**Longest common Subsequence Problem**

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

**Subsequence**

Let us consider a sequence S = <s1, s2, s3, s4, …,sn>.

A sequence Z = <z1, z2, z3, z4, …,zm> over S is called a subsequence of S, if and only if it can be derived from S deletion of some elements.

**Common Subsequence**

Suppose, ***X*** and ***Y*** are two sequences over a finite set of elements. We can say that ***Z*** is a common subsequence of ***X*** and ***Y***, if ***Z*** is a subsequence of both ***X*** and ***Y***.

**Longest Common Subsequence**

If a set of sequences are given, the longest common subsequence problem is to find a common subsequence of all the sequences that is of maximal length.

The longest common subsequence problem is a classic computer science problem, the basis of data comparison programs such as the diff-utility, and has applications in bioinformatics. It is also widely used by revision control systems, such as SVN and Git, for reconciling multiple changes made to a revision-controlled collection of files.

**Naïve Method**

Let ***X*** be a sequence of length ***m*** and ***Y*** a sequence of length ***n***. Check for every subsequence of ***X*** whether it is a subsequence of ***Y***, and return the longest common subsequence found.

There are ***2m*** subsequences of ***X***. Testing sequences whether or not it is a subsequence of ***Y*** takes ***O(n)*** time. Thus, the naïve algorithm would take ***O(n2m)*** time.

**Dynamic Programming**

Let *X = < x1, x2, x3,…, xm >* and *Y = < y1, y2, y3,…, yn >* be the sequences. To compute the length of an element the following algorithm is used.

In this procedure, table ***C[m, n]*** is computed in row major order and another table ***B[m,n]*** is computed to construct optimal solution.

# **Algorithm of Longest Common Sequence**

**LCS-LENGTH (X, Y)**

1. m ← length [X]

2. n ← length [Y]

3. for i ← 1 to m

4. do c [i,0] ← 0

5. for j ← 0 to m

6. do c [0,j] ← 0

7. for i ← 1 to m

8. do for j ← 1 to n

9. do if xi= yj

10. then c [i,j] ← c [i-1,j-1] + 1

11. b [i,j] ← "↖"

12. else if c[i-1,j] ≥ c[i,j-1]

13. then c [i,j] ← c [i-1,j]

14. b [i,j] ← "↑"

15. else c [i,j] ← c [i,j-1]

16. b [i,j] ← "← "

17. return c and b.

## Example of Longest Common Sequence

**Example 1:**

Given two sequences X [1...m] and Y [1.....n]. Find the longest common subsequences to both.

Example of Longest Common Sequence

here X = (A,B,C,B,D,A,B) and Y = (B,D,C,A,B,A)

m = length [X] and n = length [Y]

m = 7 and n = 6

Here x1= x [1] = A y1= y [1] = B

x2= B y2= D

x3= C y3= C

x4= B y4= A

x5= D y5= B

x6= A y6= A

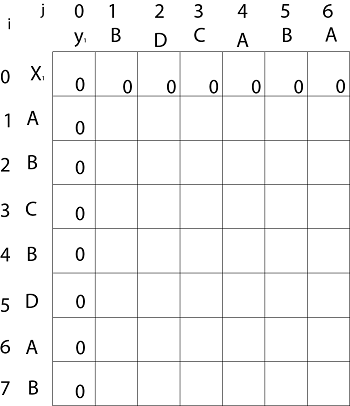
x7= B

Now fill the values of c [i, j] in m x n table

Initially, for i=1 to 7 c [i, 0] = 0

For j = 0 to 6 c [0, j] = 0

That is:



**Now for i=1 and j = 1**

x1 and y1 we get x1 ≠ y1 i.e. A ≠ B

And c [i-1,j] = c [0, 1] = 0

c [i, j-1] = c [1,0 ] = 0

That is, c [i-1,j]= c [i, j-1] so c [1, 1] = 0 and b [1, 1] = ' ↑ '

**Now for i=1 and j = 2**

x1 and y2 we get x1 ≠ y2 i.e. A ≠ D

c [i-1,j] = c [0, 2] = 0

c [i, j-1] = c [1,1 ] = 0

That is, c [i-1,j]= c [i, j-1] and c [1, 2] = 0 b [1, 2] = ' ↑ '

**Now for i=1 and j = 3**

x1 and y3 we get x1 ≠ y3 i.e. A ≠ C

c [i-1,j] = c [0, 3] = 0

c [i, j-1] = c [1,2 ] = 0

so c [1,3] = 0 b [1,3] = ' ↑ '

**Now for i=1 and j = 4**

x1 and y4 we get. x1=y4 i.e A = A

c [1,4] = c [1-1,4-1] + 1

= c [0, 3] + 1

= 0 + 1 = 1

c [1,4] = 1

b [1,4] = ' ↖ '

**Now for i=1 and j = 5**

x1 and y5 we get x1 ≠ y5

c [i-1,j] = c [0, 5] = 0

c [i, j-1] = c [1,4 ] = 1

Thus c [i, j-1] > c [i-1,j] i.e. c [1, 5] = c [i, j-1] = 1. So b [1, 5] = '←'

**Now for i=1 and j = 6**

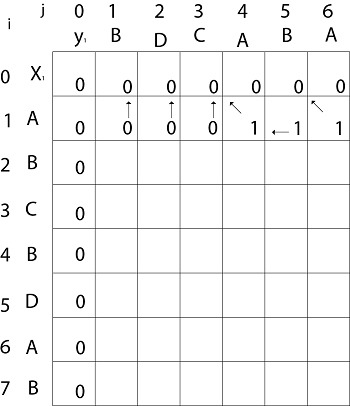
x1 and y6 we get x1=y6

c [1, 6] = c [1-1,6-1] + 1

= c [0, 5] + 1 = 0 + 1 = 1

c [1,6] = 1

b [1,6] = ' ↖ '



**Now for i=2 and j = 1**

We get x2 and y1 B = B i.e. x2= y1

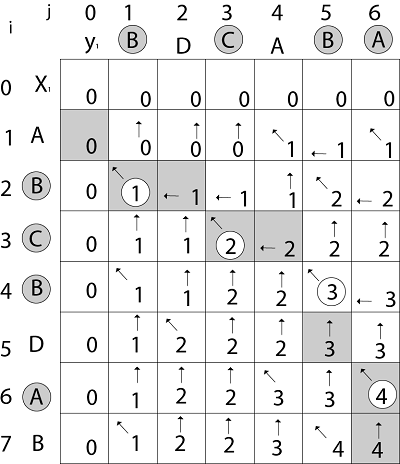
c [2,1] = c [2-1,1-1] + 1

= c [1, 0] + 1

= 0 + 1 = 1

c [2, 1] = 1 and b [2, 1] = ' ↖ '

Similarly, we fill the all values of c [i, j] and we get



**Step 4: Constructing an LCS:** The initial call is PRINT-LCS (b, X, X.length, Y.length)

**PRINT-LCS (b, x, i, j)**

1. if i=0 or j=0

2. then return

3. if b [i,j] = ' ↖ '

4. then PRINT-LCS (b,x,i-1,j-1)

5. print x\_i

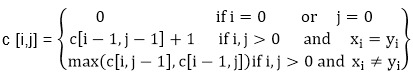
6. else if b [i,j] = ' ↑ '

7. then PRINT-LCS (b,X,i-1,j)

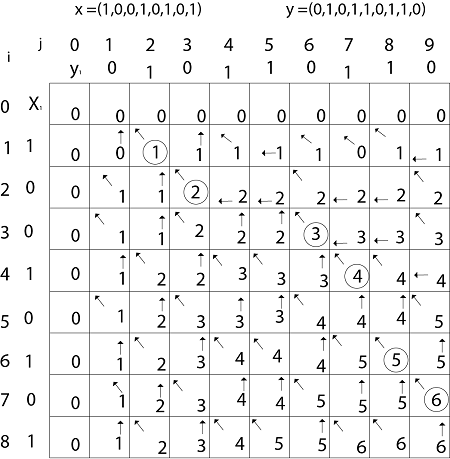
8. else PRINT-LCS (b,X,i,j-1)

**Example:** Determine the LCS of (1,0,0,1,0,1,0,1) and (0,1,0,1,1,0,1,1,0).

**Solution:** let X = (1,0,0,1,0,1,0,1) and Y = (0,1,0,1,1,0,1,1,0).



We are looking for c [8, 9]. The following table is built.



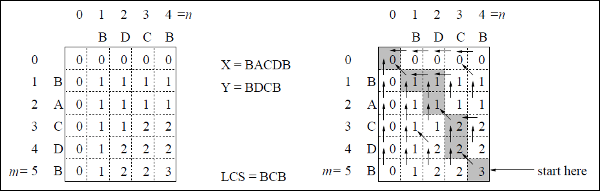
From the table we can deduct that LCS = 6. There are several such sequences, for instance (1,0,0,1,1,0) (0,1,0,1,0,1) and (0,0,1,1,0,1)

**Example 2:**

In this example, we have two strings ***X = BACDB*** and ***Y = 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 BCB.



**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.javatpoint.com/longest-common-sequence-algorithm>
2. <https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_longest_common_subsequence.htm>

**Lecture Video:**

1. <https://youtu.be/HgUOWB0StNE>

**Online Notes:**

1. <http://vssut.ac.in/lecture_notes/lecture1428551222.pdf>

**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**