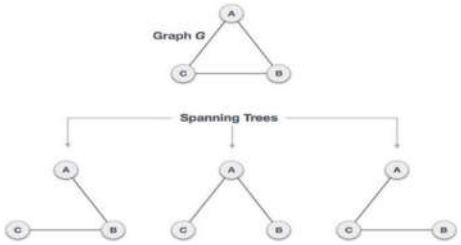
Spanning Tree and Minimum Spanning Tree

Spanning Tree:

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

Some properties:

- Every connected and undirected Graph G has at least one spanning tree.
- A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.
- A complete undirected graph can have maximum $\mathbf{n^{n-2}}$ number of spanning trees, where **n** is the number of nodes.
- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G, have the same number of edges and vertices.
- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.
- Spanning tree has **n-1** edges, where **n** is the number of nodes (vertices).
- From a complete graph, by removing maximum **e - n + 1** edges, we can construct a spanning tree.



Application of Spanning Tree:

Spanning tree is basically used to find a minimum path to connect all nodes in a graph. Common application of spanning trees are −

- **Civil Network Planning**
- **Computer Network Routing Protocol**
- **Cluster Analysis**

Let us understand this through a small example. Consider, city network as a huge graph and now plans to deploy telephone lines in such a way that in minimum lines we can connect to all city nodes. This is where the spanning tree comes into picture.

Minimum Spanning Trees:

A weighted graph is a graph G in which each edge e has been assigned a real number w(e) called the weight of e.If H is a subgraph of a weighted graph,the weight w(H) of H is the sum of the weights $w(e_1)+\ldots..+w(e_k)$,where $\{e_1,\ldots.,e_k\}$ is the set of edges of H.

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

We shall learn about two most important spanning tree algorithms here −

Kruskal's Algorithm

Prim's Algorithm

Both are greedy algorithms.

2.6.1 Algorithm : Kruskal's Algorithm

In this algorithm we choose an edge of G which has smallest weight among the edges of G which are not loops. This algorithm gives an acyclic subgraph T of G and the theorem given below proves that T is a minimal spanning tree of G. Following steps are required :

Step 1 : Choose e_1, an edge of G, such that weight of e_1, $\omega(e_1)$ is as small as possible and e_1 is not a loop.

Step 2 : If edges $e_1, e_2, \ldots\ldots\ldots\ldots, e_i$ have been chosen then choose an edge e_{i+1}, not already chosen, such that

(i) the induced subgraph $G[\{e_1, \ldots\ldots\ldots\ldots, e_{i+1}\}]$ is acyclic and

(ii) $\omega(e_{i+1})$ is as small as possible.

Step 3 : If G has n vertices, stop after n – 1 edges have been chosen. Otherwise repeat step 2.

Theorem : Let G be a weighted connected graph in which the weights of the edges are all non-negative numbers. Let T be a subgraph of G obtained by Kruskal's algorithm. Then T is a minimal Spanning tree of G.

Example : Given below is a weighted graph G with 5 vertices. Apply Kruskal's Algorithm to determine a minimal spanning tree.

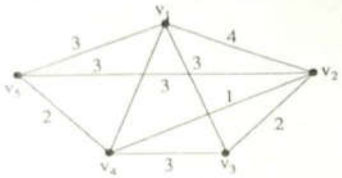


Figure 6

Solution : According to step 1, we will choose, $e = v_2v_4$ as it has the minimum weight.

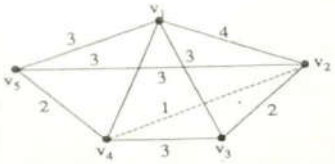
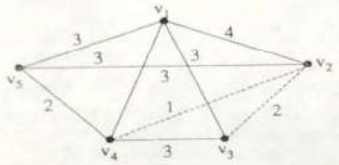
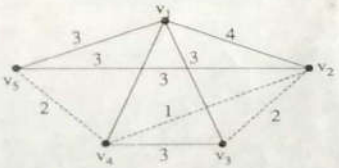


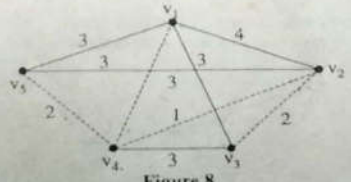
Figure 7



Choose $e = v_2\, v_3$



Choose $e = v_4\, v_5$



Choose $e = v_1\, v_4$

Figure 8

Since vertices are 5 and we have choosen 4 edges stop. The minimal spanning tree of graph is

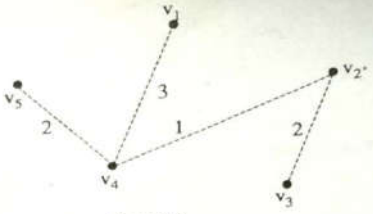


Figure 9

Note : Kruskal's algorithm is an example of a type of algorithm known as **greedy**. Greedy algorithms are essentially algorithms that proceed by selecting the choice that looks best at the moment. All processes cannot be handled so simply.

2.6.2 Algorithm : Prim's Algorithm

Another Algorithm for finding a minimal spanning tree is Prim's Algorithm. It chooses a vertex first and chooses an edge with smallest weight incident on that vertex. The algorithm involves following steps.

Step 1 : Choose any vertex v_1 of G

Step 2 : Choose an edge $e_1 = v_1 v_2$ of G such that $v_2 \neq v_1$ and e_1 has smallest weight among the edges of G incident with v_1.

Step 3 : If edges $e_1, e_2, \ldots\ldots\ldots\ldots e_i$ have been chosen involving end points $v_1, v_2, \ldots\ldots$, v_{i+1}. Choose an edge $e_{i+1} = v_j v_k$ with $v_j \in \{ v_i, \ldots\ldots\ldots\ldots v_{i+1}\}$ and $v_k \notin \{ v_1, \ldots\ldots, v_{i+1}\}$ such that e_{i+1} has smallest weight among the edges of G with precisely one end in $\{v_1, \ldots\ldots\ldots\ldots, v_{i+1}\}$.

Step 4 : Stop after n – 1 edges have been chosen. Otherwise go to step 3.

Theorem : Let G be a weighted connected graph in which the weights of the edges are all non-negative numbers. Let T be a subgraph of G obtained by Prim's Algorithm. Then T is a minimal spanning tree of G.

Example : Find the minimal spanning tree of the weighted graph in previous example using Prim's Algorithm.

Solution :

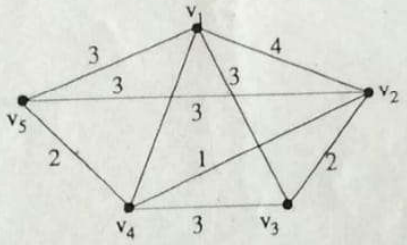
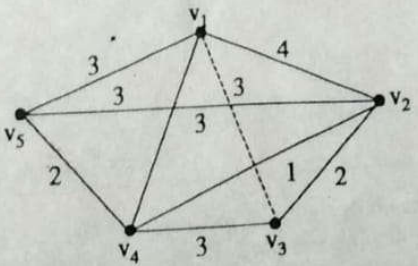


Figure 10

According to step 1 we choose vertex v_1. Now edge with smallest weight incident on v_1 is $e = v_1, v_3$ so we choose $e = v_1 v_3$ (or $e = v_1 v_5$)



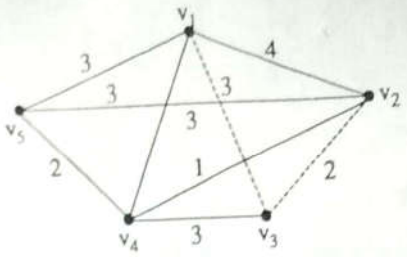
Choose $e = v_1 v_3$

Figure 11

Now we look on to the weights,

$w(v_1, v_2) = 4$, $w(v_1, v_4) = 3$, $w(v_1, v_5) = 3$.
$w(v_3, v_2) = 2$, $w(v_3, v_4) = 3$.

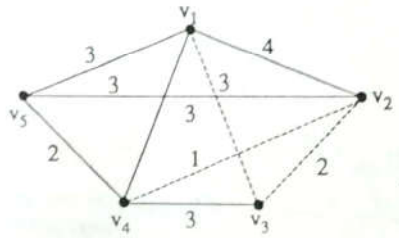
since minimum is $w(v_3, v_2)$, we choose $e = v_3, v_2$



Choose $e = v_3\, v_2$

Figure 12

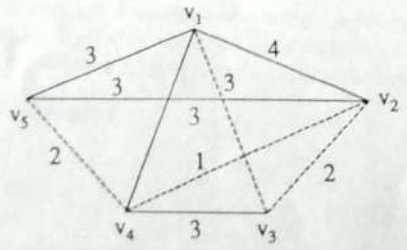
Again, $w(v_2, v_5) = 3$, $w(v_2, v_4) = 1$,
and $w(v_3, v_4) = 3$



Choose $e = v_2\, v_4$

Figure 13

Now, $w(v_4, v_5) = 2$



Choose $e = v_4\, v_5$

Figure 14

The minimal spanning tree is

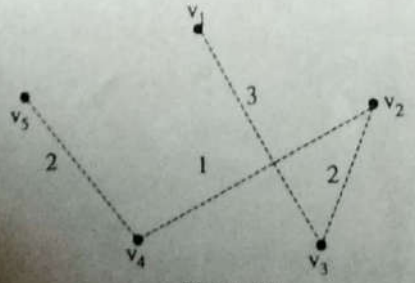
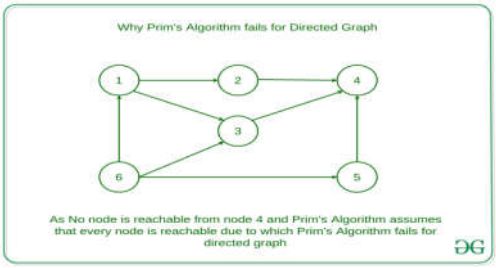


Figure 15

Why Prim’s and Kruskal’s MST algorithm fails for Directed Graph?

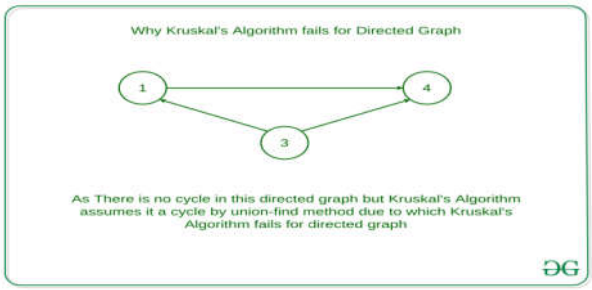
Why Prim’s Algorithm Fails for Directed Graph ?

Prim’s algorithm assumes that all vertices are connected. But in a directed graph, every node is not reachable from every other node. So, Prim’s algorithm fails due to this reason.
For Example:



As it is visible in the graph, no node is reachable from node 4. Directed graphs fail the requirement that all vertices are connected.

Why Kruskal’s Algorithm fails for directed graph ?
In Kruskal’s algorithm, In each step, it is checked that if the edges form a cycle with the spanning-tree formed so far. But Kruskal’s algorithm fails to detect the cycles in a directed graph as there are cases when there is no cycle between the vertices but Kruskal’s Algorithm assumes it to cycle and don’t take consider some edges due to which Kruskal’s Algorithm fails for directed graph.
For example:



This graph will be reported to contain a cycle with the Union-Find method, but this graph has no cycle.