**Experiment No. : 6**

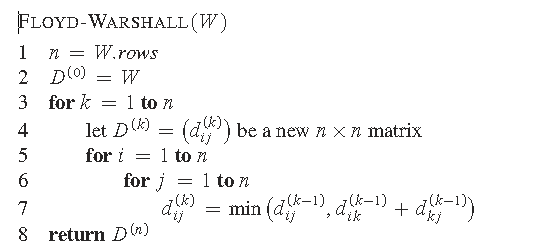
**Title: Floyd-Warshall Algorithm using Dynamic**

**programming approach**

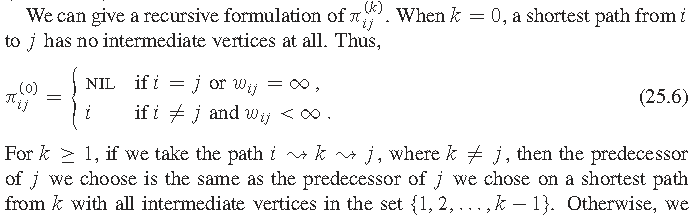
**Batch: B2 Roll No.: 16010421119 Experiment No.: 6**

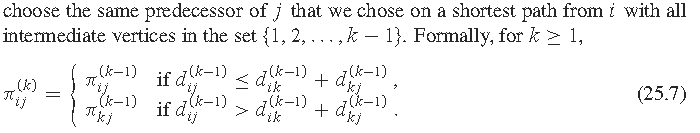
**Aim:** To Implement All pair shortest path Floyd-Warshall Algorithm using Dynamic programming approach and analyse its time Complexity.

**Algorithm of Floyd-Warshall Algorithm:**



**Constructing Shortest Path:**



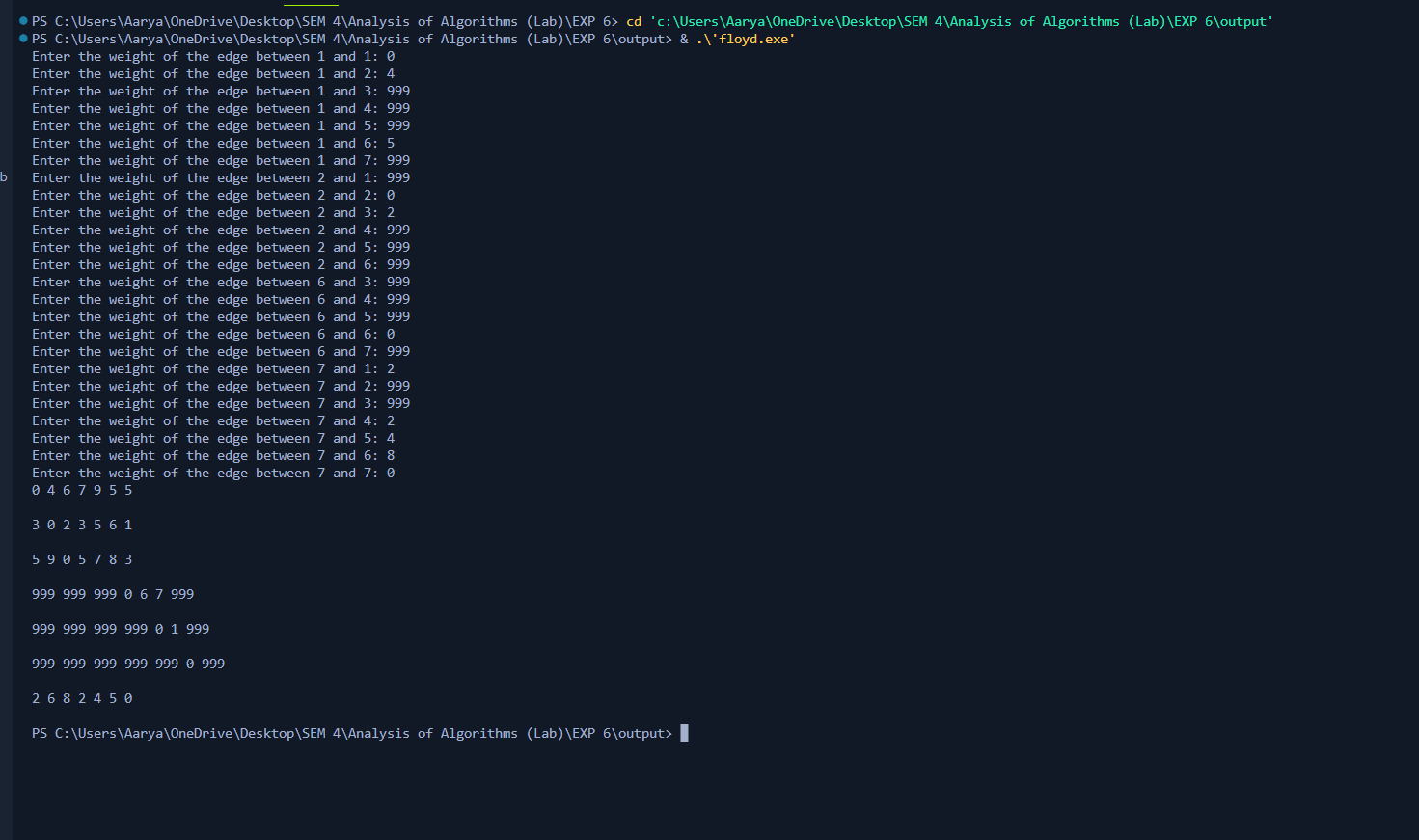


**Working of Floyd-Warshall Algorithm:**

**Problem Statement**

Find Shortest Path for each source to all destinations using Floyd-Warshall Algorithm for the following graph

****

****

**Derivation of Floyd-Warshall Algorithm:**

* Time complexity Analysis

**There are three loops. Each loop has constant complexities. So, the time complexity of the Floyd-Warshall algorithm is O(n3).**

* Space complexity Analysis

**The space complexity of the Floyd-Warshall algorithm is O(n2).**

**Program(s) of Floyd-Warshall Algorithm:**

**#include <iostream>**

**using namespace std;**

**#define size 4**

**// #define INF 999**

**void printMatrix(int matrix[][size]);**

**void floydWarshall(int graph[][size])**

**{**

**int matrix[size][size], i, j, k;**

**// for (i = 0; i < size; i++)**

**// for (j = 0; j < size; j++)**

**// matrix[i][j] = graph[i][j];**

**for (k = 0; k < size; k++)**

**{**

**for (i = 0; i < size; i++)**

**{**

**for (j = 0; j < size; j++)**

**{**

**if (matrix[i][k] + matrix[k][j] < matrix[i][j])**

**matrix[i][j] = matrix[i][k] + matrix[k][j];**

**}**

**}**

**}**

**printMatrix(matrix);**

**}**

**void printMatrix(int matrix[][size])**

**{**

**for (int i = 0; i < size; i++)**

**{**

**for (int j = 0; j < size; j++)**

**{**

**// if (matrix[i][j] == INF)**

**// printf("%4s", "INF");**

**// else**

**cout<< matrix[i][j]<<" ";**

**}**

**cout<<endl;**

**printf("\n");**

**}**

**}**

**int main()**

**{**

**int graph[size][size];**

**for (int i = 0; i < size; i++)**

**{**

**for (int j = 0; j < size; j++)**

**{**

**cout << "Enter the weight of the edge between " << i+1 << " and " << j+1 << ": ";**

**cin >> graph[i][j];**

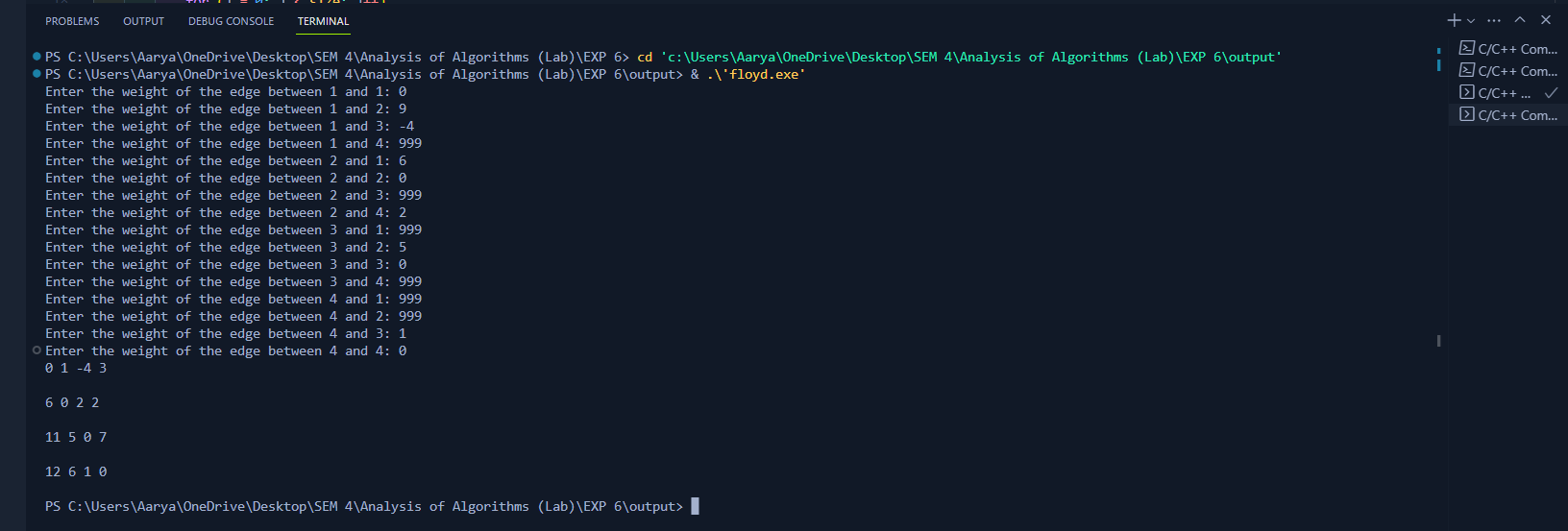
**}**

**}**

**floydWarshall(graph);**

**}**

**Output(o) of Floyd-Warshall Algorithm:**



**Post Lab Questions:-** Explain dynamic programming approach for Floyd-Warshall algorithm and write the various applications of it.

Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. This algorithm works for both the directed and undirected weighted graphs. But, it does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative).

Applications of the Floyd Warshall Algorithm

* Fast computation of Pathfinder networks.
* Inversion of real matrices.
* The transitive closure of directed graphs.
* Checking if an undirected graph is bipartite or not.
* Shortest path in a graph.

**Conclusion: (Based on the observations):**

We can conclude that we have learnt about Floyd-Warshall Algorithm..

**Outcome:**

CO3 :- Implement Backtracking and Branch-and-bound algorithms

**References:**

1. Richard E. Neapolitan, " Foundation of Algorithms ", 5th Edition 2016, Jones & Bartlett Students Edition
2. Harsh Bhasin , " Algorithms : Design & Analysis", 1st Edition 2013, Oxford Higher education, India
3. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
4. Jon Kleinberg, Eva Tardos, " Algorithm Design", 10th Edition 2013, Pearson India Education Services Pvt. Ltd.