**Assignment No:-**

**Assignment Name**:- **Write a program to find Minimum-Cost Spanning Trees (Prim’s Algorithm).**

**Name:- WARKE PURVA DILIP.**

**Roll No:- 136.**

#include<iostream.h>

#include<conio.h>

#include<stdlib.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

{

private:

//int COST[10][10];

int n,NEAR[10],T[10][3],mincost;

//int n,A[10][10];

public:

GRAPH(int);

void READ\_GRAPH();

void SHOW\_GRAPH();

void PRIMS();

};

GRAPH::GRAPH(int par)

{

n=par;

}

void GRAPH::READ\_GRAPH()

{

cout<<"\nEnter cost matrix : ";

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

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

cin>>COST[i][j];

cout<<endl;

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

{

cout<<endl;

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

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

}

}

void GRAPH::SHOW\_GRAPH()

{

cout<<endl;

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

{

cout<<endl;

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

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

}

}

int MIN (int a,int b)

{

if(a<b) return a; else return b;

}

void GRAPH::PRIMS()

{

int j,min=9999,k,l;

// search smallest edge (k,l)

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

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

{

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

{

min=COST[i][j];

k=i;l=j;

}

}

// cout<<k<<" "<< l;

//======

mincost=COST[k][l];

T[1][1]=k;

T[1][2]=l;

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

{

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

NEAR[i]=k;

else

NEAR[i]=l;

}

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

//--------------------------

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

{

//---find j such that .....

min=9999;

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

{

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

{

j=k;

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

}

}

// add next edge in the Tree

T[i][1]=j;

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

mincost=mincost+COST[j][NEAR[j]];

NEAR[j]=0;

//update NEAR array

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

{

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

NEAR[k]=j;

}

}

if(mincost>=9999)

cout<<"\nNo spanning Tree";

else

{

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

cout<<T[k][1]<<" "<<T[k][2]<<endl;

cout<<"\nCost of spanning Tree = "<<mincost;

}

}

void main()

{

int n;

clrscr();

cout<<"\nEnter no of nodes : ";

cin>>n;

GRAPH obj(n);

//obj.READ\_GRAPH();

obj.PRIMS();

obj.SHOW\_GRAPH();

getch();

}