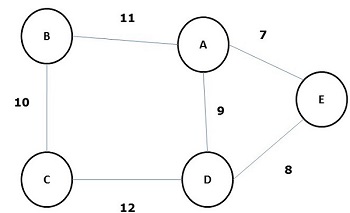
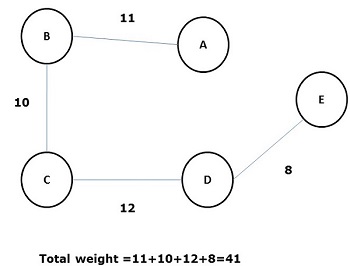
## Minimal Spanning Tree

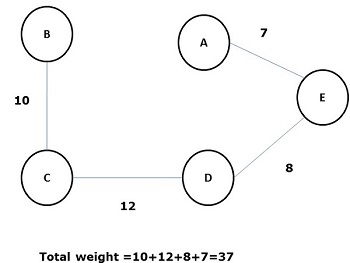
(**Prim’s & Kruskal’s Algorithm**)

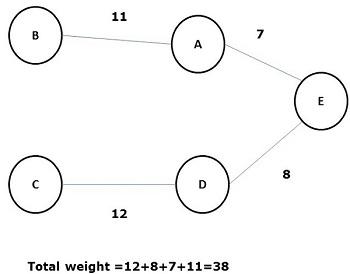
A spanning tree whose sum of weight (or length) of all its edges is less than all other possible spanning tree of graph G is known as a **minimal spanning tree** or **minimum cost spanning** tree. The following figure shows a weighted connected graph.

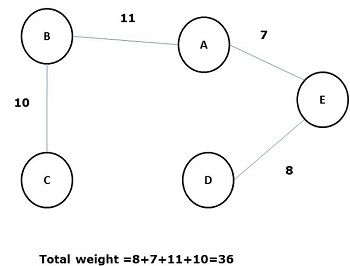


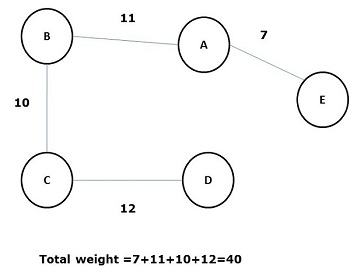
Some possible spanning trees of the above graph are shown below −

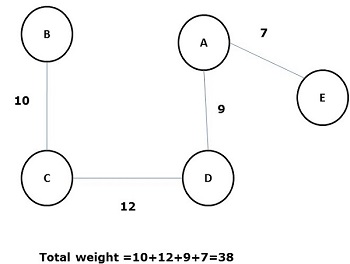


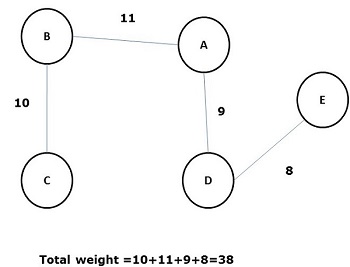












Among all the above spanning trees, figure (5) is the minimum spanning tree. The concept of minimum cost spanning tree is applied in travelling salesman problem, designing electronic circuits, Designing efficient networks, and designing efficient routing algorithms.

To implement the minimum cost-spanning tree, the following two methods are used −

* Prim’s Algorithm
* Kruskal’s Algorithm

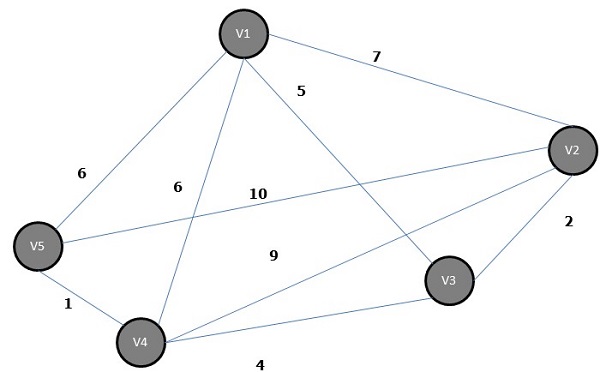
## Prim's Algorithm

Prim’s algorithm is a greedy algorithm, which helps us find the minimum spanning tree for a weighted **undirected** graph. It selects

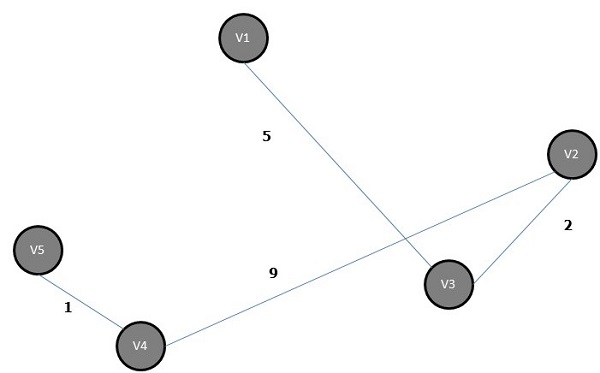
* A vertex first and
* Finds an edge with the lowest weight incident on that vertex.

### Steps of Prim’s Algorithm

* Select any **vertex**, say v1 of Graph G.
* Select an edge, say e1 of G such that e1 = v1 v2 and v1 ≠ v2 and e1 has minimum weight among the edges incident on v1 in graph G.
* Now, following step 2, select the minimum weighted edge incident on v2.
* Continue this till n–1 edges have been chosen. Here **n** is the number of vertices.



The minimum spanning tree is −

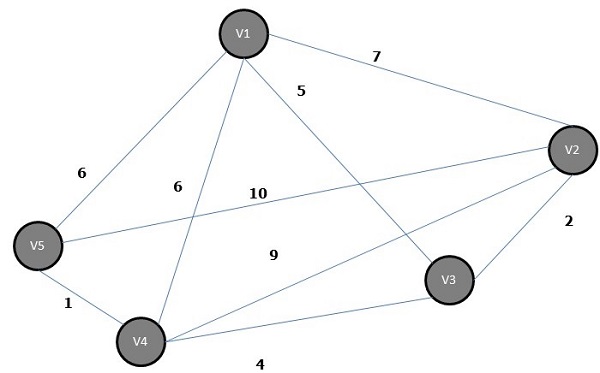


## Kruskal's Algorithm

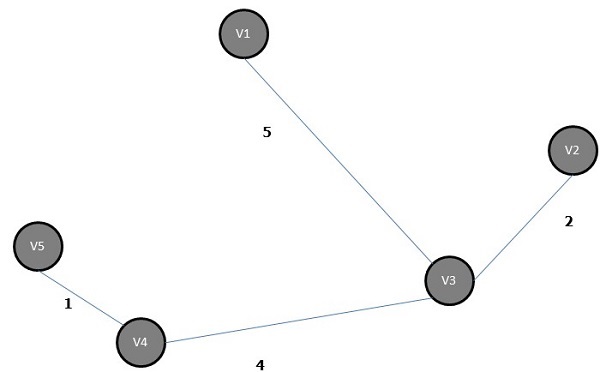
Kruskal’s algorithm is a greedy algorithm, which helps us to find the minimum spanning tree for a connected weighted graph, adding increasing cost arcs at each step. It is a minimum-spanning-tree algorithm that finds an edge of the least possible weight that connects any two trees in the forest.

### Steps of Kruskal’s Algorithm

* Select an **edge** of minimum weight; say e1 of Graph G and e1 is not a loop.
* Select the next minimum weighted edge connected to e1.
* Continue this till n–1 edges have been chosen. Here **n** is the number of vertices.



The minimum spanning tree of the above graph is −



# Prim's Spanning Tree Algorithm

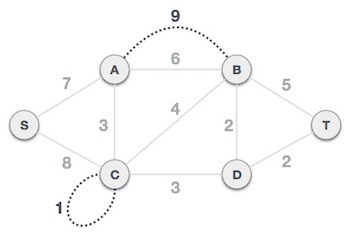
Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we shall use the same example −



## Step 1 - Remove all loops and parallel edges



Remove all loops and parallel edges from the given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.

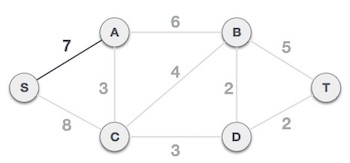


## Step 2 - Choose any arbitrary node as root node

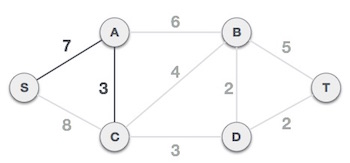
In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node.

## Step 3 - Check outgoing edges and select the one with less cost

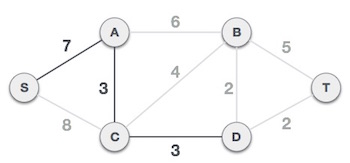
After choosing the root node **S**, we see that S, A and S, C are two edges with weight 7 and 8, respectively. We choose the edge S, A as it is lesser than the other.



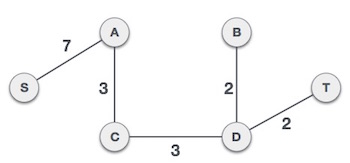
Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



After this step, S-7-A-3-C tree is formed. Now we'll again treat it as a node and will check all the edges again. However, we will choose only the least cost edge. In this case, C-3-D is the new edge, which is less than other edges' cost 8, 6, 4, etc.



After adding node **D** to the spanning tree, we now have two edges going out of it having the same cost, i.e. D-2-T and D-2-B. Thus, we can add either one. But the next step will again yield edge 2 as the least cost. Hence, we are showing a spanning tree with both edges included.



We may find that the output spanning tree of the same graph using two different algorithms is same.

# Kruskal's Spanning Tree Algorithm

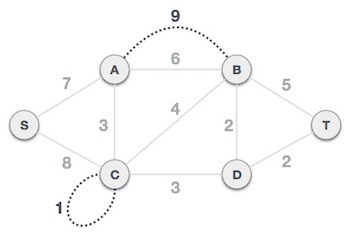
Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

To understand Kruskal's algorithm let us consider the following example −



## Step 1 - Remove all loops and Parallel Edges

Remove all loops and parallel edges from the given graph.

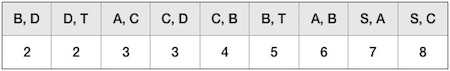


In case of parallel edges, keep the one which has the least cost associated and remove all others.



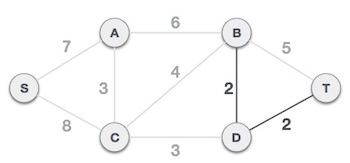
## Step 2 - Arrange all edges in their increasing order of weight

The next step is to create a set of edges and weight, and arrange them in an ascending order of weightage (cost).



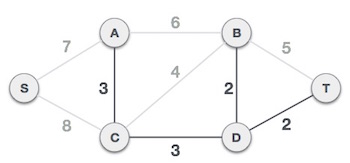
## Step 3 - Add the edge which has the least weightage

Now we start adding edges to the graph beginning from the one which has the least weight. Throughout, we shall keep checking that the spanning properties remain intact. In case, by adding one edge, the spanning tree property does not hold then we shall consider not to include the edge in the graph.

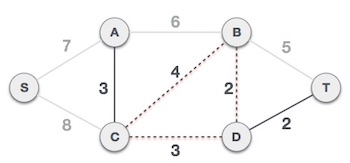


The least cost is 2 and edges involved are B,D and D,T. We add them. Adding them does not violate spanning tree properties, so we continue to our next edge selection.

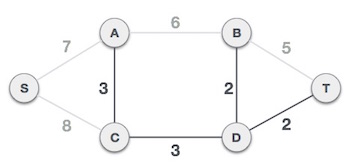
Next cost is 3, and associated edges are A,C and C,D. We add them again −



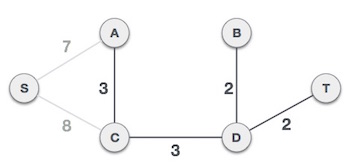
Next cost in the table is 4, and we observe that adding it will create a circuit/cycle in the graph. −



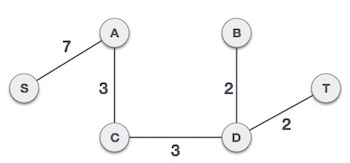
We ignore it. In the process we shall ignore/avoid all edges that create a circuit/cycle.



We observe that edges with cost 5 and 6 also create circuits. We ignore them and move on.



Now we are left with only one node to be added. Between the two least cost edges available 7 and 8, we shall add the edge with cost 7.



By adding edge S, A we have included all the nodes of the graph and we now have minimum cost spanning tree.