Содержание

Задача A. Машенька и её интерес [0.5 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

Задача B. Динамический Лес [0.5 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 C. Connect and Disconnect [0.5 sec, 256 mb]

Do you know anything about DFS, Depth First Search? For example, using this method, you can determine whether a graph is connected or not in $O(E)$ time. You can even count the number of connected components in the same time.

Do you know anything about DSU, Disjoint Set Union? Using this data structure, you can process queries like "Add an edge to the graph" and "Count the number of connected components in the graph" fast.

And do you know how to solve Dynamic Connectivity Problem? In this problem, you have to process three types of queries fast:

1. Add an edge to the graph
2. Delete an edge from the graph
3. Count the number of connected components in the graph

Формат входных данных

At the first moment, the graph is empty.

The first line of file contains two integers N and K—number of vertices and number of queries ($1 \leqslant N \leqslant 300\,000$, $0 \leqslant K \leqslant 300\,000$). Next K lines contain queries, one per line. There are three types of queries:

1. `+` u v: add an edge between vertices u and v. It is guaranteed that there is no such edge in the graph at the time of the query.
2. `-` u v: remove an edge between vertices u and v. It is guaranteed that this edge is present in the graph at the time of the query.
3. `?`: count the number of connectivity components in the graph at the time of the query.

Vertices are numbered 1 through N. No query will have $u = v$. The graph is undirected.

Формат выходных данных

For each '`?`' query, output the number of connectivity components in the graph at the time of the query on a single line.

Пример

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