Bansilal Ramnath Agarwal Charitable Trust's
Vishwakarma Institute of Information Technology

Department of

Artificial Intelligence and Data Science

Name: **Sahil Dilip Sonawane**

Class: **TY**	Division: **C**	Roll No: **373057**
Semester: **V**	Academic Year: **2023-24**	

Subject Name & Code: **Design and Analysis of Algorithm**

Title of Assignment: **Implement All Pair Shortest paths problem using Floyd's Algorithm**.

Assignment No: 04

Aim:

Implement All Pair Shortest paths problem using Floyd's Algorithm.

Problem Statement:

Implement All Pair Shortest paths problem using Floyd's Algorithm.

Background Information:

The Floyd Warshall Algorithm:

The Floyd Warshall Algorithm is for solving all pairs shortest path problems. The problem is to find the shortest distances between every pair of vertices in each edge-weighted directed Graph.

Algorithm:

- Initialize the solution matrix same as the input graph matrix as a first step.
- Then 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.
- k is not an intermediate vertex in shortest path from i to j. We keep the value of dist[i][j] as it is.
- 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]

Software Requirements:

Text Editor: VSCode, Online GDB Compiler

Environment: GCC C++

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
#define V 4
#define INF 99999
void printSolution(int dist[][V]);
void floydWarshall(int graph[][V])
{  int dist[V][V], i, j, k;
  for (i = 0; i < V; i++)
       for (j = 0; j < V; j++)
          dist[i][j] = graph[i][j];
        for (k = 0; k < V; k++) {
       for (i = 0; i < V; i++) {
          for (j = 0; j < V; j++) {
             if (dist[i][j] > (dist[i][k] + dist[k][j])
                && (dist[k][j] != INF
                   && dist[i][k] != INF))
                dist[i][j] = dist[i][k] + dist[k][j];
          }
       }
   } printSolution(dist);
}void printSolution(int dist[][V])
{ cout << "The following matrix shows the shortest "
          "distances"
          " between every pair of vertices \n";
   for (int i = 0; i < V; i++) {
       for (int j = 0; j < V; j++) {
          if (dist[i][j] == INF)
             cout << "INF"
                  << "     ";
```

```
            else
                cout << dist[i][j] << "     ";
        }  cout << endl;
    }}int main()
{
    int graph[V][V] = { { 0, 5, INF, 10 },
                        { INF, 0, 3, INF },
                        { INF, INF, 0, 1 },
                        { INF, INF, INF, 0 } };
  floydWarshall(graph);
    return 0;}
```

Output:

```
The following matrix shows the shortest distances between every pair of vertices
0     5     8     9
INF     0     3     4
INF     INF     0     1
INF     INF     INF     0

--------------------------------
```

Conclusion:

Implemented All Pairs Shortest Path Problem using Floyd Warshall Algorithm.