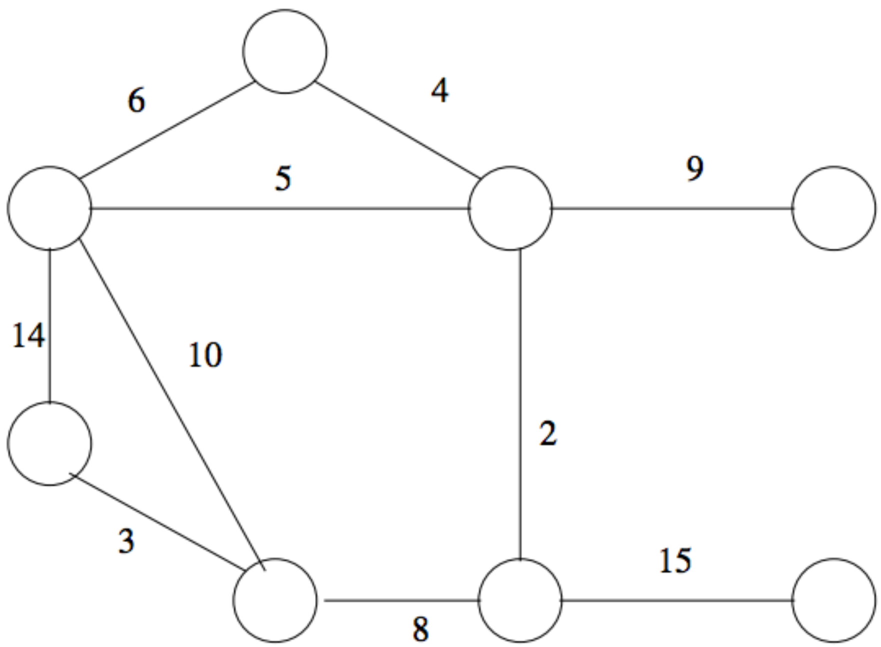
Minimum Spanning Trees

- $G = (V, E)$ 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 $w(T) = \Sigma_{e \in T} w(e)$
- A minimum spanning tree is a tree of minimum total weight.

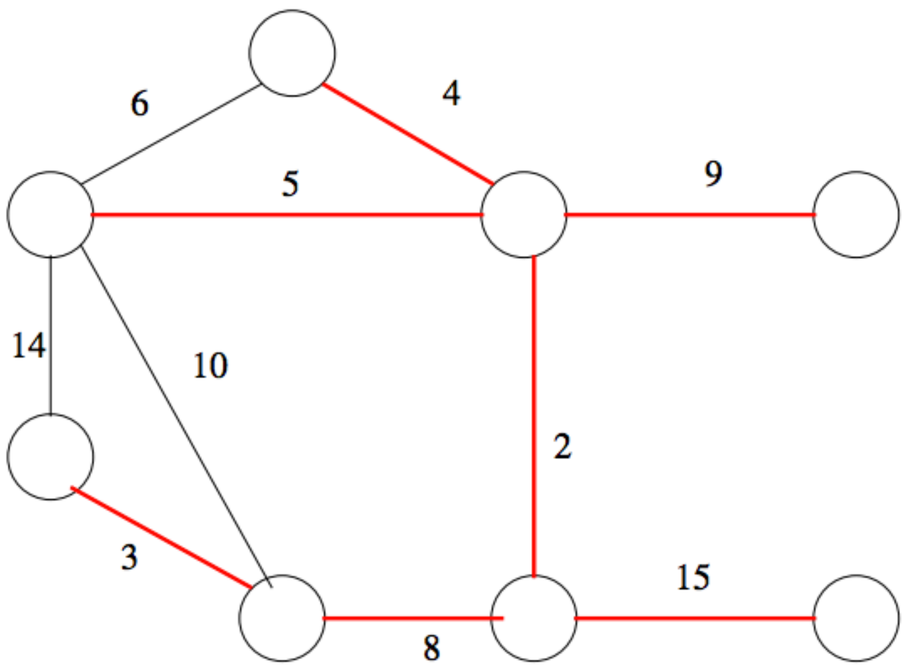

6
4
5
9
14
10
2
3
15
8

Minimum Spanning Trees

- $G = (V, E)$ **is an undirected graph with non-negative edge weights** $w : E \rightarrow Z^+$

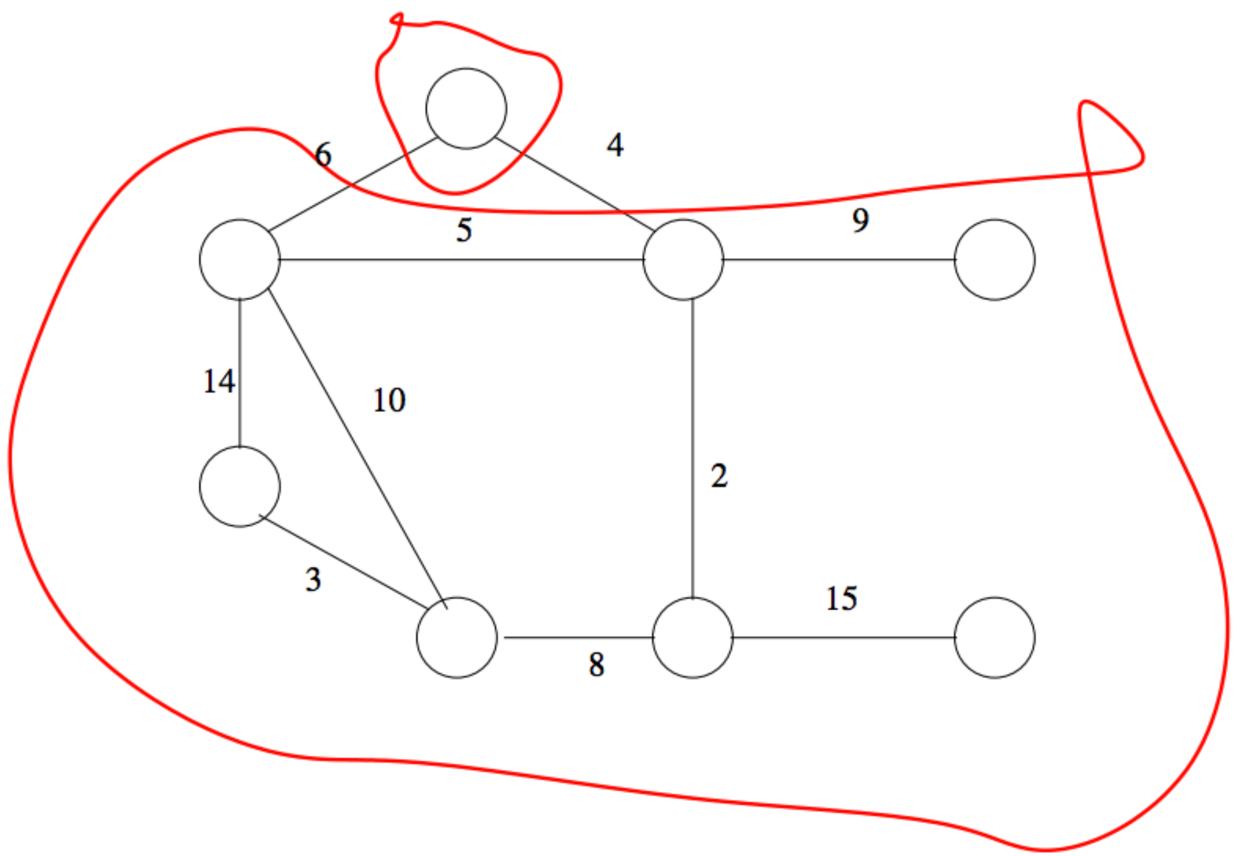
- **We assume wlog that edge weights are distinct**
- **A spanning tree is a tree with** $V - 1$ **edges, i.e. a tree that connects all the vertices.**
- **The total cost (weight) of a spanning tree T is defined as** $w(T) = \Sigma_{e \in T} w(e)$

- **A minimum spanning tree is a tree of minimum total weight.**


6
4
5
9
14
10
2
3
15
8

Cuts

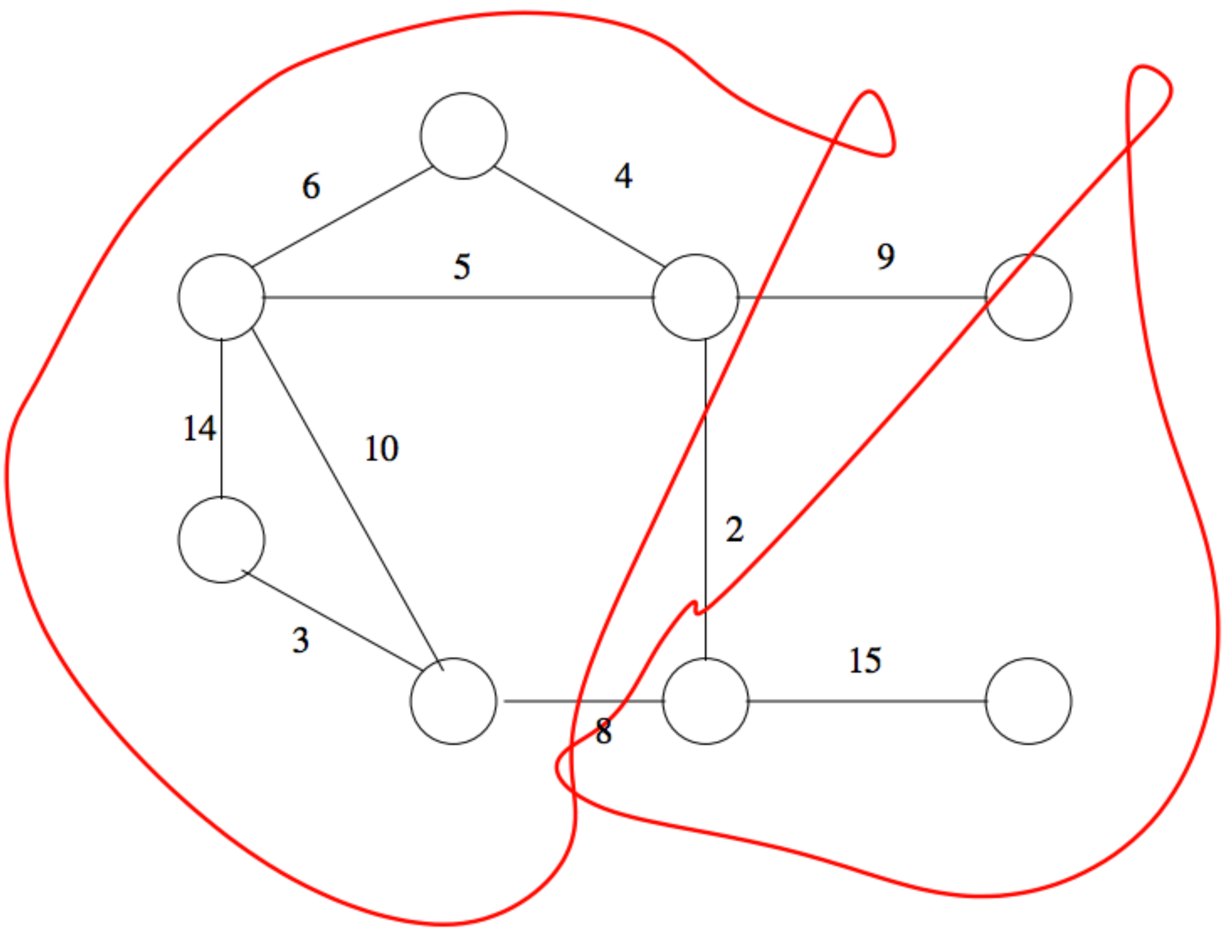
- **A cut in a graph is a partition of the vertices into two sets S and T.**
- **An edge (u, v) with $u \in S$ and $v \in T$ is said to cross the cut.**



6
4
5
9
14
10
2
3
15
8

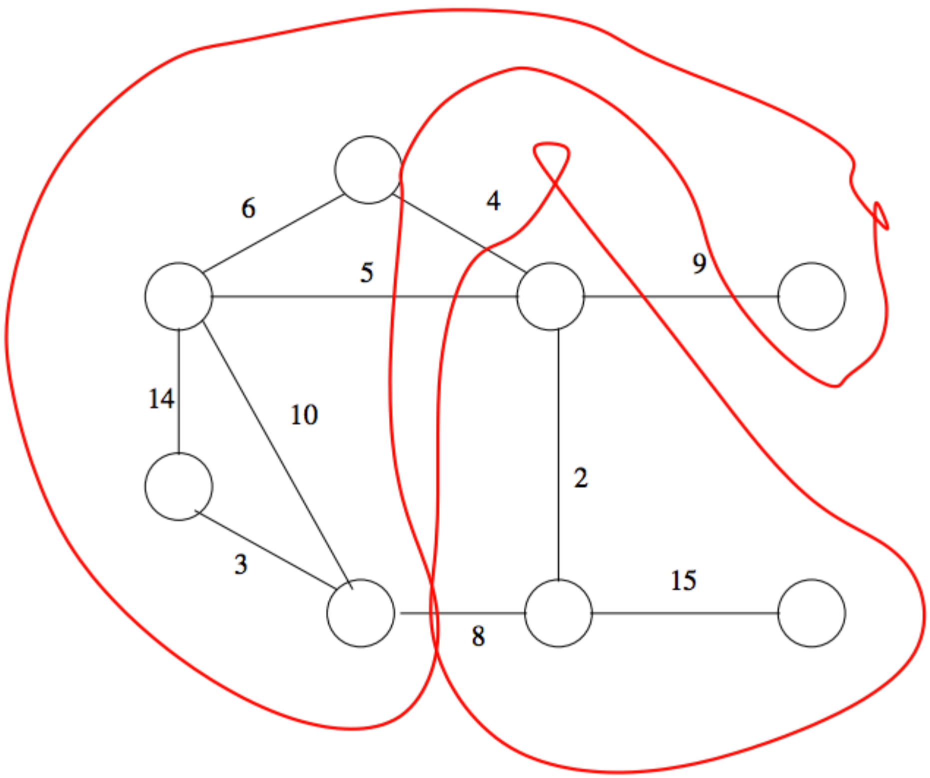
Cuts

- **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.**


6
4
5
9
14
10
2
3
15
8

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.**



6
4
5
9
14
10
2
3
15
8

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.**

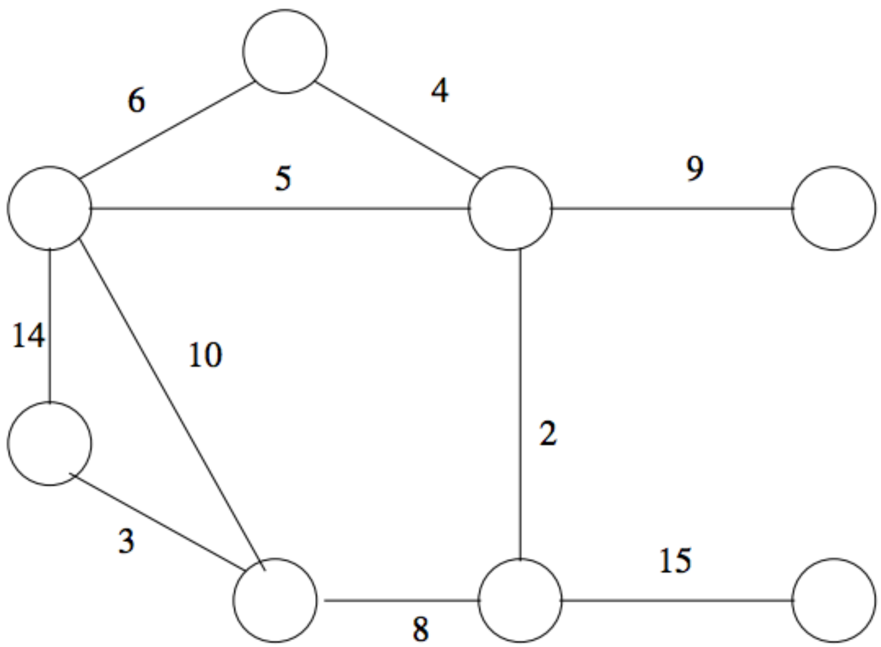
Kruskal's Algorithm

Idea: Consider edges in increasing order.

MST-Kruskal(G, w)

$A \leftarrow \emptyset$
for each vertex $v \in V[G]$
    do MAKE-SET(v)
sort the edges of E **into nondecreasing order by weight** w
for each edge $(u, v) \in E$**, taken in nondecreasing order by weight**
    do if FIND-SET$(u) \neq$ FIND-SET(v)
        then $A \leftarrow A \cup \{(u, v)\}$
            UNION(u, v)
return A

Example


6
4
5
9
14
10
2
3
15
8

Analysis

- **Sorting**
- n **UNIONS**
- m **FIND-SET ops**
- **Each UNION, FINDSET takes** $O(\log^* n)$ **time.**
- **With Sorting –** $O(m \log n)$ **time**
- **Data already sorted –** $O(m \log^* n)$ **time.**

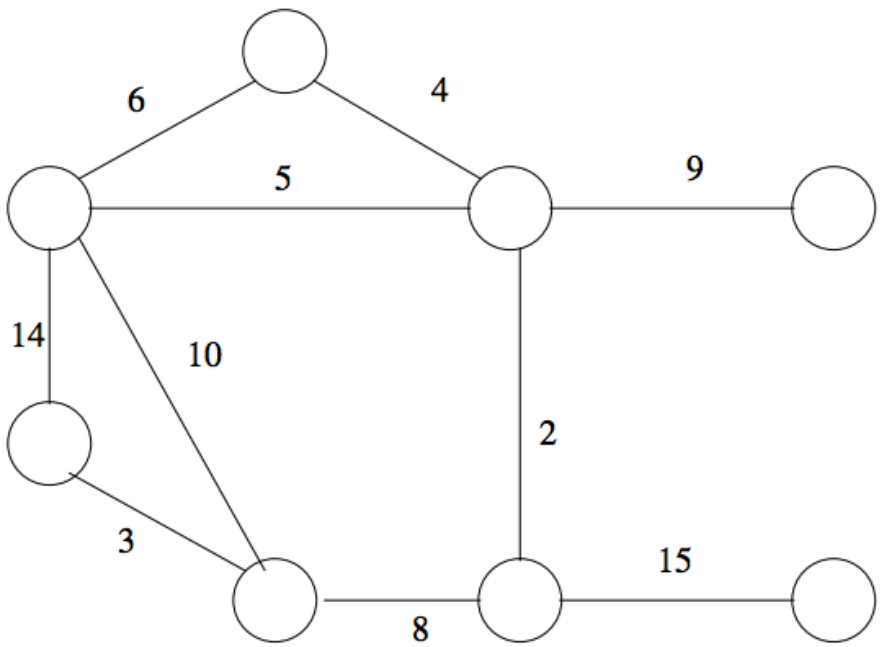
Prim's Algorithm

Idea: **Grow the MST from one node going out**

MST-Prim(G, w, r)

```
for each u ∈ V[G]
      do key[u] ← ∞
         π[u] ← NIL
         INSERT(u)
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)
                         DECREASE-KEY(v, w(u, v))
```

Example


6
4
5
9
14
10
2
3
15
8

Analysis

- n **Inserts and Delete-Mins**
- m **Decrease Keys**
- **Use a heap.** $O(\log n)$ **per operation. Total** $O(m \log n)$ **.**
- **Use a fibonacci heap. Decrease Key reduced to amortized** $O(1)$ **time. Total time** $O(m + n \log n)$