**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»

Вариант №4

Выполнил: студент группы БФИ 2001

Гежин С.А

Проверил: Симонов С.Е.

Москва 2022

**СОДЕРЖАНИЕ**

[**Цель работы:** 2](#_Toc103387833)

[**Задача 1** 3](#_Toc103387834)

[**Задача 2** 10](#_Toc103387835)

[**Задача 3** 14](#_Toc103387836)

[**Задача 4** 19](#_Toc103387837)

[**Задача 5** 24](#_Toc103387838)

[**Задача 6** 30](#_Toc103387839)

[**Задача 7** 35](#_Toc103387840)

[**Задача 8** 39](#_Toc103387841)

[**Задача 9** 45](#_Toc103387842)

[**Задача 10** 52](#_Toc103387843)

[**Вывод** 57](#_Toc103387844)

# **Цель работы:**

Необходимо найти решение 10 в соответствии с вариантом. Также дополнительно в каждой задаче нужно сделать «защиту от дурака». Номер моего варианта – 4.

# **Задача 1**

На координатной плоскости стоит робот. Изначально он расположен в точке (0,0). Его путь описывается строкой 𝑠 длины 𝑛, состоящей из символов 'L', 'R', 'U', 'D'.

Каждый из этих символов соответствует некоторому ходу:

'L' (влево): робот перемещается из точки (𝑥,𝑦) в точку (𝑥−1,𝑦);

'R' (вправо): робот перемещается из точки (𝑥,𝑦) в точку (𝑥+1,𝑦);

'U' (вверх): робот перемещается из точки (𝑥,𝑦) в точку (𝑥,𝑦+1);

'D' (вниз): робот перемещается из точки (𝑥,𝑦) в точку (𝑥,𝑦−1).

Компания, которая создала этого робота, попросила вас оптимизировать путь робота некоторым образом. Чтобы сделать это, вы можете удалить любую непустую подстроку пути. Но эта компания не хочет, чтобы покупатели заметили перемену в поведении робота. Это означает, что если до оптимизации робот заканчивал свой путь в точке (𝑥𝑒,𝑦𝑒), то после оптимизации (т.е. удаления некоторой единственной подстроки из 𝑠) робот тоже должен заканчивать свой путь в точке (𝑥𝑒,𝑦𝑒).

Эта оптимизация — малобюджетный проект, поэтому вам нужно удалить кратчайшую возможную непустую подстроку, чтобы оптимизировать путь робота так, что конечная точка его пути не изменится. Возможна ситуация, когда вы не можете оптимизировать путь. Кроме того, возможно, что после оптимизации полученный путь станет пустой строкой (т.е. удаленная подстрока — это вся строка 𝑠).

Напомним, что подстрока строки 𝑠 — это такая строка, которая может быть получена из 𝑠 с помощью удаления некоторого числа символов (возможно, нулевого) из префикса и некоторого количества символов (возможно, нулевого) из суффикса. Например, подстроками «LURLLR» являются «LU», «LR», «LURLLR», «URL», но не «RR» и «UL».

Вам нужно ответить на 𝑡 независимых наборов входных данных.

Входные данные

Первая строка входных данных содержит одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных.

Следующие 2𝑡 строк описывают наборы входных данных. Каждый набор задан двумя строками. Первая строка набора содержит одно целое число 𝑛 (1≤𝑛≤2⋅10^5) — длину пути робота. Вторая строка набора содержит одну строку 𝑠, состоящую из 𝑛 символов 'L', 'R', 'U', 'D' — путь робота.

Гарантируется, что сумма всех 𝑛 по всем наборам входных данных не превосходит 2⋅10^5 (∑𝑛≤2⋅10^5).

Выходные данные

Для каждого набора входных данных выведите ответ на него. Если вы не можете удалить такую непустую подстроку, что конечная точка пути робота не изменится, выведите -1. В противном случае выведите два целых числа 𝑙 и 𝑟 такие, что 1≤𝑙≤𝑟≤𝑛 — границы подстроки, которую вы удаляете. Значение 𝑟−𝑙+1 должно быть минимальным возможным. Если существует несколько ответов, выведите любой из них.

Пример

входные данные

4

4

LRUD

4

LURD

5

RRUDU

5

LLDDR

выходные данные

1 2

1 4

3 4

-1

**Решение задачи 1**

**Листинг 1**

def all\_substring\_ind\_combinations(x):

"""

Эта функция создает и возвращает список индексов

непустых подстрок по длинне строки.

Аргумент: х - длинна строки, int.

Возвращаемое значение: s - список кортежей, сост.

из индексов всех возможных непустых подстрок, list.

"""

s = []

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

for j in range(1, x + 1):

if i < j:

s.append((i, j))

return s

def count\_coords(s):

"""

Эта функция находит координаты конечной точки пути робота.

Аргумент: s - строка, описывающая путь робота, string.

Возвращаемое значение: x, y - координаты конечной точки пути робота, (int, int).

"""

x = 0

y = 0

for sym in s:

if sym == 'L':

x -= 1

elif sym == 'R':

x += 1

elif sym == 'U':

y += 1

elif sym == 'D':

y -= 1

return x, y

def new\_string(s, l, r):

"""

Возвращает строку с удаленной подстрокой, указанной с помошью индексов.

Аргумент: s - начальная строка, string;

Аргумент: l - индекс левого конца подстроки, int;

Аргумент: r - индекс правого конца подстроки, int.

Возвращаемое значение:

"""

new\_s = ''

for i in range(len(s)):

if not l <= i + 1 <= r:

new\_s += s[i]

return new\_s

def Task\_1(n, s):

combinations = all\_substring\_ind\_combinations(n)

destination = count\_coords(s)

result = ''

minn = math.inf

for tup in combinations:

l, r = tup[0], tup[1]

if count\_coords(new\_string(s, l, r)) == destination\

and (r - l + 1 < minn):

minn = r - l + 1

result = (l, r)

if minn == math.inf:

return -1

else:

return result

def main():

print('Входные данные:\n')

t = input('Введите количество наборов входных данных, t: ')

if is\_num\_int(t) is False:

exit('Введенная переменная не является целым числом!')

t = int(t)

if double\_inequality\_validation(t, 1, 1000) is False:

exit('Введенное число не соответствует условию!')

answers = []

summ\_n = 0

for i in range(t):

n = input('Введите длинну пути робота: ')

if is\_num\_int(n) is False:

exit('Введенная переменная не является целым числом!')

n = int(n)

if double\_inequality\_validation(n, 1, 2 \* 10 \*\* 5) is False:

exit('n не соответствует условию!')

summ\_n += n

if inequality\_validation(summ\_n, 2 \* 10 \*\* 5) is False:

exit('Сумма всех 𝑛 по всем наборам входных данных'

' превысила допустимую!')

s = input('Введите путь робота: ')

if string\_length\_validation(s, n) is False:

exit('Длина строки отличается от ожидаемой!')

if string\_symb\_validation(s, 'LRUD') is False:

exit('Символы введенной строки отличаются от ожидаемых!')

answer = Task\_1(n, s)

answers.append(answer)

print('\nВыходные данные:\n')

for ans in answers:

if ans == -1:

print(ans)

else:

l, r = ans

print(l, r)

main()

# **Задача 2**

Для некоторой 01-строки 𝑠 (то есть такой, что каждый символ 𝑠𝑖 — это либо '0', либо '1') выписали все пары подряд идущих символов (все подстроки длины 2). Для каждой пары (подстроки длины 2) посчитали количество символов '1' (единиц) в ней.

Вам заданы три числа:

𝑛0 — количество пар подряд идущих символов (подстрок), в которых 0 единиц;

𝑛1 — количество пар подряд идущих символов (подстрок), в которых 1 единица;

𝑛2 — количество пар подряд идущих символов (подстрок), в которых 2 единицы.

Например, для строки 𝑠=«1110011110» были бы выписаны следующие подстроки:

«11», «11», «10», «00», «01», «11», «11», «11», «10». Таким образом, 𝑛0=1, 𝑛1=3, 𝑛2=5.

Ваша задача — по заданным 𝑛0,𝑛1,𝑛2 восстановить любую подходящую 01-строку (бинарную строку) 𝑠. Гарантируется, что хотя бы одно из чисел 𝑛0,𝑛1,𝑛2 отлично от 0.

Входные данные

В первой строке записано целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных в тесте. Далее во входных данных записаны сами наборы.

Каждый набор состоит из одной строки, которая содержит три целых числа 𝑛0,𝑛1,𝑛2 (0≤𝑛0,𝑛1,𝑛2≤100; 𝑛0+𝑛1+𝑛2>0). Гарантируется, что ответ для заданных 𝑛0,𝑛1,𝑛2 существует.

Выходные данные

Выведите 𝑡 строк. Каждая из строк должна содержать ответ в виде 01-строки (бинарной строки) на соответствующий набор входных данных. Если возможных ответов несколько, выведите любой из них.

Пример

входные данные

7

1 3 5

1 1 1

3 9 3

0 1 0

3 1 2

0 0 3

2 0 0

выходные данные

1110011110

0011

0110001100101011

10

0000111

1111

000

**Решение задачи 2**

**Листинг 2**

def Task\_2(n0, n1, n2):

output = ''

if (n1 != 0):

output += '0' \* (n0 + 1)

output += '1' \* (n2 + 1)

for i in range(n1 - 1):

if i % 2 == 1:

output += '1'

else:

output += '0'

elif n2 == 0:

output += '0' \* (n0 + 1)

else:

output += '1' \* (n2 + 1)

return output

def main():

print('Входные данные:\n')

t = input('Введите количество наборов входных данных: ')

if is\_num\_int(t) is False:

exit('Введенная переменная не является целым числом!')

t = int(t)

if double\_inequality\_validation(t, 1, 1000) is False:

exit('Введенное число не соответствует условию!')

answers = []

for i in range(t):

test = input('Введите n0, n1, n2 через пробел: ')

if mas\_length\_validation(test, 3) is False:

exit('Введенное кол-во переменных отличается от ожидаемого!')

if int\_mas\_validation(test) is False:

exit('Не все введенные переменные целочисленны!')

n0, n1, n2 = tuple(map(int, test.split()))

if (double\_inequality\_validation(n0, 0, 100) is False) or\

(double\_inequality\_validation(n1, 0, 100) is False) or\

(double\_inequality\_validation(n2, 0, 100) is False):

exit('Одна или несколько переменных вышли за границы допустимых значений!')

if inequality\_validation(1, n0 + n1 + n2) is False:

print('Одна из переменных должна быть отлична от 0!')

answer = Task\_2(n0, n1, n2)

answers.append(answer)

print('\nВыходные данные:\n')

for ans in answers:

print(ans)

main()

# **Задача 3**

У вашего одноклассника, которого вы не очень любите за его занудство, но уважаете за его ум, были обнаружены две строки: строка 𝑠 длины 𝑛 и строка 𝑡 длины 𝑚.

Последовательность индексов 𝑝1,𝑝2,…,𝑝𝑚, где 1≤𝑝1<𝑝2<…<𝑝𝑚≤𝑛, называется хорошей, если 𝑠𝑝𝑖=𝑡𝑖 для всех 𝑖 от 1 до 𝑚. Шириной последовательности называется величина max1≤𝑖<𝑚(𝑝𝑖+1−𝑝𝑖).

Помогите однокласснику найти хорошую последовательность индексов с максимальной шириной. Одноклассник обещал вам, что строки 𝑠 и 𝑡 таковы, что хотя бы одна хорошая последовательность точно существует.

Входные данные

Первая строка входных данных содержит два числа 𝑛 и 𝑚 (2≤𝑚≤𝑛≤2⋅10^5) — длины строк 𝑠 и 𝑡 соответственно.

Во второй строке входных данных задана строка 𝑠, состоящая из 𝑛 строчных букв латинского алфавита.

В третьей строке задана строка 𝑡, состоящая из 𝑚 строчных букв латинского алфавита.

Гарантируется, что существует хотя бы одна хорошая последовательность индексов.

Выходные данные

Выведите одно число — максимальную ширину хорошей последовательности.

Примеры

входные данные

5 3

abbbc

abc

выходные данные

3

входные данные

5 2

aaaaa

aa

выходные данные

4

входные данные

5 5

abcdf

abcdf

выходные данные

1

входные данные

2 2

ab

ab

выходные данные

1

Примечание

В первом примере существует две хороших последовательности с шириной 3: это {1,2,5} и {1,4,5}.

Во втором примере хорошая последовательность максимальной ширины — это {1,5}.

В третьем примере есть лишь одна хорошая последовательность — это {1,2,3,4,5}.

В четвёртом примере есть лишь одна хорошая последовательность — это {1,2}.

**Решение задачи 3**

**Листинг 3**

def Task\_3(n, m, s, t):

l = [m] \* m

r = [0] \* m

j = 0

for i in range(n):

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

l[j] = i

j += 1

if j >= m:

break

j = m - 1

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

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

r[j] = i

j -= 1

if j < 0:

break

ans = 0

for i in range(m - 1):

ans = max(r[i + 1] - l[i], ans)

return ans

def main():

print('Входные данные:\n')

test = input('Введите m, n через пробел: ')

if mas\_length\_validation(test, 2) is False:

exit('Введенное кол-во переменных отличается от ожидаемого!')

elif int\_mas\_validation(test) is False:

exit('Не все введенные переменные целочисленны!')

n, m = tuple(map(int, test.split()))

if (double\_inequality\_validation(m, 2, n) is False) or\

(double\_inequality\_validation(n, m, 2 \* 10 \*\* 5) is False):

exit('Одна или несколько переменных вышли за границы допустимых значений!')

s = input('Введите строку s: ')

if string\_length\_validation(s, n) is False:

exit('Длинна строки s отличается от n!')

if string\_symb\_validation(s, 'abcdefghijklmnopqrstuvwxyz') is False:

exit('Строка s должна состоять только из лат. букв!')

t = input('Введите строку t: ')

if string\_length\_validation(t, m) is False:

exit('Длинна строки t отличается от m!')

if string\_symb\_validation(t, 'abcdefghijklmnopqrstuvwxyz') is False:

exit('Строка t должна состоять только из лат. букв!')

answer = Task\_3(n, m, s, t)

if answer == 0:

exit('Не существует ни одной хорошей последовательности индексов!')

print('\nВыходные данные:\n')

print(answer)

main()

# **Задача 4**

В Семи Королевствах n городов и n - 1 дорога, каждая дорога соединяет два города, и возможно достичь каждый город из каждого, передвигаясь по дорогам.

Теон и Яра Грейджой начинают путешествовать на лошади по дорогам из первого города. Из-за тумана они не видят, куда лошадь их везет. Когда лошадь входит в город (в том числе в первый), она направляется в один из городов, соединенных с ним. Но это странная лошадь, поэтому она пойдет только в город, в котором до этого не была. Она идет равновероятно в каждый из таких городов и останавливается, если таких городов нет.

Пусть длина каждой дороги 1. Путешествие начинается в городе 1. Чему равно математическое ожидание длины их путешествия? Формально, математическое ожидание — это ожидаемое (то есть среднее) значение. Подробнее можно прочесть по ссылке https://ru.wikipedia.org/wiki/Математическое\_ожидание.

Входные данные

Первая строка содержит одно целое число n (1 ≤ n ≤ 100000) — количество городов.

Далее следует n - 1 строк. i-я из них содержит два целых числа ui и vi (1 ≤ ui, vi ≤ n, ui ≠ vi) — номера городов, соединенных i-й дорогой.

Гарантируется, что возможно достичь каждый город из каждого, передвигаясь по дорогам.

Выходные данные

Выведите одно число — математическое ожидание длины путешествия. Путешествие начинается в городе 1.

Ваш ответ будет считаться правильным, если его абсолютная или относительная точность не превосходит 10 ^- 6.

А именно, если ваш ответ равен a, а ответ жюри равен b, то ваш ответ будет зачтен, если .

Примеры

входные данные

4

1 2

1 3

2 4

выходные данные

1.500000000000000

входные данные

5

1 2

1 3

3 4

2 5

выходные данные

2.000000000000000

Примечание

В первом примере путешествие может закончиться в городах 3 или 4 с равной вероятностью. Расстояние до городов 3 и 4 равно 1 и 2, соответственно, поэтому матожидание длины равно 1.5.

Во втором примере путешествие может закончиться в городах 4 или 5. Расстояние до обоих городов 2, поэтому матожидание равно 2.

**Решение задачи 4**

**Листинг 4**

def Task\_4(my\_dict):

my\_set = set()

res = 0

s = [(1, 1, 0)]

while s:

town, probability, depth = s.pop()

my\_set.add(town)

d\_counter = 0

for i in my\_dict[town]:

if i not in my\_set:

d\_counter += 1

if d\_counter == 0:

res += probability \* depth

else:

for i in my\_dict[town]:

if i not in my\_set:

s.append((i, probability / d\_counter, depth + 1))

if my\_set != set(my\_dict):

return False

return res

def main():

print('Входные данные:\n')

n = input('Введите n:')

if is\_num\_int(n) is False:

exit('n должно быть целым числом!')

n = int(n)

if double\_inequality\_validation(n, 1, 100000) is False:

exit('Значение числа n выходит за границы разрешенных!')

my\_dict = {x: [] for x in range(1, n + 1)}

for i in range(n - 1):

test = input('Введите u и v через пробел: ')

if mas\_length\_validation(test, 2) is False:

exit('Нужно ввести 2 переменных!')

if int\_mas\_validation(test) is False:

exit('Введенные переменные должы быть целочисленными!')

u, v = map(int, test.split())

if (double\_inequality\_validation(u, 1, n) is False) or\

(double\_inequality\_validation(v, 1, n) is False):

exit('Номера городов должны быть в диапазоне от 1 до n!')

if num1\_num2\_equal(u, v):

exit('u и v не должны быть равны!')

my\_dict[u].append(v)

my\_dict[v].append(u)

answer = Task\_4(my\_dict)

if answer is False:

exit('Есть город, которого нельзя достичь, передвигаясь по городам!')

print('\nВыходные данные:\n')

print('{:.15f}'.format(answer))

main()

# **Задача 5**

Дарк собирается посетить день рождения Мотарака. Дарк решил, что он хочет подарить Мотараку массив 𝑎, состоящий из 𝑛 неотрицательных целых чисел.

Дарк создал этот массив еще 1000 лет назад, поэтому некоторые его элементы пропали. Дарк знает, что Мотарак не любит, когда у двух соседних элементов большая абсолютная разница. У него не так много времени, поэтому он хочет выбрать целое число 𝑘 (0≤𝑘≤10^9) и заменить все пропущенные элементы в массиве 𝑎 числом 𝑘.

Обозначим за 𝑚 максимальную абсолютную разницу между всеми парами соседних элементов (то есть максимальное значение |𝑎𝑖−𝑎𝑖+1| по всем 1≤𝑖≤𝑛−1) в массиве 𝑎 после того, как Дарк заменит все пропавшие элементы на число 𝑘.

Дарк хочет выбрать число 𝑘 так, что 𝑚 будет минимально. Можете ли вы помочь ему?

Входные данные

Входные данные состоят из нескольких тестовых случаев. Первая строка содержит единственное целое число 𝑡 (1≤𝑡≤10^4) — количество тестовых случаев. Далее следует описание тестовых случаев в следующем формате.

В первой строке описания каждого тестового случае содержится одно целое число 𝑛 (2≤𝑛≤10^5) — размер массива 𝑎.

Во второй строке описания каждого тестового случая содержится 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (−1≤𝑎𝑖≤10^9). Если 𝑎𝑖=−1, то 𝑖-е число пропало. Гарантируется, что хотя бы одно число пропало в массиве в каждом тестовом случае.

Гарантируется, что сумма 𝑛 по всем тестовым случаям не превосходит 4⋅10^5.

Выходные данные

Выведите ответы для каждого тестового случая в следующем формате:

Вы должны вывести два целых числа, минимальное возможное значение 𝑚 и целое число 𝑘 (0≤𝑘≤10^9), которое делает максимальную абсолютную разницу между соседними элементами массива 𝑎 равной 𝑚.

Обратите внимание, что после замены всех пропавших элементов на 𝑘 максимальная абсолютная разница между всеми парами соседних элементов должна стать равной 𝑚.

Если существует более, чем одно подходящее 𝑘, вы можете вывести любое из них.

Пример

входные данные

7

5

-1 10 -1 12 -1

5

-1 40 35 -1 35

6

-1 -1 9 -1 3 -1

2

-1 -1

2

0 -1

4

1 -1 3 -1

7

1 -1 7 5 2 -1 5

выходные данные

1 11

5 35

3 6

0 42

0 0

1 2

3 4

Примечание

В первом тестовом случае, после замены всех пропавших элементов на 11 массив станет равным [11,10,11,12,11]. Абсолютная разница между любой парой соседних элементов равна 1. Выбрать такое значение 𝑘, чтобы максимальная абсолютная разница между всеми парами соседних элементов стала ≤0 невозможно, поэтому ответ равен 1.

В третьем тестовом случае, после замены всех пропавших элементов массива на 6 массив станет равным [6,6,9,6,3,6].

|𝑎1−𝑎2|=|6−6|=0;

|𝑎2−𝑎3|=|6−9|=3;

|𝑎3−𝑎4|=|9−6|=3;

|𝑎4−𝑎5|=|6−3|=3;

|𝑎5−𝑎6|=|3−6|=3.

Поэтому максимальная абсолютная разница между всеми парами соседних элементов будет равна 3.

**Решение задачи 5**

**Листинг 5**

def is\_lost\_num\_in\_mas(mas):

if mas.count(-1) == 0:

return False

return True

def Task\_5(n, a):

if a == [-1] \* n:

m = 0

k = 0

else:

mass = set(a) - set([-1])

maxx = max(mass)

minn = min(mass)

k = math.floor((maxx + minn) / 2)

if k > 10 \*\* 9:

k = 10 \*\* 9

m = -100000000

for i in range(n):

if a[i] == -1:

a[i] = k

if i != 0:

temp = abs(a[i-1]-a[i])

if temp > m:

m = copy(temp)

return m, k

def main():

print('Входные данные:\n')

t = input('Введите t:')

if is\_num\_int(t) is False:

exit('t должно быть целым числом!')

t = int(t)

if double\_inequality\_validation(t, 1, 10 \*\* 4) is False:

exit('Значение числа t выходит за границы разрешенных!')

n\_summ = 0

answers = []

for i in range(t):

n = input('Введите n: ')

if is\_num\_int(n) is False:

exit('n должно быть целым числом!')

n = int(n)

n\_summ += n

if inequality\_validation(n\_summ, 4 \* 10 \*\* 5) is False:

exit('сумма n по всем тестовым случаям не должна превосходить 2⋅10^5')

if double\_inequality\_validation(n, 2, 10 \*\* 5) is False:

exit('Значение числа n выходит за границы разрешенных!')

a = input('Введите элементы массива а через пробел: ')

if mas\_length\_validation(a, n) is False:

exit('Размер массива а не соответствует введеному числу n!')

if int\_mas\_validation(a) is False:

exit('Массив а должен состоять из целых чисел!')

a = list(map(int, a.split()))

if mas\_double\_inequality\_validation(a, -1, 10 \*\* 9) is False:

exit('Значения элементов массива а должны быть в диапазоне от -1 до 10^9!')

if is\_lost\_num\_in\_mas(a) is False:

exit('В массиве а дожно быть хотя бы одно число -1 ("пропавшее" число)!')

answer = Task\_5(n, a)

answers.append(answer)

print('\nВыходные данные:\n')

for ans in answers:

m, k = ans

print(m, k)

main()

# **Задача 6**

Вам задан массив 𝑎, состоящий из 𝑛 целых чисел.

Вы можете удалить не более одного элемента из этого массива. Таким образом, после этой операции длина массива равна либо 𝑛−1, либо 𝑛.

Ваша задача — посчитать максимально возможную длину строго возрастающего последовательного подмассива получившегося массива.

Напомним, что последовательным подмассивом 𝑎 с индексами от 𝑙 до 𝑟 называется 𝑎[𝑙…𝑟]=𝑎𝑙,𝑎𝑙+1,…,𝑎𝑟. Массив 𝑎[𝑙…𝑟] называется строго возрастающим, если 𝑎𝑙<𝑎𝑙+1<⋯<𝑎𝑟.

Входные данные

Первая строка входных данных содержит одно целое число 𝑛 (2≤𝑛≤2⋅10^5) — количество элементов в 𝑎.

Вторая строка входных данных содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤10^9), где 𝑎𝑖 равно 𝑖-му элементу 𝑎.

Выходные данные

Выведите одно целое число — максимально возможную длину строго возрастающего последовательного подмассива массива 𝑎 после удаления не более одного элемента.

Примеры

входные данные

5

1 2 5 3 4

выходные данные

4

входные данные

2

1 2

выходные данные

2

входные данные

7

6 5 4 3 2 4 3

выходные данные

2

Примечание

В первом примере можно удалить 𝑎3=5. Тогда получившийся массив будет равен [1,2,3,4] и длина его наибольшего возрастающего подмассива будет равна 4.

**Решение задачи 6**

**Листинг 6**

def Task\_6(n, a):

length = 0

len1 = 0

len2 = 0

isCur = False

isPrev = False

for i in range(1, n):

if a[i] <= a[i - 1]:

if isCur == True:

isCur = False

if isPrev == False:

length = max(length, len2)

if i != n - 1:

if a[i + 1] > a[i - 1]:

isPrev = True

len1 = len2

len2 = 0

elif a[i] > a[i - 2]:

isCur = True

else:

isCur = False

isPrev = False

length = max(length, max(len2, len1 + len2 - 1))

else:

if isCur == True:

len2 += 1

else:

len2 = 2

isCur = True

length = max(length, len2, len2 + len1 - 1)

return length

def main():

print('Входные данные:\n')

n = input('Введите n: ')

if is\_num\_int(n) is False:

exit('n должно быть целым числом!')

n = int(n)

if double\_inequality\_validation(n, 2, 2 \* 10 \*\* 5) is False:

exit('Значение числа n выходит за границы разрешенных!')

a = input('Введите элементы массива а через пробел: ')

if mas\_length\_validation(a, n) is False:

exit('Размер массива а не соответствует введеному числу n!')

if int\_mas\_validation(a) is False:

exit('Массив а должен состоять из целых чисел!')

a = list(map(int, a.split()))

if mas\_double\_inequality\_validation(a, 1, 10 \*\* 9) is False:

exit('Значения элементов массива а должны быть в диапазоне от 1 до 10^9!')

answer = Task\_6(n, a)

print('\nВыходные данные:\n')

print(answer)

main()

# **Задача 7**

У Коли есть массив целых чисел 𝑎1,𝑎2,…,𝑎𝑛. Числа в массиве могут быть и положительными, и отрицательными, но так как Коля не любит число 0, то таких чисел в его массиве нет.

Коле не нравится, что сумма элементов на некоторых подотрезках его массива может быть равна 0. Под подотрезком массива следует понимать некоторую непустую последовательность подряд идущих элементов массива.

Вы должны помочь Коле и сделать так, чтобы в его массиве не было подотрезков с суммой элементов равной 0. Для этого вы можете вставлять в его массив произвольные целые числа между любыми соседними элементами массива (числа могут быть любые: положительные, отрицательные, 0, любые по абсолютной величине, даже не представимые в стандартных типах данных во многих языках программирования).

Определите минимальное количество произвольных целых чисел, которые нужно добавить в массив Коли таким образом, чтобы после этого в массиве не было подотрезков с суммой элементов равной 0.

Входные данные

В первой строке следует целое число 𝑛 (2≤𝑛≤200000) — количество элементов в массиве Коли.

Во второй строке следует 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (−10^9≤𝑎𝑖≤10^9,𝑎𝑖≠0) — описание массива Коли.

Выходные данные

Выведите минимальное количество произвольных целых чисел, которые нужно добавить в массив Коли таким образом, чтобы после этого в массиве не было подотрезков с суммой элементов равной 0.

Примеры

входные данные

4

1 -5 3 2

выходные данные

1

входные данные

5

4 -2 3 -9 2

выходные данные

0

входные данные

9

-1 1 -1 1 -1 1 1 -1 -1

выходные данные

6

входные данные

8

16 -5 -11 -15 10 5 4 -4

выходные данные

3

Примечание

В первом примере есть один подотрезок, сумма элементов которого равна 0. Он начинается со второго элемента и заканчивается в четвертом элементе.

Достаточно добавить один элемент, чтобы в массиве не было подотрезков с суммой элементов равной нулю. Например, можно добавить число 1 между вторым и третьим элементами массива.

Во втором примере изначально нет подотрезков с суммой элементов равной 0, поэтому никаких чисел в массив добавлять не нужно.

**Решение задачи 7**

**Листинг 7**

def zero\_in\_mas\_validation(mas):

if mas.count(0) == 0:

return True

return False

def Task\_7(n, a):

res = 0

summ = 0

my\_set = {0}

for i in range(n):

summ += a[i]

if summ in my\_set:

my\_set = {0}

res += 1

summ = a[i]

my\_set.add(summ)

return res

def main():

print('Входные данные:\n')

n = input('Введите n: ')

if is\_num\_int(n) is False:

exit('n должно быть целым числом!')

n = int(n)

if double\_inequality\_validation(n, 2, 200000) is False:

exit('Значение числа n выходит за границы разрешенных!')

a = input('Введите элементы массива а через пробел: ')

if mas\_length\_validation(a, n) is False:

exit('Размер массива а не соответствует введеному числу n!')

if int\_mas\_validation(a) is False:

exit('Массив а должен состоять из целых чисел!')

a = list(map(int, a.split()))

if mas\_double\_inequality\_validation(a, -1 \* 10 \*\* 9, 10 \*\* 9) is False:

exit('Значения элементов массива а должны быть в диапазоне от -10^9 до 10^9!')

if zero\_in\_mas\_validation(a) is False:

exit('В массиве а не должно быть нулей!')

answer = Task\_7(n, a)

print('\nВыходные данные:\n')

print(answer)

main()

# **Задача 8**

В линию выстроены 𝑛 монстров, пронумерованных числами с 1 по 𝑛. У 𝑖-го монстра есть ℎ𝑖 очков здоровья (hp). Вы можете атаковать с силой, равной 𝑎 hp, а ваш соперник может атаковать с силой, равной 𝑏 hp.

Вы и ваш соперник сражаетесь с этими монстрами. Сначала вы и ваш соперник идете к первому монстру и сражаетесь с ним до тех пор, пока он не будет мертв, затем вы и ваш соперник отправляетесь ко второму монстру и сражаетесь с ним до тех пор, пока и он не будет мертв, и так далее. Смерть монстра наступает, когда значение его hp становится меньшим или равным 0.

Сражение с монстром происходит поочередно.

Вы атакуете монстра с силой 𝑎 hp. Если он погибает после вашего удара, вы получаете одно очко и затем вы и соперник переходите к следующему монстру. Ваш соперник атакует монстра с силой 𝑏 hp. Если монстр погибает после этой атаки, никто не получает очко и вы с соперником переходите к следующему монстру.

У вас есть некоторая секретная техника, с помощью которой можно заставить вашего соперника пропустить свой ход. Вы можете использовать эту технику не более, чем 𝑘 раз всего (например, если есть два монстра и 𝑘=4, то вы можете использовать эту технику 2 раза во время сражения с первым монстром и 1 один раз — со вторым, но не 2 раза с первым монстром и 3 раза со вторым).

Ваша задача — определить максимальное число очков, которое вы можете получить, если будете использовать секретную технику оптимально.

Входные данные

Первая строка входных данных содержит четыре целых числа 𝑛,𝑎,𝑏 and 𝑘 (1≤𝑛≤2⋅10^5,1≤𝑎,𝑏,𝑘≤10^9) — количество монстров, сила вашей атаки, сила атаки соперника и количество раз, которое вы можете использовать секретную технику.

Вторая строка входных данных содержит 𝑛 целых чисел ℎ1,ℎ2,…,ℎ𝑛 (1≤ℎ𝑖≤10^9), где ℎ𝑖 количество очков здоровья у 𝑖-го монстра.

Выходные данные

Выведите одно целое число — максимальное число очков, которое вы можете получить, если будете использовать секретную технику оптимально.

Примеры

входные данные

6 2 3 3

7 10 50 12 1 8

выходные данные

5

входные данные

1 1 100 99

100

выходные данные

1

входные данныеСкопировать 7 4 2 1

1 3 5 4 2 7 6

выходные данные

6

**Решение задачи 8**

**Листинг 8**

def how\_many\_hits(damage, hp):

if hp < damage:

return 1

elif hp % damage == 0:

return hp // damage

return hp // damage + 1

def Task\_8(n, a, b, k, h):

points = 0

""" Массив, очередным элементом котороого является кол-во применений

техники для получения очка за "спорного" монстрa"""

k\_mas = []

for i in range(n):

hp = h[i]

round\_damage = a + b

rounds = how\_many\_hits(round\_damage, hp)

if rounds == 1:

if hp - a <= 0:

points += 1

else:

a\_add\_hits\_needed = how\_many\_hits(a, hp - a)

k\_mas.append(a\_add\_hits\_needed)

else:

last\_round\_hp = hp - round\_damage \* (rounds - 1)

if last\_round\_hp - a <= 0:

points += 1

else:

a\_add\_hits\_needed = how\_many\_hits(a, last\_round\_hp - a)

k\_mas.append(a\_add\_hits\_needed)

k\_mas.sort()

while k >= 0 and k\_mas != []:

if k >= k\_mas[0]:

k -= k\_mas[0]

points += 1

del k\_mas[0]

else:

break

return points

def main():

print('Входные данные:\n')

nabk = input('Введите n, a, b, k через пробел:')

if mas\_length\_validation(nabk, 4) is False:

exit('Нужно ввести 4 переменных!')

if int\_mas\_validation(nabk) is False:

exit('Все переменные должны быть целочисленными!')

n, a, b, k = tuple(map(int, nabk.split()))

if double\_inequality\_validation(n, 1, 2 \* 10 \*\* 5) is False:

exit('Значение переменной n должно быть в диапазоне от 1 до 2 \* 10^5!')

if (double\_inequality\_validation(a, 1, 10 \*\* 9) is False) or\

(double\_inequality\_validation(b, 1, 10 \*\* 9) is False) or\

(double\_inequality\_validation(k, 1, 10 \*\* 9) is False):

exit('Значения переменных a, b, k должны быть в диапазоне от 1 до 10^9')

h = input('Введите элементы массива h (очки здоровья монстров) через пробел:')

if mas\_length\_validation(h, n) is False:

exit('Размер массива h не соответствует введеному числу n!')

if int\_mas\_validation(h) is False:

exit('Все переменные ассива h должны быть целочисленными!')

h = list(map(int, h.split()))

if mas\_double\_inequality\_validation(h, 1, 10 \*\* 9) is False:

exit('Значения элементов массива h должны быть в диапазоне от 1 до 10^9!')

answer = Task\_8(n, a, b, k, h)

print('\nВыходные данные:\n')

print(answer)

main()

# **Задача 9**

Олимпийские игры только что начались, и Федерико с нетерпением ждет начала марафонского забега.

В марафоне будут участвовать 𝑛 спортсменов, от 1 до 𝑛, и все они в прошлом участвовали в 5 важных марафонах, пронумерованных от 1 до 5. Для каждого 1≤𝑖≤𝑛 и 1≤𝑗≤5 Федерико помнит, что спортсмен 𝑖 занял место 𝑟𝑖,𝑗 в марафоне 𝑗 (например, 𝑟2,4=3 означает, что спортсмен 2 был третьим в марафоне 4).

Федерико считает, что спортсмен 𝑥 превосходит спортсмена 𝑦, если спортсмен 𝑥 занял лучшее место, чем спортсмен 𝑦, по крайней мере в 3 прошлых марафонах, т.е. 𝑟𝑥,𝑗<𝑟𝑦,𝑗 по крайней мере для 3 различных значений 𝑗.

Федерико считает, что у спортсмена хорошие шансы получить золотую медаль на Олимпийских играх, если он превосходит всех остальных спортсменов.

Найдите любого спортсмена, у которого хорошие шансы получить золотую медаль (то есть спортсмена, который превосходит всех остальных спортсменов), или определите, что такого спортсмена нет.

Входные данные

Первая строка содержит одно целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных. Затем следуют 𝑡 наборов входных данных.

Первая строка каждого набора входных данных содержит одно целое число 𝑛 (1≤𝑛≤50000) — количество спортсменов.

Затем следуют 𝑛 строк, каждая из которых описывает места одного спортсмена.

𝑖-я из этих строк содержит 5 целых чисел 𝑟𝑖,1,𝑟𝑖,2,𝑟𝑖,3,𝑟𝑖,4,𝑟𝑖,5 (1≤𝑟𝑖,𝑗≤50000) — места спортсмена 𝑖 в последних 5 марафонах.

Гарантируется, что сумма 𝑛 по всем наборам входных данных не превышает 50000.

Выходные данные

Для каждого набора входных данных выведите одно целое число — номер спортсмена, у которого хорошие шансы получить золотую медаль (то есть спортсмена, который превосходит всех остальных спортсменов). Если таких спортсменов нет, выведите −1. Если таких спортсменов больше, чем один, выведите любого из них.

Пример

входные данные

4

1

50000 1 50000 50000 50000

3

10 10 20 30 30

20 20 30 10 10

30 30 10 20 20

3

1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

6

9 5 3 7 1

7 4 1 6 8

5 6 7 3 2

6 7 8 8 6

4 2 2 4 5

8 3 6 9 4

выходные данные

1

-1

1

5

Примечание

Объяснение первого набора входных данных: Есть только один спортсмен, поэтому он превосходит всех остальных (так как больше никого нет), и поэтому у него хорошие шансы получить золотую медаль.

Пояснение второго набора входных данных: Есть 𝑛=3 спортсменов.

Спортсмен 1 превосходит спортсмена 2. Действительно, спортсмен 1 лучше спортсмена 2 в марафонах 1, 2 и 3.

Спортсмен 2 превосходит спортсмена 3. Действительно, спортсмен 2 превосходит спортсмена 3 в марафонах 1, 2, 4 и 5.

Спортсмен 3 превосходит спортсмена 1. Действительно, спортсмен 3 лучше, чем спортсмен 1 в марафонах 3, 4 и 5.

Объяснение третьего набора входных данных: Есть 𝑛=3 спортсменов.

Спортсмен 1 превосходит спортсменов 2 и 3. Поскольку он превосходит всех остальных спортсменов, у него хорошие шансы получить золотую медаль.

Спортсмен 2 превосходит спортсмена 3.

Спортсмен 3 не превосходит ни одного другого спортсмена.

Пояснение четвертого набора входных данных: Есть 𝑛=6 спортсменов.

Спортсмен 1 превосходит спортсменов 3, 4, 6.

Спортсмен 2 превосходит спортсменов 1, 4, 6.

Спортсмен 3 превосходит спортсменов 2, 4, 6.

Спортсмен 4 не превосходит ни одного другого спортсмена.

Спортсмен 5 превосходит спортсменов 1, 2, 3, 4, 6. Поскольку он превосходит всех остальных спортсменов, у него хорошие шансы получить золотую медаль.

Спортсмен 6 превосходит только спортсмена 4.

**Решение задачи 9**

**Листинг 9**

def compare\_sportsmen(s1, s2):

s1\_counter = 0

for i in range(5):

if s1[i] <= s2[i]:

s1\_counter += 1

if s1\_counter >= 3:

return s1

return s2

def Task\_9(n, r):

active\_winner = None

active\_winner\_num = -1

losers = []

for j in range(n):

sportsman = r.pop()

if active\_winner is None:

active\_winner = sportsman

active\_winner\_num = n - j

elif j == n - 1:

if compare\_sportsmen(active\_winner, sportsman) == active\_winner:

pass

else:

active\_winner = sportsman

active\_winner\_num = n - j

else:

if compare\_sportsmen(active\_winner, sportsman) == active\_winner:

losers.append(sportsman)

else:

losers.append(active\_winner)

active\_winner = sportsman

active\_winner\_num = n - j

flag = True

while flag and losers != []:

if compare\_sportsmen(active\_winner, losers[0]) != active\_winner:

flag = False

break

else:

del losers[0]

if flag:

return active\_winner\_num

else:

return -1

def main():

print('Входные данные:\n')

t = input('Введите t: ')

if is\_num\_int(t) is False:

exit('t должно быть целым числом!')

t = int(t)

if double\_inequality\_validation(t, 1, 1000) is False:

exit('Значение числа n выходит за границы разрешенных!')

answers = []

n\_summ = 0

for i in range(t):

r = []

n = input('Ввежите n: ')

if is\_num\_int(n) is False:

exit('n должно быть целым числом!')

n = int(n)

if double\_inequality\_validation(n, 1, 50000) is False:

exit('Значение числа n выходит за границы разрешенных!')

n\_summ += n

if inequality\_validation(n\_summ, 50000) is False:

exit('сумма n по всем тестовым случаям не должна превосходить 50000!')

for j in range(n):

sportsman = input('Введите массив, описывающий места спорстмена:')

if mas\_length\_validation(sportsman, 5) is False:

exit('Для каждого спортсмена должно быть введено 5 мест!')

if int\_mas\_validation(sportsman) is False:

exit('Каждое место спортсмена должно описываться целым числом!')

sportsman = list(map(int, sportsman.split()))

if mas\_double\_inequality\_validation(sportsman, 1, 50000) is False:

exit('Значения мест спортсменов должны быть в диапазоне от 1 до 50000!')

r.append(sportsman)

answer = Task\_9(n, r)

answers.append(answer)

print('\nВыходные данные:\n')

for ans in answers:

print(ans)

main()

# **Задача 10**

Лео-младший рисует в тетради в клеточку (каждая страница размечена квадратной сеткой). Можно считать, что страницы бесконечно большие в любом направлении.

Лео-младший закрашивает некоторые клетки на странице серым цветом. Он считает получившийся рисунок красивым, если выполнены следующие условия:

Рисунок связен, то есть, из любой серой клетки можно попасть в любую другую по цепочке из серых клеток, в которой каждые две последовательные клетки — соседи (то есть, имеют общую сторону).

Каждая серая клетка имеет чётное количество серых соседей. В рисунке ровно 𝑛 серых клеток, у которых все соседи серые. Количество остальных серых клеток может произвольным (но разумным, чтобы их можно было перечислить).

Лео-младший пытается нарисовать красивый рисунок для выбранного значения 𝑛. Помогите ему, найдите любой пример красивого рисунка.

Чтобы вывести координаты клеток в ответе, предполагайте, что на странице введена декартова система координат, в которой одна из клеток является началом координат (0,0), оси 0𝑥 и 0𝑦 перпендикулярны и сонаправлены линиям сетки, и шаг единичной длины в любом направлении вдоль любой оси ведёт в соседнюю клетку.

Входные данные

В единственной строке записано одно целое число 𝑛 (1≤𝑛≤500) — количество серых клеток, у которых все соседи серые.

Выходные данные

В первой строке выведите одно целое число 𝑘 — количество серых клеток в вашем рисунке. По техническим причинам 𝑘 не должно превосходить 5⋅10^5.

Каждая из следующих 𝑘 строк должна содержать две целых числа — координаты очередной серой клетки в вашем рисунке. Все перечисленные клетки должны быть различны, и рисунок должен отвечать всем требованиям, перечисленным выше. Все координаты не должны превосходить 10^9 по абсолютной величине.

Можно показать, что существует ответ, который удовлеторяет всем требованиям и укладывается в ограничение на 𝑘.

Пример

входные данные

4

выходные данные

12

1 0

2 0

0 1

1 1

2 1

3 1

0 2

1 2

2 2

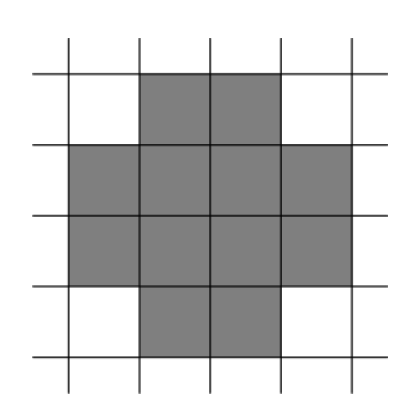
3 2

1 3

2 3

Примечание

Ответ для примера изображен ниже:



**Решение задачи 10**

**Листинг 10**

def Task\_10(n):

k = 3 \* n + 4

answers = [(0, 0), (0, 1), (1, 0), (1, 1)]

first = 2

last = n + 2

for coord in range(first, last):

answers.append((coord, coord))

answers.append((coord, coord - 1))

answers.append((coord - 1, coord))

return k, answers

def main():

print('Выходные данные:\n')

n = input('Введите n: ')

if is\_num\_int(n) is False:

exit('n должно быть целым числом!')

n = int(n)

if double\_inequality\_validation(n, 1, 500) is False:

exit('Значение числа n выходит за границы разрешенных!')

k, answers = Task\_10(n)

print('\nВыходные данные:\n')

print(k)

for ans in answers:

x, y = ans

print(x, y)

main()

# **Вывод**

Мною были решены все 10 задач для 4 варианта из задания на курсовую работу. Также в каждой задаче был произведен рефакторинг кода, реализована «проверка на дурака». Курсовая работа успешно выполнена.