Федеральное государственное автономное образовательное учреждение
высшего образования
**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**
Факультет систем управления и робототехники

Лабораторная работа №4
Задачи №1726, 1067, 1494

Студент: Сайфуллин Д.Р.
Поток: АиСД R23 1.3
Преподаватель: Тропченко А.А.

Санкт-Петербург
2025 г.

Задача №1726. Кто ходит в гости...

Программный комитет школьных соревнований по программированию, проходящих в УрГУ — многочисленная, весёлая и дружная команда. Дружная настолько, что общения в университете им явно не хватает, поэтому они часто ходят друг к другу в гости. Все ребята в программном комитете очень спортивные и ходят только пешком.

Однажды хранитель традиций олимпиадного движения УрГУ подумал, что на пешие прогулки от дома к дому члены программного комитета тратят слишком много времени, которое могли бы вместо этого потратить на придумывание и подготовку задач. Чтобы доказать это, он решил посчитать, какое расстояние в среднем преодолевают члены комитета, когда ходят друг к другу в гости. Хранитель традиций достал карту Екатеринбурга, нашёл на ней дома всех членов программного комитета и выписал их координаты. Но координат оказалось так много, что хранитель не смог справиться с этой задачей самостоятельно и попросил вас помочь ему.

Город Екатеринбург представляет собой прямоугольник со сторонами, ориентированными по сторонам света. Все улицы города идут строго с запада на восток или с севера на юг, проходя через весь город от края до края. Дома всех членов программного комитета расположены строго на пересечении каких-то двух перпендикулярных улиц. Известно, что все члены комитета ходят только по улицам, поскольку идти по тротуару гораздо приятнее, чем по дворовым тропинкам. И, конечно, при переходе от дома к дому они всегда выбирают кратчайший путь. Программный комитет очень дружный, и все его члены ходят в гости ко всем одинаково часто.

Исходные данные:

> Первая строка содержит целое число n — количество членов программного комитета ($2 \le n \le 10^5$). В i-й из следующих n строк через пробел записаны целые числа x_i, y_i — координаты дома i-го члена программного комитета ($1 \le x_i\, y_i \le 10^6$).

Результат:

> Выведите среднее расстояние, которое проходит член программного комитета от своего дома до дома своего товарища, округлённое вниз до целых.

Рабочий код

```python
def main():
    n = int(input())
    x_coords = []
    y_coords = []
    for _ in range(n):
        x, y = map(int, input().split())
        x_coords.append(x)
        y_coords.append(y)

    x_coords.sort()
    y_coords.sort()

    total = 0

    for i in range(1, n):
        dx = x_coords[i] - x_coords[i - 1]
        dy = y_coords[i] - y_coords[i - 1]
        total += (dx + dy) * i * (n - i) * 2
```

```
19
20      average_distance = total // (n * (n - 1))
21      print(average_distance)
22
23  if __name__ == "__main__":
24      main()
```

Объяснение алгоритма

Алгоритм сначала сортирует координаты точек по осям x и y. Затем он вычисляет сумму расстояний между каждой парой точек по оси x и по оси y, суммируя вклад каждой точки в общую сумму на основе ее позиции в отсортированном списке. В итоге, он выводит среднее значение суммы расстояний между всеми парами точек.

Задача №1067. Структура папок

Хакер Билл случайно потерял всю информацию с жесткого диска своего компьютера, и у него нет резервных копий его содержимого. Но он сожалеет не о потере самих файлов, а о потере очень понятной и удобной структуры папок, которую он создавал и сохранял в течение многих лет работы.

К счастью, у Билла есть несколько копий списков папок с его жесткого диска. С помощью этих списков он смог восстановить полные пути к некоторым папкам (например, «WINNT\SYSTEM32\CERTSRV\CERTCO 1\X86»). Он поместил их все в файл, записав каждый найденный путь в отдельную строку.

Напишите программу, которая восстановит структуру папок Билла и выведет ее в виде отформатированного дерева.

Исходные данные:

> Первая строка содержит целое число N – количество различных путей к папкам ($1 \leq N \leq 500$). Далее следуют N строк с путями к папкам. Каждый путь занимает одну строку и не содержит пробелов, в том числе, начальных и конечных. Длина каждого пути не превышает 80 символов. Каждый путь встречается в списке один раз и состоит из нескольких имен папок, разделенных обратной косой чертой («\»).
>
> Имя каждой папки состоит из 1-8 заглавных букв, цифр или специальных символов из следующего списка: восклицательный знак, решетка, знак доллара, знак процента, амперсанд, апостроф, открывающаяся и закрывающаяся скобки, знак дефиса, собаки, циркумфлекс, подчеркивание, гравис, открывающаяся и закрывающаяся фигурная скобка и тильда (`«!\#\$\%\&'()-@^_‘\{\}~»`).

Результат:

> Выведите отформатированное дерево папок. Каждое имя папки должно быть выведено в отдельной строке, перед ним должно стоять несколько пробелов, указывающих на глубину этой папки в иерархии. Подпапки должны быть перечислены в лексикографическом порядке непосредственно после их родительской папки; перед их именем должно стоять на один пробел больше, чем перед именем их родительской папки. Папки верхнего уровня выводятся без пробелов и также должны быть перечислены в лексикографическом порядке.

Рабочий код

```
class Dir:
def __init__(self, name: str):
    self.name = name
    self.subdirs = []

def add_subdir(self, subdirs):
    node = self
    for curr in subdirs:
        if curr not in [i.name for i in node.subdirs]:
            curr_dir = Dir(curr)
            node.subdirs.append(curr_dir)
        else:
            curr_dir = [i for i in node.subdirs if i.name == curr][0]
        node = curr_dir

```

```
    def print_tree(self, depth=0):
        for subdir in sorted(self.subdirs, key=lambda x: x.name):
            print(' ' * depth + subdir.name)
            subdir.print_tree(depth + 1)


if __name__ == '__main__':
    n = int(input())
    root = Dir('')
    for _ in range(n):
        path = input().split('\\')
        root.add_subdir(path)
    root.print_tree()
```

Объяснение алгоритма

Алгоритм использует классы, которые хранят информацию о дочерних папках. Это позволяет создавать вложенные структуры, имитирующие файловую систему.

Задача №1494. Монобильярд

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 $1, 2, \ldots, N$ (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 ($1 \leq N \leq 10^5$). 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».

Рабочий код

```python
def main():
    n = int(input())
    seq = [int(input()) for i in range(n)]

    stack = []
    next_ball = 1
    cheater = False

    for ball in seq:
        if cheater:
            break

        if ball >= next_ball:
            for b in range(next_ball, ball):
                stack.append(b)
            next_ball = ball + 1
        else:
            if stack and stack[-1] == ball:
                stack.pop()
            else:
                cheater = True

    print("Cheater" if cheater else "Not a proof")


if __name__ == "__main__":
    main()
```

Объяснение алгоритма

Алгоритм предполагает использования стека для отслеживания ожидаемых чисел. Если следующее число больше вершины стека, оно добавляет недостающие числа в стек. Если число совпадает с вершиной стека, оно удаляется. Если число меньше и не совпадает с вершиной, алгоритм определяет, что было совершено «читерство».

Статус проверки

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10939343	16:34:18 19 апр 2025	Dinislam	1494. Монобильярд	PyPy 3.10 x64	Accepted		0.531	7 916 КБ
10939325	16:20:13 19 апр 2025	Dinislam	1067. Структура папок	PyPy 3.10 x64	Accepted		0.171	10 348 КБ
10939293	15:36:06 19 апр 2025	Dinislam	1726. Кто ходит в гости...	PyPy 3.10 x64	Accepted		0.546	8 576 КБ

Рис. 1: Результат проверки