**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудового Красного Знамени**

**федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

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

Кафедра “Математическая кибернетика и информационные технологии”

Курсовая работа по дисциплине:

«Структуры и алгоритмы обработки данных» на тему

«Решение задач»

Выполнил: студент группы БФИ 2001

Кожанов М.С.

Руководитель: Симонов С.Е

Москва 2022

**Содержание**

[**Техническое задание** 3](#_Toc103690122)

[**Выполнение задания** 3](#_Toc103690123)

[**Задание №1** 3](#_Toc103690124)

[**Задание №2** 5](#_Toc103690125)

[**Задание №3** 6](#_Toc103690126)

[**Задание №4** 7](#_Toc103690127)

[**Задание №5** 9](#_Toc103690128)

[**Задание №6** 10](#_Toc103690129)

[**Задание №7** 11](#_Toc103690130)

[**Задание №8** 12](#_Toc103690131)

[**Задание №9** 13](#_Toc103690132)

[**Задание №10** 14](#_Toc103690133)

[**Листинг кода** 16](#_Toc103690134)

[**Решение задания №1** 16](#_Toc103690135)

[**Решение задания №2** 16](#_Toc103690136)

[**Решение задания №3** 16](#_Toc103690137)

[**Решение задания №4** 16](#_Toc103690138)

[**Решение задания №5** 16](#_Toc103690139)

[**Решение задания №6** 16](#_Toc103690140)

[**Решение задания №7** 16](#_Toc103690141)

[**Решение задания №8** 16](#_Toc103690142)

[**Решение задания №9** 16](#_Toc103690143)

[**Решение задания №10** 16](#_Toc103690144)

[**Вывод** 16](#_Toc103690145)

# **Техническое задание**

**Целью курсовой работы является** решение задач в соответствии с 11 вариантом.

# **Выполнение задания**

## **Задание №1**

Пока Патрик бегал по магазинам, Спанч Боб решил немного подшутить над своим другом. Порывшись в его вещах, проказник обнаружил последовательность a1, a2, ..., am длины m, составленную из целых чисел от 1 до n, не обязательно различных. Далее Спанч Боб придумал последовательность f1, f2, ..., fn длины n и получил для каждого числа ai число bi = fai. Исходную последовательность Спанч Боб, разумеется, стёр.

Сложно передать словами, как же расстроился Патрик когда вернулся из магазина! Скажем лишь, что Спанч Боб быстро пожалел о содеянном и пытается восстановить исходную последовательность. Помогите ему это сделать или определите, что это невозможно.

**Входные данные**

В первой строке входных данных находятся два целых числа n и m (1 ≤ n, m ≤ 100 000) — длины последовательностей fi и bi соответственно.

Во второй строке содержатся n целых чисел, определяющих последовательность f1, f2, ..., fn (1 ≤ fi ≤ n).

В последней строке записаны m целых чисел, определяющих последовательность b1, b2, ..., bm (1 ≤ bi ≤ n).

**Выходные данные**

Если существует ровно одна последовательность ai, такая что bi = fai для всех i от 1 до m, то выведите "Possible". Далее выведите m целых чисел a1, a2, ..., am.

Если существует несколько подходящих последовательностей ai, то выведите "Ambiguity".

Если Спанч Боб ошибся в своих вычислениях, и ни одной подходящей последовательности ai не существует, то выведите "Impossible".

**Примеры**

входные данные

3 3

3 2 1

1 2 3

выходные данные

Possible

3 2 1

входные данные

3 3

1 1 1

1 1 1

выходные данные

Ambiguity

входные данные

3 3

1 2 1

3 3 3

выходные данные

Impossible

## **Задание №2**

Гена любит последовательности чисел. Недавно он открыл для себя новый тип последовательностей, который он назвал — почти арифметическая прогрессия. Назовем последовательность почти арифметической прогрессией, если ее элементы можно представить в виде:

a1 = p, где p — некоторое целое число;

ai = ai - 1 + ( - 1)i + 1·q (i > 1), где q — некоторое целое число.

Сейчас у Гены на листке записана последовательность b, состоящая из n целых чисел. Помогите Гене, найдите в ней самую длинную подпоследовательность чисел, которая является почти арифметической прогрессией.

Последовательность s1,  s2,  ...,  sk называется подпоследовательностью последовательности b1,  b2,  ...,  bn, если найдется такая возрастающая последовательность индексов i1, i2, ..., ik (1  ≤  i1  <  i2  < ...   <  ik  ≤  n), что bij  =  sj. Иными словами, последовательность s может быть получена из b путем вычеркивания некоторых элементов.

**Входные данные**

В первой строке задано целое число n (1 ≤ n ≤ 4000). В следующей строке задано n целых чисел b1, b2, ..., bn (1 ≤ bi ≤ 106).

**Выходные данные**

Выведите единственное целое число — длину максимальной по длине, искомой подпоследовательности.

**Примеры**

входные данные

2

3 5

выходные данные

2

## **Задание №3**

Сережа любит самые разные алгоритмы. Недавно он придумал новый алгоритм, на вход которому подается строка. Обозначим входную строку алгоритма как q = q1q2... qk. Алгоритм состоит из двух шагов:

Найти любую непрерывную подпоследовательность (подстроку) из трех символов строки q, которая не равна ни одной из строк «zyx», «xzy», «yxz». Если q не содержит ни одной такой подпоследовательности, завершить алгоритм, иначе перейти к шагу 2.

Переставить буквы в найденной подпоследовательности случайным образом, перейти к шагу 1.

Сережа считает, что алгоритм корректно работает на строке q, если существует хоть какая-то ненулевая вероятность того, что алгоритм завершится. Если же алгоритм в любом случае будет работать бесконечно долго на строке, считается, что он работает некорректно на ней.

Сережа хочет протестировать свой алгоритм. Для этого у него есть строка s = s1s2... sn, состоящая из n символов. Мальчик проводит серию из m тестов. В качестве i-го теста он подает на вход алгоритму строку slisli + 1... sri (1 ≤ li ≤ ri ≤ n). К сожалению, его реализация алгоритма работает слишком долго, поэтому Сережа обратился к вам за помощью. Для каждого теста (li, ri) определите, отработает алгоритм корректно на этом тесте или нет.

**Входные данные**

В первой строке содержится непустая строка s, ее длина (n) не превосходит 105. Гарантируется, что строка s состоит только из символов: 'x', 'y', 'z'.

Вторая строка содержит целое число m (1 ≤ m ≤ 105) — количество тестов. Следующие m строк содержат сами тесты. В i-ой строке записана пара целых чисел li, ri (1 ≤ li ≤ ri ≤ n).

**Выходные данные**

Для каждого теста выведите «YES» (без кавычек), если алгоритм работает корректно на соответствующем тесте и «NO» (без кавычек) в противном случае.

**Примеры**

входные данные

zyxxxxxxyyz

5

5 5

1 3

1 11

1 4

3 6

выходные данные

YES

YES

NO

YES

NO

## **Задание №4**

Chouti и его одноклассникам скоро предстоит поступать в университет. Чтобы попрощаться друг с другом, класс устроил большую прощальную вечеринку, на которой одноклассники, учителя и родители пели и танцевали.

Chouti вспомнил, что в этой вечеринке приняли участие 𝑛 человек. Чтобы сделать вечеринку смешнее, каждый надел одну шляпу из 𝑛 разновидностей странных шляп с номерами 1,2,…𝑛. Возможно, что одну разновидность шляпы надели несколько человек. Некоторые разновидности шляп могли остаться невостребованными кем-либо.

После вечеринки 𝑖-й человек сказал, что было ровно 𝑎𝑖 людей, на которых была надета шляпа, отличающаяся от его собственной.

Прошло несколько дней, поэтому Chouti забыл, кто был в каких шляпах, но ему очень интересно что-то про это вспомнить! Пусть 𝑏𝑖 будет видом той шляпы, которую надел 𝑖-й человек. Chouti хочет, чтобы вы нашли какие-нибудь 𝑏1,𝑏2,…,𝑏𝑛, которые не противоречат утверждениям каждого человека. Так как у некоторых людей могут быть проблемы с памятью, решения может не существовать.

**Входные данные**

Первая строка содержит одно целое число 𝑛 (1≤𝑛≤105) — количество людей на вечеринке.

Вторая строка содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (0≤𝑎𝑖≤𝑛−1), и задающая утверждения каждого человека.

**Выходные данные**

Если решения не существует, выведите «Impossible».

Иначе, выведите «Possible» а затем 𝑛 целых чисел 𝑏1,𝑏2,…,𝑏𝑛 (1≤𝑏𝑖≤𝑛).

Если существует несколько возможных решений, вы можете вывести любое из них.

**Примеры**

входные данные

3

0 0 0

выходные данные

Possible

1 1 1

входные данные

4

0 1 2 3

выходные данные

Impossible

## **Задание №5**

Строительство метро в Бергороде подходит к концу! Президент Берляндии решил посетить этот город и взглянуть на новое метро своими глазами.

Метро состоит из n станций. Оно построено по всем правилам Бергородского Транспортного Закона:

Для каждой станции i существует ровно один поезд, идущий от этой станции. Его конечная станция — pi, возможно, что pi = i;

Для каждой станции i существует ровно одна станция j такая, что pj = i.

Президент рассмотрит выгодность метро после посещения. Выгодность — это количество таких упорядоченных пар (x, y), что посетитель начнет на станции x, а после, воспользовавшись несколькими поездами (возможно нулевым количеством), прибудет на станцию y (1 ≤ x, y ≤ n).

Мэр Бергорода считает, что если выгодность метро недостаточно высока, то Президент может заменить мэра в городе (разумеется, текущий мэр не хочет, чтобы это произошло). Перед посещением Президентом города мэр успеет перестроить некоторые части метро, таким образом, изменяя значения pi для не более чем двух станций метро. Конечно, нарушать Бергородский Транспортный Закон недопустимо, поэтому метро должно подчиняться Закону и после изменений.

Мэр хочет перестроить такие части метро, чтобы выгодность метро стала максимальной возможной. Помогите ему посчитать, какую максимальную выгодность он может получить.

**Входные данные**

В первой строке записано одно целое число n (1 ≤ n ≤ 100000) — количество станций.

Во второй строке записаны n целых чисел p1, p2, ..., pn (1 ≤ pi ≤ n) — текущая структура метро. Все эти числа различны.

**Выходные данные**

Выведите одно число — максимальное возможное значение выгодности.

**Примеры**

входные данные

3

2 1 3

выходные данные

9

## **Задание №6**

В детском саду в группе 𝑛 детей, пронумерованных от 1 до 𝑛. Воспитательница дала 𝑖-му ребенку 𝑎𝑖 (1≤𝑎𝑖≤𝑛) конфет. Дети сели в ряд в порядке от 1 до 𝑛 слева направо и стали есть конфеты.

Во время поедания конфет 𝑖-й ребенок заметил, что нескольким детям, сидящим левее него, и нескольким детям, сидящим правее него, дали больше конфет, чем ему самому. Поэтому каждый ребенок посчитал по два числа 𝑙𝑖 и 𝑟𝑖 — количество детей слева от него, которым дали больше конфет, чем ему, и количество детей справа от него, которым дали больше конфет, чем ему.

Формально, 𝑙𝑖 есть количество индексов 𝑗 (1≤𝑗<𝑖), таких что 𝑎𝑖<𝑎𝑗, а 𝑟𝑖 есть количество индексов 𝑗 (𝑖<𝑗≤𝑛), таких что 𝑎𝑖<𝑎𝑗.

Каждый ребенок сказал воспитательнице посчитанные им числа 𝑙𝑖 и 𝑟𝑖. К сожалению, она уже забыла, сколько конфет кому она дала. Поэтому она просит помощи у вас — по массивам 𝑙 и 𝑟 определите, могла ли она дать детям конфеты так, что дети правильно посчитали свои значения 𝑙𝑖 и 𝑟𝑖, или кто-то из ребят точно ошибся, и она не могла так раздать конфеты. Если она могла раздать конфеты, то определите любой возможный способ того, как она могла это сделать.

**Входные данные**

В первой строке входных данных находится одно целое число 𝑛 (1≤𝑛≤1000) — количество детей.

Во второй строке находится 𝑛 целых чисел 𝑙1,𝑙2,…,𝑙𝑛 (0≤𝑙𝑖≤𝑛), разделенных пробелами.

Во третьей строке находится 𝑛 целых чисел 𝑟1,𝑟2,…,𝑟𝑛 (0≤𝑟𝑖≤𝑛), разделенных пробелами.

**Выходные данные**

Если воспитательница не могла так раздать конфеты, и кто-то из детей точно ошибся в своих расчетах, то в единственной строке выведите «NO» (без кавычек).

Иначе в первой строке выведите «YES» (без кавычек), а во второй строке выведите 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛, разделённых пробелами — количества конфет, которые можно было выдать детям с номерами 1,2,…,𝑛, соответственно. Обратите внимание, что эти числа не обязаны быть различными, но для них должно выполняться ограничение 1≤𝑎𝑖≤𝑛. Также, должно быть выполнено, что количество детей левее 𝑖-го, которым дали больше конфет, чем ему, было равно 𝑙𝑖, а количество детей правее 𝑖-го, которым дали больше конфет, чем ему, было равно 𝑟𝑖. Если существует несколько способов раздать конфеты, найдите любой из них.

**Примеры**

входные данные

5

0 0 1 1 2

2 0 1 0 0

выходные данные

YES

1 3 1 2 1

## **Задание №7**

Снарк и Филипп готовят задачи на предстоящие кволы. У них есть банк из n задач, они хотят выбрать из него любое непустое подмножество задач.

На кволах будет участвовать k опытных команд. Для каждой задачи известно, для каких из этих команд эта задача баян, то есть, про каждую задачу и каждую команду известно, знает ли эта команда эту задачу, или нет.

Определите, можно ли выбрать подмножество задач так, чтобы каждая из команд не знала хотя бы половину задач.

**Входные данные**

Первая строка содержит два целых числа n и k (1 ≤ n ≤ 105, 1 ≤ k ≤ 4) — число задач и число опытных команд на кволах. Следующие n строк содержат по k чисел, каждое из которых либо 0, либо 1. j-е число в i-й строке равно 1, если j-я команда знает i-ю задачу, и 0 иначе.

**Выходные данные**

Выведите «YES» (без кавычек), если можно выбрать непустой набор задач, удовлетворяющий ограничениям, и «NO» в противном случае.

Вы можете выводить каждую букву как заглавной, так и строчной («YeS» и «yes» можно вывести вместо «YES»).

**Примеры**

входные данные

5 3

1 0 1

1 1 0

1 0 0

1 0 0

1 0 0

выходные данные

NO

входные данные

3 2

1 0

1 1

0 1

выходные данные

YES

## **Задание №8**

Дана непустая строка s, состоящая из строчных латинских букв. Найдите количество пар непересекающихся подстрок-палиндромов этой строки.

Более формально: найдите количество четверок (a, b, x, y) таких, что 1 ≤ a ≤ b < x ≤ y ≤ |s| и подстроки s[a... b], s[x... y] являются палиндромами.

Палиндромом называется строка, которая одинаково читается слева направо и справа налево. Например, строки «abacaba», «z», «abba» — палиндромы.

Подстрокой s[i... j] (1 ≤ i ≤ j ≤ |s|) строки s = s1s2... s|s| называется строка, равная sisi + 1... sj. Например, подстрока s[2...4] строки s = «abacaba» равна «bac».

**Входные данные**

В первой строке записана непустая строка s, состоящая из строчных букв латинского алфавита ('a' ... 'z'). Длина строки s не более 2000 символов.

**Выходные данные**

Выведите единственное число — количество пар непересекающихся подстрок-палиндромов заданной строки.

Пожалуйста, не используйте спецификатор %lld для чтения или записи 64-х битовых чисел на С++. Рекомендуется использовать потоки cin, cout или спецификатор %I64d.

**Примеры**

входные данные

aa

выходные данные

1

входные данные

aaa

выходные данные

5

## **Задание №9**

Вам задана таблица, состоящая из n строк и m столбцов.

Числа в каждой строке образуют перестановку от чисел 1 до m.

Разрешается не более одного раза для каждой строки поменять в ней два числа местами. Также разрешается не более одного раза поменять местами два столбца целиком. Таким образом, суммарно можно совершить от 0 до n + 1 действия. Описанные действия можно выполнять в любом порядке.

Проверьте, можно ли с помощью указанных действий добиться ситуации, чтобы в каждой строке была записана тождественная перестановка 1, 2, ..., m. Другими словами, существует ли такая последовательность действий, в результате которой числа в каждой из строк будут отсортированы по возрастанию.

**Входные данные**

В первой строке входных данных записаны два числа n и m (1 ≤ n, m ≤ 20) — количество строк и столбцов в заданной таблице.

В следующих n строках записаны по m чисел — содержимое таблицы. Гарантируется, что числа в каждой строке образуют перестановку чисел от 1 до m.

**Выходные данные**

Выведите «YES» (без кавычек), если с помощью указанных действий можно получить тождественную перестановку одновременно во всех строках таблицы. В противном случае выведите «NO» (без кавычек).

**Примеры**

входные данные

2 4

1 3 2 4

1 3 4 2

выходные данные

YES

входные данные

4 4

1 2 3 4

2 3 4 1

3 4 1 2

4 1 2 3

выходные данные

NO

## **Задание №10**

Сегодня Маша пришла на урок математики. Зайдя в класс, она увидела на доске две последовательности целых чисел одинаковой длины 𝑛−1 и задание учителя, записанное ниже. Обозначим члены первой последовательности как 𝑎𝑖 (0≤𝑎𝑖≤3), а члены второй последовательности как 𝑏𝑖 (0≤𝑏𝑖≤3).

Маша быстро решила задачу учителя, и ей стало интересно, существует ли такая последовательность длины 𝑛, члены которой обозначим как 𝑡𝑖 (0≤𝑡𝑖≤3), что для любого 𝑖 (1≤𝑖≤𝑛−1) верно:

𝑎𝑖=𝑡𝑖|𝑡𝑖+1 (где | обозначает операцию побитового ИЛИ) и

𝑏𝑖=𝑡𝑖&𝑡𝑖+1 (где & обозначает операцию побитового И).

Такая задача оказалась слишком сложной для Маши, поэтому она просит Вас, как лучшего друга, проверить, существует ли такая последовательность 𝑡𝑖 длины 𝑛 подходящая под условия, описанные выше. Если такая последовательность существует, то Маша также хочет посмотреть на нее. Если есть несколько таких последовательностей, то Вы можете показать Маше любую из них.

**Входные данные**

Первая строка содержит одно целое число 𝑛 (2≤𝑛≤105) — длина последовательности 𝑡𝑖.

Вторая строка содержит 𝑛−1 целое число 𝑎1,𝑎2,…,𝑎𝑛−1 (0≤𝑎𝑖≤3) — первая последовательность, записанная на доске.

Третья строка содержит 𝑛−1 целое число 𝑏1,𝑏2,…,𝑏𝑛−1 (0≤𝑏𝑖≤3) — вторая последовательность, записанная на доске.

**Выходные данные**

В первой строке выведите одно слово: «YES» (без кавычек), если существует такая последовательность 𝑡𝑖, подходящая под условия, или «NO» (без кавычек), если такой последовательности не существует.

Если последовательность существует, то во второй строке выведите 𝑛 целых чисел 𝑡1,𝑡2,…,𝑡𝑛 (0≤𝑡𝑖≤3) — последовательность, подходящая под ограничения из условия.

Если существует несколько ответов, выведите любой из них.

**Примеры**

входные данные

4

3 3 2

1 2 0

выходные данные

YES

1 3 2 0

входные данные

3

1 3

3 2

выходные данные

NO

# **Листинг кода**

class WrongInputException(Exception):

pass

def driver(input\_func, process\_func):

data = input\_func()

res = process\_func(\*data)

print(f"Result for {process\_func.\_\_name\_\_} with {data} : {res}")

def input\_list(size, left, rigth):

a = [int(i) for i in input().split()]

if len(a) != size:

raise WrongInputException(f"Len of {a} must be equals {size}")

for i in a:

if i < left or i > rigth:

raise WrongInputException(f"Element is not in given range")

return a

def input\_int(left, right):

try:

n = int(input())

if n < left or n > right:

raise WrongInputException(f"Size of number is not in given range")

except ValueError:

raise WrongInputException("Input a number")

## **Решение задания №1**

def input\_task1\_data():

n = input\_int(1, 100\_000)

m = input\_int(1, 100\_000)

f = input\_list(n, 1, n)

b = input\_list(m, 1, n)

return f, b

def task1(f, b):

s1 = set(f) # избавляемся от дубликатов

s2 = set(b) # избавляемся от дубликатов

ans = [] # результирующая последовательность

if s2 <= s1: # если длины разные, то невозможно

d = {} # ключ - элементы из f, значение - список элементов из f

for i, fi in enumerate(f):

if fi in d: # если элемент из f уже есть в словаре, то добавляем

# его индекс

d[fi].append(i + 1) # делаем +1 тк минимальное значения n >= 1

else: # иначе инициализируем список

d[fi] = [i + 1]

for bi in b:

if bi in d and len(d[bi]) > 1: # если по одному индексу несколько

# значений, то будет несколько

# последовательностей а

return 'Ambiguity'

elif bi in d: # иначе добавляем единственный элемент f в результат

ans.append(d[bi][0])

return 'Possible', ' '.join([str(i) for i in ans])

return 'Impossible'

driver(input\_task1\_data, task1)

## **Решение задания №2**

def input\_task2\_data():

n = input\_int(1, 4000)

t = input\_list(n, 1, 106)

return n, t

def task2(n, t):

# матрица с длинами последовательностей для каждого элемента

dp = [[0] \* (n + 1) for \_ in range(n + 1)]

# проходим по всем элементам

for i in range(n):

p = -1

# проходим по элементам до текущего

for j in range(i):

# обновляем счётчик длины послдовательности

dp[i][j] = max(dp[i][j], dp[j][p] + 1)

# при выполнении равенства от j до i находится последовательность

if t[j] == t[i]:

p = j

# находим максимумальную длину последовательности

return max(max(dp[i]) for i in range(n)) + 1

driver(input\_task2\_data, task2)

## **Решение задания №3**

def input\_task3\_data():

s = input()

n = input\_int(1, 105)

if len(s) != n:

raise WrongInputException(f"Length of {s} != {n}")

letters = ("x", "y", "Z")

for letter in s:

if letter not in letters:

raise WrongInputException("All letters must be on of x, y, z")

return s, n

# FIXME: Fix s and test input

def task3(s, n):

tests = []

for \_ in range(n):

ti, b = map(int, input().split())

print(ti,b)

ti -= 1

b -= 1

tests.append((ti, b))

count\_left = {}

count\_left[-1] = (0, 0, 0)

x, y, z = 0, 0, 0

# считаем общее колво x, y, z до каждого элемента в строке (префиксная сумма)

for i, c in enumerate(s):

if c == 'x':

x += 1

elif c == 'y':

y += 1

elif c == 'z':

z += 1

count\_left[i] = (x, y, z)

# вычисляем колво x, y, z в заданном промежутке от a до b

def count(a, b):

x2, y2, z2 = count\_left[b]

x1, y1, z1 = count\_left[a - 1]

x = x2 - x1

y = y2 - y1

z = z2 - z1

return x, y, z

def is\_valid(a, b):

if b - a < 2: # последовательность из одного или меньше элементов

return True

x, y, z = count(a, b)

# попарная разность x, y, z

diffs = [abs(i) for i in (x-y, x-z, y-z)]

# колво x, y, z не должно отличаться

return not any(d > 1 for d in diffs)

result = []

for ti, b in tests:

# вывод ответа через тернаный оператор

result.append("YES" if is\_valid(ti, b) else "NO")

return ' '.join(result)

driver(input\_task3\_data, task3)

## **Решение задания №4**

def input\_task4\_data():

n = input\_int(1, 105)

t = input\_list(n, 0, n-1)

return n, t

def task4(n, t):

# список кортежей, где 1ый элемент - колво людей, у которых отличается

# шляпа от текущего гостя, 2ой элемент - индекс гостя

s = sorted([(n-ti, i) for i, ti in enumerate(t)])

ans = []

hat\_type = 0 # тип шляпы (on 1 до n)

one\_type\_hats\_processed\_count = 0 # колво одинаковых шляп

for i in range(n):

# начинаем обрабатывать новый тип шляп

if one\_type\_hats\_processed\_count == 0:

hat\_type += 1

ans.append((s[i][1], hat\_type))

# обновляем количество шляп того же типа по словам человека

one\_type\_hats\_processed\_count = s[i][0]-1

# сюда попадаем во время обработки одного типа шляп

else:

# если нынешний человек не наврал, то есть его слова идентичны

# словам предыдущего

if s[i][0] == s[i-1][0]:

one\_type\_hats\_processed\_count -= 1

ans.append((s[i][1], hat\_type))

else:

return 'Impossible'

# если колво разных шляп больше n по условию быть не может

# или остались необработанные шляпы, то тоже кто-то наврал

if hat\_type > n or one\_type\_hats\_processed\_count > 0:

return 'Impossible'

else:

ans.sort()

return 'Possible', [i[1] for i in sorted(ans)]

driver(input\_task4\_data, task4)

## **Решение задания №5**

def input\_task5\_data():

n = input\_int(1, 100\_000)

p = input\_list(n, 1, n)

return p

def task5(t):

p = [0] + t

# считаем длины путей от каждой станции

val = sorted([dfs(u, p) for u in p])

# сумма текущих путей + две перестановки

return sum([x\*x for x in val]) + 2 \* val[-1] \* val[-2]

# вычисляет длину путей для текущей станции

# dfs - depth first search (поиск в глубину)

def dfs(u, p):

res = 0

# пока можем двигаться дальше по станциям

while p[u] != 0:

# увеличиваем количество станций

res += 1

# меняем указатель на следующую станцию

p[u], u = 0, p[u]

return res

driver(input\_task5\_data, task5)

## **Решение задания №6**

def input\_task6\_data():

n = input\_int(1, 1000)

l = input\_list(n, 0, n)

r = input\_list(n, 0, n)

return l, r

def task6(n, l, r):

candies = [] # колво детей с меньшим или равным количеством конфет для каждого ребёнка

i = 0 # колво детей с большим колвом конфет для каждого ребёнка

l1 = []

r1 = []

for i in range(n):

# если слева или справа детей меньше, чем сказал ребёнок

# то он врёт

if l[i] > i or r[i] > n - i - 1:

print('NO')

break

# иначе считаем колво детей с меньшим или равным колвом кофент

candies.append(candies - l[i] - r[i])

for i in range(n):

# для каждого ребёнка слева

for i in range(i):

# если колво детей с меньшим колвом конфет для текущего ребёнка

# меньше, чем колов детей с меньшим коловм конфет для предыдущего

# то увеличиваем колво детей с большим колвом конфет

if candies[i] < candies[i]:

i += 1

l1.append(i)

i = 0

# то же самое, но для всех детей справа

for i in range(i + 1, n):

if candies[i] < candies[i]:

i += 1

r1.append(i)

i = 0

if l1 == l and r1 == r:

return "YES", candies

return "NO"

driver(input\_task6\_data, task6)

## **Решение задания №7**

def input\_task7\_data():

n = input\_int(1, 105)

k = input\_int(1, 4)

a = [

input\_list(k, 0, 1)

for \_ in range(n)

]

return k, a

def task7(k, a):

sq\_k = 2 << (k - 1) # возведение в k степень (все возможные варианты знания задачи командами)

rec = [0] \* sq\_k # список маркеров существующих расстановок "заниний" задач командами

# находим существующие "расстаноки" знаний команд

for i in a:

ind = int(''.join(i), 2)

rec[ind] += 1

ans = 'YES'

# если есть задача, которую не знает ни одна команда, то да

if rec[0] > 0:

return ans

for k in range(sq\_k):

for k in range(sq\_k):

# если найдётся пара задач, которую знает только одна из команд

if k & k == 0 and rec[k] > 0 and rec[k] > 0:

return ans

return 'NO'

driver(input\_task7\_data, task7)

## **Решение задания №8**

def input\_task8\_data():

t = input()

if len(t) > 2000:

raise WrongInputException(f"String is too long")

return t

def task8(t):

t\_size = len(t)

i = t\_size // 2 # середина строки

# списки счётчики палиндромов для каждого символа

n, a = [1] \* t\_size, [1] \* t\_size

# подсчёт нечётных палиндромов в первой половине строки

for x, y in [(i, i + 2) for i in range(i - 1)]:

# пока символы равны двигаем границы палиндрома и сравниваем символы

while x >= 0 and t[x] == t[y]:

n[y] += 1

a[x] += 1

x -= 1

y += 1

# подсчёт нечётных палиндромов во второй половине строки

for x, y in [(i, i + 2) for i in range(i - 1, t\_size - 1)]:

while y < t\_size and t[x] == t[y]:

n[y] += 1

a[x] += 1

x -= 1

y += 1

# аналогично для чётных палиндромов

for x, y in [(i, i + 1) for i in range(i)]:

while x >= 0 and t[x] == t[y]:

n[y] += 1

a[x] += 1

x -= 1

y += 1

for x, y in [(i, i + 1) for i in range(i, t\_size)]:

while y < t\_size and t[x] == t[y]:

n[y] += 1

a[x] += 1

x -= 1

y += 1

# обновляем счётчики для каждого символа

for i in range(t\_size - 1):

n[i + 1] += n[i]

# считаем сумму всех возможных расстановок

return sum(n[i] \* a[i + 1] for i in range(t\_size - 1))

driver(input\_task8\_data, task8)

## **Решение задания №9**

def input\_task9\_data():

n, m = input().split()

try:

n = int(n)

m = int(m)

except ValueError:

raise WrongInputException(f"Wrong input format")

if n < 1 or m < 1 or n > 20 or m > 20:

raise WrongInputException(f"Wrong range of values")

a = [

input\_list(m, 0, m)

for \_ in range(n)

]

return n, m, a

def task9(n, m, a):

for i in range(m): # i - указатель на начальный элемент строки

for j in range(i, m): # j - указатель на следующий элемент строки

cnt = 0 # счётчик перестановок

# проходим по всем строкам матрицы

for row in range(n):

# меняем два элемента местами

a[row][i], a[row][j] = a[row][j], a[row][i]

# считаем количество непоследовательных элементов

for k in range(m):

# каждый элемент строки должен быть равен своему индуксу+1

if a[row][k] != k + 1:

cnt += 1

# если перестановок 2 и меньше, то возвращаем да

if cnt < 3:

return "YES"

# иначе меняем элементы обратно

for row in range(n):

a[row][i], a[row][j] = a[row][j], a[row][i]

return "NO"

driver(input\_task9\_data, task9)

## **Решение задания №10**

def input\_task10\_data():

n = input\_int(2, 105)

a = input\_list(n-1, 0, 3)

b = input\_list(n-1, 0, 3)

return n, a, b

def task10(n, a, b):

t = [5] \* n # результирующий списк, значения 5 тк это первое невозможное

# значение для элемента t

for i in range(4):

t[0] = i # обновляем элемент

# проходим по всем последовательностям

for j in range(n-1):

# проходим по всем возможным значениям ti

for k in range(4):

# если нашли нужное значение, сохраняем

if t[j] | k == a[j] and t[j] & k == b[j]:

t[j+1] = k

# если все элементы списка найдеaы

if 5 not in t:

return "YES", t

return "NO"

driver(input\_task10\_data, task10)

# **Вывод**

Была выполнена цель курсовой работы, а именно были решены задачи в соответствии с 11 вариантом