E RecTangles ! RecTangles ! RecTangles !

张星尧，樊伟富，渣诚

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会相互重叠。
矩形的两边与坐标轴平行，也就是说不会出现斜置的矩形。

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坐标（x1,y1）与(x2,y2)。
第二行为四个整数，表示第二个矩形的两个对角顶点的坐标（x3,y3）与(x4,y4)。
(0≤x1,y1,x2,y2,x3,y3,x4,y4≤100)

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

HINT

重叠一个顶点或者一条边也算重叠

请考虑矩形合理性

解题思路

咳咳，这道题跟第四次上机里某道题的题目基本类似，说差别的话大家可能都会注意到合理性和边相切的判断。不过不知道大家有没有发现一个细节：$(0\leqslant x1, y1, x2, y2, x3, y3, x4, y4\leqslant 100)$，这个范围是上次题里木有的，刨除掉 Arthur&Thor 闲的没事恶作剧的可能，这个很可能是一种新思路的成立条件。于是顺着这个角度想 ,我们就能发现：**矩形可以化简成一个个点构成的点阵 ,两个矩形相交的充要条件就是两个点阵有公共点。**这样就很好的避免了对各种蛋疼情况的讨论。

参考代码：

上面两位同学的思路是非常好的，但是其代码却使用了四层嵌套，这样就不是很好喽，其实这四层嵌套完全可以变成两层的。于是这一次使用渣诚的代码.....

```
#include <iostream>
#include<fstream>
#include<algorithm>
#include<cmath>
using namespace std;
typedef struct{
    int x,y;
}Point;    //结构体，不用被吓到了，就当成一个坐标就好了
char map[111][111];  //全局地图，用来绘图的
void Init()   //初始化的时候全置为’0’
{
    for(int i=0;i<=100;i++)
       for(int j=0;j<=100;j++)
         map[i][j]='0';
}
class Rect{//上次说的矩形类
      public:
     Rect(int x1,int y1,int x2,int y2){  //构造函数
           legal=true;
           if(x1==x2||y1==y2)
               legal=false;    //legal 记录矩形是否合理,不合理为 false
           else{   //否则构造矩形四个顶点
              p1.x=min(x1,x2);
              p1.y=min(y1,y2);
              p2.x=min(x1,x2);
              p2.y=max(y1,y2);
              p3.x=max(x1,x2);
              p3.y=max(y1,y2);
              p4.x=max(x1,x2);
              p4.y=min(y1,y2);
           }
     }
     bool isLegal(){return legal;}  //返回是否合理
     Point getP1(){return p1;}      //下面四个函数分别返回四个顶点
     Point getP2(){return p2;}
     Point getP3(){return p3;}
     Point getP4(){return p4;}
     private:
```

```
        bool legal;          //legal 记录是否合理
        Point p1,p2,p3,p4;  //四个顶点坐标
};

bool draw(Rect r,int sign)
{//从左下绘制到右上
    for(int i=r.getP1().x;i<=r.getP3().x;i++)
       for(int j=r.getP1().y;j<=r.getP3().y;j++)
          if(map[i][j]!='0')  //如果出现不是最初的’0’的情况
             return false;    //说明不可以填充
          else
             map[i][j]=(char)('0'+sign); //否则，标记，表明已填充
    return true;   //如果上述填充完毕，则返回 true，说明完整填充
}

int main()
{
     int x1,x2,x3,x4,y1,y2,y3,y4;
     while(cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4)
     {
         Init();
         Rect r1(x1,y1,x2,y2),r2(x3,y3,x4,y4);//构造矩形
         //四个判断条件，顺序很重要，可以省略不必要的计算
(r1.isLegal()&&r2.isLegal()&&!(draw(r1,1)&&draw(r2,2)))?cout<<"YES"<<
endl:cout<<"NO"<<endl;
     }
     return 0;
}
```