jongminl-311-hw5

jongminl Lee

April 2024

1 Question 1

	s	t	x	y	z
0	0	∞	∞	∞	∞
1	0	6	∞	7	∞
2	0	6	4	7	2
3	0	2	4	7	2
4	0	0	4	7	-2

2 Question 2

(a) Recursive Definition:
The recursive definition of finding the longest common subsequence (LCS) between two strings can be defined as follows:
Let $LCS(X[1...m], Y[1...n])$ be the length of the longest common subsequence of strings X and Y, where $X[1...m]$ and $Y[1...n]$ are substrings of X and Y respectively.

$$LCS(X[1...m], Y[1...n]) =$$

$$\begin{cases} 0 & \text{if } m = 0 \text{ or } n = 0 \\ 1 + LCS(X[1...m-1], Y[1...n-1]) & \text{if } X[m] = Y[n] \\ \max(LCS(X[1...m], Y[1...n-1]), LCS(X[1...m-1], Y[1...n])) & \text{otherwise} \end{cases} \tag{1}$$

(b) Pseudocode for Iterative Dynamic Programming:

```
function longestCommonSubsequence(X, Y):
    m = length(X)
    n = length(Y)
    create a table T of size (m+1) x (n+1)

    for i from 0 to m:
        for j from 0 to n:
            if i == 0 or j == 0:
                T[i][j] = 0
            else if X[i-1] == Y[j-1]:
                T[i][j] = 1 + T[i-1][j-1]
            else:
                T[i][j] = max(T[i-1][j], T[i][j-1])

    // Constructing the longest common subsequence
    lcs = []
    i = m, j = n
    while i > 0 and j > 0:
        if X[i-1] == Y[j-1]:
            lcs.push(X[i-1])
            i--, j--
        else if T[i-1][j] > T[i][j-1]:
            i--
        else:
            j--

    return reverse(lcs)
```

(c) Runtime Analysis:

The time complexity of the above algorithm is $O(m*n)$, where m and n are the lengths of the input strings X and Y respectively. This is because we construct a table of size $(m+1)*(n+1)$ and fill it in a nested loop, where each cell computation takes constant time. The space complexity is also $O(m*n)$ for the same reason.