Содержание

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.

Задачи

Задача 3A. Динамический Лес [0.6 sec, 256 mb]

Вам нужно научиться обрабатывать 3 типа запросов:

1. Добавить ребро в граф (`link`).
2. Удалить ребро из графа (`cut`).
3. По двум вершинам a и b, определить, лежат ли они в одной компоненте связности (`get`).

Изначально граф пустой (содержит N вершин, не содержит ребер). Гарантируется, что в любой момент времени граф является лесом. При добавлении ребра гарантируется, что его сейчас в графе нет. При удалении ребра гарантируется, что оно уже добавлено.

Формат входных данных

Числа N и M ($1 \leqslant N \leqslant 10^5 + 1$, $1 \leqslant M \leqslant 10^5$) — количество вершин в дереве и, соответственно, запросов. Далее M строк, в каждой строке команда (`link` или `cut`, или `get`) и 2 числа от 1 до N — номера вершин в запросе.

Формат выходных данных

В выходной файл для каждого запроса `get` выведите 0, если не лежат, или 1, если лежат.

Пример

stdin	stdout
3 7 get 1 2 link 1 2 get 1 2 cut 1 2 get 1 2 link 1 2 get 1 2	0101
5 10 link 1 2 link 2 3 link 4 3 cut 3 4 get 1 2 get 1 3 get 1 4 get 2 3 get 2 4 get 3 4	110100

Задача 3B. Жесть [6 sec, 256 mb]

Дам массив из N чисел. Нужно уметь обрабатывать 3 типа запросов:
- `get(L, R, x)` — сказать, сколько элементов отрезка массива $[L..R]$ не меньше x.
- `set(L, R, x)` — присвоить всем элементам массива на отрезке $[L..R]$ значение x.
- `reverse(L, R)` — перевернуть отрезок массива $[L..R]$.

Формат входных данных

Число N ($1 \leqslant N \leqslant 10^5$) и массив из N чисел. Далее число запросов M ($1 \leqslant M \leqslant 10^5$) и M запросов. Формат описания запросов предлагается понять из примера. Для всех отрезков верно $1 \leqslant L \leqslant R \leqslant N$. Исходные числа в массиве и числа x в запросах — целые от 0 до 10^9.

Формат выходных данных

Для каждого запроса типа `get` нужно вывести ответ.

Пример

stdin	stdout
5 1 2 3 4 5 6 get 1 5 3 set 2 4 2 get 1 5 3 reverse 1 2 get 2 5 2 get 1 1 2	3 1 3 1

Задача 3C. Connect and Disconnect [1 sec, 256 mb]

Вы когда-нибудь слышали про обход в глубину? Например, используя этот алгоритм, вы можете проверить является ли граф связным за время $O(E)$. Вы можете даже посчитать количество компонент связности за то же время.

А вы когда нибудь слышали про систему непересекающихся множеств? Используя эту структуру, вы можете быстро обрабатывать запросы "Добавить ребро в граф" и "Посчитать количество компонент связности в графе".

А вы когда-нибудь слышали о *динамической* задаче связности? В этой задаче вам необходимо обрабатывать три типа запросов:

1. Добавить ребро в граф.
2. Удалить ребро из графа.
3. Посчитать количество компонент связности в графе.

Граф является неориентированным. Изначально граф пустой.

Формат входных данных

В первой строке находятся два целых числа N и K — количество вершин и количество запросов, сответственно ($1 \leqslant N \leqslant 300\,000$, $0 \leqslant K \leqslant 300\,000$). Следующие K строк содержат запросы, по одному в строке. Каждый запрос имеет один из трех типов:

1. `+` u v: Добавить ребро между вершинами u и v. Гарантируется, что такого ребра нет.
2. `-` u v: Удалить ребро между u и v. Гарантируется, что такое ребро есть.
3. `?`: Посчитать количество компонент связности в графе.

Вершины пронумерованы целыми числами от 1 до N. Во всех запросах $u \neq v$.

Формат выходных данных

Для каждого запроса '`?`' выведите количество компонент связности в момент запроса.

Пример

stdin	stdout
5 11 ? + 1 2 + 2 3 + 3 4 + 4 5 + 5 1 ? - 2 3 ? - 4 5 ?	5 1 1 2

Гробы

Задача 3D. Динамический Лес [0.8 sec, 256 mb]

Вам нужно научиться обрабатывать 3 типа запросов:

1. Добавить ребро в граф (`link`).
2. Удалить ребро из графа (`cut`).
3. По двум вершинам a и b вернуть длину пути между ними (или -1, если они лежат в разных компонентах связности) (`get`).

Изначально граф пустой (содержит N вершин, не содержит ребер). Гарантируется, что в любой момент времени граф является лесом. При добавлении ребра гарантируется, что его сейчас в графе нет. При удалении ребра гарантируется, что оно уже добавлено.

Формат входных данных

Числа N и M ($1 \leqslant N \leqslant 10^5 + 1$, $1 \leqslant M \leqslant 10^5$) — количество вершин в дереве и, соответственно, запросов. Далее M строк, в каждой строке команда (`link` или `cut`, или `get`) и 2 числа от 1 до N — номера вершин в запросе.

Формат выходных данных

В выходной файл для каждого запроса `get` выведите одно число — расстояние между вершинами, или -1, если они лежат в разных компонентах связности.

Пример

stdin	stdout
3 7 get 1 2 link 1 2 get 1 2 cut 1 2 get 1 2 link 1 2 get 1 2	-1 1 -1 1
5 10 link 1 2 link 2 3 link 4 3 cut 3 4 get 1 2 get 1 3 get 1 4 get 2 3 get 2 4 get 3 4	1 2 -1 1 -1 -1

Для тестирования

Задача 3E. Машенька и её интерес [2 sec, 256 mb]

Есть n мальчиков и девочка Маша. Изначально каждый мальчик стоит сам по себе и с точки зрения Маши имеет нулевую интересность. Девочка Маша хочет провести некоторый эксперимент, в течение которого каждый мальчик стоит в некоторой шеренге. Мальчики несговорчивые, участвовать в эксперименте не хотят, поэтому Маша собирается прибегнуть к математическому моделированию. Для этого ей нужно научиться быстро обрабатывать следующие запросы:

- `link(a, b)` – взять мальчиков с номерами a и b, если они стоят в разных шеренгах, то объединить шеренгу в одну: в начале шеренга мальчика a, затем шеренга мальчика b.
- `split(a, k)` – взять шеренгу, в которой стоит мальчик с номером a и разбить её на две: первые k мальчиков и все остальные. Если размер шеренги не больше k, ничего делать не нужно.
- `interest(a, x)` – сделать интересность мальчика a равной x (целое от 0 до 10^9).
- `sum(a)` – суммарная интересность мальчиков в шеренге, в которой стоит мальчик a.

Формат входных данных

В первой строке n ($1 \leqslant n \leqslant 100\,000$) – количество мальчиков и m ($1 \leqslant m \leqslant 250\,000$) – количество запросов. Далее m строк. Для понимания формата смотри пример. Мальчики нумеруются числами от 1 до n.

Формат выходных данных

Для каждого запроса "`sum`" на отдельной строке одно число – суммарная интересность.

Примеры

stdin	stdout
5 12 sum 1 interest 5 10 sum 5 interest 3 7 link 3 1 link 3 5 sum 1 interest 1 20 sum 1 split 1 2 sum 3 sum 5	0 10 17 37 27 10