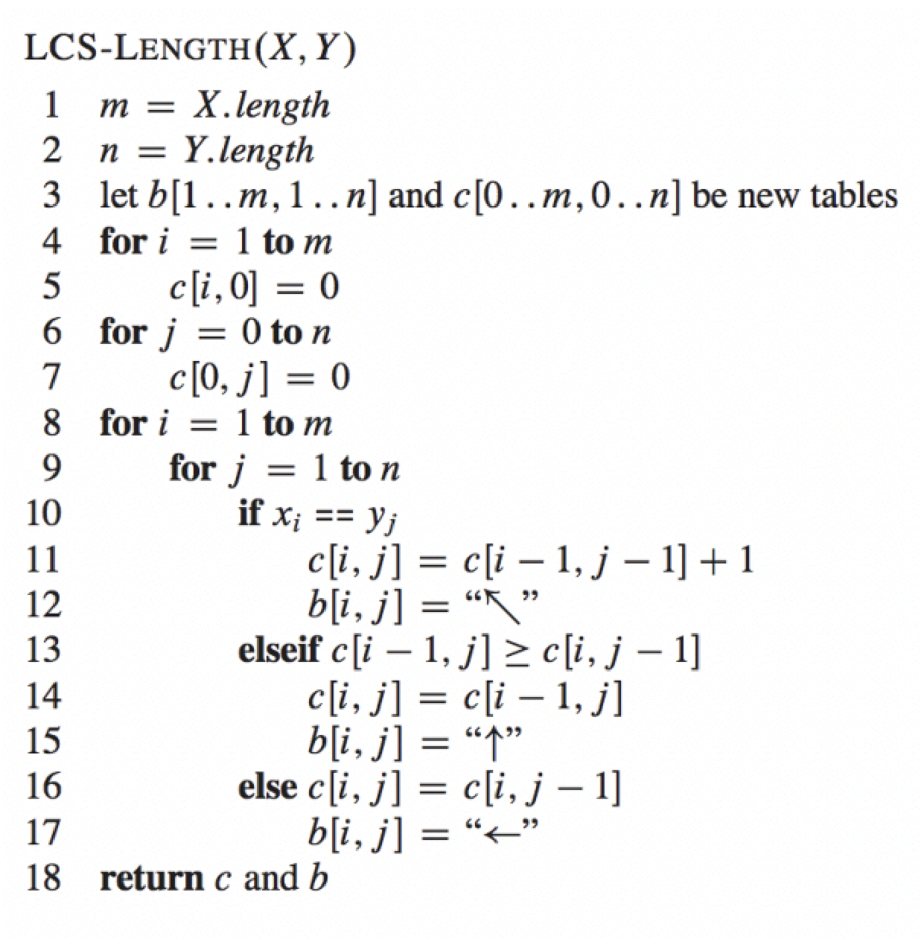
Experiment No. : 6

Title: Longest Common Subsequence using Dynamic Programming

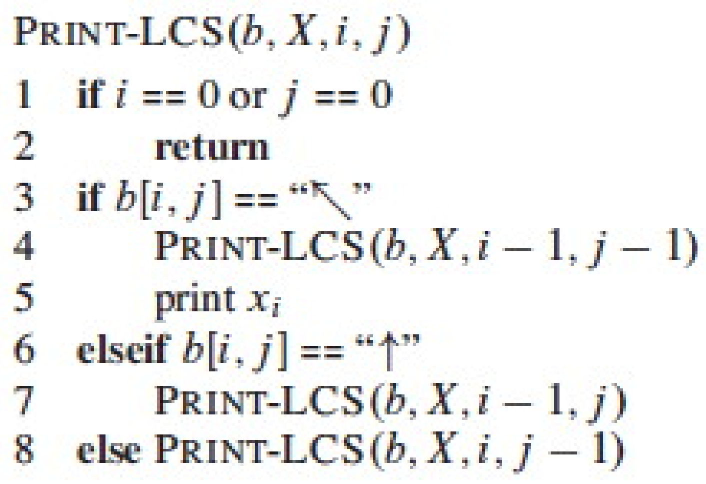
Batch:SY-IT(B3) Roll No.: 16010423076 Experiment No.:6

Aim: To Implement Longest Common Subsequence using Dynamic Programming and analyse its time Complexity.

Algorithm of Longest Common Subsequence:



```
LCS-LENGTH(X, Y)
1   m = X.length
2   n = Y.length
3   let b[1..m, 1..n] and c[0..m, 0..n] be new tables
4   for i = 1 to m
5       c[i, 0] = 0
6   for j = 0 to n
7       c[0, j] = 0
8   for i = 1 to m
9       for j = 1 to n
10          if $x_i == y_j$
11              c[i, j] = c[i − 1, j − 1] + 1
12              b[i, j] = "↖"
13          elseif c[i − 1, j] ≥ c[i, j − 1]
14              c[i, j] = c[i − 1, j]
15              b[i, j] = "↑"
16          else c[i, j] = c[i, j − 1]
17              b[i, j] = "←"
18  return c and b
```



```
PRINT-LCS(b, X, i, j)
1   if i == 0 or j == 0
2       return
3   if b[i, j] == "↖"
4       PRINT-LCS(b, X, i − 1, j − 1)
5       print $x_i$
6   elseif b[i, j] == "↑"
7       PRINT-LCS(b, X, i − 1, j)
8   else PRINT-LCS(b, X, i, j − 1)
```

Explanation and Working of Longest Common Subsequence with Example:

Initialize a 2D array `dp[m+1][n+1]`, where m and n are the lengths of the two given sequences.

Fill the table using the recurrence relation:

- If `X[i] == Y[j]`, then `dp[i][j] = dp[i-1][j-1] + 1` (include the character).
- Otherwise, `dp[i][j] = max(dp[i-1][j], dp[i][j-1])` (exclude one character).

Backtrack through the table to construct the LCS.
Return the final result from `dp[m][n]`.

Consider two sequences:
X = "ABCDGH"
Y = "AEDFHR"

The LCS of these two strings is "ADH", with a length of 3.

The table for LCS calculation is filled dynamically using the above algorithm. The final LCS is derived by tracing back the table values.

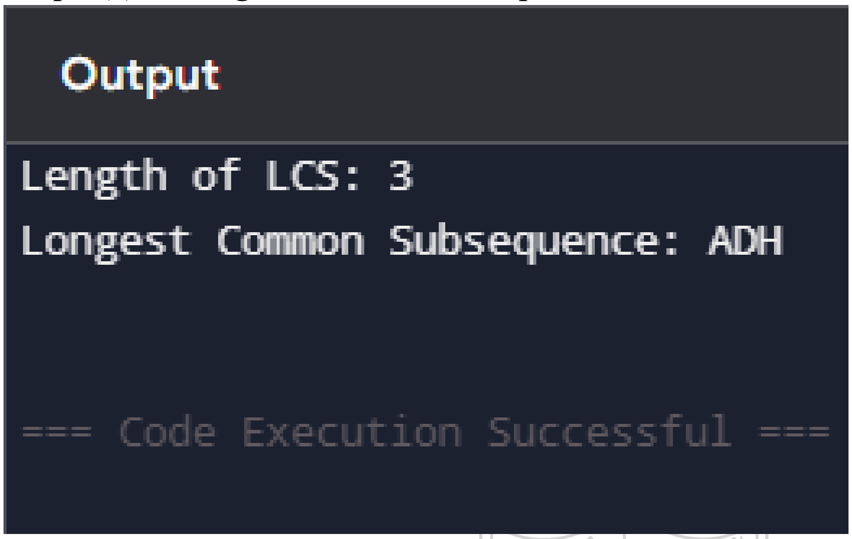
LCS Table (Dynamic Programming Approach)

	∅	**A**	**E**	**D**	**F**	**H**	**R**
∅	0	0	0	0	0	0	0
A	0	1	1	1	1	1	1
B	0	1	1	1	1	1	1
C	0	1	1	1	1	1	1
D	0	1	1	2	2	2	2
G	0	1	1	2	2	2	2
H	0	1	1	2	2	3	3

Program(s) of Longest Common Subsequence:
#include <stdio.h>

```
#include <string.h>

// Function to find the LCS
void lcs(char *X, char *Y, int m, int n) {
    int dp[m + 1][n + 1];

    // Fill dp table using bottom-up approach
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (X[i - 1] == Y[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];
        }
    }

    // Length of LCS
    int length = dp[m][n];
    printf("Length of LCS: %d\n", length);

    char lcsStr[length + 1];
    lcsStr[length] = '\0';

    // Trace the LCS path
    int i = m, j = n, index = length;
    while (i > 0 && j > 0) {
        if (X[i - 1] == Y[j - 1]) {
            lcsStr[index - 1] = X[i - 1];
            i--;
            j--;
            index--;
        } else if (dp[i - 1][j] > dp[i][j - 1])
            i--;
        else
            j--;
    }

    // Print the result
    printf("Longest Common Subsequence: %s\n", lcsStr);
}

// Driver Code
int main() {
    char X[] = "ABCDGH";
    char Y[] = "AEDFHR";
    int m = strlen(X);
    int n = strlen(Y);
    lcs(X, Y, m, n);
    return 0;
}
```

Output(o) of Longest Common Subsequence:



```
Output

Length of LCS: 3
Longest Common Subsequence: ADH


=== Code Execution Successful ===
```

Post Lab Questions:-

Explain dynamic programming approach and write the various applications of it.

Solution :

Dynamic Programming Approach:
Dynamic programming is a method for solving problems by breaking them into subproblems, solving each subproblem once, and storing its result to avoid redundant computations. It is useful for optimization problems with overlapping subproblems and optimal substructure properties.

Applications of Dynamic Programming:

1. Longest Common Subsequence (LCS)
2. Knapsack Problem
3. Fibonacci Series
4. Matrix Chain Multiplication
5. Shortest Path Algorithms (Floyd-Warshall, Bellman-Ford)
6. Subset Sum Problem
7. Edit Distance Problem

Conclusion: (Based on the observations): From this experiment, I learned how to implement the Longest Common Subsequence using Dynamic Programming. I understood how dynamic programming optimizes problems by storing intermediate results and avoiding redundant calculations. Additionally, I analyzed its time complexity and observed how it efficiently finds the longest subsequence between two sequences.

Outcome: CO3: Implement Advance Programming algorithms with its application.

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.