上机题解

IV

2013-11-10

BUAA
北航沙河校区

目录

本次写过题解的同学：**李晨豪、户建坤、赵天宇、董舒印、普俊韬、从硕、张潮鹏、张馨如、武丽莎、李闪、闾佑霖**

以上同学已得到相应加分，若有问题再联系助教。

A：户建坤

题目描述

话说不知何年何月何日，天臧散人 Arthur 受神魔两界邀请，前往神魔之井对其进行了一次大规模的改造。
改造完成后，他和他的小伙伴惊讶地发现，自己被自己设计的迷宫困住出不去了……
幸运的是，Arthur 对迷宫的基本运作原理还是很清楚的（只不过这原理超出了他的探索范围而已）。
已知新的神魔之井迷宫由若干个房间构成，这些房间的变换遵从一个规律。如果当前时刻的房间数为偶数，那么下一个时辰，房间的个数就会减半。而如果是奇数，那么下一个时辰，房间的个数会乘以三再加上一。
于是机智的 Arthur 做出一个明智的决定——坐在原地等待。
已知当前房间的个数为 n，请你计算出他想要脱出迷宫需要等待的时间（单位：时辰）。

输入

多组测试数据。
每组数据为一行，包含一个整数 n（1≤n≤10000000），为当前时刻房间的个数。

输出

对于每组数据，输出一行，包含一个整数，为 Arthur 需要等待的时间。

输入样例

2
4
3

输出样例

1
2
7

HINT

只有一个房间的迷宫，还出不去么？

解题思路

本体是本次上机题中最简单的一题，但是知识量和注意事项却是因有尽有。现在分条列述，希望大家在以后遇到难题时不会因这种小细节而 Wrong ---Answer.

1 分清时间的开始。现在状态是等了 0 个时辰，第一次变化是等了 1 个时辰。注意是否需要加 1。

2 开始本体的核心算法。输入 n，判断 n 的奇偶，然后再把值赋给 n,完成一次循环，直到 n == 1；过程中有 counter 记数。**本题需要注意的是：为什么按照这种方法可以到最后让 n 等于 1？**或许这只对出题人需要考虑，但我们做题的人难道不应该弄懂原理来大约估算计算次数而防止超时吗？希望大家想一下原理。（反正我没想出来……）

3 输入的数据。一定要注意**输入的类型**，根据要求会应用不同的类型：如 int , long long , 还记得高精度摸的一题吗？那题应该用 char 或者 string 。此外，**数据的大小**也会根本上影响算法的类型，这样做是为了防止超时。比如：我神奇的室友李晨豪出的大只的回文数。至于本题，数据很小（并且算法步骤都不知道多少），所以正常的写就行了。

4 输出。注意格式

5 具体写出代码，参见下文。

综上所述，一道题应该小心的是 1--本身的算法正确性（结果输出的要对）2--算法要够快（防止超时）。（helped by Chenhao Li)

参考代码

附上一个最普通的代码：

```
#include <iostream>
using namespace std;


int main()
{
    int n,i;// i 即是计数器。本题用 int。
    while(cin >> n)//对应题目“多组数据”
    {
        i = 0;//切记!!!
        while ( n != 1)//出循环。
        {
            if ( n % 2 == 0)
            n= n /2;
            else
            n = 3 * n + 1;
            i++;
        }
        cout << i << endl;//输出。
    }
    return 0;//好习惯 - -
}
```

PS：大家一定要看这里！

虽然做题应该弄懂题目的原理，但是本题是不可证的！为什么最后会等于1？百度一下发现……

咳咳~~本题是著名的“角谷定理”和哥德巴赫猜想一样没有人能证明。

B：李晨豪

Description

Thor 是一个苦逼的 ACMer，要去 regional 被各路大神虐了。没办法，谁让 Thor 这么弱呢……还是收拾行李吧。唉，看到行李，Thor 的智商又捉鸡了。
Thor 有 4 个包，好吧忽略它们的高度，你可以把它们俯视看成一个个的正方形，Thor 的行李箱忽略高度也可以看成一个正方形。包之间不能互相压着，问 Thor 的行李箱至少要多大才能装的下所有行李。你只需要输出行李箱看成正方形的边长。

Input

多组测试数据。
每组数据只有一行，四个整数 a,b,c,d(1<=a,b,c,d<=1000)。为这些忽略高度后的包作为正方形的边长。

Output

对于每组测试数据，输出一个数，即你的答案。

Sample

输入：

```
2 2 2 2
2 2 1 2
```

输出：

```
4
4
```

Hint

第一个样例，就是个田字形
第二个样例，还可以看成田字形，只是右上角只有一个 1*1 的小块。
（就是个正方形里面套正方形，Thor 真不会描述）

题目分析

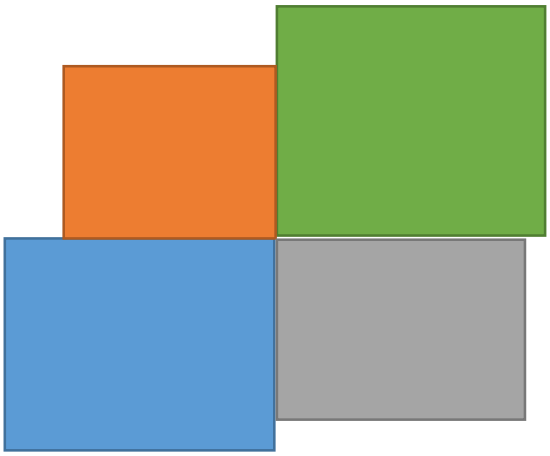
我们设答案为 ans，四个包的边长分别为 a>=b>=c>=d，那么一定有

Ans >= a + b;

现在我们直接构造 ans = a + b 这样的方案

右上角为 A，左下角为 B
那么对于边长小一点的 C，D
我们便可以把它们放在两个角落里

此时 ans = a + b



参考代码 ：

```
#include <iostream>
#include <cstdio>
#include <cstring>
int a[5];
int ans;
using namespace std;
int main()
{
    while(cin>>a[1]>>a[2]>>a[3]>>a[4])
    {
        ans = 0;
        int temp;
        for (int i = 1;i <= 4;i++)
            for (int j = i+1;j <= 4;j++)
            if (a[i] < a[j])
            {
                temp = a[i];a[i] = a[j];a[j] = temp;
```

```
            }       //只用找到最大的两个就行了，傻逼的排了序，忽略这段
                    //其实排序的有一种思想就是找到 n 个数中最大的放
                    //到第一个，再找出其余 n-1 中的最大的放到第二个
                    //直到 n==1。这种算法复杂度 o（n^2），然后用点数据
                    //结构可以优化到 o（nln（n））（堆之类的，很稳定）
                    //迷之音：你就是想为自己贴了麻烦的代码找借口吧！
                    //！！！喂！！！
        ans = max(ans,a[1]+a[2]);
        cout<<ans<<endl;
    }
}
/*看不懂数组的话可以把数组 a 看成 5 个变量*/
```

C：赵天宇 董舒印

Ps：

我觉得这道题蛮好的，于是就找来两位大神一起写题解，这样理解会深刻一点~~
好，接下来一起 OrzOrzOrzOrzOrzOrzOrz

Description

有一天天臧散人 Arthur 到了渝州东南的仙剑客栈，发现李逍遥这小子粗心大意在上酒的时候，有的桌上多上了酒，有的桌上没上酒。唉，谁让 Arthur 心软呢，不忍心看李逍遥被他婶婶骂，决定帮他——收拾一排桌子。
给你一个数 n 表示有多少个桌子，接下来给你一段整数序列表示每个桌上需要的酒量（假设这些桌子在一条直线上，且每个桌子之间的距离都是 1），正数表示多放了几瓶酒，负数表示应该放多少瓶酒。请你帮 Arthur 算一下他提着酒走的最短路程是多少。对了，Arthur 体力太渣，一次只能拿一瓶酒。

Input

第一行一个数 T 表示有 T 组数据。
接下来 T 组数据，每组数据有 2 行。
第一行一个数 n(1<=n<=1000)，表示桌子数量，
接下来第二行有一段数列 Ai(|Ai|<=1000)，表示每个桌子上应该放的酒。保证数列总和为 0。

Output

对于每组测试数据，输出一个数，表示 Arthur 拿着酒走的最少总路程。

Sample Input：

2
3
-1 2 -1
6
-1 -1 -1 3 -1 1

Sample Output：

2
7

Hint

其实就是每瓶酒走的路程之和。

解题思路：天宇版

题目已经说明是每瓶酒的路程之和，对于最简单的-1　1 局面，从 1 出发到-1 与从-1 出发到 1 结果是一样的，因此我们可以说，酒的路线是可逆的。以情况-1　-1　-1　3　-1　1 为例，根据可逆原理，从最左端的数开始读入，分别记录酒数（若为正值则表示拥有的酒数，若为负值则表示需要的酒数）和步数。

第一步读入-1　　酒数：-1　　步数：0
第二步读入-1　　酒数：-2　　步数：|-1|
第三步读入-1　　酒数：-3　　步数：|-1|+|-2|
第四步读入 3　　酒数：0　　步数：|-1|+|-2|+|-3|
第五步读入-1　　酒数：-1　　步数：|-1|+|-2|+|-3|+|0|
第六步读入 1　　酒数：0　　步数：|-1|+|-2|+|-3|+|0|+|-1|

总结为公式：若数据为 $x_1,x_2,\cdots,x_n$；则所需步数为：$|x_1|+|x_1+x_2|+\cdots+|x_1+x_2+\cdots+x_n|$,即：

$$STEP = \sum_{i=1}^{n} abs(\sum_{j=1}^{i} x_j)$$

参考代码：

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
	int t;
	cin >> t;
	while (t--)
	{
		int n,sum=0,step=0;			//初始化酒数和步数
		cin >> n;				//接收桌子数量值
		while (n--)
		{
			int num_wine;
			cin >> num_wine;	//接收当前桌子的酒数
			sum = sum + num_wine;	//计入当前积累的酒数
			step = step + abs(sum);//计入当前积累的步数
		}
		cout << step << endl;
	}
	return 0;
}
```

题目分析：舒印版

建议大家多写几个数据试一下就明白了，不管你怎么分配，只要你不把放在桌子上的酒再次拿走，并且路途没有交叉（什么叫交叉我举个例子。1 -1 1 -1 只要我们不把第一个1拿到最后一个-1就可以，这个还是多试一下比较容易理解，很难表达。），路途就是最短的，换句话说，每一个酒只要走向最近的需要酒的地方。就可以了。

Ps：看到这个题我第一反应是数组，但是发现数组非常麻烦，又想起这次的题是不需要数组的，在试了几组数之后做出了这个题。

代码示例：

```
#include<iostream>

#include<cmath>

using namespace std;



int main()

{

    int t,x,n;

    cin>>t;

    while(t--)

    {

        cin>>n;

        int s=0,c=0;

        while(n--)

        {

            cin>>x;/*核心算法，我们从左边走到右边，然后算手上的酒的个数，不管是正数还是负数都需要每瓶酒+1的路程。这里要注意是每次输入之后立刻计算。而每两个数之间的距离为1。正数表示从左向右拿，负数表示从右向左，但是这样肯定不会交叉的。*/

            c=c+x;

            s=s+abs(c);//也可以写成 s=s+abs(c)*1方便理解，abs 是取绝对值。
```

```
        }
        cout<<s<<endl;
    }
}
```

D：普俊韬

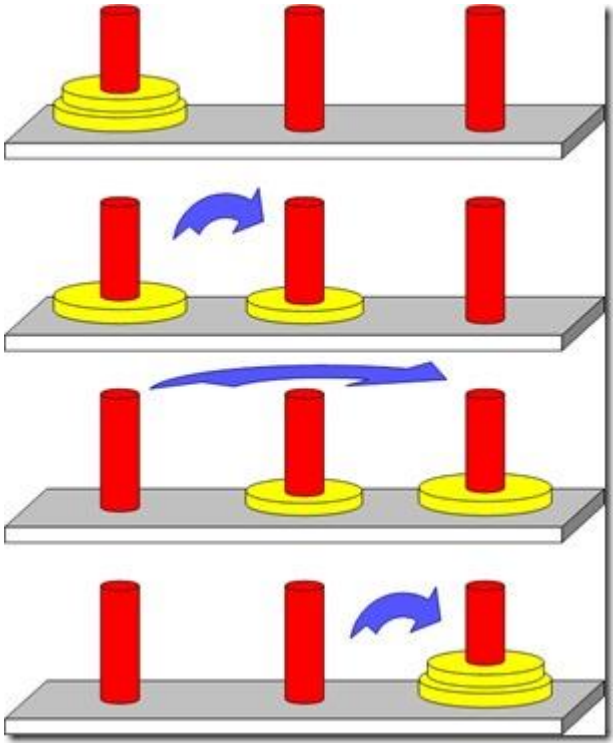
Problem Description

汉诺塔游戏，我们都非常熟悉吧~

汉诺塔是源自印度神话里的玩具。上帝创造世界的时候做了三根金刚石柱子，在一根柱子上从下往上大小顺序摞着 64 片黄金圆盘。
上帝命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。
但是，如果真正把 64 片黄金圆盘移动完成之后，那将会是世界末日了！！！！！！

现在给你盘子的个数，你知道按照什么步骤移动盘子才能达到目的么？

下面给出的两个盘子的例子：



Input

第一行输入一个整数 t（t <= 10），表示测试的组数。
接下来的 t 行，每行输入一个正整数 n（0 < n <= 12）,表示盘子的个数。

Output

对于每组测试数据，首先输出一行 Case #X：，表示第 X 组测试数据。
接下来，每行都输出移动盘子的步骤，对于每次移动，以"Move from X to Y"表示，表示需要将 X 柱子上最顶层的盘子移动到 Y 柱子上，
其中 X，Y 是 A,B,C 中的一个，当然，X 和 Y 不可能相同

Sample Input

```
2
2
3
```

Sample Output

```
Case #1:
Move from A to B
Move from A to C
Move from B to C
Case #2:
Move from A to C
Move from A to B
Move from C to B
Move from A to C
Move from B to A
Move from B to C
Move from A to C
```

解题思路：

咳咳，老师讲过，记不住了？好吧，大蚂蚁讲递归用的例子，不知道了？好吧，首先这个东西用递归毫无悬念，然后，怎么用递归？
最基本的情况是移动一个盘子，所以，递归结束的地方就是只移动一个盘子的时候。
非基本情况是移动 n 个盘子，那么这样想。

要让 n 个盘子从 A 移动到 C，得先把第 n 个盘子上面的 n-1 个盘子搬到 B，然后把第 n 个搬到 C，然后把 B 上的所有盘子搬到 C。
要把 n-1 个盘子从 A 搬到 B，得先把第 n-1 个盘子上面的 n-2 个盘子搬到 C，然后把第 n-1 个盘子搬到 B，然后把 C 上的所有盘子搬到 B。

按照这样的想法，n 就在逐步减小，最后变成基本情况，递归结束。

所以，请看下面的代码：

```
#include <iostream>
#include <iomanip>
#include <cstdlib>

using namespace std;

//移动盘子函数
void function (int n ,  char begin ,  char end ,  char middle)
{
    if (n==1)
    cout << "Move from " << begin << " to " << end << endl;//基本情况，起始到终点

    //注意函数参量位置的变化

    else
    {
        //n-1 个从起始到中间
        function (n-1 ,  begin ,  middle ,  end);

        //最后一个从起始到终点
        cout << "Move from " << begin << " to " << end << endl;

        //n-1 个从中间到终点
        function (n-1 ,  middle ,  end ,  begin);
    }
}


//函数写好了，主函数就无脑输出就行
int t , n;
char A = 'A';
char B = 'B';
char C = 'C';
```

```
int main ()
{
    cin >> t;
    for (int j = 1 ; j<=t ; j++ )
    {
        cin >> n;

        cout << "Case #" << j << ":" << endl;

        function (n ,  A ,  C ,  B);//起始柱为 A，终点柱为 C，中间柱点为 B
    }
}
```

对了，对于发生 Compiler Error 的同学，注意先运行一次再交哈（即使，用的是老师的代码），以备不测~0.0

E：李晨豪

题目描述

Arthur 这次也准备走一次简单路线，于是题面很简单。
给你平面直角坐标系下两个矩形中位于对角线上两个顶点的坐标，请判断这两个矩形是否相交。
矩形的两边与坐标轴平行，也就是说不会出现斜置的矩形。
相切的情况不计入考虑范围。

输入

多组测试数据，每组数据为两行。
对于每组数据，第一行为四个整数，表示第一个矩形的两个对角顶点的 x,y 坐标。第二行为四个整数，表示第二个矩形的两个对角顶点的 x,y 坐标。
保证以上数据在 int 范围内。

输出

对于每组数据，如果相交，则输出“YES”，否则输出“NO”

输入样例

0 0 2 2
1 1 3 3
0 0 0 1
1 1 1 2

输出样例

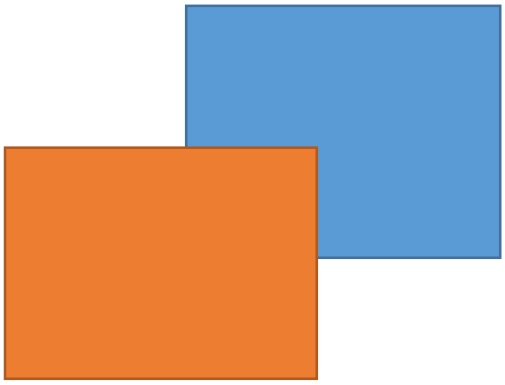
YES
NO

Analysis:

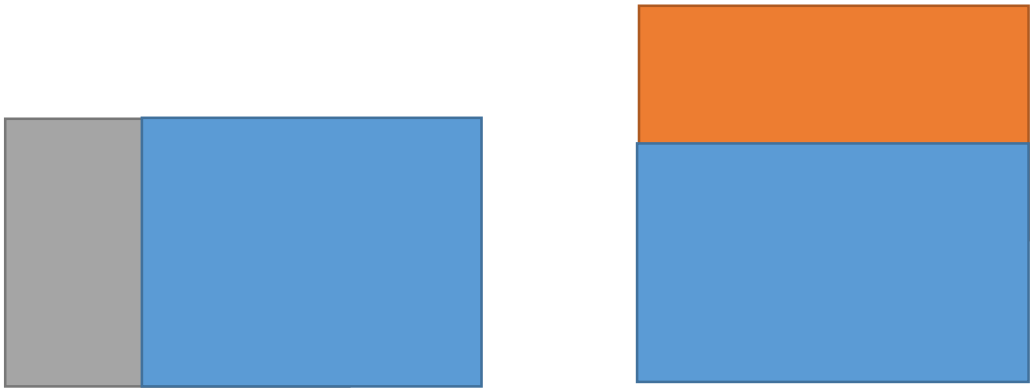
刚开始我以为这些矩形可以斜着 orz，然后就悲剧了。所以大家一定要好好看题。

首先我们来分析相交的几种情况：

(1)

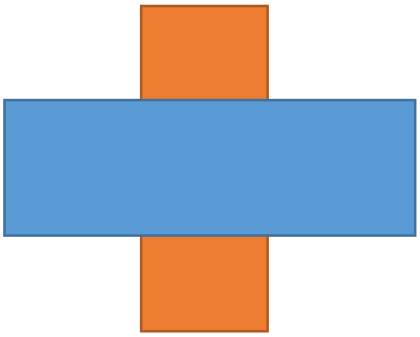


(2)



(3)完全重合

(4)



Solve:

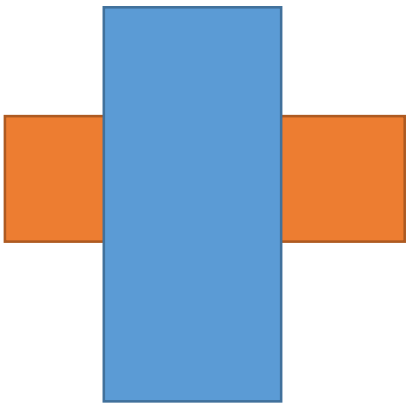
（x1,y1,x2,y2,
x3,y3,x4,y4）

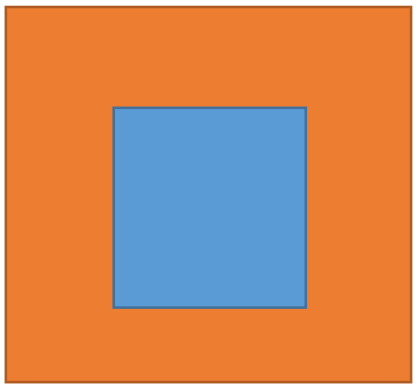
(1) 我们就用学长说的，判断点是否在对方里面。

(2) 利用重心距离，dx <= (d1 + d2)(d1,d2 为重合的那条边的长度，dx 为重心距离)。

(3) 这个不用多说。

(4) (x3-x1)*(x4-x2)<0 && (y3-y1)*(y4-y2)<0.我们这两个条件表示把二维的分为一维的 x 轴，y 轴分别考虑（因为和坐标轴平行）。
(x3-x1)*(x4-x2)<0 表示第一个矩形在 x 轴上，被第二个矩形包着（或相反）。
(y3-y1)*(y4-y2)<0 表示第一个矩形在 y 轴上，被第二个矩形包着（或相反）。





这个条件利用了（1）（2）（3）的筛选，因为有点在对方里面的，和（2）（3）的情况已经判断完毕。

错误代码：

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cmath>
int x1,y_1;//1
int x2,y2;//1
int x3,y3;//2
int x4,y4;//2
```

```
int x5,y5;//1
int x6,y6;//1
int x7,y7;//2
int x8,y8;//2
using namespace std;
int main()
{
	while (cin>>x1>>y_1>>x2>>y2)
	{
		cin>>x3>>y3>>x4>>y4;
		if (x1 == x2 || y_1 == y2 ||x3 == x4 || y3 == y4)
		{
			cout<<"NO"<<endl;
			continue;
		}
		x5 = x1;y5 = y2;
		x6 = x2;y6 = y_1;
		x7 = x3;y7 = y4;//*************
		x8 = x4;y8 = y3;//*************

		//cout<<x3<<" "<<y3;
		if (x7 > min(x1,x2) && x7 < max(x1,x2) && y7 > min(y_1,y2) && y7 <max(y_1,y2))
		{
			cout<<"YES"<<endl;
			continue;
		}
		if (x8 > min(x1,x2) && x8 < max(x1,x2) && y8 > min(y_1,y2) && y8 <max(y_1,y2))
		{
			cout<<"YES"<<endl;
			continue;
		}
		if (x3 > min(x1,x2) && x3 < max(x1,x2) && y3 > min(y_1,y2) && y3 <max(y_1,y2))
		{
			cout<<"YES"<<endl;
			continue;
		}
		if (x4 > min(x1,x2) && x4 < max(x1,x2) && y4 > min(y_1,y2) && y4 <max(y_1,y2))
		{
			cout<<"YES"<<endl;
			continue;
		}
		if (x1 > min(x3,x4) && x1 < max(x3,x4) && y_1 > min(y3,y4) && y_1 <max(y3,y4))
		{
			cout<<"YES"<<endl;
```

```
            continue;
        }
        if (x2 > min(x3,x4) && x2 < max(x3,x4) && y2 > min(y3,y4) && y2 <max(y3,y4))
        {
            cout<<"YES"<<endl;
            continue;
        }
        if (x5 > min(x3,x4) && x5 < max(x3,x4) && y5 > min(y3,y4) && y5 <max(y3,y4))
        {
            cout<<"YES"<<endl;
            continue;
        }
        if (x6 > min(x3,x4) && x6 < max(x3,x4) && y6 > min(y3,y4) && y6 <max(y3,y4))
        {
            cout<<"YES"<<endl;
            continue;
        }
        if (abs(x2-x1)==abs(x4-x3) || abs(y2-y_1)==abs(y4-y3))
        {
            int xx1,yy_1,xx2,yy2;
            xx1 = (x1 + x2)/2;yy_1 = (y_1 + y2)/2;
            xx2 = (x3 + x4)/2;yy2 = (y3 + y4)/2;
            int d = (xx2 - xx1)*(xx2 - xx1) + (yy2 - yy_1) * (yy2 - yy_1);
            if (yy_1 == yy2 && abs(y2-y_1)==abs(y4-y3))
            if (d < (x2 - x1) * (x2 - x1))
            {
                cout<<"YES"<<endl;
                continue;
            }
            if (xx1 == xx2 && abs(x2-x1)==abs(x4-x3))
            if (d < (y2 - y_1) * (y2 - y_1))
            {
                cout<<"YES"<<endl;
                continue;
            }
        }
        if (((x3-x1)*(x4-x2)<0)&&((y3-y_1)*(y4-y2)<0))
        {
            cout<<"YES"<<endl;
            continue;
        }
        cout<<"NO"<<endl;
    }
}
```
这个程序还是漏了一种情况，大家想想是哪个？

So：

我们发现相交竟然有如此多的情况 orz，因此我们试着考虑不相交的情况。

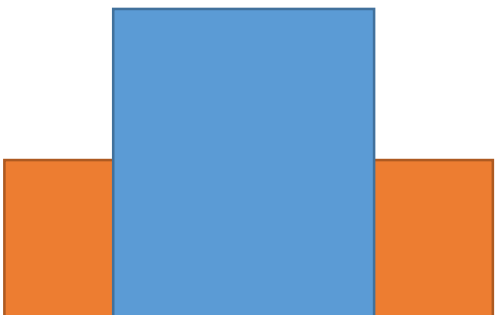
我们先求出左上角的坐标（xx1，yy1）宽为 w1 = abs（x2 – x1） h1 = abs（y2 – y1）

（xx2，yy2）宽为 w2 = abs（x4 – x3） h2 = abs（y4 – y3）

因此，只要有（xx1+w1<=xx2）||（xx2+w2<=xx1）||（yy1+h1<=yy2）||（yy2+h2<=yy1）即可判断不相交.(包含相切)

```
#include <iostream>
#include <cstdio>
#include <cmath>
int x1,y_1;//不能声明 y1，说是 cmath 里面声明过了，我也不知道 orz
int x2,y2;//
int x3,y3;//
int x4,y4;//
int w1,w2;
int h1,h2;
int xx1,yy1;
int xx2,yy2;
using namespace std;
int main()
{
    while (cin>>x1>>y_1>>x2>>y2)
    {
        cin>>x3>>y3>>x4>>y4;
        if (x1 == x2 || y_1 == y2 ||x3 == x4 || y3 == y4)//判断数据是否合法
        {
            cout<<"NO"<<endl;
            continue;
        }
        xx1 = min(x1,x2);yy1 = min(y_1,y2);
        xx2 = min(x3,x4);yy2 = min(y3,y4);
        h1 = abs(y2-y_1);w1 = abs(x2-x1);
        h2 = abs(y4-y3);w2 = abs(x4-x3);
        if ((xx1+w1<=xx2)||(xx2+w2<=xx1)||(yy1+h1<=yy2)||(yy2+h2<=yy1))
            cout<<"NO"<<endl;
        else
            cout<<"YES"<<endl;
    }
}//求大神出数据验程序
```

Ps：



这个就是第一个程序的问题，有兴趣的同学可以自己写写。

```
/*
十分抱歉，因为能力原因以及出题目时候的考虑不严谨以至于此题数据有误，故所有交过该题的人都算作 AC（虽然对一些人不太公平），同时此题暂不发布。
助教表示以后尽最大努力不再犯这种错误。
同样的，等你们中有人当助教的时候也尽量不要犯我们犯过的错误。
再一次，十分抱歉 T—T
*/
```

F：丛硕

题目描述

有这样一个传说中的世界……

> 凡《南次三以》之首，自天虞之山以至南禺之山，凡一十四山，六千五百三十里。其神皆龙身而人面。其祠皆一白狗祈，稍用稌……
>
> 凡《西次四经》自阴山以下至于崦嵫之山，凡十九山，三千六百八十里。其神祠礼，皆用一白鸡祈。糈以稻米，白菅为席……
>
> 凡《北次三经》之首，自太行山以至于毋逢之山，凡四十六山，万二千三百五十里。其神状皆马身人面者廿神。其祠之，皆用一藻茝瘗之。
>
> 其十四神状皆彘身而载玉。其祠之，皆用一璧瘗之。大凡四十四神，皆用稌糈米祠之，此皆不火食……
>
> 凡《东次四经》之首，自北号之山至于太山，凡八山，一千七百二十里……

上述这传奇般的世界，名为山海，有书载之，曰《山海经》。

于是让我们腾云驾雾，离开那大荒之中的不周之山，来到千里之外，这四方高低错落的云崖岚雾之间。
与上次不同，这次我们将要见到的，不仅仅是一座山峰，也不仅仅是山峰，还会有峡谷。

与上次相同，无论是山峰还是峡谷，我们都将它们假设为二维平面上的一圆锥体，山峰正置（凸起），峡谷倒置（凹陷）。
现给你这山海四方的地势起伏，以及某一地点的坐标（x,y），请你计算该点所在位置的高度。保留小数点后2位。
具体规则如下：

① 每座山峰（或峡谷）的信息为——底面圆心的坐标（a,b）、底面半径 r、高 h 以及一个字符 c，表示其类型，‘M’为山峰，‘C’为峡谷。

② 如果该点所在坐标同时位于多座山峰的区域范围内，取最高的。

③ 如果该点所在坐标同时位于多座峡谷的区域范围内，取最低的。

④ 如果该点所在坐标同时位于山峰与峡谷的区域范围内，取峡谷的。

⑤ 除山峰与峡谷外，其他位置高度均为0。

输入

多组测试数据。
对于每组数据，第一行为两个整数，用空格隔开，为某一点的坐标 x,y。
第二行一个整数 n（1≤n≤100），表示地势中山峰与峡谷的总数目。
接下来 n 行，每行包含4个整数与一个字符，用空格隔开。依次分别代表 a,b,r,h,c
保证输入数据合理。

输出

对于每组数据，输出一个实数，保留小数点后两位，为该点所在位置的高度。具体见样例

输入样例

0 0
1
0 0 1 10 M

输出样例

10.00

Analysis：

$$\text{Distance} = \sqrt[2]{(x - x0)*(x - x0) + (y - y0)*(y - y0)}$$

当山体或峡谷的半径大于距离时方构成影响
当半径大于距离时，计算锥体的高度（或深度），根据山谷还是峡谷来判断是否需要取用这一组数据。
计算高度(深度)公式为：H = h * (r-distance)/r
不多说，见代码。

Code for Reference：

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int main()
{
    double a,b,r,h;          //a,b,r,h,x,y 的意义和题里的意思相同，虽然是整数，但此处设为double
    double x,y;          //方便后面的计算（不必转换类型）
    double currentH,realH,distance;
    //currentH:当前计算所得高度（深度）的绝对值，因为题给数据是绝对值。
    //realH:最终记录的高度，就是用来输出的那个

    char ls,c;
    //ls: LandScape，记录当前的地形，这样当山峰和峡谷都存在时可以判断应该记录哪个。
    int n,i;
    while(cin>>x>>y)
    {
        ls = 0;    //地形初值，不确定记录哪个
        realH = 0;//记录高度
        cin >> n;
        for(i=0;i<n;i++)
        {
            cin >> a>>b>>r>>h>>c;
            distance = sqrt((a-x)*(a-x)+(b-y)*(b-y)); //计算距离
            if(distance>r)continue;                    //判断是否构成影响，非常关键！
            currentH = (r-distance)*h/r;               //运用公式计算当前锥体高度
            if(!ls)ls=c;                              //如果不知道地形，记录当前地形
```

```
                //如果此处为山峰，而这个新锥体也是山峰
                if(c=='M' && ls=='M')
                {
                    if(realH < currentH)
                        realH = currentH;
                }
                //如果此处为山谷
                if(c=='C')
                {
                    if(ls=='M')
                    {
                        //如果此处原来是山峰，则强制转换为山谷，且记录当前峡谷深度
                        ls='C';
                        realH = -currentH;
                    }
                    else if(realH>-currentH)realH = -currentH;
                }
            }
            //输出，注意-0.00 的处理
            cout << fixed << setprecision(2)<<(fabs(realH)>0.001?realH:0.00) << endl;
        }
        return 0;
}
```

G：张潮鹏

题目描述

实在闲得没事做的天臧散人 Arthur 一日跑到了鬼界，找起了转轮镜台的麻烦来。既没有找天青叙旧，也没有和风雅颂猜拳，猜猜他干什么了？
“咦，随便拿个数字放在这镜子上，就成了一个回文数？好神奇~”(—_—|||)
鬼界至宝之一怎能任你如此戏弄？阎罗勃然大怒，作为惩罚，他决定让 Arthur 计算闭区间[a,b]上全部的回文数的个数。于是走投无路的 Arthur 找到了你……

输入

多组测试数据。
每组数据为一行，包含两个整数 a,b（1≤a≤b≤INT_MAX），用空格隔开。

输出

对于每组数据，输出一行，包含一个整数，为区间内回文数的个数。

输入样例

1 1
1 10

输出样例

1
9

——

——

Think:

这道题略坑。题目简单，但超时无数。很多人习惯用 for 循环对从 a 到 b 中的每个数进行是否为回文数的判断后再累加的算法。但是，结果。。。。

因此，我们要想方设法减少循环的次数，可以从 1 到 a 的回文数 h(a)和 1 到 b 的回文数 h(b)再由 h(b)-h(a)得出结果。问题转化为如何快速地求 h(a)与 h(b)。

先有个基础，就是知道以下常识：

各个区间的回文数个数：

1——10： 9
10——100：9
100——1000： 90=9*10^1
1000——10000：90=9*10^1
10000——100000： 90=9*10^2
……
依此类推

然后求 h(a)与 h(b)，不妨这么想：我们要求 1 到 12345 的回文数个数，分为以下步骤

1：求[1,10000]的回文个数，用累加，不要枚举
2：求[10000,12000]的回文个数
3：求[12000,12300]的回文个数
……

以下看代码，关键部分已红色标注

Code:

```
#include <iostream>
using namespace std;
int aint[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
void initarray()//两次用到同个数组，因此设置一个对数组赋初始值的函数
{
	for (int i = 0; i < 10; i++) aint[i] = 0;
}
int shixp(int n)//求 10^n 的函数
{
	int result = 1;
	for (int i = 0; i < n; i++) result *= 10;
	return result;
}
int huiwen(int nd)//求区间[1, nd]之间的回文数个数。主要用累加的方法求，无需每个判断
//枚举
{
	int d = nd;
	int i = 0;
	for ( ; d >= 10; i++)//把一个 i 位数的整数 n 用 i 个数组的形式依次来存储
	{
		aint[i] = d % 10;
		d /= 10;
	}
	aint[i] = d;
	int result = 0;
	for (int j = 0; j <= i - 1; j++) result += 9 * shixp(j / 2);
	result += (aint[i] - 1) * shixp(i / 2);
	for (int j = i - 1; j > (i + 1) / 2 - 1; j--)
	result += aint[j] * shixp(j - (i + 1) / 2);
	for (int j = 0; j <= i / 2; j++)
		aint[j] = aint[i - j];
	int sd = 0;
	for (int j = 0; j <= i; j++)
		sd = sd * 10 + aint[j];
	if (nd >= sd)
		result++;
	initarray();
	return result;
}

bool ishuiwen(int n)//判断一个数是否为回文数
```

```
{
    bool result = true;
    int i = 0;
    for ( ; n >= 10; i++)//把一个 i 位数的整数 n 用 i 个数组的形式依次来存储
    {
        aint[i] = n % 10;
        n /= 10;
    }
    aint[i] = n;
    for (int j = 0; j < (i + 1) / 2; j++)
        if (aint[j] != aint[i - j]) result = false;
    initarray();
    return result;
}

int main()
{
    int a, b;
    while (cin >> a >> b)
    {
        int c = huiwen(b) - huiwen(a);
        if (ishuiwen(a))
            c++;
        cout <<c<<endl;
    }
}
```

PS:zcp 大神是这次除了学长与出题人唯一 AC 的人。大家 orz。

其实我们也可以想一想 12345，先把 10000 以内的算出来，我们发现五位数的回文数是由三位数生成的，345 –>54345，由此对于一个数 12345，我们直接先以 123 对称得 12321，然后我们发现 12321<12345,那么回文数的数量为 123-100+1 = 24 个。若对称后的数大于原数，那么不加一。

有问题联系助教(TA)

刘翰诚

E-mail: trashLHC@163.com

QQ: 773799131

梁明阳

E-mail: lmysoar@hotmail.com
QQ: 1012004860