题目地址（1143.最长公共子序列）

https://leetcode-cn.com/problems/longest-common-subsequence

入选理由

1. DP 另一个重要内容： LCS（最长公众子序列）

题目描述

给定两个字符串 text1 和 text2，返回这两个字符串的最长公共子序列的长度。

一个字符串的 子序列 是指这样一个新的字符串：它是由原字符串在不改变字符的相对顺序的情况下删除某些字符（也可以不删除任何字符）后组成的新字符串。 例如，"ace" 是 "abcde" 的子序列，但 "aec" 不是 "abcde" 的子序列。两个字符串的「公共子序列」是这两个字符串所共同拥有的子序列。

若这两个字符串没有公共子序列，则返回 0。

示例 1:

输入：text1 = "abcde", text2 = "ace" 输出：3
解释：最长公共子序列是 "ace"，它的长度为 3。 示例 2:

输入：text1 = "abc", text2 = "abc" 输出：3 解释：最长公共子序列是 "abc"，它的长度为 3。 示例 3:

输入：text1 = "abc", text2 = "def" 输出：0 解释：两个字符串没有公共子序列，返回 0。

提示:

1 <= text1.length <= 1000 1 <= text2.length <= 1000 输入的字符串只含有小写英文字符。

前置知识

- 数组
- 动态规划

标签

- 动态规划

难度

- 中等

思路

LCS（最长公共子序列） 系列我写过专题，非常详细了，大家可以参考一下。地址：https://lucifer.ren/blog/2020/07/01/LCS/

当然最长上升子序列问题我也讲过了，大家一起复习。地址：https://lucifer.ren/blog/2020/06/20/LIS/

关键点解析

- dp 建模套路

代码

代码支持：Python，CPP

Python Code:

```python
class Solution:
    def longestCommonSubsequence(self, A: str, B: str) -> int:
        m, n = len(A), len(B)
        ans = 0
        dp = [[0 for _ in range(n + 1)] for _ in range(m + 1)]
        for i in range(1, m + 1):
            for j in range(1, n + 1):
                if A[i - 1] == B[j - 1]:
                    dp[i][j] = dp[i - 1][j - 1] + 1
                    ans = max(ans, dp[i][j])
                else:
                    dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])
        return ans
```

CPP Code：

```cpp
class Solution {
public:
    int longestCommonSubsequence(string text1, string text2) {
        vector<vector<int>> dp(text1.size() + 1, vector<int>(text2.size() + 1, 0));
        for (int i = 1; i <= text1.size(); i++) {
            for (int j = 1; j <= text2.size(); j++) {
                if (text1[i - 1] == text2[j - 1]) {
```

```
                    dp[i][j] = dp[i - 1][j - 1] + 1;
                } else {
                    dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
                }
            }
        }
        return dp[text1.size()][text2.size()];
    }
};
```

复杂度分析

- 时间复杂度：$O(m * n)$，其中 m 和 n 分别为 A 和 B 的 长度。
- 空间复杂度：$O(m * n)$，其中 m 和 n 分别为 A 和 B 的 长度。

上一页 下一页