COMP 3011 Assignment 1

JAHJA Darwin, 16094501d

1.

I have chosen Java 11 as my coding language to implement this algorithm.

To compile the java file, run: `javac LCS16094501D.java`

To execute the program, run: `java LCS16094501D nine singing ninjas`

To further test the program with larger input size (each input strings have a length of 512), simply copy the command from the `Inp16094501D.txt` file and execute in the command line. The result should be:

```
thmocxintgwsfhdygnihzglkenlomccvgqkdceagllngjmusnwsp
saajzyspofocotvxtfwrsaqofgscybbhtwcpdrirpvqlylyzxqi
```

2.

The Java program uses bottom-up dynamic programming to find out the LCS of 3 strings (represented as a string).

Based on the Theorem 15.1 from **Lecture 4 Slide 40**, we can extend this to obtain the following recurrence:

Define $c[i, j, k]$ = the length of an LCS of X_i, Y_j and Z_k, Then:

$$c[i,j,k] = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0 \text{ or } k = 0 \\ c[i-1, j-1, k-1] + 1, & \text{if } i, j, k > 0 \text{ and } x_i = y_j = z_k \\ max\{c[i-1,j,k], c[i,j-1,k], c[i,j,k-1]\}, & \text{otherwise} \end{cases}$$

The algorithm's procedures to obtain the LCS of 3 strings are similar to that to obtain from 2 strings. The difference is that the *count* table (*c-table*) and *trace* table (*b-table*) are defined in 3-Dimensional instead of 2-Dimensional, and their values are filled in based on this new recurrence.

Both time complexity and space complexity for this algorithm are $O(|X| \cdot |Y| \cdot |Z|)$. When running the Java program on my laptop (16GB RAM), the largest input length is around **940** before running out of my memory. In this case, running out of memory is the main issue that terminates the program.

3.

One example: Three binary sequences $X = \langle 1,1,1,1,1,1,1,1 \rangle, Y = \langle 0,0,0,0,0,0,1,1 \rangle, Z = \langle 1,1,0,0,0,0,0,0 \rangle$

In this case, the LCS of X, Y, Z is $\langle 1, 1 \rangle$, length = 2, while the LCS of W is $\langle 0, 0, 0, 0, 0, 0 \rangle$. Thus, the LCS of X and W is empty, length = 0, which is different from the length of LCS of X, Y, Z.

4.

n	$A(n)$	$A(n)/n$
20	12.9	0.645
50	33.8	0.6760
100	71.2	0.712
150	108.2	0.7213
200	145.1	0.7255
400	292.6	0.7315
600	443.8	0.7397
700	514.8	0.7354
800	590.9	0.7386
900	664.1	0.7379

(Note: The values of $A(n)/n$ have been rounded to 4 decimal places)

The value of $A(n)/n$ seems to converge to a constant between 0.71 to 0.74 for $n \geq 100$, according to my experiments.

5.

Assuming that when $c = \sqrt{2}$, there exists 2 sequences of size-n so that there will be $\Omega(c^n)$ combinations of LCS. Then, let $f(n)$ be the combination of the 2 sequences of size-n, and denote X_n and Y_n as those 2 sequences.

Prove by induction: Assume that $f(n) \geq c^n/2 = 2^{n/2-1}$ holds, we define an alphabet set of $s = \{s_1, s_2, ..., s_n\}$

Bases cases:

$n = 1$: Let $X_1 = Y_1 = \{s_1\}$. Thus, $F(n) = 1 = 2^0 > 2^{-0.5} = c^n/2$ holds.

$n = 2$: Let $X_2 = \{s_1, s_2\}, Y_2 = \{s_2, s_1\}$. Thus, $F(n) = 2 > 2^{-0.5} = c^n/2$ holds.

Induction:

For $n > 2$, let $X_n = X_{n-2} + \{s_{n-1}, s_n\}, Y_n = Y_{n-2} + \{s_n, s_{n-1}\}$.

Thus, for each LCS of (X_{n-2}, Y_{n-2}), one can generate 2 distinct LCS of (X_n, Y_n) by adding either a_n or a_{n-1} to the end of that LCS so that $f(n) = f(n-2) \cdot 2 \geq 2^{((n-2)/2)-1} \cdot 2 = 2^{n/2-1} = c^n/2$ holds.

Finally, we can conclude that when $c = \sqrt{2}$, there exists 2 sequences of size-n such that there are $\Omega(c^n)$ combination of LCS for $n \in \mathbb{Z}^+$.