（选做）编辑距离（动态规划）

题目描述

编辑距离是衡量两个字符串差异的一种常用方法。其定义为：一个字符串转化成另一个字符串最少需要的增、删、改的字符数。给定两个字符串（字符串均由小写的a-z字符组成），计算其编辑距离

输入格式

输入为1行，共有两个字符串，字符串间以空格隔开，两个字符串的长度分别为len1、len2

输出格式

输出为一个数字，表示编辑距离

样例 #1

样例输入 #1

```
abcd aebcd
```

样例输出 #1

```
1
```

提示

100≤len1,len2≤1000

迭代法

```
def levenshtein(i, j):
    matrix = [[0] * j for _ in range(i)]
    matrix[0][:] = range(j)
    for k in range(i):
        matrix[k][0] = k
```

```
    for ii in range(1, i):
        for jj in range(1, j):
            matrix[ii][jj] = min(
                matrix[ii - 1][jj],
                matrix[ii][jj - 1],
                matrix[ii - 1][jj - 1] - (s1[ii - 1] == s2[jj - 1])
            ) + 1

    return matrix[-1][-1]


if __name__ == '__main__':
    s1, s2 = input().split()
    print(levenshtein(len(s1) + 1, len(s2) + 1))
```

递归法

```
def cache(func):
    memo = {}

    def inner(i, j):
        ans = memo.get((i, j), None)
        if ans is None:
            ans = memo[i, j] = func(i, j)
        return ans

    return inner


@cache
def levenshtein(i, j):
    if min(i, j) == -1:
```

```
        return max(i, j) + 1
    else:
        return min(
            levenshtein(i - 1, j) + 1,
            levenshtein(i, j - 1) + 1,
            levenshtein(i - 1, j - 1) + int(s1[i] != s2[j])
        )


if __name__ == '__main__':
    from sys import setrecursionlimit

    setrecursionlimit(12345678)
    s1, s2 = input().split()

    print(levenshtein(len(s1) - 1, len(s2) - 1))
```