**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”

**Отчет по курсовой работе**

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы

БВТ2005

Тимошин Алексей

Витальевич

Проверил:

Мкртчян Грач Маратович

Москва 2022

**Оглавление**

[Задача 1 3](#_Toc102738224)

[Задача 2 5](#_Toc102738225)

[Задача 3 8](#_Toc102738226)

[Задача 4 12](#_Toc102738227)

[Задача 5 14](#_Toc102738228)

[Задача 6 17](#_Toc102738229)

[Задача 7 21](#_Toc102738230)

[Задача 8 23](#_Toc102738231)

[Задача 9 27](#_Toc102738232)

[Задача 10 33](#_Toc102738233)

[Вывод 37](#_Toc102738234)

Задача 1

В Берляндии есть особый туристический маршрут — Золотое кольцо. Этот маршрут состоит из 𝑛 городов и циклического железнодорожного маршрута. Города пронумерованы от 1 до 𝑛 таким образом, что: следующим городом для 1 является город 2; следующим городом для 2 является город 3; ... следующим городом для 𝑛 является город 1. Таким образом, маршрут замкнут в цикл, города пронумерованы по ходу движения (маршрут — односторонний).

Блогер Поликарп хочет начать своё путешествие в городе 1. Про каждый город он знает величину 𝑎𝑖 — сколько селфи он хочет сделать в 𝑖-м городе. За одно посещение города он может сделать там не более одного селфи. Так как он едет на поезде, то города он пропускать не может (он всегда перемещается из города 𝑖 в город 𝑖+1 для 1≤𝑖<𝑛 и из 𝑛 в 1). Таким образом, когда поезд останавливается в городе, то Поликарп обязательно посещает этот город. Если Поликарп посещает город многократно, то все посещения учитываются отдельно.

Какое наименьшее количество посещений городов придётся Поликарпу совершить, чтобы выполнить свой план по количеству селфи для каждого города? Учтите, что первым он всегда посещает город 1, так как именно в этом городе начинается его путешествие.

Входные данные В первой строке записано целое число 𝑛 (3≤𝑛≤2⋅105) — количество городов в Золотом кольце Берляндии.

Следующая строка содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (0≤𝑎𝑖≤109), где 𝑎𝑖 равно требуемому количеству селфи в 𝑖-м городе.

Гарантируется, что хотя бы одно из чисел 𝑎𝑖 строго больше нуля.

Выходные данные Выведите едиственное целое число — искомое минимальное количество посещений.

Примеры входные данные 3 1 0 0 выходные данные 1, входные данные 3 2 0 2 выходные данные 6

**Решение:**

def solution\_task1():

n = int(input())

selfie\_count = [int(el) for el in input().split(" ")]

max\_selfie = -1

max\_index = 0

for i in range(len(selfie\_count)):

if selfie\_count[i] >= max\_selfie:

max\_selfie = selfie\_count[i]

max\_index = i

return max\_index + 1 + ((max\_selfie - 1) \* n)

result = solution\_task1()

print(result)

Задача 2

Вы помните, как гордый Кай льдинками выкладывал слово «вечность»? Маленький Шелдон играет с кубиками, на каждом из которых изображена одна цифра от 0 до 9 (включительно). Шелдон выкладывает свое любимое число t. Он понял, что цифры 6 и 9 настолько похожи, что перевернув кубик с цифрой 6 получается кубик с цифрой 9 (и наоборот). Аналогичная ситуация имеет место и для кубиков с цифрами 2 и 5. Остальные цифры не подвержены подобному феномену, который маленький Шелдон называет «цифровой мимикрией».

Шелдон очень любит число t. Настолько, что хочет его видеть перед собой в наибольшем количестве экземпляров. Какое количество раз Шелдон сможет сложить свое любимое число t, используя заданный набор кубиков? Разумеется, каждый кубик может быть использован не более чем в одном экземпляре числа t. Сложив некоторый экземпляр числа t, Шелдон более не трогает его и кубики, использованные в этом экземпляре.

Входные данные В первой строке входного файла записано целое положительное число t (1 ≤ t ≤ 10000), t — любимое число Шелдона. Вторая строка содержит последовательность цифр, изображенных на кубиках Шелдона. Длина последовательности от 1 до 200, она содержит только цифры от 0 до 9 и не содержит пробелов.

Выходные данные Выведите максимально количество экземпляров числа t, которое может сложить Шелдон. Допускается использование кубика 6 в качестве кубика 9 (и наоборот), аналогично с кубиками 2 и 5. Каждый кубик может быть использован не более чем в одном экземпляре числа t.

Примеры входные данные 42 23454 выходные данные 2 входные данные 169 12118999 выходные данные 1

**Решение:**

def solve\_task2():

goal = input()

counter = 0

cubes = input()

i = 0

while i < len(cubes):

if cubes[i] == "2" or cubes[i] == "5":

cubes = cubes[0 : i] + "25" + cubes[i + 1:]

i += 1

elif cubes[i] == "6" or cubes[i] == "9":

cubes = cubes[0 : i] + "69" + cubes[i + 1:]

i += 1

i += 1

flag = True

while flag:

for el in goal:

if el in cubes:

if el == "2" or el == "5":

cubes = cubes.replace("2", "", 1)

cubes = cubes.replace("5", "", 1)

elif el == "6" or el == "9":

cubes = cubes.replace("6", "", 1)

cubes = cubes.replace("9", "", 1)

else:

cubes = cubes.replace(el, "", 1)

else:

flag = False

break

if flag:

counter += 1

return counter

result = solve\_task2()

print(result)

Задача 3

Рядом с университетом находится остановка маршрутки. Пары закончились, и n студентов приходят на остановку. i-ый студент появляется на остановке в момент времени ti (все ti различны). Будем считать, что остановка находится в точке координатной прямой x = 0, а маршрутка ходит вдоль луча Ox, то есть в сторону положительного направления координатной прямой и обратно. i-ому студенту нужно попасть в точку с координатой xi (xi > 0). Маршрутка перемещается по следующему алгоритму. Изначально она находится в точке 0. Студенты последовательно приходят на остановку и садятся в нее. Как только в маршрутку сядет m студентов или сядет последний студент из n, она отправится в сторону положительного направления координатной прямой. Маршрутка двигается со скоростью 1 единица расстояния в 1 единицу времени, то есть за время y она проезжает расстояние y.

Каждый раз, когда маршрутка проезжает точку, в которой нужно выйти хотя бы одному студенту, она останавливается, и эти студенты выходят. На высадку студентов тратится 1 + [k / 2] единиц времени, где k — количество студентов, которые выходят в этой точке. Запись [k / 2] обозначает целую часть при делении k на 2. Как только из маршрутки выходит последний студент, она разворачивается и едет без остановок обратно в точку x = 0. Там она вновь наполняется студентами, и история повторяется.

Если студенты приходят на остановку, когда маршрутки там нет, они встают в очередь и садятся в маршрутку в том порядке, в котором они пришли. Посадка любого количества студентов и остальные действия происходит мгновенно, а других пассажиров нет.

Напишите программу, которая определит для каждого студента момент времени, когда он вышел из маршрутки. Моментом выхода студента из маршрутки считайте момент ее остановки в точке выхода студента (несмотря на то, что высадка группы студентов произойдет не мгновенно).

Входные данные В первой строке записаны через пробел два целых числа n, m (1 ≤ n, m ≤ 105) — количество студентов и количество мест в маршрутке, соответственно. Далее в n строках описаны студенты, по одному в строке. Каждая строка содержит пару целых чисел ti, xi (1 ≤ ti ≤ 105, 1 ≤ xi ≤ 104). Строки заданы в порядке строгого возрастания ti. Значения xi могут совпадать.

Выходные данные Выведите n чисел w1, w2, ..., wn, wi — момент времени, когда i-ый студент вышел из маршрутки. Числа выводите в одну строку и разделяйте единичными пробелами. Примеры входные данные 1 10 3 5 выходные данные 8 входные данные 2 1 3 5 4 5 выходные данные 8 19

**Решение:**

def solution\_task3():

data = input().split(" ")

n = int(data[0])

m = int(data[1])

students = {}

for i in range(n):

data = input().split(" ")

students[int(data[0])] = int(data[1])

total\_time = 0

bus = []

result\_dict = {}

for key, value in students.items():

bus.append((key, value))

if len(bus) == m:

total\_time = max(total\_time, bus[-1][0])

bus.sort(key = lambda x : x[1])

total\_time += bus[0][1]

current\_stop = bus[0][1]

current\_out = 0

for i in range(m):

if bus[i][1] == current\_stop:

current\_out += 1

result\_dict[bus[i][0]] = total\_time

else:

total\_time += bus[i][1] - bus[i - 1][1]

current\_stop = bus[i][1]

total\_time += 1 + (current\_out // 2)

result\_dict[bus[i][0]] = total\_time

current\_out = 1

if current\_out == 1:

total\_time += 1

else:

for i in range(m - current\_out, m):

result\_dict[bus[i][0]] = total\_time

total\_time += 1 + (current\_out // 2)

total\_time += bus[-1][1]

bus = []

if len(bus) > 0:

total\_time = max(total\_time, bus[-1][0])

bus.sort(key = lambda x : x[1])

total\_time += bus[0][1]

current\_stop = bus[0][1]

current\_out = 0

for i in range(len(bus)):

if bus[i][1] == current\_stop:

current\_out += 1

result\_dict[bus[i][0]] = total\_time

else:

total\_time += bus[i][1] - bus[i - 1][1]

current\_stop = bus[i][1]

total\_time += 1 + (current\_out // 2)

result\_dict[bus[i][0]] = total\_time

current\_out = 1

if current\_out == 1:

total\_time += 1

else:

for i in range(len(bus) - current\_out, len(bus)):

result\_dict[bus[i][0]] = total\_time

total\_time += 1 + (current\_out // 2)

result\_list = []

sort\_keys = list(result\_dict.keys())

sort\_keys.sort()

for key in sort\_keys:

result\_list.append(result\_dict[key])

return " ".join([str(el) for el in result\_list])

result = solution\_task3()

print(result)

Задача 4

В Берляндии снова реформы! На этот раз парламент обсуждает реформу календаря. Чтобы сделать жизнь граждан Берляндии разнообразнее, решено изменить календарь. Так как все большее количество граждан жалуются, что «годы летят...», решено что с нового года количество дней в году будет неуклонно расти. Так в предстоящем году будет ровно a дней, в следующем a + 1 день, затем a + 2 и так далее. Такой график планируется на ближайшие n лет (в n-ый год длина года будет составлять a + n - 1 день).

Что станет с месяцами — пока не решено. Депутат Палевный выступил со следующим предложением.

Календарь каждого месяца удобно печатать на квадратном листе бумаге. Предлагается сделать так, что количество дней в каждом месяце должно быть квадратом некоторого натурального числа. Количество дней в месяце должно быть одинаковым для каждого месяца любого года, но может отличаться для разных лет. Количество дней в месяце должно делить количество дней в году этого месяца. Это правило гарантирует, что количество месяцев в каждом году — это целое число. Количество дней в месяце для каждого года надо подбирать так, чтобы сэкономить максимальное количество бумаги на печати календарей. Иными словами, количество дней в месяце должно быть как можно больше. Эти правила дают однозначное правило выбора количества дней в каждом месяце года при фиксированной длине года. Например, согласно предложению Палевного, год, состоящий из 108 дней, будет содержать три месяца по 36 дней в каждом, год из 99 дней будет состоять из 11 месяцев по 9 дней, а год из 365 дней — из 365 месяцев по одному дню.

Предложение вызвало бурное обсуждение в обществе, известный математик Перельманов быстро посчитал, что если предложение будет поддержано, то в период n лет, начиная с года в котором a дней, будет израсходовано p листов бумаги для печати комплекта календарей всех лет. В расчетах Перельманов учитывал, что комплект будет содержать по одному календарю каждого года, а каждый месяц будет напечатан на отдельном листе.

Повторите достижение Перельманова и выведите искомое число p по заданным натуральным a и n. Перельманов предостерегает вас, что ваша программа не должна работать более четырех секунд на максимальном тесте.

Входные данные Единственная строка входных данных содержит пару целых чисел a, n (1 ≤ a, n ≤ 107; a + n - 1 ≤ 107).

Выходные данные Выведите искомое число p.

**Решение:**

def solve\_task4():

data = input().split(" ")

a, n = int(data[0]), int(data[1])

b = a + n

p = 0

squares = [i for i in range(a + n)]

i = 2

while i \* i < b:

j = 1

while j \* i \* i < b:

squares[j \* i \* i] = j

j += 1

i += 1

for i in range(a, b):

p += squares[i]

return p

result = solve\_task4()

print(result)

Задача 5

— О, милая сердцу Бобрунья, не желаете ли вы прогуляться со мной по расчудесной лесополосе?

— Конечно, мой Умный Бобр, давайте насладимся прекрасными видами вместе. Как насчет вечера пятницы?

Умный Бобер засуетился. К пятнице все должно быть идеально, поэтому срочно нужно подготовить лесополосу к предстоящей прогулке — выпилить некоторые деревья.

Рассмотрим лесополосу как последовательность деревьев. Каждое дерево i характеризуется своей эстетической привлекательностью ai — одни деревья очень красивы, другие так себе, а третьи даже пугают своим внешним видом!

Умный Бобер вычислил, что для завоевания сердца Бобруньи нужно добиться следующих эффектов: во-первых, Бобрунью нужно порадовать: сумма эстетических привлекательностей оставшихся в лесополосе деревьев должна быть максимально возможной; во-вторых, Бобрунью нужно удивить: эстетические привлекательности первого и последнего деревьев лесополосы должны быть равны; ну и, конечно, прогулка должна состояться: нужно оставить хотя бы два дерева в лесополосе. Теперь помогите Умному Бобру! Какие деревья придется выпилить для завоевания сердца Бобруньи?

Входные данные В первой строке содержится единственное целое число n — изначальное количество деревьев в лесополосе, 2 ≤ n. Во второй строке через пробел перечислены целые числа ai — эстетические привлекательности каждого из деревьев. Все эстетические привлекательности не превосходят 109 по модулю, n ≤ 3·105. Выходные данные В первой строке выведите два целых числа — суммарную эстетическую привлекательность лесополосы после ее обработки Умным Бобром и количество выпиливаемых деревьев k.

В следующей строке выведите k чисел — номера деревьев, которые стоит выпилить. Считайте, что деревья пронумерованы от 1 до n слева направо.

Если существует несколько решений, то выведите любое из них. Гарантируется, что хотя бы два дерева имеют одинаковую эстетическую привлекательность.

Примеры входные данные 5 1 2 3 1 2 выходные данные 8 1 1

**Решение:**

from math import inf

def solve\_task5():

n = int(input())

trees = tuple(map(int, input().split(" ")))

d = {}

l\_sum = {}

max\_rating = -1 \* inf

begin\_tree = 0

end\_tree = 0

current\_sum = 0

for i in range(n):

if trees[i] in d:

d[trees[i]][1], d[trees[i]][2] = i, current\_sum - l\_sum[trees[i]]

else:

d[trees[i]] = [i, -1, current\_sum]

l\_sum[trees[i]] = current\_sum

if trees[i] > 0:

current\_sum += trees[i]

for key in d.keys():

if d[key][1] == -1:

continue

if key > 0:

temp\_sum = d[key][2] + key

else:

temp\_sum = d[key][2] + (key \* 2)

if temp\_sum > max\_rating:

max\_rating, begin\_tree, end\_tree = temp\_sum, d[key][0], d[key][1]

deleted\_trees = list(range(begin\_tree)) + list(range(end\_tree + 1, n))

for i in range(begin\_tree + 1, end\_tree):

if trees[i] < 0:

deleted\_trees.append(i)

deleted\_trees.sort()

deleted\_trees = [el + 1 for el in deleted\_trees]

k = len(deleted\_trees)

return " ".join([str(max\_rating), str(k)]) + "\n" + " ".join([str(el) for el in deleted\_trees])

result = solve\_task5()

print(result)

Задача 6

Существует язык программирования, в котором каждая программа представлена непустой последовательностью символов «<» и «>» и цифр. Объясним, как работает интерпретатор этого языка программирования. Программа интерпретируется посредством движения указателя команды (IP), который состоит из двух частей.

Указатель текущего символа (CP); Указатель направления (DP), который может указывать налево или направо; Изначально CP указывает на крайний левый символ в последовательности, а DP указывает направо.

Мы повторяем следующие шаги до тех пор, пока CP впервые укажет куда-то за пределы последовательности.

Если CP указывает на цифру, то интерпретатор выводит эту цифру. Затем CP перемещается на один шаг согласно направлению DP. После этого значение написанной в последовательности цифры уменьшается на один. Если выведенная цифра была 0, значит, ее нельзя уменьшить. Поэтому она удаляется из последовательности, и длина последовательности уменьшается на один. Если CP указывает на «<» или «>», то направление DP меняется на «влево» или «вправо», соответственно. Затем CP двигается на один шаг согласно DP. Если новый символ, на который указывает CP — это «<» или «>», то предыдущий символ будет удален из последовательности. Если в какой-то момент выполнения программы CP выходит за пределы последовательности, то выполнение программы завершается.

Очевидно, что любая программа на этом языке завершается после некоторого количества шагов.

У Вас есть последовательность s1, s2, ..., sn, состоящая из символов «<», «>» и чисел. Вы должны ответить на q запросов. Каждый запрос характеризуется двумя целыми числами l и r и спрашивает, сколько раз будет выведена каждая цифра, если мы запустим последовательность sl, sl + 1, ..., sr как независимую программу на описанном языке.

Входные данные В первой строке входных данных содержатся два целых числа n и q (1 ≤ n, q ≤ 100) — длина последовательности s и количество запросов.

Вторая строка содержит s, последовательность из «<», «>» и цифр (0..9), записанных слева направо. Заметьте, что символы s не разделяются пробелами.

Каждая из следующих q строк содержит по два целых числа li и ri (1 ≤ li ≤ ri ≤ n) — i-ый запрос.

Выходные данные Для каждого запроса выведите 10 целых чисел, записанных через пробел: x0, x1, ..., x9, где xi равняется количеству выведенных цифр i при выполнении соответствующей программы из запроса. Выводите ответы на запросы в том порядке, в каком запросы даны во входных данных.

Примеры входные данные

7 4

1>3>22<

1 3

4 7

7 7

1 7

выходные данные

0 1 0 1 0 0 0 0 0 0

2 2 2 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

2 3 2 1 0 0 0 0 0 0

**Решение:**

def solve\_task6():

n, q = map(int, input().split(" "))

program = input()

result = []

for k in range(q):

l, r = map(int, input().split(" "))

temp\_program = list(program[l - 1 : r])

cp = 0

dp = 1

temp\_result = [0 for i in range(10)]

flag = False

while 0 <= cp < len(temp\_program):

current\_symbol = temp\_program[cp]

if current\_symbol.isdigit():

flag = False

if current\_symbol == "0":

temp\_result[0] += 1

temp\_program.pop(cp)

if dp == 1:

cp -= 1

else:

temp\_result[int(temp\_program[cp])] += 1

temp\_program[cp] = str(int(current\_symbol) - 1)

else:

if flag:

temp\_program.pop(cp - dp)

if dp == 1:

cp -= dp

if current\_symbol == ">":

dp = 1

else:

dp = -1

flag = True

cp += dp

result.append(temp\_result)

return result

result = solve\_task6()

for line in result:

print(" ".join([str(el) for el in line]) + " ")

Задача 7

В цветочный магазин привезли n букетов, причем i-й букет состоит из ai цветов. Продавец Вася решил сделать из имеющихся букетов большие букеты.

Под большим букетом Вася понимает букет, составленный из двух или более привезенных букетов, причем количество цветов в большом букете обязательно должно быть нечетным. Каждый из привезенных букетов может стать частью не более одного большого букета. Если привезенный букет выбирается для составления большого букета, то все цветы, которые есть в нем, обязательно берутся в большой букет.

Определите максимальное количество больших букетов, которые может составить Вася.

Входные данные В первой строке следует целое положительное число n (1 ≤ n ≤ 105) — количество привезенных букетов.

Во второй строке следует последовательность целых чисел a1, a2, ..., an (1 ≤ ai ≤ 106) — количество цветов в каждом из привезенных букетов.

Выходные данные Выведите максимальное количество больших букетов, которые может составить Вася.

Примеры

входные данные

5

2 3 4 2 7

выходные данные

2

входные данные

6

2 2 6 8 6 12

выходные данные

0

**Решение:**

def solve\_task7():

n = int(input())

flowers = [int(el) for el in input().split(" ")]

divisible = 0

non\_divisible = 0

for el in flowers:

if el % 2 == 0:

divisible += 1

else:

non\_divisible += 1

return min(divisible, non\_divisible) if divisible >= non\_divisible else divisible + ((non\_divisible - divisible) // 3)

result = solve\_task7()

print(result)

Задача 8

В суматохе современной жизни люди часто забывают насколько прекрасен мир. Времени насладиться окружающим у них настолько мало, что некоторые даже занимают очередь в поликлинике сразу в несколько кабинетов одновременно, бегая из одной очереди в другую.

Наверняка многие из вас сталкивались с тем, когда незнакомец спрашивает у вас, кто последний в очереди, высказывает мнение, что он будет крайним, после чего скрывается в неизвестном направлении — это именно те представители современного мира, у которых коэффициент нехватки времени настолько велик, что они даже не смотрят зарубежные рейтинговые сериалы. Такие люди часто создают проблемы в очередях, ведь вновь пришедший человек не видит последнего в очереди и занимает место за «виртуальным» звеном этой цепочки, недоумевая, куда же делась легендарная личность.

Умный Бобер после длительной болезни записался на прием к терапевту. Кратко и лаконично терапевт сообщил Умному Бобру печальное известие: необходимо сделать ЭКГ. На следующий день Умный Бобер встал пораньше, поставил на скачивание известный сериал (до полной загрузки необходимо около трех часов) и, стиснув зубы, мужественно пошел занимать очередь в кабинет электрокардиограммы, который славится самыми большими очередями в поликлинике.

Отстояв около трех часов в очереди, Умный Бобер понял, что многие не видели, за кем занимали очередь, из-за чего получилась неразбериха. Он спросил у каждого бобра, имеющего желание посетить кабинет ЭКГ, за кем тот занимал. Если бобр не знал за кем он, то, возможно, сейчас его очередь на прием, а может ему еще стоять и стоять...

Как Вы догадались, Умный Бобер очень спешил домой, поэтому он дал Вам все необходимые данные, чтобы Вы помогли ему определить, каким же по номеру в очереди он может следовать.

Входные данные В первой строке записаны два целых числа n (1 ≤ n ≤ 103) и x (1 ≤ x ≤ n) — количество бобров, стоявших в очереди, и номер Умного Бобра соответственно. Все желающие попасть на прием к врачу пронумерованы от 1 до n.

Во второй строке содержится n целых чисел a1, a2, ..., an (0 ≤ ai ≤ n) — номер бобра, за которым следует i-ый бобер. Если ai = 0, то i-ый бобер не знает, за кем он занимал. Гарантируется, что значения ai соответствуют корректной очереди. Другими словами нет циклических зависимостей в очереди, а также за любым бобром в очереди может следовать не более одного бобра.

Количество нулевых элементов ai произвольно. Выходные данные Выведите все возможные позиции Умного Бобра в очереди в порядке возрастания.

Примеры

входные данные

6 1

2 0 4 0 6 0

выходные данные

2

4

6

**Решение:**

import copy

def \_\_solver(cur\_node, open\_indicator, data, n, path):

result = []

while data[cur\_node] != -1:

if cur\_node in path:

return []

path.add(cur\_node)

cur\_node = data[cur\_node]

if data[cur\_node] == -1:

result.append(len(path) + 1)

for i in range(len(open\_indicator)):

if open\_indicator[i] in path:

continue

new\_open = copy.deepcopy(open\_indicator)

new\_open.pop(i)

new\_data = copy.deepcopy(data)

new\_data[cur\_node] = open\_indicator[i]

new\_path = copy.deepcopy(path)

new\_path.add(cur\_node)

result.extend(\_\_solver(open\_indicator[i], new\_open, new\_data, n, new\_path))

return result

def solve\_task8():

n, x = map(int, input().split())

data = [int(el) - 1 for el in input().split(" ")]

open\_indicator = []

closed\_indicator = set()

for i in range(n):

if data[i] != -1:

closed\_indicator.add(data[i])

open\_indicator = list(set(list(range(n))) - closed\_indicator)

path = set()

positions = \_\_solver(x - 1, open\_indicator, data, n, path)

positions = list(set(positions))

positions.sort()

return positions

result = solve\_task8()

for el in result:

print(el)

Задача 9

В городе N. очень плохо с дорогами, общепитом и IT-инфраструктурой. Всего в городе n перекрестков, некоторые пары которых соединены двусторонними дорогами. Дорожная сеть состоит из n - 1 дороги, по дорогам можно добраться с любого перекрестка на любой другой. Да, вы правы — дорожная сеть образует неориентированное дерево.

Недавно мэр города придумал способ, устраняющий проблемы с общепитом и IT-инфраструктурой, причем одновременно! Решено поставить на перекрестках города ресторанчики двух известных сетей кафе для IT-шников: «iMac D0naldz» и «Burger Bing». Так как владельцы сетей не дружат, категорически запрещается размещать рестораны двух разных сетей на соседних перекрестках. Есть и другие требования. Вот полный список:

в каждом перекрестке должен находится не более чем один ресторан; каждый ресторан принадлежит либо «iMac D0naldz», либо «Burger Bing»; каждая сеть должна построить не менее одного ресторана; не существует пары перекрестков, которые соединены дорогой и на которых стоят рестораны разных сетей. Мэр собирается брать неплохой налог с каждого ресторана, поэтому он заинтересован в том, чтобы общее число ресторанов было максимальным.

Помогите мэру проанализировать ситуацию. Найдите все такие пары (a, b), что a ресторанов может принадлежать «iMac D0naldz», b — «Burger Bing», а сумма a + b максимальна.

Входные данные В первой строке входных данных содержится целое число n (3 ≤ n ≤ 5000) — количество перекрестков в городе. Далее в n - 1 строке перечислены все дороги, по одной дороге в строке. Каждая дорога задана парой чисел xi, yi (1 ≤ xi, yi ≤ n) — номерами соединяемых перекрестков. Считайте, что перекрестки пронумерованы от 1 до n.

Гарантируется, что заданная дорожная сеть представляет собой неориентированное дерево с n вершинами.

Выходные данные В первую строку выведите целое число z — количество искомых пар. Далее выведите все искомые пары (a, b) в порядке увеличения первой компоненты a.

Примеры

входные данные

5

1 2

2 3

3 4

4 5

выходные данные

3

1 3

2 2

3 1

**Решение:**

import copy

def \_\_solver(a\_nodes: list, connections: dict, initial\_nodes: set, n: int):

result = []

b\_nodes = []

challenge\_b = []

for node in initial\_nodes:

b\_nodes.append(node)

for chall in connections[node]:

if chall not in b\_nodes and chall not in challenge\_b:

challenge\_b.append(chall)

while len(challenge\_b) > 0:

node = challenge\_b.pop(0)

flag = False

temp\_challenge = []

for chall in connections[node]:

if chall not in a\_nodes:

if chall not in b\_nodes and chall not in challenge\_b:

temp\_challenge.append(chall)

else:

flag = True

break

if not flag:

challenge\_b.extend(temp\_challenge)

b\_nodes.append(node)

a, b = len(a\_nodes), len(b\_nodes)

if a and b and (a + b == n - 1):

result.append([len(a\_nodes), len(b\_nodes)])

for node in a\_nodes:

send\_initial\_nodes = copy.deepcopy(initial\_nodes)

for chall in connections[node]:

if chall not in a\_nodes:

a\_nodes.append(chall)

if chall in initial\_nodes:

send\_initial\_nodes.remove(chall)

result.extend(\_\_solver(a\_nodes, connections, send\_initial\_nodes, n))

return result

def solve\_task9():

n = int(input())

connections = {}

for i in range(n):

connections[i] = []

for i in range(n - 1):

data = input().split(" ")

x, y = int(data[0]) - 1, int(data[1]) - 1

connections[x].append(y)

connections[y].append(x)

initial\_nodes = []

for key, values in connections.items():

if len(values) == 1:

initial\_nodes.append(key)

result = []

i = 0

a\_nodes = []

changing\_initial\_nodes = copy.deepcopy(initial\_nodes)

for i in range(len(initial\_nodes) // 2):

a\_nodes.append(initial\_nodes[i])

send\_a\_nodes = copy.deepcopy(a\_nodes)

changing\_initial\_nodes.pop(0)

result.extend(\_\_solver(send\_a\_nodes, connections, changing\_initial\_nodes, n))

\_\_result = set()

for i in range(len(result)):

result[i].sort()

if tuple(result[i]) not in \_\_result:

\_\_result.add(tuple(result[i]))

result[i].sort(reverse=True)

\_\_result.add(tuple(result[i]))

\_\_result = list(\_\_result)

\_\_result.sort()

return \_\_result

result = solve\_task9()

print(len(result))

for el in result:

print(el[0], el[1])

Задача 10

К своим трем годам Умный Бобер в совершенстве овладел всеми арифметическими операциями, а от своего учителя по математике одаренный ученик получил следующее домашнее задание на лето:

Дана последовательность целых чисел a1, a2, ..., an. Необходимо последовательно произвести над ней m операций следующего вида:

Для заданных чисел xi и vi присвоить элементу axi значение vi. Для заданных чисел li и ri необходимо вычислить сумму , где f0 = f1 = 1 и при i ≥ 2: fi = fi - 1 + fi - 2. Для тройки чисел li ri di необходимо для всех x (li ≤ x ≤ ri) увеличить значение ax на di. Этим летом Умный Бобер решил отправиться в тур по великим озерам Канады, потому просит Вас помочь ему в решении данной задачи.

Входные данные Первая строка содержит два целых числа n и m (1 ≤ n, m ≤ 2·105) — количество чисел в последовательности и количество операций соответственно. Вторая строка содержит n целых чисел a1, a2, ..., an (0 ≤ ai ≤ 105). Далее идет m строк, каждая из которых описывает операцию. Каждый строка начинается с целого числа ti (1 ≤ ti ≤ 3) — тип операции:

если ti = 1, то далее идет два целых числа xi vi (1 ≤ xi ≤ n, 0 ≤ vi ≤ 105); если ti = 2, то далее идет два целых числа li ri (1 ≤ li ≤ ri ≤ n); если ti = 3, то далее идет три целых числа li ri di (1 ≤ li ≤ ri ≤ n, 0 ≤ di ≤ 105).

Выходные данные Для каждого запроса второго типа выведите вычисленную сумму по модулю 1000000000 (109).

Примеры

входные данные

5 5

1 3 1 2 4

2 1 4

2 1 5

2 2 4

1 3 10

2 1 5

выходные данные

12

32

8

50

**Решение:**

def solve\_task10():

n, m = map(int, input().split(" "))

data = list(map(int, input().split(" ")))

result = []

for k in range(m):

comand = list(map(int, input().split(" ")))

operation = comand[0]

if operation == 1:

data[comand[1] - 1] = comand[2]

elif operation == 3:

for i in range(comand[1] - 1, comand[2]):

data[i] += comand[3]

else:

temp\_sum = 0

dif = comand[2] - comand[1]

f1, f0 = 1, 1

for x in range(dif + 1):

if x < 2:

temp\_sum += data[comand[1] + x - 1]

else:

f2 = f1 + f0

temp\_sum += f2 \* data[comand[1] + x - 1]

f1, f0 = f2, f1

result.append(temp\_sum % 1000000000)

return result

result = solve\_task10()

for el in result:

print(el)

Вывод

В ходе выполнения курсовой работы я успешно выполнил все поставленные задачи и закрепил полученные в ходе изучения данной дисциплины знания.