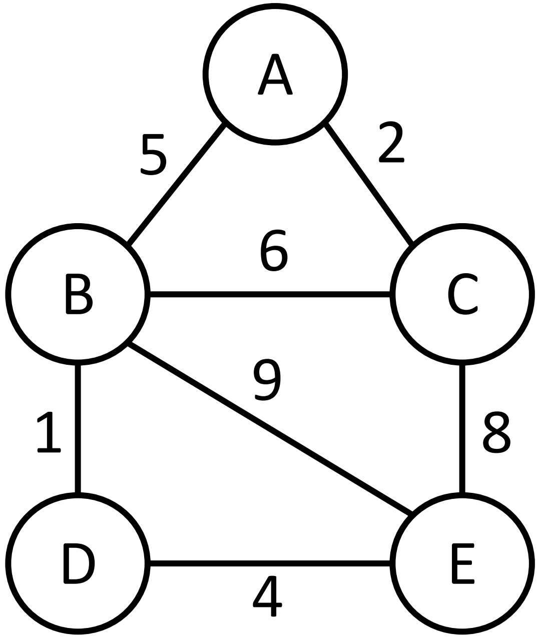
Minimum Spanning Trees

CS313E – Elements Of Software Design

Weighted Graphs

Weighted graphs: graphs with weight-associated edges

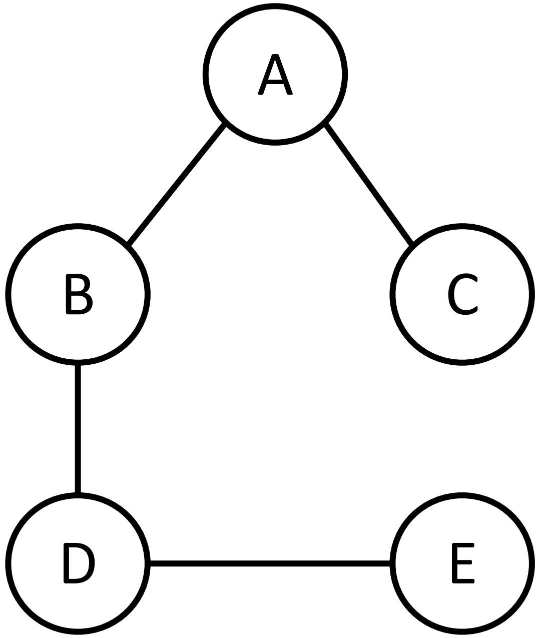
- This means that each edge is associated with a cost
- Taking a certain path may result in a higher or lower cost than taking another path

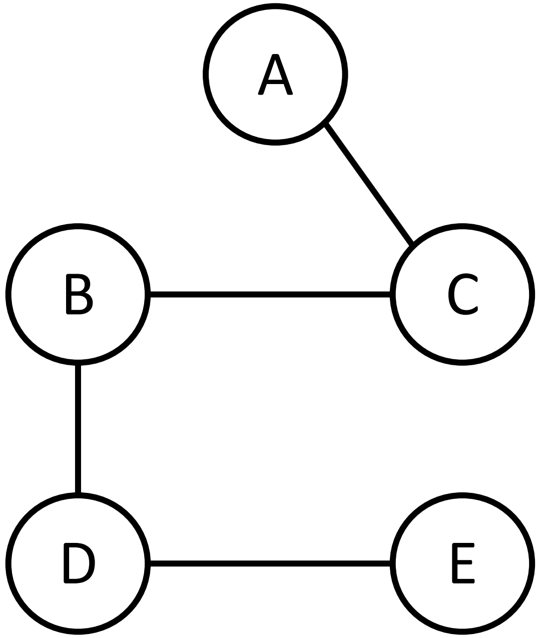


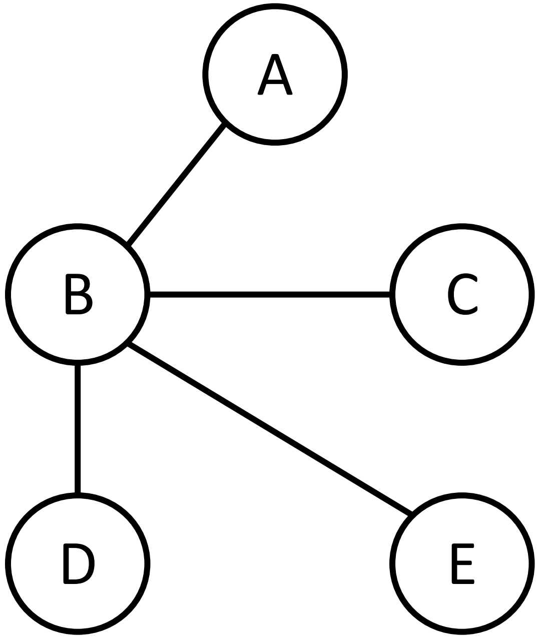
What is a Spanning tree?

Connected, acyclic graphs

- Vertices connected with the minimum number of edges (n – 1)
- Graphs may contain more than one spanning tree



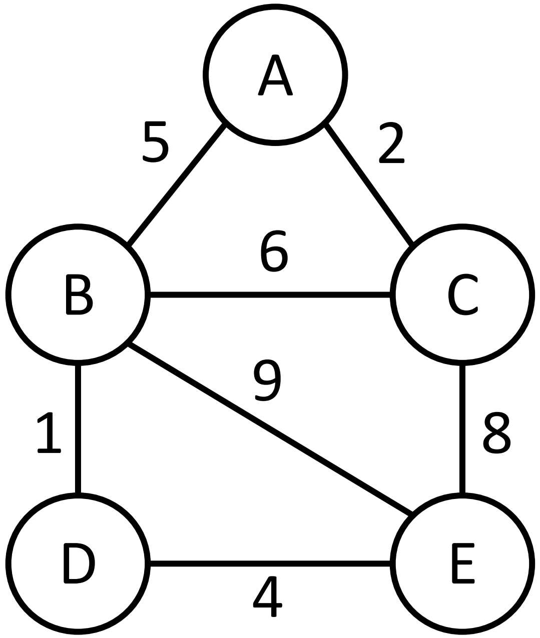


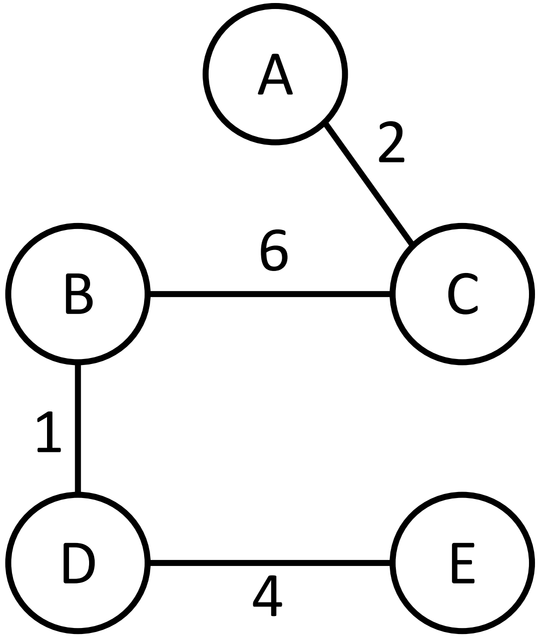


Minimum Spanning Trees (MSTs)

A spanning tree of minimum cost

- Minimum sum of edge weights
- Some graphs have exactly one MST
- In some cases, graphs can have multiple MSTs





Minimum Spanning tree

- In an undirected, weighted graph G = (V , E) with weights $w(u, v)$ for each edge $(u, v) \in$ E

- Find an acyclic subset T $\subseteq E$ that connects all of the vertices V and minimizes the total weight:
  $w(\mathrm{T}) = \sum_{(u,\, v)\, \in\ T} w\ (u, w)$

- The minimum spanning tree is (V,T)
  - There may be a multiple minimum spanning tree

Minimum Spanning tree

Algorithms for determining MSTs:

– Prim’s Algorithm

– Kruskal’s Algorithm

- Both are Greedy algorithms which produce optimal minimum solutions

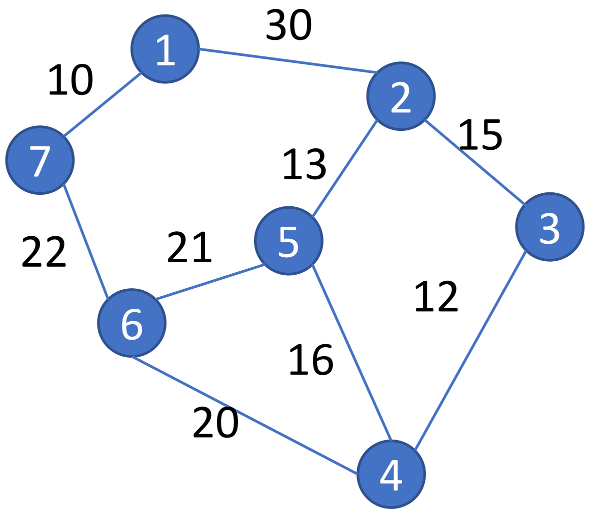
Kruskal’s Algorithm

Joseph Bernard Kruskal, Jr. (1928 - 2010)
https://en.wikipedia.org/wiki/Joseph_Kruskal

- It runs on edges
- Two Steps:
  - Sort edges by increase edge weight
  - Select the first |V| -1 edges that do not generate a cycle

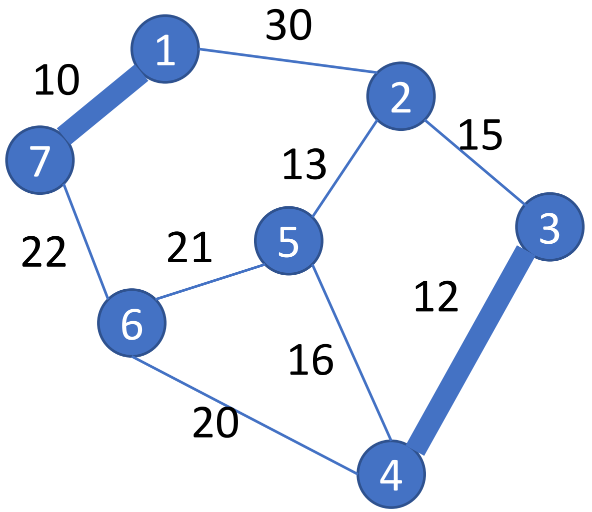
Kruskal’s Algorithm

- Always select a minimum cost edge
- If it form a cycle don’t select it



Edge	Cost	Selection
1-2	30	
2-3	15	
3-4	12	
4-5	16	
5-2	13	
5-6	21	
6-4	20	
6-7	22	
7-1	10	

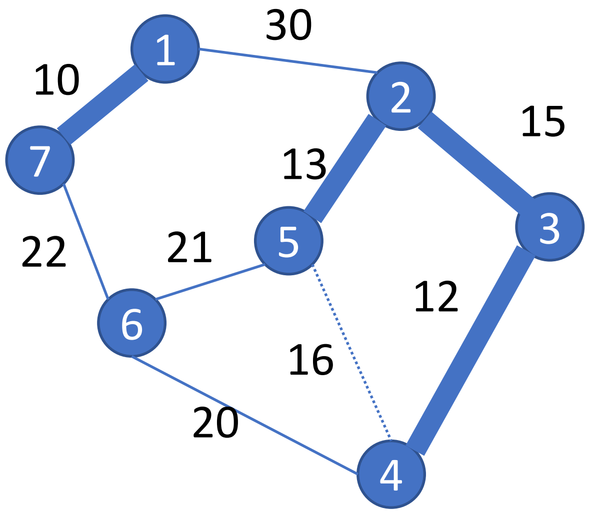
- Kruskal's Algorithm
  - Always select a minimum cost edge
  - If it form a cycle don't select it



Edge	Cost	Selection
1-2	30	
2-3	15	
3-4	12	✓
4-5	16	
5-2	13	
5-6	21	
6-4	20	
6-7	22	
7-1	10	✓

Kruskal’s Algorithm

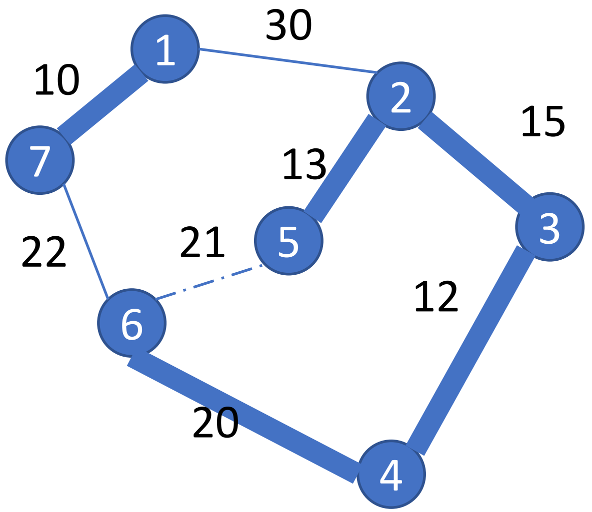
- Always select a minimum cost edge
- If a form a cycle don’t select it



Edge	Cost	Selection
1-2	30	
2-3	15	✓
3-4	12	✓
~~4-5~~	~~16~~	Cycle
5-2	13	✓
5-6	21	
6-4	20	
6-7	22	
7-1	10	✓

Kruskal’s Algorithm

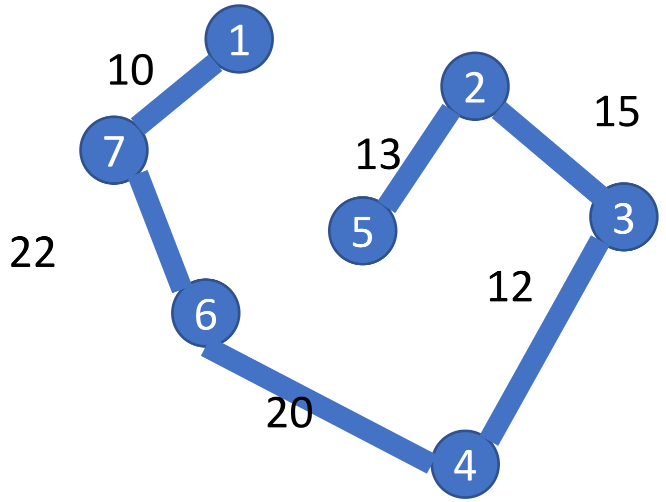
- Kruskal’s Algorithm
  - Always select a minimum cost edge
  - If it form a cycle don’t select it



Edge	Cost	Selection
1-2	30	
2-3	15	✓
3-4	12	✓
~~4-5~~	~~16~~	Cycle
5-2	13	✓
~~5-6~~	~~21~~	Cycle
6-4	20	✓
6-7	22	
7-1	10	✓

Kruskal’s Algorithm

- Kruskal’s Algorithm
  - Always select a minimum cost edge
  - If it form a cycle don’t select it
  - If we reach (V-1) edges, discard remain
  - The total cost = 92



Edge	Cost	Selection
~~1-2~~	~~30~~	Discarded
2-3	15	✓
3-4	12	✓
~~4-5~~	~~16~~	Cycle
5-2	13	✓
~~5-6~~	~~21~~	Cycle
6-4	20	✓
6-7	22	✓
7-1	10	✓

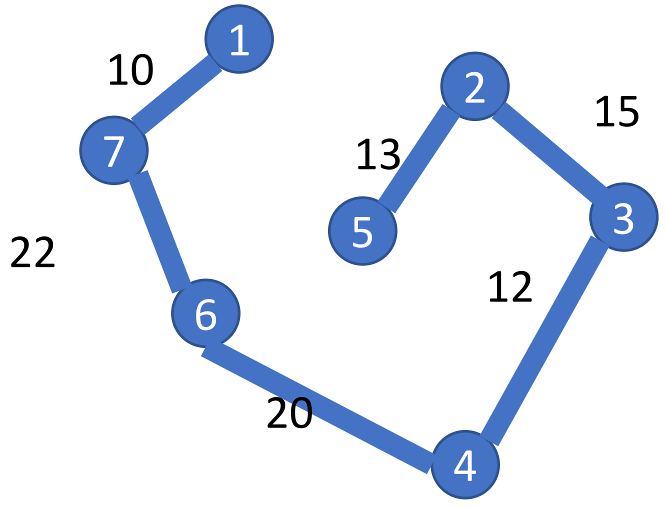
Kruskal’s Algorithm Pseudocode

MST-KRUSKAL (G, w)

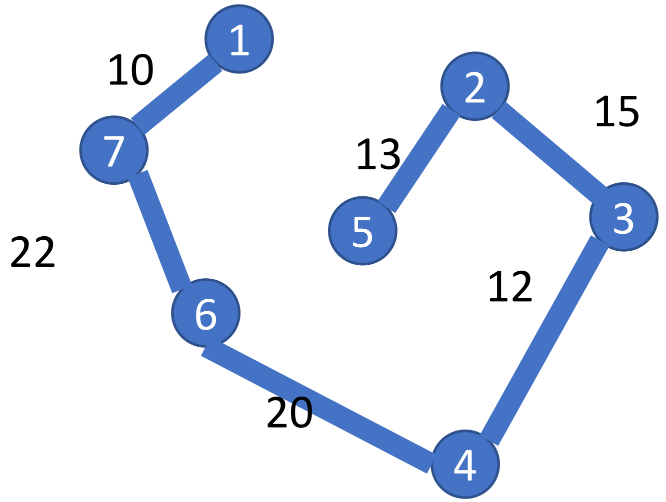
A=0
for each vertex $v \in$ G.V
        MAKE-SET (v)

Sort the edges of (G,E) into non decreasing order by weight w

for each edge $(u, v) \in$ G.E ,   taken in non decreasing order by weight
        if FIND-SET(u) $\neq$ FIND-SET(v)
                A = A $\cup$ $\{(u, v)\}$
                Union (u, v)
Return A



Kruskal’s Algorithm Time complexity



MST-KRUSKAL (G,w)
A=0 O(1)
for each vertex $v \in$ G.V
    Make-Set (v) O(V)
Sort the edges of (G,E) into non decreasing order by weight w O (E log E)
for each edge $(u,v) \in$ G.E.taken in non decreasing order by weight
    if Find-Set(u) $\neq$ Find-Set(v)
        A = A U $\{(u,v)\}$ O (E log V)
        Union (u,v)
Return A

Total running time O (E log E) Observing $|E| < |V^2|$, O (E * log V)

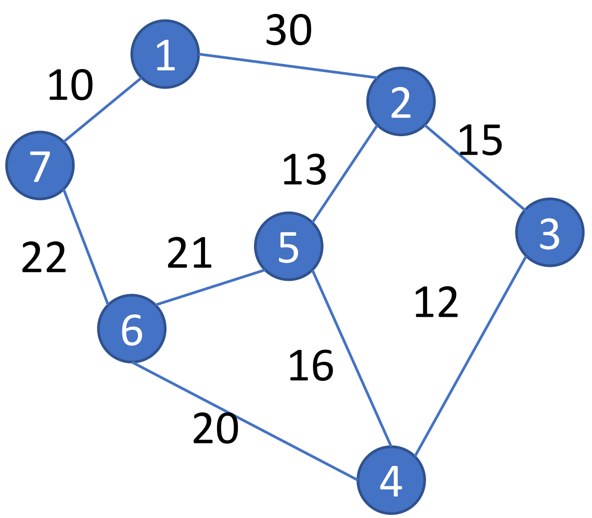
- Kruskal MST Simulator https://www.cs.usfca.edu/~galles/visualization/Kruskal.html

Prim’s Algorithm

- The simplest MST algorithm
- Prim’s algorithm operates much like Dijkstra’s algorithm for finding shortest paths in a graph
- Best MST algorithm for densely-populated graphs
- How does Prim’s Algorithm work?
  - Begin from a starting vertex, N
  - Pick the lowest-cost edge connecting N to another vertex, M
  - Add edge (N, M) to the MST
  - Pick the next lowest-cost edge from either N or M (or any previously-visited nodes) that connects them to an **unvisited** vertex
  - Continue this process until all nodes have been incorporated to the MST

Prim’s Algorithm

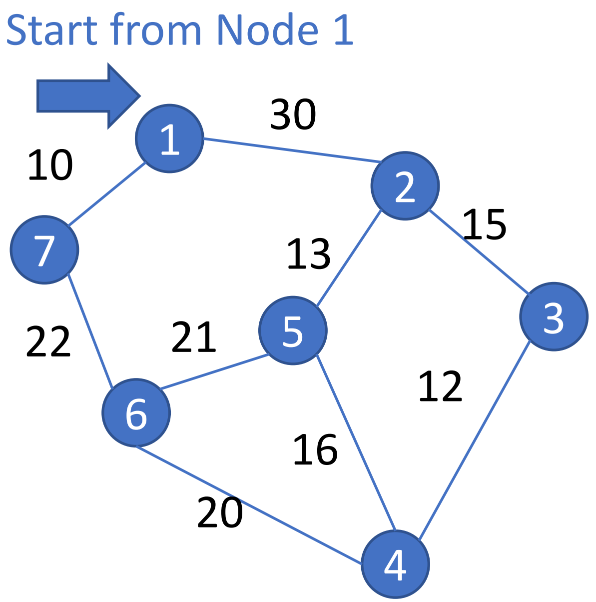
- Always select a node with minimum cost
- Sect select next minimum cost edge, which is connected to selected vertices



Node	In Tree	Distance to Tree	Closet Node in Tree
1	No		
2	No		
3	No		
4	No		
5	No		
6	No		
7	No		

Prim’s Algorithm

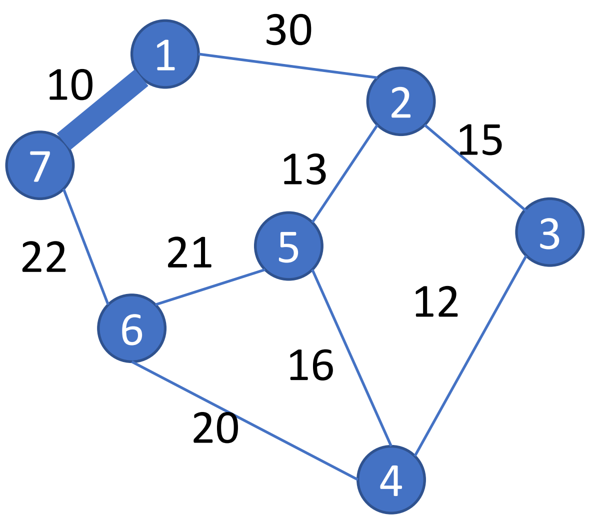
- Select a node to start (better a node with minimum cost edge)
- Select next minimum cost edge which is connected to selected vertices



Node	In Tree	Distance to Tree	Closet Node in Tree
1	Yes	0	-
2	No		
3	No		
4	No		
5	No		
6	No		
7	No		

Prim’s Algorithm

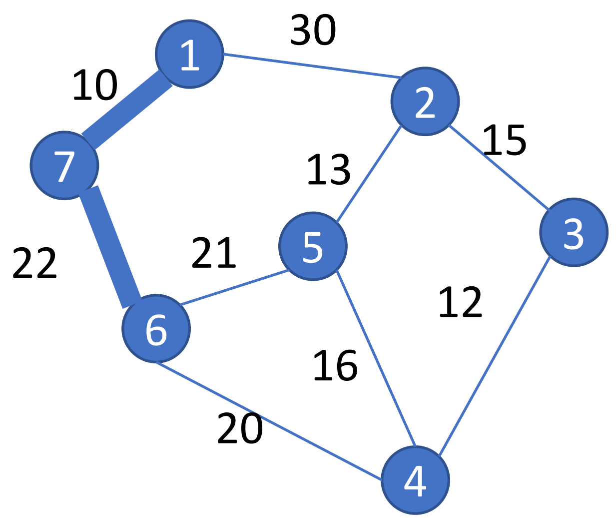
- Select a node to start (better a node with minimum cost edge)
- Select next minimum cost edge which is connected to selected vertices



Node	In Tree	Distance to Tree	Closet Node in Tree
1	Yes	0	1
2	No		
3	No		
4	No		
5	No		
6	No		
7	Yes	10	1

Prim’s Algorithm

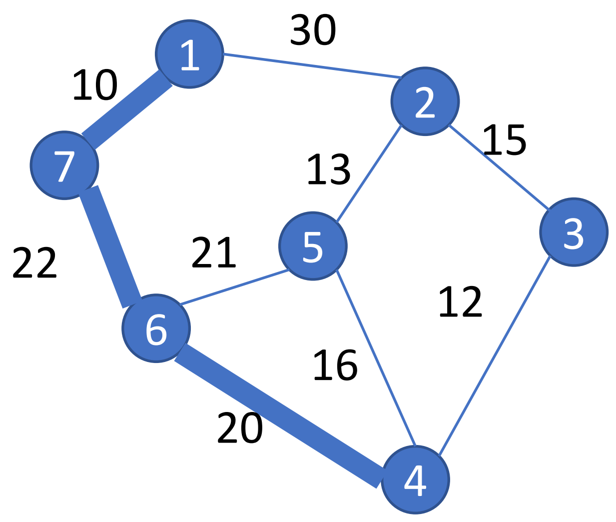
- Select a node to start (better a node with minimum cost edge)
- Select next minimum cost edge which is connected to selected vertices



Node	In Tree	Distance to Tree	Closet Node in Tree
1	Yes	0	1
2	No		
3	No		
4	No		
5	No		
6	Yes	22	7
7	Yes	10	1

Prim’s Algorithm

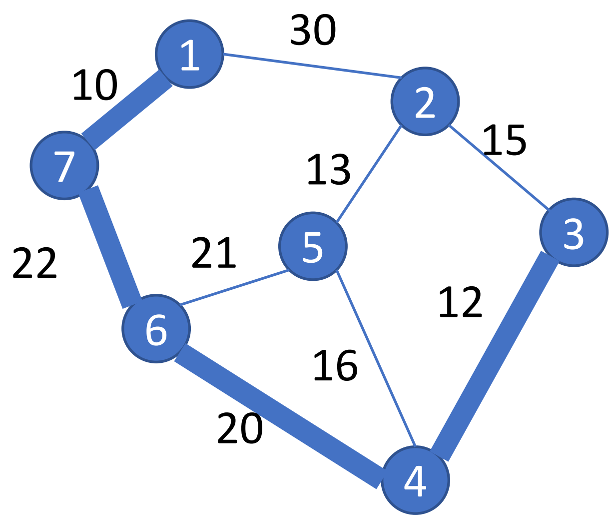
- Select a node to start (better a node with minimum cost edge)
- Select next minimum cost edge which is connected to selected vertices



Node	In Tree	Distance to Tree	Closet Node in Tree
1	Yes	0	1
2	No		
3	No		
4	Yes	20	6
5	No		
6	Yes	22	7
7	Yes	10	1

Prim’s Algorithm

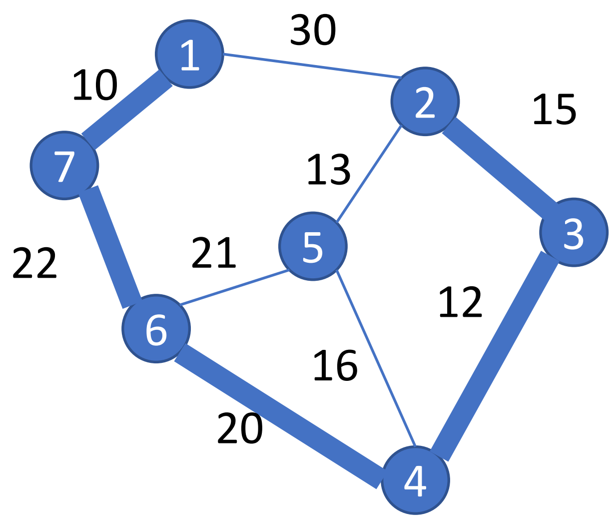
- Select a node to start (better a node with minimum cost edge)
- Select next minimum cost edge which is connected to selected vertices



Node	In Tree	Distance to Tree	Closet Node in Tree
1	Yes	0	1
2	No		
3	Yes	12	4
4	Yes	20	6
5	No		
6	Yes	22	7
7	Yes	10	1

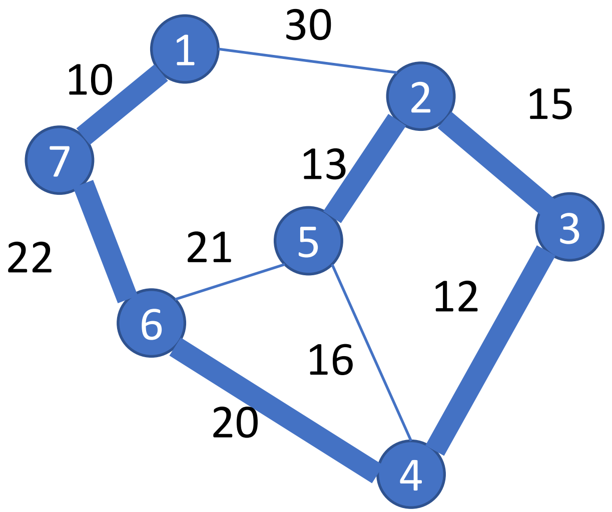
Prim’s Algorithm

- Select a node to start (better a node with minimum cost edge)
- Select next minimum cost edge which is connected to selected vertices



Node	In Tree	Distance to Tree	Closet Node in Tree
1	Yes	0	1
2	Yes	15	3
3	Yes	12	4
4	Yes	20	6
5	No		
6	Yes	22	7
7	Yes	10	1

Prim's Algorithm



- Always select a node with minimum cost
- Select next minimum cost edge which is connected to selected vertices

MST with total weight cost = 92

Node	In Tree	Distance to Tree	Closet Node in Tree
1	Yes	0	1
2	Yes	15	3
3	Yes	12	4
4	Yes	20	6
5	Yes	13	2
6	Yes	22	7
7	Yes	10	1

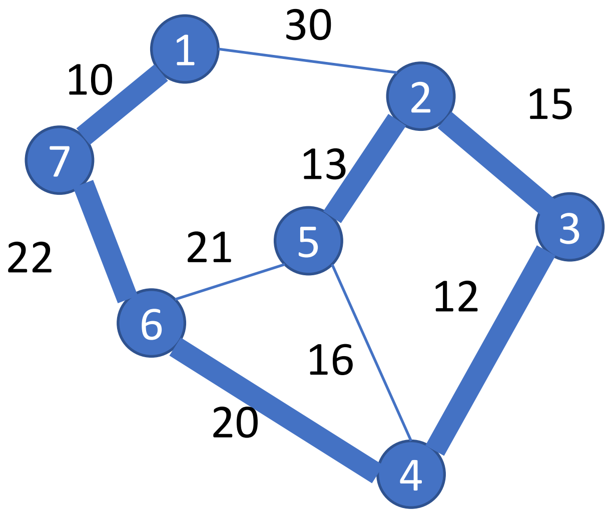
Prim's Algorithm Pseudocode

When the algorithm terminates, the min-priority queue Q is empty; the minimum spanning tree A for G is thus

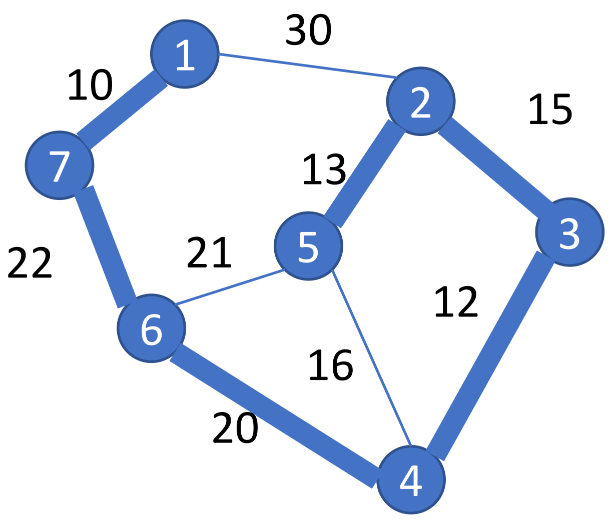
$A = \{(u\ ,\ v.\pi) : v \in V - \{r\}\ \}$
r is the root of the minimum spanning tree

MST-PRIM (G, w, r)
for each $\in$ G.V
    u.key = ∞
    u.π = NIL

r.key = 0
Q $=$ G.V
While Q $\neq$ empty
    u $=$ ExtractMin(Q)
    for each of v $\in$ G.Adj[u]
        if v $\in$ Q and $w(u,\ v)$ < v.key
            v.π = u
            v.key = $w(u,\ v)$



- Prim’s Algorithm complexity



If we implement a priority queue using **binary heap O(E * lg V)**

If we use **Fibonacci heaps**, the running time can be reduced to **O(E + V* lg V)**

- Prim MST Simulator https://www.cs.usfca.edu/~galles/visualization/Prim.html

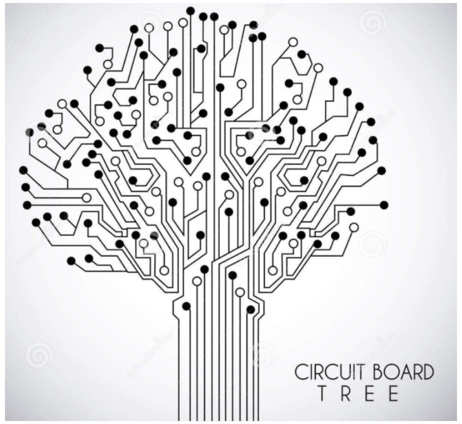
Kruskal’s vs Prim’s Algorithms

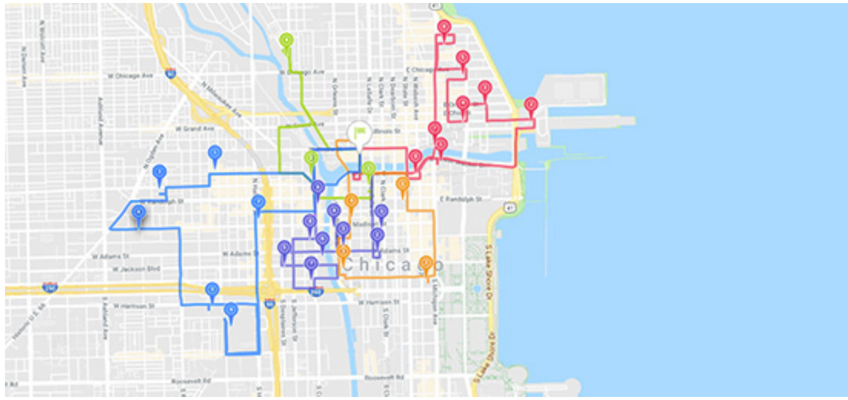
Prim’s Algorithm	Kruskal’s Algorithm
The tree that we are making or growing always remains connected.	The tree that we are making or growing usually remains disconnected.
Prim’s Algorithm grows a solution from a random vertex by adding the next cheapest vertex to the existing tree.	Kruskal’s Algorithm grows a solution from the cheapest edge by adding the next cheapest edge to the existing tree / forest.
Prim’s Algorithm is faster for dense graphs.	Kruskal’s Algorithm is faster for sparse graphs.
O(E + V * log(v))	O (E * log(v))
Requires Priority Queue	Requires Disjoin Set
Harder to implement	Easier to implement

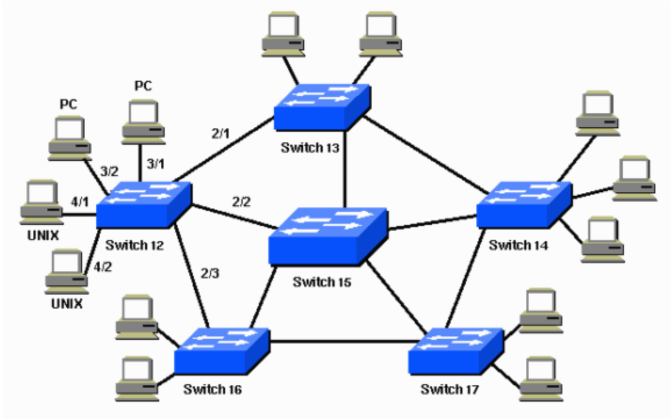
- The Kruskal algorithm is better to use regarding the easier implementation and the best control over the resulting MST. However, Prim’s algorithm offers better complexity.

Minimum Spanning tree Applications

- Applications
  - Computer networks
  - Circuit Design
  - Approximating graphs







Further Readings/References

- 1.P624- 642.Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. 3rd ed. The MIT Press (CLRS Chapter 23)

- Kruskal MST Simulator
  https://www.cs.usfca.edu/~galles/visualization/Kruskal.html

- Prim MST Simulator
  https://www.cs.usfca.edu/~galles/visualization/Prim.html