Longest common Subsequence

Longest common Subsequence mean, we need to find common string between 2 strings where ~~ordo~~ index of the string doesn't matter, but order of the string does matter.

For example,

LCS for 'abcde' & 'ace' ⇒ 3 [ace]

LCS for 'ABCDGH' & 'AEDFHR' ⇒ 3 [ADH]

To Find the LCS,
we use following algorithm,
we use "DPapproch"

* Declare an arrayof length [len(text1)+1, len(text2)+1]
* we iterate the DP array from (len(text1)+1, len(text2)+1) to (0,0)
* The LCS will be accumulated finally at (0,0)
* In b/w we accumulate DParray as follows

```
if(text1[i] == text2[j]):
    dp[i][j] = 1+dp[i+1][j+1]
else:
    dp[i][j] = max(dp[i+1][j], dp[i][j+1])
```

After iterating the whole DParray we return dp[0][0]

(0,0) ⇒ a=a

⇓

dp[0][0] = 1+dp[1][1]

dp[0][0] = 1+2 = 3

$\boxed{dp[0][0] = 3}$

↳ LCS

text1 = 'abcde'

text2 = 'ace'

j

i	a	c	e	
a	3	2	1	0
b	2	2	1	0
c	2	2	1	0
d	1	1	1	0
e	1	1	1	0
	0	0	0	0

Initialize DP arraywith 0

(2,4) ⇒ e=e
j i

⇒ 1+dp[i+1][j+1]

dp[2][4] ⇒ 1+dp[3][4] = 1+0 = 1

$\boxed{dp[4][2] = 1}$

(4,1)(4,0)(3,2)(3,1)(3,0)
(2,2)

⇓

max(dp[i+1][j], dp[i][j+1])

(2,1) ⇒ c=c

dp[2][1] ⇒ 1+dp[i+1][j+1] ⇒ 1+1 = 2

$\boxed{dp[2][1] = 2}$

(2,0)(1,2)(1,1)(1,0)(0,2)(0,1)

⇓

max(dp[i+1][j], dp[i][j+1])