题目描述

- 给出3组点坐标(x, y, w, h)，-1000<x,y<1000，w,h为正整数。
- (x, y, w, h)表示平面直角坐标系中的一个矩形；
- x, y为矩形左上角坐标点，w, h**向右w，向下h。**
- (x, y, w, h)表示x轴(x, x+w)和y轴(y, y-h)围成的矩形区域；
- (0, 0, 2, 2)表示 x轴(0, 2)和y 轴(0, -2)围成的矩形区域；
- (3, 5, 4, 6)表示x轴(3, 7)和y轴(5, -1)围成的矩形区域；
- 求3组坐标构成的矩形区域重合部分的面积。

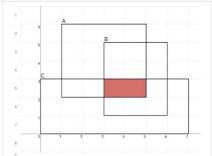
输入描述

3行输入分别为3个矩形的位置，分别代表"左上角x坐标"，"左上角y坐标"，"矩形宽"，"矩形高" -1000 <= x,y < 1000

输出描述

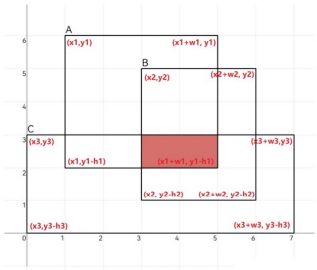
输出3个矩形相交的面积，不相交的输出0。

用例

输入	1 6 4 4 3 5 3 4 0 3 7 3
输出	2
说明	

题目解析

为图示如上坐标



如上图所示，交叉区域的

长：Math.min(x1+w1, x2+w2, x3+w3) - Math.max(x1, x2, x3)

宽：Math.min(y1, y2, y3) - Math.max(y1-h1, y2-h2, y3-h3)

如果长度 <=0 或者 宽度<=0，则三个矩形不存在交叉区域，返回0

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int x1 = sc.nextInt();
    int y1 = sc.nextInt();
    int w1 = sc.nextInt();
    int h1 = sc.nextInt();

    int x2 = sc.nextInt();
    int y2 = sc.nextInt();
    int w2 = sc.nextInt();
    int h2 = sc.nextInt();

    int x3 = sc.nextInt();
    int y3 = sc.nextInt();
    int w3 = sc.nextInt();
    int h3 = sc.nextInt();

    int wid = getMin(x1 + w1, x2 + w2, x3 + w3) - getMax(x1, x2, x3);
    if (wid <= 0) {
      System.out.println(0);
      return;
    }

    int hei = getMin(y1, y2, y3) - getMax(y1 - h1, y2 - h2, y3 - h3);
    if (hei <= 0) {
      System.out.println(0);
      return;
    }

    System.out.println(wid * hei);
  }

  public static int getMax(int n1, int n2, int n3) {
    int max = Math.max(n1, n2);
    max = Math.max(max, n3);
    return max;
  }

  public static int getMin(int n1, int n2, int n3) {
    int min = Math.min(n1, n2);
    min = Math.min(min, n3);
    return min;
  }
}
```

JS算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 3) {
    let arr = lines.map((line) => line.split(" ").map((ele) => parseInt(ele)));

    console.log(calSquare(arr));

    lines.length = 0;
  }
});

function calSquare(arr) {
  let [rect1, rect2, rect3] = arr;

  let [x1, y1, w1, h1] = rect1;
  let [x2, y2, w2, h2] = rect2;
  let [x3, y3, w3, h3] = rect3;

  let width = Math.min(x1 + w1, x2 + w2, x3 + w3) - Math.max(x1, x2, x3);
  if (width <= 0) return 0;

  let height = Math.min(y1, y2, y3) - Math.max(y1 - h1, y2 - h2, y3 - h3);
  if (height <= 0) return 0;

  return width * height;
}
```

Python算法源码

```python
# 输入获取
x1, y1, w1, h1 = map(int, input().split())
x2, y2, w2, h2 = map(int, input().split())
x3, y3, w3, h3 = map(int, input().split())


# 算法入口
def getResult():
    wid = min(x1 + w1, x2 + w2, x3 + w3) - max(x1, x2, x3)
    if wid <= 0:
        return 0

    hei = min(y1, y2, y3) - max(y1 - h1, y2 - h2, y3 - h3)
    if hei <= 0:
        return 0

    return wid * hei


# 调用算法
print(getResult())
```