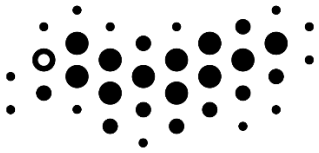
Санкт-Петербургский национальный
исследовательский университет ИТМО

Факультет систем управления и робототехники



ITMO UNIVERSITY

Алгоритмы и структуры данных.
Отчет №3.

Выполнил студент гр. R32362

Лаптев Максим Сергеевич

Санкт-Петербург 2023

Содержание

Цель

Решить данные задачи, написав код, работа которого удовлетворяет условиям

№1401 «Игроки»

Ограничение времени: 2.0 секунды
Ограничение памяти: 64 МБ

Известно, что господин Чичиков зарабатывал свой капитал и таким способом: он спорил со всякими недотёпами, что сможет доказать, что квадратную доску размера 512 × 512 нельзя замостить следующими фигурами:

```
x      xx      x      xx
xx     x       xx      x
```

и всегда выигрывал. Однако один из недотёп оказался не так уж глуп, и сказал, что сможет замостить такими фигурами доску размера 512 × 512 без правой верхней клетки. Чичиков, не подумав, ляпнул, что он вообще может любую доску размера $2^n \times 2^n$ без одной произвольной клетки замостить такими фигурами. Слово за слово, они поспорили. Чичиков чувствует, что сам он не докажет свою правоту. Помогите же ему!

Исходные данные

В первой строке записано целое число n ($1 \le n \le 9$). Во второй строке через пробел даны два целых числа x, y: координаты «выколотой» клетки доски ($1 \le x, y \le 2^n$), x — номер строки, y — номер столбца. Левый верхний угол доски имеет координаты (1, 1).

Результат

Ваша программа должна выдать 2^n строчек по 2^n чисел в каждой строке. На месте выбитой клетки должно стоять число 0. На месте остальных клеток должны стоять числа от 1 до $(2^{2n} - 1) / 3$ — номер фигуры, закрывающей данную клетку. Разумеется, одинаковые номера должны образовывать фигуры. Если же такую доску нельзя покрыть фигурами, выведите «−1».

Пример

исходные данные	результат
2 1 1	0 1 3 3 1 1 4 3 2 4 4 5 2 2 5 5

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

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10193627	21:23:34 6 мар 2023	Maxim	1401. Игроки	Python 3.8 x64	Accepted		0.468	5 340 КБ

№1444 «Накормить элефпотама»

Ограничение времени: 0.5 секунды
Ограничение памяти: 64 МБ

Гарри Поттер сдаёт экзамен по предмету «Уход за магическими существами». Его задание — накормить карликового элефпотама. Гарри помнит, что элефпотамы отличаются прямолинейностью и невозмутимостью. Они настолько прямолинейны, что ходят строго по прямой, и настолько невозмутимы, что заставить их идти можно, только если привлечь его внимание к чему-нибудь действительно вкусному. И главное, наткнувшись на цепочку своих собственных следов, элефпотам впадает в ступор и отказывается идти куда-либо. По словам Хагрида, элефпотамы обычно возвращаются домой, идя в обратную сторону по своим собственным следам. Поэтому они никогда не пересекают их, иначе могут заблудиться. Увидев свои следы, элефпотам детально вспоминает все свои перемещения от выхода из дома (поэтому-то они и ходят только по прямой и лишний раз не меняют направление — так легче запоминать). По этой информации элефпотам вычисляет, в какой стороне расположена его нора, после чего поворачивается и идет прямо к ней. Эти вычисления занимают у элефпотама некоторое (довольно большое) время. А то, что некоторые невежды принимают за ступор, на самом деле есть проявление выдающихся вычислительных способностей этого чудесного, хотя и медленно соображающего животного!

Любимое лакомство элефпотамов — слоновьи тыквы, именно они и растут на лужайке, где Гарри должен сдавать экзамен. Перед началом испытания Хагрид притащит животное к одной из тыкв. Скормив элефпотаму очередную тыкву, Гарри может направить его в сторону любой оставшейся тыквы. Чтобы сдать экзамен, надо провести элефпотама по лужайке так, чтобы тот съел как можно больше тыкв до того, как наткнется на свои следы.

Исходные данные

В первой строке входа находится число N ($3 \le N \le 30000$) — количество тыкв на лужайке. Тыквы пронумерованы от 1 до N, причем номер один присвоен той тыкве, у которой будет стоять элефпотам в начале экзамена. В следующих N строках даны координаты всех тыкв по порядку. Все координаты — целые числа от −1000 до 1000. Известно, что положения всех тыкв различны, и не существует прямой, проходящей сразу через все тыквы.

Результат

В первой строке выхода вы должны вывести K — максимальное количество тыкв, которое может съесть элефпотам. Далее по одному числу в строке выведите K чисел — номера тыкв в порядке их обхода. Первым в этой последовательности всегда должно быть число 1.

Пример

исходные данные	результат
4 0 0 10 10 0 10 10 0	4 1 3 2 4

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

10193765	23:08:16 6 мар 2023	Maxim	1444. Накормить элефпотама	Python 3.8 x64	Accepted		0.328	7 824 КБ

№1494 «Монобильярд»

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Стол для монобильярда, установленный в игровом доме уездного города N, оказался очень прибыльным вложением. До того, как в городе появился небезызвестный господин Чичиков. Раз за разом он выигрывал, и хозяин, подсчитывая убытки, понимал, что дело тут нечисто. Однако уличить подлеца в жульничестве не удавалось до прибытия в город N ревизора из Петербурга.

Правила игры в монобильярд очень просты: нужно последовательно закатить в единственную лузу шары с номерами 1, 2, …, N (именно в этом порядке). Пока господин Чичиков играл, ревизор несколько раз подходил к столу и забирал из лузы последний закатившийся туда шар. В конце концов, оказалось, что Чичиков закатил в лузу все шары, а ревизор все шары достал и обследовал. Аферист утверждал, что закатил шары в правильном порядке. Хозяин понял, что это его шанс: ревизор должен помнить, в каком порядке он доставал шары. Однако так ли легко будет доказать жульничество?

Исходные данные

В первой строке записано целое число N — количество бильярдных шаров ($1 \le N \le 100000$). В следующих N строках даны номера этих шаров в том порядке, в котором ревизор забирал их из лузы.

Результат

Выведите слово «Cheater», если Чичиков не мог закатить все N шаров в правильном порядке. Иначе выведите «Not a proof».

Примеры

исходные данные	результат
2 2 1	Not a proof
3 3 1 2	Cheater

Замечания

В первом примере Чичиков мог закатить шары в правильном порядке, если ревизор достал их оба по очереди уже после того, как Чичиков закатил второй шар. Во втором примере Чичиков мог закатить шары в любом порядке, кроме правильного 1-2-3.

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:

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10193889	00:40:26 7 мар 2023	Maxim	1494. Монобильярд	PyPy 3.8 x64	Accepted		0.703	8 772 КБ

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).