Содержание

Вы не умеете читать/выводить данные, открывать файлы? Воспользуйтесь примерами.

В некоторых задачах большой ввод и вывод. Пользуйтесь быстрым вводом-выводом.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу.

Обратите внимание на компилятор `GNU C++11 5.1.0 (TDM-GCC-64) inc`, который позволяет пользоваться дополнительной библиотекой. Под ним можно сдать вот это.

В этом контесте всего одна обязательная задача на корневую. Если хотите потренировать мастерство корневой, можно взять все задачи прошлого контеста и сдать корневой.

Must have

Задача 10A. Persistent Array [1 sec, 256 mb]

Дан массив (вернее, первая, начальная его версия).

Нужно уметь отвечать на два запроса:

∘ $a_i[j] = x$ — создать из i-й версии новую, в которой j-й элемент равен x, а остальные элементы такие же, как в i-й версии.

∘ `get` $a_i[j]$ — сказать, чему равен j-й элемент в i-й версии.

Формат входных данных

Количество чисел в массиве N ($1 \leqslant N \leqslant 10^5$) и N элементов массива. Далее количество запросов M ($1 \leqslant M \leqslant 10^5$) и M запросов. Формат описания запросов можно посмотреть в примере. Если уже существует K версий, новая версия получает номер $K + 1$. И исходные, и новые элементы массива — целые числа от 0 до 10^9. Элементы в массиве нумеруются числами от 1 до N.

Формат выходных данных

На каждый запрос типа `get` вывести соответствующий элемент нужного массива.

Пример

stdin	stdout
6 1 2 3 4 5 6 11 create 1 6 10 create 2 5 8 create 1 5 30 get 1 6 get 1 5 get 2 6 get 2 5 get 3 6 get 3 5 get 4 6 get 4 5	6 5 10 5 10 8 6 30

Замечание

Вы умеете эту задачу решать и в offline, и в online.

Обязательные задачи

Задача 10B. CHM [1 sec, 256 mb]

Ваша задача — реализовать **Persistent Disjoint-Set-Union**. Что это значит?
Про **Disjoint-Set-Union**:
Изначально у вас есть n элементов. Нужно научиться отвечать на 2 типа запросов.

- `+ a b` — объединить множества, в которых лежат вершины a и b
- `? a b` — сказать, лежат ли вершины a и b сейчас в одном множестве

Про **Persistent**:
Теперь у нас будет несколько копий (версий) структуры данных **Disjoint-Set-Union**.
Запросы будут выглядеть так:

- `+ i a b` — запрос к i-й структуре, объединить множества, в которых лежат вершины a и b. При этом i-я структура остается не изменной, создается новая версия, ей присваивается новый номер (какой? читайте дальше)
- `? i a b` — запрос к i-й структуре, сказать, лежат ли вершины a и b сейчас в одном множестве

Формат входных данных

На первой строке 2 числа N ($1 \leqslant N \leqslant 10^5$) и K ($0 \leqslant K \leqslant 10^5$) — число элементов и число запросов. Изначально все элементы находятся в различных множествах. Эта изначальная копия (версия) структуры имеет номер 0.

Далее следуют K строк, на каждой описание очередного запроса. Формат запросов описан выше. Запросы нумеруются целыми числами от 1 до K.

Пусть j-й из K запросов имеет вид «`+ i a b`». Тогда новая версия получит номер j. Запросы вида «`? i a b`» не порождают новых структур.

Формат выходных данных

Для каждого запроса вида `? i a b` на отдельной строке нужно вывести `YES` или `NO`.

Пример

stdin	stdout
4 7 + 0 1 2 ? 0 1 2 ? 1 1 2 + 1 2 3 ? 4 3 1 ? 0 4 4 ? 4 1 4	NO YES YES YES NO

Замечание

И эту задачу Вы умеете решать и в offline, и в online.
Хотите уметь больше? Попробуйте второй способ!

Задача 10C. Менеджер памяти [1 sec, 256 mb]

Одно из главных нововведений новейшей операционной системы Indows 7 — новый менеджер памяти. Он работает с массивом длины N и позволяет выполнять три самые современные операции:

- `copy(a, b, l)` — скопировать отрезок длины $[a, a+l-1]$ в $[b, b+l-1]$
- `sum(l, r)` — посчитать сумму элементов массива на отрезке $[l, r]$
- `print(l, r)` — напечатать элементы с l по r, включительно

Вы являетесь разработчиком своей операционной системы, и Вы, безусловно, не можете обойтись без инновационных технологий. Вам необходимо реализовать точно такой же менеджер памяти.

Формат входных данных

Первая строка входного файла содержит целое число N ($1 \leqslant N \leqslant 1\,000\,000$) — размер массива, с которым будет работать Ваш менеджер памяти.

Во второй строке содержатся четыре числа $1 \leqslant X_1, A, B, M \leqslant 10^9 + 10$. С помощью них можно сгенерировать исходный массив чисел $X_1, X_2, \ldots, X_N$. $X_{i+1} = (A \cdot X_i + B) \mod M$

Следующая строка входного файла содержит целое число K ($1 \leqslant K \leqslant 200\,000$) — количество запросов, которые необходимо выполнить Вашему менеджеру памяти.

Далее в K строках содержится описание запросов. Запросы заданы в формате:

- `cpy` a b l — для операции `copy`
- `sum` l r — для операции `sum` ($l \leqslant r$)
- `out` l r — для операции `print` ($l \leqslant r$)

Гарантируется, что суммарная длина запросов `print` не превышает 3 000. Также гарантируется, что все запросы корректны.

Формат выходных данных

Для каждого запроса *sum* или *print* выведите в выходной файл на отдельной строке результат запроса.

Пример

stdin	stdout
6 1 4 5 7 8 out 1 6 sum 1 6 cpy 1 3 2 out 3 4 sum 3 4 cpy 1 2 4 out 1 6 sum 1 6	1 2 6 1 2 6 18 1 2 3 1 1 2 1 2 6 13

Замечание

Эту задачу вы умеете решать персистентным деревом и корневой.

Предполагается, что Вы напишите персистентное дерево.

Задача 10D. Менеджер памяти [1 sec, 256 mb]

Абсолютно такая же, как предыдущая.
Только ограничения жестче.

Замечание

Добавьте сборку мусора в своё персистентное дерево (ссылочный garbage collector).
Любители `shared_ptr` получат `TL` или `ML`.

Задача 10E. Фаброзавры-дизайнеры [1 sec, 256 mb]

Фаброзавры известны своим тонким художественным вкусом и увлечением ландшафтным дизайном. Они живут около очень живописной реки и то и дело перестраивают тропинку, идущую вдоль реки: либо насыпают дополнительной земли, либо срывают то, что есть. Для того, чтобы упростить эти работы, они поделили всю тропинку на горизонтальные участки, пронумерованные от 1 до N, и их переделки устроены всегда одинаково: они выбирают часть дороги от L-ого до R-ого участка (включительно) и изменяют (увеличивают или уменьшают) высоту на всех этих участках на одну и ту же величину (если до начала переделки высоты были разными, то и после переделки они останутся разными).

Поскольку, как уже говорилось, у фаброзавров тонкий художественный вкус, каждый из них считает, что их река лучше всего выглядит с определенной высоты. Поэтому им хочется знать, есть ли поблизости от их дома место на тропинке, где высота на их взгляд оптимальна. Помогите им в этом разобраться.

Формат входных данных

Первая строка входного файла содержит два числа N и M — длину дороги и количество запросов соответственно ($1 \leqslant N, M \leqslant 10^5$). На второй строке содержатся N чисел, разделенных пробелами — начальные высоты соответствующих частей дороги; высоты не превосходят 10^4 по модулю. В следующих M строках содержатся запросы по одному на строке.

Запрос `+` L R X означает, что высоту частей дороги от L-ой до R-ой (включительно) нужно изменить на X. При этом $1 \leqslant L \leqslant R \leqslant N$, а $|X| \leqslant 10^4$.

Запрос `?` L R X означает, что нужно проверить, есть ли между L-ым и R-ым участками (включая эти участки) участок, где дорога проходит точно на высоте X. Гарантируется, что $1 \leqslant L \leqslant R \leqslant N$, а $|X| \leqslant 10^9$.

Формат выходных данных

На каждый запрос второго типа нужно вывести в выходной файл на отдельной строке одно слово «`YES`» (без кавычек), если нужный участок существует, и «`NO`» в противном случае.

Примеры

stdin	stdout
10 5 0 1 1 3 3 3 2 0 0 1 ? 3 5 2 + 1 4 1 ? 3 5 2 + 7 10 2 ? 9 10 3	NO YES YES

Замечание

Классическая корневая...

Авторское решение – корневая без `split`, разбиение на куски статично.

Дополнительные задачи

Задача 10F. Опять k-я статистика [1 sec, 256 mb]

Изначально вам дан массив целых чисел.
Нужно уметь отвечать на три запроса:

- `+ i x` — Вставить на i-ю позицию число x (размер массива увеличивается на 1)
- `- i` — Удалить число на i-й позиции (размер массива уменьшается на 1)
- `? L R x` — Сказать, сколько чисел y на позициях $L \leqslant i \leqslant R$ таких, что $y \leqslant x$ ($|x| \leqslant 10^9$)

Все индексы i, L, R нумеруются с нуля.
Все числа в запросах целые. Все запросы корректны.
Пример запроса: "`+ 0 x`" означает "добавление x в начало массива".
Длина массива: $0 \leqslant N \leqslant 10^5$. Число запросов: $1 \leqslant K \leqslant 10^5$.
Числа в массиве по модулю не превышают 10^9.

Пример

stdin	stdout
10 455184306 359222813 948543704 914773487 861885581 253523 770029097 193773919 581789266 457415808 - 1 ? 2 5 527021001 ? 0 5 490779085 ? 0 5 722862778 + 9 448694272 - 5 ? 1 2 285404014 - 4 ? 3 4 993634734 + 0 414639071	1 2 2 0 2

Задача 10G. Формула-1 [1 sec, 256 mb]

Фотокорреспондент планирует съёмки соревнований "Формулы-1". Для этого он собирается пройти вдоль самой длинной прямой трека и сделать снимки последовательно с каждой из n запланированных точек обзора. Для каждой точки обзора известны расстояния a и b от данной точки до каждого из концов отрезка, по которому болиды проезжают мимо неё, находясь в прямой видимости.

У корреспондента есть m объективов, для каждого из которых заданы два параметра: минимальное расстояние c и максимальное расстояние d до болида, необходимое для получения качественных снимков с помощью этого объектива. Так как смена объектива — трудоёмкое занятие, то фотокорреспондент хочет для каждого объектива вычислить, каково максимальное число **последовательных** точек обзора, на которых данный объектив может быть полезен. Объектив с параметрами $[c, d]$ является полезным в некоторой точке обзора, если болиды могут проехать мимо этой точки на некотором расстоянии x таком, что $c \leqslant x \leqslant d$.

Подсчитайте интересующее фотографа число для каждого из имеющихся в наличии объективов.

Формат входных данных

В первой строке входного файла заданы два целых числа n и m ($1 \leqslant n \leqslant 50\,000, 1 \leqslant m \leqslant 200\,000$) — количество точек обзора и количество объективов у фотокорреспондента. В последующих n строках заданы параметры точек обзора в порядке их обхода корреспондентом. Для i-й точки заданы два целых числа a_i, b_i ($1 \leqslant a_i \leqslant b_i \leqslant 10^9$), обозначающие, что болиды могут проезжать на расстоянии от a_i до b_i от данной точки, включительно. Далее следуют m строк, задающих параметры объективов. Каждый объектив задан двумя целыми числами c_j, d_j ($1 \leqslant c_j \leqslant d_j \leqslant 10^9$) — минимальным и максимальным расстоянием до болида, требуемым для получения качественных снимков.

Формат выходных данных

Для каждого из m объективов в порядке их перечисления во входном файле выведите одно число — наибольшее количество идущих подряд точек обзора, на которых данный объектив является полезным.

Примеры

stdin	stdout
3 4 2 5 1 3 6 6 3 5 1 10 7 9 5 6	2 3 0 1

Задача 10H. Точки в полуплоскости [1 sec, 256 mb]

Есть N точек на плоскости.

Точки равномерно распределены внутри квадрата $[0..C] \times [0..C]$.

Вам нужно научиться отвечать на запрос "сколько точек лежит в полуплоскости"?

Формат входных данных

Число точек N ($1 \leqslant N \leqslant 50\,000$), число запросов M ($1 \leqslant M \leqslant 50\,000$), константа C (целое число от 1 до 10^4). Далее N точек (X, Y) с целочисленными координатами. Далее M полуплоскостей (a, b, c). Числа a, b, c — целые, по модулю не превосходят 10^4. $a^2 + b^2 \neq 0$. Считается, что точка лежит в полуплоскости тогда и только тогда, когда $ax + by + c \geqslant 0$.

Формат выходных данных

Для каждого из M запросов одно целое число — количество точек в полуплоскости.

Пример

stdin	stdout
3 4 10 5 5 1 7 7 4 1 1 -9 1 1 -10 1 1 -11 1 1 -12	2 2 1 0