Prim's Spanning Tree Algorithm

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

Prim's algorithm is use to find minimum cost spanning tree for a weighted undirected graph.

A **spanning tree** of a graph is just a subgraph that contains all the vertices and is a tree with no cycle. A graph may have many spanning trees.

A spanning tree of a graph with minimum cost is called '**minimum cost spanning tree'**.

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

Prim's Algorithm:

Step 1: Remove all the loops (Any edge that starts and ends at the same vertex is loop)

Step 2: Remove all parallel edges between two vertices except the one with least weight.

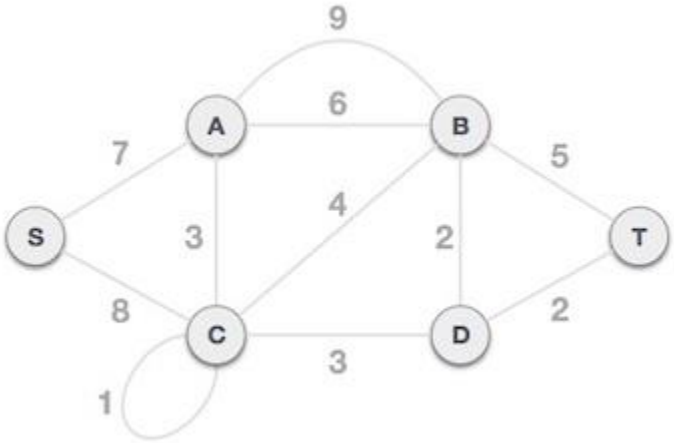
Step 3 - Choose any arbitrary vertex as root vertex.

(for all vertices repeat step 4)

Step 4 - Check outgoing edges and select the one with less cost and with no cycle.

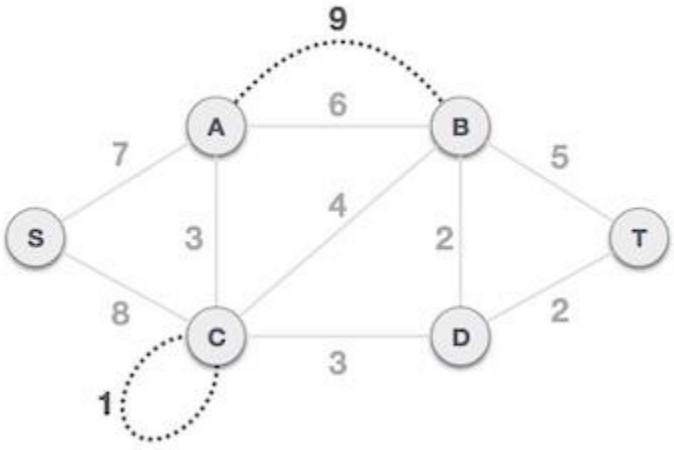
Let’s take an example to understand Prim's algorithm better.

Find the minimum cost spanning tree from the following graph:

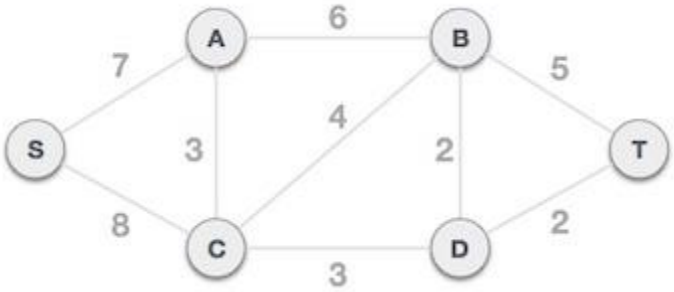


Step 1 and Step 2 –

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 –

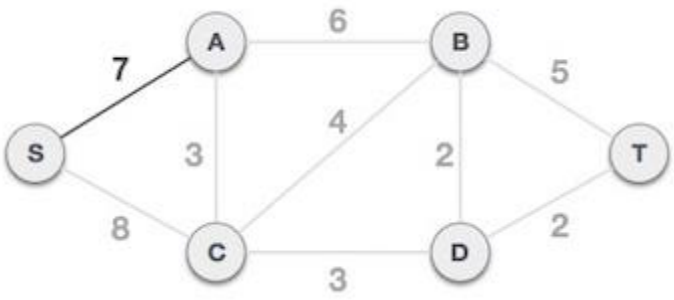
Choose any arbitrary node as root node

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.

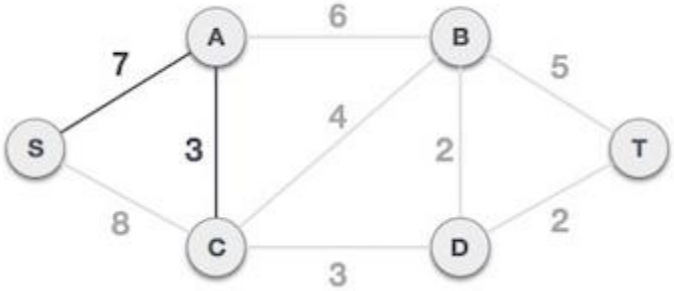
Step 3 –

Check outgoing edges and select the one with less cost and with no cycle.

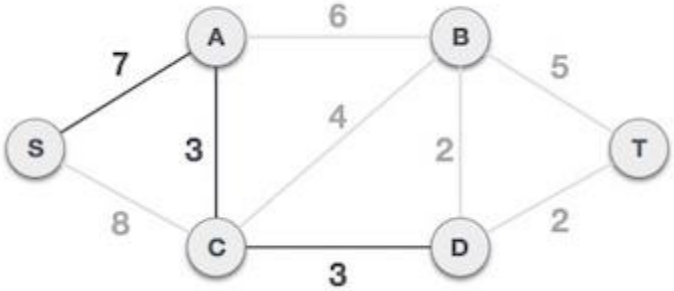
After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.



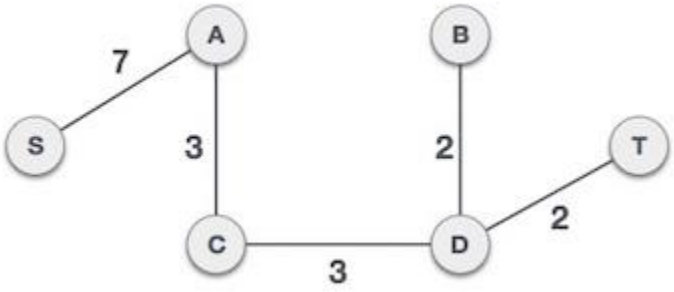
Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



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.



Now, calculate minimum cost.

Minimum cost of this spanning tree is: 7+3+3+2+2 = 17

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