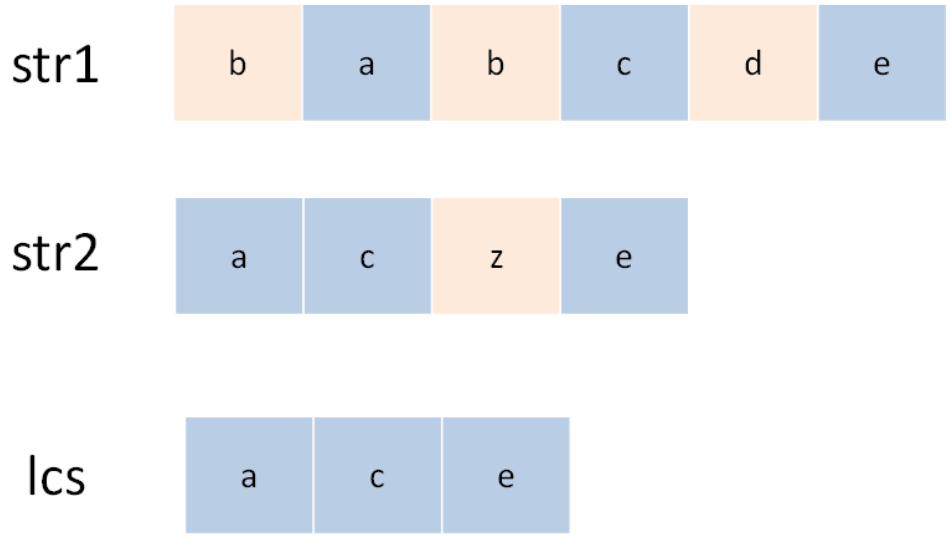
1143. Longest Common Subsequence

Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0. A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.
For example, "ace" is a subsequence of "abcde".
A common subsequence of two strings is a subsequence that is common to both strings.
Example 1:
Input: text1 = "abcde", text2 = "ace" Output: 3
Explanation: The longest common subsequence is "ace" and its length is 3.

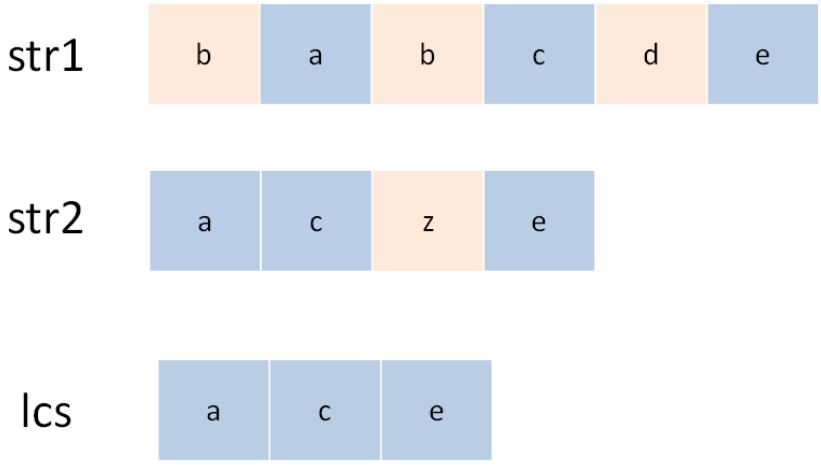
基本思路

PS:这道题在《算法小抄》的第 117 页。
和 编辑距离 同为经典的双字符串动态规划问题。用两个指针 i, j 在两个字符串上游走，这就是「状态」，字符串中的每个字符都有两种「选择」，要么在 lcs 中，要么不在。



dp[i][j] 的含义是：对于 **s1[1..i]** 和 **s2[1..j]**，它们的 **LCS** 长度是 **dp[i][j]**。

详细题解 :详解最长公共子序列问题，秒杀三道动态规划题目
标签：子序列 动态规划，二维动态规划




```java
class Solution {
    public int longestCommonSubsequence(String s1, String s2) {
        int m = s1.length(), n = s2.length();
        // 定义：s1[0..i-1] 和 s2[0..j-1] 的 lcs 长度为 dp[i][j]
        int[][] dp = new int[m + 1][n + 1];
        // 目标：s1[0..m-1] 和 s2[0..n-1] 的 lcs 长度，即 dp[m][n]
        // base case: dp[0][..] = dp[..][0] = 0

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                // 现在 i 和 j 从 1 开始，所以要减一
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    // s1[i-1] 和 s2[j-1] 必然在 lcs 中
                    dp[i][j] = 1 + dp[i - 1][j - 1];
                } else {
                    // s1[i-1] 和 s2[j-1] 至少有一个不在 lcs 中
                    dp[i][j] = Math.max(dp[i][j - 1], dp[i - 1][j]);
                }
            }
        }

        return dp[m][n];
    }
}
```

类似题目：

- 583. 两个字符串的删除操作（中等）
- 712. 两个字符串的最小 ASCII 删除和（中等）