第三次题解

777. 在LR字符串中交换相邻字符

难度 中等 257 收藏 分享 切换为英文 接收动态 反馈

在一个由 'L' , 'R' 和 'X' 三个字符组成的字符串（例如"RXXLRXRXL"）中进行移动操作。一次移动操作指用一个"LX"替换一个"XL"，或者用一个"XR"替换一个"RX"。现给定起始字符串start和结束字符串end，请编写代码，当且仅当存在一系列移动操作使得start可以转换成end时， 返回True。

示例：

```
输入: start = "RXXLRXRXL", end = "XRLXXRRLX"
输出: True
解释:
我们可以通过以下几步将start转换成end:
RXXLRXRXL ->
XRXLRXRXL ->
XRLXRXRXL ->
XRLXXRRXL ->
XRLXXRRLX
```

提示：

- 1 <= len(start) = len(end) <= 10000。
- start和end中的字符串仅限于'L', 'R'和'X'。

样例：

输入：

> RXXLRXRXL
>
> XRLXXRRLX

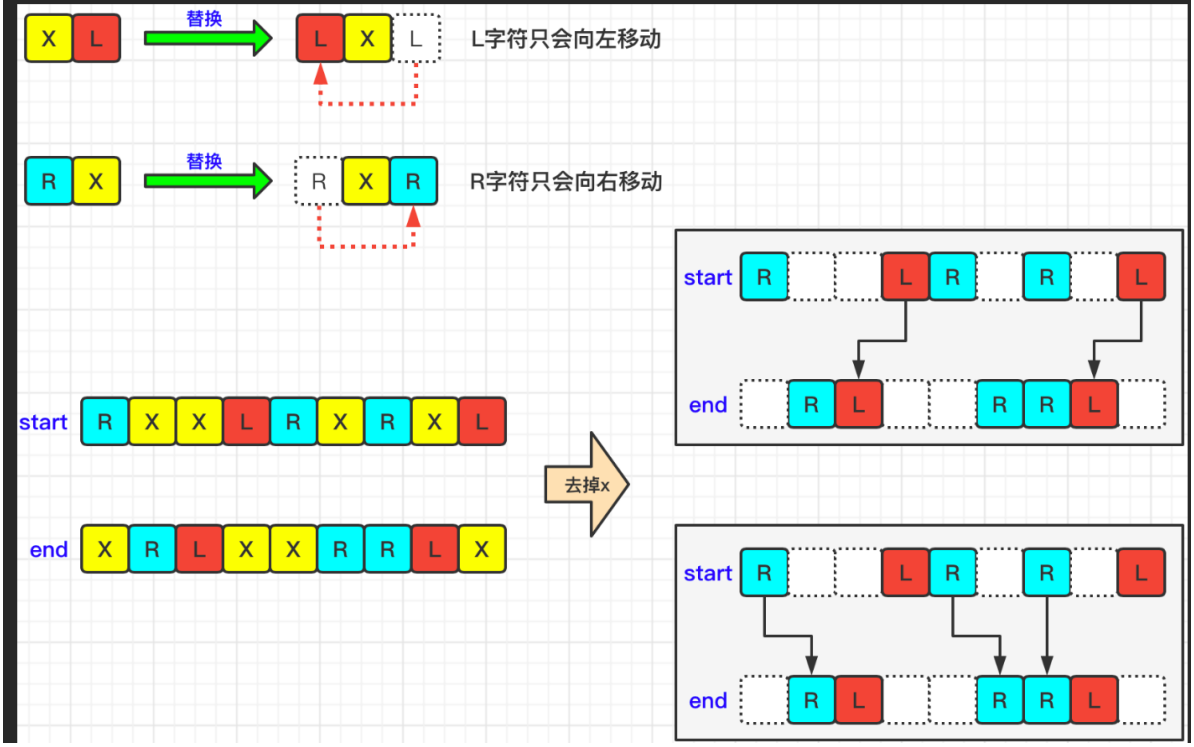
输出：

> true

那么我们发现这种规律之后，我们就可以只关注字符'R'和'L'了。判断规律如下所示：

> **条件一**：start中每位字符'L'的index，都要 **大于等于** end中每位字符'L'的index。
> **条件二**：start中每位字符'R'的index，都要 **小于等于** end中每位字符'R'的index。

只有同时满足了上述两个条件，才可以确定start中的字符'R'和字符'L'是可以移动成为end中的字符'R'和字符'L'。而且，题目中的提示部分，已经告诉我们**start和len的长度相同**，并且start和end中的字符串仅限于'`L`','`R`'和'`X`'，那么我们只需要再确定start与end中的字符'R'与字符'L'是一一对应的即可（即：个数都是相同的，可以一一对应上）。具体逻辑，请看下图所示：




```c
#include <stdio.h>
#include <string.h>
char start[500];
char end[500];
int canTransform(char * start, char * end);

int main()
{
    int a;
    gets(start);
    gets(end);
    a = canTransform(start,end);
    printf("%d",a);
    return 0;
}

int canTransform(char start[], char end[]) {
    int n = strlen(start);
    int i = 0, j = 0;
    while (i < n || j < n) {
        while (i < n && start[i] == 'X') i++;
        while (j < n && end[j] == 'X') j++;//对于x可以直接不处理
        if (i == n || j == n) return i == j;//到末尾了，如果都到末尾说明元素个数一样可以换到最后
```

```
        if (start[i] != end[j]) return false;//如果不一样说明L或者R的相对位置不一样
        if (start[i] == 'L' && i < j) return false;//一样则相对位置需要一样
        if (start[i] == 'R' && i > j) return false;
        i++; j++;
    }
    return i == j;
}
```

『MdOI R3』Operations

题目背景

这是这场比赛唯一一道没有题目背景的题，这就是本题的题目背景。

题目描述

给定非负整数 a, b，有两种操作：

1. 任意选择一个正整数 x，将两数都减去 x，执行一次该操作的代价为 c；
2. 任意选择一个正整数 x，将两数其中一个数乘以 x，另一个除以 x 后向下取整，执行一次该操作的代价为 d。

在这里，向下取整指使一个数变为**不大于它的最大的整数**，比如 3.5 向下取整为 3，-0.07 向下取整为 -1。

选择的 x 可以为任意正整数。在操作的过程中，可以把 a, b 变为负数。

你可以任意多次对这两个数操作，求将 a, b 都变成 0 的代价最小值。

输入格式

一行四个整数 a, b, c, d，用空格隔开，分别表示 a, b 的初始值和两种操作的代价。

输出格式

一行一个整数，表示代价的最小值。

样例

样例输入

```
9 36 1 3
```

样例输出

```
4
```

提示

【样例解释】

先使用一次 2 操作，选择 $x = 2$，将 a 乘 2，将 b 除以 2，得 $a = 18, b = 18$。
再使用一次 1 操作，选择 $x = 18$，将两个数都减去 18，得 $a = 0, b = 0$。
可以证明没有比上述操作代价更小的方案。

更多样例请到这里领取。

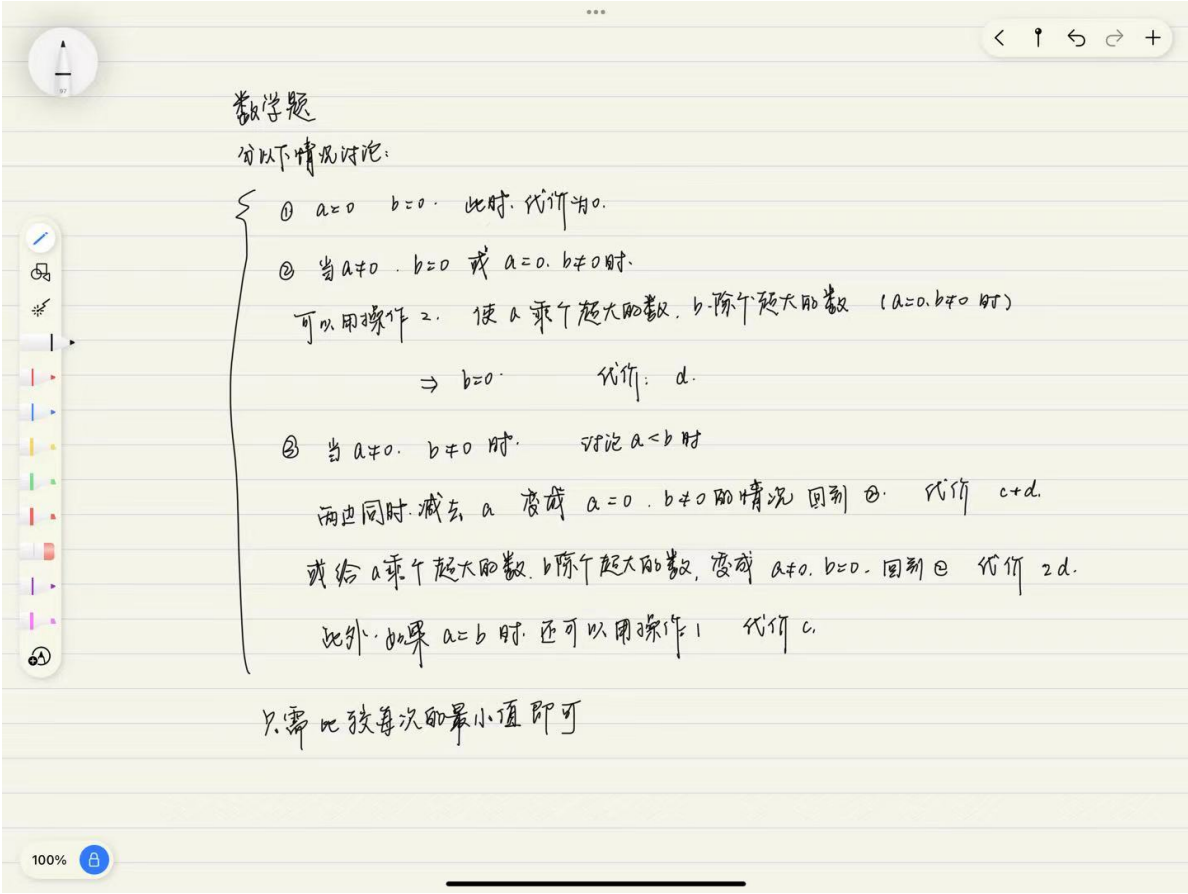
【数据范围】

本题采用捆绑测试，换言之，你只有通过一个子任务的所有测试点，才可以拿到该子任务对应分数。

子任务编号	$a = 0$	$b = 0$	$a = b$	$c = 1, d \geq 10^5$	$c \geq 10^5, d = 1$	分值
1	√	√				10
2	√					10
3			√	√		10
4				√		10
5			√		√	10
6					√	10
7						40

特殊性质如上所示，其中，√ 表示保证有该特殊性质，空格表示不保证有该性质。

对于所有数据，$0 \leq a, b, c, d \leq 10^9$。




```
1 #include <stdio.h>
```

```c
int main()
{
    int a,b,c,d;
    scanf("%d%d%d%d",&a,&b,&c,&d);
    int sum;

    if(a==0&&b==0)
        sum = 0;
    else if(a!=0&&b!=0){
        if(a!=b)
        sum = c < d? c+d:d+d;
        else
        sum = c < d+d? c:d+d;
    }
    else
        sum = d;

    printf("%d",sum);
    return 0;
}
```