Design & Analysis of Algorithm

Name – Jay Lapani

Roll No. - 21BCP150

Div - 3 , G - 5

```
import java.util.Arrays;
public class FloydWarshallAlgorithm {
 private static final int INF = Integer.MAX_VALUE;
// Infinity value for unavailable paths
 // Main function to find shortest distances between every
pair of vertices
 public static int[][] floydWarshall(int[][]
graph) {
 int n = graph.length; // Number of vertices in the graph
 int[][] dist = new int[n][n]; // Initialize the distance
matrix
 // Copy the input graph matrix to the distance matrix
 for (int i = 0; i < n; i++) {
 for (int j = 0; j < n; j++) {
 dist[i][j] = graph[i][j];
 }
 }
 // Calculate shortest path between every pair of vertices
through intermediate vertices
 for (int k = 0; k < n; k++) {
 for (int i = 0; i < n; i++) {
 for (int j = 0; j < n; 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];
 }
 }
 }
 }
 return dist; // Return the resulting distance matrix
 }
 public static void main(String[] args) {
 // Test graph
 int[][] graph = {{0, 5, INF, 10},
 {INF, 0, 3, INF},
 {INF, INF, 0, 1},
 {INF, INF, INF, 0}};
```

```
 int[][] dist = floydWarshall(graph);
 // Print the resulting distance matrix
 for (int i = 0; i < dist.length; i++) {

System.out.println(Arrays.toString(dist[i]));
 }
 }
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                    Code + v

PS D:\Assignment> cd "d:\Assignment\" ; if ($?) { javac FloydWarshallAlgorithm.java } ; if ($?) { java FloydWarshallAlgorithm }
[0, 5, 8, 9]
[2147483647, 0, 3, 4]
[2147483647, 2147483647, 0, 1]
[2147483647, 2147483647, 2147483647, 0]
PS D:\Assignment>
```