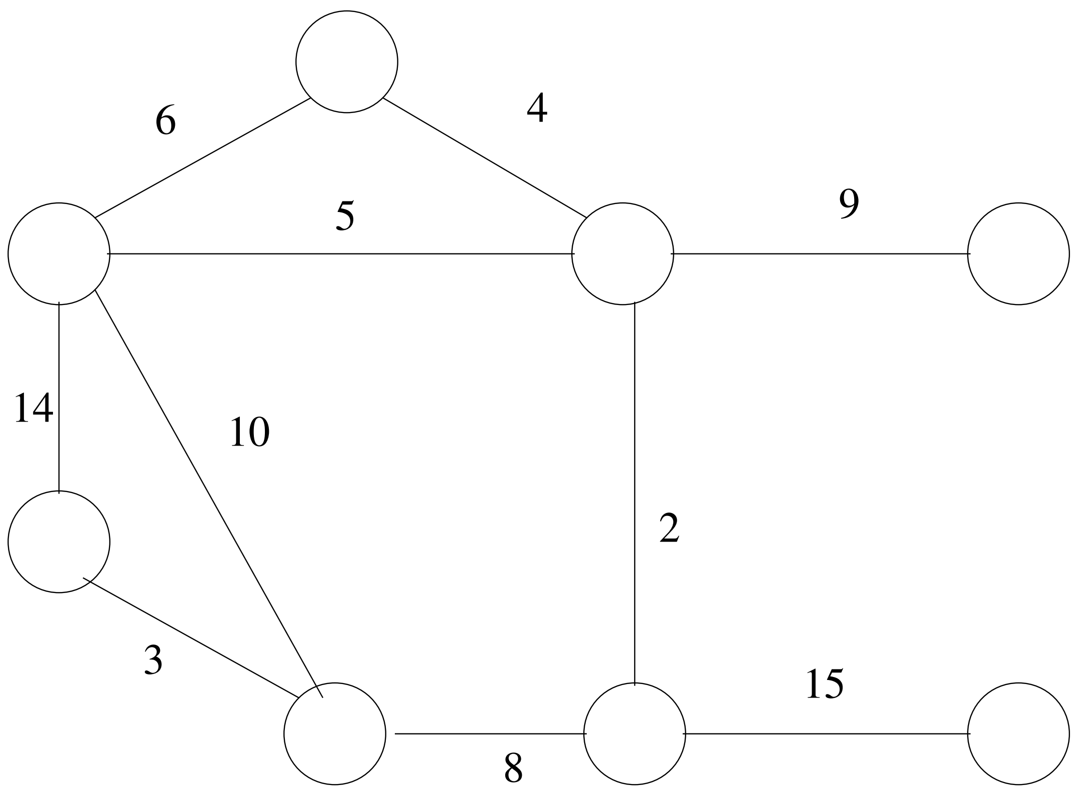
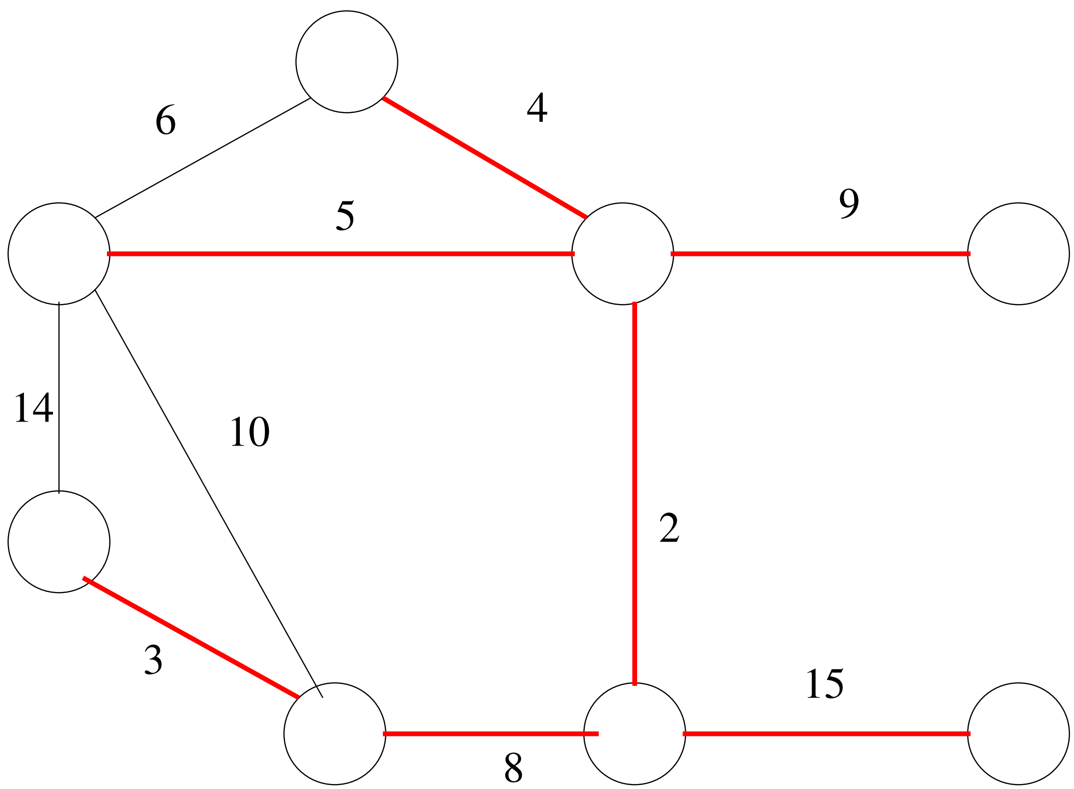
Minimum Spanning Trees

- $G = (V, E)$ is an undirected graph with non-negative edge weights $w : E \rightarrow Z^+$
- We assume wlog that edge weights are distinct
- A spanning tree is a tree with $V - 1$ edges, i.e. a tree that connects all the vertices.
- The total cost (weight) of a spanning tree T is defined as $\Sigma_{e \in T} w(e)$
- A minimum spanning tree is a tree of minimum total weight.



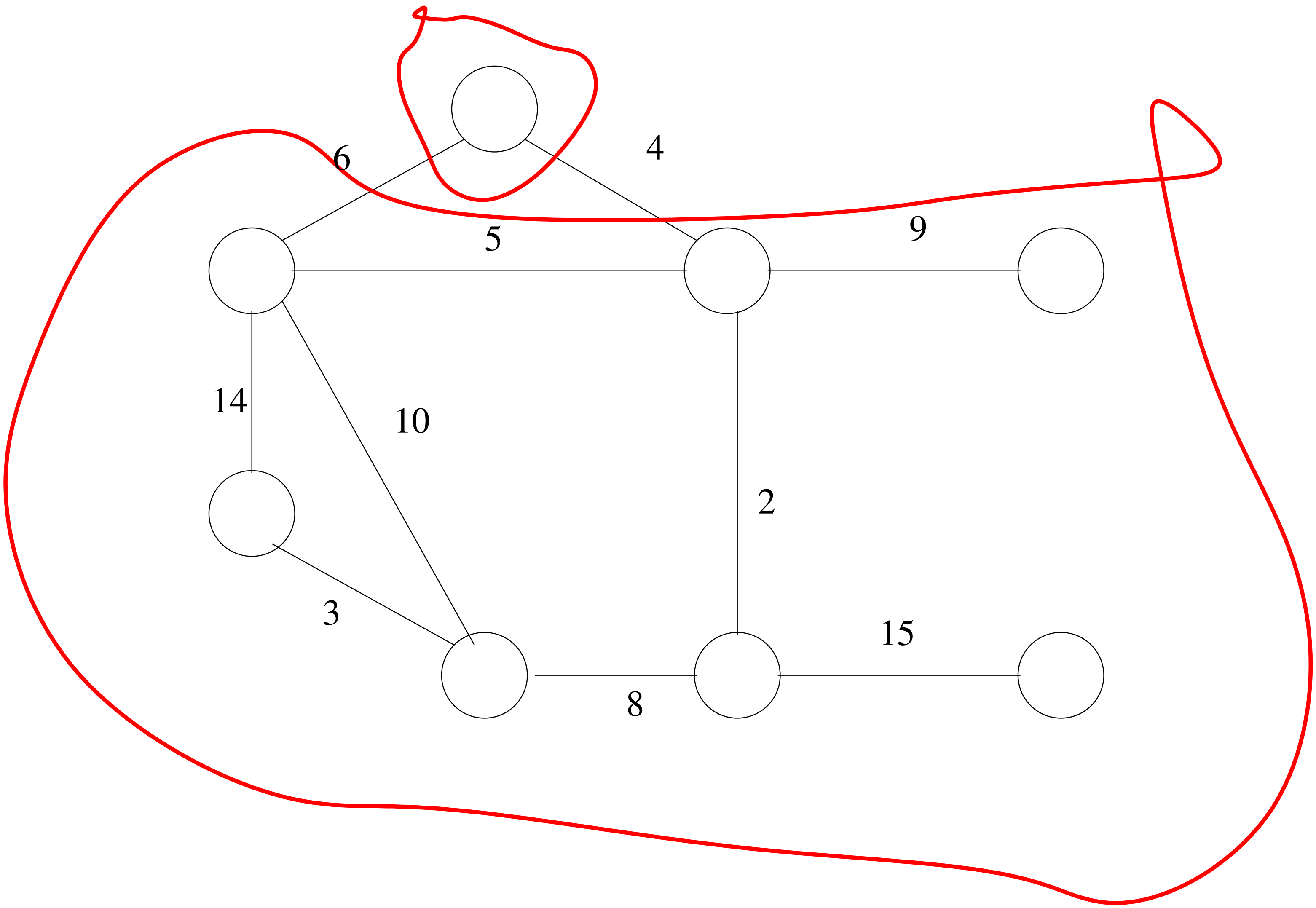
Minimum Spanning Trees

- $G = (V, E)$ is an undirected graph with non-negative edge weights $w : E \rightarrow Z^+$
- We assume wlog that edge weights are distinct
- A **spanning tree** is a tree with $V - 1$ edges, i.e. a tree that connects all the vertices.
- The total cost (weight) of a spanning tree T is defined as $\Sigma_{e \in T} w(e)$
- A **minimum spanning tree** is a tree of minimum total weight.



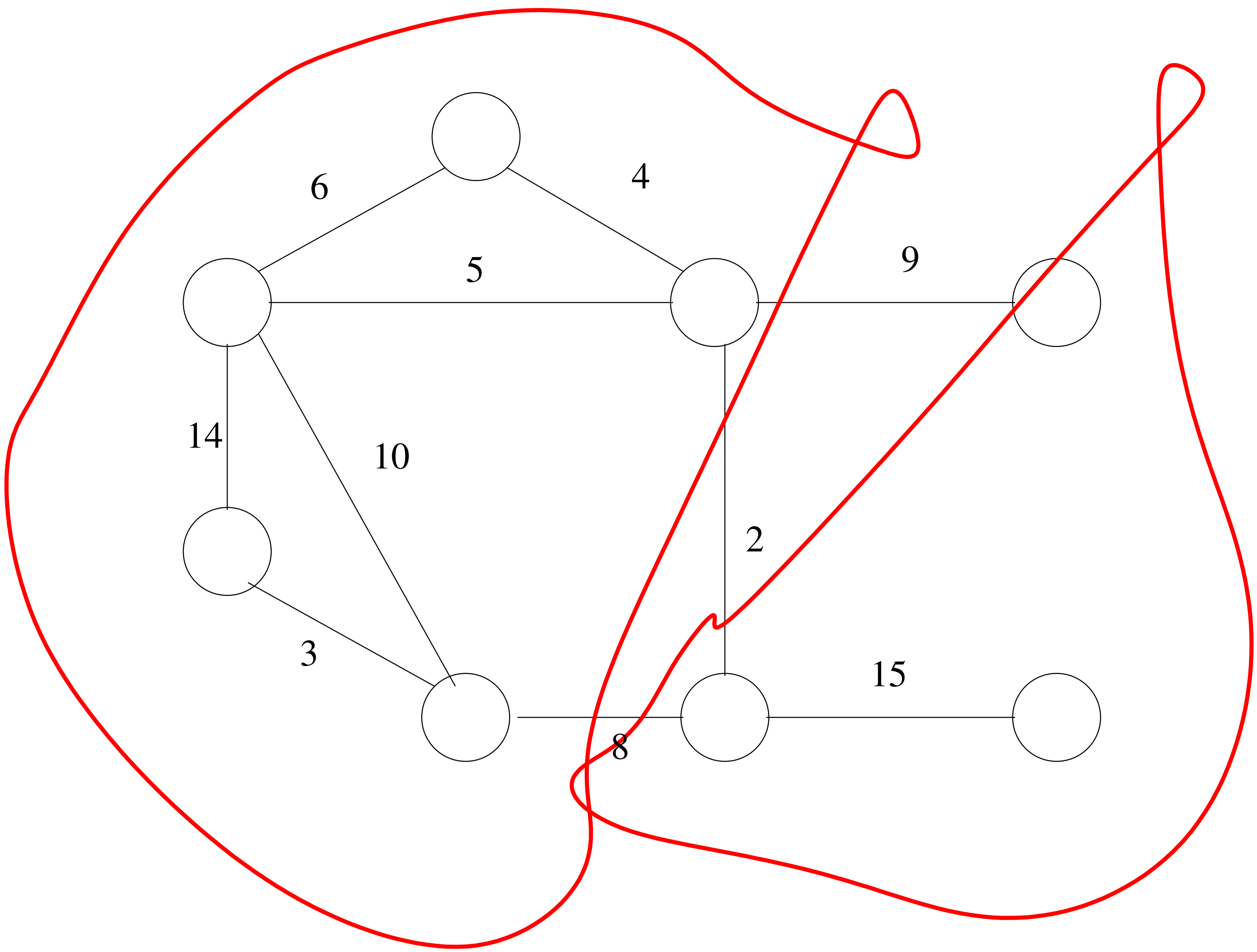
Cuts

- **A cut in a graph is a partition of the vertices into two sets S and T.**
- **An edge (u, v) with $u \in S$ and $v \in T$ is said to cross the cut.**



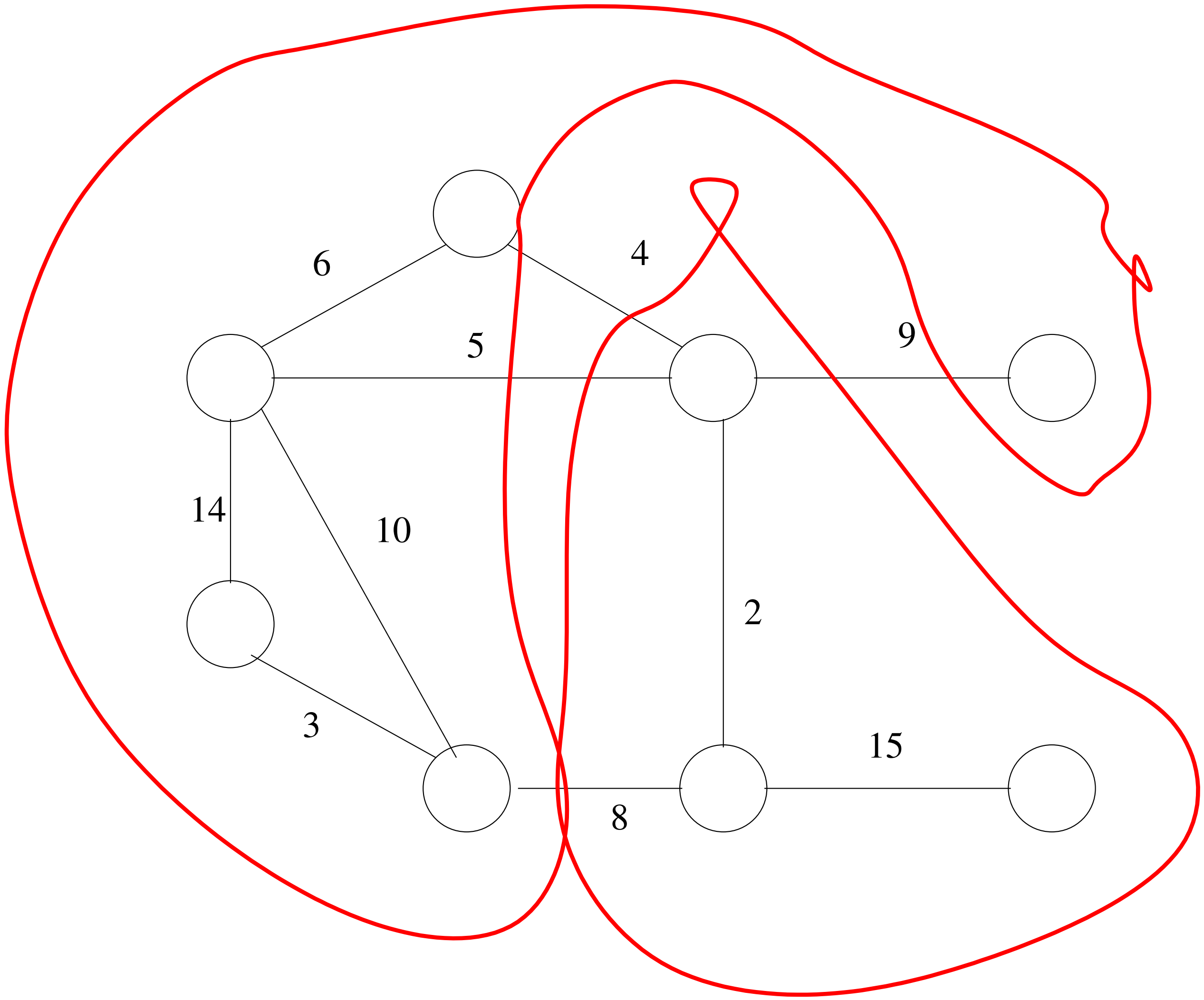
Cuts

- **A cut in a graph is a partition of the vertices into two sets S and T.**
- **An edge (u, v) with $u \in S$ and $v \in T$ is said to cross the cut.**



Cuts

- **A cut in a graph is a partition of the vertices into two sets S and T.**
- **An edge (u, v) with $u \in S$ and $v \in T$ is said to cross the cut.**



Greedy Property

Recall that we assume all edges weights are unique.

Greedy Property: The minimum weight edge crossing a cut is in the minimum spanning tree.

Proof Idea: Assume not, then remove an edge crossing the cut and replace it with the minimum weight edge.

Restatement Lemma: Let $G = (V, E)$ be an undirected graph with edge weights w. Let $A \subseteq E$ be a set of edges that are part of a minimum spanning tree. Let (S, T) be a cut with no edges from A crossing it. Then the minimum weight edge crossing (S, T) can be added to A.

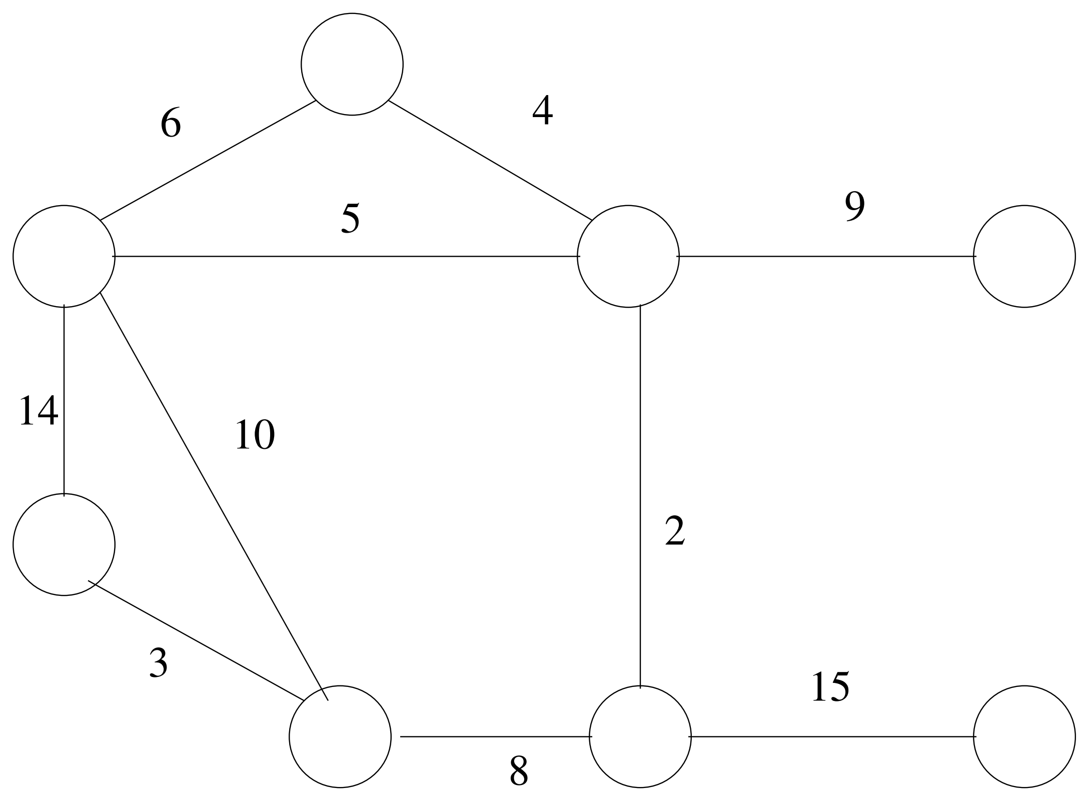
Algorithm Idea: Repeatedly choose an edge according to the Lemma, add to MST.

Challenge: Finding the edge to add.

Two standard algorithms:

- Kruskal – consider the edges in increasing order of weight
- Prim – start at one vertex and grow the tree.

Example: Run both agorithms



Kruskal's Algorithm: detailed implementation

Idea: Consider edges in increasing order.

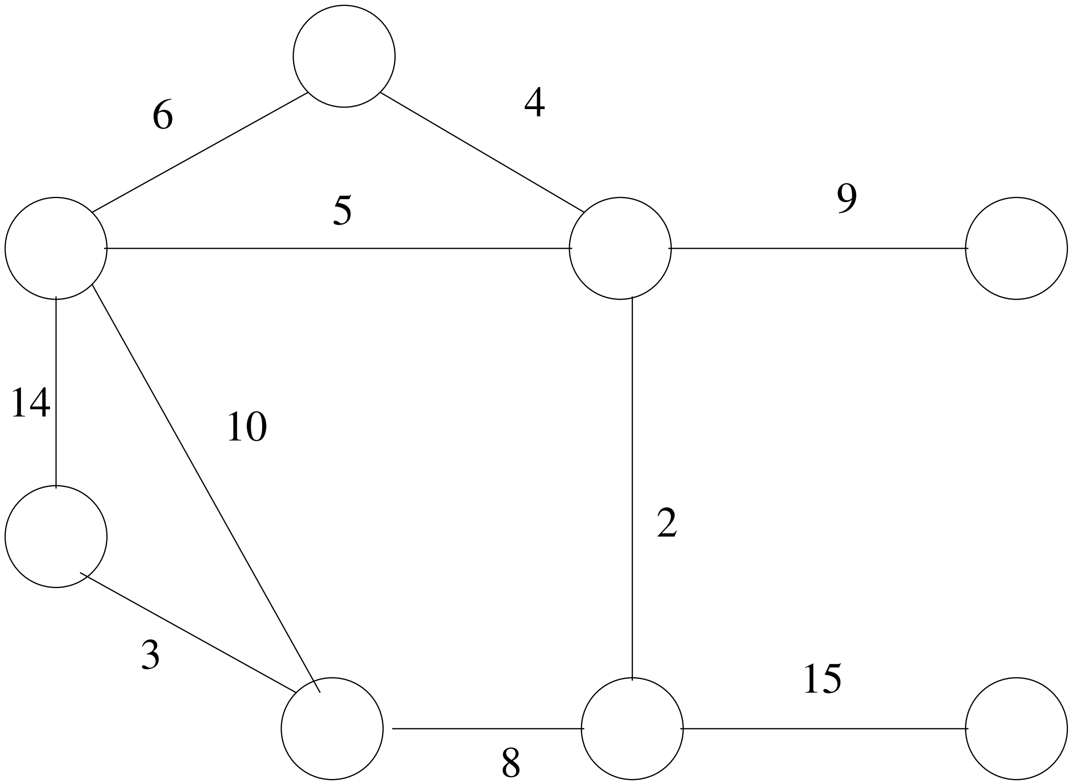
Need: a data structure to maintain the sets of vertices in each component of the current forrest

- Make-Set(v) puts v in a set by itself
- Find-Set(v) returns the name of v's set
- Union(u, v) combines the sets that u and v are in

MST-Kruskal(G, w)

```
A ← ∅
for each vertex v ∈ V[G]
        do Make-Set(v)
sort the edges of E into nondecreasing order by weight w
for each edge (u, v) ∈ E, taken in nondecreasing order by weight
        do if Find-Set(u) ≠ Find-Set(v)
                then A ← A ∪ {(u, v)}
                        Union(u, v)
return A
```

Example



Prim's Algorithm

Idea: Grow the MST from one node going out

Need: a data structure to maintain the edges crossing the cut, and choose minimum. We will maintain, for each vertex, the minimum weight incident edge crossing the cut

- INSERT(v) **puts** v **in the structure**
- EXTRACT-MIN() **finds and returns the node with minimum key value**
- DECREASE-KEY(v, w) **updates (decreases) the key of** v

MST-Prim(G, w, r)

```
for each u ∈ V[G]
        do key[u] ← ∞
           π[u] ← NIL
key[r] ← 0
Q ← V[G]
while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
           for each v ∈ Adj[u]
               do if v ∈ Q and w(u, v) < key[v]
                     then π[v] ← u
                          key[v] ← w(u, v)
```

Example