**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»

Выполнил:

студент

группы БФИ 2001

Писаков А.А.

Руководитель:

Симонов С.Е.

Москва 2022

Оглавление

[1 Цель работы 3](#_Toc103034008)

[2 Техническое задание 3](#_Toc103034009)

[3 Практическая часть 13](#_Toc103034010)

[4 Вывод 18](#_Toc103034011)

1 Цель работы

Закрепление полученных за время семестра знаний по структурам и алгоритмам обработки данных.

# 2 Техническое задание

Необходимо найти решение нескольких задач в соответствии с вариантом (вариант 16).

Задача 1

Финалистами соревнований «Russian Code Cup» в 2214 году будут участники, ставшие победителями в одном из отборочных раундов.

Отборочные раунды делятся на основные и дополнительные. Каждый из основных отборочных раундов должен состоять из c задач, а победителями раунда считаются n человек, занявшие первые места в этом раунде. Каждый из дополнительных отборочных раундов состоит из d задач. Победителем дополнительного раунда становится один человек. Кроме этого, на финал без конкурса приглашаются k победителей финалов прошлых лет.

В результате всех отборочных раундов в финал должно пройти не менее n·m человек. Каким образом нужно организовать отборочные раунды, чтобы в результате всех отборочных раундов в финал прошли не менее n·m человек, а при этом суммарное количество использованных в раундах задач было как можно меньше?

Входные данные Первая строка содержит два целых числа c и d (1 ≤ c, d ≤ 100) — количество задач в основном и дополнительном раундах соответственно. Вторая строка содержит два целых числа n и m (1 ≤ n, m ≤ 100). Наконец, третья строка содержит целое число k (1 ≤ k ≤ 100) — число заранее отобранных победителей.

Выходные данные В первой строке выведите единственное целое число — минимальное количество задач, которое нужно подготовить членам жюри.

Примеры  
входные данные  
1 10  
7 2  
1  
выходные данные  
2

Задача 2

Ярослав, Андрей и Роман могут часами напролет играть в кубики. Но это игра для трех человек, поэтому, когда Роман отсутствует, Ярослав и Андрей играют в другую игру.

Роман оставляет каждому из них по слову, состоящему из 2·n символов «0» или «1». После этого игроки начинают ходить по очереди. Ярослав ходит первым. На своем ходе игрок должен выбрать целое число от 1 до 2·n, которое не выбиралось никем из игроков ранее, и выписать себе на листочек соответствующий символ из своей строки.

Обозначим слово, которое оставили Ярославу, как s = s1s2... s2n. Аналогично, обозначим слово, которое оставили Андрею, как t = t1t2... t2n. Тогда, если Ярослав выбрал на своем ходе число k, то он выпишет на свой листочек символ sk. Аналогично, если Андрей выбрал на своем ходе число r, то он выпишет на свой листочек символ tr.

Игра заканчивается, когда ни один из игроков не может сделать ход. После окончания игры, Ярослав составляет из символов, записанных на своем листочке, некоторое целое число (символы можно переставлять в любом порядке), тоже самое делает Андрей (полученные числа могут содержать лидирующие нули). Побеждает тот, у кого получилось большее число. Если получившиеся числа равны, то объявляется ничья.

Вам заданы две строки s и t. Определите исход игры при оптимальной игре Ярослава и Андрея.

Входные данные В первой строке записано целое число n (1 ≤ n ≤ 106). Во второй строке записана строка s — слово Ярослава. В третьей строке записана строка t — слово Андрея.

Гарантируется, что оба слова состоят из 2·n символов «0» и «1».

Выходные данные Выведите «First», если при оптимальной игре побеждает Ярослав, «Second» — в случае победы Андрея и «Draw», если при оптимальной игре будет ничья. Слова выводите без кавычек.

входные данные  
3  
110110  
001001  
выходные данные  
First  
входные данные  
3  
111000  
000111  
выходные данные  
Draw

Задача 3

Счастливый PMP учится в ВУЗе на первом курсе, где он изучает алгоритмические задачи. PMP обожает алгоритмические игры.

Один студент постарше дал счастливому PMP занятную игру. PMP даны две перестановки чисел от 1 до n, от него требуется преобразовать первую перестановку во вторую. За один шаг можно убрать из перестановки чисел последнее число и вставить его обратно в произвольное место. Другими словами, можно либо вставить последнее число между любыми двумя числами, идущими одно за другим, либо вставить число в начало перестановки.

Счастливый PMP знает алгоритм, решающий эту задачу, но он слишком медленный. PMP хочет знать минимальное количество шагов, за которое он может преобразовать первую перестановку во вторую.

Входные данные В первой строке записано единственное целое число n (1 ≤ n ≤ 2·105) количество чисел в каждой из двух заданных перестановок.

В следующей строке записано n целых чисел, разделенных пробелом — первая перестановка. Каждое число от 1 до n встретится в перестановке ровно один раз.

Следующая строка описывает вторую перестановку в аналогичном формате.

Выходные данные Выведите единственное целое число, обозначающее минимальное количество шагов, необходимых для преобразования первой перестановки во вторую.

Примеры  
входные данные  
3  
3 2 1  
1 2 3  
выходные данные  
2  
входные данные  
5  
1 2 3 4 5  
1 5 2 3 4  
выходные данные  
1

Задача 4

Мальчик Петя обожает шахматы. Он даже придумал свою собственную шахматную фигуру — полуконя. Полуконь может ходить в любом из следующих четырех направлений: на 2 клетки вперед и 2 вправо, на 2 клетки вперед и 2 влево, на 2 клетки назад и 2 вправо или на 2 клетки назад и 2 влево. Разумеется, полуконь не может выходить за пределы шахматной доски.

На стандартной шахматной доске Петя поставил двух полуконей. Петя одновременно делает ходы обоими полуконями. Клетки доски достаточно большие, поэтому полукони после некоторого хода могут встретиться, то есть оказаться в одной и той же клетке. После встречи полукони могут продолжить ходить, поэтому возможно, что в дальнейшем они вновь встретятся. Пете интересно, существует ли такая последовательность ходов, при которой полукони встретятся. Некоторые клетки Петя посчитал плохими, то есть не подходящими для встречи. Полукони могут ходить по этим клеткам, но встречи в этих клетках не засчитываются.

Петя подготовил несколько шахматных досок. Помогите Пете узнать для каждой доски, могут ли полукони встретиться на некоторой подходящей для этого клетке.

Пожалуйста, обратите внимание на разбор тестового примера.

Входные данные В первой строке задано число t (1 ≤ t ≤ 50) — количество досок. Каждая доска описывается матрицей символов, состоящей из 8 строк и 8 столбцов. Матрица состоит из символов «.», «#», «K», обозначающих соответственно пустую клетку, плохую клетку и клетку, в которой стоит полуконь. Гарантируется, что на доске стоят ровно 2 полуконя. Клетки, занимаемые полуконями, считаются подходящими для встречи. Тесты разделены пустой строкой.

Выходные данные Выведите для каждого теста в отдельной строке ответ на задачу: «YES», если полукони смогут встретиться, и «NO» в противном случае.

Примеры  
входные данные  
2  
........  
........  
......#.  
K..##..#  
.......#  
...##..#  
......#.  
K.......  
  
........  
........  
..#.....  
..#..#..  
..####..  
...##...  
........  
....K#K#  
выходные данные  
YES  
NO  
  
Примечание  
Рассмотрим первую доску из примера. Будем считать, что строки и столбцы матрицы нумеруются от 1 до 8 сверху вниз и слева направо соответственно. Полукони могут встретиться, например, в клетке (2, 7). Полуконь из клетки (4, 1) переходит в клетку (2, 3), а полуконь из клетки (8, 1) — в клетку (6, 3). Далее оба полуконя переходят в (4, 5), но эта клетка — плохая, поэтому они вместе переходят в клетку (2, 7).  
На второй доске полукони никогда не встретятся.

Задача 5

Маленький пингвин Поло любит свою родную деревню. В этой деревне n домов, пронумерованных целыми числами от 1 до n. На каждом доме висит табличка с целым числом, на i-том доме висит табличка с целым числом pi (1 ≤ pi ≤ n).

Маленький пингвин Поло очень любит гулять по этой деревне. Прогулка выглядит следующим образом. Сначала он стоит около дома с номером x. Потом пингвин идет к дому, номер которого написан на табличке дома x (то есть к дому px), затем к дому, номер которого написан на табличке дома px (то есть к дому ppx), и так далее.

Известно, что:

Начав прогулку от любого дома с номером от 1 до k, включительно, он может дойти до дома с номером 1. Начав прогулку от любого дома с номером от k + 1 до n, включительно, он точно не может дойти до дома с номером 1. Начав прогулку от дома с номером 1, пингвин Поло может попасть обратно к дому с номером 1 через некоторое ненулевое количество переходов от дома к дому.

Вам нужно найти, сколькими способами можно написать числа на табличках домов, чтобы описанные выше три условия выполнялись. Выведите остаток от деления этого количества на 1000000007 (109 + 7).

Входные данные В единственной строке через пробел заданы два целых числа n и k (1 ≤ n ≤ 1000, 1 ≤ k ≤ min(8, n)) — количество домов и число k из условия.

Выходные данные В единственной строке выведите целое число — ответ на задачу по модулю 1000000007 (109 + 7).

Примеры  
входные данные  
5 2  
выходные данные  
54

Задача 6

Facetook — широко известная социальная сеть. Недавно в ней появилась новая функция — стена с приоритетами. Эта функция позволяет сортировать всех ваших друзей по коэффициенту дружбы. Чем больше вы общаетесь с другом, тем больше будет его коэффициент дружбы.

На коэффициент дружбы влияет 3 типа действий:

человек X написал на стене человек Y ("X posted on Y's wall") — 15 очков;

человек X прокомментировал сообщение человека Y ("X commented on Y's post") — 10 очков;

человеку X понравилось сообщение человека Y ("X likes Y's post") — 5 очков. X и Y — два различных имени. Каждое действие увеличивает коэффициент дружбы между X и Y (и наоборот) на данное количество очков (коэффициент дружбы между X и Y равен коэффициенту дружбы между Y и X).

Вам дано n действий в формате, описанном выше. Выведите все различные имена, которые встречаются в списке действий, в отсортированном порядке в соответствие с коэффициентом дружбы с вами.

Входные данные В первой строке записано ваше имя. Во второй строке записано целое число n, количество действий (1 ≤ n ≤ 100). Далее следует n строк. Гарантируется, что каждая строка содержит описание ровно одного действия в описанном выше формате. Между каждой парой слов в строке есть ровно один пробел, никаких других пробелов в строке нет. Все буквы имеют нижний регистр. Все имена во входных данных имеют длину от 1 до 10 строчных латинских букв.

Выходные данные Выведите m строк, где m — количество различных имен во входных данных (исключая вас). Каждая строка должна содержать ровно одно имя. Имена должны быть отсортированы в по убыванию коэффициента дружбы с вами (т. е. люди с большим коэффициентом дружбы должны идти раньше). Если два или больше человек имеют одинаковый коэффициент дружбы, выводите их в алфавитном (лексикографическом) порядке.

Учтите, что нужно выводить все имена, присутствующие во входных данных (исключая вас), даже если человек имеет нулевой коэффициент дружбы с вами.

Лексикографическое сравнение реализует оператор "<" в современных языках программирования. Строка a лексикографически меньше строки b, если либо a является префиксом b, либо существует такое i (1 ≤ i ≤ min(|a|, |b|)), что ai < bi, а для любого j (1 ≤ j < i) aj = bj, где |a| и |b| обозначают длины строк a и b соответственно.

Примеры  
входные данные  
ahmed  
3  
ahmed posted on fatma's wall  
fatma commented on ahmed's post  
mona likes ahmed's post  
выходные данные  
fatma  
mona

Задача 7

Дальнейшие исследования мыслительных процессов зомби дали интересные результаты. Как мы знаем из предыдущей задачи, нервная система зомби состоит из n мозгов и m мозговых соединений, связывающих их воедино. Было замечено, что интеллектуальные способности зомби сильно зависят от топологии нервной системы. Точнее, определим как расстояние между двумя мозгами u и v (1 ≤ u, v ≤ n) как минимальное количество мозговых соединений, которые потребуется использовать, чтобы передать мысль между этими двумя мозгами. Мозговая задержка зомби определяется как максимальное расстояние между какой-нибудь парой мозгов. Ученые подозревают, что мозговая задержка является ключевым параметром определяющим, насколько умён данный зомби. Помогите им протестировать их гипотезу, написав программу для вычисления мозговой задержки данной нервной системы.

В данной задаче вы можете считать, что данная вам во входных данных мозговая система является правильной, то есть удовлетворяет пунктам (1) и (2) из лёгкой версии данной задачи.

Входные данные В первой строке входных данных записаны два целых числа n и m (1 ≤ n, m ≤ 100000), определяющих количество мозгов (пронумерованных от 1 до n) и количество мозговых связей в нервной системе, соответственно. В следующих m строках содержатся описания мозговых связей. Каждая связь задаётся как пара мозгов a b, которую она соединяет (1 ≤ a, b ≤ n, a ≠ b).

Выходные данные Выведите одно число — мозговую задержку данной нервной системы.

Примеры  
входные данные  
4 3  
1 2  
1 3  
1 4  
выходные данные  
2

Задача 8

В перерывах между конкурсами топ-модель Изабелла старается не скучать и развиваться. Например, сейчас она учится собирать кубик Рубика 2x2x2.

Сразу научиться собирать кубик довольно непросто, поэтому она сначала хочет научиться понимать, можно ли из текущего положения собрать кубик поворотом ровно одной грани на 90 градусов в любую сторону.

Чтобы проверять правильность своих ответов, ей хочется иметь программу, которая будет по заданной раскраске граней кубика говорить, можно ли его собрать требуемым образом.

Кубик является собранным, если для каждой грани верно, что все квадратики этой грани имеют один цвет.

[https://ru.wikipedia.org/wiki/Кубик\_Рубика](https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1%D0%B8%D0%BA_%D0%A0%D1%83%D0%B1%D0%B8%D0%BA%D0%B0)

Входные данные В первой строке дана последовательность из 24 целых чисел ai (1 ≤ ai ≤ 6), где ai — номер цвета i-го квадратика. Гарантируется, что каждый из цветов встречается ровно 4 раза.

Выходные данные Выведите «YES», если кубик можно собрать поворотом одной грани, и «NO» иначе.

Примеры  
входные данные  
2 5 4 6 1 3 6 2 5 5 1 2 3 5 3 1 1 2 4 6 6 4 3 4  
выходные данные  
NO  
входные данные  
5 3 5 3 2 5 2 5 6 2 6 2 4 4 4 4 1 1 1 1 6 3 6 3  
выходные данные  
YES

Задача 9

Задана последовательность a1, a2, ..., an одномерных отрезков, пронумерованных от 1 до n. Ваша задача — найти два различных индекса i и j таких, что отрезок ai лежит внутри отрезка aj.

Отрезок [l1, r1] лежит внутри отрезка [l2, r2] тогда и только тогда, когда l1 ≥ l2 и r1 ≤ r2.

Выведите позиции i и j. Если существует несколько возможных ответов, выведите любой из них. Если ответа не существует, то выведите -1 -1.

Входные данные В первой строке записано одно целое число n (1 ≤ n ≤ 3·105) — количество отрезков.

В каждой из следующих n строк записаны по два целых числа li и ri (1 ≤ li ≤ ri ≤ 109) — i-й отрезок.

Выходные данные Выведите две различные позиции i и j такие, что отрезок ai лежит внутри отрезка aj. Если существует несколько возможных ответов, выведите любой из них. Если ответа не существует, то выведите -1 -1.

Примеры  
входные данные  
5  
1 10  
2 9  
3 9  
2 3  
2 9  
выходные данные  
2 1

Задача 10

Автобус курсирует по координатной прямой Ox от точки x = 0 до точки x = a. Выехав из точки x = 0, он доезжает до точки x = a, сразу разворачивается и едет обратно в точку x = 0. Вернувшись в точку x = 0, он сразу снова едет в точку x = a и так далее. Таким образом, автобус курсирует от x = 0 до x = a и обратно. Переезд из точки x = 0 в x = a или из точки x = a в x = 0 называется рейсом автобуса. Всего автобус должен совершить k рейсов.

Бензобак автобуса вмещает b литров бензина. На преодоление единицы расстояния автобус тратит ровно один литр бензина. Свой путь автобус начинает с полным бензобаком.

Между x = 0 и x = a в точке x = f находится заправка. Других заправок на пути автобуса нет. Проезжая мимо заправки в любом из двух направлений, автобус может остановиться и пополнить запас бензина в бензобаке. Таким образом, после остановки на заправку в бензобаке автобуса будет b литров бензина.

Какое минимальное количество раз автобус должен дозаправиться в точке x = f, чтобы начав путь в точке x = 0 совершить k рейсов?

Входные данные В первой строке записаны четыре целых числа a, b, f, k (0 < f < a ≤ 106, 1 ≤ b ≤ 109, 1 ≤ k ≤ 104) — точка окончания первого автобусного рейса, вместимость бензобака автобуса, точка, в которой расположена заправка, а также требуемое количество рейсов.

Выходные данные Выведите минимальное количество дозаправок автобуса, которые необходимы, чтобы совершить k рейсов. Если автобус не сможет совершить k рейсов, выведите -1.

Примеры  
входные данные  
6 9 2 4  
выходные данные  
4

# 3 Практическая часть

Программный код для решения задачи 1 представлен в листинге 1.

Листинг 1

c, d = map(int, input().split())

n, m = map(int, input().split())

k = int(input())

target = n \* m - k

if target <= 0:

    print(0)

else:

    if c / n < d:

        winners = target // n

        zad = winners \* c

        winners \*= n

        target -= winners

        print(zad + min(c, target \* d))

    else:

        print(target \* d)

Программный код для решения задачи 2 представлен в листинге 2.

Листинг 2

n = int(input())

s1 = input()

s2 = input()

count1 = s1.count('1')

count2 = s2.count('1')

bit\_and = format(int(s1, 2) & int(s2, 2), 'b').count('1')

a = count1 - bit\_and // 2

b = count2 - bit\_and // 2 - bit\_and % 2

if b - a == 1 or a == b:

    print('Draw')

elif a > b:

    print('First')

else:

    print('Second')

Программный код для решения задачи 3 представлен в листинге 3.

Листинг 3

n=int(input())

p1=list(map(int,input().split()))

p2=list(map(int,input().split()))

count = 0

for i in range(n):

    if p2[i] == p1[count]:

        count += 1

print(n - count)

Программный код для решения задачи 4 представлен в листинге 4.

Листинг 4

n = int(input())

for l in range(n):

    pos = []

    for i in range(8):

        s = input()

        for j in range(8):

            if s[j] == 'K':

                pos.append((i, j))

    print('YES' if abs(pos[0][0] - pos[1][0]) % 4 == abs(pos[0][1] - pos[1][1]) % 4 == 0 else 'NO')

    if l != n - 1: input()

Программный код для решения задачи 5 представлен в листинге 5.

Листинг 5

n, k = map(int, input().split())

print((k \*\* (k - 1)) \* ((n - k) \*\* (n - k)) % 1000000007)

Программный код для решения задачи 6 представлен в листинге 6.

Листинг 6

def input\_score(s, name: str) -> int:

    if name + "'s" not in s and name not in s:

        balls = 0

    else:

        if s[1] == 'posted':

            balls = 15

        elif s[1] == 'commented':

            balls = 10

        else:

            balls = 5

    return balls

def input\_second\_name(s) -> str:

    return s[2].replace("'s", "") if s[1] == 'likes' else s[3].replace("'s", "")

name = input()

n = int(input())

dic = {}

for i in range(n):

    s = input().split()

    first\_name = s[0]

    score = input\_score(s, name)

    second\_name = input\_second\_name(s)

    if first\_name == name:

        if second\_name in dic:

            dic[second\_name] += score

        else:

            dic[second\_name] = score

    elif second\_name == name:

        if first\_name in dic:

            dic[first\_name] += score

        else:

            dic[first\_name] = score

    else:

        if first\_name not in dic:

            dic[first\_name] = 0

        if second\_name not in dic:

            dic[second\_name] = 0

b = list(dic.items())

b.sort(key=lambda item: (5000 - item[1], item[0]))

for item in b:

    print(item[0])

Программный код для решения задачи 7 представлен в листинге 7.

Листинг 7

def bfs(x, g):

    n, q = len(g), [x]

    dist = [0 if y == x else -1 for y in range(n)]

    i = 0

    while i < len(q):

        v = q[i]

        i += 1

        for to in g[v]:

            if dist[to] < 0:

                dist[to] = dist[v] + 1

                q.append(to)

    return q[-1], dist[q[-1]]

n, m = map(int, input().split())

g = [[] for \_ in range(n)]

for \_ in range(m):

    a, b = map(lambda x: int(x) - 1, input().split())

    g[a].append(b)

    g[b].append(a)

print(bfs(bfs(0, g)[0], g)[1])

Программный код для решения задачи 8 представлен в листинге 8.

Листинг 8

def cube() -> bool:

    r = [

    "111155223333224466554466",

    "111144223333664422555566",

    "114422225533434315156666",

    "115522224433414135356666",

    "111122553333442255666644",

    "111122443333446655226655",

    "212132326363444455556161",

    "616112122323444455556363",

    "121223233636444455551616",

    "161621213232444455553636",

    "551122223344141453536666",

    "441122223355343451516666"

    ]

    a = "".join(input().split(" "))

    for b in r:

        is\_assembled = True

        for i in range(24):

            for j in range(24):

                if (b[i] == b[j] and a[i] != a[j]) or (b[i] != b[j] and a[i] == a[j]):

                    is\_assembled = False

        if is\_assembled: return True

    return False

print("YES" if cube() else "NO")

Программный код для решения задачи 9 представлен в листинге 9.

Листинг 9

n = int(input())

dic = {}

for j in range(n):

    l, r = map(int, input().split())

    dic[j + 1] = (l, r)

lst = sorted(dic.keys(), key=lambda i: (dic.get(i)[0], 1\_000\_000\_001 - dic.get(i)[1]))

maxR = num = -1

flag = False

for i in range(len(lst)):

    r = dic[lst[i]][1]

    if r <= maxR:

        flag = True

        print(lst[i], num)

        break

    maxR = max(maxR, r)

    num = lst[i]

if not flag:

    print(-1, -1)

Программный код для решения задачи 10 представлен в листинге 10.

Листинг 10

def is\_refuel(course: bool, fuel: int, f: int, a: int, is\_last: bool) -> bool:

    if course:

        return fuel < a - f if is\_last else fuel < (a - f) \* 2

    else:

        return fuel < f if is\_last else fuel < f \* 2

def is\_enough\_fuel(k: int, a: int, b: int, f: int) -> bool:

    if k == 1:

        return not (b < f or b < a - f)

    elif k == 2:

        return not (b < f or b < 2 \* (a - f))

    else:

        return not (b < 2 \* f or b < 2 \* (a - f))

a, b, f, k = map(int, input().split())

if not is\_enough\_fuel(k, a, b, f):

    print(-1)

    exit()

currentF = b - f

currentK = refueling = 0

course = True

while currentK != k:

    if is\_refuel(course, currentF, f, a, k - currentK == 1):

        refueling += 1

        currentF = b

    currentK += 1

    currentF -= 2 \* (a - f) if course else 2 \* f

    course = not course

print(refueling)

# 4 Вывод

В ходе выполнения курсовой работы были закреплены знания, полученные в ходе семестра.