**Problem Statement**[#](https://www.educative.io/courses/grokking-dynamic-programming-patterns-for-coding-interviews/RMkk7NwE44R#problem-statement)

Given two strings ‘s1’ and ‘s2’, find the length of the longest substring which is common in both the strings.

**Example 1:**

Input: s1 = "abdca"  
       s2 = "cbda"  
Output: 2  
Explanation: The longest common substring is "bd".

### Basic Solution [#](https://www.educative.io/courses/grokking-dynamic-programming-patterns-for-coding-interviews/RMkk7NwE44R#basic-solution)

A basic brute-force solution could be to try all substrings of ‘s1’ and ‘s2’ to find the longest common one. We can start matching both the strings one character at a time, so we have two options at any step:

1. If the strings have a matching character, we can recursively match for the remaining lengths and keep a track of the current matching length.
2. If the strings don’t match, we start two new recursive calls by skipping one character separately from each string and reset the matching length.
3. class LCS {
4. public int findLCSLength(String s1, String s2) {
5. return findLCSLengthRecursive(s1, s2, 0, 0, 0);
6. }
7. private int findLCSLengthRecursive(String s1, String s2, int i1, int i2, int count) {
8. if(i1 == s1.length() || i2 == s2.length())
9. return count;
10. if(s1.charAt(i1) == s2.charAt(i2))
11. count = findLCSLengthRecursive(s1, s2, i1+1, i2+1, count+1);
12. int c1 = findLCSLengthRecursive(s1, s2, i1, i2+1, 0);
13. int c2 = findLCSLengthRecursive(s1, s2, i1+1, i2, 0);
14. return Math.max(count, Math.max(c1, c2));
15. }
16. public static void main(String[] args) {
17. LCS lcs = new LCS();
18. System.out.println(lcs.findLCSLength("abdca", "cbda"));
19. System.out.println(lcs.findLCSLength("passport", "ppsspt"));
20. }
21. }

Because of the three recursive calls, the time complexity of the recursive algorithm is exponential O(3^{m+n})*O*(3​*m*+*n*​​), where ‘m’ and ‘n’ are the lengths of the two input strings. The space complexity is O(m+n)*O*(*m*+*n*), this space will be used to store the recursion stack.

Top Down ,Bottom up 2D and 1D in Java file..