72. 编辑距离

难度 困难 1149

给你两个单词 *word1* 和 *word2*，请你计算出将 *word1* 转换成 *word2* 所使用的最少操作数 。

你可以对一个单词进行如下三种操作：

1. 插入一个字符
2. 删除一个字符
3. 替换一个字符

示例 1：

```
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：

```
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：

dp[i][j]表示word1到i位置转换成word2到j位置的需要的最少步数

当 word1[i] = word2[j] 时，dp[i][j] = dp[i-1][j-1]

当word1[i] != word2[j] 时，dp[i][j] = min(dp[i-1][j-1] + dp[i][j-1] + dp[i-1][j]) + 1

op[i][j-1]：插入操作， op[i-1][j]：删除操作，op[i-1][j-1]：替换操作

	‘’	r	o	s
‘’	0	1	2	3
h	1			
o	2			
r	3			
s	4			
e	5			

第一行，是word1为空变成word2的最小步数，就是插入操作

第一列，是word2位空，需要的最少步数，就是删除操作

代码：

自底向上

```python
def minDistance(self, word1: str, word2: str) -> int:
        n1 = len(word1)
        n2 = len(word2)

        op = [ [0]*(n2 + 1) for _ in range(n1 + 1) ]

        for i in range(1, n1 + 1):
            op[i][0] = op[i - 1][0] + 1
```

```
        for j in range(1, n2 + 1):
            op[0][j] = op[0][j - 1] + 1

        for i in range(1, n1 + 1):
            for j in range(1, n2 + 1):
                if word1[i - 1] == word2[j - 1]:
                    op[i][j] = op[i - 1][j - 1]
                else:
                    op[i][j] = min(op[i - 1][j - 1], op[i][j - 1], op[i - 1][j]) + 1

        return op[-1][-1]
```

时间复杂度：O(N1 * N2)
空间复杂度：O(N1 * N2)

自顶向下：

```
def minDistance(self, word1: str, word2: str) -> int:
        import functools
        @functools.lru_cache(None)
        def helper(i, j):
            # 递归终止条件
            if i == len(word1) or j == len(word2):
                return len(word1) - i + len(word2) - j
            # 如果word1[i], word2[j]的值是相同的，那么就不需要考虑i, j
            # 就去考虑 i+1, j+1
            if word1[i] == word2[j]:
                return helper(i+1, j+1)
            else:
            # 如果word1[i], word2[j]的值不同，那就就考虑是通过 插入、删除、还是替换
            # 哪一种方式能得到i+1,  j+1的值所需操作最小。
                inserted = helper(i, j + 1)
                deleted = helper(i + 1, j)
                replaced = helper(i + 1, j+ 1)
                return min(inserted, deleted, replaced) + 1
        return helper(0, 0)
```