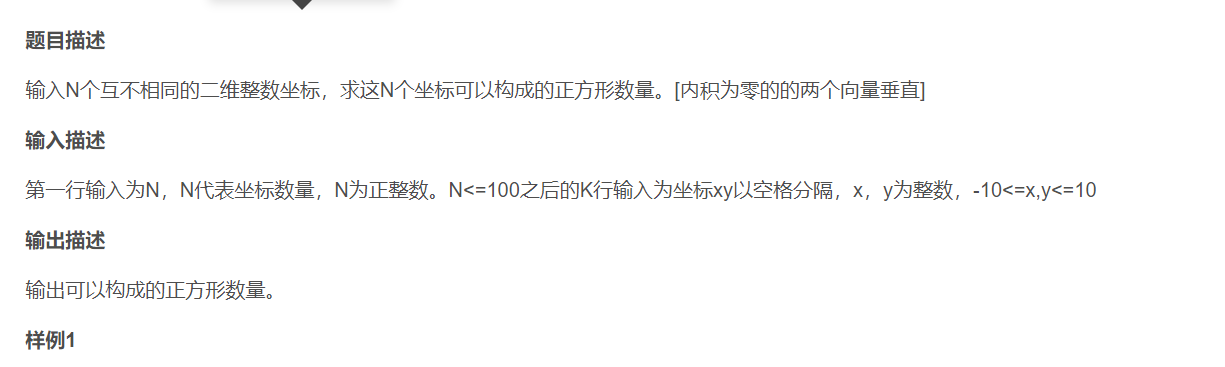
# **E卷-构成的正方形数量[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**









import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// 输入 n 表示点的数量

int n = scanner.nextInt();

// 创建一个列表存储所有输入的点（x, y）

List<int[]> a1 = new ArrayList<>();

// 读取 n 个点的坐标并添加到列表中

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

int x = scanner.nextInt();

int y = scanner.nextInt();

a1.add(new int[] {x, y});

}

// 创建一个列表存储所有可能的线段，由两个点组成

List<int[]> a2 = new ArrayList<>();

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

for (int j = i + 1; j < n; j++) {

int a = a1.get(i)[0]; // 第一个点的 x 坐标

int b = a1.get(i)[1]; // 第一个点的 y 坐标

int c = a1.get(j)[0]; // 第二个点的 x 坐标

int d = a1.get(j)[1]; // 第二个点的 y 坐标

a2.add(new int[] {a, b, c, d});

}

}

int cnt = 0; // 统计满足条件的矩形数量

n = a2.size(); // 线段的总数量

// 枚举所有线段对

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

for (int j = i + 1; j < n; j++) {

int[] rect1 = a2.get(i); // 第一条线段

int[] rect2 = a2.get(j); // 第二条线段

int a = rect1[0], b = rect1[1], c = rect1[2],

d = rect1[3]; // 线段1的两个端点

int e = rect2[0], f = rect2[1], g = rect2[2],

h = rect2[3]; // 线段2的两个端点

// 检查是否有共同端点，如果有则继续下一组线段对

if ((a == e && b == f) || (a == g && b == h) || (c == e && d == f) || (c == g &&

d == h)) {

continue;

}

// 检查对角线的中点是否相同

if ((a + c != e + g) || (b + d != f + h)) {

continue;

}

// 检查线段的长度平方是否相等

if ((d - b) \* (d - b) + (c - a) \* (c - a) != (h - f) \* (h - f) + (g - e) \*

(g - e)) {

continue;

}

// 检查线段之间是否垂直

if ((c - a) \* (g - e) + (d - b) \* (h - f) == 0) {

cnt++;

}

}

}

// 输出满足条件的矩形数量

System.out.println(cnt);

}

}



# 读取输入点的数量

n = int(input())

# 初始化一个列表存储所有点的坐标

a1 = []

# 读取每个点的坐标并存储到列表a1中

for \_ in range(n):

x = list(map(int, input().split())) # 读取每个点的x, y并转换为整数

a1.append([x[0], x[1]]) # 将点的坐标添加到列表a1中

# 初始化一个列表存储所有由两点形成的线段

a2 = []

for i in range(n):

for j in range(i + 1, n):

a, b = a1[i] # 第一个点的坐标

c, d = a1[j] # 第二个点的坐标

a2.append([a, b, c, d]) # 将由两个点形成的线段坐标添加到列表a2中

# 初始化计数器，统计满足条件的矩形数量

cnt = 0

n = len(a2) # 更新n为线段的总数量

# 遍历所有可能的线段对，检查是否构成矩形

for i in range(n):

for j in range(i + 1, n):

a, b, c, d = a2[i] # 第一个矩形对角线的两个端点坐标

e, f, g, h = a2[j] # 第二个矩形对角线的两个端点坐标

# 检查是否有共享端点的情况，如果有则跳过

if (a == e and b == f) or (a == g and b == h) or (c == e and d == f) or (c == g and d == h):

continue

# 检查对角线的中点是否相同

if (a + c != e + g) or (b + d != f + h):

continue

# 检查两条对角线的长度平方是否相等

if (d - b) \*\* 2 + (c - a) \*\* 2 != (h - f) \*\* 2 + (g - e) \*\* 2:

continue

# 检查两条对角线是否垂直

if (c - a) \* (g - e) + (d - b) \* (h - f) == 0:

cnt += 1 # 如果满足条件，计数器加1

# 输出满足条件的矩形数量

print(cnt)



#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

int main() {

// 输入点的数量

int n;

cin >> n;

// 用于存储点的坐标的向量

vector<pair<int, int>> a1;

// 读取每个点的坐标并存储到向量a1中

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

int x, y;

cin >> x >> y;

a1.push\_back({x, y});

}

// 用于存储所有可能的线段的向量

vector<vector<int>> a2;

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

for (int j = i + 1; j < n; j++) {

int a = a1[i].first; // 第一个点的x坐标

int b = a1[i].second; // 第一个点的y坐标

int c = a1[j].first; // 第二个点的x坐标

int d = a1[j].second; // 第二个点的y坐标

a2.push\_back({a, b, c, d}); // 将两个点形成的线段加入到a2中

}

}

// 初始化计数器，用于统计符合条件的矩形数量

int cnt = 0;

n = a2.size(); // 更新n为线段的总数量

// 遍历所有可能的线段对，检查是否构成矩形

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

for (int j = i + 1; j < n; j++) {

// 获取第一个矩形对角线的两个端点坐标

int a = a2[i][0], b = a2[i][1], c = a2[i][2], d = a2[i][3];

// 获取第二个矩形对角线的两个端点坐标

int e = a2[j][0], f = a2[j][1], g = a2[j][2], h = a2[j][3];

// 检查线段是否有共同端点，如果有则跳过

if ((a == e && b == f) || (a == g && b == h) || (c == e && d == f) || (c == g && d == h)) {

continue;

}

// 检查对角线的中点是否相同

if ((a + c != e + g) || (b + d != f + h)) {

continue;

}

// 检查两条对角线的长度平方是否相等

if ((int)pow(d - b, 2) + (int)pow(c - a, 2) != (int)pow(h - f, 2) + (int)pow(g - e, 2)) {

continue;

}

// 检查两条对角线是否垂直

if ((c - a) \* (g - e) + (d - b) \* (h - f) == 0) {

cnt++; // 如果满足条件，计数器加1

}

}

}

// 输出满足条件的矩形数量

cout << cnt << endl;

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main() {

int n;

// 读取点的数量

scanf("%d", &n);

// 动态分配内存来存储点的坐标

int \*\*a1 = (int \*\*)malloc(n \* sizeof(int \*));

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

a1[i] = (int \*)malloc(2 \* sizeof(int));

scanf("%d %d", &a1[i][0], &a1[i][1]); // 读取点的x和y坐标

}

// 计算所有可能的线段对的数量，并初始化存储这些线段的数组

int a2\_size = n \* (n - 1) / 2; // 计算组合数C(n, 2)

int \*\*a2 = (int \*\*)malloc(a2\_size \* sizeof(int \*));

int index = 0;

// 将所有可能的线段存储到a2中

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

for (int j = i + 1; j < n; j++) {

a2[index] = (int \*)malloc(4 \* sizeof(int));

a2[index][0] = a1[i][0]; // 线段起点x1

a2[index][1] = a1[i][1]; // 线段起点y1

a2[index][2] = a1[j][0]; // 线段终点x2

a2[index][3] = a1[j][1]; // 线段终点y2

index++;

}

}

// 初始化计数器，统计满足条件的矩形数量

int cnt = 0;

// 遍历所有可能的线段对，检查是否构成矩形

for (int i = 0; i < a2\_size; i++) {

for (int j = i + 1; j < a2\_size; j++) {

int a = a2[i][0], b = a2[i][1], c = a2[i][2], d = a2[i][3]; // 第一个矩形对角线的两个端点坐标

int e = a2[j][0], f = a2[j][1], g = a2[j][2], h = a2[j][3]; // 第二个矩形对角线的两个端点坐标

// 检查线段是否有共同端点，如果有则跳过

if ((a == e && b == f) || (a == g && b == h) || (c == e && d == f) || (c == g && d == h)) {

continue;

}

// 检查对角线的中点是否相同

if ((a + c != e + g) || (b + d != f + h)) {

continue;

}

// 检查两条对角线的长度平方是否相等

if ((d - b) \* (d - b) + (c - a) \* (c - a) != (h - f) \* (h - f) + (g - e) \* (g - e)) {

continue;

}

// 检查两条对角线是否垂直

if ((c - a) \* (g - e) + (d - b) \* (h - f) == 0) {

cnt++; // 如果满足条件，计数器加1

}

}

}

// 输出满足条件的矩形数量

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

// 释放动态分配的内存

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

free(a1[i]);

}

free(a1);

for (int i = 0; i < a2\_size; i++) {

free(a2[i]);

}

free(a2);

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 () {

// 读取输入的点数量

let n = parseInt(await readline());

// 存储所有点的坐标

let a1 = new Array();

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

// 读取每个点的坐标并转换为数字类型

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

// 将坐标存入数组a1

a1.push([x[0], x[1]]);

}

// 存储所有可能的线段

let a2 = new Array();

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

for (let j = i + 1; j < n; ++j) {

// 获取点的坐标

let [a, b] = a1[i];

let [c, d] = a1[j];

// 形成一条线段并存入a2数组

a2.push([a, b, c, d]);

}

}

// 初始化计数器，统计符合条件的矩形数量

let cnt = 0;

// 更新n为线段的总数量

n = a2.length;

// 遍历所有可能的线段对，检查是否构成矩形

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

for (let j = i + 1; j < n; ++j) {

let [a, b, c, d] = a2[i]; // 第一个矩形对角线的两个端点坐标

let [e, f, g, h] = a2[j]; // 第二个矩形对角线的两个端点坐标

// 检查线段是否有共同端点，如果有则跳过

if (a == e && b == f) continue;

if (a == g && b == h) continue;

if (c == e && d == f) continue;

if (c == g && d == h) continue;

// 检查对角线的中点是否相同

if (a + c != e + g || b + d != f + h) continue;

// 检查两条对角线的长度平方是否相等

if (

(d - b) \* (d - b) + (c - a) \* (c - a) !=

(h - f) \* (h - f) + (g - e) \* (g - e)

)

continue;

// 检查两条对角线是否垂直

if ((c - a) \* (g - e) + (d - b) \* (h - f) == 0) cnt += 1;

}

}

// 输出符合条件的矩形数量

console.log(cnt);

})();



package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

// 从输入读取一行并去除首尾空白字符

func readline(reader \*bufio.Reader) string {

line, \_ := reader.ReadString('\n')

return strings.TrimSpace(line)

}

func main() {

// 创建一个读取器以从标准输入中读取数据

reader := bufio.NewReader(os.Stdin)

// 读取点的数量 n

n, \_ := strconv.Atoi(readline(reader))

// 创建一个大小为 n 的数组 a1 来存储点的坐标

a1 := make([][2]int, n)

// 依次读取每个点的坐标并存储到 a1 中

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

line := readline(reader) // 读取一行输入

split := strings.Split(line, " ") // 按空格分割字符串

x, \_ := strconv.Atoi(split[0]) // 将第一个分割部分转换为整数 x

y, \_ := strconv.Atoi(split[1]) // 将第二个分割部分转换为整数 y

a1[i] = [2]int{x, y} // 存储坐标点

}

// 定义一个动态数组 a2 用于存储所有可能的线段对

var a2 [][4]int

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

for j := i + 1; j < n; j++ {

a, b := a1[i][0], a1[i][1] // 线段的起点坐标

c, d := a1[j][0], a1[j][1] // 线段的终点坐标

a2 = append(a2, [4]int{a, b, c, d}) // 将这个线段添加到 a2 中

}

}

// 初始化计数器，统计满足条件的矩形数量

cnt := 0

// 更新 n 为 a2 数组的长度

n = len(a2)

// 遍历所有可能的线段对，检查是否构成矩形

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

for j := i + 1; j < n; j++ {

a, b, c, d := a2[i][0], a2[i][1], a2[i][2], a2[i][3] // 第一个矩形对角线的两个端点坐标

e, f, g, h := a2[j][0], a2[j][1], a2[j][2], a2[j][3] // 第二个矩形对角线的两个端点坐标

// 检查线段是否有共同端点，如果有则跳过

if (a == e && b == f) || (a == g && b == h) || (c == e && d == f) || (c == g && d == h) {

continue

}

// 检查对角线的中点是否相同

if (a+c != e+g) || (b+d != f+h) {

continue

}

// 检查两条对角线的长度的平方是否相等

if (d-b)\*(d-b)+(c-a)\*(c-a) != (h-f)\*(h-f)+(g-e)\*(g-e) {

continue

}

// 检查两条对角线是否垂直

if (c-a)\*(g-e)+(d-b)\*(h-f) == 0 {

cnt++ // 如果满足条件，计数器加 1

}

}

}

// 输出满足条件的矩形数量

fmt.Println(cnt)

}