I. Пираты Баренцева моря

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Вася играет в настольную игру «Пираты Баренцева моря», которая посвящена морским битвам. Игровое поле представляет собой квадрат из N×N клеток, на котором расположено N кораблей (каждый корабль занимает одну клетку).

Вася решил воспользоваться линейной тактикой, для этого ему необходимо выстроить все N кораблей в одном столбце. За один ход можно передвинуть один корабль в одну из четырёх соседних по стороне клеток. Номер столбца, в котором будут выстроены корабли, не важен. Определите минимальное количество ходов, необходимых для построения кораблей в одном столбце. В начале и процессе игры никакие два корабля не могут находиться в одной клетке.

Формат ввода

В первой строке входных данных задаётся число N (1≤N≤100).

В каждой из следующих N строк задаются координаты корабля: сначала номер строки, затем номер столбца (нумерация начинается с единицы).

Формат вывода

Выведите одно число — минимальное количество ходов, необходимое для построения.

Пример

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

Примечания

В примере необходимо выстроить корабли в столбце номер 2. Для этого необходимо переставить корабль из клетки 3 3 в клетку 3 2 за один ход, а корабль из клетки 1 1 в клетку 2 2 за два хода. Существуют и другие варианты перестановки кораблей, однако ни в одном из них нет меньше трёх ходов.

using System;

public struct Point

{

public int X;

public int Y;

public Point(int x, int y)

{

X = x;

Y = y;

}

}

class Program

{

static void Main()

{

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

int result = 0;

int[] xs = new int[N];

int[] ys = new int[N];

int[,] field1 = new int[N, N];

int[,] field2 = new int[N, N];

int[,] field3 = new int[N, N];

int[] stolbA = new int[N];

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

{

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

xs[i] = int.Parse(input[0]);

ys[i] = int.Parse(input[1]);

stolbA[i] = ys[i];

field1[(xs[i] - 1), (ys[i] - 1)] = 1;

field2[(xs[i] - 1), (ys[i] - 1)] = 1;

field3[(xs[i] - 1), (ys[i] - 1)] = 1;

}

Array.Sort(stolbA);

int length = stolbA.Length;

// Вычисляем индекс центрального элемента

int middle = length / 2;

// Если длина массива четная, вычисляем медиану как среднее двух центральных элементов

int median;

if (length % 2 == 0)

{

median = (stolbA[middle - 1] + stolbA[middle]) / 2;

}

else

{

// Если длина нечетная, медиана - это центральный элемент

median = stolbA[middle];

}

// double average = (double)stolbA / N;

// stolbA = (int)Math.Ceiling(average);

Point[] points = new Point[N];

int count = 0;

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

{

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

{

if (field1[r, j] == 1)

{

int g = count++;

points[g].X = r + 1;

points[g].Y = j + 1;

}

}

}

int find\_way(int x, int y, int stolb, int[,] field)

{

int res = 0;

if (y <= stolb)

{

if (y == stolb) res = 0;

else

{

res = stolb - y;

int min = N + 1;

bool find = false;

for (int j = 0; j < N && !find; j++)

{

bool empty = true;

for (int u = y < N ? y : N - 1; u < stolb && empty; u++)

{

if (field[j, u] == 1) empty = false;

}

if (empty)

{

min = Math.Max(x, j + 1) - Math.Min(x, j + 1);

field[j, stolb - 1] = 1;

find = true;

}

}

res += min;

field[x - 1, y - 1] = 0;

}

}

else

{

res = y - stolb;

int min = N + 1;

bool find = false;

for (int j = 0; j < N && !find; j++)

{

bool empty = true;

for (int u = y - 2 >= 0 ? y - 2 : 0; u >= stolb - 1 && empty; u--)

{

if (field[j, u] == 1) empty = false;

}

if (empty)

{

min = Math.Max(x, j + 1) - Math.Min(x, j + 1);

field[j, stolb - 1] = 1;

find = true;

}

}

res += min;

field[x - 1, y - 1] = 0;

}

return res;

}

if (N != 1)

{

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

{

result += find\_way(points[i].X, points[i].Y, median, field1);

}

if(median-1 >= 1)

{

int res = 0;

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

{

res += find\_way(points[i].X, points[i].Y, median - 1, field2);

}

result = Math.Min(res, result);

}

if (median + 1 <= N)

{

int res = 0;

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

{

res += find\_way(points[i].X, points[i].Y, median + 1, field3);

}

result = Math.Min(res, result);

}

}

Console.WriteLine(result);

}

}