K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: B2	**Roll No.:** 16010121110
Experiment No._7_	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implementation of All Pair Shortest Path using Dynamic Programming

Objective To learn the All-Pair Shortest Path using Floyd-Warshall's algorithm

CO to be achieved:

CO 2 Describe various algorithm design strategies to solve different problems and analyse Complexity.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajasekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algorithms",2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://users.cecs.anu.edu.au/~Alistair.Rendell/Teaching/apac_comp3600/module4/all_pairs_shortest_paths.xhtml**
4. **https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/**
5. **http://www.cs.bilkent.edu.tr/~atat/502/AllPairsSP.ppt**

Theory:

It aims to figure out the shortest path from each vertex v to every other u.

1. In all pairs shortest path, when a weighted graph is represented by its weight matrix W then the objective is to find the distance between every pair of nodes.
2. Apply dynamic programming to solve the all pairs shortest path.
3. In the all pair shortest path algorithm, we first decomposed the given problem into subproblems.
4. In this, the principle of optimality is used for solving the problem.
5. It means any subpath of shortest path is a shortest path between the end nodes.

Algorithm:

```
Algorithm All_pair(W, A)
{
For i = 1 to n do
For j = 1 to n do
A [i , j] = W [i , j]
For k = 1 to n do
        {
        For i = 1 to n do
                {
                For j = 1 to n do
                {
                A [i , j] = min(A [i, j], A [i, k] + A [k, j])
                }
                }}}
```

Example and Solution for the example:

B2 16010121110 APSP

$k=1$ $A^1(1,2) =$ Min $(d(1,2)) =$ No change
$A^1(1,3) =$ Min $(d(1,3)) =$ No change
$A^1(2,3) =$ Min$(d(2,3), d(2,1)+d(1,3)) =$ No change
$A^1(3,2) =$ Min$(d(3,2), d(3,1)+d(1,2)) =$ Min$(\infty, 7) = 7$

$k=2$ $A^2(1,2) \rightarrow$ No change
$A^2(1,3) \rightarrow$ Min$(A^0(1,2) + A^1(2,3), A(1,3))$
$=$ Min$(6, 11) = 6$
$A(3,1) \rightarrow$ Min$(\infty, 7) = 7$

$k=3$ $A(1,2) \rightarrow$ No change
$A(2,3) \rightarrow$ No change

Floyd Warshal $\rightarrow$ Dynamic programming

$\rightarrow$ Used to find shortest path from one node to another for all Nodes

$$A^k(i,j) = \text{Min}\begin{cases} A^{k-1}(i,j) \\ \underset{\forall k}{\text{Min}}\, A^{k-1}(i,k) + A^{k-1}(k,j) \end{cases}$$

Matrix of Previous iteration is used for the Next iteration

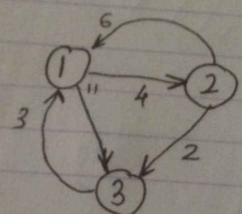
Time Complexity $i \times j \times k$
$\downarrow \downarrow \downarrow$
$n \; n \; n = O(n^3)$

Space complexity $\rightarrow$ $O(n^3)$ if we keep all matrices
$O(n^2)$ if only required (2) matrices used.

1601012110 Aatmaj B2

$$W_{ij} = \begin{cases} 0 & i=j \\ w(i,j) & i\neq j \\ \infty & \text{Not Included} \end{cases}$$

$$A^k(i,j) = \text{Min}\left(A^{k-1}(i,j),\ \text{Min}\left\{ A^{k-1}(i,k) + A^{k-1}(k,j) \right\} \right) \quad \forall k$$



$$\rightarrow A^0 = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 11 \\ 2 & 6 & 0 & 2 \\ 3 & 3 & \infty & 0 \end{array}$$

$$A^1 = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 4 & 11 \\ 2 & 6 & 0 & 2 \\ 3 & 3 & \underline{\underline{7}} & 0 \end{array}$$

Go to 1 in between
eg 2 3 is 2-1-3

$$A^2 = \begin{bmatrix} 0 & 4 & \underline{\underline{6}} \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

(1,2) → (1-2)
(1-3) → 1 → 2 → 3
(3-2) → 3 → 1 → 2
⋮

$$A^3 = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Analysis of algorithm:

O(n^3) As there are three loops with size n

```
/*
Warshall algo for all pair shortest path


*/
public class Main
{
        public static void main(String[] args) {
    int [] [] matrix = {{ 0, 3, 1000, 7 },
                        { 8, 0, 2, 1000 },
                        { 5, 1000, 0, 1 },
                        { 2, 1000, 1000, 0 } };

    int[][] matrix2 = new int[4][4];

    for(int k=0;k<4;k++){//hor

    for(int i=0;i<4;i++){//hor
        for(int j=0;j<4;j++){//vert
       if(i==k | j==k | i==j){ //for target element and diagonal
          matrix2[i][j]=matrix[i][j];
       }
     //  System.out.println(matrix[i][k]+matrix[k][j]);
       if((matrix[i][k]+matrix[k][j])<matrix[i][j]){
          matrix2[i][j]=matrix[i][k]+matrix[k][j];
       }
       else{
          matrix2[i][j]=matrix[i][j];
       }
    }
    }

    //copy matrix
    for(int i=0;i<4;i++){//hor
        for(int j=0;j<4;j++){//vert
        // System.out.print(matrix2[i][j]+",");
       matrix[i][j]=matrix2[i][j];
        }
     //  System.out.println();

       }
    }
```

```
        for(int i=0;i<4;i++){//hor
         for(int j=0;j<4;j++){//vert
         System.out.print(matrix2[i][j]+",");
         }
         System.out.println();

        }
                System.out.println("Hello World");
          }
    }
```

CONCLUSION: Thus we have understood the working of all pair shortest path algorithm using the Floyd-warshall algorithm. We have written code and solved examples for the same. Floyd-warshall algorithm is an algorithm which is used to find the shortes path from, one node to all other nodes in any graph. It takes O(n^3) time complexity and O(n^2) space complexity.