COMP3011 Homework 2

ZHANG Caiqi 18085481d

October 17, 2020

1

See *LCS18085481D.java*.

2

Explanation: I developed a dynamic programming algorithm for this LCS. As this question is about comparing three sequences, we need a 3-dimensional table to store the sub-problem results.
Suppose that the three strings are X, Y, and Z. Also, we define the three directions to be $x - axis$, $y - axis$, and $z - axis$. And 0, 1, 2, 3 are defined as followed.

```
/*
 * 0: go one step in x - axis, y - axis, and z - axis
 * 1: go one step in z - axis
 * 2: go one step in y - axis
 * 3: go one step in x - axis
 */
```

Figure 1: Codes fragment

The recurrence formula is as followed.

$$a[i,j,k] = \begin{cases} 0, & i = 0, j = 0, or\ k = 0, \\ a[i-1,j-1,k-1]+1, & i,j,k > 0, X_i = Y_j = Z_k \\ max\{a[i-1,j,k], a[i,j-1,k], a[i,j,k-1]\}, & else. \end{cases} \quad (1)$$

The time complexity and space complexity of this algorithm are both $O(lmn)$, where l, m, n are the length of X, Y, and Z.

3

The largest n in my computer is around 810 (MacBook Pro 2016 with 16GB memory). It can be find that the main issue here is running out of memory.

4

Here is an example: $X =$ "11111111", $Y =$ "00100011", $Z =$ "11100000".
The LCS of X, Y, and Z is "111". The $W =$ "00000". The LCS of X and W is empty.

5

n	A(n)	A(n)/n
80	56.8	0.7100
160	114.3	0.7144
240	174.8	0.7283
320	234.5	0.7328
400	292.2	0.7305
480	351.4	0.7321
560	411.4	0.7346
640	469.5	0.7336
720	531.2	0.7378
800	589.4	0.7368

Because that the maximum number my personal computer can process is around 800, so I made this table with 10 different n values. Every time I did 10 independent runs.
According to the table, we may guess that $A(n)/n$ will converge to some value between 0.70 to 0.80. However, there is not enough evidence only from this experiment.

6

If n is even and X_n, Y_n are defined as followed,

$$X_n = (a_1, a_2, a_3, a_4, ..., a_{n-1}, a_n) \tag{2}$$

$$Y_n = (a_2, a_1, a_4, a_3, ..., a_n, a_{n-1}) \tag{3}$$

By choosing one term from each pair, there must be at least $(2)^{\frac{n}{2}}$ longest common subsequences. If every time we can get one possible LCS, it takes $\Omega((\sqrt{2})^n)$ time to get all the LCS.

If n is odd, because of the similar reason, X_n, Y_n can be defined as followed,

$$X_n = (a_1, a_2, a_3, a_4, ..., a_{n-1}, a_n) \tag{4}$$

$$Y_n = (a_2, a_1, a_4, a_3, ..., a_{n-1}, a_{n-2}, a_n) \tag{5}$$

Apart from the last one a_n, by choosing one term from each pair, there must be at least $(2)^{\frac{n-1}{2}}$ longest common subsequences. If every time we can get one possible LCS, it also takes $\Omega((\sqrt{2})^{n-1}) = \Omega((\sqrt{2})^n)$ time to get all the LCS, the constant c is greater than 1.

Therefore, every algorithm that outputs all longest common subsequences of two input sequences has a worst-case running time that is exponential.