Assignment 5.

Aim :

You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures

Objective:

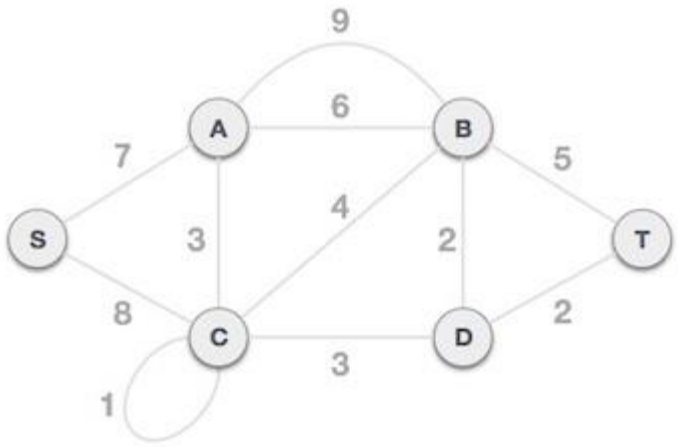
To understand the application of Prims algorithm to find the minimum spanning tree.

Theory:

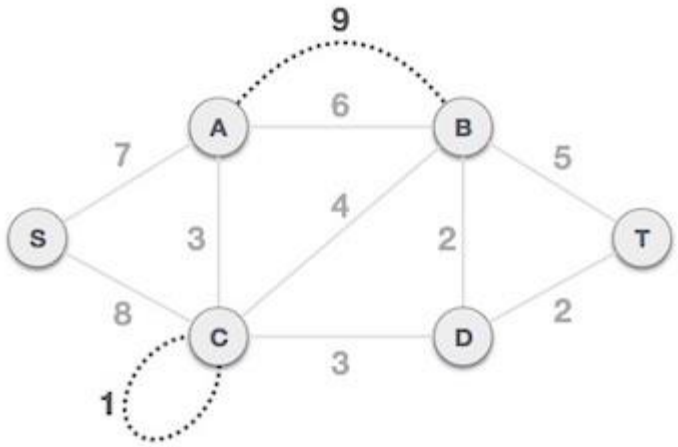
Prim's algorithm to find minimum cost spanning tree (as Kruskal's algorithm) uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

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.

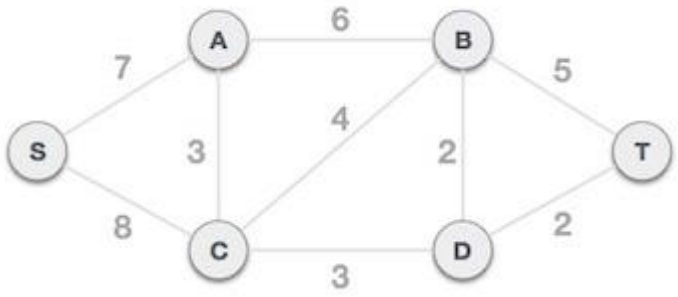
To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we shall use the same example –



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.

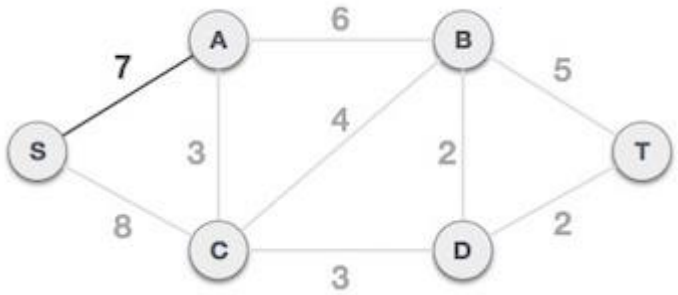


Step 2 - Choose any arbitrary node as root node

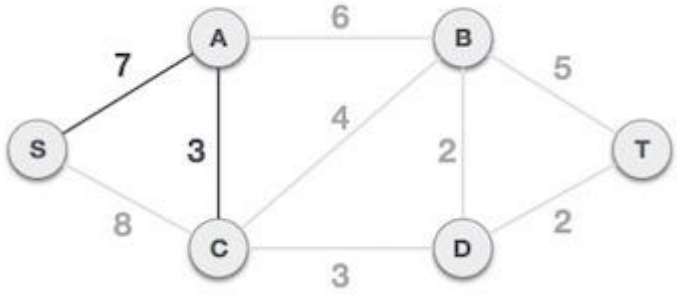
In this case, we choose **S** node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be a root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then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

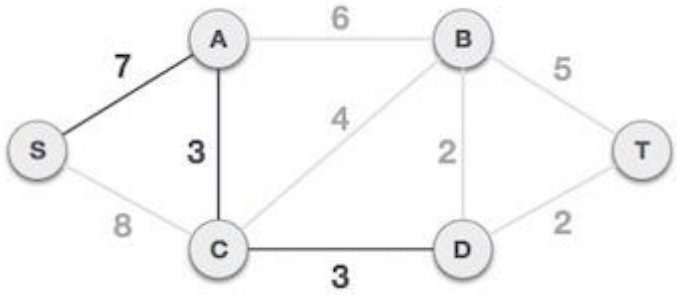
After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.



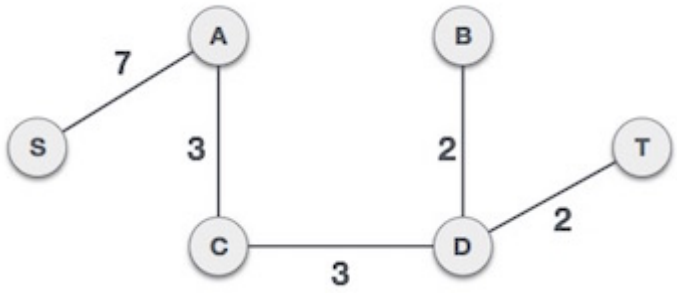
Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



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

}

Code:

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

int minimum(int *v,int *d, int n)
{

        int index;
```

```
        int min=9999;

        for(int i=0;i<n;i++)
          {
                if(d[i]<min && v[i]==0)
                {
                        min= d[i];
                        index=i;

                  }


          }
          return index;

}



int main()
{
        int n_v,n_e,u,v,value;

        cout<<"enter the no of CITIES and no of PATHS between them";
        cin>>n_v>>n_e;

        int g[n_v][n_v];
        int parent[n_v];
        int visited[n_v];
        int distance[n_v];
```

```
        for(int i=0;i<n_v;i++)
        {
                distance[i]=9999;
                parent[i]=0;
                visited[i]=0;

        }

for(int i=0;i<n_v;i++)
        for(int j=0;j<n_v;j++)
            g[i][j]=0;

        distance[0]=0;









        for(int i=0;i<n_e;i++)
         {
                cout<<"enter the stating city- destination- charge by phone company ";
                cin>>u>>v>>value;

        g[u][v]=g[v][u]=value;
         }

cout<<"the cost matrix is";
```

```
  for(int i=0;i<n_v;i++)
  {
        cout<<endl;
        for(int j=0;j<n_v;j++)
        {
        cout<<g[i][j]<<"\t";
        }
  }
```

```
for(int j=0;j<n_v-1;j++)
{
        int v= minimum(visited, distance,n_v);
            visited[v]=1;

            cout<<"the "<<j<<"loop run"<<endl;
            cout<<"the minimum value is"<<v<<endl;


        for(int i=0;i<n_v;i++)
        {
                if(g[v][i]!=0 && (distance[i]>g[v][i]) && visited[i]==0)
                {
                        distance[i]=g[v][i];
                        parent[i]=v;

                }
```

```
        }

/*      cout<<"the distance matrix is"<<endl;
        for(int i=0;i<n_v;i++)
  cout<<"0 ->"<<i<<" "<<"= "<<distance[i]<<endl;

        cout<<"the parent matrix is "<<endl;
        for(int i=0;i<n_v;i++)
  cout<<i<<" "<<"= "<<parent[i]<<endl;*/

}
cout<<"*****************************************0";
cout<<"\n\n\n\n";
cout<<"the path summary is";
for(int i=0;i<n_v;i++)
  {
        cout<<" vertex1  "<<" vertex2 "<< " distance ";
        cout<<i<<" --->"<<parent[i]<<" == "<<distance[i];
        cout<<endl;
  }

int sum1=0;
for(int i=0;i<n_v;i++)
    sum1+=distance[i];

cout<<"total cost for all telephone line setup "<<sum1<<endl;

int p,sum;
```

```
cout<<"enter -1 to close";
do
{
        sum=0;
        cout<<"enter the destination";
cin>>p;

cout<<"required path"<<p;
sum=sum+distance[p];
        while(p!=0)
{
        p=parent[p];
        sum+=distance[p];
        cout<<"<--"<<p;

}
cout<<"total path length "<<sum<<endl;

}while(p!=-1);



        return 0;
}
```

Output:

Conclusion:

We understood the implementation of Prims algorithm in real life problems.