Лабораторная работа №2

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 методы сортировки, использование встроенных не допускается.

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 2. Алгоритмы сортировки

Задача 1. Поиск в сломанном массиве

Ограничение по времени: 0.001 с. Ограничение по памяти: 64 Mb.

Алла ошиблась при копировании из одной структуры данных в другую. Она хранила массив чисел в кольцевом буфере. Массив был отсортирован по возрастанию, и в нем можно было найти элемент за логарифмическое время. Алла скопировала данные из кольцевого буфера в обычный массив, но сдвинула данные исходной отсортированной последовательности. Теперь массив не является отсортированным Тем не менее, нужно обеспечить возможность находить внем элемент за O($\log n$).

Можно предполагать, что в массиве только уникальные элементы.

Задачу необходимо сдавать, выбрав компилятор Make! Решение отправляется файлом. Требуемые сигнатуры функций лежат в заготовках кода на диске.

От вас требуется реализовать функцию, осуществляющую поиск в сломанном массиве. Файлы с заготовками кода, содержащими сигнатуры Функций и базовый тест для поддерживаемых языков, находятся на Яндекс Диске по ссылке. Обратите внимание, что считывать данные и выводить ответ не требуется

Разрешение файла должно соответствовать языку на котором вы пишете (.cpp, .java, .go, .js, .py). Если вы пишете на Java, назовите файл с решением Solution.java, для C# — Solution.cs. Для остальных языков название может быть любым, кроме solution.ext где ext — разрешение для вашего языка.

Формат входных данных:

Функция принимает массив натуральных чисел и искомое число k. Длина массива не превосходит 10000. Элементы массива и число k не превосходят по значению 10000.

В примерах:

В первой строке записано число n — длина массива.

Во второй строке записано положительное число k — искомый элемент

Далее в строку через пробел записано n натуральных чисел. каждое из которых не превосходит 200000

Формат выходных данных:

Функция должна вернуть индекс элемента, равного k, если такой есть в массиве (нумерация с нуля). Если элемент не найден, функция должна вернуть — 1.

Изменять массив нельзя.

Для отсечения неэффективных решений ваша функция будет запускаться от 100000 до 1000000 раз.

Примеры:

Стандартный ввод	Стандартный вывод
2 5 19 21 100 101 1 4 5 7 12	6
2 1 5 1	1

Задача 2. Эффективная быстрая сортировка

Ограничение по времени: 3 с. Ограничение по памяти: 64 Mb.

Тимофей решил организовать соревнование по спортивному программированию, чтобы найти талантливых стажёров. Задачи подобраны,

участники зарегистрированы, тесты написаны. Осталось придумать, как конце соревнования будет определяться победитель.

Каждый участник имеет уникальный логин. Когда соревнование закончится, к нему будут привязаны два показателя: количество решённых задач P_i размер штрафа F_i, Штраф начисляется за неудачные попытки и время, затраченное на задачу.

Тимофей решил сортировать таблицу результатов следующим образом: при сравнении двух участников выше будет идти тот, у которого решено больше задач. При равенстве числа решенных задач первым идет участник с меньшим штрафом. Если же и штрафы совпадают, то первым будет тот. у которого логин идёт раньше в алфавитном (лексикографическом) порядке.

Тимофей заказал толстовки для победителей и накануне поехал за ними в магазин. В своё отсутствие он поручил вам реализовать алгоритм быстрой сортировки (англ. quick sort) для таблицы результатов. Так как Тимофей любит спортивное программирование и не любит зря расходовать оперативную память, то ваша реализация сортировки не может потреблять O(n) дополнительной памяти для промежуточных данных (такая модификация быстрой сортировки называется "in-place”).

Как работает in-place quick sort

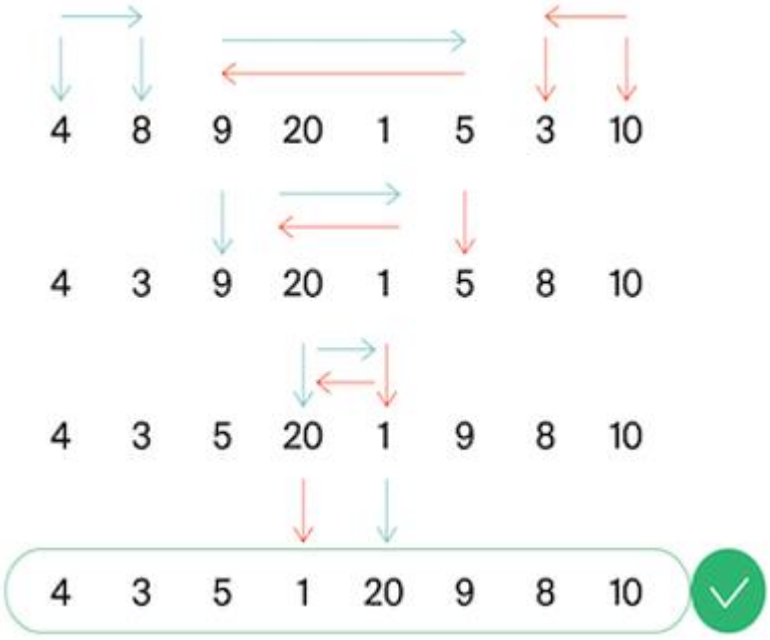
Как и в случае обычной быстрой сортировки, которая использует дополнительную память, необходимо выбрать опорный элемент (англ. pivot), а затем переупорядочить массив. Сделаем так, чтобы сначала шли элементы, не превосходящие опорного, а затем — большие опорного.

Затем сортировка вызывается рекурсивно для двух полученных частей. Именно на этапе разделения элементов на группы в обычном алгоритме используется дополнительная память. Теперь разберёмся, как реализовать этот in-place.

Пусть мы как-то выбрали опорный элемент. Заведем два указателя left и right, которые изначально будут указывать на левый и правый концы отрезка

соответственно. Затем будем двигать первый указатель вправо до тех пор. пока он указывает на элемент, меньший опорного. Аналогично двигаем правый указатель влево, пока он стоит на элементе, превосходящем опорный. В итоге окажется, что что левее от left все элементы точно принадлежат первой группе, а правее от right — второй. Элементы, на которых стоят указатели, нарушают порядок. Поменяем их местами (в большинстве языков программирования используется функция swap() и продвинем указатели на следующие элементы. Будем повторять это действие до тех пор, пока left и right не столкнутся.

На рисунке представлен пример разделения при pivot=5. Указатель left — голубой, right — оранжевый.



Формат входных данных:

В первой строке задано число участников n, $1 \leq n \leq 100000$.

В каждой из следующих n строк задана информация про одного из участников.

i-й участник описывается тремя параметрами:

- уникальным логином (строкой из маленьких латинских букв длиной не более 20)
- числом решённых задач P_i
- штрафом F_i

F_i и P_i – целые числа, лежащие в диапазоне от 0 до 10^9.

Формат выходных данных:

Для отсортированного списка участников выведите по порядку их логины по одному в строке.

Примеры:

Стандартный ввод	Стандартный вывод
5 alla 4 100 gena 6 1000 gosha 2 90 rita 2 90 Timofey 4 80	gena timofey alla gosha rita
5 alla 0 0 gena 0 0 gosha 0 0 rita 0 0 Timofey 0 0	alla gena gosha rita timofey

Задача 3. Максимальная тройка

Ограничение по времени: 1.5 с. Ограничение по памяти: 8 Mb.

Имеется не более 1000000 целых чисел, каждое из которых лежит в диапазоне от -1000000 до 1000000. Найти максимально возможное значение произведений любых трех различных по номерам элементов массива.

Формат входных данных:

N
A1
A2
…
AN

Формат выходных данных:

MaxPossibleProduct

Примеры:

Стандартный ввод	Стандартный вывод
10 -1 2 3 -4 -2 5 -1 5 -3 -2	75

Задача 4. Сортировка по многим полям

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

В базе данных хранится N записей, вида (Name, a_1, a_2, …, a_k) – во всех записях одинаковое число параметров. На вход задачи подается приоритет полей – перестановка на числах 1, …, k – записи нужно вывести по невозрастанию в соответствии с этим приоритетом. В случае, если приоритет полей таков: 3 4 2 1, то это следует воспринимать так: приоритет значений из 3 колонки самый высокий, приоритет значений из колонки 4 ниже, приоритет значений из колонки 2 еще ниже, а приоритет значений из колонки 1 самый низкий.

Формат входных данных:

$N \leq 1000$

k: $1 \leq k \leq 10$

p_1 p_2 … p_k – перестановка на k числах, разделитель – пробел

N строк вида

Name a_1 a_2 … a_k

Формат выходных данных:

N строк с именами в порядке, согласно приоритету

Примеры:

Стандартный ввод	Стандартный вывод
3 3 2 1 3 A 1 2 3 B 3 2 1 C 3 1 2	B A C

Замечание. Так как колонка под номером 2 самая приоритетная, то переставить записи можно только двумя способами: (A, B, C) и (B, A, C). Следующий по приоритетности столбец – первый, и он позволяет выбрать из возможных перестановок только (B, A, C). Так как осталась ровно одна перестановка, третий приоритет не имеет значения.

Задача 5. Оболочка.

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

Имеется массив из N целочисленных точек на плоскости.

Требуется найти периметр наименьшего охватывающего многоугольника, содержащего все точки.

Формат входных данных:

N

x1 y1

x2 y2

…

xn yn

$5 \leq N \leq 500000$

$-10000 \leq x_i, y_i \leq 10000$

Формат выходных данных:

Одно вещественное число – периметр требуемого многоугольника с двумя знаками после запятой.

Примеры:

Стандартный ввод	Стандартный вывод
5 2 1 2 2 2 3 3 2 1 2	5.66

Задача 6. Очень быстрая сортировка.

Ограничение по времени: 1.5 с. Ограничение по памяти: 512 Mb.

Имеется рекуррентная последовательность A1, A2, …, AN, строящаяся по следующему правилу:

$A1 = K$

$A_{i+1} = (A_i \times M) \% (2^{32} - 1) \% L$

Требуется найти сумму всех нечетных по порядку элементов в отсортированной по неубыванию последовательности по модулю L.

Для входных данных

5 7 13 100

последовательность будет такой:

{7; 7 × 13%100 = 91; 91 × 13%100 = 83; 83 × 13%100 = 79; 79 × 13%100 = 27}, то есть {10; 91; 83; 79; 27}.

Отсортированная последовательность {7; 27; 79; 83; 91}.

Сумма элементов на нечетных местах = (7 + 79 + 91)%100 = 77.

Формат входных данных:

N K M L

$5000000 \le N \le 60000000$, $0 \le K, L, M \le 2^{32} - 1$

Формат выходных данных:

RESULT

Примеры:

Стандартный ввод	Стандартный вывод
5 7 13 100	77

Замечание. Для представления элементов последовательности необходимо использовать тип данных unsigned int.

Для получения массива используйте цикл:

```
a[0] = K;
for (int i = 0; i < N-1; i++)
a[i+1] = (unsigned int) ((a[i]*unsigned long long)M)&0xFFFFFFFFU)%L;
```

Внимание! Изменение типа данных и/или метода генерации элементов массива может привести (и на различных компиляторах приводит) к другой последовательности!

Задача 7. Внешняя сортировка.

Ограничение по времени: 2 с. Ограничение по памяти: 2 Mb.

В файле «input.txt» содержатся строки символов, длина каждой строки не превышает 10000 байт. Файл нужно отсортировать в лексикографическом порядке и вывести результат в файл «output.txt». Вот беда, файл занимает много мегабайт, а в Вашем распоряжении оказывается вычислительная система с очень маленькой оперативной памятью. Но файл должен быть отсортирован!

Примеры:

input.txt	output.txt
qwertyuiopasdffghhj qpoiuytredgfhfd asdfghjjklvcvx alkjghcdysdfgsr pquytrgsdjdsa akjhfgdghshhfuushvdfs	akjhfgdghshhfuushvdfs alkjghcdysdfgsr asdfghjjklvcvx pquytrgsdjdsa qpoiuytredgfhfd qwertyuiopasdffghhj

Задача 8. Музей.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

В музее регистрируется в течение суток время прихода и ухода каждого посетителя. Таким образом, за день получены N пар значений, где первое значение в паре показывает время прихода посетителя и второе значение - время его ухода. Требуется найти максимальное число посетителей, которые находились в музее одновременно.

Формат входных данных:

В первой строке входного файла INPUT.TXT записано натуральное число N ($N < 10^5$) – количество зафиксированных посетителей в музее в течении суток. Далее, идут N строк с информацией о времени визитов посетителей: в каждой строке располагается отрезок времени посещения в формате «ЧЧ:ММ ЧЧ:ММ» ($00:00 \leq$ ЧЧ:ММ $\leq 23:59$).

Формат выходных данных:

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число — максимальное количество посетителей, одновременно находящихся в музее.

Примеры:

input.txt	output.txt
6 09:00 10:07 10:20 11:35 12:00 17:00 11:00 11:30 11:20 12:30 11:30 18:15	4

Источник задачи: здесь.

Задача 9. Охрана.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

На секретной военной базе работает N охранников. Сутки поделены на 10000 равных промежутков времени, и известно когда каждый из охранников приходит на дежурство и уходит с него. Например, если охранник приходит в 5, а уходит в 8, то значит, что он был в 6, 7 и 8-ой промежуток. В связи с уменьшением финансирования часть охранников решено было сократить. Укажите: верно ли то, что для данного набора охранников, объект

охраняется в любой момент времени хотя бы одним охранником и удаление любого из них приводит к появлению промежутка времени, когда объект не охраняется.

Формат входных данных:

В первой строке входного файла INPUT.TXT записано натуральное число K ($1 \le K \le 30$) – количество тестов в файле. Каждый тест начинается с числа N ($1 \le N \le 10000$), за которым следует N пар неотрицательных целых чисел A и B - время прихода на дежурство и ухода ($0 \le A < B \le 10000$) соответствующего охранника. Все числа во входном файле разделены пробелами и/или переводами строки.

Формат выходных данных:

В выходной файл OUTPUT.TXT выведите K строк, где в M-ой строке находится слово Accepted, если M-ый набор охранников удовлетворяет описанным выше условиям. В противном случае выведите Wrong Answer.

Примеры:

input.txt	output.txt
2 3 0 3000 2500 7000 2700 10000 2 0 3000 2700 10000	Wrong Answer Accepted

Источник задачи: здесь.

Задача 10. Аттракцион.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

На протяжении многих лет в Байтландии существует парк развлечений “Funny byte”, в котором представлено много различных аттракционов: колесо вычислений, веселые горки с трассами в форме интегралов, равнобедренная ромашка и многие другие.

В последние годы популярность “Funny byte” стала неуклонно падать. В целях привлечения посетителей руководство парка решило открыть новый аттракцион, который представляет собой сложный механизм.

Кресла аттракциона расположены в N рядов по M кресел в каждом. То есть каждое кресло характеризуется номером ряда и номером колонки, в котором оно находится. Ряды нумеруются последовательно сверху вниз начиная с единицы, колонки нумеруются слева направо начиная с единицы. Каждое кресло имеет свой уникальный номер. Кресло, находящееся в i-м ряду и в j-ой колонке, имеет номер (i-1)*M + j.

После посадки отдыхающих, кресла поднимают вверх над землей. И начинается веселье! Механизм аттракциона случайным образом производит некоторое количество операций. Под одной операцией понимается взаимная перестановка двух рядов либо двух колонок. При взаимной перестановке двух рядов или колонок каждое кресло в ряду или колонке заменятся на соответствующее ему кресло в другом ряду или колонке.

И вот аттракцион завершил свою работу. Но есть одна трудность! Кресла надо вернуть в начальное положение при помощи таких же операций. Как количество, так и сами операции не обязательно должны быть идентичны тем, которые производил аттракцион во время сеанса. Руководство парка решило, что вернуть кресла в начальное положение необходимо не более чем за 1000 операций.

Такая задача оказалась не по силам разработчикам механизма. Помогите им! От вас требуется разработать программу, позволяющую вернуть кресла в начальное положение. Гарантируется, что решение существует.

Формат входных данных:

В первой строке входного файла INPUT.TXT заданы два натуральных числа N и M ($1 \le N, M \le 250$). В последующих N строках задано по M натуральных чисел, где j-е число в i+1-й строке соответствует номеру кресла после окончания сеанса аттракциона. Числа в строках разделяются одиночными пробелами.

Формат выходных данных:

В первой строке выходного файла OUTPUT.TXT должно быть выведено количество операций перестановки K, которое не должно превышать 1000. Каждая из следующих K строк описывает одну операцию. Каждая операция описывается строкой вида Q X Y, где Q – символ 'R'(ASCII 82) либо символ 'C'(ASCII 67). Если Q равно 'R', то данная операция является перестановкой рядов, если Q равно 'C', то операция является перестановкой колонок. X и Y – два натуральных числа, соответствующие номерам рядов (колонок), которые будут переставлены в результате данной операции. Операции должны быть выведены в порядке осуществления, то есть последовательное применение которых позволит вернуть кресла в начальное положение.

Примеры:

input.txt	output.txt
2 2 4 3 2 1	2 C 1 2 R 1 2
3 5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	0
4 5 10 7 9 8 6 15 12 14 13 11 20 17 19 18 16 5 2 4 3 1	5 R 1 4 C 1 5 C 3 4 R 2 4 R 3 4

Источник задачи: здесь.