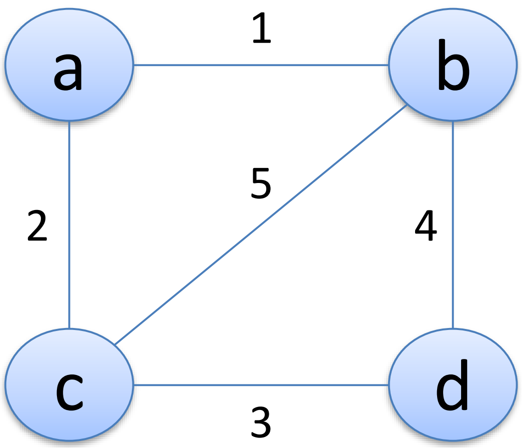
Minimum cost spanning tree (MCST)

- What is a minimum cost spanning tree?
  - Tree
    - No cycles; equivalently, for each pair of nodes u and v, there is only one path from u to v
  - Spanning
    - Contains every node in the graph
  - Minimum cost
    - Smallest possible total weight of any spanning tree

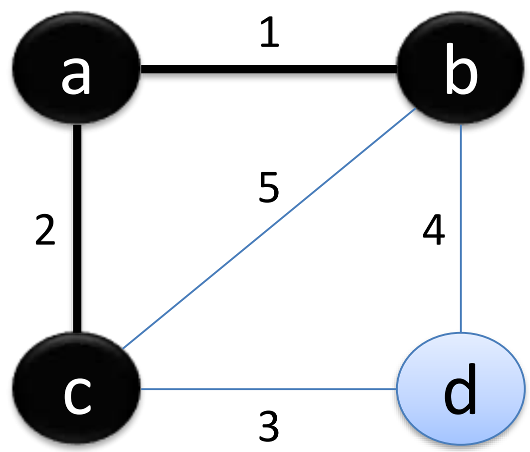
Minimum cost spanning tree (MCST)

- Let’s think about simple MCSTs on this graph:



Minimum cost spanning tree (MCST)

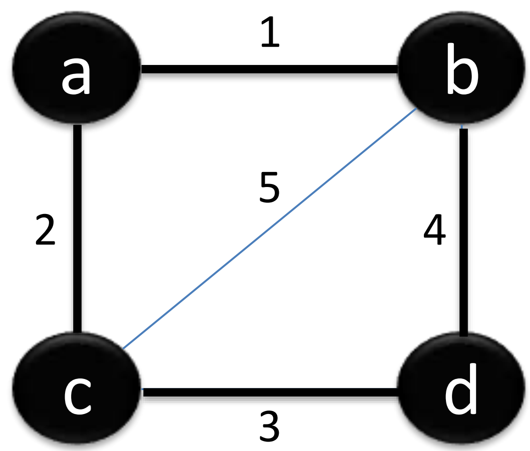
- Black edges and nodes are in T
- Is T a minimum cost spanning tree?



- Not spanning; d is not in T.

Minimum cost spanning tree (MCST)

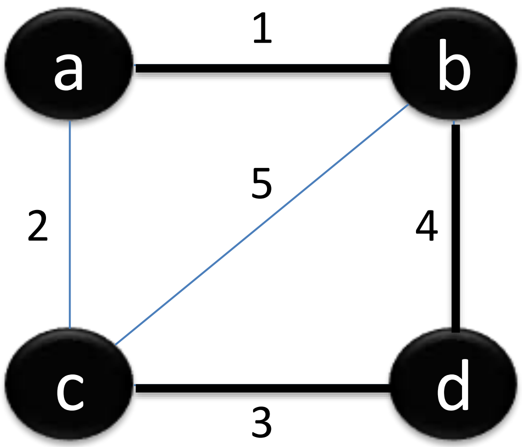
- Black edges and nodes are in T
- Is T a minimum cost spanning tree?



- Not a tree; has a cycle.

Minimum cost spanning tree (MCST)

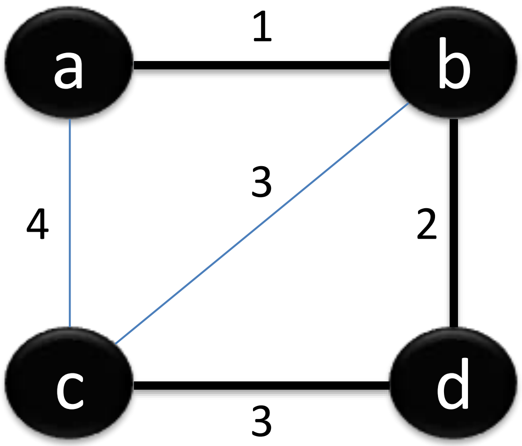
- Black edges and nodes are in T
- Is T a minimum cost spanning tree?

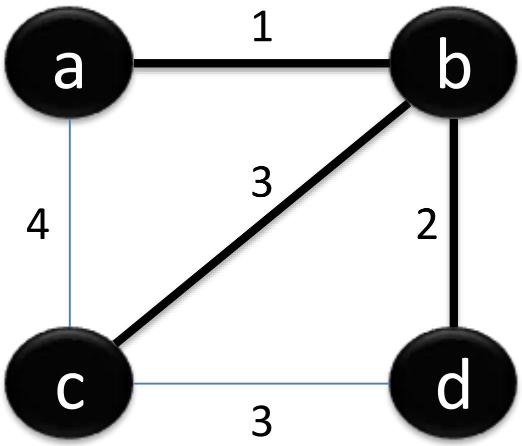


- Not minimum cost; can swap edges 4 and 2.

Minimum cost spanning tree (MCST)

- Which edges form a MCST?



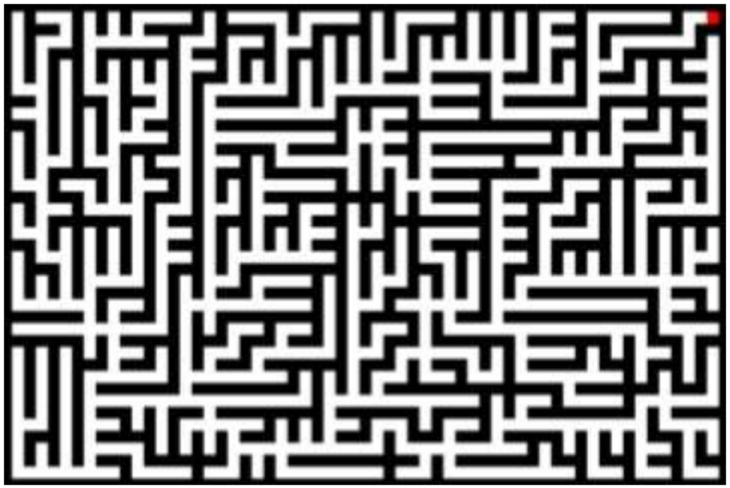


An application of MCSTs

- Electronic circuit designs (from Cormen et al.)
  - Circuits often need to wire together the pins of several components to make them electrically equivalent.
  - To connect *n* pins, we can use *n* - 1 wires, each connecting two pins.
  - Want to use the minimum amount of wire.
  - Model problem with a graph where each pin is a node, and every possible wire between a pair of pins is an edge.

A few other applications of MCSTs

- Planning how to lay network cable to connect several locations to the internet
- Planning how to efficiently bounce data from router to router to reach its internet destination
- Creating a 2D maze (to print on cereal boxes, etc.)



Building a MCST

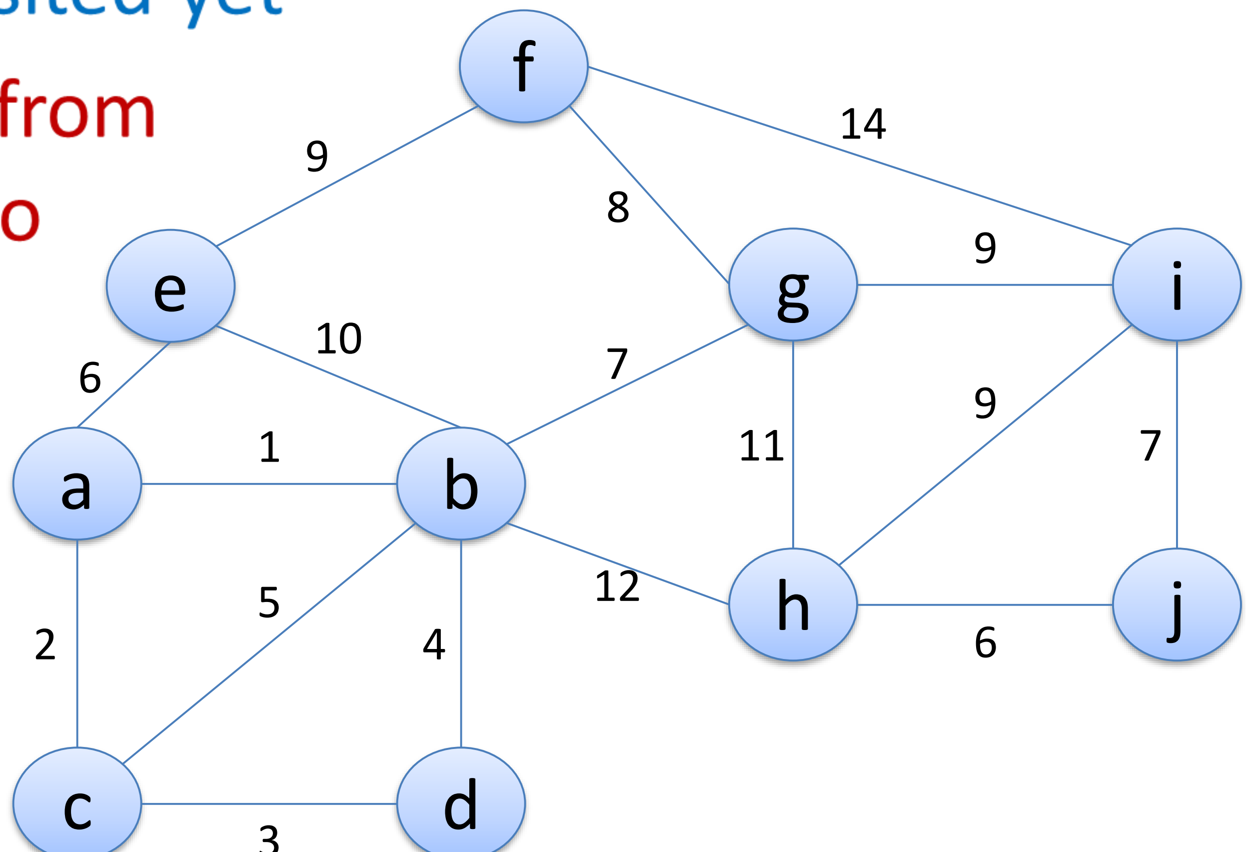
- Prim's algorithm takes a graph G = (V, E) and builds an MCST ***T***
- PrimMCST(V, E)
  - Pick an arbitrary node **r** from V
  - Add **r** to *T*
  - While *T* contains < |V| nodes
    - Find a **minimum weight edge (u, v)** where $\mathbf{u} \in \boldsymbol{T}$ and $\mathbf{v} \notin \boldsymbol{T}$
    - Add node v to *T*

In the book's terminology, we find a **light edge crossing the cut (T, V-T)**

The book proves that adding |V|-1 such edges will create a MCST

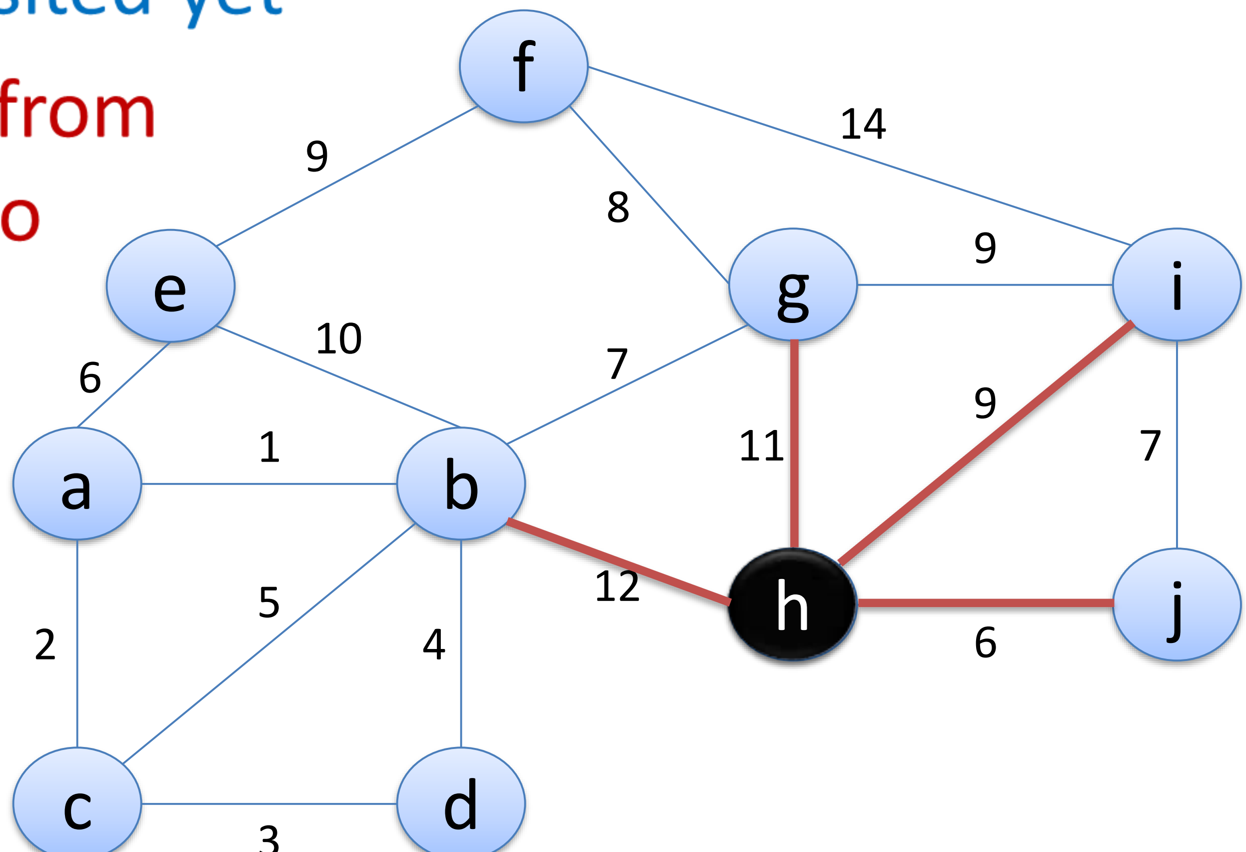
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



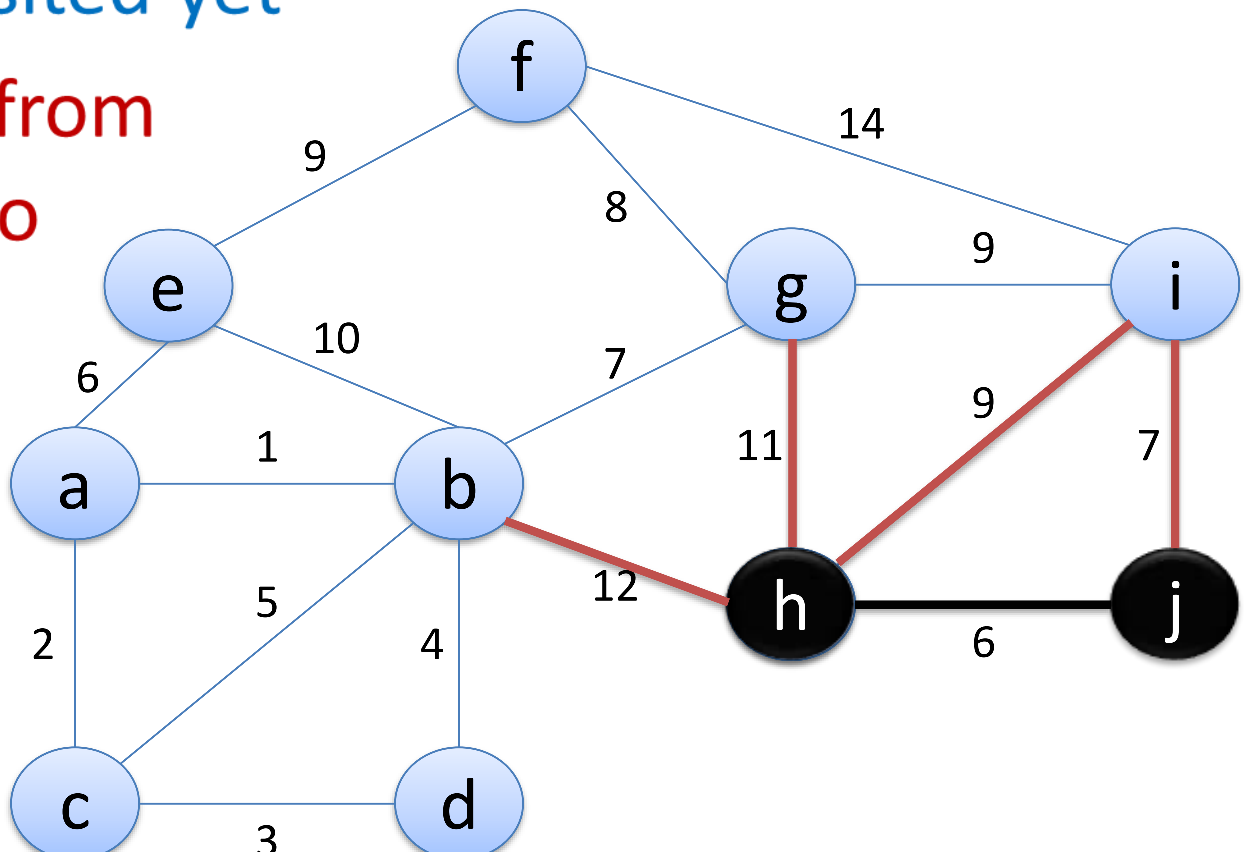
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



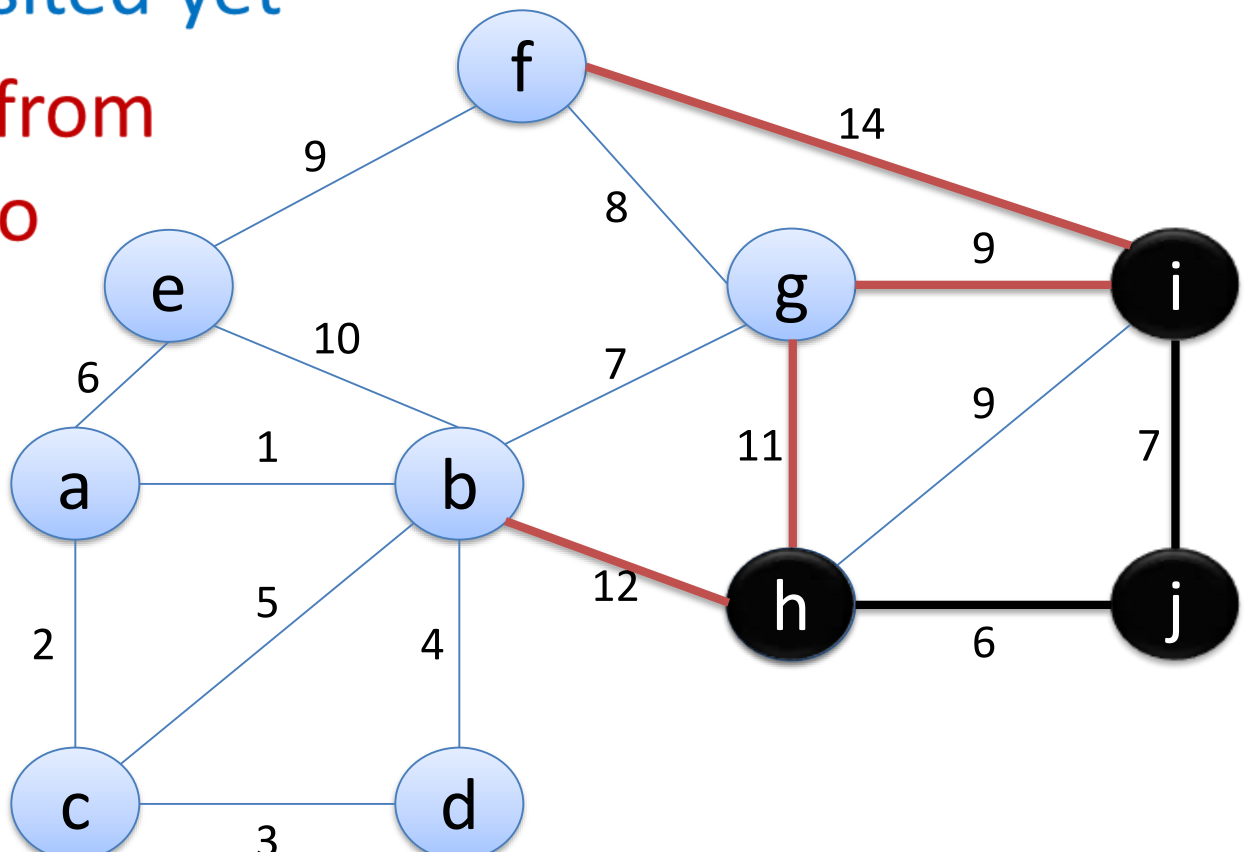
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



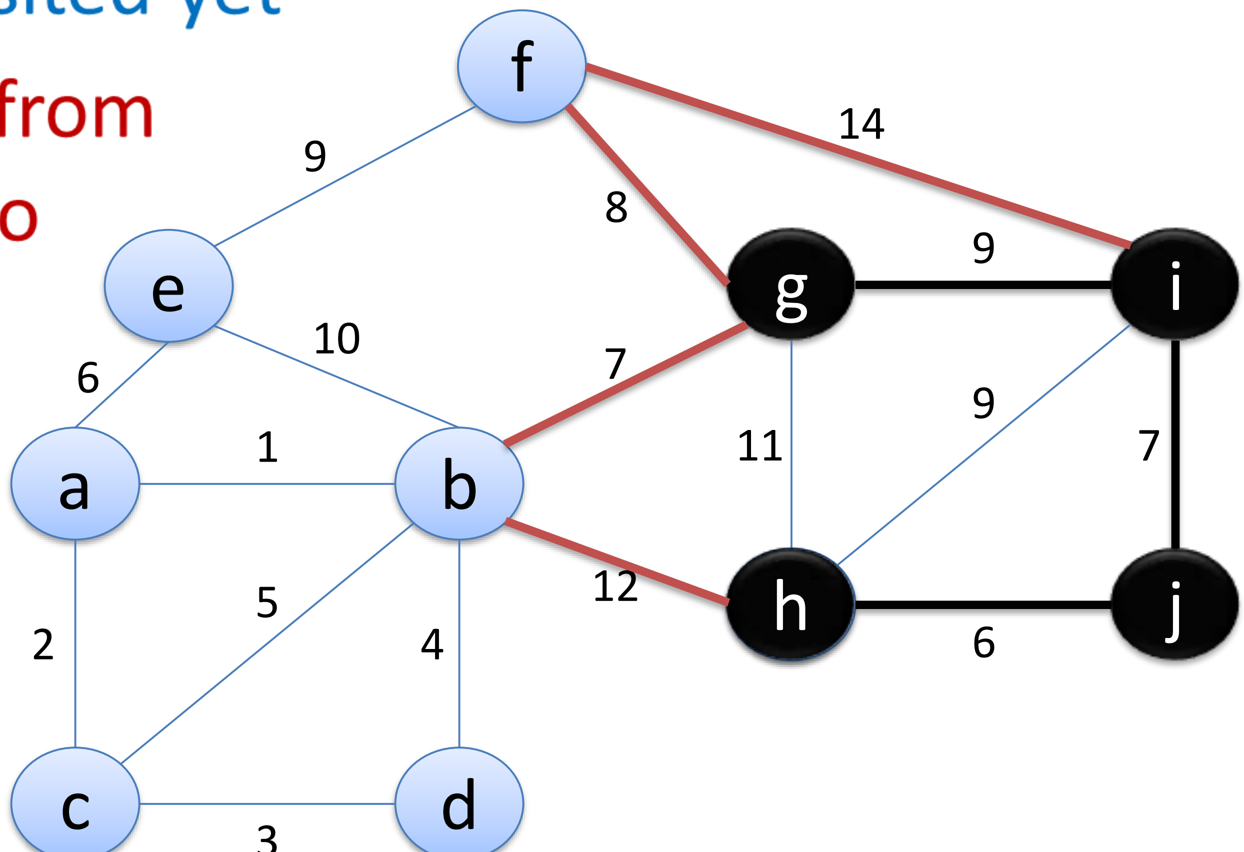
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



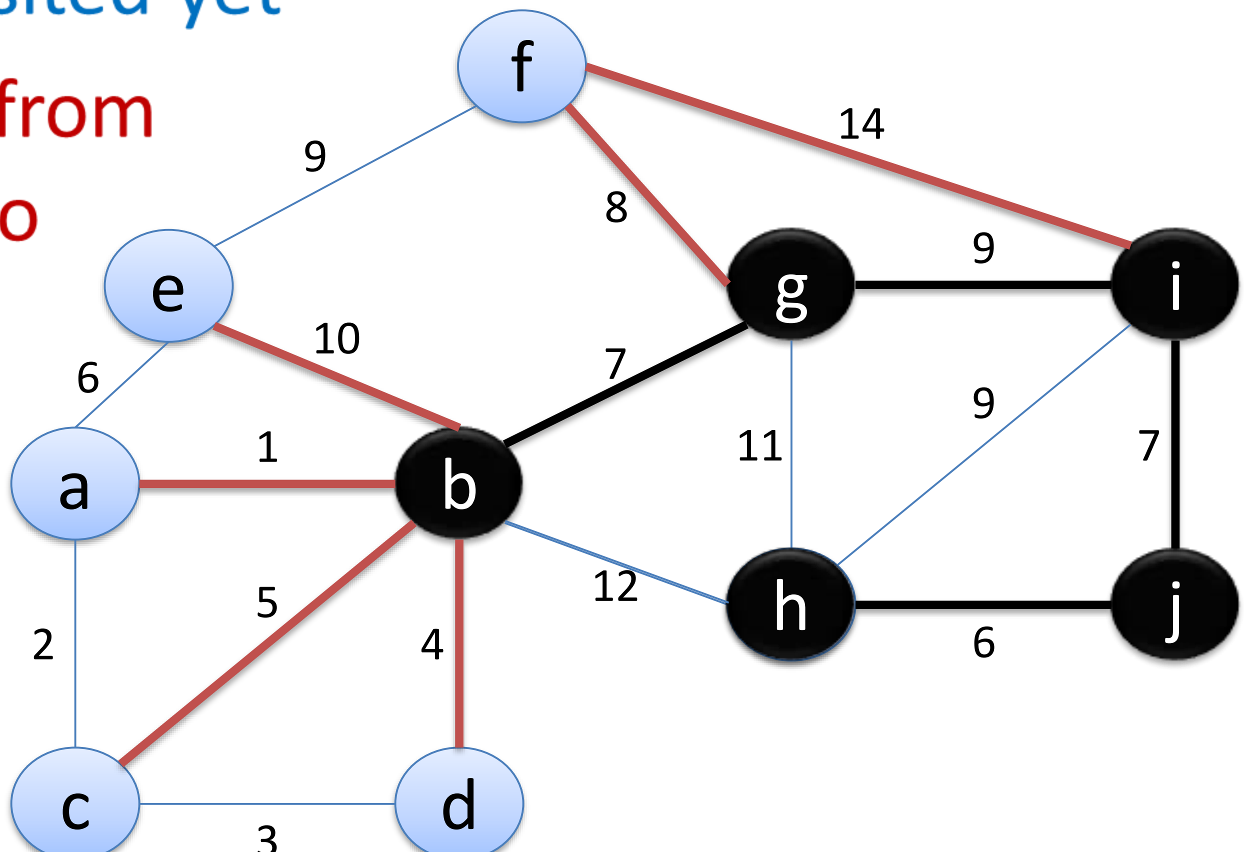
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



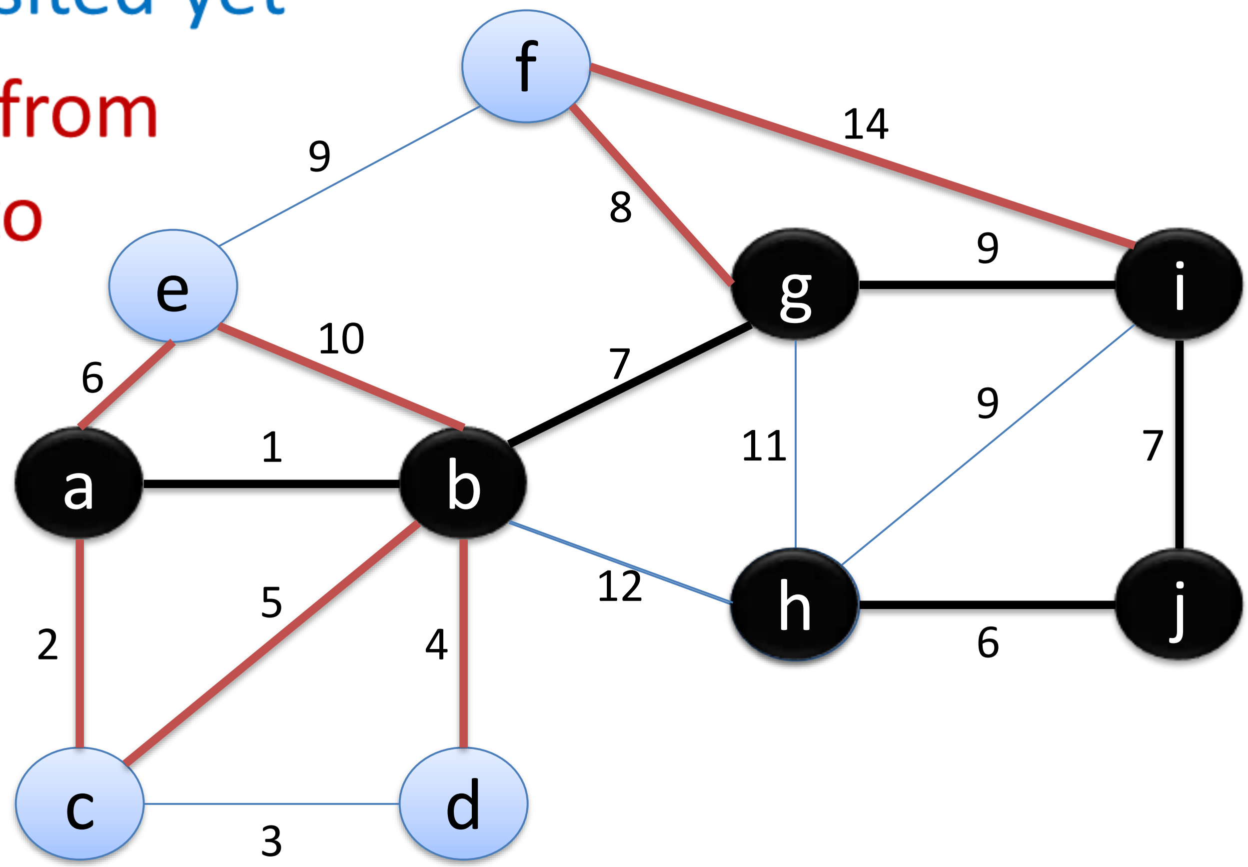
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



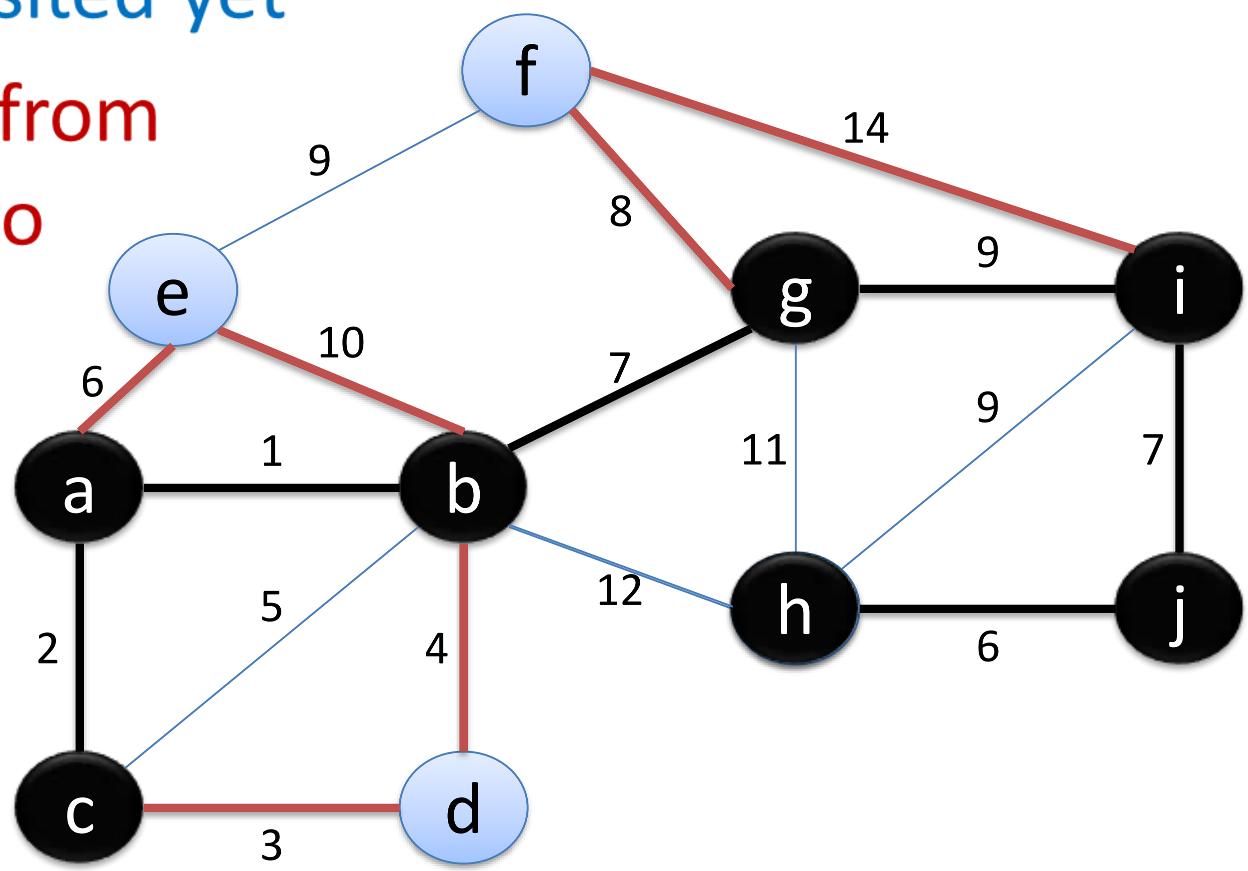
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



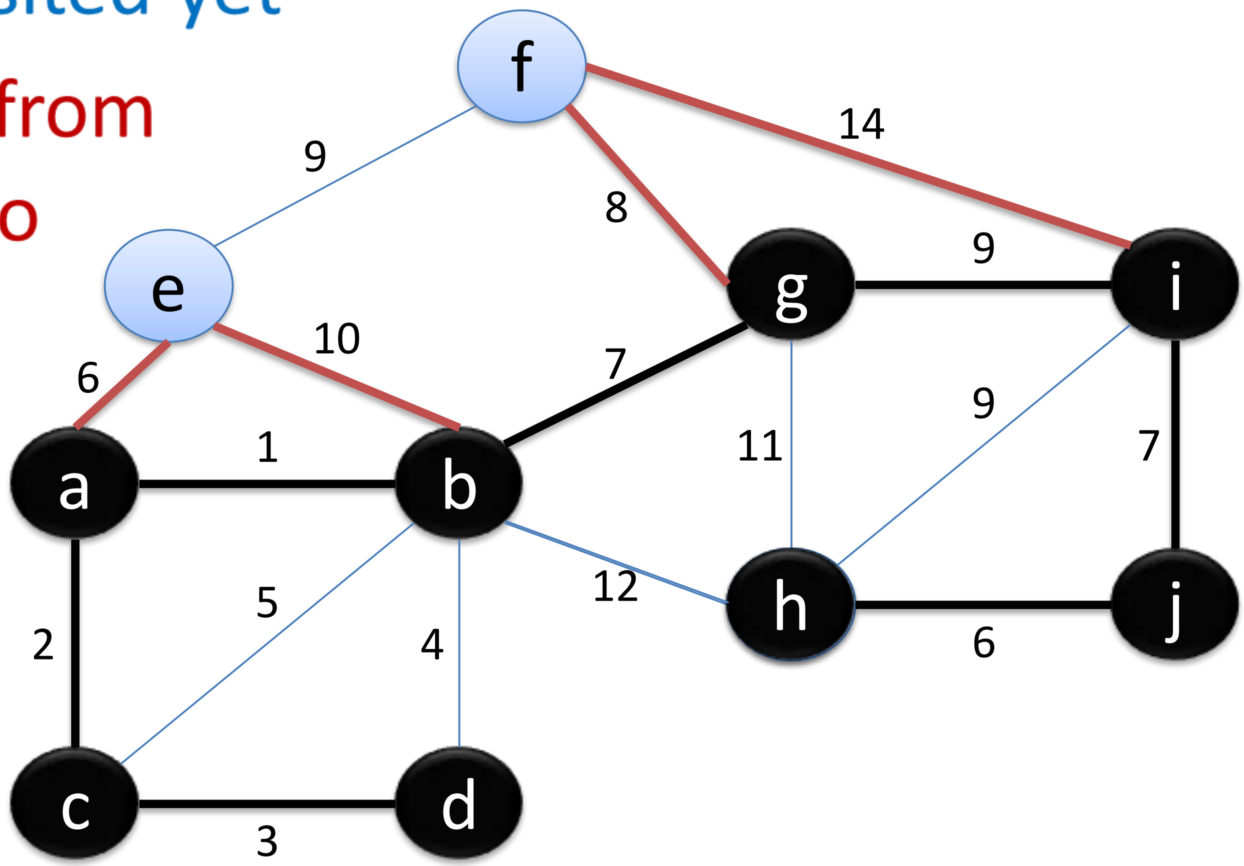
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



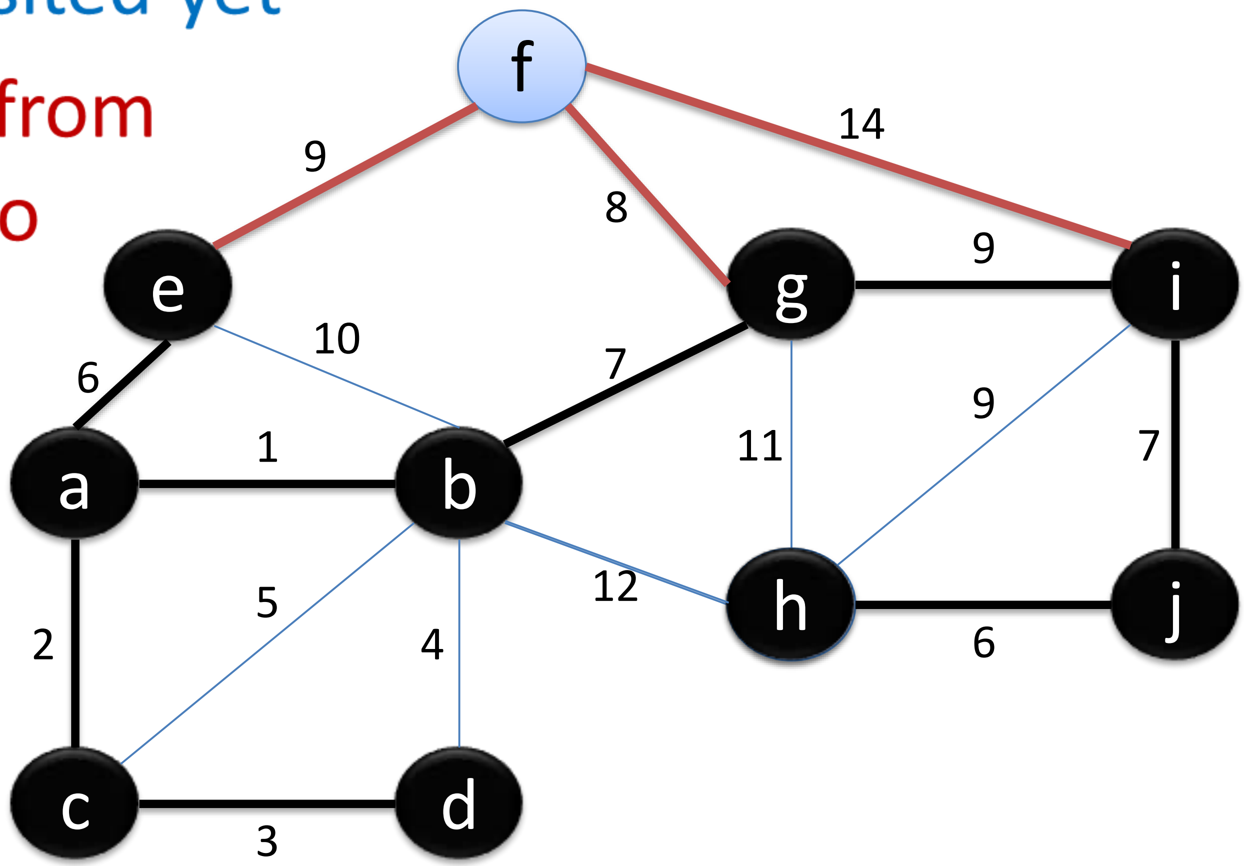
Running Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



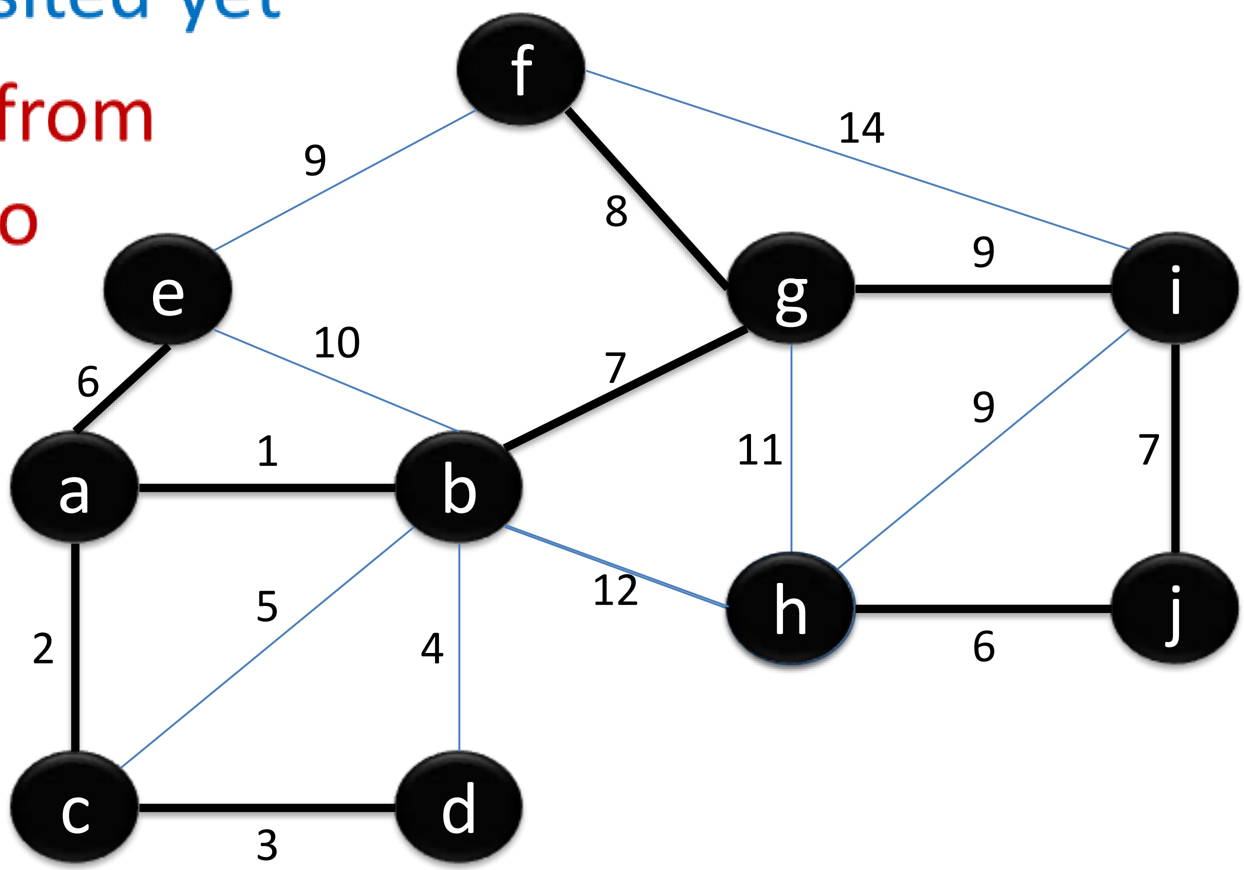
Running Prim's algorithm

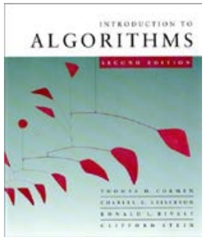
- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



Running Prim’s algorithm

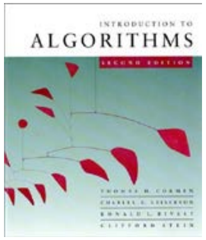
- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T
- Minimum Cost: 47





Analysis of Prim

$Q \leftarrow V$
$key[v] \leftarrow \infty$ for all $v \in V$
$key[s] \leftarrow 0$ for some arbitrary $s \in V$
while $Q \neq \emptyset$
    do $u \leftarrow$ EXTRACT-MIN(Q)
        for each $v \in Adj[u]$
            do if $v \in Q$ and $w(u, v) < key[v]$
                then $key[v] \leftarrow w(u, v)$
                      $\pi[v] \leftarrow u$



Analysis of Prim

$\Theta(V)$ total:

$Q \leftarrow V$

$key[v] \leftarrow \infty$ for all $v \in V$

$key[s] \leftarrow 0$ for some arbitrary $s \in V$

while $Q \neq \emptyset$

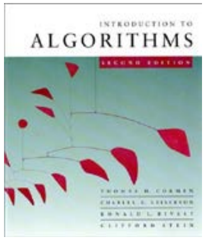
    do $u \leftarrow$ EXTRACT-MIN(Q)

        for each $v \in Adj[u]$

            do if $v \in Q$ and $w(u, v) < key[v]$

                then $key[v] \leftarrow w(u, v)$

                        $\pi[v] \leftarrow u$



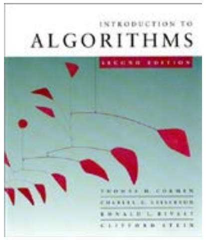
Analysis of Prim

$\Theta(V)$ total:

$Q \leftarrow V$
$key[v] \leftarrow \infty$ for all $v \in V$
$key[s] \leftarrow 0$ for some arbitrary $s \in V$

$|V|$ times:

while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)
    for each $v \in Adj[u]$
      do if $v \in Q$ and $w(u, v) < key[v]$
        then $key[v] \leftarrow w(u, v)$
          $\pi[v] \leftarrow u$



Analysis of Prim

$\Theta(V)$ total

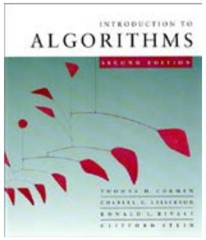
$Q \leftarrow V$
$key[v] \leftarrow \infty$ for all $v \in V$
$key[s] \leftarrow 0$ for some arbitrary $s \in V$

$|V|$ times

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

$degree(u)$ times

  for each $v \in Adj[u]$
    do if $v \in Q$ and $w(u, v) < key[v]$
      then $key[v] \leftarrow w(u, v)$
        $\pi[v] \leftarrow u$



Analysis of Prim

$\Theta(V)$ total:

$Q \leftarrow V$

$key[v] \leftarrow \infty$ for all $v \in V$

$key[s] \leftarrow 0$ for some arbitrary $s \in V$

$|V|$ times:

while $Q \neq \emptyset$

    do $u \leftarrow$ EXTRACT-MIN(Q)

$degree(u)$ times:

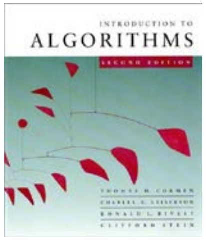
        for each $v \in Adj[u]$

            do if $v \in Q$ and $w(u, v) < key[v]$

                then $key[v] \leftarrow w(u, v)$

                      $\pi[v] \leftarrow u$

Handshaking Lemma $\Rightarrow \Theta(E)$ implicit DECREASE-KEY's.



Analysis of Prim

$\Theta(V)$ total:

$Q \leftarrow V$
$key[v] \leftarrow \infty$ for all $v \in V$
$key[s] \leftarrow 0$ for some arbitrary $s \in V$

$|V|$ times:

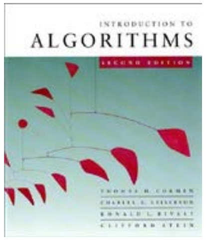
while $Q \neq \emptyset$
  do $u \leftarrow$ EXTRACT-MIN(Q)

$degree(u)$ times:

  for each $v \in Adj[u]$
    do if $v \in Q$ and $w(u, v) < key[v]$
      then $key[v] \leftarrow w(u, v)$
        $\pi[v] \leftarrow u$

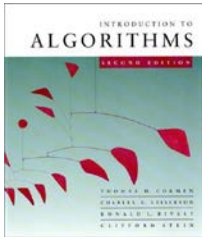
Handshaking Lemma $\Rightarrow \Theta(E)$ implicit DECREASE-KEY's.

$$\text{Time} = \Theta(V)\cdot T_{\text{EXTRACT-MIN}} + \Theta(E)\cdot T_{\text{DECREASE-KEY}}$$



Analysis of Prim (continued)

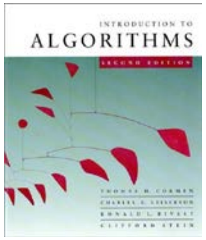
Time $= \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$



Analysis of Prim (continued)

Time = $\Theta(V)\cdot T_{\text{EXTRACT-MIN}} + \Theta(E)\cdot T_{\text{DECREASE-KEY}}$

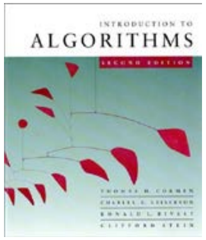
Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total



Analysis of Prim (continued)

Time = $\Theta(V)\cdot T_{\text{EXTRACT-MIN}} + \Theta(E)\cdot T_{\text{DECREASE-KEY}}$

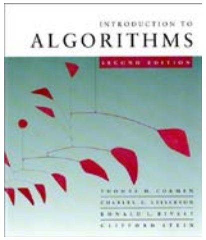
Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$



Analysis of Prim (continued)

Time = $\Theta(V)\cdot T_{\text{EXTRACT-MIN}} + \Theta(E)\cdot T_{\text{DECREASE-KEY}}$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$

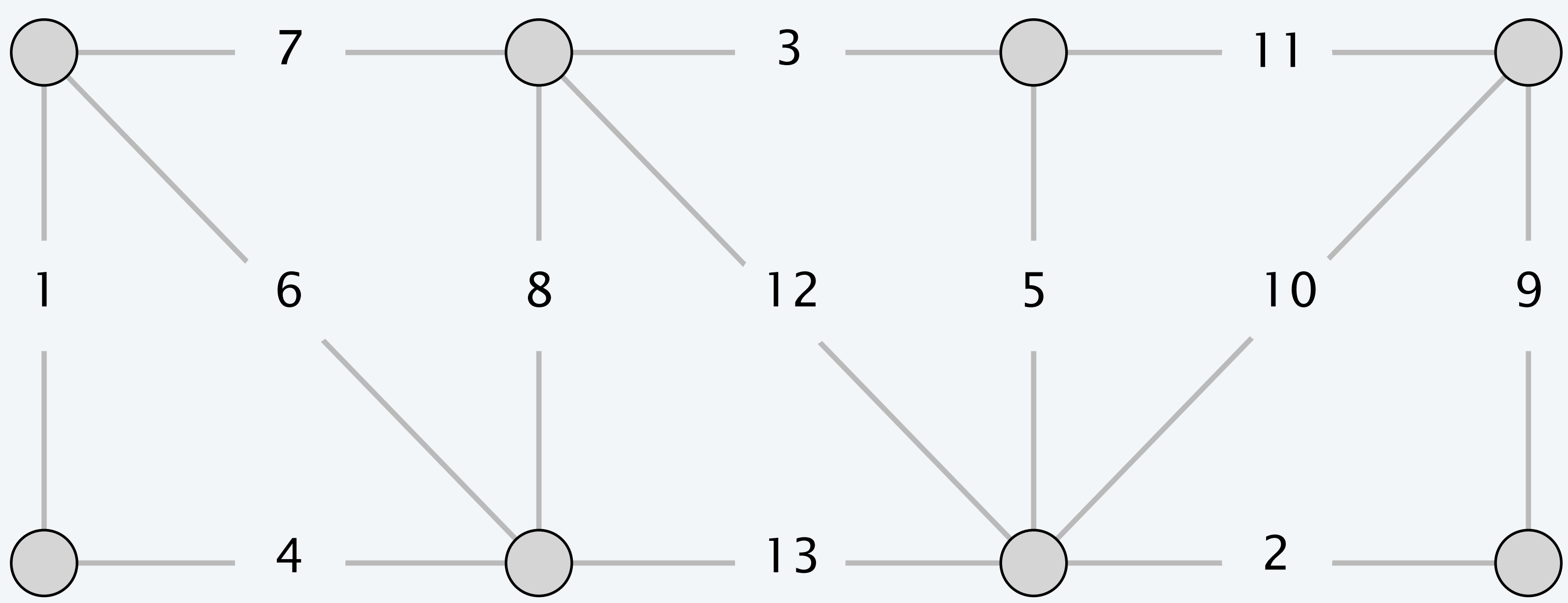


Analysis of Prim (continued)

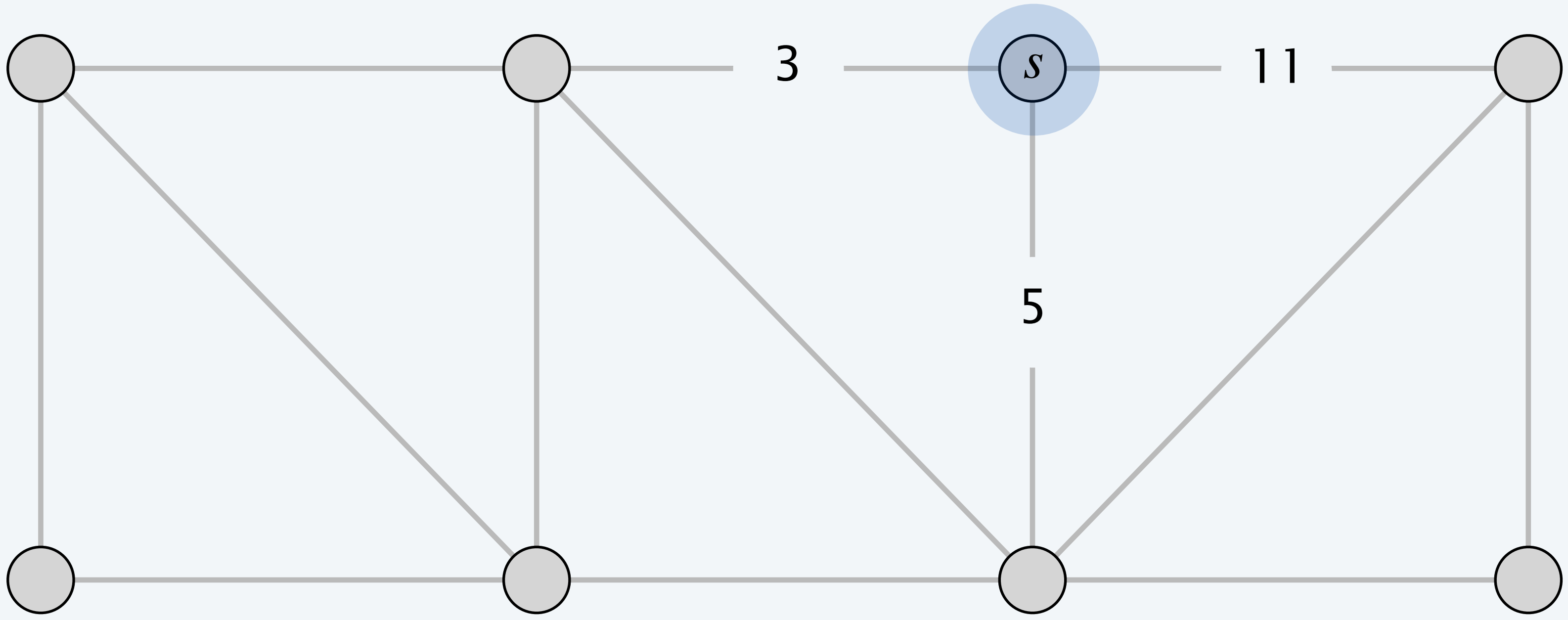
Time = $\Theta(V)\cdot T_{\text{EXTRACT-MIN}} + \Theta(E)\cdot T_{\text{DECREASE-KEY}}$

Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci heap	$O(\lg V)$ amortized	$O(1)$ amortized	$O(E + V \lg V)$ worst case

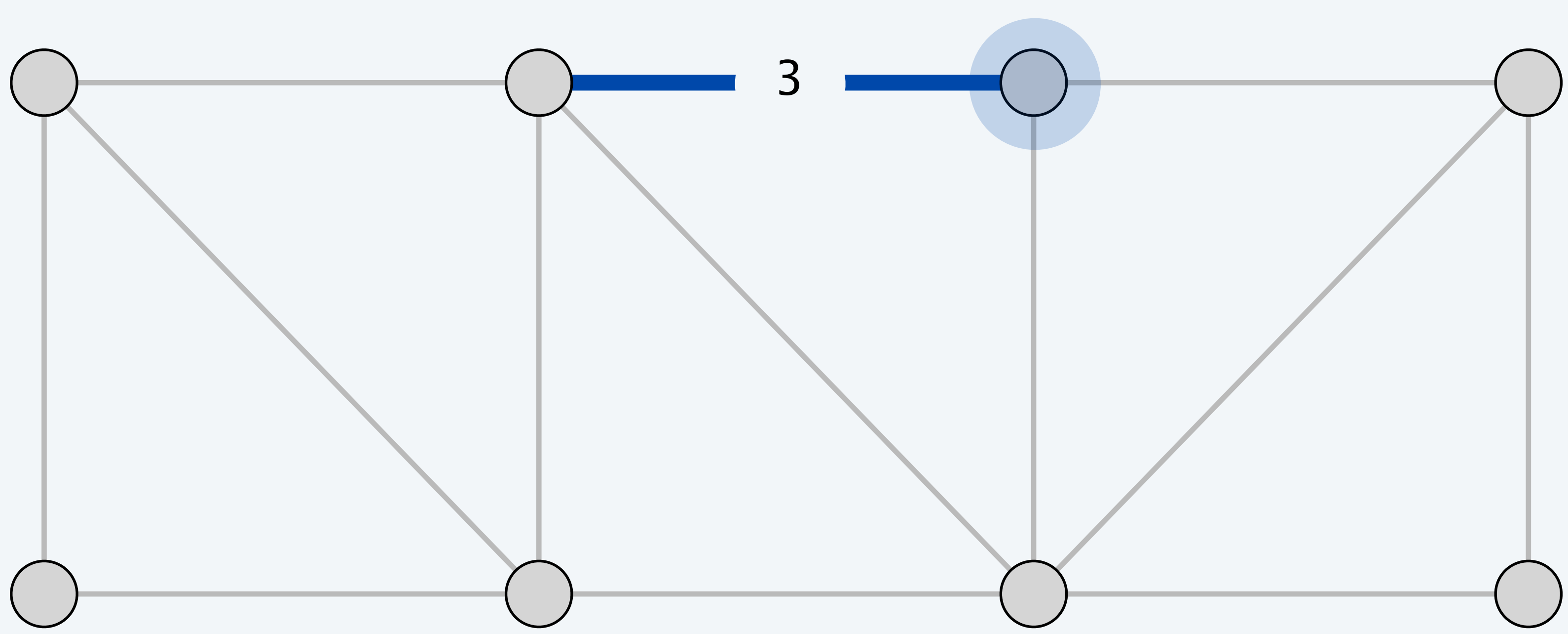
Prim's algorithm demo



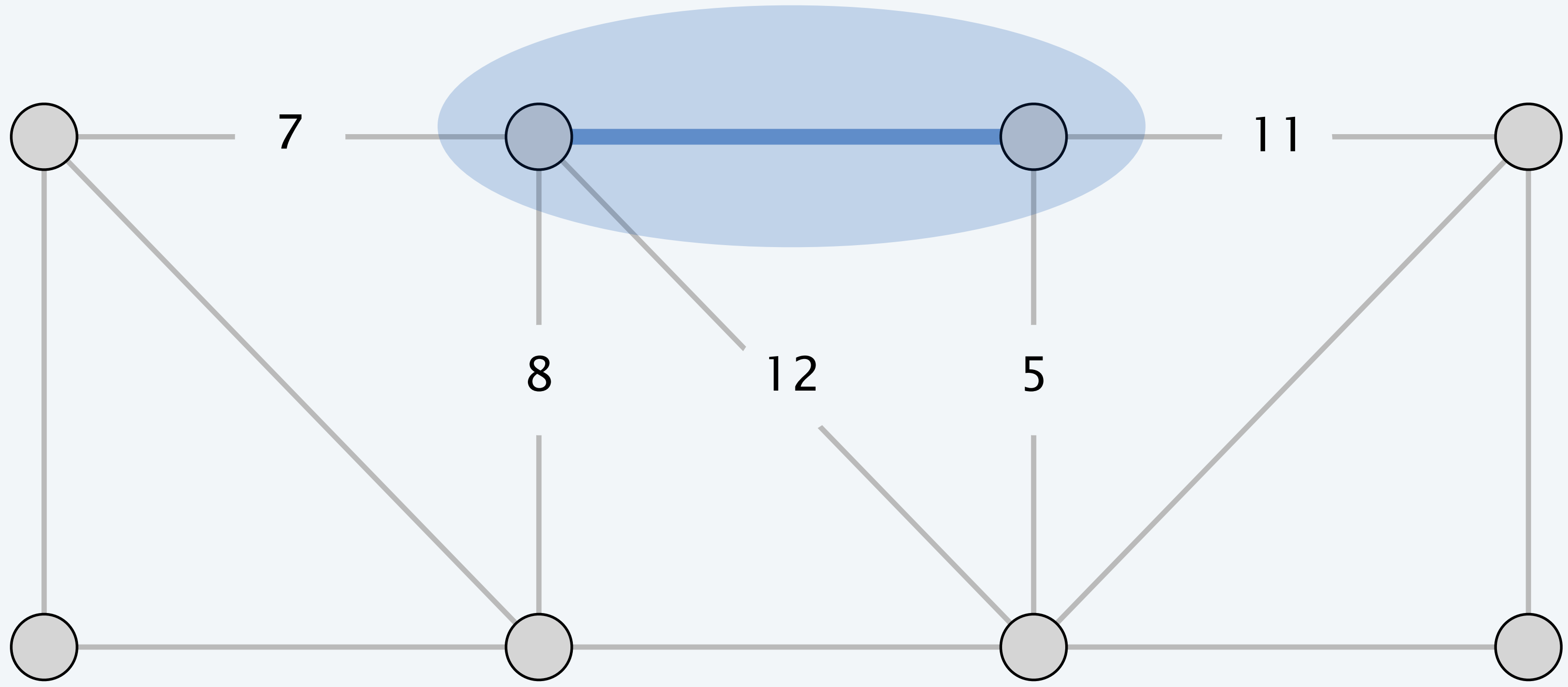
Prim's algorithm demo



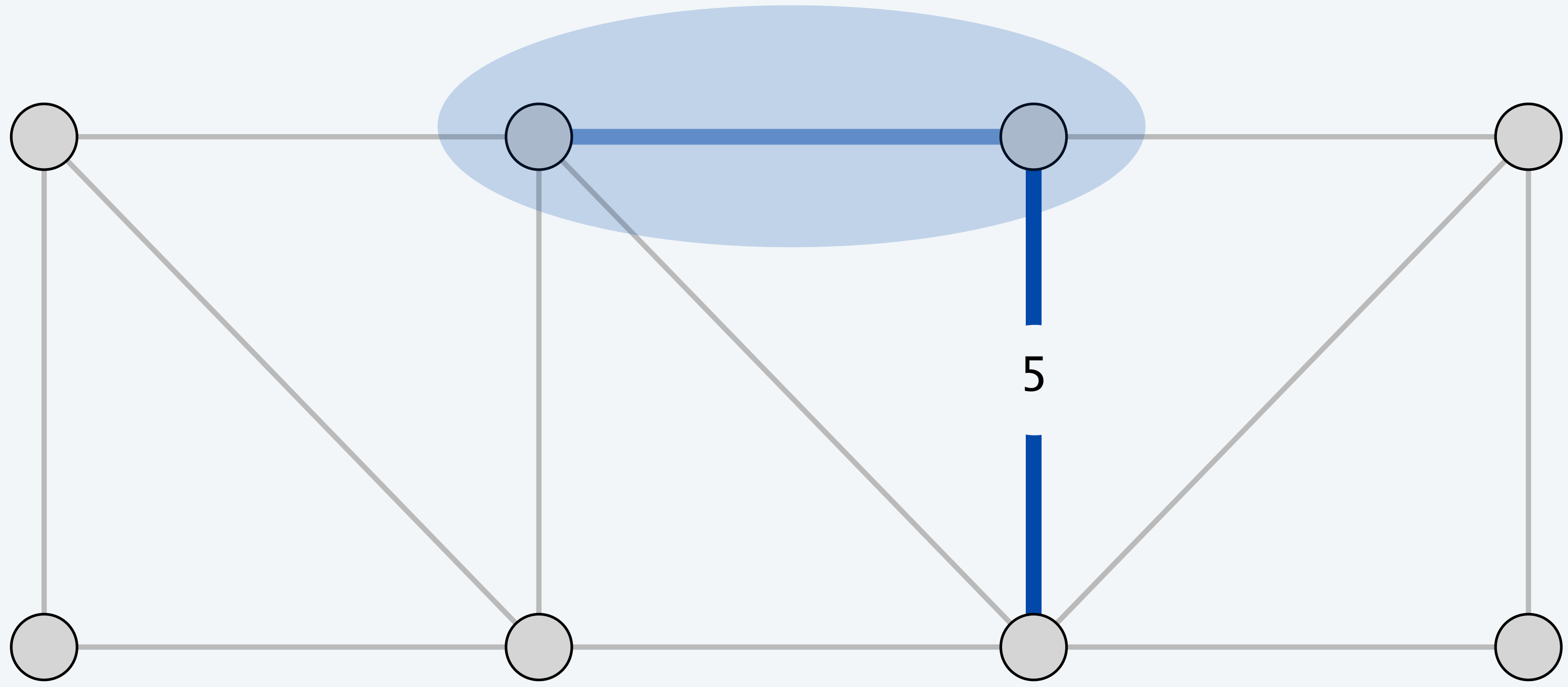
Prim's algorithm demo



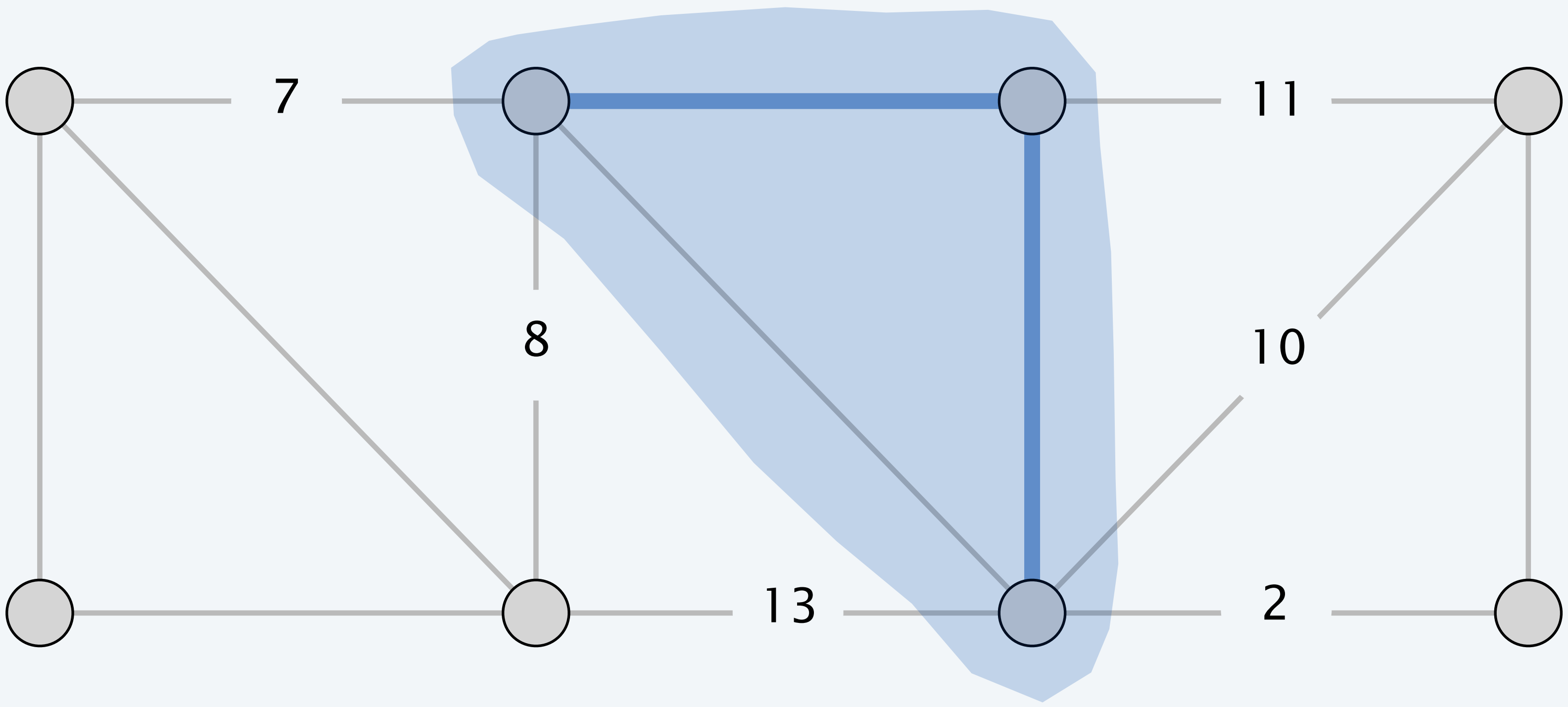
Prim's algorithm demo



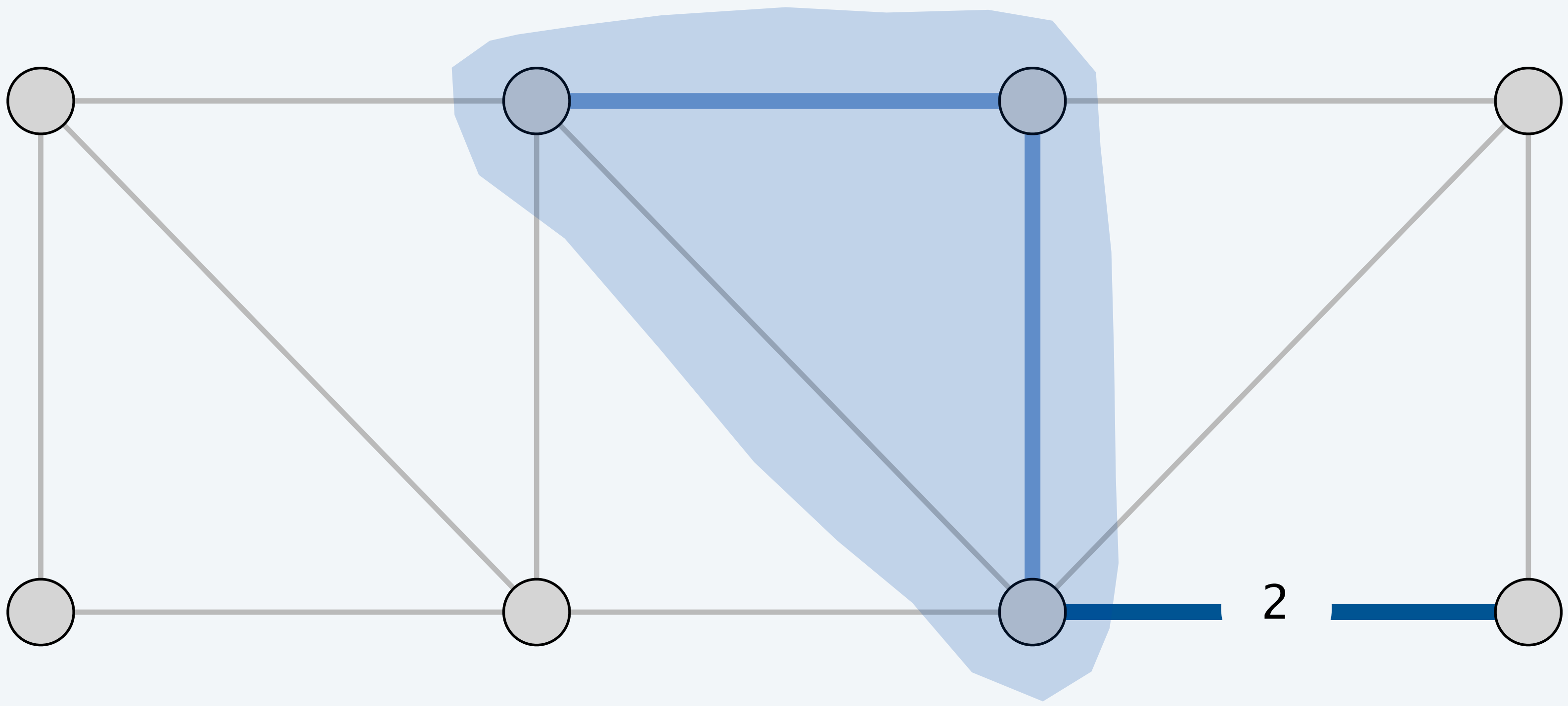
Prim's algorithm demo



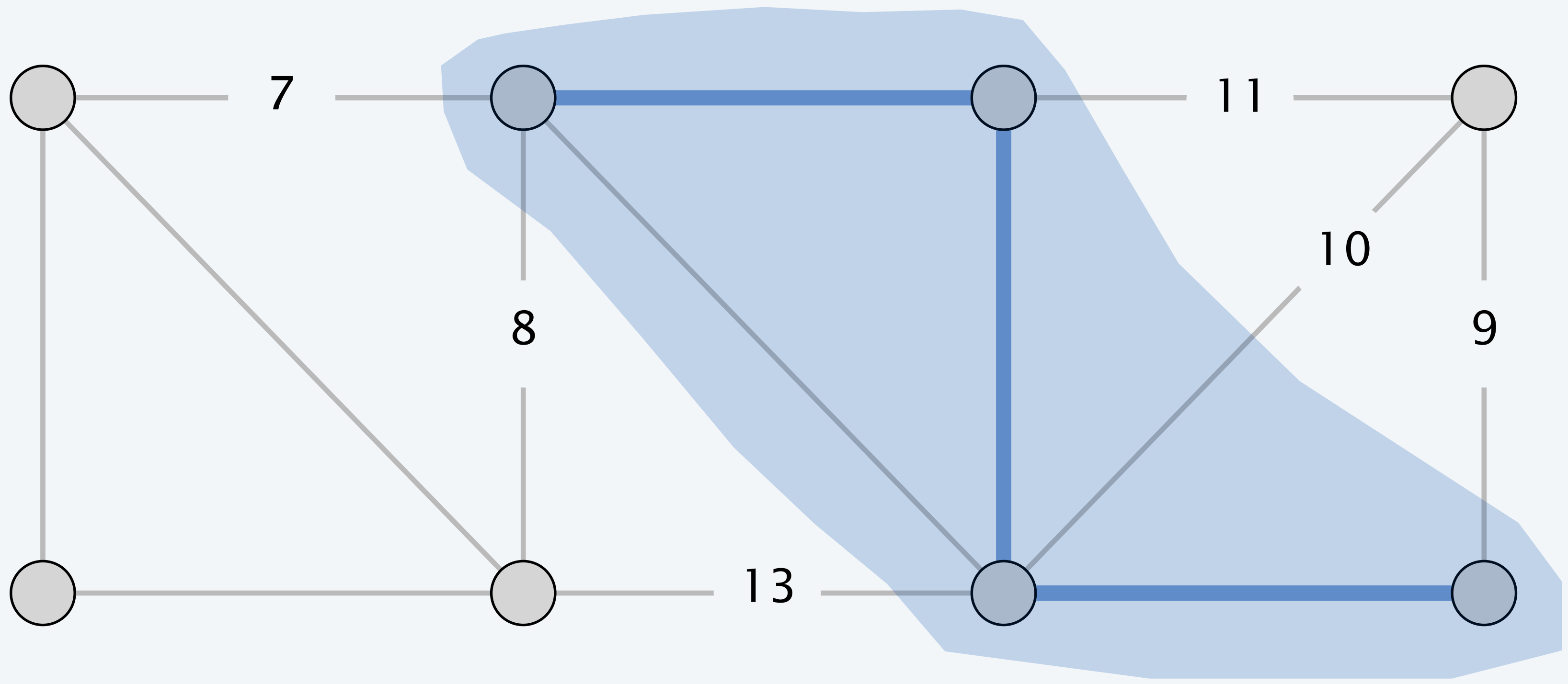
Prim's algorithm demo



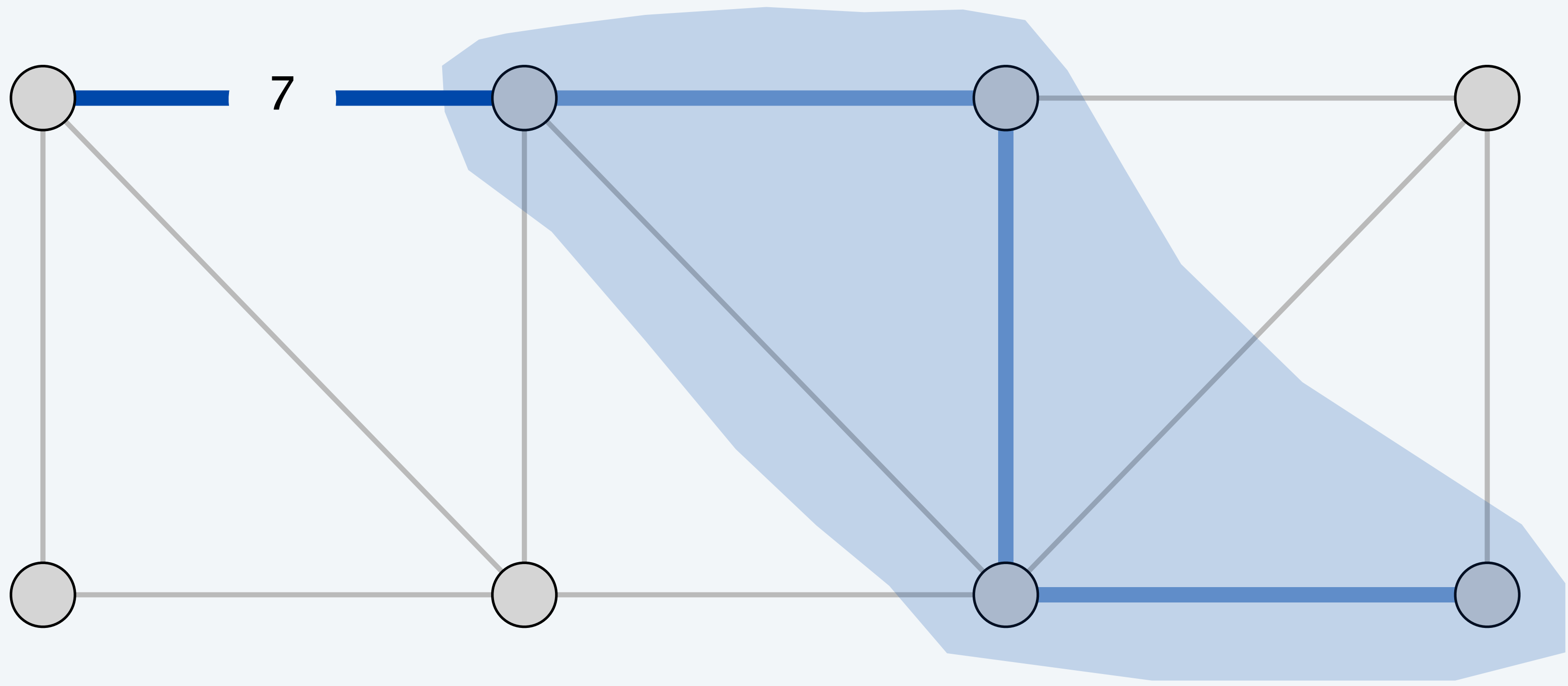
Prim's algorithm demo



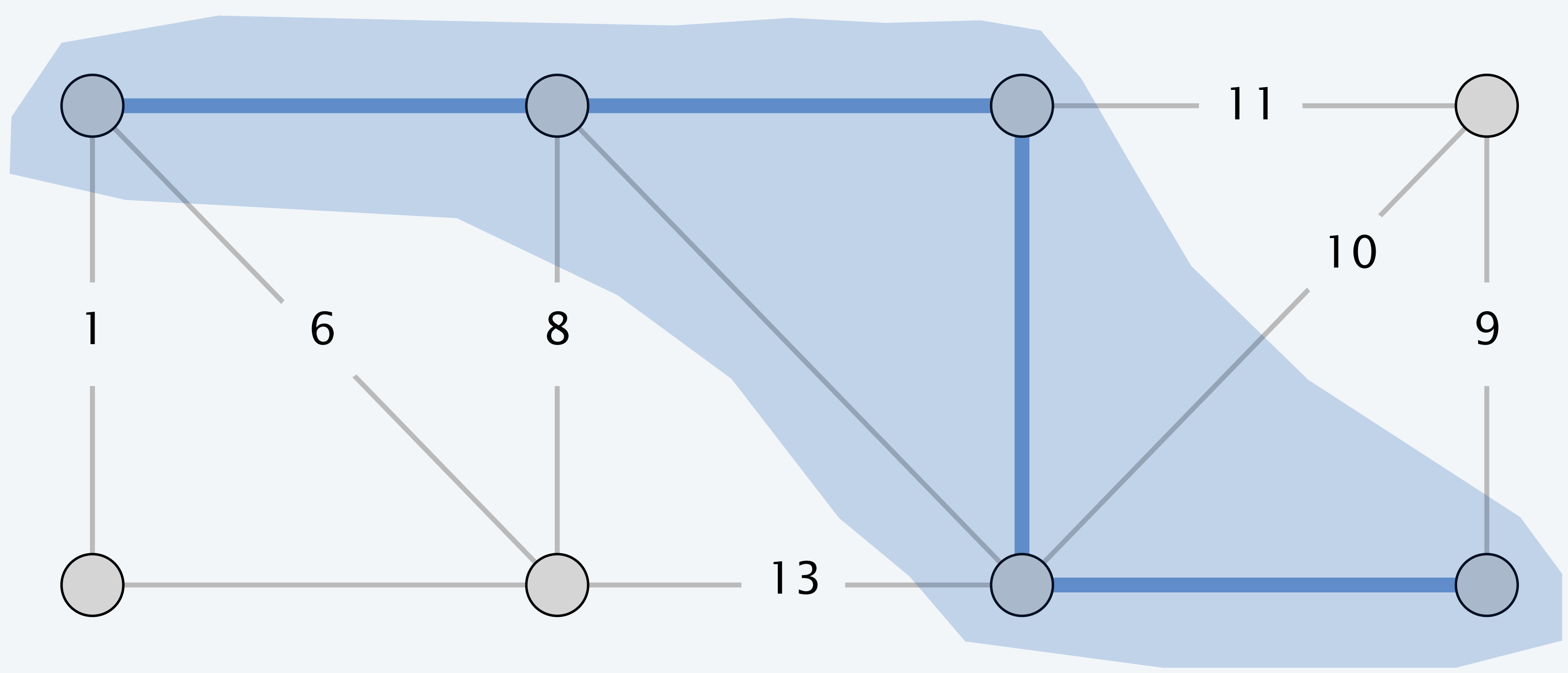
Prim's algorithm demo



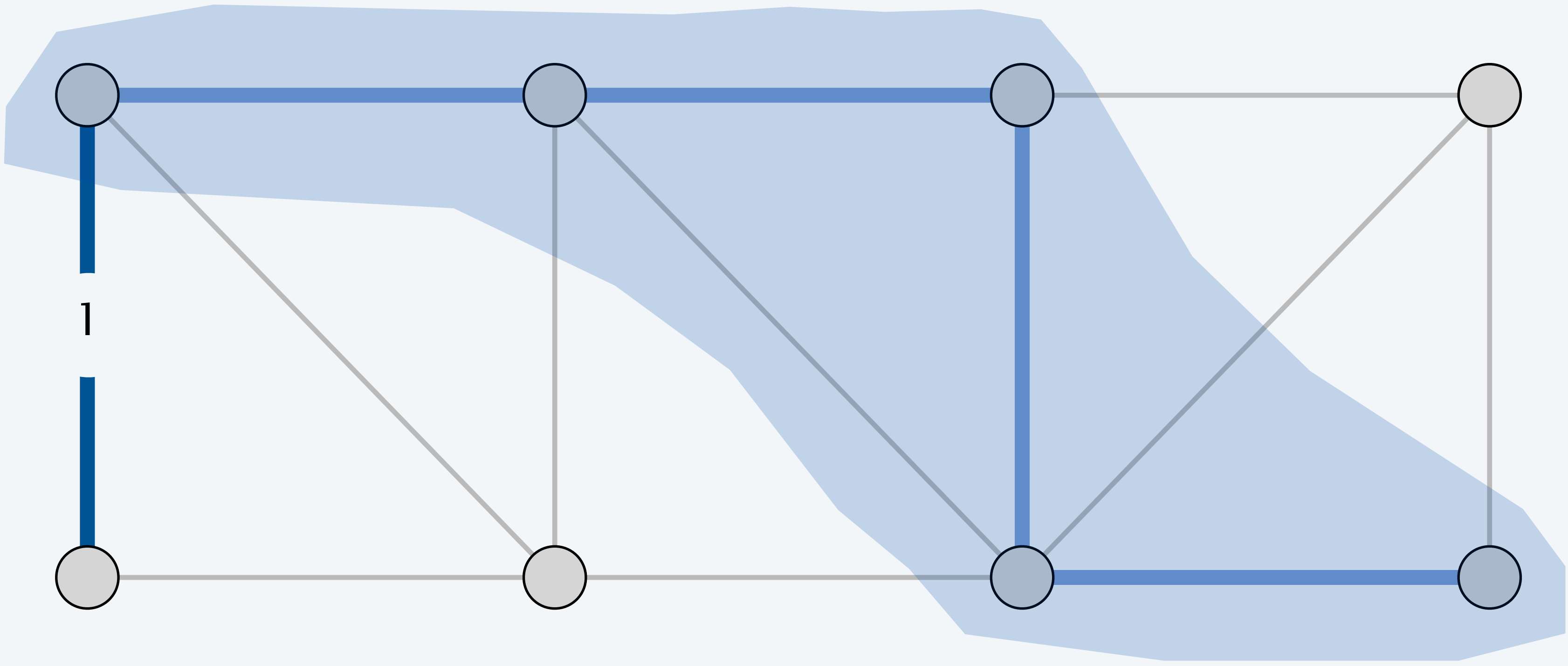
Prim's algorithm demo



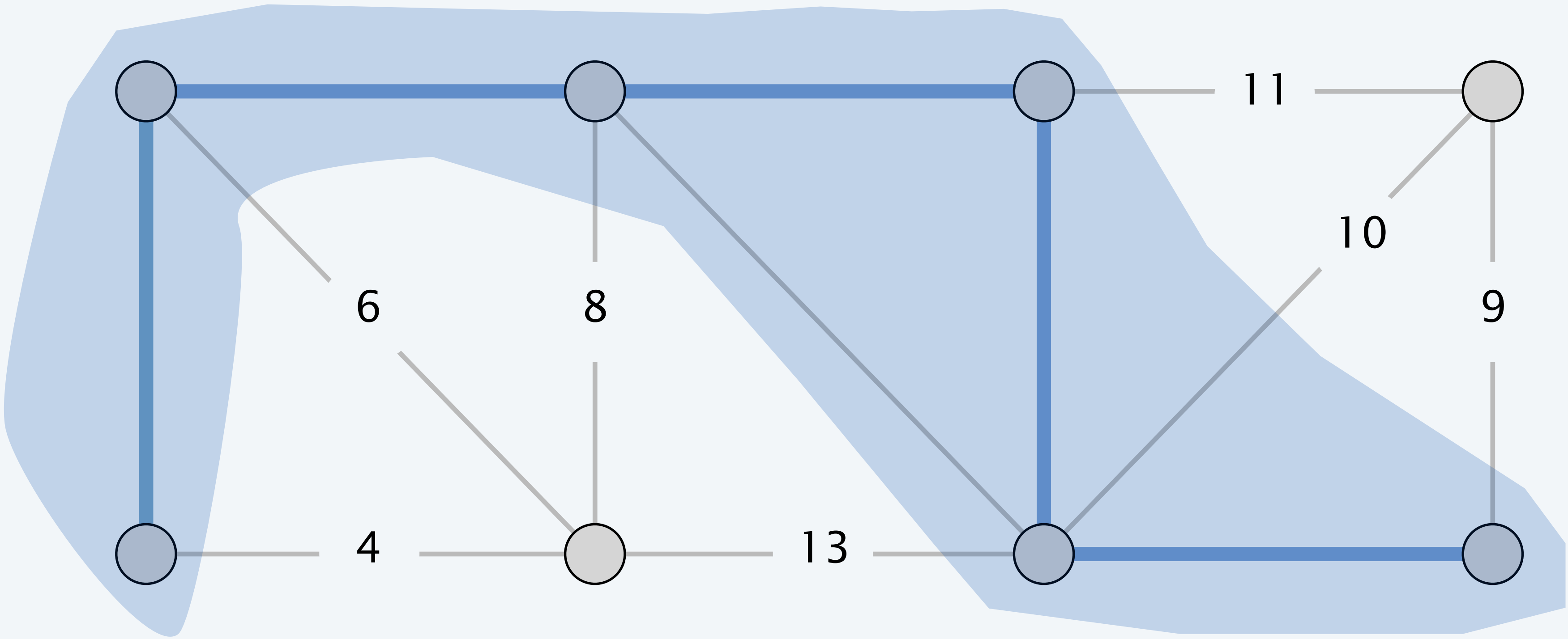
Prim's algorithm demo



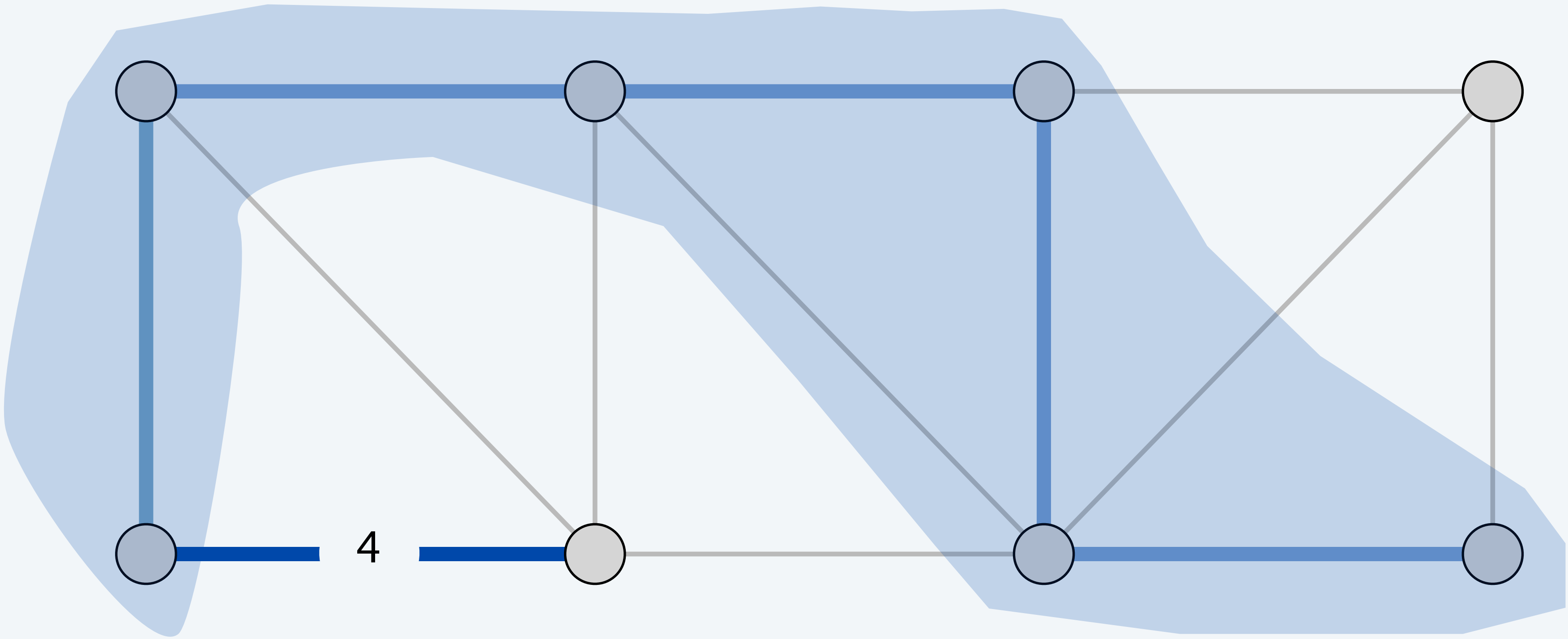
Prim's algorithm demo



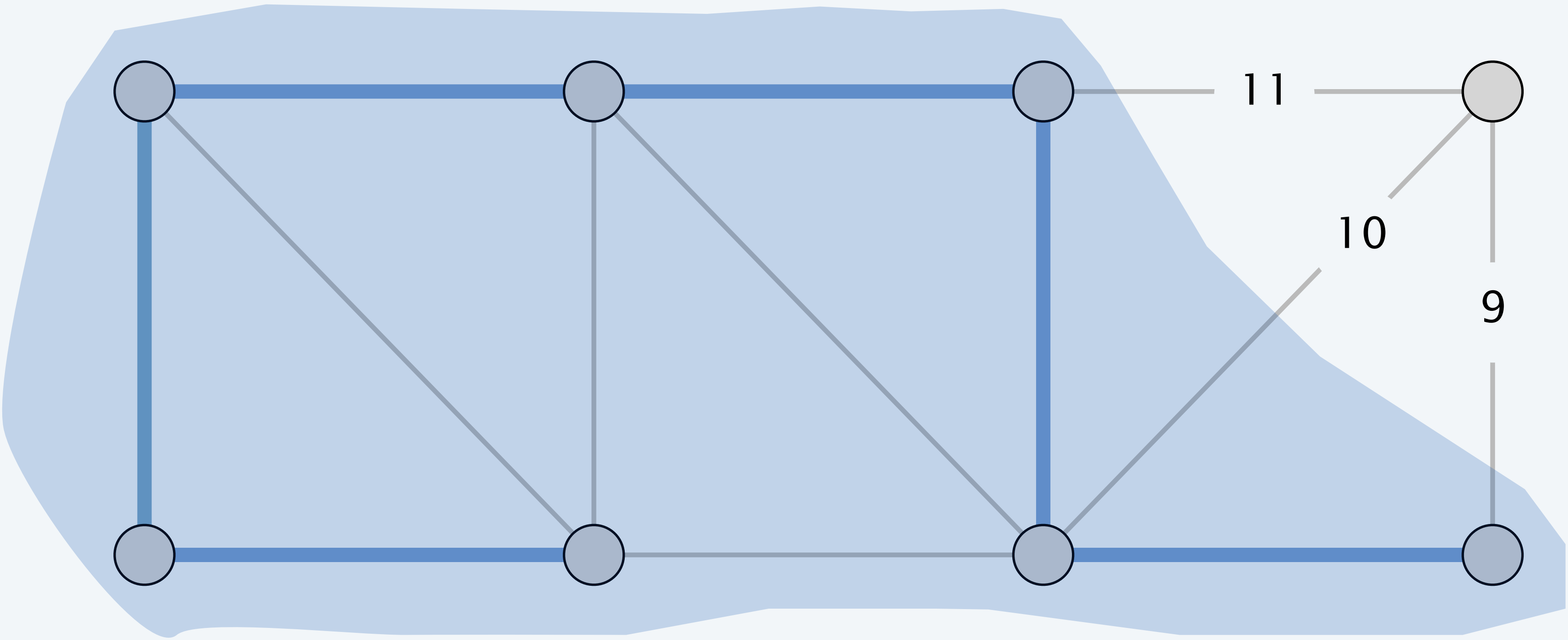
Prim's algorithm demo



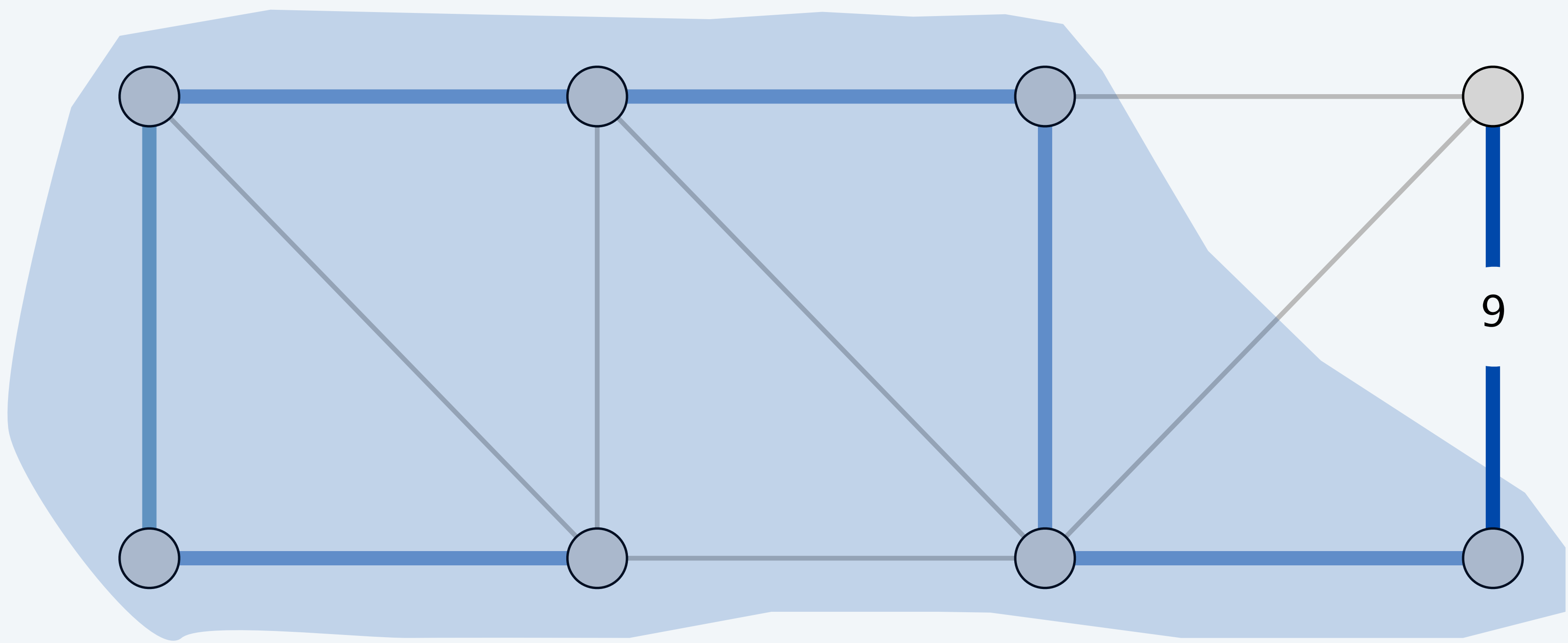
Prim's algorithm demo



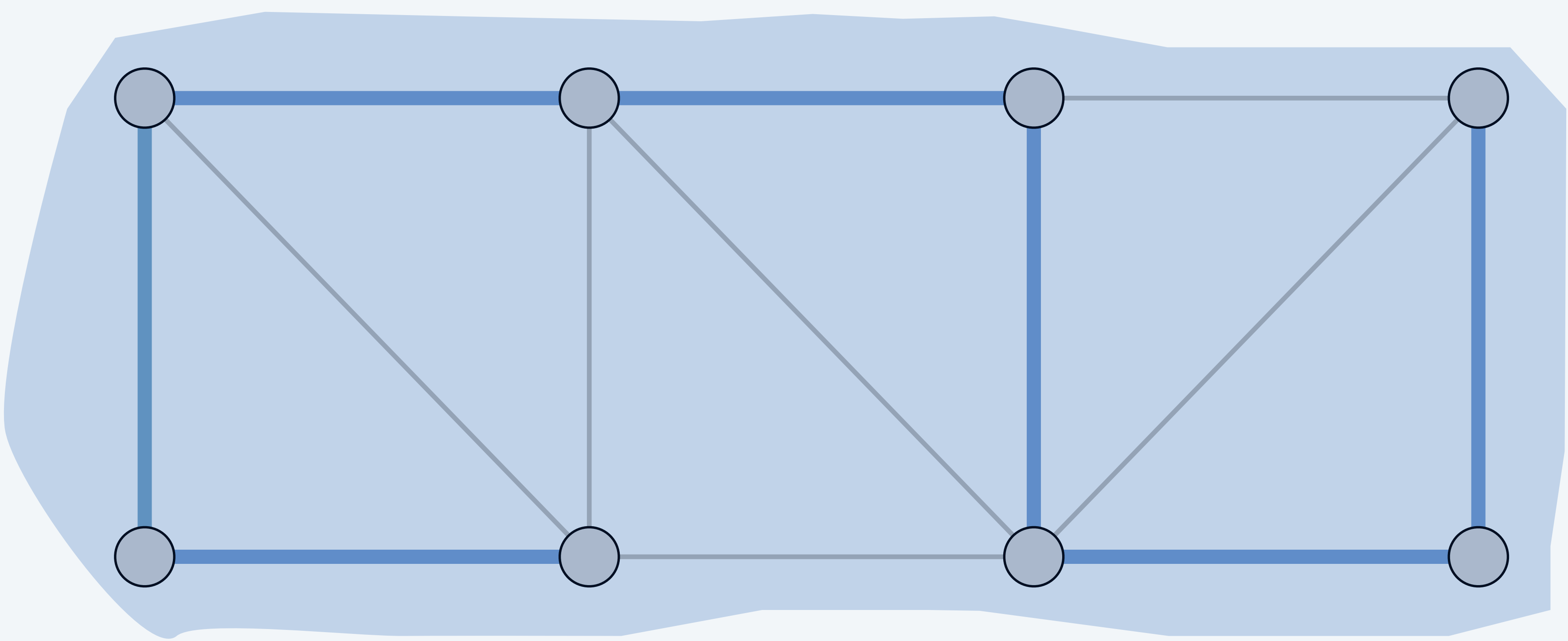
Prim's algorithm demo



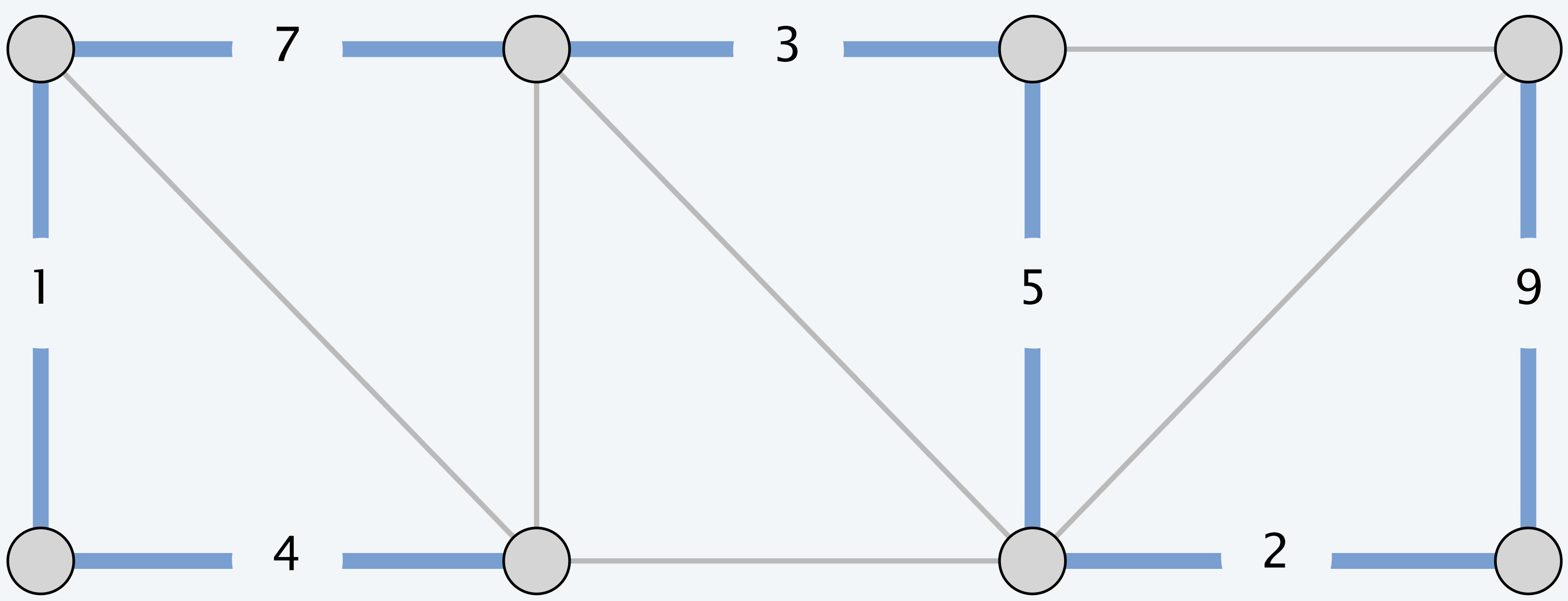
Prim's algorithm demo



Prim's algorithm demo



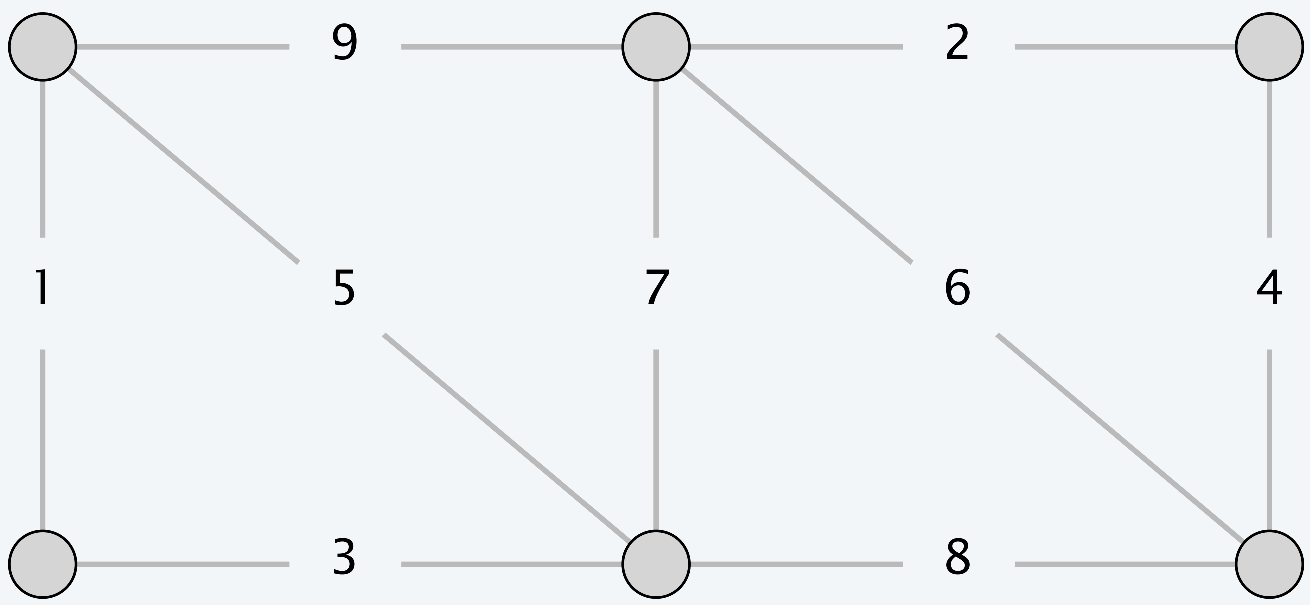
Prim's algorithm demo



Kruskal's algorithm demo

Consider edges in ascending order of weight:

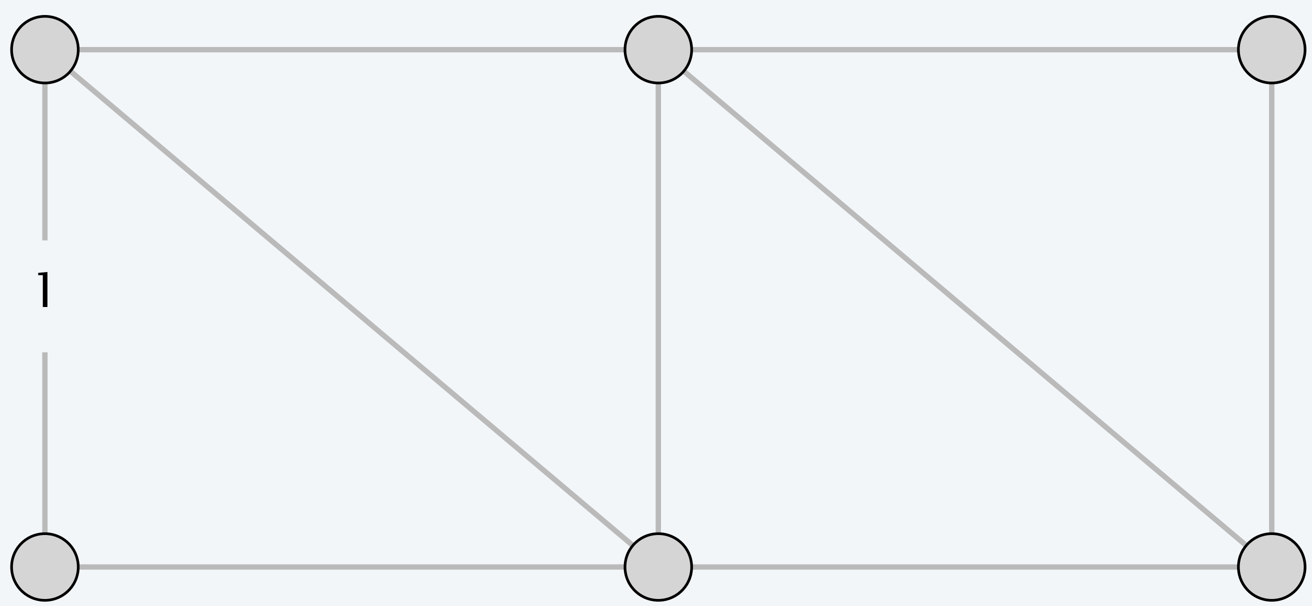
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

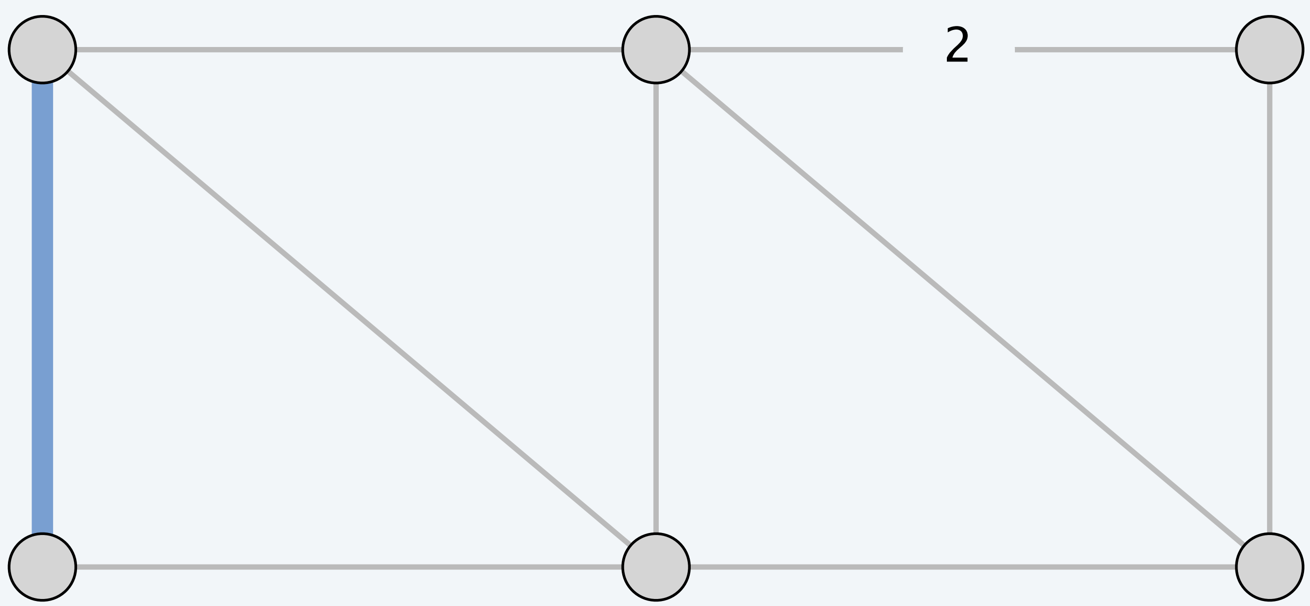
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

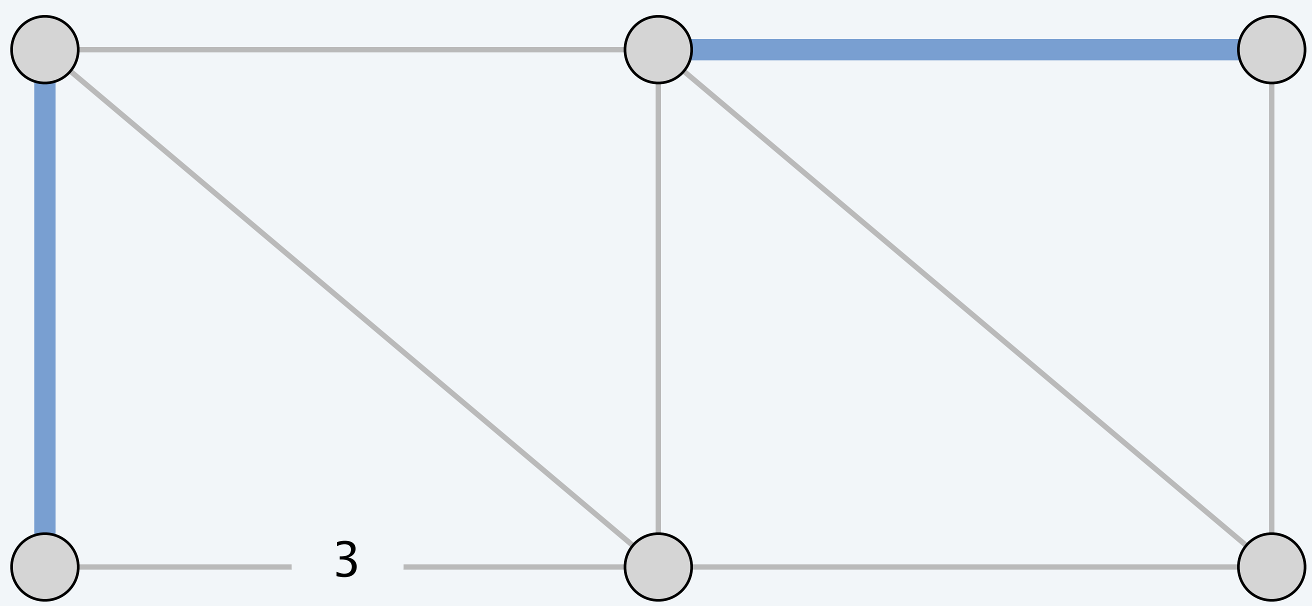
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

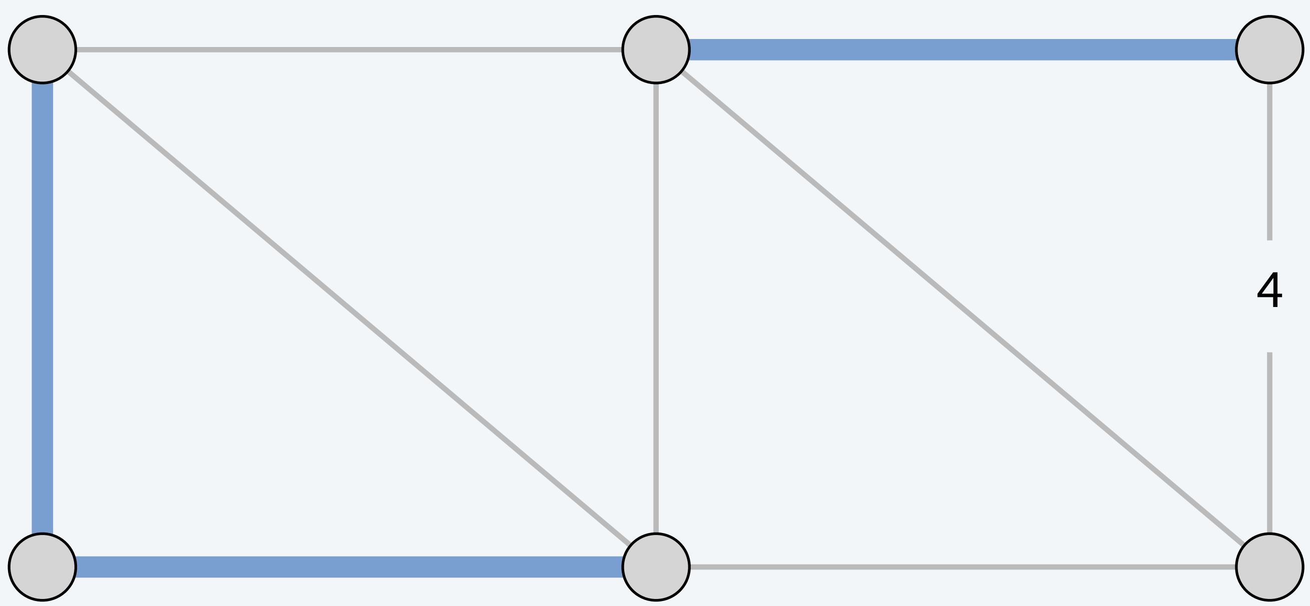
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

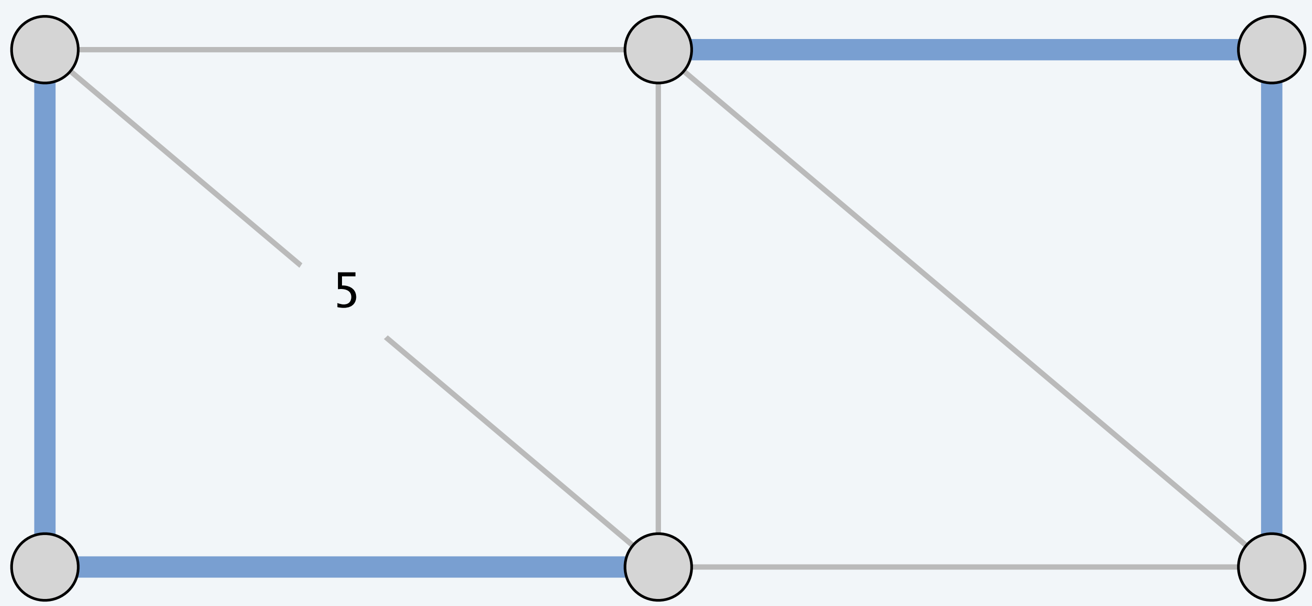
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

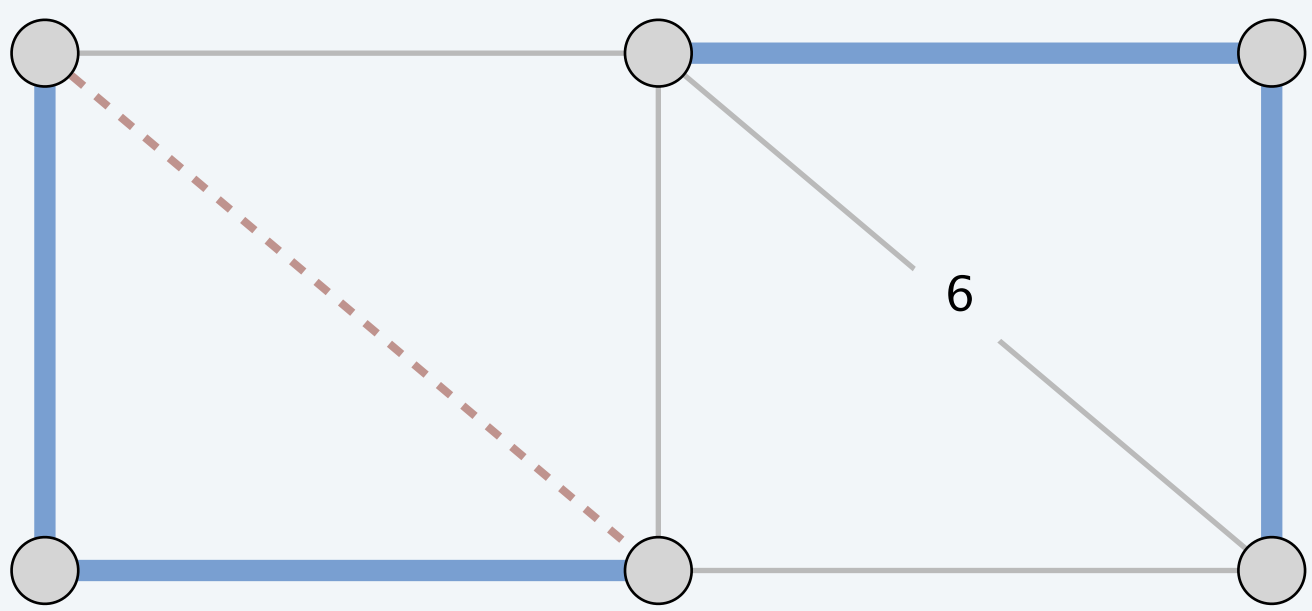
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

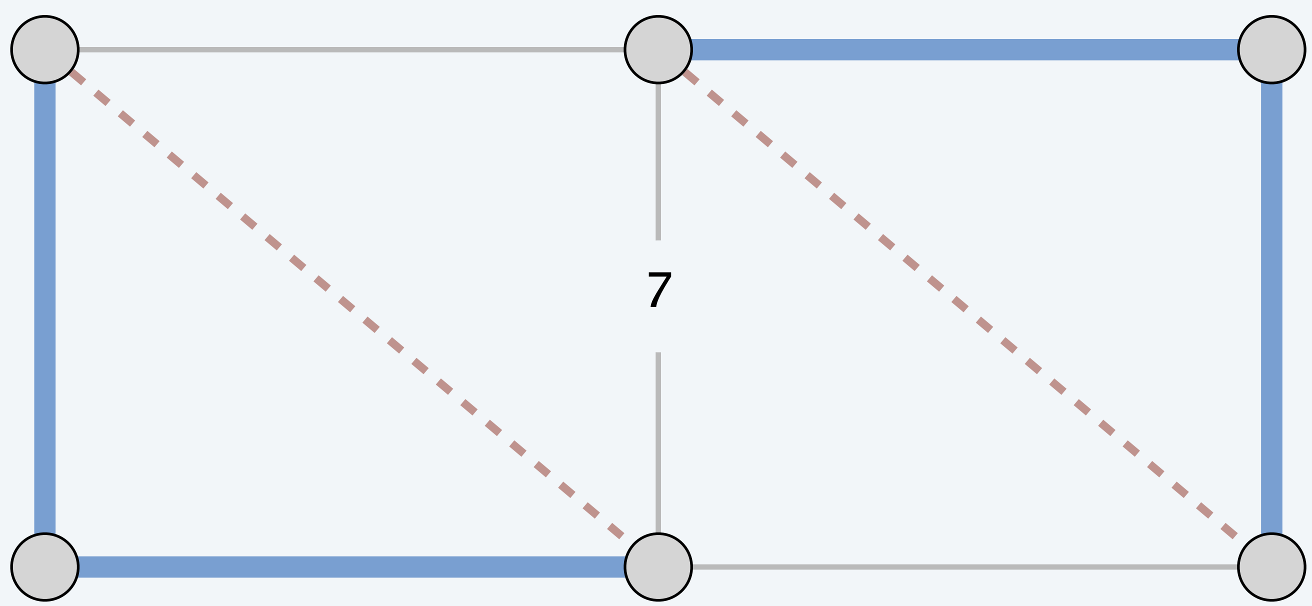
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

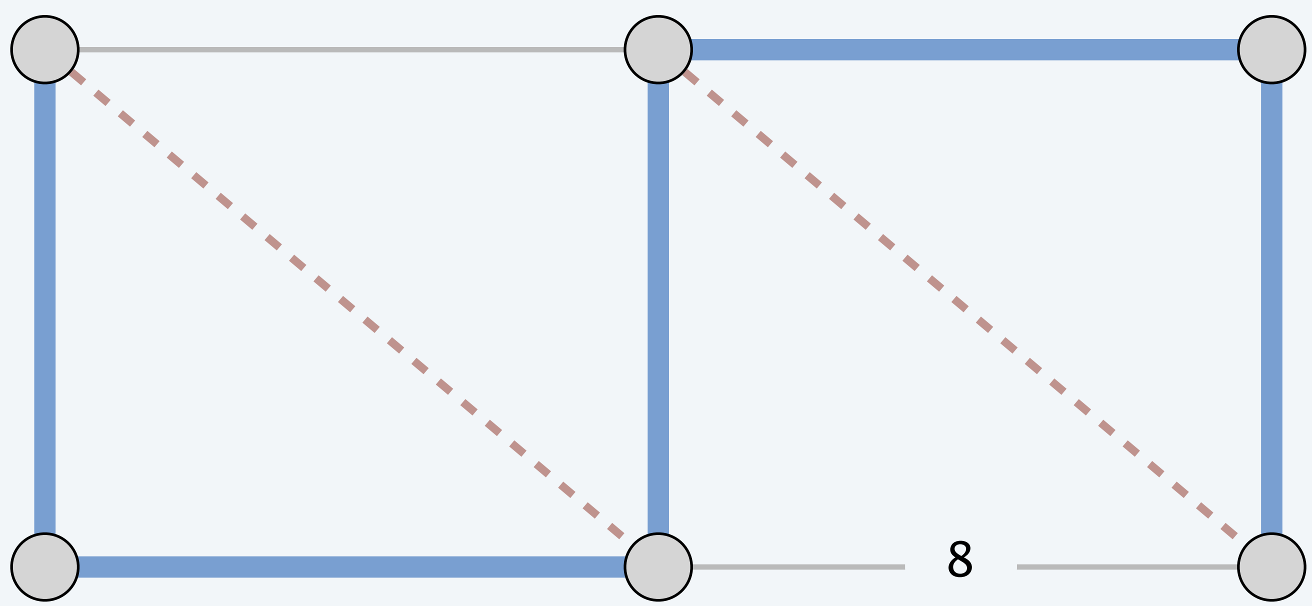
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

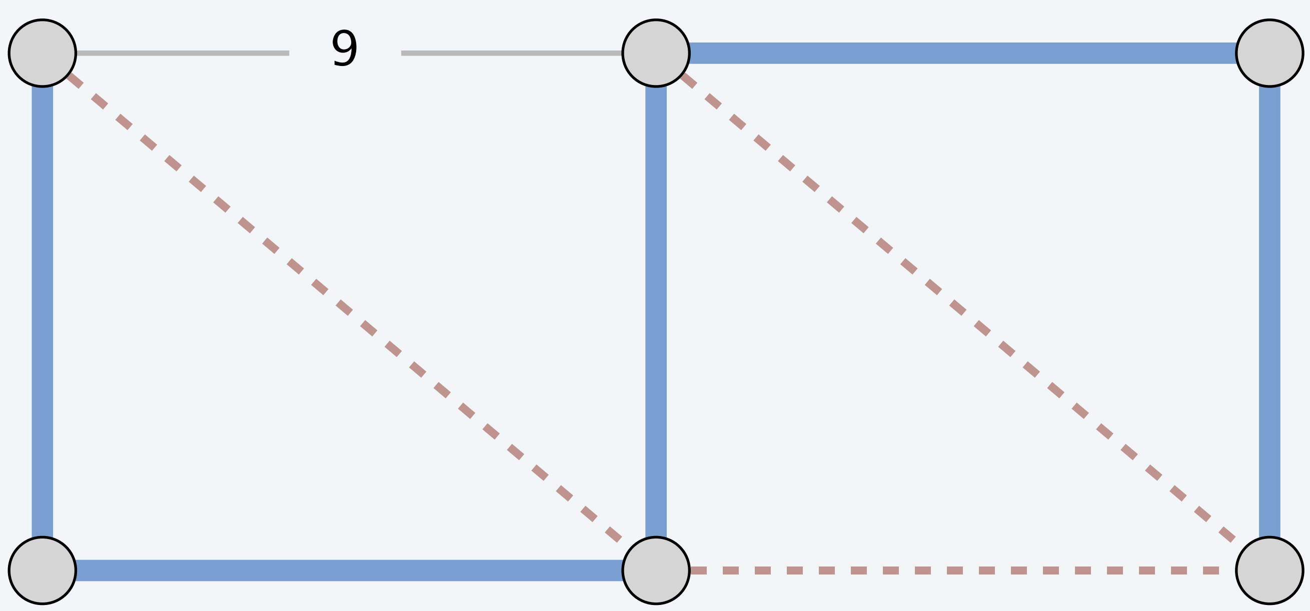
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

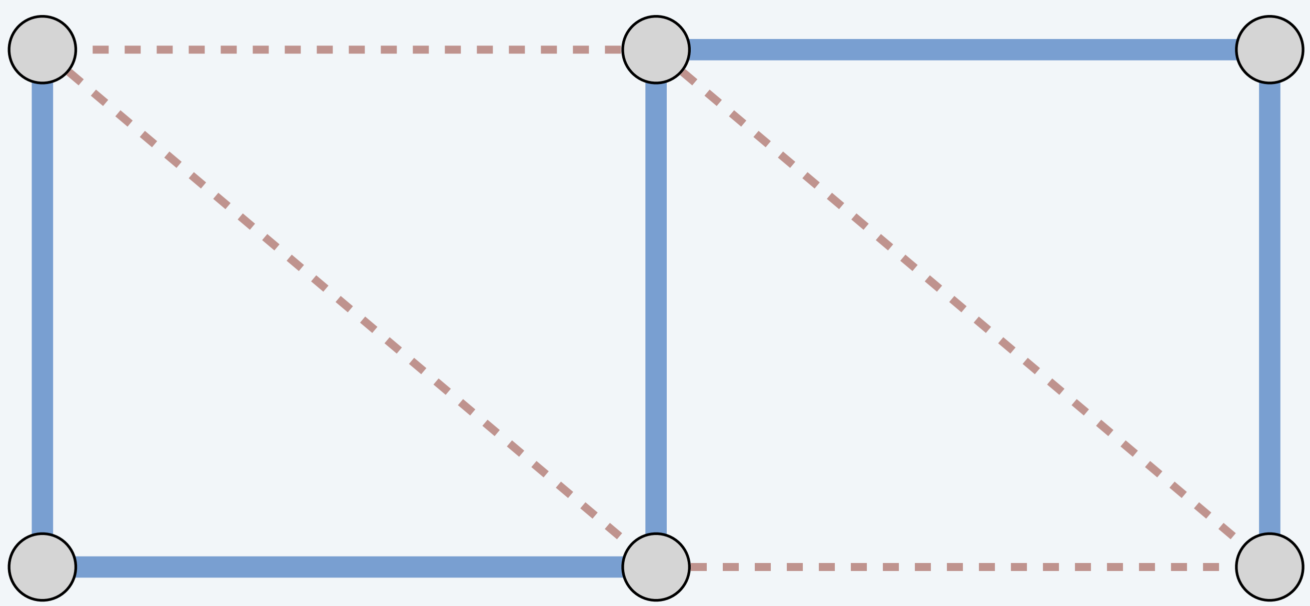
- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

- Add to T unless it would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight:

- Add to T unless it would create a cycle.