DATA ANALYSIS AND ALGORITHM

PRACTICAL NO 5

027_Ojha Abhishek

Experiment No – 5 **Date of Experiment : 24th September 2021**

Program : - Write a Program to implement Longest Common Subsequence.

Input :-

S1 : ACADB

S2 : CBDA

LCS: CB

Algorithm :-

```
X and Y be two given sequences
Initialize a table LCS of dimension X.length * Y.length
X.label = X
Y.label = Y
LCS[0][] = 0
LCS[][0] = 0
Start from LCS[1][1]
Compare X[i] and Y[j]
If X[i] = Y[j]
LCS[i][j] = 1 + LCS[i-1, j-1]
Point an arrow to LCS[i][j]
Else
LCS[i][j] = max(LCS[i-1][j], LCS[i][j-1])
Point an arrow to max(LCS[i-1][j], LCS[i][j1])
```

Fig :-

The First Sequence

X	A	C	A	D	B

The Second Sequence

Y	C	B	D	A

The following steps are followed for finding the longest common subsequence.

Create a table of dimension n+1*m+1 where n and m are the lengths of X and Y respectively.

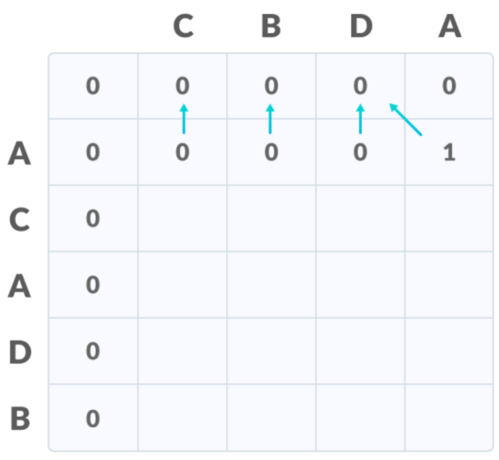
The first row and the first column are filled with zeros.

		C	B	D	A
	0	0	0	0	0
A	0				
C	0				
A	0				
D	0				
B	0				

Fill each cell of the table using the following logic.

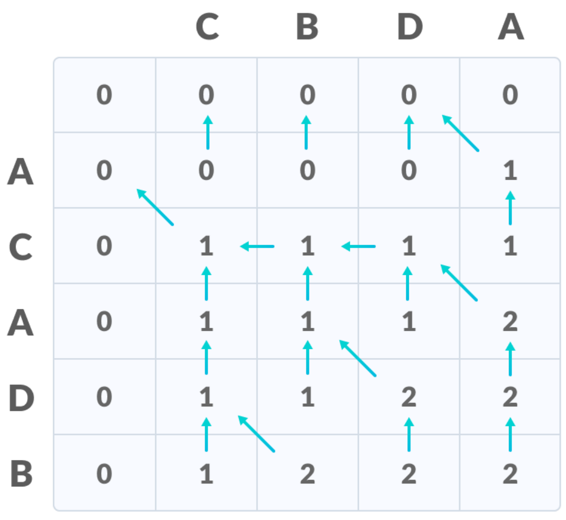
If the character correspoding to the current row and current column are matching, then fill the current cell by adding one to the diagonal element. Point an arrow to the diagonal cell.

Else take the maximum value from the previous column and previous row element for filling the current cell. Point an arrow to the cell with maximum value. If they are equal, point to any of them.



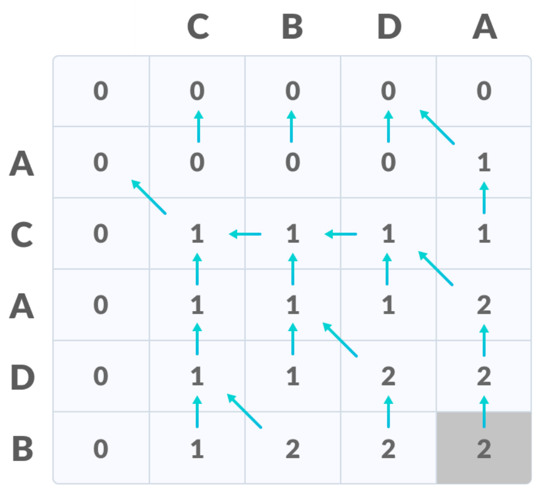
		C	B	D	A
	0	0	0	0	0
A	0	0	0	0	1
C	0				
A	0				
D	0				
B	0				

Step 2 **is repeated until the table is filled**.

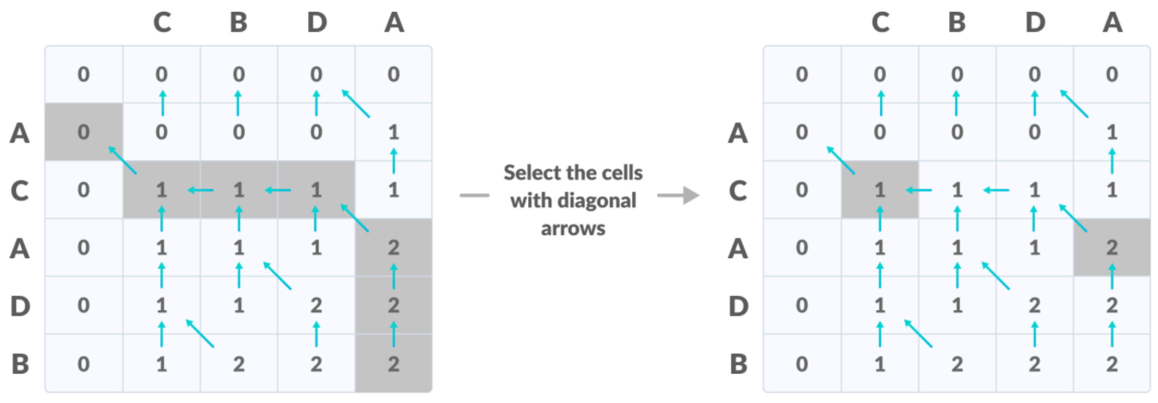


		C	B	D	A
	0	0	0	0	0
A	0	0	0	0	1
C	0	1	1	1	1
A	0	1	1	1	2
D	0	1	1	2	2
B	0	1	2	2	2

The value in the last row and the last column is the length of the longest common
subsequence.



In order to find the longest common subsequence, start from the last element and follow the direction of the arrow. The elements corresponding to () symbol form the longest common subsequence.

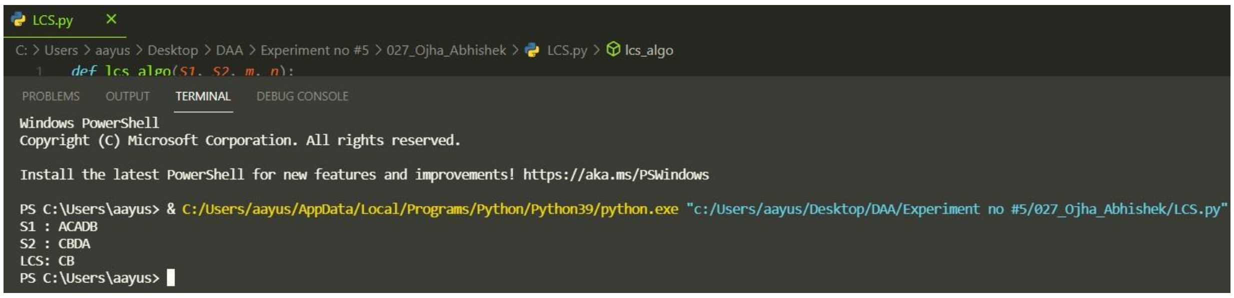


Thus, the longest common subsequence is CA .

Practical Implementation of Longest Common Subsequence :-

```
def lcs_algo(S1, S2, m, n):
   L = [[0 for x in range(n+1)] for x in range(m+1)]
    for i in range(m+1):         for j in
range(n+1):             if i == 0 or j == 0:
L[i][j] = 0            elif S1[i-1] == S2[j-1]:
L[i][j] = L[i-1][j-1] + 1             else:
            L[i][j] = max(L[i-1][j], L[i][j-1])     index =
L[m][n]     lcs_algo = [""] * (index+1)
lcs_algo[index] = ""     i = m     j = n     while i > 0 and
j > 0:
      if S1[i-1] == S2[j-1]:
         lcs_algo[index-1] = S1[i-1]
i -= 1            j -= 1            index -= 1
elif L[i-1][j] > L[i][j-1]:
i       -= 1        else:
j       -= 1
          print("S1 : " + S1 + "\nS2 : " +
S2)     print("LCS: " + "".join(lcs_algo))
S1 = "ACADB" S2 =
"CBDA" m = len(S1) n =
len(S2) lcs_algo(S1,
S2, m, n)
```

Output:



LCS.py ×

C: > Users > aayus > Desktop > DAA > Experiment no #5 > 027_Ojha_Abhishek > LCS.py > lcs_algo

```
def lcs_algo(S1, S2, m, n):
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\aayus> & C:/Users/aayus/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/aayus/Desktop/DAA/Experiment no #5/027_Ojha_Abhishek/LCS.py"
S1 : ACADB
S2 : CBDA
LCS: CB
PS C:\Users\aayus> 
```

Time complexity of the above naive recursive approach is O(2^n) in worst case and worst case happens when all characters of X and Y mismatch i.e., length of LCS is 0.

Conclusion:

Time Complexity of the above implementation is O(mn) which is much better than the worst-case time complexity.