Floyd warshall’s algorithm is used to find the shortest path between two vertices of a graph. It is mostly used for weighted, directed graphs.

Suppose we have a weighted directed graph:

10

0 -------> 3

| /|\

3 | |

| | 7

\|/ |

1 -------> 2

5

Here, we make a 2-d array storing the distance of a particular vertex from all other vertices. The distance of a vertex from itself is 0, and we assume the distance of a vertex from another vertex which is not joined to it directly as infinite(basically a large number).

So, our 2-d array looks like this:

graph[][]={ {0 , 3 , INF , 10},

{INF , 0 , 5 , INF}, //1 to 0 is INF because the graph is directed from 0 to 1 and not 1 to 0

{INF , INF , 0 , 7},

{INF , INF , INF , 0}}

The algorithm states that, for graph[i][j], if there is an intermediate point k, such that distance of graph[i][k]+graph[k][j] is less than graph[i][j], then the shortest distance between points i and j is the distance from i to k plus distance from k to j. Given that if we are considering k, then all intermediate points {1,2,3…k-1} have been already considered.

The C++ code for the same is as follows:

#include<bits/stdc++.h>

#define INF 999

#define n 4

using namespace std;

int fwgraph(int graph[n][n]) {

int k, shortdist[n][n],i,j;

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

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

shortdist[i][j]=graph[i][j];

//we initially define the shortest distance the same as the input distances, considering no intermediate vertices

}

//We now consider all intermediate vertices uptil n,

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

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

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

if(graph[i][k]+graph[k][j]<graph[i][j]) {

shortdist[i][j]=graph[i][k]+graph[k][j];

}

}

}

}

print(shortdist);

}

int print(int shortdist[n][n]) {

int i,j;

cout<<”The following elements show the shortest distance b/w the respective vertices: “ <<endl;

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

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

if(shortdist[i][j]==INF) cout<<” INF ”;

else cout<<” “<<shortdist[i][j]<<” “;

}

cout<<endl;

}

}

int main() {

int graph[n][n]={ {0 , 3 , INF , 10},

{INF , 0 , 5 , INF},

{INF , INF , 0 , 7},

{INF , INF , INF , 0}};

fwgraph(graph);

return 0;

}

The output given by this will be:

The following elements show the shortest distance b/w the respective vertices:

0 3 8 10

INF 0 5 12

INF INF 0 7

INF INF INF 0