583. 两个字符串的删除操作

算法 动态规划

> 题目： Delete Operation for Two Strings
> 英文版链接：https://leetcode.com/problems/delete-operation-for-two-strings/description/
> 中文版链接：https://leetcode-cn.com/problems/delete-operation-for-two-strings/

题目分析

```
示例 1:

输入: "sea", "eat"
输出: 2
解释: 第一步将"sea"变为"ea"，第二步将"eat"变为"ea"
```

这道题可以转换为求两个字符串的最长公共子序列问题，求出了最长公共子序列，我们将其相减，即可得到最终的步骤。关于LCS的理解，可以参考这篇文章：动态规划 最长公共子序列 过程图解

根据这篇文章中的原理，实际上这道题的递推公式出来了，就非常简单了。
将递推公式搬移过来，方便写代码的时候对照着一起看：

$$C[i,j]=\begin{cases}0 & 若\ i=0或j=0\\ C[i-1,j-1]+1 & 若\ i,j>0,x_i=y_j\\ \max\{C[i,j-1],C[i-1,j]\} & 若\ i,j>0,x_i\neq y_j\end{cases}$$

答案

```python
class Solution:
    def minDistance(self, word1: str, word2: str) -> int:
        m = len(word1)
        n = len(word2)
        dp = [[0] * (n+1) for _ in range(m+1)]
        for i in range(1, m + 1):
            for j in range(1, n + 1):
                if word1[i - 1] == word2[j - 1]:
```

```
                    dp[i][j] = dp[i - 1][j - 1] + 1
                else:
                    dp[i][j] = max(dp[i][j - 1], dp[i - 1][j])

        return m + n - 2 * dp[m][n]
```