**UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre**

**EXPERIMENT NO: 09**

**EXPERIMENT TITLE:** To implement Floyd Warshall problem

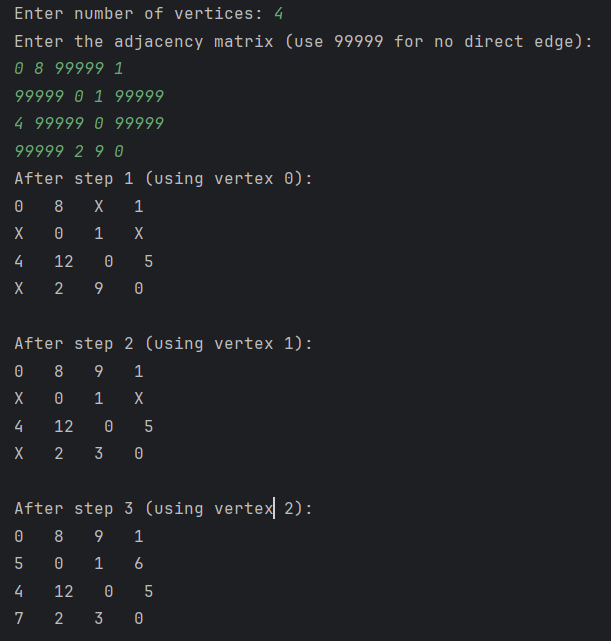
5.1 To implement Floyd Warshall problem.

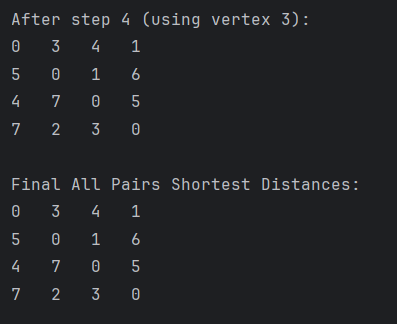
**Objective:**

* To understand the working of the Floyd Warshall algorithm.
* To learn how dynamic programming can be applied to solve the all-pairs shortest path problem.
* To implement a graph as an adjacency matrix and apply the Floyd Warshall algorithm.
* To update and display the shortest path matrix after each intermediate step.
* To represent unreachable paths using a symbolic value (X) for infinity.

**Program code:** -

import java.util.Scanner;  
  
public class FloydWarshall {  
  
 final static int *INF* = 99999; // Representation of infinity  
  
 public static void floydWarshall(int[][] graph, int V) {  
 int[][] dist = new int[V][V];  
  
 // Initialize distance matrix  
 for (int i = 0; i < V; i++)  
 for (int j = 0; j < V; j++)  
 dist[i][j] = graph[i][j];  
  
 // Floyd Warshall core logic with intermediate printing  
 for (int k = 0; k < V; k++) {  
 for (int i = 0; i < V; i++) {  
 for (int j = 0; j < V; j++) {  
 if (dist[i][k] != *INF* && dist[k][j] != *INF* && dist[i][k] + dist[k][j] < dist[i][j])  
 dist[i][j] = dist[i][k] + dist[k][j];  
 }  
 }  
  
 // Print matrix after each step  
 System.*out*.println("After step " + (k + 1) + " (using vertex " + k + "):");  
 *printMatrix*(dist, V);  
 }  
  
 // Final result  
 System.*out*.println("Final All Pairs Shortest Distances:");  
 *printMatrix*(dist, V);  
 }  
  
 public static void printMatrix(int[][] dist, int V) {  
 for (int i = 0; i < V; ++i) {  
 for (int j = 0; j < V; ++j) {  
 if (dist[i][j] == *INF*)  
 System.*out*.print("X ");  
 else  
 System.*out*.print(dist[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
  
 System.*out*.print("Enter number of vertices: ");  
 int V = sc.nextInt();  
  
 int[][] graph = new int[V][V];  
  
 System.*out*.println("Enter the adjacency matrix (use " + *INF* + " for no direct edge (infinity)):");  
 for (int i = 0; i < V; i++) {  
 for (int j = 0; j < V; j++) {  
 graph[i][j] = sc.nextInt();  
 }  
 }  
  
 *floydWarshall*(graph, V);  
 sc.close();  
 }  
}

**Output:**

****

**Conclusion:**

In this experiment, we successfully implemented the Floyd Warshall algorithm to compute the shortest distances between all pairs of vertices in a weighted graph. The algorithm uses dynamic programming to iteratively improve the solution by considering each vertex as an intermediate point. By printing the distance matrix after each step, we clearly observed how the shortest paths evolve. This experiment deepened our understanding of graph algorithms and how they can be applied to real-world problems like network optimization, routing, and transitive closure in databases.