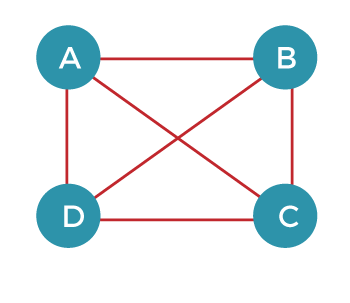
**2.1.2 Minimum Spanning Tree- Prim**

## Minimum Spanning Tree

Before knowing about the minimum spanning tree, we should know about the spanning tree.

**To understand the concept of spanning tree, consider the below graph:**



The above graph can be represented as G(V, E), where 'V' is the number of vertices, and 'E' is the number of edges. The spanning tree of the above graph would be represented as G`(V`, E`). In this case, V` = V means that the number of vertices in the spanning tree would be the same as the number of vertices in the graph, but the number of edges would be different. The number of edges in the spanning tree is the subset of the number of edges in the original graph. Therefore, the number of edges can be written as:

E` € E

It can also be written as:

E` = |V| - 1

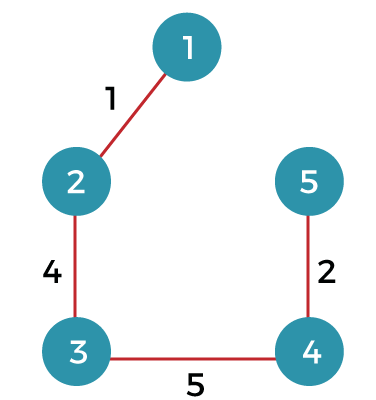
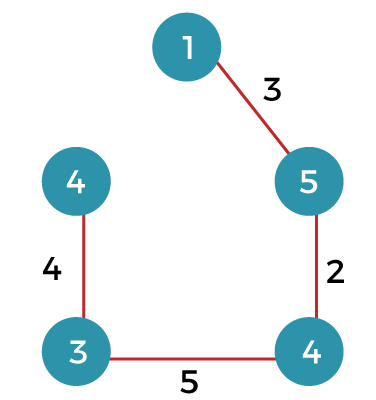
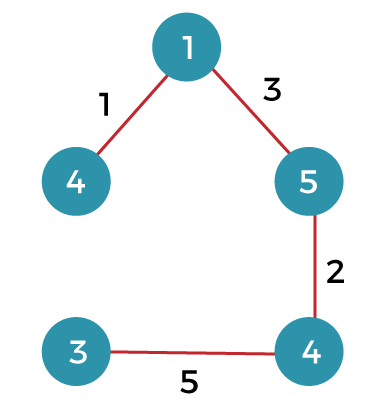
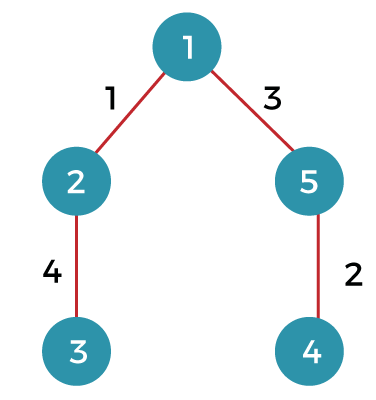
Two conditions exist in the spanning tree, which is as follows:

* The number of vertices in the spanning tree would be the same as the number of vertices in the original graph.  
  **V` = V**
* The number of edges in the spanning tree would be equal to the number of edges minus 1.  
  **E` = |V| - 1**
* The spanning tree should not contain any cycle.
* The spanning tree should not be disconnected.

**Note: A graph can have more than one spanning tree.**

Consider the below graph:

The above graph contains 5 vertices. As we know, the vertices in the spanning tree would be the same as the graph; therefore, V` is equal 5. The number of edges in the spanning tree would be equal to (5 - 1), i.e., 4. The following are the possible spanning trees:

**What is a minimum spanning tree?**

The minimum spanning tree is a spanning tree whose sum of the edges is minimum. Consider the below graph that contains the edge weight:

**The following are the spanning trees that we can make from the above graph.**

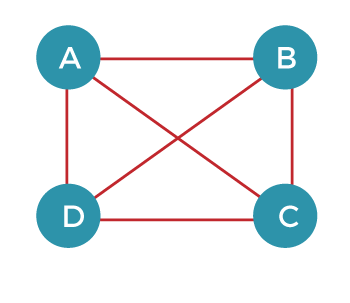
* The first spanning tree is a tree in which we have removed the edge between the vertices 1 and 5 shown as below:  
  The sum of the edges of the above tree is (1 + 4 + 5 + 2): 12
* The second spanning tree is a tree in which we have removed the edge between the vertices 1 and 2 shown as below:  
  The sum of the edges of the above tree is (3 + 2 + 5 + 4) : 14
* The third spanning tree is a tree in which we have removed the edge between the vertices 2 and 3 shown as below:  
  The sum of the edges of the above tree is (1 + 3 + 2 + 5) : 11
* The fourth spanning tree is a tree in which we have removed the edge between the vertices 3 and 4 shown as below:  
  The sum of the edges of the above tree is (1 + 3 + 2 + 4) : 10. The edge cost 10 is minimum so it is a minimum spanning tree.

General properties of minimum spanning tree:

* If we remove any edge from the spanning tree, then it becomes disconnected. Therefore, we cannot remove any edge from the spanning tree.
* If we add an edge to the spanning tree then it creates a loop. Therefore, we cannot add any edge to the spanning tree.
* In a graph, each edge has a distinct weight, then there exists only a single and unique minimum spanning tree. If the edge weight is not distinct, then there can be more than one minimum spanning tree.
* A complete undirected graph can have an nn-2 number of spanning trees.
* Every connected and undirected graph contains atleast one spanning tree.
* The disconnected graph does not have any spanning tree.
* In a complete graph, we can remove maximum (e-n+1) edges to construct a spanning tree.

**Let's understand the last property through an example.**

Consider the complete graph which is given below:



The number of spanning trees that can be made from the above complete graph equals to nn-2 = 44-2 = 16.

Therefore, 16 spanning trees can be created from the above graph.

The maximum number of edges that can be removed to construct a spanning tree equals to e-n+1 = 6 - 4 + 1 = 3.

**Prim's Algorithm**

It is a greedy algorithm. It starts with an empty spanning tree. The idea is to maintain two sets of vertices:

* Contain vertices already included in MST.
* Contain vertices not yet included.

At every step, it considers all the edges and picks the minimum weight edge. After picking the edge, it moves the other endpoint of edge to set containing MST.

**Steps for finding MST using Prim's Algorithm:**

1. Create MST set that keeps track of vertices already included in MST.
2. Assign key values to all vertices in the input graph. Initialize all key values as INFINITE (∞). Assign key values like 0 for the first vertex so that it is picked first.
3. While MST set doesn't include all vertices.
   1. Pick vertex u which is not is MST set and has minimum key value. Include 'u'to MST set.
   2. Update the key value of all adjacent vertices of u. To update, iterate through all adjacent vertices. For every adjacent vertex v, if the weight of edge u.v less than the previous key value of v, update key value as a weight of u.v.

**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 ? ∅

7. do u ← EXTRACT - MIN (Q)

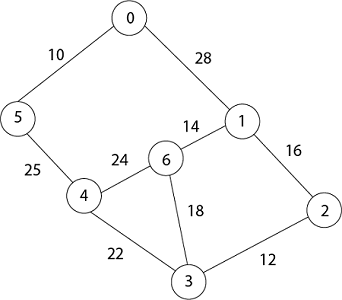
8. for each v ∈ Adj [u]

9. do if v ∈ Q and w (u, v) < key [v]

10. then π [v] ← u

11. key [v] ← w (u, v)

**Example:** Generate minimum cost spanning tree for the following graph using Prim's algorithm.

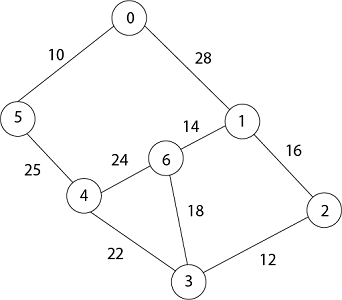


**Solution:** In Prim's algorithm, first we initialize the priority Queue Q. to contain all the vertices and the key of each vertex to ∞ except for the root, whose key is set to 0. Suppose 0 vertex is the root, i.e., r. By EXTRACT - MIN (Q) procure, now u = r and Adj [u] = {5, 1}.

Removing u from set Q and adds it to set V - Q of vertices in the tree. Now, update the key and π fields of every vertex v adjacent to u but not in a tree.

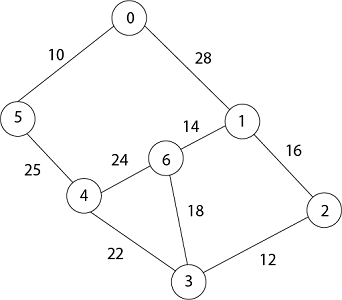


1. Taking 0 as starting vertex
2. Root = 0
3. Adj [0] = 5, 1
4. Parent, π [5] = 0 and π [1] = 0
5. Key [5] = ∞ and key [1] = ∞
6. w [0, 5) = 10  and w (0,1) = 28
7. w (u, v) < key [5] , w (u, v) < key [1]
8. Key [5] = 10 and key [1] = 28
9. So update key value of 5 and 1 is:

Now by EXTRACT\_MIN (Q) Removes 5 because key [5] = 10 which is minimum so u = 5.

1. Adj [5] = {0, 4} and 0 is already in heap
2. Taking 4, key [4] = ∞      π [4] = 5
3. (u, v) < key [v] then key [4] = 25
4. w (5,4) = 25
5. w (5,4) < key [4]
6. date key value and parent of 4.

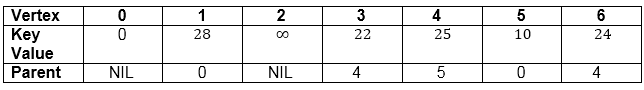
Now remove 4 because key [4] = 25 which is minimum, so u =4

1. Adj [4] = {6, 3}
2. Key [3] = ∞         key [6] = ∞
3. w (4,3) = 22        w (4,6) = 24
4. w (u, v) < key [v]    w (u, v) < key [v]
5. w (4,3) < key [3]      w (4,6) < key [6]

Update key value of key [3] as 22 and key [6] as 24.

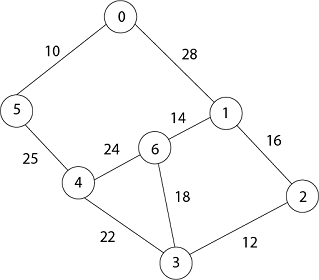
And the parent of 3, 6 as 4.

1. π[3]= 4       π[6]= 4



1. u = EXTRACT\_MIN (3, 6)            [key [3] < key [6]]
2. u = 3              i.e.  22 < 24

Now remove 3 because key [3] = 22 is minimum so u =3.

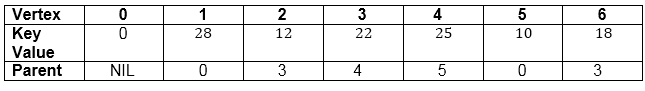


1. Adj [3] = {4, 6, 2}
2. 4 is already in heap
3. 4 ≠ Q key [6] = 24 now becomes key [6] = 18
4. Key [2] = ∞            key [6] = 24
5. w (3, 2) = 12          w (3, 6) = 18
6. w (3, 2) < key [2]         w (3, 6) < key [6]

Now in Q, key [2] = 12, key [6] = 18, key [1] = 28 and parent of 2 and 6 is 3.

1. π [2] = 3      π[6]=3

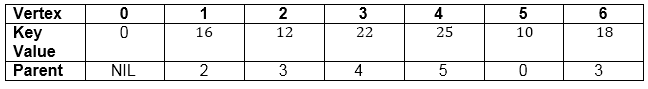
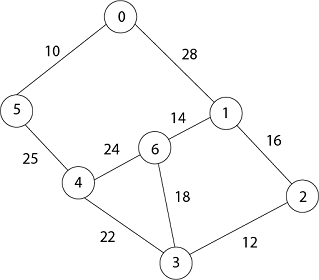
Now by EXTRACT\_MIN (Q) Removes 2, because key [2] = 12 is minimum.



1. u = EXTRACT\_MIN (2, 6)
2. u = 2          [key [2] < key [6]]
3. 12 < 18
4. Now the root is 2
5. Adj [2] = {3, 1}
6. 3 is already in a heap
7. Taking 1, key [1] = 28
8. w (2,1) = 16
9. w (2,1) < key [1]

So update key value of key [1] as 16 and its parent as 2.

1. π[1]= 2

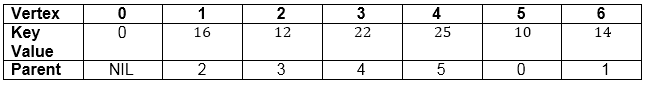
  


Now by EXTRACT\_MIN (Q) Removes 1 because key [1] = 16 is minimum.

1. Adj [1] = {0, 6, 2}
2. 0 and 2 are already in heap.
3. Taking 6, key [6] = 18
4. w [1, 6] = 14
5. w [1, 6] < key [6]

Update key value of 6 as 14 and its parent as 1.

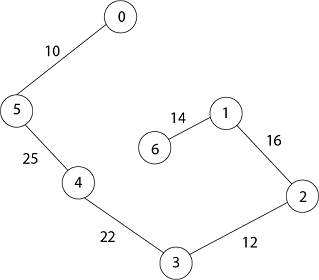
1. Π [6] = 1



Now all the vertices have been spanned, Using above the table we get Minimum Spanning Tree.

1. 0 → 5 → 4 → 3 → 2 → 1 → 6
2. [Because Π [5] = 0, Π [4] = 5, Π [3] = 4, Π [2] = 3, Π [1] =2, Π [6] =1]

**Thus the final spanning Tree is**



**Total Cost = 10 + 25 + 22 + 12 + 16 + 14 = 99**