ASSIGNMENT NO.4.

Aim:

For a weighted graph G, find the minimum spanning tree using Prims Algorithm.

Objective:

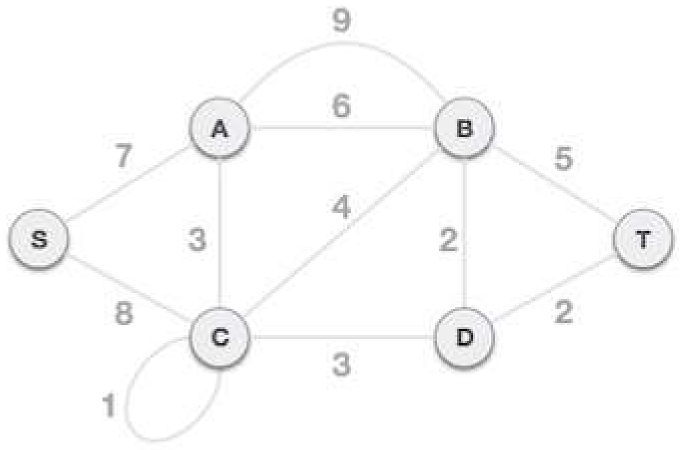
Understand the concepts of prims algorithm

Theory:

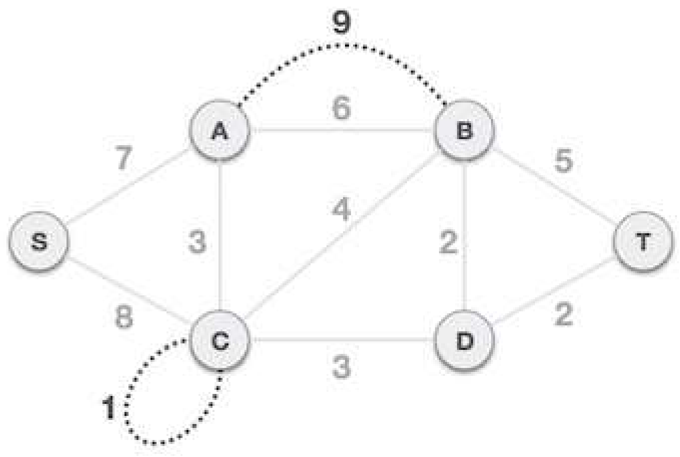
Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

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

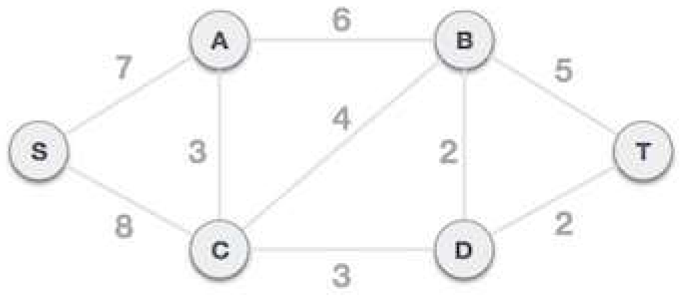
To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we shall use the same example –



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.

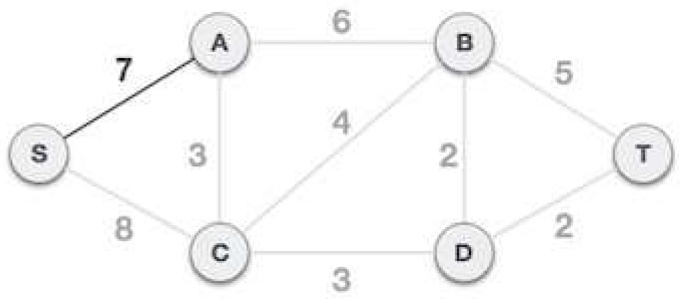


Step 2 - Choose any arbitrary node as root node

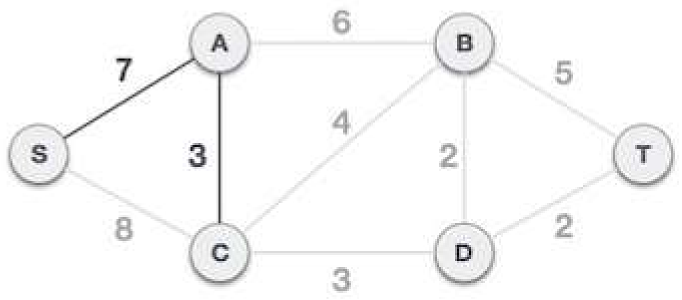
In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then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

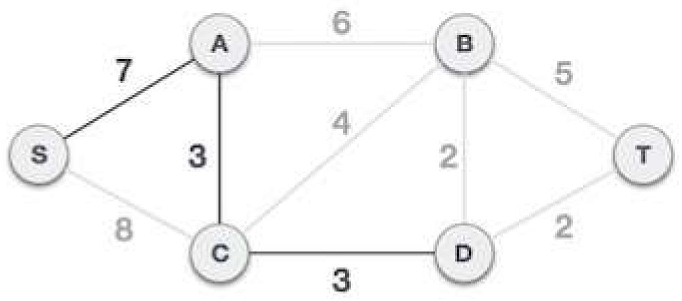
After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.



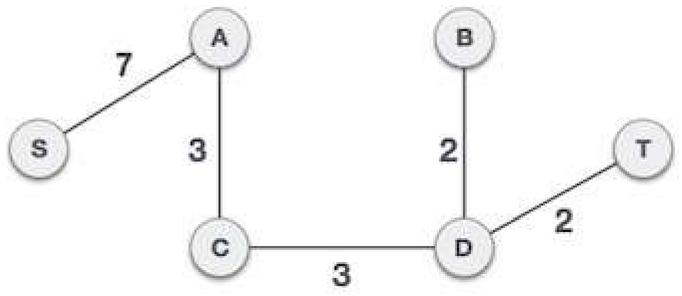
Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



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.

Algorithm:

```
Algorithm Prims(E,cost,n,t)
{1.Let (k,l) be the edge of minimum cost
2.mincost=cost(k,l)
3.t[1,1]=k;t[1,2]=l;
4.for i=1 to n do
   If(cost[i,l]<cost[i,k] then near[i]=l
  Else  near[i]=k;
5.near[k]=near[l]=0
6.for i=2 to n-1 do
  6.1Let j be the index  such that near[j]!=0 and
      Cost[j,near[j]] is minimum
   6.2t[i,1]=j ;t[i ,2]=near[j]
   6.3  mincost=mincost+cost[j,near[j]];
   6.4 near[j]=0

   6.5for k=1 to n do

     if ((near[k]!=0) and (cost[k,near[k]]>cost[k,j]))

              then near[k]=j}

Return mincost

}
```

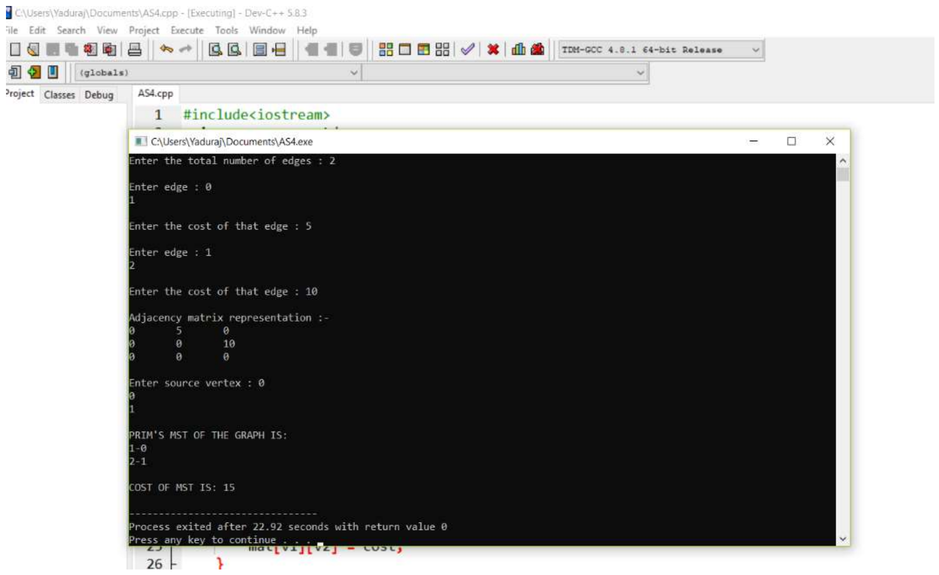
C++ Code:

```
#include<iostream>
using namespace std;

void create(int mat[][10], int v)
{
   int v1,v2,cost;
   int edges;
   cout << "\nEnter the total number of edges : ";
   cin >> edges;

   for(int i=0;i<v;i++)
            {
                        for(int j=0;j<v;j++)
                        {
                                    mat[i][j] = 0;
                        }
            }

   for(int i=0;i<edges;i++)
   {
      cout << "\nEnter edge : ";
      cin >> v1 >> v2;
      cout << "\nEnter the cost of that edge : ";
      cin >> cost;
      mat[v1][v2] = cost;
   }
}

void display_matrix(int mat[][10],int v)
```

```
{
   cout << "\nAdjacency matrix representation :- " << endl;
   for(int i=0;i<v;i++)
   {
      for(int j=0;j<v;j++)
      {
         cout << mat[i][j] << "\t";
      }
      cout << endl;
   }
}

int min_dist(int dist[],int visited[],int v)
{
   int min = 32767;
   int min_index;
   for(int i=0;i<v;i++)
   {
      if(visited[i] == 0 && dist[i] <= min)
      {
         min = dist[i];
         min_index = i;
      }
   }
   cout << min_index <<endl;
   return min_index;
}

void print_dist(int mat[][10],int dist[],int v,int parent[])
{
   int sum=0;
   cout<<"\nPRIM'S MST OF THE GRAPH IS: ";
   for(int i = 1; i<v; i++)
   {
      cout<<"\n"<<i<<"-"<<parent[i];
      sum = sum + mat[parent[i]][i];
   }
   cout<<endl;
   cout<<"\nCOST OF MST IS: "<<sum<<endl;
}

void prims(int mat[][10],int s,int v)
{
   int dist[v];
   int visited[v];
   int parent[v];
```

```
    for(int i=0;i<v;i++)
    {
        dist[i] = 32767;
        visited[i] = 0;
    }
    dist[s] = 0;
    int p=0;
    for(int j=0;j<v-1;j++)
    {
        p = min_dist(dist,visited,v);
        visited[p] = 1;
        for(int q=0;q<v;q++)
        {
            if(mat[p][q] != 0)
            {
                if(visited[q] == 0 && mat[p][q] < dist[q])
                {
                    dist[q] = mat[p][q];
                    parent[q] = p;
                }
            }
        }
    }
    print_dist(mat,dist,v,parent);
}


int main()
{
    int v;
    int s;
    cout << "\nEnter the number of vertices : ";
    cin >> v;
    int mat[v][10];
            create(mat,v);
    display_matrix(mat,v);
    cout << "\nEnter source vertex : ";
    cin >> s;
    prims(mat,s,v);
    return 0;
}
```

Output:



```
Enter the total number of edges : 2

Enter edge : 0
1

Enter the cost of that edge : 5

Enter edge : 1
2

Enter the cost of that edge : 10

Adjacency matrix representation :-
0       5       0
0       0       10
0       0       0

Enter source vertex : 0
0
1

PRIM'S MST OF THE GRAPH IS:
1-0
2-1

COST OF MST IS: 15

--------------------------------
Process exited after 22.92 seconds with return value 0
Press any key to continue . . .
```

Conclusion:

This assignment is used how prims algorithm is used in solving the example using vertex edge .