**Kruskal’s Algorithm-**

* Kruskal’s Algorithm is a famous greedy algorithm.
* It is used for finding the Minimum Spanning Tree (MST) of a given graph.
* To apply Kruskal’s algorithm, the given graph must be weighted, connected and undirected.

**Kruskal’s Algorithm Implementation-**

The implementation of Kruskal’s Algorithm is explained in the following steps-

**Step-01:**

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

**Step-02:**

* 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-03:**

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

|  |
| --- |
| **Thumb Rule to Remember**    The above steps may be reduced to the following thumb rule-   * Simply draw all the vertices on the paper. * Connect these vertices using edges with minimum weights such that no cycle gets formed. |

**Kruskal’s Algorithm Time Complexity-**

|  |
| --- |
| Worst case time complexity of Kruskal’s Algorithm  = O(ElogV) or O(ElogE) |

**Analysis-**

* The edges are maintained as min heap.
* The next edge can be obtained in O(logE) time if graph has E edges.
* Reconstruction of heap takes O(E) time.
* So, Kruskal’s Algorithm takes O(ElogE) time.
* The value of E can be at most O(V2).
* So, O(logV) and O(logE) are same.

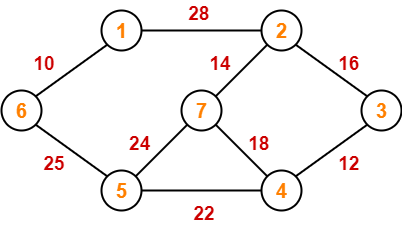
**Special Case-**

* If the edges are already sorted, then there is no need to construct min heap.
* So, deletion from min heap time is saved.
* In this case, time complexity of Kruskal’s Algorithm = O(E + V)

**PRACTICE PROBLEMS BASED ON KRUSKAL’S ALGORITHM-**

**Problem-01:**

Construct the minimum spanning tree (MST) for the given graph using Kruskal’s Algorithm-

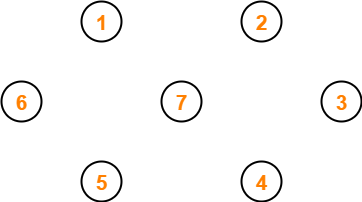


**Solution-**

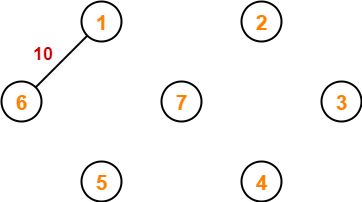
To construct MST using Kruskal’s Algorithm,

* Simply draw all the vertices on the paper.
* Connect these vertices using edges with minimum weights such that no cycle gets formed.

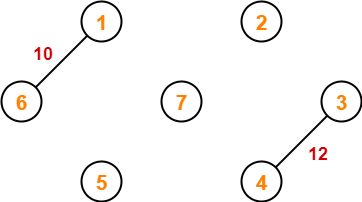
**Step-01:**



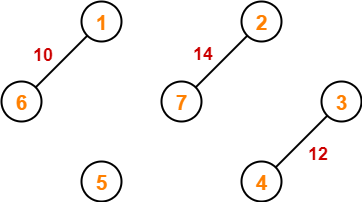
**Step-02:**



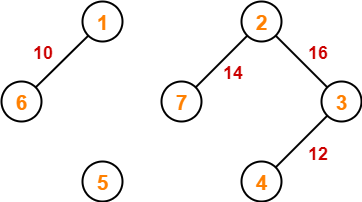
**Step-03:**



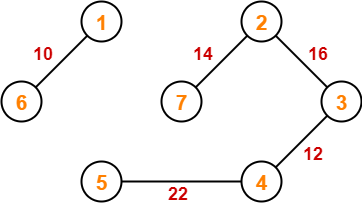
**Step-04:**



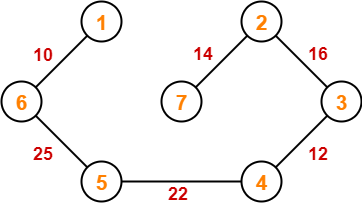
**Step-05:**



**Step-06:**



**Step-07:**



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

Weight of the MST

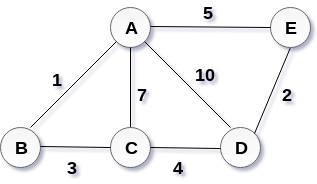
= Sum of all edge weights

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

= 99 units

### Example :

**Apply the Kruskal's algorithm on the graph given as follows.**



## Solution:

the weight of the edges given as :

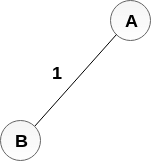
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Edge** | **AE** | **AD** | **AC** | **AB** | **BC** | **CD** | **DE** |
| Weight | 5 | 10 | 7 | 1 | 3 | 4 | 2 |

Sort the edges according to their weights.

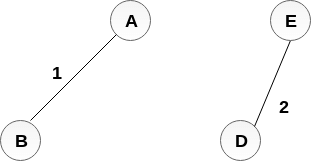
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Edge** | **AB** | **DE** | **BC** | **CD** | **AE** | **AC** | **AD** |
| Weight | 1 | 2 | 3 | 4 | 5 | 7 | 10 |

Start constructing the tree;

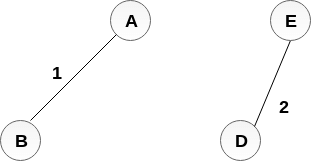
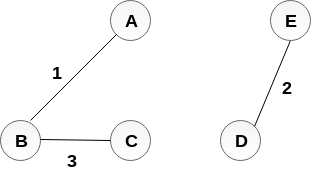
Add AB to the MST;



Add DE to the MST;



Add BC to the MST;

The next step is to add AE, but we can't add that as it will cause a cycle.

The next edge to be added is AC, but it can't be added as it will cause a cycle.

The next edge to be added is AD, but it can't be added as it will contain a cycle.

Hence, the final MST is the one which is shown in the step 4.

the cost of MST = 1 + 2 + 3 + 4 = 10.

**Prim’s Algorithm-**

* Prim’s Algorithm is a famous greedy algorithm.
* It is used for finding the Minimum Spanning Tree (MST) of a given graph.
* To apply Prim’s algorithm, the given graph must be weighted, connected and undirected.

**Prim’s Algorithm Implementation-**

The implementation of Prim’s Algorithm is explained in the following steps-

**Step-01:**

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

**Step-02:**

* 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-03:**

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

**Prim’s Algorithm Time Complexity-**

Worst case time complexity of Prim’s Algorithm is-

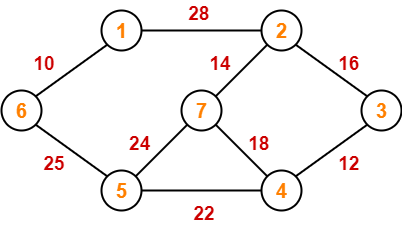
* O(ElogV) using binary heap
* O(E + VlogV) using Fibonacci heap

|  |
| --- |
| **Time Complexity Analysis**     * If adjacency list is used to represent the graph, then using breadth first search, all the vertices can be traversed in O(V + E) time. * We traverse all the vertices of graph using breadth first search and use a min heap for storing the vertices not yet included in the MST. * To get the minimum weight edge, we use min heap as a priority queue. * Min heap operations like extracting minimum element and decreasing key value takes O(logV) time.     So, overall time complexity  = O(E + V) x O(logV)  = O((E + V)logV)  = O(ElogV)    This time complexity can be improved and reduced to O(E + VlogV) using Fibonacci heap. |

**PRACTICE PROBLEMS BASED ON PRIM’S ALGORITHM-**

**Problem-01:**

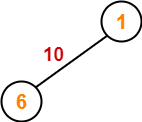
Construct the minimum spanning tree (MST) for the given graph using Prim’s Algorithm-



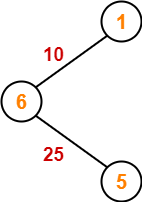
**Solution-**

The above discussed steps are followed to find the minimum cost spanning tree using Prim’s Algorithm-

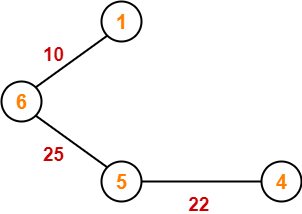
**Step-01:**



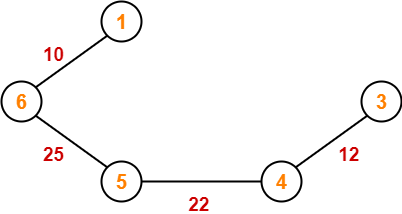
**Step-02:**



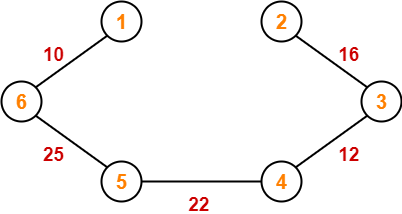
**Step-03:**



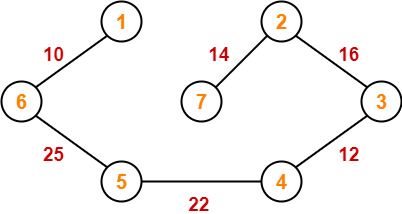
**Step-04:**



**Step-05:**



**Step-06:**



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

Now, Cost of Minimum Spanning Tree

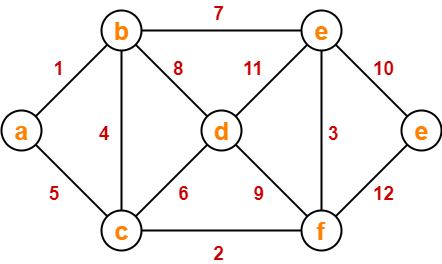
= Sum of all edge weights

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

= 99 units

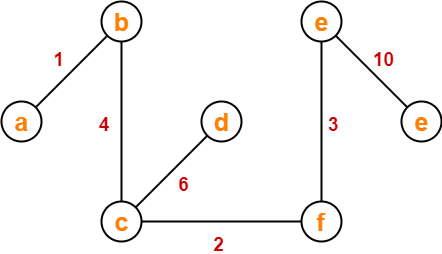
**Problem-02:**

Using Prim’s Algorithm, find the cost of minimum spanning tree (MST) of the given graph-



**Solution-**

The minimum spanning tree obtained by the application of Prim’s Algorithm on the given graph is as shown below-



Now, Cost of Minimum Spanning Tree

= Sum of all edge weights

= 1 + 4 + 2 + 6 + 3 + 10

= 26 units