G. Построить квадрат

|  |  |
| --- | --- |
| Ограничение времени | 2 секунды |
| Ограничение памяти | 256Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Задано множество, состоящее из *N* различных точек на плоскости. Координаты всех точек — целые числа. Определите, какое минимальное количество точек нужно добавить во множество, чтобы нашлось четыре точки, лежащие в вершинах квадрата.

Формат ввода

В первой строке вводится число *N* (*1 ≤ N ≤ 2000*) — количество точек.

В следующих *N* строках вводится по два числа *xi*, *yi* (*-108 ≤ xi, yi ≤ 108*) — координаты точек.

Формат вывода

В первой строке выведите число *K* — минимальное количество точек, которые нужно добавить во множество.

В следующих *K* строках выведите координаты добавленных точек *xi*, *yi* через пробел. Координаты должны быть целыми и не превышать *109* по модулю.

Если решений несколько — выведите любое из них.

Пример 1

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 2  0 1  1 0 | 2  0 0  1 1 |

Пример 2

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 3  0 2  2 0  2 2 | 1  0 0 |

Пример 3

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 4  -1 1  1 1  -1 -1  1 -1 | 0 |

using System;

using System.Collections.Generic;

using System.Linq;

public struct Point

{

public int X;

public int Y;

public Point(int x, int y)

{

X = x;

Y = y;

}

public void Print()

{

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

}

}

class Program

{

static void Main()

{

// HashSet<Point> allPoints = new HashSet<Point>();

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

// Dictionary<int, HashSet<int>> aXis = new Dictionary<int, HashSet<int>>();

Dictionary<int, HashSet<int>> aYis = new Dictionary<int, HashSet<int>>();

Point[] points = new Point[N];

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

{

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

int x = int.Parse(input[0]);

int y = int.Parse(input[1]);

// if (aXis.ContainsKey(x))

// {

// aXis[x].Add(y);

// }

// else aXis.Add(x, new HashSet<int>() {y});

points[i].X = x;

points[i].Y = y;

if (aYis.ContainsKey(y))

{

aYis[y].Add(x);

}

else aYis.Add(y, new HashSet<int>() {x});

}

if (N == 1)

{

Console.WriteLine("3");

foreach (var key in aYis.Keys)

{

Console.WriteLine(key + " " + (aYis[key].Max() + 1));

Console.WriteLine((key + 1) + " " + aYis[key].Max());

Console.WriteLine((key + 1) + " " + (aYis[key].Max() + 1));

}

return;

}

else

{

int result = 0;

int who = 0;

Point[] result\_points = new Point[4];

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

{

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

{

int max = 2;

Point[] current\_points = new Point[4];

current\_points[0] = points[i];

current\_points[1] = points[j];

current\_points[2].Y = current\_points[3].Y = 1000000000;

int dX = Math.Abs(current\_points[0].X - current\_points[1].X);

int dY = Math.Abs(current\_points[0].Y - current\_points[1].Y);

if (current\_points[0].X == current\_points[1].X && current\_points[0].Y == current\_points[1].Y) continue;

if (aYis.ContainsKey(current\_points[0].Y + dX))

{

if ((current\_points[0].Y <= current\_points[1].Y && current\_points[0].X <= current\_points[1].X) || current\_points[0].Y >= current\_points[1].Y && current\_points[0].X >= current\_points[1].X) //current\_points[0].Y <= current\_points[1].Y && current\_points[0].X <= current\_points[1].X

{

// Console.WriteLine(1);

if (aYis[current\_points[0].Y + dX].Contains(current\_points[0].X - dY))

{

max++;

current\_points[2].X = current\_points[0].X - dY;

current\_points[2].Y = current\_points[0].Y + dX;

}

if (aYis.ContainsKey(current\_points[1].Y + dX))

{

if (aYis[current\_points[1].Y + dX].Contains(current\_points[1].X - dY))

{

max++;

current\_points[3].X = current\_points[1].X - dY;

current\_points[3].Y = current\_points[1].Y + dX;

}

}

if (max > result)

{

result\_points = current\_points;

result = max;

who = 1;

if (result == 4)

{

Console.WriteLine("0");

return;

}

}

}

else if(current\_points[0].Y >= current\_points[1].Y && current\_points[0].X <= current\_points[1].X)

{

// Console.WriteLine(2);

if (aYis[current\_points[0].Y + dX].Contains(current\_points[0].X + dY))

{

max++;

current\_points[2].X = current\_points[0].X + dY;

current\_points[2].Y = current\_points[0].Y + dX;

}

if (aYis.ContainsKey(current\_points[1].Y + dX))

{

if (aYis[current\_points[1].Y + dX].Contains(current\_points[1].X + dY))

{

max++;

current\_points[3].X = current\_points[1].X + dY;

current\_points[3].Y = current\_points[1].Y + dX;

}

}

if (max > result)

{

result\_points = current\_points;

result = max;

who = 2;

if (result == 4)

{

Console.WriteLine("0");

return;

}

}

}

}

else if (aYis.ContainsKey(current\_points[0].Y - dX))

{

if (current\_points[0].X <= current\_points[1].X && current\_points[0].Y <= current\_points[1].Y)

{

// Console.WriteLine(3);

if (aYis[current\_points[0].Y - dX].Contains(current\_points[0].X + dY))

{

max++;

current\_points[2].X = current\_points[0].X + dY;

current\_points[2].Y = current\_points[0].Y - dX;

}

if (aYis.ContainsKey(current\_points[1].Y - dX))

{

if (aYis[current\_points[1].Y - dX].Contains(current\_points[1].X + dY))

{

max++;

current\_points[3].X = current\_points[1].X + dY;

current\_points[3].Y = current\_points[1].Y - dX;

}

}

if (max > result)

{

result\_points = current\_points;

result = max;

who = 3;

if (result == 4)

{

Console.WriteLine("0");

return;

}

}

}

//current\_points[2].X = current\_points[2].Y = current\_points[3].X = current\_points[3].Y = 0;

else if (current\_points[0].X <= current\_points[1].X && current\_points[0].Y >= current\_points[1].Y)

{

// Console.WriteLine(4);

if (aYis[current\_points[0].Y - dX].Contains(current\_points[0].X - dY))

{

max++;

current\_points[2].X = current\_points[0].X - dY;

current\_points[2].Y = current\_points[0].Y - dX;

}

if (aYis.ContainsKey(current\_points[1].Y - dX))

{

if (aYis[current\_points[1].Y - dX].Contains(current\_points[1].X - dY))

{

max++;

current\_points[3].X = current\_points[1].X - dY;

current\_points[3].Y = current\_points[1].Y - dX;

}

}

if (max > result)

{

result\_points = current\_points;

result = max;

who = 4;

if (result == 4)

{

Console.WriteLine("0");

return;

}

}

}

}

}

}

// Console.WriteLine(who);

if (who == 0)

{

result\_points[0] = points[0];

result\_points[1] = points[1];

result = 2;

if ((result\_points[0].Y <= result\_points[1].Y && result\_points[0].X < result\_points[1].X) ||

(result\_points[0].X < result\_points[1].X && result\_points[0].Y <= result\_points[1].Y))

{

who = 1;

}

else who = 2;

}

// Console.WriteLine(who);

// Console.WriteLine(result);

// foreach (var pop in result\_points)

// {

// pop.Print();

// }

if (result == 2)

{

int dX = Math.Abs(result\_points[0].X - result\_points[1].X);

int dY = Math.Abs(result\_points[0].Y - result\_points[1].Y);

Console.WriteLine(2);

if (who == 1 || who == 3)

{

Console.WriteLine((result\_points[0].X - dY) + " " + (result\_points[0].Y + dX));

Console.WriteLine((result\_points[1].X - dY) + " " + (result\_points[1].Y + dX));

}

else

{

Console.WriteLine((result\_points[0].X + dY) + " " + (result\_points[0].Y + dX));

Console.WriteLine((result\_points[1].X + dY) + " " + (result\_points[1].Y + dX));

}

return;

}

if (result == 3)

{

int dX = Math.Abs(result\_points[0].X - result\_points[1].X);

int dY = Math.Abs(result\_points[0].Y - result\_points[1].Y);

Console.WriteLine(1);

if (who == 1)

{

if (result\_points[2].Y == 1000000000)Console.WriteLine((result\_points[0].X - dY) + " " + (result\_points[0].Y + dX));

else Console.WriteLine((result\_points[1].X - dY) + " " + (result\_points[1].Y + dX));

return;

}

if (who == 2)

{

if (result\_points[2].Y == 1000000000)Console.WriteLine((result\_points[0].X + dY) + " " + (result\_points[0].Y + dX));

else Console.WriteLine((result\_points[1].X + dY) + " " + (result\_points[1].Y + dX));

return;

}

if (who == 3)

{

if (result\_points[2].Y == 1000000000)Console.WriteLine((result\_points[0].X + dY) + " " + (result\_points[0].Y - dX));

else Console.WriteLine((result\_points[1].X + dY) + " " + (result\_points[1].Y - dX));

return;

}

if (who == 4)

{

if (result\_points[2].Y == 1000000000)Console.WriteLine((result\_points[0].X - dY) + " " + (result\_points[0].Y - dX));

else Console.WriteLine((result\_points[1].X - dY) + " " + (result\_points[1].Y - dX));

return;

}

}

// Console.WriteLine(result);

// foreach (var pop in result\_points)

// {

// pop.Print();

// }

}

}

}