**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»

Выполнила: студентка группы БФИ 2001

Ахматова А.И

Проверил: Мкртчян Г.М.

Вариант №2

Москва 2022

**Оглавление**

[Цель работы 4](#_Toc103105440)

[Задача №1 4](#_Toc103105441)

[Условие задачи 4](#_Toc103105442)

[Решение задачи 5](#_Toc103105443)

[Задача №2 5](#_Toc103105444)

[Условие задачи 5](#_Toc103105445)

[Решение задачи 7](#_Toc103105446)

[Задача №3 8](#_Toc103105447)

[Условие задачи 8](#_Toc103105448)

[Решение задачи 9](#_Toc103105449)

[Задача №4 10](#_Toc103105450)

[Условие задачи 10](#_Toc103105451)

[Решение задачи 12](#_Toc103105452)

[Задача №5 13](#_Toc103105453)

[Условие задачи 13](#_Toc103105454)

[Решение задачи 15](#_Toc103105455)

[Задача №6 16](#_Toc103105456)

[Условие задачи 16](#_Toc103105457)

[Решение задачи 17](#_Toc103105458)

[Задача №7 18](#_Toc103105459)

[Условие задачи 18](#_Toc103105460)

[Решение задачи 19](#_Toc103105461)

[Задача №8 20](#_Toc103105462)

[Условие задачи 20](#_Toc103105463)

[Решение задачи 22](#_Toc103105464)

[Задача №9 22](#_Toc103105465)

[Условие задачи 22](#_Toc103105466)

[Решение задачи 24](#_Toc103105467)

[Задача №10 24](#_Toc103105468)

[Условие задачи 24](#_Toc103105469)

[Решение задачи 26](#_Toc103105470)

[Вывод 26](#_Toc103105471)

[Приложение А 26](#_Toc103105472)

# **Цель работы**

Необходимо найти решение для 10 задач в соответствии с 2 Вариантом.

# **Задача №1**

## **Условие задачи**

Вам задано целое положительное число 𝑛. За один ход вы можете увеличить 𝑛 на единицу (то есть сделать 𝑛:=𝑛+1). Ваша задача — найти минимальное количество ходов, которое надо совершить, чтобы сделать сумму цифр 𝑛 не превышающей 𝑠.

Вам необходимо ответить на 𝑡 независимых наборов тестовых данных.

**Входные данные**

Первая строка входных данных содержит одно целое число 𝑡 (1≤𝑡≤2⋅104) — количество наборов тестовых данных. Затем следуют 𝑡 наборов тестовых данных.

Единственная строка набора тестовых данных содержит два целых числа 𝑛 и 𝑠 (1≤𝑛≤1018; 1≤𝑠≤162).

**Выходные данные**

На каждый набор тестовых данных выведите ответ: минимальное количество ходов, которое надо совершить, чтобы сделать сумму цифр 𝑛 не превышающей 𝑠.

**Пример**

**входные данные**

**5**

**2 1**

**1 1**

**500 4**

**217871987498122 10**

**100000000000000001 1**

**выходные данные**

**8**

**0**

**500**

**2128012501878**

**899999999999999999**

## **Решение задачи**

Решение задачи представлено на рисунке 1. Код программы в текстовом виде представлен в приложении А.

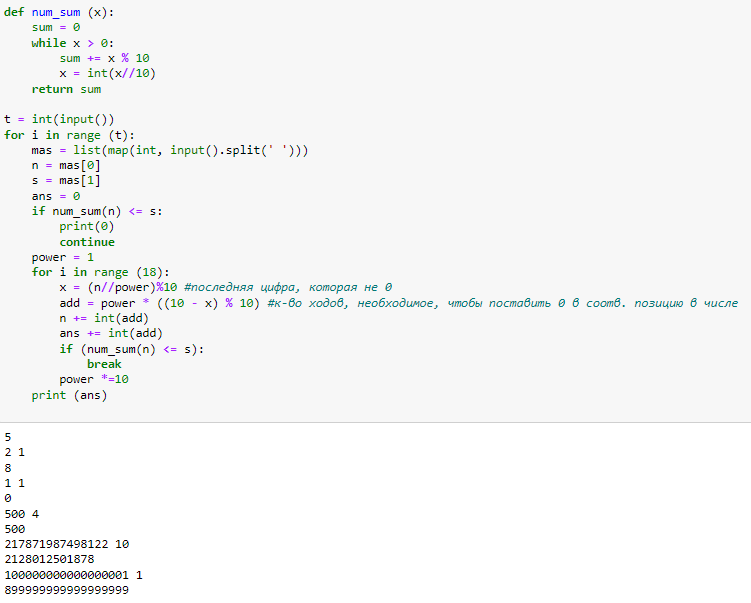


Рисунок 1 – решение задачи 1

На рисунке 2 представлен результат работы программы.

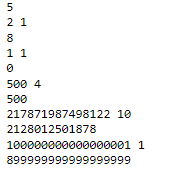


Рисунок 2 – Результат работы программы 1

# **Задача №2**

## **Условие задачи**

Назовем левым циклическим сдвигом некоторой строки 𝑡1𝑡2𝑡3…𝑡𝑛−1𝑡𝑛 следующую строку: 𝑡2𝑡3…𝑡𝑛−1𝑡𝑛𝑡1.

Аналогично, назовем правым циклическим сдвигом строки 𝑡 строку 𝑡𝑛𝑡1𝑡2𝑡3…𝑡𝑛−1.

Скажем, что строка 𝑡 является хорошей, если ее левый циклический сдвиг равен правому циклическому сдвигу.

Вам дана строка 𝑠, состоящая из цифр 0–9.

Какое минимальное количество символов необходимо удалить из строки 𝑠, чтобы она стала хорошей?

**Входные данные**

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

Следующие 𝑡 строк содержат описание наборов входных данных. Первая и единственная строка каждого набора содержит строку 𝑠 (2≤|𝑠|≤2⋅105). Каждый символ 𝑠𝑖 является цифрой 0–9.

Гарантируется, что суммарная длина строк не превышает 2⋅105.

**Выходные данные**

Для каждого набора входных данных выведите минимальное количество символов, которое необходимо удалить из строки 𝑠, чтобы она стала хорошей.

**Пример**

**входные данные**

3

95831

100120013

252525252525

**выходные данные**

3

5

0

Гарантируется, что суммарная длина строк не превышает 2⋅105.

Выходные данные

**Примечание**

В первом примере можно стереть любые 3 символа, например 1-й, 3-й и 4-й. Вы получите строку 51, и это хорошая строка.

Во втором примере можно стереть все символы, кроме 0: оставшаяся строка 0000 — хорошая.

В третьем примере заданная строка 𝑠 уже является хорошей.

## **Решение задачи**

Решение задачи представлено на рисунке 3. Код программы в текстовом виде представлен в приложении А.

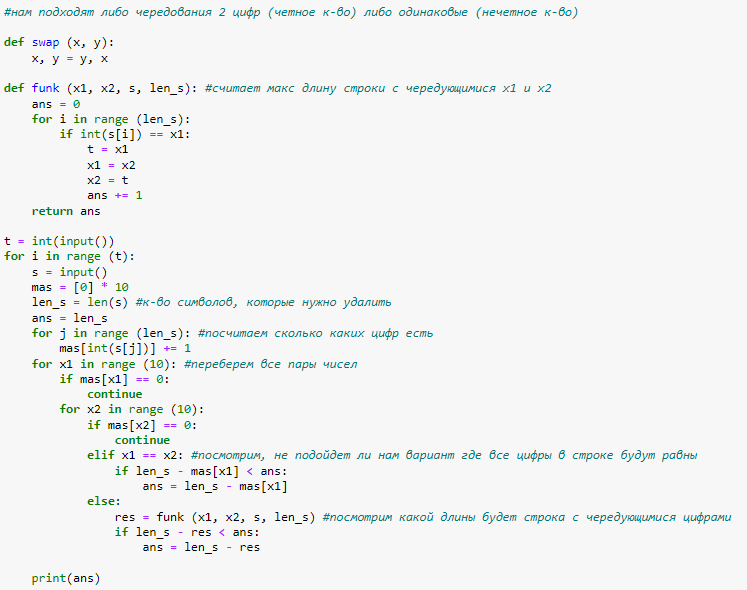


Рисунок 3 – решение задачи 2

На рисунке 4 представлен результат работы программы.

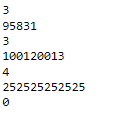


Рисунок 4 – Результат работы программы 2

# **Задача №3**

## **Условие задачи**

Вам задана бинарная строка 𝑠, состоящая из 𝑛 нулей или единиц.

Ваша задача – разделить заданную строку на минимальное число подпоследовательностей таким образом, что каждый символ строки принадлежит ровно одной подпоследовательности и каждая подпоследовательность выглядит подобно «010101 ...» или «101010 ...» (т.е. подпоследовательность не должна содержать два соседних нуля или единицы).

Напомним, что подпоследовательность — это последовательность, которая может быть получена путем удаления из заданной последовательности с помощью удаления нуля или более элементов без изменения порядка остальных элементов. Например, подпоследовательностями «1011101» являются «0», «1», «11111», «0111», «101», «1001», но не «000», «101010» и «11100».

Вам необходимо ответить на 𝑡 независимых наборов тестовых данных.

**Входные данные**

Первая строка теста содержит одно целое число 𝑡 (1≤𝑡≤2⋅104) — количество наборов тестовых данных. Затем следуют 𝑡 наборов тестовых данных.

Первая строка набора тестовых данных содержит одно целое число 𝑛 (1≤𝑛≤2⋅105) — длину 𝑠. Вторая строка набора тестовых данных содержит 𝑛 символов '0' и '1' — строку 𝑠.

Гарантируется, что сумма всех 𝑛 не превосходит 2⋅105 (∑𝑛≤2⋅105).

**Выходные данные**

Для каждого набора тестовых данных выведите ответ на него: первой строкой выведите одно целое число 𝑘 (1≤𝑘≤𝑛) — минимальное количество подпоследовательностей, на которые вы можете разделить строку 𝑠. Второй строкой выведите 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤𝑘), где 𝑎𝑖 — номер подпоследовательности, к которой принадлежит -й символ строки 𝑠.

Если существует несколько ответов, вы можете вывести любой.

**Пример**

**входные данные**

4

4

0011

6

111111

5

10101

8

01010000

**выходные данные**

2

1 2 2 1

6

1 2 3 4 5 6

1

1 1 1 1 1

4

1 1 1 1 1 2 3 4

## **Решение задачи**

Решение задачи представлено на рисунке 5. Код программы в текстовом виде представлен в приложении А.

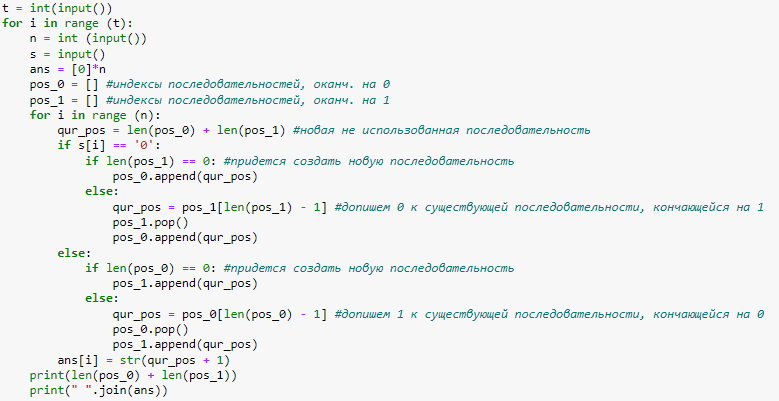


Рисунок 5 – решение задачи 3

На рисунке 6 представлен результат работы программы.

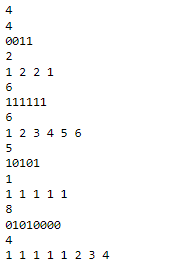


Рисунок 6 – Результат работы программы 3

# **Задача №4**

## **Условие задачи**

Патрик любит играть в бейсбол, но иногда он тратит так много часов на пробежки, что его разум начинает затуманиваться! Патрик уверен, что его набранные очки за 𝑛 игр соответствуют тождественной перестановке (т.е. в первой игре он набирает 1, во второй игре он набирает 2 и так далее). Однако, когда он посмотрел на свои записи, он увидел, что все значения перепутаны!

Определим специальный обмен следующим образом: выберите любой подмассив очков и переставьте местами его элементы так, чтобы ни один элемент не оказался в той же позиции, где он был до обмена. Например, выполнение специального обмена на [1,2,3] может дать [3,1,2], но не может дать [3,2,1], так как 2 находится в той же позиции.

Вам дана перестановка из 𝑛 целых чисел. Пожалуйста, помогите Патрику найти минимальное количество специальных обменов, необходимых для того, чтобы сделать ее отсортированной! Можно доказать, что при данных ограничениях это число не превышает 1018.

Массив 𝑎 является подмассивом массива 𝑏, если 𝑎 можно получить из 𝑏, удалив несколько (возможно, ноль или все) элементов из начала и несколько (возможно, ноль или все) элементов с конца.

**Входные данные**

Каждый тест содержит несколько наборов входных данных. Первая строка содержит количество наборов входных данных 𝑡 (1≤𝑡≤100). Описание наборов входных данных приведено ниже.

Первая строка каждого набора входных данных содержит целое число 𝑛 (1≤𝑛≤2⋅105) — длину данной перестановки.

Вторая строка каждого набора входных данных содержит 𝑛 целых чисел 𝑎1,𝑎2,...,𝑎𝑛 (1≤𝑎𝑖≤𝑛) — начальную перестановку.

Гарантируется, что сумма 𝑛 по всем наборам входных данных не превышает 2⋅105.

**Выходные данные**

Для каждого набора входных данных выведите одно целое число: минимальное количество специальных обменов, необходимых для сортировки перестановки.

**Пример**

**входные данные**

2

5

1 2 3 4 5

7

3 2 4 5 1 6 7

**выходные данные**

0

2

**Примечание**

Первая перестановка она уже отсортирована, поэтому обмены не нужны.

Можно показать, что для сортировки второй перестановки нужно как минимум 2 обмена.

[3,2,4,5,1,6,7]

Сделаем специальный обмен для диапазона (1,5)

[4,1,2,3,5,6,7]

Сделаем специальный обмен для диапазона (1,4)

[1,2,3,4,5,6,7]

## **Решение задачи**

Решение задачи представлено на рисунке 7. Код программы в текстовом виде представлен в приложении А.

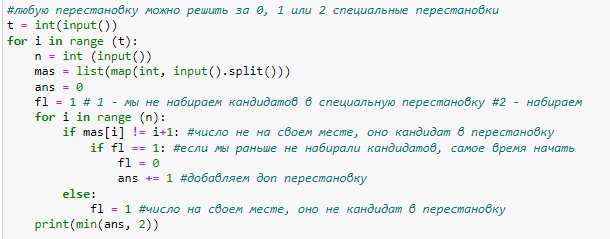


Рисунок 7 – решение задачи 4

На рисунке 8 представлен результат работы программы.

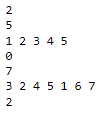


Рисунок 8 – Результат работы программы 4

# **Задача №5**

## **Условие задачи**

Вам дана строка 𝑠[1…𝑛], состоящая из строчных латинских букв. Гарантируется, что 𝑛=2𝑘 для некоторого целого числа 𝑘≥0.

Строка 𝑠[1…𝑛] называется -хорошей, если выполняется как минимум одно из следующих условий:

Длина строки 𝑠 равна 1 и она состоит из единственного символа 𝑐 (то есть 𝑠1=𝑐);

Длина строки 𝑠 больше 1, первая половина строки состоит только из символа 𝑐 (то есть 𝑠1=𝑠2=⋯=𝑠𝑛2=𝑐), а вторая половина строки (то есть строка 𝑠𝑛2+1𝑠𝑛2+2…𝑠𝑛) является (𝑐+1)-хорошей строкой;

Длина строки 𝑠 больше 1, вторая половина строки состоит только из символа 𝑐 (то есть 𝑠𝑛2+1=𝑠𝑛2+2=⋯=𝑠𝑛=𝑐), а первая половина строки (то есть строка 𝑠1𝑠2…𝑠𝑛2) является (𝑐+1)-хорошей строкой.

Например: «aabc» является 'a'-хорошей, «ffgheeee» является 'e'-хорошей.

За один ход вы можете выбрать один индекс 𝑖 от 1 до 𝑛 и заменить 𝑠𝑖 на любую строчную латинскую букву (любой символ от 'a' до 'z').

Ваша задача — найти минимальное количество ходов, необходимое, чтобы получить 'a'-хорошую строку из 𝑠 (т.е. -хорошую строку для 𝑐= 'a'). Гарантируется, что ответ всегда существует.

Вам нужно ответить на 𝑡 независимых наборов тестовых данных.

Еще один пример 'a'-хорошей строки является следующим. Например, рассмотрим строку 𝑠=«cdbbaaaa». Это 'a'-хорошая строка, потому что:

вторая половина строки («aaaa») состоит только из символов 'a';

первая половина строки («cdbb») — 'b'-хорошая строка, потому что:

вторая половина строки («bb») состоит только из символов 'b';

первая половина строки («cd») — 'c'-хорошая строка, потому что:

первая половина строки («c») состоит из единственного символа 'c';

вторая половина строки («d») — 'd'-хорошая строка.

**Входные данные**

Первая строка теста содержит одно целое число 𝑡 (1≤𝑡≤2⋅104) — количество наборов тестовых данных. Затем следуют 𝑡 наборов тестовых данных.

Первая строка набора тестовых данных содержит одно целое число 𝑛 (1≤𝑛≤131 072) — длину 𝑠. Гарантируется, что 𝑛=2𝑘 для некоторого целого числа 𝑘≥0. Вторая строка набора тестовых данных содержит строку 𝑠, состоящую из 𝑛 строчных латинских букв.

Гарантируется, что сумма всех 𝑛 не превосходит 2⋅105 (∑𝑛≤2⋅105).

**Выходные данные**

Для каждого набора тестовых данных выведите ответ на него — наименьшее количество ходов, необходимое, чтобы получить 'a'-хорошую строку 𝑠 (т.е. -хорошую строку для 𝑐= 'a'). Гарантируется, что ответ существует.

**Пример**

**входные данные**

6

8

bbdcaaaa

8

asdfghjk

8

ceaaaabb

8

bbaaddcc

1

z

2

ac

**выходные данные**

0

7

4

5

1

1

## **Решение задачи**

Решение задачи представлено на рисунке 9. Код программы в текстовом виде представлен в приложении А.

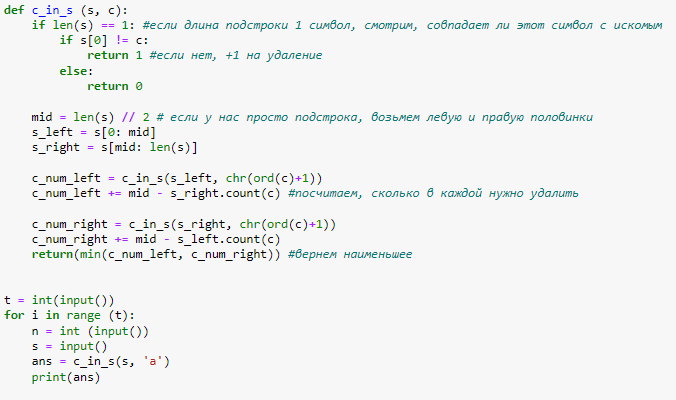


Рисунок 9 – решение задачи 5

На рисунке 10 представлен результат работы программы.

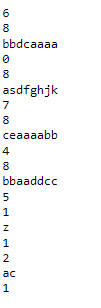


Рисунок 10 – Результат работы программы 5

# **Задача №6**

## **Условие задачи**

Рассмотрим следующий процесс. У вас есть бинарная строка (строка, состоящая только из символов 0 и 1) 𝑤 длины 𝑛 и число 𝑥. Вы создаете новую бинарную строку 𝑠 длины 𝑛; -й символ новой строки 𝑠 выбирается следующим образом:

если символ 𝑤𝑖−𝑥 существует и равен 1, то 𝑠𝑖 равно 1 (более формально, если 𝑖>𝑥 и 𝑤𝑖−𝑥= 1, то 𝑠𝑖= 1);

если символ 𝑤𝑖+𝑥 существует и равен 1, то 𝑠𝑖 равно 1 (более формально, если 𝑖+𝑥≤𝑛 и 𝑤𝑖+𝑥= 1, то 𝑠𝑖= 1);

если оба описанных выше условия неверны, то 𝑠𝑖 равно 0.

Вам заданы число 𝑥 и строка 𝑠. Восстановите изначальную строку 𝑤.

**Входные данные**

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

Каждый набор входных данных содержит две строки. Первая строка содержит строку 𝑠 (2≤|𝑠|≤105, каждый символ строки 𝑠 равен либо 0, либо 1). Вторая строка содержит целое число 𝑥 (1≤𝑥≤|𝑠|−1).

Суммарная длина всех длин строк 𝑠 во входных данных не превосходит 105.

**Выходные данные**

На каждый набор входных данных выведите ответ:

если не существует строки 𝑤, которая может породить строку 𝑠, то выведите −1;

иначе, выведите бинарную строку 𝑤 состоящую из |𝑠| символов. Если возможных ответов несколько — вы можете вывести любой из них. Пример

**входные данные**

3

101110

2

01

1

110

1

**выходные данные**

111011

10

-1

## **Решение задачи**

Решение задачи представлено на рисунке 11. Код программы в текстовом виде представлен в приложении А.

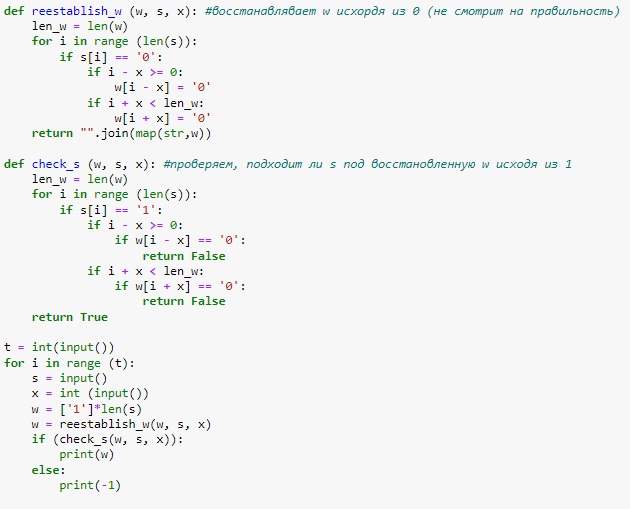


Рисунок 11 – решение задачи 6

На рисунке 12 представлен результат работы программы.

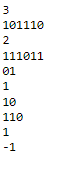


Рисунок 12 – Результат работы программы 6

# **Задача №7**

## **Условие задачи**

В линию выстроены 𝑛 настоек, причем настойка 1 находится слева, а настойка 𝑛 — справа. Каждая настойка увеличит ваше здоровье на 𝑎𝑖, если ее выпить. 𝑎𝑖 может быть отрицательным, что означает, что настойка уменьшит ваше здоровье.

Вы начинаете с 0 здоровья и будете идти слева направо, от первой настойки до последней. Для каждой настойки вы можете выбрать, выпить ли ее. Вы должны следить за тем, чтобы ваше здоровье всегда было неотрицательным.

Какое наибольшее количество настоек вы можете выпить?

**Входные данные**

Первая строка содержит одно целое число 𝑛 (1≤𝑛≤2000) — количество настоек.

Следующая строка содержит 𝑛 целых чисел 𝑎1, 𝑎2, ... ,𝑎𝑛 (−109≤𝑎𝑖≤109), которые обозначают изменения здоровья после употребления данных настоек.

**Выходные данные**

Выведите одно целое число — максимальное количество настоек, которое вы можете выпить, чтобы ваше здоровье всегда было неотрицательным.

**Пример**

**входные данные**

6

4 -4 1 -3 1 -3

**выходные данные**

5

**Примечание**

В примере, вы можете выпить 5 настоек, приняв настойки 1, 3, 4, 5 и 6.\ Невозможно выпить все 6 настоек, потому что в какой-то момент ваше здоровье станет отрицательным.

## **Решение задачи**

Решение задачи представлено на рисунке 13. Код программы в текстовом виде представлен в приложении А.

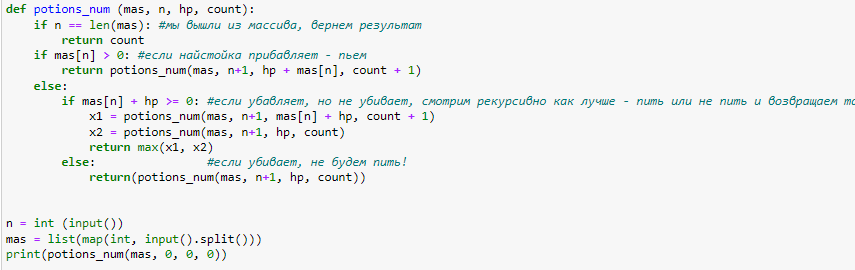


Рисунок 13 – решение задачи 7

На рисунке 14 представлен результат работы программы.



Рисунок 14 – Результат работы программы 7

# **Задача №8**

## **Условие задачи**

Задан массив 𝑎=[𝑎1,𝑎2,…,𝑎𝑛] (1≤𝑎𝑖≤𝑛). Его элемент 𝑎𝑖 называется особым, если существует такая пара индексов 𝑙 и 𝑟 (1≤𝑙<𝑟≤𝑛), что 𝑎𝑖=𝑎𝑙+𝑎𝑙+1+…+𝑎𝑟. Иными словами, элемент называется особым, если он представим в виде суммы двух или более подряд идущих элементов массива (не важно, особых или нет).

Выведите количество особых элементов заданного массива 𝑎.

Например, если 𝑛=9 и 𝑎=[3,1,4,1,5,9,2,6,5], то ответ равен 5:

𝑎3=4 — особый элемент, так как 𝑎3=4=𝑎1+𝑎2=3+1;

𝑎5=5 — особый элемент, так как 𝑎5=5=𝑎2+𝑎3=1+4;

𝑎6=9 — особый элемент, так как 𝑎6=9=𝑎1+𝑎2+𝑎3+𝑎4=3+1+4+1;

𝑎8=6 — особый элемент, так как 𝑎8=6=𝑎2+𝑎3+𝑎4=1+4+1;

𝑎9=5 — особый элемент, так как 𝑎9=5=𝑎2+𝑎3=1+4.

Обратите внимание, что среди элементов массива 𝑎 могут быть равные — если несколько элементов равны и являются особыми, то все они должны быть посчитаны в ответе.

Входные данные

В первой строке записано целое число 𝑡 (1≤𝑡≤1000) — количество наборов входных данных в тесте. Далее следуют описания 𝑡 наборов входных данных.

Каждый набор задается двумя строками. В первой строке записано целое число 𝑛 (1≤𝑛≤8000) — длина массива 𝑎. Во второй строке записаны целые числа 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤𝑛).

Гарантируется, что сумма значений 𝑛 по всем наборам входных данных не превосходит 8000.

Выходные данные

Выведите 𝑡 чисел — количества особых элементов для каждого из заданных массивов.

Пример

входные данные

5

9

3 1 4 1 5 9 2 6 5

3

1 1 2

5

1 1 1 1 1

8

8 7 6 5 4 3 2 1

1

1

выходные данные

5

1

0

4

0

## **Решение задачи**

Решение задачи представлено на рисунке 15. Код программы в текстовом виде представлен в приложении А.

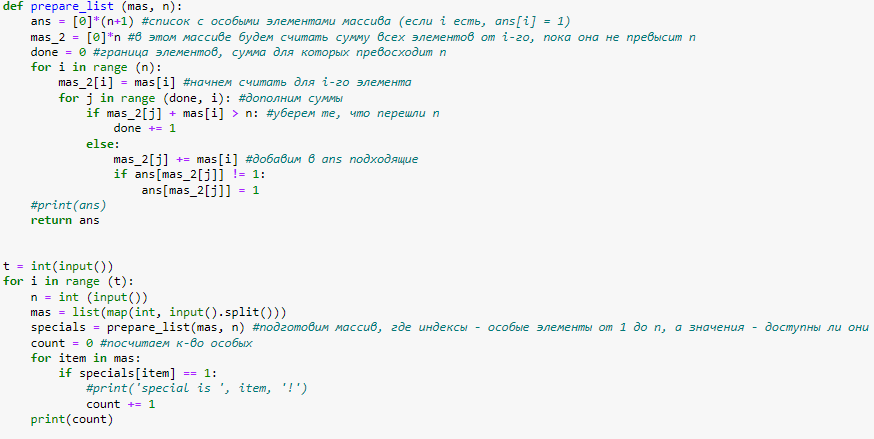


Рисунок 15 – решение задачи 8

На рисунке 16 представлен результат работы программы.

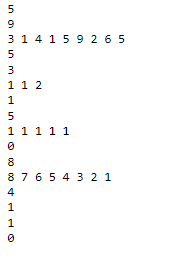


Рисунок 16 – Результат работы программы 8

# **Задача №9**

## **Условие задачи**

Вам дан массив 𝑎1,𝑎2,…,𝑎𝑛.

За одну операцию вы можете взять любые два элемента 𝑎𝑖 и 𝑎𝑗 (𝑖≠𝑗) и уменьшить каждый из них на единицу.

Вам нужно проверить можно ли сделать все элементы массива равными нулю или нет.

**Входные данные**

Первая строка содержит одно целое число 𝑛 (2≤𝑛≤105) — размер массива.

Вторая строка содержит 𝑛 целых чисел 𝑎1,𝑎2,…,𝑎𝑛 (1≤𝑎𝑖≤109) — элементы массива.

**Выходные данные**

Выведите «YES», если можно сделать все элементы массива равными нулю. Иначе выведите «NO».

**Примеры**

**входные данные**

4

1 1 2 2

**выходные данные**

YES

**входные данные**

6

1 2 3 4 5 6

**выходные данные**

NO

**Примечание**

В первом примере можно сделать все элементы равными нулю за 3 операции:

Уменьшить 𝑎1 и 𝑎2,

Уменьшить 𝑎3 и 𝑎4,

Уменьшить 𝑎3 и 𝑎4

Во втором примере можно показать, что сделать все элементы равными нулю нельзя.

## **Решение задачи**

Решение задачи представлено на рисунке 17. Код программы в текстовом виде представлен в приложении А.

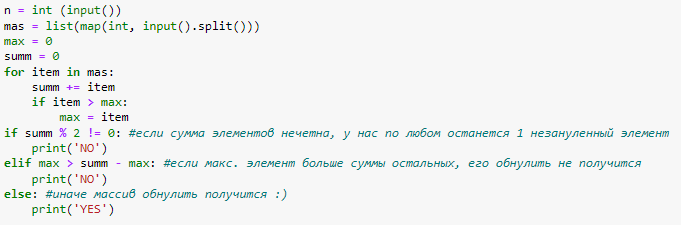


Рисунок 17 – решение задачи 9

На рисунке 18 представлен результат работы программы.



Рисунок 18 – Результат работы программы 9

# **Задача №10**

## **Условие задачи**

Вам задан тетраэдр. Обозначим его вершины буквами A, B, C и D соответственно (рисунок 19).

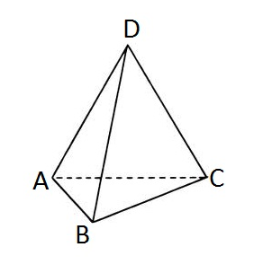


Рисунок 19 - тетраэдр

В вершине тетраэдра D находится муравей. Муравей очень подвижный и не любит стоять на месте. В каждый момент времени он совершает один шаг от одной вершины к другой по некоторому ребру тетраэдра, оставаться на месте он не может.

От Вас в этой задаче требуется совсем немногое: нужно посчитать каким количеством способов муравей может прийти из исходной вершины D в себя ровно за n шагов. Другими словами, Вас просят узнать количество различных циклических путей длины n из вершины D в себя. Поскольку это количество может быть достаточно большим, ответ требуется посчитать по модулю

1000000007 (109 + 7).

Входные данные В первой строке записано единственное целое число n (1 ≤ n ≤ 107) — требуемая длина циклического пути.

Выходные данные

Выведите единственное целое число — искомое количество способов по модулю

1000000007 (109 + 7).

Примеры

входные данные

2

выходные данные

3

входные данные

4

выходные данные

21

Примечание

Искомые пути в первом примере:

D - A - D

D - B - D

D - C - D

## **Решение задачи**

Решение задачи представлено на рисунке 20. Код программы в текстовом виде представлен в приложении А.

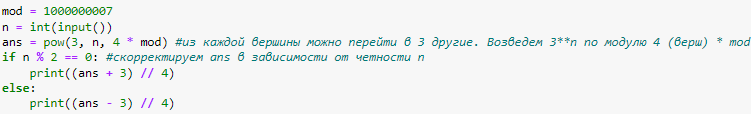


Рисунок 20 – решение задачи 10

На рисунке 21 представлен результат работы программы.



Рисунок 21 – Результат работы программы 10

# **Вывод**

В ходе выполнения данной курсовой работы были разработаны программы для решения 10 задач в соответствии с условием 2 Варианта.

# **Приложение А**

Код программы 1:

def num\_sum (x):

sum = 0

while x > 0:

sum += x % 10

x = int(x//10)

return sum

t = int(input())

for i in range (t):

mas = list(map(int, input().split(' ')))

n = mas[0]

s = mas[1]

ans = 0

if num\_sum(n) <= s:

print(0)

continue

power = 1

for i in range (18):

x = (n//power)%10 #последняя цифра, которая не 0

add = power \* ((10 - x) % 10) #к-во ходов, необходимое, чтобы поставить 0 в соотв. позицию в числе

n += int(add)

ans += int(add)

if (num\_sum(n) <= s):

break

power \*=10

print (ans)

Код программы 2:

def swap (x, y):

x, y = y, x

def funk (x1, x2, s, len\_s): #считает макс длину строки с чередующимися х1 и х2

ans = 0

for i in range (len\_s):

if int(s[i]) == x1:

t = x1

x1 = x2

x2 = t

ans += 1

return ans

t = int(input())

for i in range (t):

s = input()

mas = [0] \* 10

len\_s = len(s) #к-во символов, которые нужно удалить

ans = len\_s

for j in range (len\_s): #посчитаем сколько каких цифр есть

mas[int(s[j])] += 1

for x1 in range (10): #переберем все пары чисел

if mas[x1] == 0:

continue

for x2 in range (10):

if mas[x2] == 0:

continue

elif x1 == x2: #посмотрим, не подойдет ли нам вариант где все цифры в строке будут равны

if len\_s - mas[x1] < ans:

ans = len\_s - mas[x1]

else:

res = funk (x1, x2, s, len\_s) #посмотрим какой длины будет строка с чередующимися цифрами

if len\_s - res < ans:

ans = len\_s - res

print(ans)

Код программы 3:

t = int(input())

for i in range (t):

n = int (input())

s = input()

ans = [0]\*n

pos\_0 = [] #индексы последовательностей, оканч. на 0

pos\_1 = [] #индексы последовательностей, оканч. на 1

for i in range (n):

qur\_pos = len(pos\_0) + len(pos\_1) #новая не использованная последовательность

if s[i] == '0':

if len(pos\_1) == 0: #придется создать новую последовательность

pos\_0.append(qur\_pos)

else:

qur\_pos = pos\_1[len(pos\_1) - 1] #допишем 0 к существующей последовательности, кончающейся на 1

pos\_1.pop()

pos\_0.append(qur\_pos)

else:

if len(pos\_0) == 0: #придется создать новую последовательность

pos\_1.append(qur\_pos)

else:

qur\_pos = pos\_0[len(pos\_0) - 1] #допишем 1 к существующей последовательности, кончающейся на 0

pos\_0.pop()

pos\_1.append(qur\_pos)

ans[i] = str(qur\_pos + 1)

print(len(pos\_0) + len(pos\_1))

print(" ".join(ans))

Код программы 4:

#любую перестановку можно решить за 0, 1 или 2 специальные перестановки

t = int(input())

for i in range (t):

n = int (input())

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

ans = 0

fl = 1 # 1 - мы не набираем кандидатов в специальную перестановку #2 - набираем

for i in range (n):

if mas[i] != i+1: #число не на своем месте, оно кандидат в перестановку

if fl == 1: #если мы раньше не набирали кандидатов, самое время начать

fl = 0

ans += 1 #добавляем доп перестановку

else:

fl = 1 #число на своем месте, оно не кандидат в перестановку

print(min(ans, 2))

Код программы 5:

def c\_in\_s (s, c):

if len(s) == 1: #если длина подстроки 1 символ, смотрим, совпадает ли этот символ с искомым

if s[0] != c:

return 1 #если нет, +1 на удаление

else:

return 0

mid = len(s) // 2 # если у нас просто подстрока, возьмем левую и правую половинки

s\_left = s[0: mid]

s\_right = s[mid: len(s)]

c\_num\_left = c\_in\_s(s\_left, chr(ord(c)+1))

c\_num\_left += mid - s\_right.count(c) #посчитаем, сколько в каждой нужно удалить

c\_num\_right = c\_in\_s(s\_right, chr(ord(c)+1))

c\_num\_right += mid - s\_left.count(c)

return(min(c\_num\_left, c\_num\_right)) #вернем наименьшее

t = int(input())

for i in range (t):

n = int (input())

s = input()

ans = c\_in\_s(s, 'a')

print(ans)

Код программы 6:

def reestablish\_w (w, s, x): #восстанавлявает w исхордя из 0 (не смотрит на правильность)

len\_w = len(w)

for i in range (len(s)):

if s[i] == '0':

if i - x >= 0:

w[i - x] = '0'

if i + x < len\_w:

w[i + x] = '0'

return "".join(map(str,w))

def check\_s (w, s, x): #проверяем, подходит ли s под восстановленную w исходя из 1

len\_w = len(w)

for i in range (len(s)):

if s[i] == '1':

if i - x >= 0:

if w[i - x] == '0':

return False

if i + x < len\_w:

if w[i + x] == '0':

return False

return True

t = int(input())

for i in range (t):

s = input()

x = int (input())

w = ['1']\*len(s)

w = reestablish\_w(w, s, x)

if (check\_s(w, s, x)):

print(w)

else:

print(-1)

Код программы 7:

def potions\_num (mas, n, hp, count):

if n == len(mas): #мы вышли из массива, вернем результат

return count

if mas[n] > 0: #если найстойка прибавляет - пьем

return potions\_num(mas, n+1, hp + mas[n], count + 1)

else:

if mas[n] + hp >= 0: #если убавляет, но не убивает, смотрим рекурсивно как лучше - пить или не пить и возвращаем max

x1 = potions\_num(mas, n+1, mas[n] + hp, count + 1)

x2 = potions\_num(mas, n+1, hp, count)

return max(x1, x2)

else: #если убивает, не будем пить!

return(potions\_num(mas, n+1, hp, count))

n = int (input())

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

print(potions\_num(mas, 0, 0, 0))

Код программы 8:

def prepare\_list (mas, n):

ans = [0]\*(n+1) #список с особыми элементами массива (если i есть, ans[i] = 1)

mas\_2 = [0]\*n #в этом массиве будем считать сумму всех элементов от i-го, пока она не превысит n

done = 0 #граница элементов, сумма для которых превосходит n

for i in range (n):

mas\_2[i] = mas[i] #начнем считать для i-го элемента

for j in range (done, i): #дополним суммы

if mas\_2[j] + mas[i] > n: #уберем те, что перешли n

done += 1

else:

mas\_2[j] += mas[i] #добавим в ans подходящие

if ans[mas\_2[j]] != 1:

ans[mas\_2[j]] = 1

#print(ans)

return ans

t = int(input())

for i in range (t):

n = int (input())

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

specials = prepare\_list(mas, n) #подготовим массив, где индексы - особые элементы от 1 до n, а значения - доступны ли они в последовательности

count = 0 #посчитаем к-во особых

for item in mas:

if specials[item] == 1:

#print('special is ', item, '!')

count += 1

print(count)

Код программы 9:

n = int (input())

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

max = 0

summ = 0

for item in mas:

summ += item

if item > max:

max = item

if summ % 2 != 0: #если сумма элементов нечетна, у нас по любом останется 1 незануленный элемент

print('NO')

elif max > summ - max: #если макс. элемент больше суммы остальных, его обнулить не получится

print('NO')

else: #иначе массив обнулить получится :)

print('YES')

Код программы 10:

mod = 1000000007

n = int(input())

ans = pow(3, n, 4 \* mod) #из каждой вершины можно перейти в 3 другие. Возведем 3\*\*n по модулю 4 (верш) \* mod

if n % 2 == 0: #скорректируем ans в зависимости от четности n

print((ans + 3) // 4)

else:

print((ans - 3) // 4)