**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра “Математическая кибернетика и информационные технологии”

**Отчет по курсовой работе**

по дисциплине «Структуры и алгоритмы обработки данных»

Вариант №15.

Выполнил: студент группы БВТ2004

Казыханов Рамиль Ильясович

Проверил:

Мкртчян Грач Маратович

Москва, 2022

# Задачи

## Задача №1

Мария участвует в велосипедной гонке.

Гоночная трасса проходит по берегу озера Люцерн, точно повторяя его контур. Как известно, берег озера состоит только из прямых участков, направленных на север, юг, восток или запад.

Введём систему координат, направив ось Ox с запада на восток, а ось Oy — с юга на север. В качестве позиции старта гонки выбирается самая южная (а если таких несколько, то самая западная из них) точка трассы. Участники начинают гонку, двигаясь на север. На всех прямых участках трассы участники едут в одном из четырёх направлений (север, юг, восток или запад) и меняют направление движения только на поворотах между прямыми участками. Участники, конечно, никогда не поворачивают назад, то есть не меняют направление движения с северного на южное или с восточного на западное (или наоборот).

Мария ещё юна, поэтому она чувствует себя уверенно не на всех поворотах. А именно, Мария чувствует себя неуверенно, если при неудачном или несвоевременном повороте она попадёт в воду. Иными словами, Мария считает поворот опасным, если при его игнорировании она сразу попадает в воду.

Помогите Маше подготовиться к соревнованию — определите количество опасных поворотов на трассе.

*Входные данные:* В первой строке входных данных находится целое число n (4 ≤ n ≤ 1000) — количество прямых участков трассы.В следующих (n + 1)-й строке находятся пары целых чисел (xi, yi) ( - 10 000 ≤ xi, yi ≤ 10 000). Первая из этих точек является позицией старта. i-й прямой участок трассы начинается в точке (xi, yi) и заканчивается в точке (xi + 1, yi + 1).

*Гарантируется, что*:

* первый прямой участок направлен на север;
* самая южная (а если таких несколько, то самая западная из них) точка трассы — первая точка;
* последняя точка совпадает с первой (то есть с позицией старта);
* любая пара прямых участков трассы не имеет общих точек (за исключением соседних, имеющих ровно одну общую точку);
* никакая пара точек (кроме первой и последней) не совпадает;
* никакие два соседних прямых участка не направлены в одну и ту же сторону или в противоположные стороны.

*Выходные данные:* выведите единственное целое число — количество опасных поворотов на трассе.

## Задача №2

В организации Alternative Cake Manufacturing (ACM) работает n сотрудников. Сейчас там проводится очень важное голосование, и все ведущие мировые СМИ пытаются спрогнозировать его результат.

Так получилось, что каждый из сотрудников имеет ярко выраженные политические взгляды и принадлежит либо к фракции депубликанцев, либо к фракции ремократов, и каждой фракции выгоден свой результат голосования. Процедура голосования устроена крайне необычно:

Каждый из n сотрудников высказывается по очереди, начиная с сотрудника номер 1 и заканчивая сотрудником номер n. При этом если к моменту, когда сотруднику пора высказаться, его уже лишили права голоса, то он просто пропускает свой ход (и уже никак не участвует в голосовании в последствии).

Высказывание сотрудника состоит в том, чтобы лишить права голоса любого другого сотрудника или ничего не сделать. Лишать права голоса можно как тех кто уже высказался, так и тех, кому только предстоит высказаться. Лишение права голоса действует в течение всего процесса голосования.

Когда все сотрудники выскажутся, процедура повторяется: снова по очереди высказываются все от сотрудника 1 до сотрудника n, кто ещё не лишился права голоса.

Повторения процедуры происходят до тех пор, пока не останется только один сотрудник с правом голоса, то он в одиночку решает исход голосования. Разумеется, он принимает решение в пользу своей фракции.

Зная порядок голосования и то, что все сотрудники действуют оптимально (и знают порядок голосования и кто принадлежит какой фракции), вы должны предсказать, какая фракция одержит верх.

*Входные данные:* в первой строке записано целое число n (1 ≤ n ≤ 200 000) — количество сотрудников.Далее записана строка из n символов, i-й из которых равен «D», если i-й человек поддерживает депубликанцев, либо «R», если i-й человек поддерживает ремократов.

*Выходные данные:* выведите «D», если на голосовании будет принято решение, устраивающее фракцию депубликанцев, и «R», если ремократов.

## Задача №3

Алёна работает в аэропорту Мегаполиса, составляя расписание отправления рейсов. Сегодня должны отправиться n рейсов, i-й из которых должен вылететь в минуту i.

Как вы помните из задачи вчерашнего дня, аэропорт Мегаполиса является основным транспортным узлом в Метрополии, а в высоконагруженных аэропортах нередко случаются накладки. Ровно так и произошло сегодня — из-за технических неполадок в течение первых k минут ни один рейс не смог вылететь из аэропорта.

Теперь все запланированные n рейсов должны вылететь в различные минуты от (k + 1)-й до (k + n)-й включительно. Однако рейсы не обязаны вылетать в исходном порядке — Алёна может составить любое расписание отправления рейсов. При этом должно быть выполнено важное условие: ни один рейс не может вылететь раньше своего изначально запланированного времени вылета.

Алёна знает, что задержка вылета i-го рейса на одну минуту стоит аэропорту ci бурлей. Помогите Алёне определить, в каком порядке следует вылетать рейсам, чтобы суммарная стоимость задержки оказалась минимально возможной.

*Входные данные:* в первой строке входных данных находятся два целых числа n и k (1 ≤ k ≤ n ≤ 300 000), где n — количество рейсов, а k — количество минут, в течение которых рейсы не вылетали из аэропорта.Во второй строке находятся n чисел c1, c2, ..., cn (1 ≤ ci ≤ 107), где ci — стоимость задержки i-го рейса на одну минуту.

*Выходные данные:* в первой строке выведите минимальную возможную суммарную стоимость задержки всех рейсов.Во второй строке выведите n различных целых чисел t1, t2, ..., tn (k + 1 ≤ ti ≤ k + n), где ti означает время вылета i-го рейса. Если расписаний с минимальной стоимостью несколько, разрешается вывести любое из них.

## Задача №4

У Ивана есть массив, состоящий из n элементов. Каждый из элементов — целое число от 1 до n.

Совсем недавно Иван узнал о перестановках и их лексикографическом порядке. Теперь он хочет изменить значения минимального количества элементов в своём массиве таким образом, чтобы его массив стал перестановкой (то есть каждое из целых чисел от 1 до n встречалось в его массиве ровно по одному разу). Если существует много способов сделать заданный массив перестановкой за минимальное количество изменений, то среди них Иван хочет выбрать такой, что полученная перестановка лексикографически минимальна.

Таким образом, в первую очередь Иван хочет минимизировать количество измененных элементов, а во вторую — минимизировать лексикографически полученную перестановку.

Для того, чтобы определить, какая из двух перестановок лексикографически меньше, они сначала сравниваются по первому элементу. При их равенстве — по второму, и так далее. Среди двух перестановок x и y лексикографически меньше будет x, если xi < yi, где i — первый индекс, в котором перестановки x и y различаются.

Определите, как будет выглядеть массив Ивана, если он осуществит задуманное.

*Входные данные*: в первой строке следует целое число n (2 ≤ n ≤ 200 000) — количество элементов в массиве Ивана. Во второй строке следует последовательность целых чисел a1, a2, ..., an (1 ≤ ai ≤ n) — описание массива Ивана.

*Выходные данные*: в первую строку выведите q — минимальное количество элементов, которые нужно изменить в массиве Ивана для того, чтобы массив стал перестановкой. Во вторую строку выведите лексикографически минимальную перестановку, которая получается с помощью q изменений.

## Задача №5

Вы — амбициозный король, который хочет быть Императором Действительных чисел. Но перед этим вам нужно стать Императором Целых чисел.

Рассмотрим числовую ось. Столица вашей империи изначально находится в точке 0. На прямой есть 𝑛 незахваченных королевств в позициях 0<𝑥1<𝑥2<…<𝑥𝑛. Вы хотите захватить все эти королевства.

Вы можете делать два вида действий:

Вы можете переместить свою столицу (пусть ее текущая координата 𝑐1) в любое захваченное королевство (пусть его позиция 𝑐2), стоимость такого действия 𝑎⋅|𝑐1−𝑐2|.

Вы можете из текущей столицы (пусть ее текущая координата 𝑐1) захватить королевство (пусть его позиция 𝑐2), стоимость такого действия 𝑏⋅|𝑐1−𝑐2|. Вы не можете захватывать королевство, если между вашей столицей и целью есть другие незахваченные королевства.

Обратите внимание, что вы не можете расположить столицу в точке, где нет королевства. Другими словами, в любой момент времени ваша столица может быть только в точке 0 или в 𝑥1,𝑥2,…,𝑥𝑛. Также обратите внимание, что захват королевства не изменяет положение вашей столицы.

Выведите минимальную суммарную стоимость захвата всех королевств. Ваша столица может оказаться в итоге в любой точке.

*Входные данные*: первая строка содержит одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных. Далее следуют описания наборов. Первая строка каждого набора содержит 3 целых числа 𝑛, 𝑎 и 𝑏 (1≤𝑛≤2⋅105; 1≤𝑎,𝑏≤105). Вторая строка каждого набора содержит 𝑛 целых чисел 𝑥1,𝑥2,…,𝑥𝑛 (1≤𝑥1<𝑥2<…<𝑥𝑛≤108). Гарантируется, что сумма значений 𝑛 по всем наборам входных данных не превосходит 2⋅105.

*Выходные данные*: для каждого набора входных данных выведите одно число: минимальную стоимость захвата всех королевств.

## Задача №6

Малек нашел карту сокровищ. В поисках сокровищ он нашел закрытую дверь. На двери была записана строка s из символов '(', ')' и '#'. Ниже была записана инструкция, как открыть дверь. Покорпев над инструкцией, Малек смог её расшифровать, в результате чего он узнал, что надо заменить каждый символ '#' одним или более символом ')' так, чтобы итоговая строка стала красивой.

Ещё ниже было написано, что строка называется красивой, если для каждого i (1 ≤ i ≤ |s|) среди первых i символов строки s символов ')' не больше, чем символов '(', а также общее число символов '(' в строке равняется общему числу символов ')'.

Помогите Малеку открыть дверь, указав ему, сколькими символами ')' надо заменить каждый символ '#'.

*Входные данные*: в первой строке ввода записана строка s (1 ≤ |s| ≤ 105). Каждый символ этой строки — это один из символов '(', ')', '#'. Гарантируется, что s содержит как минимум один символ '#'.

*Выходные данные*: если невозможно заменить символы '#' так, чтобы получилась красивая строка, выведите -1. В противном случае для каждого символа '#' выведите в отдельной строке положительное число — количество символов ')', которыми надо заменить этот символ. Если возможных ответов несколько, разрешается вывести любой.

## Задача №7

Скобочной последовательностью называется строка, состоящая только из символов «(» и «)».

Правильной скобочной последовательностью называется скобочная последовательность, которую можно преобразовать в корректное арифметическое выражение путем вставок между ее символами символов «1» и «+». Например, скобочные последовательности «()()», «(())» — правильные (полученные выражения: «(1)+(1)», «((1+1)+1)»), а «)(» и «(» — нет.

Вам задано 𝑛 скобочных последовательностей 𝑠1,𝑠2,…,𝑠𝑛. Найдите количество пар 𝑖,𝑗(1≤𝑖,𝑗≤𝑛), таких что скобочная последовательность 𝑠𝑖+𝑠𝑗 является правильной скобочной последовательностью. Операция + означает конкатенацию, т. е. "()(" + ")()" = "()()()".

Если 𝑠𝑖+𝑠𝑗 и 𝑠𝑗+𝑠𝑖 являются правильными скобочными последовательностями и 𝑖≠𝑗, в ответе должны учитываться обе пары (𝑖,𝑗) и (𝑗,𝑖). Также если строка 𝑠𝑖+𝑠𝑖 является правильной скобочной последовательностью, пара (𝑖,𝑖) должна учитываться в ответе.

*Входные данные*: в первой строке задано число 𝑛(1≤𝑛≤3⋅105) — количество скобочных последовательностей. В следующих 𝑛 строках содержатся скобочные последовательности — непустые строки, состоящие только из символов «(» и «)». Сумма длин всех скобочных последовательностей не превосходит 3⋅105.

*Выходные данные*: в единственной строке выведите целое число — количество пар 𝑖,𝑗(1≤𝑖,𝑗≤𝑛), таких, что скобочная последовательность 𝑠𝑖+𝑠𝑗 является правильной скобочной последовательностью.

## Задача №8

Для обработки запросов к некоторому сетевому сервису используются n серверов. Известен текущий план распределения запросов между серверами: i-ый сервер должен обработать mi запросов.

Было решено выполнить балансировку нагрузки, переназначая задания между серверами. Более формально, пусть ma — количество запросов, которые обрабатывает наиболее загруженный сервер, а mb — количество запросов, которые обрабатывает наименее загруженный сервер. Нагрузка будет считаться сбалансированной, если разность ma - mb будет минимально возможной.

За одну секунду можно переназначить один запрос. То есть, за одну секунду можно выбрать любую пару серверов, снять ровно один запрос с одного сервера и назначить этот же запрос другому серверу.

Перед вами стоит задача найти минимальное количество секунд, по истечении которых нагрузка серверов будет сбалансирована.

*Входные данные*: в первой строке содержится целое положительное число n (1 ≤ n ≤ 105) — количество серверов. Во второй строке содержится последовательность неотрицательных целых чисел m1, m2, ..., mn (0 ≤ mi ≤ 2·104), где mi — количество запросов, изначально предназначенных для обработки i-м сервером.

*Выходные данные*: выведите одно целое неотрицательное число — минимальное количество секунд, по истечении которых нагрузка серверов будет сбалансирована.

## Задача №9

Вам в руки попала шкатулка с кодовым замком. На замке есть дисплей, который показывает n цифр. Рядом с дисплеем расположены две кнопки, каждая из которых изменяет цифры на дисплее. Вы быстро догадались, что первая кнопка прибавляет 1 ко всем цифрам (при этом все цифры 9 превращаются в цифры 0), а вторая — сдвигает все цифры на дисплее на одну позицию вправо (при этом последняя цифра становится первой). Например, если дисплей в данный момент показывает число 579, то если нажать первую кнопку, дисплей станет показывать 680, а если после этого нажать вторую кнопку, дисплей станет показывать 068.

Вам известно, что замок откроется, если на дисплее будет отображаться минимальное число из всех, которые можно получить нажатием на кнопки в каком-то порядке (каждая кнопка может быть нажать произвольное число раз). При сравнении чисел ведущие нули игнорируются. Теперь вам требуется определить, что это за число.

*Входные данные*: в первой строке записано одно целое число n (1 ≤ n ≤ 1000) — количество цифр на дисплее. Во второй строке записано n цифр — исходное состояние дисплея.

*Выходные данные*: выведите одну строку из n цифр — искомое состояние дисплея, содержащее минимально возможное число.

## Задача №10

На большой перемене все n студентов Берляндского государственного университета выстроились в очередь в столовой. Однако оказалось, что у столовой тоже есть перерыв на обед, и она временно перестала работать.

Стоять в очереди, пока она не обслуживается, так скучно! Поэтому каждый из студентов записал номер студенческого билета того студента, что стоит в очереди перед ним, и того, что стоит в очереди непосредственно за ним. Если перед или после студента никого нет (то есть он первый или последний в очереди), то в качестве номера он записал число 0 (билеты студентов Берляндского государственного университета нумеруются с 1).

После этого все студенты разошлись по своим делам. Когда же они вернулись, то оказалось, что восстановить очередь не такая простая задача, как кажется на первый взгляд.

Помогите студентам восстановить состояние очереди по номерам студенческих билетов соседей в очереди.

*Входные данные*: в первой строке записано целое число n (2 ≤ n ≤ 2·105) — количество студентов в очереди. Далее следует n строк, где i-я строка содержит пару целых чисел ai, bi (0 ≤ ai, bi ≤ 106), где ai — номер студенческого билета того, кто стоит перед очередным студентом, а bi — номер студенческого билета того, кто стоит после очередного студента. Строки заданы в произвольном порядке. В качестве номера студенческого билета используется значение 0, если такого соседа нет. У всех студентов номера студенческих билетов различны. Гарантируется, что записи соответствуют очереди, в которой стоят все студенты в каком-то порядке.

*Выходные данные*: выведите последовательность n целых чисел x1, x2, ..., xn — последовательность номеров студенческих билетов всех студентов в порядке очереди от первого к последнему.

# Выполнение

## Листинг кода задачи №1

*# метод, возвращающий True, если два отрезка пересекаются. Необходим для того,  
# чтобы проверять на корректность вводимые данные***def** intersects(p1: tuple[int, int], p2: tuple[int, int], p3: tuple[int, int], p4: tuple[int, int]) -> bool:  
 common = (p2[0] - p1[0])\*(p4[1] - p3[1]) - (p2[1] - p1[1])\*(p4[0] - p3[0])  
  
 **if** common == 0:  
 **return False** r\_h = (p1[1] - p3[1]) \* (p4[0] - p3[0]) - (p1[0] - p3[0]) \* (p4[1] - p3[1])  
 s\_h = (p1[1] - p3[1]) \* (p2[0] - p1[0]) - (p1[0] - p3[0]) \* (p2[1] - p1[1])  
  
 r = r\_h / common  
 s = s\_h / common  
  
 **if** 0 <= r <= 1 **and** 0 <= s <= 1:  
 **return True  
 else**:  
 **return False  
  
  
def** danger\_turns(k: int, input\_func) -> int:  
 **if not** 4 <= k <= 1000:  
 **raise** ValueError(**'нарушено условие 4 ≤ k ≤ 1000'**)  
  
 passed\_points = []  
  
 **def** validate\_input():  
 x, y = input\_func()  
  
 **if** len(passed\_points) == 1 **and** (x - passed\_points[0][0] != 0 **or** y –

passed\_points[0][1] <= 0):  
 **raise** ValueError(**'первый прямой участок должен быть направлен**

**на север'**)  
  
 **if** (x, y) **in** passed\_points **and** len(passed\_points) != k:  
 **raise** ValueError(**'никакая пара точек (кроме первой и последней)**

**не может совпадать'**)  
  
 **if** len(passed\_points) == k **and** (x, y) != passed\_points[0]:  
 **raise** ValueError(**'последняя точка должна совпадать с первой'**)  
  
 **if** len(passed\_points) > 0:  
 **if** y < passed\_points[0][1] **or** (y == passed\_points[0][1] **and** x <

passed\_points[0][0]):  
 **raise** ValueError(**'самая южная и западная точка трассы**

**должна быть первой точкой'**)  
  
 **if** x - passed\_points[-1][0]!=0 **and** y-passed\_points[-1][1] != 0:  
 **raise** ValueError(**'движение может быть только на север, юг,**

**запад или восток'**)  
  
 **for** num **in** [x, y]:  
 **if not** -10000 <= num <= 10000:  
 **raise** ValueError(**'нарушено условие -10000 ≤ xi,yi ≤ 10000'**)  
  
 **for** i **in** range(0 **if** len(passed\_points) != k **else** 1,

len(passed\_points) - 2):  
 **if** intersects(passed\_points[i], passed\_points[i + 1],

passed\_points[-1], (x, y)):  
 **raise** ValueError(**'любая пара прямых участков трассы не**

**должна иметь общих точек'**)  
  
 passed\_points.append((x, y))  
 **return** x, y  
  
 ax, ay = validate\_input()  
 bx, by = validate\_input()  
 cx, cy = validate\_input()  
  
 result = 0  
  
 **for** i **in** range(k - 2):  
 vec\_a\_x = bx - ax  
 vec\_a\_y = by - ay  
  
 vec\_b\_x = cx - bx  
 vec\_b\_y = cy - by  
  
 z = vec\_a\_x \* vec\_b\_y - vec\_a\_y \* vec\_b\_x  
  
 **if** z == 0:  
 **raise** ValueError(**'два соседних участка не могут быть направлены**

**в одну и ту же сторону или в противоположные стороны'**)  
 **elif** z > 0:  
 result += 1  
  
 ax, ay = bx, by  
 bx, by = cx, cy  
 cx, cy = validate\_input()  
  
 **return** result  
  
  
print(**'входные данные: '**)  
k = int(input())  
result = danger\_turns(k, **lambda**: map(int, input().split()))  
  
print(**'\nвыходные данные: '**)  
print(result)

## Листинг кода задачи №2

**import** copy  
  
  
**def** get\_winner(members: str) -> str:  
 **if not** 1 <= len(members) <= 200000:  
 **raise** ValueError(**'нарушено условие 1 ≤ длина members ≤ 200000'**)  
  
 members = copy.copy(members)  
 i = 0  
  
 **while 'R' in** members **and 'D' in** members:  
 member = members[i]  
  
 **if** member == **'D'**:  
 members = members.replace(**'R'**, **'-'**, 1)  
 **elif** member == **'R'**:  
 members = members.replace(**'D'**, **'-'**, 1)  
  
 i = (i + 1) % len(members)  
  
 **for** member **in** members:  
 **if** member == **'D'**:  
 **return 'D'  
 elif** member == **'R'**:  
 **return 'R'  
  
  
assert** get\_winner(**'DDRRR'**) == **'D'  
assert** get\_winner(**'DDRRRR'**) == **'R'  
assert** get\_winner(**'D'**) == **'D'  
assert** get\_winner(**'R'**) == **'R'  
assert** get\_winner(**'DR'**) == **'D'  
assert** get\_winner(**'RD'**) == **'R'  
assert** get\_winner(**'RDD'**) == **'D'  
assert** get\_winner(**'DRRD'**) == **'D'  
assert** get\_winner(**'RDRD'**) == **'R'  
assert** get\_winner(**'RRRDDDDDR'**) == **'R'**print(**'входные данные: '**)  
members = input()  
  
print(**'\nвыходные данные: '**)  
print(get\_winner(members))

## Листинг кода задачи №3

**def** cheapest(delay: int, costs: list[int]):  
 **if not** 1 <= delay <= len(costs) <= 300000:  
 **raise** ValueError(**'нарушено условие 1 ≤ delay ≤ costs count ≤**

**300000'**)  
  
 **for** cost **in** costs:  
 **if not** 1 <= cost <= 10 \*\* 7:  
 **raise** ValueError(**f'нарушено условие 1 ≤ {**cost**} (стоимость**

**задержки рейса) ≤ 10^7'**)  
  
 sorted\_costs = []  
  
 **for** i **in** range(len(costs)):  
 index = max(0, i - delay)  
  
 **if** len(sorted\_costs) == 0:  
 sorted\_costs.append((costs[i], i))  
 **continue  
  
 while** index < len(sorted\_costs) **and** costs[i] <

sorted\_costs[index][0]:  
 index += 1  
  
 sorted\_costs.insert(index, (costs[i], i))  
  
 print(sorted\_costs)  
  
 price = 0  
 best\_order = []  
  
 **for** i **in** range(len(sorted\_costs)):  
 flight = sorted\_costs[i]  
 price += flight[0] \* (abs(flight[1] - (i + delay)))  
 best\_order.append(flight[1] + delay + 1)  
  
 **return** price, best\_order  
  
  
**assert** cheapest(2, [4, 2, 1, 10, 2]) == (20, [3, 6, 7, 4, 5])  
**assert** cheapest(2, [3, 1, 2]) == (11, [3, 5, 4])  
**assert** cheapest(5, [5, 5, 9, 100, 3]) == (321, [9, 8, 7, 6, 10])  
**assert** cheapest(1, [1]) == (1, [2])  
**assert** cheapest(1, [10000000]) == (10000000, [2])print(**'входные данные: '**)  
delay = int(input())  
costs = [int(cost) **for** cost **in** input().split()]  
  
result = cheapest(delay, costs)  
  
print(**'\nвыходные данные: '**)  
print(result[0])  
print(result[1])

## Листинг кода задачи №4

**def** permutation(nums: list[int]) -> tuple[int, list[int]]:  
 **if not** 2 <= len(nums) <= 200000:  
 **raise** ValueError(**'нарушено условие 2 ≤ nums count ≤ 200000'**)  
  
 **for** num **in** nums:  
 **if not** 1 <= num <= len(nums):  
 **raise** ValueError(**f'нарушено условие 1 ≤ {**num**} (число в массиве**

**Ивана) ≤ {**len(nums)**} (nums count)'**)  
  
 positions = dict()  
 result = []  
 count = 0  
**for** i **in** range(len(nums)):  
 num = nums[i]  
  
 **if** num **in** positions:  
 positions[num] += [i]  
 **else**:  
 positions[num] = [i]  
  
 **for** i **in** range(len(nums)):  
 num = nums[i]  
 *# если число на "своём" месте или встречается только один раз* **if** num == i + 1 **or** len(positions[num]) == 1:  
 result.append(num)  
 **continue** *# иначе находим минимальный ключ, которого в таблице еще нет* **for** j **in** range(1, len(nums) + 1):  
 **if** j > num **and** positions[num][0] == i:  
 result.append(num)  
 **break  
  
 if** j **not in** positions:count += 1  
 result.append(j)  
 positions[num].remove(i)  
 positions[j] = [i]  
 **break  
  
 return** count, result  
  
  
**assert** permutation([2, 2, 4, 4]) == (2, [1, 2, 3, 4])  
**assert** permutation([2, 2, 3, 3]) == (2, [1, 2, 3, 4])  
**assert** permutation([3, 2, 2, 2, 3]) == (3, [1, 2, 4, 5, 3])  
**assert** permutation([1, 2, 2, 2, 3]) == (2, [1, 2, 4, 5, 3])  
**assert** permutation([6, 8, 4, 6, 7, 1, 6, 3, 4, 5]) == (3, [2, 8, 4, 6, 7, 1, 9, 3, 10, 5])  
  
print(**'входные данные:'**)  
nums = [int(num) **for** num **in** input().split()]  
  
result = permutation(nums)  
  
print(**'\nвыходные данные:'**)  
print(result[0])  
print(result[1])

## Листинг кода задачи №5

**import** math  
**def** min\_capture\_cost(a: int, b: int, kingdoms: list[int]) -> int:  
 **if not** 1 <= len(kingdoms) <= 2 \* (10 \*\* 5):  
 **raise** ValueError(**'нарушено условие 1 ≤ kingdoms count ≤ 2 \* 10^5'**)  
  
 **if not** 1 <= a <= 10 \*\* 5:  
 **raise** ValueError(**'нарушено условие 1 ≤ a ≤ 10^5'**)  
  
 **if not** 1 <= b <= 10 \*\* 5:  
 **raise** ValueError(**'нарушено условие 1 ≤ b ≤ 10^5'**)  
  
 **for** i **in** range(len(kingdoms)):  
 prev = kingdoms[i - 1] **if** i > 0 **else** 0  
 next = kingdoms[i + 1] **if** i < len(kingdoms) - 1 **else** 10 \*\* 8 + 1  
 **if not** prev < kingdoms[i] < next:  
 **raise** ValueError(**'нарушено условие 1 ≤ x1 < x2 < … < xn ≤**

**10^8'**)  
  
 **if** a <= b:  
 current\_kingdom = 0  
 min\_cost = 0  
  
 **for** kingdom **in** kingdoms[:-1]:  
 min\_cost += b \* (kingdom - current\_kingdom)  
 min\_cost += a \* (kingdom - current\_kingdom)  
 current\_kingdom = kingdom  
  
 min\_cost += b \* (kingdoms[-1] - current\_kingdom)  
  
 **return** min\_cost  
 **else**:  
 min\_cost = math.inf  
  
 **for** i **in** range(len(kingdoms)):  
 temp\_min = 0  
 current\_kingdom = 0  
 remaining\_moves = i  
  
 **for** kingdom **in** kingdoms:  
 temp\_min += b \* (kingdom - current\_kingdom)  
  
 **if** remaining\_moves > 0:  
 remaining\_moves -= 1  
 temp\_min += a \* (kingdom - current\_kingdom)  
 current\_kingdom = kingdom  
  
 **if** temp\_min < min\_cost:  
 min\_cost = temp\_min  
 **else**:  
 **break  
  
 return** min\_cost  
  
  
**assert** min\_capture\_cost(2, 7, [3, 5, 12, 13, 21]) == 173  
**assert** min\_capture\_cost(6, 3, [1, 5, 6, 21, 30]) == 171  
**assert** min\_capture\_cost(9, 3, [10, 15]) == 75  
  
print(**'входные данные:'**)  
t = int(input())  
results = []  
n\_sum = 0  
  
**for** i **in** range(t):  
 a, b = map(int, input().split())  
 kingdoms = list(map(int, input().split()))  
 n\_sum += len(kingdoms)  
  
 **if** n\_sum > 2 \* (10 \*\* 5):  
 **raise** ValueError(**'нарушено условие: сумма значений 𝑛 по всем**

**наборам входных данных не превосходит 2⋅10^5'**)  
  
 results.append(min\_capture\_cost(a, b, kingdoms))  
  
print(**'\nвыходные данные:'**)  
print(results)

## Листинг кода задачи №6

*# проходимся по списку результатов и 'убираем' одну закрывающуюся  
# скобку, если это возможно. Это необходимо для того, чтобы убранную  
# скобку поставить вместо новой найденной решётки***def** grab\_closing(result: list[int]) -> bool:  
 **for** i **in** range(len(result) - 1, -1, -1):  
 **if** result[i] > 1:  
 result[i] -= 1  
 **return True  
  
 return False  
  
  
def** beautify\_count(src: str) -> list[int] **or** int:  
 **if not** 1 <= len(src) <= 10 \*\* 5:  
 **raise** ValueError(**'нарушено условие 1 ≤ длина src ≤ 10^5'**)  
  
 result = []  
  
 sharps\_count = 0  
  
 *# кол-во закрывающих скобок для соблюдения баланса* to\_balance\_count = 0  
  
 *# кол-во закрывающих скобок, которое можно применить для того, чтобы  
 # соблюсти условие: для каждого i (1≤i≤|s|) среди первых i символов  
 # строки s символов ')' не больше, чем символов '('* available\_count = 0  
  
 **for** symbol **in** src:  
 **if** symbol == **'('**:  
 to\_balance\_count += 1  
 **elif** symbol == **')'**:  
 to\_balance\_count -= 1  
 **elif** symbol == **'#'**:  
 sharps\_count += 1  
  
 **if** to\_balance\_count < 1 **or** sharps\_count < 1:  
 **return** -1  
  
 **for** symbol **in** src:  
 **if** symbol == **'('**:  
 available\_count += 1  
 **elif** symbol == **')'**:  
 available\_count -= 1  
  
 **if** grab\_closing(result):  
 to\_balance\_count += 1  
 **elif** symbol == **'#'**:  
 count = min(available\_count, to\_balance\_count)  
  
 **if** count < 1 **and** grab\_closing(result):  
 count += 1  
  
 result.append(count)  
 to\_balance\_count -= count  
 sharps\_count -= 1  
  
 **if** sharps\_count == 0:  
 **break  
  
 for** count **in** result:  
 **if** count < 1:  
 **return** -1  
  
 **return** result  
  
  
**assert** beautify\_count(**'(((#)((#)'**) == [2, 1]  
**assert** beautify\_count(**'(((#((#'**) == [3, 2]  
**assert** beautify\_count(**'(#((('**) == [1]  
**assert** beautify\_count(**'#((('**) == -1  
**assert** beautify\_count(**'(((#'**) == [3]  
**assert** beautify\_count(**'(#(#(#'**) == [1, 1, 1]  
**assert** beautify\_count(**'('**) == -1  
**assert** beautify\_count(**'((#)(((#'**) == [1, 3]  
**assert** beautify\_count(**'()((#((#(#()'**) == [2, 2, 1]  
  
print(**'входные данные:'**)  
src = input()  
  
print(**'\nвыходные данные:'**)  
print(beautify\_count(src))

## Листинг кода задачи №7

**def** is\_correct\_sequence(sequence: str) -> bool:  
 brackets\_stack = []  
 failed = **False  
  
 for** char **in** sequence:  
 **if** char == **'('**:  
 brackets\_stack.append(**'('**)  
 **elif** char == **')'**:  
 **if** len(brackets\_stack) > 0:  
 brackets\_stack.pop()  
 **else**:  
 failed = **True  
 break  
  
 return not** failed **and** len(brackets\_stack) == 0  
  
**assert not** is\_correct\_sequence(**'('**)  
**assert not** is\_correct\_sequence(**')'**)  
**assert not** is\_correct\_sequence(**'()()('**)  
**assert not** is\_correct\_sequence(**')('**)  
**assert** is\_correct\_sequence(**'()'**)  
**assert** is\_correct\_sequence(**'()()'**)  
**assert** is\_correct\_sequence(**'(()())'**)  
  
**def** correct\_combinations\_count(k: int, input\_func):  
 **if not** 1 <= k <= 3 \* (10 \*\* 5):  
 **raise** ValueError(**'нарушено условие 1 ≤ k ≤ 3 \* 10^5'**)  
  
 sequences = []  
 lengths\_sum = 0  
 count = 0  
  
 **for** i **in** range(k):  
 sequences.append(input\_func())  
 lengths\_sum += len(sequences[-1])  
  
 **if** lengths\_sum > 3 \* (10 \*\* 5):  
 **raise** ValueError(**'нарушено условие: сумма длин всех скобочных**

**последовательностей не превосходит 3⋅10^5'**)  
  
 **for** j **in** range(len(sequences)):  
 sequence = sequences[j]  
  
 **if** is\_correct\_sequence(sequences[-1] + sequence):  
 count += 1  
  
 **if** i != j **and** is\_correct\_sequence(sequence + sequences[-1]):  
 count += 1  
  
 **return** count  
  
  
print(**'входные данные:'**)  
k = int(input())  
  
count = correct\_combinations\_count(k, **lambda**: input())  
print(**'\nвыходные данные:'**)  
print(count)

## Листинг кода задачи №8

**def** balance\_duration(loads: list[int]) -> int:  
 **if not** 1 <= len(loads) <= 10 \*\* 5:  
 **raise** ValueError(**'нарушено условие 1 ≤ длина loads ≤ 10^5'**)  
  
 **for** load **in** loads:  
 **if not** 0 <= load <= 2 \* (10 \*\* 4):  
 **raise** ValueError(**'нарушено условие 0 ≤ кол-во запросов ≤ 2 ⋅**

**10^4'**)  
  
 loads = sorted(loads)  
 k = 0  
  
 start = 0  
 end = len(loads) - 1  
  
 **while** start < end:  
 **while** (loads[end] - 1) - (loads[start] + 1) >= 0:  
 loads[start] += 1  
 loads[end] -= 1  
 k += 1  
  
 start += 1  
 end -= 1  
  
 **return** k  
  
  
**assert** balance\_duration([1, 6]) == 2  
**assert** balance\_duration([10, 11, 10, 11, 10, 11, 11]) == 0  
**assert** balance\_duration([1, 2, 3, 4, 5]) == 3  
  
print(**'входные данные:'**)  
loads = list(map(int, input().split()))  
  
print(**'\nвыходные данные:'**)  
print(balance\_duration(loads))

## Листинг кода задачи №9

**def** min\_code(code: str) -> str:  
 **if not** 1 <= len(code) <= 1000:  
 **raise** ValueError(**'нарушено условие 1 ≤ длина code ≤ 1000'**)  
  
 digits = [int(digit) **for** digit **in** code]  
 n = len(digits)  
  
 min\_index = 0  
 min\_digit = digits[0]  
  
 **for** i **in** range(1, n):  
 digit = digits[i]  
 **if** digit < min\_digit:  
 min\_digit = digit  
 min\_index = i  
  
 shifted\_digits = [0] \* n  
 shift = n - min\_index  
  
 **for** i **in** range(n):  
 digit = digits[i]  
 shifted\_digits[(i + shift) % n] = digit  
  
 **if** shifted\_digits[0] > 0:  
 add\_count = 10 - shifted\_digits[0]  
  
 **for** i **in** range(n):  
 shifted\_digits[i] = (shifted\_digits[i] + add\_count) % 10  
  
 **return ''**.join(map(str, shifted\_digits))  
  
  
**assert** min\_code(**'579'**) == **'024'  
assert** min\_code(**'2014'**) == **'0142'  
assert** min\_code(**'328'**) == **'061'**print(**'входные данные:'**)  
code = input()  
  
print(**'\nвыходные данные:'**)  
print(min\_code(code))

## Листинг кода задачи №10

**def** find\_by\_first(data: list[tuple[int, int]], first: int) -> int:  
 **for** pair **in** data:  
 **if** pair[0] == first:  
 **return** pair[1]  
  
 **return** -1  
  
  
**def** ordered\_queue(k: int, students\_neighbors: list[tuple[int, int]]) -> list[int]:  
 result = [0] \* k  
 first = **None  
  
 if not** 2 <= k <= 2 \* (10 \*\* 5):  
 **raise** ValueError(**'нарушено условие 2 ≤ k ≤ 2 ⋅ 10^5'**)  
  
 **for** neighbors **in** students\_neighbors:  
 **for** number **in** neighbors:  
 **if not** 0 <= number <= 10 \*\* 6:  
 **raise** ValueError(**'нарушено условие 0 ≤ номер студ билета ≤**

**10^6'**)  
  
 *# Находим первого студента в очереди* **for** i **in** range(len(students\_neighbors)):  
 found = **True  
  
 for** j **in** range(len(students\_neighbors)):  
 **if** i != j **and** students\_neighbors[i][0] ==

students\_neighbors[j][1]:  
 found = **False  
 break  
  
 if** found:  
 first = students\_neighbors[i][0]  
 **break** *# далее, зная первого студента и то, что перед ним стоит ноль,  
 # можно легко восстановить очередь, например:  
 # в исходных данных имеются пары (0, 7) и (92, 15), при этом  
 # первым элементом является число 92. Отсюда следует, что число 7 стоит  
 # после 92 (от нуля сделав два шага вправо), а число 15 после 7 (от 92 два шага вправо).  
 # Получаем: 92, 7, 15* result[0] = first  
 **for** i **in** range(0, k - 1, 2):  
 result[i + 1] = find\_by\_first(students\_neighbors, 0 **if** i - 1 < 0

**else** result[i - 1])  
  
 **if** i + 2 < len(result):  
 result[i + 2] = find\_by\_first(students\_neighbors, result[i])  
  
 **return** result  
  
  
**assert** ordered\_queue(  
 8,  
 [(17, 0), (31, 20), (141, 17), (20, 2), (7, 31), (15, 141), (0, 7), (92, 15)]  
) == [92, 7, 15, 31, 141, 20, 17, 2]  
  
**assert** ordered\_queue(  
 4,  
 [(92, 31), (0, 7), (31, 0), (7, 141)]  
) == [92, 7, 31, 141]  
  
  
print(**'входные данные:'**)  
k = int(input())  
  
students\_neighbors = []  
  
**for** i **in** range(k):  
 a, b = map(int, input().split())  
 students\_neighbors.append((a, b))  
  
print(**'\nвыходные данные:'**)  
print(ordered\_queue(k, students\_neighbors))

# Вывод

В ходе проделанной курсовой работы, я выполнил 10 алгоритмических задач на языке Python.