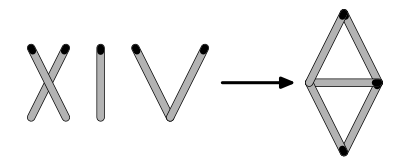
H. Спички детям не игрушка!

|  |  |
| --- | --- |
| Ограничение времени | 3 секунды |
| Ограничение памяти | 256Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Вася любит решать головоломки со спичками. Чаще всего они формулируется следующим образом: дано изображение *A*, составленное из спичек; переложите в нем минимальное количество спичек так, чтобы получилось изображение *B*.

Например, из номера текущего командного чемпионата школьников Санкт-Петербурга по программированию, можно получить ромб с диагональю, переложив всего три спички.



Головоломки, которые решает Вася, всегда имеют решение. Это значит, что набор спичек, используемый в изображении *A*, совпадает с набором спичек, используемым в изображении *B*. Кроме того, в одном изображении никогда не встречаются две спички, у которых есть общий участок ненулевой длины (то есть спички могут пересекаться, но не могут накладываться друг на друга).

Вася устал решать головоломки вручную, и теперь он просит вас написать, программу, которая будет решать головоломки за него. Программа будет получать описания изображений *A* и *B* и должна найти минимальное количество спичек, которые надо переложить в изображении *A*, чтобы полученная картинка получалась из *B* параллельным переносом.

Формат ввода

В первой строке входного файла содержится целое число *n* — количество спичек в каждом из изображений (*1 ≤ n ≤ 1000*).

В следующих *n* строках записаны координаты концов спичек на изображении *A*. Спичка номер *i* описывается целыми числами *x1i*, *y1i*, *x2i*, *y2i* — координатами ее концов. Следующие *n* строк содержат описание изображения *B* в таком же формате. Набор длин этих спичек совпадает с набором длин спичек с изображения *A*.

Все координаты по абсолютной величине не превосходят *104*. Все спички имеют ненулевую длину, то есть *x1i ≠ x2i* или *y1i ≠ y2i*.

Формат вывода

Выведите в выходной файл минимальное количество спичек, которые следует переложить, чтобы изображение *A* совпало с изображением *B*, с точностью до параллельного переноса.

Пример 1

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 5  0 0 1 2  1 0 0 2  2 0 2 2  4 0 3 2  4 0 5 2  9 -1 10 1  10 1 9 3  8 1 10 1  8 1 9 -1  8 1 9 3 | 3 |

Пример 2

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 1  3 4 7 9  -1 3 3 8 | 0 |

Пример 3

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 1  -4 5 2 -3  -12 4 -2 4 | 1 |

using System;

using System.Collections.Generic;

using System.IO;

public struct Point

{

public int dX;

public int dY;

public Point(int x, int y)

{

dX = x;

dY = y;

}

public void Print()

{

Console.WriteLine("\t\tPoint coordinates: X = {0}, Y = {1}", dX, dY);

}

public override bool Equals(object? obj)

{

return obj is Point ? (Point)obj == this: false;

}

public static bool operator == (Point p1, Point p2)

{

return p1.dX == p2.dX && p1.dY == p2.dY;

}

public static bool operator != (Point p1, Point p2)

{

return p1.dX != p2.dX || p1.dY != p2.dY;

}

public override int GetHashCode()

{

return dX.GetHashCode() \* dY.GetHashCode();

}

}

public struct PointWhisLen

{

public int dX;

public int dY;

public int X;

public int Y;

public PointWhisLen(int dx, int dy, int x, int y)

{

dX = dx;

dY = dy;

X = x;

Y = y;

}

public void Print1()

{

Console.WriteLine("Point dif: X = {0}, Y = {1}", dX, dY);

}

}

class Program

{

static void Main()

{

StreamReader reader = new StreamReader("input.txt");

int N = Convert.ToInt32(reader.ReadLine());

Dictionary<Point, HashSet<PointWhisLen>> start = new Dictionary<Point, HashSet<PointWhisLen>>();

for (int i = 0; i < N; i++)

{

string[] input = reader.ReadLine().Split();

int x1 = Convert.ToInt32(input[0]);

int y1 = Convert.ToInt32(input[1]);

int x2 = Convert.ToInt32(input[2]);

int y2 = Convert.ToInt32(input[3]);

int dX = Math.Abs(x2 - x1);

if ((x1 <= x2 && y1 >= y2) || (x1>=x2 && y2>=y1)) dX = -dX;

int dY = Math.Abs(y2 - y1);

int lenX = x1 + x2, lenY = y1 + y2;

Point point = new Point(dX, dY);

PointWhisLen pointWhisLen = new PointWhisLen(dX, dY, lenX, lenY);

if (start.ContainsKey(point)) start[point].Add(pointWhisLen);

else start.Add(point, new HashSet<PointWhisLen>(){pointWhisLen});

}

Dictionary<Point, int> end = new Dictionary<Point, int>();

for (int i = 0; i < N; i++)

{

string[] input = reader.ReadLine().Split();

int x1 = Convert.ToInt32(input[0]);

int y1 = Convert.ToInt32(input[1]);

int x2 = Convert.ToInt32(input[2]);

int y2 = Convert.ToInt32(input[3]);

int dX = Math.Abs(x2 - x1);

if ((x1 <= x2 && y1 >= y2) || (x1>=x2 && y2>=y1)) dX = -dX;

int dY = Math.Abs(y2 - y1);

int lenX = x1 + x2, lenY = y1 + y2;

Point this\_point = new Point(dX, dY);

if (start.ContainsKey(this\_point))

{

foreach (var spichka in start[this\_point])

{

Point this\_len = new Point(spichka.X - lenX, spichka.Y - lenY);

if (end.ContainsKey(this\_len))

{

end[this\_len]++;

}

else

{

end.Add(this\_len, 1);

}

}

}

}

int Max = 0;

foreach (var VARIABLE in end)

{

if (VARIABLE.Value > Max) Max = VARIABLE.Value;

}

Console.WriteLine(N - Max);

}

}