

Minimale spaendetraeer

Diskret Matematik og Formelle Sprog
Københavns Universitet, marts 2023

Rasmus Pagh
Slides med lyseblå baggrund er baseret på slides af Kevin Wayne



Hvad sker der?

Mål

- Forståelse af algoritmer til topologisk sortering og stærke sammenhængskomponenter
- Forståelse af algoritmer til at beregne mindste udspændende træ

Motiverende case

(forfattet af Thore Husfeldt)

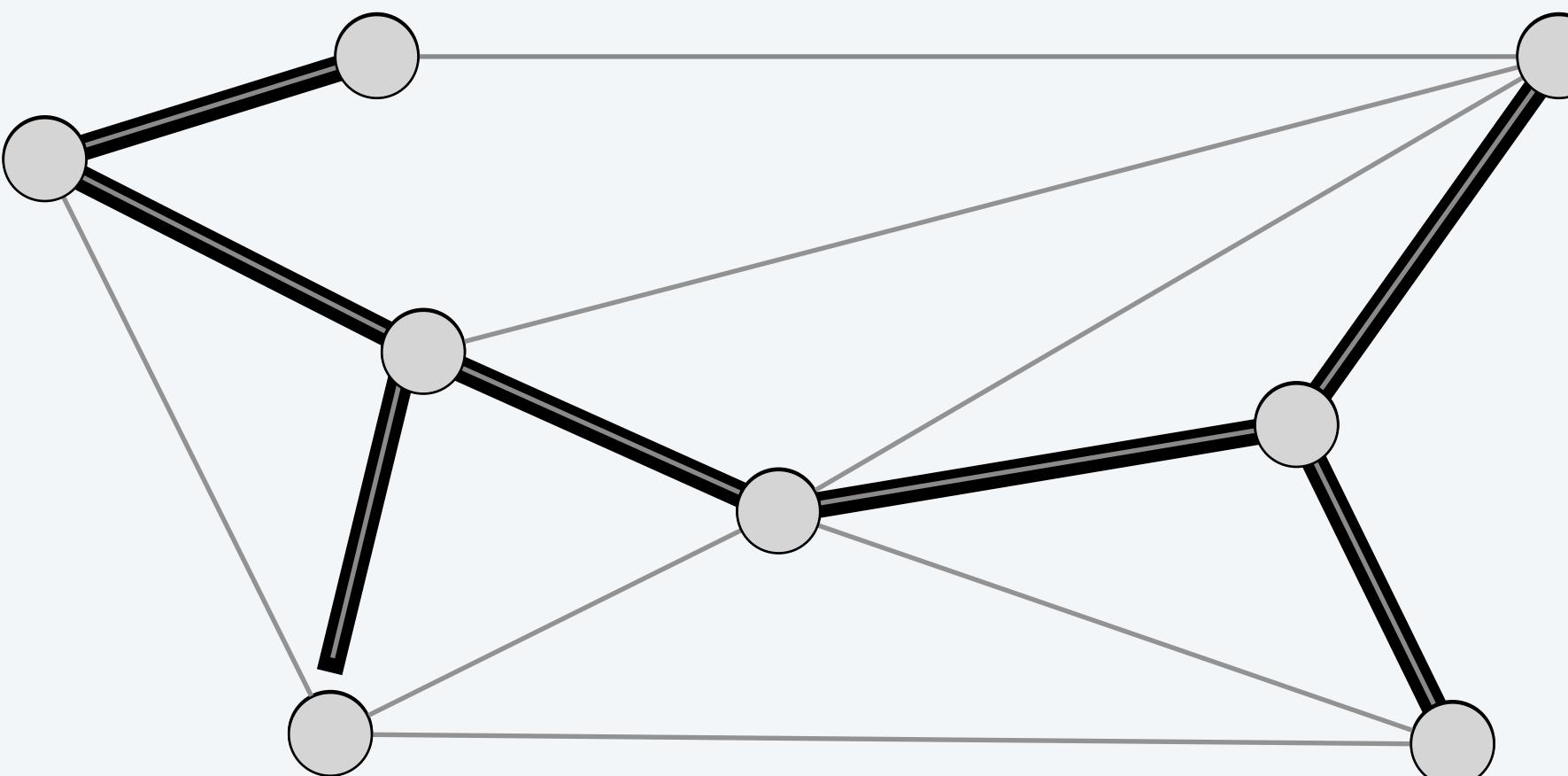
- Det sydfynske øhav skal gøres landfast.
- Den totale udgift for alle de nødvendige broer være så lav som muligt.
- Direkte forbindelser mellem øerne er ikke påkrævede, bare enhver ø kan nås fra enhver af de andre.
- Du har en liste over mulige broforbindelser mellem par af øer og deres forventede omkostning.
- Hvilke projekter sætter du i værk?



Spændetræ (aka. udspændende træ)

Def. Lad $H = (V, T)$ være en delgraf af en uorienteret graf $G = (V, E)$.

H er et **spændetræ** for G hvis H er både **acyklisk** (uden nogen kredse) og **sammenhængende**.

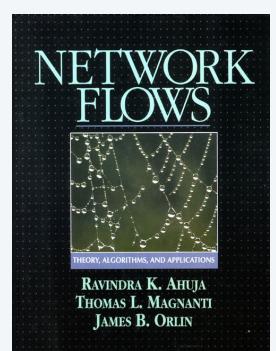


graf $G = (V, E)$
spændetræ $H = (V, T)$

Anvendelser

MST er et fundamentalt problem med mange anvendelser

- Dithering.
- Cluster analysis.
- Max bottleneck paths.
- Real-time face verification.
- LDPC codes for error correction.
- Image registration with Renyi entropy.
- Find road networks in satellite and aerial imagery.
- Model locality of particle interactions in turbulent fluid flows.
- Reducing data storage in sequencing amino acids in a protein.
- Autoconfig protocol for Ethernet bridging to avoid cycles in a network.
- Approximation algorithms for NP-hard problems (e.g., TSP, Steiner tree).
- Network design (communication, electrical, hydraulic, computer, road).

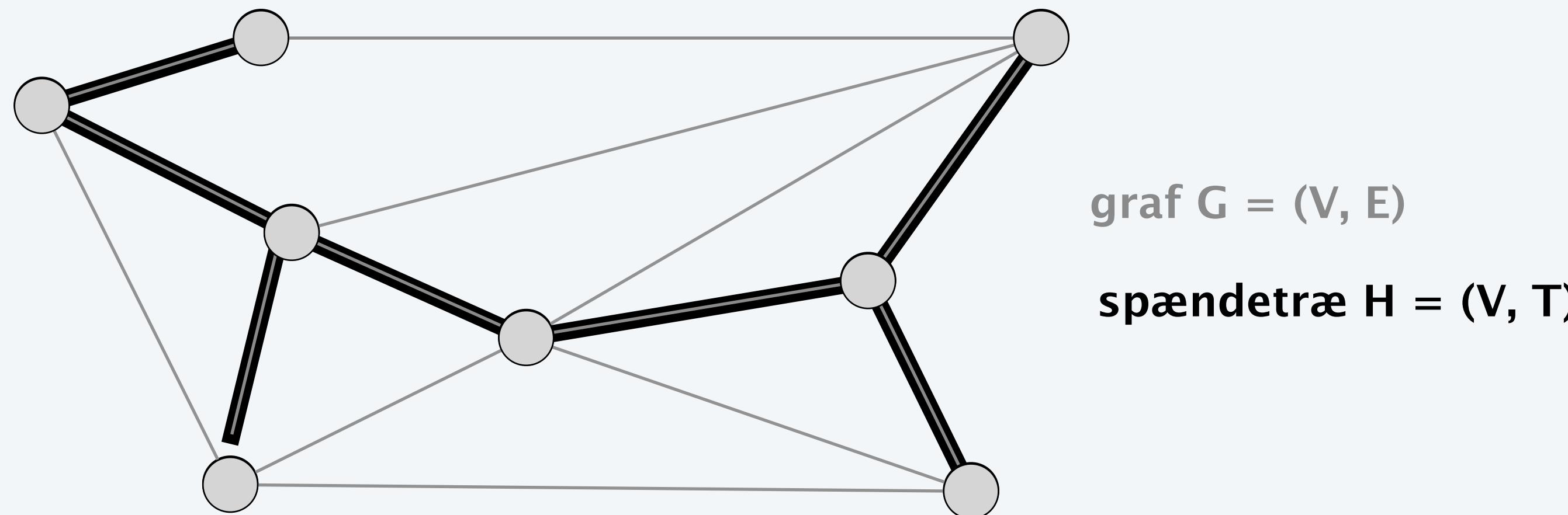


Network Flows: Theory, Algorithms, and Applications,
by Ahuja, Magnanti, and Orlin, Prentice Hall, 1993.

Egenskaber for spændetræer

Lad $H = (V, T)$ være en delgraf af en uorienteret graf $G = (V, E)$. Følgende udsagn er ækvivalente (dvs. enten gælder ingen, eller også gælder alle):

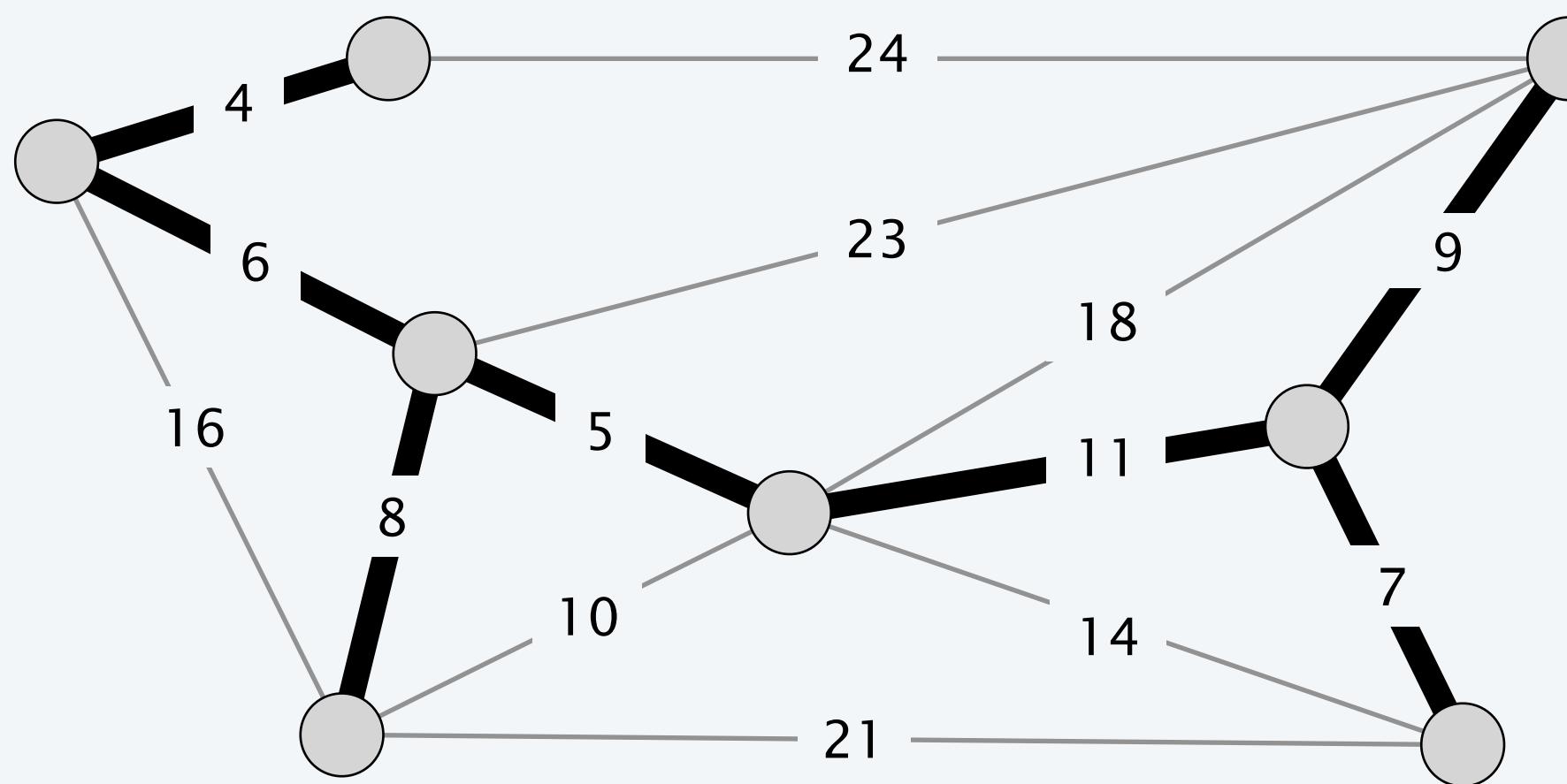
- H er acyklisk og sammenhængende
- H er sammenhængende og har $|V| - 1$ kanter
- H er acyklisk og har $|V| - 1$ kanter
- H er minimalt sammenhængende: at fjerne en kant gør den ikke-sammenhængende
- H er maksimalt acyklisk: at tilføje en kant skaber en kreds





Minimalt spændetræ (MST, aka. minimalt udspændende træ)

Def. Givet en sammenhængende, uorienteret graf $G = (V, E)$ med kantvægte c_e , er et **minimalt spændetræ** (V, T) et spændetræ for G således at summen af kantvægte i T er minimeret.

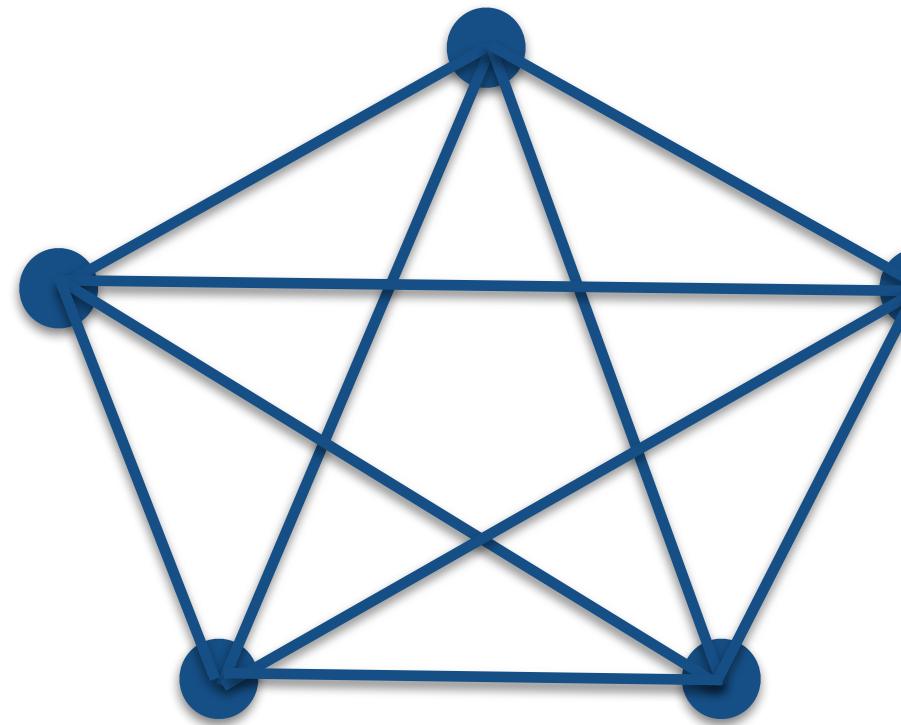


MST omkostning : $4 + 6 + 8 + 5 + 11 + 9 + 7 = 50$

Øvelse

- **Cayley's teorem:**

Den *komplette graf* med $n > 1$ knuder har n^{n-2} spændetræer.



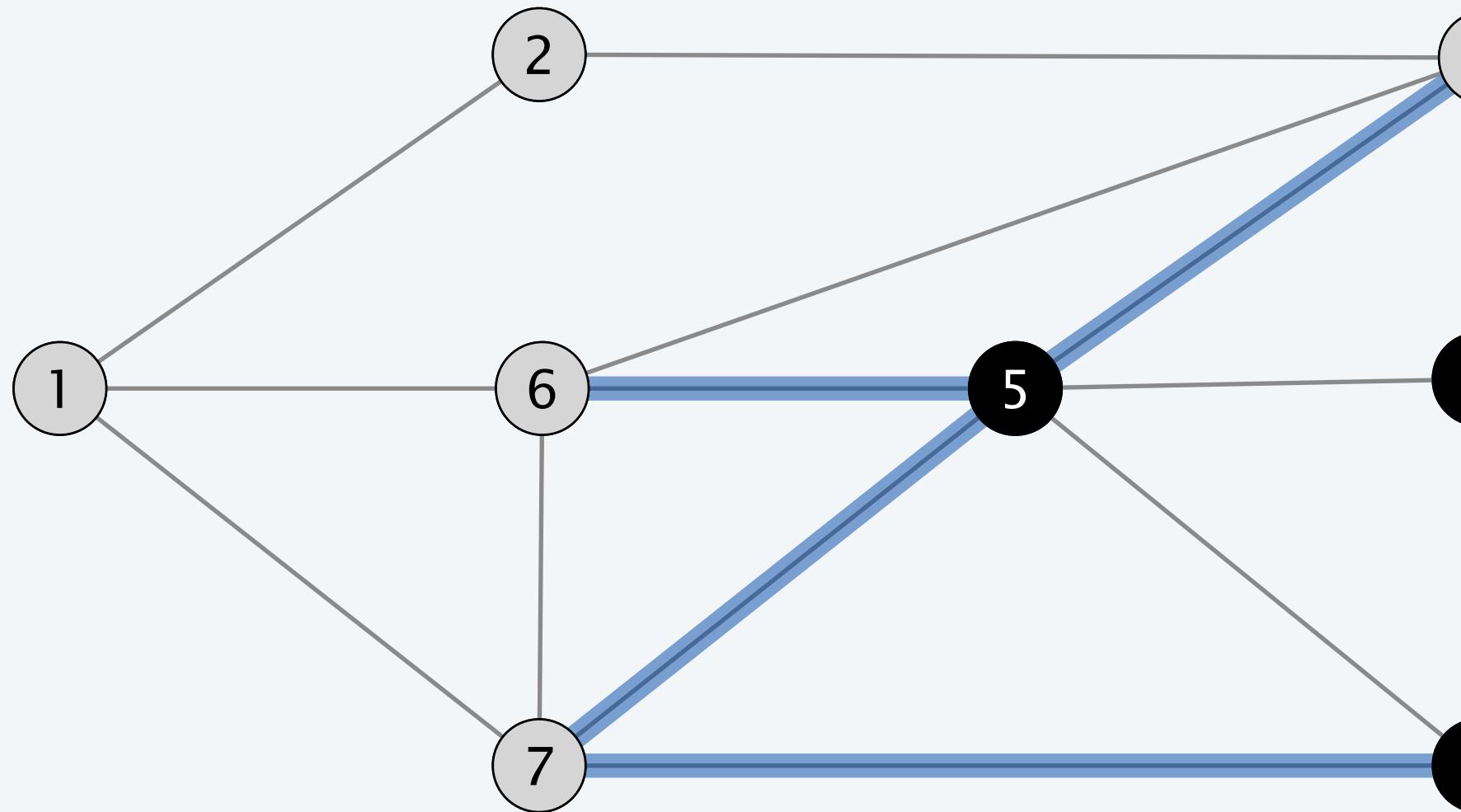
$$n^{n-2} = 5^{5-2} = 125$$

- Vi kan beregne MST ved at evaluere vægten af *alle* spændetræer.
- Antag at din computer kan beregne vægten på 1 million spændetræer i sekundet. Hvor stor en graf kan du håndtere på 1 minut? På 1 år?

Snit

Def. Et snit er en opdeling af knuderne i to ikke-tomme delmængder S og $V-S$.

Def. Snitmængden af et snit S er mængden af kanter, der har præcis én knude i S .



snit for $S = \{ 4, 5, 8 \}$

snitmængde: $\{ (3, 4), (3, 5), (5, 6), (5, 7), (8, 7) \}$

Generisk (grådig) MST algoritme

- Antag at vi allerede har fundet en mængde A af kanter, der er en delmængde af et MST (invariant)
- Vi siger at en kant (u, v) er *sikker* (safe) for A hvis $A \cup \{(u, v)\}$ også er en delmængde af et MST

GENERIC-MST(G, w)

$A = \emptyset$

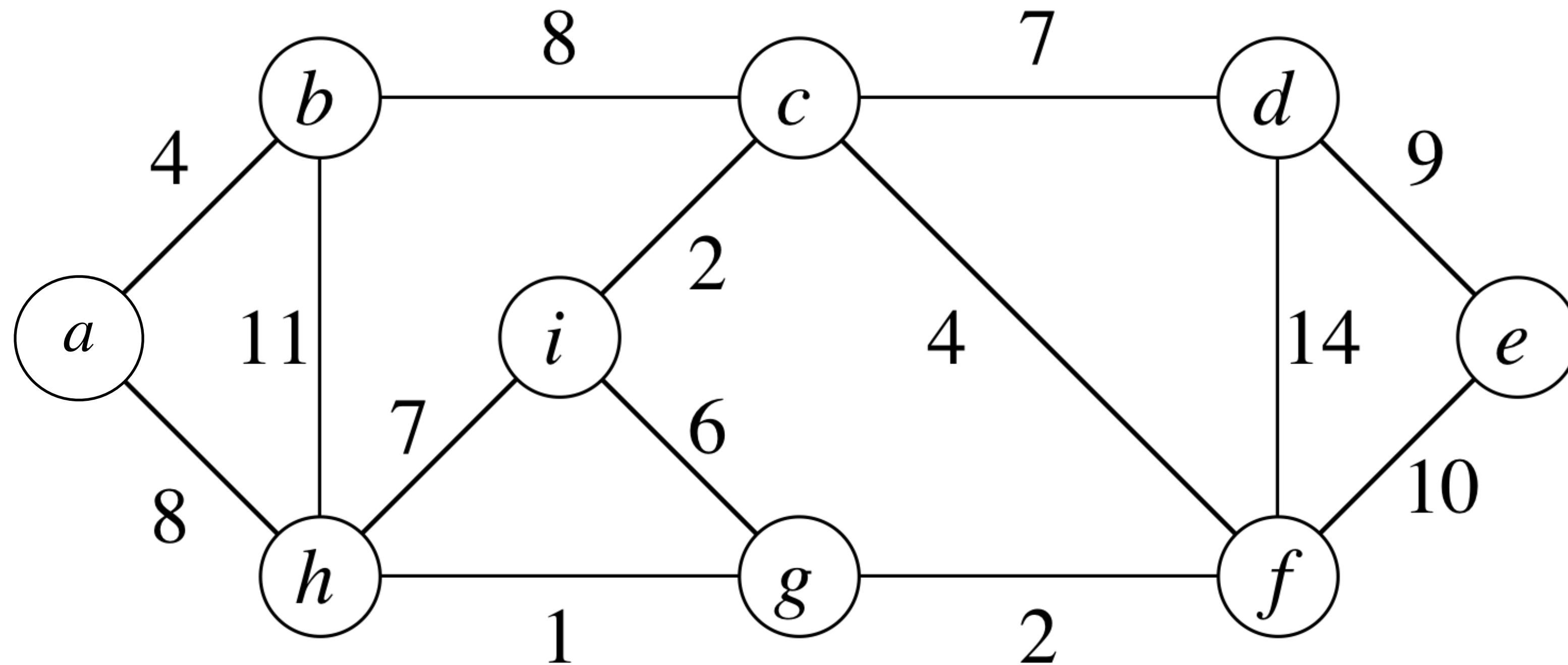
while A is not a spanning tree

 find an edge (u, v) that is safe for A

$A = A \cup \{(u, v)\}$

return A

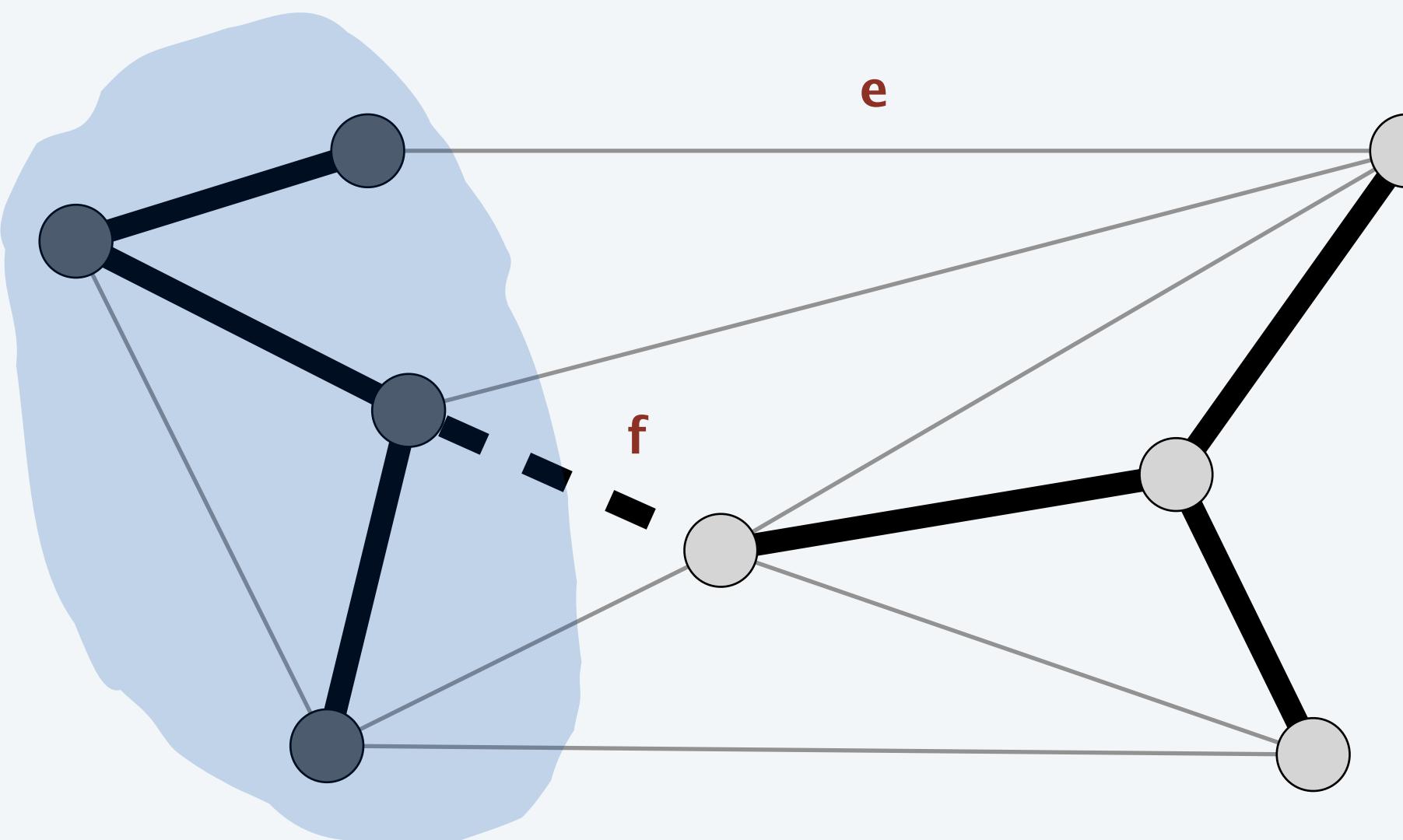
Eksempel på sikre kanter



Fundamental snitegenskab

Fundamental snitegenskab. Lad $H = (V, T)$ være et spændetræ for $G = (V, E)$:

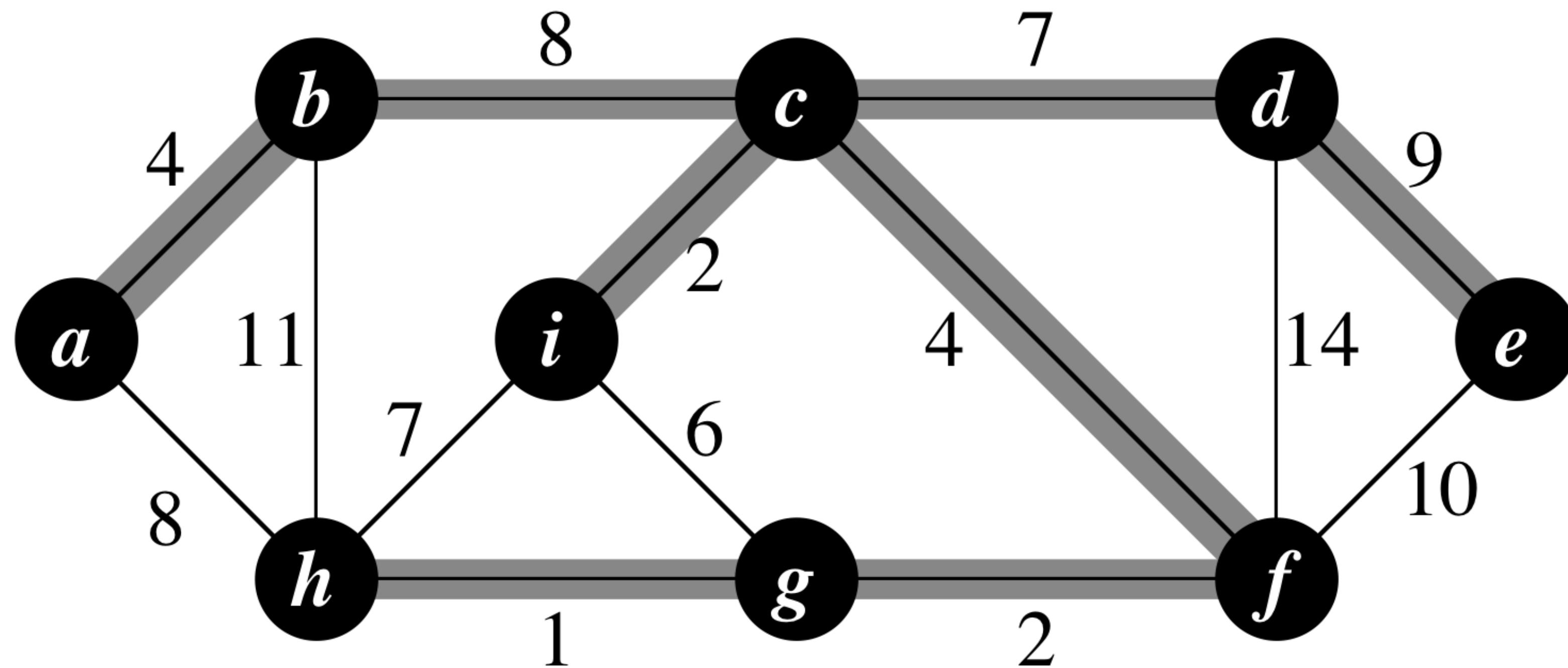
- For enhver kant i træet $f \in T$ har $(V, T - \{f\})$ to sammenhængskomponenter
- Lad D betegne den tilsvarende snitmængde
- For enhver kant $e \in D$ er $(V, T - \{f\} \cup \{e\})$ et spændetræ



Observation 1. Hvis $c_e < c_f$
så er (V, T) ikke et MST

Observation 2. En kant der
har **minimal vægt** i en
snitmængde D er altid sikker
for A hvis $D \cap A = \emptyset$.

Eksempel på snitegenskab



Stigler's law of eponymy

From Wikipedia, the free encyclopedia

Stigler's law of eponymy, proposed by University of Chicago statistics professor Stephen Stigler in his 1980 publication *Stigler's law of eponymy*,^[1] states that no scientific discovery is named after its original discoverer. Examples include Hubble's law, which was derived by Georges Lemaître two years before Edwin Hubble, the Pythagorean theorem, which was known to Babylonian mathematicians before Pythagoras, and Halley's Comet, which was observed by astronomers since at least 240 BC (although its official designation is due to the first ever mathematical prediction of such astronomical phenomenon in the sky, not to its discovery). Stigler himself named the sociologist Robert K. Merton as the discoverer of "Stigler's law" to show that it follows its own decree, though the phenomenon had previously been noted by others.^[2]

Prim's algorithm

From Wikipedia, the free encyclopedia

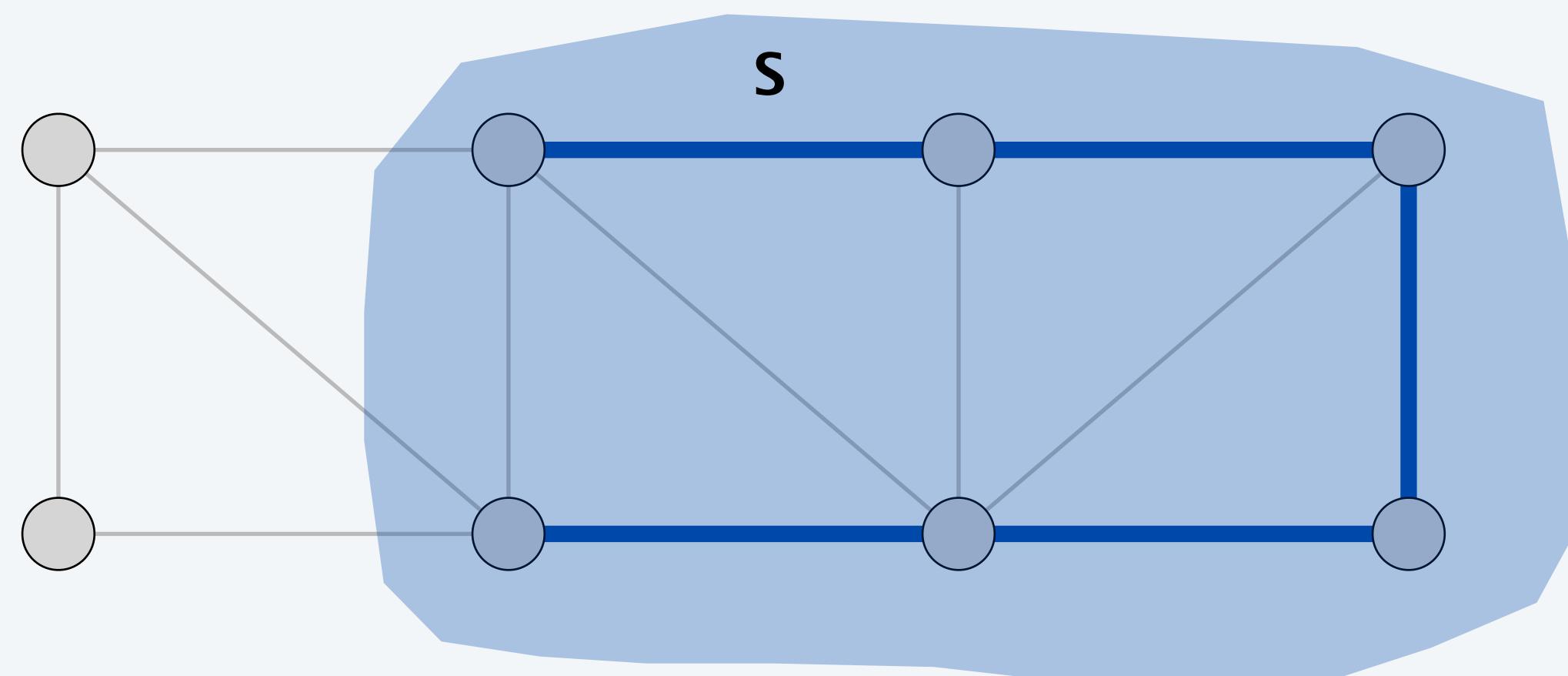
In computer science, **Prim's algorithm** (also known as Jarník's algorithm) is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.

The algorithm was developed in 1930 by Czech mathematician Vojtěch Jarník^[1] and later rediscovered and republished by computer scientists Robert C. Prim in 1957^[2]

Jarník-Prim's MST algoritme

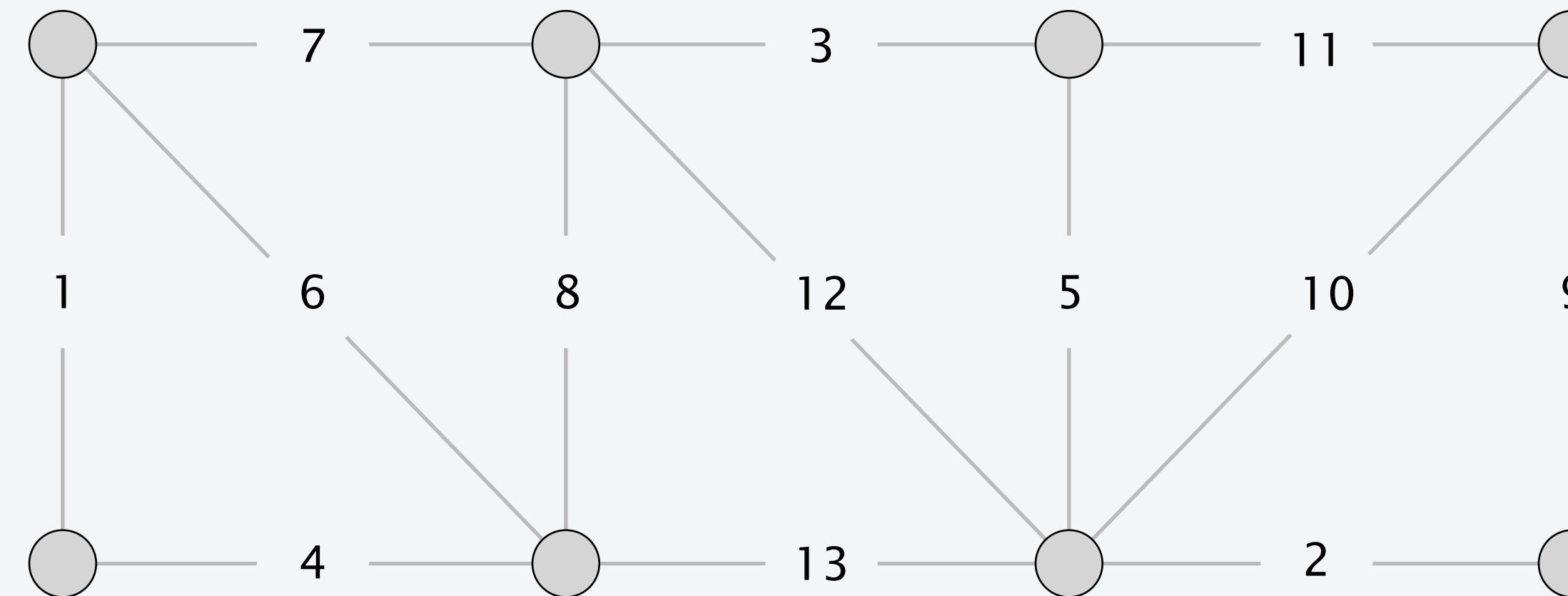
- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S ← Kanten er sikker, fordi den har mindst vægt i snitmængden for S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .

Invariant: Kantmængden A er et MST for knuderne i S .



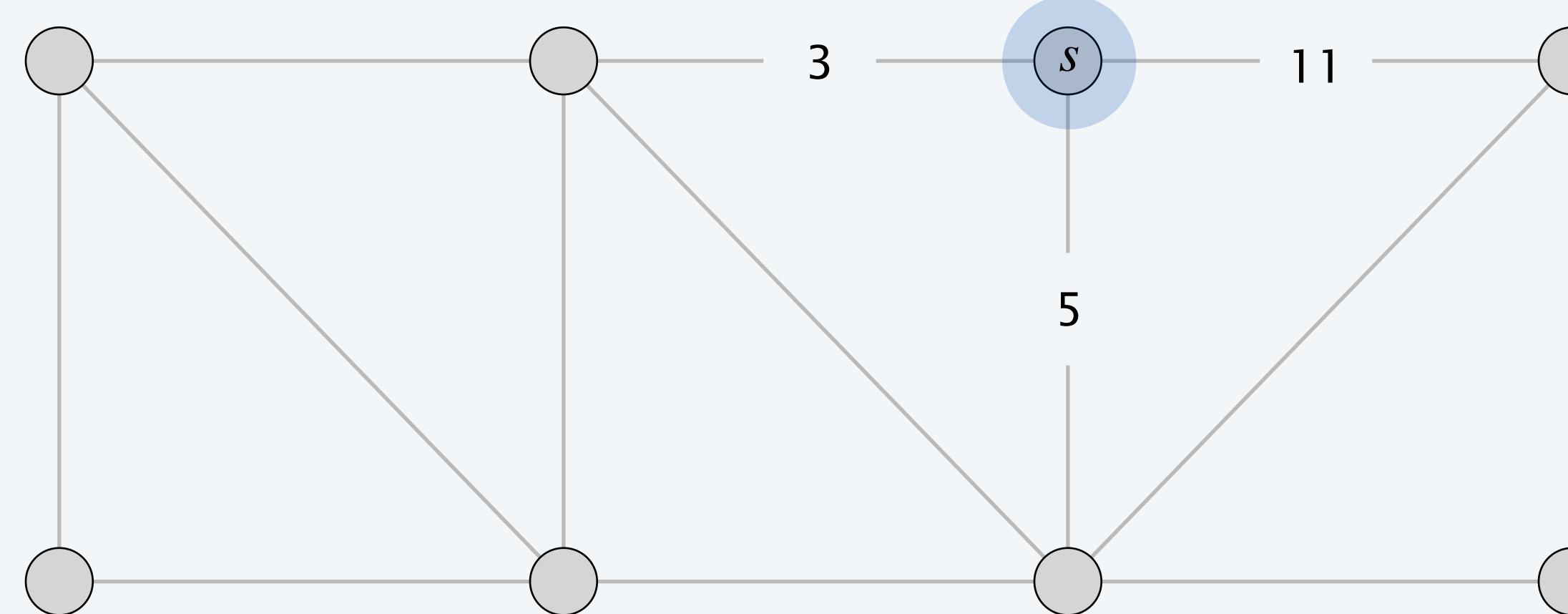
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



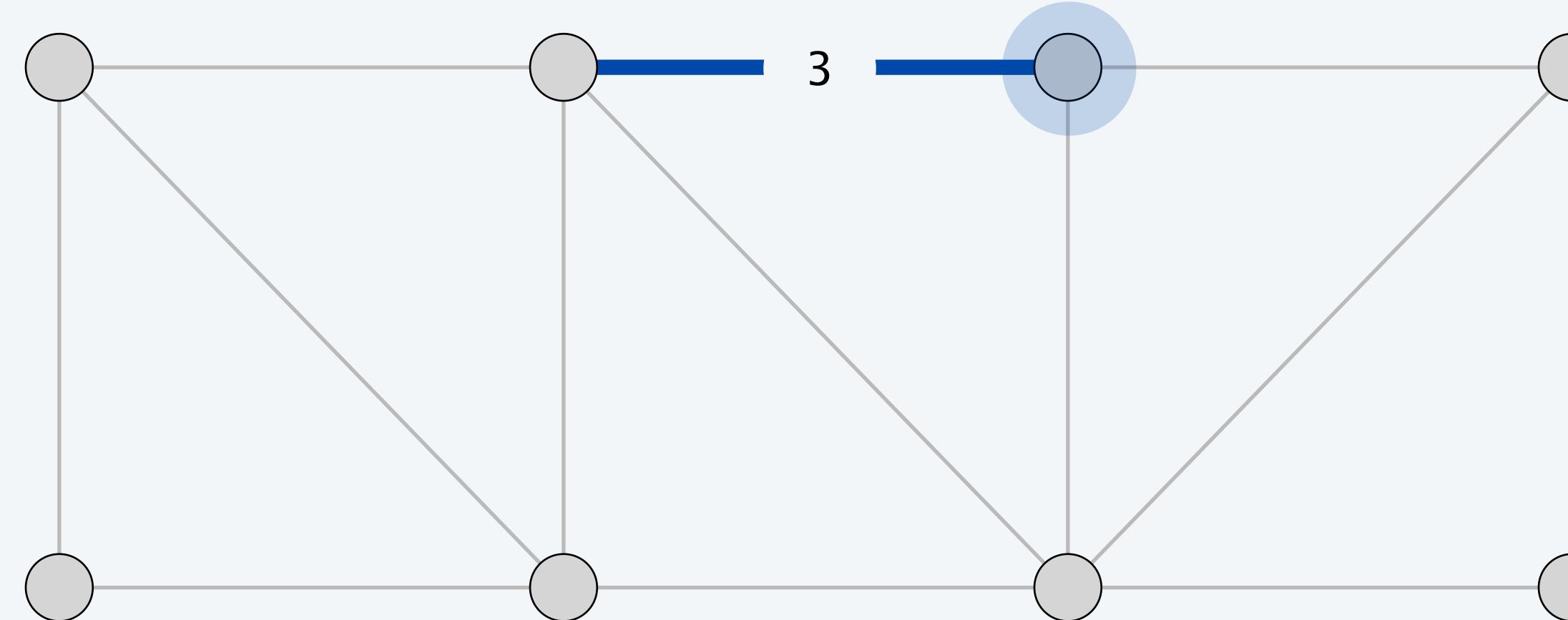
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



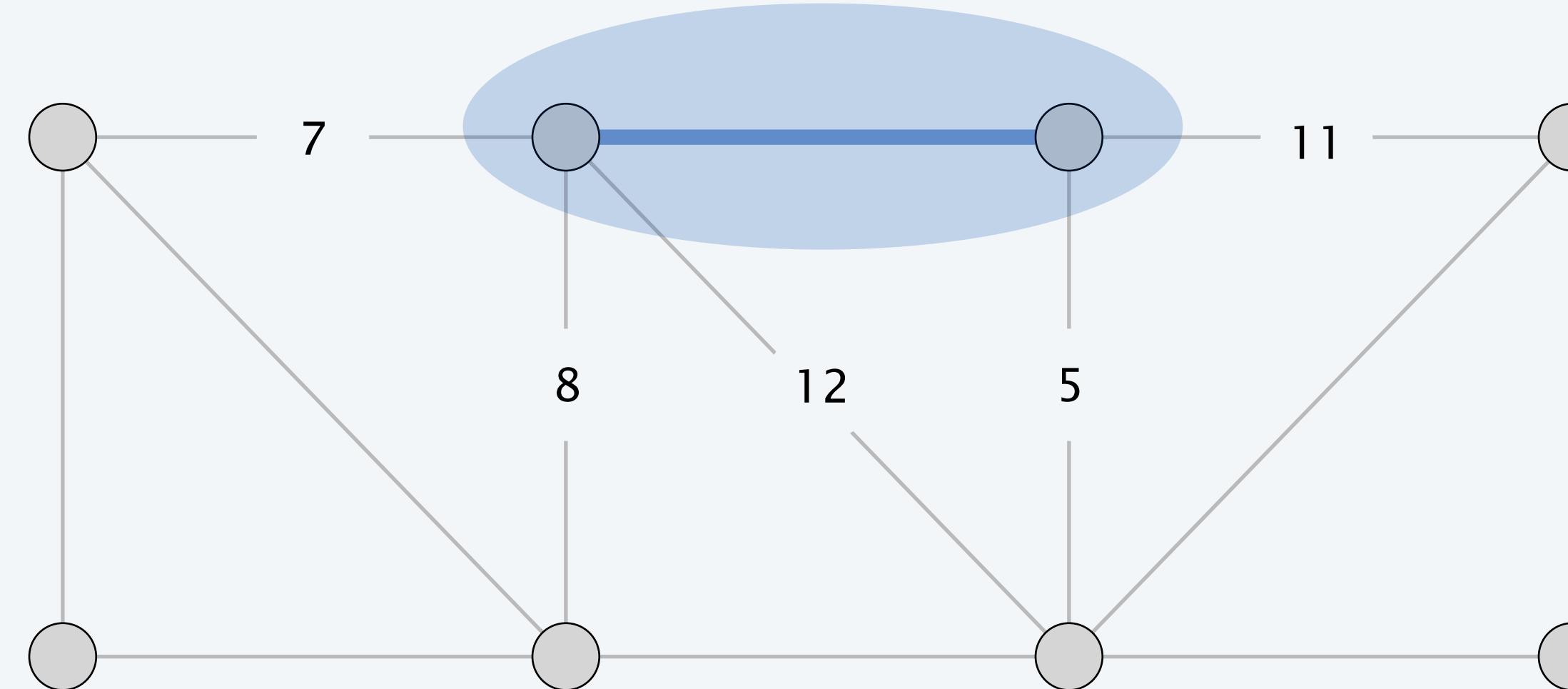
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



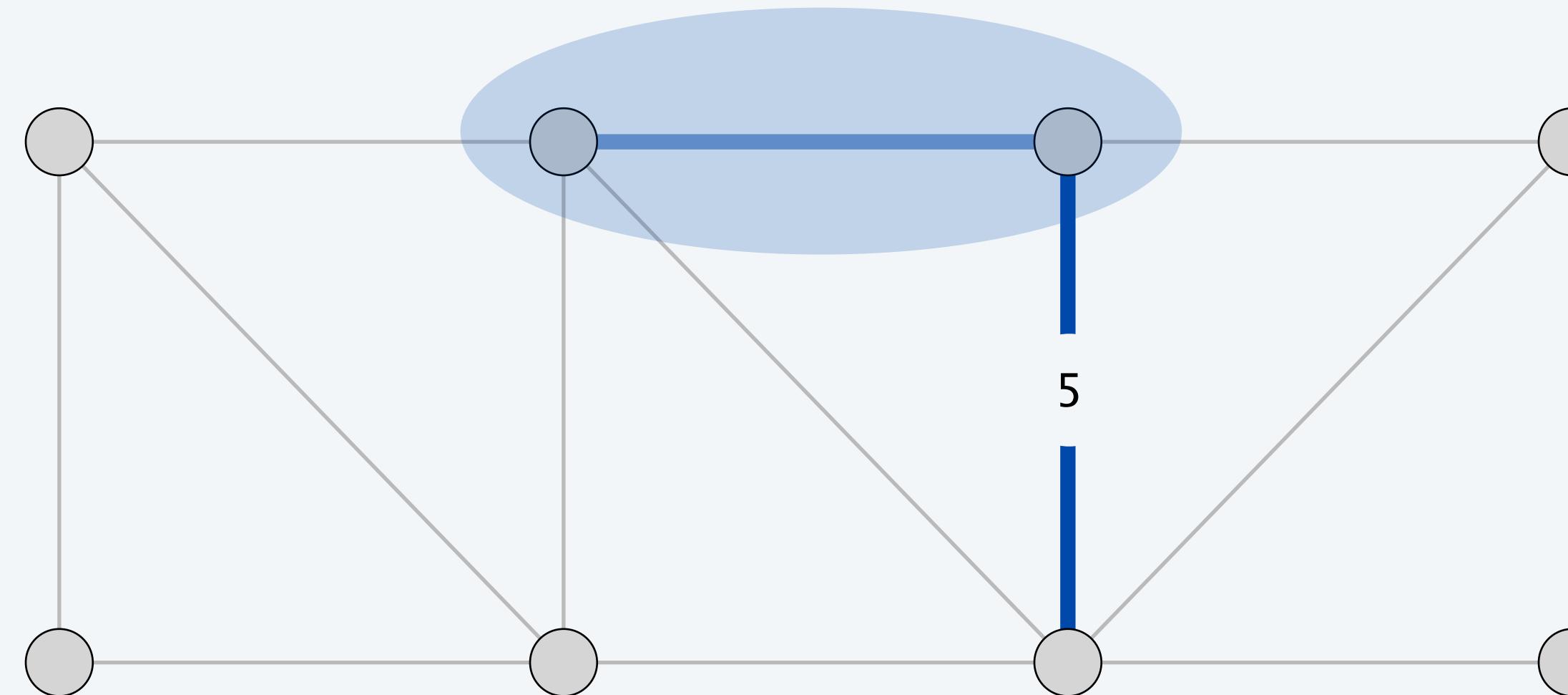
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



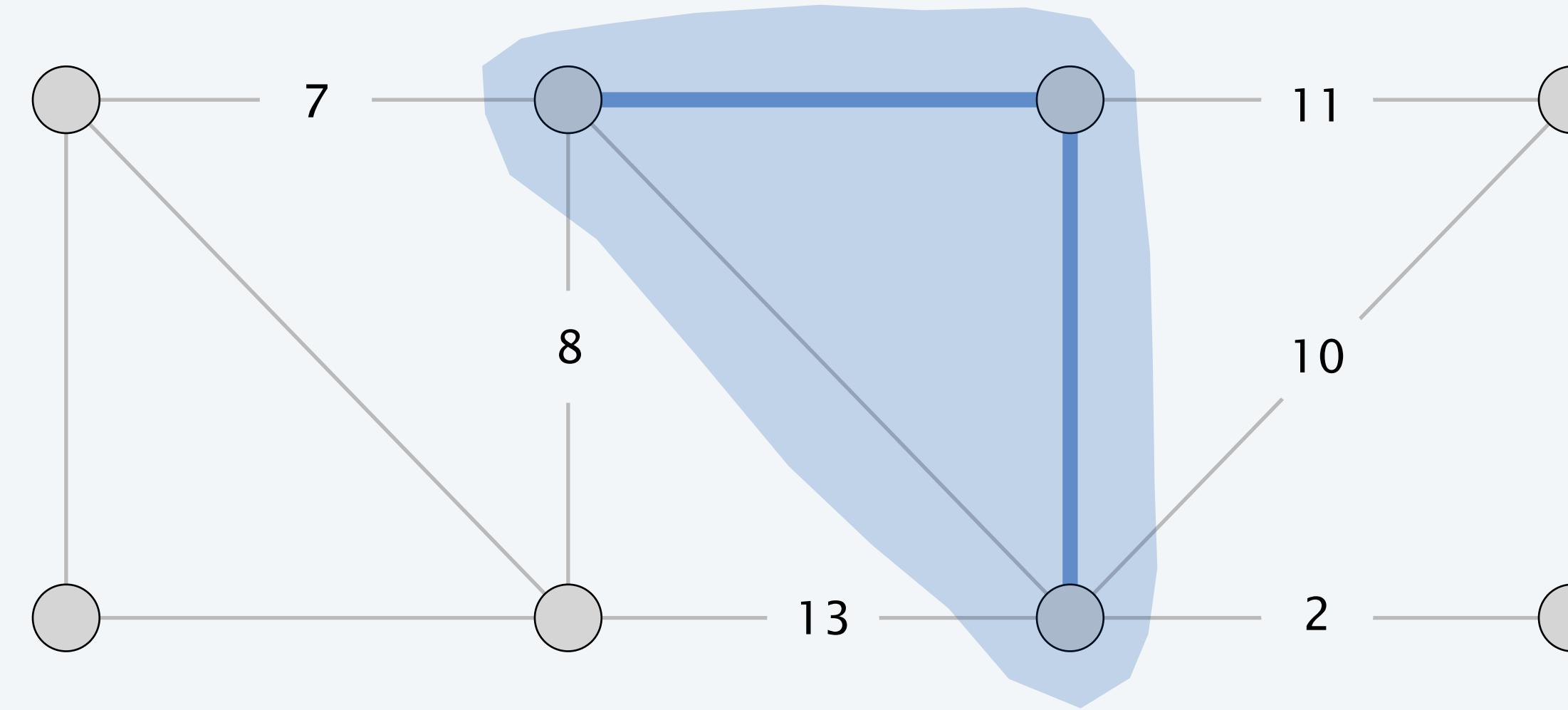
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



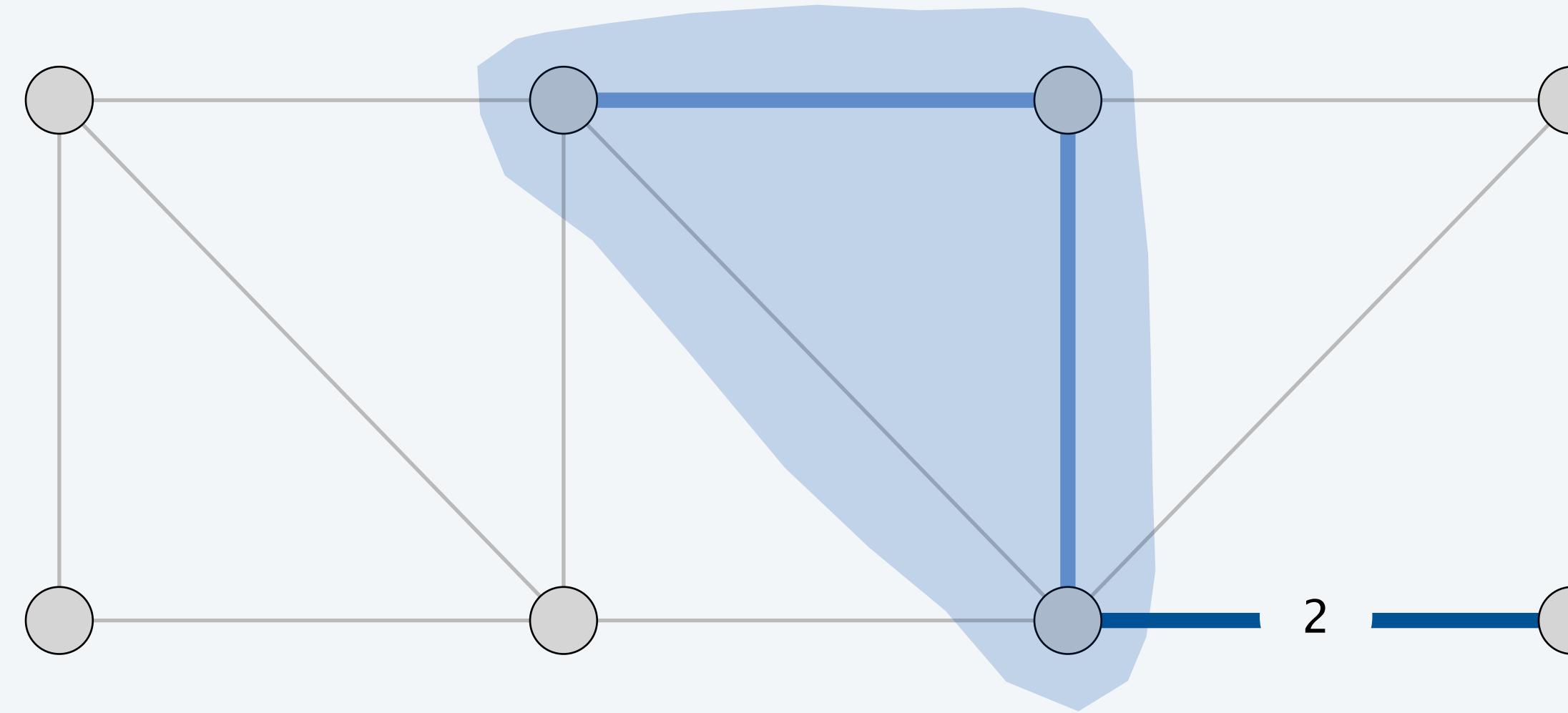
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



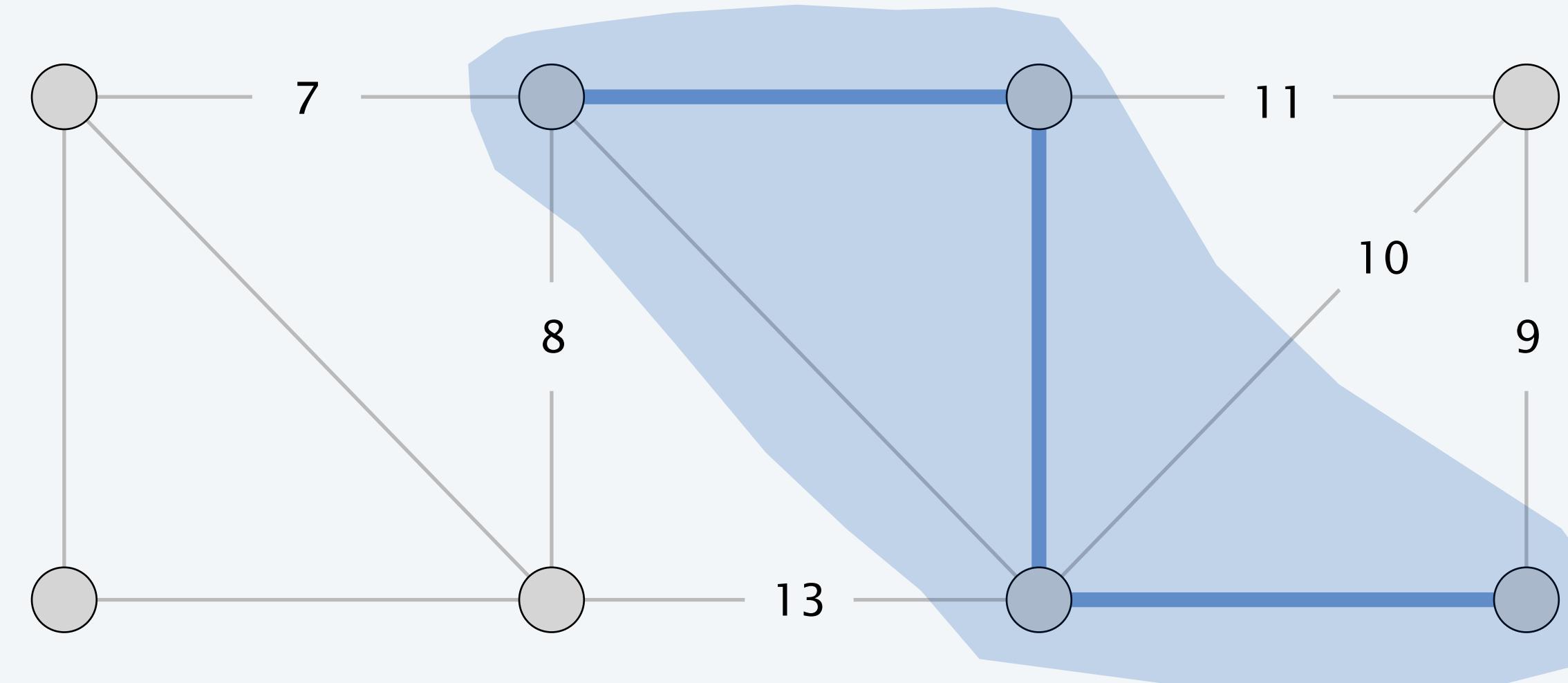
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



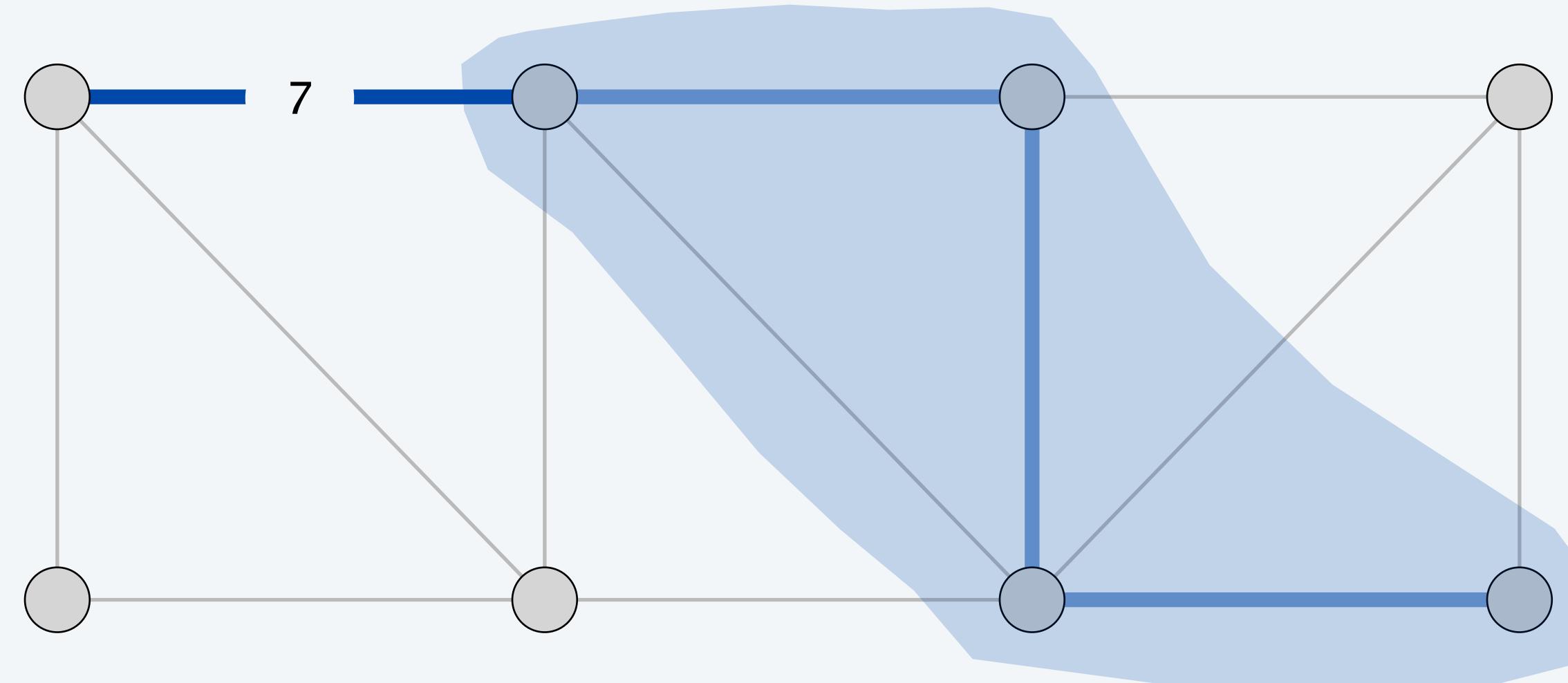
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



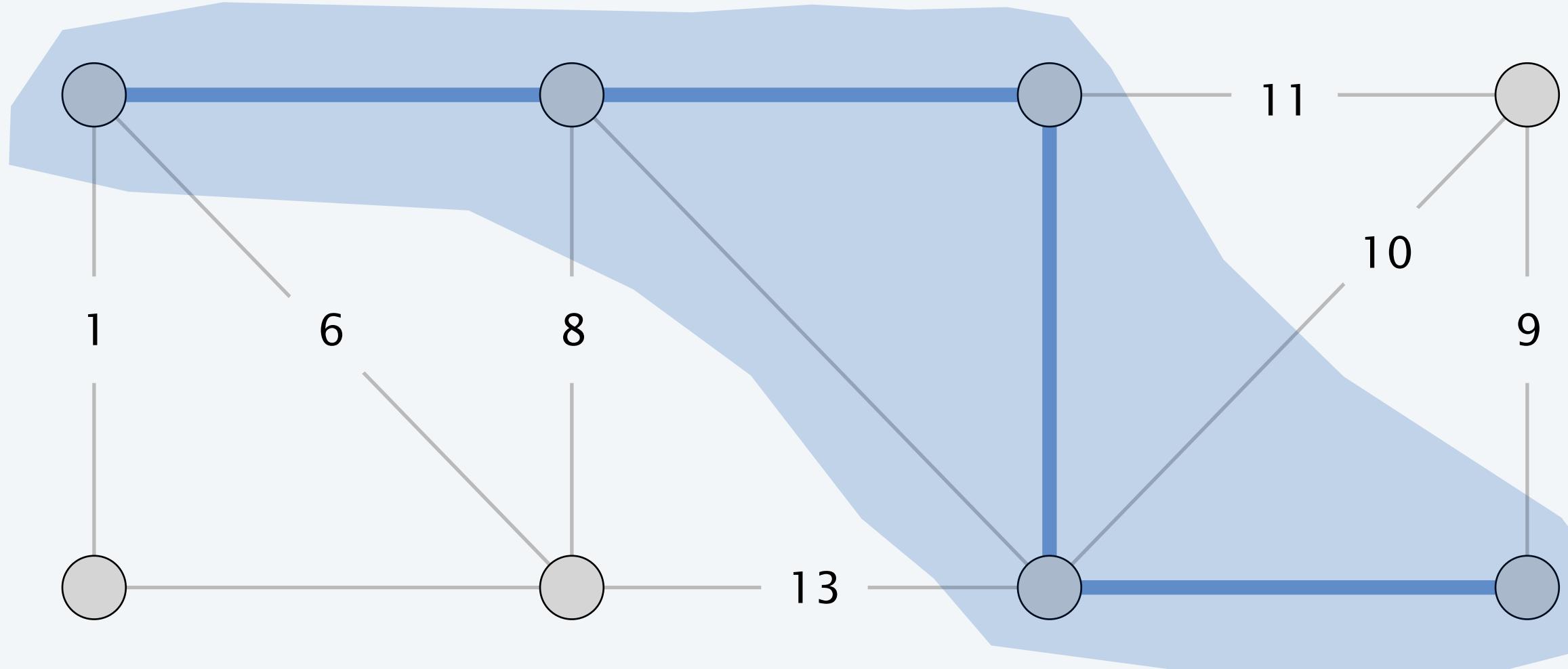
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



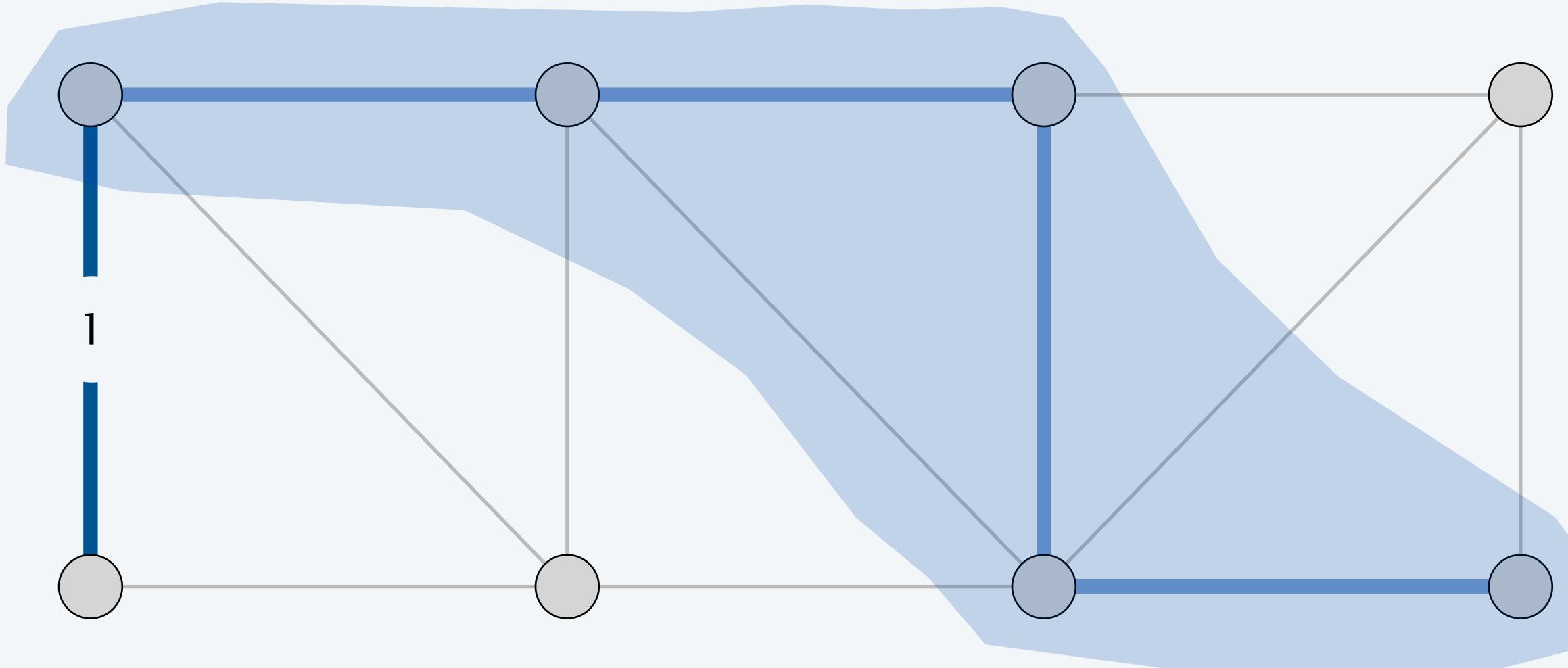
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



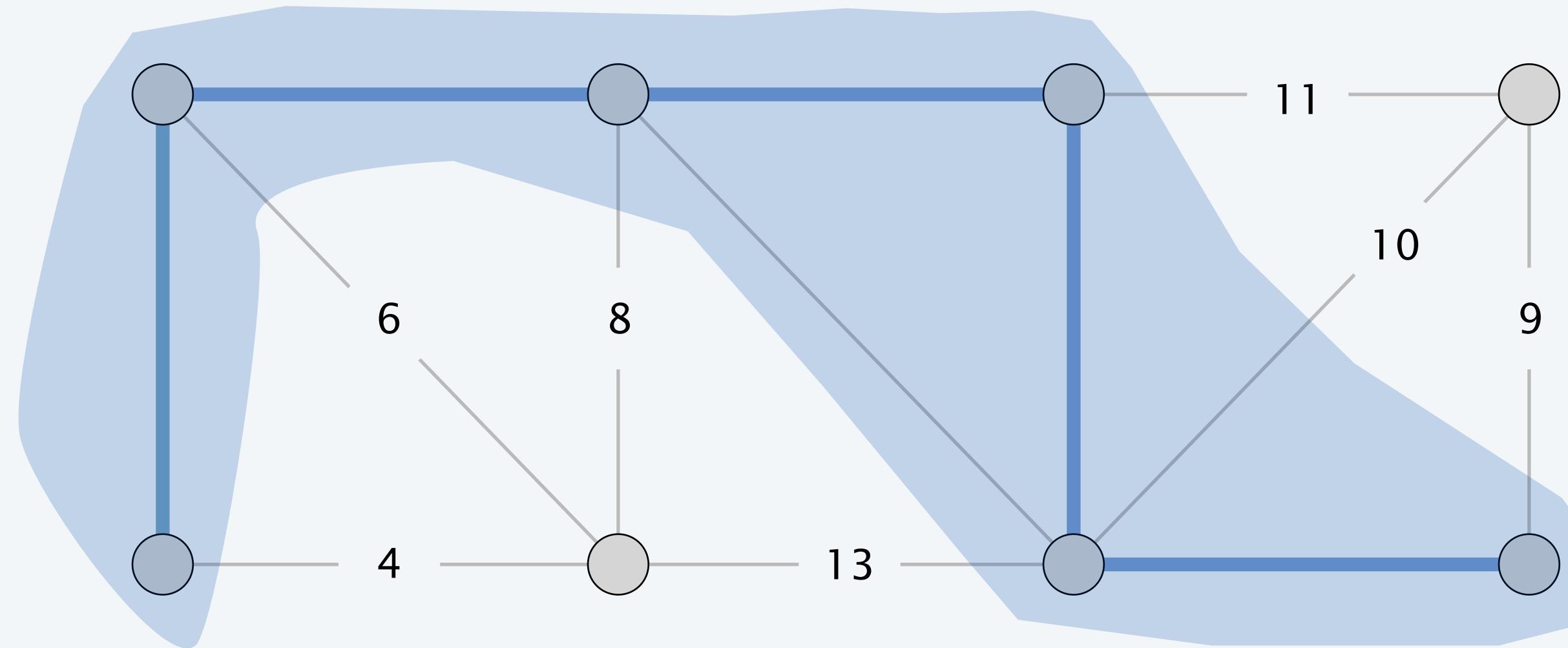
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



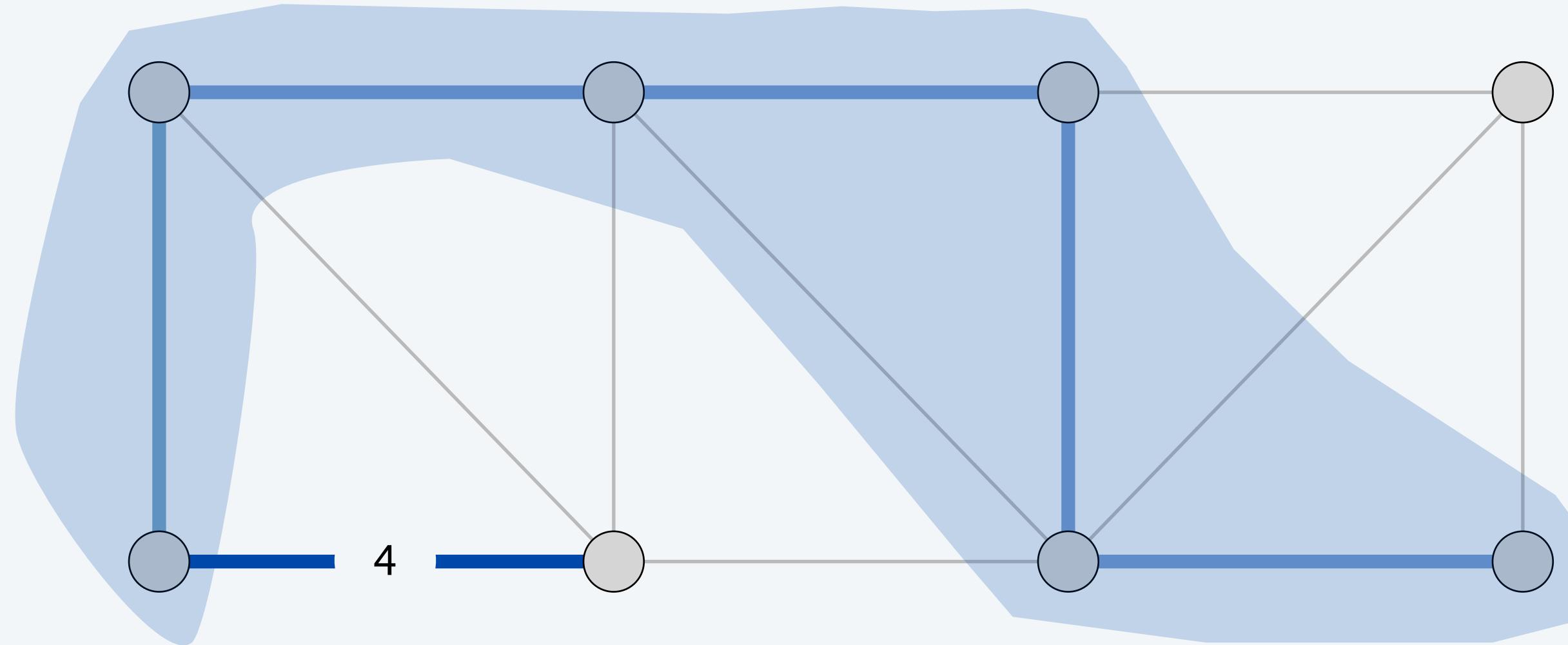
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



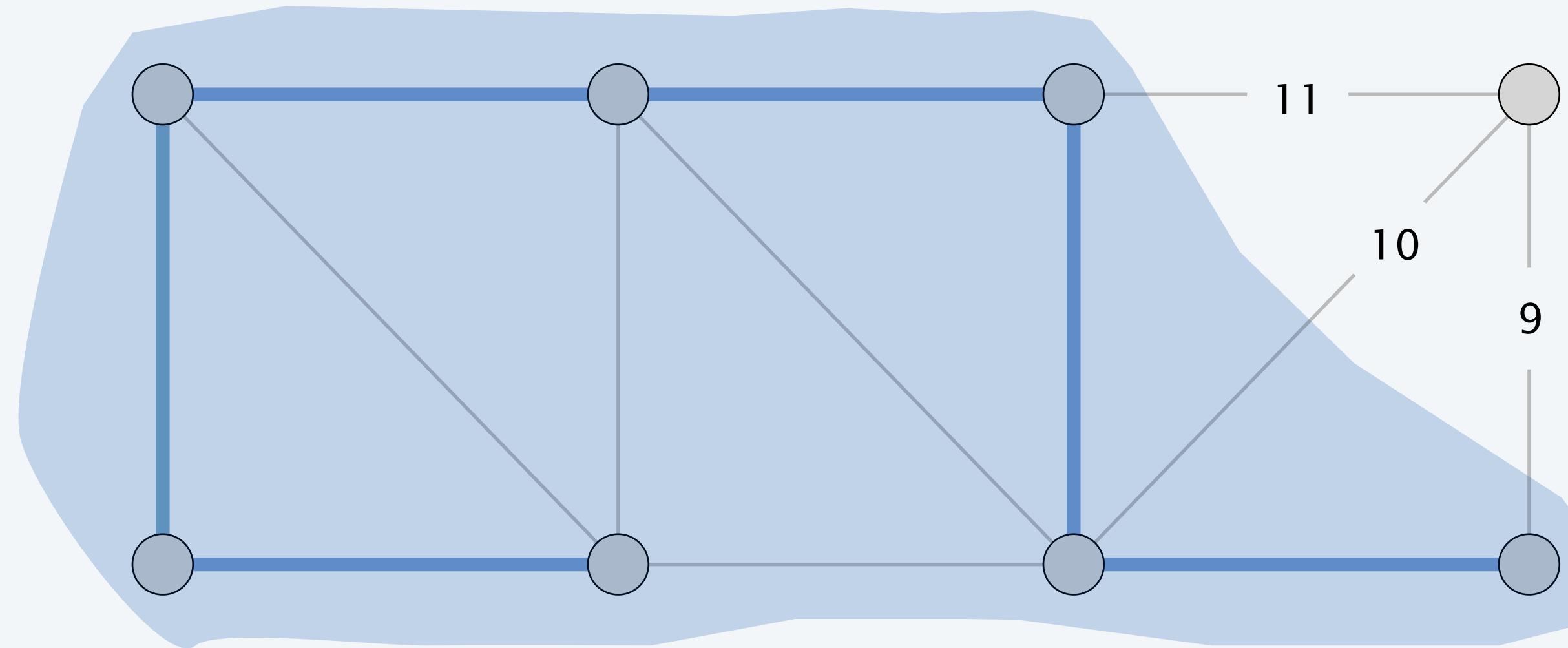
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



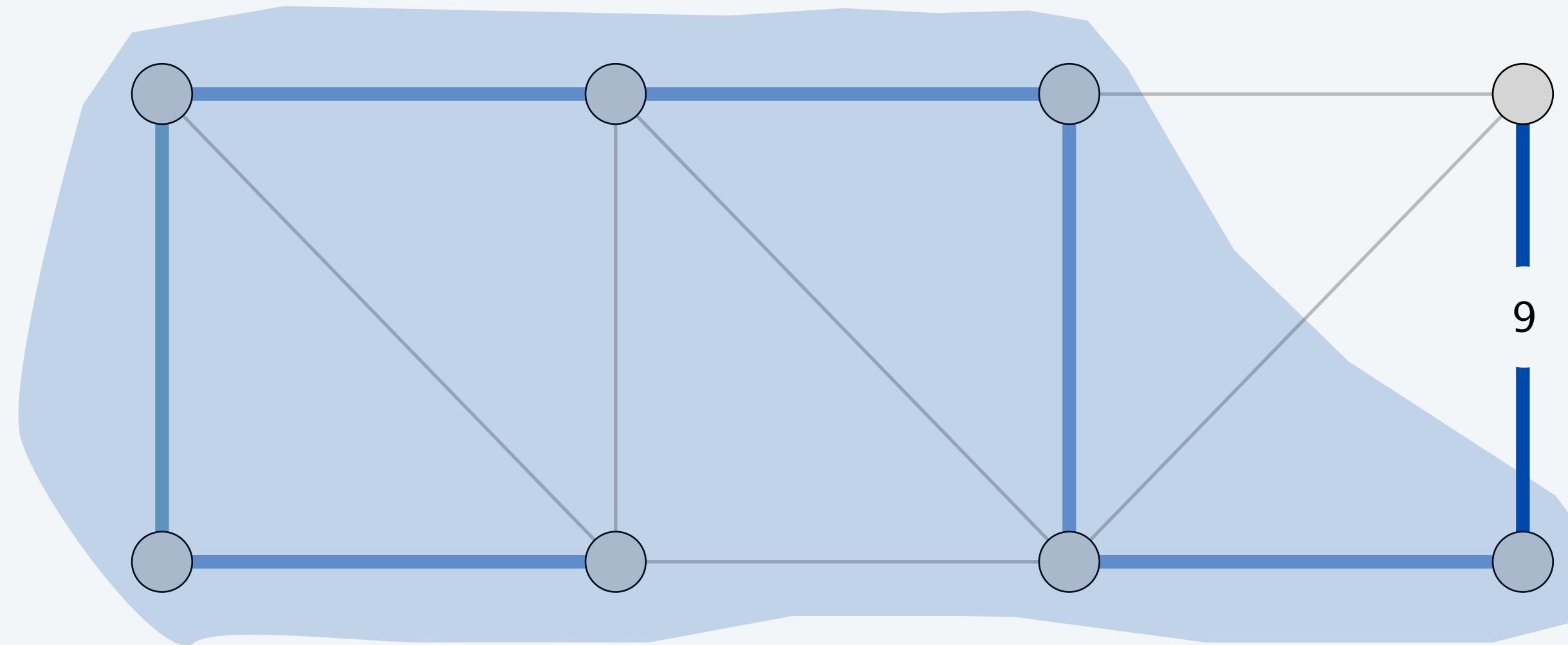
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



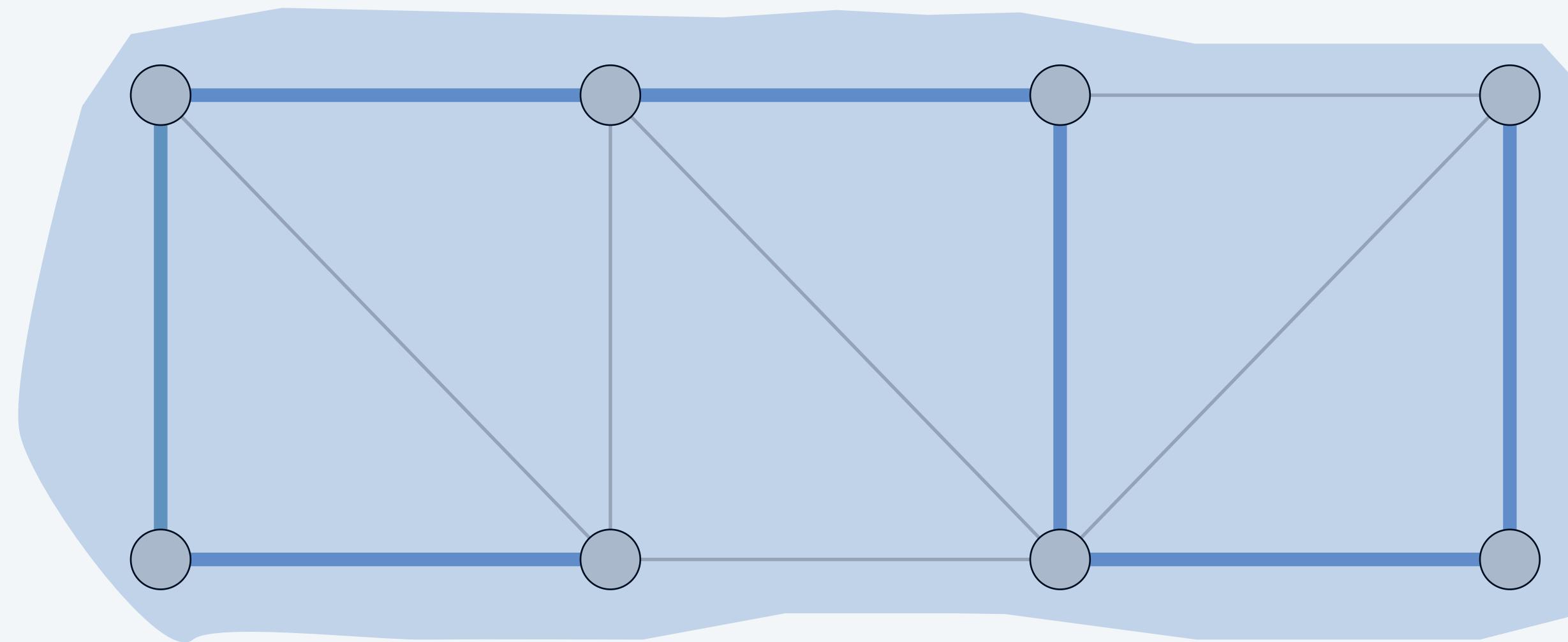
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



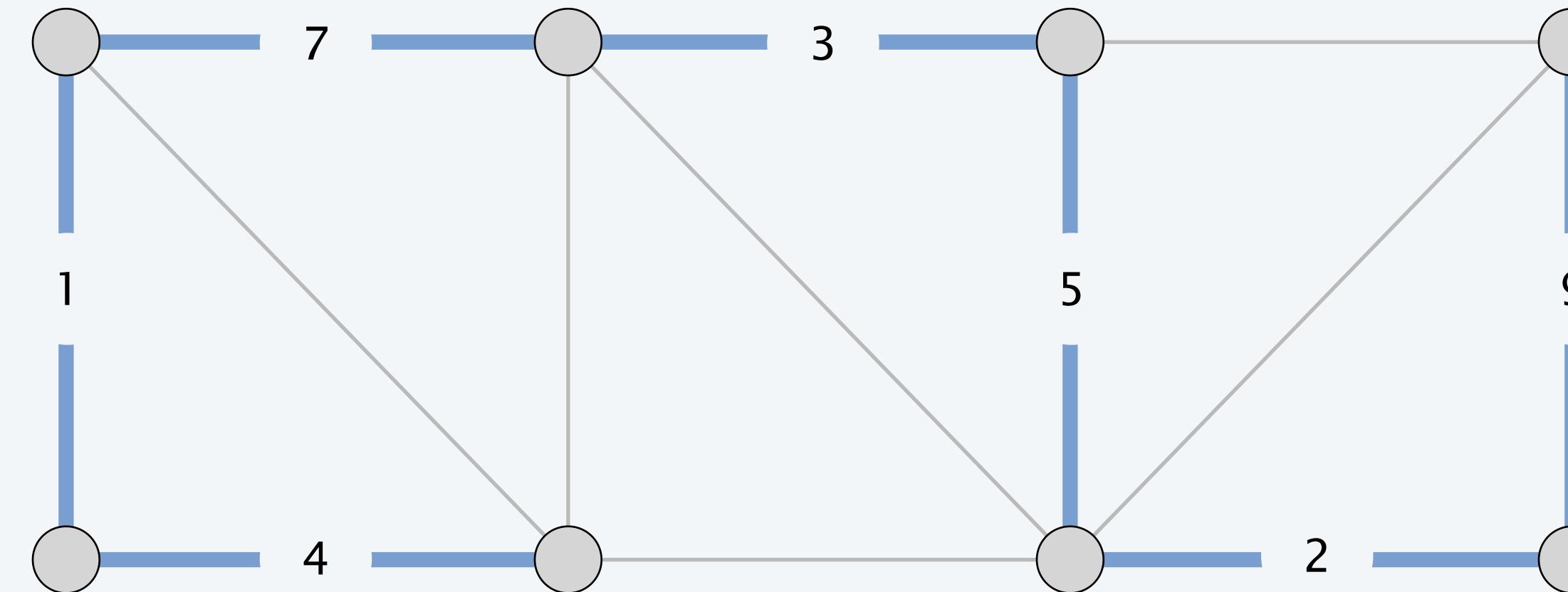
Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



Jarník-Prim's MST algoritme – eksempel

- Initialisér $A = \emptyset$ og definér $S = \{s\}$ hvor s er en vilkårlig knude.
- Gentag $n - 1$ gange:
 - Find en kant e med minimal vægt, der har præcis én knude i S
 - Tilføj kanten e til A
 - Tilføj den nye knude fra e til S .



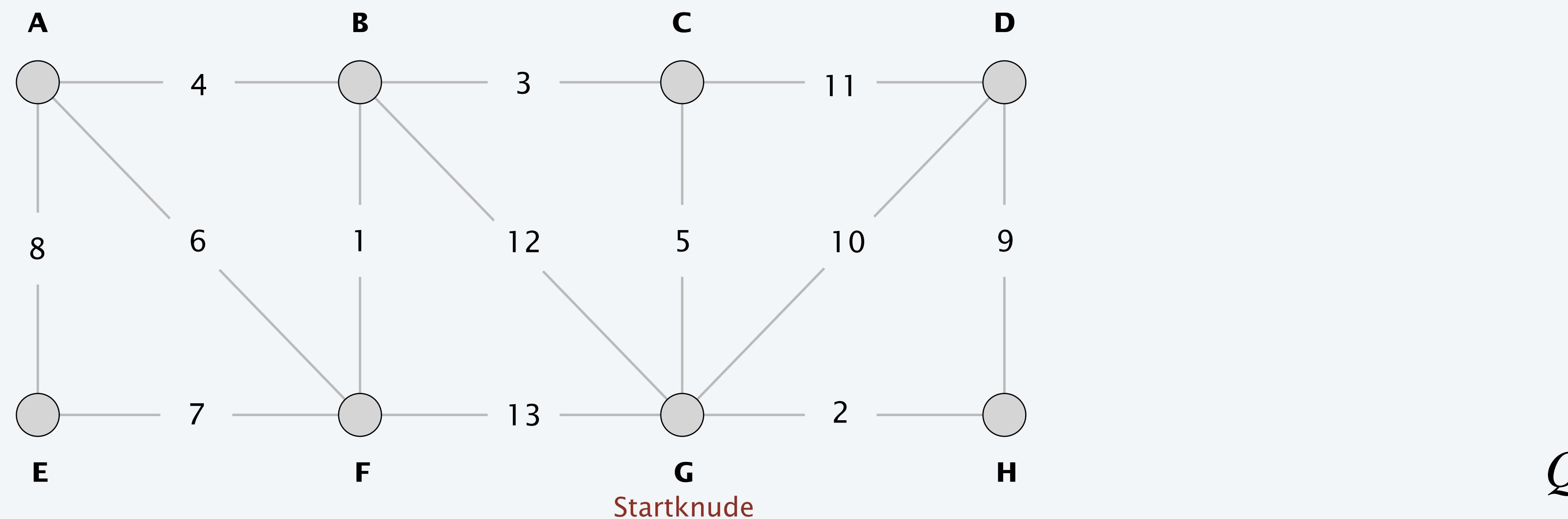
Implementation af Jarník-Prim's algoritme

- Variant af Dijkstra's algoritme: Brug en *prioritetskø* til at finde den letteste kant
- Tidsanalyse:
 $n \times$ Insert
 $m \times$ Decrease-Key
- Med en binær høb:
Tid $O(m \log n)$

```
PRIM( $G, w, r$ )
 $Q = \emptyset$ 
for each  $u \in G.V$ 
     $u.key = \infty$  ← Vægt af letteste kant til  $u$ 
     $u.\pi = \text{NIL}$  ← Kant til  $u$  i et MST
    INSERT( $Q, u$ )
DECREASE-KEY( $Q, r, 0$ ) ← Startknude  $r$ 
while  $Q \neq \emptyset$ 
     $u = \text{EXTRACT-MIN}(Q)$ 
    for each  $v \in G.Adj[u]$ 
        if  $v \in Q$  and  $w(u, v) < v.key$ 
             $v.\pi = u$ 
            DECREASE-KEY( $Q, v, w(u, v)$ )
```

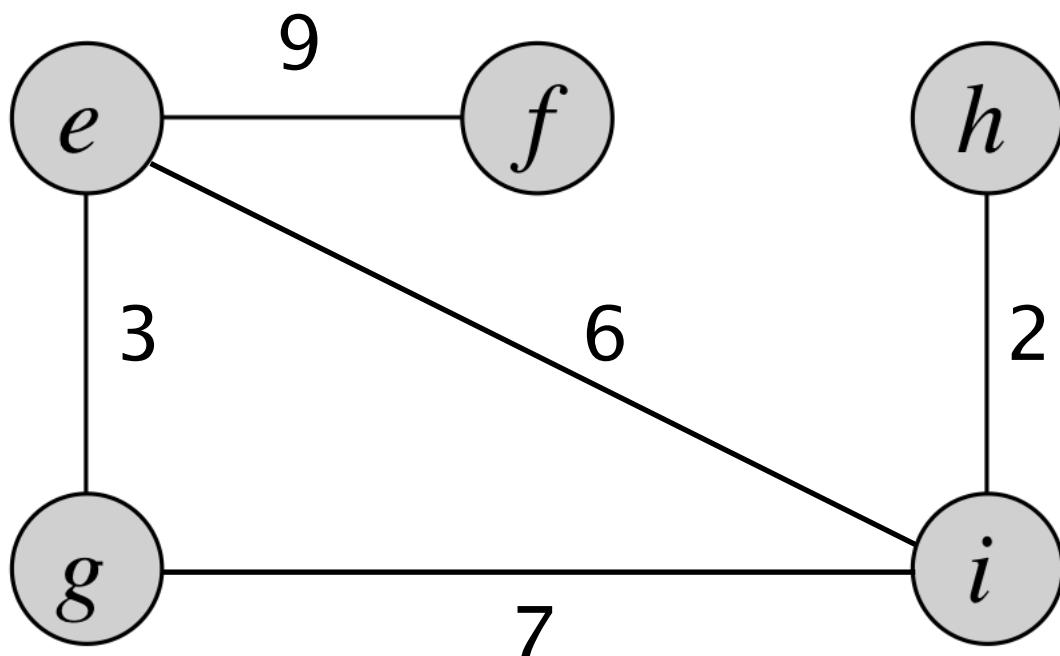
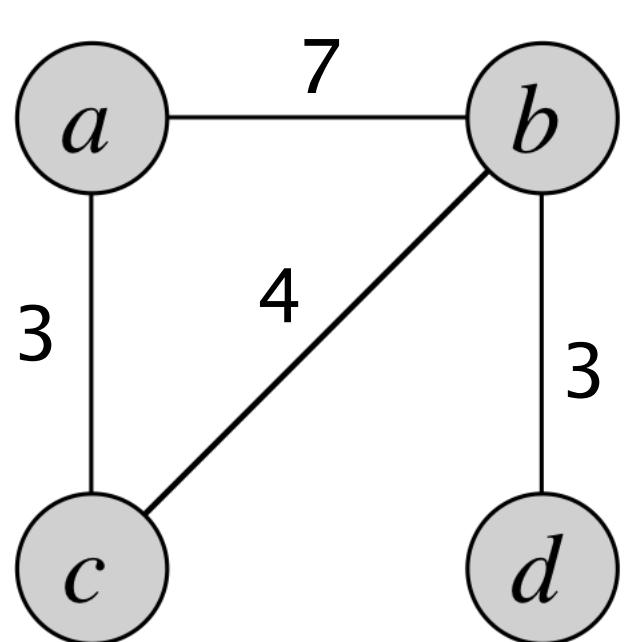
Forskel til Dijkstra's
algoritme er vægten hér

Jarník-Prim's MST algoritme – eksempel med prioritetskø



Øvelse

- Antag at din graf ikke er sammenhængende
- Hvad beregner Jarník-Prim?
(Prøv evt. på eksemplet nedenfor)



```
PRIM( $G, w, r$ )
 $Q = \emptyset$ 
for each  $u \in G.V$ 
     $u.key = \infty$ 
     $u.\pi = \text{NIL}$ 
    INSERT( $Q, u$ )
    DECREASE-KEY( $Q, r, 0$ )      //  $r.key = 0$ 
while  $Q \neq \emptyset$ 
     $u = \text{EXTRACT-MIN}(Q)$ 
    for each  $v \in G.Adj[u]$ 
        if  $v \in Q$  and  $w(u, v) < v.key$ 
             $v.\pi = u$ 
            DECREASE-KEY( $Q, v, w(u, v)$ )
```

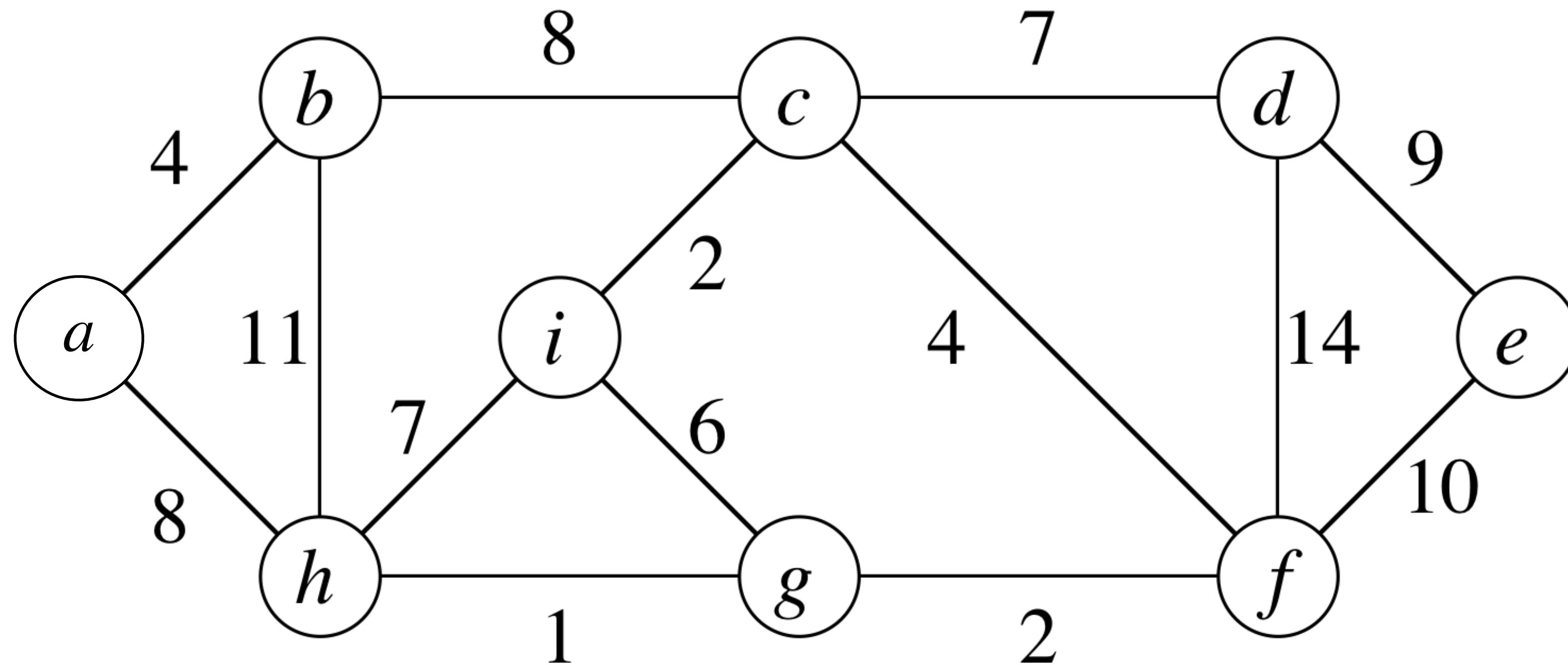
Kruskal's algorithm

From Wikipedia, the free encyclopedia

Kruskal's algorithm finds a [minimum spanning forest](#) of an undirected [edge-weighted graph](#). If the graph is [connected](#), it finds a [minimum spanning tree](#). (A minimum spanning tree of a connected graph is a subset of the [edges](#) that forms a tree that includes every [vertex](#), where the sum of the [weights](#) of all the edges in the tree is minimized. For a disconnected graph, a minimum spanning forest is composed of a minimum spanning tree for each [connected component](#).) It is a [greedy algorithm](#) in [graph theory](#) as in each step it adds the next lowest-weight edge that will not form a [cycle](#) to the minimum spanning forest.^[1]

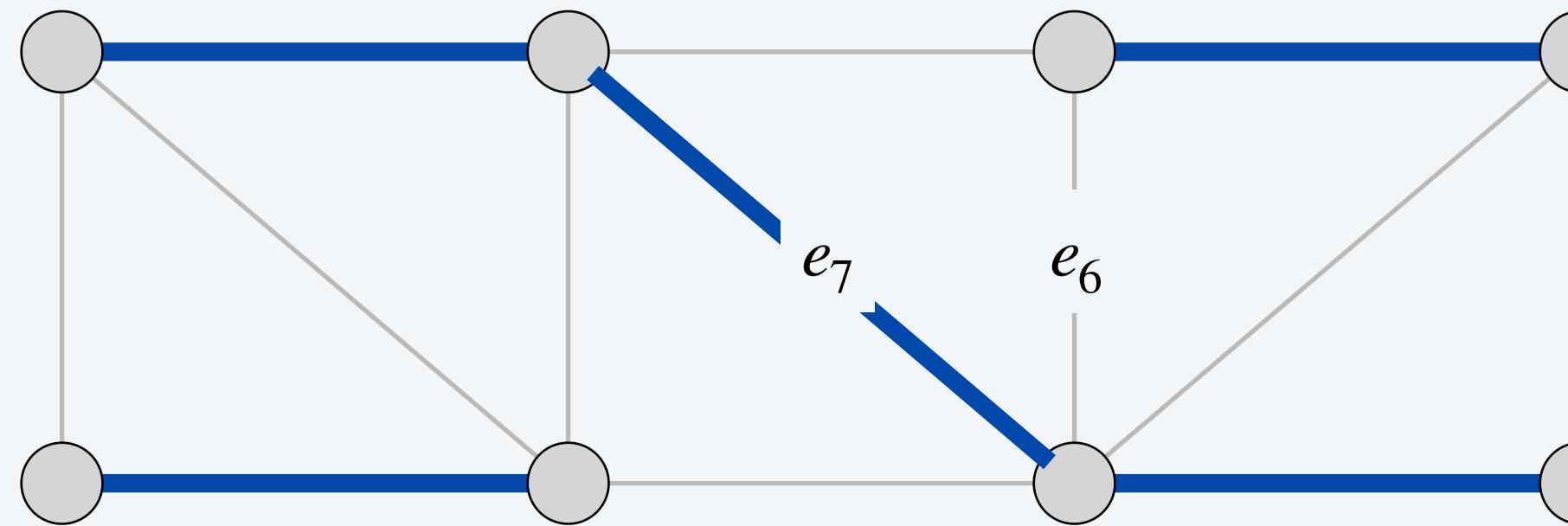
This algorithm first appeared in *Proceedings of the American Mathematical Society*, pp. 48–50 in 1956, and was written by Joseph Kruskal.^[2]

Eksempel på sikre kanter



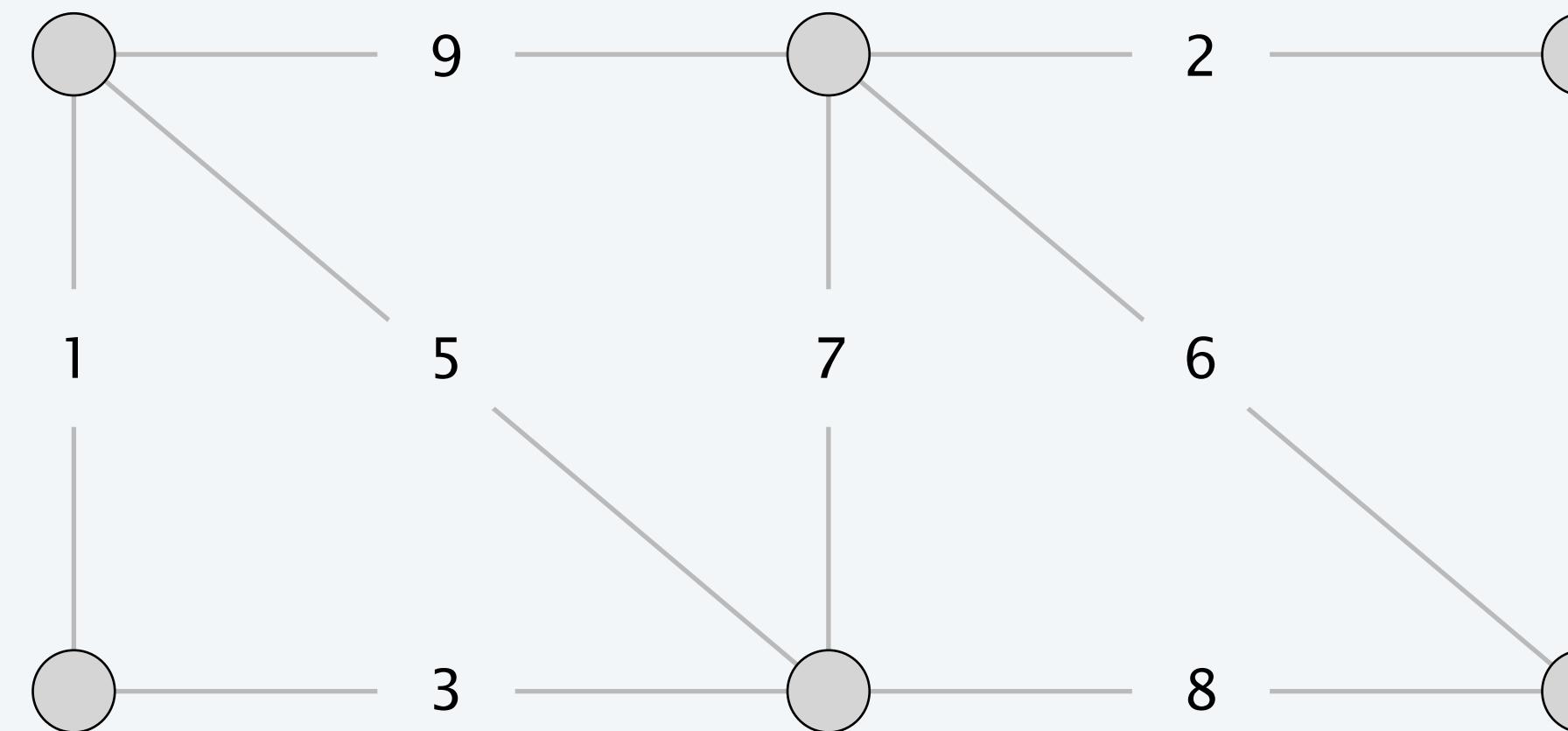
Kruskal's algoritme

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



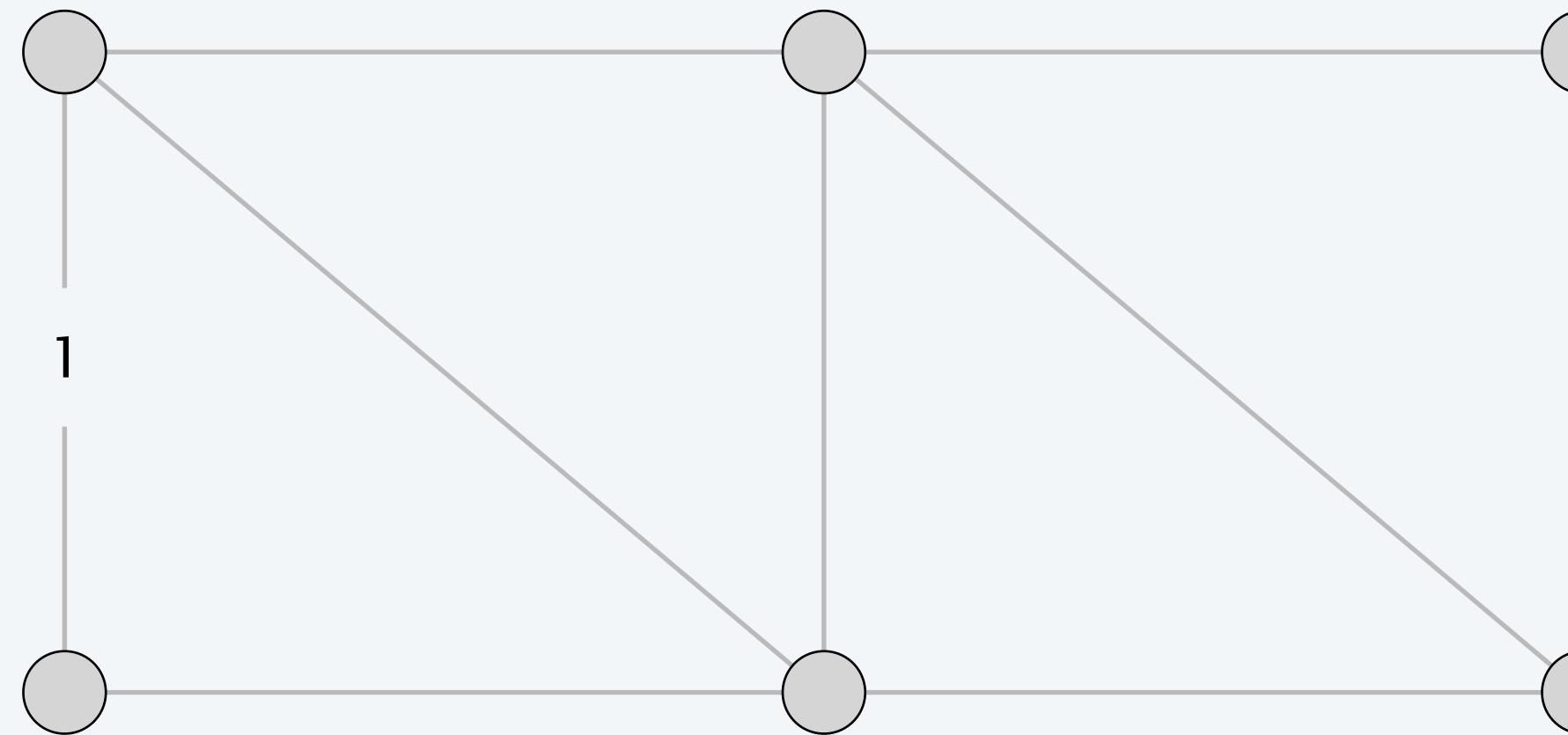
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



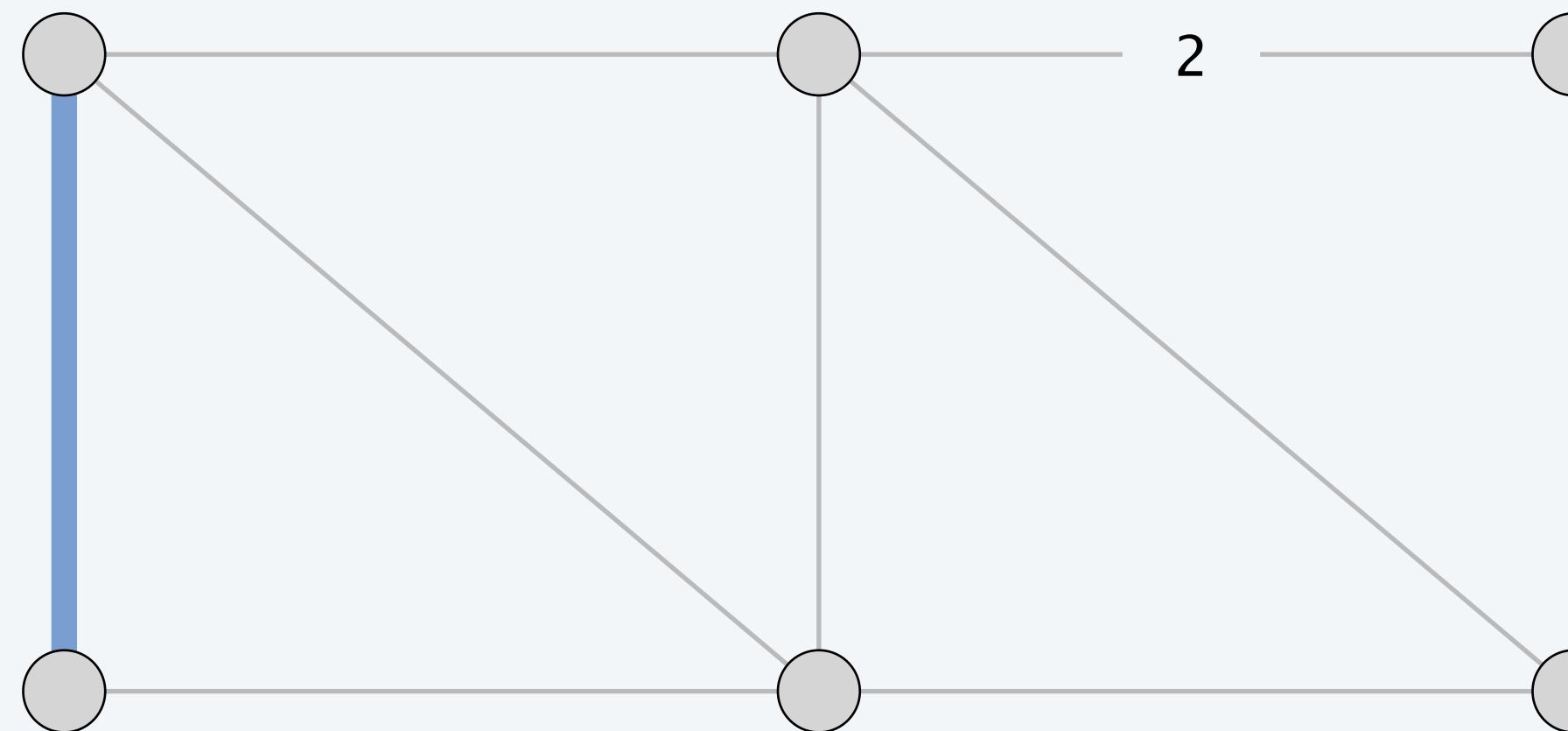
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



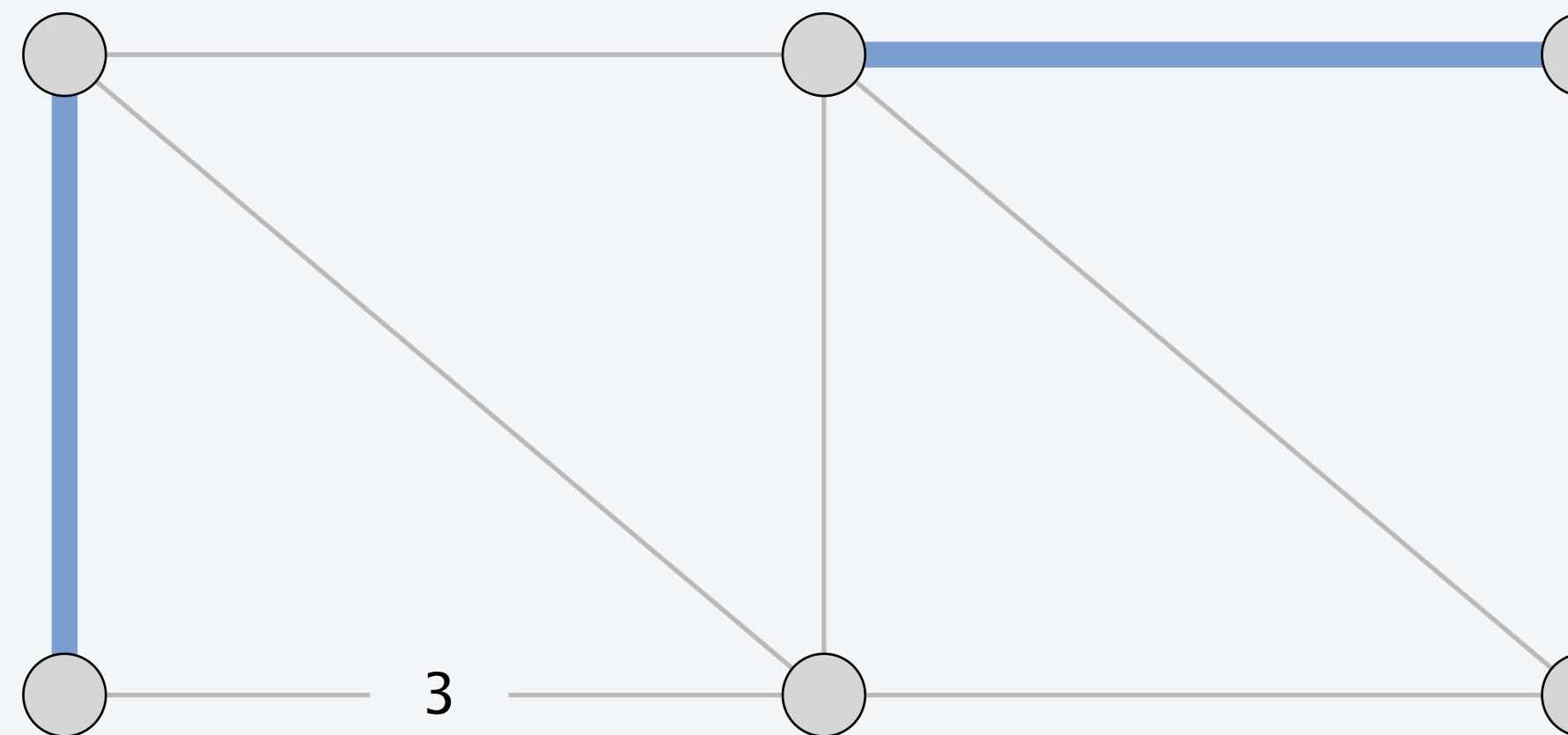
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



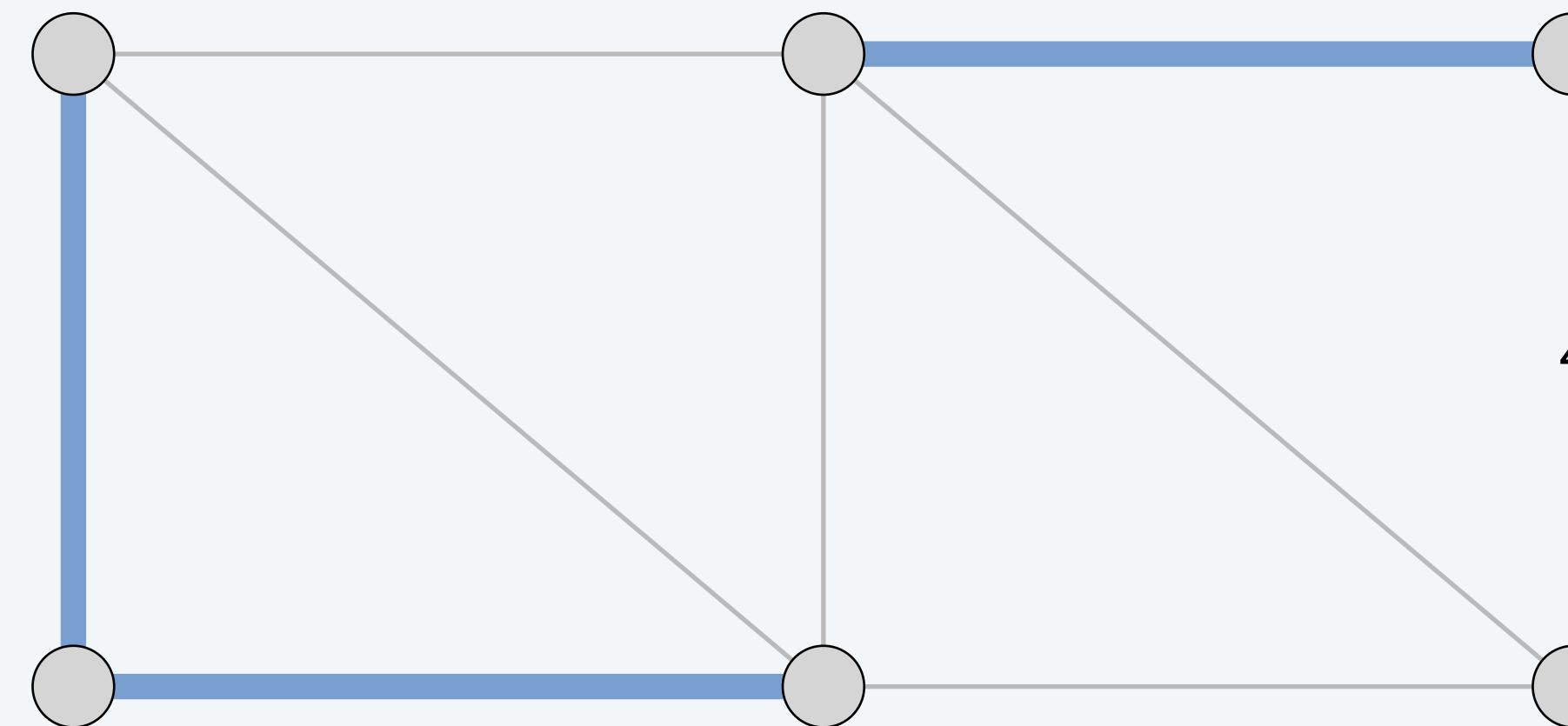
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



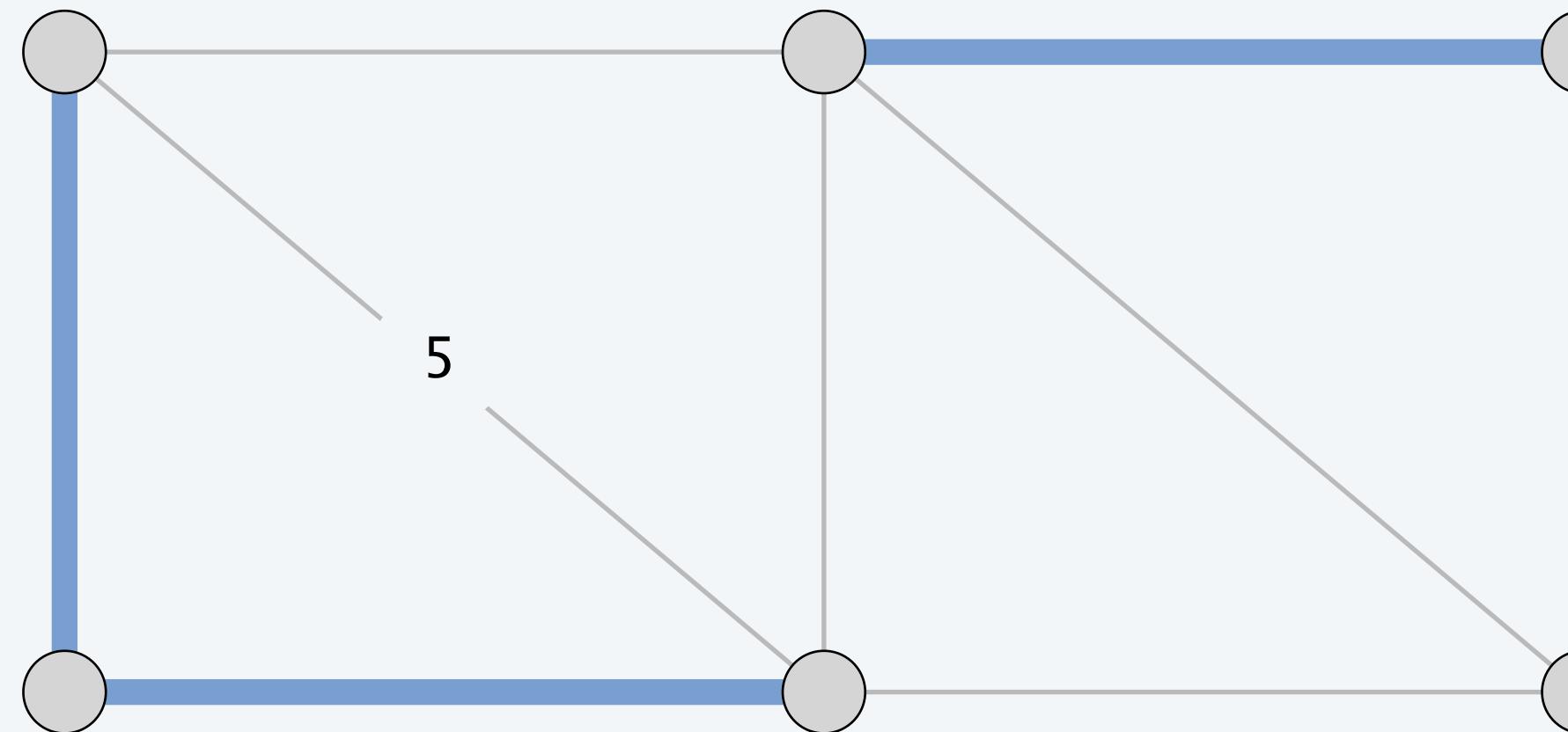
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



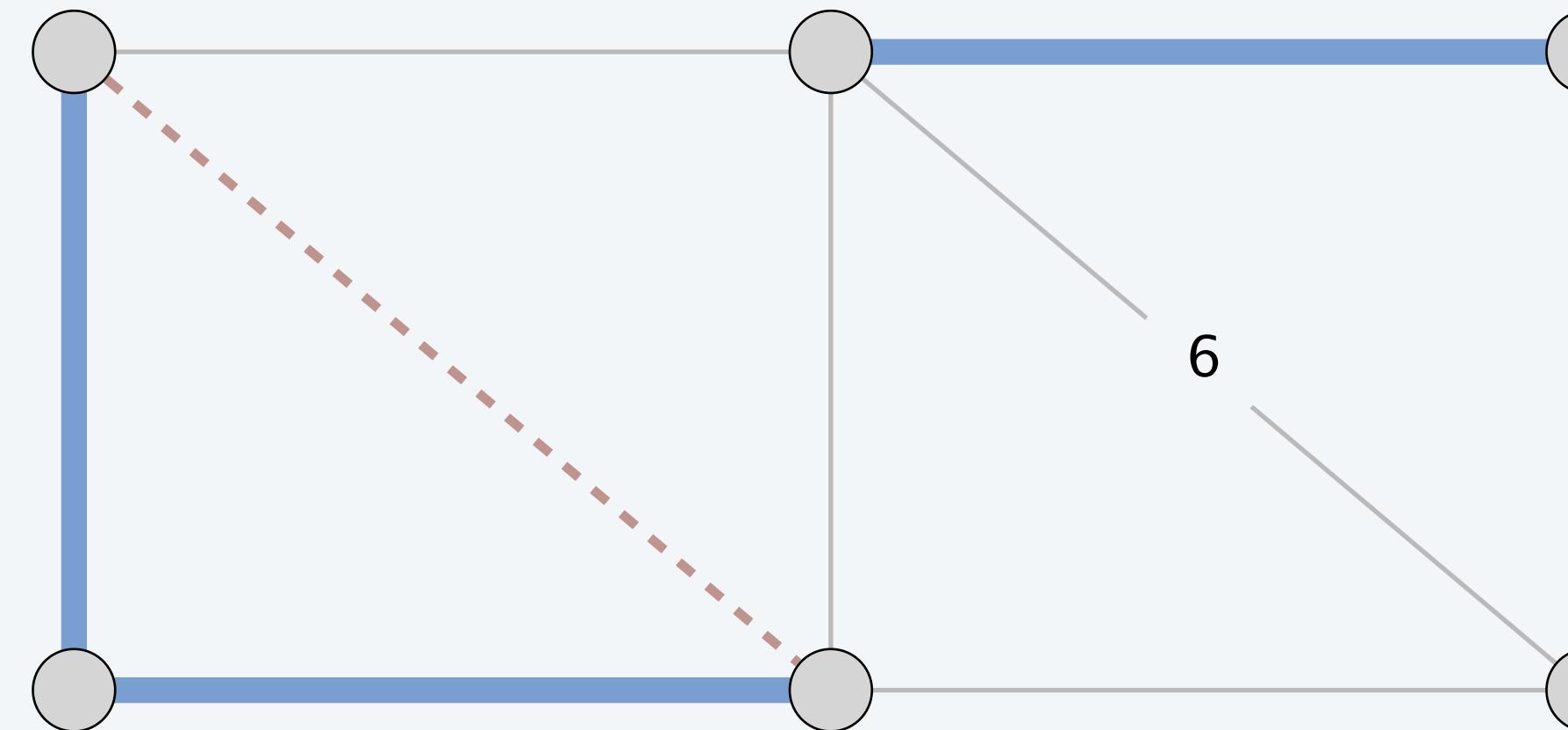
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



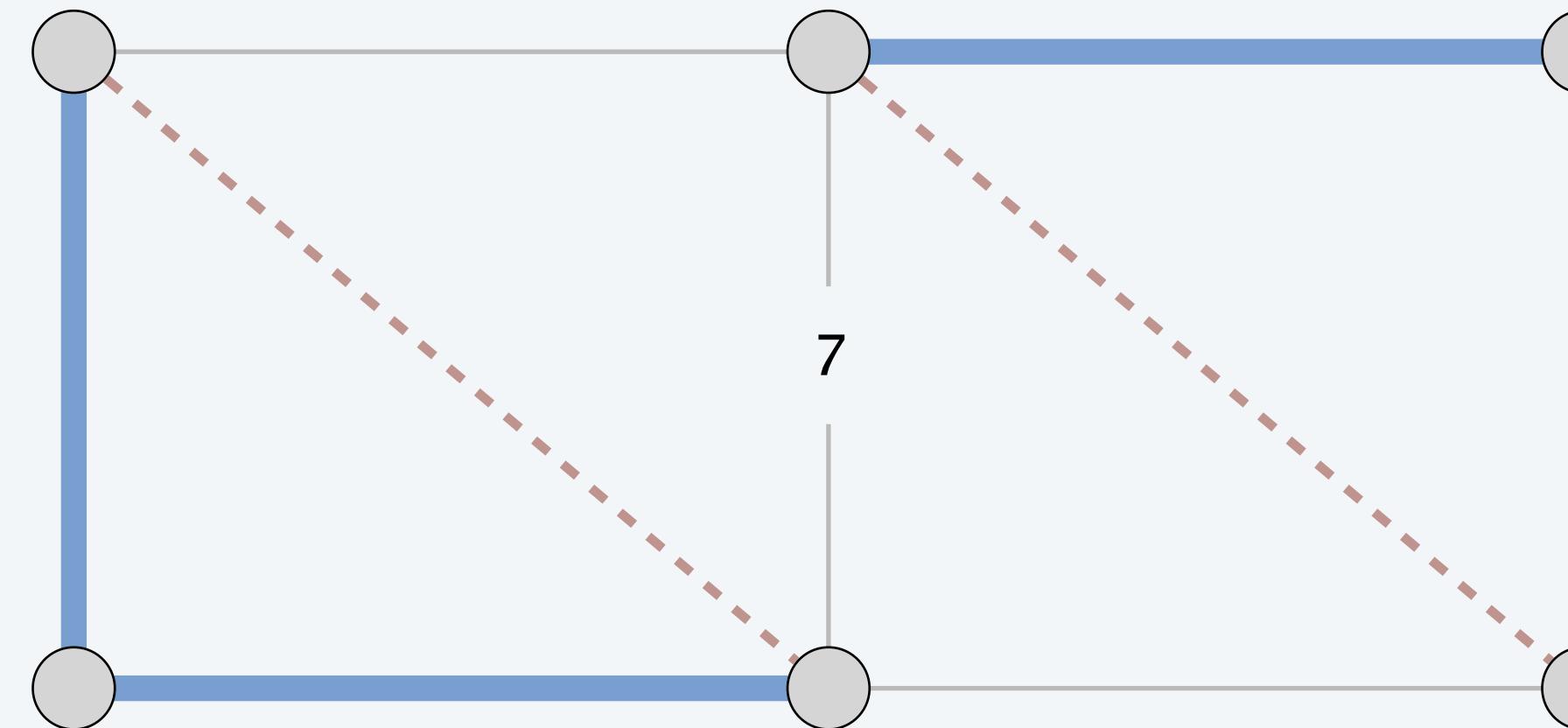
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



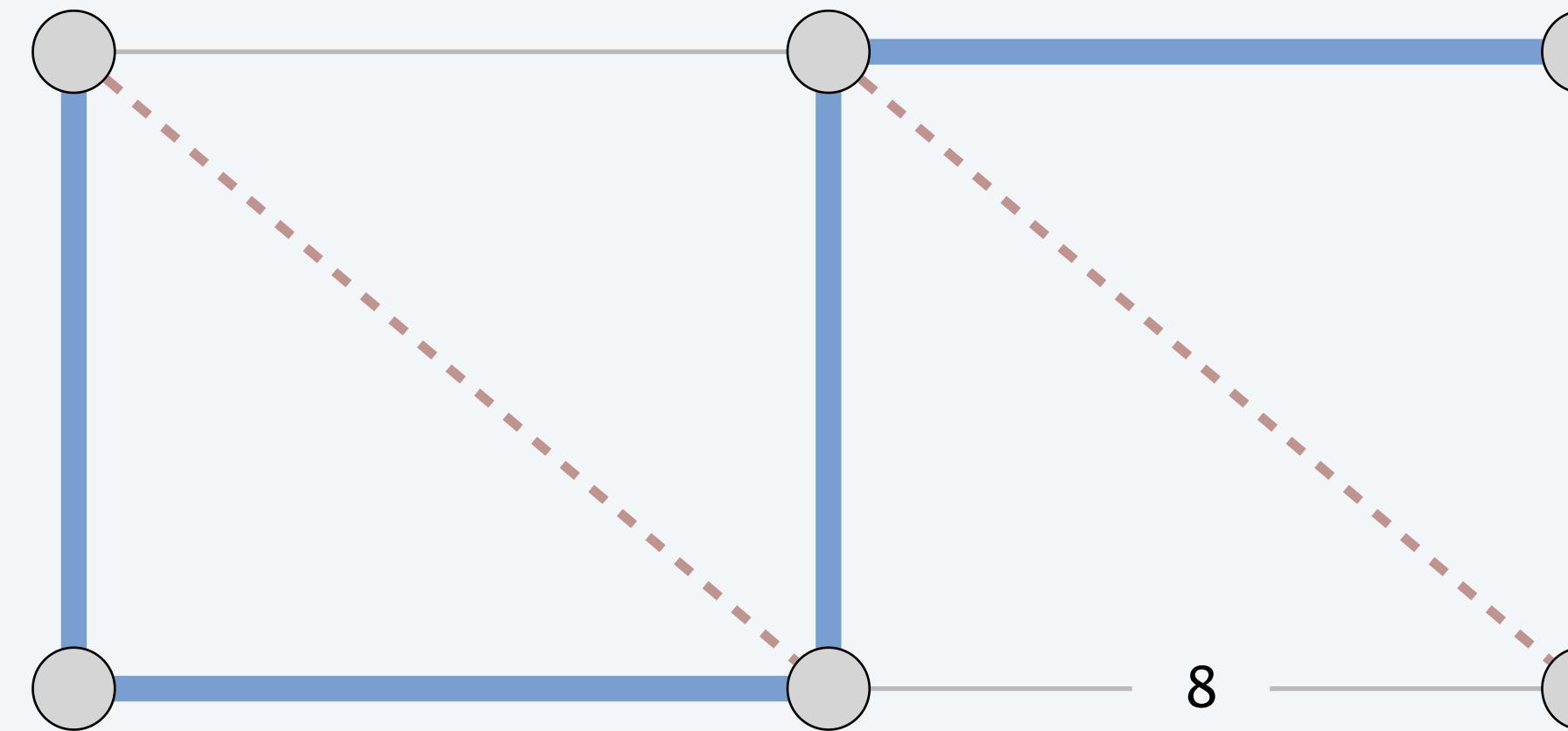
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



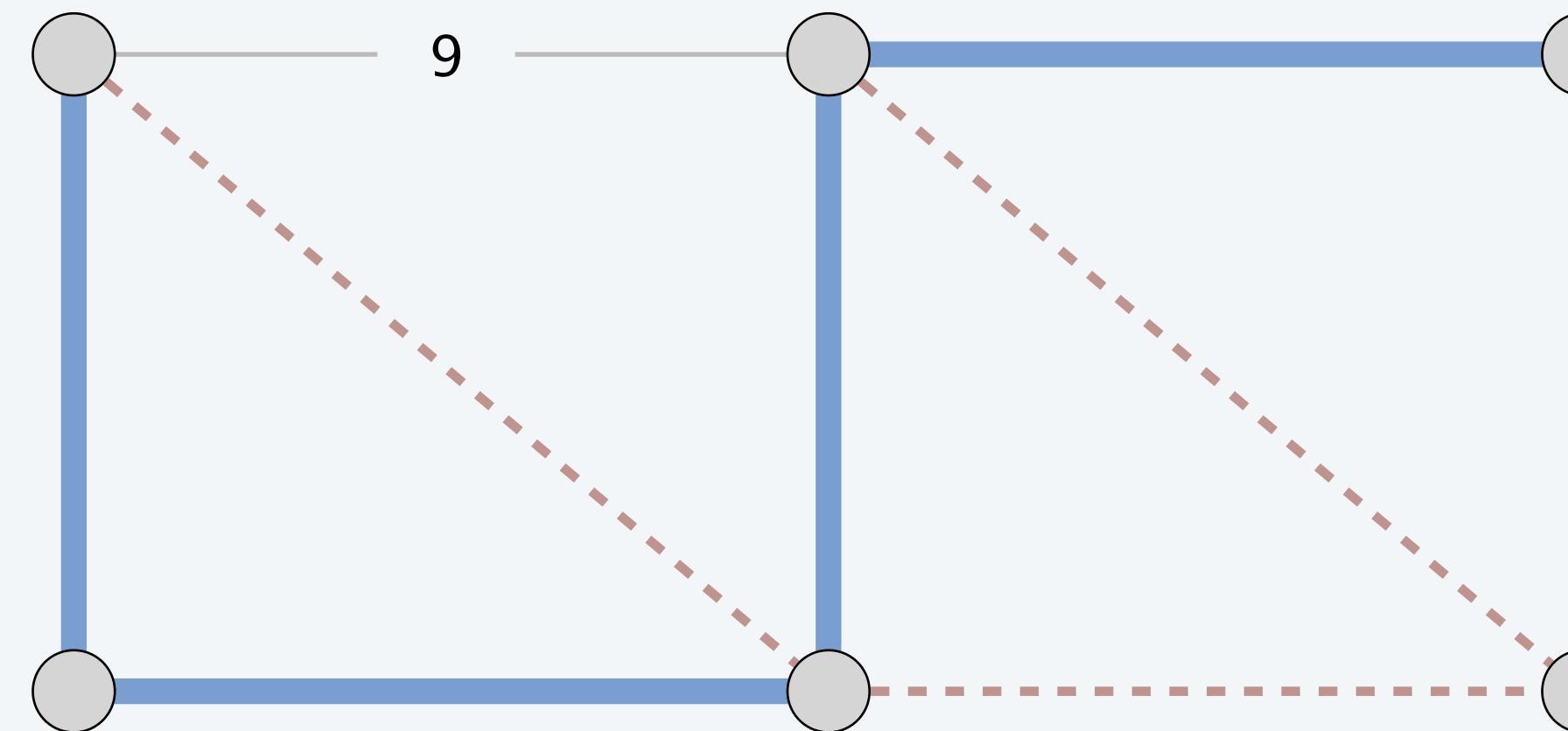
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



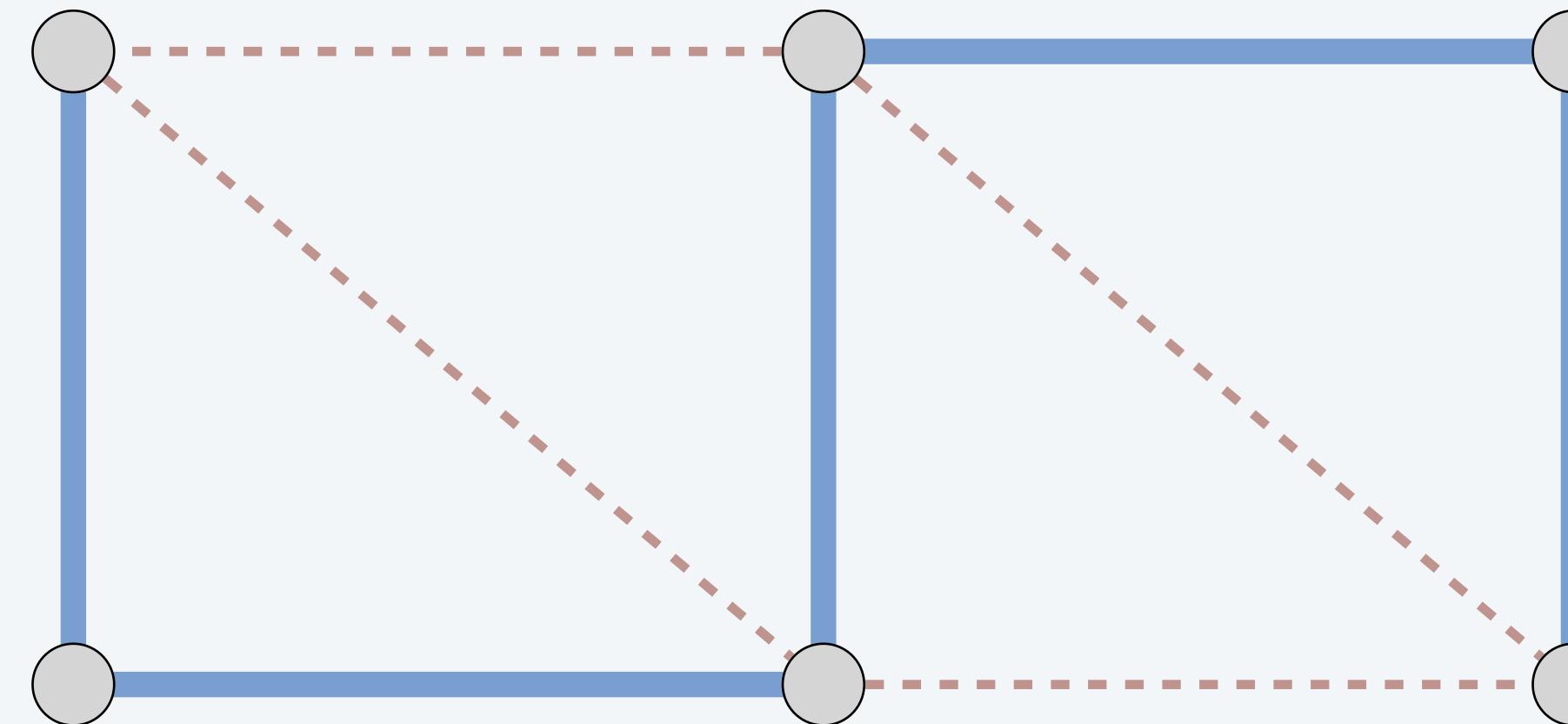
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



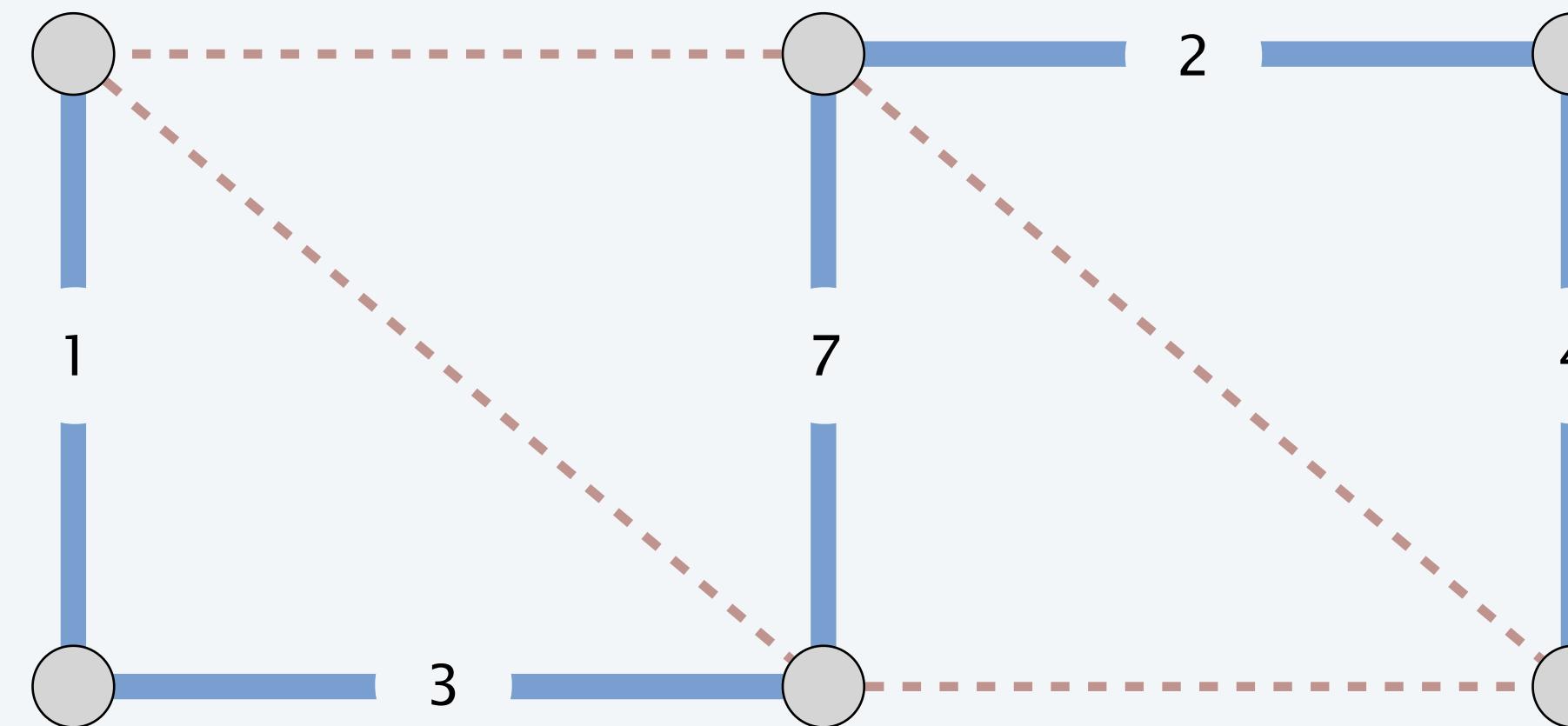
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



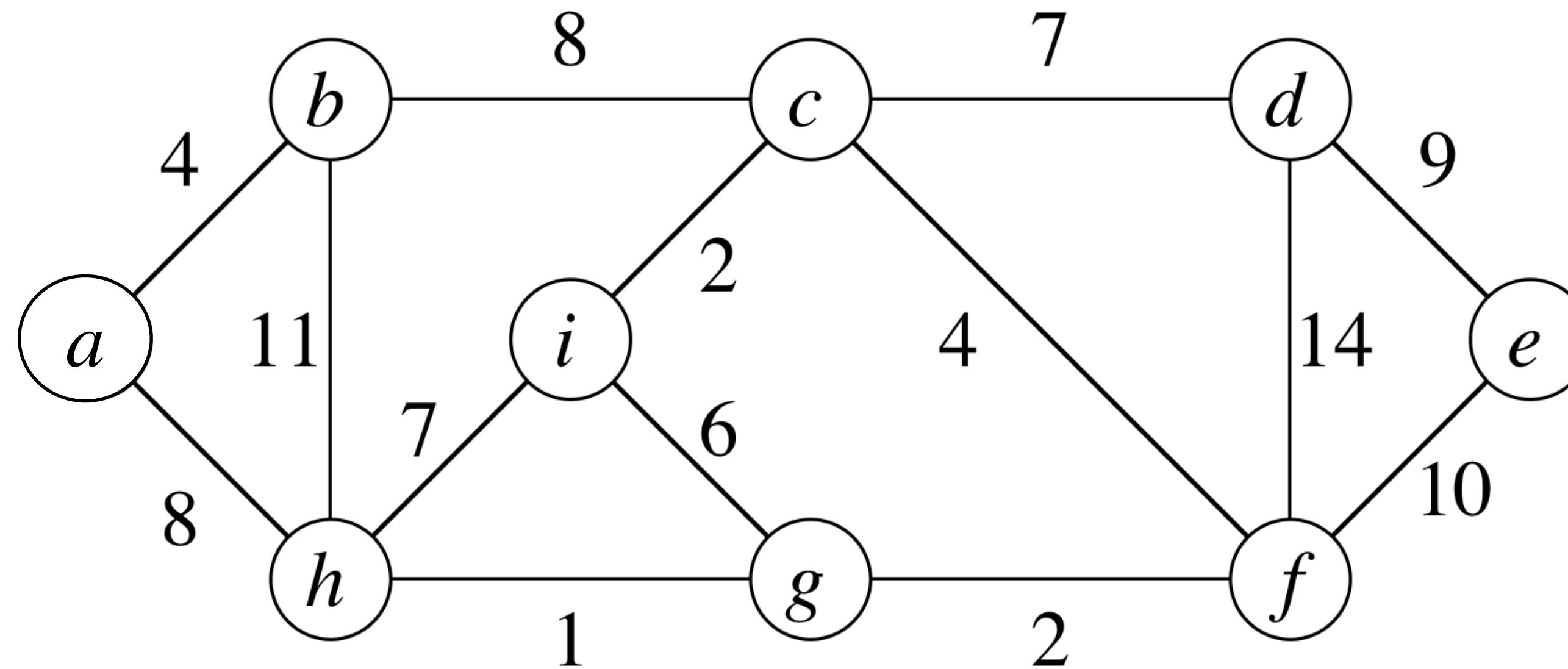
Kruskal's MST algoritme – eksempel

- Initialisér $A = \emptyset$
- Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds



Øvelse

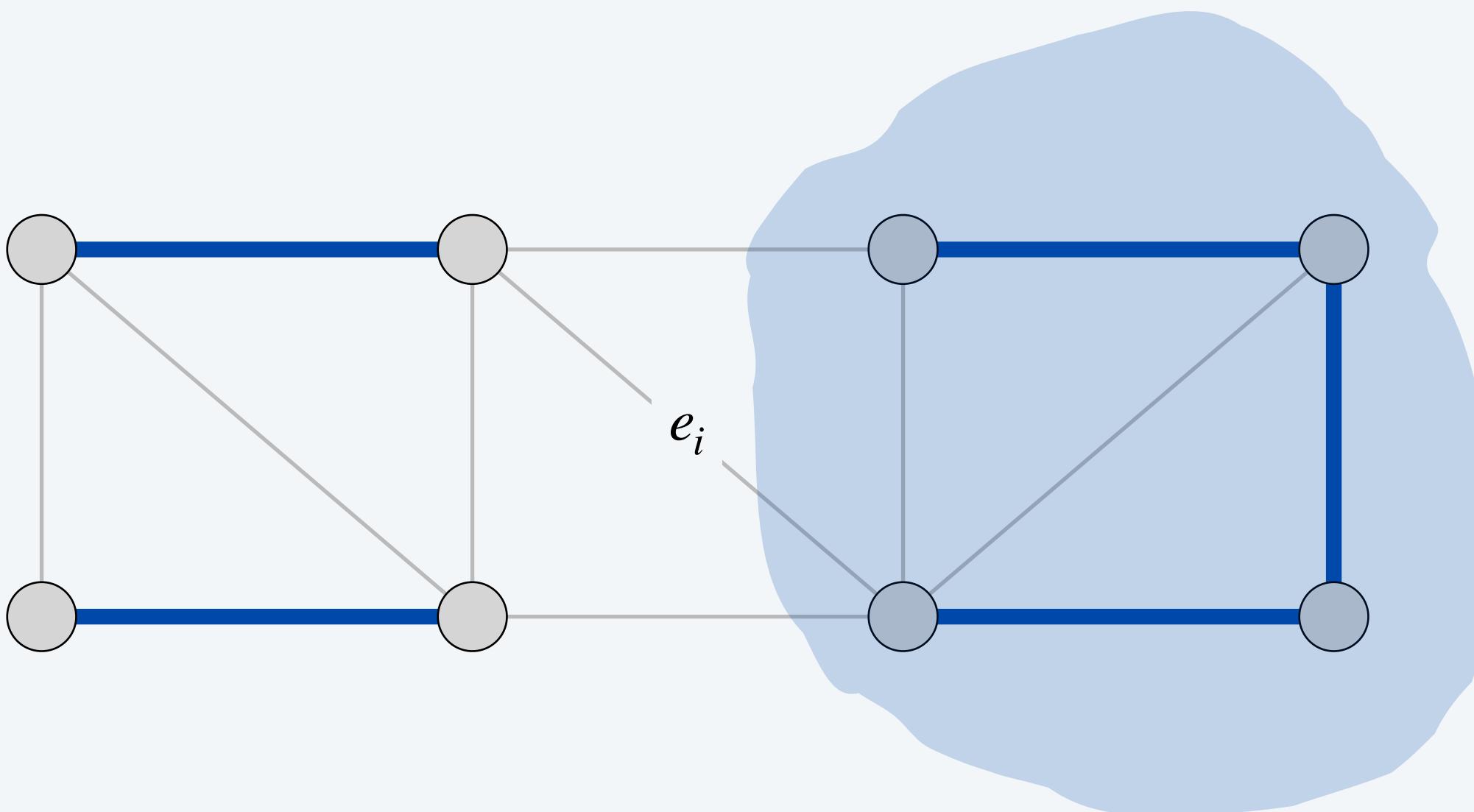
- Udfør Kruskal's algoritme på denne graf:



- For hver kant, du føjer til træet, find et snit der viser at kanten er sikker

Korrekthed af Kruskal's algoritme

- Initialisér $A = \emptyset$
 - Gennemløb kanterne e_1, e_2, \dots, e_m med ikke-faldende vægte, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$
 - Tilføj kant e_i til A med mindre det ville skabe en kreds
- ← Kanten er sikker, fordi den har mindst vægt i en snitmængde, der ikke indeholder kanter fra A : snitmængden mellem de to sammenhængskomponenter



Datastruktur til disjunkte mængder (ikke pensum)

Mål. Understøtte tre operationer på en samling af disjunkte mængder:

- $\text{MAKE-SET}(x)$: skab en ny mængde, der kun indeholder elementet x .
- $\text{FIND-SET}(x)$: returner et “kanonisk” element i mængden, der indeholder x .
- $\text{UNION}(x, y)$: udskift mængderne der indeholder x og y med deres foreningsmængde.

Parametre:

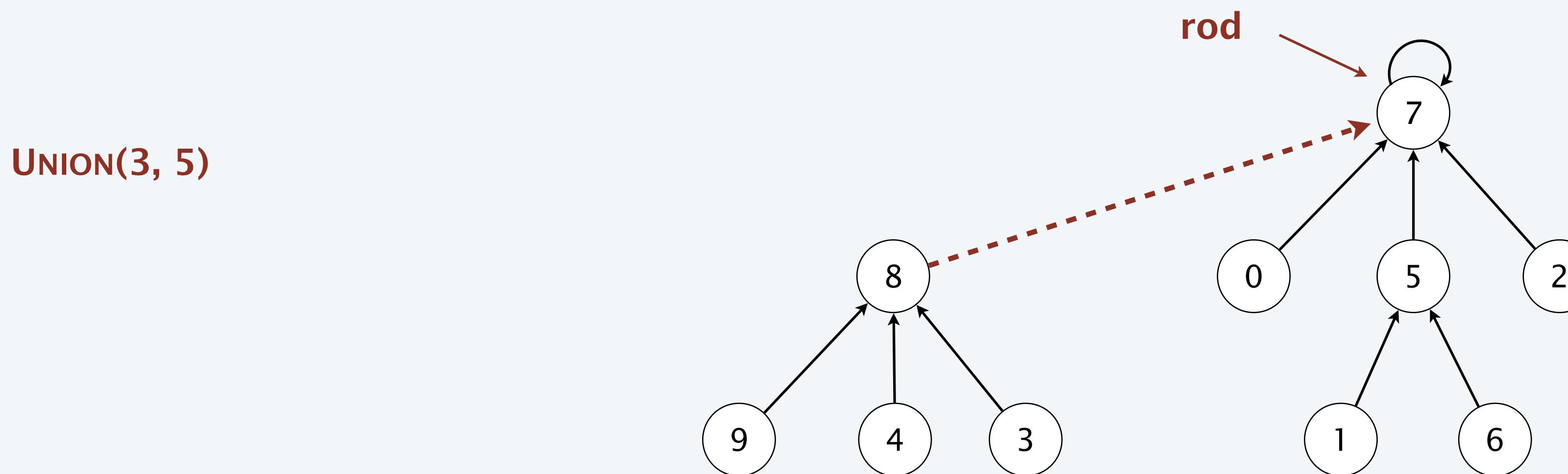
- m = antal kald til MAKE-SET , FIND-SET , og UNION .
- n = antal elementer = antal kald til MAKE-SET .

Trærepræsentation

Forældre-pointer repræsentation.

Repræsenter hver mængde som et **træ** af elementer (træerne danner en **skov**).

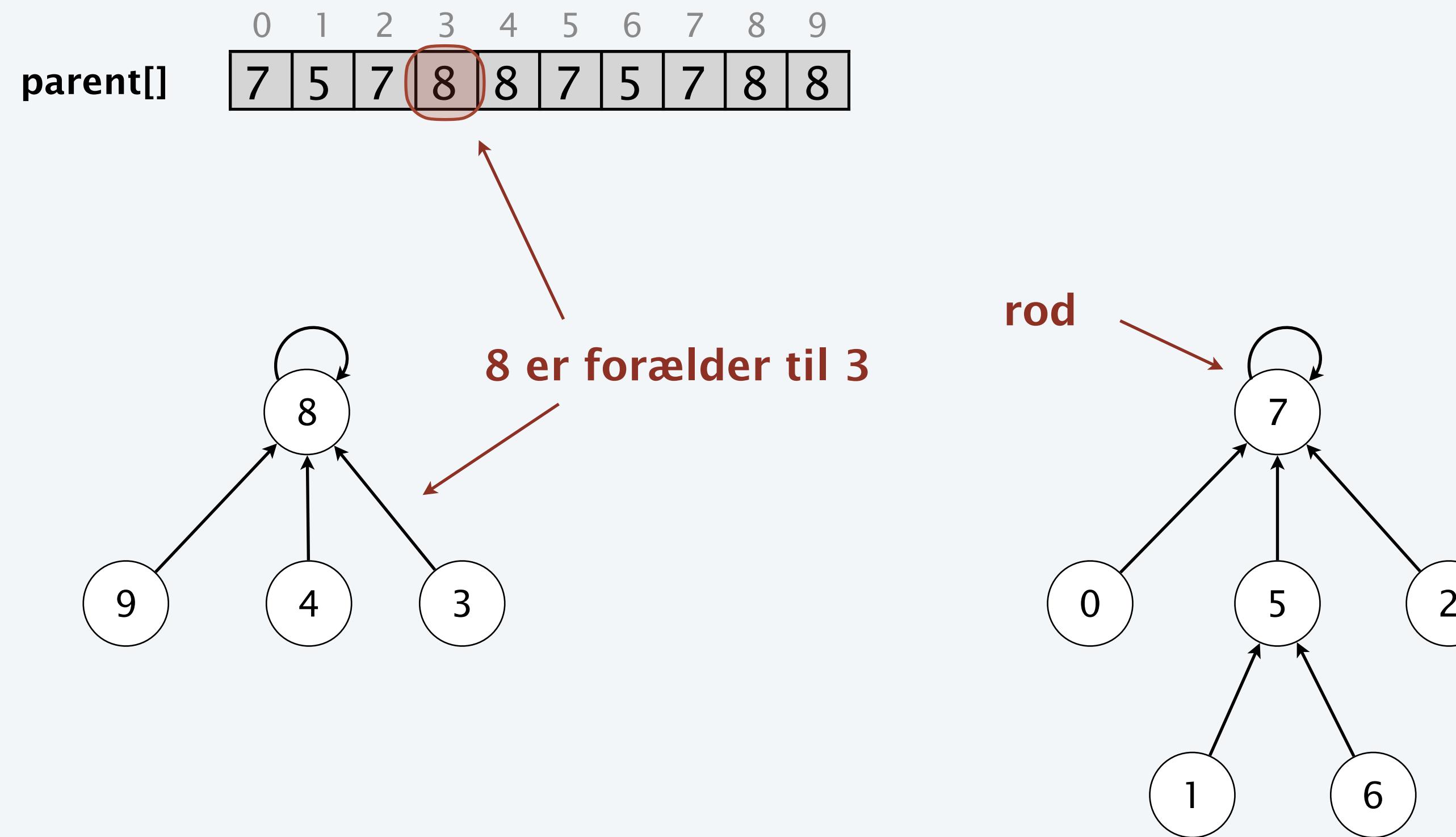
- Hvert element har en eksplisit pointer til sin forælder i træet.
- Roden er mængdens kanoniske element (og har pointer til sig selv).
- FIND-SET(x): find rodens af træet der indeholder x .
- UNION(x, y): lav forening af træerne der indeholder x og y (ved at lade den ene rod pege på den anden rod).



Implementation af trærepræsentation

Tabelrepræsentation. Representér hver mængde som et træ af elementer.

- Allokér en tabel $parent[]$ af længde n . ← Er nødt til at kende antal elementer n i forvejen
- $parent[i] = j$ betyder at j er forælder til element i .

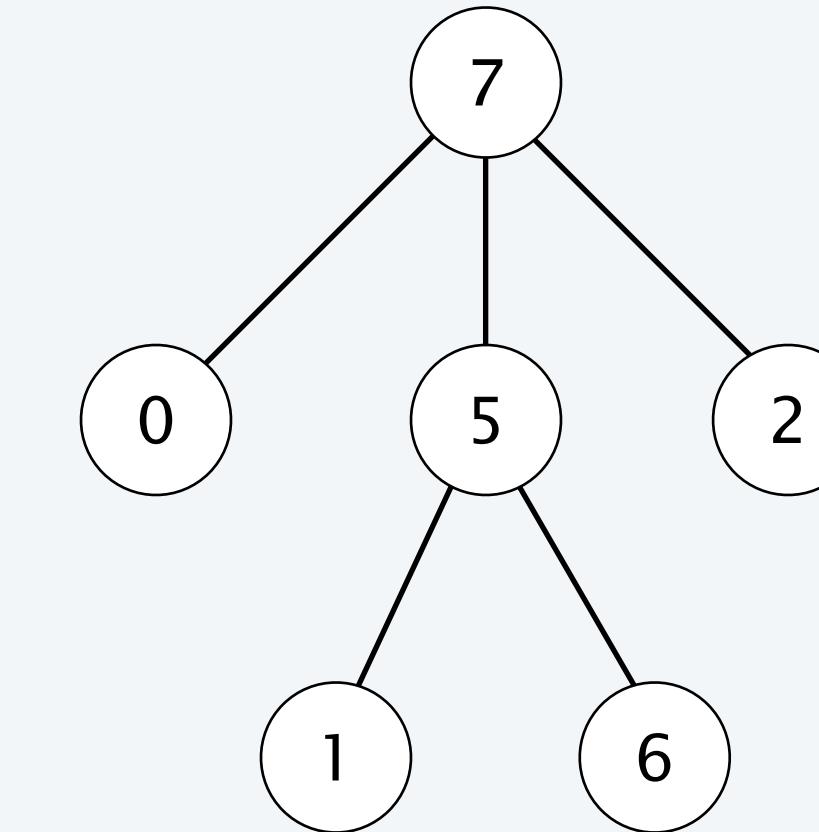
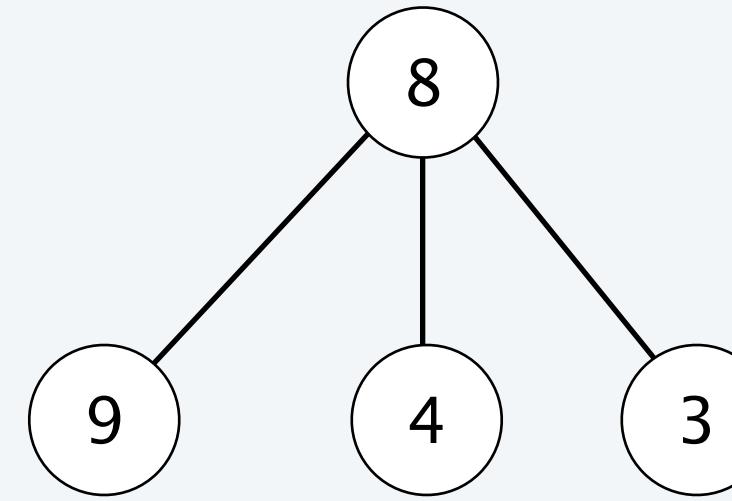


NB! I de følgende slides udelader vi pile (altid opad) og selvreferencer til roden

Naiv hægtning

Naiv hægtning. Lad roden fra det første træ pege på rodens for det andet træ.

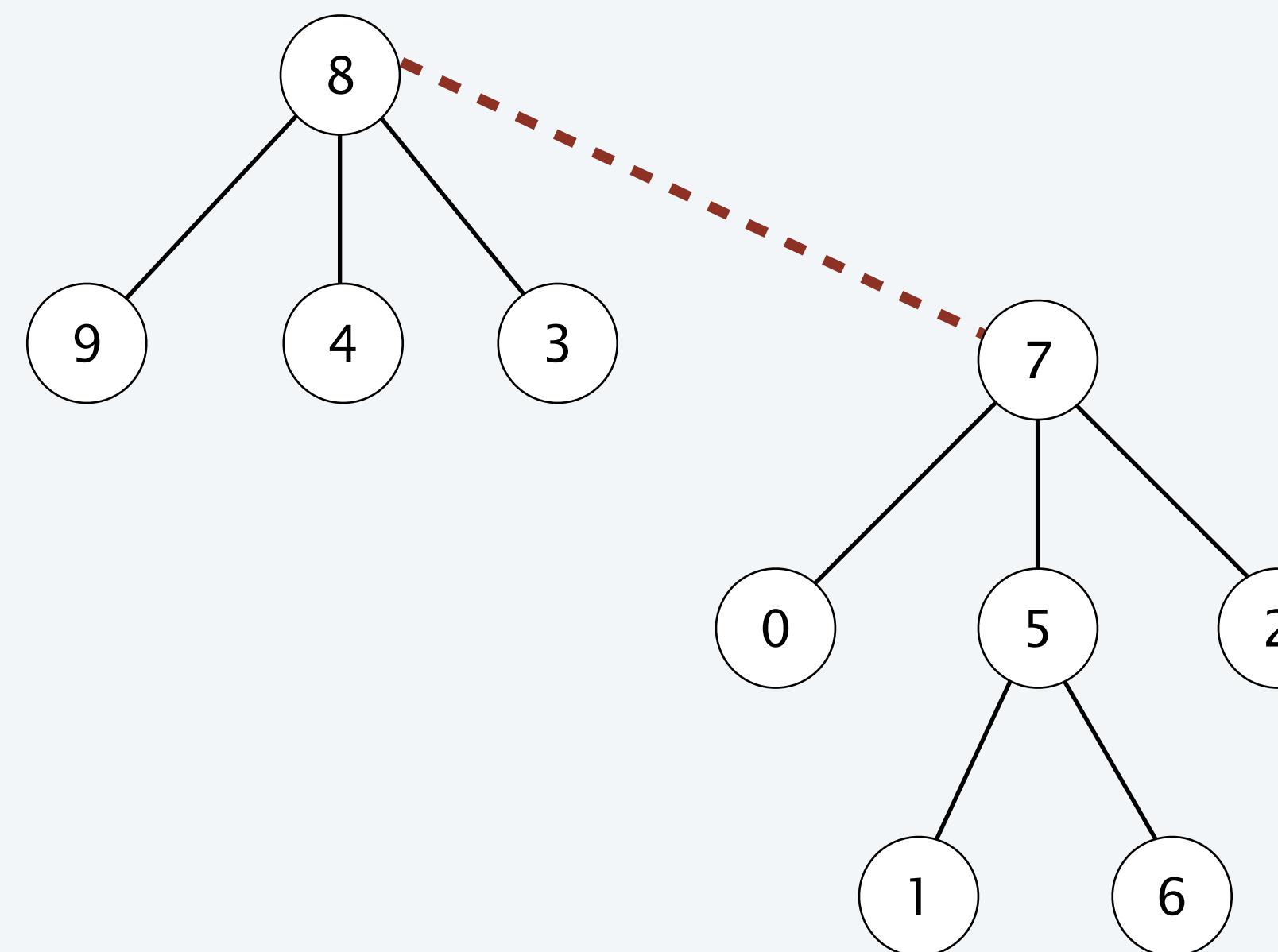
UNION(5, 3)



Naiv hægtning

Naiv hægtning. Lad roden fra det første træ pege på rodens for det andet træ.

UNION(5, 3)



Naiv hægtning

Naiv hægtning. Lad roden fra det første træ pege på roden for det andet træ.

MAKE-SET(x)

$parent[x] \leftarrow x.$

UNION(x, y)

$r \leftarrow \text{FIND-SET}(x).$

$s \leftarrow \text{FIND-SET}(y).$

$parent[r] \leftarrow s.$

FIND-SET(x)

WHILE ($x \neq parent[x]$)

$x \leftarrow parent[x].$

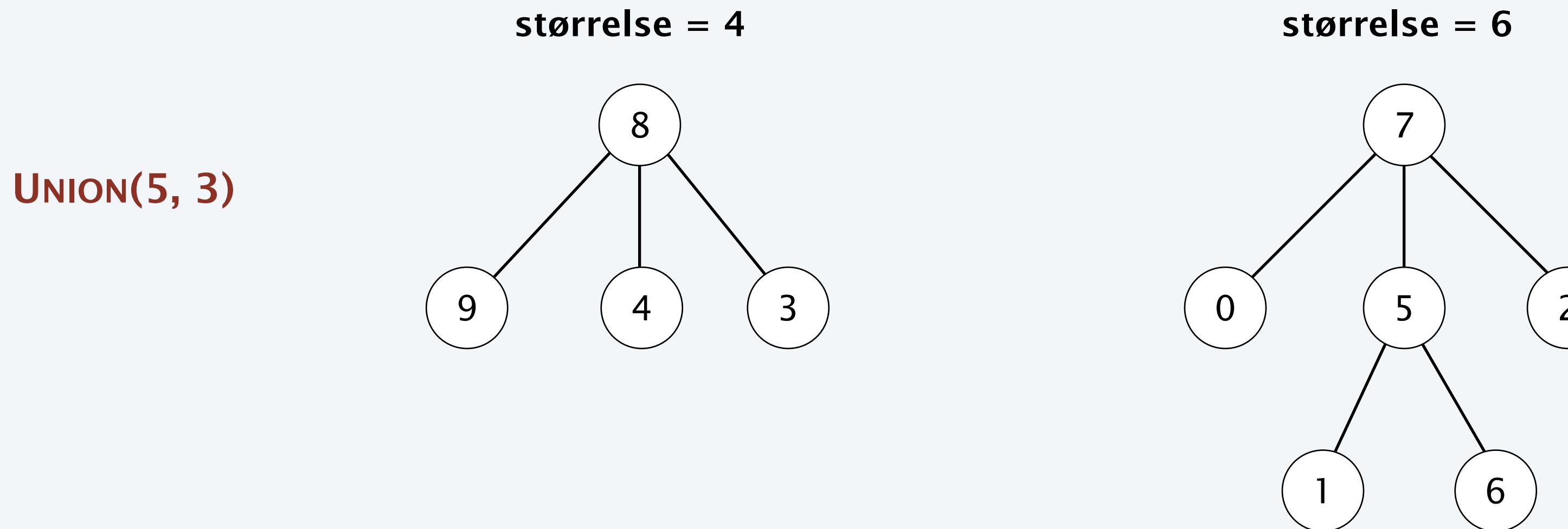
RETURN $x.$

Hægt-efter-størrelse (CLRS: “Weighted union heuristic”)

Hægt-efter-størrelse:

Vedligehold en **størrelse** (antal elementer i mængden) for hver rodknude.

Hægt roden af det mindre træ på det større træ
(hvis de er lige store så vælg den nye rod arbitrært).

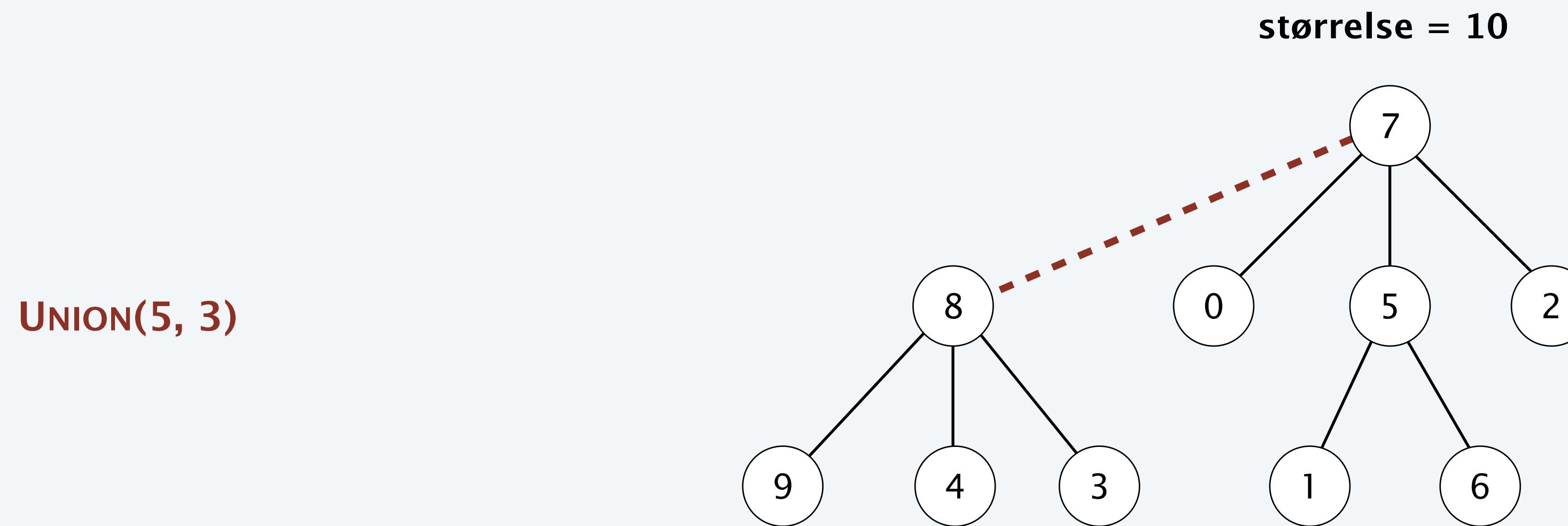


Hægt-efter-størrelse

Hægt-efter-størrelse:

Vedligehold en **størrelse** (antal elementer i mængden) for hver rodknude.

Hægt roden af det mindre træ på det større træ
(hvis de er lige store så vælg den nye rod arbitrært).



Hægt-efter-størrelse

MAKE-SET(x)

$parent[x] \leftarrow x.$

$size[x] \leftarrow 1.$

FIND-SET(x)

WHILE ($x \neq parent[x]$)

$x \leftarrow parent[x].$

RETURN $x.$

UNION(x, y)

$r \leftarrow \text{FIND-SET}(x).$

$s \leftarrow \text{FIND-SET}(y).$

IF ($r = s$) **RETURN**.

ELSE IF ($size[r] > size[s]$)

$parent[s] \leftarrow r.$

$size[r] \leftarrow size[r] + size[s].$

ELSE

$parent[r] \leftarrow s.$

$size[s] \leftarrow size[r] + size[s].$

← hægt-efter-størrelse

FIND-SET(x) kræver tid proportionalt med dybden af x i træet

$height(r) =$ største dybde i træet med rodknude r

Hægt-efter-størrelse: Analyse

Teorem. Ved brug af hægt-efter-størrelse bruger enhver UNION eller FIND-SET operation tid $O(\log n)$ i værste fald, hvor n er antal elementer.

Bevisskitse

- Køretiden af hver operation er proportional med træets højde.
- Man kan vise at højden er $\leq \lfloor \lg n \rfloor$. ■

$$\lfloor \lg n \rfloor = \log_2 n$$


Implementation af Kruskal's algoritme

- Brug datastruktur til disjunkte mængder til at bestemme om en kant (u, v) forbinder to forskellige sammenhængskomponenter eller ej
- Tidsanalyse:
 $n \times$ Make-Set
 $n - 1 \times$ Union
 $2m \times$ Find-Set
- Med link-by-rank:
Tid $O(n + m \log n)$
- Med stiforkortning:
Tid $O(n + m\alpha(n))$

```
KRUSKAL( $G, w$ )
     $A = \emptyset$ 
    for each vertex  $v \in G.V$ 
        MAKE-SET( $v$ )
    sort the edges of  $G.E$  into nondecreasing order by weight  $w$ 
    for each  $(u, v)$  taken from the sorted list
        if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ )
             $A = A \cup \{(u, v)\}$ 
            UNION( $u, v$ )
    return  $A$ 
```

Findes der en MST algoritme som bruger tid $O(n + m)$ i værste fald?

år	worst case tid	opdaget af
1975	$O(m \log \log n)$	Yao
1976	$O(m \log \log n)$	Cheriton-Tarjan
1984	$O(m \log^* n), O(m + n \log n)$	Fredman-Tarjan
1986	$O(m \log (\log^* n))$	Gabow-Galil-Spencer-Tarjan
1997	$O(m \alpha(n) \log \alpha(n))$	Chazelle
2000	$O(m \alpha(n))$	Chazelle
2002	<i>asymptotisk optimal</i>	Pettie-Ramachandran
20xx	$O(m) ?$???

deterministiske sammenligningsbaserede MST algoritmer

Opsummering

- Et minimalt spændetræ kan findes ved hjælp af en grådig algoritme, der bygger et træ ved hele tiden at tilføje en *sikker* kant
- Vi så to måder at finde sikre kanter på:
 - Jarník-Prim: Find den letteste kant, der har én ende i det nuværende træ
 - Kruskal: Find den letteste kant og check at den ikke skaber en kreds
- Flere detaljer i CLRS 21