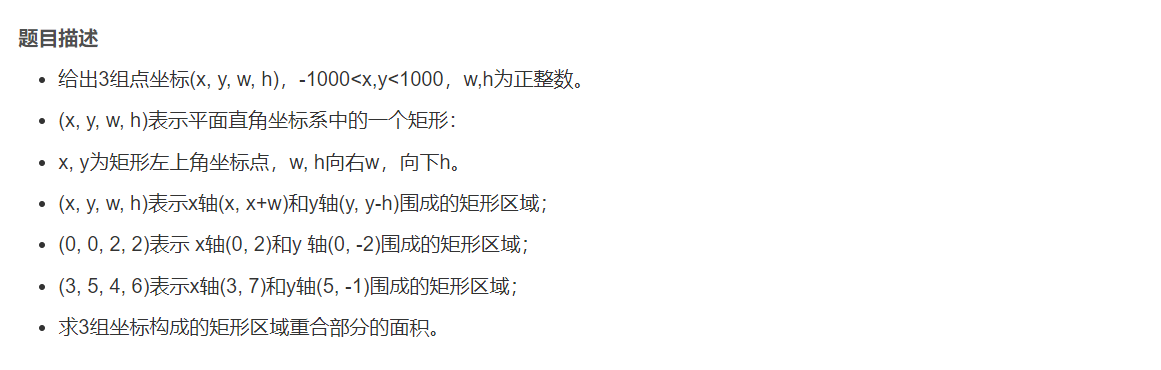
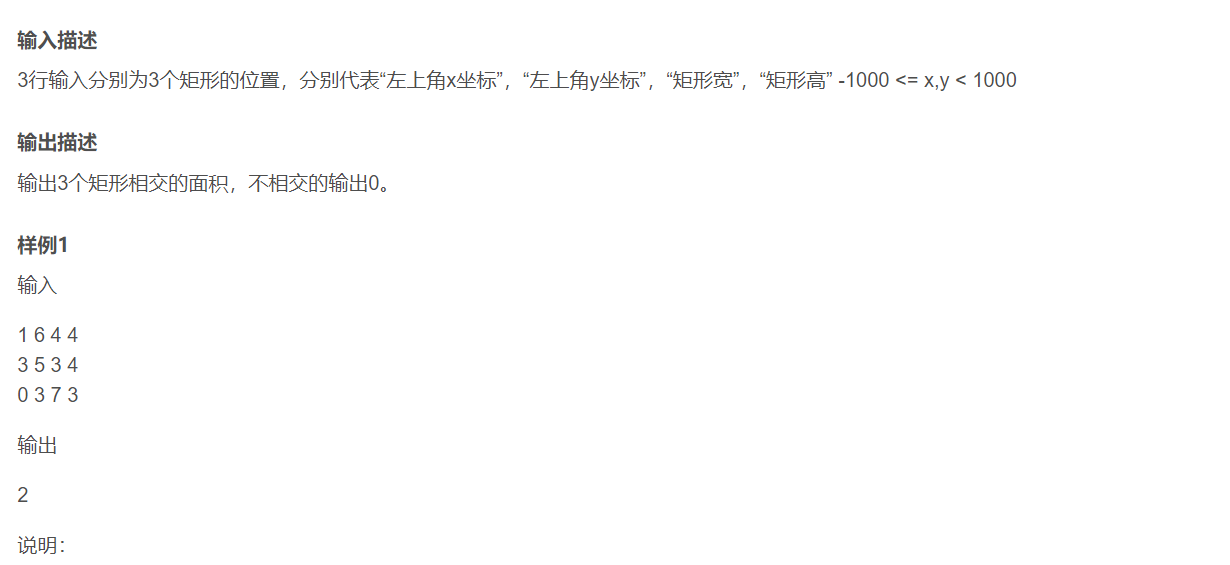
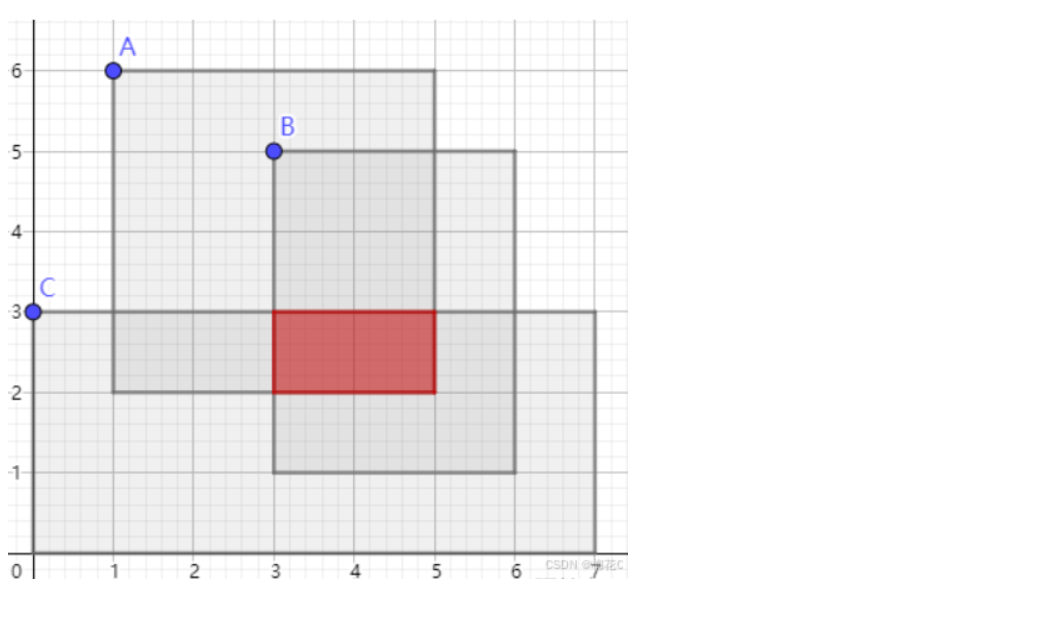
# **E卷-矩形相交的面积[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**









import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 定义一个3x4的二维数组

int[][] v = new int[3][4];

// 读取输入，填充二维数组

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 4; j++) {

v[i][j] = scanner.nextInt();

if (j == 2) {

// 第三列的值等于第一列的值加上原来的值

v[i][j] += v[i][0];

}

if (j == 3) {

// 第四列的值等于第二列的值减去原来的值

v[i][j] = v[i][1] - v[i][j];

}

}

}

// 计数，记录符合条件的点的数量

int ans = 0;

// 遍历所有可能的坐标点

for (int i = -2000; i <= 2000; i++) {

for (int j = -2000; j <= 2000; j++) {

int kk = 0; // 记录点 (i, j) 满足的区域数量

// 检查点 (i, j) 是否在每个矩形区域内

for (int k = 0; k < 3; k++) {

// 检查条件：在矩形的x范围内并且在矩形的y范围内

if (i >= v[k][0] && i < v[k][2] && j >= v[k][3] && j < v[k][1]) {

kk++;

}

}

// 如果点 (i, j) 在所有三个矩形区域内，则计数加1

if (kk == 3) ans++;

}

}

// 输出符合条件的点的数量

System.out.println(ans);

}

}



# 定义一个3x4的二维数组

v = [[0] \* 4 for \_ in range(3)]

# 读取输入，填充二维数组

for i in range(3):

v[i] = list(map(int, input().split()))

v[i][2] += v[i][0] # 第三列的值等于第一列的值加上原来的值

v[i][3] = v[i][1] - v[i][3] # 第四列的值等于第二列的值减去原来的值

# 计数，记录符合条件的点的数量

ans = 0

# 遍历所有可能的坐标点

for i in range(-2000, 2001):

for j in range(-2000, 2001):

kk = 0 # 记录点 (i, j) 满足的区域数量

# 检查点 (i, j) 是否在每个矩形区域内

for k in range(3):

if i >= v[k][0] and i < v[k][2] and j >= v[k][3] and j < v[k][1]:

kk += 1

# 如果点 (i, j) 在所有三个矩形区域内，则计数加1

if kk == 3:

ans += 1

# 输出符合条件的点的数量

print(ans)



#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

int main() {

// 定义一个3x4的二维向量并初始化

vector<vector<int>> v(3, vector<int>(4));

// 读取输入，填充二维向量

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 4; j++) {

cin >> v[i][j];

if (j == 2) {

// 第三列的值等于第一列的值加上原来的值

v[i][j] += v[i][0];

}

if (j == 3) {

// 第四列的值等于第二列的值减去原来的值

v[i][j] = v[i][1] - v[i][j];

}

}

}

// 计数，记录符合条件的点的数量

int ans = 0;

// 遍历所有可能的坐标点

for (int i = -2000; i <= 2000; i++) {

for (int j = -2000; j <= 2000; j++) {

int kk = 0; // 记录点 (i, j) 满足的区域数量

// 检查点 (i, j) 是否在每个矩形区域内

for (int k = 0; k < 3; k++) {

// 检查条件：在矩形的x范围内并且在矩形的y范围内

if (i >= v[k][0] && i < v[k][2] && j >= v[k][3] && j < v[k][1]) {

kk++;

}

}

// 如果点 (i, j) 在所有三个矩形区域内，则计数加1

if (kk == 3) ans++;

}

}

// 输出符合条件的点的数量

cout << ans << endl;

}



#include <stdio.h>

int main() {

// 定义一个3x4的二维数组并初始化为0

int v[3][4] = {0};

// 读取输入，填充二维数组

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 4; j++) {

scanf("%d", &v[i][j]);

if (j == 2) {

// 第三列的值等于第一列的值加上原来的值

v[i][j] += v[i][0];

}

if (j == 3) {

// 第四列的值等于第二列的值减去原来的值

v[i][j] = v[i][1] - v[i][j];

}

}

}

// 计数，记录符合条件的点的数量

int ans = 0;

// 遍历所有可能的坐标点

for (int i = -2000; i <= 2000; i++) {

for (int j = -2000; j <= 2000; j++) {

int kk = 0; // 记录点 (i, j) 满足的区域数量

// 检查点 (i, j) 是否在每个矩形区域内

for (int k = 0; k < 3; k++) {

// 检查条件：在矩形的x范围内并且在矩形的y范围内

if (i >= v[k][0] && i < v[k][2] && j >= v[k][3] && j < v[k][1]) {

kk++;

}

}

// 如果点 (i, j) 在所有三个矩形区域内，则计数加1

if (kk == 3) ans++;

}

}

// 输出符合条件的点的数量

printf("%d\n", ans);

return 0;

}



const rl = require("readline").createInterface({

input: process.stdin,

output: process.stdout,

});

var iter = rl[Symbol.asyncIterator]();

const readline = async () => (await iter.next()).value;

void (async function () {

let v = [];

for (let i = 0; i < 3; ++i) {

let x = (await readline()).split(" ").map(Number);

v.push(x);

}

for (let i = 0; i < 3; ++i) {

for (let j = 0; j < 4; ++j) {

if (j == 2) v[i][j] += v[i][0];

if (j == 3) v[i][j] = v[i][1] - v[i][j];

}

}

let ans = 0;

for (let i = -2000; i <= 2000; i++) {

for (let j = -2000; j <= 2000; j++) {

let kk = 0; // 记录点 (i, j) 满足的区域数量

for (let k = 0; k < 3; k++) {

if (

i >= v[k][0] &&

i < v[k][2] &&

j >= v[k][3] &&

j < v[k][1]

) {

kk++;

}

}

if (kk === 3) ans++;

}

}

console.log(ans);

})();



package main

import (

"fmt"

)

func main() {

// 定义一个3x4的二维数组

var v [3][4]int

// 读取输入，填充二维数组

for i := 0; i < 3; i++ {

for j := 0; j < 4; j++ {

fmt.Scan(&v[i][j])

if j == 2 {

// 第三列的值等于第一列的值加上原来的值

v[i][j] += v[i][0]

}

if j == 3 {

// 第四列的值等于第二列的值减去原来的值

v[i][j] = v[i][1] - v[i][j]

}

}

}

// 计数，记录符合条件的点的数量

ans := 0

// 遍历所有可能的坐标点

for i := -2000; i <= 2000; i++ {

for j := -2000; j <= 2000; j++ {

kk := 0 // 记录点 (i, j) 满足的区域数量

// 检查点 (i, j) 是否在每个矩形区域内

for k := 0; k < 3; k++ {

if i >= v[k][0] && i < v[k][2] && j >= v[k][3] && j < v[k][1] {

kk++

}

}

// 如果点 (i, j) 在所有三个矩形区域内，则计数加1

if kk == 3 {

ans++

}

}

}

// 输出符合条件的点的数量

fmt.Println(ans)

}