**翻硬币**

问题描述

小明正在玩一个“翻硬币”的游戏。

桌上放着排成一排的若干硬币。我们用 \* 表示正面，用 o 表示反面（是小写字母，不是零）。

比如，可能情形是：\*\*oo\*\*\*oooo

如果同时翻转左边的两个硬币，则变为：oooo\*\*\*oooo

现在小明的问题是：如果已知了初始状态和要达到的目标状态，每次只能同时翻转相邻的两个硬币,那么对特定的局面，最少要翻动多少次呢？

我们约定：把翻动相邻的两个硬币叫做一步操作，那么要求：

输入格式

两行等长的字符串，分别表示初始状态和要达到的目标状态。每行的长度<1000

输出格式

一个整数，表示最小操作步数。

样例输入1

\*\*\*\*\*\*\*\*\*\*  
o\*\*\*\*o\*\*\*\*

样例输出1

5

样例输入2

\*o\*\*o\*\*\*o\*\*\*  
\*o\*\*\*o\*\*o\*\*\*

样例输出2

1

锦囊1

贪心。

本题的C++参考代码如下：

#include <iostream>

using namespace std;

char reverse(char c)

{

if(c=='\*')

c='o';

else

c='\*';

return c;

}

int main()

{

char a[1000],b[1000];

cin>>a>>b;

int move=0;

for(int i=0;a[i]!='\0';i++)

{

if(a[i]!=b[i])

{

a[i]=reverse(a[i]);

a[i+1]=reverse(a[i+1]);

move++;

}

}

cout<<move<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <stdlib.h>

int turn(char a[], char b[])

{

int i;

int n = 0;

for(i = 0;a[i]!='\0';i++)

{

if(a[i] == b[i])

{

continue;

}

else

{

b[i+1]=(b[i+1]=='\*'?'o':'\*');

n++;

}

}

return n;

}

int main(void)

{

char a[1000];

char b[1000];

gets(a);

gets(b);

printf("%d\n", turn(a, b));

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Main{

public static void main(String[] args)throws Exception {

BufferedReader bu = new BufferedReader(

new InputStreamReader(System.in));

String s = bu.readLine();

char[] c1 = s.toCharArray();

s = bu.readLine();

char[] c2 = s.toCharArray();

int i = 0;

int c = 0;

boolean[] b = new boolean[c1.length];

boolean x = false;

while(!x){

if(c1[i] == c2[i]){

i++;

b[i] =true;

}else{

if(i < c1.length -1){

c1[i] = fz(c1[i]);

c1[i+1] = fz(c1[i+1]);

c++;

i++;

}

}

for(int i1 = 0;i1 < b.length;i1++){

if(b[i1] == true){

x = true;

} else{

x = false;

}

}

}

System.out.println(c);

}

public static char fz(char c ){

if(c == '\*'){

return 'o';

}else{

return '\*';

}

}

}