Experiment No: 07

Aim: To implement LCS problem using dynamic programming approach.

Theory:

If a set of sequences are given, the longest common subsequence problem is to find a common subsequence of all the sequences that is of maximal length.

```
Algorithm: LCS-Length-Table-Formulation (X, Y)
m :=
length(X
)n :=
length(Y
)
for i = 1
   to m do
   C[i, 0]
   := 0
for j = 1
   to n do
   C[0, j]
   := 0
for i = 1 to m
   do for j =
   1 to n do
      if xi = yj
         C[i, j] := C[i - 1, j -
         1] + 1B[i, j] := 'D'
      else
         if C[i -1, j] ≥ C[i, j -1]
            C[i, j] := C[i - 1, j] + 1
            B[i, j] := 'U'
         else
         C[i, j] := C[i, j - 1]
         B[i, j] := 'L'
return C and B
Algorithm: Print-LCS (B, X, i, j)
if i = 0 and
   j = 0
   return
if B[i, j] = 'D'
   Print-LCS(B, X, i-1, j-1)
   Print(xi)
else if B[i, j] = 'U'
   Print-LCS(B, X,
i-1, j)else
   Print-LCS(B, X, i, j-1)
```

Conclusion:

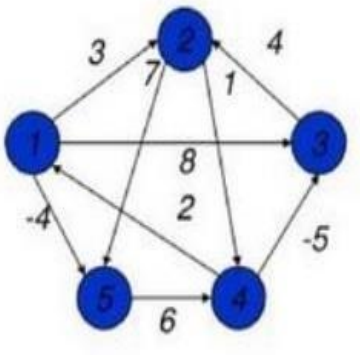
To populate the table, the outer **for** loop iterates ***m*** times and the inner **for** loop iterates ***n*** times.Hence, the complexity of the algorithm is *O(m, n),* where ***m*** and ***n*** are the length of two strings.

Experiment No: 08

Aim: Implement All Pair Shortest Path Algorithm

Theory:

The all pair shortest path algorithm is also known as Floyd-Warshall algorithm is used to find all pair shortest path problem from a given weighted graph. As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to all other nodes in the graph.



$$\begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

At first the output matrix is same as given cost matrix of the graph. After that the output matrix will be updated with all vertices k as the intermediate vertex.

The time complexity of this algorithm is O(V3), here V is the number of vertices in the graph.

Input − The cost matrix of the graph.

```
0 3 6 ∞ ∞ ∞ ∞
3 0 2 1 ∞ ∞ ∞
6 2 0 1 4 2 ∞
∞ 1 1 0 2 ∞ 4
∞ ∞ 4 2 0 2 1
∞ ∞ 2 ∞ 2 0 1
∞ ∞ ∞ 4 1 1 0
```

Output − Matrix of all pair shortest path.

```
0 3 4 5 6 7 7
3 0 2 1 3 4 4
4 2 0 1 3 2 3
5 1 1 0 2 3 3
6 3 3 2 0 2 1
7 4 2 3 2 0 1
7 4 3 3 1 1 0
```

Algorithm:

floydWarshal(cost)
Input − The cost matrix of given Graph.
Output − Matrix to for shortest path between any vertex to any vertex.

```
Begin
   for k := 0 to n, do
      for i := 0 to n, do
         for j := 0 to n, do
            if cost[i,k] + cost[k,j] < cost[i,j], then
               cost[i,j] := cost[i,k] + cost[k,j]
            done
         done
      done
      display the current cost matrix
End
```

Analysis:
The time complexity of this algorithm is $O(V^3)$, here V is the number of vertices in the graph.

Conclusion: Thus we implemented all pair shortest path algorithm.