牛客网-华为机试练习题 80

题目描述

对于不同的字符串，我们希望能有办法判断相似程度，我们定义了一套操作方法来把两个不相同的字符串变得相同，具体的操作方法如下：

1 修改一个字符，如把“a”替换为“b”。

2 增加一个字符，如把“abdd”变为“aebdd”。

3 删除一个字符，如把“travelling”变为“traveling”。

比如，对于“abcdefg”和“abcdef”两个字符串来说，我们认为可以通过增加和减少一个“g”的方式来达到目的。上面的两种方案，都只需要一次操作。把这个操作所需要的次数定义为两个字符串的距离，而相似度等于“距离 + 1”的倒数。也就是说，“abcdefg”和“abcdef”的距离为1，相似度为1/2=0.5.

给定任意两个字符串，你是否能写出一个算法来计算出它们的相似度呢?

请实现如下接口

```
/* 功能：计算字符串的相似度
 \* 输入：pucAExpression/ pucBExpression：字符串格式，如："abcdef"
 \* 返回：字符串的相似度,相似度等于“距离 + 1”的倒数,结果请用1/字符串的形式,如1/2
 */
public static  String  calculateStringDistance(String expressionA, String expressionB)
{
    /* 请实现*/
    return null;
}
```

约束：

1、PucAExpression/ PucBExpression字符串中的有效字符包括26个小写字母。

2、PucAExpression/ PucBExpression算术表达式的有效性由调用者保证；
3、超过result范围导致信息无法正确表达的，返回null。

输入描述:

输入两个字符串

输出描述:

输出相似度，string类型

示例1

输入

```
abcdef
abcdefg
```

输出

```
1/2
```

解决代码：

```
import java.util.*;
import java.io.*;

public class Main {
    public static void main(String[] args) throws IOException {
                BufferedReader in = new BufferedReader(
                    new InputStreamReader(System.in)
                );
                String s1 = "";
                while ( null != (s1 = in.readLine())  ){
                //将两个输入字符串转为数组
                String s2=in.readLine();
                char[] cs1=s1.toCharArray();
                char[] cs2=s2.toCharArray();
                int[][] dp=new int[s1.length()+1][s2.length()+1];
                //用动态规划的方式获取一个数组变为另一个数组的步骤次数
                //初始化二维数组
                for(int row=1;row<=s1.length();row++){
                    dp[row][0]=row;
                }
                for(int col=1;col<=s2.length();col++){
                    dp[0][col]=col;
                }
                //动态规划
                for(int row=1;row<=s1.length();row++){
                    for(int col=1;col<=s2.length();col++){
                        if(cs1[row-1]==cs2[col-1]){
                            dp[row][col]=dp[row-1][col-1];
                        }
                        else{
                            int min1=Math.min(dp[row-1][col],dp[row][col-1])+1;
                            dp[row][col]=Math.min(min1,dp[row-1][col-1]+1);

                        }
                    }
                }
                System.out.println("1/"+ (dp[s1.length()][s2.length()]+1) );

            }
        }
    }
```