

Лабораторная работа №4

В рамках данной лабораторной работы необходимо решить предложенные задачи на языке программирования высокого уровня из предложенного перечня. Варианты задач находятся в отдельном файле. В качестве результата выполнения практической работы необходимо приложить **архив с файлами решенных задач, а также отчет о выполнении работы**. Файлы решенных задач представляют собой файлы с исходным кодом с установленным расширением. При решении задач необходимо писать свои собственные структуры данных и методы сортировки, использование встроенных не допускается (кроме массивов).

Отчет о выполнении практической работы должен содержать:

1. Титульный лист

2. Содержание

3. 4 параграфа, в которых раскрыто решение каждой задачи. По каждой задаче необходимо представить следующую информацию:

3.1. *Условие задачи*. Берется из файла.

3.2. *Ход решения задачи*. Студент описывает логику решения данной задачи. Какие алгоритмы и для чего использованы, как построена программа. Данная часть является описательной. Здесь следует говорить именно о построении алгоритма, опуская процессы ввода и вывода данных (если это не является основной сутью алгоритма).

3.3. *Листинг программы с комментариями*. Копируется весь программный код.

3.4. *Тестирование программы*. Если для задачи предусмотрены автотесты, приложить скриншот их прохождения. Если нет: Составляется таблица, содержащая следующие поля: номер теста, входные данные, результат выполнения программы, корректность, время выполнения (мс), затраченная память (Мб). Составляется не менее 10-ти тестовых наборов данных согласно условию задачи. Тестовые наборы входных данных студент составляет самостоятельно. В обязательном порядке программа тестируется на граничных наборах входных данных (например, если N варьируется от 0 до 10^9 , то обязательно рассмотреть решение задачи при $N=0$ и при N близком к 10^9). Если написанная программа не позволяет решить задачу при граничных входных данных, все равно включить в тест и в качестве результата написать "Не решено". В столбце "входные данные" данные впечатываются вручную, в столбце "результат..." представляется скриншот выполнения программы (если не влезает на одну страницу, делать несколько скриншотов).

Каждая лабораторная работа защищается на занятии преподавателю.

Задачи по теме 4. Деревья

Задача 1. Провода

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

На складе есть провода различной целочисленной длины. Их можно разрезать на части. Необходимо получить K кусочков одинаковой целочисленной и как можно большей длины. Найти максимальную длину M , при которой можно получить по меньшей мере K кусочков этой длины. Все оставшиеся на складе куски проводов длиной меньше M в подсчете не участвуют.

Формат входных данных:

В первой строке – количество проводов на складе N и необходимое количество кусочков K . В следующих N строках – длины проводов.

$$1 \leq N \leq 100000$$

$$1 \leq K \leq 100000$$

Формат выходных данных:

M – максимальная длина, на которую можно разрезать все провода так, чтобы получилось не менее K кусочков.

Примеры:

Стандартный ввод	Стандартный вывод
5 7 15 12 5 13 6	6

Задача 2. Брокеры

Ограничение по времени: 1 секунда

Ограничение по памяти: 64 мегабайта

В стране Бурляндии фирма «Котлетный рай» имеет много отделений, работающих сравнительно автономно. После неких экономических преобразований такая форма функционирования оказалась невыгодной и фирма решила сливать капиталы отделений, образуя укрупненные департаменты, отвечающие за несколько отделений сразу. Цель фирмы – слить все отделения в один громадный департамент, владеющий всеми капиталами. Проблема заключается в том, что по законам Бурляндии операция слияния капиталов должна проводиться государственной брокерской службой, которая не может производить более одной операции слияния в одной фирме одновременно. Вторая проблема состоит в том, что брокерская служба берет за свои услуги один процент всех средств, получившихся в результате слияния двух подразделений. Важно спланировать порядок операций слияния таким образом, чтобы фирма потратила на слияние как можно меньшую сумму.

Формат входных данных:

На вход программы подается число отделение $2 \leq N \leq 1000000$, за которым следует N капиталов отделений $1 \leq C_i \leq 1000000$.

Формат выходных данных:

T – минимальная сумма из возможных, которую должна заплатить брокерам фирма «Котлетный рай», с двумя знаками после запятой.

Примеры:

Стандартный ввод	Стандартный вывод
5 1 2 3 4 5	0.33
10 2 10 100 30 7 4 15 2 15 80	6.52

Задача 3. Пересечение множеств

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

Задано $2 \leq N \leq 1000$ множеств из $3 \leq M \leq$ элементов, значения которых могут находиться в диапазоне от -2000000000 до 2000000000. Значения каждого множества задаются в произвольном порядке. Гарантируется, что для одного множества все задаваемые значения различные.

Требуется найти наибольший размер множества, получаемого при пересечении какой-либо из пар из заданных множеств.

Формат входных данных:

N M

A_1[1] A_1[2] ... A_1[M]

A_2[1] A_2[2] ... A_2[M]

...

A_N[1] A_N[2] ... A_N[M]

Формат выходных данных:

Одно целое число.

Примеры:

Стандартный ввод	Стандартный вывод
3 4 9 7 1 8 5 7 6 3	2

5 9 8 6	
4 5 -2 6 8 4 -1 5 3 10 -5 -1 7 8 -5 -1 -2 -1 8 4 9 0	3

Задача 4. Составные строки

Ограничение по времени: 0.5 секунд

Ограничение по памяти: 16 мегабайт

В первой строке входного файла содержится число $4 \leq N \leq 1000$, последующие N строк состоят только из заглавных букв латинского алфавита и образуют множество, то есть, равных элементов среди них нет. Длина строки не превышает 1000 букв. Некоторые из этих строк можно составить, прописав друг за другом каких-то два элемента множества, возможно, совпадающие. То есть, если исходное множество содержит элементы A, B, AB, AA, C, ABC, то элемент AB можно составить из элементов A и B, а строку AA – из элементов A и A.

Ваша задача – вывести все элементы множества, которые можно составить из пары элементов множества по одному на строку в лексикографически возрастающем порядке.

Примеры:

Стандартный ввод	Стандартный вывод
5 A AB B AA ABC	AA AB
10 ABC DEFG AB ABCAB DEFGA FG ABFG ABCAFG FGFG ABABC	ABCAB ABFG FGFG

Задача 5. Ксерокопии

Ограничение по времени: 0,2 секунды

Ограничение по памяти: 16 мегабайт

Секретарша Людочка сегодня опоздала на работу и ей срочно нужно успеть к обеду сделать N копий одного документа. В ее распоряжении имеются два ксерокса, один из которых копирует лист за x секунд, а другой – за y секунд. (Разрешается использовать как

один ксерокс, так и оба одновременно. Можно копировать не только с оригинала, но и с копии.) Помогите ей выяснить, какое минимальное время для этого потребуется.

Формат входных данных:

Во входном файле INPUT.TXT записаны три натуральных числа N , x и y , разделенные пробелом ($1 \leq N \leq 2 \cdot 10^8$, $1 \leq x, y \leq 10$).

Формат выходных данных:

В выходной файл OUTPUT.TXT выведите одно число – минимальное время в секундах, необходимое для получения N копий.

Пример

INPUT.TXT	OUTPUT.TXT
4 1 1	3
5 1 2	4

Источник: [здесь](#).

Задача 6. Роман в томах

Ограничение по времени: 1 секунда

Ограничение по памяти: 16 мегабайт

В романе N глав. В i -той главе a_i страниц. Требуется издать роман в K томах так, чтобы объем самого «толстого» тома был минимален. В каждом томе главы располагаются по порядку своих номеров.

Требуется написать программу, которая найдет количество страниц в самом «толстом» томе.

Формат входных данных:

Входной текстовый файл INPUT.TXT содержит в первой строке число N ($1 \leq N \leq 100$). Во второй строке через пробел записаны N чисел – количество страниц в каждой главе. Количество страниц в романе не превышает 32767. В третьей строке записано число K ($1 \leq K \leq N$).

Формат выходных данных:

Выходной файл OUTPUT.TXT должен содержать количество страниц в самом «толстом» томе.

Пример

INPUT.TXT	OUTPUT.TXT
3 1 2 1 2	3
4 1 2 1 1 3	2

Источник: [здесь](#).

Задача 7. Запросы сумм

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

В первой строке файла содержатся два числа: количество элементов в массиве V : $10 \leq N \leq 500000$ и количество запросов $1 \leq M \leq 500000$. Каждый элемент массива лежит в интервале $[0 \dots 2^{32}]$.

Каждый запрос – отдельная строка, состоящая из кода запроса, который может быть равен 1 или 2 и аргументов запроса.

Запрос с кодом один содержит два аргумента, начало L и конец отрезка R массива. В ответ на этот запрос программа должна вывести значения суммы элементов массива от V[L] до V[R] включительно.

Запрос с кодом два содержит тоже два аргумента, первый из которых есть номер элемента массива V, а второй – его новое значение.

Количество выведенных строк должно совпадать с количеством запросов первого типа.

Пример

Стандартный ввод	Стандартный вывод
10 8	93
1	39
7	70
15	49
8	38
9	
15	
15	
19	
5	
19	
1 1 8	
1 6 8	
1 0 6	
2 6 6	
2 1 6	
2 0 9	
1 4 7	
1 3 6	

Задача 8. Поиск множеств

Ограничение по времени: 1 секунда

Ограничение по памяти: 64 мегабайта

В первой строке файла содержится три числа: N – количество эталонных множеств, M – размер каждого из множеств и K – количество пробных множеств.

Каждое из множеств содержит целые числа от 0 до 10^9 , числа могут повторяться.

Требуется для каждого из пробных множеств вывести в отдельной строке цифру '1', если это множество в точности совпадает с каким-либо из эталонных множеств и цифру '0', если оно ни с одним не совпадает, то есть выведено должно быть в точности K строк.

$$5 \leq N \leq 50000$$

$$3 \leq M \leq 1000$$

$$5 \leq K \leq 50000$$

Примеры:

Стандартный ввод	Стандартный вывод
------------------	-------------------

10 3 5	1
6 5 1	1
7 9 3	0
2 3 2	0
7 2 9	1
9 6 2	
6 6 6	
9 4 1	
8 4 4	
8 3 2	
1 2 6	
9 7 2	
1 6 5	
3 7 7	
4 4 6	
3 9 7	
10 7 5	1
8 4 0 3 6 9 2	1
3 5 0 4 3 1 1	1
7 1 0 3 1 2 4	1
7 1 5 1 5 5 1	0
3 4 0 0 3 4 0	
3 3 3 6 3 9 3	
3 4 1 3 1 8 1	
1 1 6 8 6 8 2	
5 6 8 1 3 9 3	
7 5 7 1 4 0 3	
1 1 3 3 8 4 1	
2 1 1 6 6 8 8	
1 1 5 7 5 1 5	
3 4 1 3 1 1 8	
0 0 1 2 8 2 6	

Задача 9. Телефонная книга

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

Необходимо разработать программу, которая является промежуточным звеном в реализации телефонной книги. На вход подается $N \leq 1000$ команд вида

ADD User Number

DELETE User

EDITPHONE User Number

PRINT User

Согласно этим командам нужно соответственно добавить пользователя в телефонную книгу, удалить пользователя, изменить его номер и вывести на экран его

данные. В случае невозможности выполнить действие, необходимо вывести **ERROR**.
Добавлять пользователя, уже существующего в телефонной книге, нельзя.

Необходимо вывести протокол работы телефонной книги

Формат входных данных:

См. пример.

Формат выходных данных:

В случае невозможности выполнения действия требуется вывести **ERROR**

В случае команды **PRINT** User

User Number

Примеры:

Стандартный ввод	Стандартный вывод
9 ADD IVAN 1178927 PRINT PETER ADD EGOR 123412 PRINT IVAN EDITPHONE IVAN 112358 PRINT IVAN PRINT EGOR DELETE EGOR EDITPHONE EGOR 123456	ERROR IVAN 1178927 IVAN 112358 EGOR 123412 ERROR

Задача 10. Анаграммы

Ограничение по времени: 0.5 секунд

Ограничение по памяти: 256 мегабайт

Как известно, анаграммами называются слова, которые могут получиться друг из друга путем перестановки букв, например LOOP, POOL, POLO. Будем называть все слова такого рода *комплект*.

На вход программы подается число слов $1 \leq N \leq 100000$. В каждой из очередных N строк присутствует одно слово, состоящее из заглавных букв латинского алфавита. Все слова имеют одинаковую длину $3 \leq L \leq 10000$.

Требуется определить число комплектов во входном множестве.

Формат входных данных:

N

W1

W2

...

WN

Формат выходных данных:

NumberOfComplexes

Примеры:

Стандартный ввод	Стандартный вывод
8 BCB ABA BCB BAA BBC CCB CBC CBC	3

Задача 11. Кеширующий сервер

Ограничение по времени: 3 секунды

Ограничение по памяти: 64 мегабайта

Дормидонт работает в компании, которая занимается обработкой больших данных. Обработываемые данные находятся где-то в распределенной системе. Количество различных данных в системе ограничено и каждое данное имеет свой номер. Эти данные регулярно требуются различным клиентам и, поскольку время обращения к ним достаточно велико, для ускорения обработки информации Дормидонту поручено написать часть *middleware* — сервер-посредник, к которому и обращаются теперь клиенты за данными. Так как система — распределенная, а сервер — нет, все требуемые данные на сервер не помещаются, но он имеет возможность запоминать результаты своих запросов к распределенной системе. Для этого на сервере выделена ограниченная память на N запросов. Важно, что клиент не имеет возможности обращаться к распределенной системе и результат запроса к распределенной системе всегда должны оказаться на сервере.

К большой радости Дормидонта оказалось, что самые крупные и значимые клиенты всегда обращаются за одними и теми же данными в одном и том же порядке, так что у него есть последовательность запросов. Дормидонт придумал такой алгоритм, что как можно большее количество запросов исполняется из кеша сервера, без обращения к распределенной системе. Придумаете ли вы что-то подобное?

Формат входных данных:

На вход программы подается размер памяти под кеширование запросов $1 \leq N \leq 100000$, количество запросов $1 \leq M \leq 100000$ и ровно M запросов с номерами $0 \leq R_i \leq 10^{18}$. Количество различных номеров запросов ограничено и не превосходит 100000.

Формат выходных данных:

Требуется вывести одно число: сколько раз пришлось обратиться к распределенной системе за данными, отсутствующими в кеше. В начале работы кеш пуст.

Пример

стандартный ввод	стандартный вывод
5 15 3 1 4 1 5	9

9	
2	
6	
5	
3	
5	
8	
7	
9	
3	

Замечание

В приведенном примере первые три запроса произойдут к данным под номерами 3, 1 и 4, так как их нет в кеше. Следующий запрос, 1, есть в кеше и обращения к распределенной системе не произойдет. Запросы 5 и 9 занесут их в кеш. Следующий запрос — 2 — в кеше отсутствует, но мы выкинем из кеша запрос 1 и запрос 2 займет его место. Далее, запрос 6 вытеснит из кеша значение 2 (у нас есть информация о дальнейших запросах и из нее мы видим, что запрос под номером 2 больше не повториться и нет причин хранить его далее), после чего следующие три запроса удовлетворятся из кеша. Затем произойдет еще два вытеснения — 8 и 7. Итого 9 обращений к распределенной системе. Нетрудно установить, что меньше сделать нельзя.