1265：最长公共子序列

题目描述

一个给定序列的子序列是在该序列中删去若干元素后得到的序列。确切地说，若给定序列X=<x1,x2,…,xm>，则另一序列Z＝<z1，z2，…，zk>是X的子序列是指存在一个严格递增的下标序列<i1,i2,…,ik>,使得对于所有j=1,2,…,k有：Xij=Zj

例如，序列Z=<B,C,D,B>是序列X=<A,B,C,B,D,A,B>的子序列,相应的递增下标序列为<2,3,5,7>。给定两个序列X和Y，当另一序列Z既是X的子序列又是Y的子序列时，称Z是序列X和Y的公共子序列。例如，若X＝<A,B,C,B,D,A,B>和Y＝<B,D,C,A,B,A>，则序列<B,C,A>是X和Y的一个公共子序列,序列 <B,C,B,A>也是X和Y的一个公共子序列。而且，后者是X和Y的一个最长公共子序列．因为X和Y没有长度大于4的公共子序列。

给定两个序列X＝<x1，x2，…，xm>和Y=<y1,y2….yn>．要求找出X和Y的一个最长公共子序列。

输入

共有两行。每行为一个由大写字母构成的长度不超过1000的字符串，表示序列X和Y。

输出

第一行为一个非负整数。表示所求得的最长公共子序列的长度。若不存在公共子序列．则输出文件仅有一行输出一个整数0。

输入样例

ABCBDAB
BDCABA

输出样例

4

解析

最长公共子序列LCS(Longest Common Subsequence)

先来做几个简单的例子，来理清思路。

Str1 = "a"，Str2 = "abc"，则最长公共子序列为"a"；

Str1 = "ac"，Str2 = "abcd"，则最长公共子序列为"ac"；

我们定义dp[i][j]代表处于Str1的第i个位置Str2的第j个位置的最长公共子序列。

	0	a	b	c	d
0	0	0	0	0	0
a	0	1	1	1	1
b	0	1	2	2	2

空串和任意字符串的最长公共子序列都为0

如果两个字符串中某个位置的字符相同，表达式为：Str1[i]==Str2[j]，则说明它可能在LCS中，即：dp[i][j] = dp[i-1][j-1]+1。

这是什么原理呢？我们可以想象一下，对于两个字符串，如果它们的最后一个字母相同，我们就完全可以把这两个字母分别从这两个字符串当中删掉！这是一种所小规模的思想。如下所示：

假设str1 = "abcd"，str2="werewd"，这道题的最长公共子序列即为dp[4][6]，因为两个字符串的最后一个字母相同，因此我们可以同时删掉它，使原来的两个字符串变成了"abc"和"werew"，继而将原来求dp[4][6]就转变成了求dp[3][5]+1。

如果两个字符串中某个位置的字符不同，表达式为：Str1[i]!=Str2[j]，则说明两个字符至少有一个不在LCS中，但是到底是哪个不在呢，我们需要比较一下，即：dp[i][j] = max(dp[i-1][j],dp[i][j-1])。

编码

```
#include<bits/stdc++.h>

using namespace std;
//数据初始化
string S, T;
int F[1001][1001] = {{0}};

//主函数
int main() {
    //初始化数据
    //输入数据
    cin >> S;
    cin >> T;
    int ls = S.length();
    int lt = T.length();
    //事务处理
```

```
    for (int i = 1; i <= ls; i++) {
        for (int j = 1; j <= lt; j++) {
            //索引从1开始，所以需要-1，才是字符串的第一位
          if (S[i - 1] == T[j - 1])
                //相等，则说明可能在LCS中，则将原题缩小规模
              F[i][j] = F[i - 1][j - 1] + 1;
            else
                //不相等，说明至少有一个字母不在LCS中，需要比较判断
              F[i][j] = max(F[i - 1][j], F[i][j - 1]);
        }
    }

    //输出数据
    cout << F[ls][lt] << endl;
    return 0;
}
```