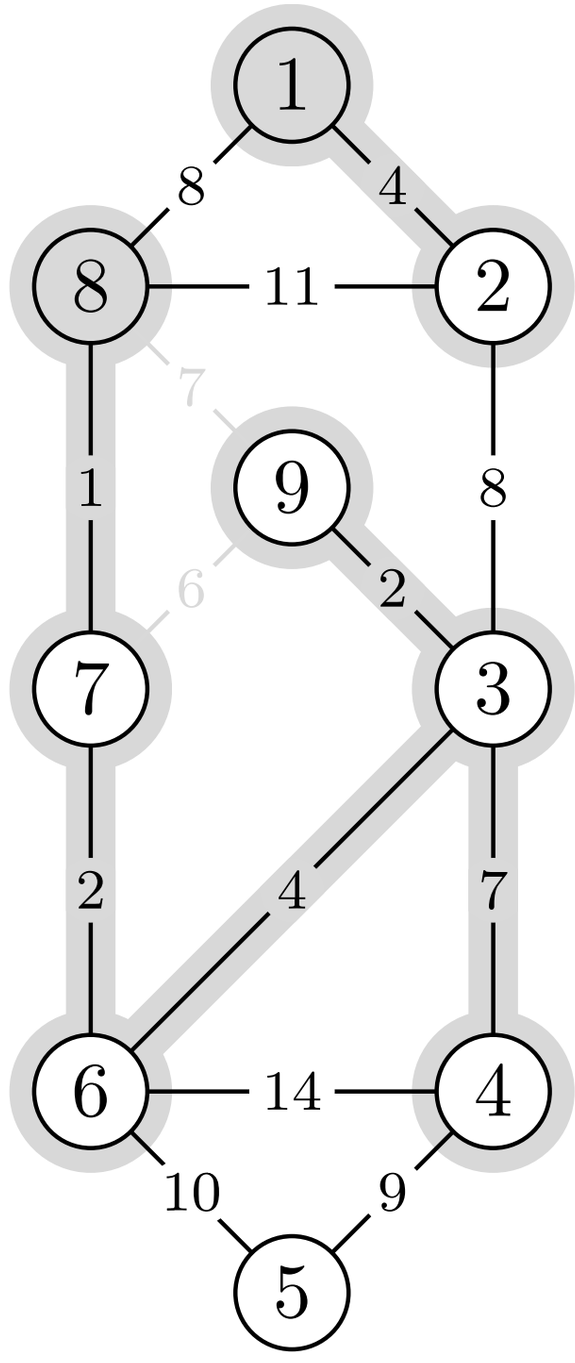
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



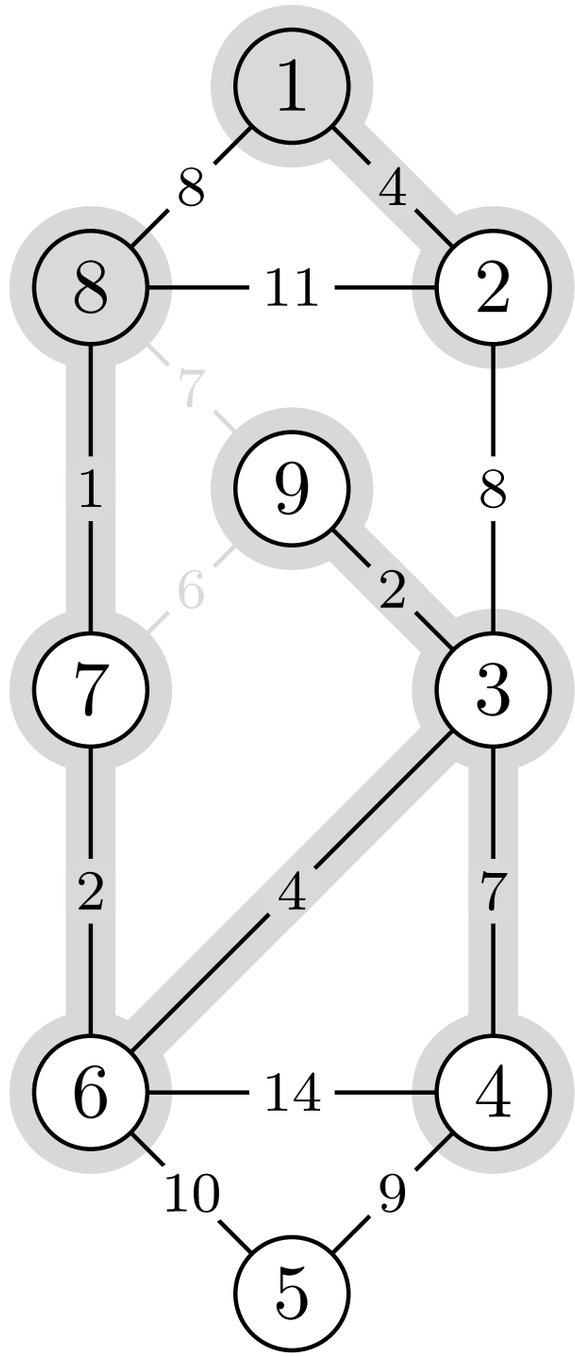
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



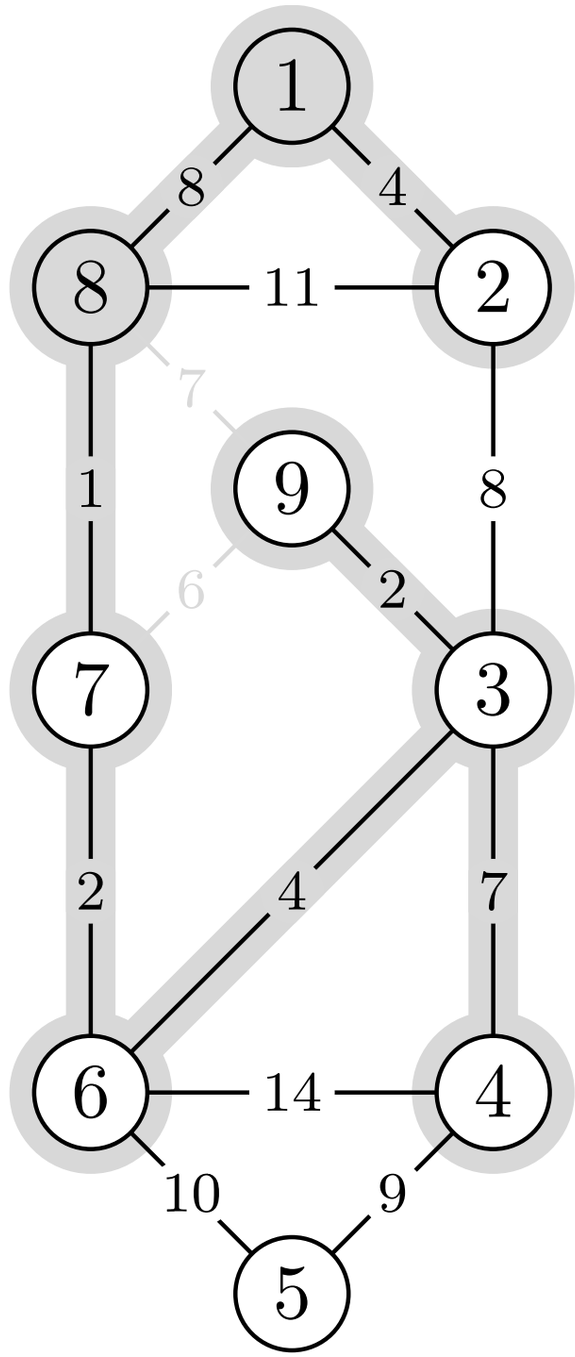
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



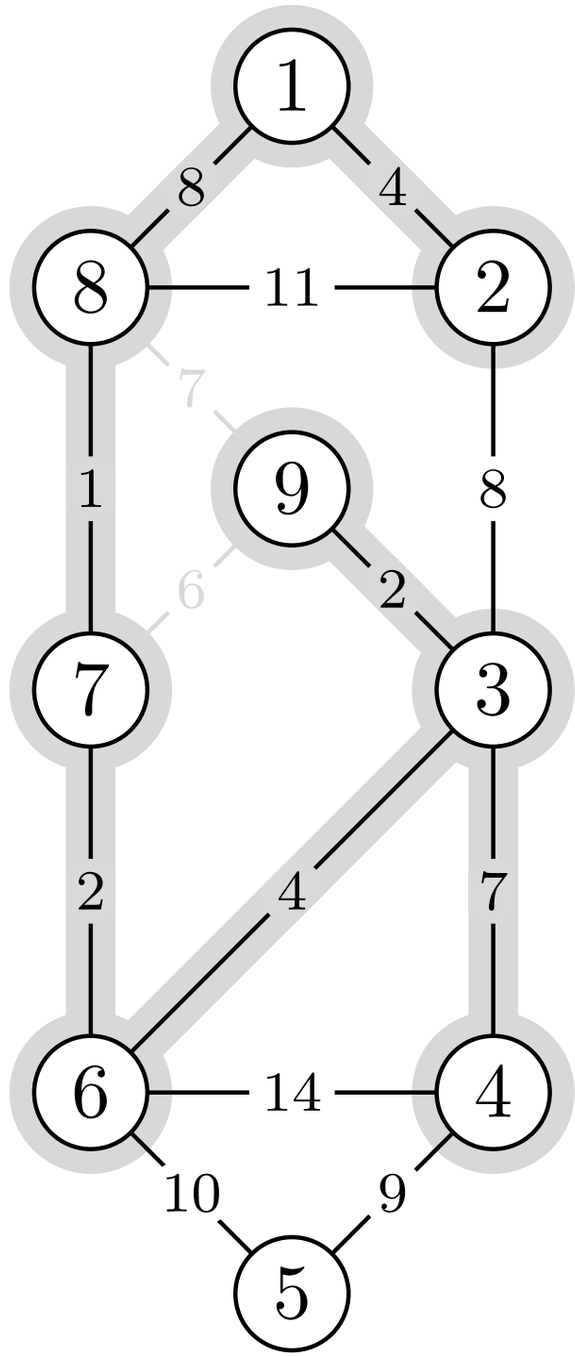
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



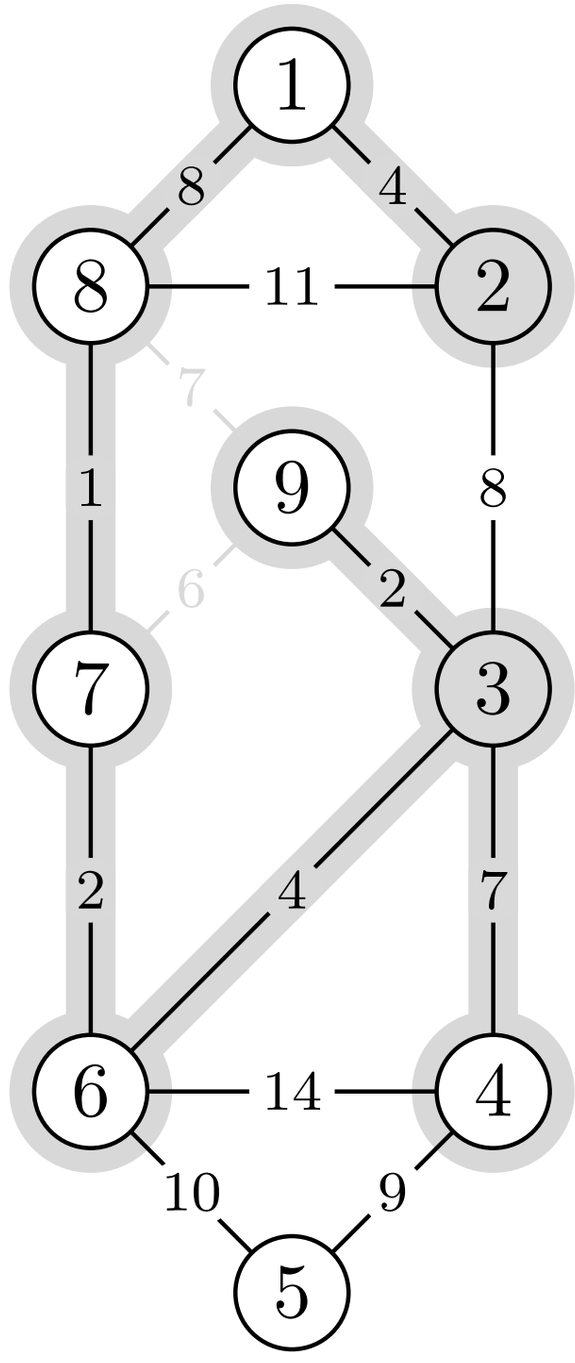
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3        Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6        if Find-Set(u) ≠ Find-Set(v)
7              A = A ∪ {(u, v)}
8              Union(u, v)
9  return A
```



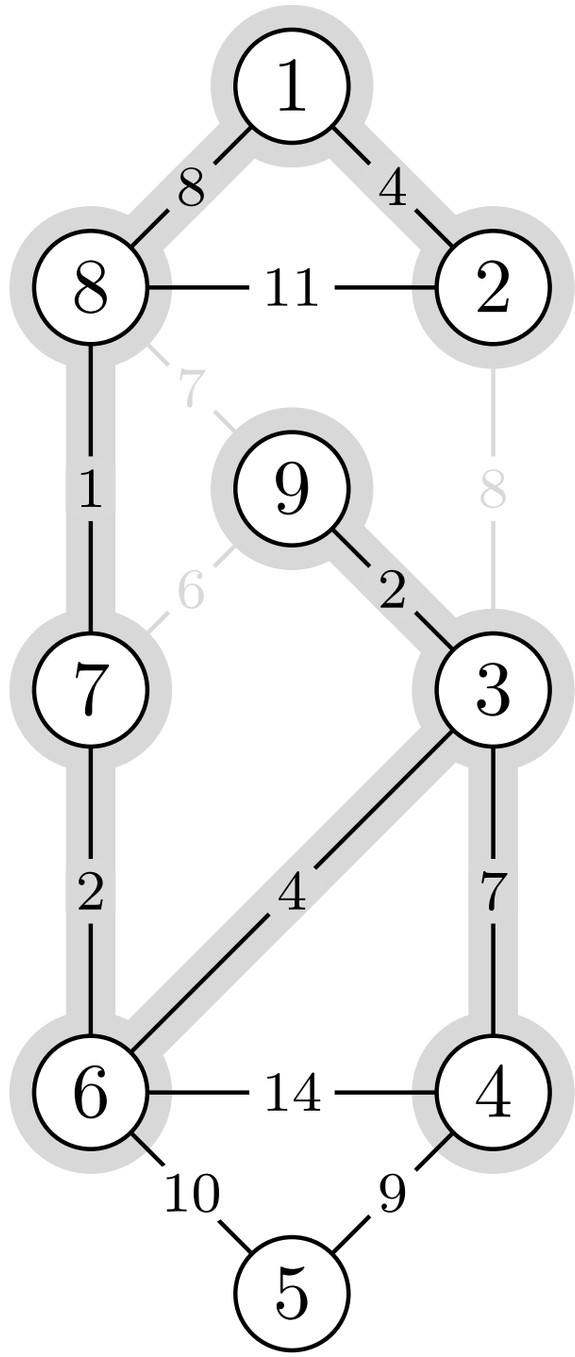
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```



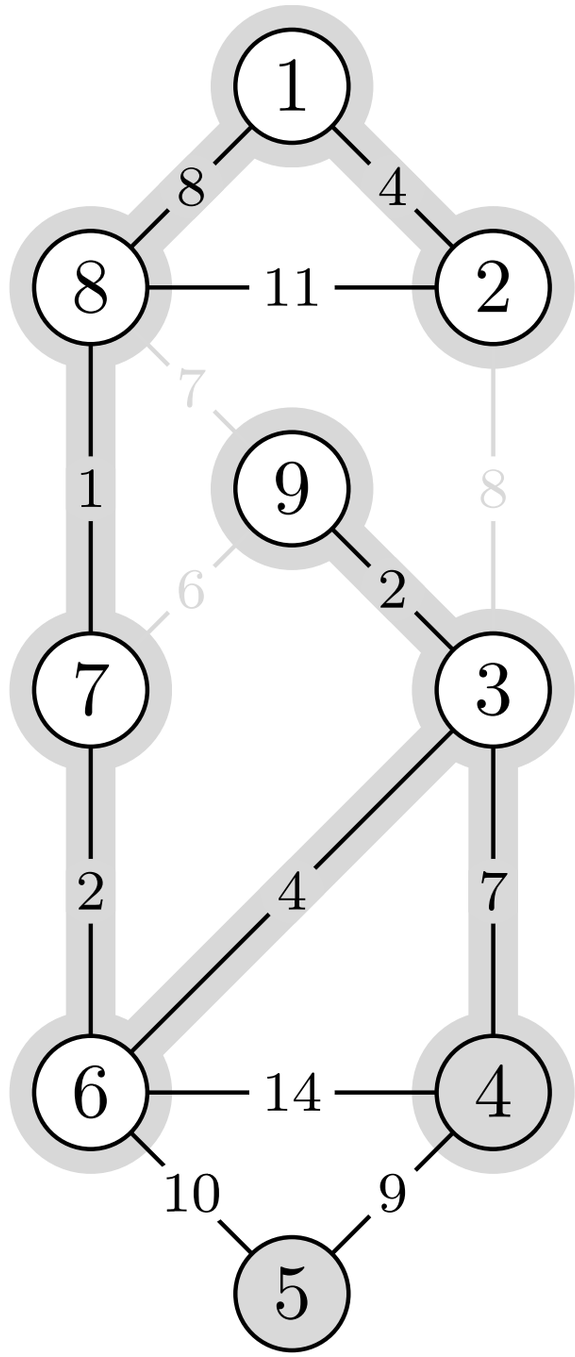
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



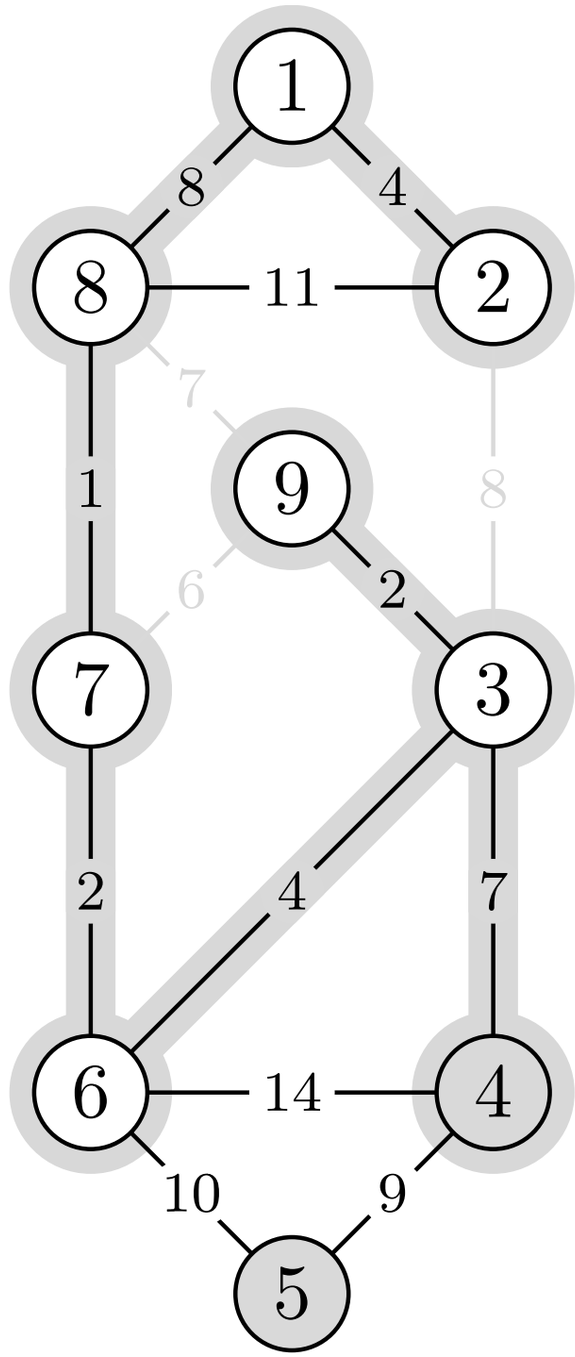
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



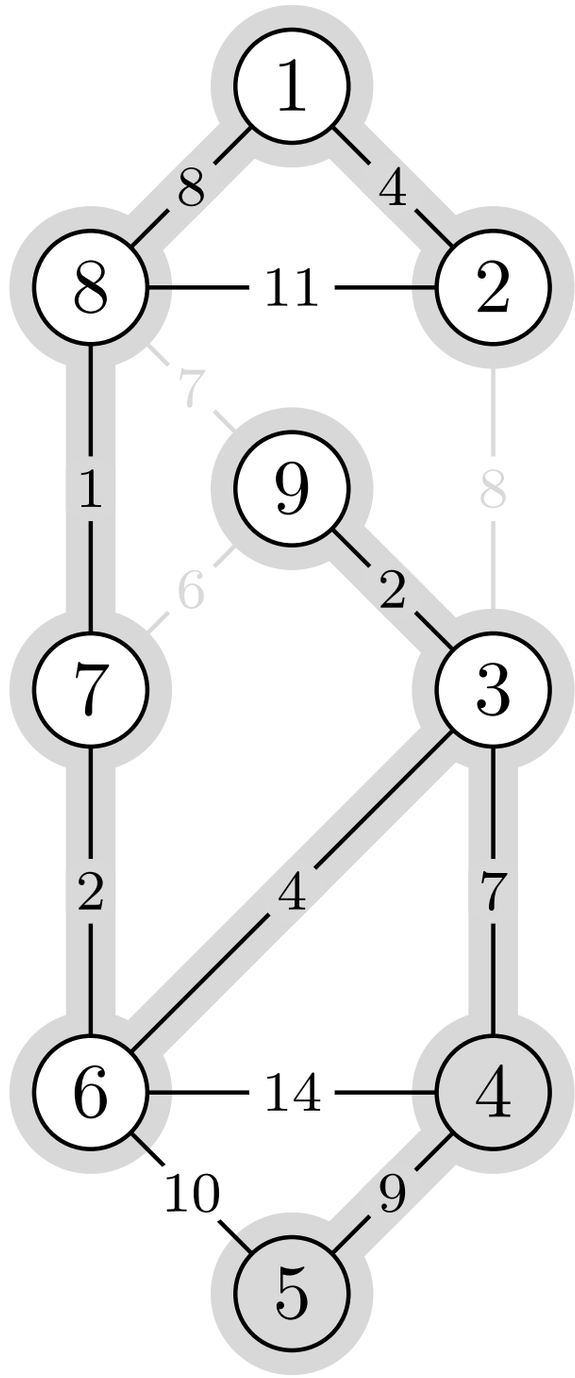
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



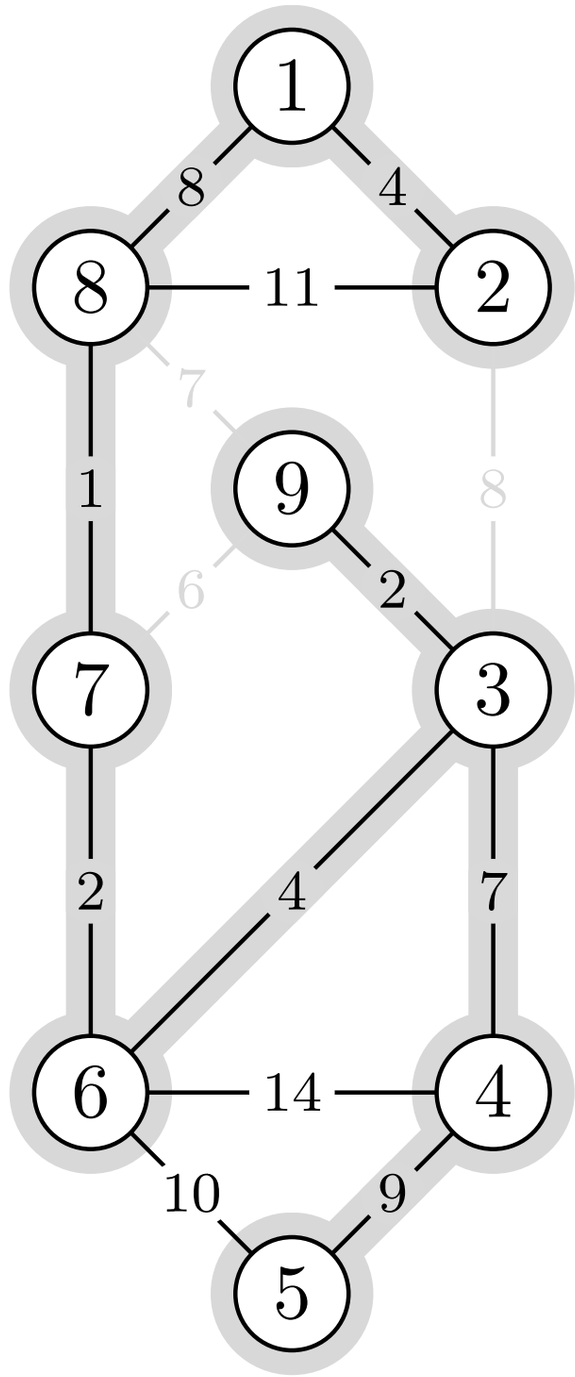
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



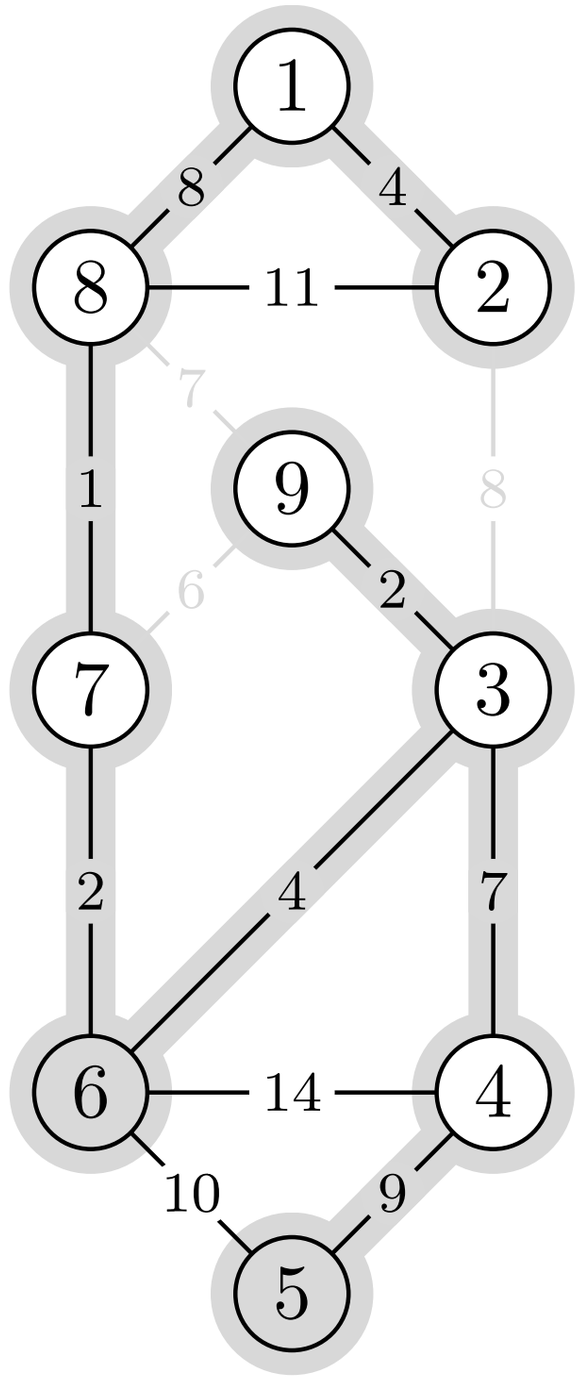
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



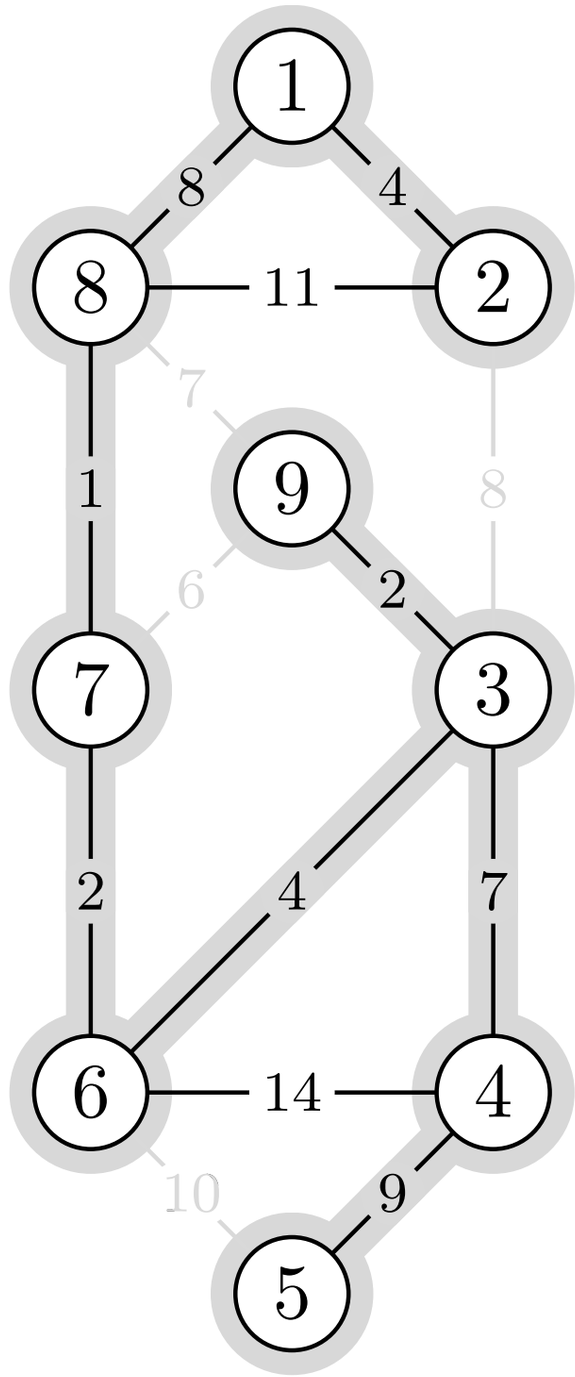
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



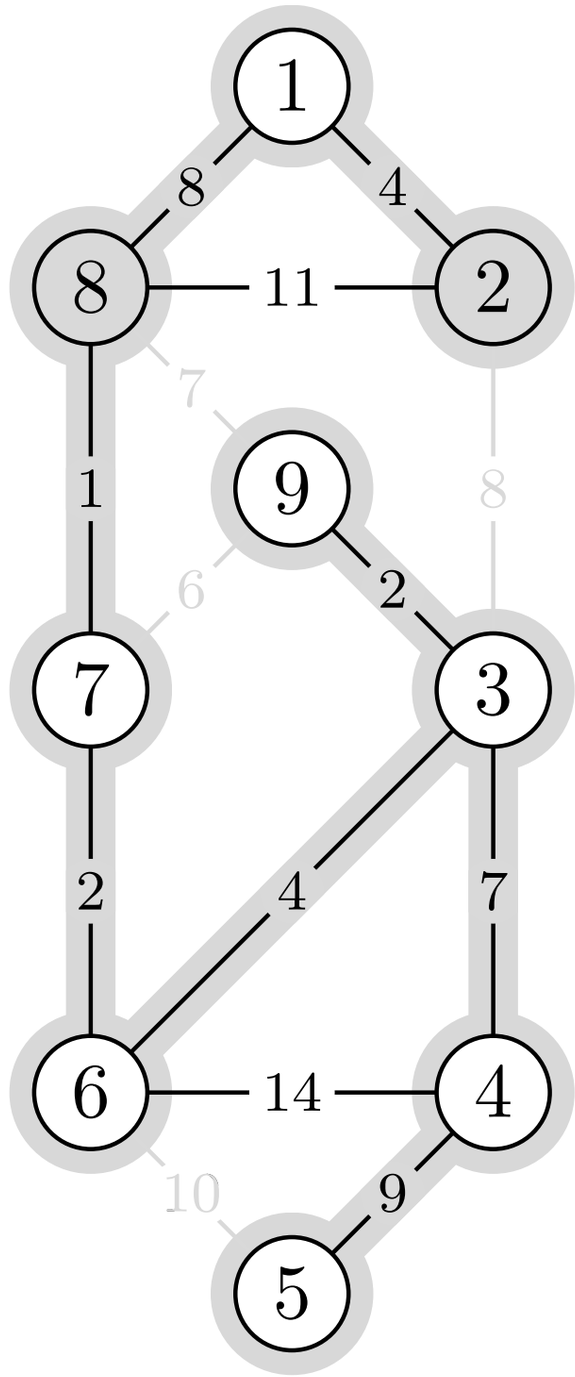
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



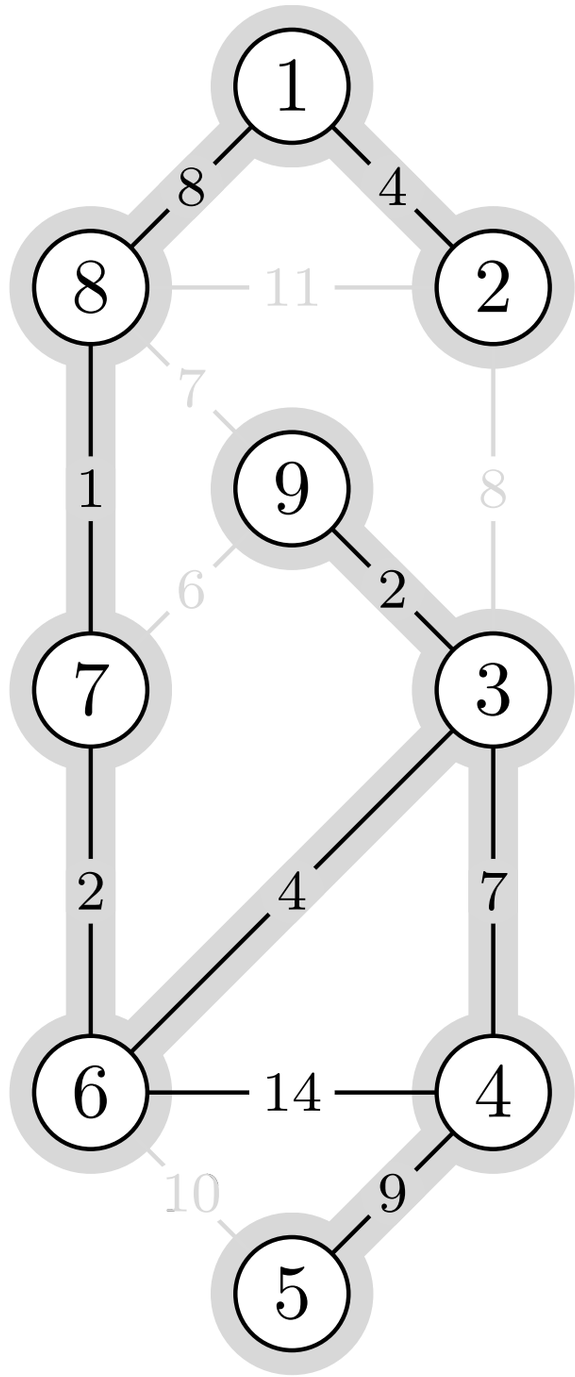
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



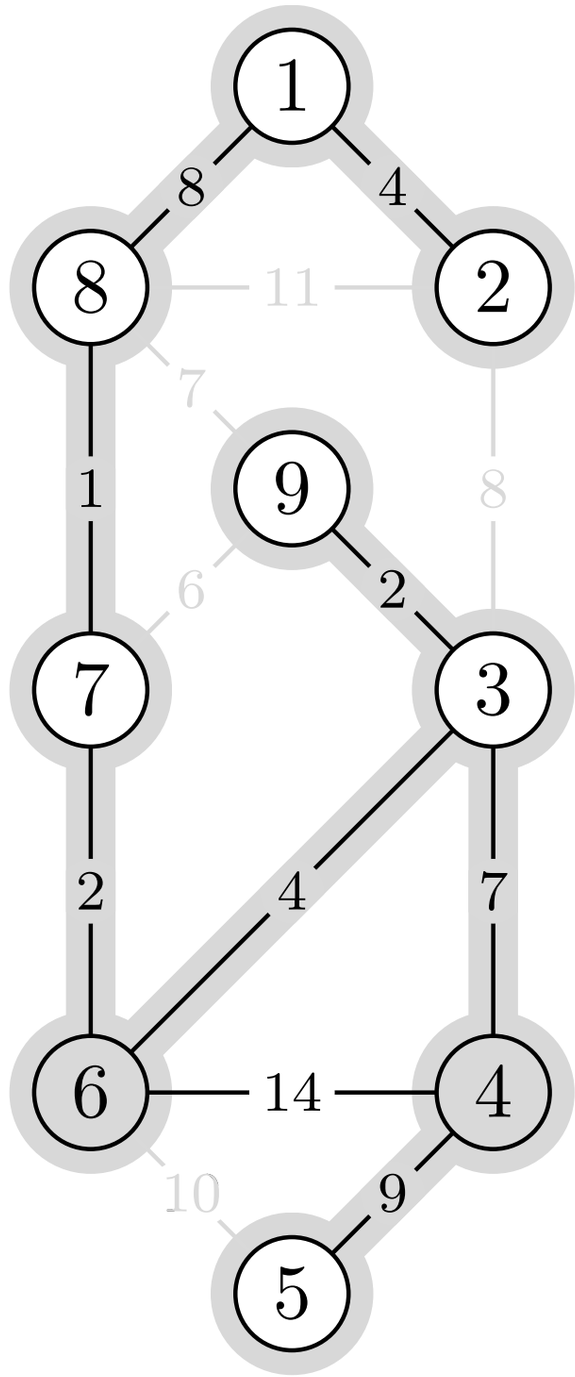
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



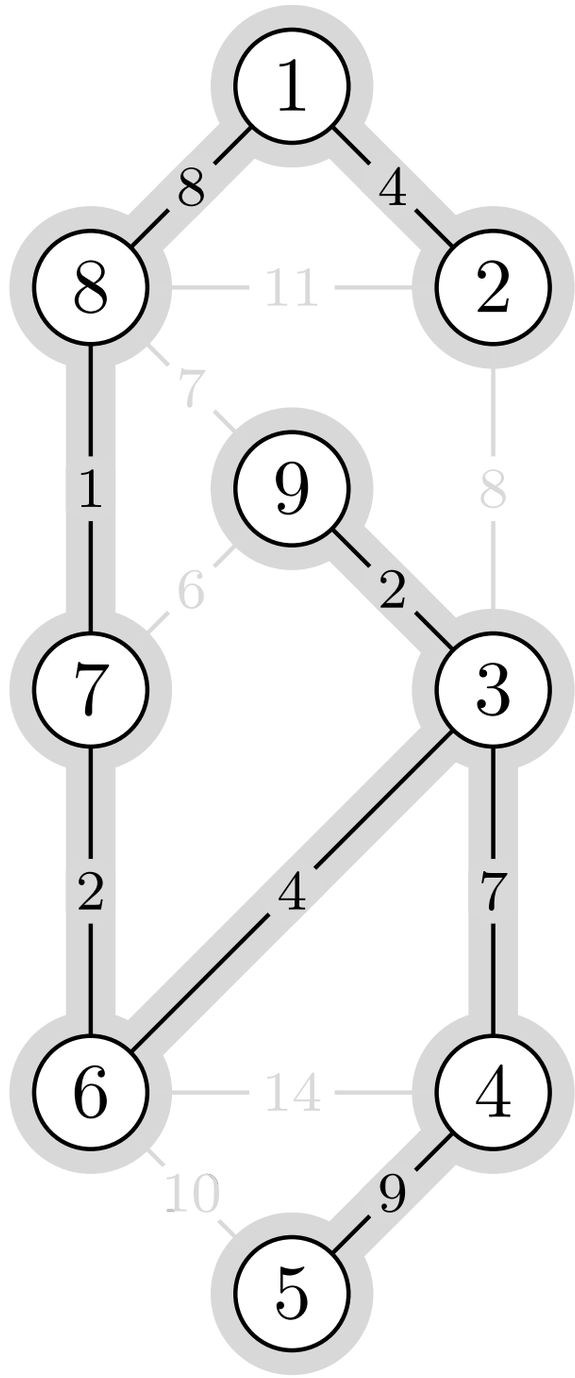
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



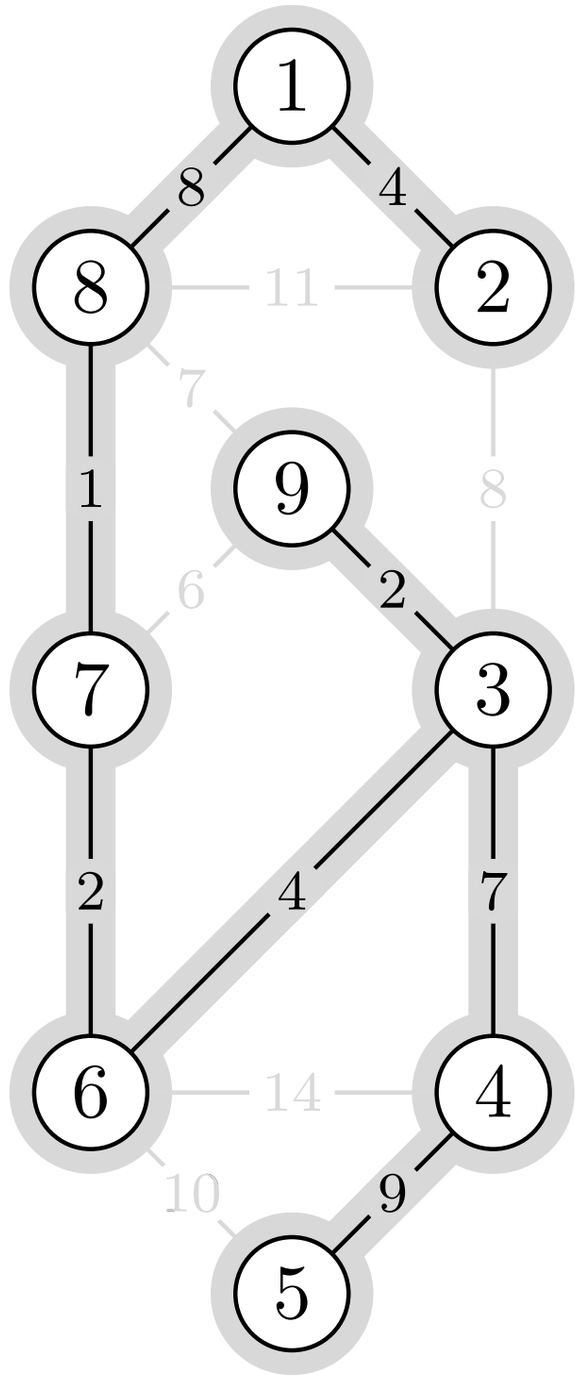
```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3        Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6        if Find-Set(u) ≠ Find-Set(v)
7              A = A ∪ {(u, v)}
8              Union(u, v)
9  return A
```



```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3        Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6        if Find-Set(u) ≠ Find-Set(v)
7              A = A ∪ {(u, v)}
8              Union(u, v)
9  return A
```



```
MST-Kruskal(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       Make-Set(v)
4  sort G.E by w
5  for each edge (u, v) ∈ G.E
6       if Find-Set(u) ≠ Find-Set(v)
7            A = A ∪ {(u, v)}
8            Union(u, v)
9  return A
```



Operasjon	Antall	Kjøretid

Operasjon	Antall	Kjøretid
Make-Set	V	O(1)

Operasjon	Antall	Kjøretid
Make-Set	V	O(1)

Merk: Boka kombinerer Make-Set med Find-Set og Union

Operasjon	Antall	Kjøretid
Make-Set	V	O(1)
Sortering	1	O(E lg E)

Operasjon	Antall	Kjøretid
Make-Set	V	$O(1)$
Sortering	1	$O(E \lg E)$
Find-Set	$O(E)$	$O(\alpha(V))$
Union	$O(E)$	$O(\alpha(V))$

Operasjon	Antall	Kjøretid
Make-Set	V	$O(1)$
Sortering	1	$O(E \lg E)$
Find-Set	O(E)	$O(\alpha(V))$
Union	O(E)	$O(\alpha(V))$

$$\alpha(n) = O(\lg n)$$

Operasjon	Antall	Kjøretid
Make-Set	V	$O(1)$
Sortering	1	$O(E \lg E)$
Find-Set	O(E)	$O(\alpha(V))$
Union	O(E)	$O(\alpha(V))$

$\alpha(n) = O(\lg n)$; i praksis har vi $\alpha(n) \leq 4$

Operasjon	Antall	Kjøretid
Make-Set	V	O(1)
Sortering	1	O(E lg E)
Find-Set	O(E)	O(lg V)
Union	O(E)	O(lg V)

$\alpha(n) = \mathrm{O}(\lg n)$; i praksis har vi $\alpha(n) \leq 4$

Operasjon	Antall	Kjøretid
Make-Set	V	O(1)
Sortering	1	O(E lg E)
Find-Set	O(E)	O(lg V)
Union	O(E)	O(lg V)

$|E| < |V|^2$

Operasjon	Antall	Kjøretid
Make-Set	V	$O(1)$
Sortering	1	$O(E \lg E)$
Find-Set	$O(E)$	$O(\lg V)$
Union	$O(E)$	$O(\lg V)$

$$|E| < |V|^2 \implies \lg|E| < 2\lg|V|$$

Operasjon	Antall	Kjøretid
Make-Set	V	$O(1)$
Sortering	1	$O(E \lg E)$
Find-Set	$O(E)$	$O(\lg V)$
Union	$O(E)$	$O(\lg V)$

$$|E| < |V|^2 \implies \lg|E| < 2\lg|V| \implies \lg E = O(\lg V)$$

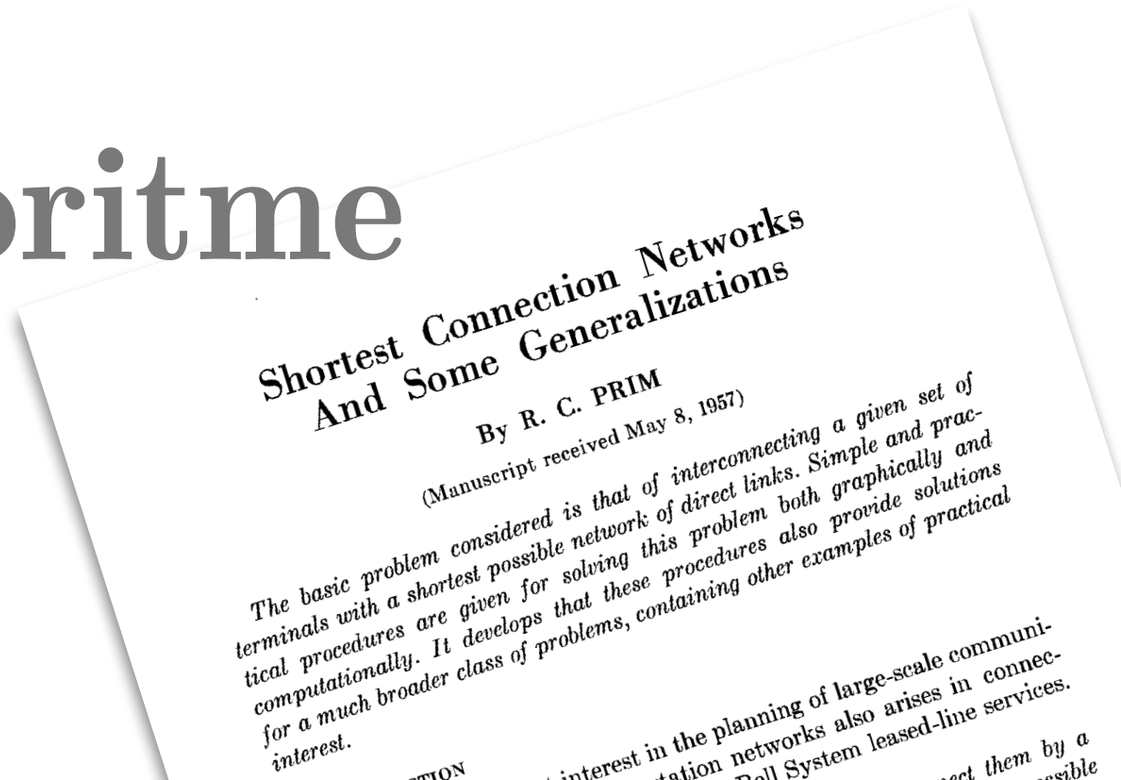
Operasjon	Antall	Kjøretid
Make-Set	V	$O(1)$
Sortering	1	$O(E \lg E)$
Find-Set	$O(E)$	$O(\lg V)$
Union	$O(E)$	$O(\lg V)$

Totalt: $O(E \lg V)$

$$|E| < |V|^2 \implies \lg|E| < 2\lg|V| \implies \lg E = O(\lg V)$$

4:4

Prims algoritme



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.

- Kan implementeres vha. traversering
- Der BFS bruker FIFO og DFS bruker LIFO, så bruker Prim en min-prioritets-kø
- Prioriteten er vekten på den letteste kanten mellom noden og treet
- For enkelhets skyld: Legg alle noder inn fra starten, med uendelig dårlig prioritet

MST-Prim(G, w, r)

G graf
w vekting
r startnode

Start i r. Gjenta: Inkludér noden nærmest treet

```
MST-Prim(G, w, r)
 1  for each u ∈ G.V
```

G graf
w vekting
r startnode

Initialisering

```
MST-Prim(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
```

G graf
w vekting
r startnode

Vekt til letteste kant til treet

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
```

G graf
w vekting
r startnode

Noden den letteste kanten kommer fra; utgjør MST til slutt

MST-PRIM(G, w, r)

```
1  for each u ∈ G.V
2      u.key = ∞
3      u.π = NIL
4  r.key = 0
```

G graf
w vekting
r startnode

r er startnoden vår

mlh@ntnu.no

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
 4  r.key = 0
 5  Q = G.V
```

G graf
w vekting
r startnode
Q pri-kø

Vi kunne ha traversert ... men trenger ikke

```
MST-Prim(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
```

G graf
w vekting
r startnode
Q pri-kø

Som ved traversering: Står det noe på huskelista vår?

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7        u = EXTRACT-MIN(Q)
```

G graf
w vekting
r startnode
Q pri-kø

Noden nærmest treet

```
MST-Prim(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7        u = Extract-Min(Q)
 8        for each v ∈ G.Adj[u]
```

G graf
w vekting
r startnode
Q pri-kø

«Oppdag» – eller i hvert fall oppdatér – naboene

```
MST-Prim(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7        u = Extract-Min(Q)
 8        for each v ∈ G.Adj[u]
 9              if v ∈ Q and w(u, v) < v.key
```

G graf
w vekting
r startnode
Q pri-kø

Er v fortsatt ubrukt? Fant vi en bedre kant fra treet til v?

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7        u = EXTRACT-MIN(Q)
 8        for each v ∈ G.Adj[u]
 9              if v ∈ Q and w(u, v) < v.key
10                    v.π = u
```

G graf
w vekting
r startnode
Q pri-kø

Da bruker vi denne nye kanten i stedet . . .

```
MST-Prim(G, w, r)
 1  for each u ∈ G.V
 2        u.key = ∞
 3        u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7        u = Extract-Min(Q)
 8        for each v ∈ G.Adj[u]
 9              if v ∈ Q and w(u, v) < v.key
10                    v.π = u
11                    v.key = w(u, v)
```

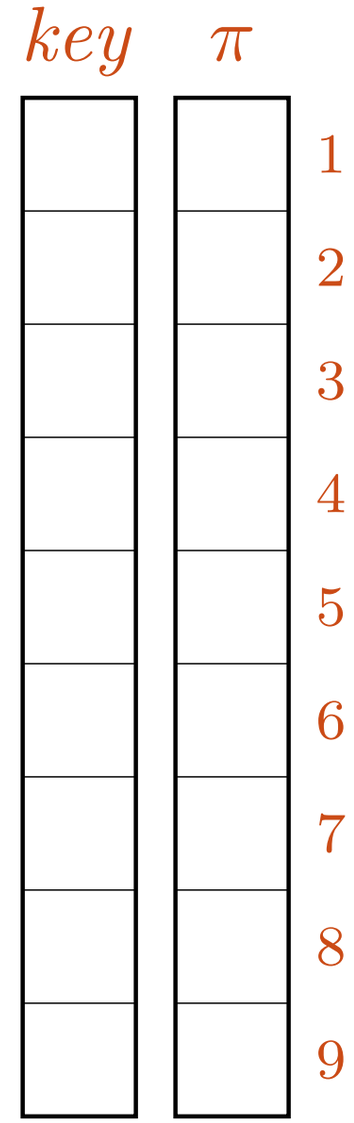
G graf
w vekting
r startnode
Q pri-kø

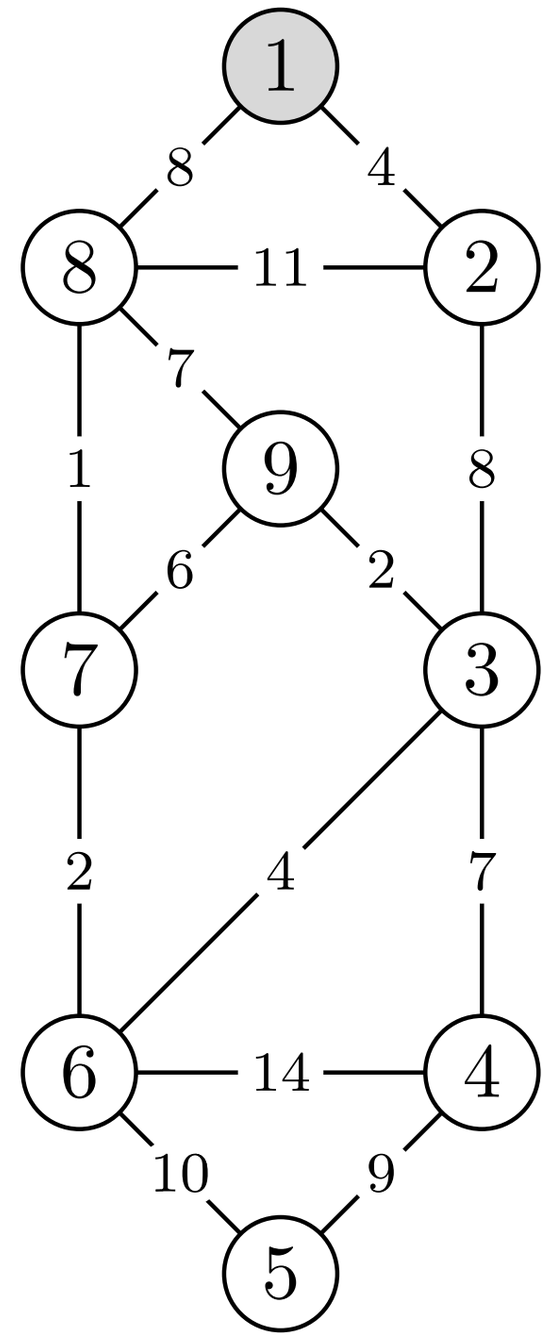
. . . og oppdaterer prioriteten til v i Q

- I det følgende: Farging som for BFS
- Kanter mellom svarte noder er endelige
- Beste kanter for grå noder også uthevet
- Boka uthever bare kantene i spenntreet

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

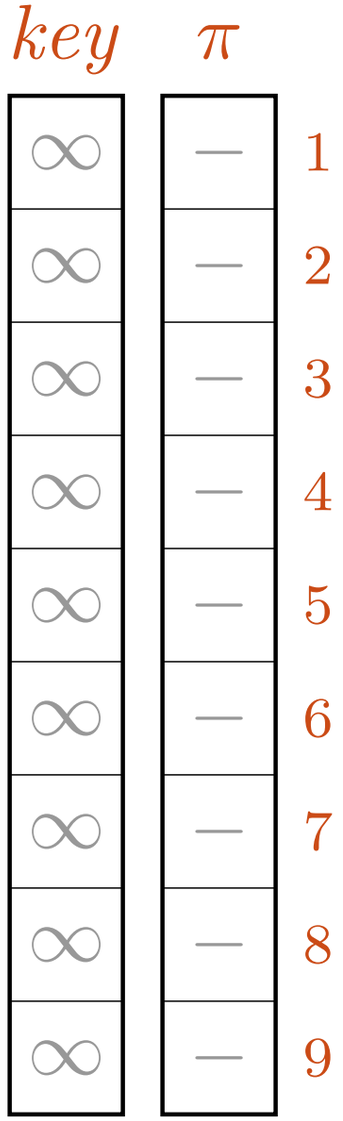
$u, v = -, -$

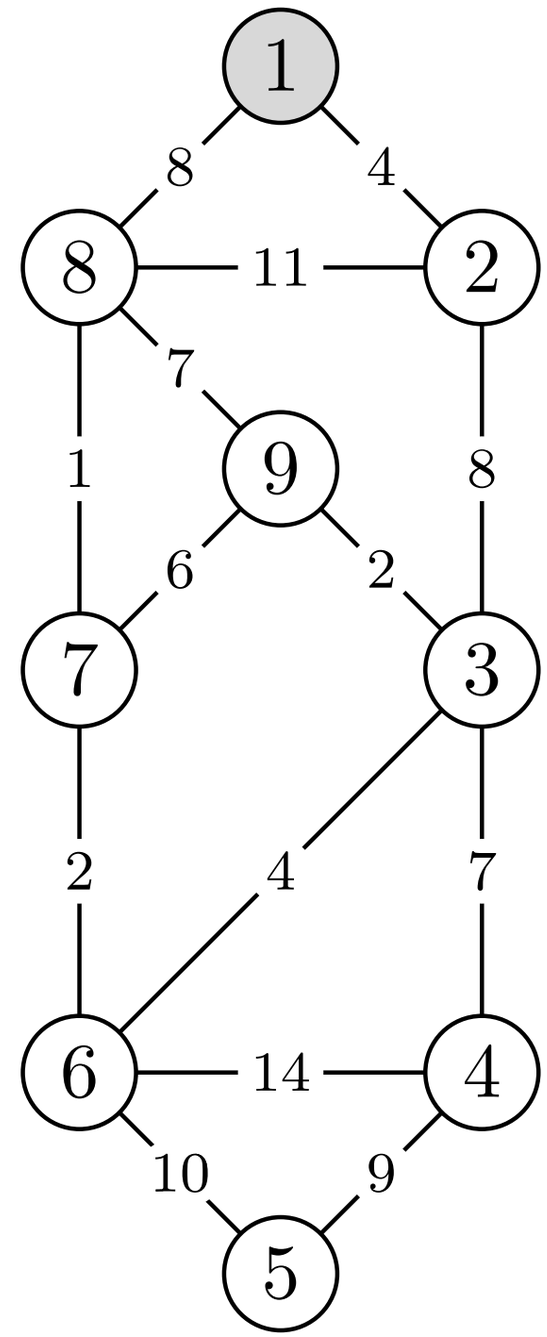




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

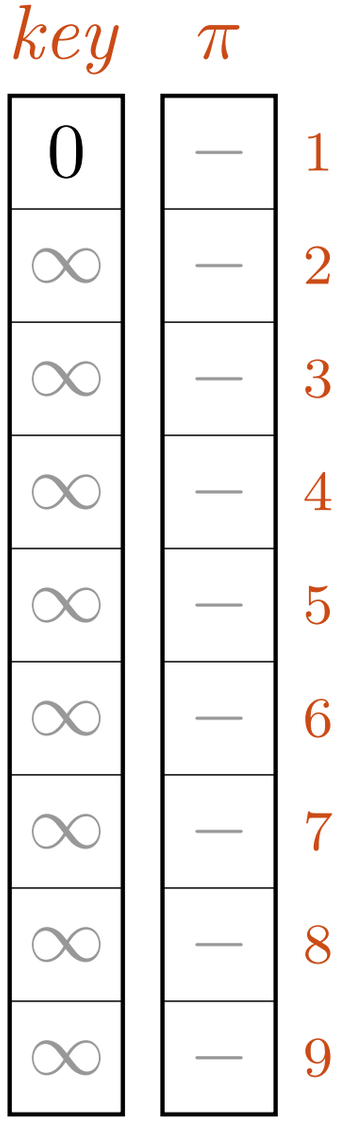
$u, v = -, -$

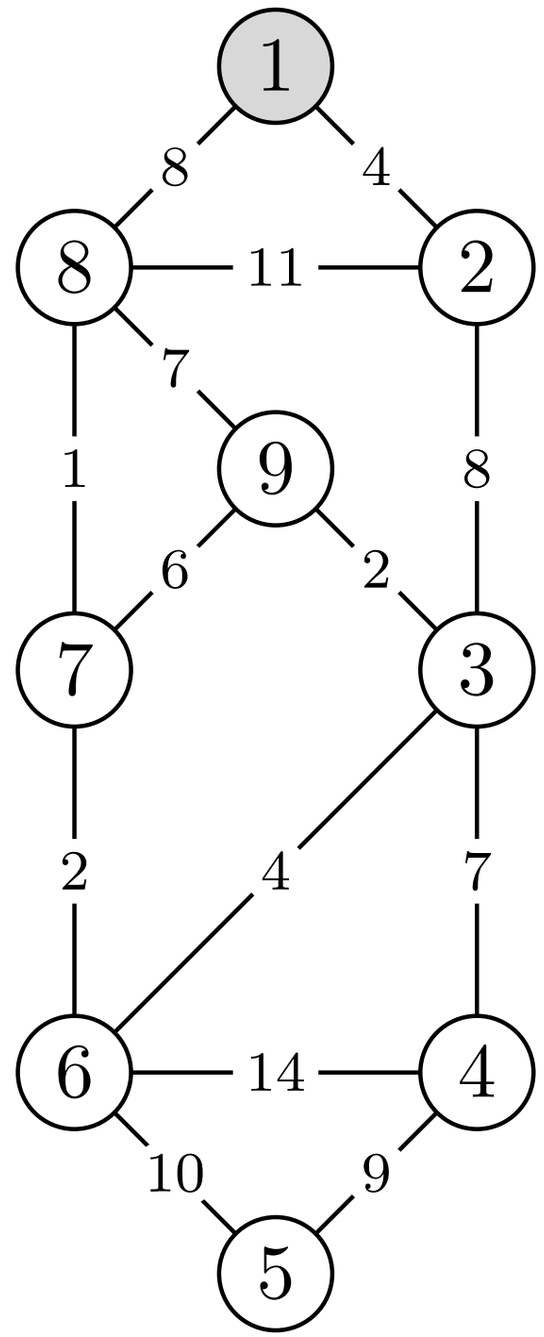




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

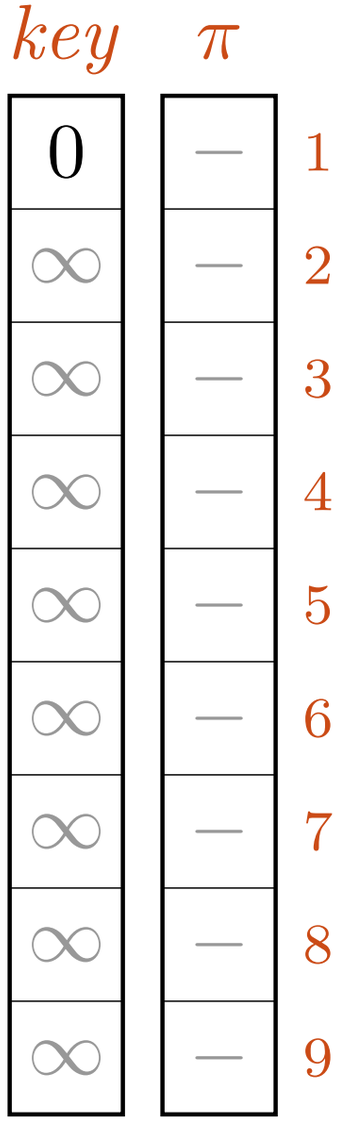
$u, v = -, -$

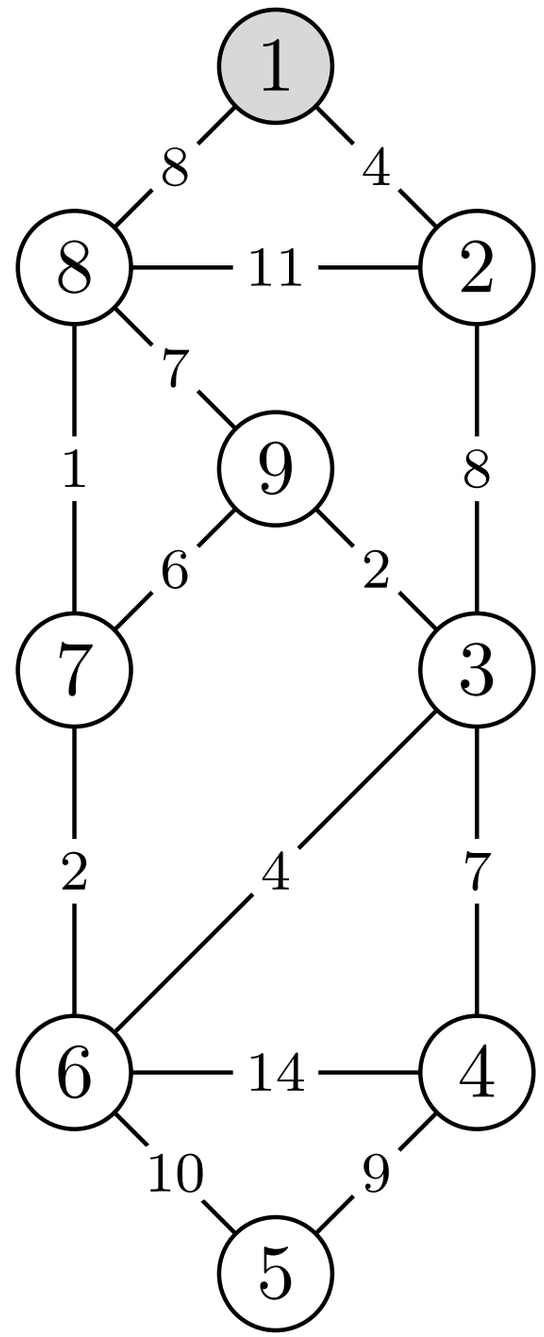




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

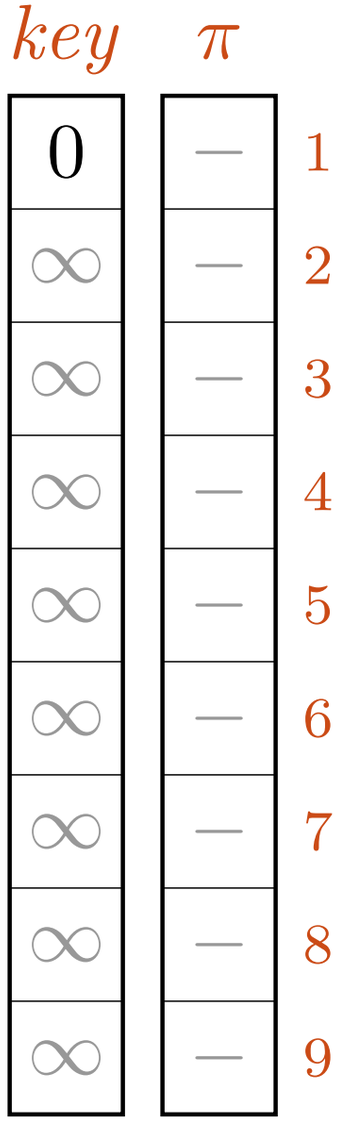
$u, v = -, -$

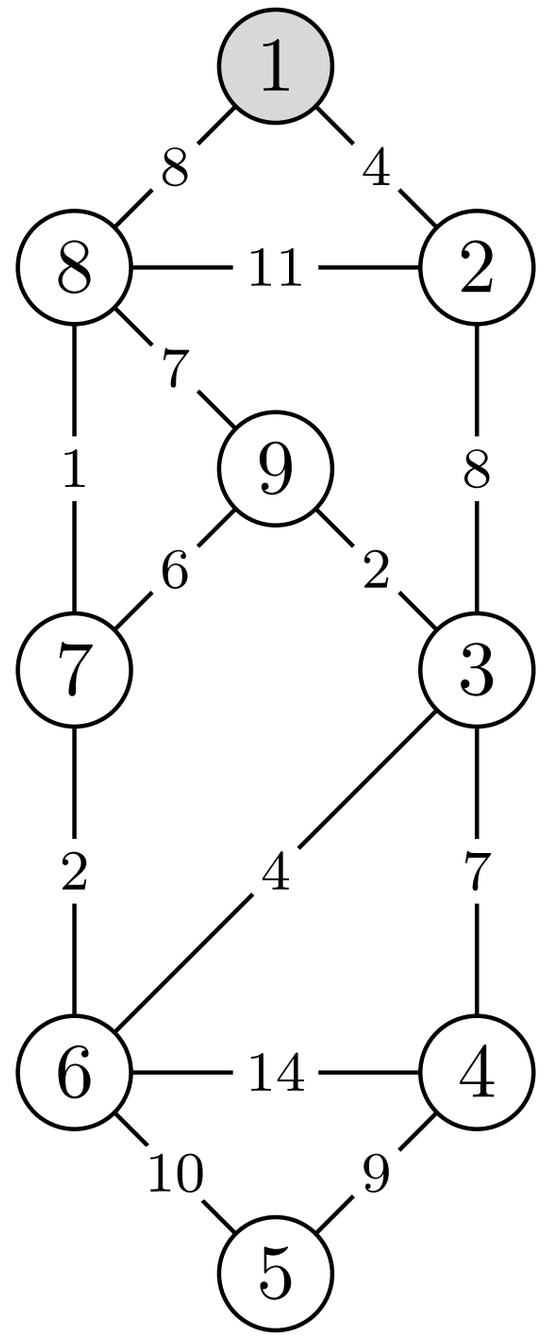




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

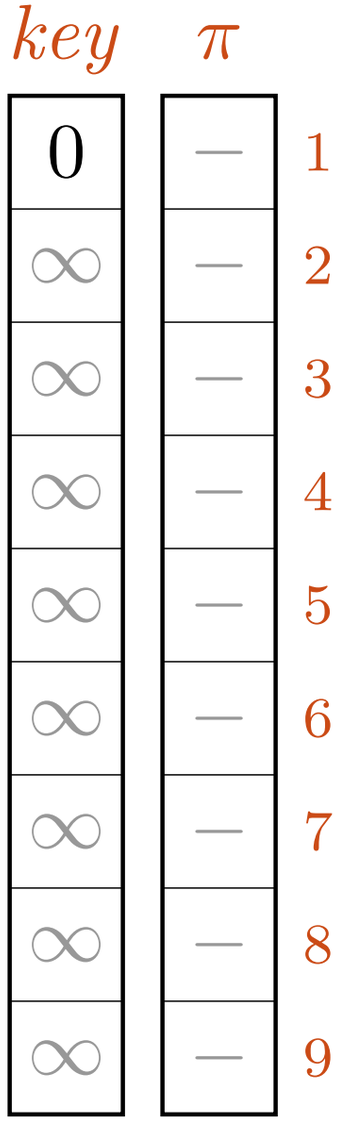
$u, v = -, -$

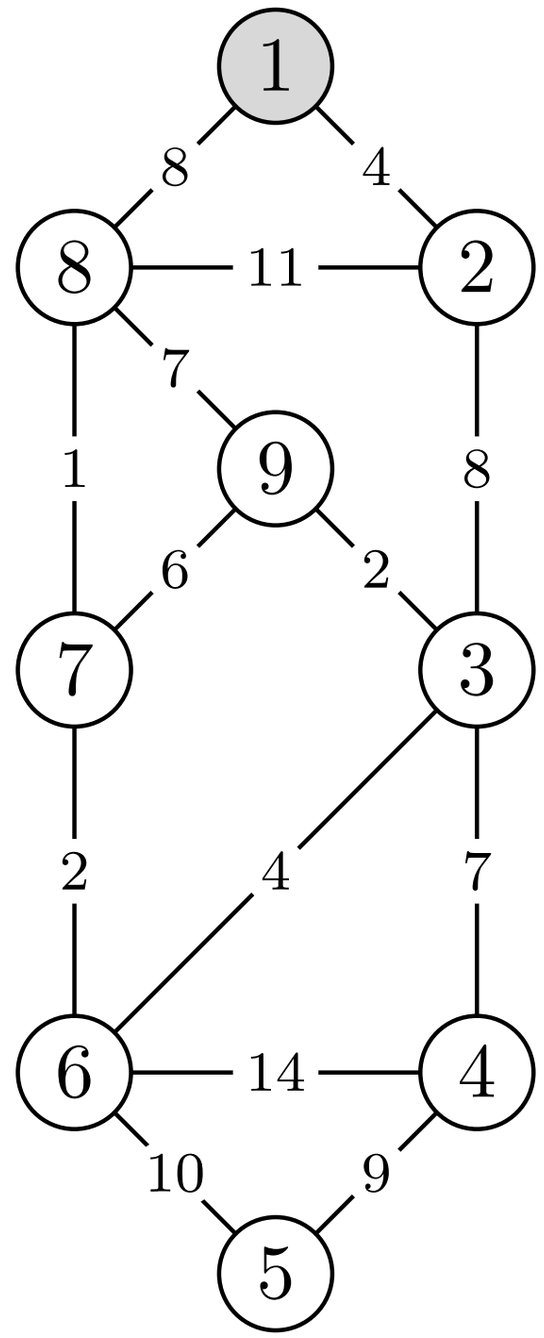




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

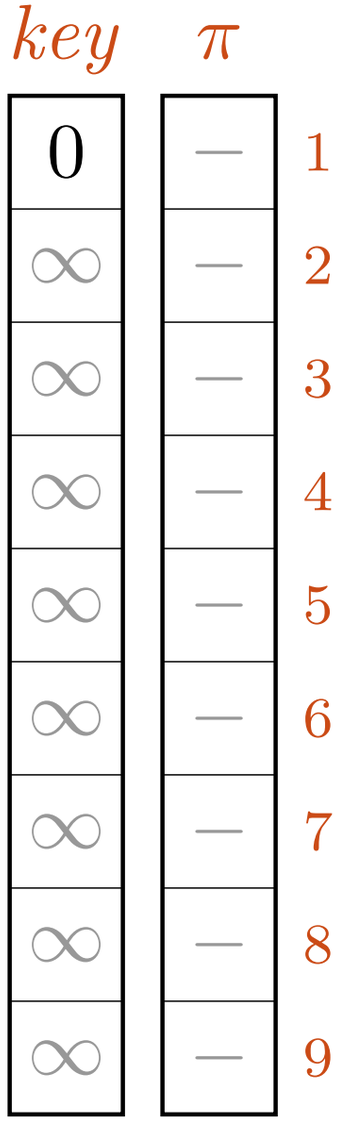
$u, v = 1, -$

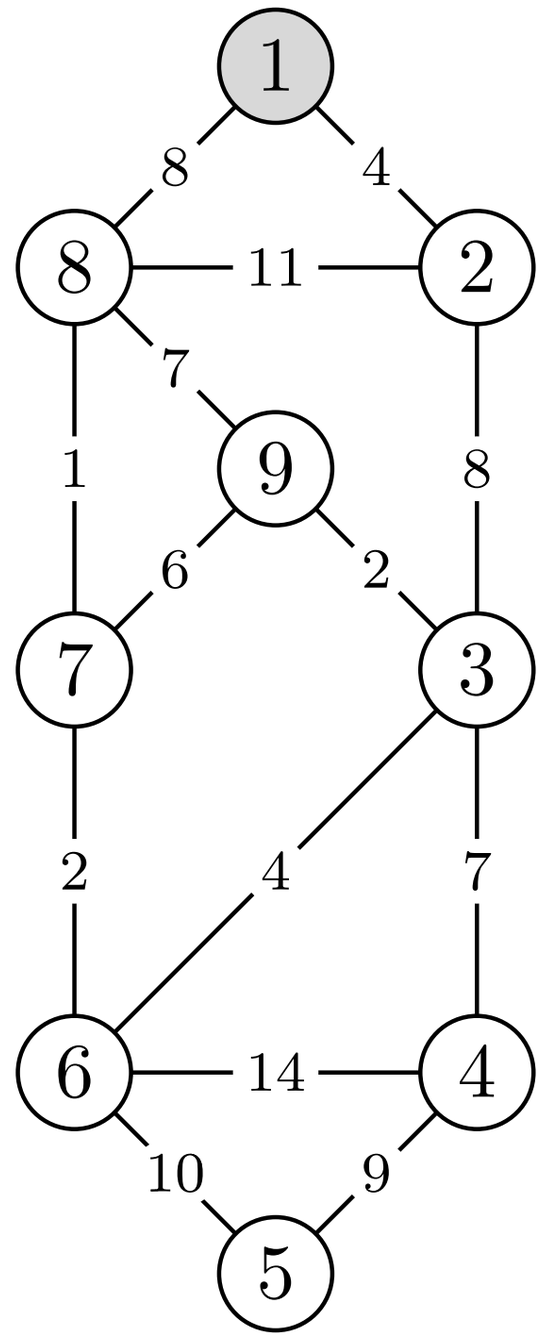




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

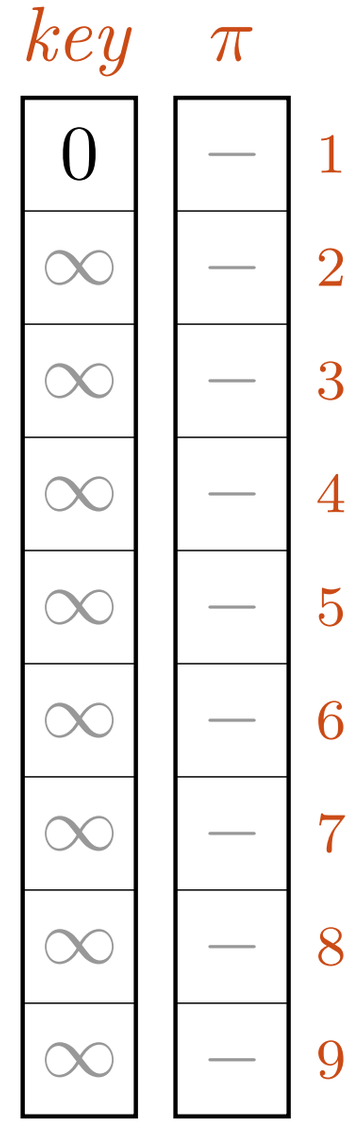
$u, v = 1, 2$

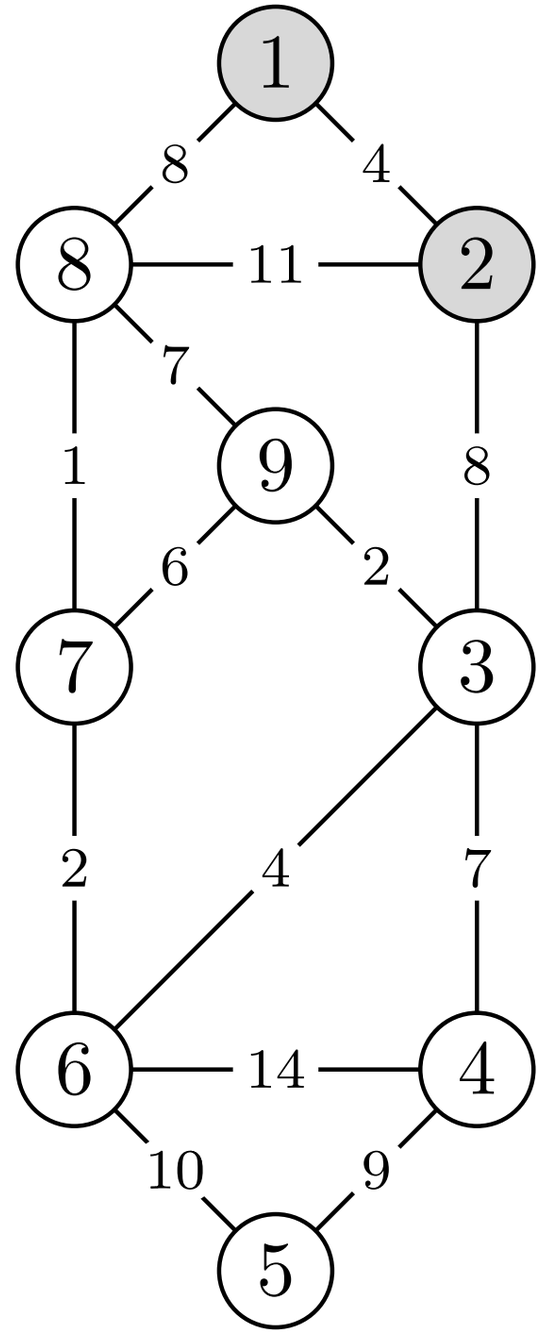




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

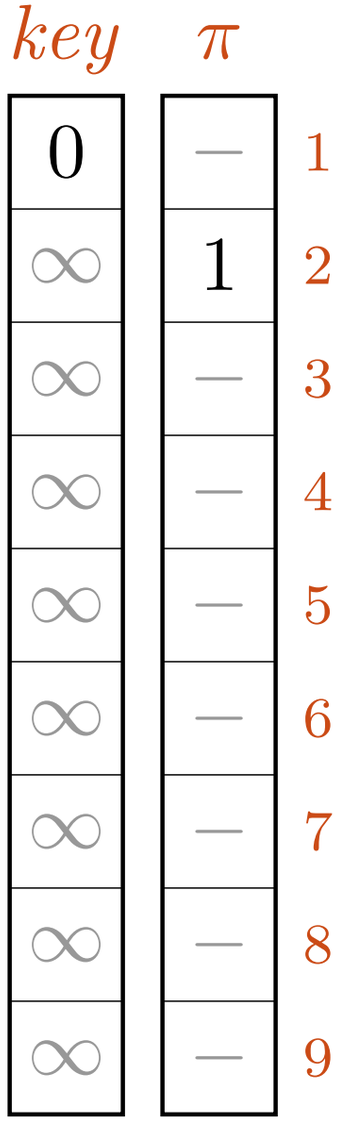
$u, v = 1, 2$

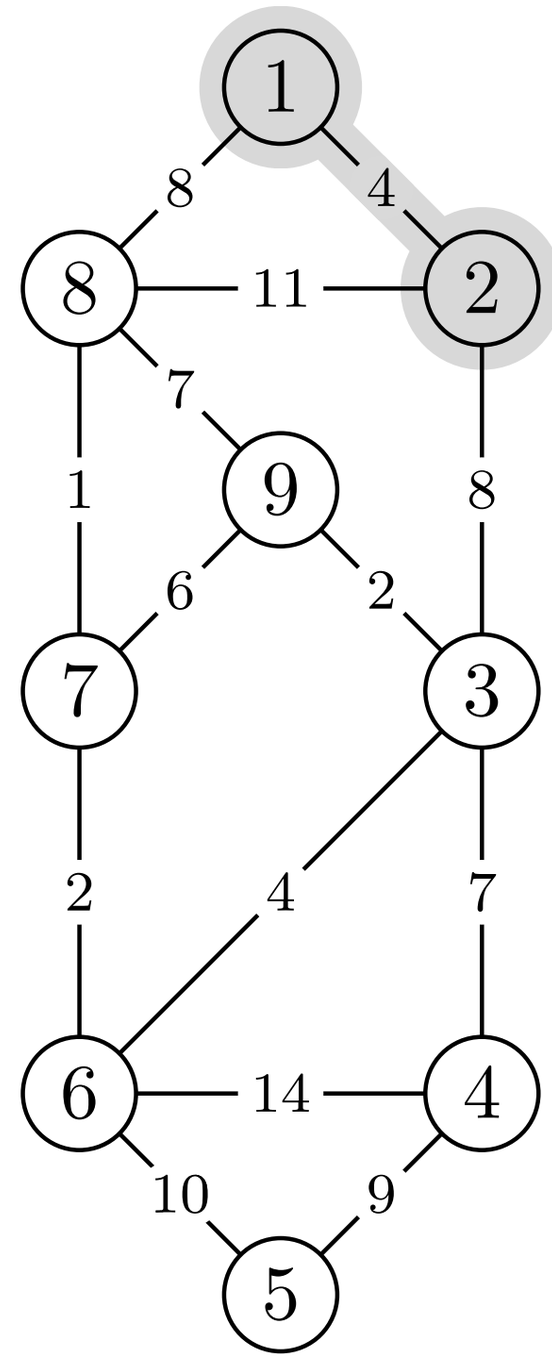




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

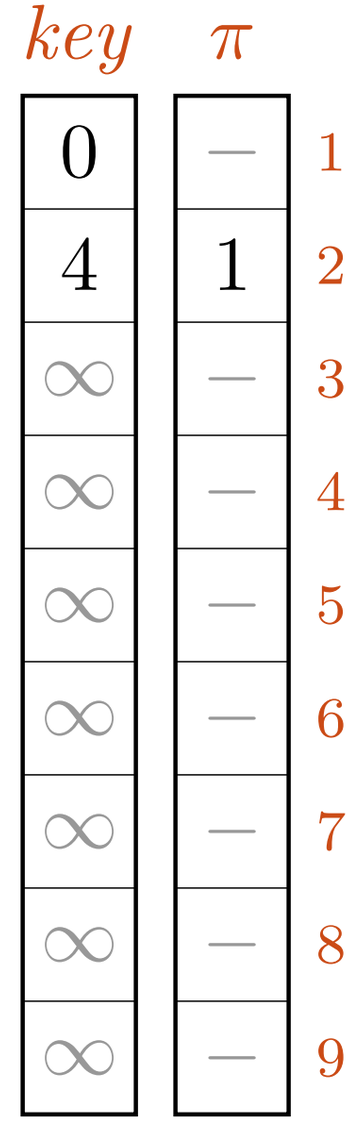
$u, v = 1, 2$

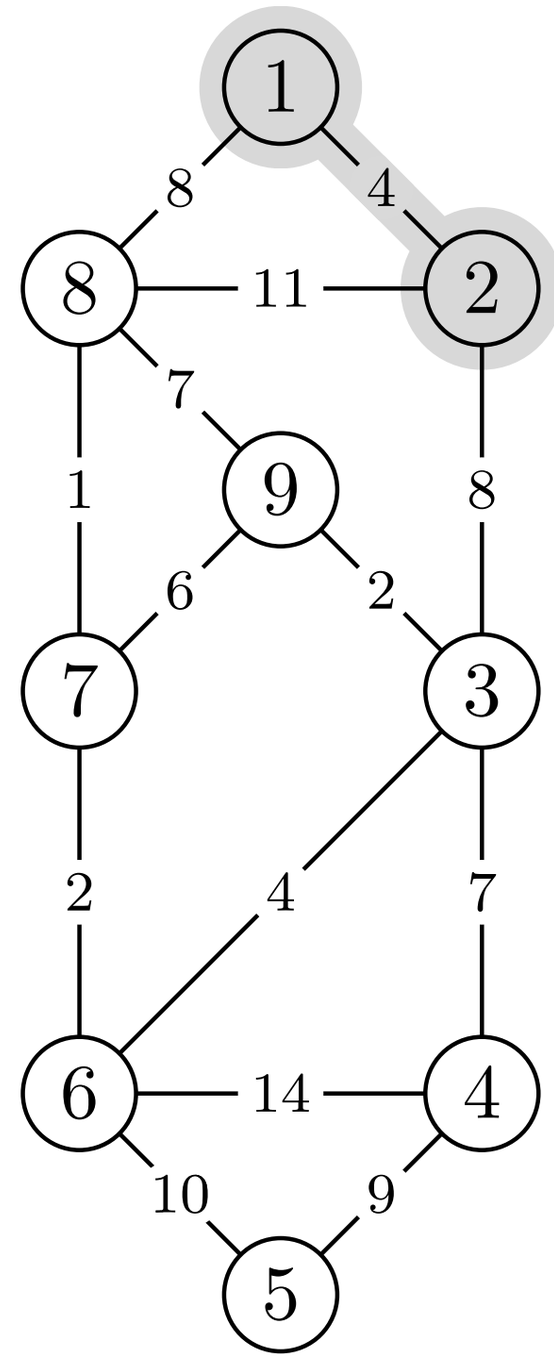




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

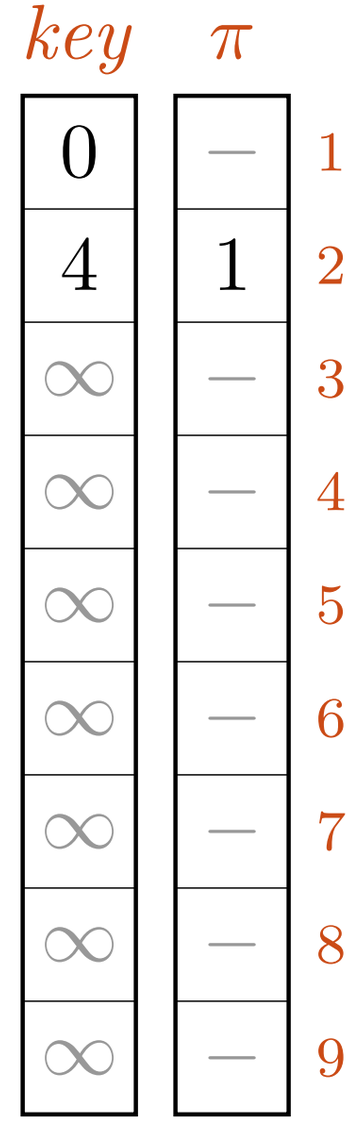
$u, v = 1, 2$

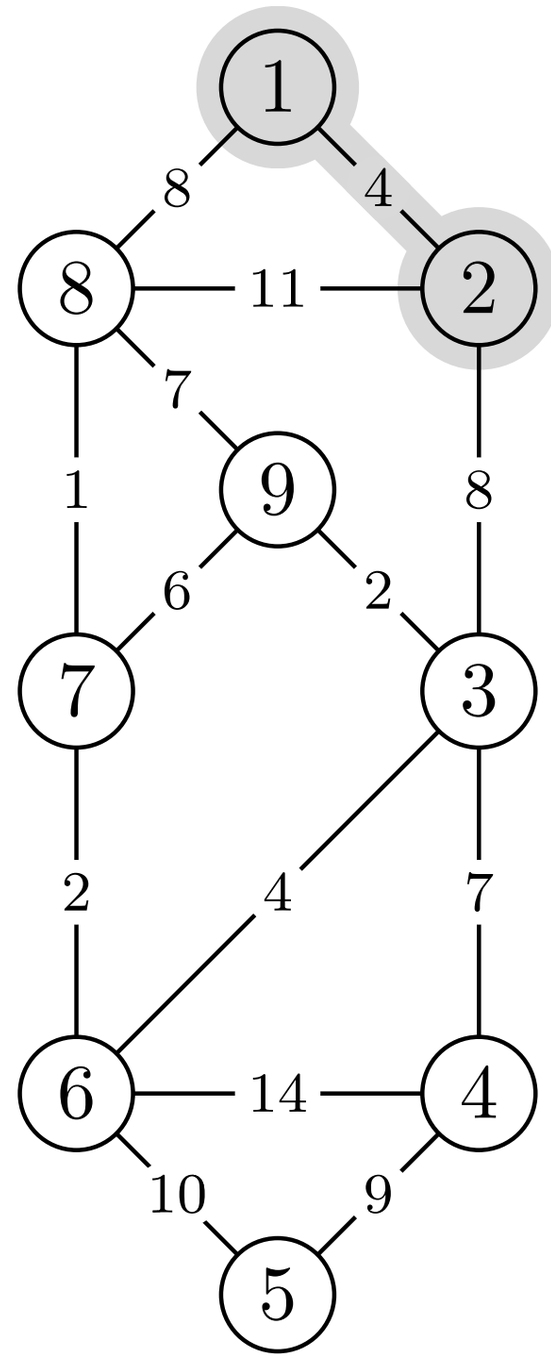




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

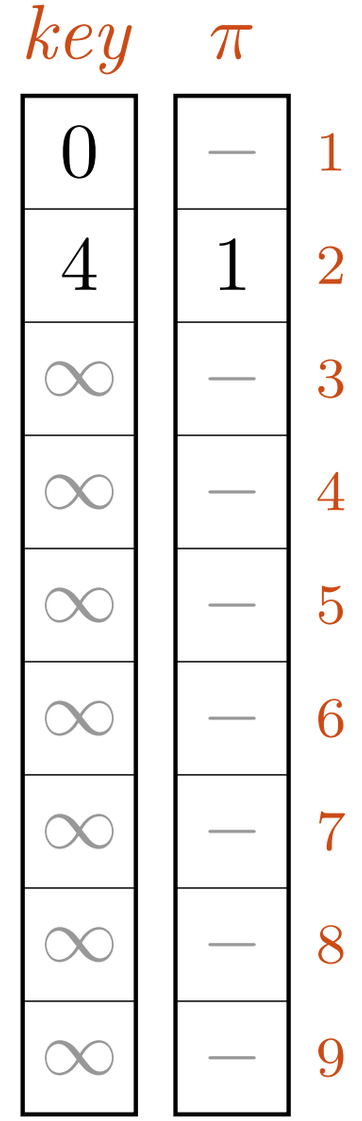
$u, v = 1, 8$

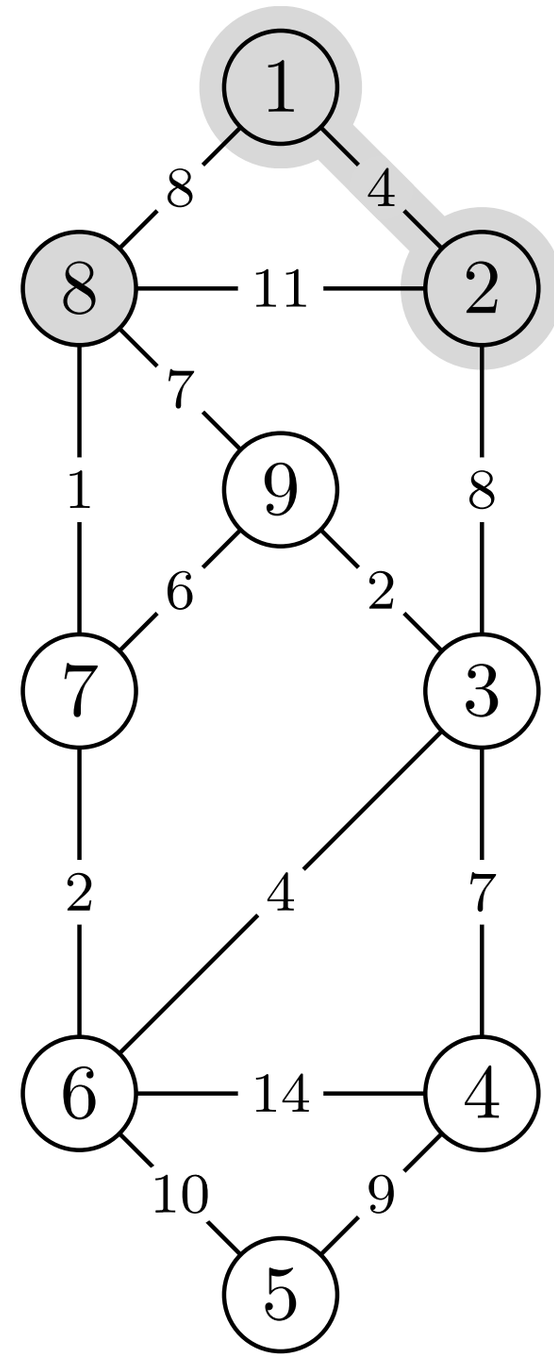




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

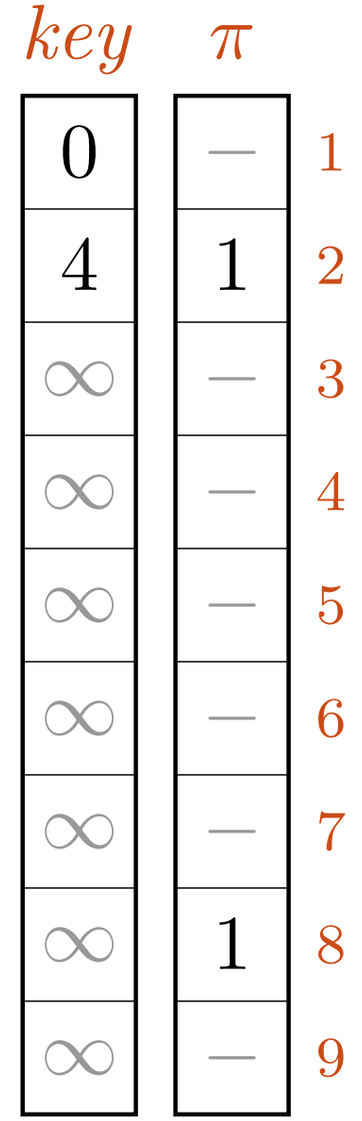
$u, v = 1, 8$

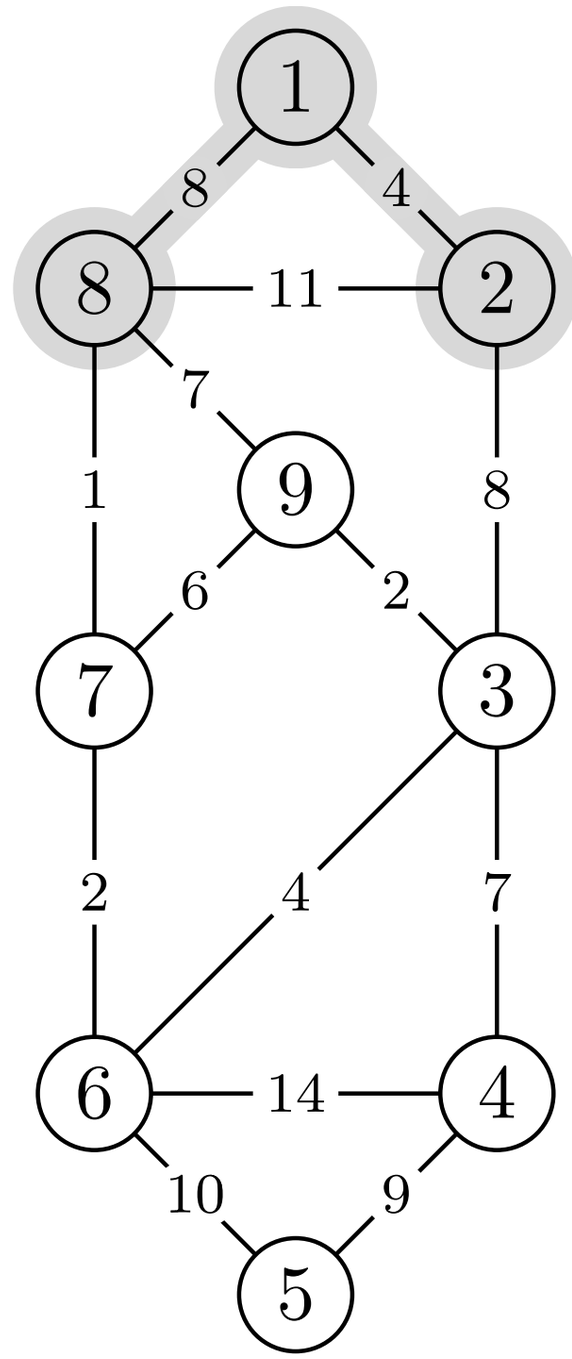




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 1, 8$

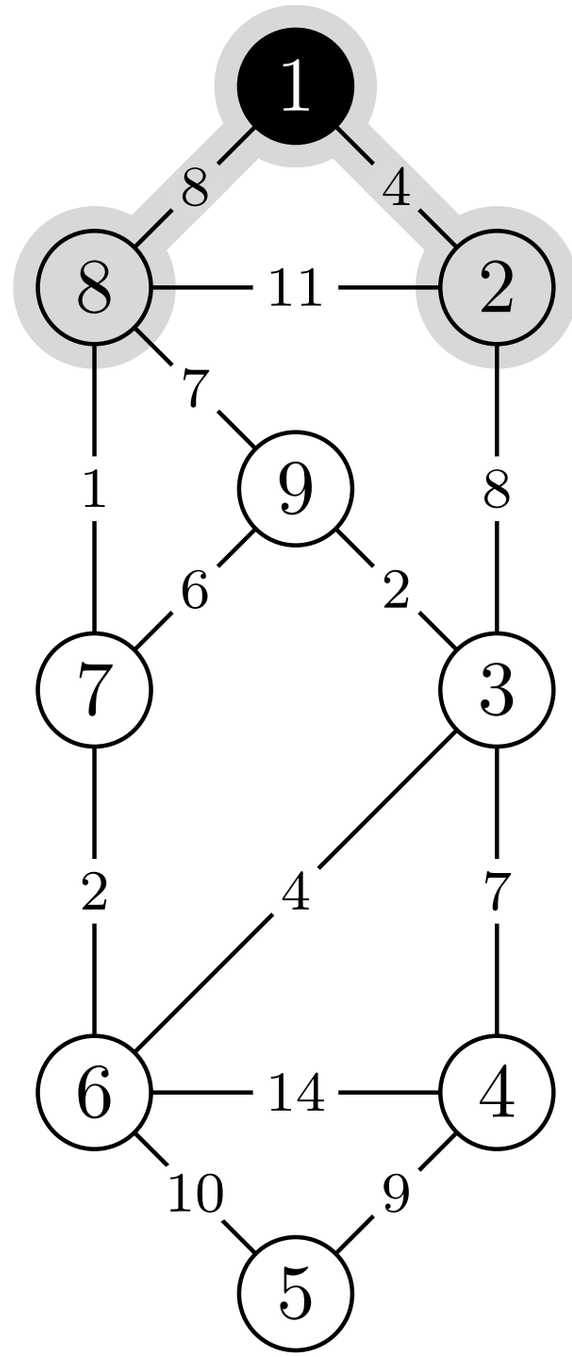




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 1, -$

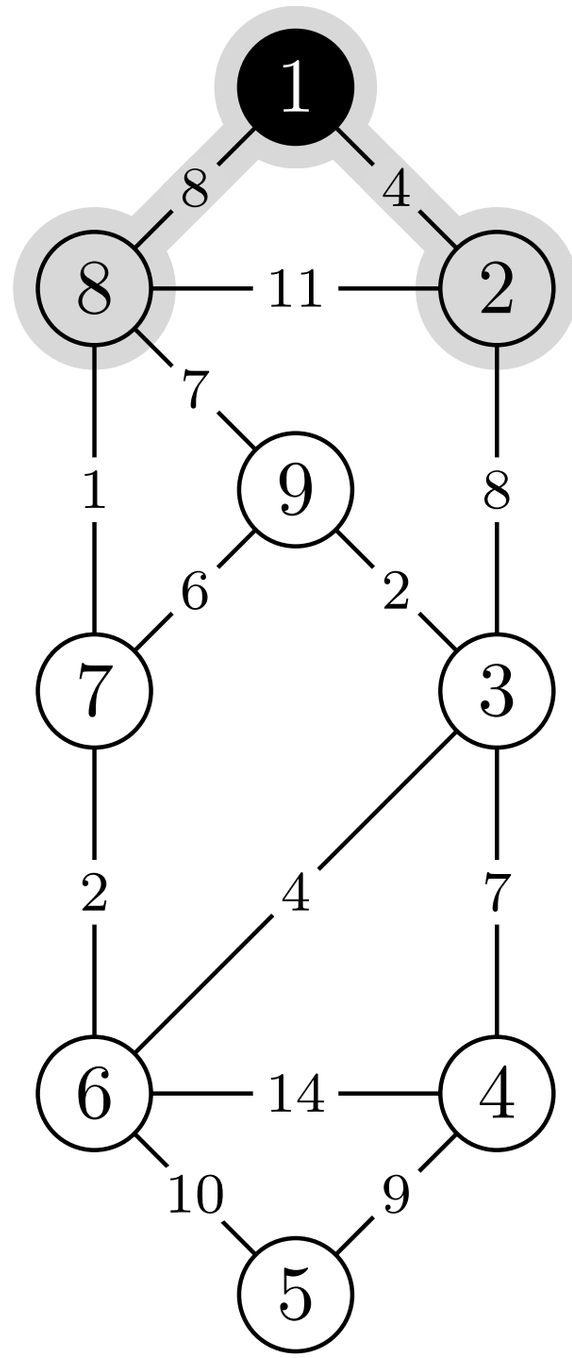
	key	π	
1	0	—	1
2	4	1	2
3	∞	—	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 2, -$

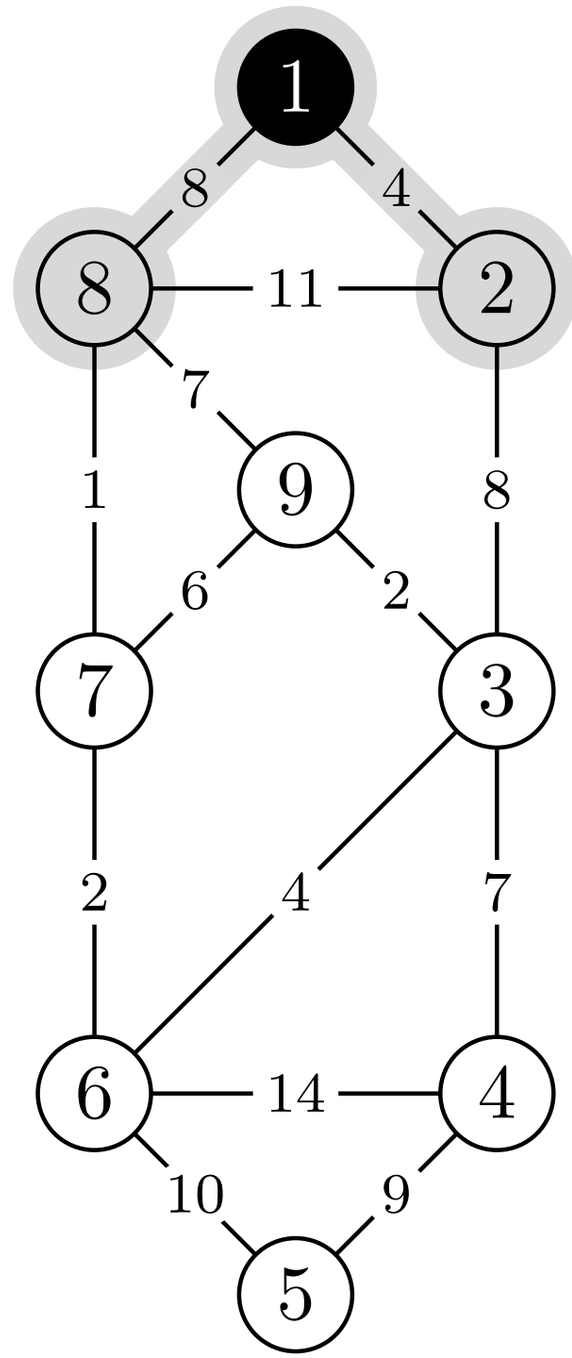
	key	π	
1	0	—	1
2	4	1	2
3	∞	—	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

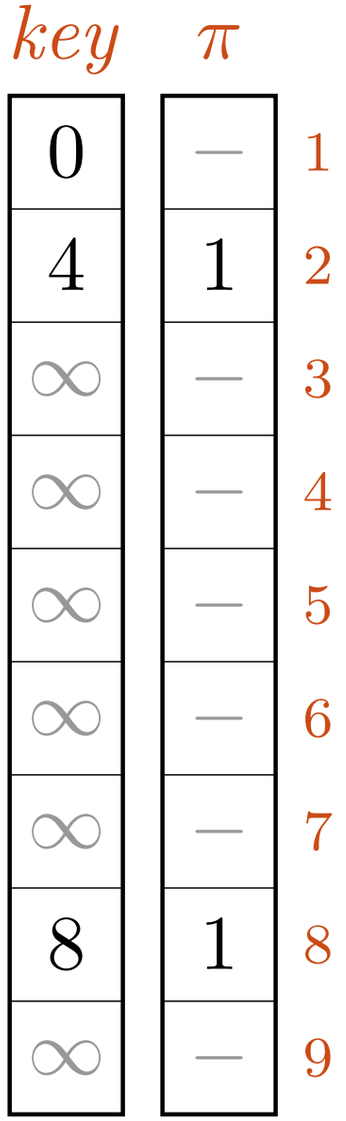
$u, v = 2, 3$

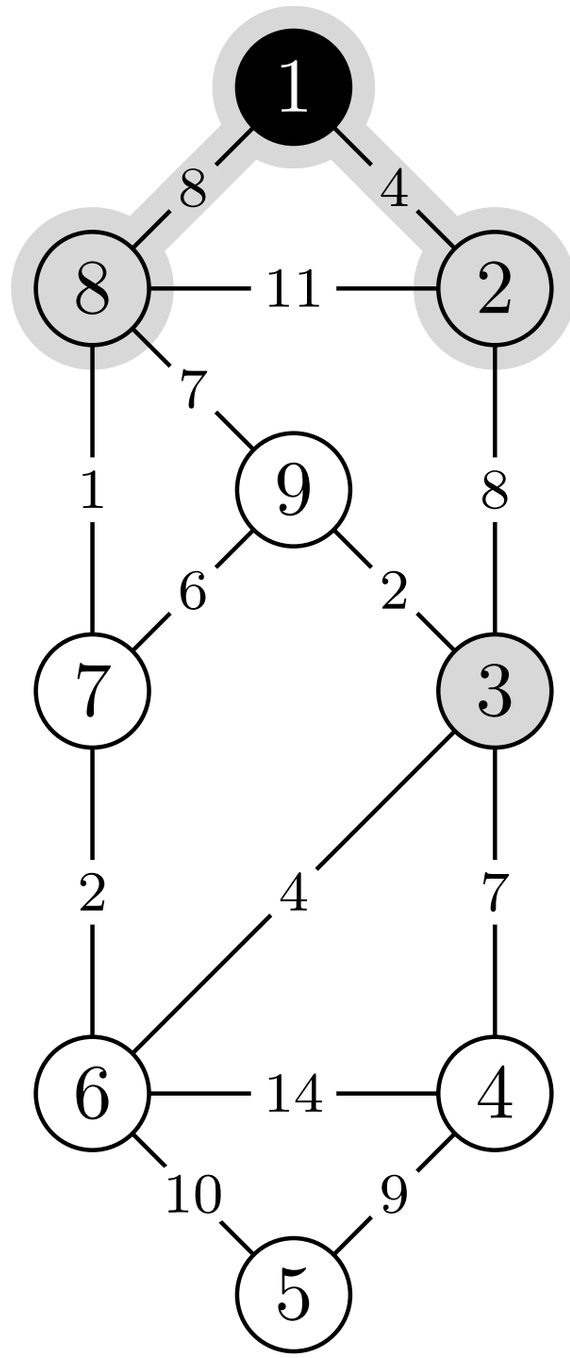
	key	π	
1	0	—	1
2	4	1	2
3	∞	—	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

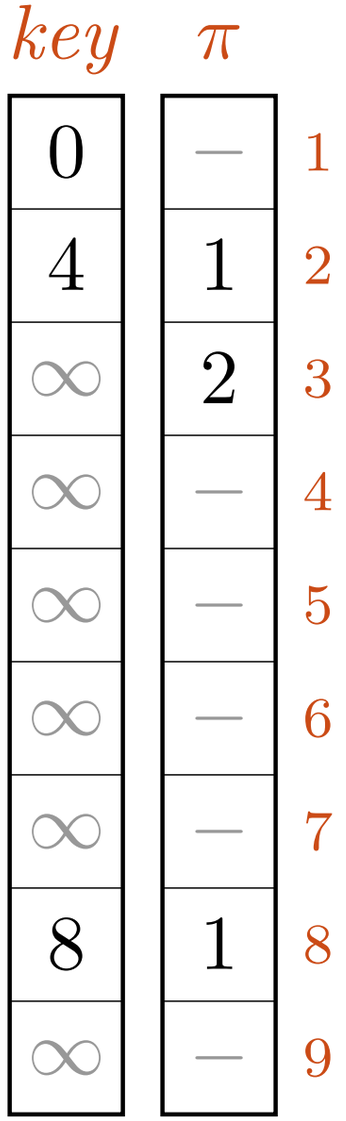
$u, v = 2, 3$

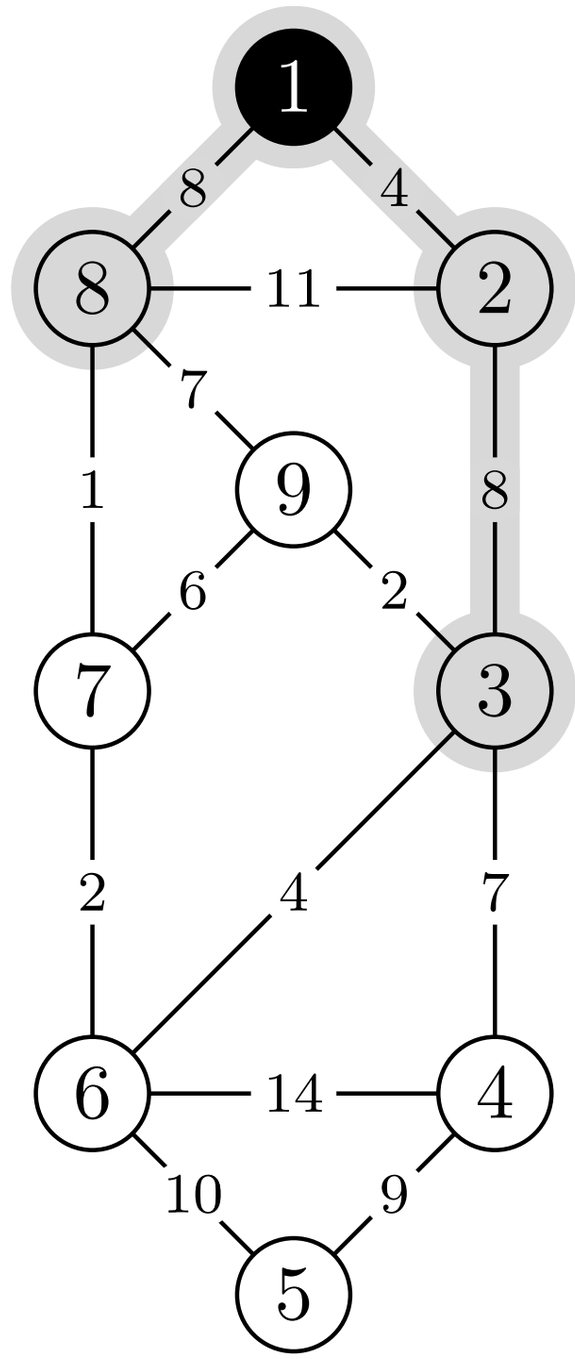




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 2, 3$

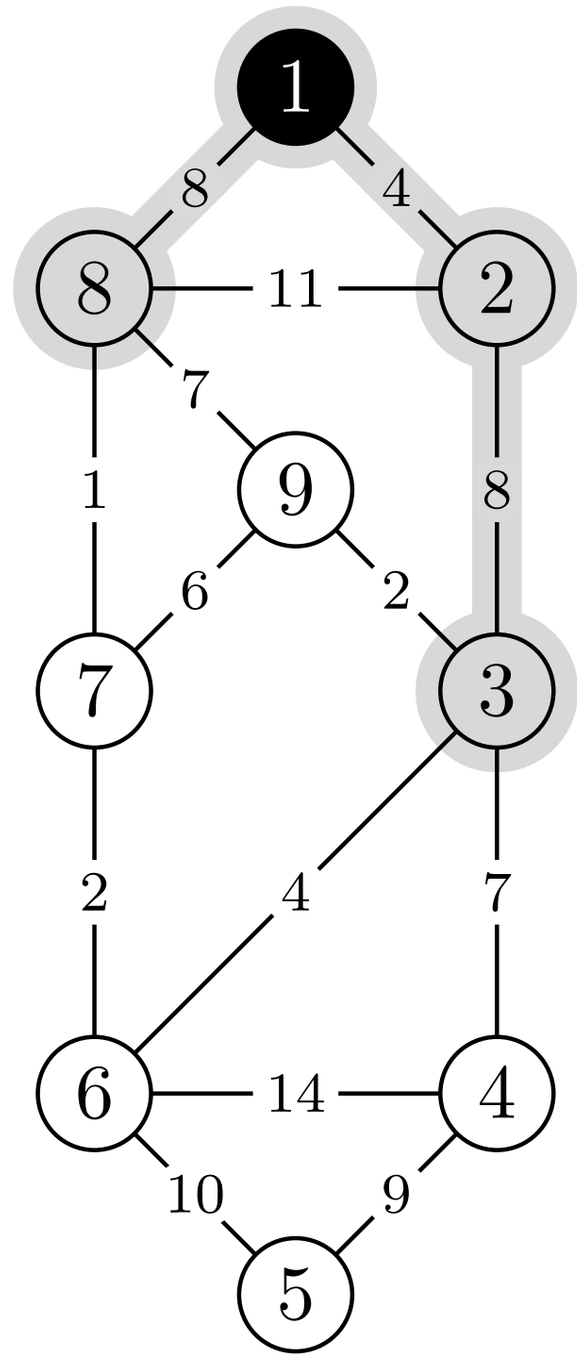




```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 2, 3$

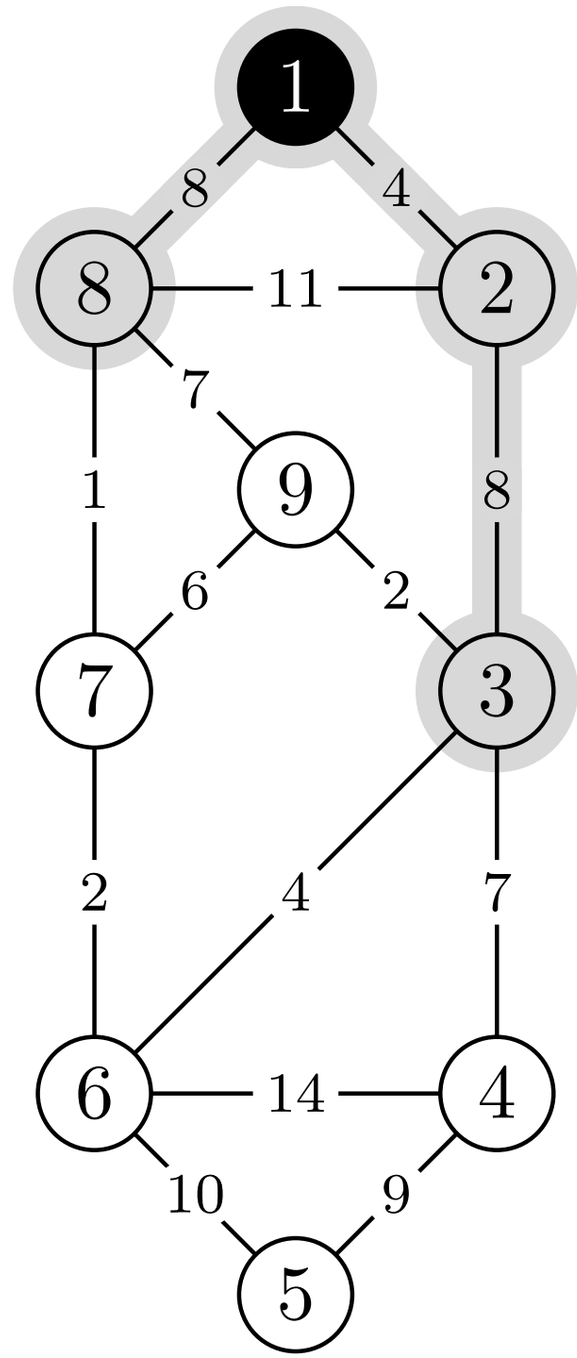
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 2, 8$

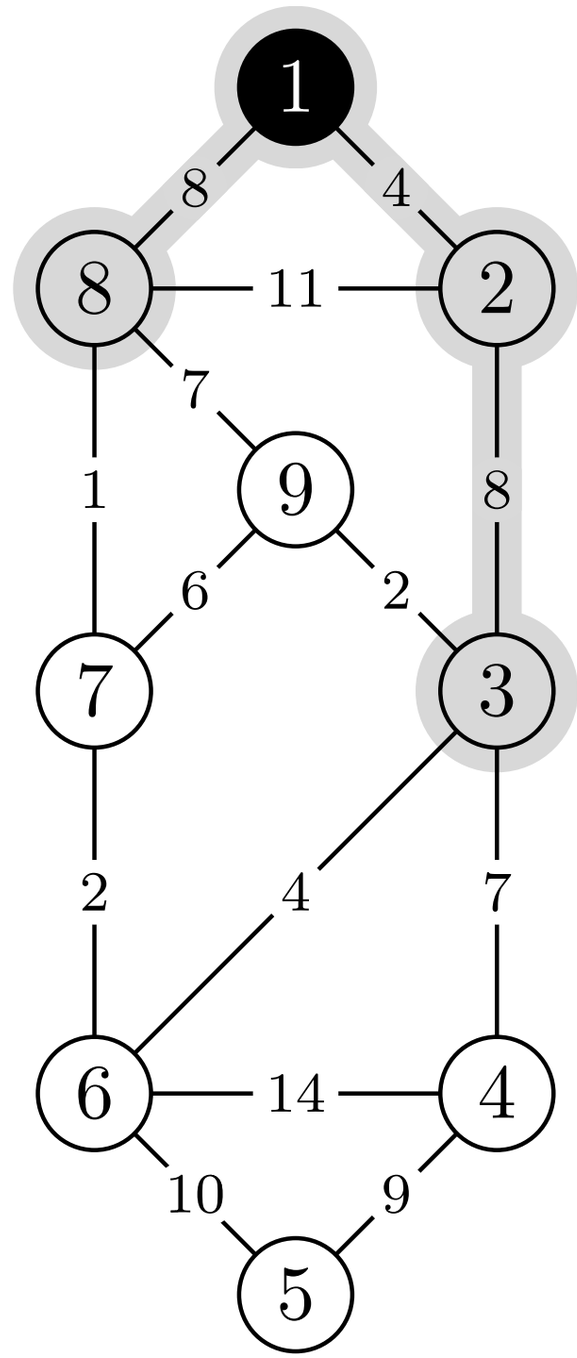
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 2, 8$

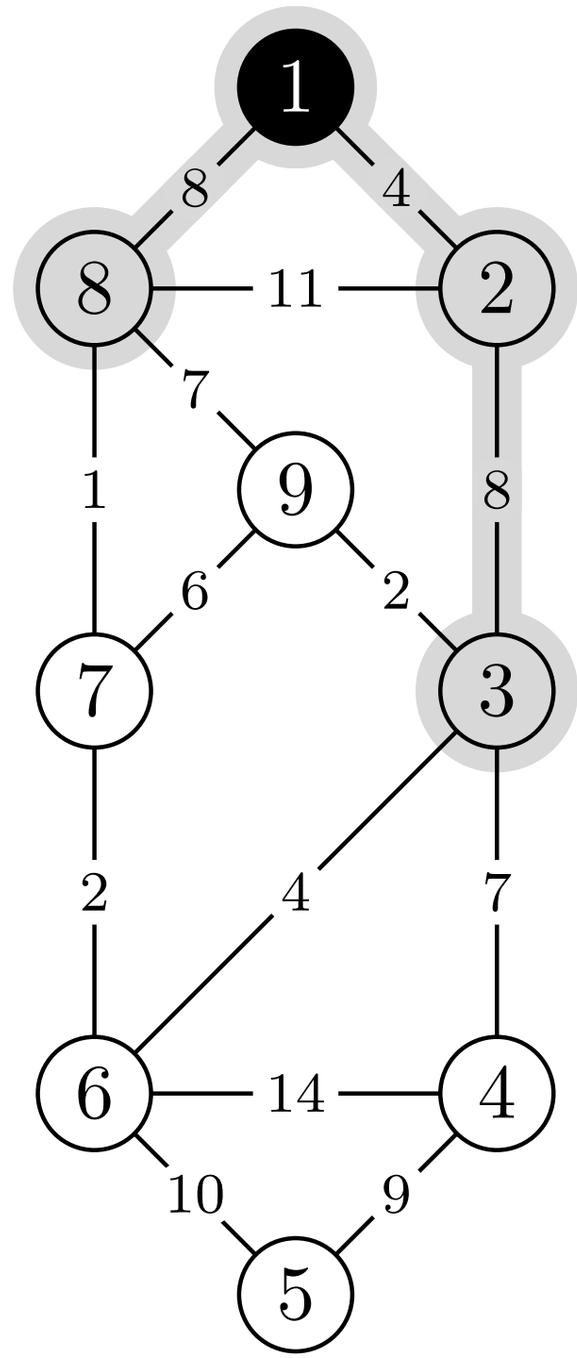
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 2, 1$

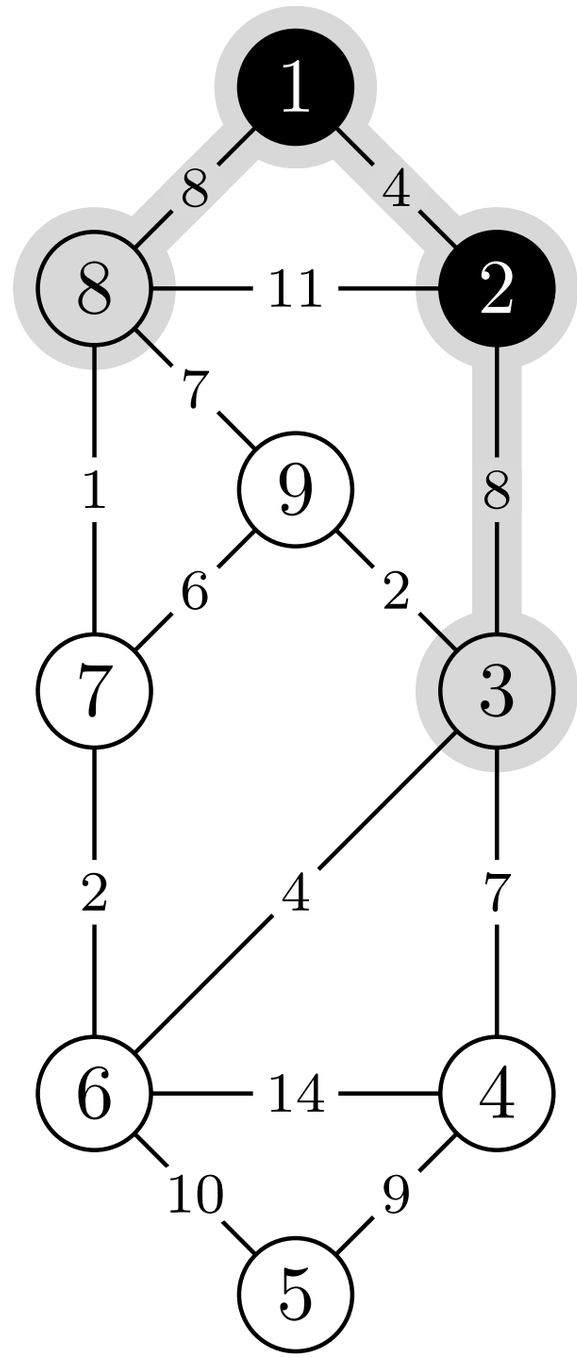
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 2, -$

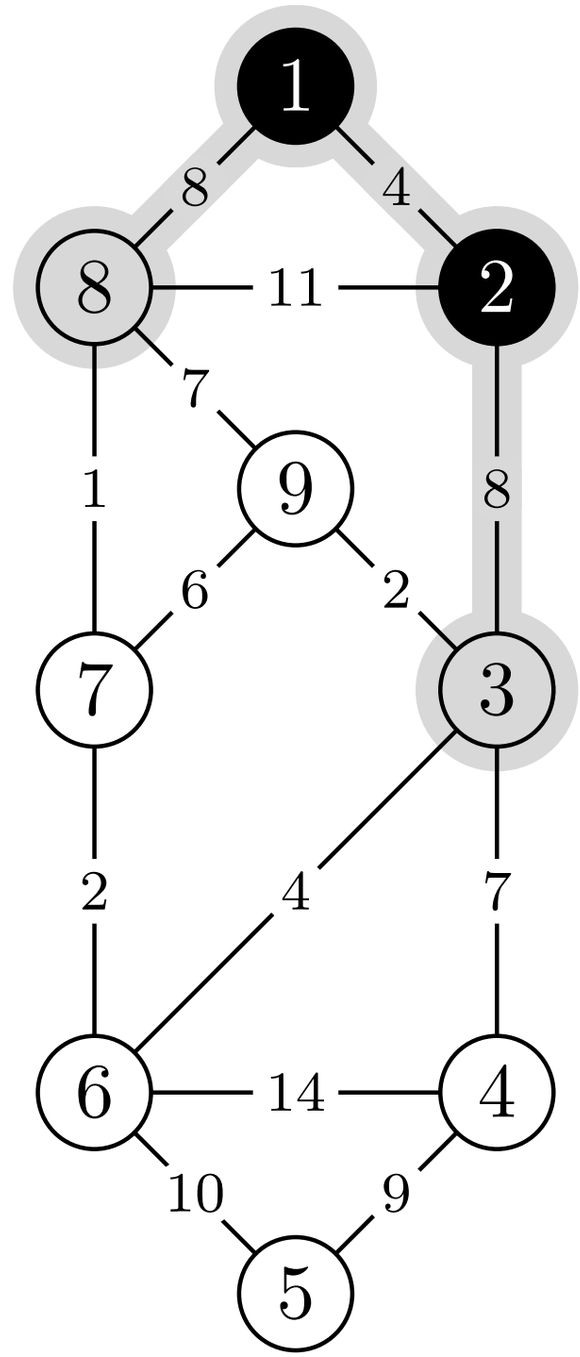
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, -$

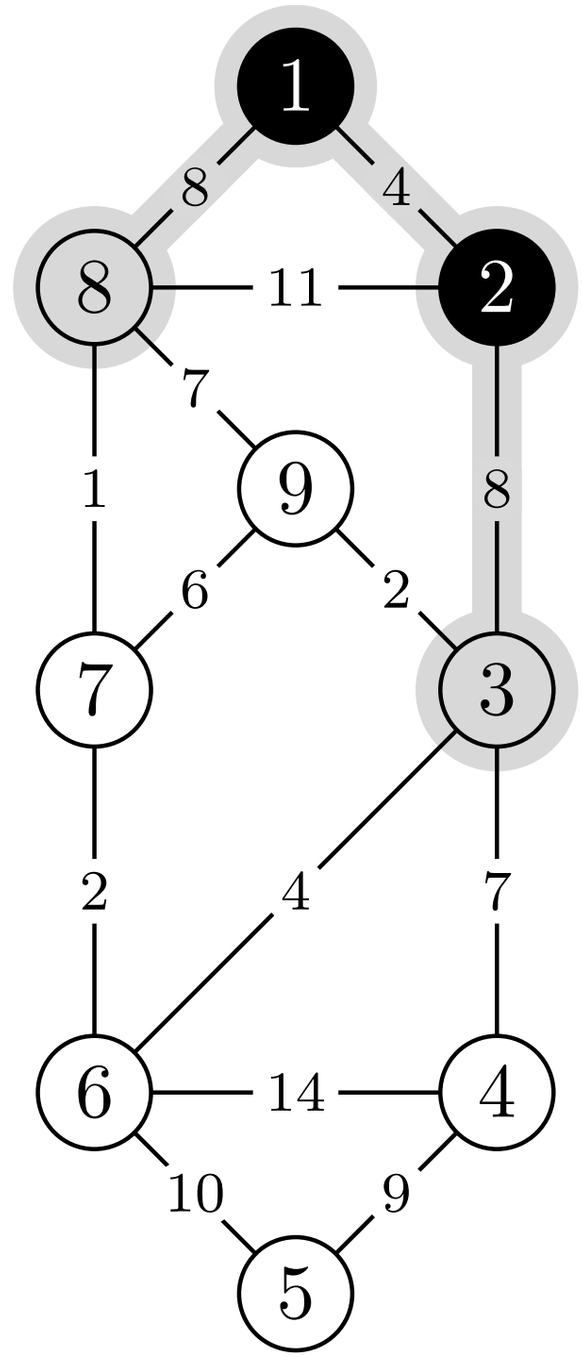
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 9$

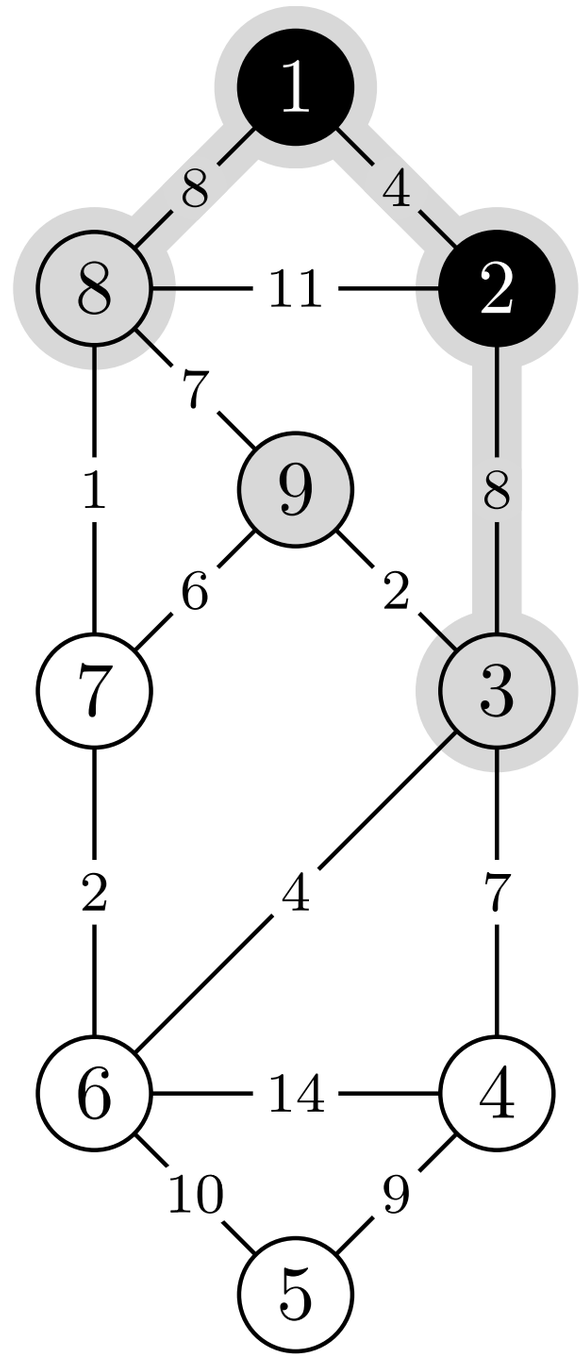
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 9$

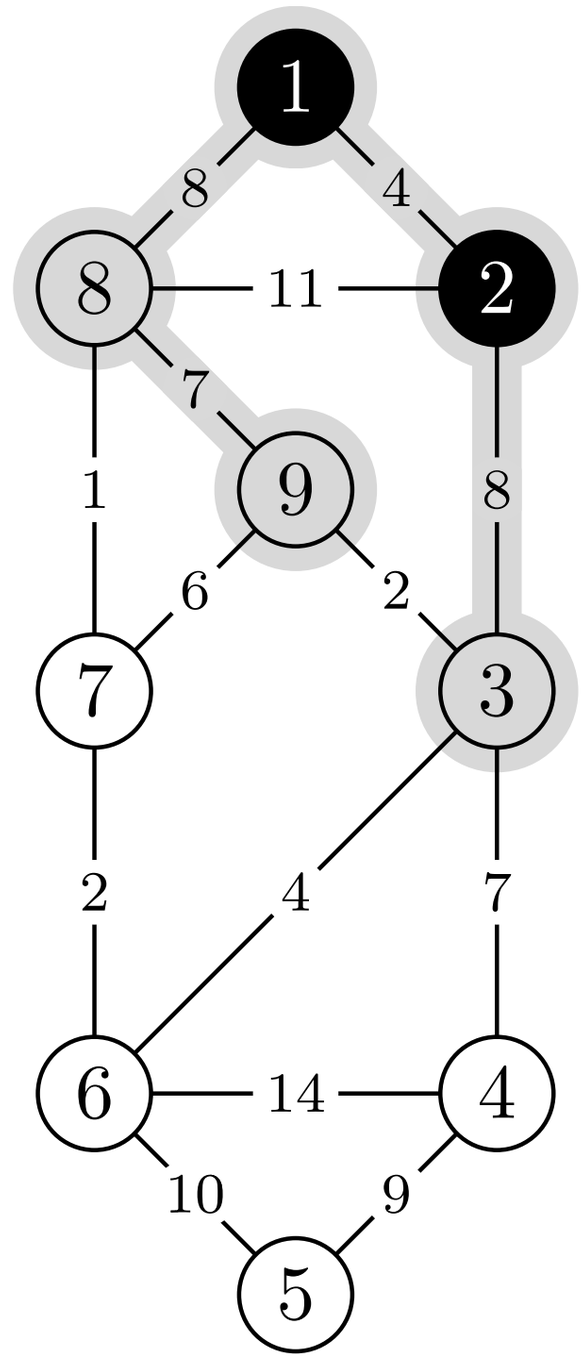
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	—	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 9$

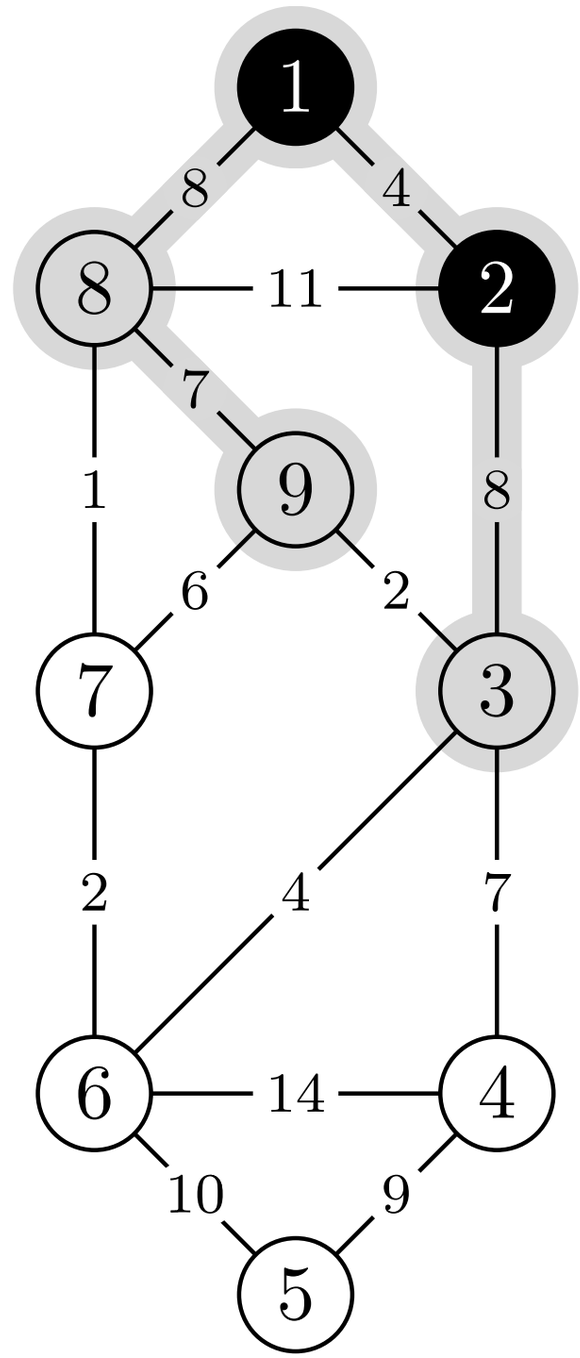
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	∞	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 9$

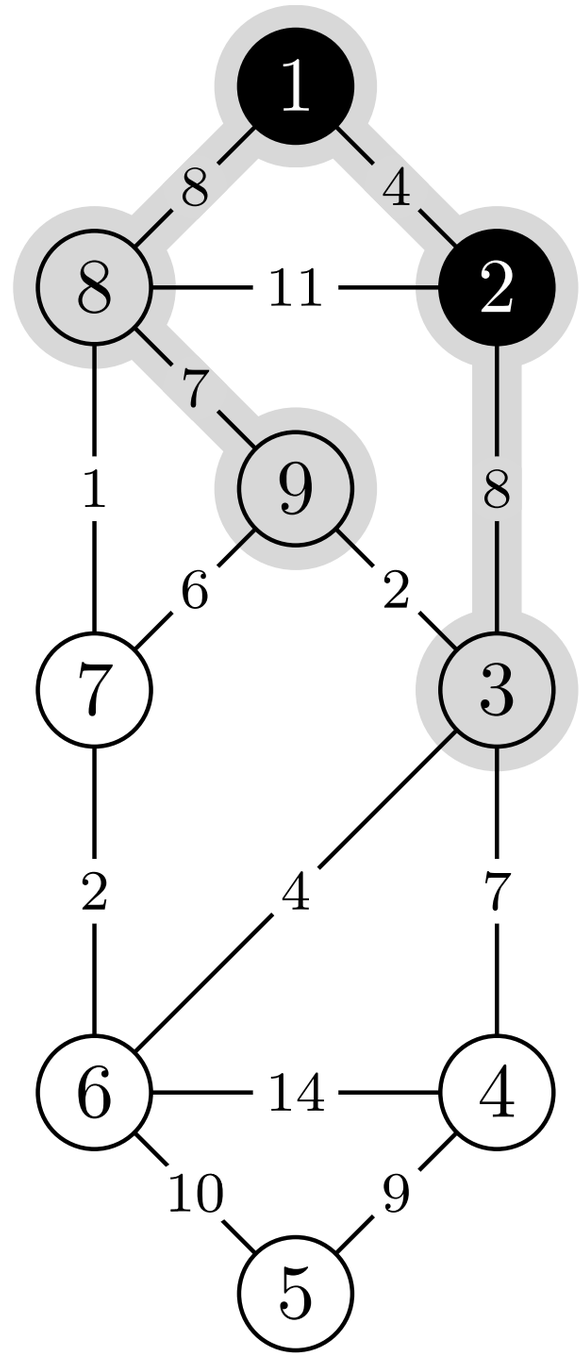
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 1$

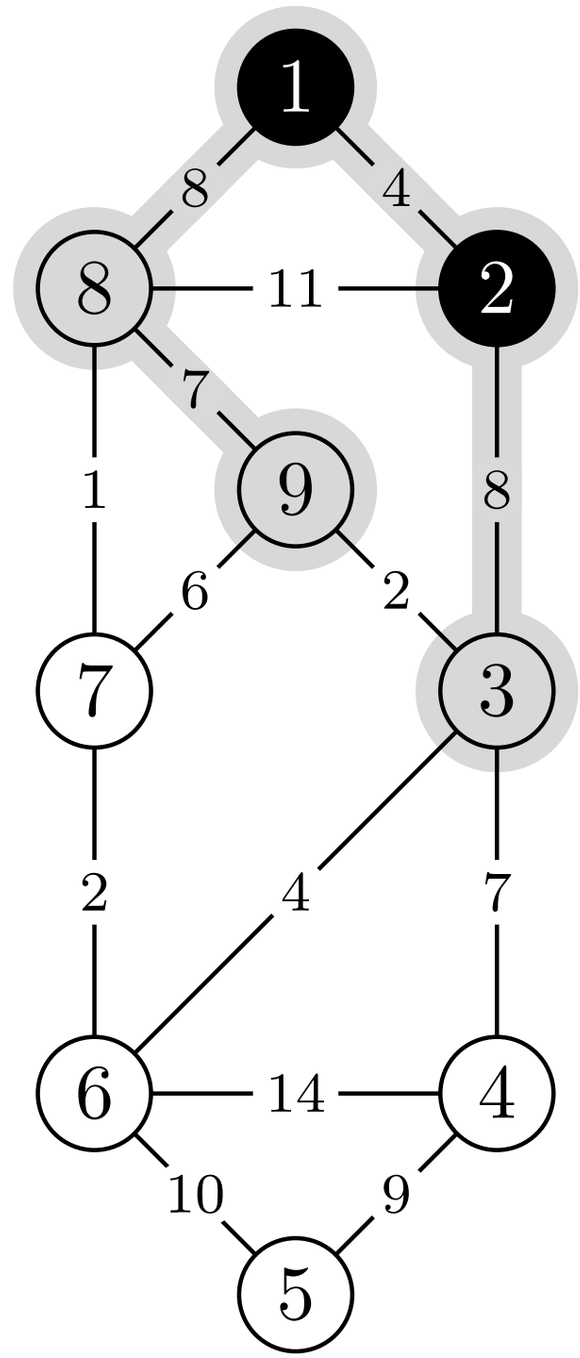
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 1$

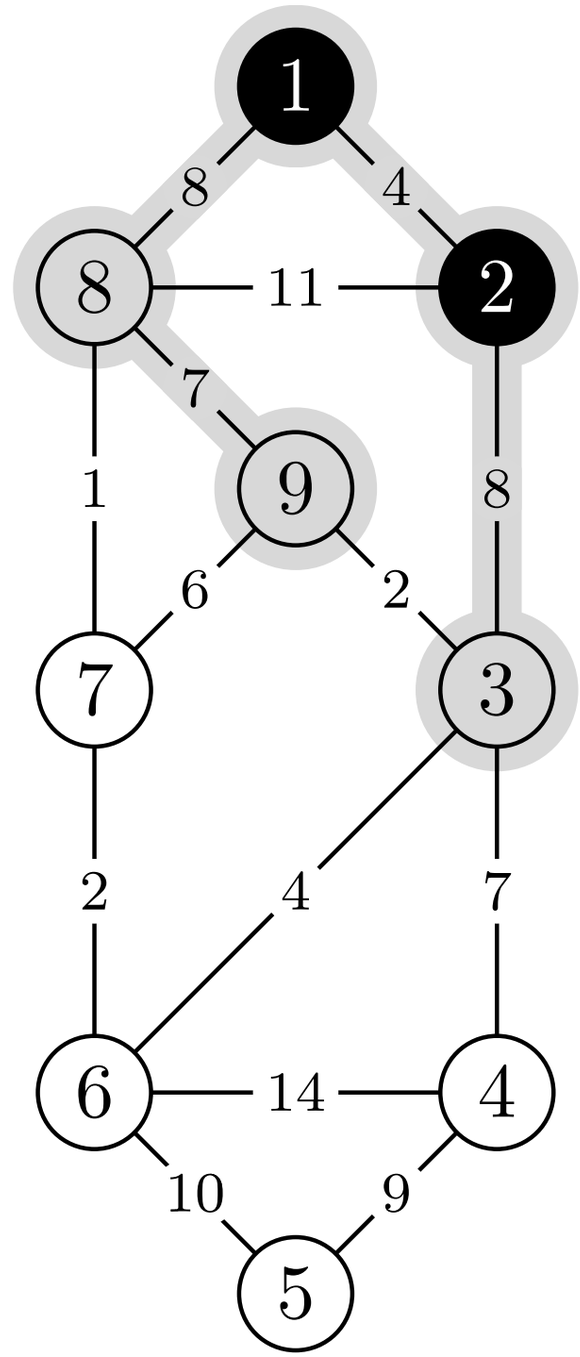
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 2$

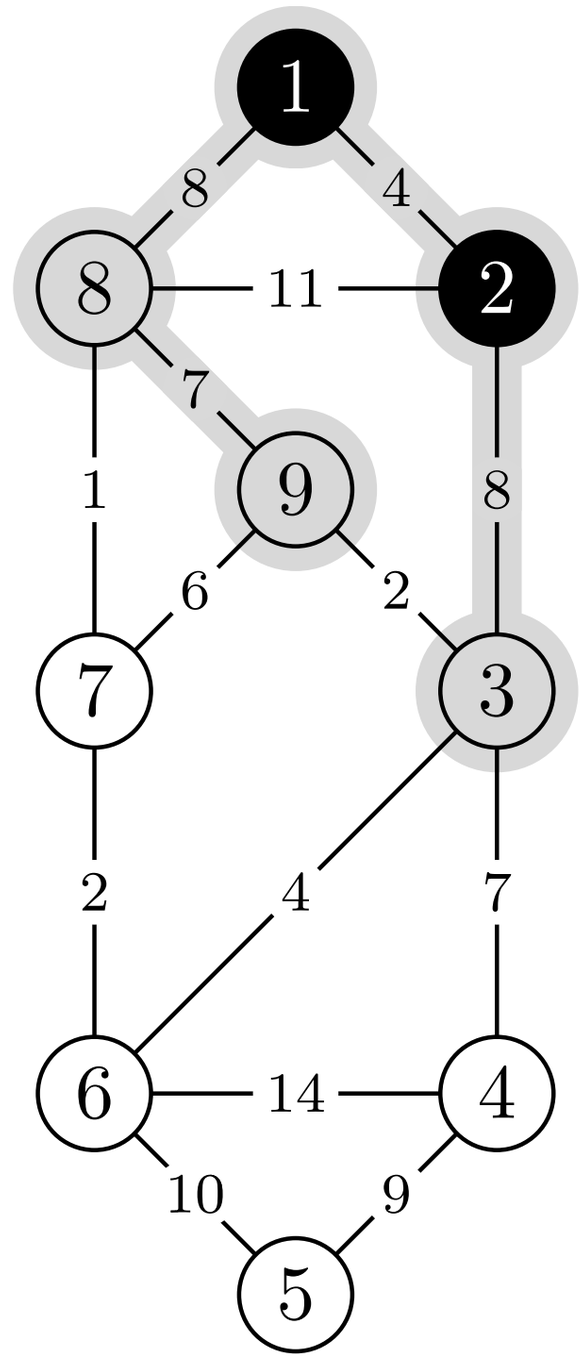
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 2$

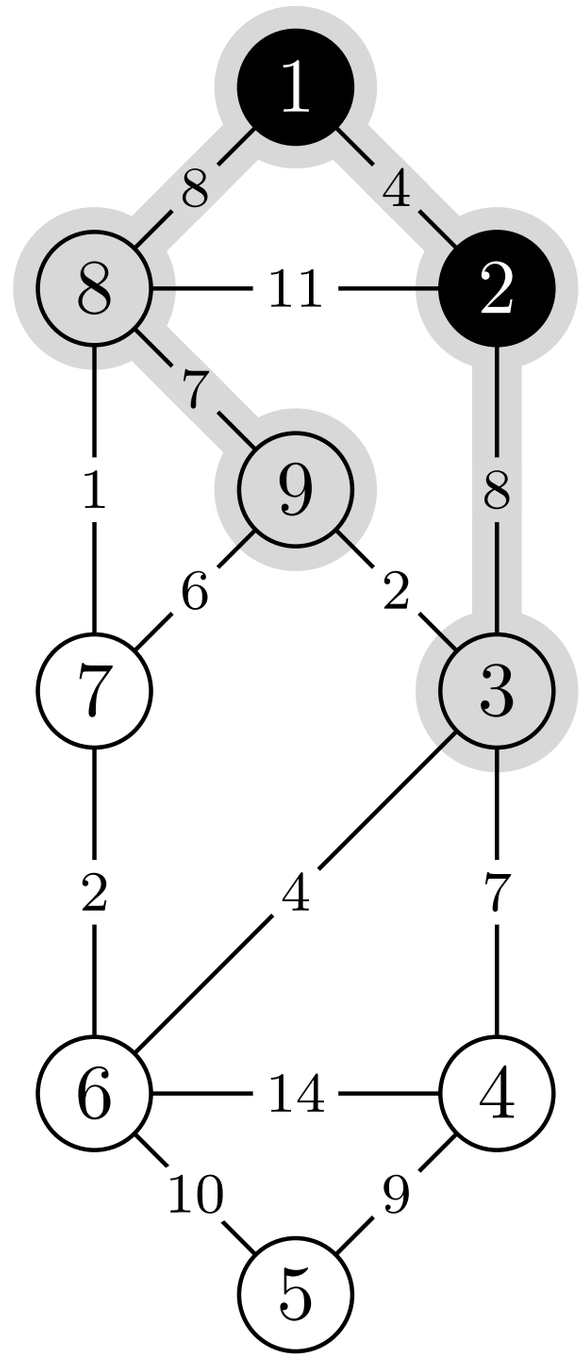
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 7$

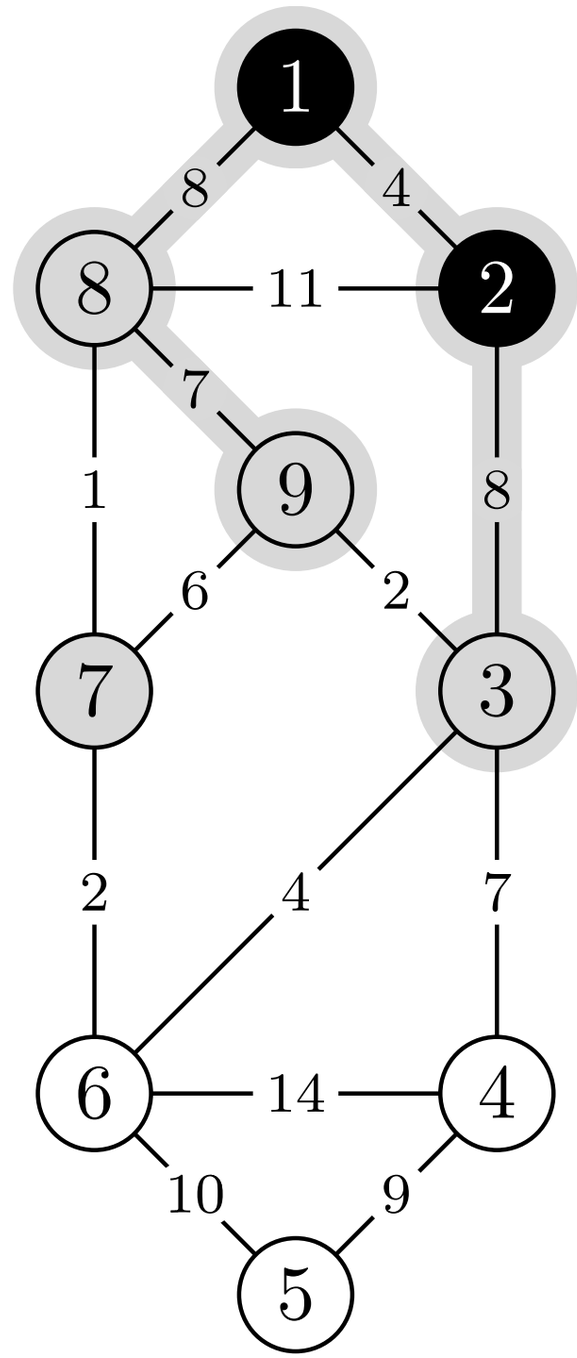
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 7$

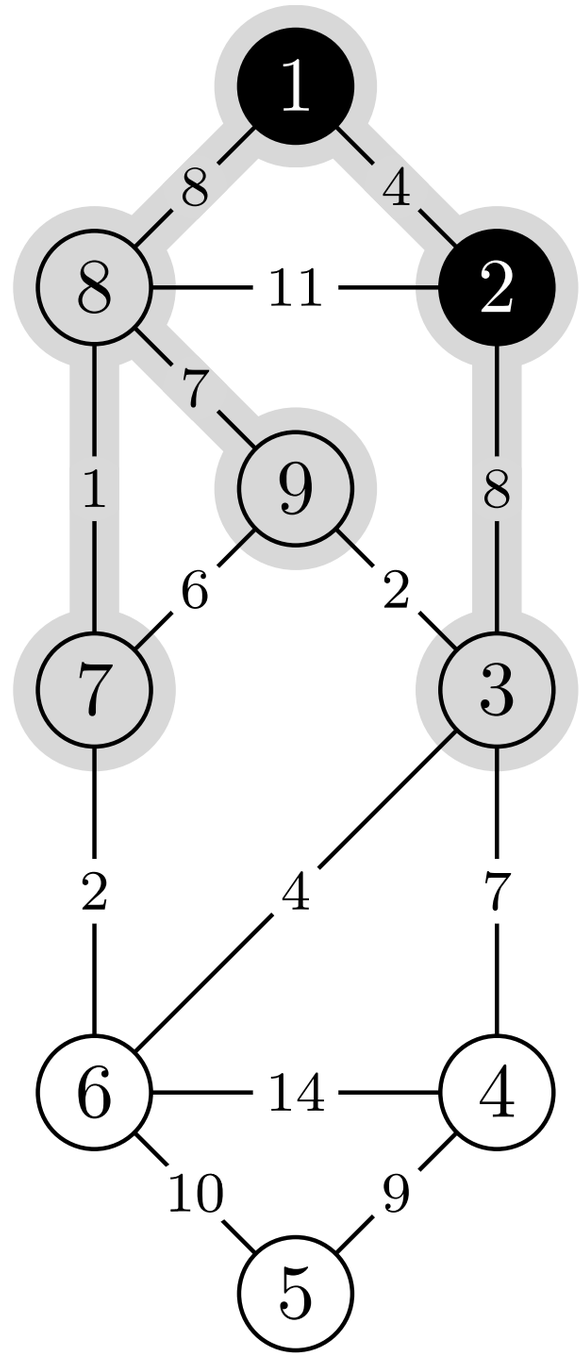
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	—	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 8, 7$

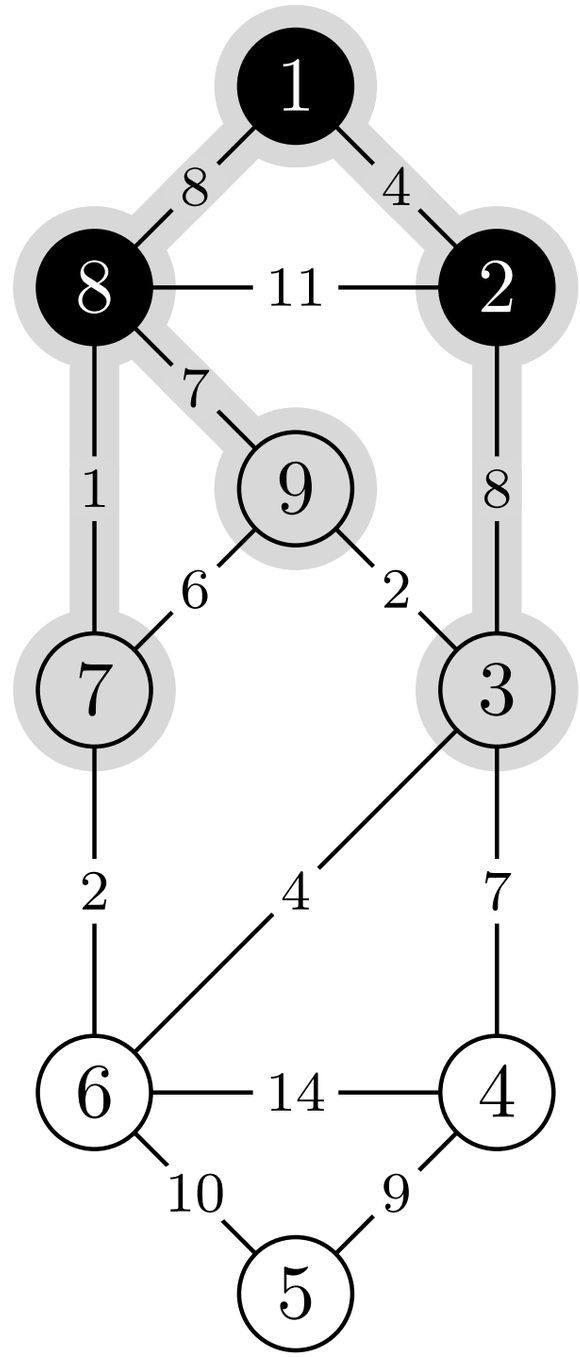
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	∞	8	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 8, -$

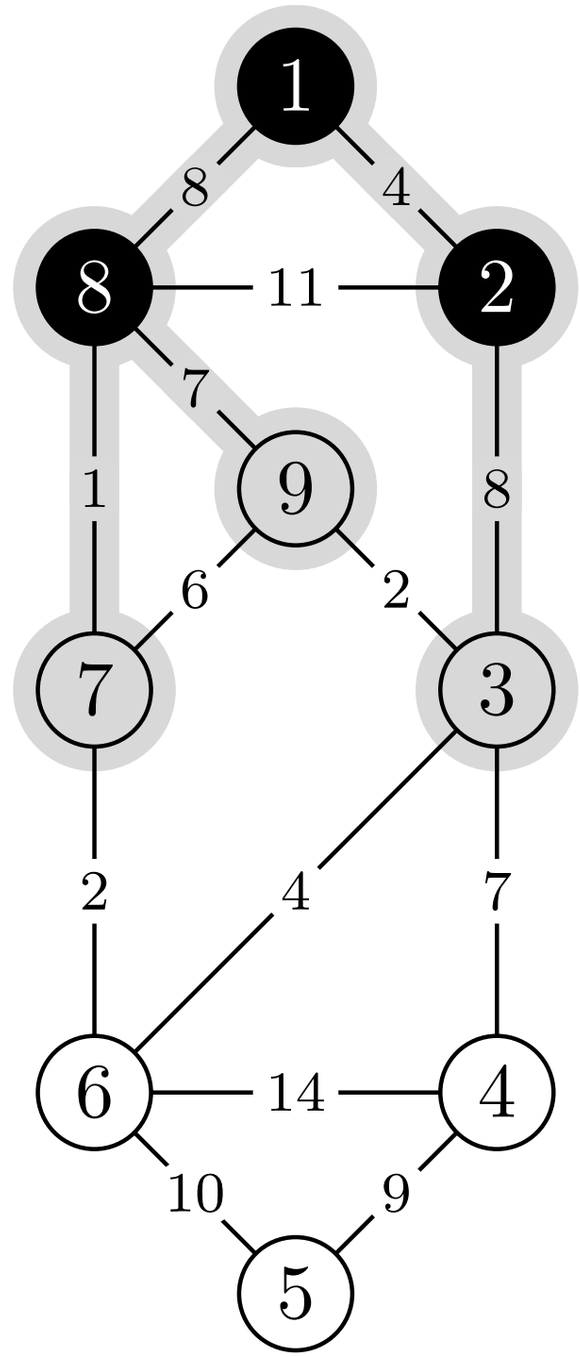
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	1	8	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 7, -$

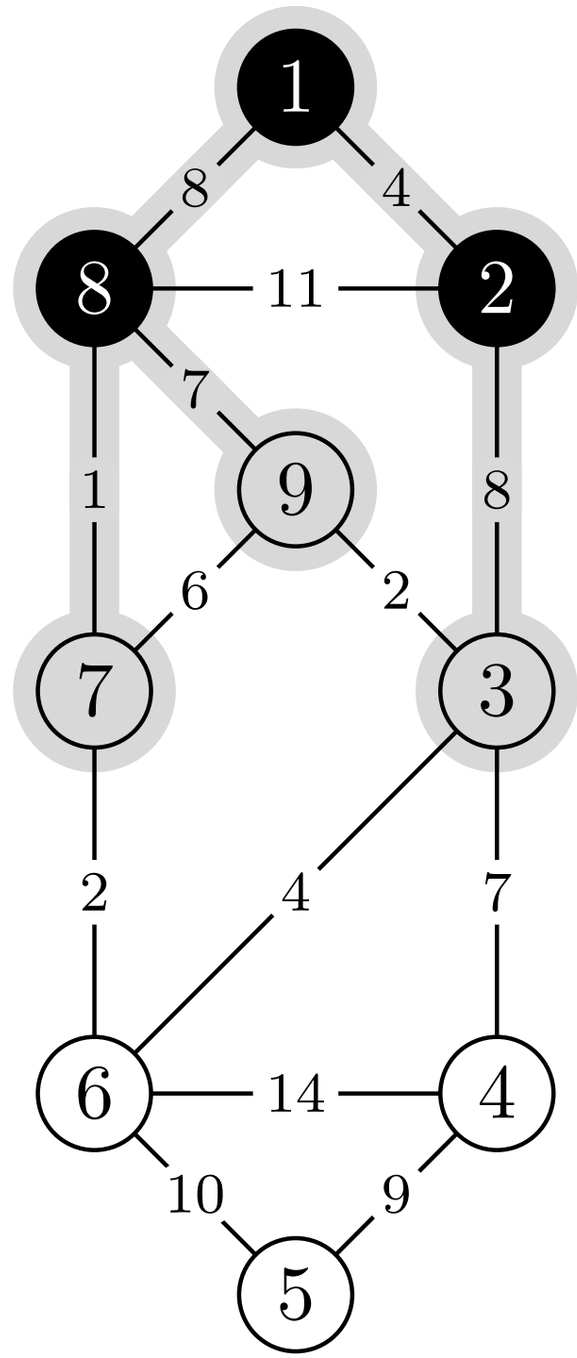
	key	π
1	0	—
2	4	1
3	8	2
4	∞	—
5	∞	—
6	∞	—
7	1	8
8	8	1
9	7	8



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 7, 8$

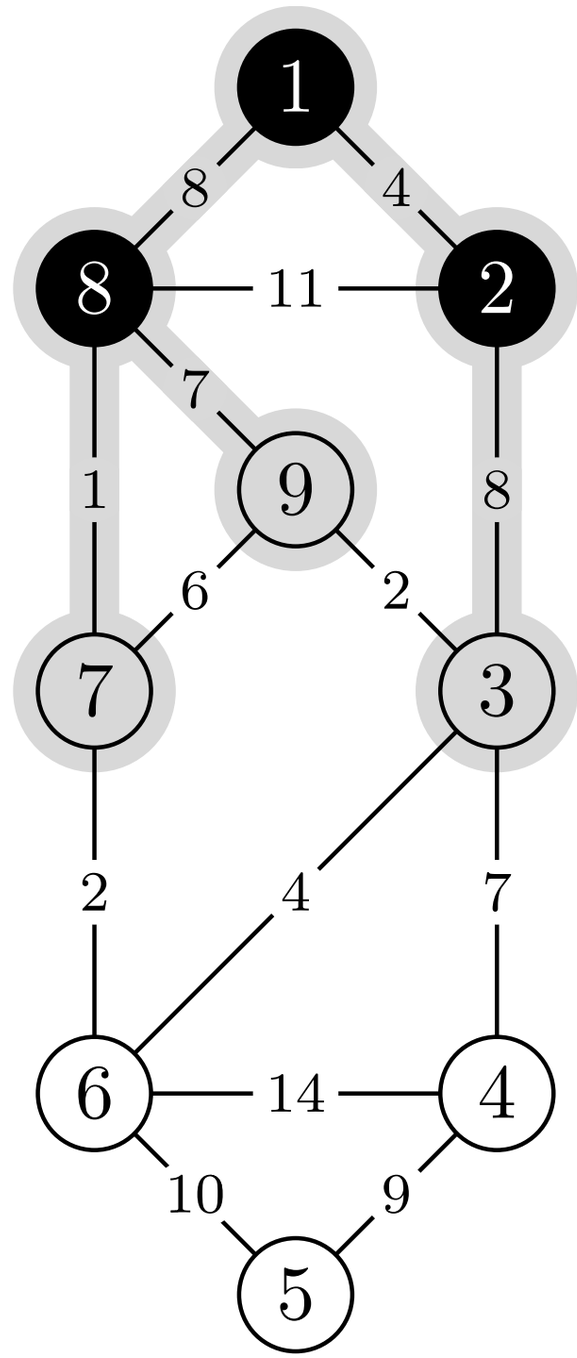
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	1	8	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 7, 8$

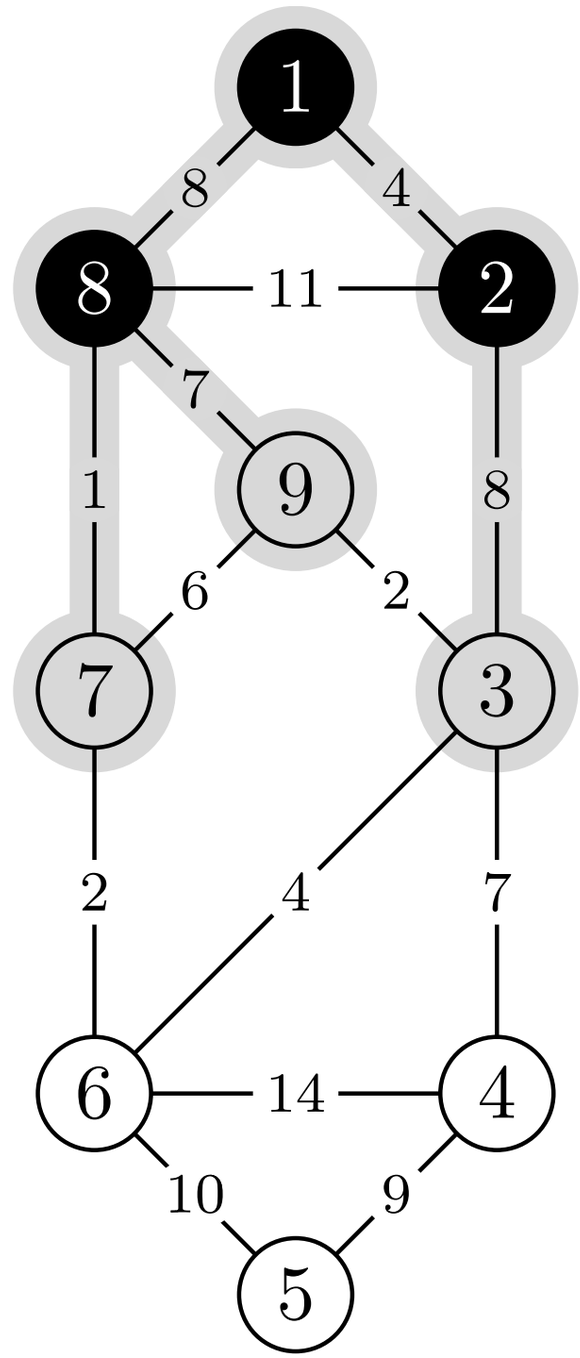
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	1	8	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 7, 9$

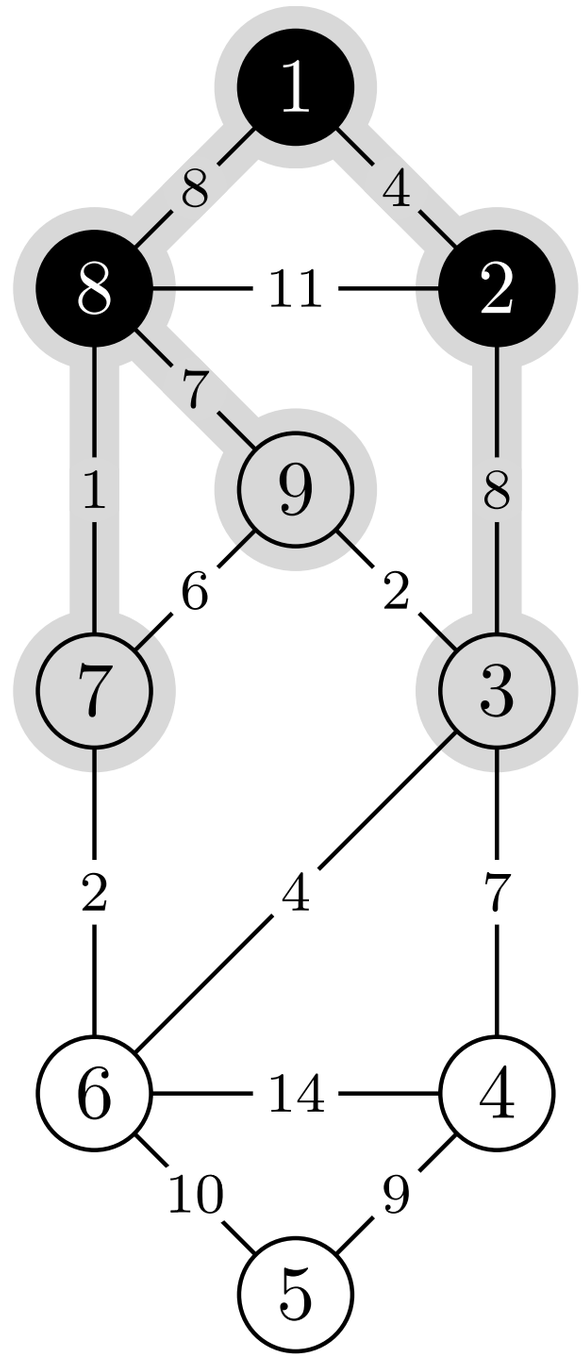
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	1	8	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 7, 9$

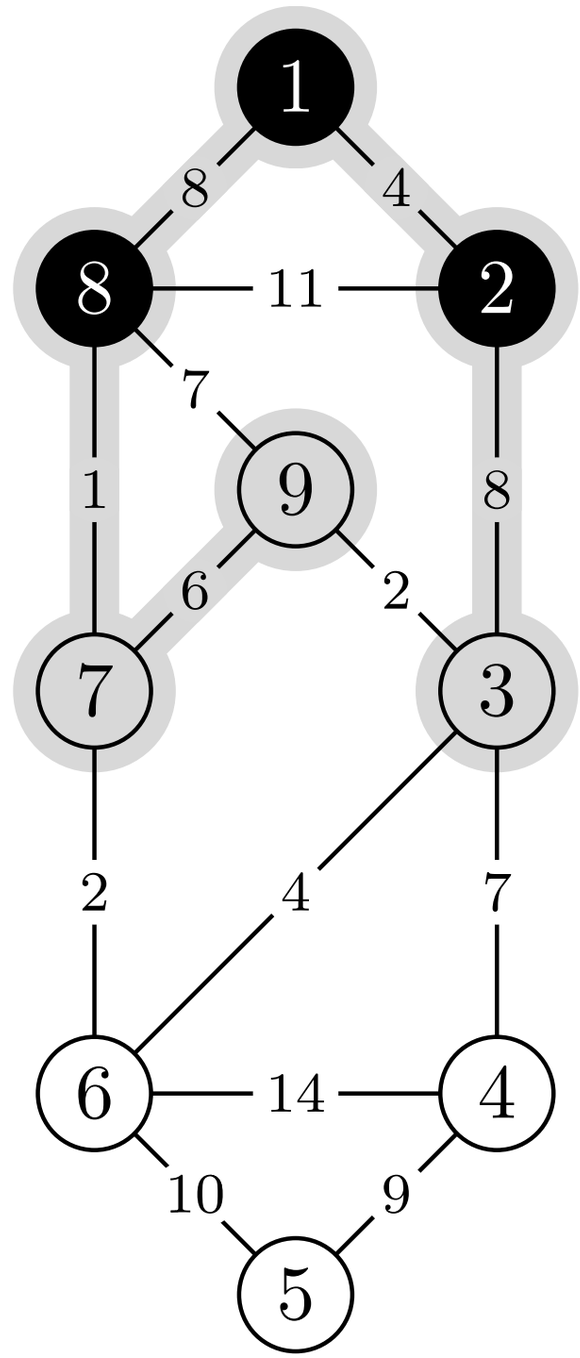
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	1	8	7
8	8	1	8
9	7	8	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 7, 9$

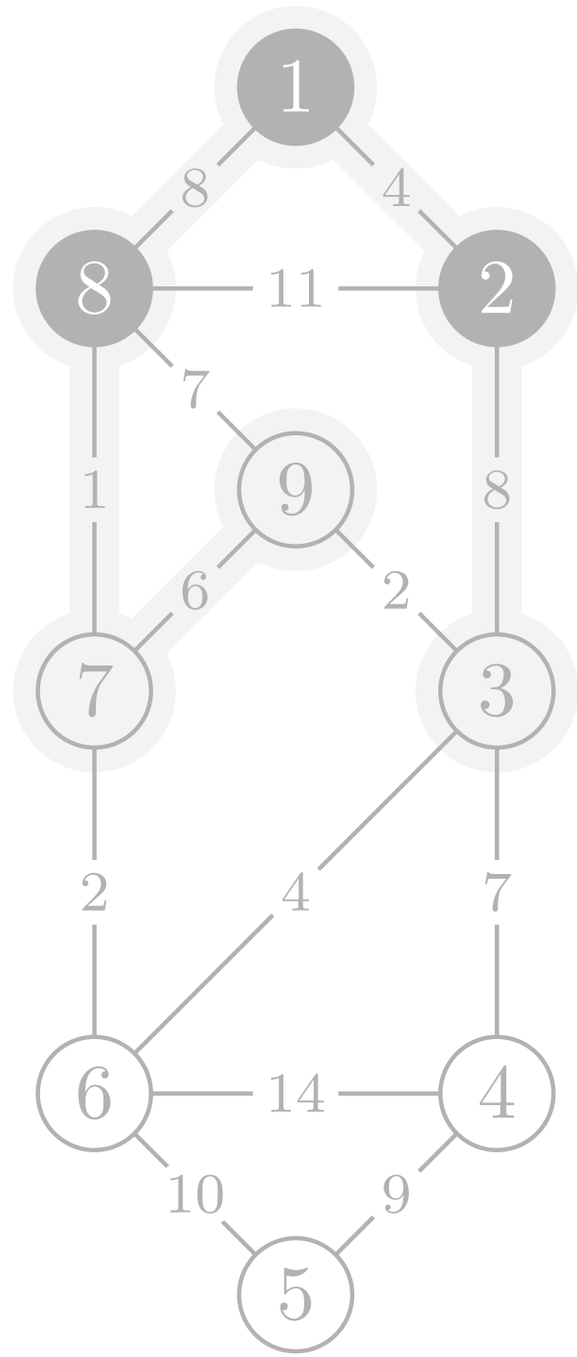
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	1	8	7
8	8	1	8
9	7	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 7, 9$

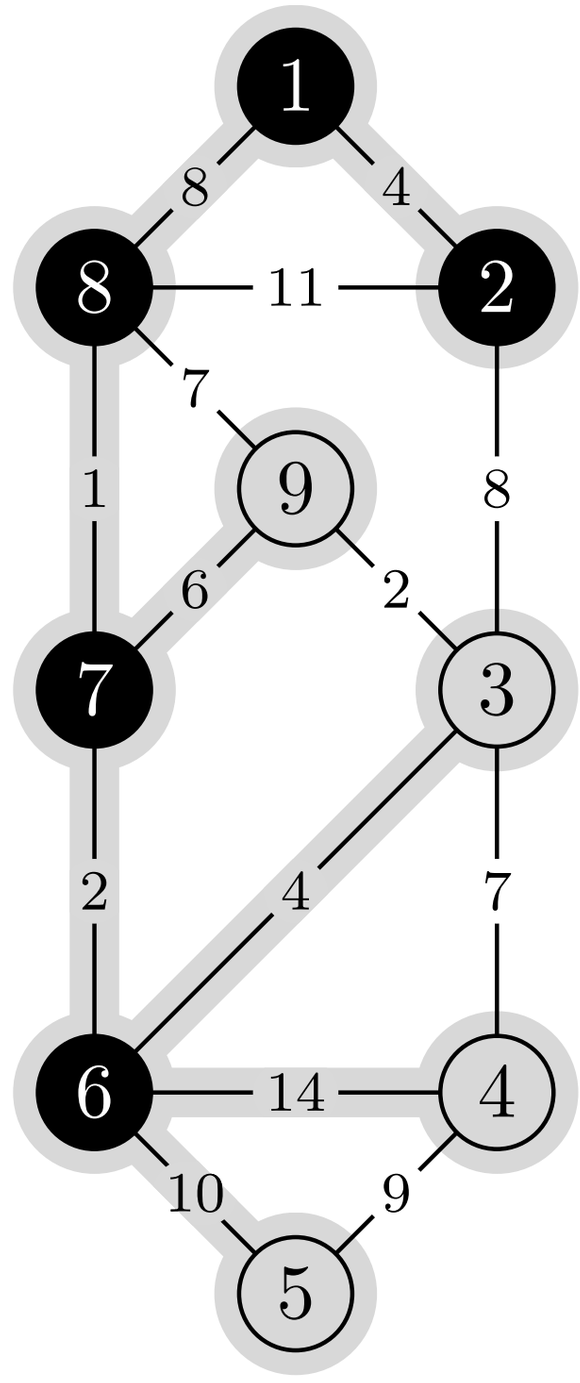
	key	π	
1	0	—	1
2	4	1	2
3	8	2	3
4	∞	—	4
5	∞	—	5
6	∞	—	6
7	1	8	7
8	8	1	8
9	6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 6, -$

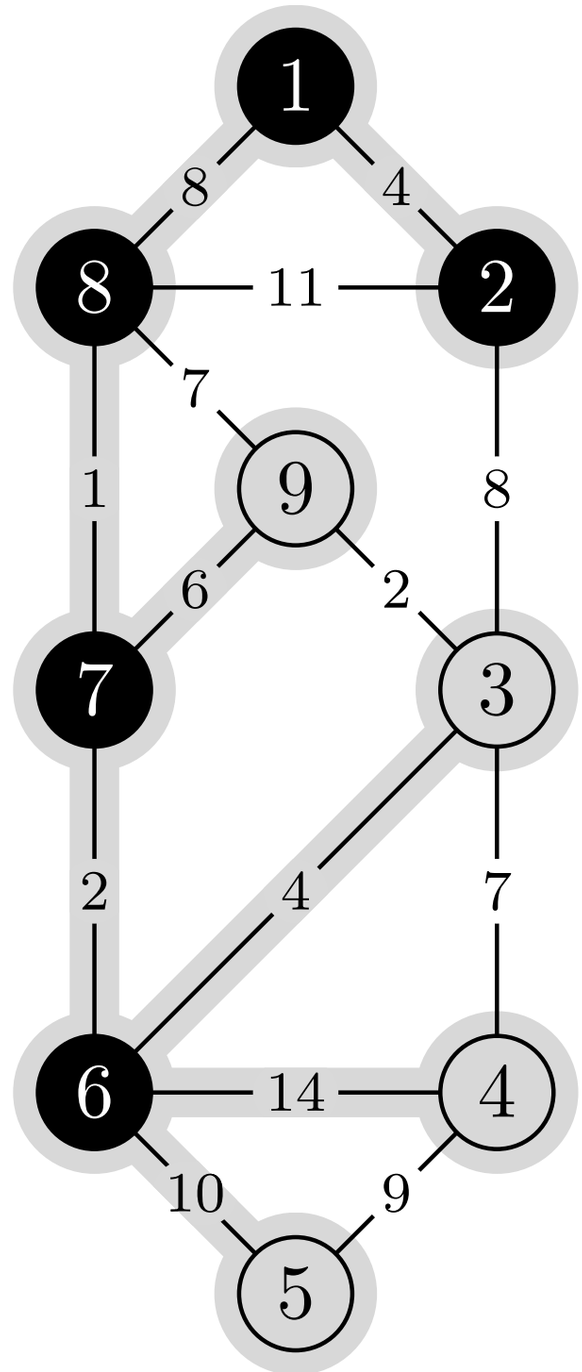
	key	π	
0	—	1
4	1	2
4	6	3
14	6	4
10	6	5
2	7	6
1	8	7
8	1	8
6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 3, -$

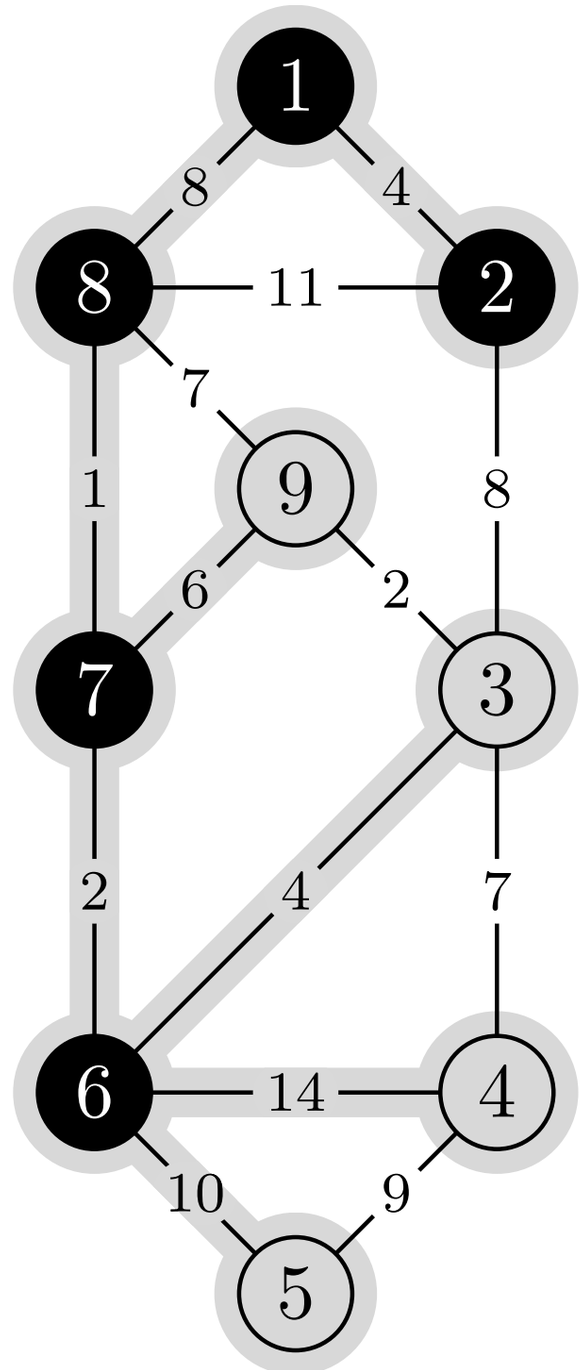
key	π	
0	—	1
4	1	2
4	6	3
14	6	4
10	6	5
2	7	6
1	8	7
8	1	8
6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 4$

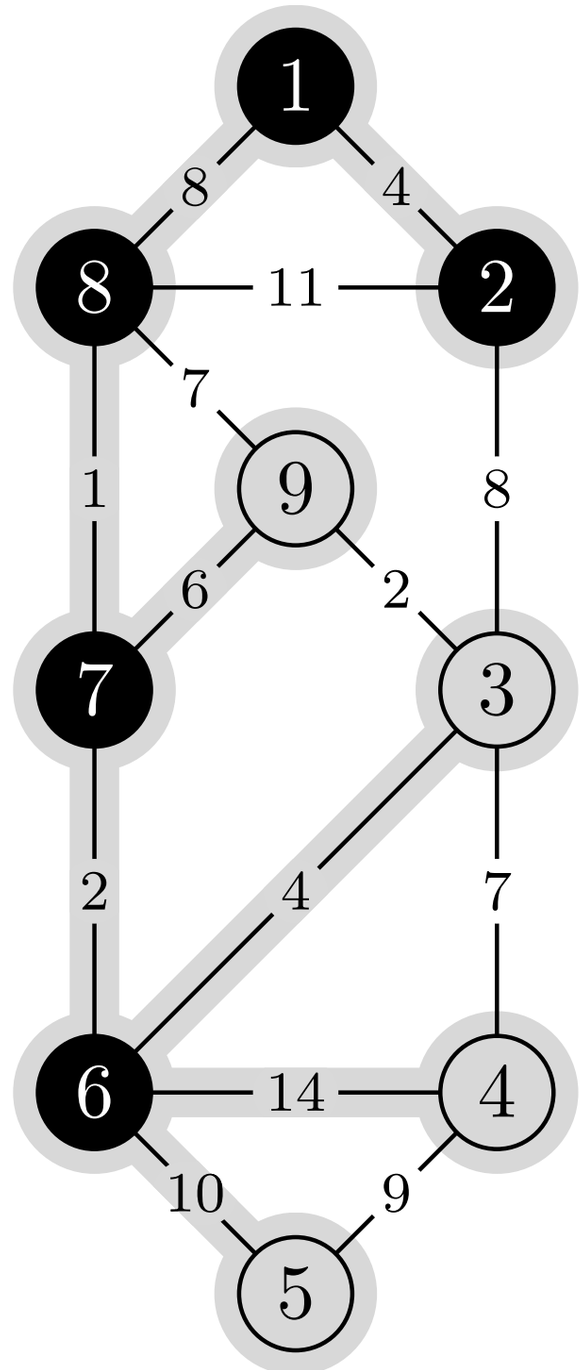
	key	π
1	0	—
2	4	1
3	4	6
4	14	6
5	10	6
6	2	7
7	1	8
8	8	1
9	6	7



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 4$

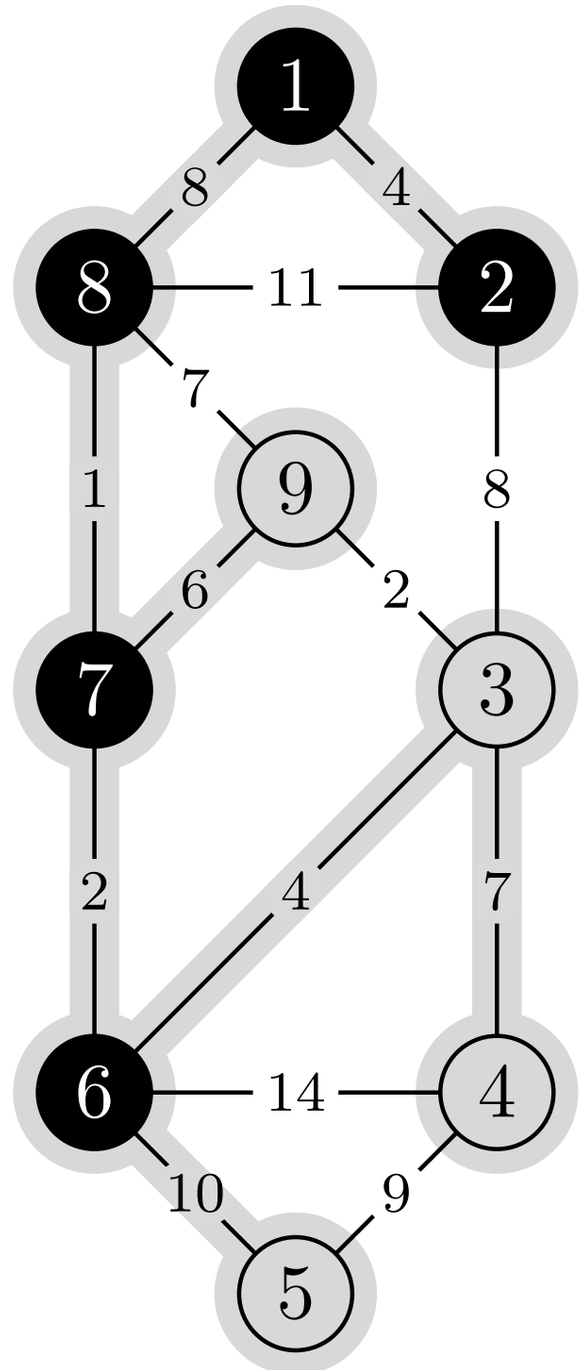
	key	π
1	0	—
2	4	1
3	4	6
4	14	6
5	10	6
6	2	7
7	1	8
8	8	1
9	6	7



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 4$

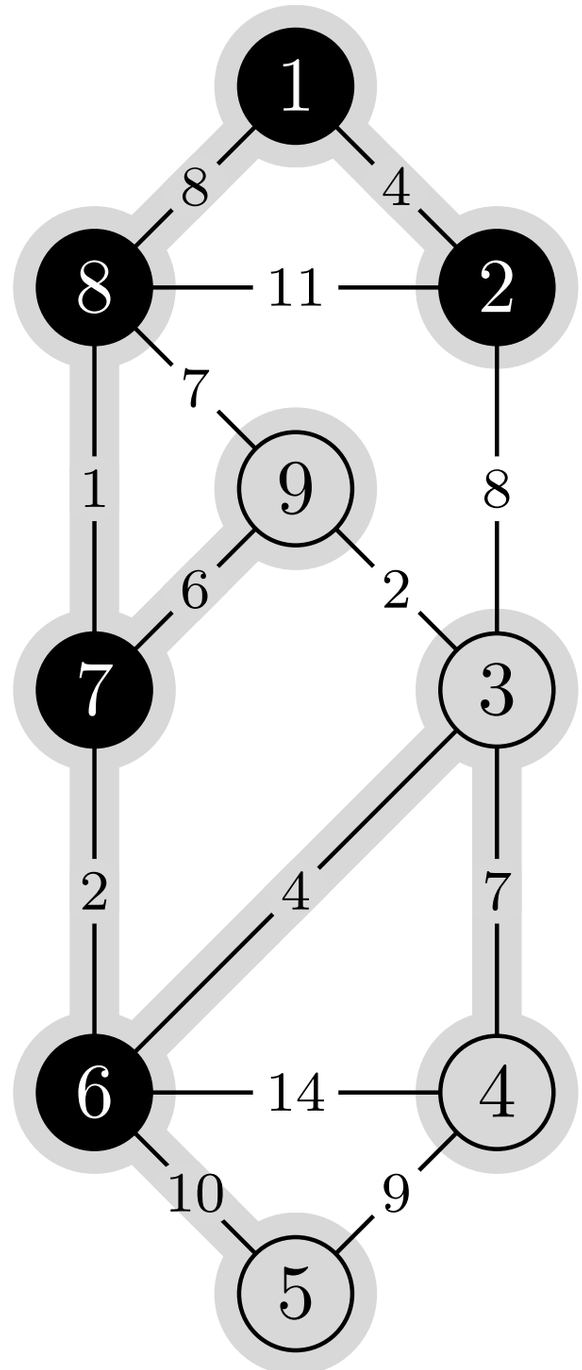
	key	π	
1	0	—	1
2	4	1	2
3	4	6	3
4	14	3	4
5	10	6	5
6	2	7	6
7	1	8	7
8	8	1	8
9	6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 4$

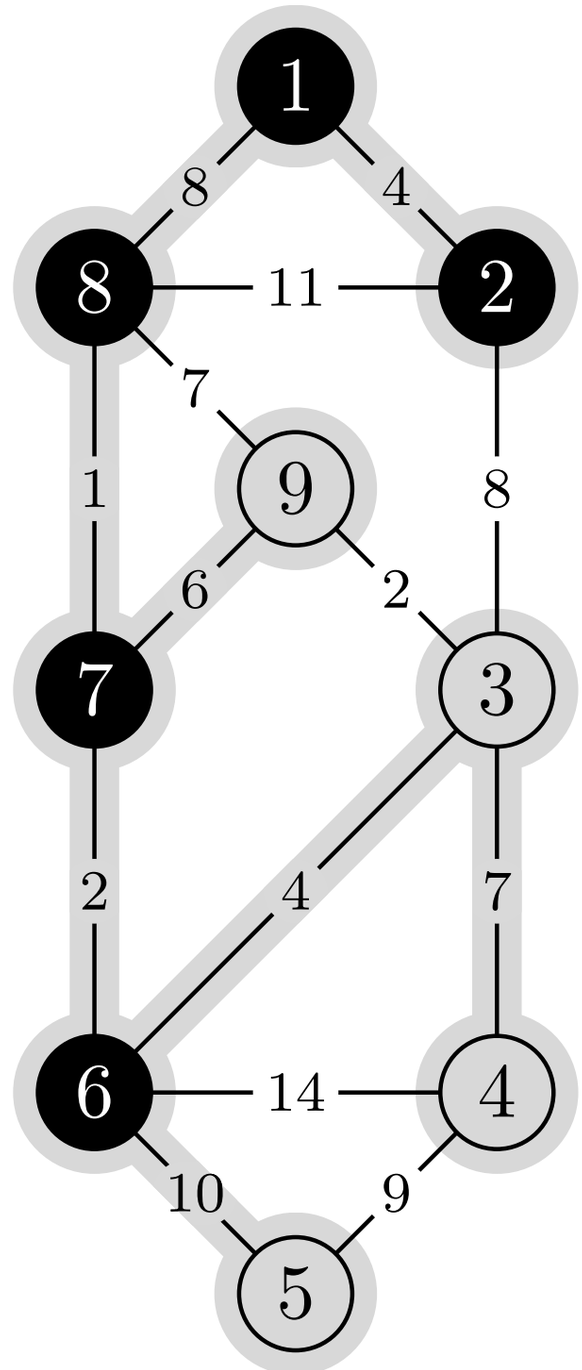
key	π	
0	—	1
4	1	2
4	6	3
7	3	4
10	6	5
2	7	6
1	8	7
8	1	8
6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 6$

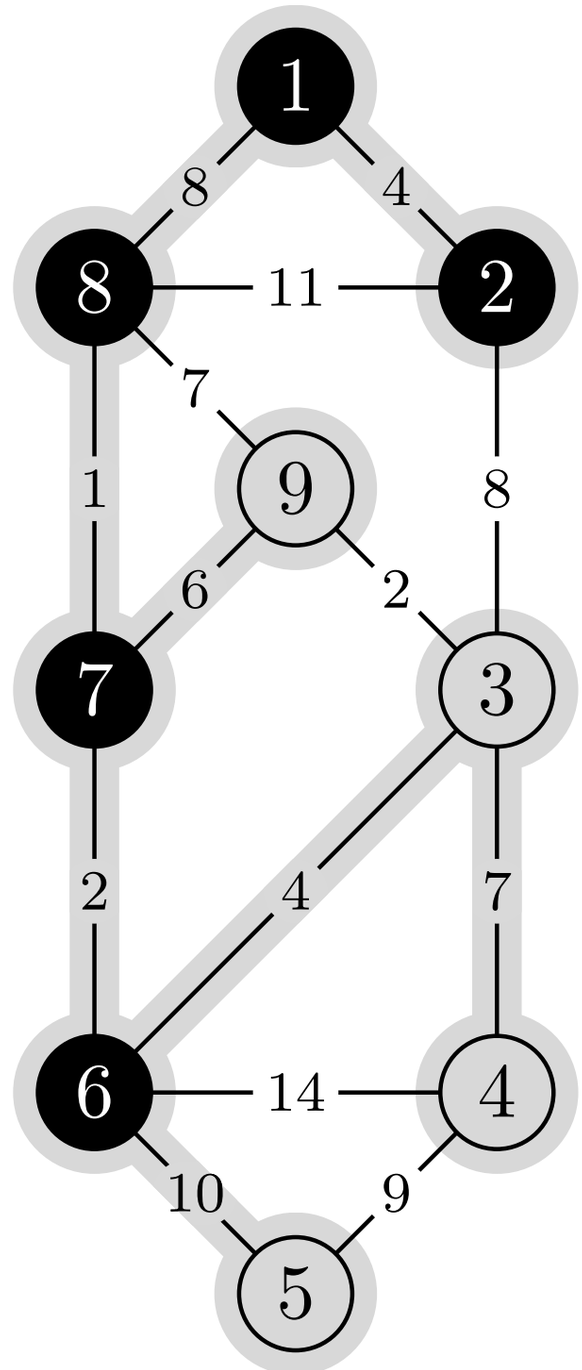
	key	π	
0	—	1
4	1	2
4	6	3
7	3	4
10	6	5
2	7	6
1	8	7
8	1	8
6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 6$

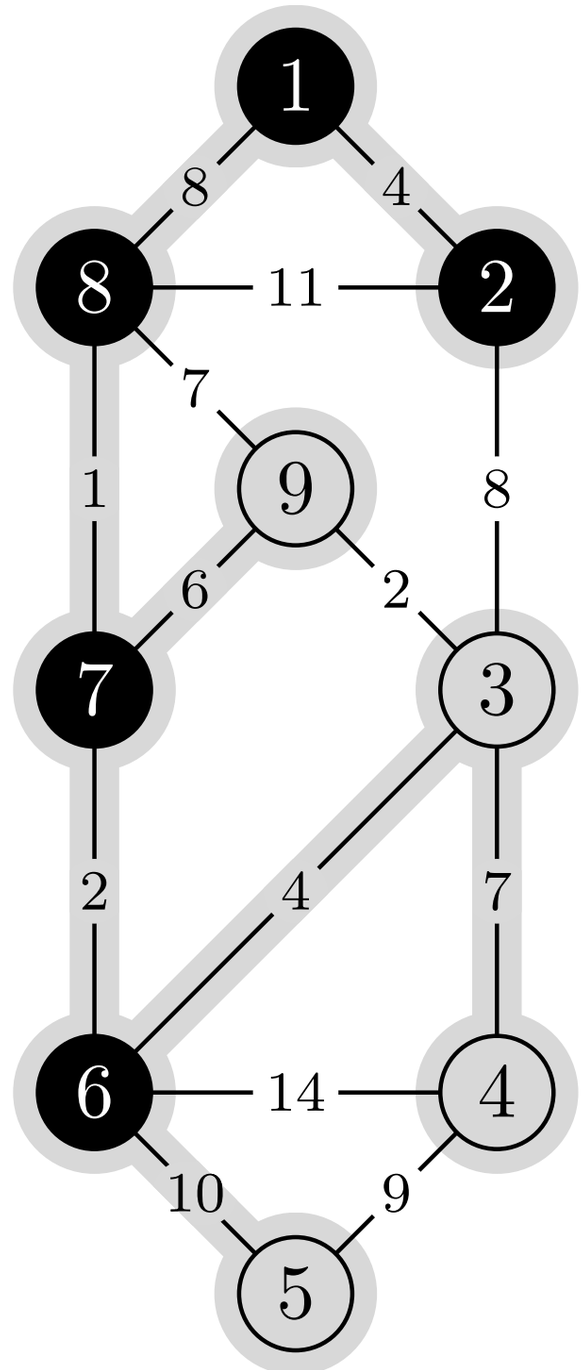
	key	π	
0	—	1
4	1	2
4	6	3
7	3	4
10	6	5
2	7	6
1	8	7
8	1	8
6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 3, 9$

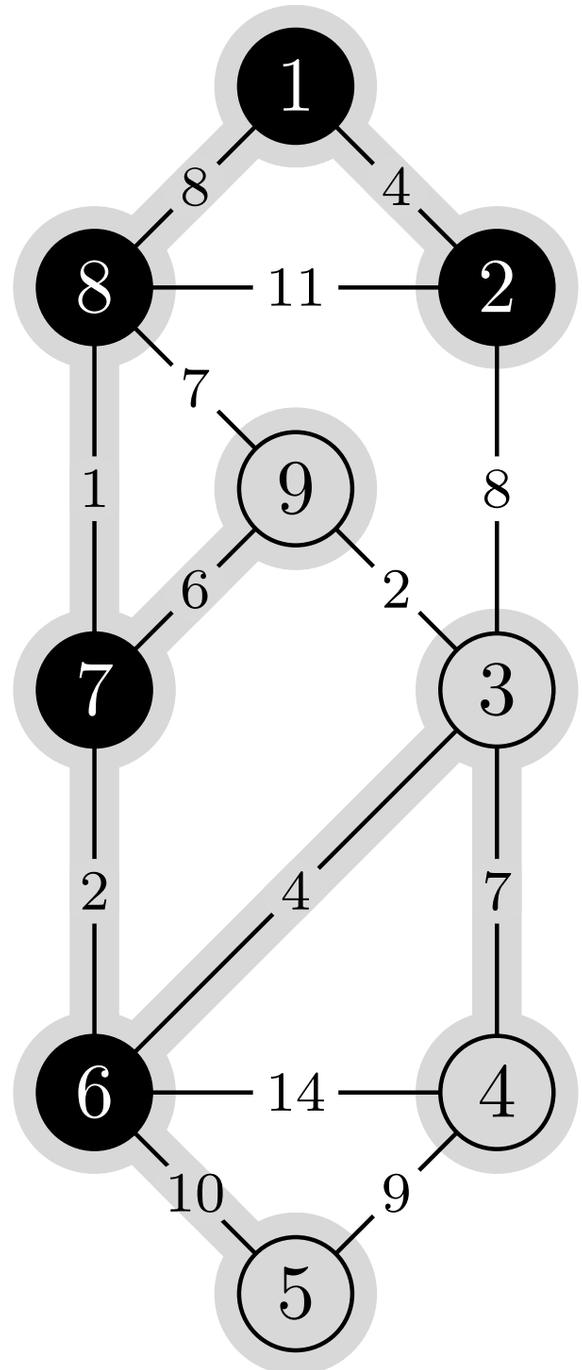
	key	π	
0	—	1
4	1	2
4	6	3
7	3	4
10	6	5
2	7	6
1	8	7
8	1	8
6	7	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 9$

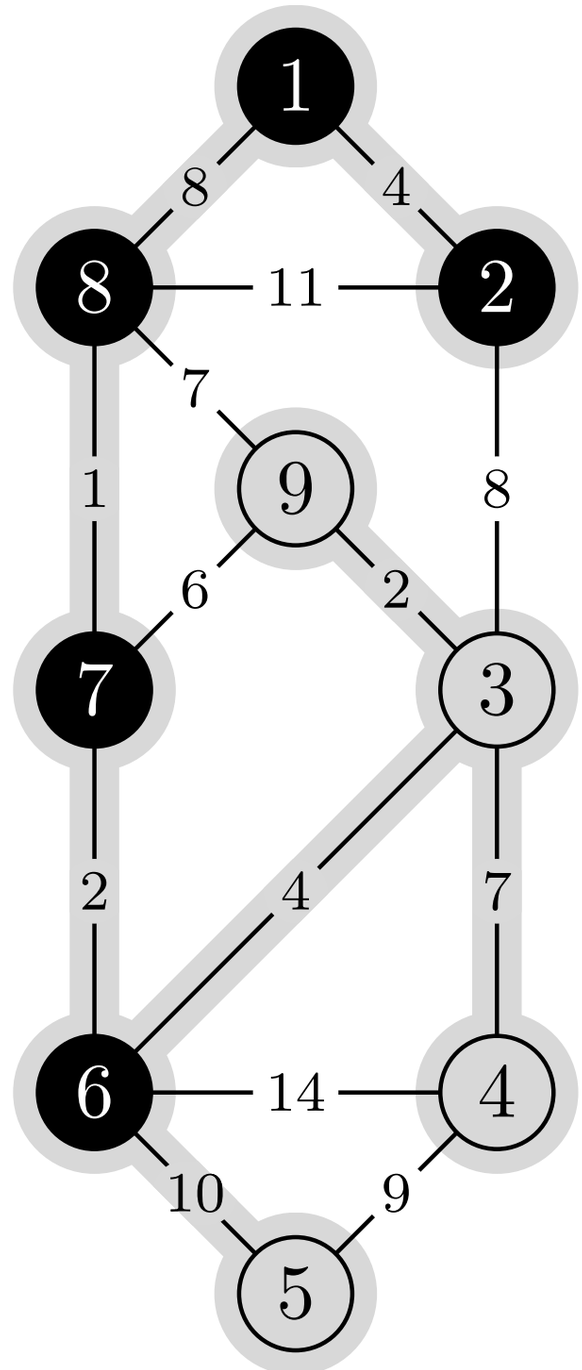
	key	π	
0	—		1
4	1		2
4	6		3
7	3		4
10	6		5
2	7		6
1	8		7
8	1		8
6	7		9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 3, 9$

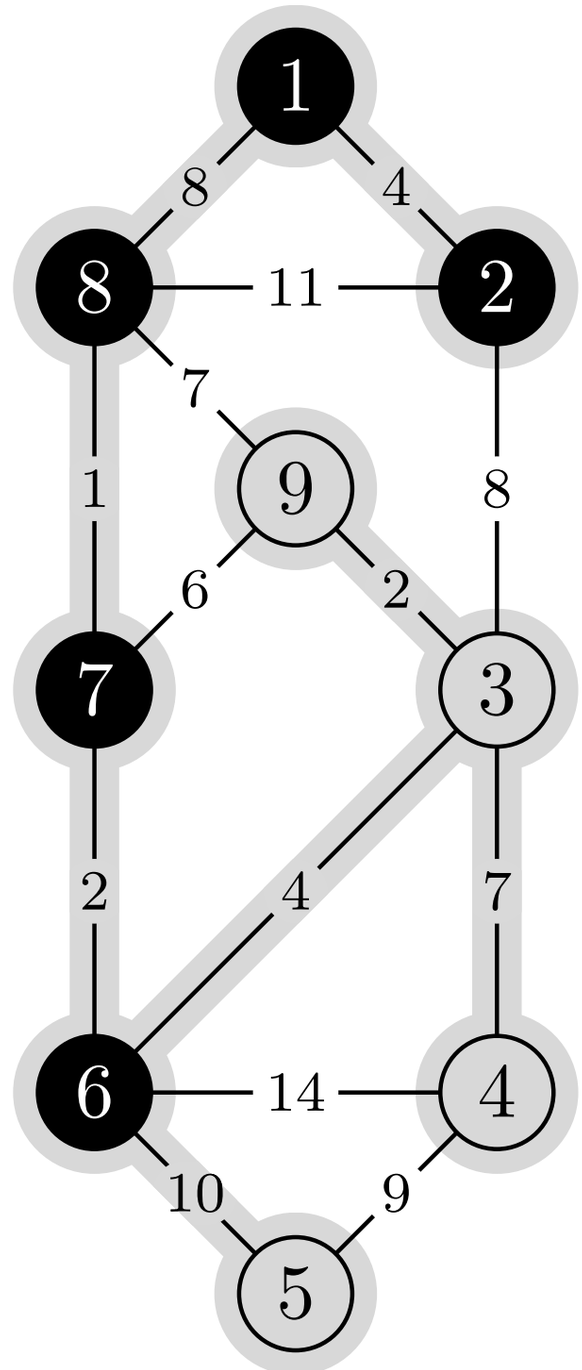
	key	π	
1	0	—	1
2	4	1	2
3	4	6	3
4	7	3	4
5	10	6	5
6	2	7	6
7	1	8	7
8	8	1	8
9	6	3	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 3, 9$

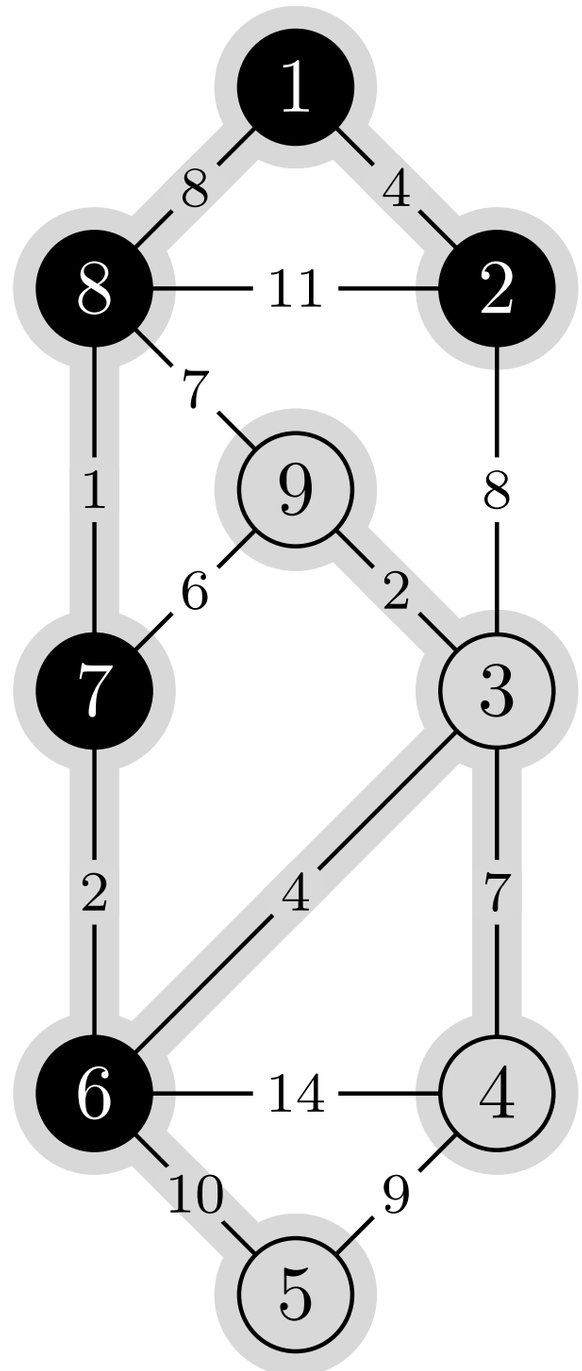
	key	π	
1	0	—	1
2	4	1	2
3	4	6	3
4	7	3	4
5	10	6	5
6	2	7	6
7	1	8	7
8	8	1	8
9	2	3	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 3, 2$

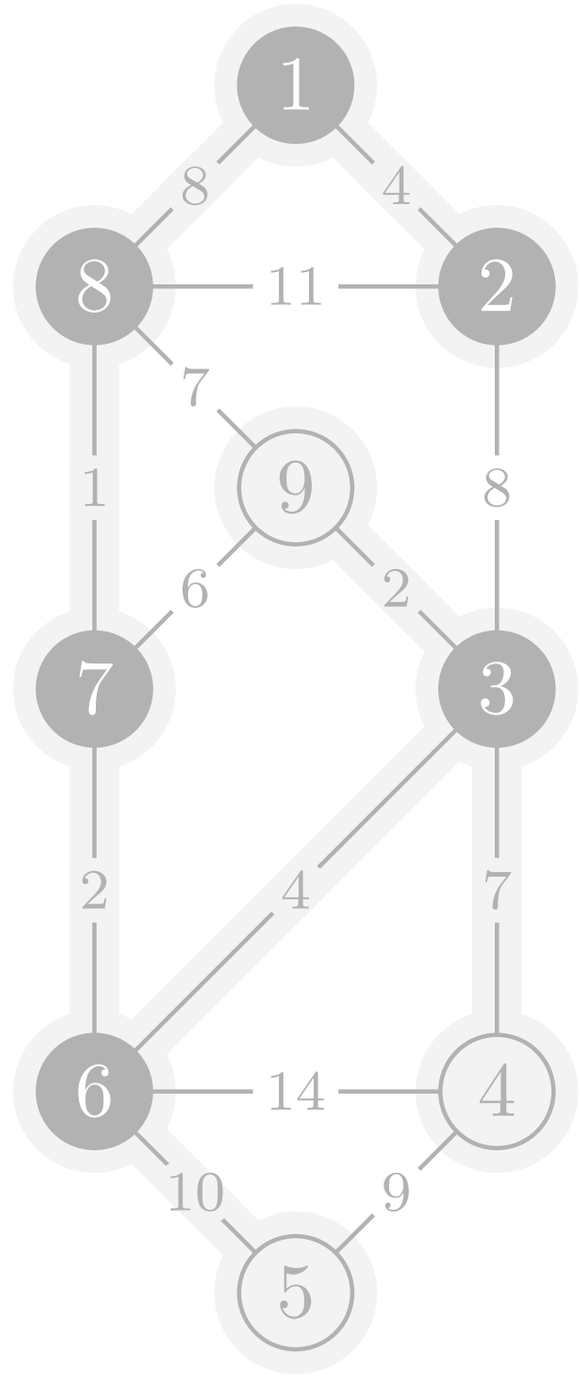
	key	π
1	0	—
2	4	1
3	4	6
4	7	3
5	10	6
6	2	7
7	1	8
8	8	1
9	2	3



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 3, -$

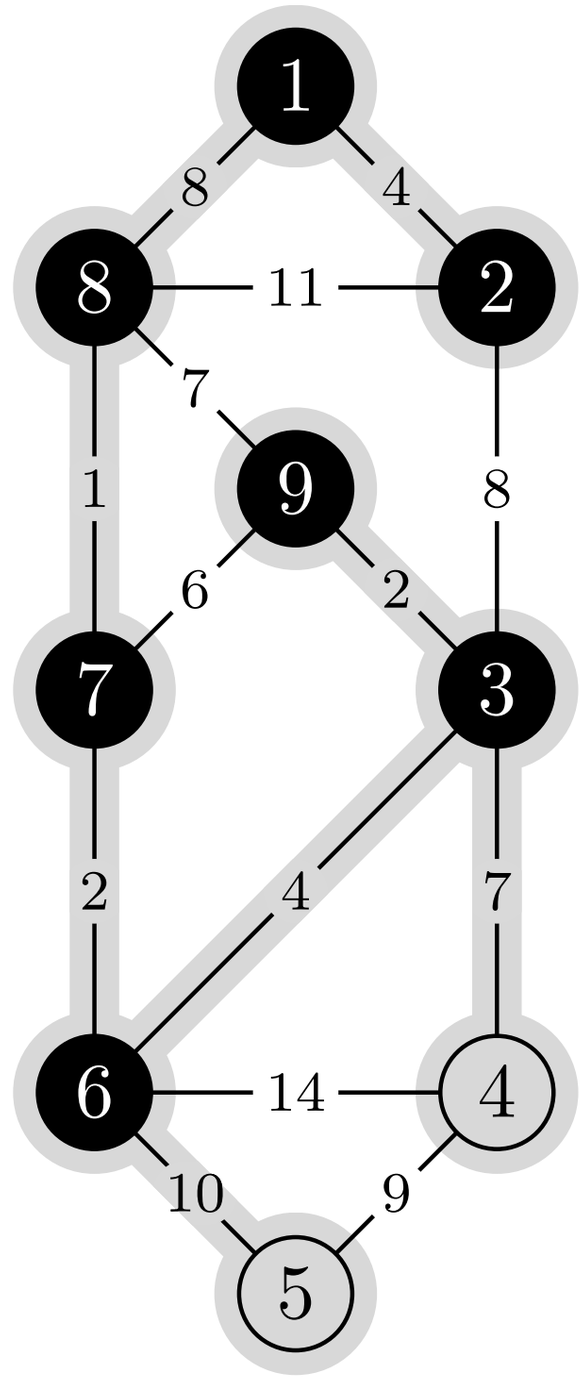
	key	π	
0	—	1
4	1	2
4	6	3
7	3	4
10	6	5
2	7	6
1	8	7
8	1	8
2	3	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 9, -$

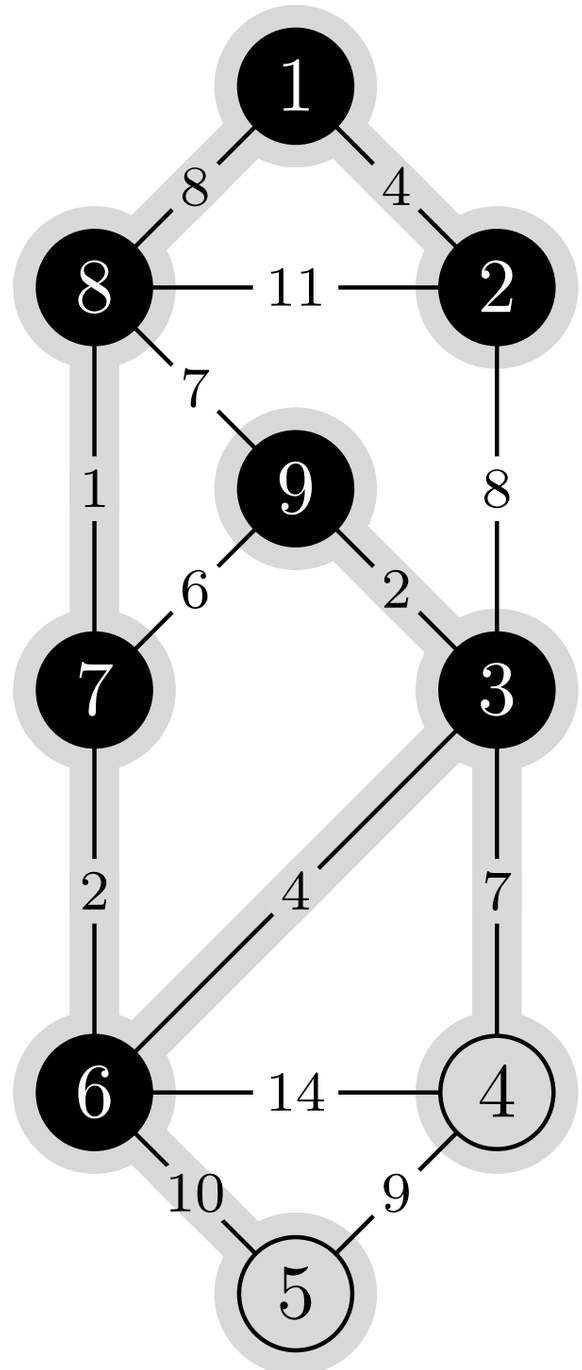
	key	π	
0	—	1
4	1	2
4	6	3
7	3	4
10	6	5
2	7	6
1	8	7
8	1	8
2	3	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 4, -$

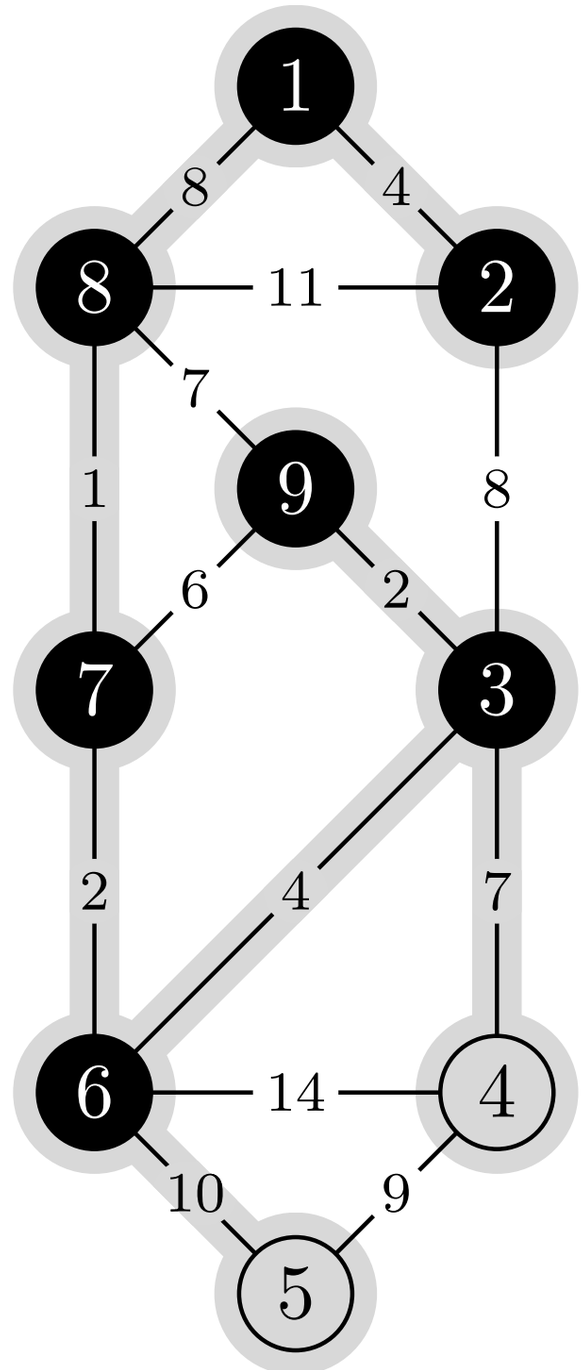
	key	π	
1	0	–	1
2	4	1	2
3	4	6	3
4	7	3	4
5	10	6	5
6	2	7	6
7	1	8	7
8	8	1	8
9	2	3	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 4, 5$

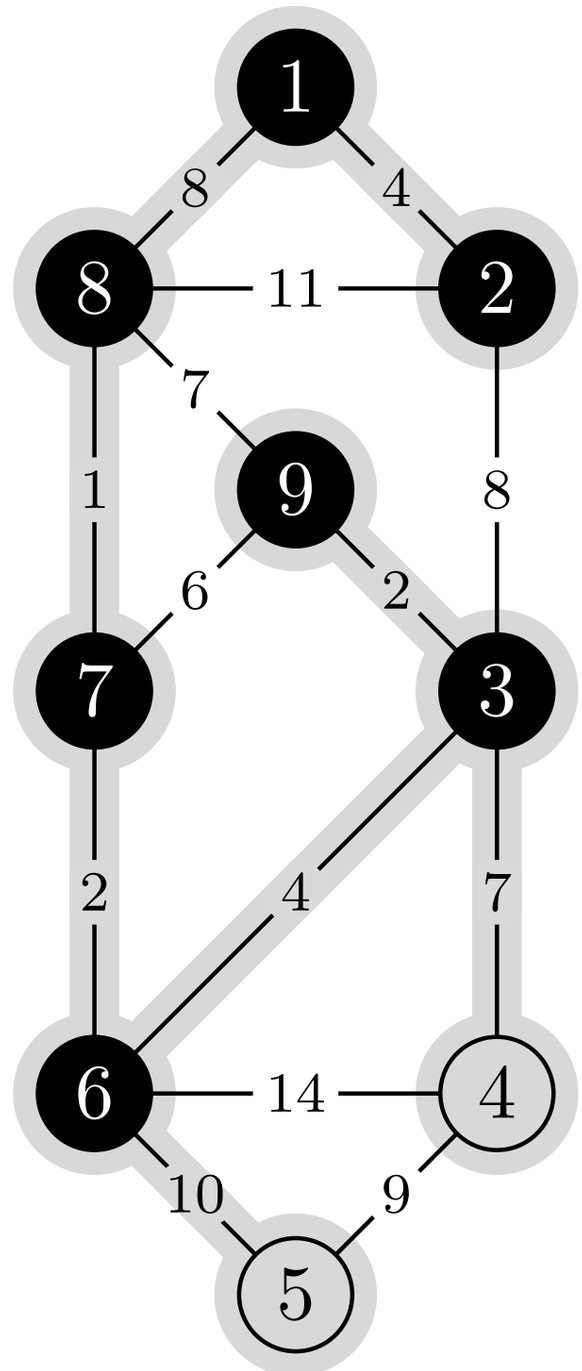
	key	π
1	0	—
2	4	1
3	4	6
4	7	3
5	10	6
6	2	7
7	1	8
8	8	1
9	2	3



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 4, 5$

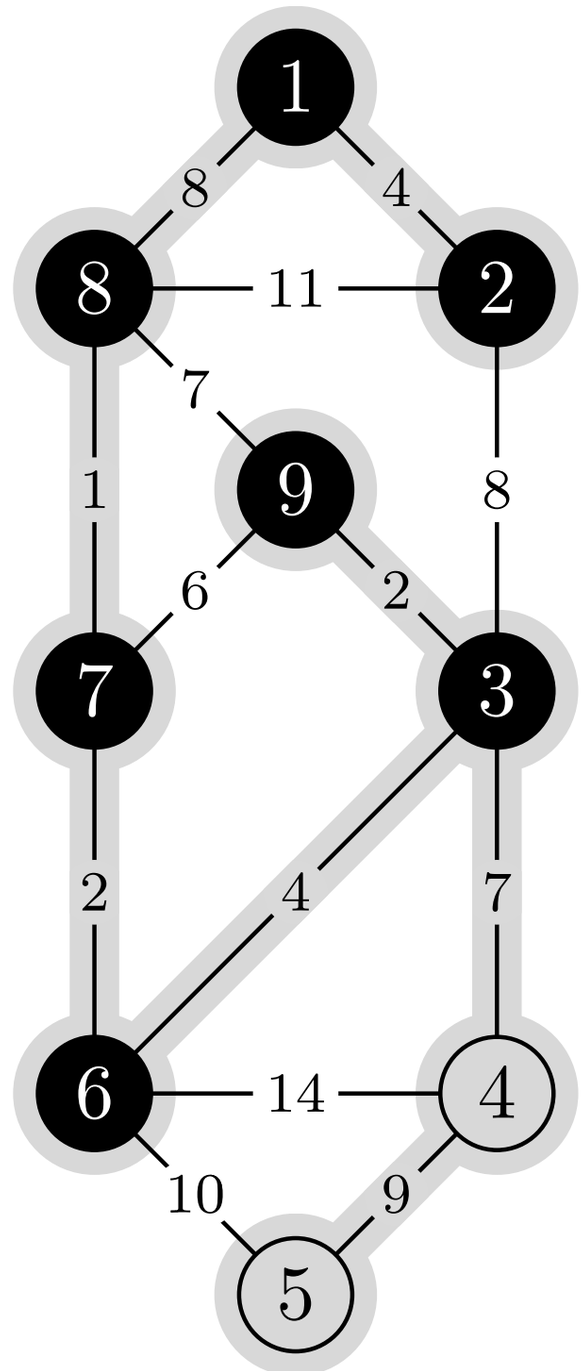
	key	π
1	0	—
2	4	1
3	4	6
4	7	3
5	10	6
6	2	7
7	1	8
8	8	1
9	2	3



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v = 4, 5$

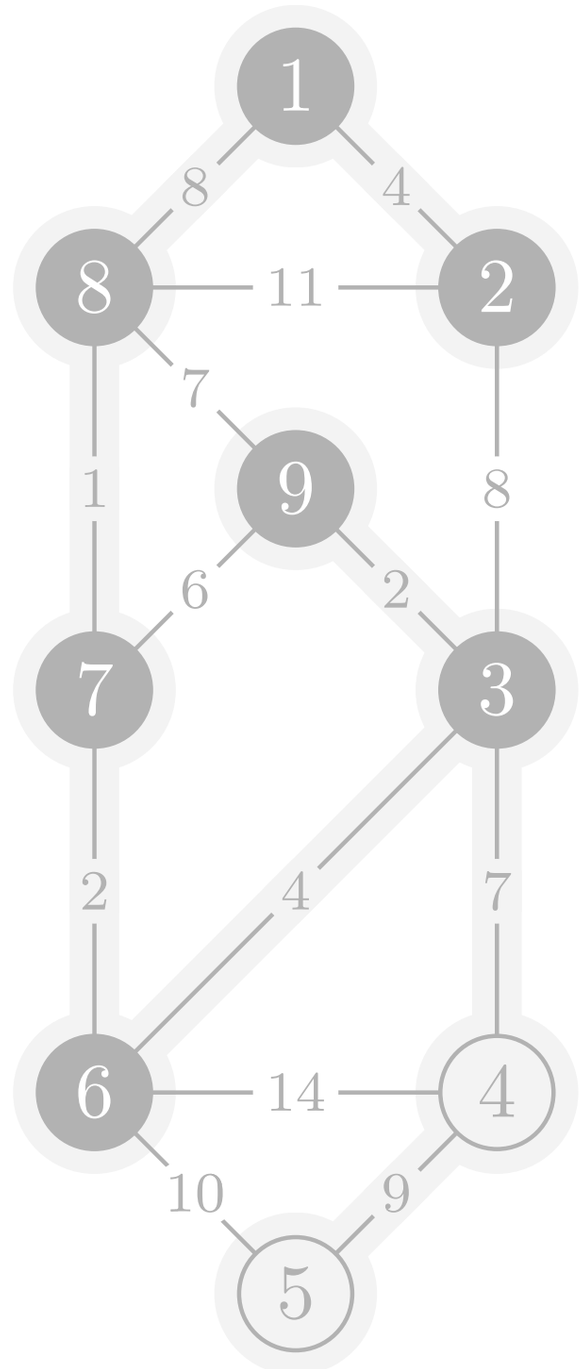
	key	π
1	0	—
2	4	1
3	4	6
4	7	3
5	10	4
6	2	7
7	1	8
8	8	1
9	2	3



```
MST-Prim(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = Extract-Min(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 4, 5$

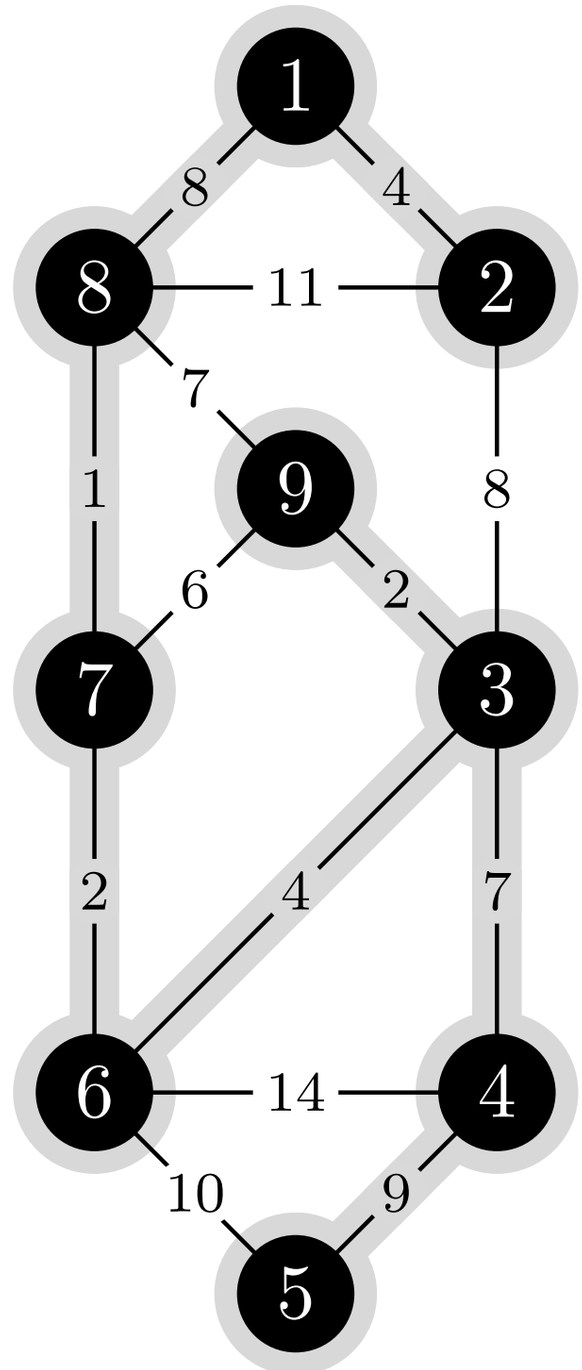
	key	π	
0	—	1
4	1	2
4	6	3
7	3	4
9	4	5
2	7	6
1	8	7
8	1	8
2	3	9



```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

$u, v \;=\; 5, -$

	key	π	
1	0	–	1
2	4	1	2
3	4	6	3
4	7	3	4
5	9	4	5
6	2	7	6
7	1	8	7
8	8	1	8
9	2	3	9



Operasjon	Antall	Kjøretid

Operasjon	Antall	Kjøretid
Build-Min-Heap	1	O(V)

Vi legger alle noder i køen til å begynne med

Operasjon	Antall	Kjøretid
Build-Min-Heap	1	$O(V)$
Extract-Min	V	$O(\lg V)$

Alle noder må hentes ut av køen én gang, før de besøkes

Operasjon	Antall	Kjøretid
Build-Min-Heap	1	$O(V)$
Extract-Min	V	$O(\lg V)$
Decrease-Key	E	$O(\lg V)$

Langs hver kant (u, v) vi finner, oppdatér prioriteten til v

Operasjon	Antall	Kjøretid
Build-Min-Heap	1	$O(V)$
Extract-Min	V	$O(\lg V)$
Decrease-Key	E	$O(\lg V)$

Totalt: $O(E \lg V)$

Operasjon	Antall	Kjøretid
Build-Min-Heap	1	$O(V)$
Extract-Min	V	$O(\lg V)$
Decrease-Key	E	$O(\lg V)$

Totalt: $O(E \lg V)$

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

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

1. Disjunkte mengder

2. Generisk MST

3. Kruskals algoritme

4. Prims algoritme