Minimum Spanning Tree

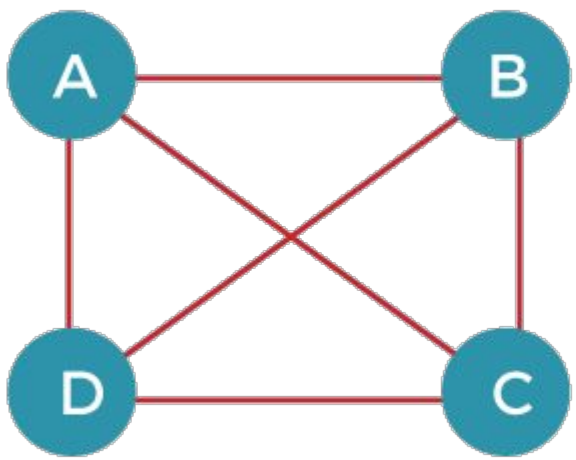
Greedy Algorithm

CSE (AIML)

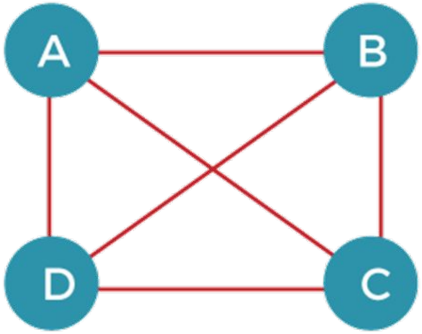
6^{TH} SEM

Spanning Tree

- A connected subgraph ‘s’ of Graph G(V, E) is said to be spanning tree iff:
- 1. ‘s’ should contain all vertices of ‘G’.
- 2. ‘s’ should contain (|V| - 1) edges.



Spanning Tree

- 

- The number of spanning trees that can be made from the above complete graph equals to $n^{n-2} = 4^{4-2}$=$4^2 = 16$
- Therefore, 16 spanning trees can be created from the above graph.

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.

Kruskal's Algorithm

- The inputs taken by the Kruskal's algorithm are the graph G {V, E}, where V is the set of vertices and E is the set of edges, and the source vertex S and the minimum spanning tree of graph G is obtained as an output.

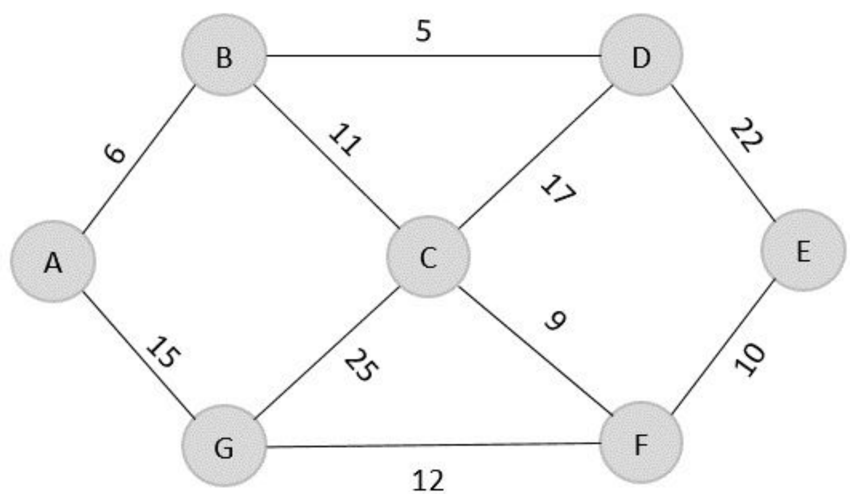
Kruskal's Algorithm

- Sort all the edges in the graph in an ascending order and store it in an array edge[].
- Construct the forest of the graph on a plane with all the vertices in it.
- Select the least cost edge from the edge[] array and add it into the forest of the graph. Mark the vertices visited by adding them into the visited[] array.
- Repeat the steps 2 and 3 until all the vertices are visited without having any cycles forming in the graph
- When all the vertices are visited, the minimum spanning tree is formed.
- Calculate the minimum cost of the output spanning tree formed.

Kruskal's Algorithm

EXAMPLE

Construct a minimum spanning tree using Kruskal's algorithm for the graph given below –



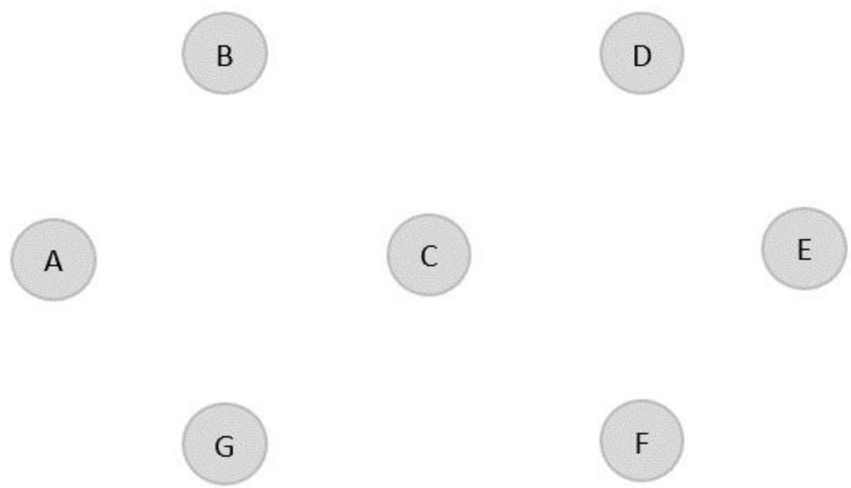
Kruskal's Algorithm

Solution

- As the first step, sort all the edges in the given graph in an ascending order and store the values in an array.
- Edge B→D A→B C→F F→E B→C G→F A→G C→D D→E C→G
- Cost 5 6 9 10 11 12 15 17 22 25

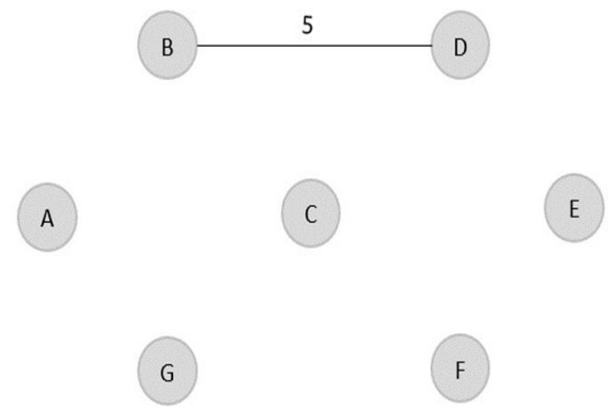
Kruskal's Algorithm

construct a forest of the given graph on a single plane.



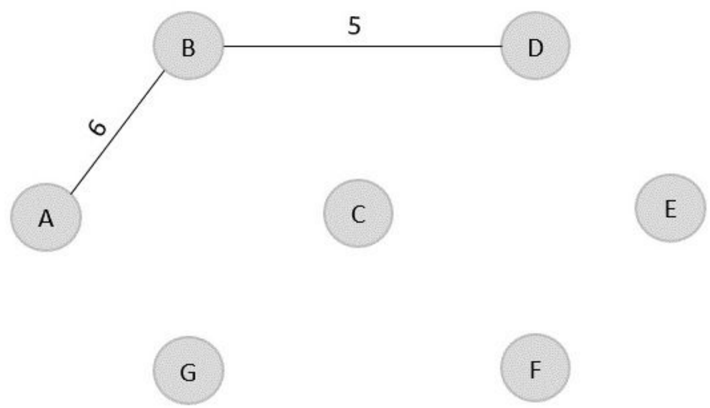
Kruskal's Algorithm

- From the list of sorted edge costs, select the least cost edge and add it onto the forest in output graph.
- B → D = 5
- Minimum cost = 5
- Visited array, v = {B, D}



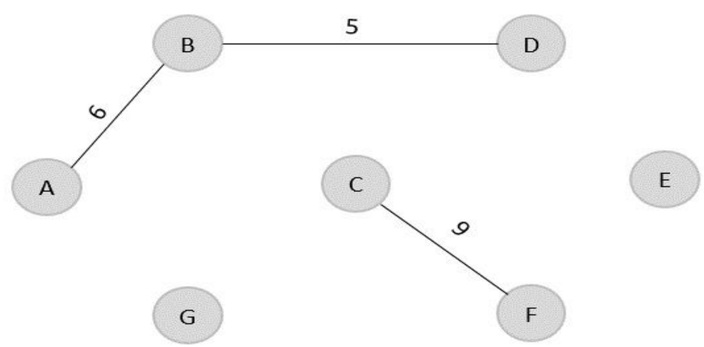
Kruskal's Algorithm

- Similarly, the next least cost edge is B → A = 6; so we add it onto the output graph.
- Minimum cost = 5 + 6 = 11
- Visited array, v = {B, D, A}



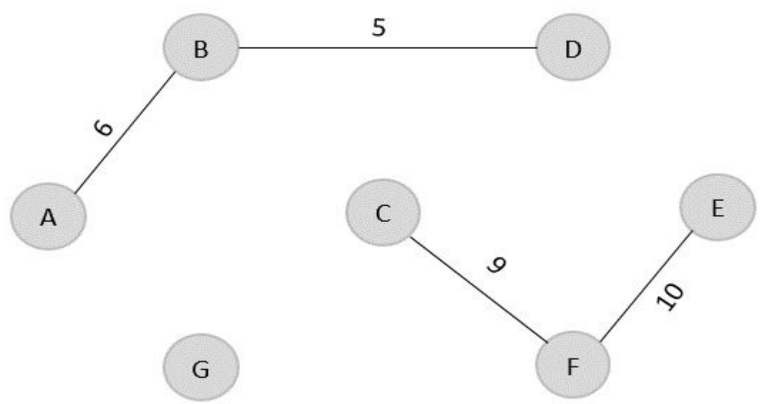
Kruskal's Algorithm

- The next least cost edge is C → F = 9; add it onto the output graph.
- Minimum Cost = 5 + 6 + 9 = 20
- Visited array, v = {B, D, A, C, F}



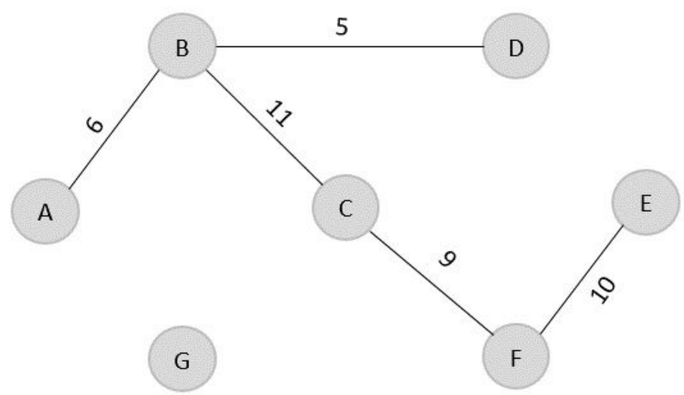
Kruskal's Algorithm

- The next edge to be added onto the output graph is F → E = 10.
- Minimum Cost = 5 + 6 + 9 + 10 = 30
- Visited array, v = {B, D, A, C, F, E}



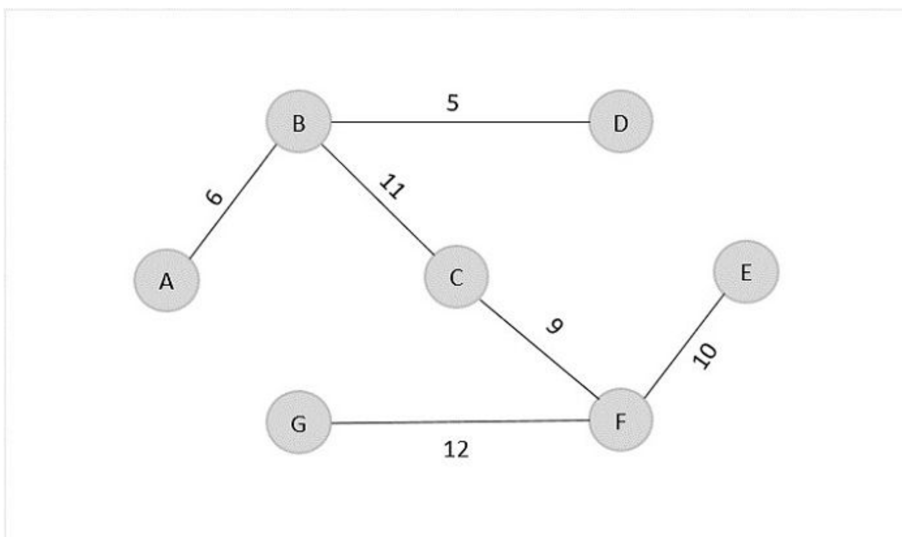
Kruskal's Algorithm

- The next edge from the least cost array is B → C = 11, hence we add it in the output graph.
- Minimum cost = 5 + 6 + 9 + 10 + 11 = 41
- Visited array, v = {B, D, A, C, F, E}



Kruskal's Algorithm

- The last edge from the least cost array to be added in the output graph is F → G = 12.
- Minimum cost = 5 + 6 + 9 + 10 + 11 + 12 = 53
- Visited array, v = {B, D, A, C, F, E, G}



Kruskal's Algorithm

- The obtained result is the minimum spanning tree of the given graph with cost = 53.

Prim’s Algorithm

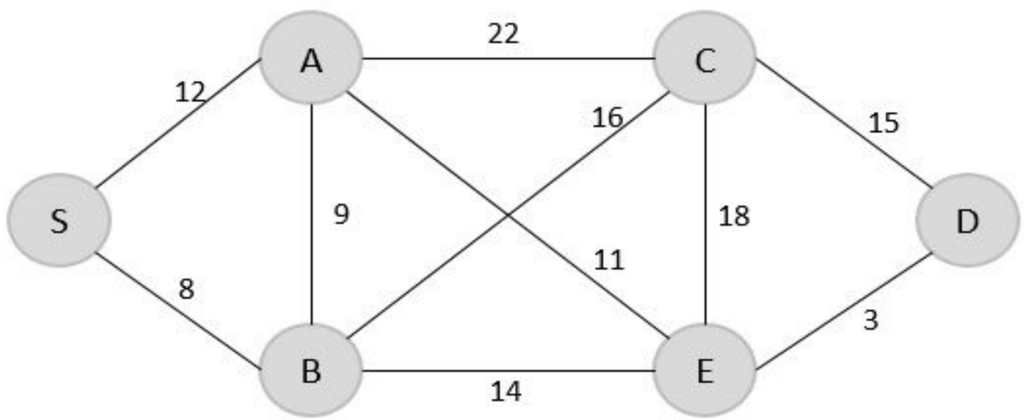
- To execute the prim’s algorithm, the inputs taken by the algorithm are the graph G {V, E}, where V is the set of vertices and E is the set of edges, and the source vertex S. A minimum spanning tree of graph G is obtained as an output.

Prim’s Algorithm

- Declare an array visited[] to store the visited vertices and firstly, add the arbitrary root, say S, to the visited array.
- Check whether the adjacent vertices of the last visited vertex are present in the visited[] array or not.
- If the vertices are not in the visited[] array, compare the cost of edges and add the least cost edge to the output spanning tree.
- The adjacent unvisited vertex with the least cost edge is added into the visited[] array and the least cost edge is added to the minimum spanning tree output.
- Steps 2 and 4 are repeated for all the unvisited vertices in the graph to obtain the full minimum spanning tree output for the given graph.
- Calculate the cost of the minimum spanning tree obtained.

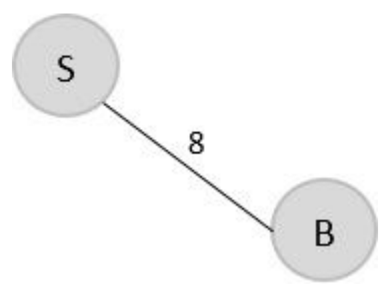
Prim’s Algorithm Examples

- Find the minimum spanning tree using prim’s method (greedy approach) for the graph given below with S as the arbitrary root.



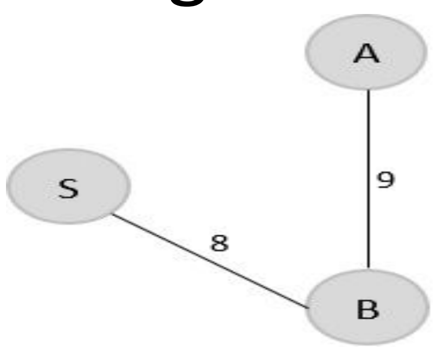
Prim’s Algorithm

- Step 1: Create a visited array to store all the visited vertices into it.
- V = { }
- The arbitrary root is mentioned to be S, so among all the edges that are connected to S we need to find the least cost edge.
- S → B = 8
- V = {S, B}



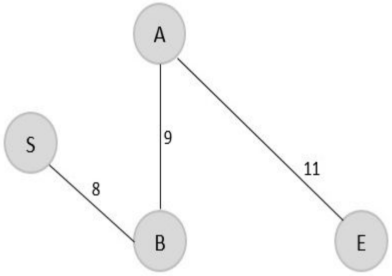
Prim’s Algorithm

- Step 2: Since B is the last visited, check for the least cost edge that is connected to the vertex B.
- B → A = 9
- B → C = 16
- B → E = 14
- Hence, B → A is the edge added to the spanning tree.
- V = {S, B, A}



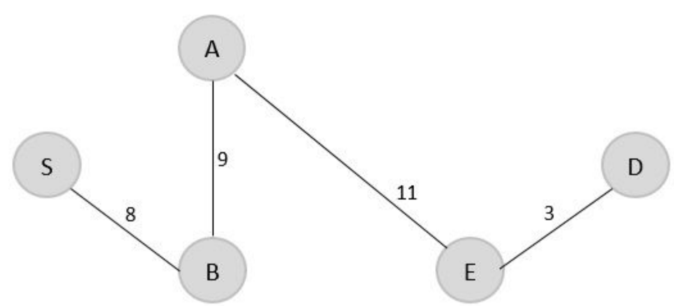
Prim’s Algorithm

- Step 3: Since A is the last visited, check for the least cost edge that is connected to the vertex A.
- A → C = 22
- A → B = 9
- A → E = 11
- But A → B is already in the spanning tree, check for the next least cost edge. Hence, A → E is added to the spanning tree.
- V = {S, B, A, E}



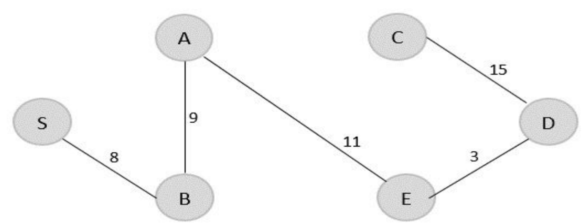
Prim’s Algorithm

- Step 4: Since E is the last visited, check for the least cost edge that is connected to the vertex E.
- E → C = 18
- E → D = 3
- Therefore, E → D is added to the spanning tree.
- V = {S, B, A, E, D}



Prim’s Algorithm

- Step 5: Since D is the last visited, check for the least cost edge that is connected to the vertex D.
- D → C = 15
- E → D = 3
- Therefore, D → C is added to the spanning tree.
- V = {S, B, A, E, D, C}



- The minimum spanning tree is obtained with the minimum cost = 46