A picture containing text, clipart, businesscard

Description automatically generated

**Assignment**

**(Lab-09 Floyd Warshall)**

Sub Code: CSE246

Sec: 03

**Course Title:** Algorithm

**Semester:** Fall-2021

**Course Instructor:** Jeson Ahammed Ovi, Senior Lecturer, Dept. of

Computer Science & Engineering.

**Submitted To:**

**Jeson Ahammed Ovi**

Senior Lecturer, Dept. of

Computer Science & Engineering

**Submitted By:**

Name     : Kazi Sifat Al Maksud

ID           : 2019-3-60-050

**Problem statement:** Implement Floyd Warshall algorithm in a weighted directed graph to find all pair shortest paths. Print the all-pair shortest paths.

**Theory :** For each vertex pair in a weighted directed graph, the Floyd-Warshall method is a common approach for determining the shortest path. In the all-pair shortest path issue, we must discover all shortest pathways from each vertex in the graph to all other vertices.

As input, we'll use a directed weighted graph G (V, E). And then, from the supplied graph, we create a graph matrix. The graph's edge weights are included in this matrix.

Now, let’s jump into the algorithm:

Graphical user interface, text, application, email

Description automatically generated

Then, at the diagonal points of the matrix, we enter textbfmathsf0. The edge weights from the input graph are used to fill the remaining places.

After that, we must calculate the distance between two vertices. While looking for

**Code:**

#include<iostream>>

using namespace std;

int nameArray[100];

void warshall(int graph[100][100], int n){

for(int m=0; m < n; m++){

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

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

if(graph[i][m] != 666 && graph[m][j] != 666){

graph[i][j] = min(graph[i][j], graph[i][m] + graph[m][j]);

}

}

}

}}

int checkVertexLocation(int u1, int n){

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

if(nameArray[i] == u1){

return i;

}

}

}

int main(){

int graph[100][100], w, nvertice, nedge;

int c, s, e;

cout << "Enter number of Vertices and Edges : ";

cin >> nvertice >> nedge;

for(int i = 0; i < nvertice; i++){

for(int j = 0; j < nvertice; j++){

if(i == j){

graph[i][j] = 0;

} else{

graph[i][j] = 666;

}

}

} cout << "Enter Vertex name: ";

for(int i = 0; i < nvertice; i++){

cin >> c;

nameArray[i] = c;

} for(int i = 0; i < nedge; i++){

cout << "Enter source then destination then weight: ";

cin >> s >> e >> w;

graph[checkVertexLocation(s, nvertice)][checkVertexLocation(e, nvertice)] = w;

} warshall(graph, nvertice);

cout << "\n\nshortest path’s cost of all possible pairs graph is given bellow : \n";

for(int i = 0; i < nvertice; i++){

for(int j = 0; j < nvertice; j++){

if(graph[i][j] != 666){

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

}

else{

cout << "Inf\t\t";

}

}

cout << "\n";

}

cout << "\n\nOne By one distance:\n\n";

for(int i = 0; i < nvertice; i++){

for(int j = 0; j < nvertice; j++){

if(graph[i][j] != 666){

cout << "From " << nameArray[i] << " To " << nameArray[j] << " is: " << graph[i][j] << "\n";

} else{

cout << "From " << nameArray[i] << " To " << nameArray[j] << " is: Inf\n";

}

}

}

}

**Result:**

Text

Description automatically generated

**Conclusion:** Floyd-Warshal performs the same in calculations and in determining rectangular calculations and frameworks. For phase j ≥ 1, as may be the case, the rectangular calculation determines the more rapidly related structure due to the reduction of the sum of the calculations.

==The End==