

CTA

PRACTICAL 9

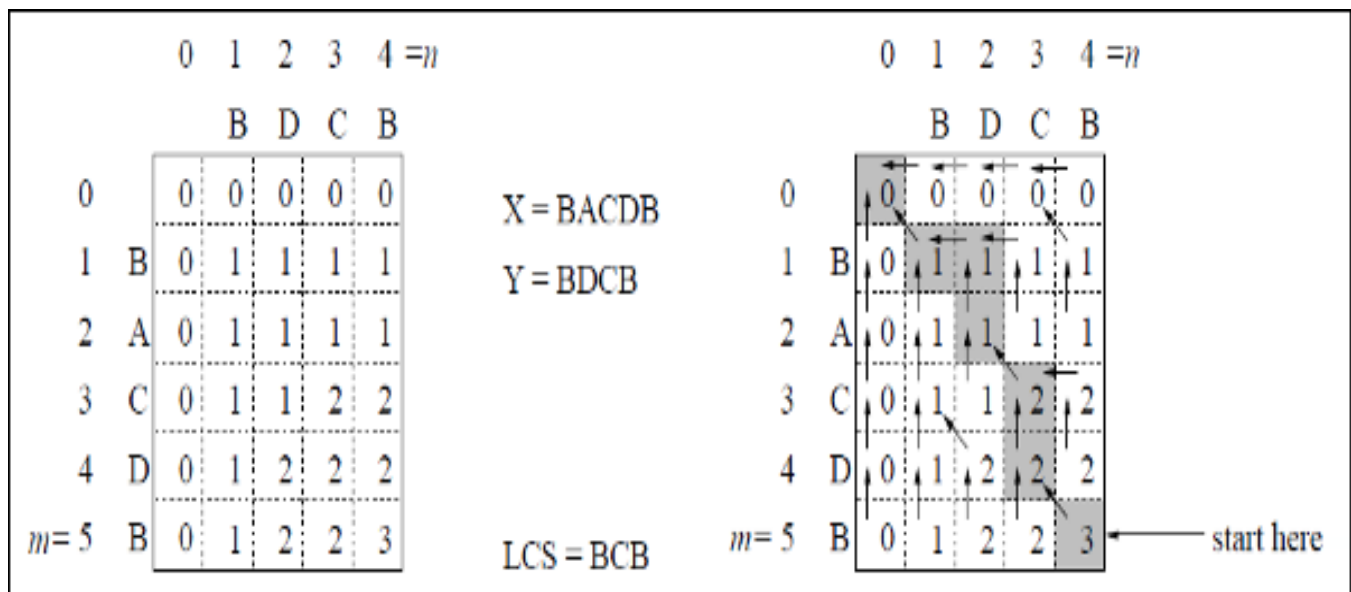
20MCED08

## Practical 9

Given two sequences X and Y, find the longest common sub-sequence (LCS) of X and Y using dynamic programming.

### Introduction

The longest common subsequence problem is finding the longest sequence which exists in both the given strings.



In this example, we have two strings  $X = \mathbf{BACDB}$  and  $Y = \mathbf{BDCB}$  to find the longest common subsequence.

Following the algorithm LCS-Length-Table-Formulation (as stated above), we have calculated table C (shown on the left hand side) and table B (shown on the right hand side).

In table B, instead of 'D', 'L' and 'U', we are using the diagonal arrow, left arrow and up arrow, respectively. After generating table B, the LCS is determined by function LCS-Print. The result is BDB.

# Printing Longest Common Subsequence

```
# Dynamic programming implementation of LCS problem

def lcs(X, Y, 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 X[i-1] == Y[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]
    print("Length of LCS =", index)

    lcs = [""] * (index+1)
    lcs[index] = ""
    # print(lcs)

    i = m
    j = n
    while i > 0 and j > 0:
        if X[i-1] == Y[j-1]:
            lcs[index-1] = X[i-1]
            # print(lcs)
            i-=1
            j-=1
            index-=1
        elif L[i-1][j] > L[i][j-1]:
            i-=1
        else:
            j-=1

    print ("LCS of " + X + " and " + Y + " is " +"".join(lcs))

X = "BACDB"
Y = "BDCB"
m = len(X)
n = len(Y)
lcs(X, Y, m, n)
```

## Output :

Length of LCS = **3**

LCS of BACDB and BDCB is **BDB**

## Observation:

To fill the table, the outer **for** loop iterates ***m*** times and the inner **for** loop iterates ***n*** times. Hence, the complexity of the algorithm is  $O(m, n)$ , where ***m*** and ***n*** are the length of two strings.