/\*

**The Floyd Warshall Algorithm is for solving the All Pairs Shortest Path problem.**

The problem is to find shortest distances between every pair of vertices in a given edge weighted directed Graph.

Time Complexity: O(V^3)

Example:

Input:

graph[][] = { {0, 5, INF, 10},

{INF, 0, 3, INF},

{INF, INF, 0, 1},

{INF, INF, INF, 0} }

which represents the following graph

10

(0)------->(3)

| /|\

5 | |

| | 1

\|/ |

(1)------->(2)

3

Note that the value of graph[i][j] is 0 if i is equal to j

And graph[i][j] is INF (infinite) if there is no edge from vertex i to j.

Output:

Shortest distance matrix

0 5 8 9

INF 0 3 4

INF INF 0 1

INF INF INF 0

Floyd Warshall Algorithm :

We initialize the solution matrix same as the input graph matrix as a first step. Then we update the solution matrix by considering all vertices as an intermediate vertex. The idea is to one by one pick all vertices and updates all shortest paths which include the picked vertex as an intermediate vertex in the shortest path. When we pick vertex number k as an intermediate vertex, we already have considered vertices {0, 1, 2, .. k-1} as intermediate vertices. For every pair (i, j) of the source and destination vertices respectively, there are two possible cases.

1) k is not an intermediate vertex in shortest path from i to j. We keep the value of dist[i][j] as it is.

2) k is an intermediate vertex in shortest path from i to j. We update the value of dist[i][j] as dist[i][k] + dist[k][j] if dist[i][j] > dist[i][k] + dist[k][j]

\*/

**Code :**

#include<stdio.h>

#include<stdlib.h>

//#include<limits.h>

#define INF 999

int V;

void floydWarshall(int[V][V]);

void printSolution(int[V][V],int[V][V]);

void printpath(int[V][V],int,int);

void main()

{

int i,j;

printf("enter no of vertices = ");

scanf("%d",&V);

int dist[V][V];

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

{

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

{

printf("\nenter distance from vetex %d to %d (-1 for infinity) : ",i,j);

scanf("%d",&dist[i][j]);

if(dist[i][j]==-1)

dist[i][j]=INF;

if(i==j&&dist[i][j]!=0)

{

printf("NEGATIVE CYCLE..");

exit(0);

}

}

}

floydWarshall(dist);

return;

}

void floydWarshall(int dist[V][V])

{

int temp[V][V],path[V][V],i,j,k;

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

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

{

temp[i][j]=dist[i][j];

if(i==j) path[i][j]=0;

else if(temp[i][j]!=INF) path[i][j]=i;

else path[i][j]=-1;

}

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

{

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

{

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

{

if(temp[i][k]!=INF&&temp[k][j]!=INF&&temp[i][k]+temp[k][j]<temp[i][j])

{

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

path[i][j]=path[k][j];

}

}

}

}

printSolution(temp,path);

}

void printSolution(int op[V][V],int p[V][V])

{

int i,j;

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

{

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

{

if(op[i][j]<INF)

{

printf("%7d",op[i][j]);

}

else printf("%7s","INF");

}

printf("\n");

}

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

{

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

{

if(i!=j&&p[i][j]!=-1)

{

printf("path from %d to %d : ",i,j);

printf("%d->",i);

printpath(p,i,j);

printf("%d\n",j);

}

}

}

return;

}

void printpath(int p[V][V],int i,int j)

{

if(p[i][j]==i) return;

printpath(p,i,p[i][j]);

printf("%d->",p[i][j]);

}