Задача 1. Построение списка добавлением в голову

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить односвязный динамический список. Каждое новое число добавлять в начало списка. Затем пройти по построенному списку и посчитать количество отрицательных чисел, входящих в список, кратных 7. После этого память освободить.

Формат входных данных

Входной файл содержит заданную последовательность целых чисел. Числа в файле записаны через пробел. Их величина по модулю не превосходит 1000. Количество чисел может изменяться от 1 до 1000.

Формат выходных данных

В выходной файл нужно вывести одно целое число — количество отрицательных чисел, кратных 7.

Пример

input.txt	output.txt
10 -1 14 8 -21 -35 35 16	2

Задача 2. Построение списка добавлением в хвост

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить односвязный динамический список. Каждое новое число добавлять в конец списка. Затем пройти по построенному списку и посчитать среднее арифметическое чисел, входящих в список. После этого память освободить.

Формат входных данных

Входной файл содержит заданную последовательность целых чисел. Числа в файле записаны через пробел. Их величина по модулю не превосходит 1000. Количество чисел может изменяться от 1 до 1000.

Формат выходных данных

В выходной файл нужно вывести одно целое число — среднее арифметическое элементов списка.

Пример

input.txt	output.txt
1 5 4 6 3	3

Задача 3. Удаление повторов

Источник:	базовая*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Требуется написать программу, которая из целых чисел, записанных во входном файле, строит динамический список. Числа должны располагаться в списке в том же порядке, что и во входном файле.

Затем из полученного списка нужно удалить все повторяющиеся элементы, после чего пройтись по преобразованному списку и в выходной файл выдать через пробел оставшиеся числа.

В конце программы память освободить.

Формат входных данных

Во входном файле через пробел записаны целые числа. Количество чисел от 1 до 10^5.

Формат выходных данных

В выходном файле должны быть записаны через пробел числа, которые остались в списке после удаления повторов.

Пример

input.txt	output.txt
1 4 5 6 6 4 2 3 3 3	1 4 5 6 4 2 3

Задача 4. Удаление предшественников

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить односвязный список. Каждое новое число добавлять в начало списка. Затем пройти по построенному списку и удалить элемент, предшествующий элементу, содержащему заданное число, для всех таких вхождений. После этого распечатать полученный список, память освободить.

Формат входных данных

Первая строка входного файла содержит заданное число, которое нужно удалить.

В следующей строке записана последовательность целых чисел. Числа в файле записаны через пробел. Их величина по модулю не превосходит 1000. Количество чисел может изменяться от 1 до 1000.

Формат выходных данных

В выходной файл нужно вывести последовательность элементов, оставшихся в списке после удаления.

Пример

input.txt	output.txt
4 1 5 4 6 4 3	4 4 5 1

Задача 5. Построение упорядоченного списка

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить односвязный динамический список. Каждое новое число добавлять в список так, чтобы он оставался упорядоченным по возрастанию. Если такое число в списке уже есть, то его не добавлять. Затем пройти по построенному списку от начала до конца и распечатать его элементы. После этого память освободить.

Формат входных данных

Входной файл содержит заданную последовательность целых чисел. Числа в файле записаны через пробел. Их величина по модулю не превосходит 1000. Количество чисел может изменяться от 1 до 1000.

Формат выходных данных

В выходной файл нужно вывести упорядоченную последовательность заданных чисел без повторений. Числа выводить через пробел в одну строку.

Пример

input.txt	output.txt
10 -1 14 8 -21 -3 35 16 -3 10	-21 -3 -1 8 10 14 16 35

Задача 6. Задача Иосифа

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Согласно легенде, еврейский историк Флавий Иосиф (37 н.э. – ок. 100 н.э.) был вовлечен в сражение между евреями и римлянами и прятался в пещере еще с 40 людьми. Пещера была окружена римскими воинами. Казалось неизбежным, что людей Иосифа обнаружат и захватят в плен. Для большинства из находившихся там смерть была более почетной, чем плен, и группа решила совершить самоубийство следующим образом: все становились в круг, начинали считать людей по кругу и убивали каждого третьего, пока не оставался один человек, который должен был убить себя сам. В отличие от остальных, Иосифу и его другу идея плена нравилась больше; они быстро (лихорадочно) вычислили, где им стать в круге, чтобы оказаться двумя последними.

Вам нужно решить более общую задачу: исключать из группы, первоначально состоящей из N элементов, каждого K-го до тех пор, пока в ней не останется один элемент. Для решения ее нужно использовать двусвязный циклический список.

Формат входных данных

Входной файл содержит два натуральных числа N и K – количество элементов в группе и номер каждого удаляемого ($1 \leqslant N, K \leqslant 1000$).

Из чисел от 1 до N построить двусвязный циклический список.

Формат выходных данных

Программа должна удалять из списка каждый K-й элемент, начинать счет нужно с первого элемента, а продолжать со следующего за удаленным.

В выходной файл нужно вывести одно целое число – номер элемента, который останется единственным неудаленным в списке.

Пример

input.txt	output.txt
10 3	4

Задача 7. Слияние списков

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Дано два односвязных упорядоченных по неубыванию списка, содержащих целые числа. Нужно эти два списка слить в один, упорядоченный по невозрастанию, т.е. построить этот список из элементов двух данных. Новой памяти не выделять.

Формат входных данных

В первой строке входного файла записаны через пробел два положительных целых числа N и — длины первого и второго списков ($1 \leqslant N, M \leqslant 1000$).

В следующих двух строках через пробел записаны N и целых чисел соответственно – элементы первого и второго списков. В каждой строке числа даны в неубывающем порядке. Из них нужно построить два списка в том же порядке.

Формат выходных данных

Из двух списков нужно построить один, в котором элементы будут упорядочены по невозрастанию.

В выходной файл нужно вывести значения элементов полученного списка в одну строку через пробел. После этого память освободить.

Пример

input.txt	output.txt
4 5 1 1 7 9 2 3 7 8 9	9 9 8 7 7 3 2 1 1

Задача 8. Сортировка со списками

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В первой строке записано целое число N — количество записей ($1 \leqslant N \leqslant 2 \cdot 10^5$). В остальных N строках содержатся записи, по одной в строке.

Для каждой записи указаны ключ и значение через пробел. Ключ — это целое число в диапазоне от 0 до 10^6 включительно, а значение — это строка от одного до семи символов включительно, состоящая только из маленьких букв латинского алфавита.

Требуется вывести ровно те же самые N записей, но в другом порядке. Записи должны быть упорядочены по возрастанию ключа. Если у нескольких записей ключ равный, то нужно упорядочить их в том порядке, в котором они встречаются по входном файле.

Важно: Решать задачу **нужно** следующим образом (другие решения засчитываться **не** будут). Нужно завести 10^6 связных списков, и в каждый k-ый список складывать все записи с ключом, равным k. Тогда после раскидывания записей по спискам достаточно будет пробежаться по спискам в порядке увеличения k и распечатать их.

Пример

input.txt	output.txt
7 3 qwerty 3 string 6 good 1 a 3 ab 2 hello 5 world	1 a 2 hello 3 qwerty 3 string 3 ab 5 world 6 good

Пояснение к примеру

В примере 7 записей с ключами 1, 2, 3, 5 и 6 — именно в таком порядке записи и выведены в выходном файле. Обратите внимание, что есть три записи с ключом 3: `qwerty`, `string`, `ab`. Они выведены ровно в том порядке, в котором они идут во входном файле.

Задача 9. Список с индексами

Источник:	повышенной сложности*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда*
Ограничение по памяти:	разумное

У связного списка есть серьёзная проблема: в отличие от массива в нём нельзя быстро получить k-ый по порядку элемент. Это можно сделать лишь перебором узлов списка за $O(k) \approx O(N)$ времени. В данном задаче предлагается ускорить поиск узла по индексу.

Рассмотрим односвязный список. Пусть в списке помимо обычных узлов ("маленьких" узлов) есть ещё немного особенных узлов ("больших" узлов). В любом узле списка нужно хранить значение узла и указатель на следующий по порядку элемент. В каждом большом узле списка предлагается дополнительно хранить указатель на следующий **большой** узел списка и расстояние до него. Под расстоянием здесь понимается то, сколько переходов вперёд нужно сделать из одного узла, чтобы попасть по второй узел.

Дополнительные ссылки в больших узлах позволяют быстрее найти k-ый элемент, так как можно проскакивать сразу по много узлов, изначально проходя исключительно по большим узлам. Можно заметить, что если доля больших узлов в списке примерно $\frac{1}{B}$, то найти k-ый элемент можно примерно за $O(\frac{k}{B} + B)$. (**Вопрос:** как лучше выбрать B?)

При добавлении элементов важно поддерживать указанную выше структуру. Вставку узла будем выполнять по индексу, на котором должен стоять новый элемент. При этом сначала нужно решить, будет ли новый узел большим — это можно делать случайным образом, так чтобы поддерживать желаемую долю больших узлов. Затем нужно найти по индексу последний больший и малый узлы перед тем местом, куда нужно вставить новый узел. Наконец, можно вставить новый узел, корректно обновив все ссылки и расстояния так, чтобы сохранилась структура списка.

Предлагается реализовать эту структуру данных, и обработать с её помощью серию из N запросов двух типов (в описаниях L — текущее количество узлов в списке):

0. Вставить на k-ую позицию новый узел с заданным значением V. Гарантируется, что во всех запросах k лежит в пределах от 0 до L.
1. Вывести значение узла, находящегося на k-ой позиции в списке. Гарантируется, что во всех запросах k лежит в пределах от 0 до $L - 1$.

Решение в целом должно работать за время $O(N\sqrt{N})$.

Формат входных данных

В первой строке задано целое число N — общее количество запросов ($1 \leqslant N \leqslant 10^5$). В каждой из следующих N строк дан один запрос. Запрос вставки задаётся тремя целыми числами 0 k V, а запрос на вывод элемента задаётся двумя целыми числами 1 k. Все числа V в списке целые, по модулю не превышают 10^9.

Формат выходных данных

Для каждого запроса на вывод (типа 1) нужно вывести значение k-ого узла в списке.

Пример

input.txt	output.txt
10 0 0 7 0 0 5 0 1 3 0 0 2 1 0 1 1 1 2 1 3 0 1 -5 1 2	2 5 3 7 5

Комментарий

Несмотря на формально лучшую асимптотику, полученная структура данных будет обрабатывать заданные запросы не быстрее простого массива.