编辑距离

题目：

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')
```

思路：

动态规划：

op[i][j] 表示从 word1的i 位置变到word2 的 j位置需要的最小步数。

所以：

word1[i] == word[j] ， 不需要操作,所以，op[i][j] == op[i−1][j−1]

word1[i] ！ = word[j]， op[i][j] = min(op[i][j−1], op[i−1][j], op[i−1][j−1]) + 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			

代码：

```
def minDistance(self, word1: str, word2: str) -> int:
        n1 = len(word1)
        n2 = len(word2)
        op = [[0] *(n2+1) for _ in range(n1+1)]
        %23 第一列：
        for i in range(1, n1 +1):
            op[i][0] = op[i-1][0] + 1

        %23 第一行：
        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][j-1], op[i-1][j], op[i-1][j-1]) + 1

        return op[-1][-1]

```

时间复杂度：O(N1 * N2)

空间复杂度：O(N1 * N2)