Day 20

Assignment 2: Longest Common Subsequence

Implement int LCS(string text1, string text2) to find the length of the longest common subsequence between two strings.

A)

Java implementation of the function LCS that computes the length of the longest common subsequence (LCS) between two strings using dynamic programming.

Java:

```
public class LongestCommonSubsequence {

    public static int LCS(String text1, String text2) {
        int m = text1.length();
        int k = text2.length();
        int[][] dp = new int[m + 1][k + 1];

        // Build dp[][] in bottom-up manner
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= k; j++) {
                if (text1.charAt(i - 1) == text2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1] + 1;
                } else {
                    dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }

        return dp[m][k];
    }

    public static void main(String[] args) {
        String text1 = "abcde"; // Example string 1
        String text2 = "ace";  // Example string 2

        System.out.println("Length of LCS = " + LCS(text1, text2));
    }
}
```

Explanation:

1. Initialization:

- dp[i][j] will hold the length of the LCS of the substrings text1[0..i-1] and text2[0..j-1].

- dp[0][*] = 0 and dp[*][0] = 0 because the LCS with any empty string is 0.

2. Filling the DP Table:

- Iterate over each character of text1 and text2.

- If text1[i-1] == text2[j-1], then the characters match, and the LCS up to i and j is dp[i-1][j-1] + 1.

- If the characters do not match, then the LCS up to i and j is the maximum of the LCS without the current character of text1 (dp[i-1][j]) or without the current character of text2 (dp[i][j-1]).

3. Result:

- The result will be in dp[m][k], which represents the length of the LCS of text1 and text2.

This solution efficiently computes the length of the LCS using dynamic programming with a time complexity of \(O(m \times k)\) and a space complexity of \(O(m \times k)\), where \(m\) and \(k\) are the lengths of text1 and text2, respectively.