**Name:-Bindiya Chaudhari**

**Roll no:-30**

**Pratical no:-**

**Pratical Name:-Implementation of program to find minimum spanning tree using prim.**

#include"iostream.h"

#include"conio.h"

int COST[7][7]={

{0, 0,0,0,0,0,0},

{0,9999,10,9999,30,45,9999},

{0,10,9999,50,9999,40,25},

{0,9999,50,9999,9999,35,15},

{0,30,9999,9999,9999,9999,20},

{0,45,40,35,9999,9999,55},

{0,9999,25,15,20,55,9999},

};

class GRAPH

{

int n,min\_cost;

//int COST[9][9];

public:

GRAPH(int);

void READ\_GRAPH();

void SHOW\_GRAPH();

void PRIM();

};

GRAPH::GRAPH(int para)

{

n=para;

}

void GRAPH::READ\_GRAPH()

{

cout<<endl<<"Enter adjuncy Matrix";

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

cin>>COST[i][j];

}

void GRAPH::SHOW\_GRAPH()

{

cout<<endl<<"adjuncy Matrix is";

for(int i=1;i<=n;i++)

{

cout<<endl;

for(int j=1;j<=n;j++)

cout<<COST[i][j]<<" ";

}

}

void GRAPH::PRIM()

{

int T[10][3],min=9999,j,k,l,NEAR[10];

min\_cost=0;

for(int i=1;i<=n;i++)

for(j=1;j<=n;j++)

if(COST[i][j]<min)

{

min=COST[i][j];

k=i;

l=j;

}

min cost=COST[k][l];

T[1][1]=k;

T[1][2]=l;

for(i=1;i<=n;i++)

{

if(COST[i][l]<COST[i][k])

NEAR[i]=l;

else

NEAR[i]=k;

}

NEAR[k]=NEAR[l]=0;

for(i=2;i<=n-1;i++)

{

min=9999;

for(k=1;k<=n;k++)

{

if(NEAR[k]!=0 && COST[k]]<min)

{

min=COST[k][NEAR[k]];

j=k;

}

}

T[i][l]=j;

T[i][2]=NEAR[j];

min\_cost=min\_cost+COST[j][NEAR[j]];

NEAR[j]=0;

for(k=1;k<=n;k++)

{

if(NEAR[K]!=0 && COST[k][NEAR[k]]>COST[k][j])

NEAR[k]=j;

}

}

if (min\_cost>=9999)

cout<<endl<<"No spanning tree";

else

cout<<endl<<min\_cost;

}

void main()

{

clrscr();

int v,n;

cout<<endl<<"Enter no of vertices:";

cin>>n;

GRAPH obj(n);

obj.SHOW\_GRAPH();

obj.PRIM();

getch(); }