Minimum Spanning Trees

Spanning Trees

- Given (connected) graph $G(V,E)$,
  a spanning tree $T(V',E')$:
  - Is a subgraph of G; that is, $V' \subseteq V$, $E' \subseteq E$.
  - Spans the graph $(V' = V)$
  - Forms a tree (no cycle);
  - So, E' has $|V| - 1$ edges

Minimum Spanning Trees

- Edges are weighted: find minimum cost spanning tree
- Applications
  - › Find cheapest way to wire your house
  - › Find minimum cost to send a message on the Internet

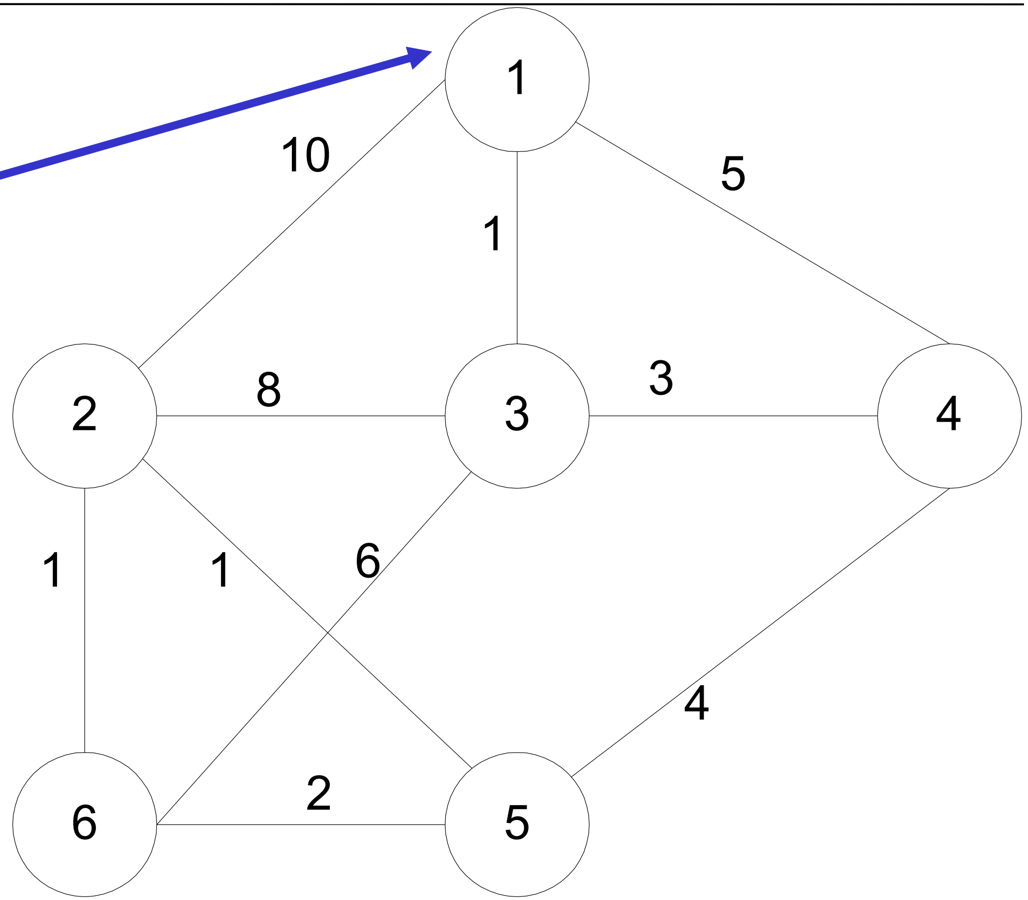
Two Algorithms

- Prim: (build tree incrementally)
  - Pick lower cost edge connected to known (incomplete) spanning tree that does not create a cycle and expand to include it in the tree
- Kruskal: (build forest that will finish as a tree)
  - Pick lowest cost edge not yet in a tree that does not create a cycle. Then expand the set of included edges to include it. (It will be somewhere in the forest.)

Prim’s algorithm

Starting from empty T, choose a vertex at random and initialize

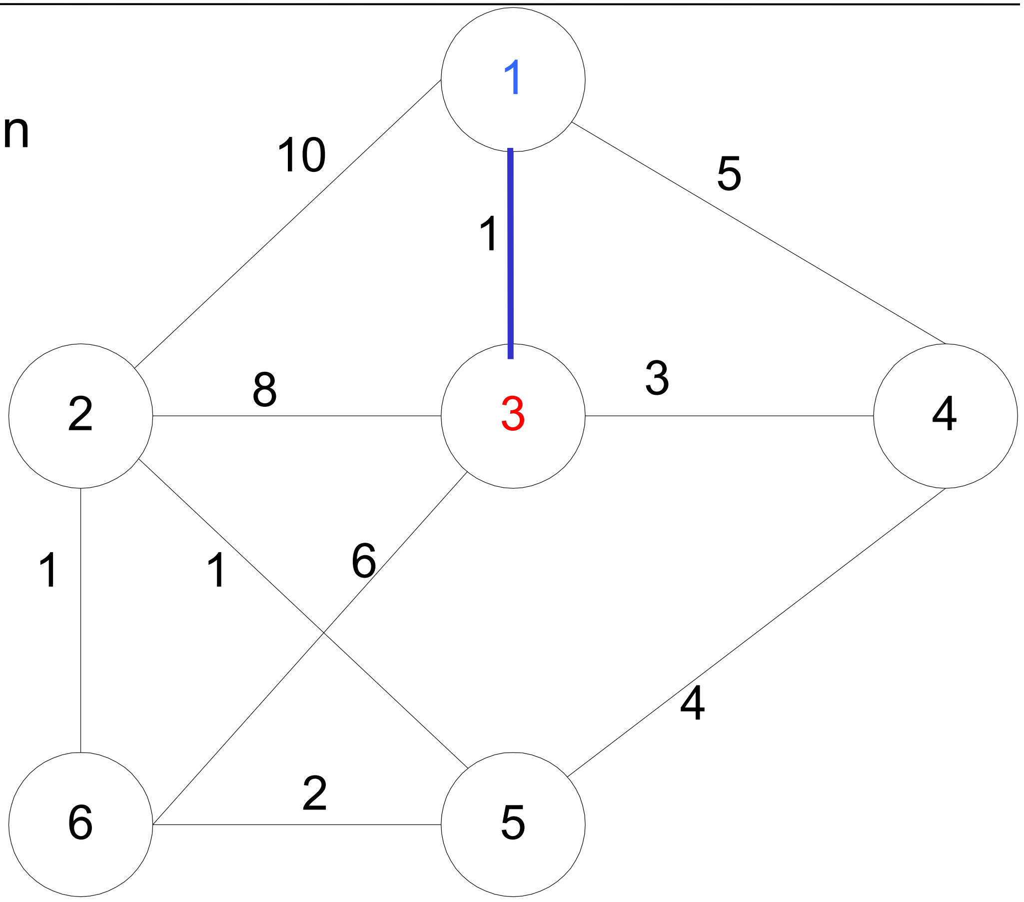
V = {1), E’ ={}



Prim's algorithm

Choose the vertex u not in V such that edge weight from u to a vertex in V is minimal (greedy!)

V={1,3} E'= {(1,3) }



Prim's algorithm

Repeat until all vertices have been chosen

Choose the vertex u not in V such that edge weight from v to a vertex in V is minimal (greedy!)

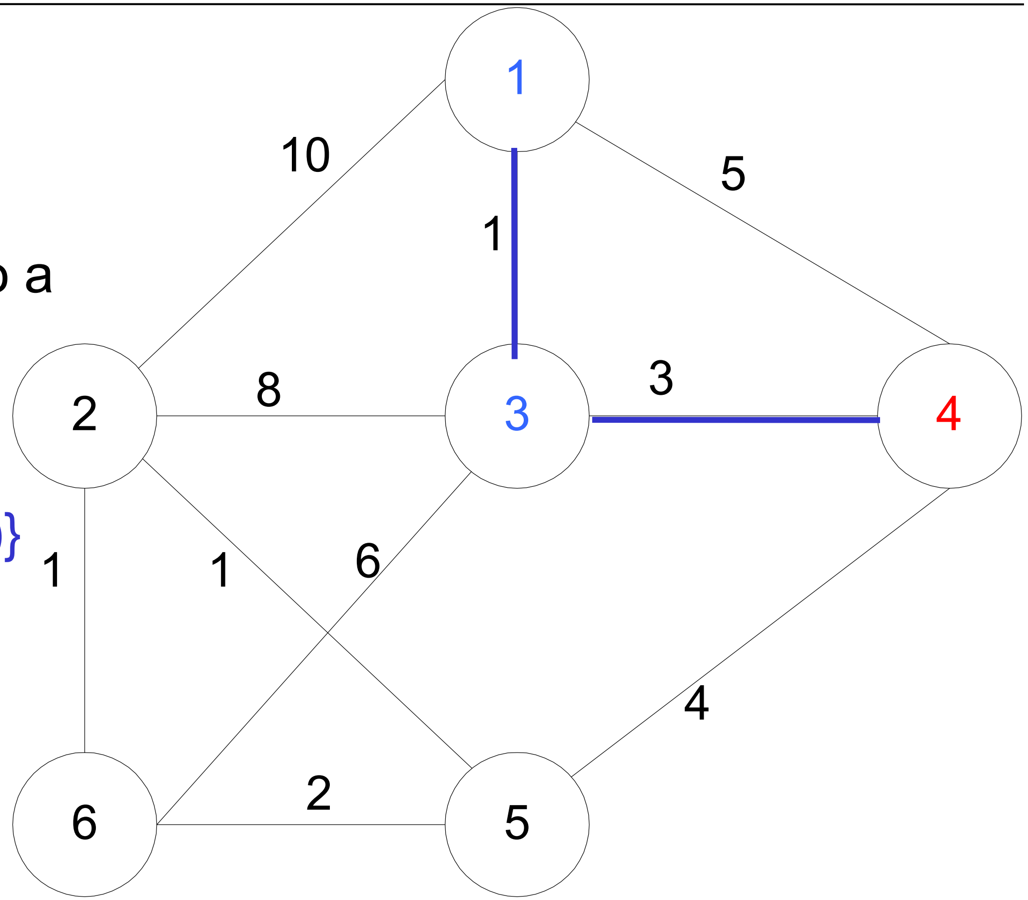
V= {1,3,4} E'= {(1,3),(3,4)}

V={1,3,4,5} E'={(1,3),(3,4),(4,5)}

….

V={1,3,4,5,2,6}

E'={(1,3),(3,4),(4,5),(5,2),(2,6)}



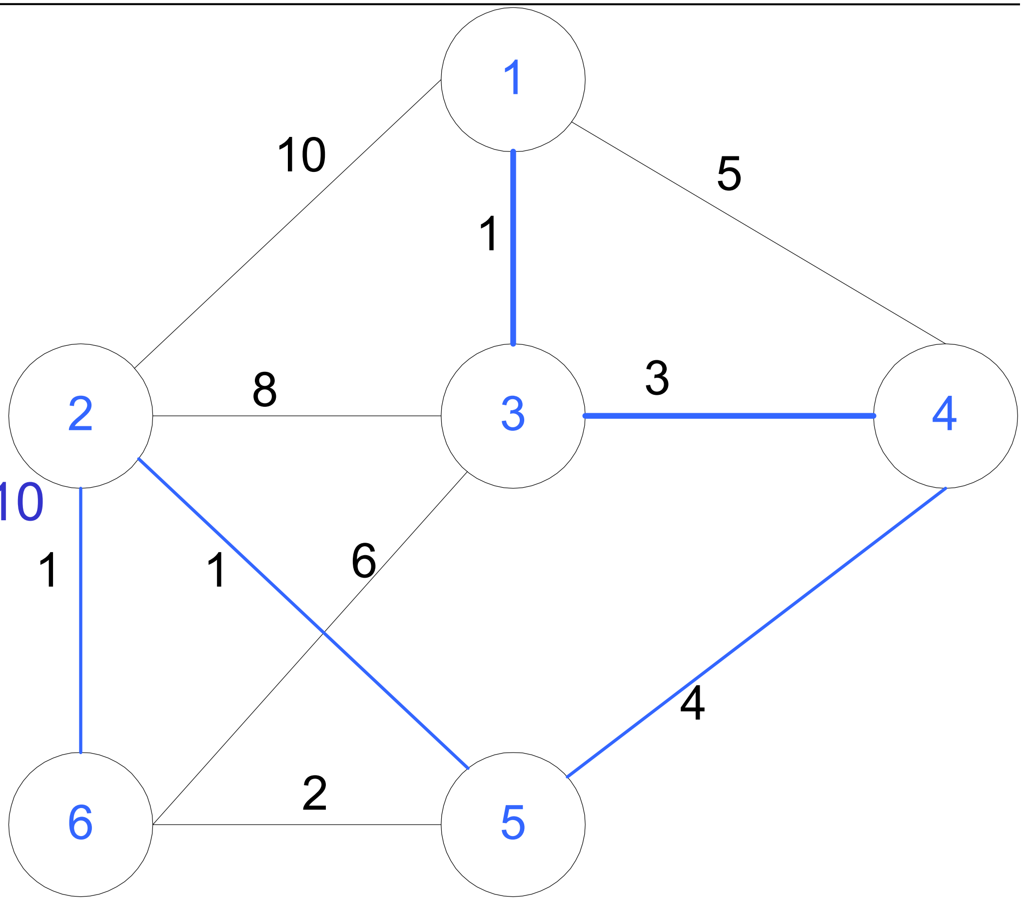
Prim's algorithm

Repeat until all vertices have been chosen

V={1,3,4,5,2,6}

E'={(1,3),(3,4),(4,5),(5,2),(2,6)}

Final Cost: 1 + 3 + 4 + 1 + 1 = 10



Kruskal's Algorithm

- Select edges in order of increasing cost
- Accept an edge to expand tree or forest only if it does not cause a cycle

Kruskal's Algorithm

Initialize a forest of trees, each tree being a single node

Build a priority queue of edges with priority being lowest cost

Repeat until |V| -1 edges have been accepted {

Deletemin edge from priority queue

If it forms a cycle then discard it

else accept the edge – It will join 2 existing trees yielding a larger tree and reducing the forest by one tree

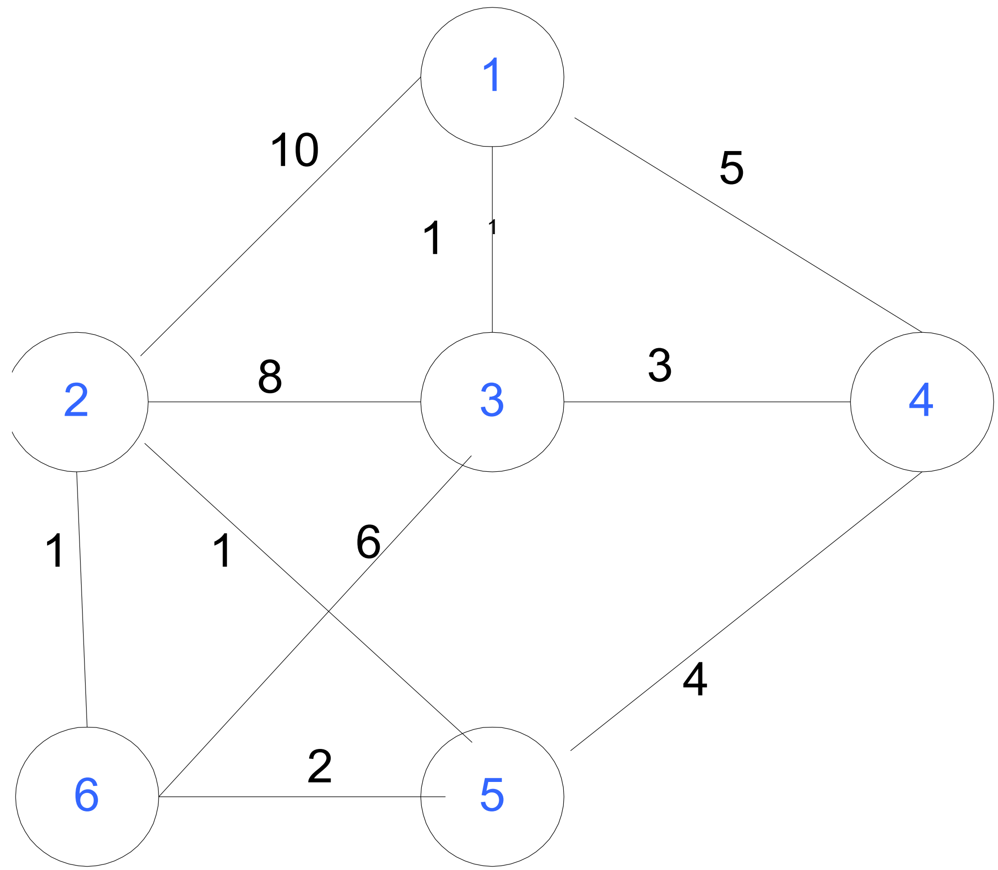
}

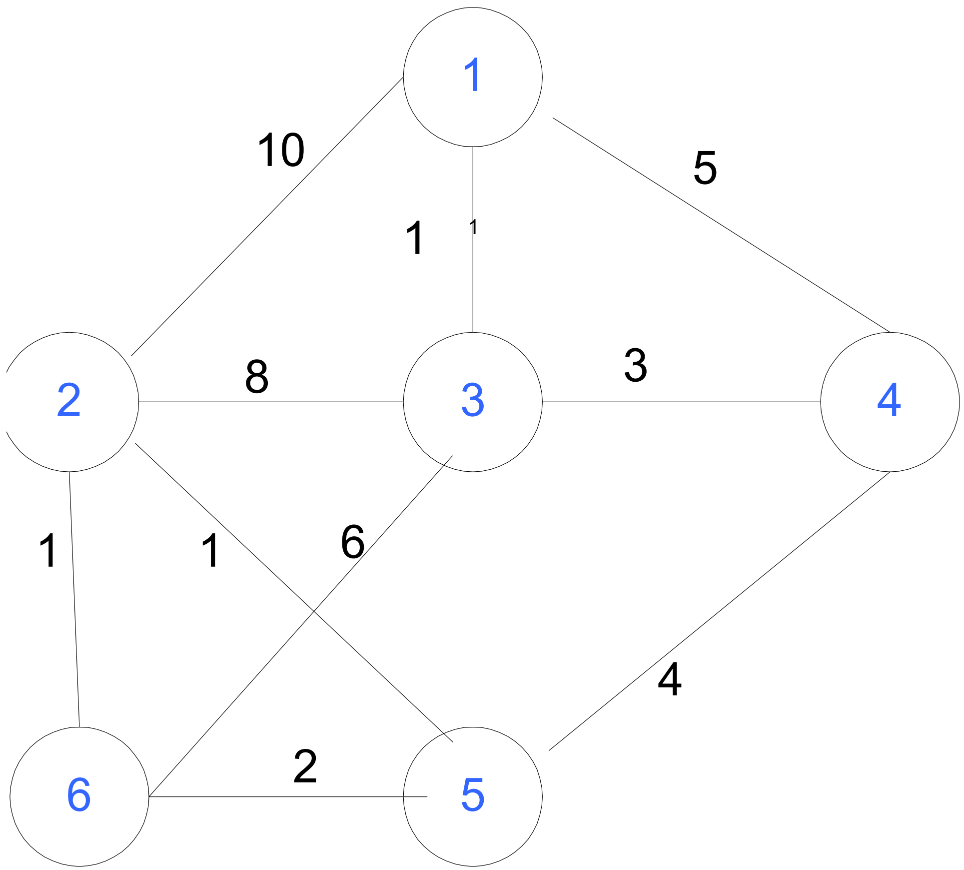
The accepted edges form the minimum spanning tree

Detecting Cycles

- If the edge to be added (u,v) is such that vertices u and v belong to the same tree, then by adding (u,v) you would form a cycle

Example



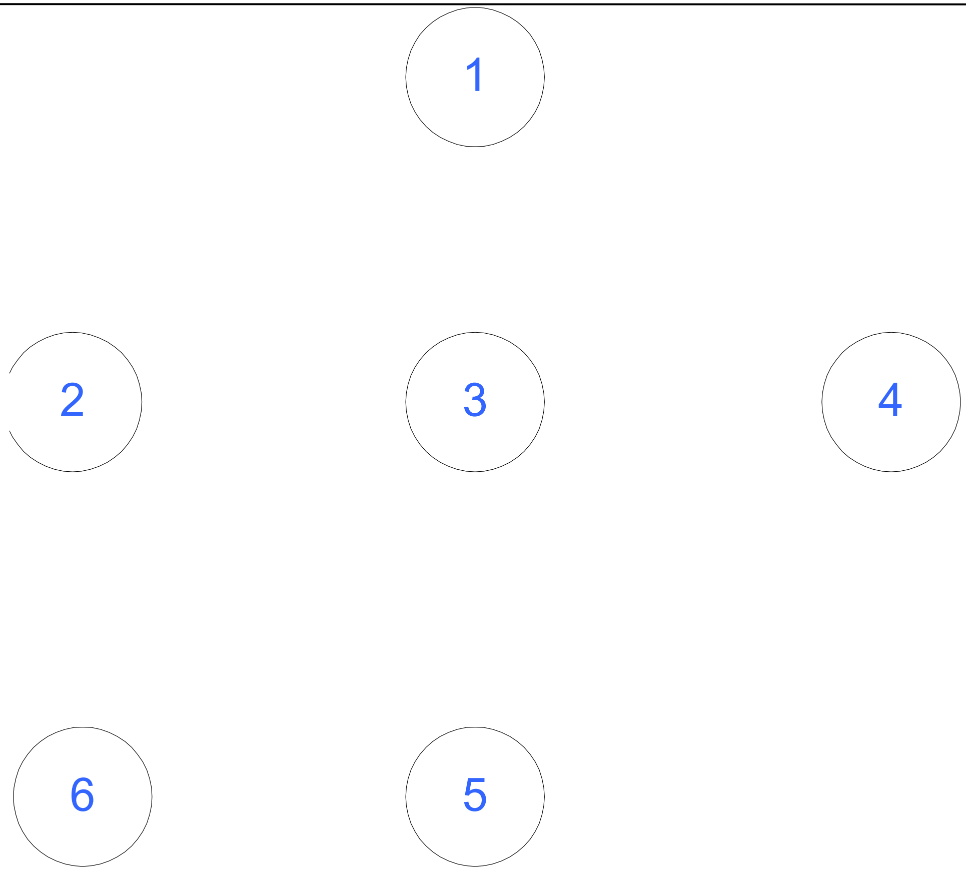

1
10
5
1
1
2
8
3
3
4
1
1
6
4
2
6
5

Initialization

Initially, Forest of 6 trees

F= {{1},{2},{3},{4},{5},{6}}

Edges in a heap (not shown)



Step 1

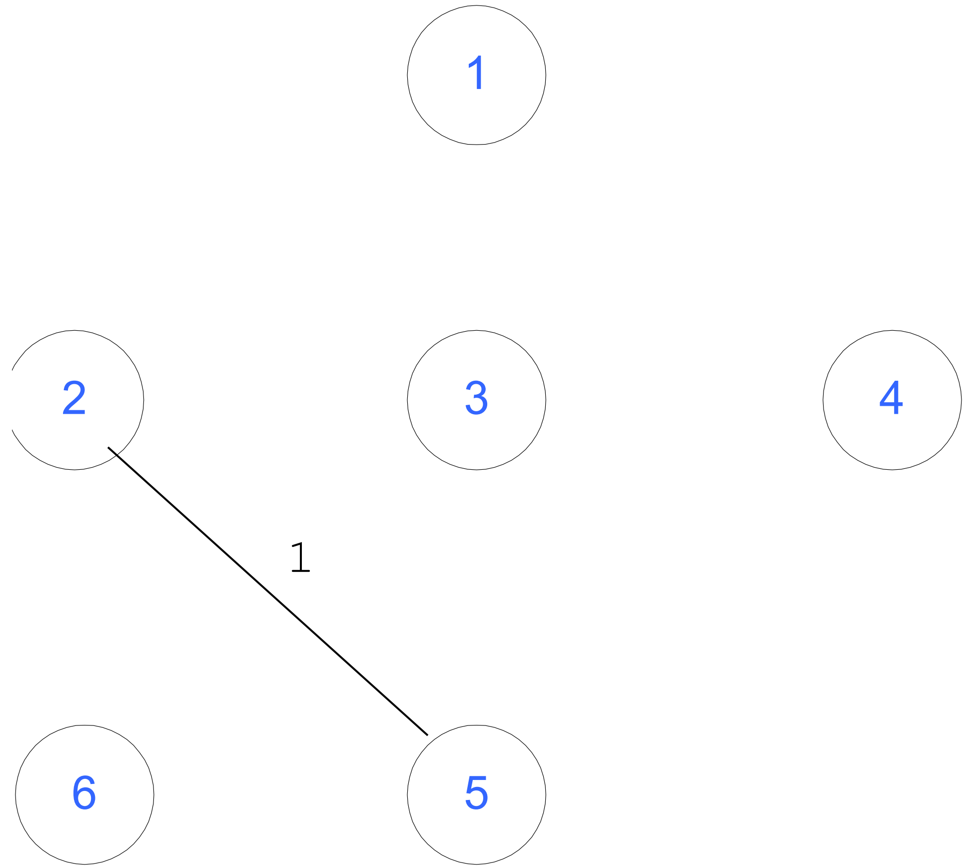
Select edge with lowest cost (2,5)

Find(2) = 2, Find (5) = 5

Union(2,5)

F= {{1},{2,5},{3},{4},{6}}

1 edge accepted



Step 2

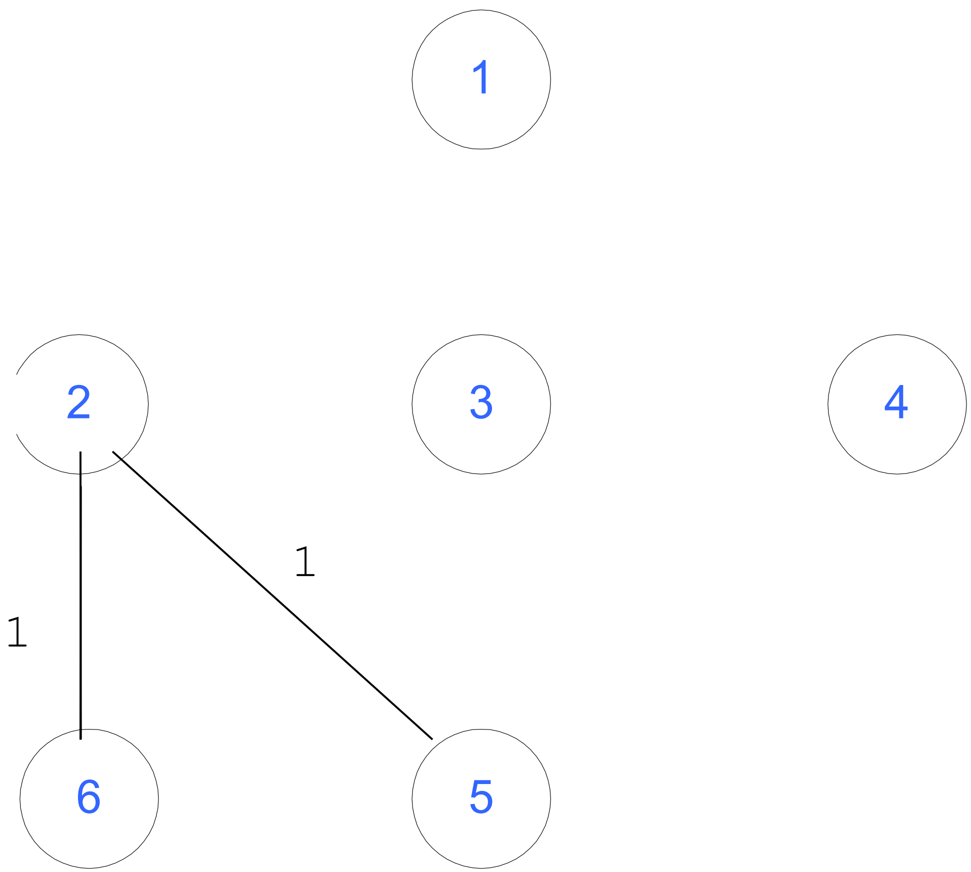
Select edge with lowest cost (2,6)

Find(2) = 2, Find (6) = 6

Union(2,6)

F= {{1},{2,5,6},{3},{4}}

2 edges accepted



Step 3

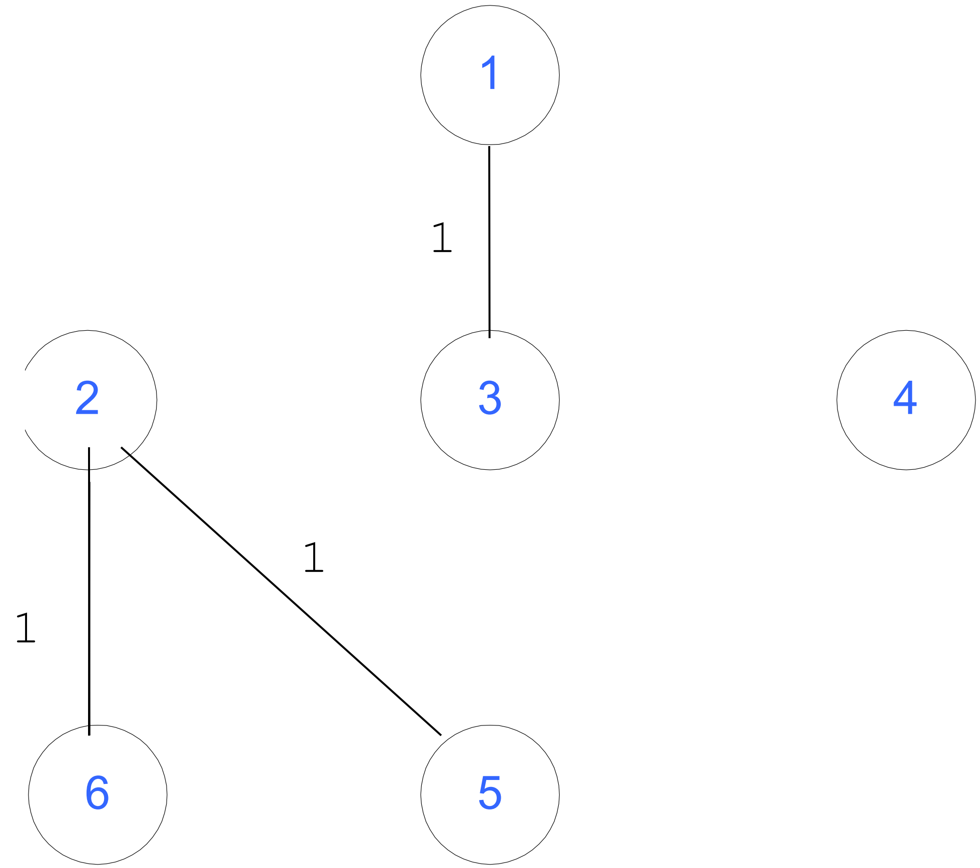
Select edge with lowest cost (1,3)

Find(1) = 1, Find (3) = 3

Union(1,3)

F= {{1,3},{2,5,6},{4}}

3 edges accepted



Step 4

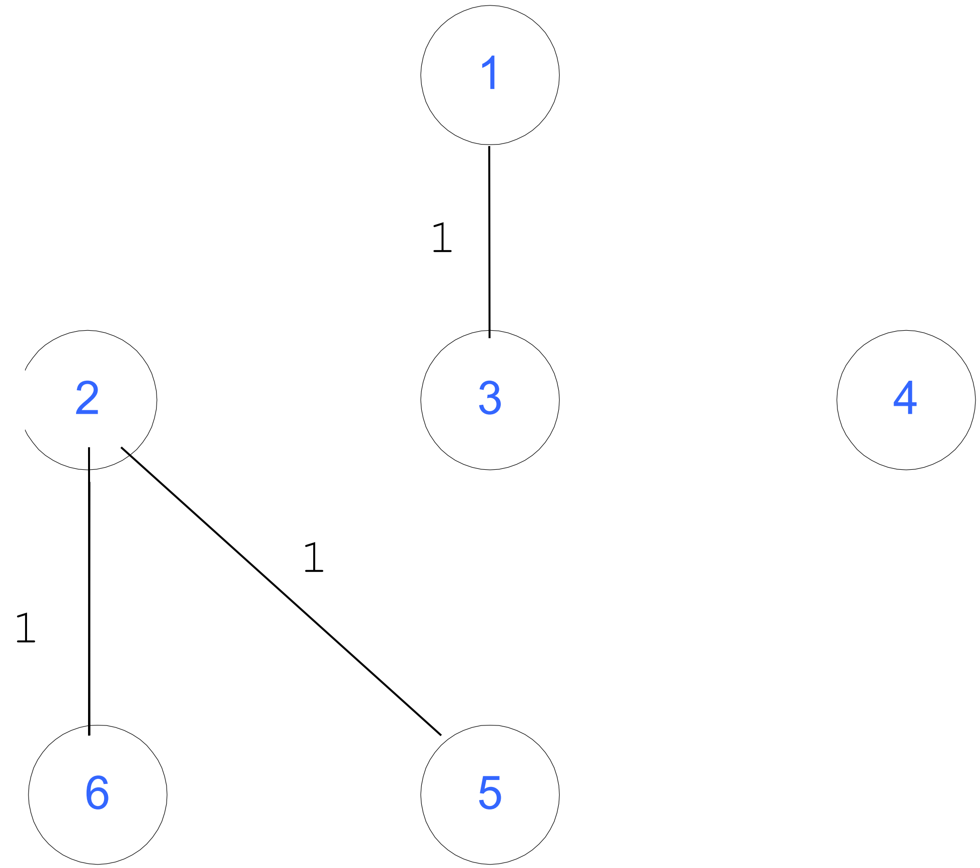
Select edge with lowest cost (5,6)

Find(5) = 2, Find (6) = 2

Do nothing

F= {{1,3},{2,5,6},{4}}

3 edges accepted



Step 5

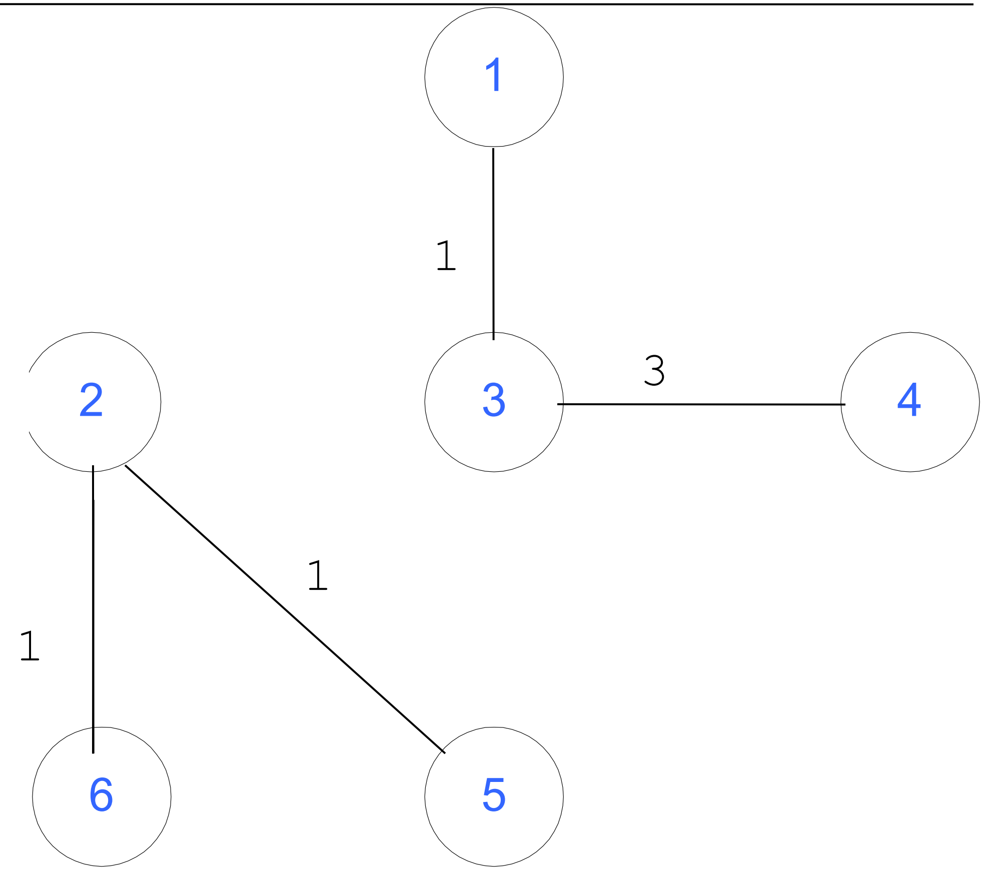
Select edge with lowest cost (3,4)

Find(3) = 1, Find (4) = 4

Union(1,4)

F= {{1,3,4},{2,5,6}}

4 edges accepted



Step 6

Select edge with lowest cost (4,5)

Find(4) = 1, Find (5) = 2

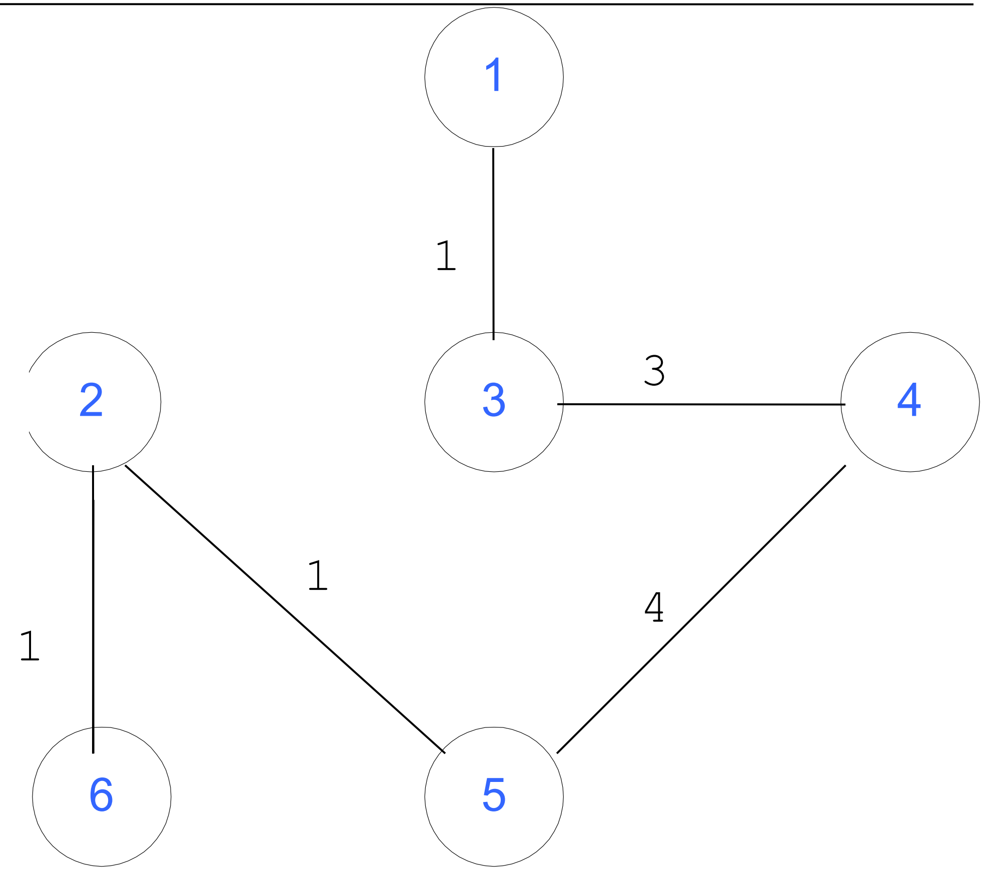
Union(1,2)

F= {{1,3,4,2,5,6}}

5 edges accepted : end

Total cost = 10

Although there is a unique spanning tree in this example, this is not generally the case



Time Complexity

- Prim's algorithm has a time complexity of O(V2), V being the number of vertices and can be improved up to O(E log V) using Fibonacci heaps.
- Kruskal's algorithm's time complexity is O(E log V), V being the number of vertices.
- Prim's algorithm runs faster in dense graphs. Kruskal's algorithm runs faster in sparse graphs.

Thank you

Any Question ????????