**Assignment No : 03**

**1.Title of Assignment :** Implement Greedy search algorithm for any of the following application:

Kruskal's Minimal Spanning Tree Algorithm

**2.Prerequisite**: Basic knowledge of:

Tree data structure, Graph and searching algorithms

**3. Objective:** In this experiment, we will learn about :

* Minimum Spanning Tree
* Greedy Method for searching algorithms
* Implementation and working of Kruskal’s Algorithm

Outcome: Successfully able to implement Kruskal’s Algorithm

**4. S/w and H/w Requirements :**

Open Source python programming tool like JupyterNotebook.

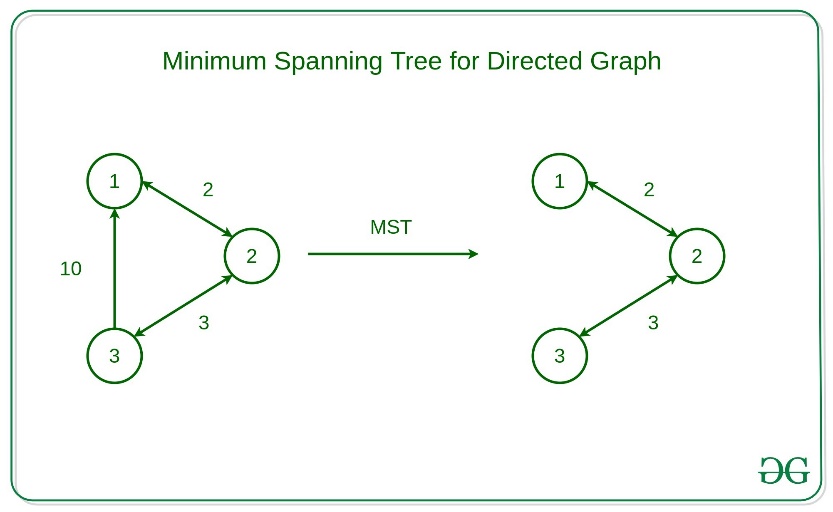
**5. Theory**:

**Minimum spanning tree:**

A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree.

A **spanning tree** is defined as a tree-like subgraph of a connected, undirected graph that includes all the vertices of the graph. Or, to say in Layman’s words, it is a subset of the edges of the graph that forms a tree (**acyclic**) where every node of the graph is a part of the tree.

The minimum spanning tree has all the properties of a spanning tree with an added constraint of having the minimum possible weights among all possible spanning trees. Like a spanning tree, there can also be many possible MSTs for a graph.



**Kruskal’s algorithm :**

In Kruskal’s algorithm, sort all edges of the given graph in increasing order. Then it keeps on adding new edges and nodes in the MST if the newly added edge does not form a cycle. It picks the minimum weighted edge at first and the maximum weighted edge at last. Thus we can say that it makes a locally optimal choice in each step in order to find the optimal solution. Hence this is a [**Greedy Algorithm**](https://www.geeksforgeeks.org/introduction-to-greedy-algorithm-data-structures-and-algorithm-tutorials/).

**How to find MST using Kruskal’s algorithm?**

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

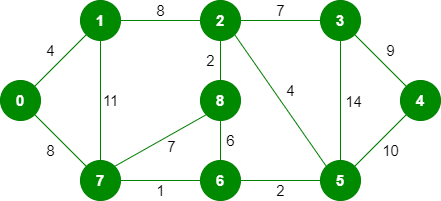
1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If the cycle is not formed, include this edge. Else, discard it.
3. Repeat step#2 until there are (V-1) edges in the spanning tree.

Kruskal’s algorithm to find the minimum cost spanning tree uses the greedy approach. 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:

### Illustration:

Below is the illustration of the above approach:

***Input Graph:***

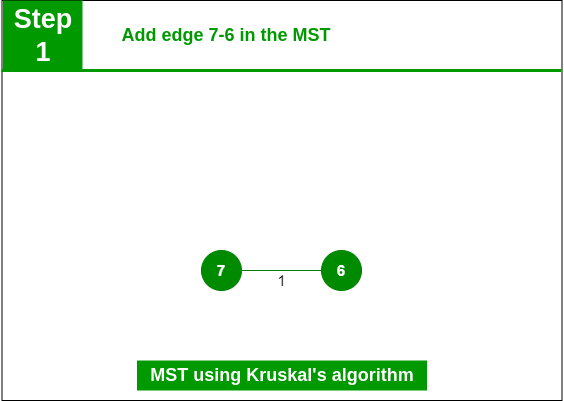
******

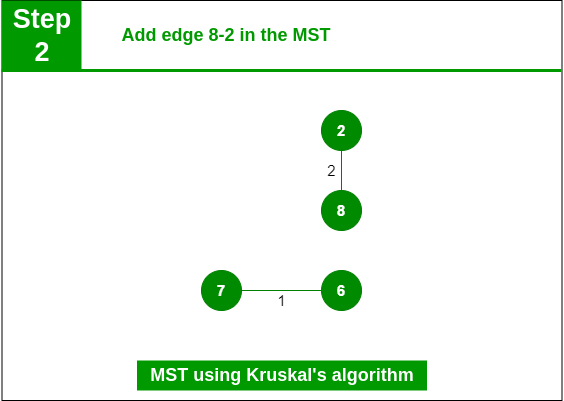
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having (9 – 1) = 8 edges.   
After sorting:

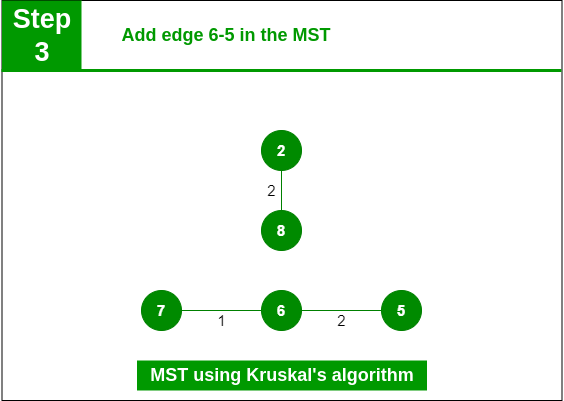
|  |  |  |
| --- | --- | --- |
| *Weight* | *Source* | *Destination* |
| *1* | *7* | *6* |
| *2* | *8* | *2* |
| *2* | *6* | *5* |
| *4* | *0* | *1* |
| *4* | *2* | *5* |
| *6* | *8* | *6* |
| *7* | *2* | *3* |
| *7* | *7* | *8* |
| *8* | *0* | *7* |
| *8* | *1* | *2* |
| *9* | *3* | *4* |
| *10* | *5* | *4* |
| *11* | *1* | *7* |
| *14* | *3* | *5* |

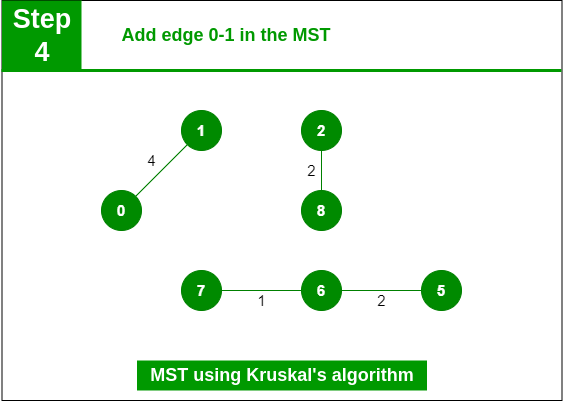
Now pick all edges one by one from the sorted list of edges

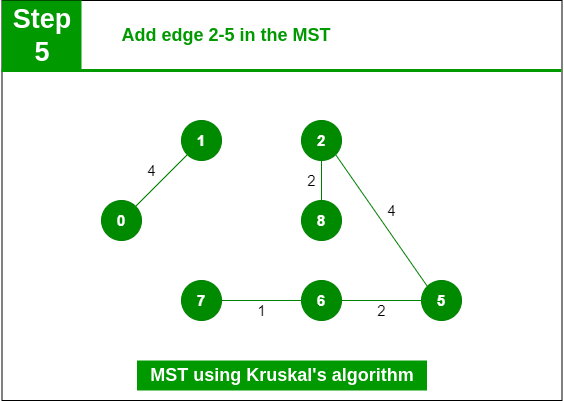
**Step 1:** Pick edge 7-6. No cycle is formed, include it.

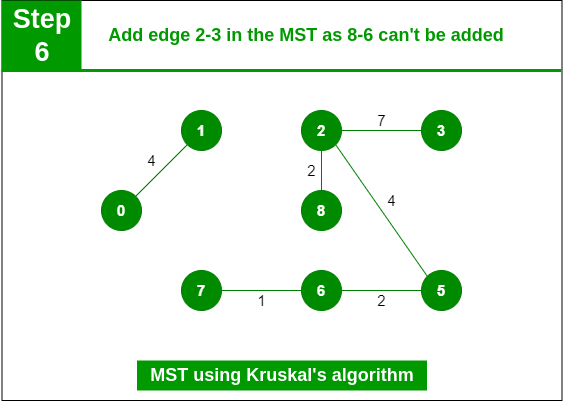


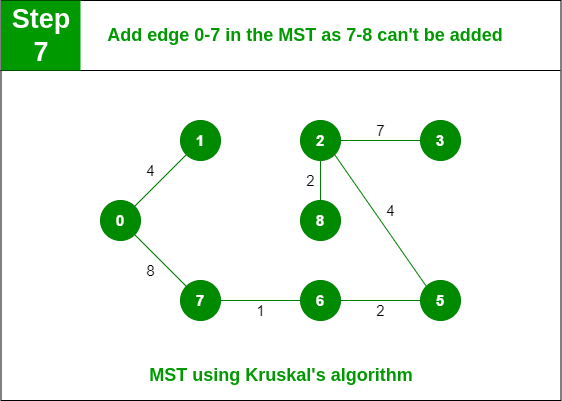
**Step 2:**  Pick edge 8-2. No cycle is formed, include it.

**Step 3:** Pick edge 6-5. No cycle is formed, include it.

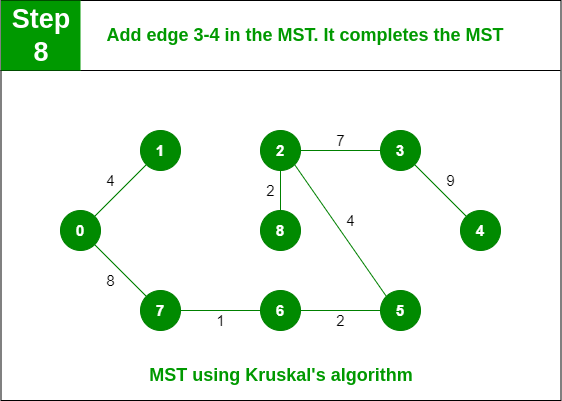
**Step 4:** Pick edge 0-1. No cycle is formed, include it.

**Step 5:** Pick edge 2-5. No cycle is formed, include it.

**Step 6:** Pick edge 8-6. Since including this edge results in the cycle, discard it. Pick edge 2-3: No cycle is formed, include it.

 **Step 7:** Pick edge 7-8. Since including this edge results in the cycle, discard it. Pick edge 0-7. No cycle is formed, include it.

**Step 8:** Pick edge 1-2. Since including this edge results in the cycle, discard it. Pick edge 3-4. No cycle is formed, include it.



**Note:** Since the number of edges included in the MST equals to (V – 1), so the algorithm stops here

**Time Complexity:** O(E \* logE) or O(E \* logV)

* Sorting of edges takes O(E \* logE) time.
* After sorting, we iterate through all edges and apply the find-union algorithm. The find and union operations can take at most O(logV) time.
* So overall complexity is O(E \* logE + E \* logV) time.
* The value of E can be at most O(V2), so O(logV) and O(logE) are the same. Therefore, the overall time complexity is O(E \* logE) or O(E\*logV)

**Auxiliary Space:**O(V + E), where V is the number of vertices and E is the number of edges in the graph.

**6.Conclusion :**

In this way we have studied Greedy method , Minimum spanning tree, Kruskal’s Algorithm and their Implementation.