Algorithm – 07
– Longest-Common-Subsequence

A. Problem Description

In the *longest-common-subsequence problem,* we are given two sequences X = {x1, x2, ..., xm} and Y = {y1, y2, ..., yn} and wish to find a maximum- length common subsequence Z = {z1, z2, ..., zk} of X and Y.

B. Description of Algorithm

First, let 'LCS' be short for 'Longest-Common-Sequence'.

1. If x[m] = y[n], then Z[k] = x[m] = y[n] and Zk-1 is the LCS of Xm-1 and Yn-1.
2. If x[m] != y[n] and z[k] != x[m], then Z is the LCS of Xm-1 and Y.
3. If x[m] != y[n] and z[k] != y[n], then z is the LCS of Xm and Yn-1.

Let us define c[i][j] to be the length of an LCS of the sequences Xi and Yj, so the optimal substructure of the LCS problem gives the recursive formula as follows:

$$c[i][j] = \begin{cases} 0 & i = 0, j = 0 \\ c[i-1][j-1] + 1 & i, j > 0;\ x_i = y_j \\ \max\{c[i-1][j], c[i][j-1]\} & i, j > 0;\ x_i \neq y_j \end{cases}$$

```
LCSLength(m, n, arrayX, arrayY, c, b)
        for i = 1 to m + 1
                for j = 1 to n + 1
                        if arrayX[i] == arrayY[j]
                                c[i][j] = c[i - 1][j - 1] + 1
                                b[i][j] = 1
                        else if c[i - 1][j] >= c[i][j - 1]
                                c[i][j] = c[i - 1][j]
                                b[i][j] = 2
                        else
                                c[i][j] = c[i][j - 1]
                                b[i][j] = 3
```

--> T-LCSLength(m, n) = O(mn)

```
LCS(i, j, arrayX, b)
    if i == 0 or j == 0
        return
    if b[i][j] == 1
        LCS(i - 1, j - 1, arrayX, b)
        print arrayX[i]
    else if b[i][j] == 2
        LCS(i - 1, j, arrayX, b)
    else
        LCS(i, j - 1, arrayX, b)
```

--> T-LCS(m, n) = O(m + n)

==>T(m, n) = T-LCSLength(m, n) + T-LCS(m, n) = O(mn)

C. Code.[Python]

```
#!/usr/bin/python
# Filename: LCSLength.py

def LCSLength(m, n, arrayX, arrayY, c, b):
    '''
    for     i in range(0, m + 1):
            c[i][0] = 0
    for     i in range(0, n + 1):
            c[0][i] = 0
    '''
    for i in range(1, m + 1):
            for     j in range(1, n + 1):
                    if      arrayX[i] == arrayY[j]:
                            c[i][j] = c[i - 1][j - 1] + 1
                            b[i][j] = 1
                    elif c[i - 1][j] >= c[i][j - 1]:
                            c[i][j] = c[i - 1][j]
                            b[i][j] = 2
                    else:
                            c[i][j] = c[i][j - 1]
                            b[i][j] = 3
```

```
#!/usr/bin/python
# Filename: LCS.py

def LCS(i, j, arrayX, b):
    if      i == 0 or j == 0:
            return
    if      b[i][j] == 1:
            LCS(i - 1, j - 1, arrayX, b)
```

```
		print arrayX[i]
	elif b[i][j] == 2:
		LCS(i - 1, j, arrayX, b)
	else:
		LCS(i,j - 1, arrayX, b)
```

```
#!/usr/bin/python
# Filename: Longest-Common-Subsequence.py

import random
import LCSLength
import LCS

arrayX = [0, 'a', 'b', 'c', 'b', 'd', 'a', 'b']
arrayY = [0, 'b', 'd', 'c', 'a', 'b', 'a']

m = len(arrayX) - 1
n = len(arrayY) - 1

c = []
b = []

for i in range(0, m + 1):
	c.append([])
	for j in range(0, n + 1):
		c[i].append(0)

for i in range(0, m + 1):
	b.append([])
	for j in range(0, n + 1):
		b[i].append(0)

LCSLength.LCSLength(m, n, arrayX, arrayY, c, b)
LCS.LCS(m, n, arrayX, b)
```