Задача A. Set

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Дано q запросов.

$+\ x$	$1 \leqslant x \leqslant 10^9$	`добавить x во множество`
$?\ x$	$1 \leqslant x \leqslant 10^9$	`проверить, что` $x \in set$

Формат входных данных

В первой строке находится единственное число q — количество запросов. ($1 \leqslant q \leqslant 10^6$).
В следующих q строках находятся сами запросы.

Формат выходных данных

В ответ на каждый запрос «?» выведите «YES» или «NO».

Пример

стандартный ввод	стандартный вывод
10 + 9 + 6 ? 4 ? 1 ? 10 + 7 ? 6 ? 3 + 2 ? 7	NO NO NO YES NO YES

Замечание

Напишите splay tree.

Задача B. Двоичное дерево поиска

Имя входного файла:	`bst.in`
Имя выходного файла:	`bst.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Реализуйте сбалансированное двоичное дерево поиска.

Формат входных данных

Входной файл содержит описание операций с деревом, их количество не превышает 100000. В каждой строке находится одна из следующих операций:

- `insert` x — добавить в дерево ключ x. Если ключ x в дереве уже есть, то ничего делать не надо.
- `delete` x — удалить из дерева ключ x. Если ключа x в дереве нет, то ничего делать не надо.
- `exists` x — если ключ x есть в дереве, выведите «`true`», иначе «`false`»
- `next` x — выведите минимальный элемент в дереве, строго больший x, или «`none`», если такого нет.
- `prev` x — выведите максимальный элемент в дереве, строго меньший x, или «`none`», если такого нет.

Все числа во входном файле целые и по модулю не превышают 10^9.

Формат выходных данных

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`. Следуйте формату выходного файла из примера.

Примеры

bst.in	bst.out
insert 2 insert 5 insert 3 exists 2 exists 4 next 4 prev 4 delete 5 next 4 prev 4	true false 5 3 none 3

Задача C. Асхат и дерево

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

Это интерактивная задача.

Дано двоичное дерево поиска размера n, в каждой вершине есть значение от 1 до n. Петух по имени Асхат каждый день нумерует вершины числами от 1 до n, даёт вам номер корня r и просит вас найти номер вершины со значением x.

Вы можете совершать запросы — по номеру вершины узнать значение в ней и номера её детей. Пусть максимальное расстояние от корня до вершины в i-й день равно d. Тогда в i-й день вы можете совершить не более, чем d запросов. За все дни вы можете совершить не более, чем 70000 запросов.

Также вы можете менять детей у каждой вершины. Пусть в i-й день длина пути от корня до вершины с номером x равна s. Тогда вам в i-й день разрешено сделать не более, чем s запросов вида «установить у вершины новых детей». За все дни вы можете совершить не более, чем 70000 запросов этого типа.

Протокол взаимодействия

В первой строке ввода даны числа n и q ($1 \leqslant n \leqslant 2000$, $1 \leqslant q \leqslant 2000$) — размер дерева и число запросов, соответственно.

Для каждого запроса даны числа r и x ($1 \leqslant r, x \leqslant n$) — номер корня дерева и значение, вершину с которым требуется найти. Вы не сможете считать эти числа для следующего запроса, пока не дадите ответ на текущий.

Чтобы обратиться к вершине с номером i выведите «`val i`» в отдельной строке. В ответ даются три числа val, L и R ($1 \leqslant val \leqslant n$, $0 \leqslant L, R \leqslant n$) — значение в этой вершине и номера левого и правого ребёнка, соответственно. В случае, если у вершины нет левого или правого ребёнка, $L = 0$ или $R = 0$, соответственно.

Чтобы поменять детей вершины с номером i на вершины с номером L и R выведите «`change i L R`» в отдельной строке. Чтобы у вершины с номером i не было левого или правого ребёнка, выведите 0 вместо L или R, соответственно. После выполнения этого запроса граф может перестать быть двоичным деревом поиска.

Если искомое значение находится в вершине с номером i и вы совершили все нужные изменения, выведите «`confirm i`» в отдельной строке. После этого вершины перенумеруются, а на ввод будет дан новый запрос. Если на момент выполнения этого запроса граф не является двоичным деревом поиска, вы получите вердикт «Wrong answer».

Задача D. Переворот

Имя входного файла:	`reverse.in`
Имя выходного файла:	`reverse.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дан массив. Надо научиться обрабатывать два типа запросов.

- `1 L R` - перевернуть отрезок [L, R]
- `2 L R` - найти минимум на отрезке [L, R]

Формат входных данных

Первая строка файла содержит два числа n, m. ($1 \leqslant n, m \leqslant 10^5$) Во второй строке находится n чисел a_i ($1 \leqslant a_i \leqslant 10^9$)- исходный массив. Остальные m строк содержат запросы, в формате описанном в условии. Для чисел L,R выполняется ограничение ($1 \leqslant L \leqslant R \leqslant n$).

Формат выходных данных

На каждый запрос типа 2, во входной файл выведите ответ на него, в отдельной строке.

Примеры

reverse.in	reverse.out
10 7 5 3 2 3 12 6 7 5 10 12 2 4 9 1 4 6 2 1 8 1 1 8 1 8 9 2 1 7 2 3 6	3 2 2 2

Задача E. 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 F. Динамический Лес

Имя входного файла:	`linkcut.in`
Имя выходного файла:	`linkcut.out`
Ограничение по времени:	1 секунда
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 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.

Примеры

linkcut.in	linkcut.out
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

Задача G. lca

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Есть граф на n вершинах. Изначально в нем нет ребер. Вам нужно обработать q запросов.

link u v	$1 \leqslant u, v \leqslant n$	*add the edge* $< u, v >$
cut u v	$1 \leqslant u, v \leqslant n$	*delete the edge* $< u, v >$
lca u v r	$1 \leqslant u, v, r \leqslant n$	*find lca*(u, v) *when root is r*

Формат входных данных

В первой строке находятся два числа n и q — количество вершин в графе и количество запросов соответственно. $(1 \leqslant n, q \leqslant 10^5)$.

В следующих q строках находятся сами запросы.

Формат выходных данных

В ответ на каждый запрос типа `get` выведите единственное число — lca вершин из запроса.

Примеры

стандартный ввод	стандартный вывод
10 5 link 5 6 lca 2 1 1 link 7 2 cut 7 2 cut 5 6	-1
10 10 link 5 6 lca 2 1 1 link 7 2 cut 7 2 cut 5 6 link 6 2 link 8 7 lca 9 8 3 cut 6 2 lca 6 10 3	-1 -1 -1

Задача H. Минимальный номер вершины на пути

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Есть граф на n вершинах. Изначально в нем нет ребер. Вам нужно обработать q запросов.

link u v	$1 \leqslant u, v \leqslant n$	*add the edge* $< u, v >$
cut u v	$1 \leqslant u, v \leqslant n$	*delete the edge* $< u, v >$
get u v	$1 \leqslant u, v \leqslant n$	*find* $\min p \in path(u, v)$

Формат входных данных

В первой строке находятся два числа n и q — количество вершин в графе и количество запросов соответственно. ($1 \leqslant n, q \leqslant 10^5$).

В следующих q строках находятся сами запросы.

Формат выходных данных

В ответ на каждый запрос типа `get` выведите единственное число — минимальный номер вершины на пути из запроса (или -1, если пути нет).

Примеры

стандартный ввод	стандартный вывод
10 5 link 5 6 get 2 1 get 9 7 link 2 4 link 6 2	-1 -1
10 10 link 5 6 get 2 1 get 9 7 link 2 4 link 6 2 link 8 7 get 9 8 link 9 6 get 6 10 get 3 9	-1 -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 1 -1 2 -1 -1

Задача I. Кратчайший путь

Имя входного файла:	`shortest.in`
Имя выходного файла:	`shortest.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В некотором царстве, в некотором государстве есть несколько городов, соединённых двусторонними дорогами. При этом страна разбита на провинции, внутри каждой провинции можно добраться от любого города до любого другого, а между провинциями пути не существует. Внутри каждой провинции есть один выделенный город, который называется столицей этой провинции.

Компания «AloeVera» занимается пассажирскими перевозками в этом государстве. Компания перевозит людей от одного города до другого по кратчайшему пути. Однако, дорожная сеть постоянно развивается, поэтому периодически в государстве появляются новые дороги. Строительство новой дороги — очень важное событие, поэтому при появлении новой дороги все стремятся проехать именно по ней. Более точно, при добавлении новой дороги между городами u и v существует две возможности:

1. Города u и v находятся в разных провинциях. Тогда после добавления этой дороги эти две провинции объединяются, при этом столицей становится столица первой провинции (той, в которой до добавления находилась вершина u).

2. Города u и v находятся в одной провинции. Тогда происходит перераспределение транспортных потоков. Услышав об открытии новой дороги, все продолжают добираться из столицы в другие города провинции по кратчайшему пути, но в случае, если таких путей существует несколько, то предпочтение отдается тому из них, на котором находится новая построенная дорога. После этого, если найдутся дороги, по которым никто ездить не будет, они навсегда удаляются из дорожной сети. Будьте внимательны: может так случиться, что по новой дороге никто ездить не будет, и тогда её не нужно добавлять вообще.

Иногда в государстве некоторые дороги приходят в негодность. При этом, если после удаления этой дороги некоторая часть провинции становится недостижимой из столицы, то она становится независимой провинцией, и её столицей назначается город, находящийся в этой провинции и являющийся концом удалённой дороги.

Основная же задача компании — сообщать людям о кратчайшем расстоянии между двумя какими-то городами. Ваша задача состоит в том, чтобы автоматизировать работу «AloeVera». При этом люди, сообщающие о разрушении дорог, иногда запаздывают или оперируют неверной информацией, и сообщают о разрушении дороги, которая уже была разрушена или вообще никогда не существовала.

Для решения поставленной задачи компания «AloeVera» использует следующий интерфейс:

- `1 u v` — запрос об удалении дороги между городами u и v. Будьте внимательны, эта дорога могла быть разрушена ранее, или вообще никогда не существовать!

- `2 u v w` — запрос о добавлении новой дороги между городами u и v, время проезда по которой будет равно w ($1 \leqslant w \leqslant 10^4$).

- `3 u v` — запрос кратчайшего расстояния от города u до города v. В случае, если города находятся в разных провинциях, длина полагается равной -1.

Во всех запросах номера городов корректны, то есть $1 \leqslant u, v \leqslant N$, где N обозначает общее число городов в государстве.

Формат входных данных

В первой строке входного файла содержится число N — количество городов в государстве. В следующей строке содержится число M — количество запросов. Далее в каждой строке содержится описание запроса. Описание соответствует условию.

$1 \leqslant N \leqslant 50\,000$, $1 \leqslant M \leqslant 100\,001$.

Формат выходных данных

Для каждого запроса типа 3 выведите единственное число: ответ на запрос.

Примеры

shortest.in	shortest.out
5 8 2 1 3 10 2 1 2 3 1 1 2 3 1 3 2 1 3 9 2 2 3 1 3 1 5 3 1 3	10 -1 9
5 10 2 2 3 2 2 3 1 2 2 1 5 2 2 5 4 2 2 2 3 2 2 1 5 2 1 3 1 2 3 1 1 2 3 1 2 3 3 1	1

Задача J. Connect

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 мегабайт

Дан список из m ребер, пронумерованных от 1 до m. Посчитайте количество подотрезков $[l, r]$ ($1 \leqslant l \leqslant r \leqslant m$) таких, что граф на n вершинах, содержащий только ребра из этого подотрезка, является связным.

Формат входных данных

Первая строка входных данных содержит два числа n и m ($2 \leqslant n \leqslant 50000$, $1 \leqslant m \leqslant 200000$) — число вершин и ребер. Следующие m строк содержат описания ребер. На $i + 1$-й строке даны два целых числа u_i, v_i — концы i-го ребра ($1 \leqslant u_i, v_i \leqslant n$).

Формат выходных данных

Выведите ответ на задачу.

Пример

стандартный ввод	стандартный вывод
4 4 1 2 2 4 1 3 1 4	3

Задача K. Поднимайся и вращай

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

Изначально у вас есть n чисел $1, 2, \dots n$. Каждое живёт само по себе. Далее числа будут объединяться в массивы. Вам нужно реализовать структуру, данных, умеющую отвечать на несколько запросов, за $\emptyset(\log n)$ каждый.

- `+ i j` – взять массивы, в которых живут числа i и j, и объединить их в один массив именно в таком порядке.
- `! i k` – взять массив, в котором живёт число i, и повернуть его на k влево.
- `- i k` – взять массив, в котором живёт число i, и отрезать первые k элементов. Получится два новых массива.

Гарантируется, что запросы корректны. В первом i и j живут в разных массивах, во втором и третьем, длина массива, содержащего i, строго больше k.

Формат входных данных

На первой строке число элементов n ($2 \leqslant n \leqslant 100\,000$) и число запросов m ($1 \leqslant m \leqslant 100\,000$).
На следующих m строках сами запросы.

Формат выходных данных

После всех запросов нужно вывести получившиеся массивы. На первой строке выведите число массивов. Далее k массивов в формате «число элементов и сами элементы». Отсортируйте массив массивов перед выводом (массивы сравниваются лексикографически).

Пример

стандартный ввод	стандартный вывод
5 5 + 1 2 + 1 3 + 1 4 ! 3 1 - 2 1	3 1 2 3 3 4 1 1 5