Minimum Spanning Tree

Fatemeh Panahi
Department of Computer Science
University of Toronto
CSC263-Fall 2017
Lecture 11

Today

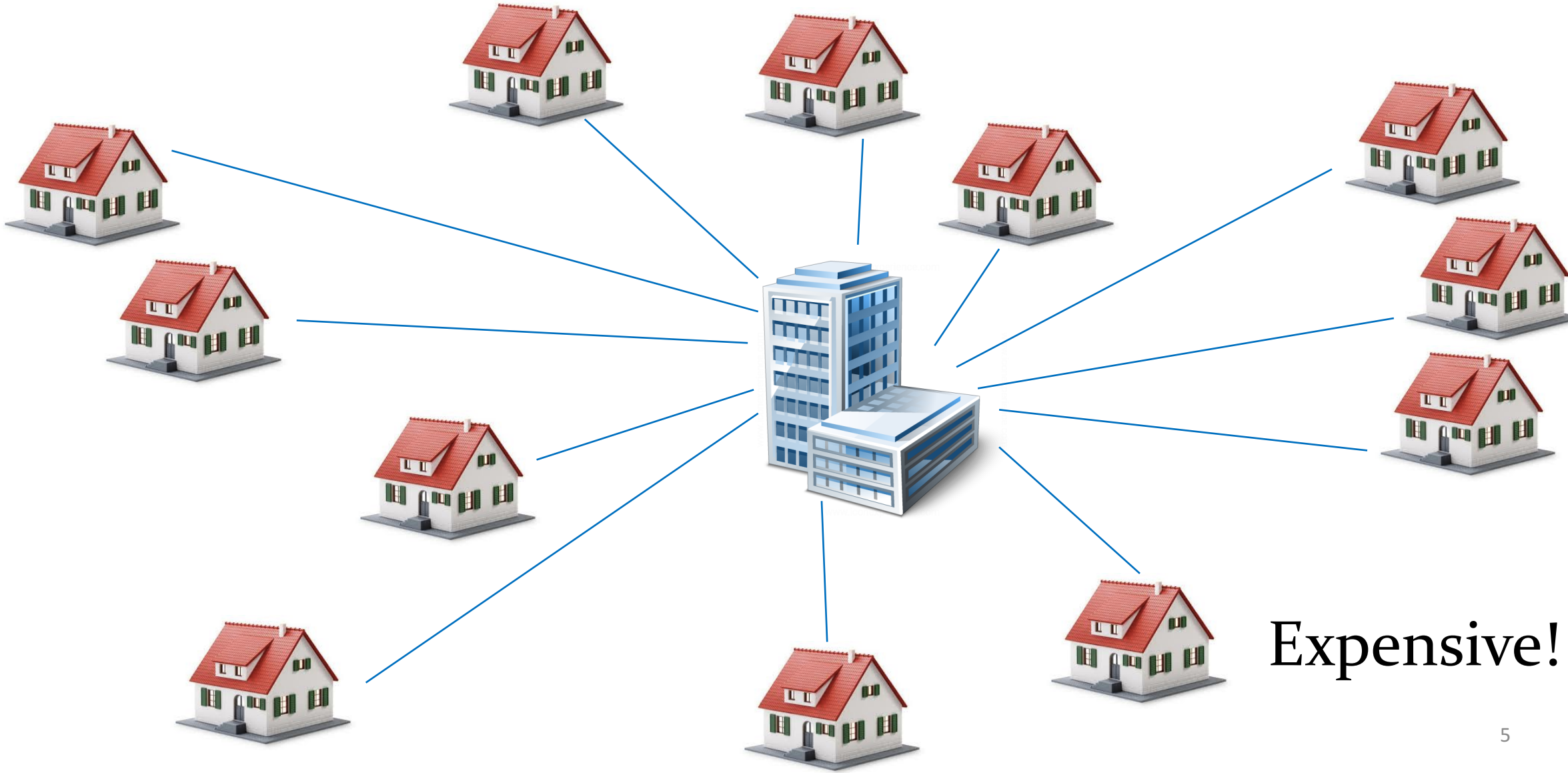
- Minimum Spaning Tree
- Prim's Algorithm
- Kruscal's Algorithm

Reading Assignments

Chapter 23 

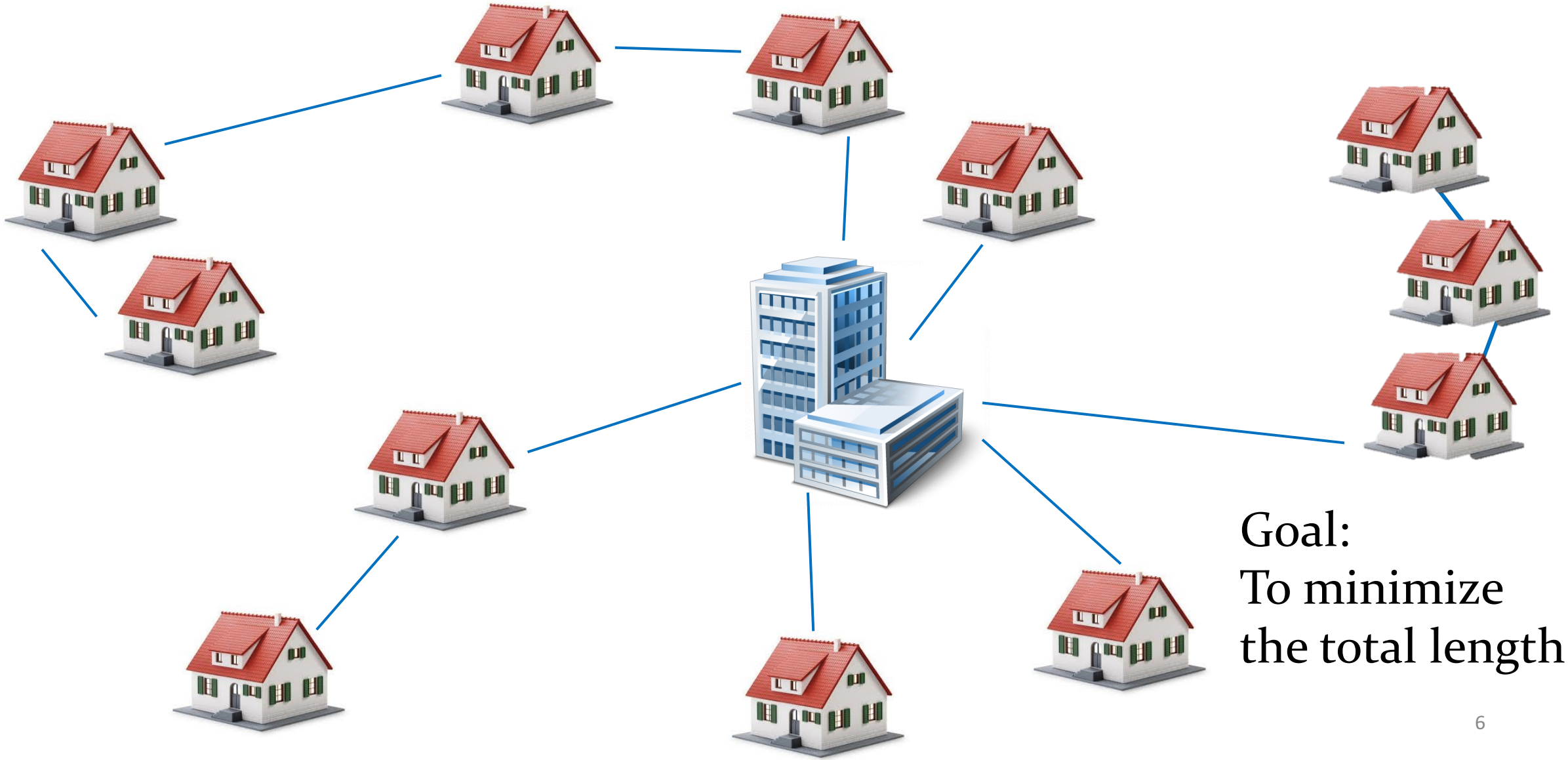
Problem: Laying Telephone wire

Problem: Laying Telephone wire



Expensive!

Problem: Laying Telephone wire



Goal:
To minimize
the total length

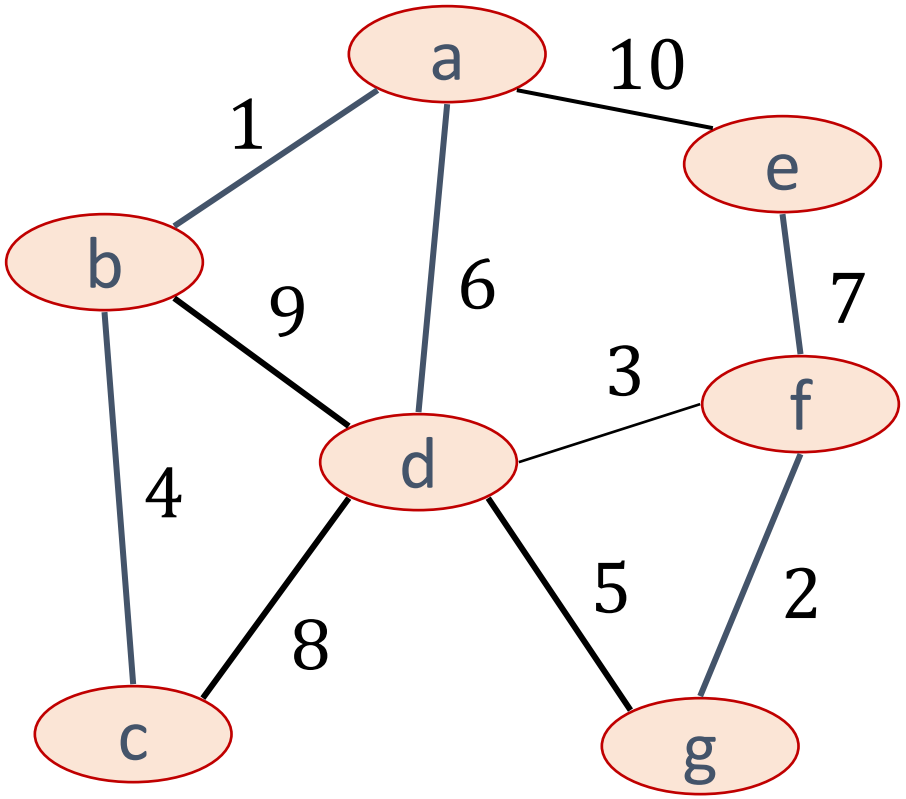
Similar Application

- Communication networks
- Transportation systems
- Electronic circuit design

Minimum Spanning Tree Problem

- **Input**: Connected, undirected, weighted graph
  - $G = (V, E)$
  - $w : E \rightarrow R$
- Find a spanning tree of G with smallest total weight.
- Spanning tree is a subset of graph G that contains all the vertices in G.
- If G is not connected, generalize to minimum spanning forest.

Example:

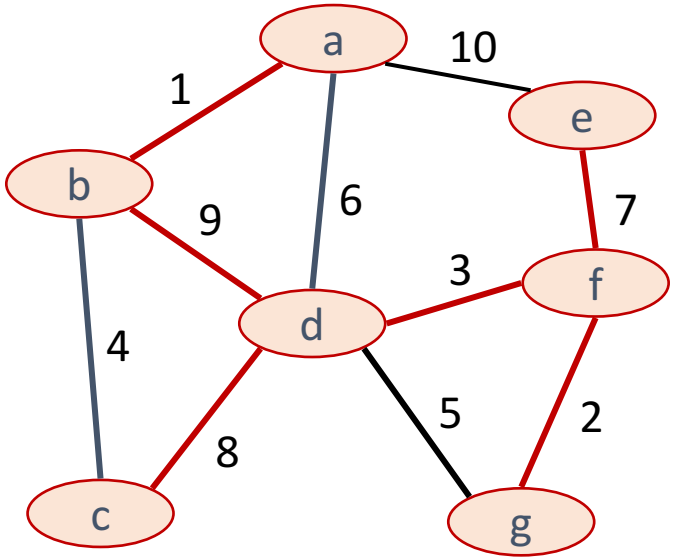


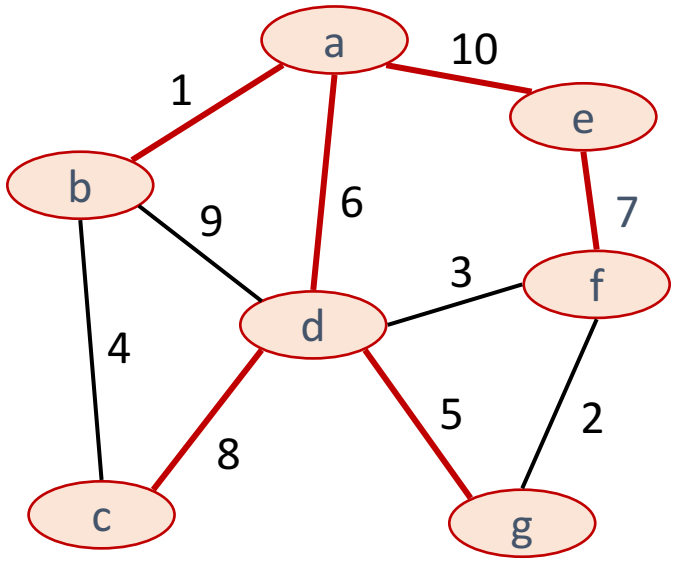
For a connected graph G = (V, E)

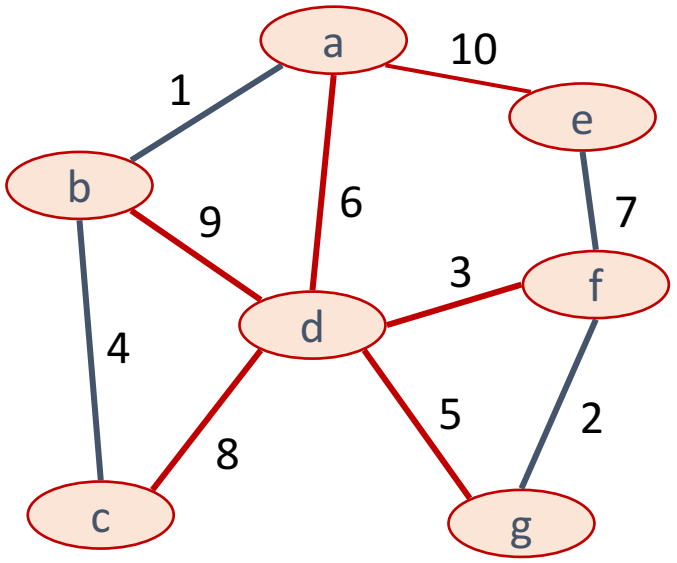
- What is the number of vertices in a minimum spanning tree?
- What is the number of edges in a minimum spanning tree?

Example:

- Some spanning trees







- We are looking for the one with minimum total weight

Greedy approach

GENERIC-MST(G)

```
1       A = {}
2       while A does not form a spanning tree
3           find an edge e that is safe for A
4           A = A ∪ {e}
5       return A
```

The greedy approach grows the minimum spanning tree one edge at a time.

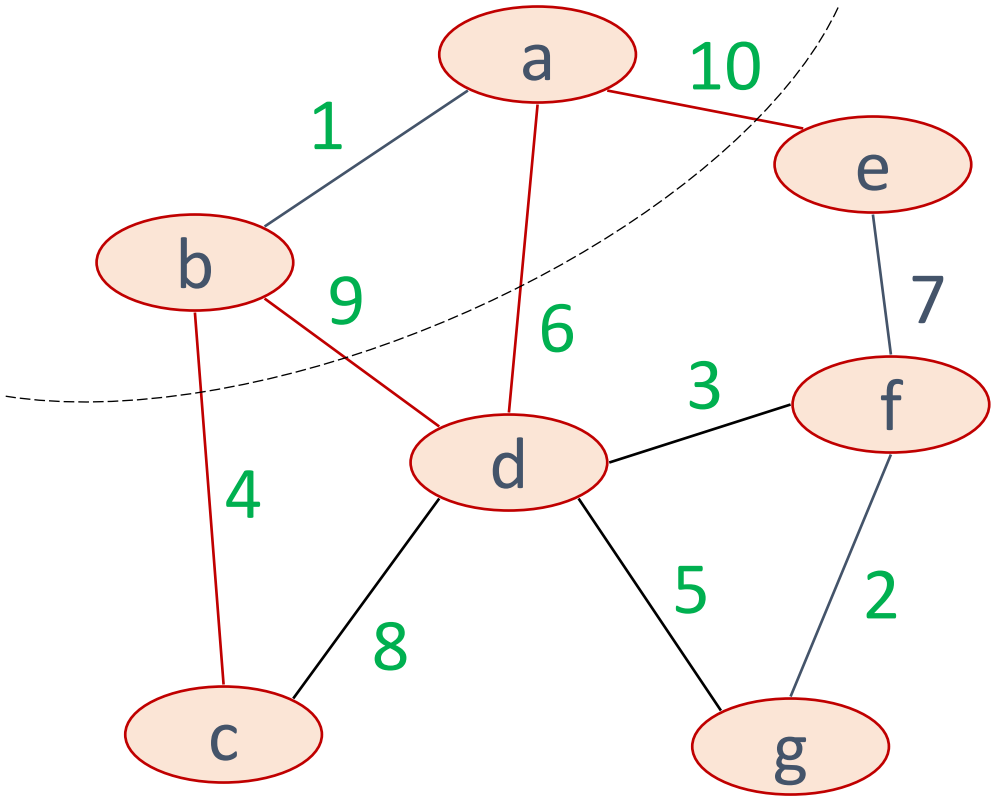
Loop invariant:

Prior to each iteration, A is a subset of some minimum spanning tree.

(u, v) is a safe edge **for** A if $A \cup \{(u, v)\}$ is also a subset of a minimum spanning tree.

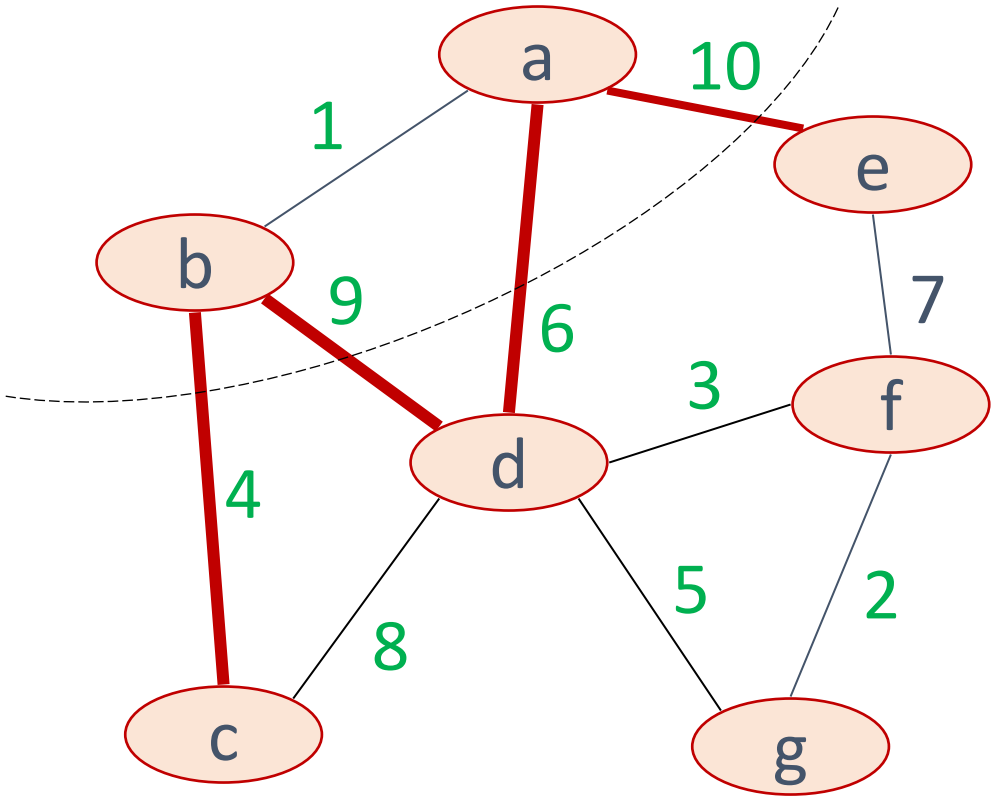
Graph partition

- Cut $(S, V - S)$
- Cross edges
- Light edge



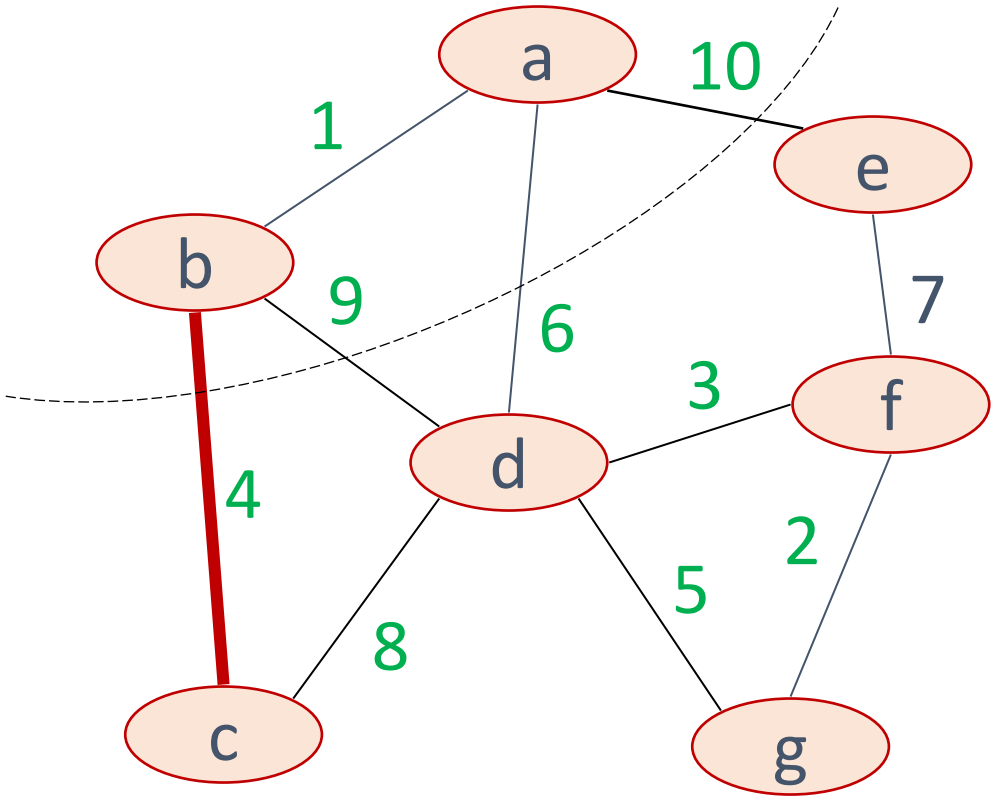
Graph partition

- Cut $(S, V - S)$
- Cross edges
- Light edge

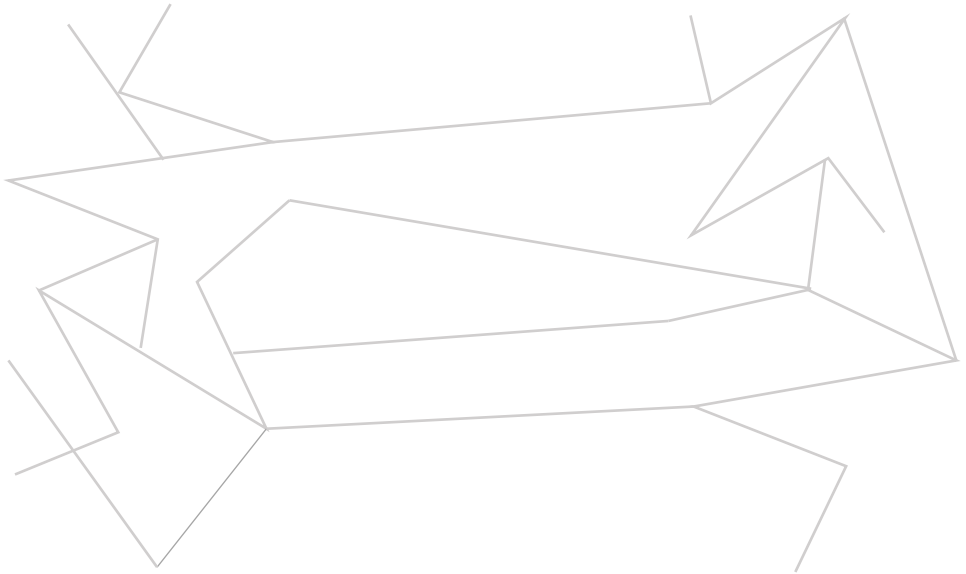


Graph partition

- Cut $(S, V - S)$
- Cross edges
- Light edge

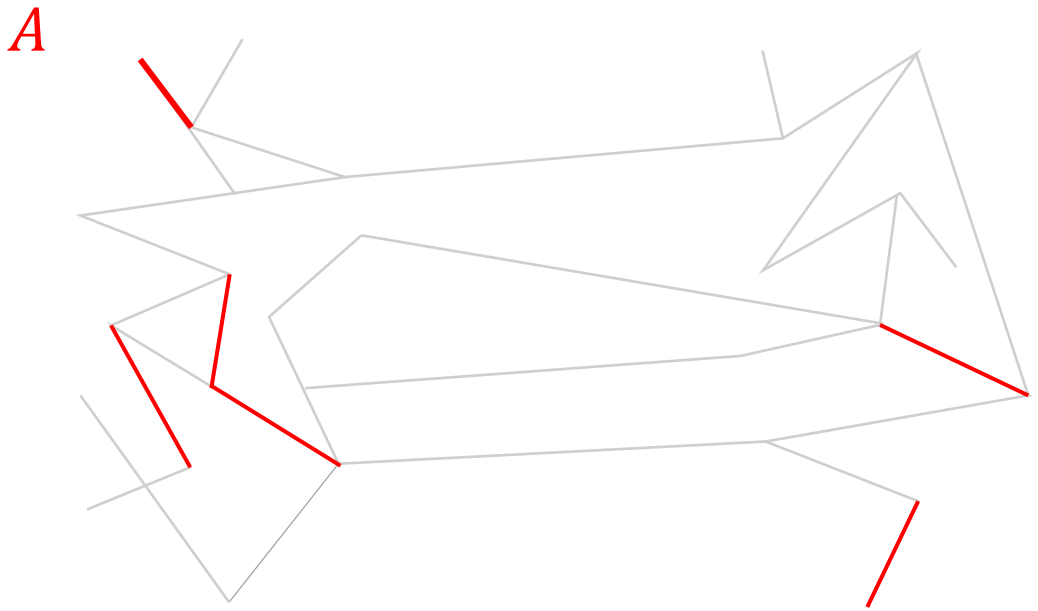


Theorem: safe edges



G a connected, undirected weighted graph.

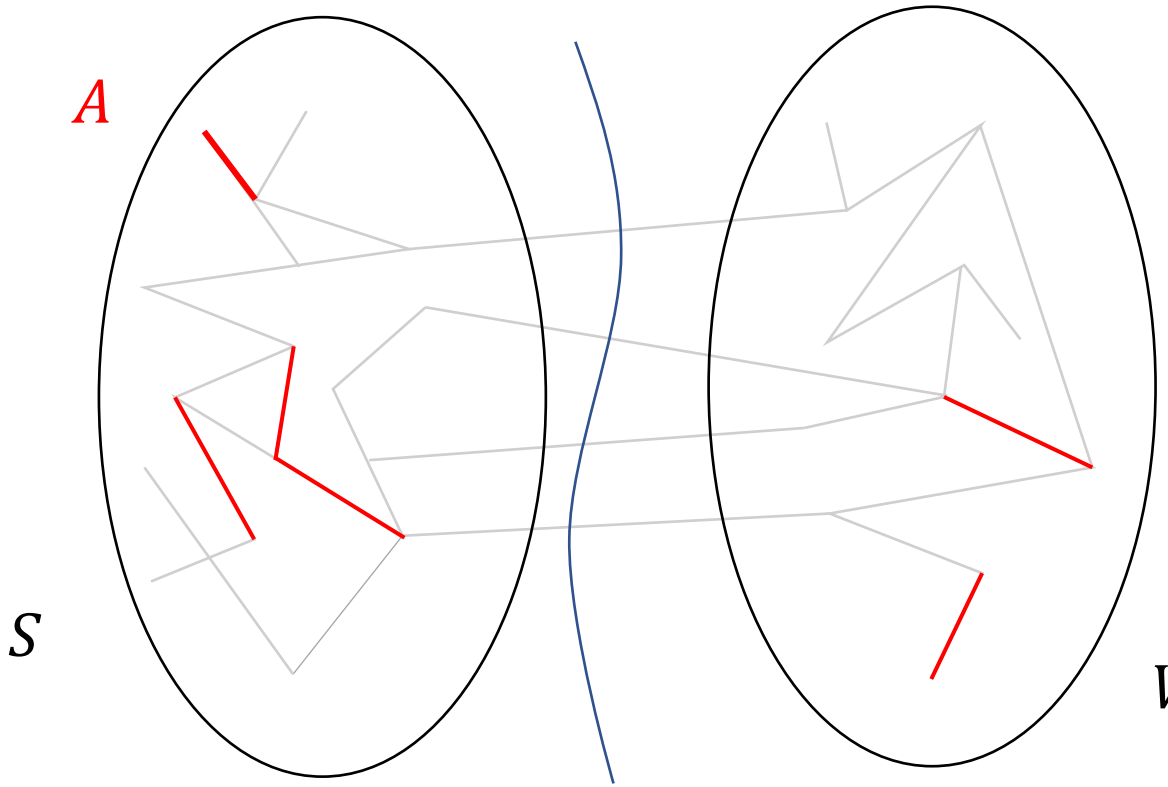
Theorem: safe edges



G a connected, undirected weighted graph.

Let A be a subset of E that is included in some minimum spanning tree.

Theorem: safe edges

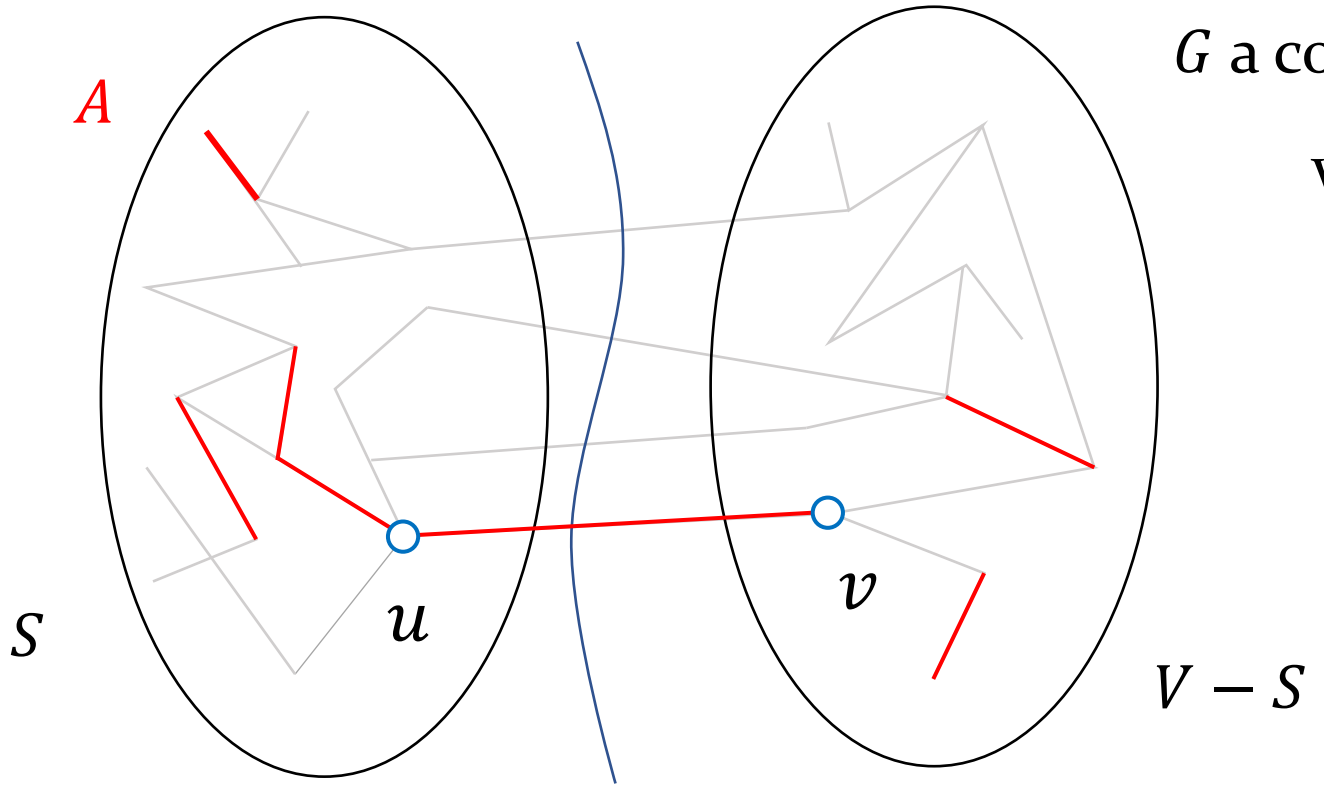


G a connected, undirected weighted graph.

Let A be a subset of E that is included in some minimum spanning tree.

Let $(S, V - S)$ be any cut of G that respects A (does not intersect edges in A)

Theorem: safe edges



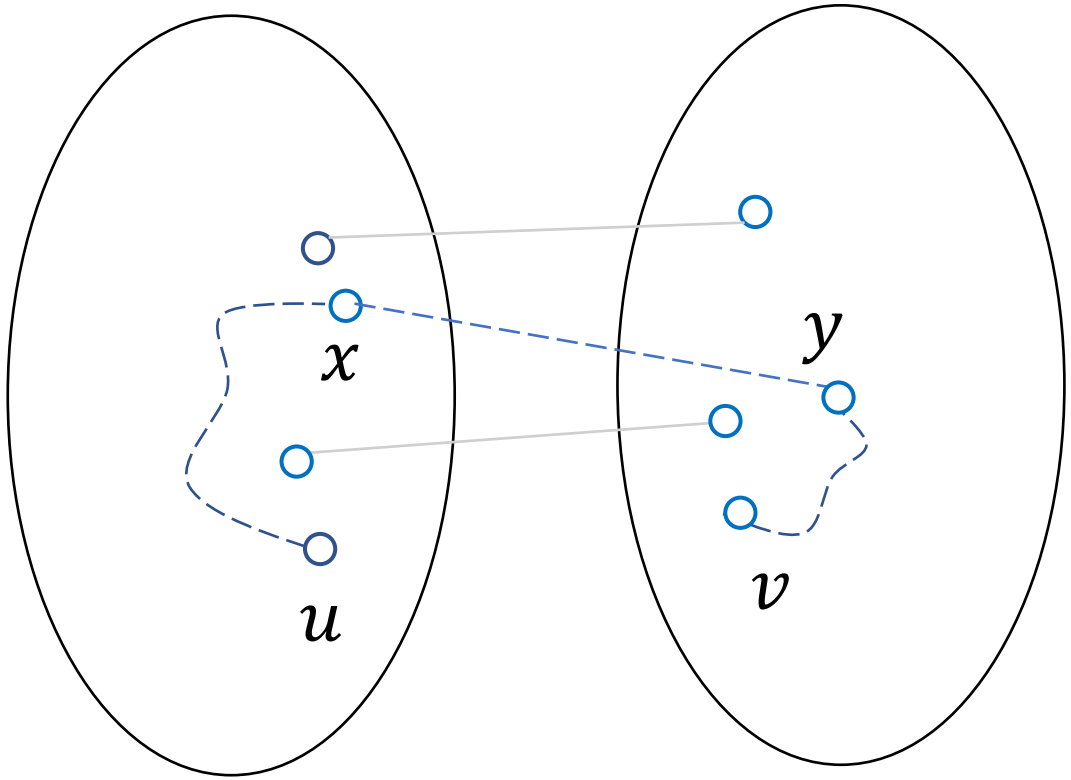
G a connected, undirected weighted graph.

Let A be a subset of E that is included in some minimum spanning tree.

Let $(S, V - S)$ be any cut of G that respects A (does not intersect edges in A)

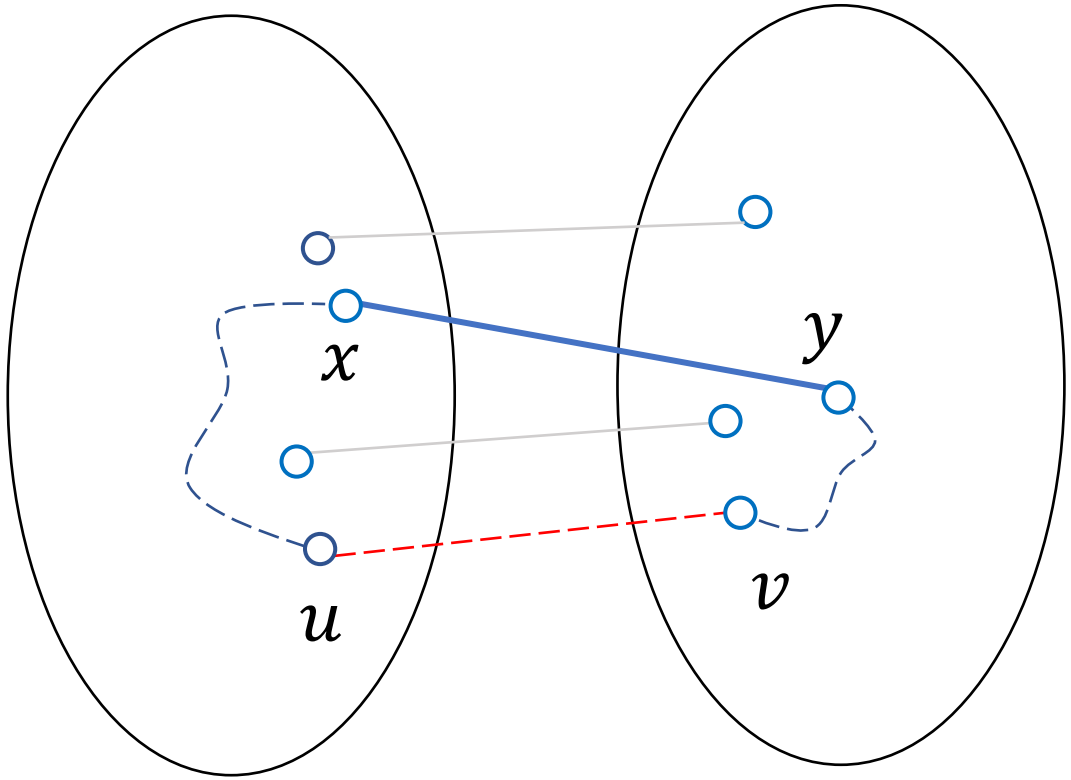
Let (u, v) be a light edge for (S, $V - S$) Then, the claim is that the edge (u, v) is safe for A

Theorem: safe edges



Proof : Let T be a minimum spanning tree that includes A but not (u, v).

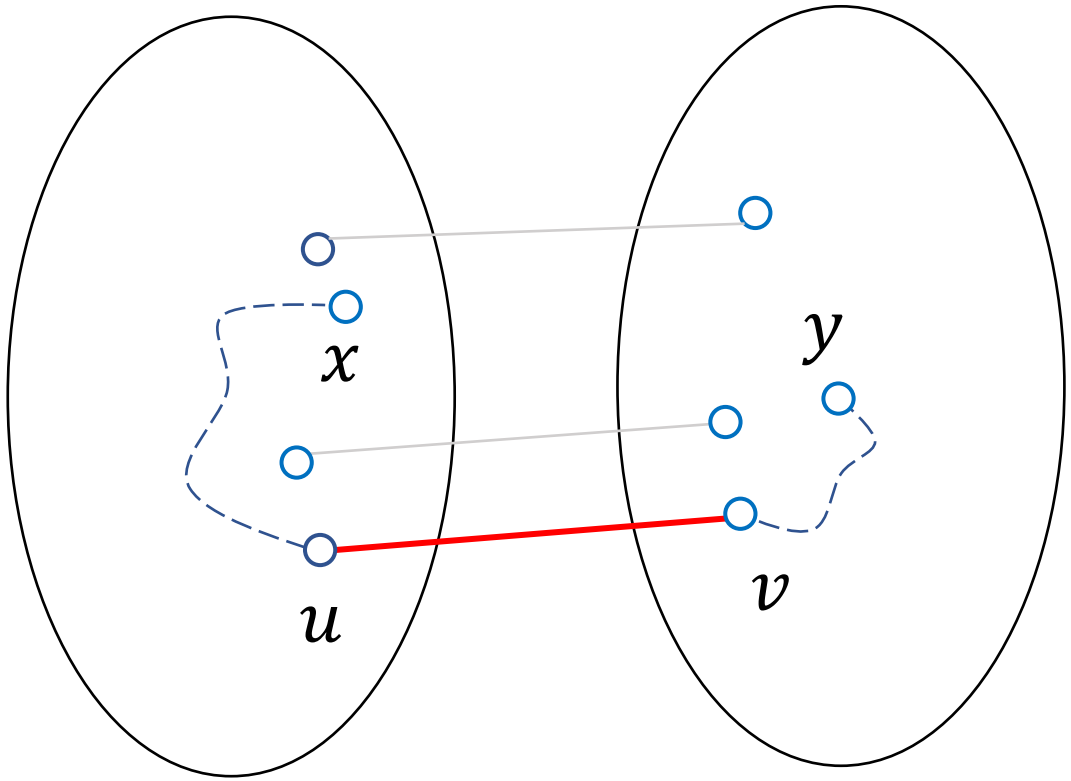
Theorem: safe edges



$w(x, y) \geq w(u, v)$

Proof : Let T be a minimum spanning tree that includes A but not (u, v).

Theorem: safe edges

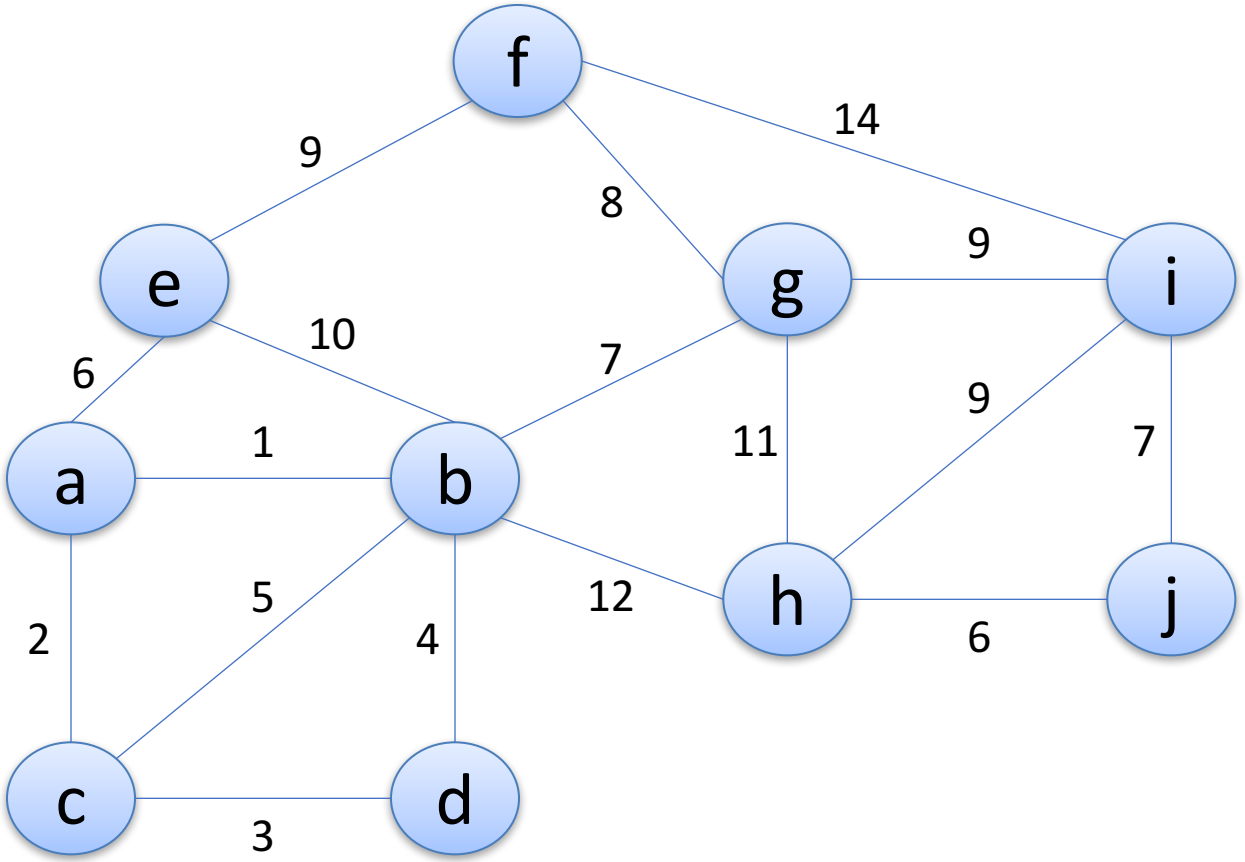


$w(x, y) \geq w(u, v)$

Proof : Let T be a minimum spanning tree that includes A but not (u, v).

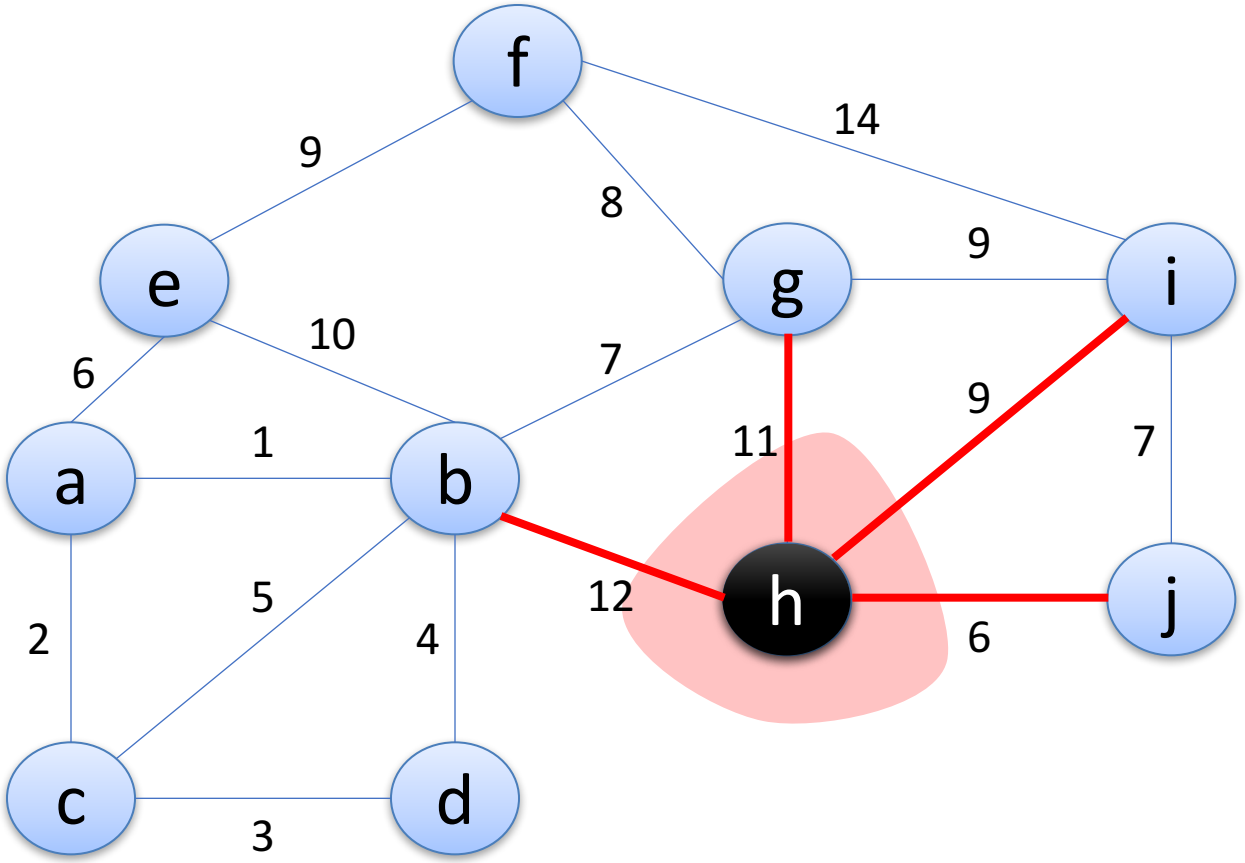
Prim's Algorithm : Example

Start from an arbitrary node h

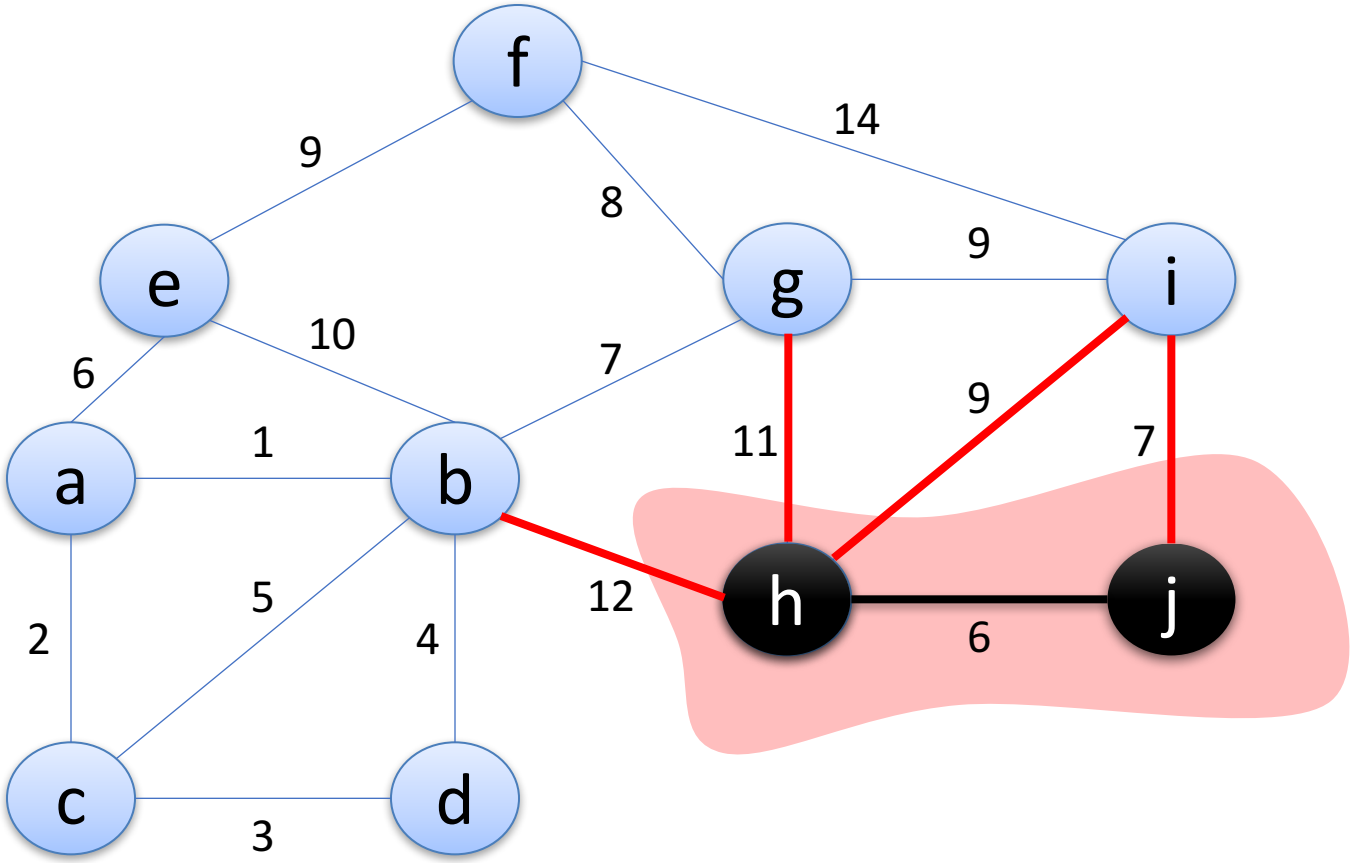


Prim's Algorithm : Example

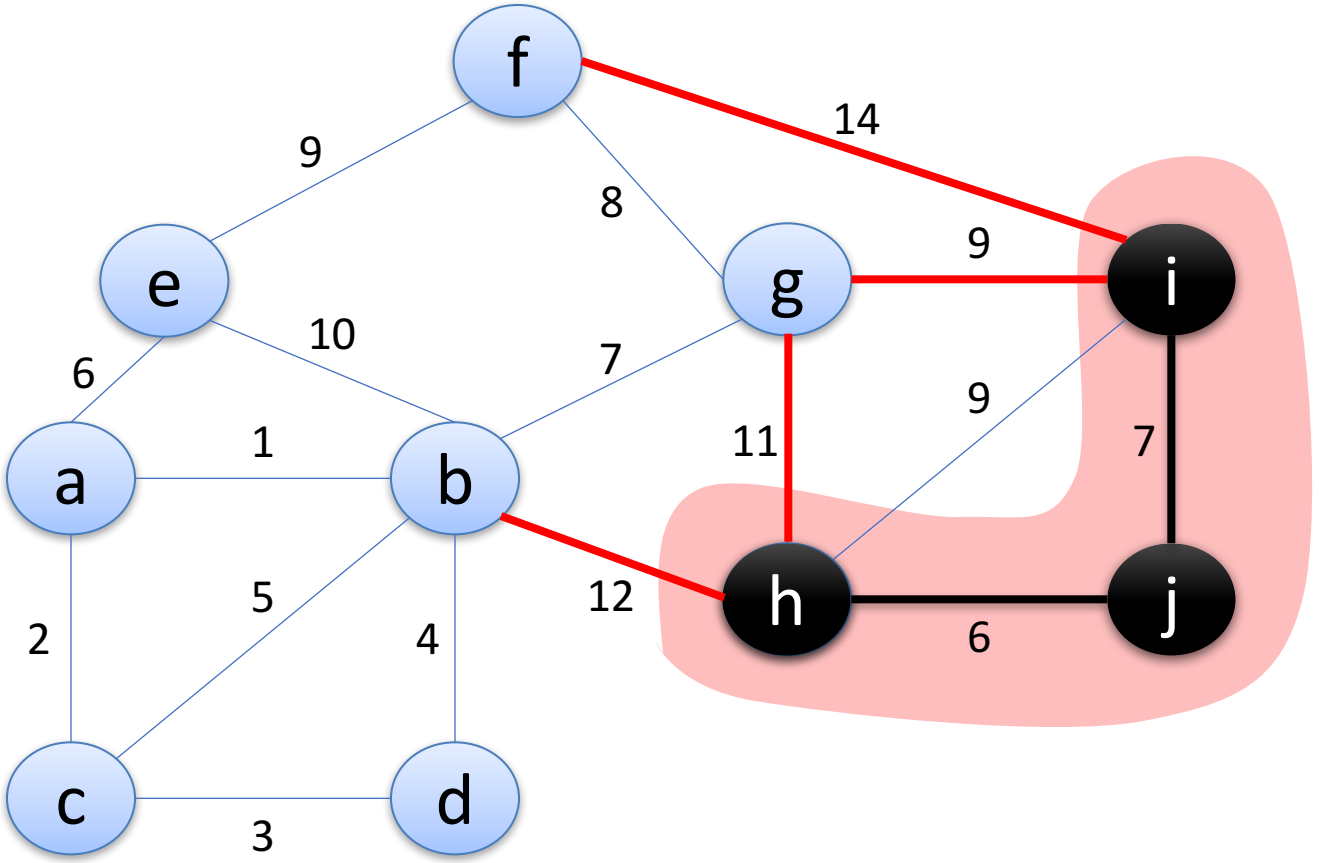
Start from an arbitrary node h



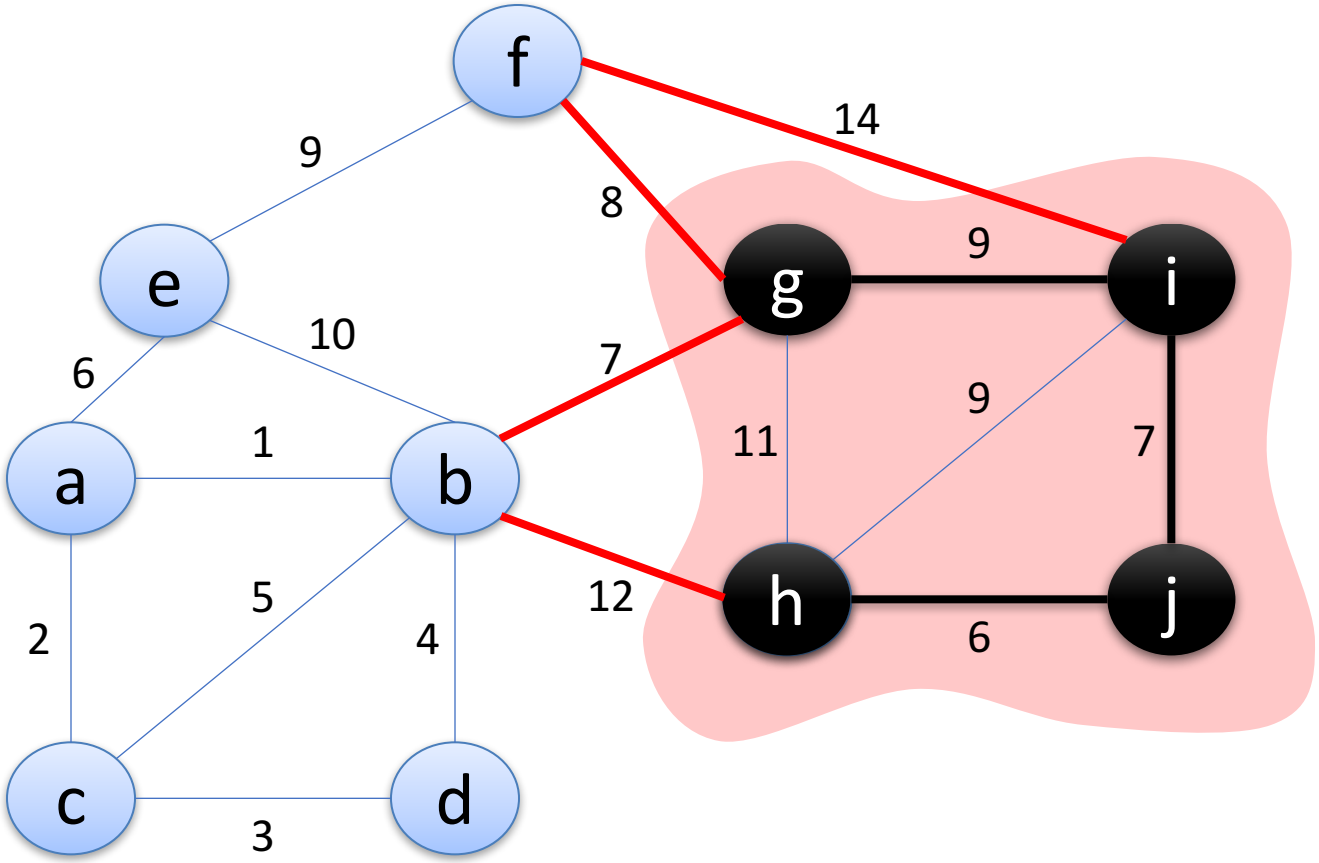
Prim's Algorithm : Example



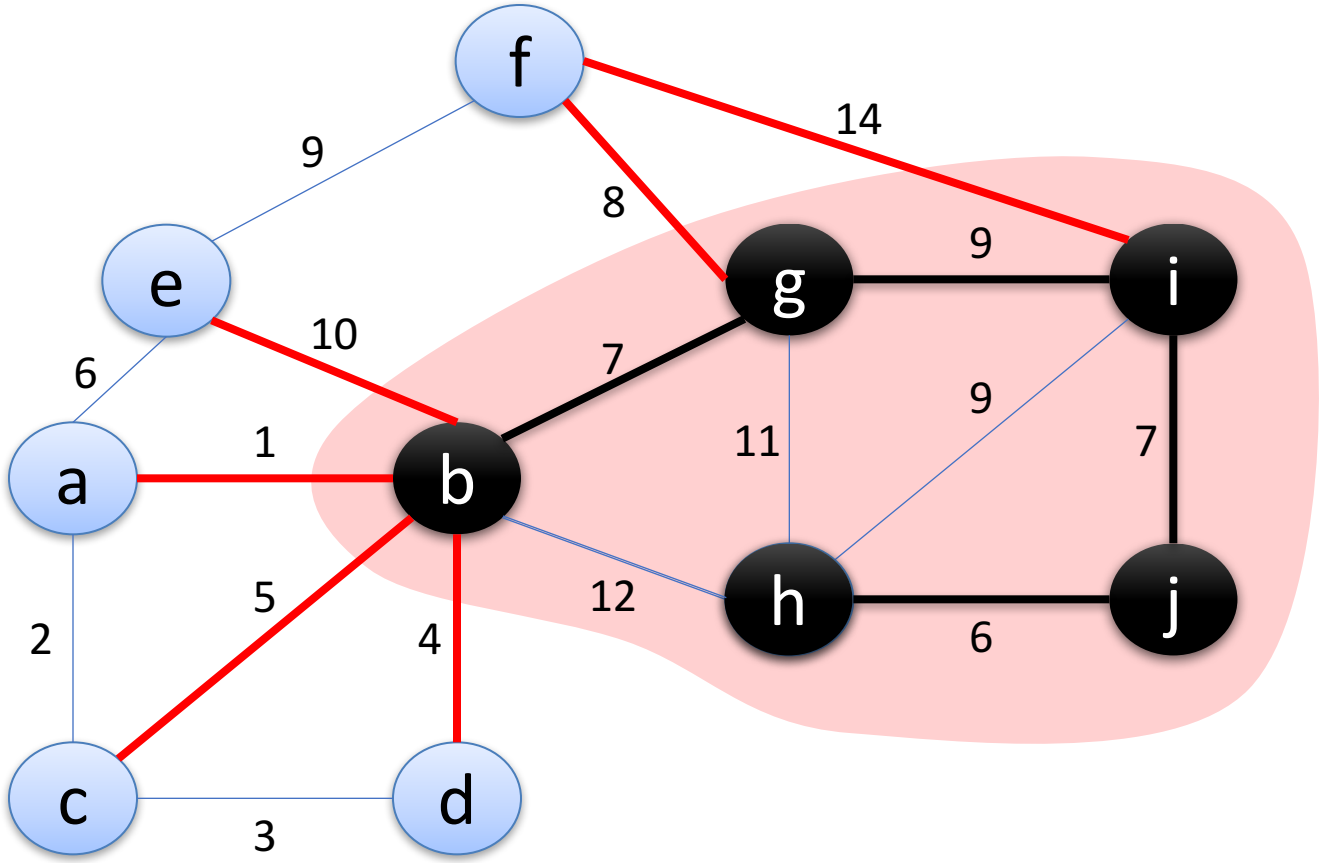
Prim's Algorithm : Example



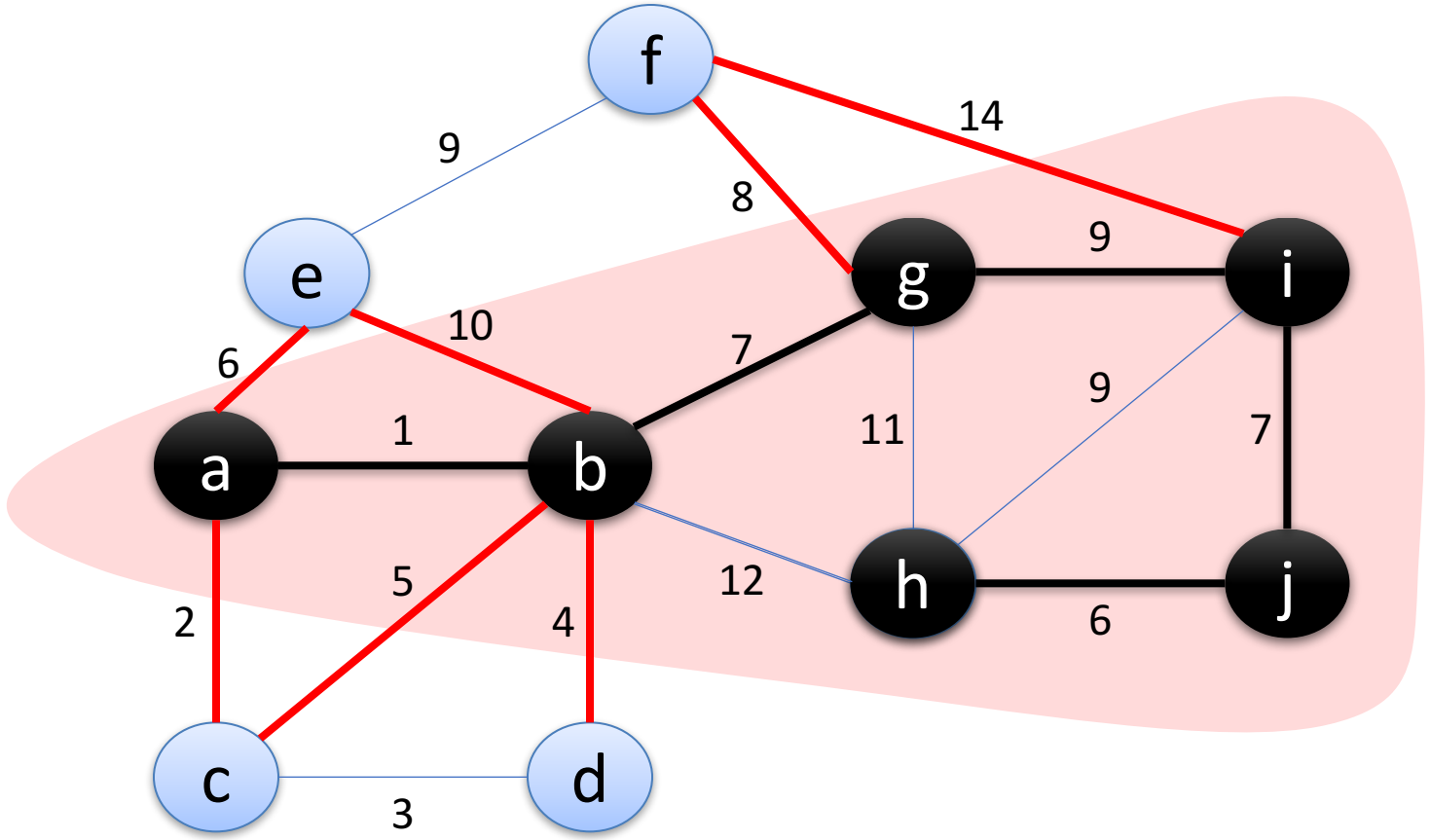
Prim's Algorithm : Example



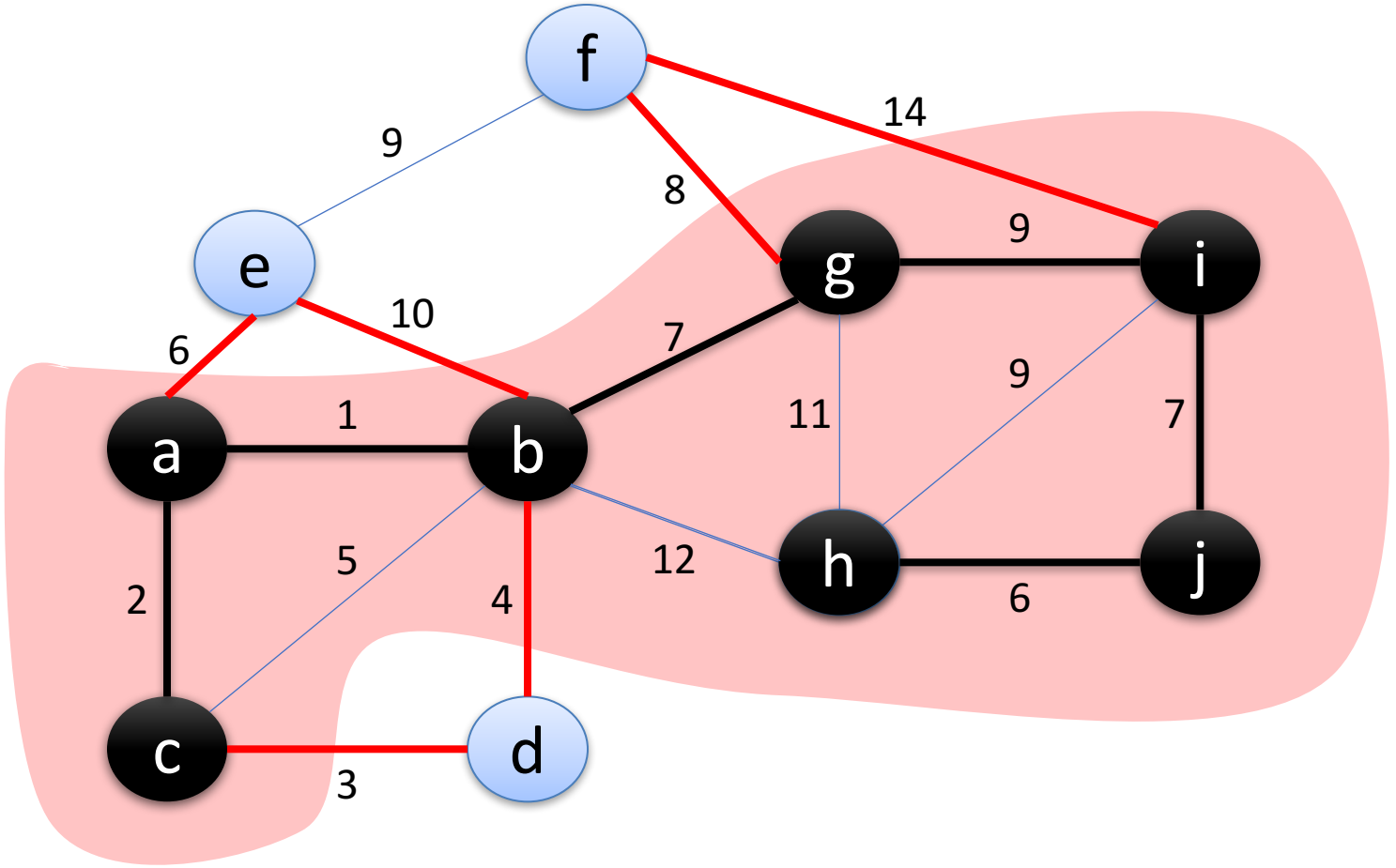
Prim's Algorithm : Example



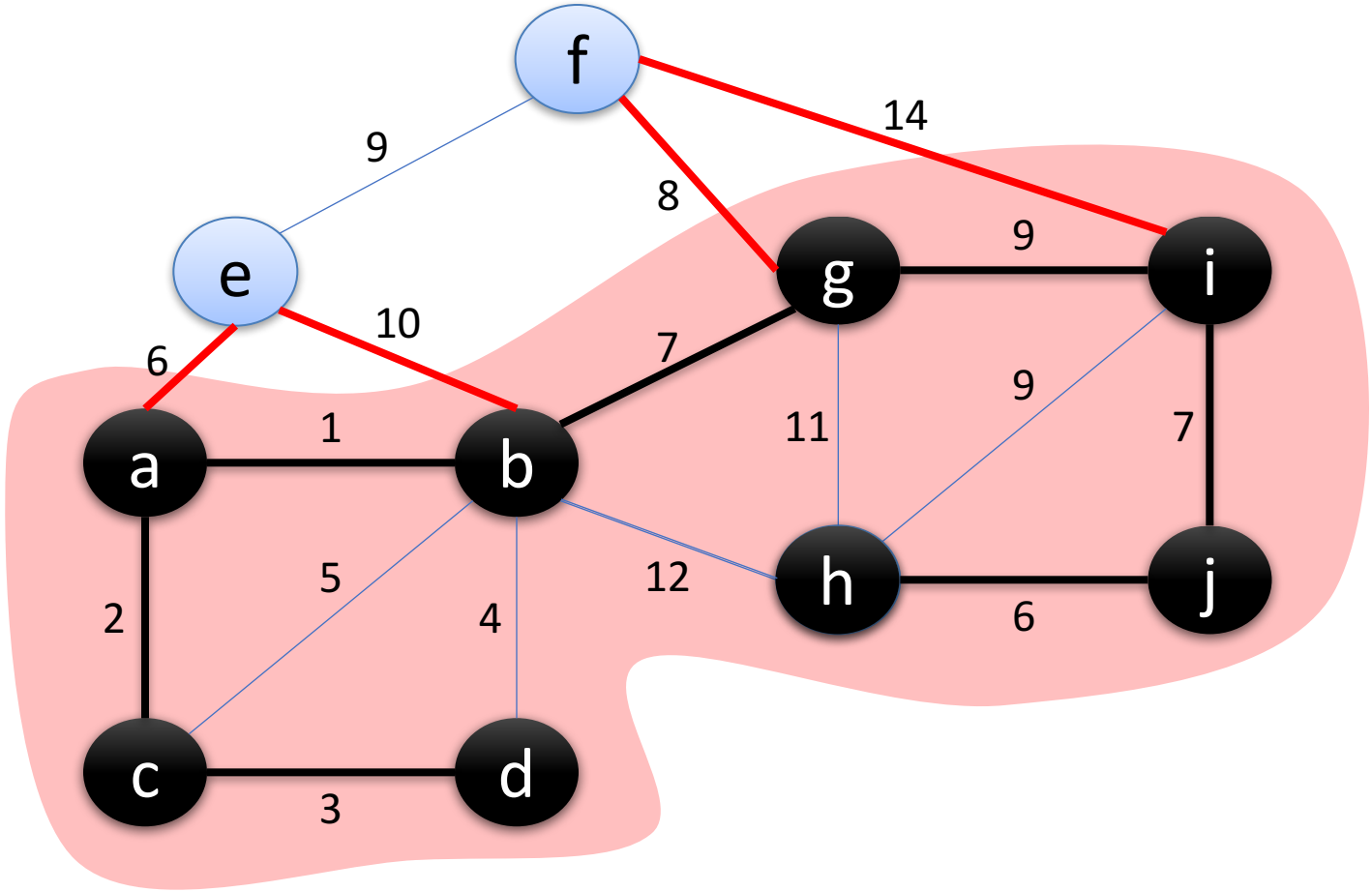
Prim's Algorithm : Example



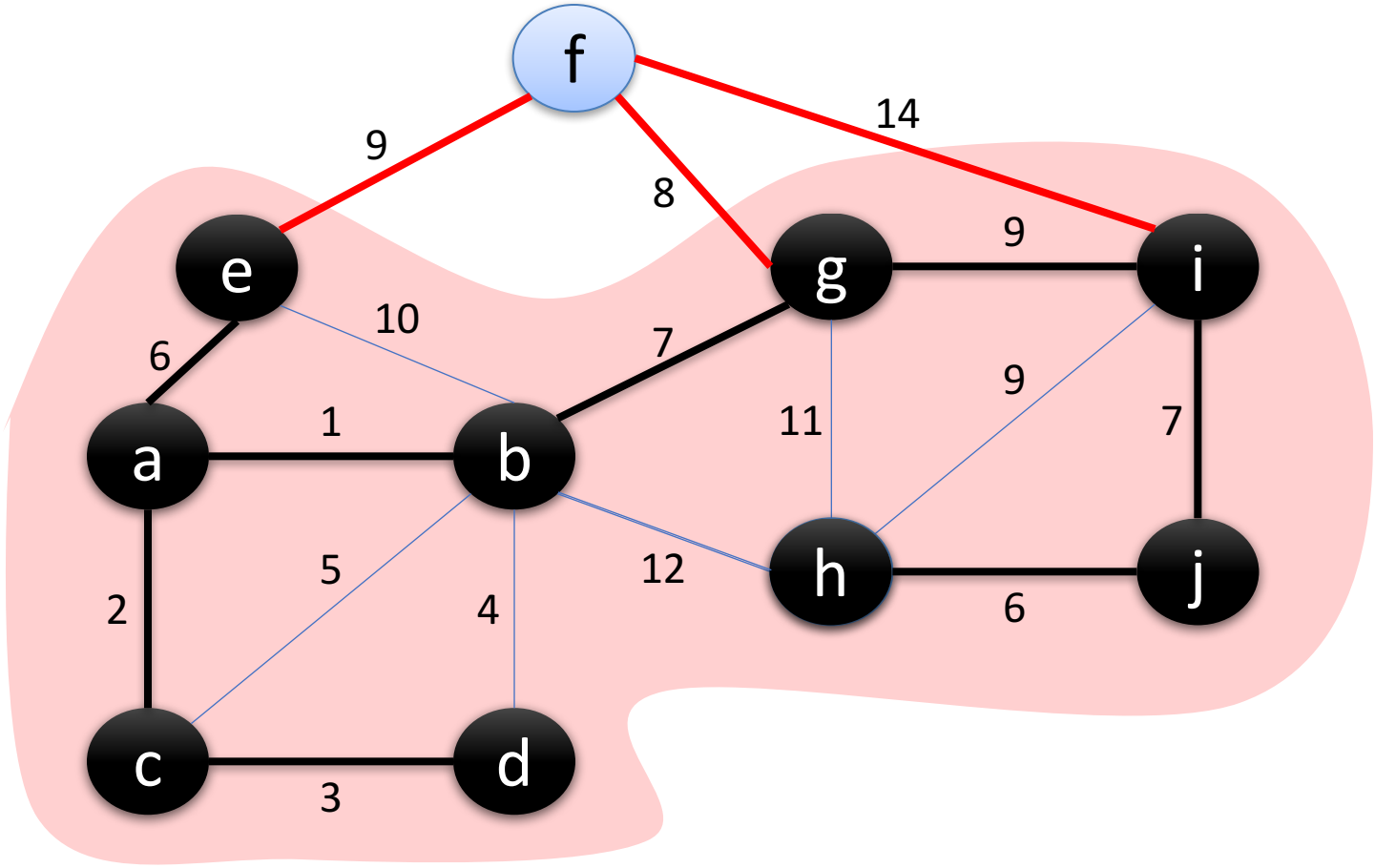
Prim's Algorithm : Example



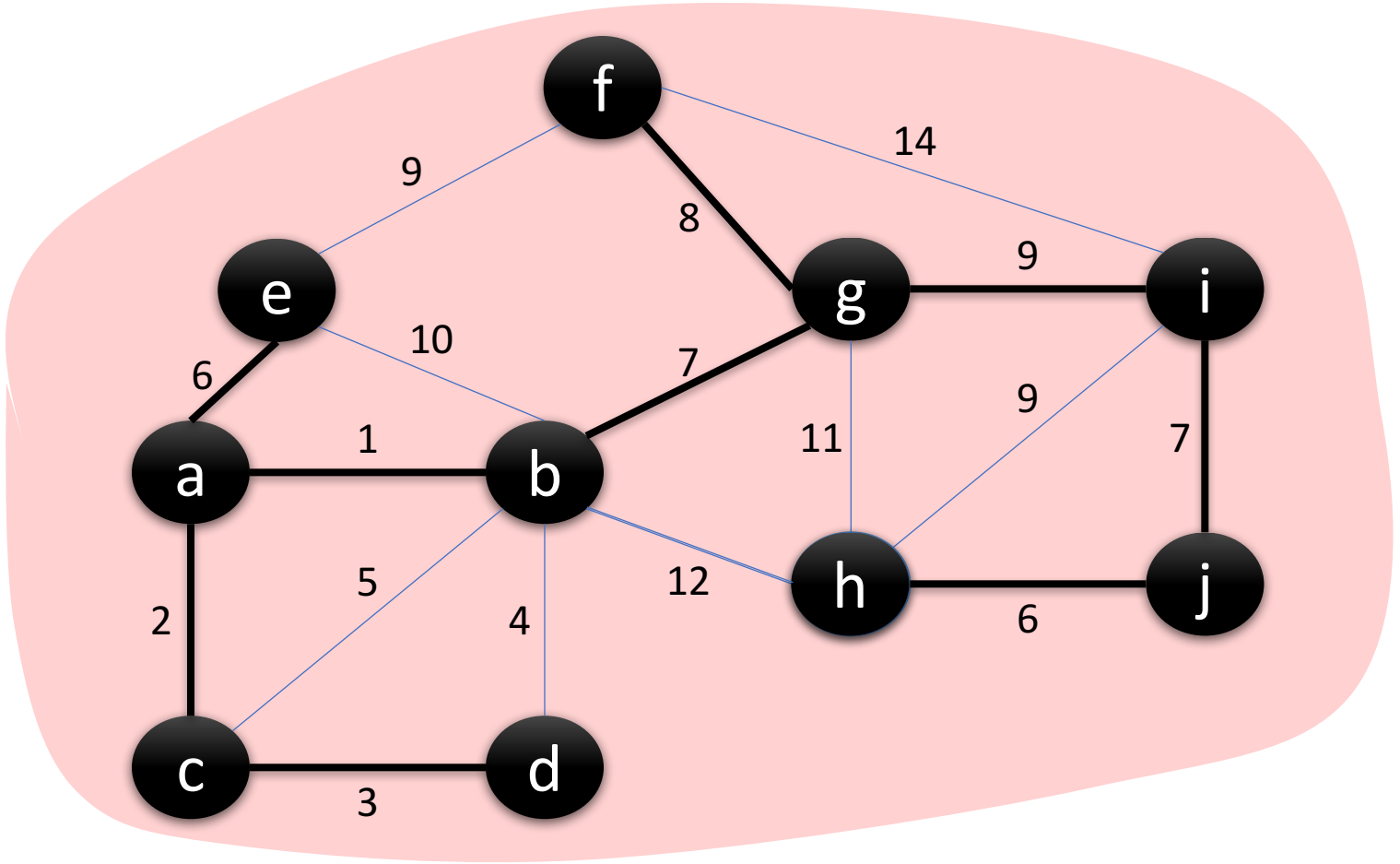
Prim's Algorithm : Example



Prim's Algorithm : Example



Prim's Algorithm : Example



Algorithm

PrimMST (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)
```

We store smallest weight of the edge connecting v to a vertex in the cloud

Select u from the light edges

For each u we update the keys

Running time

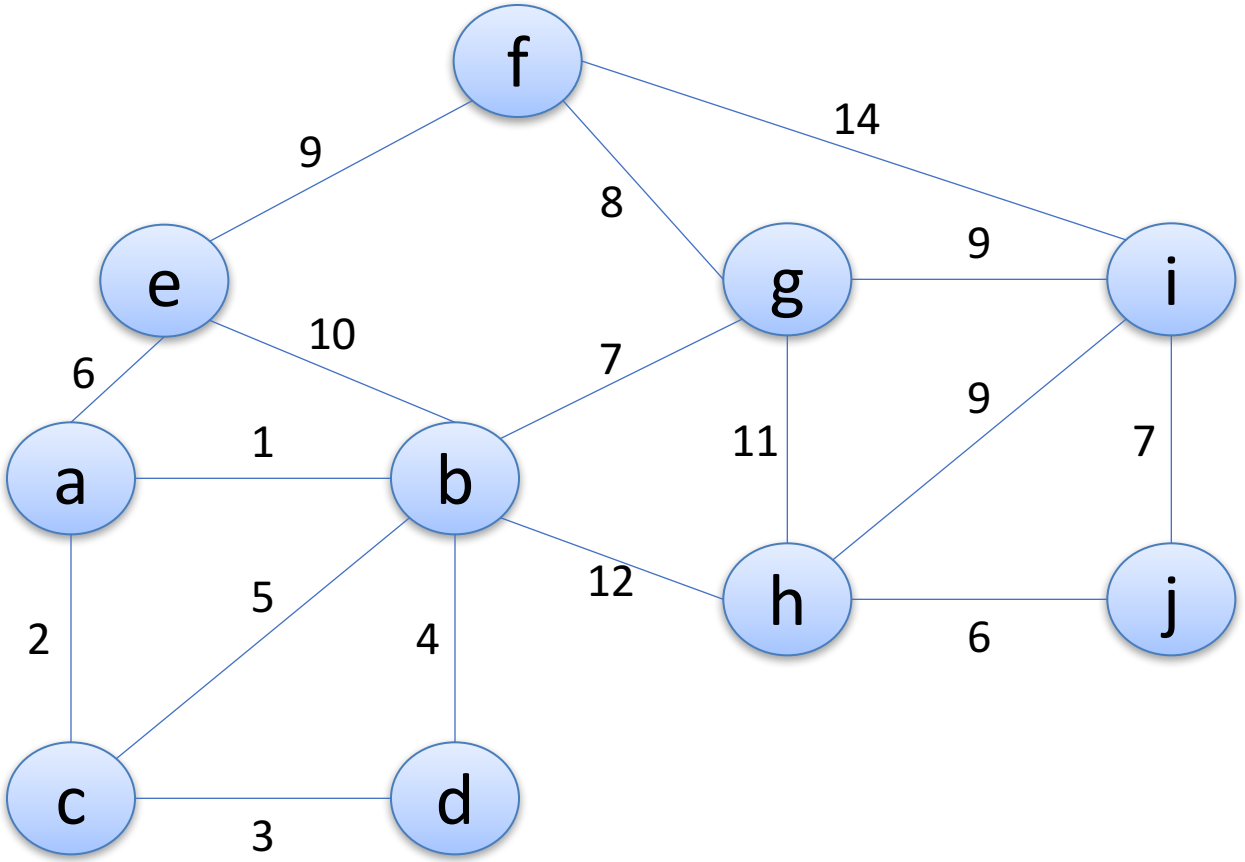
PrimMST (G, w, r)

1. **for** each $u \in G.V$
2. $\quad u.key = \infty$
3. $\quad u.\pi = NIL$
4. $r.key = 0$
5. $Q = G.V$ → $O(V)$
6. **while** $Q \neq \{\}$ → $+$
7. $\quad u = Extract - Min(Q)$ → $O(V \log V)$
8. $\quad$ **for** each $v \in G.Adj[u]$ → $+ O(E)$
9. $\quad\quad$ if $v \in Q$ and $w(u,v) < v.key$ → $*$ *(membership In O(1) by storing a bit for each vertex +*
10. $\quad\quad\quad v.\pi = u$
11. $\quad\quad\quad v.key = w(u,v)$ → $O(\log V)$ *updating the key (decrease key))*

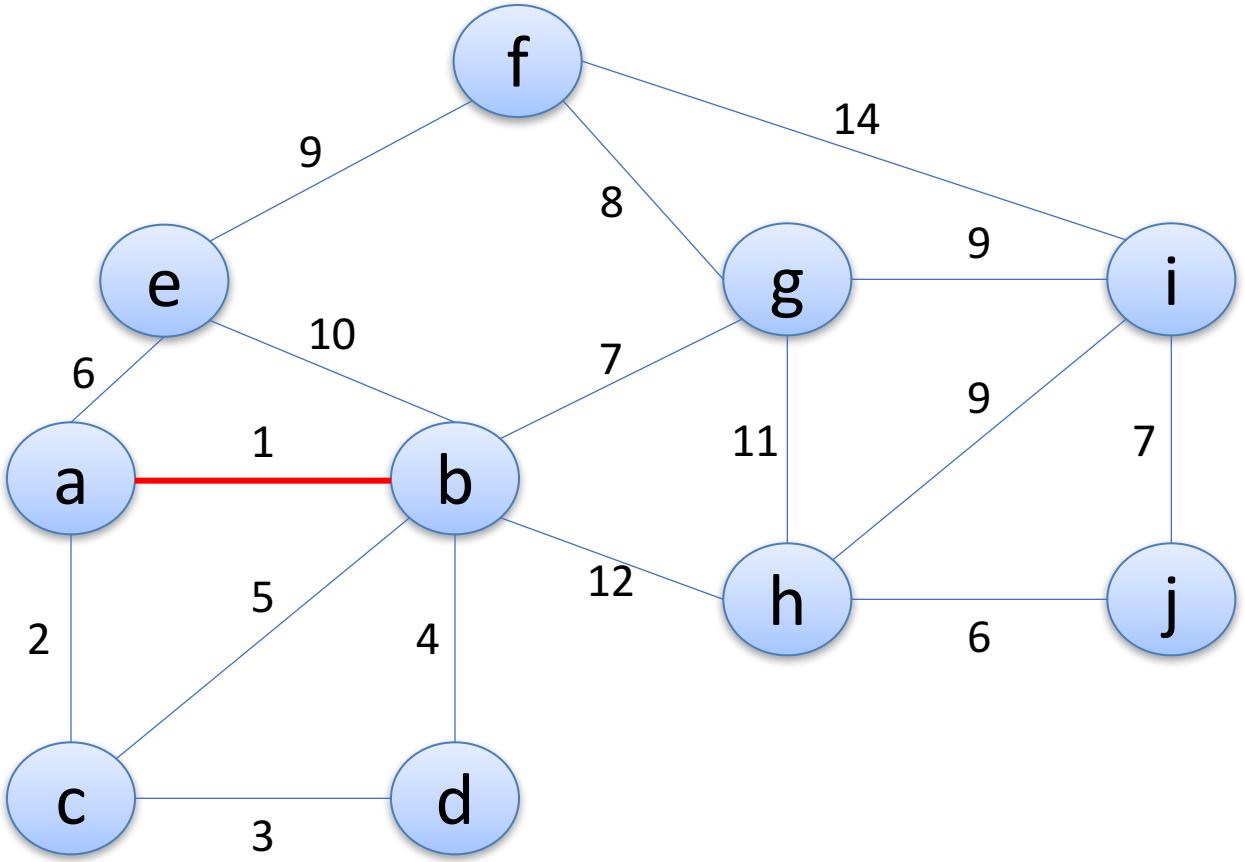
$= O(V \log V + E \log V) = O(E \log V)$

Kruskal's Algorithm

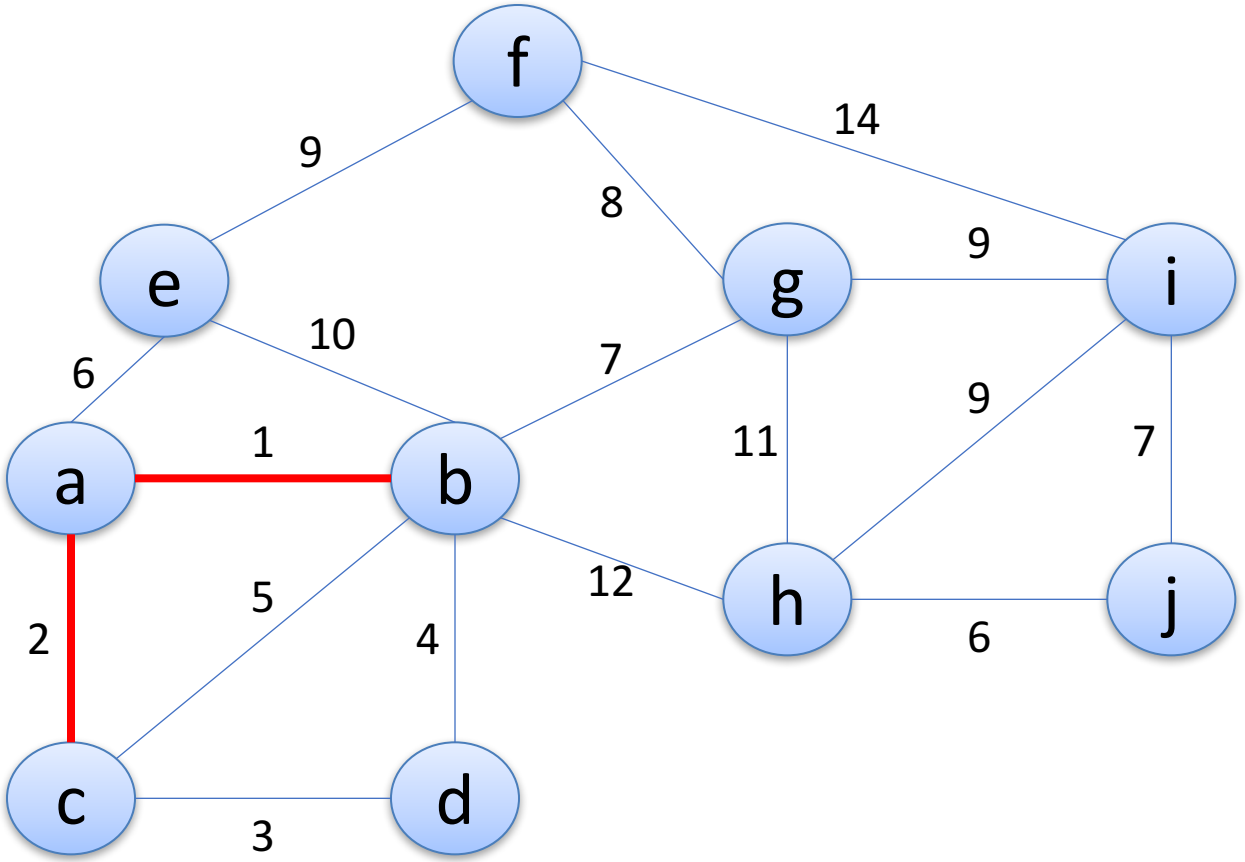
Pick the smallest edge if it does not make a cycle



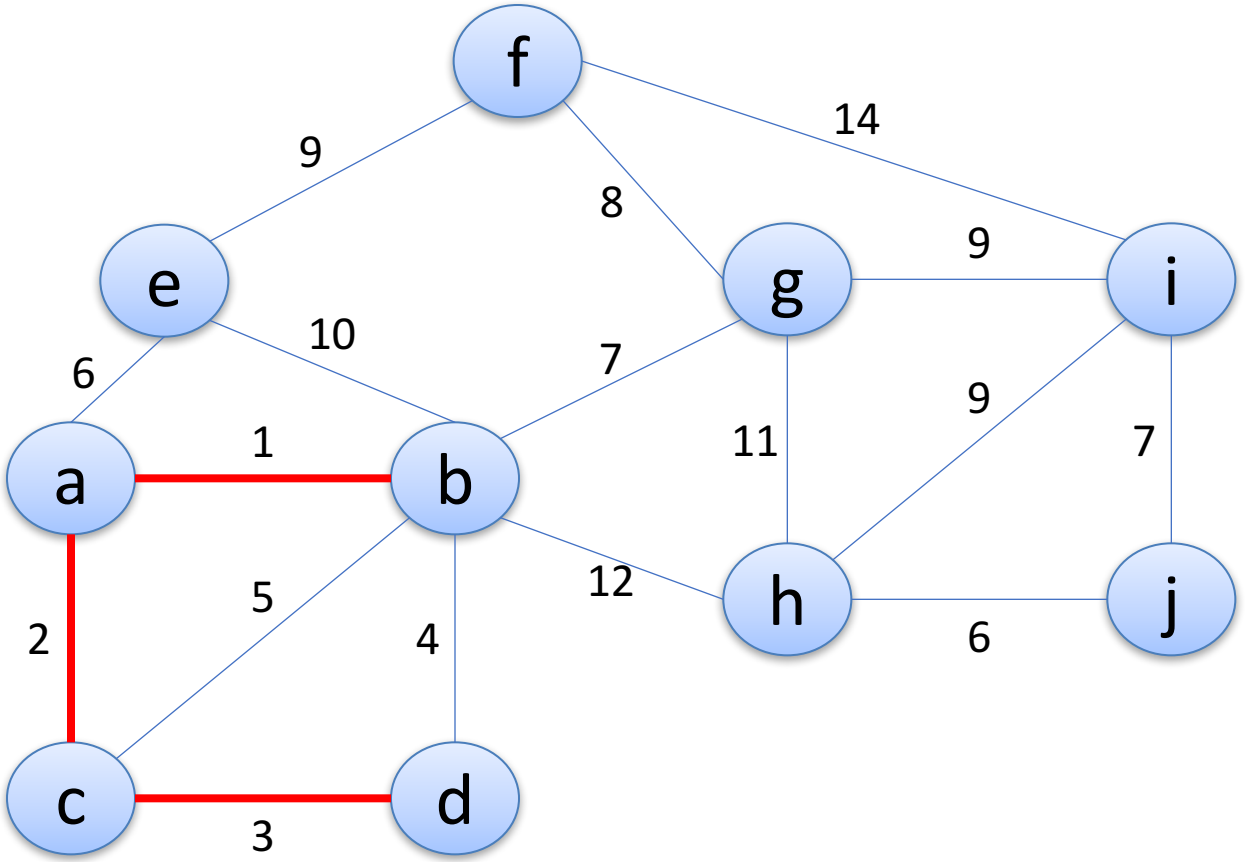
Kruskal's Algorithm



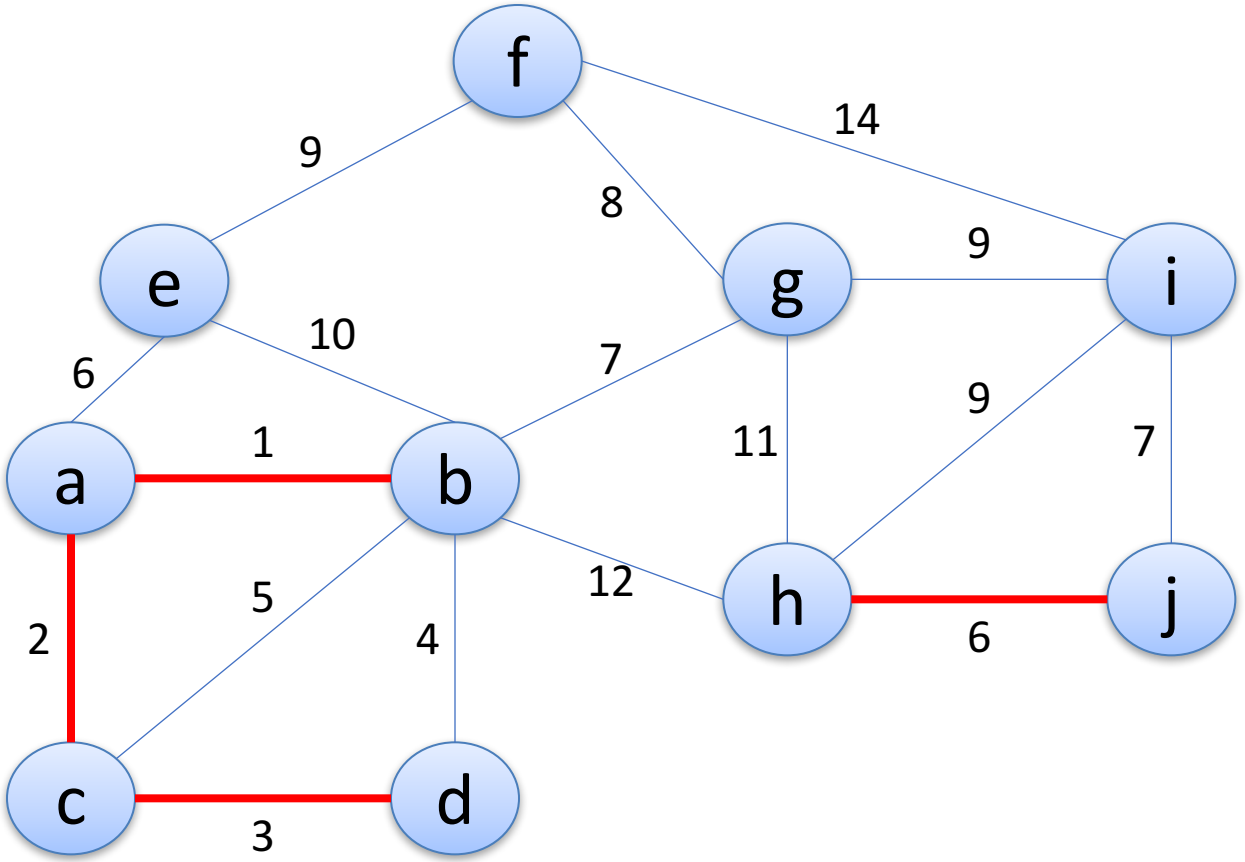
Kruskal's Algorithm



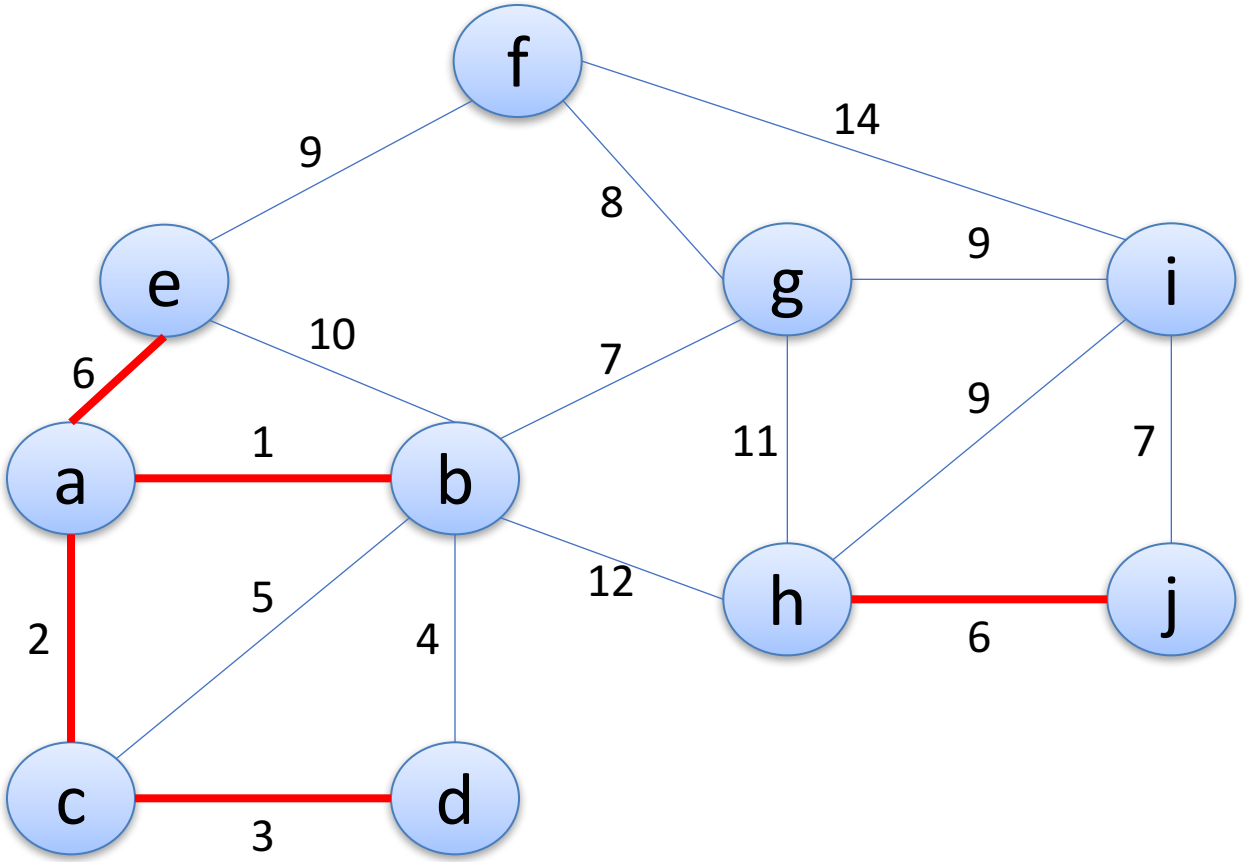
Kruskal's Algorithm



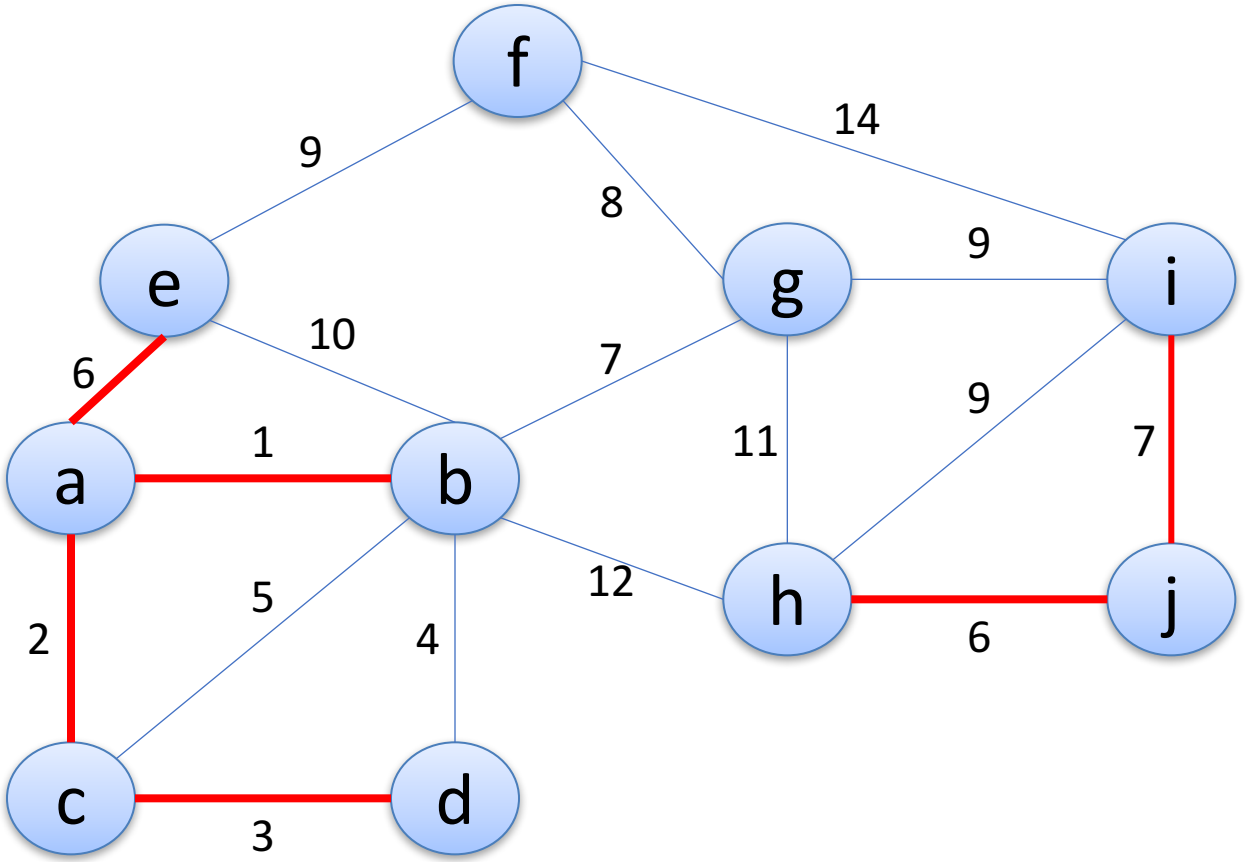
Kruskal's Algorithm



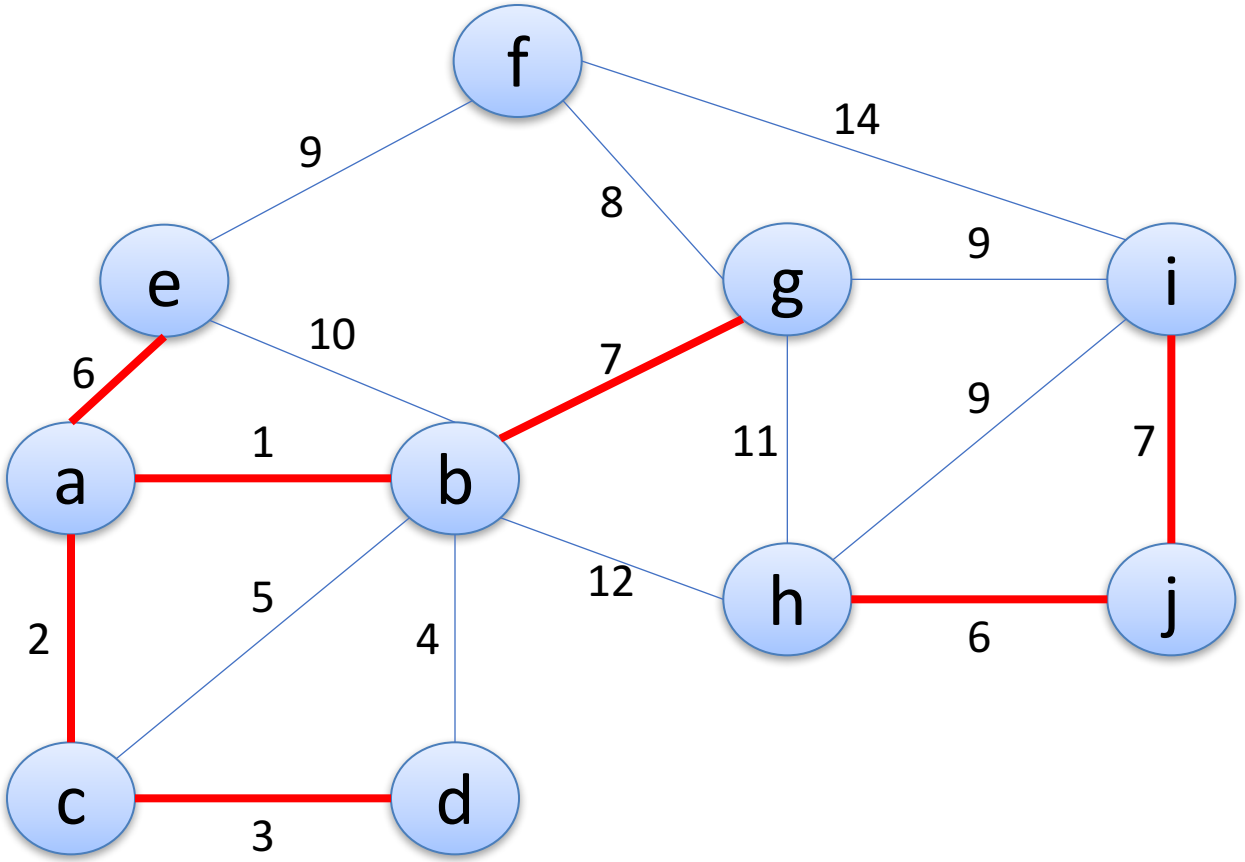
Kruskal's Algorithm



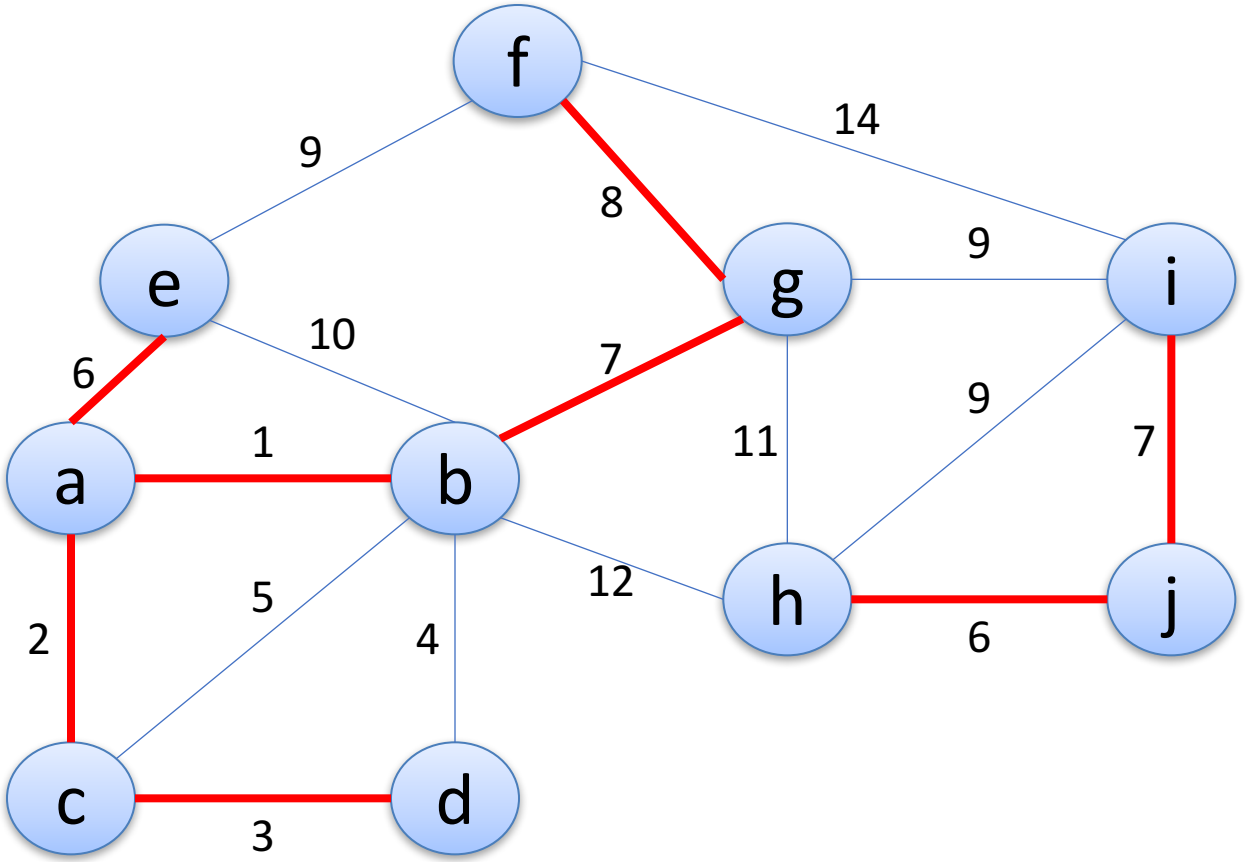
Kruskal's Algorithm



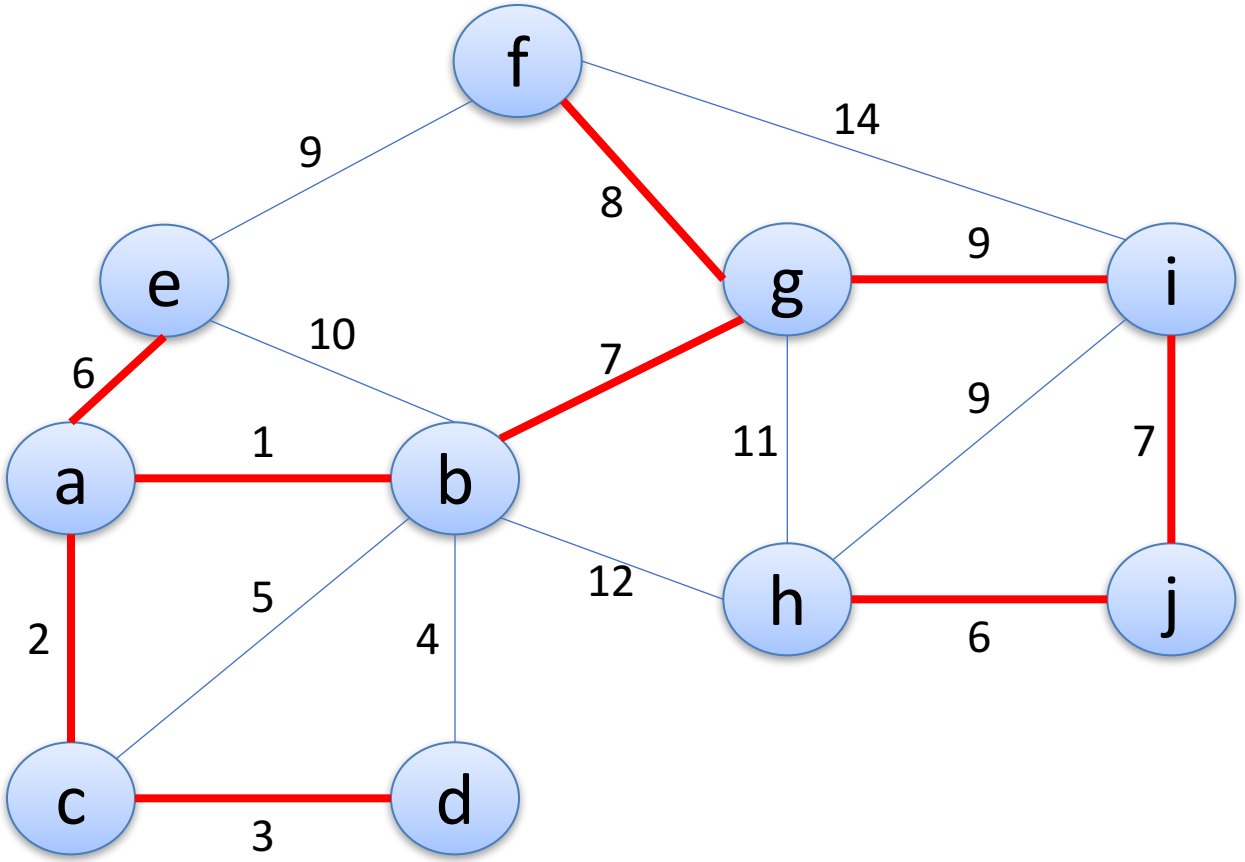
Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm

KRUSKAL-MST(G, w):

$T \leftarrow \{\}$

sort edges so $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_E)$

for each $v \in V$:

    Make-Set(v)

for $i \leftarrow 1$ to E # let $(u_i, v_i) = e_i$

  if Find-Set(u_i) $\neq$ Find-Set(v_i):

    Union(u_i,v_i)

    $T \leftarrow T \cup \{e_i\}$

return T

Running time

KRUSKAL-MST(G, w):

$T \leftarrow \{\}$

sort edges so $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_E)$

for each $v \in V$:

    Make-Set(v)

for $i \leftarrow 1$ to E # let $(u_i, v_i) = e_i$

    if Find-Set(u_i) ≠ Find-Set(v_i):

        Union(u_i,v_i)

        $T \leftarrow T \cup \{e_i\}$

return T

$O(E \log E)$ for sorting weights,

V make-set and
E Find-set and Union

$O\big((V+E)\log^* V\big)$
$= O\big((V+E)\log V\big) = O(E \log V)$

In total $O(E \log E) = O(E \log V)$
Since $E = O(V^2)$

Questions

- Will Both algorithms always give the solutions with the same length?

yes

- Will both algorithms always give the same edges as their output?

no

- How do you modify the algorithms for disconnected graphs?

DFS first, krustal algorithsm?

Questions?