**Practical No.7**

**Graph: Minimum Spanning Tree**

**Program Code:-**

**A)** **Prim’s algorithm**

#include<iostream>

using namespace std;

class graph

{

int G[20][20],n;

public:

void accept()

{

int i,j,e;

int src,dest,cost;

cout<<"\nEnter the no. of vertices: ";

cin>>n;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

G[i][j]=0;

}

}

cout<<"\nEnter the no. of Edges: ";

cin>>e;

for(i=0;i<e;i++)

{

cout<<"\nEnter Source: ";

cin>>src;

cout<<"\nDestination: ";

cin>>dest;

cout<<"\nCost: ";

cin>>cost;

G[src][dest]=cost;

G[dest][src]=cost;

}

}

void display()

{

int i,j;

for(i=0;i<n;i++)

{

cout<<"\n";

for(j=0;j<n;j++)

{

cout<<"\t"<<G[i][j];

}

}

}

void prims()

{

int i,j,R[20][20];

int src,dest,cost,count,min;

int total=0;

int visited[20];

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

if(G[i][j]==0)

{

R[i][j]=999;

}

else

R[i][j]=G[i][j];

}

}

for(i=0;i<n;i++)

{

visited[i]=0;

}

cout<<"\nEnter start vertex: ";

cin>>src;

visited[src]=1;

count=0;

while(count<n-1)

{

min=999;

for(i=0;i<n;i++)

{

if(visited[i]==1)

for(j=0;j<n;j++)

{

if(visited[j]!=1)

{

if(min>R[i][j])

{

min=R[i][j];

src=i;

dest=j;

}

}

}

}

cout<<"\nEdge from "<<src<<" to "<<dest<<" \twith cost: "<<min;

total=total+min;

visited[dest]=1;

count++;

}

cout<<"\nTotal Cost: "<<total<<"\n";

}

};

int main()

{

graph g;

g.accept();

g.display();

g.prims();

}