Содержание

В некоторых задачах большой ввод и вывод. Имеет смысл пользоваться супер быстрым вводом-выводом: `http://acm.math.spbu.ru/~sk1/algo/input-output/`

Задачи

Задача A. Сбалансированное дерево [0.5 sec, 256 mb]

Дано подвешенное за вершину с номером 1 дерево из N вершин. Вершины дерева нумеруются от 1 до N и покрашены в красный и черный цвета.

Дерево называется красно-черным, если:

- У каждой вершины количество детей или 2 или 0.
- Все листья черные
- На пути от корня до любого из листьев одинаковое количество черных вершин
- Отцом красной вершины не может быть красная вершина

Ваша задача — проверить, является ли данное вам дерево красно-черным. Если же является, то нужно построить по нему 2-3-4-дерево. Для того, чтобы построить 2-3-4-дерево, нужно сперва корень дерева сделать чёрным. Затем для каждой черной вершины провести ребро из неё в ближайшего черного предка. 2-3-4-деревом будут являться все чёрные вершины исходного и проведённые нами рёбра.

Для лучшего понимания смотрите примеры.

Формат входных данных

Число N ($1 \leqslant N \leqslant 10^5$), далее N пар «число от 1 до $i-1$, номер отца i-й вершины» и «буква, описывающая цвет i-й вершины» (`R` = red, `B` = black). Отец корня (первой вершины) — вершина с номером -1.

Формат выходных данных

Выведите `YES` или `NO`. Если ответ `YES`, выведите 2-3-4 дерево изоморфное исходному. Дерево выведите в следующем формате: количество ребер и сами ребра. Каждое ребро описывается номерами вершин-концов. Нумерация вершин должна быть такая же, как и в исходном дереве.

Пример

bst.in	bst.out
1 -1 R	NO
3 -1 R 1 B 1 B	YES 2 1 2 1 3
2 -1 B 1 B	NO
5 -1 R 1 B 1 R 3 B 3 B	NO
17 -1 B 1 R 1 B 2 B 2 B 4 B 4 B 5 B 5 R 9 B 9 B 3 R 3 R 12 B 12 B 13 B 13 B	YES 12 1 3 1 4 1 5 4 6 4 7 5 8 5 10 5 11 3 14 3 15 3 16 3 17
4 -1 B 1 B 1 B 1 B	NO

Замечание

2-3-4-дерево обладает двумя свойствами — во-первых, расстояния от корня до листьев одинаковы, во-вторых, число детей любого не листа и не корня — 2, 3 или 4.

Задача B. Сила с тобой, Люк [1.5 sec, 256 mb]

Дан массив a из n чисел, нужно научиться обрабатывать запросы двух типов.

- `change (i, y)` — сделать a_i равным y.
- `int get ()` — вернуть индекс i такой, что a_i встречается в массиве наименьшее возможное число раз. Если таких i несколько, вернуть минимально возможный.

Все индексы, встречающиеся в задаче, нумеруются с нуля.

Формат входных данных

На первой строке размер массива n. На второй сам массив — n целых чисел от 0 до 10^9-1. На третей строке число запросов q. Следующие q строк содержат запросы в формате "?" (`get`) и "= i y" (`change`).

Ограничения: $n, q \leqslant 300\,000$.

Формат выходных данных

На каждый запрос `get` выведите на отдельной строке ответ.

Замечание

Поскольку 8.5 мегабайт нельзя прочитать из файла мгновенно (как и записать 1 мегабайт данных), используйте максимально быстрые ввод вывод.

Примеры

sforce.in	sforce.out
4 1 2 3 4 5 ? = 0 2 ? = 3 3 ?	0 2 0
6 1 1 2 1 1 0 3 ? = 5 2 ?	2 2

Задача C. Persistant 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

parray.in	parray.out
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

Задача D. Менеджер памяти [8 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.

Подзадача 1 (25 баллов) $N, K \leqslant 20\,000$
Подзадача 2 (25 баллов) суммарная длина запросов `copy` не превосходит 1 000 000
Подзадача 3 (25 баллов) $K \leqslant 10\,000$
Подзадача 4 (25 баллов) Дополнительные упрощения отсутствуют

Пример

memory.in	memory.out
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

Бонус

Задача E. ВнутренНЯя точка 2 [1 sec, 256 mb]

Дан совсем невыпуклый N-угольник и K точек. Для каждой точки нужно определеить, где она находится — внутри, на границе, или снаружи.

Формат входных данных

N ($3 \leqslant N \leqslant 10^5$). Далее N точек — вершины многоугольника.
K ($0 \leqslant K \leqslant 10^5$). Далее K точек — запросы.
Все координаты — целые числа по модулю не превосходящие 10^9.

Формат выходных данных

Для каждого запроса одна строка — `INSIDE`, `BORDER` или `OUTSIDE`.

Примеры

inside2.in	inside2.out
4 0 0 2 0 2 2 0 2 4 1 1 0 0 0 1 0 3	INSIDE BORDER BORDER OUTSIDE

Задача F. Эх, дороги... [1 sec, 256 mb]

В многострадальном Тридесятом государстве опять готовится дорожная реформа. Впрочем, надо признать, дороги в этом государстве находятся в довольно плачевном состоянии. Так что реформа не повредит. Одна проблема — дорожникам не развернуться, поскольку в стране действует жесткий закон — из каждого города должно вести не более двух дорог. Все дороги в государстве двусторонние, то есть по ним разрешено движение в обоих направлениях (разумеется, разметка отсутствует). В результате реформы некоторые дороги будут строиться, а некоторые другие закрываться на бессрочный ремонт.

Петя работает диспетчером в службе грузоперевозок на дальние расстояния. В связи с предстоящими реформами, ему необходимо оперативно определять оптимальные маршруты между городами в условиях постоянно меняющейся дорожной ситуации. В силу большого количества пробок и сотрудников дорожной полиции в городах, критерием оптимальности маршрута считается количество промежуточных городов, которые необходимо проехать.

Помогите Пете по заданной последовательности сообщений об изменении структуры дорог и запросам об оптимальном способе проезда из одного города в другой, оперативно отвечать на запросы.

Формат входных данных

В первой строке входного файла заданы числа n — количество городов, m — количество дорог в начале реформы и q — количество сообщений об изменении дорожной структуры и запросов ($1 \leqslant n, m \leqslant 100\,000$, $q \leqslant 200\,000$). 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

roads.in	roads.out
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