Lab Report-11

(Longest_Common_Subsequence)

CSE-2212 (Design and Analysis of Algorithms Lab)

Submitted By:

Name: Eyasir Ahamed
Exam Roll: 413
Class Roll: 15
Registration No: 202004017

Submitted To:

Sharad Hasan
Ex. Lecturer
Dept. of CSE
Sheikh Hasina University, Netrokona

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SHEIKH HASINA UNIVERSITY
NETROKONA, BANGLADESH

#11_Longest Common Subsequence

Problem Definition: Given two strings, the problem is to find the length of the Longest Common Subsequence (LCS) between them.

Formal Statement of Algorithm (Longest Common Subsequence using Dynamic Programming):

- Define a function lcs that takes two strings s1 and s2 as input parameters.
- Initialize two variables n and m to store the lengths of s1 and s2 respectively.
- Create a 2D DP table dp of dimensions (n + 1) x (m + 1) and initialize all cells to -1.
- Initialize the base cases:
    - Set dp[i][0] = 0 for all i from 0 to n.
    - Set dp[0][i] = 0 for all i from 0 to m.
- Iterate over the characters of s1 and s2 using two nested loops:
    - If the characters at indices ind1 and ind2 match (s1[ind1 - 1] == s2[ind2 - 1]), increment the length of LCS by 1 (dp[ind1][ind2] = 1 + dp[ind1 - 1][ind2 - 1]).
    - Otherwise, set the length of LCS at the current indices to the maximum of the LCS lengths obtained by excluding each

character (dp[ind1][ind2] = max(dp[ind1 - 1][ind2], dp[ind1][ind2 - 1])).

- The final result is stored in dp[n][m], which represents the length of the Longest Common Subsequence.

Complexity Analysis:

- Time Complexity: The algorithm fills in a 2D DP table of size (n + 1) x (m + 1), so the time complexity is O(nm), where n and m are the lengths of the input strings s1 and s2 respectively.
- Space Complexity: The space complexity is also O(nm) due to the DP table.

Actual Code and Output

```cpp
#include <bits/stdc++.h>
using namespace std;

int lcs(string s1, string s2) {
    int n = s1.size();
    int m = s2.size();

    vector<vector<int>> dp(n + 1, vector<int>(m + 1, -1));

    for (int i = 0; i <= n; i++) {
        dp[i][0] = 0;
    }
    for (int i = 0; i <= m; i++) {
        dp[0][i] = 0;
    }

    for (int ind1 = 1; ind1 <= n; ind1++) {
        for (int ind2 = 1; ind2 <= m; ind2++) {
            if (s1[ind1 - 1] == s2[ind2 - 1])
                dp[ind1][ind2] = 1 + dp[ind1 - 1][ind2 - 1];
            else
                dp[ind1][ind2] = max(dp[ind1 - 1][ind2], dp[ind1][ind2 - 1]);
        }
    }

    return dp[n][m];
}

int main() {
    string s1 = "acd";
    string s2 = "ced";

    cout << "The Length of Longest Common Subsequence is " << lcs(s1, s2) << endl;

    return 0;
}
```

```
The Length of Longest Common Subsequence is 2
[Finished in 1.2s]
```