Задача A. Динамический Лес

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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.

Примеры

стандартный ввод	стандартный вывод
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 B. Дерево

Имя входного файла:	`treenum.in`
Имя выходного файла:	`treenum.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Рассмотрим корневое дерево. Изначально дерево состоит из одного лишь корня. На сыновьях каждой вершины определён порядок слева направо. Допустимые операции с деревом таковы:

- Добавить в дерево лист.
- Удалить из дерева лист.
- Найти количество вершин на пути между двумя листьями.
- Найти количество вершин «под путём» между двумя листьями.

Множество вершин, лежащих «под путём» между листьями u и v, определяется следующим образом. Рассмотрим путь $u = w_0 - w_1 - \ldots - w_k = v$ между ними. Выделим в нём ту вершину w_c, в которой оба ребра пути идут в её сыновей. Пусть для определённости левое из этих рёбер приближает нас к вершине u, а правое — к вершине v. Тогда лежащими «под путём» объявляются следующие вершины:

- Все сыновья w_c, лежащие между w_{c-1} и w_{c+1}, а также все вершины их поддеревьев;
- Для $i = 1, 2, \ldots, c-1$ — все сыновья w_i, лежащие правее сына w_{i-1}, а также все вершины их поддеревьев;
- Для $i = c+1, c+2, \ldots, k-1$ — все сыновья w_i, лежащие левее сына w_{i+1}, а также все вершины их поддеревьев.

Напишите программу, которая по последовательности запросов перестраивает дерево согласно запросам на добавление и удаление вершин, а также вычисляет ответы на запросы о количестве вершин на пути и «под путём».

Формат входных данных

В первой строке ввода содержится одно целое число n — количество запросов ($0 \leqslant n \leqslant 300\,000$). Каждая из следующих n строк описывает один запрос. Возможные виды запросов таковы:

- `l` x — добавить новый лист как самого левого сына вершины x
- `r` x — добавить новый лист как самого правого сына вершины x
- `a` x y — добавить новый лист как сына вершины x, находящегося непосредственно справа от вершины y; все сыновья x, находившиеся до этого справа от y, после добавления оказываются правее новой вершины; гарантируется, что y является сыном x.
- `d` x удалить вершину x. Гарантируется, что в этот момент вершина x не удалена и является листом.
- `p` x y найти количество вершин на пути между x и y, включая сами эти вершины; гарантируется, что x и y являются листьями
- `q` x y найти количество вершин «под путём» между x и y, включая сами эти вершины; гарантируется, что x и y являются листьями.

Добавляемые вершины нумеруются с единицы в том порядке, в котором они добавляются в запросах. Корень дерева имеет номер 0 и листом ни в какой момент времени не считается.

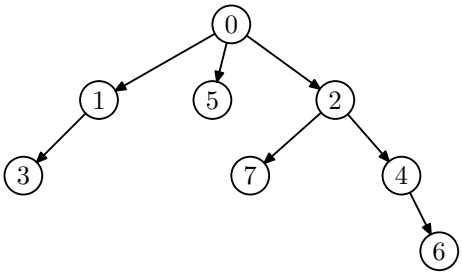
Формат выходных данных

Выполняя запросы по порядку, на каждый запрос вида `p` или `q` выведите ответ на него на отдельной строке.

Примеры

treenum.in	treenum.out
10 l 0 r 0 l 1 r 2 a 0 1 r 4 p 6 5 l 2 q 3 6 d 7	5 2

Замечание



На рисунке показано дерево до выполнения последнего запроса — удаления вершины 7.
Путь между вершинами 6 и 5 содержит пять вершин: 6 – 4 – 2 – 0 – 5.
«Под путём» между вершинами 3 и 6 находится две вершины — 5 и 7.
Эту задачу надо решать онлайн. Оффлайн решение получит Ignored.

Задача C. Связность и несвязность

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	8 секунды
Ограничение по памяти:	512 мегабайт

Это интерактивная задача. Ваше решение должно выводить ответ на каждый запрос до считывания следующего запроса.

Вы знаете всё про поиск в глубину? Например, используя его, можно определить, является ли граф связным за $O(E)$. А также, за то же самое время, можно найти количество компонент связности.

Вы знаете всё про систему непересекающихся множеств? С её помощью можно быстро отвечать на запросы «добавить ребро в граф» и «посчитать количество компонент связности графа».

И умеете ли вы решать Dynamic Connectivity Problem? В этой задаче нужно отвечать на следующие запросы:

1. добавить ребро в граф,
2. удалить ребро из графа и
3. посчитать количество компонент связности.

Формат входных данных

В начале граф пустой.

В первой строке файла содержится два целых числа n ($1 \leqslant n \leqslant 300\,000$) и k ($1 \leqslant k \leqslant 300\,000$) — число вершин и число запросов, соответственно. Далее в k строках содержатся сами запросы, по одному в строке. Есть три типа запросов:

1. `+` u v — добавить ребро между верщинами u и v. Гарантируется, что такого ребра ещё нет.
2. `-` u v — удалить ребро между верщинами u и v. Гарантируется, что такое ребро есть.
3. `?` — посчитать число компонент связности графа в текущий момент времени.

Вершины нумеруются от 1 до n. Во всех запросах $u \neq v$. Граф в этой задаче — неориентированный.

Формат выходных данных

Для каждого запроса `?`, выведите в своей строке число компонент связности.

Примеры

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

Задача D. Decremental Minimum Spanning Forest

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	8 секунд
Ограничение по памяти:	256 мегабайт

Дан взвешенный граф на n вершинах. Вам нужно уметь **online** удалять из него ребра, а также сообщать сумму весов ребер в минимальном основном лесу графа после каждого удаления.

Формат входных данных

В первой строке ввода находятся числа n и m. ($1 \leqslant n \leqslant 10^5, 1 \leqslant m \leqslant 2 \times 10^5$).

В следующих m строках находятся числа u_i, v_i, w_i, задающие рёбра исходного графа $1 \leqslant w_i \leqslant 10^9$.

В последней строке ввода находятся m чисел — перестановка p длины m, задающая порядок удаления рёбер из графа. На i-м шаге будет удалено ребро p_i.

Формат выходных данных

Выведите m чисел — ответ на задачу.

Пример

стандартный ввод	стандартный вывод
4 4 1 2 1 2 3 2 3 4 5 4 1 4 2 1 4 3	10 9 5 0

Замечание

Тесты для локального тестирования можно найти на страничке параллели.