**Minimum cost spanning tree: Prim’s Algorithm, Kruskal Algorithm**

Spanning Tree

A **spanning tree** is a subset of an undirected Graph that has all the vertices connected by minimum number of edges.

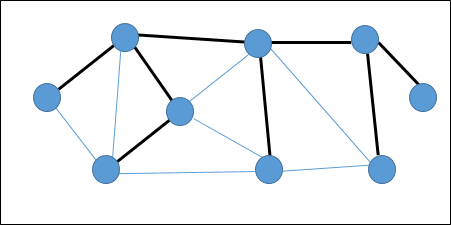
If all the vertices are connected in a graph, then there exists at least one spanning tree. In a graph, there may exist more than one spanning tree.

### **Properties**

* A spanning tree does not have any cycle.
* Any vertex can be reached from any other vertex.

### **Example**

In the following graph, the highlighted edges form a spanning tree.

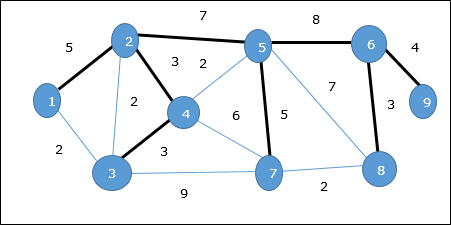


## **Minimum Spanning Tree**

A **Minimum Spanning Tree (MST)** is a subset of edges of a connected weighted undirected graph that connects all the vertices together with the minimum possible total edge weight. To derive an MST, Prim’s algorithm or Kruskal’s algorithm can be used. Hence, we will discuss Prim’s algorithm in this chapter.

As we have discussed, one graph may have more than one spanning tree. If there are **n** number of vertices, the spanning tree should have ***n* - 1** number of edges. In this context, if each edge of the graph is associated with a weight and there exists more than one spanning tree, we need to find the minimum spanning tree of the graph.

Moreover, if there exist any duplicate weighted edges, the graph may have multiple minimum spanning tree.



In the above graph, we have shown a spanning tree though it’s not the minimum spanning tree. The cost of this spanning tree is (5 + 7 + 3 + 3 + 5 + 8 + 3 + 4) = 38.

We will use Prim’s algorithm to find the minimum spanning tree.

## **Prim’s Algorithm**

Prim’s algorithm is a greedy approach to find the minimum spanning tree. In this algorithm, to form a MST we can start from an arbitrary vertex.

**Algorithm: MST-Prim’s (G, w, r)**

for each u є G.V

u.key = ∞

u.∏ = NIL

r.key = 0

Q = G.V

while Q ≠ Ф

u = Extract-Min (Q)

for each v є G.adj[u]

if each v є Q and w(u, v) < v.key

v.∏ = u

v.key = w(u, v)

The function Extract-Min returns the vertex with minimum edge cost. This function works on min-heap.

### **Example**

Using Prim’s algorithm, we can start from any vertex, let us start from vertex **1**.

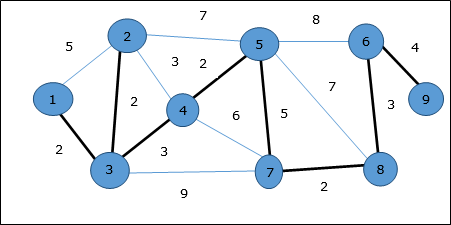
Vertex **3** is connected to vertex **1** with minimum edge cost, hence edge **(1, 2)** is added to the spanning tree.

Next, edge **(2, 3)** is considered as this is the minimum among edges **{(1, 2), (2, 3), (3, 4), (3, 7)}**.

In the next step, we get edge **(3, 4)** and **(2, 4)** with minimum cost. Edge **(3, 4)** is selected at random.

In a similar way, edges **(4, 5), (5, 7), (7, 8), (6, 8)** and **(6, 9)** are selected. As all the vertices are visited, now the algorithm stops.

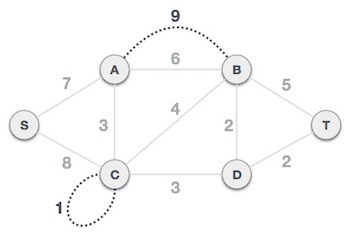
The cost of the spanning tree is (2 + 2 + 3 + 2 + 5 + 2 + 3 + 4) = 23. There is no more spanning tree in this graph with cost less than **23**.



Example 2:



**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.

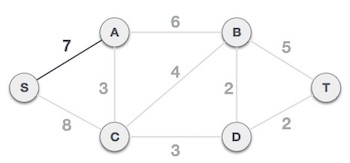


## **Step 2 - Choose any arbitrary node as root node**

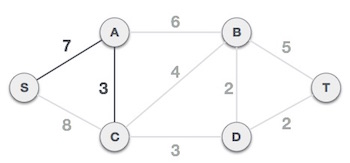
In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then there must be at least one edge, which will join it to the rest of the tree.

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

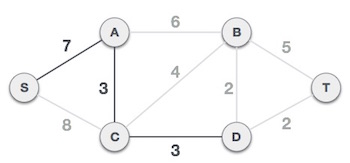
After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.



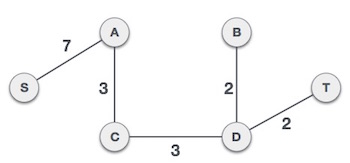
Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



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

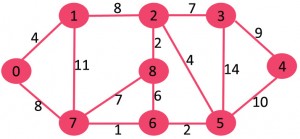
Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

Below are the steps for finding MST using Kruskal’s algorithm

***1.****Sort all the edges in non-decreasing order of their weight.****2.****Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.****3.****Repeat step#2 until there are (V-1) edges in the spanning tree.*

The step#2 uses Union-Find algorithm to detect cycle.

The algorithm is a Greedy Algorithm. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far. Let us understand it with an example: Consider the below input graph.

[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-0.jpg)

The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having (9 – 1) = 8 edges.

After sorting:

Weight Src Dest

1 7 6

2 8 2

2 6 5

4 0 1

4 2 5

6 8 6

7 2 3

7 7 8

8 0 7

8 1 2

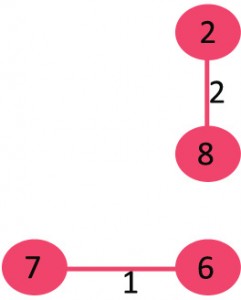
9 3 4

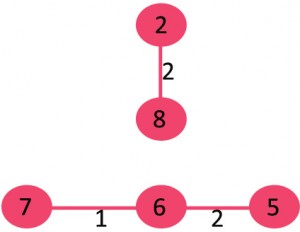
10 5 4

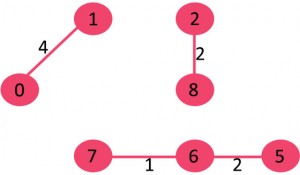
11 1 7

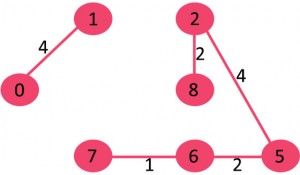
14 3 5

Now pick all edges one by one from sorted list of edges  
**1.** Pick edge 7-6: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-1.jpg)

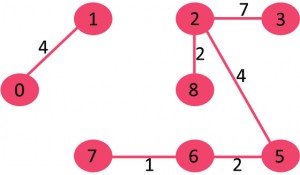
**2.** Pick edge 8-2: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-2.jpg)

**3.** Pick edge 6-5: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-3.jpg)

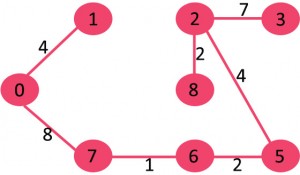
**4.** Pick edge 0-1: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-4.jpg)

**5.** Pick edge 2-5: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-5.jpg)

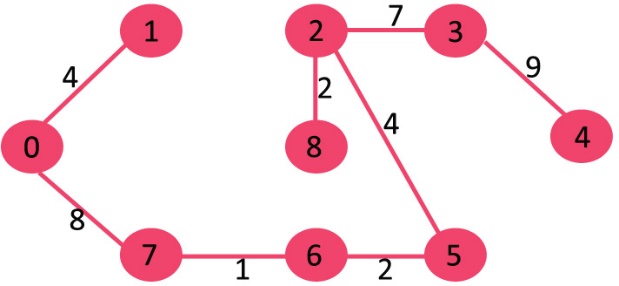
**6.** Pick edge 8-6: Since including this edge results in cycle, discard it.

**7.** Pick edge 2-3: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-6.jpg)

**8.** Pick edge 7-8: Since including this edge results in cycle, discard it.

**9.** Pick edge 0-7: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/Fig-7.jpg)

**10.** Pick edge 1-2: Since including this edge results in cycle, discard it.

**11.** Pick edge 3-4: No cycle is formed, include it.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/fig8new.jpeg)

Since the number of edges included equals (V – 1), the algorithm stops here.

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_spanning_tree.htm#:~:text=A%20Minimum%20Spanning%20Tree%20(MST,Kruskal's%20algorithm%20can%20be%20used.>
2. <https://www.tutorialspoint.com/data_structures_algorithms/prims_spanning_tree_algorithm.htm>
3. <https://www.tutorialspoint.com/data_structures_algorithms/kruskals_spanning_tree_algorithm.htm>
4. <https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/>

**Lecture Video:**

1. <https://youtu.be/eB61LXLZVqs>
2. <https://youtu.be/3rrNH_AizMA>

**Online Notes:**

1. <http://vssut.ac.in/lecture_notes/lecture1428551222.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**