Batch: E-2 Roll No.: 16010123325

Experiment No._7_

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Study, Implementation and Analysis of All Pair Shortest Path.

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.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",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:

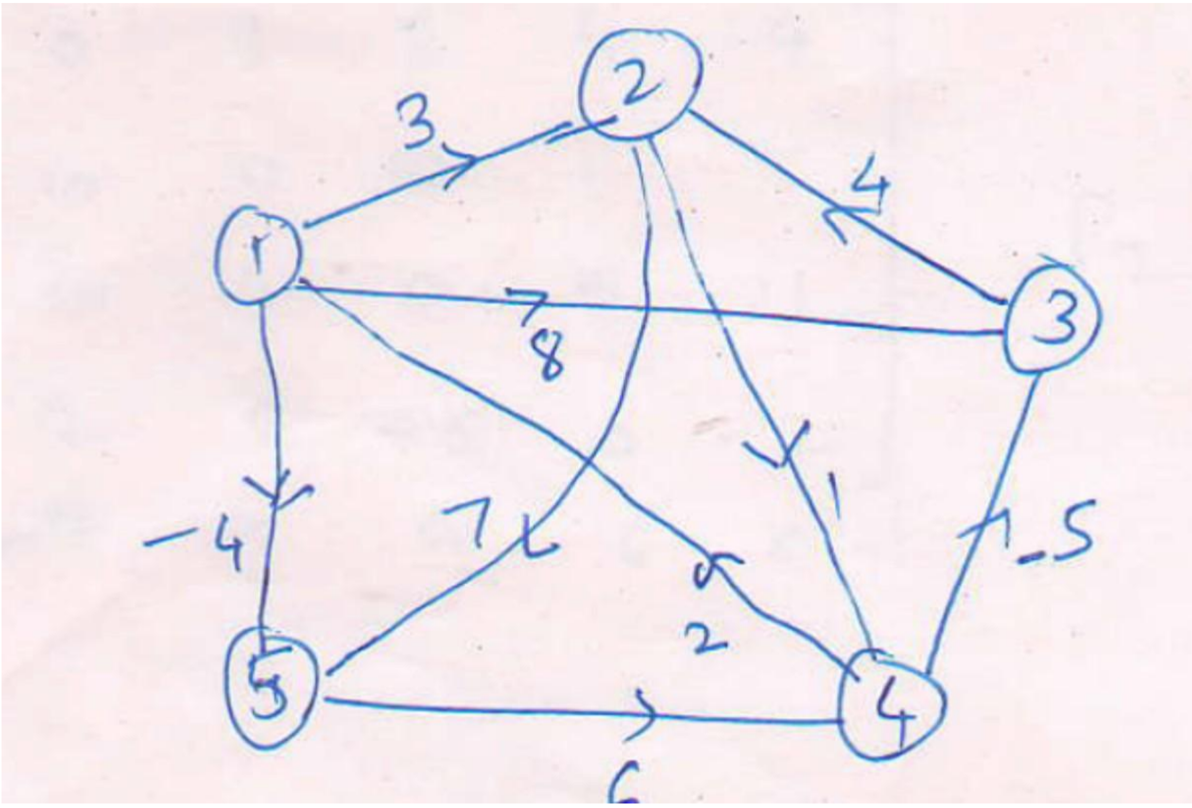
It aims to figure out the shortest path from each vertex v to every other u.

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

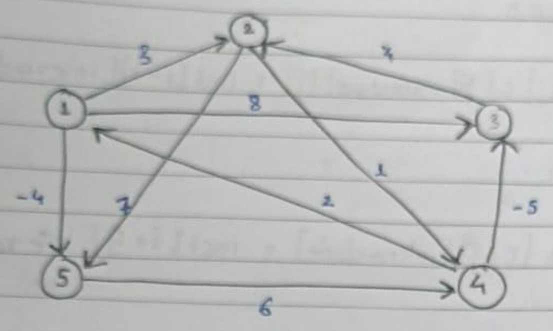
Algorithm:

```
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 :



Solution for the example:



$D[i][j] = \min(D[i][j],\ D[i][k] + D[k][j])$

Initial Matrix

D^0	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

D^1	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D[2,5] = \infty$

$D[2,4] = \min(D[2,4],\ D[2,1] + D[1,4])$

$= \min(1,\ 3)$

$D[2,4] = 1$

$[3,4]$:

$D[3,4] = \min(D[3,4],\ D[3,2] + [2,4])$

∞ 4+1

1 4

5

D^2	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

D^3	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

D^4	1	2	3	4	5
1	0	3	8	4	-4
2	3	0	-4	1	-1
3	∞	4	0	5	∞
4	2	5	-5	0	-2
5	8	∞	∞	6	0

$D^5 \Rightarrow$ Final

D^5	1	2	3	4	5
1	0	3	8	4	-4
2	3	0	-4	1	-1
3	∞	4	0	5	∞
4	2	5	-5	0	-2
5	8	11	5	6	0

Time complexity:-

```
for(int k=0; k<n; k++) {
    for(i=0; i<n; i++) {
        for(j=0; j<n; j++) {
            D[i][j] = min(D[i][j], D[i][k] + D[k][j]);
        }
    }
}
```

∴ Time complexity : $O(n^3)$

Recurrence formula

$$D^{k}[i,j] = \min\{ D^{(k-1)}(i,j),\ D^{k-1}(i,k) + D^{k-1}(k,j) \}$$

12.13 pm.

27/3

Code-

```
import java.util.*;

class floyd {
    static void floydWarshall(int[][] graph) {
        int V = graph.length;

        for (int k = 0; k < V; k++) {
            for (int i = 0; i < V; i++) {
                for (int j = 0; j < V; j++) {
                    if ((graph[i][j] == -1 || graph[i][j] > (graph[i][k] +
graph[k][j]))
                        && (graph[k][j] != -1 && graph[i][k] != -1)) {
                        graph[i][j] = graph[i][k] + graph[k][j];
                    }
                }
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of vertices: ");
        int V = scanner.nextInt();
        int[][] graph = new int[V][V];

        System.out.println("Enter the adjacency matrix (-1 for no direct edge):");
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                graph[i][j] = scanner.nextInt();
            }
        }

        floydWarshall(graph);

        System.out.println("The shortest distance matrix is:");
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                System.out.print(graph[i][j] + " ");
            }
            System.out.println();
        }

        scanner.close();
    }
}
```

Output-

```
Enter the number of vertices: 5
Enter the adjacency matrix (-1 for no direct edge):
0 4 -1 5 -1
-1 0 1 -1 6
2 -1 0 3 -1
-1 -1 1 0 2
1 -1 -1 4 0
The shortest distance matrix is:
0 4 5 5 7
3 0 1 4 6
2 6 0 3 5
3 7 1 0 2
1 5 5 4 0
```

Analysis of algorithm:

Three loops are present. The complexity of each loop never changes. Consequently, the Floyd-Warshall algorithm's time complexity is O (n^3).

The Floyd-Warshall algorithm's space complexity is O (n^2).

CONCLUSION:

The above experiment highlights implementation of Floyd Warshall algorithm to find All Pair Shortest Path.