**DAA PRACTICAL**

**Name:Parth Borse Roll No.: 07**

**Modified Warshall's**

#include<iostream>

using namespace std;

#define infinity 9999

#define MAX 100

int n; //Number of vertices

int adj[MAX][MAX]; //Weighted Adjancy matrix

int D[MAX][MAX]; //Shortest path matrix

int pred[MAX][MAX]; //Predecessor matrix

void create\_graph();

void floyd\_warshalls();

void findpath(int, int);

void display(int m[MAX][MAX], int);

int main()

{

int s, d;

create\_graph(); //Called function for taking graph as a input

floyd\_warshalls(); //Called function to perform Floyd-Warshall's algorithm

while(1)

{

cout<<"\nEnter source vertex (-1 to exit) : ";

cin>>s;

if(s==-1)

{

break;

}

cout<<"Enter destination vertex : ";

cin>>d;

if(s<0 || s>n-1 || d<0 || d>n-1)

{

cout<<"Enter valid vertices\n\n";

continue;

}

cout<<"Shortest path is : ";

findpath(s, d);

cout<<"Length of the shortest path is : "<<D[s][d]<<endl;

}

return 0;

}

//Function taking graph as an input

void create\_graph()

{

int o, d;

cout<<"Enter number of edges : ";

cin>>n;

cout<<"Enter Adjancy matrix :\n";

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

for(int d=0; d<n; d++)

cin>>adj[o][d];

}

//Function implementing Floyd-Warshall's algoritm

void floyd\_warshalls()

{

int i, j, k;

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

{

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

{

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

{

D[i][j] = infinity;

pred[i][j] = -1;

}

else

{

D[i][j] = adj[i][j];

pred[i][j] = i;

}

}

}

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

{

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

{

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

{

if(D[i][k] + D[k][j] < D[i][j])

{

D[i][j] = D[i][k] + D[k][j];

pred[i][j] = pred[k][j];

}

}

}

}

cout<<"\nShortest path matrix is :\n";

display(D, n);

cout<<"\nPredecessor matrix is :\n";

display(pred, n);

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

{

if(D[i][j]<0)

{

cout<<"Error : negative cycle\n";

exit(1);

}

}

}

//Function displays the matrix

void display(int m[MAX][MAX], int n)

{

int i, j;

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

{

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

{

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

}

cout<<"\n";

}

}

//Function finds path from source to destination

void findpath(int s, int d)

{

int i, path[MAX], count;

if(D[s][d]==infinity)

{

cout<<"There is no path between "<<s<<" to "<<d<<"\n";

return;

}

count = -1;

do

{

path[++count] = d;

d = pred[s][d];

}while(d!=s);

path[++count] = s;

for(i=count; i>0; i--)

{

cout<<path[i]<<" -> ";

}

cout<<path[i]<<endl;

}