MINIMUM SPANNING TREES

The greedy method

- A greedy algorithm is an algorithm in which at each stage a locally optimal choice is made.
- A greedy algorithm is therefore one in which no overall strategy is followed, but you simply do whatever looks best at the moment.
- For example a mountain climber using the greedy strategy to climb Everest would at every step climb in the steepest direction.
- From this analogy we get the computational search technique known as hill-climbing.
- In general greedy methods have limited use, but fortunately, the problem of finding a minimum spanning tree can be solved by a greedy method.

Spanning tree

- A subgraph T of an undirected graph G = (V, E) is a spanning tree of G if it is a tree and contains every vertex of G without a cycle.
- Every connected graph has a spanning tree.
- If a graph has n number of vertices then n (n-2) number of spanning tree can be created.
- Example as shown in Figure 1:

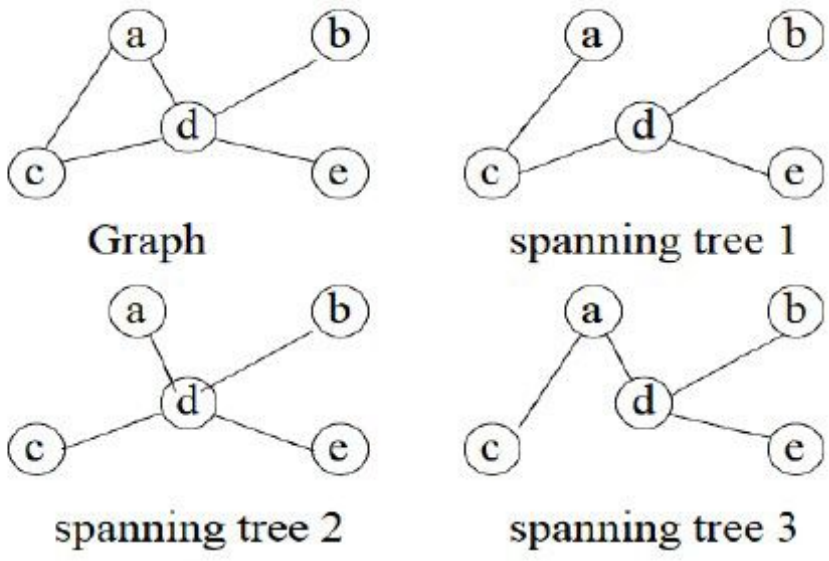


Figure 1: Example of Spanning Tree

Minimum spanning tree (MST)

- A MST in an undirected connected weighted graph is a spanning tree of minimum weight (among all spanning tree).
- The MST may not be unique.
- In other words, A minimum spanning tree in a graph is a subgraph that is:
  1) A spanning subgraph

2) A tree and
3) Has a lower weight than any other spanning tree.

- Example as shown in Figure 2:

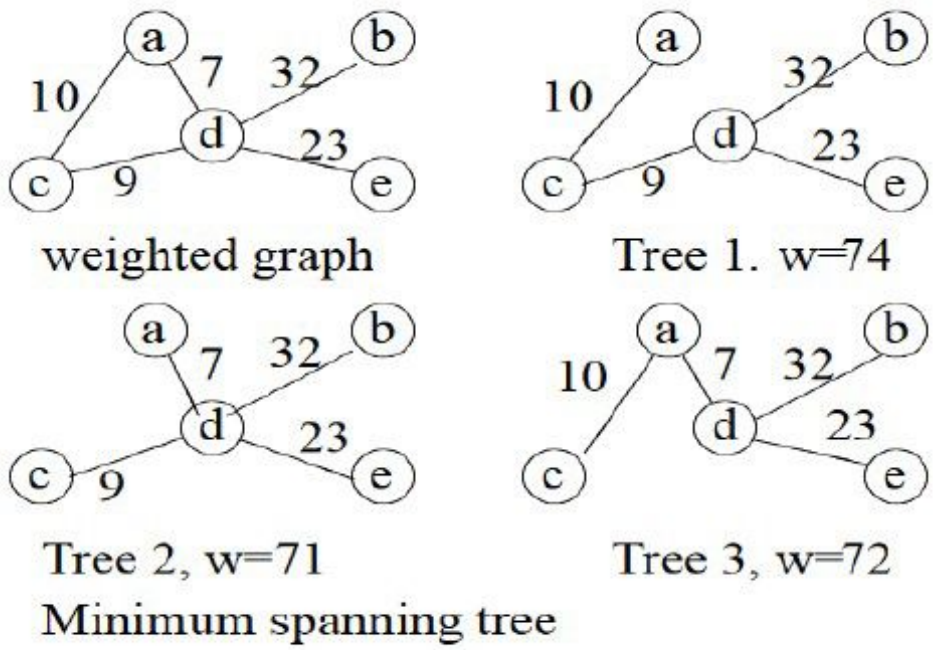


Figure 2: Example of Minimum Spanning Tree

- Consider a group of villages in a remote area that are to be connected by telephone lines. There is a certain cost associated with laying the lines between any pair of villages, depending on their distance apart, the terrain and some pairs just cannot be connected.
- Our task is to find the minimum possible cost in laying lines that will connect all the villages.
- This situation can be modeled by a weighted graph W, in which the weight on each edge is the cost of laying that line.
- It is clear that finding a MST for W is the solution to this problem.

MST algorithms

- The two classical algorithms for computing MST are: Kruskal's and Prim's algorithms.
- They both have same running time complexity, and they are both greedy algorithms, but they use complementary approaches:
  - Kruskal's algorithms: Start with a T consisting of all vertices but no edges. Consider the edges in G in increasing order of weight. Add edge e to T unless doing so would create a cycle.

    This corresponds to growing the tree from a forest of V disconnected vertices. Add one edge at a time, joining two trees in the forest together. Each join reduces the number of connected components in the forest by one.
  - Prim's algorithm: Start with an empty T, and an arbitrary vertex v. Grow T one edge at a time: at each step, add to T the edges of minimum weight that has exactly one endpoint in T. This grows a tree one edge at a time. The current tree is always connected but only covers a part of the vertices.

Kruskal's algorithm

- General idea:
  - Start with a forest consisting of |V| trees, each one consisting of one vertex
  - Consider edges E in increasing order of their weight; add an edge if it connects two trees, ie if it does not create cycles.
- Kruskal's algorithm uses a disjoint-set data structure to maintain several disjoint sets of elements. Each set contains the vertices in one tree of the current forest.
- The operation FIND-SET (u) returns a representative element from the set that contains u. Thus, we can determine whether two vertices u and v belong to the same tree by testing whether FIND-SET (u) equals FIND-SET (v).
- To combine trees, Kruskal's algorithm calls the UNION procedure.
- The complete algorithm for Kruskal's as shown in below.

```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ V[G] {
3       MAKE-SET(v) }
4  sort the edges of E[G] into nondecreasing order by weight w
5  for each edge (u, v) ∈ E[G], taken in nondecreasing order by weight {
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v) }
9  return A
```

Analysis of Kruskal's Algorithm

- The running time of Kruskal's algorithm for a graph G = (V, E) depends on how we implement the disjoint-set data structure.
- Initializing the set A in line 1 takes O (1) time, and the time to sort the edges in line 4 is O(E log E)
- Lines 2–3 take O (V) times to make V number of MAKE-SET operations for all the vertices in the given **for** loop.
- The **for** loop of lines 5–8 performs O (E) FIND-SET and UNION operations on the disjoint-set forest.
- So, Kruskal's Algorithm takes O (E logE) time.
- Assume that G is connected, we have $|E| \geq |V| - 1$. Ignore the constant values. So, O (logV) and O (logE) are same.

- So we can restate the running time of Kruskal's algorithm as O (E logV).

Implementation of Kruskal's Algorithm

Follow the procedure:

Step 1:

- Sort all the edges from low weight to high weight.

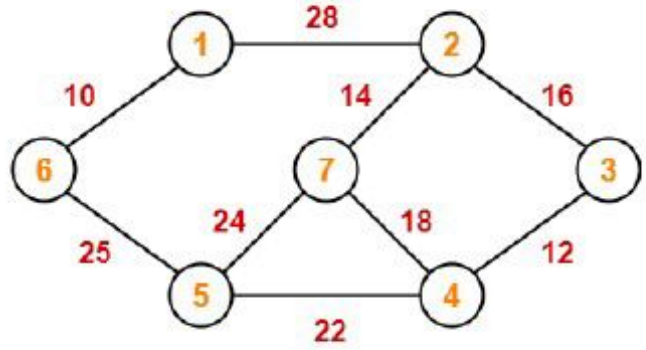
Step 2:

- Take the edge with the lowest weight and use it to connect the vertices of graph.
- If adding an edge creates a cycle, then reject that edge and go for the next least weight edge.

Step 3:

- Keep adding edges until all the vertices are connected and a Minimum Spanning Tree (MST) is obtained.

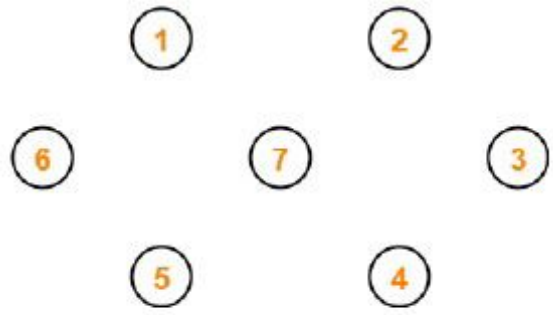
Example: Construct the minimum spanning tree (MST) for the given graph using Kruskal's Algorithm



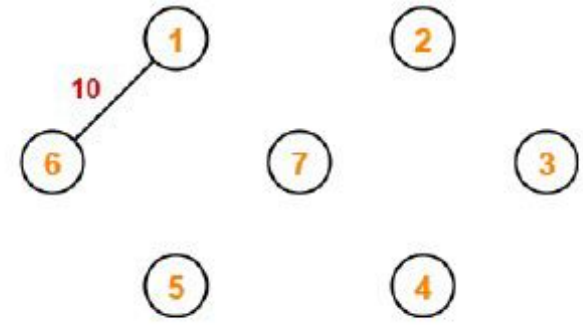
Solution:

- To construct MST using Kruskal's Algorithm,
  - Simply draw all the vertices on the paper.
  - Connect these vertices using edges with minimum weights such that no cycle gets formed.

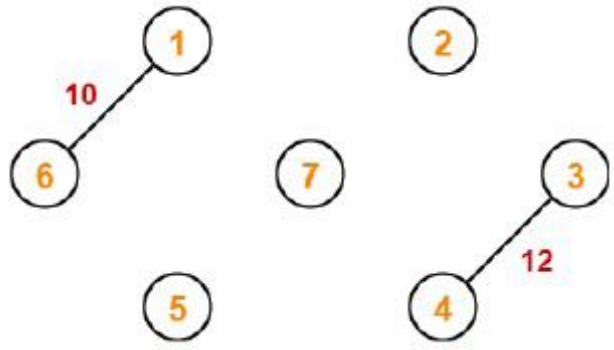
Step 1:



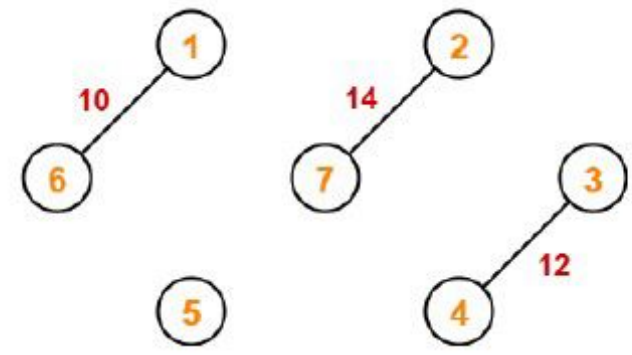
Step 2:



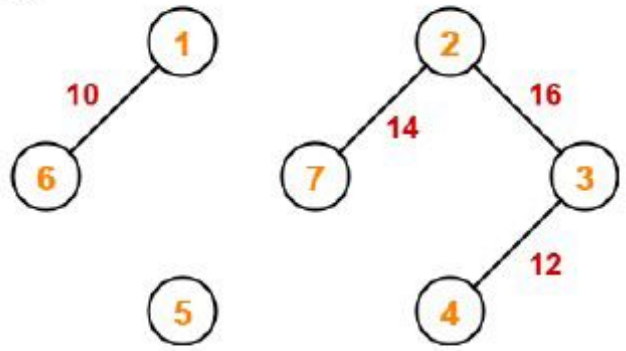
Step 3:



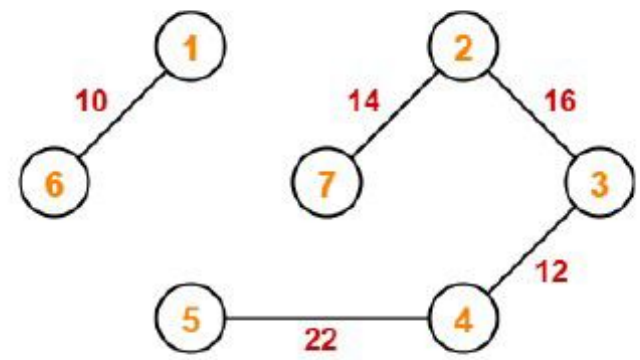
Step 4:



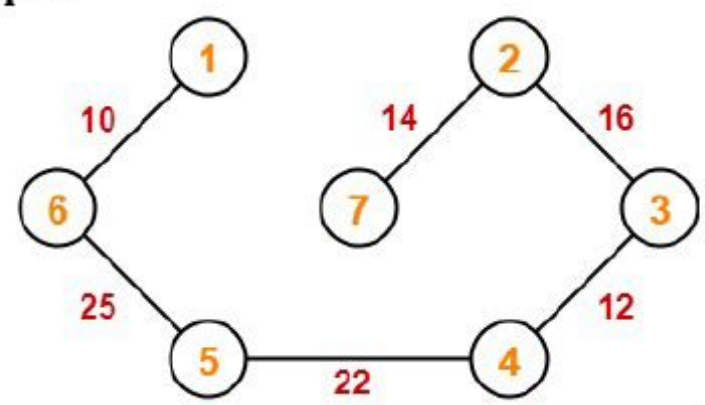
Step 5:



Step 6:



Step 7:



Since all the vertices have been connected / included in the MST, so we stop.

Weight of the MST

= Sum of all edge weights

= 10 + 25 + 22 + 12 + 16 + 14

= **99 units**

Prim's algorithms.

- Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected weighted undirected graph.
- Start by picking any vertex r to be the root of the tree.
- It finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.
- This algorithm is directly based in the MST (minimum spanning tree) property.

```
MST-PRIM(G, w, r)
1.      for each u ∈ V [G] {
2.      do key[u] ← ∞
3.      π[u] ← NIL }
4.      key[r] ← 0
5.      Q ← V [G]
6.      while Q ≠ Ø {
7.              do u ← EXTRACT-MIN(Q)
8.              for each v ∈ Adj[u] {
9.                      do if v ∈ Q and w(u, v) < key[v]{
10.                             then π[v] ← u
11.                     key[v] ← w (u, v)} } }
```

- The running time of Prim's algorithm depends on how we implement the min-priority queue Q.
- Lines 1–5 perform in O (V) time.
- The while loop executes $|V|$ times, and since each EXTRACT-MIN operation takes O (log V) time, the total time for all calls to EXTRACT-MIN is O (V log V).
- The for loop in lines 8–11 executes O (E) times altogether.
- Within in the **for** loop, in line 9 in constant time.
- The assignment in line 11 involves an implicit DECREASE-KEY operation on the min-heap, which a binary min-heap supports in O (log V) time.
- Thus, the total time for Prim's algorithm is O (V log V + E log V) = O (E log V), which is asymptotically the same as for our implementation of Kruskal's algorithm.

Implementation of Prim's Algorithm

Follow the procedure:

Step 1:

- Randomly choose any vertex.
- The vertex connecting to the edge having least weight is usually selected.

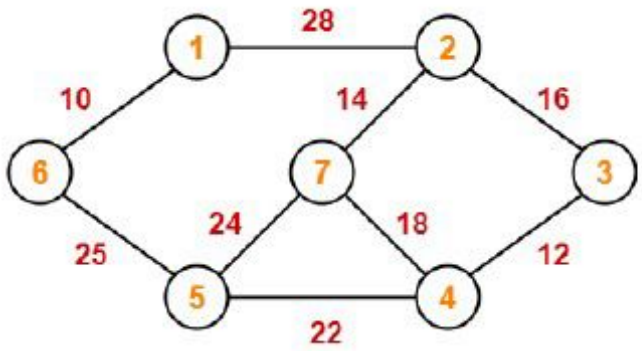
Step 2:

- Find all the edges that connect the tree to new vertices.
- Find the least weight edge among those edges and include it in the existing tree.
- If including that edge creates a cycle, then reject that edge and look for the next least weight edge.

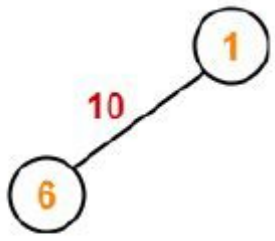
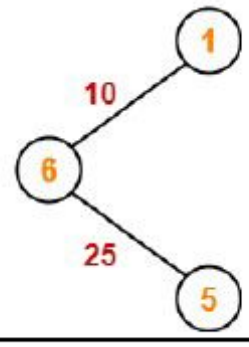
Step 3:

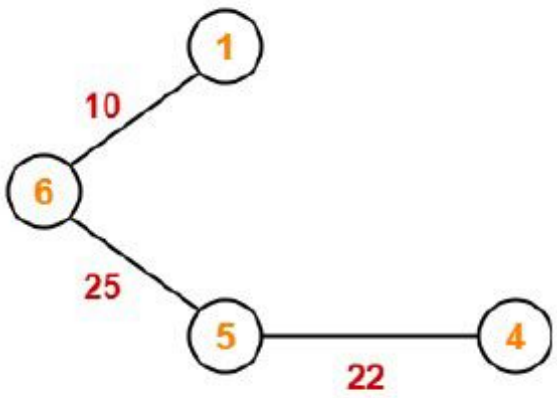
- Keep repeating step-02 until all the vertices are included and Minimum Spanning Tree (MST) is obtained.

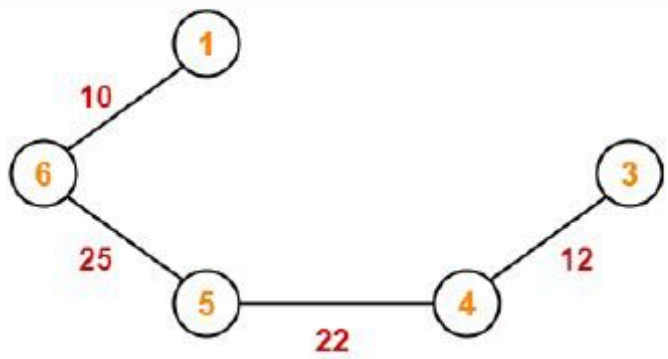
Example: Construct the minimum spanning tree (MST) for the given graph using Prim's Algorithm.



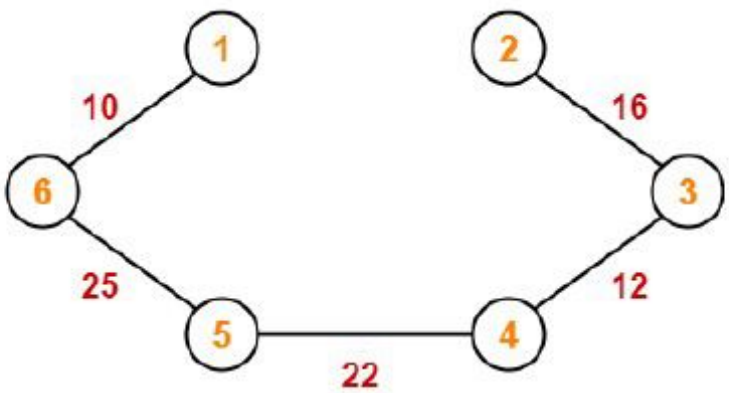
Solution:

Step 1:	**Step 2:**
	
Step 3:	**Step 4:**

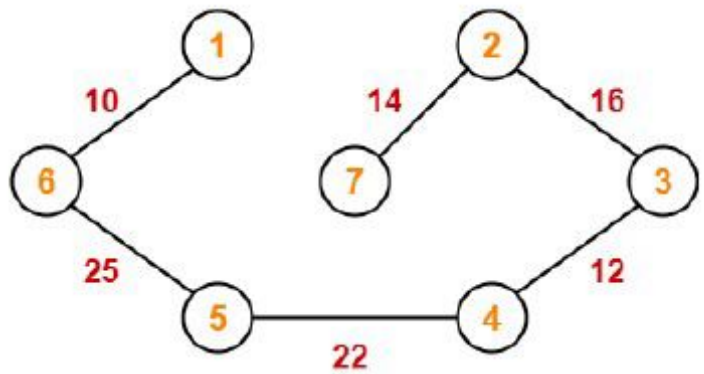




Step 5:



Step 6:



Since all the vertices have been included in the MST, so we stop.

Now, Cost of Minimum Spanning Tree

= Sum of all edge weights

= 10 + 25 + 22 + 12 + 16 + 14

= **99 units**