0072. 编辑距离

ITCharge 大约 3 分钟

- 标签：字符串、动态规划
- 难度：困难

题目链接

- 0072. 编辑距离 - 力扣

题目大意

描述：给定两个单词 $word1$、$word2$。

对一个单词可以进行以下三种操作：

- 插入一个字符
- 删除一个字符
- 替换一个字符

要求：计算出将 $word1$ 转换为 $word2$ 所使用的最少操作数。

说明：

- $0 \leq word1.length, word2.length \leq 500$。
- $word1$ 和 $word2$ 由小写英文字母组成。

示例：

- 示例 1：

```py
输入：word1 = "horse", word2 = "ros"
输出：3
解释：
horse -> rorse (将 'h' 替换为 'r')
rorse -> rose (删除 'r')
rose -> ros (删除 'e')
```

- 示例 2：

```py
输入：word1 = "intention", word2 = "execution"
输出：5
解释：
intention -> inention (删除 't')
inention -> enention (将 'i' 替换为 'e')
enention -> exention (将 'n' 替换为 'x')
exention -> exection (将 'n' 替换为 'c')
exection -> execution (插入 'u')
```

解题思路

思路 1：动态规划

1. 划分阶段

按照两个字符串的结尾位置进行阶段划分。

2. 定义状态

定义状态 $dp[i][j]$ 表示为：「以 $word1$ 中前 i 个字符组成的子字符串 $str1$」变为「以 $word2$ 中前 j 个字符组成的子字符串 $str2$」，所需要的最少操作次数。

3. 状态转移方程

1. 如果当前字符相同（$word1[i-1] = word2[j-1]$），无需插入、删除、替换。$dp[i][j] = dp[i-1][j-1]$。
2. 如果当前字符不同（$word1[i-1] \neq word2[j-1]$），$dp[i][j]$ 取源于以下三种情况中的最小情况：
    1. 替换（$word1[i-1]$ 替换为 $word2[j-1]$）：最少操作次数依赖于「以 $word1$ 中前 $i-1$ 个字符组成的子字符串 $str1$」变为「以 $word2$ 中前 $j-1$ 个字符组成的子字符串 $str2$」，再加上替换的操作数 1，即：$dp[i][j] = dp[i-1][j-1] + 1$。
    2. 插入（$word1$ 在第 $i-1$ 位置上插入元素）：最少操作次数依赖于「以 $word1$ 中前 $i-1$ 个字符组成的子字符串 $str1$」变为「以 $word2$ 中前 j 个字符组成的子字符串 $str2$」，再加上插入需要的操作数 1，即：$dp[i][j] = dp[i-1][j] + 1$。

3. 删除（$word1$ 删除第 $i-1$ 位置元素）：最少操作次数依赖于「以 $word1$ 中前 i 个字符组成的子字符串 $str1$」变为「以 $word2$ 中前 $j-1$ 个字符组成的子字符串 $str2$」，再加上删除需要的操作数 1，即：$dp[i][j] = dp[i][j-1] + 1$。

综合上述情况，状态转移方程为：

$$dp[i][j] = \begin{cases} dp[i-1][j-1] & word1[i-1] = word2[j-1] \\ min(dp[i-1][j-1], dp[i-1][j], dp[i][j-1]) + 1 & word1[i-1] \neq word2[j-1] \end{cases}$$

4. 初始条件

- 当 $i = 0$，「以 $word1$ 中前 i 个字符组成的子字符串 $str1$」为空字符串，「$str1$」变为「以 $word2$ 中前 j 个字符组成的子字符串 $str2$」时，至少需要插入 j 次，即：$dp[0][j] = j$。
- 当 $j = 0$，「以 $word2$ 中前 j 个字符组成的子字符串 $str2$」为空字符串，「以 $word1$ 中前 i 个字符组成的子字符串 $str1$」变为「$str2$」时，至少需要删除 i 次，即：$dp[i][0] = i$。

5. 最终结果

根据状态定义，最后输出 $dp[sise1][s \quad |$（即 $word1$ 变为 $word2$ 所使用的最少操作数）即可。其中 $size1$、$size2$ 分别为 $word1$、$word2$ 的字符串长度。

思路 1：代码

py

```
class Solution:
    def minDistance(self, word1: str, word2: str) -> int:
        size1 = len(word1)
        size2 = len(word2)
        dp = [[0 for _ in range(size2 + 1)] for _ in range(size1 + 1)]

        for i in range(size1 + 1):
            dp[i][0] = i
        for j in range(size2 + 1):
            dp[0][j] = j
        for i in range(1, size1 + 1):
            for j in range(1, size2 + 1):
                if word1[i - 1] == word2[j - 1]:
```

```
                    dp[i][j] = dp[i - 1][j - 1]
                else:
                    dp[i][j] = min(dp[i - 1][j - 1], dp[i - 1][j], dp[i][j - 1])
+ 1
        return dp[size1][size2]
```

思路 1：复杂度分析

- **时间复杂度**：$O(n \times m)$，其中 n、m 分别是字符串 $word1$、$word2$ 的长度。两重循环遍历的时间复杂度是 $O(n \times m)$，所以总的时间复杂度为 $O(n \times m)$。
- **空间复杂度**：$O(n \times m)$。用到了二维数组保存状态，所以总体空间复杂度为 $O(n \times m)$。