Experiment No. : 6

Title: Floyd-Warshall Algorithm using Dynamic programming approach

Batch: A2 **Roll No.: 16010421073** **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:

FLOYD-WARSHALL(W)

$n = W.rows$
$D^{(0)} = W$
for $k = 1$ **to** n
  let $D^{(k)} = \left(d_{ij}^{(k)}\right)$ be a new $n \times n$ matrix
  for $i = 1$ **to** n
    for $j = 1$ **to** n
      $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
return $D^{(n)}$

Constructing Shortest Path:

We can give a recursive formulation of $\pi_{ij}^{(k)}$. When $k = 0$, a shortest path from i to j has no intermediate vertices at all. Thus,

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty , \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty . \end{cases} \tag{25.6}$$

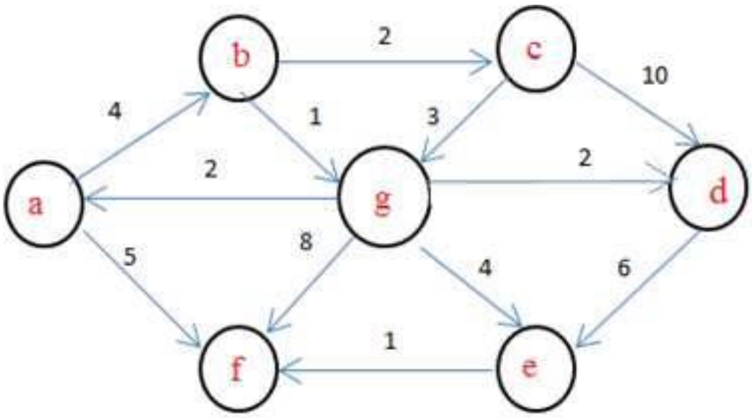
For $k \geq 1$, if we take the path $i \leadsto k \leadsto j$, where $k \neq j$, then the predecessor of j we choose is the same as the predecessor of j we chose on a shortest path from k with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. Otherwise, we choose the same predecessor of j that we chose on a shortest path from i with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. Formally, for $k \geq 1$,

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} , \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} . \end{cases} \tag{25.7}$$

Working of Floyd-Warshall Algorithm:

Problem Statement

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



Solution

Derivation of Floyd-Warshall Algorithm:

Time complexity Analysis

- Floyd Warshall Algorithm consists of three loops over all the nodes.
- The inner most loop consists of only constant complexity operations.
- Hence, the asymptotic complexity of Floyd Warshall algorithm is $O(n^3)$.
- Here, n is the number of nodes in the given graph.

Program(s) of Floyd-Warshall Algorithm:

```
#include<iostream>
using namespace std;
#define V 4
#define INF 99999

void printSolution(int dist[][V]);

void floyd_Warshall(int graph[][V])
{
    int dist[V][V], i, j, k;
    for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
            dist[i][j] = graph[i][j];
    for (k = 0; k < V; k++) {
        for (i = 0; i < V; i++)
        {
            for (j = 0; j < V; j++)
            {
                if (dist[i][j] > (dist[i][k] + dist[k][j]) && (dist[k][j] != INF
&& dist[i][k] != INF))
                    dist[i][j] = dist[i][k] + dist[k][j];
```

```
            }
        }
    }
    printSolution(dist);
}

void printSolution(int dist[][V]){
    cout << "The following matrix shows the shortest distances between every pair
of vertices \n";
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            if (dist[i][j] == INF)
                cout << "INF"<< "   ";
            else
                cout << dist[i][j] << "     ";
        }
    cout << "\n";
    }
}

int main()
{
    int graph[V][V] = { { 0, 8, INF, 16 },
                        { INF, 0, 3, INF },
                        { INF, 7, 0, 12 },
                        { INF, INF, INF, 0 } };
    floyd_Warshall(graph);
    return 0;
}
```

Output(o) of Floyd-Warshall Algorithm:

```
The following matrix shows the shortest distances between every pair of vertices
0    8    11    16
INF  0    3    15
INF  7    0    12
INF  INF  INF  0
```

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

The Floyd-Warshall algorithm is a classic example of a dynamic programming approach to finding the shortest path between all pairs of vertices in a graph.

It is based on the idea of solving subproblems and combining their solutions to obtain the optimal solution for the whole problem.

The dynamic programming approach for Floyd-Warshall algorithm can be summarized as follows:

1. We create a matrix D of size n x n, where n is the number of vertices in the graph. The entry D[i][j] will hold the length of the shortest path from vertex i to vertex j.
2. We initialize the matrix D with the lengths of the edges in the graph. If there is no edge between two vertices, we set the length to infinity.
3. We then use a nested loop to update the matrix D. For each pair of vertices i and j, we consider all intermediate vertices k and 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, we update the value of D[i][j] to the new, shorter path.
4. After the nested loop has finished executing, the matrix D will hold the shortest path between all pairs of vertices in the graph.

The Floyd-Warshall algorithm has several applications, including:

- Finding the shortest path between all pairs of vertices in a graph.
- Detecting negative cycles in a graph.
- Finding the transitive closure of a directed graph.
- Solving the all-pairs shortest path problem in a weighted graph.
- Finding the shortest path in a weighted graph with negative edges.

Conclusion: (Based on the observations):

Thus we successfully implemented Floyd-Warshall Algorithm using Dynamic programming approach and analyse its time Complexity.

Outcome:

CO2 : Implement Greedy and Dynamic Programming 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.