**How to check if question can be solved by LCS ?**

Two String/array will be given and in question it will be asked to find out longest/shortest array or String based on some condition.

A **subsequence** is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements.

1. **Longest common sub-sequence/string (Recursive approach)**

String x = "abcdgh";

String y = "abedfhr";

Here common character in string is a, b, d, h so longest common subsequence will be "abdh"

Common substring is "ab" because substring should be common and in sequence.

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/lcs/LCSRecursive.java>

1. **Longest common sub-sequence( Memoized approach)**

String x = "abcdgh";

String y = "abedfhr";

In recursive call same method call happens for many times. To escape that method call which is already had been called we use memoization.

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/lcs/LCSRecursiveMemoized.java>

1. **Longest common sub-sequence(Tabular approach)**

In Tabular approach we take matrix of size m+1, n+1 where m is size of x string and n is the size of y string.

In initialization phase we initialize the 1st row and 1st column with 0

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/lcs/LCSTabulation.java>

1. **Longest common substring**

String x = "abcdgh";

String y = "abedfhr";

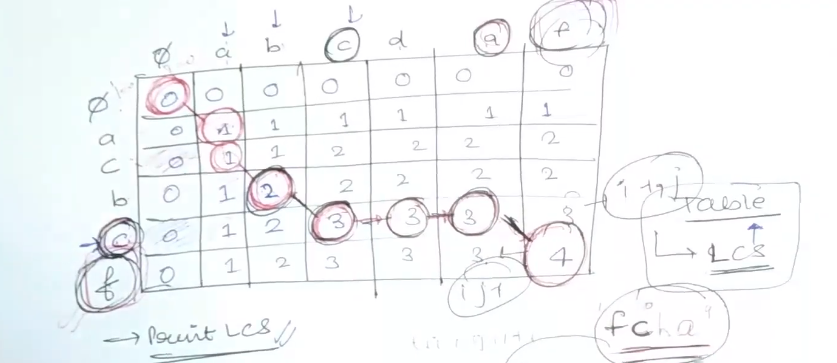
Here we have to find out Longest common substring.

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/lcs/LCSubstring.java>

1. **LCS print**

String x = "abcdgh";

String y = "abedfhr";



Here if x.charAt(i - 1) == y.charAt(j - 1) then we add the char into string. So after adding the string it will be fcba. We need to reverse it to get the answer.

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/lcs/LCSPrint.java>

1. **Shortest common super sequence String count**

String x = "aggtab";

String y = "gxtxayb";

Given two strings str1 and str2, return *the shortest string that has both*str1*and*str2*as****subsequences***. If there are multiple valid strings, return **any** of them.

Op: 9

Shortest common super sequence of the above string is aggxtxayb

Append x and y

aggtabgxtxayb

And LCS of the above string is gtab

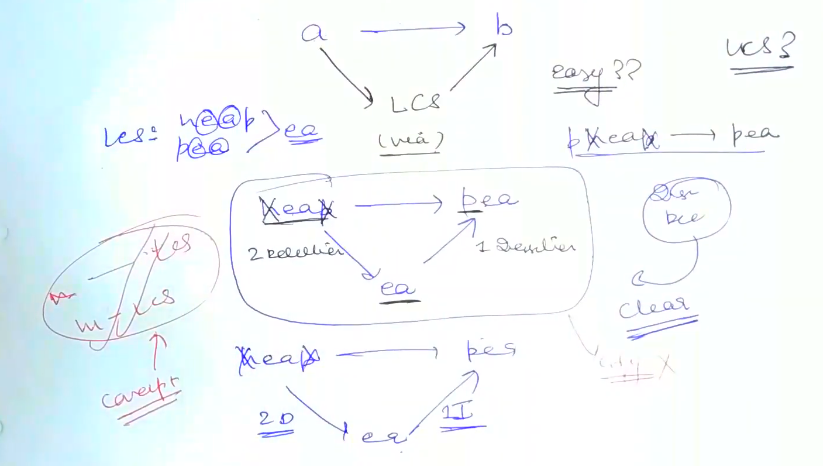
Now remove LCS from appended string

Aggxtxayb…………………………………………. Length of the above string is 9

Algo:

Append both Strings x and y length and then substract lcs from it.

1. **Minimum no of insertion and deletion to convert String A to B**



No of deletion will be X – LCS

No of insertion will be Y – LCS

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/lcs/MinNoOfInsertionDeletionToConvertAtoB.java>

1. **Longest palindromic subsequence**

String x = "aggtab";

Given a string s, find *the longest palindromic****subsequence****'s length in* s.

Steps:

To use LCS we need 2 Strings….but in the input only one String is given

For the 2nd String we can reverse the 1st String and can take as 2nd String.

So String x = “aggtab”;

String y = “batgga”;

Here LCS of x and y is agga and its length will be 4

And this will be the longest polyndromic sequence for the String “aggtab”

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/lcs/LPSTabulation.java>

**10) Minimum no of deletion in string to make it polyndrom.**

Length of LPS is inversally proportional to no of deletion.

If we find out x.length() – lcs then it will be min no of deletion.

**11) Print shortest super common sequence**

String x = "aggtab";

String y = "gxtxayb";

Algo:

Start from the end i=m, j=n

While i>0 && j>0

Check if i-1th char and j-1th char is same

If it is same then add into result. And i--; j—

If dp[i-1][j] > dp[j][i-1] then add the str1[i-1]th char to result and i--

Else add str2[j-1]th char to result and j—

If any of the string ends early…append that string to result.

And last reverse the string……this is shortest common super sequence

**12) Longest repeating subsequence**

**13) Sequence Pattern Matching**

String x = "abc";

String y = "ahbgdc";

Given two strings x and y, return true*if*x*is a****subsequence****of*y*, or*false*otherwise*.

14) Minimum no of insertion to make a String polindrom.