Содержание

Must have 2

Задача 25A. Простейшее BST [0.2, 256] 2

Задача 25B. Простейший полуявный ключ [0.2, 256] 3

Задача 25C. Простейший неявный Ключ [0.2, 256] 4

Задачи здорового человека 5

Задача 25D. Двоичное дерево поиска [0.1, 256] 5

Задача 25E. Дебаг-вывод дерева [0.2, 256] 6

Задача 25F. K-ый максимум [0.1, 256] 7

Задача 25G. Неявный Ключ [0.5, 256] 8

Для искателей острых ощущений 9

Задача 25H. Range Minimum Query [0.5, 256] 9

Задача 25I. И снова сумма... [0.6, 256] 10

Задача 25J. Перестановки [1.5 sec, 256 mb] 11

Для мастеров AI 12

Задача 25K. Эх, дороги... [0.25, 256] 13

У вас не получается читать/выводить данные?
Воспользуйтесь примерами (c++) (python).

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`.
Подни можно пользоваться дополнительной библиотекой (`optimization.h`).
То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.
И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): пример.

Для тех, кто хочет разобраться, как всё это работает.
Короткая версия быстрого ввода-вывода (тык) и короткая версия аллокатора (тык).

Must have

Задача 25A. Простейшее BST [0.2, 256]

В этой задаче вам нужно написать простейшее BST по явному ключу и отвечать им на запросы:

- `+ x` – добавить в дерево `x` (если `x` уже есть, ничего не делать).
- `> x` – вернуть минимальный элемент больше `x` или 0, если таких нет.

Формат входных данных

В каждой строке содержится один запрос.
Все x целые от 1 до 10^9, количество запросов от 1 до 300 000.
Гарантируется, что все x выбраны равномерным распределением.

Формат выходных данных

Для каждого запроса вида «`> x`» выведите в отдельной строке ответ.

Пример

stdin	stdout
+ 1 + 3 + 3 > 1 > 2 > 3 + 2 > 1	3 3 0 2

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

Вам нужно в каждой вершине поддерживать размер поддерева.
В этой задаче вам нужно написать BST по **не**явному ключу.

Задача 25B. Простейший полуявный ключ [0.2, 256]

В этой задаче вам нужно написать BST по **явному** ключу и отвечать им на запросы:

- `+ x` – добавить в дерево `x` (если `x` уже есть, ничего не делать).
- `? k` – вернуть k-й по возрастанию элемент.

Формат входных данных

В каждой строке содержится один запрос.
Все x целые от 1 до 10^9, количество запросов от 1 до 300 000.
Гарантируется, что все x выбраны равномерным распределением.
В запросах «`? k`», число k от 1 до количества элементов в дереве.

Формат выходных данных

Для каждого запроса вида «`? k`» выведите в отдельной строке ответ.

Пример

stdin	stdout
+ 1 + 4 + 3 + 3 ? 1 ? 2 ? 3 + 2 ? 3	1 3 4 3

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

Вам нужно в каждой вершине поддерживать размер поддерева.
В этой задаче вам нужно написать BST по явному ключу.

Задача 25C. Простейший неявный Ключ [0.2, 256]

Изначально есть пустой массив. Вам нужно обрабатывать запросы вида
`i x` — добавить после i-го элемента x $(0 \leqslant i \leqslant n)$, где n текущая длина массива.
В конце после всех добавлений нужно вывести полученный массив.

Формат входных данных

q $(1 \leqslant q \leqslant 100\,000)$ строк, на каждой запрос «`i x`» —
пара целых чисел $(0 \leqslant i \leqslant n,\ 1 \leqslant x < 100\,000)$.

Формат выходных данных

Выведите q целых чисел — полученный массив.

Примеры

stdin	stdout
0 1 0 2 0 3	3 2 1
0 1 1 2 2 3	1 2 3
0 1 1 2 1 3 0 4	4 1 3 2

Подсказка по решению

Случайные данные! Не нужно ничего специально балансировать.

Вам нужно в каждой вершине поддерживать размер поддерева.
В этой задаче вам нужно написать BST по **не**явному ключу.

Задачи здорового человека

Задача 25D. Двоичное дерево поиска [0.1, 256]

Реализуйте сбалансированное двоичное дерево поиска.
Или воспользуйтесь любой стандартной структурой из `C++::STL`.
Или попытайтесь запихать квадрат.

Формат входных данных

Входной файл содержит описание одной или нескольких операций с деревом.
Операций не больше 10^5. Все числа целые от -10^5 до 10^9.
В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x.
  Если ключ x в дереве уже есть, то ничего делать не надо.
- `delete` x — удалить из дерева ключ x.
  Если ключа x в дереве нет, то ничего делать не надо.
- `exists` x — если ключ x есть в дереве, «`true`», иначе «`false`»
- `next` x — минимальный элемент в дереве, $> x$, или «`none`», если такого нет.
- `prev` x — максимальный элемент в дереве, $< x$, или «`none`», если такого нет.

Формат выходных данных

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`.
Следуйте формату выходного файла из примера.

Примеры

stdin	stdout
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4	true false 5 3 none 3

Подсказка по решению

`while (!seekEof()) { ... }`
Заходят `set`, умный квадрат, сбалансированное bst.

Задача 25E. Дебаг-вывод дерева [0.2, 256]

Дана последовательность добавлений в дерево поиска (bst). После каждого добавления выводите LxR обход дерева (вывести левое поддерева, вывести ключ, вывести правое поддерева).

Формат входных данных

Последовательность из n ($1 \leqslant n \leqslant 1\,000$) целых чисел от 0 до 10^6, которые следуют добавить. Добавлять нужно именно в таком порядке. Гарантируется, что все числа **различны**.

Формат выходных данных

После каждого добавления выведите дерево в особом формате (см. примеры). Деревья для удобства чтения разделяйте строкой с одним дефисом. Обратите внимание, число пробелов перед x равно 2· глубина вершины.

Примеры

stdin	stdout

1
5
9
3
	x=1 size=1
	-
	x=1 size=2
	  x=5 size=1
	-
	x=1 size=3
	  x=5 size=2
	    x=9 size=1
	-
	x=1 size=4
	    x=3 size=1
	  x=5 size=3
	    x=9 size=1

Подсказка по решению

Здесь эта задача не просто так. Такое действительно сильно помогает при дебаге. Выводите своё дерево после каждого запроса, любуйтесь тем, как оно растёт и развивается.

Задача 25F. K-ый максимум [0.1, 256]

Напишите программу, реализующую структуру данных, позволяющую добавлять и удалять элементы, а также находить k-й максимум.

Формат входных данных

Первая строка входного файла содержит натуральное число n — количество команд ($n \leqslant 100\,000$). Последующие n строк содержат по одной команде каждая. Команда записывается в виде двух чисел c_i и k_i — тип и аргумент команды соответственно ($|k_i| \leqslant 10^9$). Поддерживаемые команды:

- $+1$ (или просто 1): Добавить элемент с ключом k_i.
- 0: Найти и вывести k_i-й максимум.
- -1: Удалить элемент с ключом k_i.

Гарантируется, что в процессе работы в структуре не требуется хранить элементы с равными ключами или удалять несуществующие элементы. Также гарантируется, что при запросе k_i-го максимума, он существует.

Формат выходных данных

Для каждой команды нулевого типа в выходной файл должна быть выведена строка, содержащая единственное число — k_i-й максимум.

Пример

stdin	stdout
11 +1 5 +1 3 +1 7 0 1 0 2 0 3 -1 5 +1 10 0 1 0 2 0 3	7 5 3 10 7 3

Подсказка по решению

Напишите AVL дерево. `tree<int>` запрещён.

Задача 25G. Неявный Ключ [0.5, 256]

Научитесь быстро делать две операции с массивом:
∘ `add i x` — добавить после i-го элемента x ($0 \leqslant i \leqslant n$)
∘ `del i` — удалить i-й элемент ($1 \leqslant i \leqslant n$)

Формат входных данных

На первой строке n_0 и m ($1 \leqslant n_0, m \leqslant 10^5$) — длина исходного массива и количество запросов. На второй строке n_0 целых чисел от 0 до $10^9 - 1$ — исходный массив. Далее m строк, содержащие запросы. Гарантируется, что запросы корректны: например, если просят удалить i-й элемент, он точно есть.

Формат выходных данных

Выведите конечное состояние массива. На первой строке количество элементов, на второй строке сам массив.

Примеры

stdin	stdout
3 4 1 2 3 del 3 add 0 9 add 3 8 del 2	3 9 2 8

Подсказка по решению

Напишите AVL дерево по неявному ключу.

Для искателей острых ощущений

Задача 25H. Range Minimum Query [0.5, 256]

Компания *Giggle* открывает свой новый офис в Судиславле, и вы приглашены на собеседование. Ваша задача — решить поставленную задачу.

Вам нужно создать структуру данных, которая представляет из себя массив целых чисел. Изначально массив пуст. Вам нужно поддерживать две операции:

- запрос: «`? i j`» — возвращает минимальный элемент между i-ым и j-м, включительно;
- изменение: «`+ i x`» — добавить элемент x после i-го элемента списка. Если $i = 0$, то элемент добавляется в начало массива.

Конечно, эта структура должна быть достаточно хорошей.

Формат входных данных

Первая строка входного файла содержит единственное целое число n — число операций над массивом ($1 \leqslant n \leqslant 200\,000$). Следующие n строк описывают сами операции. Все операции добавления являются корректными. Все числа, хранящиеся в массиве, по модулю не превосходят 10^9.

Формат выходных данных

Для каждой операции в отдельной строке выведите её результат.

Примеры

stdin	stdout
8 + 0 5 + 1 3 + 1 4 ? 1 2 + 0 2 ? 2 4 + 4 1 ? 3 5	4 3 1

Подсказка по решению

В вершине теперь нужно ещё и хранить то, что просят.

Задача 25I. И снова сумма... [0.6, 256]

Реализуйте структуру данных, которая поддерживает множество S целых чисел, с котором разрешается производить следующие операции:

- $add(i)$ — добавить в множество S число i (если он там уже есть, то множество не меняется);
- $sum(l, r)$ — вывести сумму всех элементов x из S, которые удовлетворяют неравенству $l \leqslant x \leqslant r$.

Формат входных данных

Исходно множество S пусто. Первая строка входного файла содержит n — количество операций ($1 \leqslant n \leqslant 300\,000$). Следующие n строк содержат операции. Каждая операция имеет вид либо «`+ i`», либо «`? l r`». Операция «`? l r`» задает запрос $sum(l, r)$.

Если операция «`+ i`» идет во входном файле в начале или после другой операции «`+`», то она задает операцию $add(i)$. Если же она идет после запроса «`?`», и результат этого запроса был y, то выполняется операция $add((i + y) \bmod 10^9)$.

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до 10^9.

Формат выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

Пример

stdin	stdout
6 + 1 + 3 + 3 ? 2 4 + 1 ? 2 4	3 7

Подсказка по решению

На практике разобрали, как можно отвечать на такие запросы.

Задача 25J. Перестановки [1.5 sec, 256 mb]

Рассмотрим циклический алфавит, состоящий из первых десяти букв английского алфавита. Циклическим он называется потому, что следующей буквой за ‘a’ является буква ‘b’, за ‘b’ — ‘c’ и так далее, и при этом следующей за буквой ‘j’ является буква ‘a’.

Вам дана строка S, состоящая из букв этого циклического алфавита. Вы должны обрабатывать запросы трёх типов:

- Развернуть подстроку строки S с L-го по R-й символ включительно.
- В подстроке S с L-го по R-й символ заменить каждый символ на D-й следующий символ в циклическом алфавите.
- Для подстроки S с L-го по R-й символ найти количество различных перестановок символов этой подстроки по модулю $10^9 + 7$.

Формат входных данных

Первая строка входного файла содержит одно число N ($1 \leqslant N \leqslant 10^5$) — длину строки S. Во второй строке содержится сама строка S, состоящая из строчных букв английского алфавита от ‘a’ до ‘j’. Третья строка содержит число M ($1 \leqslant M \leqslant 10^5$) — количество запросов. Затем идут M строк, содержащие описания запросов трех типов:

- `-1 L R` ($1 \leqslant L \leqslant R \leqslant N$) — разворот подстроки.
- `0 L R D` ($1 \leqslant L \leqslant R \leqslant N$, $0 < D \leqslant N$) — замена символов в подстроке.
- `1 L R` ($1 \leqslant L \leqslant R \leqslant N$) — количество различных перестановок подстроки $[L, R]$ строки S.

Формат выходных данных

На каждый запрос типа “`1 L R`” выведите ответ по модулю $10^9 + 7$ в отдельной строке.

Пример

stdin	stdout
6 abcabc 3 -1 1 6 0 1 3 1 1 1 6	180

Замечание

D-м следующим символом за x называется символ, получающийся путем D-кратного взятия следующего за x символа в циклическом алфавите.

Для мастеров AI

Правила.
Это блок задач про промтинг. Пользоваться можно только бесплатными версиями ИИ. В шапке исходника указывать последовательность промтов, и сайты, куда они были отправлены. Если в процессе использования ИИ вы получили какие-то важные идеи для решения, это тоже часть решения. Если с помощью ИИ вы написали генератор тестов или стресс-тест, это тоже следует указывать. Задокументируйте, пожалуйста, проделанную работу в шапке отосланного решения. Если это нужно, вы можете писать часть кода сами.

Задача 25K. Эх, дороги... [0.25, 256]

В диком и суровом королевстве Гранития началась очередная транспортная кампания. Правда, дороги в этом королевстве давно стали притчей во языцех – то трещины, то овцы пасутся, то мост сам развалился. Власти решили взяться за дело серьёзно, но закон суров: из каждого поселения в каждый момент времени должно вести не более двух дорог. Все дороги – двусторонние, знаков нет, направления нет – кто куда хочет, туда и едет.

По плану, часть дорог будут строить, а часть закрывать на неопределённый срок по причине «внезапного обрушения» или «атак гномов».

В этой суматохе Лея, диспетчер караванной службы, должна моментально находить кратчайшие пути между поселениями, ведь каждый дополнительный город по пути – это риск нарваться на сборщиков налогов или заблудиться в тумане.

Ваша задача – помочь Лее отвечать на запросы о кратчайших маршрутах по мере изменений в дорожной сети.

Формат входных данных

В первой строке входного файла заданы числа n — количество городов, m — количество дорог в начале кампании и q — количество сообщений об изменении дорожной структуры и запросов ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 100\,000$, $0 \leqslant q \leqslant 200\,000$). Следующие m строк содержат по два целых числа каждая — пары городов, соединенных дорогами перед реформой. Следующие q строк содержат по три элемента, разделенных пробелами. «`+` i j» означает строительство дороги от города i до города j, «`-` i j» означает закрытие дороги от города i до города j, «`?` i j» означает запрос об оптимальном пути между городами i и j.

Гарантируется, что в начале и после каждого изменения никакие два города не соединены более чем одной дорогой, и из каждого города выходит не более двух дорог. Никакой город не соединяется дорогой сам с собой.

Формат выходных данных

На каждый запрос вида «`?` i j» выведите одно число — минимальное количество промежуточных городов на маршруте из города i в город j. Если проехать из i в j невозможно, выведите -1.

Пример

stdin	stdout
5 4 6 1 2 2 3 1 3 4 5 ? 1 2 ? 1 5 - 2 3 ? 2 3 + 2 4 ? 1 5	0 -1 1 2

Подсказка по решению

Нужно увидеть массив, то есть, деревья по неявному ключу.