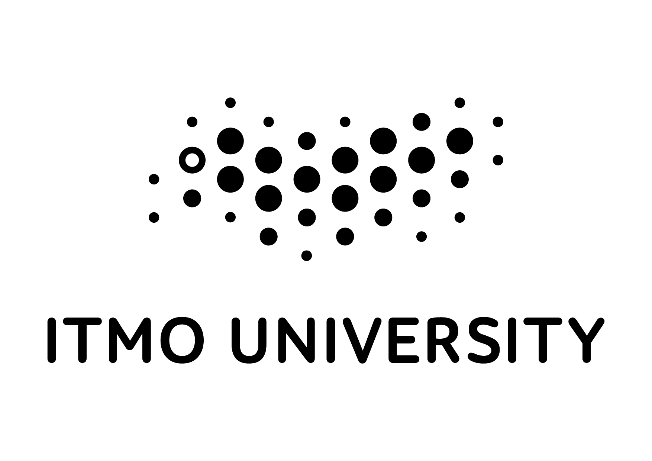
Санкт-Петербургский национальный исследовательский университет ИТМО

Факультет систем управления и робототехники



Алгоритмы и структуры данных.

Отчет №3.

Выполнил студент гр. R32362

Лаптев Максим Сергеевич

Санкт-Петербург 2023

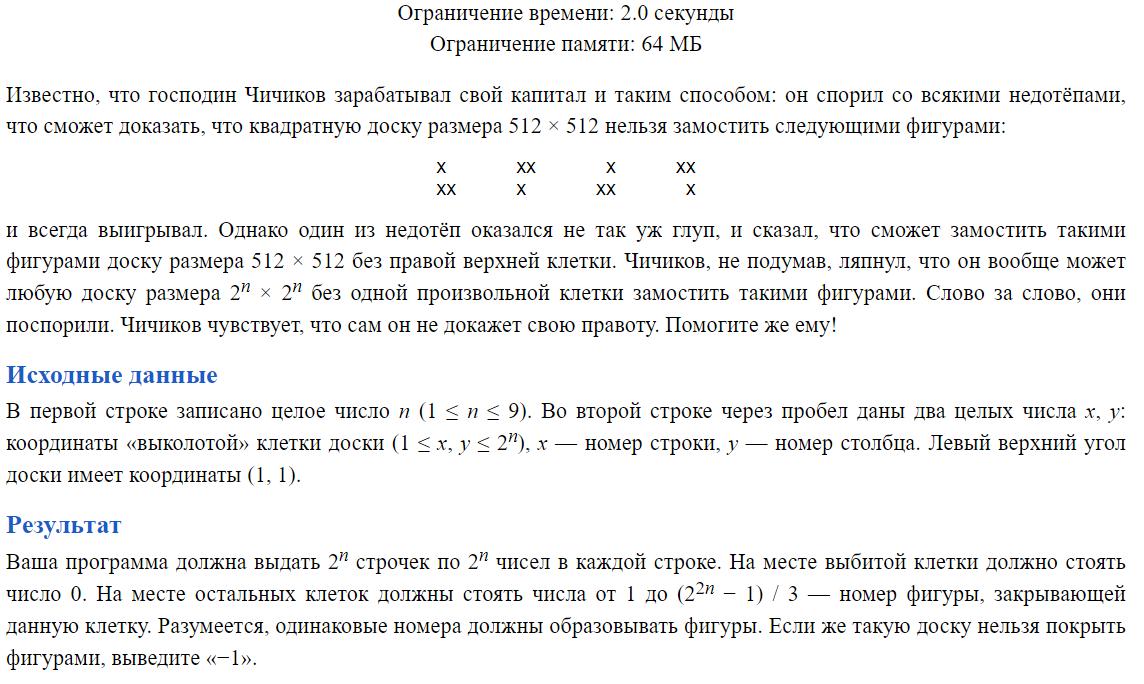
**Содержание**

1. Цель
2. Задача №1401 – Игроки
3. Задача №1444 – Накормить элефпотама
4. Задача №1494 – Монобильярд
5. Вывод

**Цель**

Решить данные задачи, написав код, работа которого удовлетворяет условиям

**№1401 «Игроки»**



Изображение выглядит как стол

Автоматически созданное описание

Исходный код:

def f(x, y, size, start, dx, dy):  
 global maxk  
 if size == 2:  
 for i in range(size):  
 for j in range(size):  
 if s[dx + i][dy + j] == -1:  
 s[dx + i][dy + j] = maxk  
 maxk += 1  
 return 0  
 if x < size // 2 and y < size // 2: # 1 quarter  
 s[dx + size//2][dy + size//2] = s[dx + size//2][dy + size//2 - 1] = s[dx + size//2 - 1][dy + size//2] = maxk  
 maxk += 1  
 f(x, y, size//2, (0, 0), dx, dy) # 1 quarter  
 f(size//2 - 1, 0, size//2, (0, size//2), dx, dy + size//2) # 2 quarter  
 f(0, size//2 - 1, size//2, (size//2, 0), dx + size//2, dy) # 3 quarter  
 f(0, 0, size//2, (size//2, size//2), dx + size//2, dy + size//2) # 4 quarter  
 elif x < size // 2 and y >= size // 2: # 2 quarter  
 s[dx + size//2][dy + size//2] = s[dx + size//2][dy + size//2 - 1] = s[dx + size//2 - 1][dy + size//2 - 1] = maxk  
 maxk += 1  
 f(x, y - size//2, size//2, (0, 0), dx, dy + size//2) # 2 quarter  
 f(size//2 - 1, size//2 - 1, size//2, (0, 0), dx, dy) # 1 quarter  
 f(0, size//2 - 1, size//2, (size//2, 0), dx + size//2, dy) # 3 quarter  
 f(0, 0, size//2, (size//2, size//2), dx + size//2, dy + size//2) # 4 quarter  
 elif x >= size // 2 and y < size // 2: # 3 quarter  
 s[dx + size//2][dy + size//2] = s[dx + size//2 - 1][dy + size//2] = s[dx + size//2 - 1][dy + size//2 - 1] = maxk  
 maxk += 1  
 f(x - size//2, y, size//2, (0, 0), dx + size//2, dy)   
 f(size//2 - 1, size//2 - 1, size//2, (0, 0), dx, dy) # 1 quarter  
 f(size//2 - 1, 0, size//2, (0, size//2), dx, dy + size//2) # 2 quarter  
 f(0, 0, size//2, (size//2, size//2), dx + size//2, dy + size//2) # 4 quarter  
 elif x >= size // 2 and y >= size // 2: # 4 quarter  
 s[dx + size//2 - 1][dy + size//2] = s[dx + size//2][dy + size//2 - 1] = s[dx + size//2 - 1][dy + size//2 - 1] = maxk  
 maxk += 1  
 f(x - size//2, y - size//2, size//2, (0, 0), dx + size//2, dy + size//2)   
 f(size//2 - 1, size//2 - 1, size//2, (0, 0), dx, dy) # 1 quarter  
 f(size//2 - 1, 0, size//2, (0, size//2), dx, dy + size//2) # 2 quarter  
 f(0, size//2 - 1, size//2, (size//2, 0), dx + size//2, dy) # 3 quarter  
   
   
  
n = 2 \*\* int(input())  
s = [[-1] \* n for i in range(n)]  
x, y = map(int, input().split())  
maxk = 1  
  
s[x - 1][y - 1] = 0  
f(x - 1, y - 1, n, (0, 0), 0, 0)  
for el in s:  
 print("\t".join([str(elem) for elem in el]))

Главная идея алгоритма – деление квадрата на четверти и прохождение рекурсией по каждой части. Дело в том, что любой квадрат (2^n – 1) можно разбить на 3 ячейки. Осталось выяснить как – если занятая клетка в первой четверти, то в оставшихся трех необходимо заполнить их углы так, чтобы они образовывали уголок (.:) и так далее для любого квадрата.

Язык программирования: python

Результат



**№1444** **«Накормить элефпотама»**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

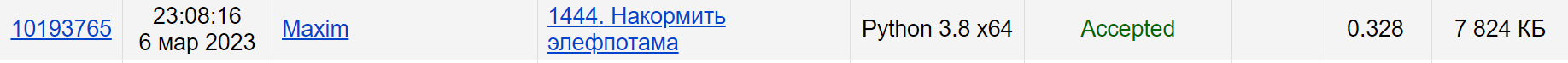
Исходный код:

from math import atan2, pi  
  
n = int(input())  
x1, y1 = map(int, input().split())  
angles = []  
  
for i in range(n - 1):  
 x, y = map(int, input().split())  
 angle = atan2((y - y1) , (x - x1))  
 dist = ((x - x1)\*\*2 + (y-y1)\*\*2)\*\*0.5  
 if (y - y1) < 0:  
 angle += 2 \* pi  
 angles.append((angle, dist, i + 2))  
  
print(n)  
print(1)  
angles.sort(key = lambda x: (x[0], x[1]))  
  
start = 0  
max\_angle = angles[0][0] - angles[n-2][0] + 2\*pi  
for i in range(n - 2):  
 if (angles[i+1][0] - angles[i][0] > max\_angle):  
 max\_angle = angles[i+1][0] - angles[i][0]  
 start = i + 1  
for i in range(start, n - 1):  
 print(angles[i][2])  
for i in range(start):  
 print(angles[i][2])

За точку отсчета начала координат возьмем 1 точку, с которой происходит поедание тыкв. Теперь пройдемся по каждой следующей тыкве и подсчитаем для нее угол и расстояние до точки отсчета. Отсортируем (сначала по углу, потом по расстоянию) полученный список. Выведем их, учитывая особенность, что точки могут противоположно с другой стороны.

Язык программирования: python

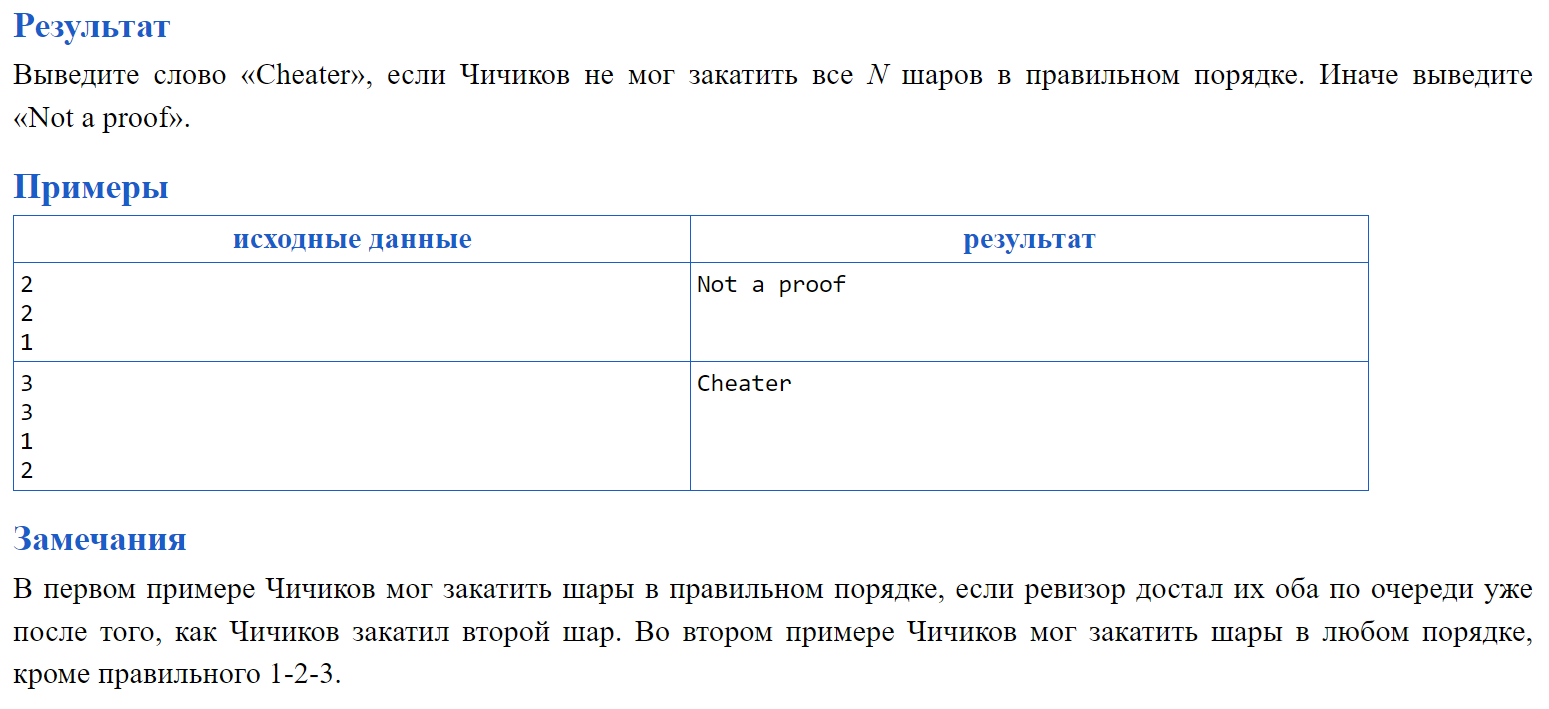
Результат



**№1494 «Монобильярд»**

Изображение выглядит как текст

Автоматически созданное описание



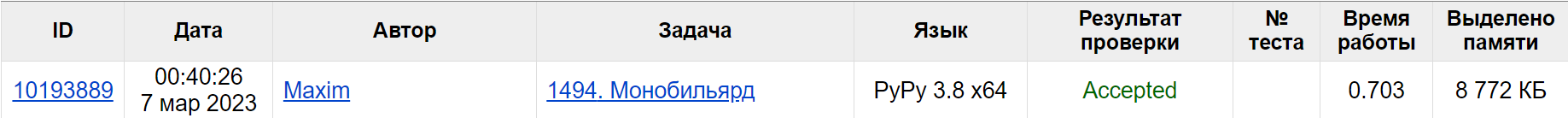
Исходный код:

n = int(input())  
stack = []  
was = 0  
for i in range(n):  
 x = int(input())  
 if stack == []:   
 for i in range(was + 1, x):  
 stack.append(i)  
 else:  
 if stack[-1] == x:  
 stack.pop(-1)  
 elif stack[-1] < x:  
 for i in range(was + 1, x):  
 stack.append(i)  
 else:  
 print("Cheater")  
 exit()  
 was = max(was, x)  
if stack == []: print("Not a proof")

Моделируем закатывание шаров, используя стек. Если первый шар, который достал Ревизор, например, 5, то значит все предыдущие шары (1, 2, 3 и 4) уже в лунке. В случае, когда смоделировать поведение не получается, в частности когда последний элемент стека больше доставаемого выводим ‘Cheater’. Иначе ‘Not a proof’

Язык программирования: python

Результат:



**Вывод**

В результате проделанной работы были успешно решены 3 задачи, используя язык программирования python. Работа программ выдает правильный ответ, а также удовлетворяет условиям задачи (проходит по времени и памяти).