Kartet er fra Prims artikkel.

Forelesning 9

Minimale spenntrær

Fig. 1 — Example of a shortest connection network.

Backlog

- Kantklassifisering, DFS
- Parentesteoremet
- Topologisk sortering

Minimale spenntrær

- Disjunkte mengder
- Generisk MST
- Kruskals algoritme
- Prims algoritme

Backlog

Kantklassifisering

- **Tre-kanter**

  Kanter i dybde-først-skogen

- **Bakoverkanter**

  Kanter til en forgjenger i DF-skogen

- **Foroverkanter**

  Kanter utenfor DF-skogen to en etterkommer i DF-skogen

- **Krysskanter**

  Alle andre kanter

Kantklassifisering

- **Møter en hvit node**

  Tre-kant

- **Møter en grå node**

  Bakoverkant

- **Møter en svart node:**

  Forover- eller krysskant

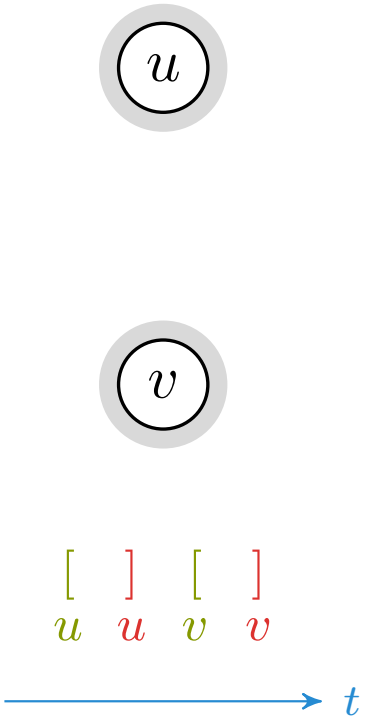
Traversering › DFS ›

Parentesteoremet

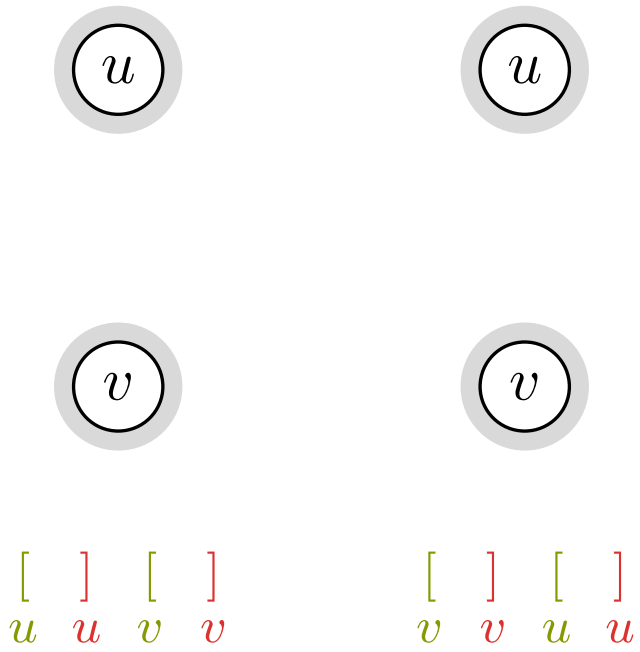
Noder oppdages før og avsluttes etter sine etterkommere

Noder oppdages før og avsluttes etter sine etterkommere

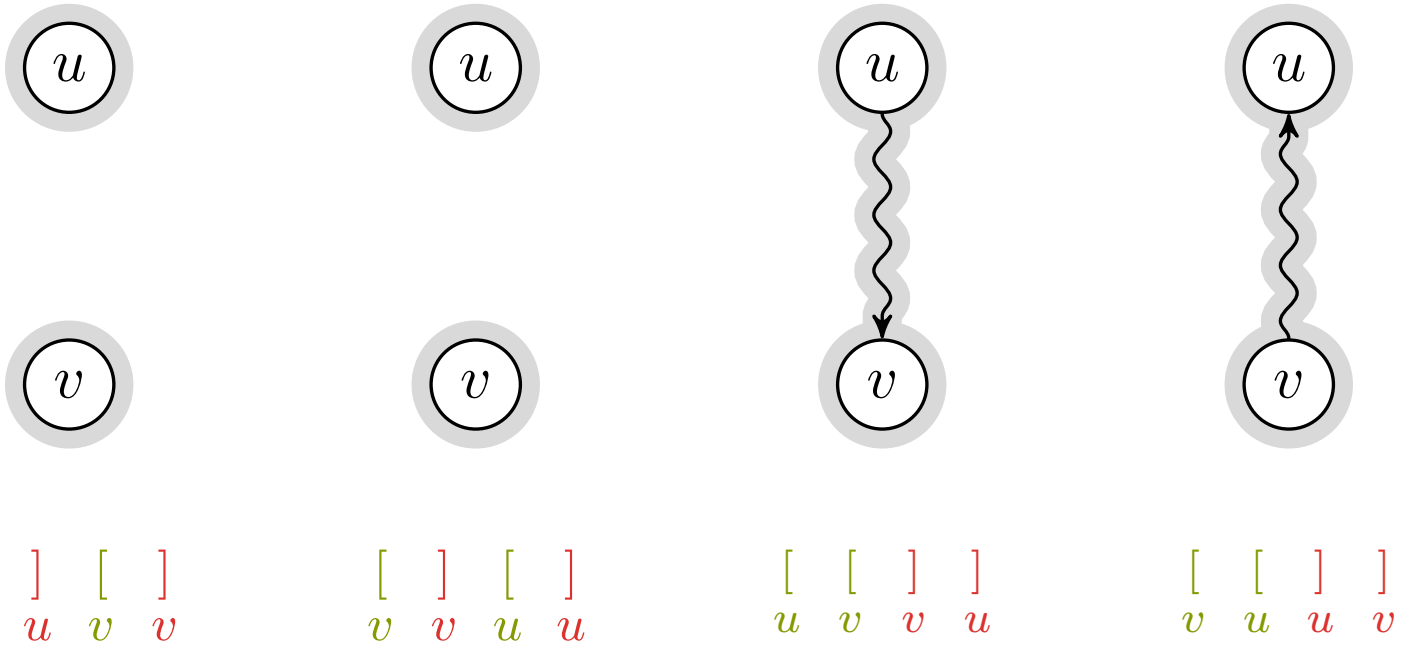
Noder oppdages før og avsluttes etter sine etterkommere



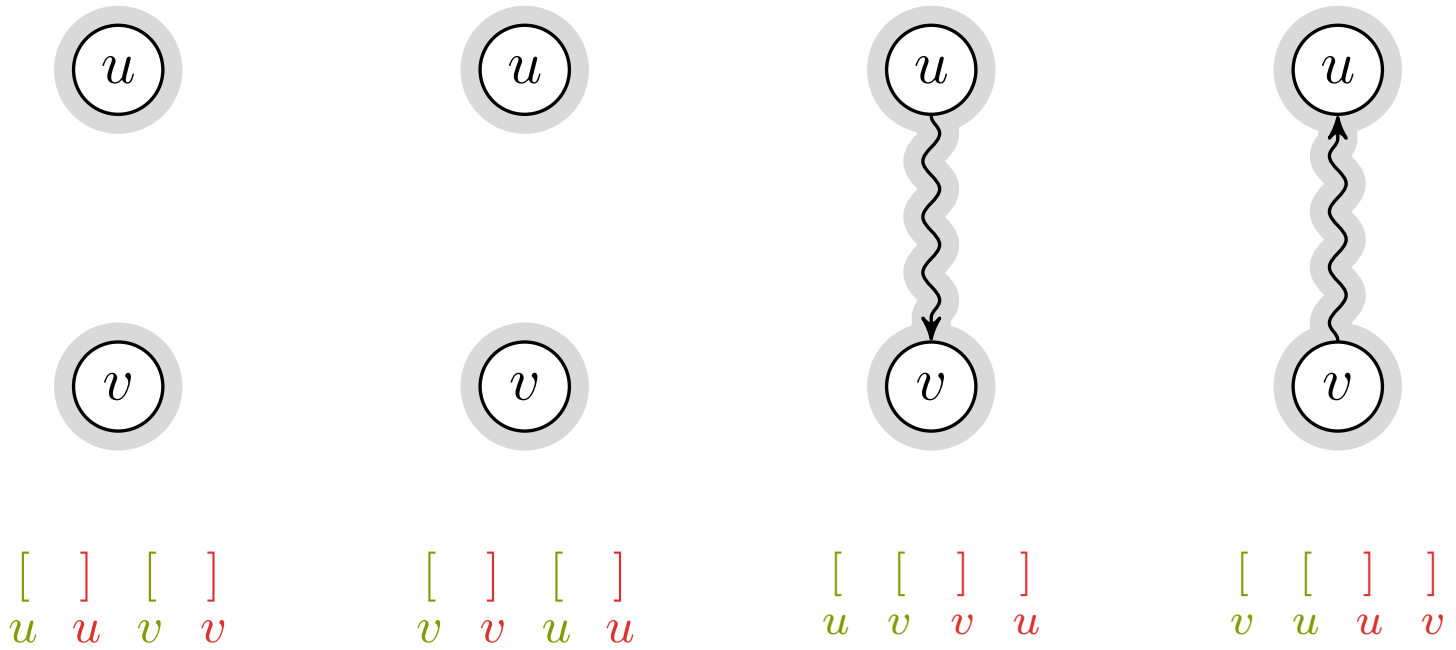
Noder oppdages før og avsluttes etter sine etterkommere



Noder oppdages før og avsluttes etter sine etterkommere



Noder oppdages før og avsluttes etter sine etterkommere

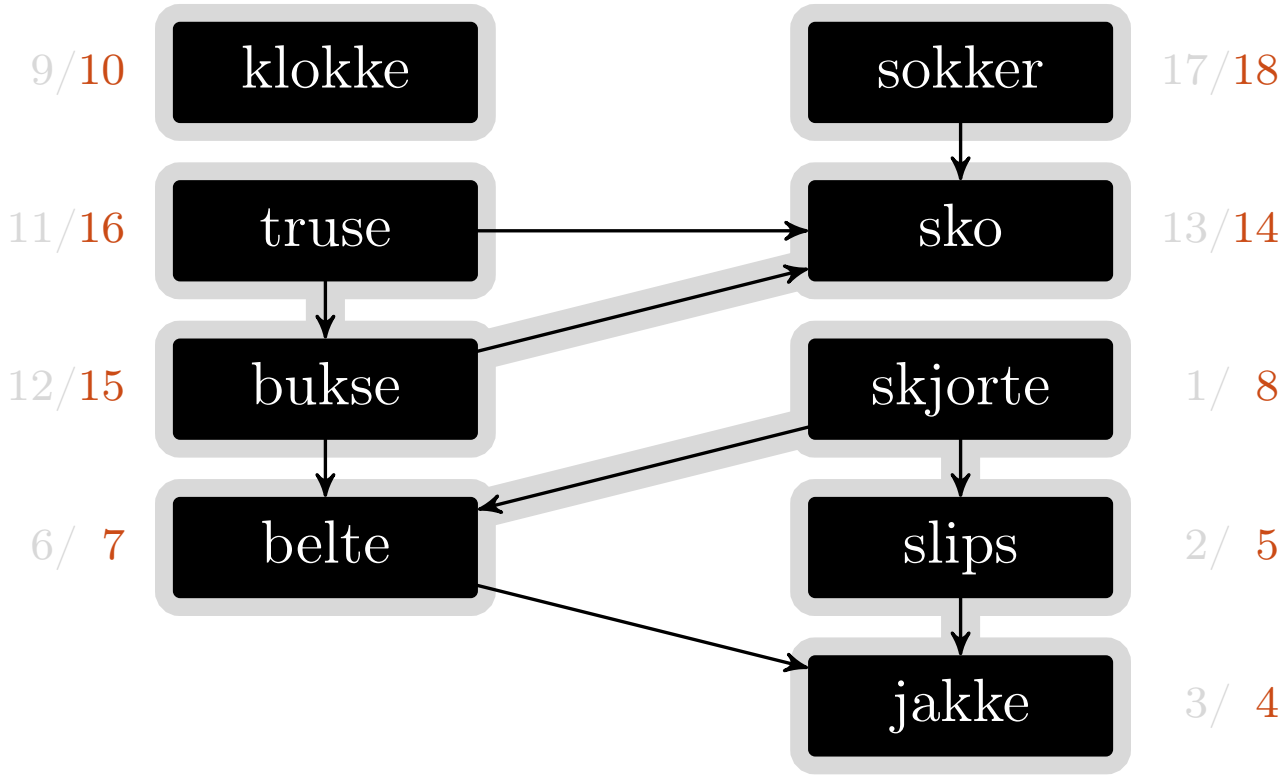


Dette er de eneste mulighetene!

Topologisk sortering

Topologisk sortering

- **Gir nodene en rekkefølge**
- **Foreldre før barn**
- **Evt.: Alle kommer etter avhengigheter**
- **Krever DAG (dvs. velfundert)!**


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

3/ 4

2/ 5

6/ 7

1/ 8

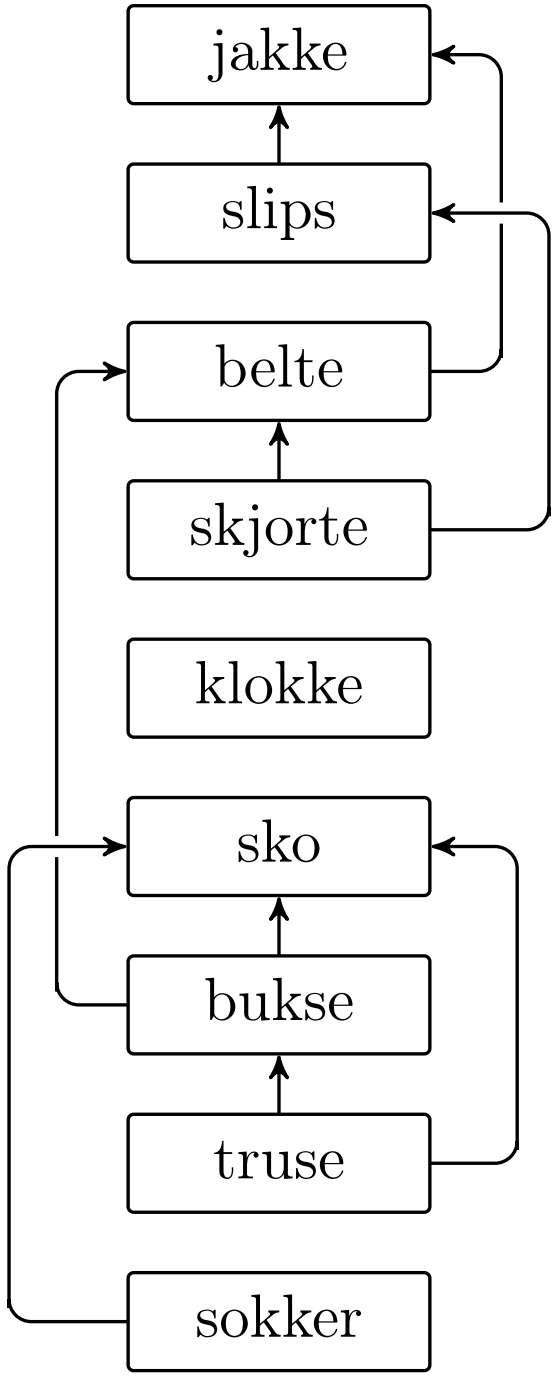
9/10

13/14

12/15

11/16

17/18



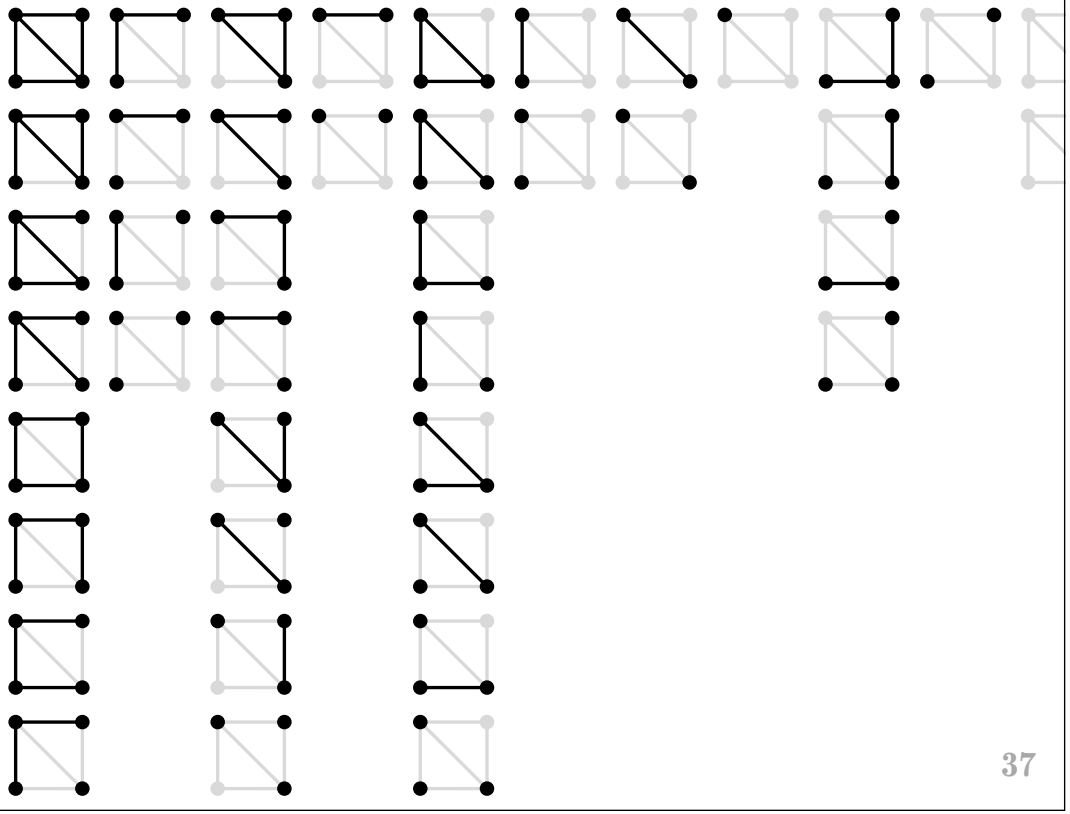
https://
en.wikipedia.org/wiki/
The_dress

Minimale spenntrær

En graf

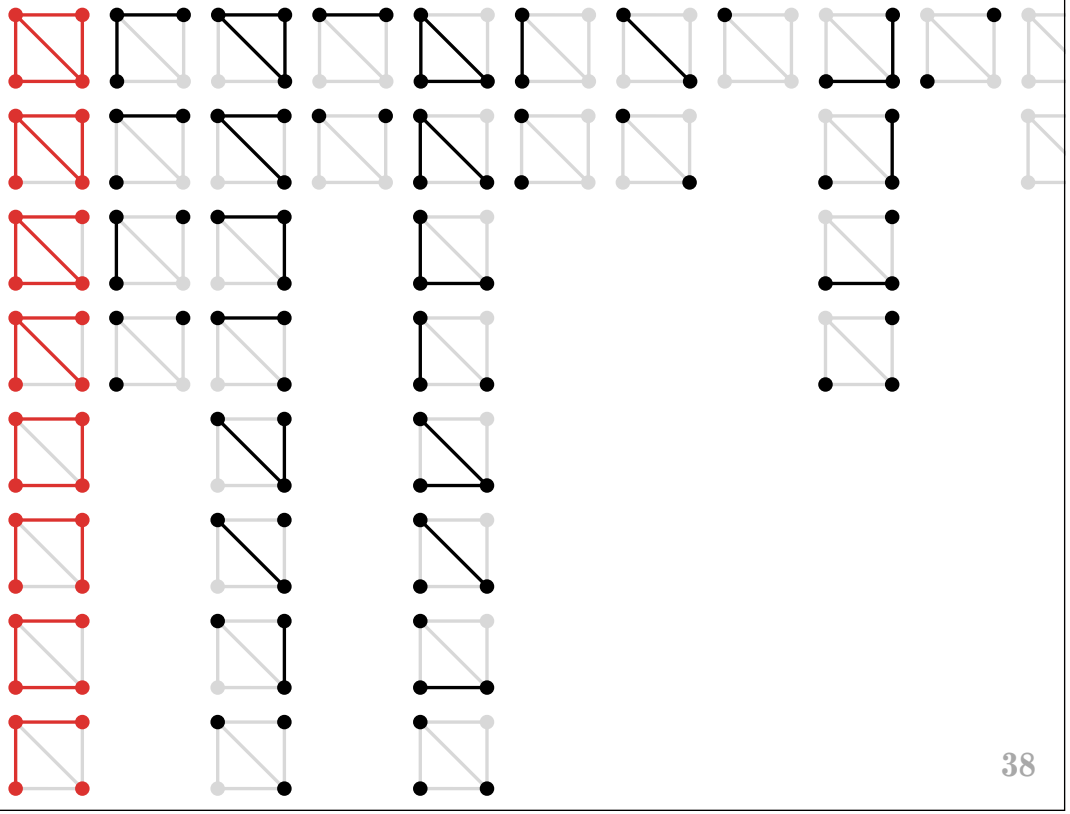
Delgrafer

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.)



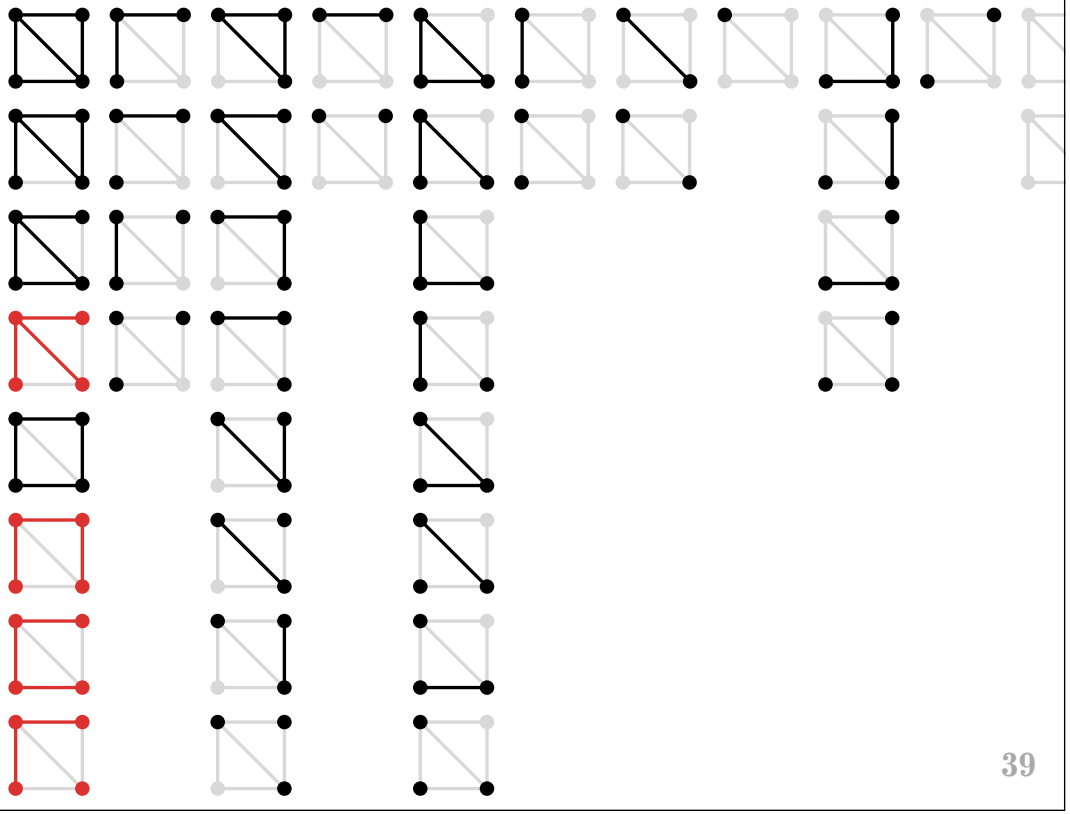
«Spenngrafer»

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



«Spennskoger»

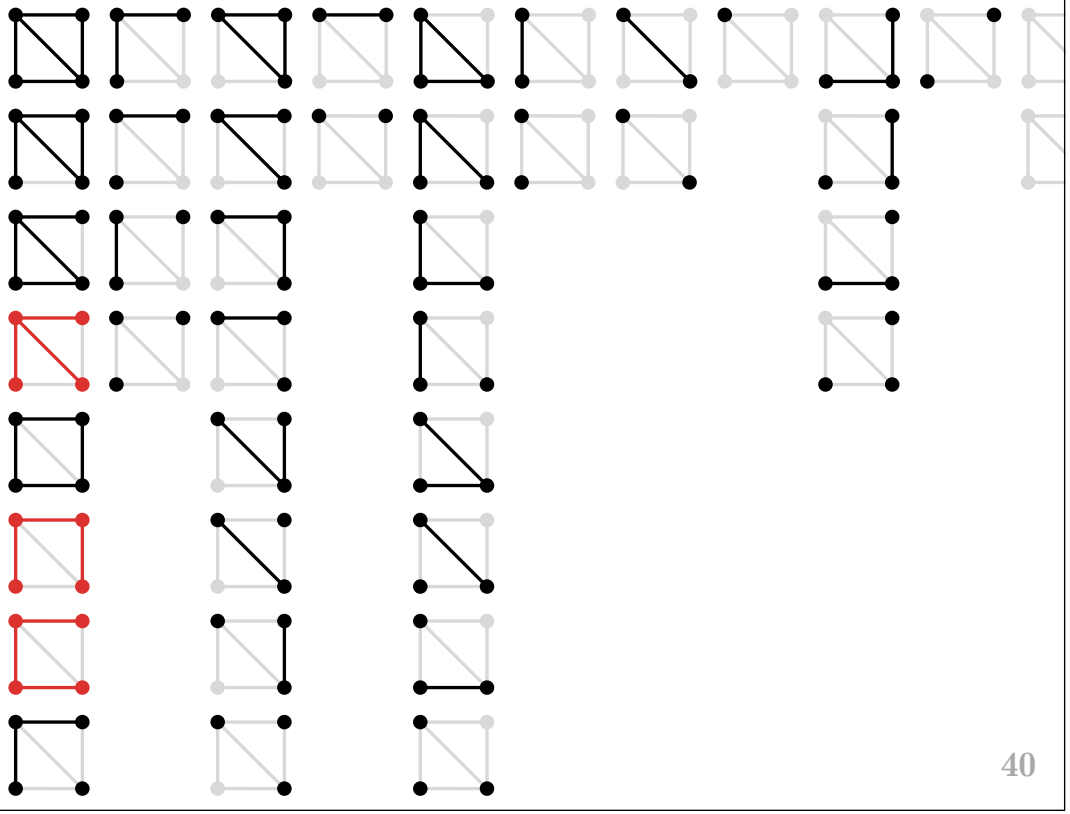
En «dekkende skog» (spanning forest) eller «spenngskog» er en asyklisk spenngraf.



Spenntrær

Et spenntre (spanning tree) er en sammenhengende spennskog.

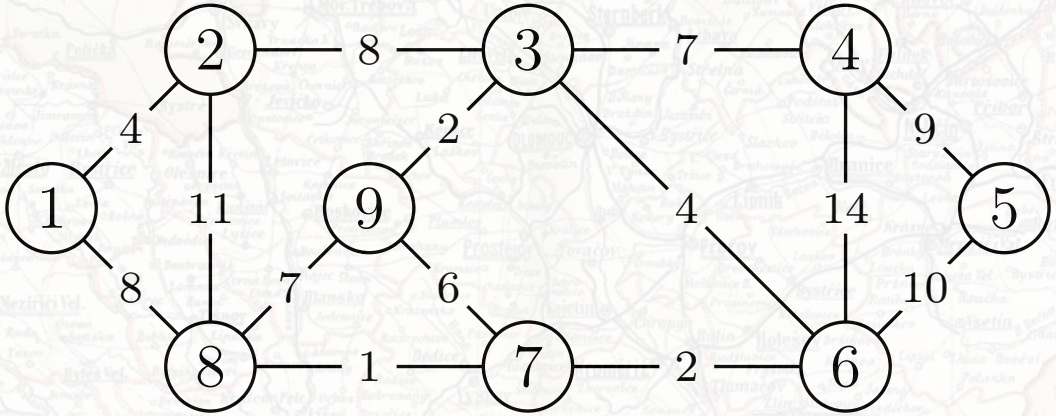
(Spenntre er et vanlig begrep på norsk. Spenngraf og spennskog er det ikke.)



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.

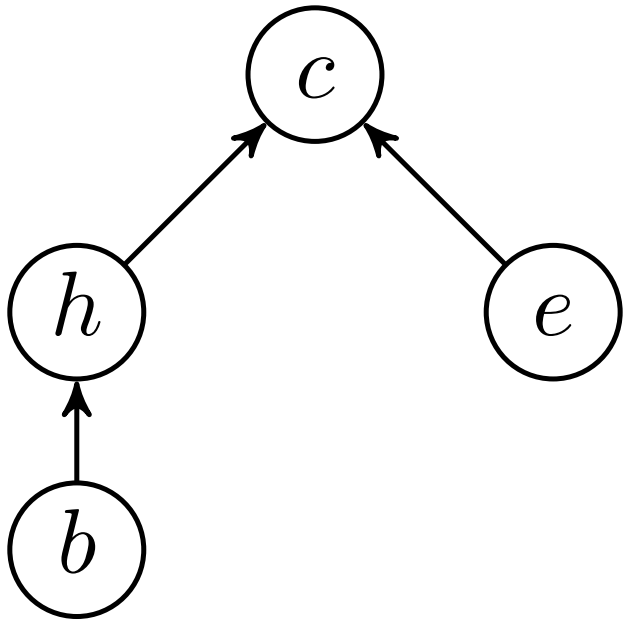
https://commons.wikimedia.org/wiki/File:Map_of_Moravia_and_Austrian_Silesia_2.jpg



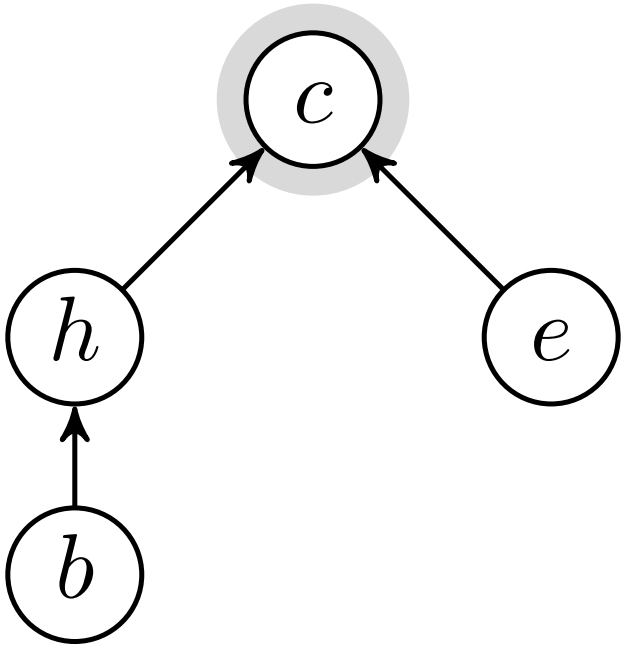
Vi vil knytte sammen nodene billigst mulig

MST ›

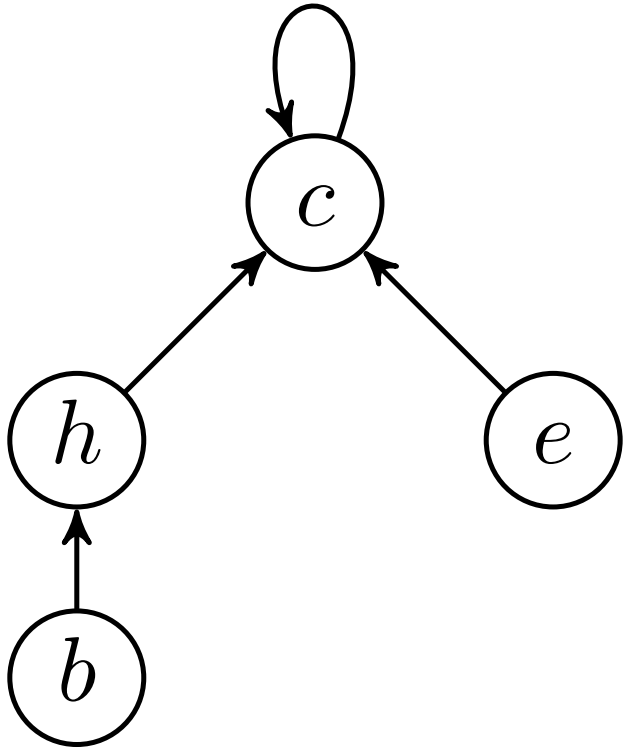
Disjunkte mengder



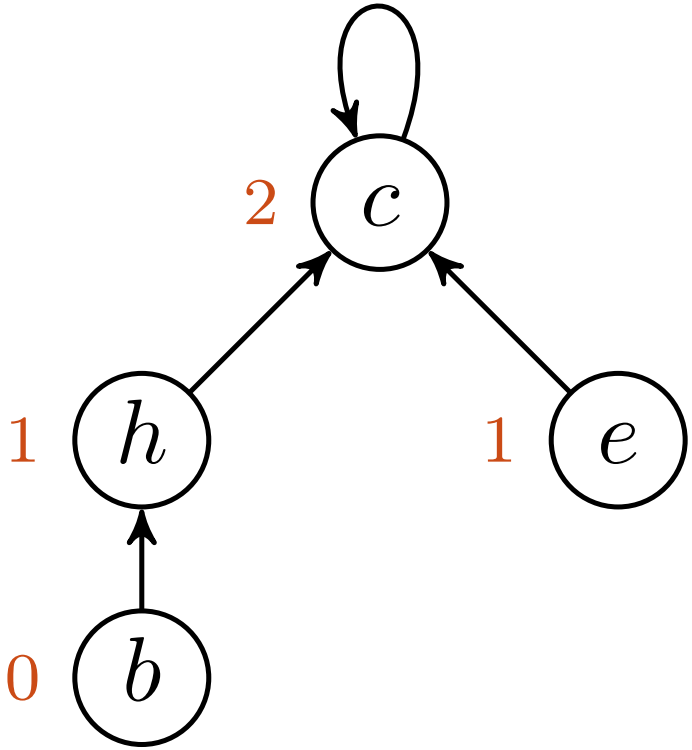
Mengder representeres som trær vha. foreldrepekere $v.p$



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



Self-loop: Fjerner spesialtilfelle fra FIND-SET

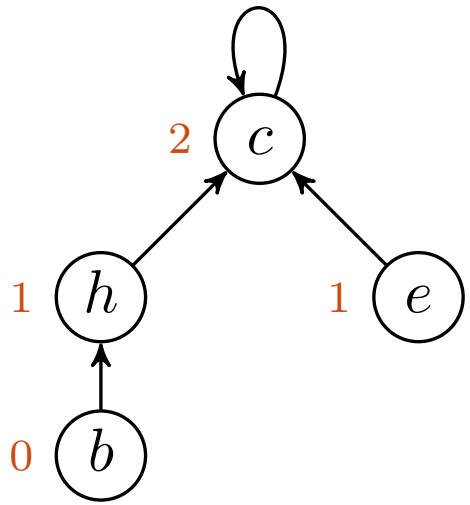


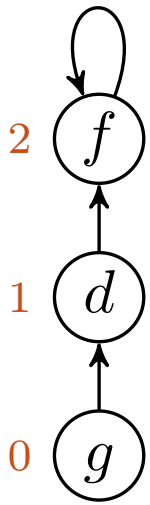
For *union by rank*-heuristikk: Rang er øvre grense for nodehøyde

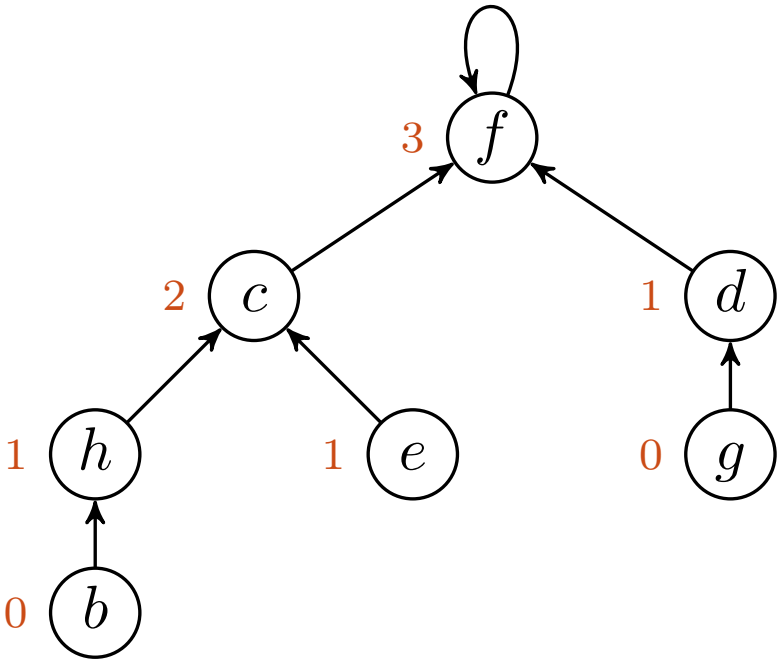
MAKE-SET(x)
1 $x.p = x$
2 $x.rank = 0$

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

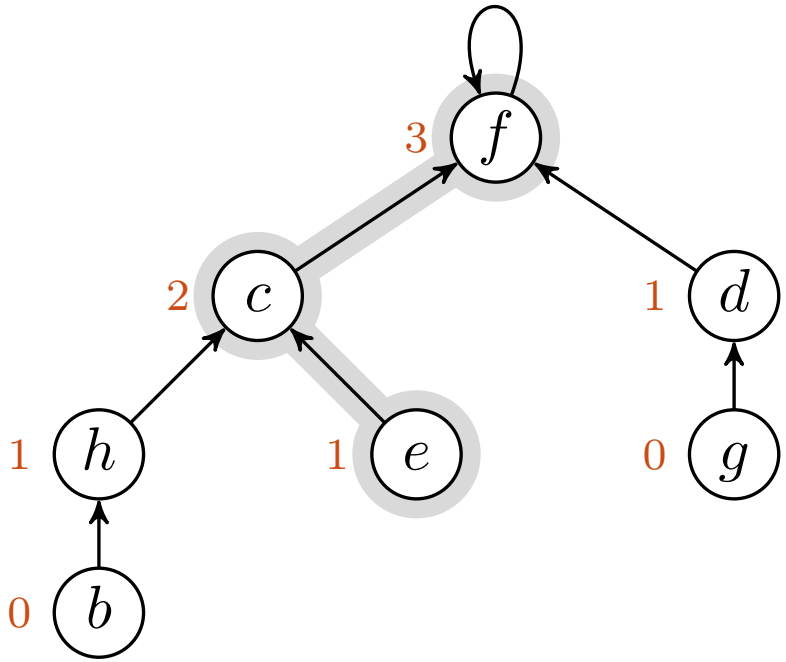
```
LINK(x, y)
1   if x.rank > y.rank
2         y.p = x
3   else x.p = y
4         if x.rank == y.rank
5               y.rank = y.rank + 1
```

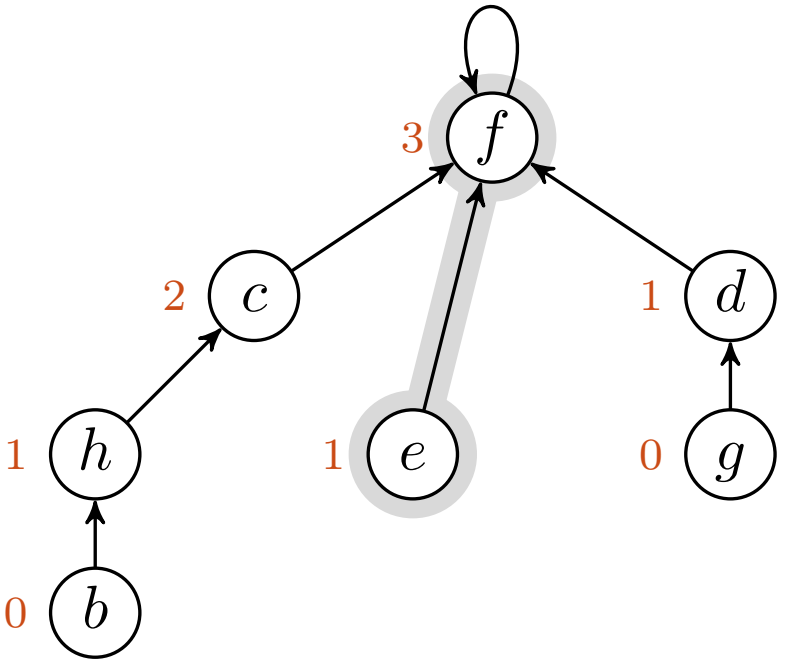

2
c
1
h
1
e
0
b

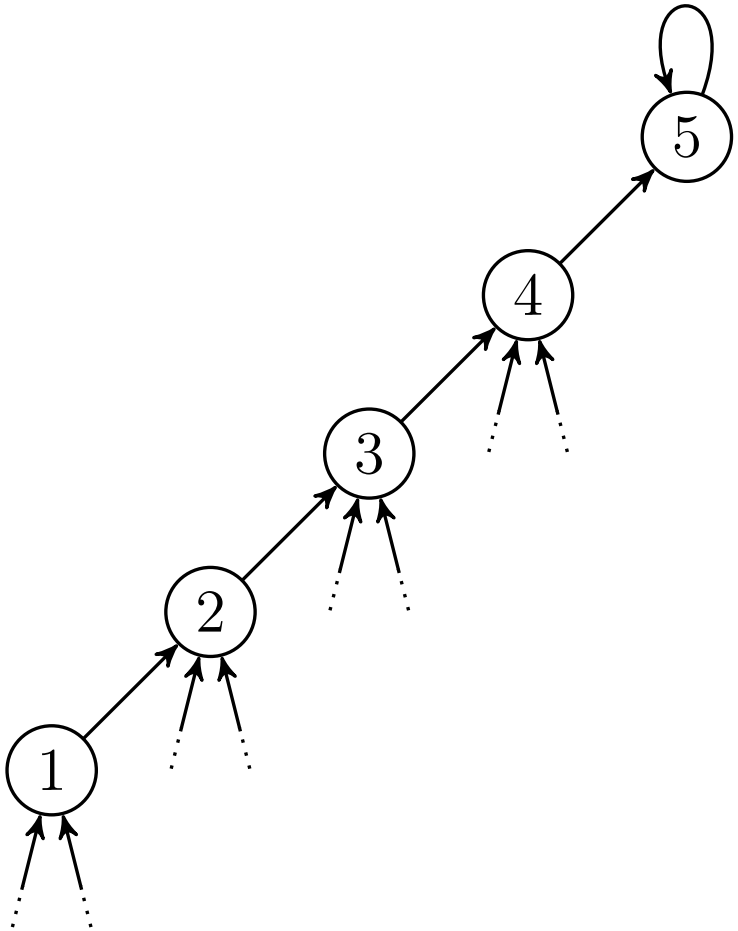

2
f
1
d
0
g


3
f
2
c
1
d
1
h
1
e
0
g
0
b

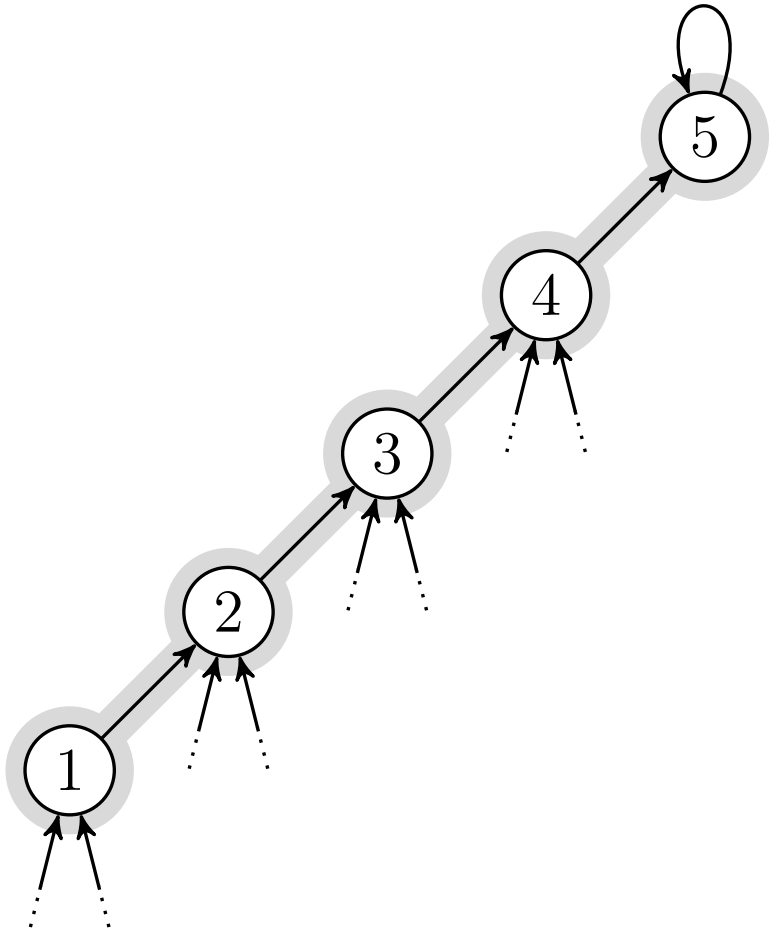
```
FIND-SET(x)
1  if x ≠ x.p
2        x.p = FIND-SET(x.p)
3  return x.p
```


3
f
2
c
1
d
1
h
1
e
0
g
0
b

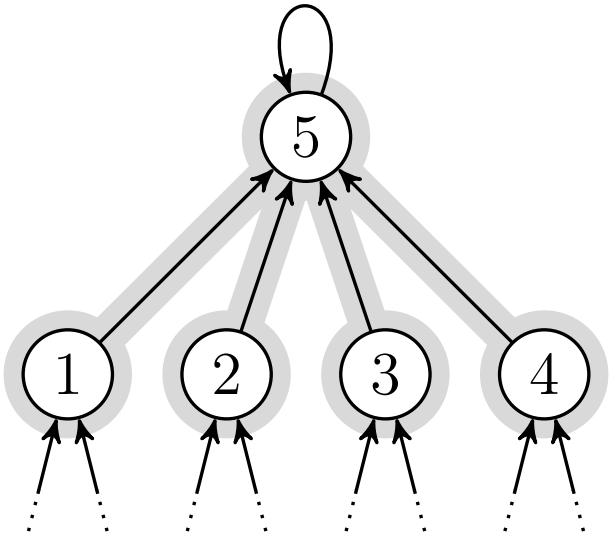

3
f
2
c
1
d
1
h
1
e
0
g
0
b



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



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

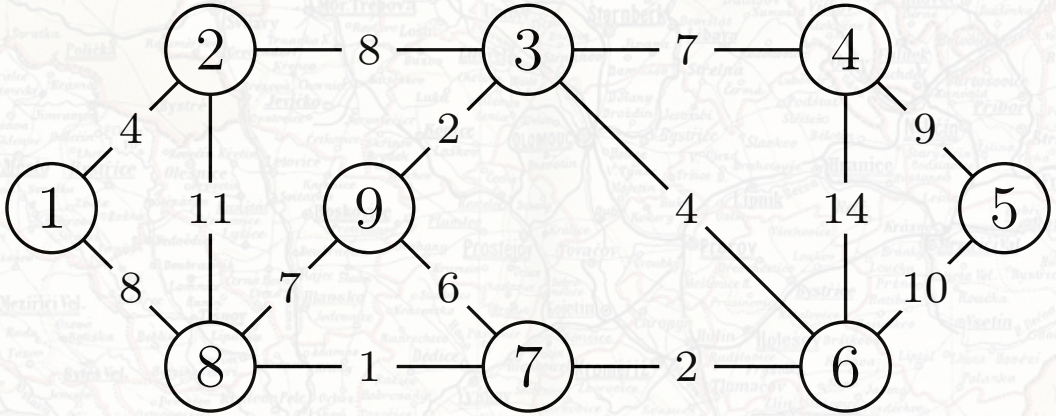


FIND-SET er rekursiv og «destruktiv»; alle på stien får $p = 5$

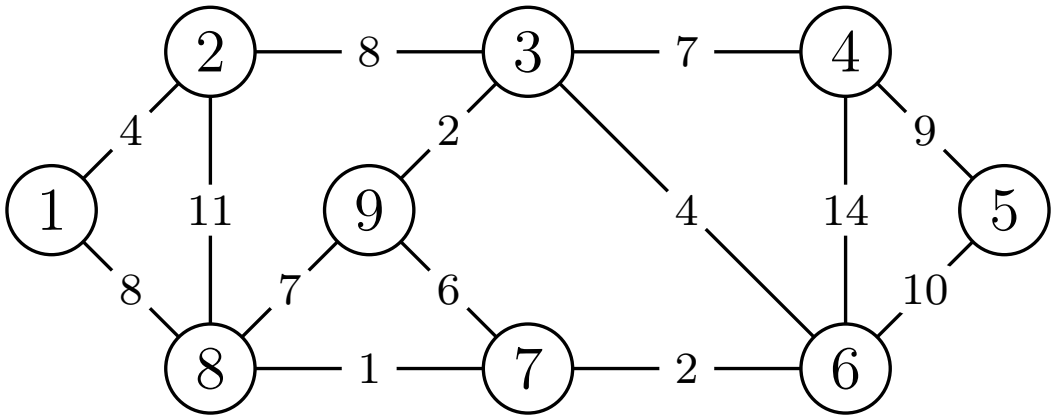
m operasjoner: $\mathrm{O}(m \cdot \alpha(n))$

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

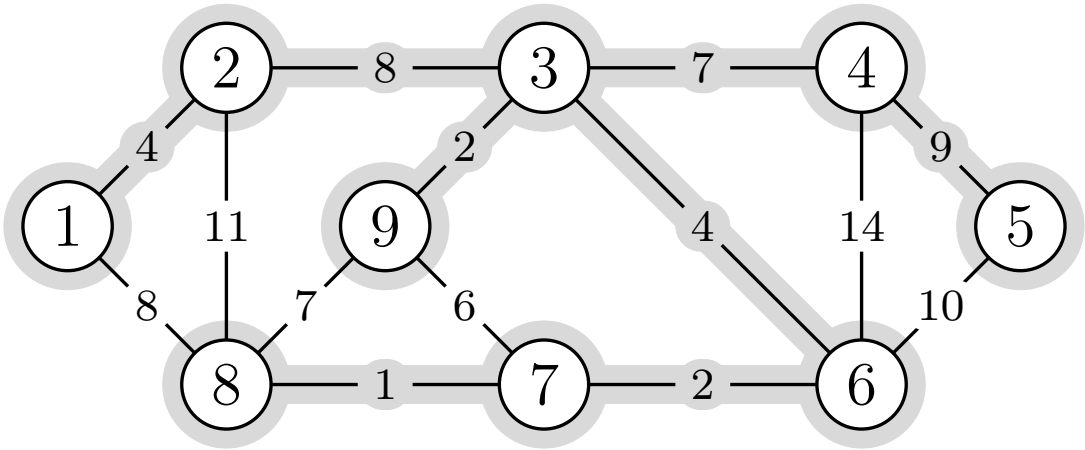
MST › Generisk



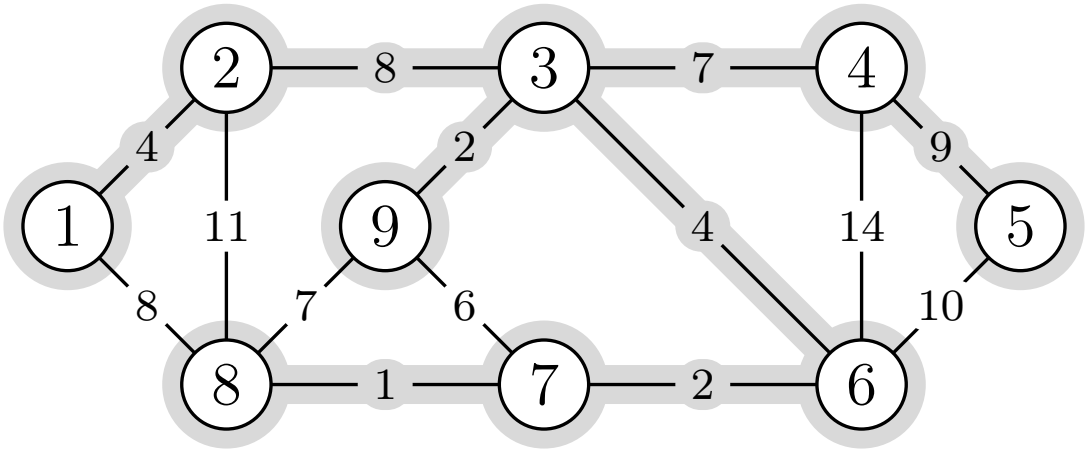
Vi vil knytte sammen nodene billigst mulig



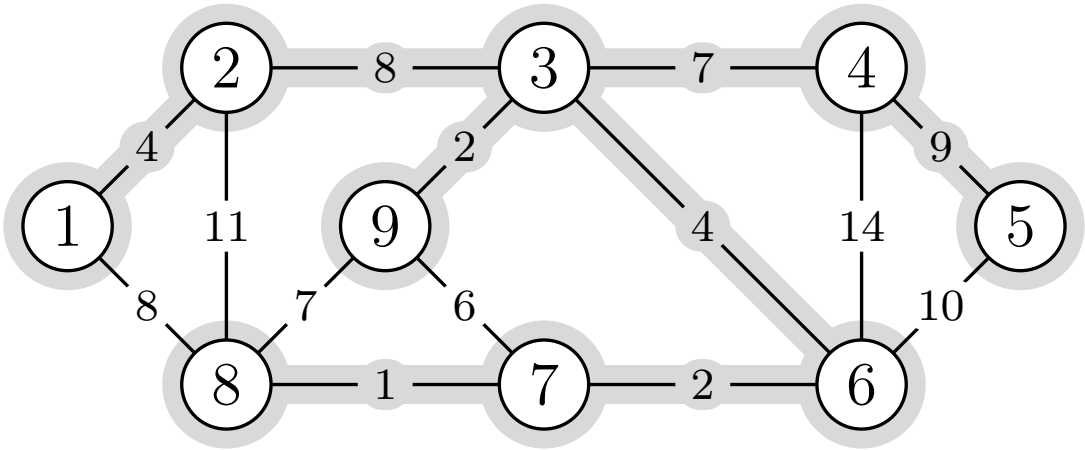
Delmengde $T \subseteq E$ som spenner over V og minimerer $\sum_{e \in T} w(e)$



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



Hvis w er positiv vil T alltid bli asyklisk



Generelt: Tillater negativ w men krever asyklisk T

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

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

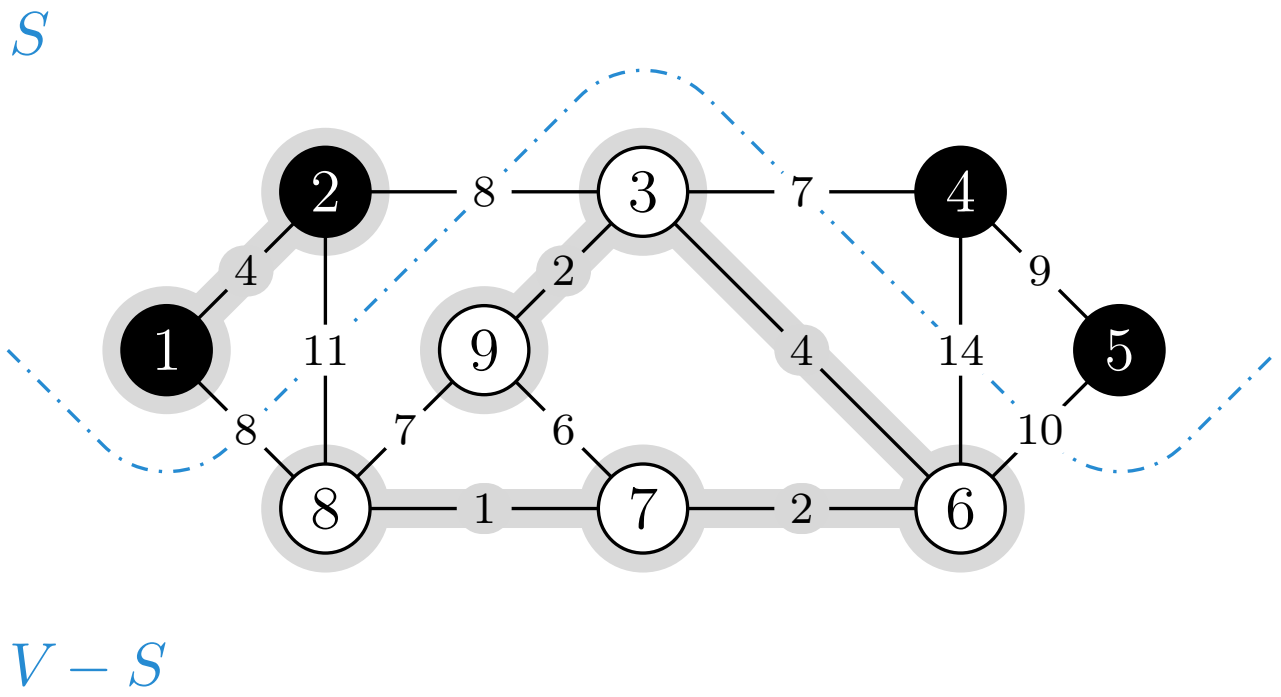
$$w(T) = \sum_{(u,v) \in T} w(u, v) \, .$$

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**

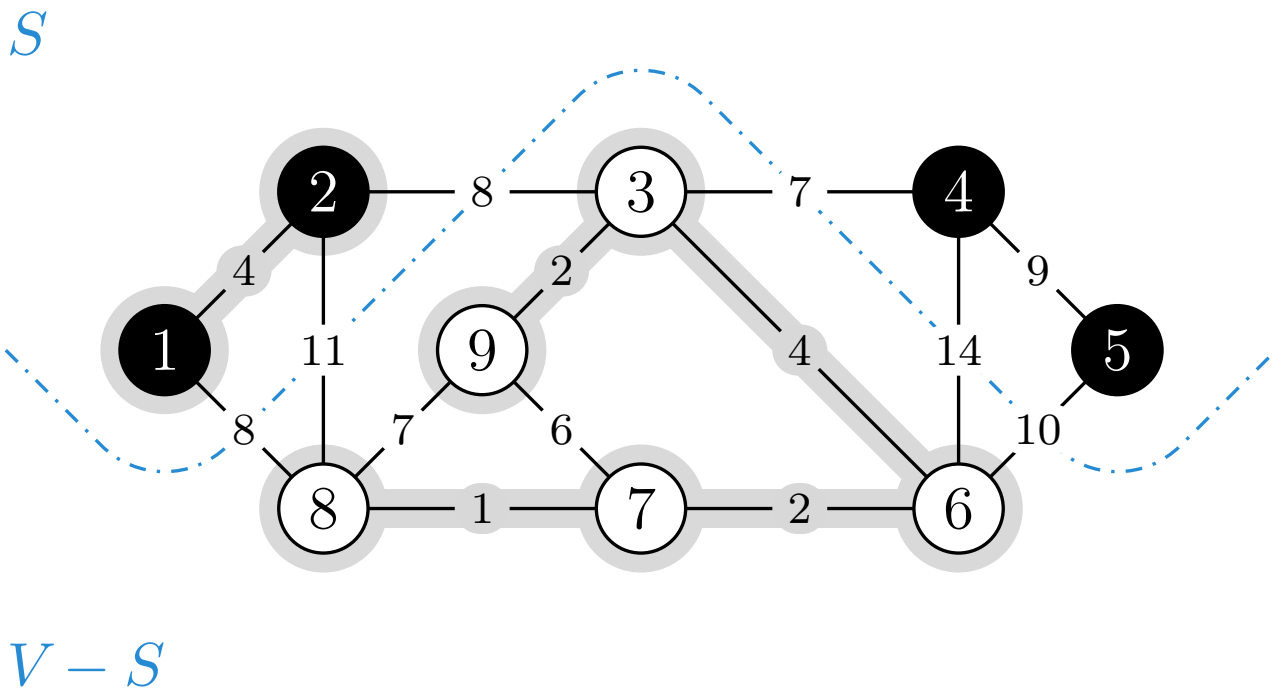
GENERIC-MST(G, w)
1 $A = \emptyset$
2 **while** A does not form a spanning tree
3 find an edge (u, v) that is safe for A
4 $A = A \cup \{(u, v)\}$
5 **return** A

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).

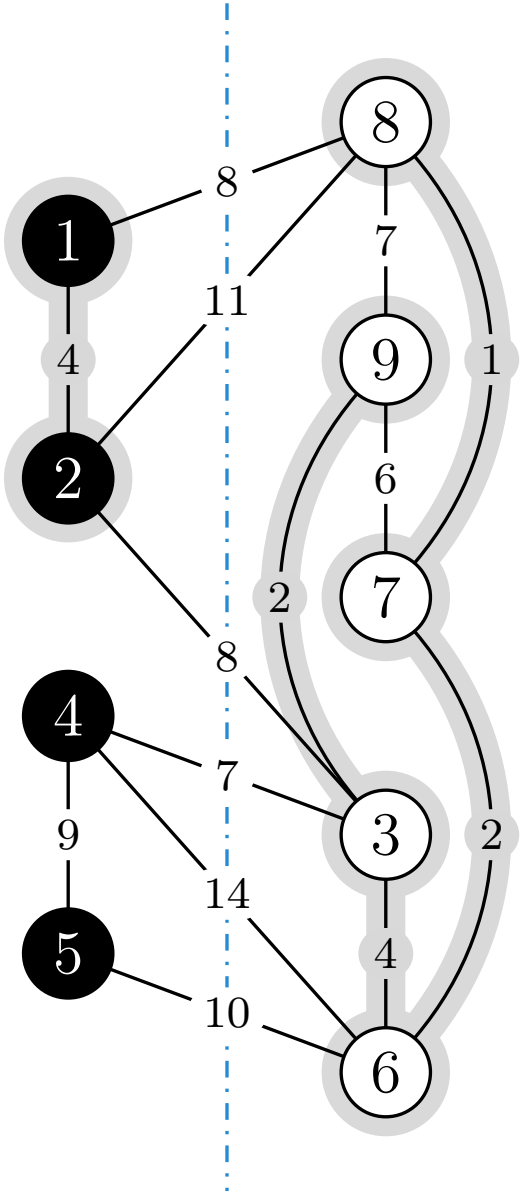


Et snitt $(S, V - S)$ som respekterer kantmengden A (uthevet)

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



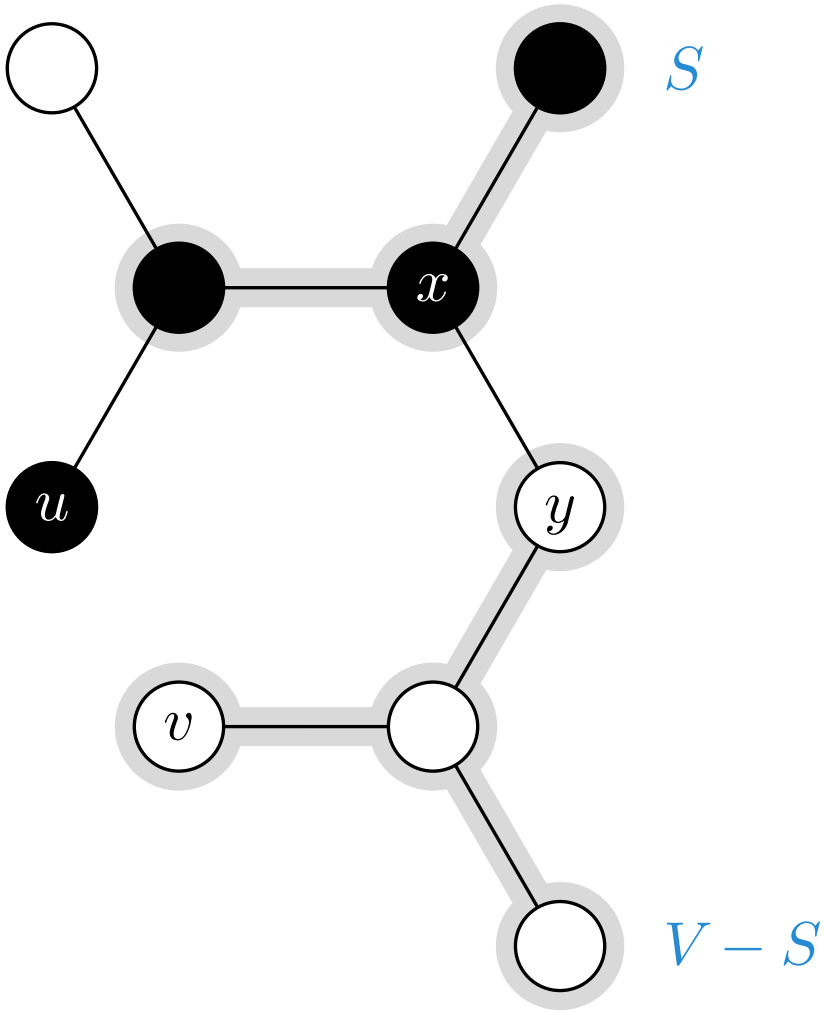
Kanten $(4, 3)$ er en (unik) lett kant over snittet



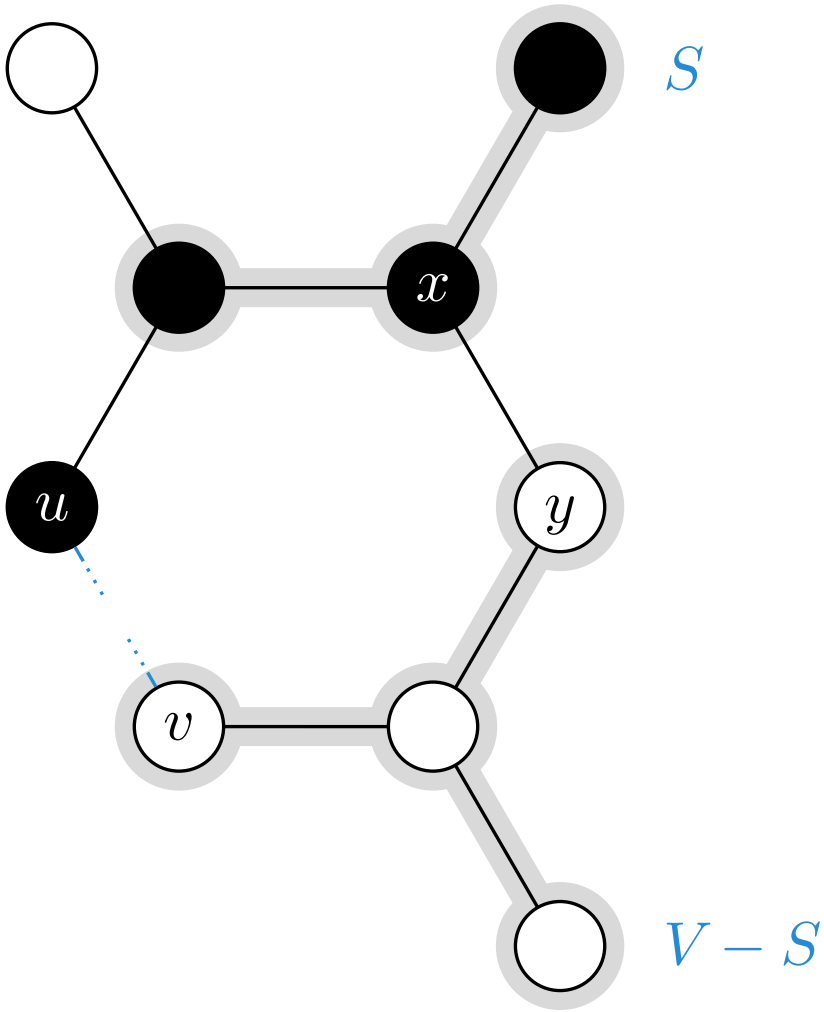
Samme snitt, tegnet på en annen måte. Vi vurderer her altså kanter som går på tvers av midtlinjen.

S

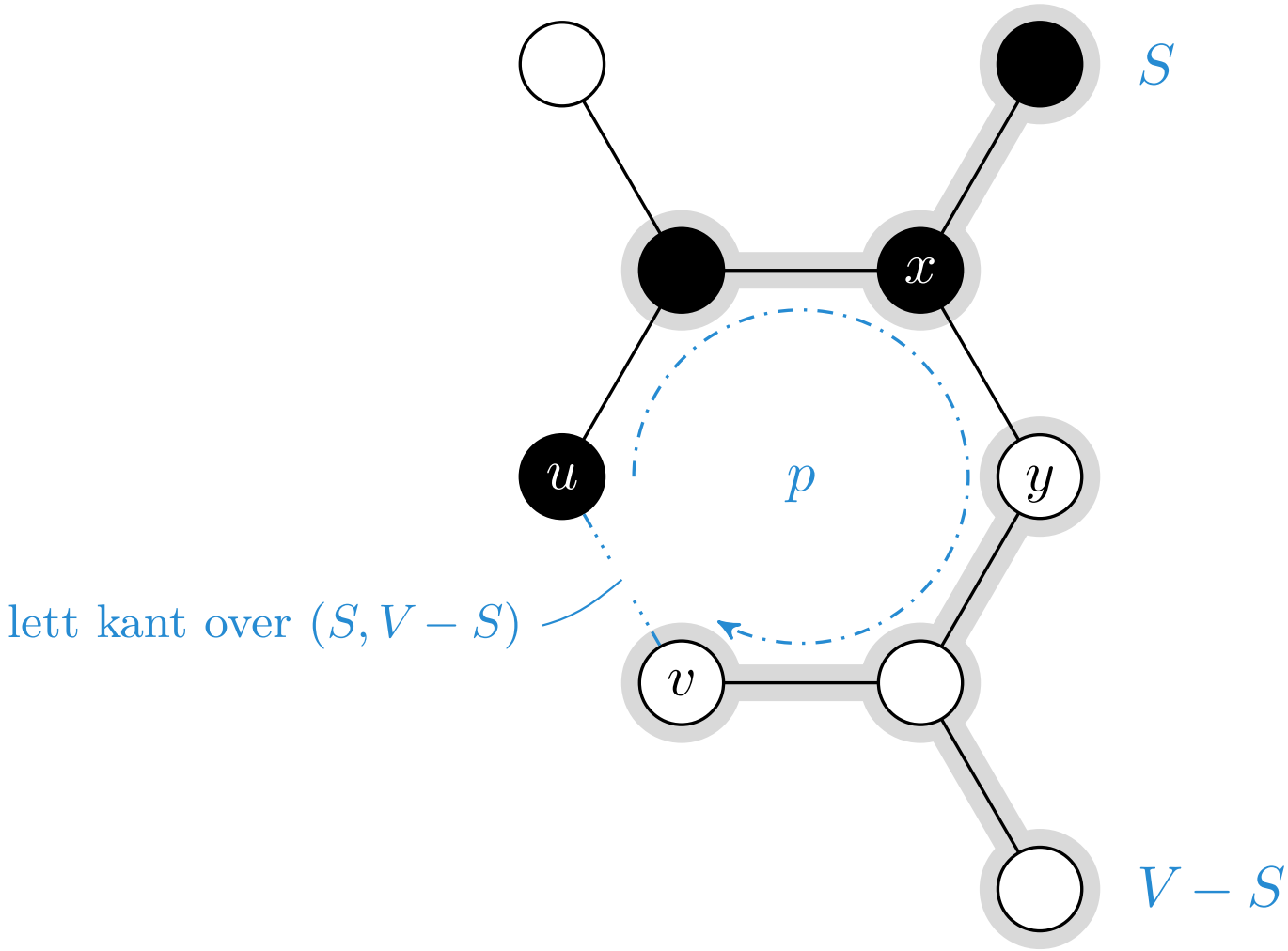
$V - S$



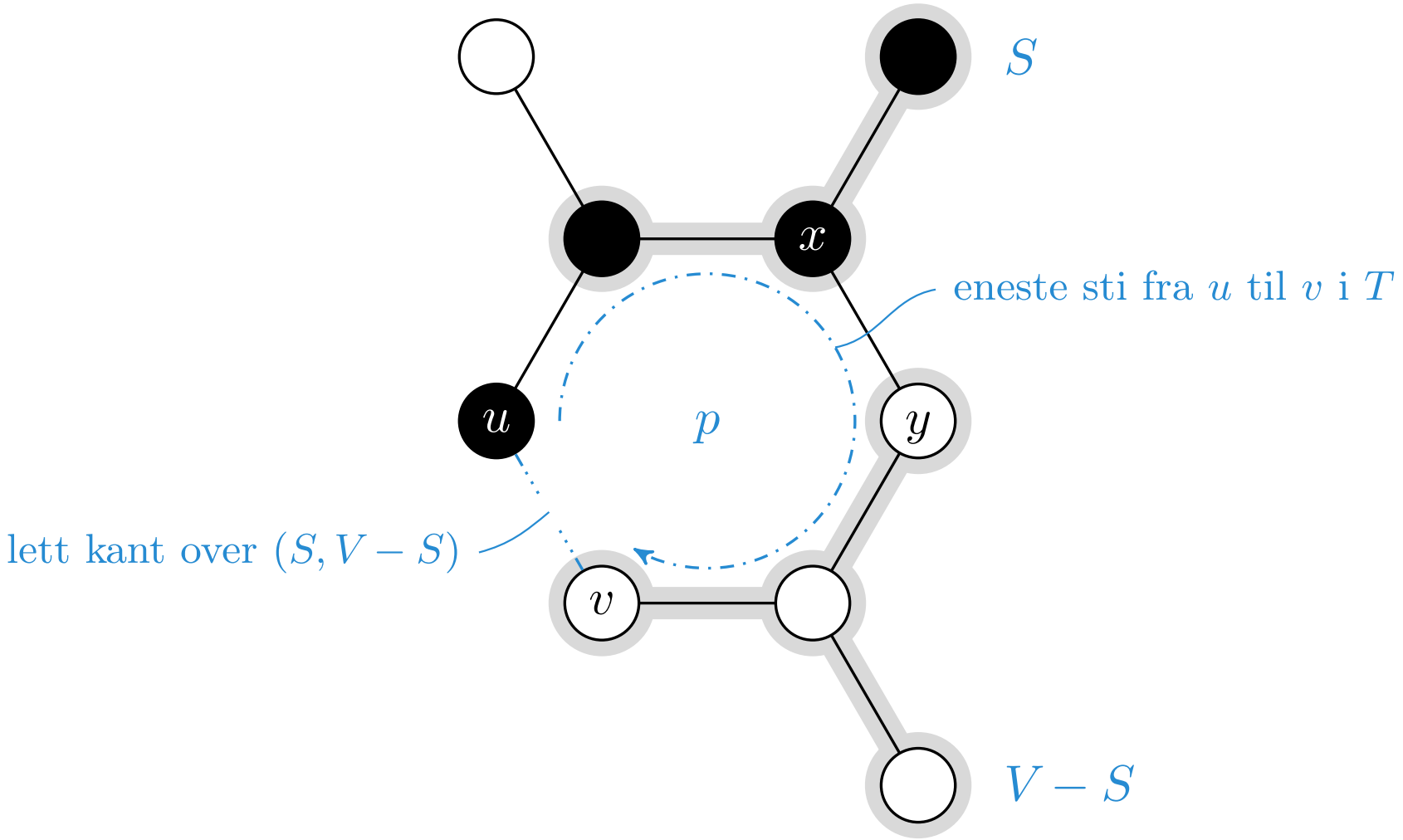
Spenntreet T inneholder A og snittet $(S, V - S)$ respekterer A



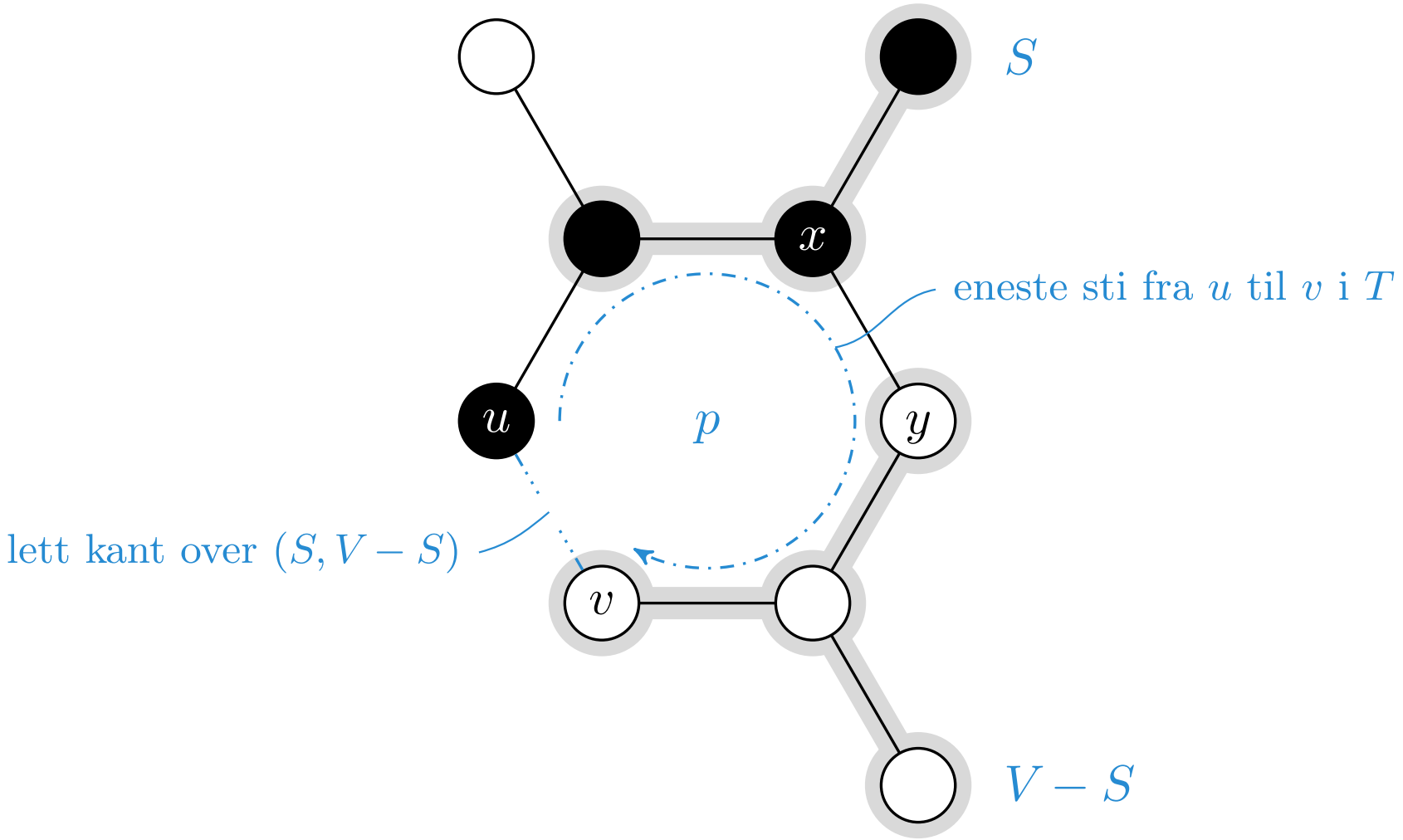
Kanten (u, v) er en lett kant over snittet $(S, V - S)$



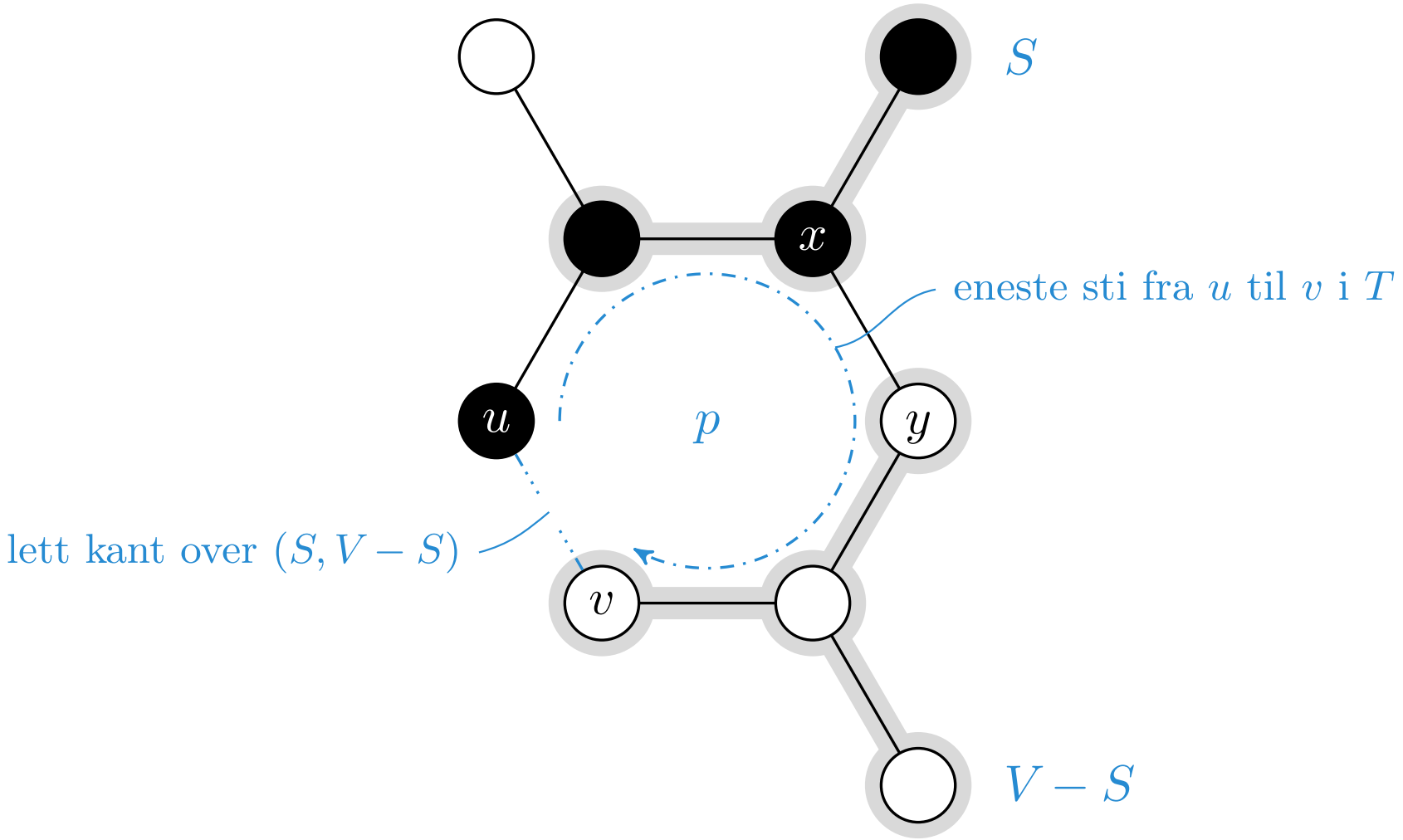
Et spenntre vil ha nøyaktig én sti fra u til v



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 $w(T') \leq w(T)$



T var vilkårlig og T' minst like bra, så (u, v) er trygg for A

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

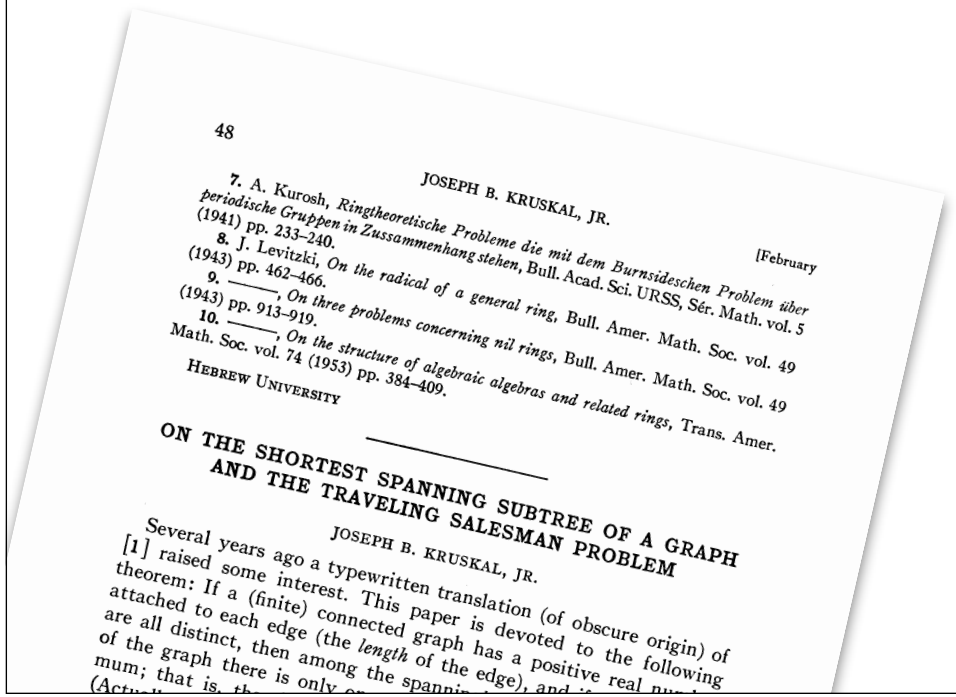
Kruskal: En kant med minimal vekt blant de gjenværende er trygg så lenge den ikke danner sykler.

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

MST › Kruskal



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 (of obscure origin) of [1] raised some interest. This paper is devoted to the following theorem: If a (finite) connected graph has a positive real number attached to each edge (the *length* of the edge), and if the lengths are all distinct, then among the spanning subtrees of the graph there is only one, whose sum is a minimum; that is, the

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.

which solves Problem 1 may be used to prove this theorem.

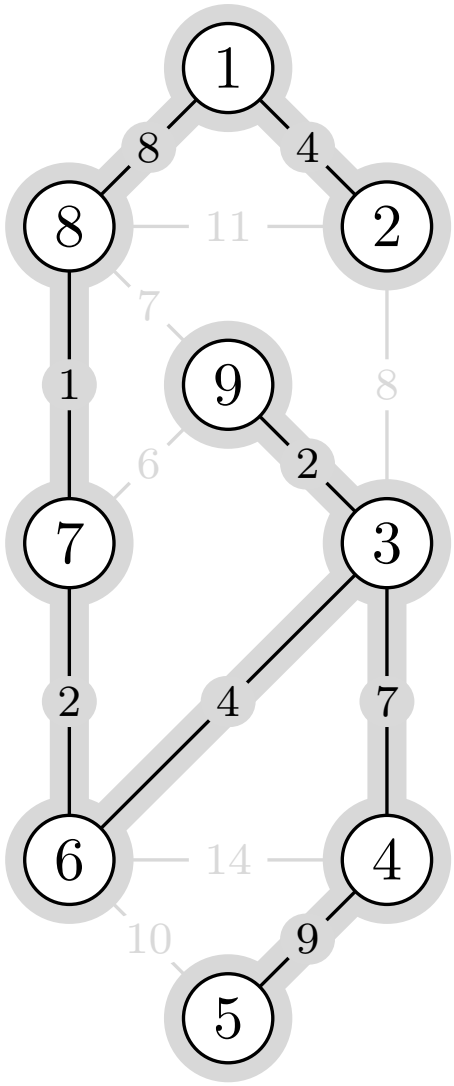
First I would like to point out that there is no loss of generality in assuming that the given connected graph G is complete, that is, that every pair of vertices is connected by an edge. For if any edge of G is "missing," an edge of great length may be inserted, and this does not alter the graph in any way which is relevant to the present purposes. Also, it is possible and intuitively appealing to think of missing edges as edges of infinite length.

CONSTRUCTION A. Perform the following step as many times as possible: Among the edges of G not yet chosen, choose the shortest edge which does not form any loops with those edges already chosen. Clearly the set of edges eventually chosen must form a spanning tree of G, and in fact it forms a shortest spanning tree.

CONSTRUCTION B. Let V be an arbitrary but fixed (nonempty) subset of the vertices of G. Then perform the following step as many times as possible: Among the edges of G which are not yet chosen but which are connected either to a vertex of V or to an edge already chosen, pick the shortest edge which does not form any loops with the edges already chosen. Clearly the set of edges eventually chosen forms a spanning tree of G, and in fact it forms a shortest spanning tree. In case V is the set of all vertices of G, then Construction B

```
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
```

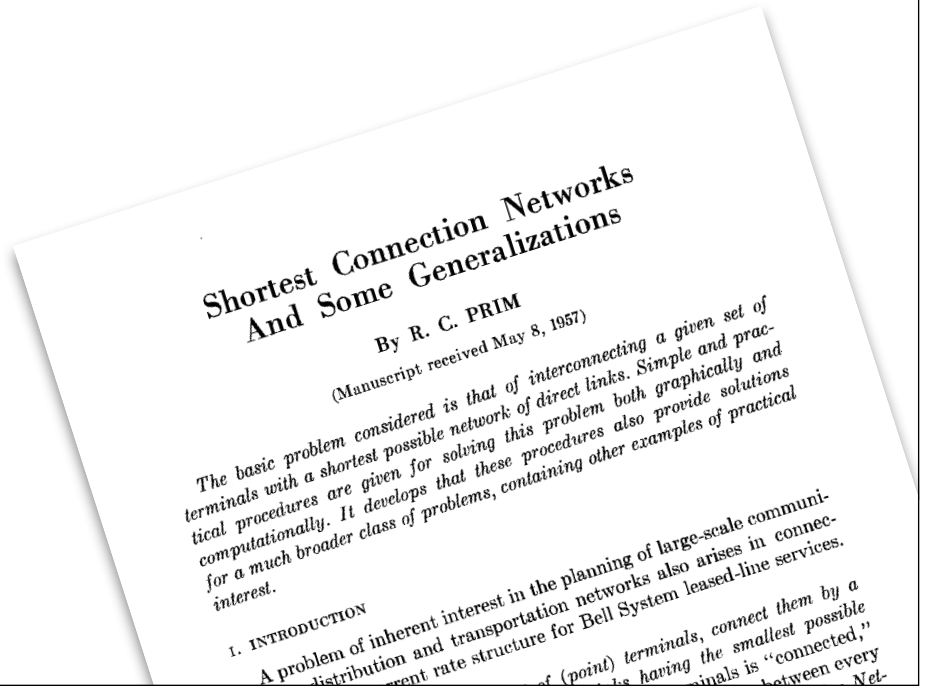


Operasjon	Antall	Kjøretid
MAKE-SET	V	$\mathrm{O}(1)$
Sortering	1	$\mathrm{O}(E \lg E)$
FIND-SET	$\mathrm{O}(E)$	$\mathrm{O}(\lg V)$
UNION	$\mathrm{O}(E)$	$\mathrm{O}(\lg V)$

Totalt: $\mathrm{O}(E \lg V)$

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

MST › Prim



Shortest Connection Networks
And Some Generalizations

By R. C. PRIM

(Manuscript received May 8, 1957)

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

I. INTRODUCTION

A problem of inherent interest in the planning of large-scale communi-
distribution and transportation networks also arises in connec-
rent rate structure for Bell System leased-line services.
of (*point*) *terminals, connect them by a*
having the smallest possible
inals is "connected,"
between every
Net-

not fragments.) The *distance of a terminal from a fragment* of which it is not an element is the minimum of its distances from the individual terminals comprising the fragment. An *isolated fragment* is a fragment to which, at a given stage of the construction, no external connections have been made. (In Fig. 2, 8-3 and 1-6-7-5 are the only isolated fragments.) A *nearest neighbor of a terminal* is a terminal whose distance from the specified terminal is at least as small as that of any other. A *nearest neighbor of a fragment*, analogously, is a terminal whose distance from the specified fragment is at least as small as that of any other.

The two fundamental construction principles (P1 and P2) for shortest connection networks can now be stated as follows:

Principle 1 — *Any isolated terminal can be connected to a nearest neighbor.*

Principle 2 — *Any isolated fragment can be connected to a nearest neighbor by a shortest available link.*

For example, the next steps in the incomplete construction of Fig. 2 could be any one of the following:

(1) add link 9-2 (P1 applied to Term. 9)

(P2), 9-2 (P1), and 1-9 (P2). Another is: 1-9 (P2), 9-2 (P2), 2-8 (P2), and 8-4 (P2).

As a second example, the construction of the network of Fig. 1 could have proceeded as follows: Olympia-Salem (P1), Salem-Boise (P2), Boise-Salt Lake City (P2), Helena-Boise (P1), Sacramento-Carson City (P1), Carson City-Boise (P2), Salt Lake City-Denver (P2), Phoenix-Santa Fe (P1), Santa Fe-Denver (P2), and so on.

The kind of intermixture of applications of P1 and P2 demonstrated here is very efficient when the shortest connection network is actually being laid out on a map on which the given terminal set is plotted to scale. With only a few minutes of practice, an example as complex as that of Fig. 1 can be solved in less than 10 minutes. Another mode of procedure, making less use of the flexibility permitted by the construction principles, involves using P1 only once to produce a single fragment, which is then extended by successive applications of P2 until the network is completed. This highly systematic variant, as will emerge later, has advantages for computer mechanization of the solution process. As applied to the example of Fig. 1, this algorithm would proceed as follows if Sacramento were the indicated initial terminal: Sacramento-Carson City, Carson City-Boise, Boise-Salt Lake City, Boise-Helena, Boise-Salem, Salem-Olympia, Salt Lake City-Denver, Denver-Cheyenne, Denver-Santa Fe, and so on.

Since each application of either P1 or P2 reduces the total number

- **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 til treet**
- **For enkelhets skyld: Legg alle noder inn fra starten, med uendelig dårlig prioritet**

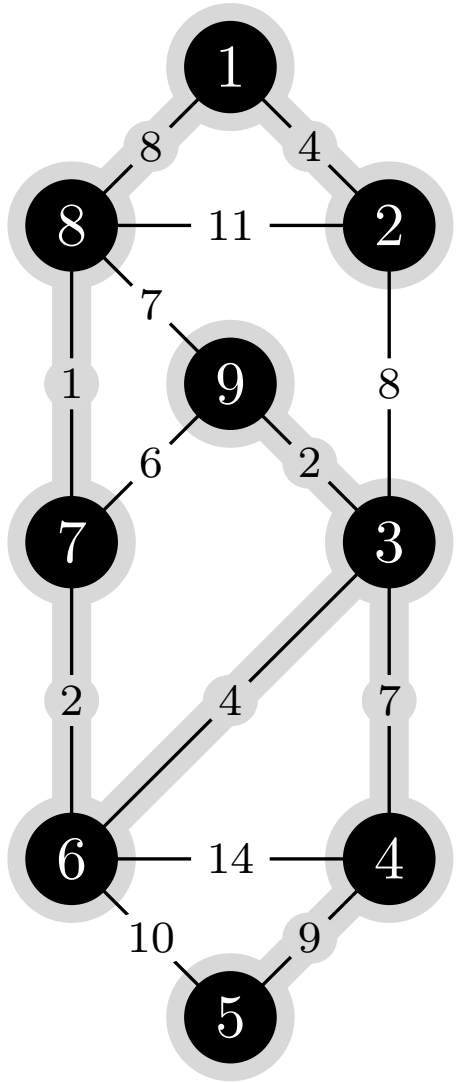
```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

- 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)
```

$u, v = 5, 4$

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



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