**ASSIGNMENT : 3**

**AIM:**

There are flight paths between cities. If there is a flight between city A

and city B then there is an edge between the cities. The cost of the edge can be

the time that flight takes to reach city B from A, or the amount of fuel used for

the journey. Represent this as a graph. The node can be represented by airport

name or name of the city. Use adjacency list representation of the graph or use

adjacency matrix representation of the graph. Justify the storage representations

used.

**CODE:**

#include<iostream>

#define MAX 20

using namespace std;

class dijkstra

{

int city;

int distance[MAX][MAX];

int d[MAX];

int visited[MAX];

public:

void city\_no();

int minvertex();

void matrix\_fill();

void dijkstra\_code();

void display();

};

void dijkstra::city\_no()

{

cout<<"\n enter the number of cities (including cities A and B) : ";

cin>>city;

}

int dijkstra::minvertex()

{

int mvertex=-1;

for(int i=0;i<city;i++)

{

if(visited[i]==0 && (mvertex==-1 || d[i]<d[mvertex]))

mvertex=i;

}

return mvertex;

}

void dijkstra::matrix\_fill()

{

cout<<"\n enter the distances between the cities : ";

for(int i=0;i<city;i++)

{

cout<<"\n For city "<<i<<endl;

for(int j=0;j<city;j++)

{

if(i==j)

distance[i][j]=0;

cin>>distance[i][j];

}

d[i]=INT\_MAX;

visited[i]=0;

}

}

void dijkstra::dijkstra\_code()

{

d[0]=0;

for(int i=0;i<city-1;i++)

{

int mvertex=minvertex();

visited[mvertex]=1;

for(int j=0;j<city;j++)

{

if((distance[mvertex][j]!=0)&&(visited[j]==0))

{

int dist=d[mvertex]+distance[mvertex][j];

if(dist<d[j])

d[j]=dist;

}

}

}

}

void dijkstra::display()

{

cout<<"\n distance of cities from city 0 \n";

cout<<"city Distance\n";

for(int i=0;i<city;i++)

cout<<i<<"\t"<<d[i]<<endl;

}

int main()

{

dijkstra sp;

sp.city\_no();

sp.matrix\_fill();

sp.dijkstra\_code();

sp.display();

return 0;

}

Output:

