Задача A. Всем чмоки в этом чатике!

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

Сегодня Мэри, как программисту социальной сети «Телеграфчик», предстоит реализовать сложную систему управления чатами.

Задача Мэри усложняется тем, что в социальную сеть «Телеграфчик» внедрена продвинутая система шифрования «ZergRus», простая, как всё гениальное. Суть её в том, что в системе хранится одна переменная zerg, которая принимает значения от 0 (включительно) до $p = 10^6 + 3$ (исключая p) и меняется в зависимости от событий в системе.

В социальной сети всего n пользователей ($1 \leqslant n \leqslant 10^5$). В начале дня каждый пользователь оказывается в своём собственном чате, в котором больше никого нет. Переменная zerg в начале дня устанавливается равной 0.

В течение дня происходят события типов:

1. Участник с номером $(i + zerg) \bmod n$ посылает сообщение всем участникам, сидящим с ним в чате (в том числе и себе самому), при этом переменная zerg заменяется на $(30 \cdot zerg + 239) \bmod p$.
2. Происходит слияние чатов, в которых сидят участники с номерами $(i + zerg) \bmod n$ и $(j + zerg) \bmod n$. Если участники и так сидели в одном чате, то ничего не происходит. Если в разных, то чаты объединяются, а переменной zerg присваивается значение $(13 \cdot zerg + 11) \bmod p$.
3. Участник с номером $(i + zerg) \bmod n$ хочет узнать, сколько сообщений он не прочитал, и прочитать их. Если участник прочитал q новых сообщений, то переменной zerg присваивается значение $(100\,500 \cdot zerg + q) \bmod p$.

Вы поможете Мэри реализовать систему, обрабатывающую эти события?

Формат входных данных

В первой строке входного файла записаны натуральные числа n ($1 \leqslant n \leqslant 10^5$) — число пользователей социальной сети. и m ($1 \leqslant m \leqslant 3 \cdot 10^5$) — число событий, произошедших за день. В следующих m строках содержится описание событий. Первое целое число в строке означает тип события t ($1 \leqslant t \leqslant 3$). Если $t = 1$, далее следует число i ($0 \leqslant i < n$), по которому можно вычислить, какой участник послал сообщение. Если $t = 2$, далее следуют числа i и j ($0 \leqslant i, j < n$), по которым можно вычислить номера участников, чаты с которыми должны объединиться. Если $t = 3$, далее следует число i ($0 \leqslant i < n$), по которому можно вычислить номер участника, желающего узнать, сколько у него сообщений, и прочитать их.

Формат выходных данных

Для каждого события типа 3 нужно вывести число непрочитанных сообщений у участника.

Примеры

chat.in	chat.out	Пояснение
4 10 1 0 1 2 1 1 1 2 3 1 2 1 2 1 3 3 3 2 3 2 3 2	1 1 2	4 10 1 0 1 1 1 2 1 3 3 0 2 0 1 1 1 3 0 2 2 1 3 1

Задача B. LCA - 2

Имя входного файла:	`lca2.in`
Имя выходного файла:	`lca2.out`
Ограничение по времени:	8 секунды
Ограничение по памяти:	256 мегабайт

Задано подвешенное дерево, содержащее n ($1 \leqslant n \leqslant 100\,000$) вершин, пронумерованных от 0 до $n-1$. Требуется ответить на m ($1 \leqslant m \leqslant 10\,000\,000$) запросов о наименьшем общем предке для пары вершин.

Запросы генерируются следующим образом. Заданы числа a_1, a_2 и числа x, y и z. Числа $a_3, \ldots, a_{2m}$ генерируются следующим образом: $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. Первый запрос имеет вид $\langle a_1, a_2 \rangle$. Если ответ на $i-1$-й запрос равен v, то i-й запрос имеет вид $\langle (a_{2i-1} + v) \bmod n, a_{2i} \rangle$.

Формат входных данных

Первая строка содержит два числа: n и m. Корень дерева имеет номер 0. Вторая строка содержит $n-1$ целых чисел, i-е из этих чисел равно номеру родителя вершины i. Третья строка содержит два целых числа в диапазоне от 0 до $n-1$: a_1 и a_2. Четвертая строка содержит три целых числа: x, y и z, эти числа неотрицательны и не превосходят 10^9.

Формат выходных данных

Выведите в выходной файл сумму номеров вершин — ответов на все запросы.

Примеры

lca2.in	lca2.out
3 2 0 1 2 1 1 1 0	2
1 2 0 0 1 1 1	0

Задача C. LCA-3

Имя входного файла:	`lca3.in`
Имя выходного файла:	`lca3.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Подвешенное дерево —это ориентированный граф без циклов, в котором в каждую вершину, кроме одной, называемой *корнем* ориентированного дерева, входит одно ребро. В корень ориентированного дерева не входит ни одного ребра. *Отцом* вершины называется вершина, ребро из которой входит в данную.

(по материалам Wikipedia)

Дан набор подвешенных деревьев. Требуется выполнять следующие операции:

1. 0 u v Для двух заданных вершин u и v выяснить, лежат ли они в одном дереве. Если это так, вывести вершину, являющуюся их наименьшим общим предком, иначе вывести 0.
2. 1 u v Для корня u одного из деревьев и произвольной вершины v другого дерева добавить ребро (v, u). В результате эти два дерева соединятся в одно.

Вам необходимо выполнять все операции online, т.е. вы сможете узнать следующий запрос только выполнив предыдущий.

Формат входных данных

На первой строке входного файла находится число n — суммарное количество вершин в рассматриваемых деревьях, $1 \leqslant n \leqslant 50000$. На следующей строке расположено n чисел — предок каждой вершины в начальной конфигурации, или 0, если соответствующая вершина является корнем. Затем следует число k — количество запросов к вашей программе, $1 \leqslant k \leqslant 100000$. Каждая из следующих строк содержит по три целых числа: вид запроса (0 — для поиска LCA или 1 — для добавления ребра) и два числа x, y. Вершины, участвующие в запросе можно вычислить по формуле: $u = (x - 1 + ans) \bmod n + 1$, $v = (y - 1 + ans) \bmod n + 1$, где ans - ответ на последний запрос типа 0 ($ans = 0$ для первого запроса).

Формат выходных данных

Для каждого запроса типа 0, выведите в выходной файл одно число на отдельной строке — ответ за этот запрос.

Примеры

lca3.in	lca3.out
5 0 0 0 0 0 12 1 5 3 0 2 5 1 4 2 1 1 5 0 1 5 1 3 4 0 1 5 0 3 1 0 4 2 0 1 4 0 5 2 0 4 1	0 5 5 3 2 3 3 2

Задача D. Ещё одна задача про деревья

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

Есть граф из N вершин. Изначально он пустой. Нужно обработать M запросов:

1. добавить ребро из вершины v_1 в вершину v_2, при этом вершины v_1 и v_2 находятся в разных деревьях и вершина v_2 является корнем какого-то дерева.
2. по двум вершинам a и b определить, лежат ли они в одном дереве.

Решение задачи: реализуем СНМ с эвристикой сжатия путей:

```
int n, m, l[NMAX];

int calc_leader (int v)
{
        if (l[v] != v)
                l[v] = calc_leader (l[v]);
        return l[v];
}

int main()
{
        scanf ("%d%d", &n, &m);
        for (int i = 1; i <= n; i ++)
                l[i] = i;
        for (int i = 1; i <= m; i ++)
        {
                int x, y, z;
                scanf ("%d%d%d", &z, &x, &y);
                if (z == 1)
                        l[y] = x;
                else
                if (calc_leader (x) == calc_leader (y))
                        printf ("YES\n");
                else
                        printf ("NO\n");
        }
}
```

Вам же предстоит сделать тест, на котором это решение будет работать долго. Более точно, нужно сделать тест, на котором количество вызовов функции `calc_leader` будет не меньше, чем $\frac{1}{4} M \log_2 M$.

Формат входных данных

Входной файл содержит два целых числа N и M. В тестах, отличных от примера $N = 1000000, M = 100000$.

Формат выходных данных

Выведите M строк. i-ая строка должна иметь вид `1` x y, если i-ый запрос заключается в добавлении ребра из вершины x в вершину y, или `0` x y, если спрашивается, лежат ли вершины x и y в одном дереве.

Примеры

trees.in	trees.out
2 2	1 2 1 0 1 1

Замечание

На тест из примера будет зачтен любой ответ, который удовлетворяет формату вывода.