,

Задача A. Соединение и разъединение

Имя входного файла:	`connect.in`
Имя выходного файла:	`connect.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

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.

Можно считать, что граф является неориентированным. Изначально граф является пустым.

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 типа '`?`', Выведите количество компонент связности в момент запроса.

Примеры

connect.in	connect.out
5 11 ? + 1 2 + 2 3 + 3 4 + 4 5 + 5 1 ? - 2 3 ? - 4 5 ?	5 1 1 2

Задача B. Динамическая рёберная двусвязность

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

Дан неориентированный мультиграф на n вершинах, пронумерованных числами от 1 до n. Изначально в графе нет ни одного ребра.

Нужно обработать q запросов трёх видов:

- `+ u v` — добавить в граф ребро между вершинами u и v; это ребро получает номер, равный номеру текущего запроса;
- `- i` — удалить из графа ребро, добавленное запросом номер i;
- `? u v` — вывести `YES`, если вершины u и v лежат в одной компоненте рёберной двусвязности текущего графа, и `NO` иначе.

Формат входных данных

В первой строке заданы два целых числа n и q — количество вершин графа и количество запросов ($1 \leqslant n \leqslant 3 \cdot 10^5$, $1 \leqslant q \leqslant 3 \cdot 10^5$).

В следующих q строках заданы запросы, по одному в строке. Запрос имеет один из трёх видов:

- `+ u v`, где $1 \leqslant u \leqslant n$, $1 \leqslant v \leqslant n$;
- `- i`, где i — номер запроса;
- `? u v`, где $1 \leqslant u \leqslant n$, $1 \leqslant v \leqslant n$.

Символ и числа разделены одним пробелом. Запросы нумеруются целыми числами от 1 до q в порядке следования.

Формат выходных данных

Для каждого запроса вида `?` выведите в отдельной строке `YES`, если вершины u и v лежат в одной компоненте рёберной двусвязности текущего графа, и `NO` иначе.

Примеры

стандартный ввод	стандартный вывод
5 13 + 1 2 ? 1 2 + 1 2 ? 1 2 - 3 ? 1 2 + 2 3 + 3 1 ? 1 3 ? 1 4 ? 4 4 - 7 ? 1 3	NO YES NO YES NO YES NO
3 9 + 1 1 ? 1 1 ? 1 2 + 1 2 + 2 3 + 3 1 ? 2 3 - 4 ? 2 3	YES NO YES NO

,

Задача C. Компоненты сильной связности

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

Дан ориентированный граф на n вершинах, занумерованных числами от 1 до n. Изначально в графе нет ни одного ребра.

Далее выполняется m операций. Операция номер i задаётся парой (u_i, v_i) и добавляет в граф ориентированное ребро из вершины u_i в вершину v_i.

После каждой операции требуется вывести текущее число компонент сильной связности графа.

Формат входных данных

В первой строке заданы два целых числа n и m ($1 \leqslant n \leqslant 3 \cdot 10^5$, $1 \leqslant m \leqslant 3 \cdot 10^5$) — количество вершин графа и количество добавляемых рёбер.

В следующих m строках заданы добавляемые рёбра в том порядке, в котором они появляются. В i-й из этих строк записаны два целых числа u_i и v_i ($1 \leqslant u_i \leqslant n$, $1 \leqslant v_i \leqslant n$) — ребро ведёт из вершины u_i в вершину v_i.

Никаких дополнительных ограничений на рёбра нет:

- допускаются петли, то есть возможно $u_i = v_i$;
- допускаются кратные рёбра, то есть пара (u_i, v_i) может повторяться произвольное число раз.

Формат выходных данных

Выведите m строк. В i-й строке должно быть записано одно целое число — количество компонент сильной связности графа после добавления первых i рёбер.

Примеры

стандартный ввод	стандартный вывод
4 5 1 2 2 3 3 1 1 4 4 1	4 4 2 2 1
3 6 1 1 1 2 1 2 2 1 3 3 2 3	3 3 3 2 2 2

Задача D. Динамический минимальный остовный лес

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

Дан взвешенный неориентированный граф на n вершинах, пронумерованных от 1 до n. Изначально в графе нет рёбер.

Требуется обработать q запросов двух типов:

- `+ u v w` — добавить в граф ребро между вершинами u и v веса w;
- `- u v` — удалить из графа ребро между вершинами u и v.

После каждого запроса выведите суммарный вес минимального остовного леса текущего графа. Если рёбер в графе нет, ответ равен 0.

Формат входных данных

В первой строке заданы два целых числа n и q — количество вершин графа и количество запросов ($1 \leqslant n \leqslant 3 \cdot 10^5$, $1 \leqslant q \leqslant 3 \cdot 10^5$).

В следующих q строках записаны запросы, по одному в строке, в порядке их выполнения. Запрос имеет один из двух видов:

- `+ u v w` — добавить ребро между u и v веса w ($1 \leqslant u, v \leqslant n$, $u \neq v$, $0 \leqslant w \leqslant 10^9$);
- `- u v` — удалить ребро между u и v ($1 \leqslant u, v \leqslant n$, $u \neq v$).

Гарантируется, что все запросы корректны, а именно:

- в запросе обоих типов $u \neq v$;
- в запросе типа `+` ребра между u и v в текущем графе **нет**;
- в запросе типа `-` ребро между u и v в текущем графе **есть**.

Формат выходных данных

Выведите q строк. В i-й строке должен быть вес минимального остовного леса графа после выполнения первых i запросов.

Примеры

стандартный ввод	стандартный вывод
4 8 + 1 2 5 + 2 3 3 + 1 3 4 + 3 4 10 - 1 3 - 2 3 + 1 4 1 - 1 2	5 8 7 17 18 15 16 11
3 6 + 1 2 0 + 2 3 7 + 3 1 7 - 1 3 - 2 1 + 1 3 4	0 7 7 7 7 11

Задача E. Ближайшая пара точек

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

Изначально множество точек на плоскости пусто. Затем последовательно обрабатываются q запросов двух видов:

- `+ x y` — добавить в множество точку с координатами (x, y);
- `- j` — удалить из множества точку, добавленную запросом номер j.

После каждого запроса нужно вывести **квадрат** расстояния между двумя ближайшими точками текущего множества, то есть минимум величины $(x_a - x_b)^2 + (y_a - y_b)^2$ по всем парам различных элементов множества. Если в множестве меньше двух точек, вывести -1.

Формат входных данных

В первой строке задано целое число q — количество запросов ($1 \leqslant q \leqslant 3 \cdot 10^5$).

В следующих q строках заданы запросы, по одному в строке. Запрос номер i ($1 \leqslant i \leqslant q$) имеет один из двух видов:

- `+ x y` — добавить точку с координатами (x, y), где $-10^9 \leqslant x \leqslant 10^9$ и $-10^9 \leqslant y \leqslant 10^9$;
- `- j` — удалить точку, добавленную запросом номер j, где $1 \leqslant j < i$.

Гарантируется, что для каждого запроса вида `- j`:

- запрос номер j — это запрос вида `+`;
- точка, добавленная запросом номер j, в этот момент присутствует в множестве, то есть она ещё не была удалена.

Формат выходных данных

Выведите q строк. В i-й строке выведите ответ после выполнения i-го запроса: квадрат расстояния между двумя ближайшими точками текущего множества или -1, если в множестве меньше двух точек.

Примеры

стандартный ввод	стандартный вывод
7 + 0 0 + 3 4 + 0 1 - 3 - 2 - 1 + 5 5	-1 25 1 25 -1 -1 -1
6 + 1000000000 1000000000 + -1000000000 -1000000000 + 1000000000 1000000000 - 3 - 1 + -1000000000 -999999999	-1 8000000000000000000 0 8000000000000000000 -1 1

,

Задача F. Раскрась одновременно

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мегабайт

Задана матрица, состоящая из n строк и m столбцов.
Вы можете выполнять над ней два типа действий:

- покрасить весь столбец в синий;
- покрасить всю строку в красный.

Обратите внимание, что вы не можете выбирать, в какой цвет красить строку или столбец.

За одну секунду можно выполнить как одно действие, так и несколько одновременно. Если сделать одно действие, то это будет бесплатно. Если же сделать $k > 1$ действий одновременно, то на это придется потратить k^2 монет. Когда несколько действий выполняются одновременно, для каждой клетки, которая затрагивается действиями обоих типов, цвет можно выбрать независимо.

Требуется обработать q запросов. Перед каждым запросом все ячейки становятся бесцветными. Изначально ограничений на итоговый цвет клеток нет. В i-м запросе добавляется ограничение вида:

- x_i y_i c_i — клетка в строке x_i в столбце y_i должна быть покрашена в цвет c_i.

Таким образом, после i запросов существует i ограничений на необходимые цвета клеток матрицы. После каждого запроса выведите минимальную стоимость покраски в соответствии с ограничениями.

Формат входных данных

В первой строке записано три целых числа n, m и q ($1 \leqslant n, m, q \leqslant 2 \cdot 10^5$) — размер матрицы и количество запросов.

В i-й из следующих q строк записаны два целых числа x_i, y_i и символ c_i ($1 \leqslant x_i \leqslant n$; $1 \leqslant y_i \leqslant m$; $c_i \in \{$`'R'`, `'B'`$\}$, где `'R'` значит красный, а `'B'` значит синий) — описание i-го ограничения. Клетки во всех ограничениях попарно различные.

Формат выходных данных

Выведите q целых чисел — После каждого запроса выведите минимальную стоимость покраски в соответствии с ограничениями.

Примеры

стандартный ввод	стандартный вывод
2 2 4 1 1 R 2 2 R 1 2 B 2 1 B	0 0 0 16
3 5 10 1 1 B 2 5 B 2 2 B 2 3 R 2 1 B 3 2 R 3 3 B 1 2 R 1 3 B 3 1 B	0 0 0 0 0 0 16 16 25 25

,

Задача G. 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 N. 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.

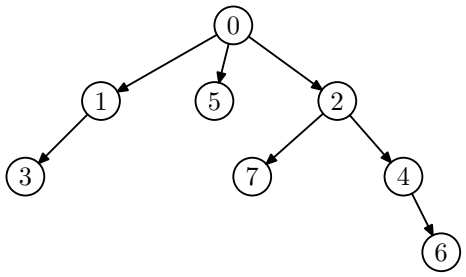
Формат выходных данных

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 I. 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 вершинами u и v. Гарантируется, что такого ребра ещё нет.
2. `-` u v — удалить ребро между вершинами 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 J. Decremental Minimum Spanning Forest

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	10 секунд
Ограничение по памяти:	512 мегабайт

Дан взвешенный граф на n вершинах. Вам нужно уметь 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

В контесте на Link Cut Tree нужно решать **offline**. В контесте на Dynamic Connectivity Online — **online**