**Practical 7: Floyd Warshall – Dynamic Programming**

**Aim**

To Implement the Floyd-Warshall Algorithm for All Pair Shortest Path Problem using Dynamic Programming.

**Algorithm**

1. Initialize distance and pred matrices with infinity and null values respectively, except for the diagonal elements which are initialized to 0.
2. For each vertex k from 1 to n, do the following:
   1. For each pair of vertices i and j from 1 to n, check if the path from i to k and then from k to j is shorter than the current path from i to j. If it is, update the distance and pred matrices accordingly.
3. Return the distance and pred matrices.

**Program**

|  |
| --- |
| public class FloydWarshall {  static int *INF* = 9999;  public static void main(String[] args) {  int graph[][] = {  {0, 5, *INF*, 10},  {*INF*, 0, 3, *INF*},  {*INF*, *INF*, 0, 1},  {*INF*, *INF*, *INF*, 0}  };  *floydWarshall*(graph);  }  static void floydWarshall(int[][] graph) {  int V = graph.length;  int[][] matrix = new int[V][V];  for(int i = 0; i < V; i ++) {  for(int j = 0; j < V; j ++) {  matrix[i][j] = graph[i][j];  }  }  for(int i = 0; i < V; i ++) {  for(int j = 0; j < V; j ++) {  for(int k = 0; k < V; k ++) {  if(matrix[j][k] > matrix[i][k] + matrix[j][i]) {  matrix[j][k] = matrix[i][k] + matrix[j][i];  }  }  }  }  *printMatrix*(matrix);  }  static void printMatrix(int[][] matrix) {  System.*out*.println("Resultant Matrix using Floyd Warshall is: ");  int V = matrix.length;  for(int i = 0; i < V; i ++) {  System.*out*.print("[");  for(int j = 0; j < V; j ++) {  if(matrix[i][j] == *INF*) {  System.*out*.print("INF ");  }  else {  System.*out*.print(matrix[i][j] + " ");  }  }  System.*out*.print("]");  System.*out*.println();  }  } } |
| **Output:** |
|  |

**Analysis of Algorithm**

**Time Complexity:**

The time complexity of the Floyd-Warshall algorithm is **O(V3)**, where V is the number of vertices in the graph. The algorithm iteratively considers all possible intermediate vertices in the shortest path calculation, resulting in a **nested triple loop** that performs V3 operations. This time complexity is independent of the specific graph structure or edge weights, making it a suitable algorithm for a wide range of graph problems.

**Space Complexity:**

The space complexity of the Floyd-Warshall algorithm is **O(V2)**, where V is the number of vertices in the graph. This is because the algorithm requires a matrix of size V x V to store the shortest path distances between all pairs of vertices.