camaro@nightsova.ru

Тренировки по алгоритмам 5.0 от Яндекса — Занятие 2 (Линейный поиск)

15 мар 2024, 04:40:33
старт: 6 мар 2024, 20:30:00
финиш: 20 мар 2024, 18:00:00
до финиша: 5д. 13ч.
начало: 6 мар 2024, 20:30:00
конец: 20 мар 2024, 18:00:00
длительность: 13д. 21ч.

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 $N \times N$ 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 ($1 \le N \le 100$).
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	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.

Язык Python 3.9 (PyPy 7.3.11)

Набрать здесь | Отправить файл

```
from collections import deque
n = int(input())

field = [] # игровое поле
columns = [] #колонны с кораблями
side_queue = deque() # двусторон очередь для распределения ходов (сверху-снизу-сверху и т.д.)

answer = 0

for i in range(1, (n+1)):   # заполняем двустороннюю очередь
    side_queue.append(i)

for f in range(n+2):      # заполняем поле (n+2)*(n+2), границы поля +, пустые яч-ки 0
    if f == 0 or f == n+1:
        field.append(['+']*(n+2))
    else:
        field.append(['+'] + [0]*n + ['+'])

for _ in range(n):      # наносим на поле корабли как #
    x, y = map(int, input().split())
    columns.append(y)
    field[x][y] = '#'

columns.sort()
column = int((columns[n//2] + columns[~(n//2)])/2)   # медиана, для подсчета оптимальной центральной колонны

side = 'left'
while side_queue:           # двигаем корабли к нужной колонне
    if side == 'left':
        i = side_queue.popleft()
        course = 0          # курс к ближайшей границе(верхней-нижней)
        side = 'right'
    else:
        i = side_queue.pop()
        course = n
        side = 'left'
    for j in range(column-1, 0, -1):  #обраб поля слева от колонны, справа-налево
```

Отправить

Предыдущая | Следующая