Design & Analysis of Algorithm (Lab)

Name: Ananya

SAPID: 590013832

B-33

Submitted to: Mr.Aryan Gupta

https://github.com/ananya438/DAALAB_ANANYA-590013832

Implement Longest Common Subsequence Problem.

```
public class LCSDynamicProgramming {
    public static String findLCS(String text1, String text2) {
        int m = text1.length();
        int n = text2.length();

        int[][] c = new int[m + 1][n + 1];

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (text1.charAt(i - 1) == text2.charAt(j - 1)) {
                    c[i][j] = c[i - 1][j - 1] + 1;
                } else {
                    c[i][j] = Math.max(c[i - 1][j], c[i][j - 1]);  }  }  }
        System.out.println("Length of LCS: " + c[m][n]);
        return reconstructLCS(c, text1, text2, m, n);
    }
    private static String reconstructLCS(int[][] c, String text1, String text2, int i, int j) {
        if (i == 0 || j == 0) {
            return "";
        }
        if (text1.charAt(i - 1) == text2.charAt(j - 1)) {
            return reconstructLCS(c, text1, text2, i - 1, j - 1) + text1.charAt(i - 1);
        } else {
            if (c[i - 1][j] > c[i][j - 1]) {
                return reconstructLCS(c, text1, text2, i - 1, j);
            } else {
                return reconstructLCS(c, text1, text2, i, j - 1);
            }
    }
      }
```

```
    public static void main(String[] args) {
        String S1 = "GTCGTTCGGAATACTGTC";
        String S2 = "GGTCGTCGGAATTGAC";

        String lcsResult = findLCS(S1, S2);
        System.out.println("LCS: " + lcsResult);
    }
}
```

Complexity Analysis for LCS (Dynamic Programming)

The complexity is mainly determined by the size of the 2D table created for the dynamic programming solution.

Time Complexity

The time complexity is O(m · n).

Dominant Factor: The main part of the algorithm is filling the 2D DP table.

Process: This task requires two nested loops: one loop for the m length of the first string and one for the n length of the second string.

Work per Step: Inside the loops, only simple, constant-time operations (like comparison and addition) are performed.

The final step to build the actual sequence takes only O(m+n) time, which is much faster (dominated by) the O(m · n) table-filling time.

Space Complexity

The space complexity is O(m · n).

Primary Factor: The algorithm requires an auxiliary 2D array (the DP table, c).

Size: This table is of size (m+1)×(n+1) to store the lengths of the LCS for all possible prefixes.

Necessity: This space is essential because the algorithm must store the results of all smaller subproblems to build the final solution.

O/P: