Longest Common Subsequence (LCS)

Problem

Given two strings S1 and S2. We need to output the length of longest common subsequence.

Example:
S1 = “RISHABH”
S2 = “SHUBHI”

Longest Common Subsequence = “SHBH”
Length of LCS = 4.

Brute force approach

1. Compute all subsequences of any of the string and check whether it is a subsequence of other string also.
2. Pick the one with maximum length.

Time Complexity: $O(2^n)$

Efficient Approach

Let f(i,j) denotes the length of LCS of S1[0, ..., i-1] and S2[0,...., j-1].
Two cases arise:

A. S1[i-1] == S2[j-1]

   f(i,j) = 1 + f(i-1, j-1) {we take element in our LCS}

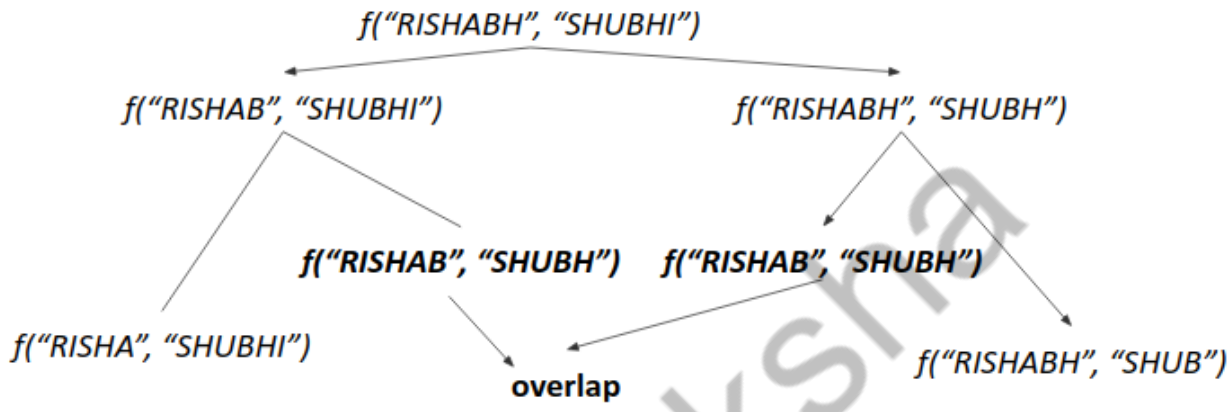
B. S1[i-1] != S2[j-1], further two possibilities
   a. Take S1[i-1] in LCS.
   b. Take S2[j-1] in LCS.

      f(i,j) = max(f(i-1, j), f(i,j-1))

Note: Either if we do not take any of them, that case is also included in the above possibilities.

Since we can write the recurrence relation of problems, hence it follows optimal substructure property.
Checking whether it follows the overlapping subproblem property also?

Making recursion tree



Since f("RISHAB", "SHUBH") repeats
It follows overlapping subproblem property.

Dry Run

	ϕ	R	I	S	H	A	B	H
ϕ	0	0	0	0	0	0	0	0
S	0	0	0	1	1	1	1	1
H	0	0	0	1	2	2	2	2
U	0	0	0	1	2	2	2	2
B	0	0	0	1	2	2	3	3
H	0	0	0	1	2	2	3	4
I	0	0	1	1	2	2	3	**4**

Hence answer = 4.

Approach 1

1. Write the recursive solution.
2. Memoize it.

Approach 2

1. Build from base.
2. For every character of S1, iterate on every character of S2, and apply the recurrence

if(S1[i-1] == S2[j-1])
dp[i][j] = 1 + dp[i-1][j-1]
else
dp[i][j] = max(dp[i-1][j], dp[i][j-1])

3. Output dp[n][m]

Where n - size of S1
m - size of S2

Time Complexity: O(n*m)

Code (Recursive)

```
int dp[N][N];

// LCS

int lcs(string &s1, string &s2, int n, int m)
{
    if(n == 0 || m == 0)
        return 0;

    if(dp[n][m] != -1)
        return dp[n][m];

    if(s1[n-1] == s2[m-1])
        dp[n][m] = 1 + lcs(s1, s2, n-1, m-1);
    else
        dp[n][m] = max(lcs(s1, s2, n-1, m), lcs(s1, s2, n, m-1));

    return dp[n][m];
}
```

```
void solve()
{
    rep(i,0,N)
    {
        rep(j,0,N)
            dp[i][j] = -1;
    }

    string s1, s2;

    cin >> s1 >> s2;

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

    cout << lcs(s1, s2, n, m);
}
```

Code (Iterative)

```
void solve()
{
    string s1,s2;
    cin >> s1 >> s2;
    int n = s1.size(), m = s2.size();

    vvi dp(n+1, vi(m+1,-1));

    rep(i,0,n+1)
    {
        rep(j,0,m+1)
        {
            if(i == 0 || j == 0)
            {
                dp[i][j] = 0;
                continue;
            }

            if(s1[i-1] == s2[j-1])
                dp[i][j] = 1 + dp[i-1][j-1];
            else
                dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
        }
    }

    cout << dp[n][m] << endl;
}
```