Задача A. Быстрое прибавление

Имя входного файла:	`fastadd.in`
Имя выходного файла:	`fastadd.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Есть массив целых чисел длины $n = 2^{24}$, изначально заполненных нулями. Вам нужно сперва обработать m случайных запросов вида "прибавление на отрезке". Затем обработать q случайных запросов вида "сумма на отрезке".

Формат входных данных

На первой строке числа m, q ($1 \leqslant m, q \leqslant 2^{24}$). На второй строке пара целых чисел a, b от 1 до 10^9, используемая в генераторе случайных чисел.

```
0. unsigned int a, b; // даны во входных данных
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur » 8; // число от 0 до 2^24 - 1.
5. }
```

Каждый запрос первого вида генерируется следующим образом:

```
1. add = nextRand(); // число, которое нужно прибавить
2. l = nextRand();
3. r = nextRand();
4. if (l > r) swap(l, r); // получили отрезок [l..r]
```

Каждый запрос второго вида генерируется следующим образом:

```
1. l = nextRand();
2. r = nextRand();
3. if (l > r) swap(l, r); // получили отрезок [l..r]
```

Сперва генерируются запросы первого вида, затем второго.

Формат выходных данных

Выведите сумму ответов на все запросы второго типа по модулю 2^{32}.

Примеры

fastadd.in	fastadd.out
5 5 13 239	811747796

Замечание

Последовательность запросов в тесте из примера:

```
[13..170] += 0
[28886..375523] += 2221
[2940943..13131777] += 4881801
[2025901..10480279] += 4677840
[4943766..6833065] += 9559505
get sum [13412991..13937319]
get sum [1871500..6596736]
get sum [7552290..14293694]
get sum [1268651..16492476]
get sum [2210673..13075602]
```

Задача B. Разреженные таблицы

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

Дан массив из n чисел. Требуется написать программу, которая будет отвечать на запросы следующего вида: найти минимум на отрезке между u и v включительно.

Формат входных данных

В первой строке входного файла даны три натуральных числа n, m ($1 \leqslant n \leqslant 10^5$, $1 \leqslant m \leqslant 10^7$) и a_1 ($0 \leqslant a_1 < 16\,714\,589$) — количество элементов в массиве, количество запросов и первый элемент массива соответственно. Вторая строка содержит два натуральных числа u_1 и v_1 ($1 \leqslant u_1, v_1 \leqslant n$) — первый запрос.

Элементы $a_2, a_3, \ldots, a_n$ задаются следующей формулой:

$$a_{i+1} = (23 \cdot a_i + 21563) \bmod 16714589.$$

Например, при $n = 10$, $a_1 = 12345$ получается следующий массив: $a = (12345, 305498, 7048017, 11694653, 1565158, 2591019, 9471233, 570265, 13137658, 1325095)$.

Запросы генерируются следующим образом:

$$u_{i+1} = \big((17 \cdot u_i + 751 + ans_i + 2i) \bmod n\big) + 1,$$
$$v_{i+1} = \big((13 \cdot v_i + 593 + ans_i + 5i) \bmod n\big) + 1,$$

где ans_i — ответ на запрос номер i.

Обратите внимание, что u_i может быть больше, чем v_i.

Формат выходных данных

В выходной файл выведите u_m, v_m и ans_m (последний запрос и ответ на него).

Примеры

sparse.in	sparse.out
10 8 12345 3 9	5 3 1565158

Задача C. Прямоугольники

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

Есть таблица T размера $N \times M$. Элементами таблицы являются прямоугольники T_{ij}, где $0 \leqslant i < N$ и $0 \leqslant j < M$. Прямоугольник T_{ij} задаётся четвёркой чисел $(x_1^{ij}, y_1^{ij}, x_2^{ij}, y_2^{ij})$, где (x_1^{ij}, y_1^{ij}) и (x_2^{ij}, y_2^{ij}) — координаты противоположных углов прямоугльника. Стороны прямоугольника параллельны осям координат.

Далее вам поступают запросы. Каждый запрос состоит из четырёх чисел: (r_1, c_1, r_2, c_2). Ответом на такой запрос является площадь фигуры, являющейся пересечением всех прямоугольников T_{ij} таких, что $\min(r_1, r_2) \leqslant i \leqslant \max(r_1, r_2)$ и $\min(c_1, c_2) \leqslant j \leqslant \max(c_1, c_2)$. Запросов очень много, поэтому мы просим вас вывести сумму ответов на все запросы по модулю $10^9 + 7$.

Формат входных данных

В первой строке записаны два целых числа N и M — размеры таблицы T ($1 \leqslant N, M \leqslant 127$). Далее в N строках описывается таблица T: в $(i+1)$-й строке $(j+1)$-я четвёрка чисел $x_1^{ij}\ y_1^{ij}\ x_2^{ij}\ y_2^{ij}$ описывает прямоугольник T_{ij}. Гаранируется, что $|x_k^{ij}|, |y_k^{ij}| \leqslant 10^6$.

Дальше в отдельной строке записано четыре числа. Первое из них, число Q — количество запросов ($1 \leqslant Q \leqslant 5 \cdot 10^6$). Следующие три числа — это A, B, v_0 ($0 \leqslant A, B, v_0 < 10^9 + 7$). При помощи этих чисел генерируется бесконечная последовательность $\{v_i\}$ по правилу $v_i = (A \cdot v_{i-1} + B) \bmod (10^9 + 7)$.

После этого k-й запрос (запросы нумеруются с единицы) задаётся следующей четвёркой чисел: $(v_{4k-3} \bmod N, v_{4k-2} \bmod M, v_{4k-1} \bmod N, v_{4k} \bmod M)$.

Формат выходных данных

Выведите сумму ответов на все запросы по модулю $10^9 + 7$.

Примеры

pail.in	pail.out
2 2 0 0 2 2 1 1 3 3 0 3 2 1 1 2 3 0 1 500000003 4 2	1
3 2 8 -1 -7 6 6 8 9 10 -4 -10 4 9 -3 -8 6 9 -2 -9 3 8 -5 7 7 3 5 303164476 273973578 65779139	85

Замечание

В первом примере запрос имеет вид `(1,0,0,1)`, то есть это запрос ко всей таблице. Пересечением всех прямоугольников является квадрат с углами в точках $(1, 1)$ и $(2, 2)$. Его площадь равна 1.

Во втором примере запросы имеют вид `(0,1,1,1)`, `(1,0,2,0)`, `(0,0,2,1)`, `(0,1,1,1)`, `(0,1,0,0)`. На второй запрос ответ — 85, на остальные — 0.

Задача D. RMQ

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

Формат входных данных

В первой строке находится число n — размер массива. ($1 \leqslant n \leqslant 500000$) Во второй строке находится n чисел a_i — элементы массива. Далее содержится описание операций, их количество не превышает 1000000. В каждой строке находится одна из следующих операций:

- `set` i x — установить $a[i]$ в x.
- `min` i j — вывести значение минимального элемента в массиве на отрезке с i по j, гарантируется, что $(1 \leqslant i \leqslant j \leqslant n)$.

В массив помещаются только целые числа, не превышающие по модулю 10^9.

Формат выходных данных

Выведите последовательно результат выполнения всех операций `min`. Следуйте формату выходного файла из примера.

Пример

rmq.in	rmq.out
5 1 2 3 4 5 min 2 5 min 1 5 min 1 4 min 2 4 set 1 10 set 2 3 set 5 2 min 2 5 min 1 5 min 1 4 min 2 4	2 1 1 2 2 3 3

Задача E. RSQ

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

Формат входных данных

В первой строке находится число n — размер массива. ($1 \leqslant n \leqslant 500000$) Во второй строке находится n чисел a_i — элементы массива. Далее содержится описание операций, их количество не превышает 1000000. В каждой строке находится одна из следующих операций:

- `set` i x — установить $a[i]$ в x.
- `sum` i j — вывести значение суммы элементов в массиве на отрезке с i по j, гарантируется, что $(1 \leqslant i \leqslant j \leqslant n)$.

Все числа во входном файле и результаты выполнения всех операций не превышают по модулю 10^{18}

Формат выходных данных

Выведите последовательно результат выполнения всех операций `sum`. Следуйте формату выходного файла из примера.

Пример

rsq.in	rsq.out
5 1 2 3 4 5 sum 2 5 sum 1 5 sum 1 4 sum 2 4 set 1 10 set 2 3 set 5 2 sum 2 5 sum 1 5 sum 1 4 sum 2 4	14 15 10 9 12 22 20 10

Задача F. Криптография

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

Задано n матриц $A_1, A_2, \ldots, A_n$ размера 2×2. Необходимо для нескольких запросов вычислить произведение матриц $A_i, A_{i+1}, \ldots, A_j$. Все вычисления производятся по модулю r.

Формат входных данных

Первая строка входного файла содержит числа r ($1 \leqslant r \leqslant 10\,000$), n ($1 \leqslant n \leqslant 200\,000$) и m ($1 \leqslant m \leqslant 200\,000$). Следующие n блоков по две строки содержащие по два числа в строке — описания матриц. Затем следуют m пар целых чисел от 1 до n, запросы на произведение на отрезке.

Формат выходных данных

Выведите m блоков по две строки,по два числа в каждой — произведения на отрезках. Разделяйте блоки пустой строкой. Все вычисления производятся по модулю r

Пример

crypto.in	crypto.out
3 4 4 0 1 0 0 2 1 1 2 0 0 0 2 1 0 0 2 1 4 2 3 1 3 2 2	0 2 0 0 0 2 0 1 0 1 0 0 2 1 1 2

Задача G. RMQ

Имя входного файла:	`rmq2.in`
Имя выходного файла:	`rmq2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Формат входных данных

В первой строке находится число n — размер массива. ($1 \leqslant n \leqslant 100000$) Во второй строке находится n чисел a_i — элементы массива. Далее содержится описание операций, их количество не превышает 200000. В каждой строке находится одна из следующих операций:

- `set` i j x — установить все $a[k]$, $i \leqslant k \leqslant j$ в x.
- `add` i j x — увеличить все $a[k]$, $i \leqslant k \leqslant j$ на x.
- `min` i j — вывести значение минимального элемента в массиве на отрезке с i по j, гарантируется, что $(1 \leqslant i \leqslant j \leqslant n)$.

Все числа во входном файле и результаты выполнения всех операций не превышают по модулю 10^{18}

Формат выходных данных

Выведите последовательно результат выполнения всех операций `min`. Следуйте формату выходного файла из примера.

Пример

rmq2.in	rmq2.out
5 1 2 3 4 5 min 2 5 min 1 5 min 1 4 min 2 4 set 1 3 10 add 2 4 4 min 2 5 min 1 5 min 1 4 min 2 4	2 1 1 2 5 5 8 8

Задача N. Парковка

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

На кольцевой парковке есть n мест пронумерованых от 1 до n. Есть два вида событий прибытие машину на парковку и отъезд машины с парковки. Если машина приезжает на парковку, а её место занято, то она едет далее по кругу и встаёт на первое свободное место.

Формат входных данных

В первой строке входного файла находится два числа n и m — размер парковки и количество запросов($1 \leqslant n, m \leqslant 100000$). В следующих m строках находятся события. Каждая из этих строк имеет следующий вид:

- enter x — приехала машина, которая хочет встать на место x. Для каждой такой команды выведите какое место займёт эта машина.
- exit x — уехала машина занимавшая место x. Гарантируется, что на этом месте была машина.

Формат выходных данных

Выведите последовательно результаты выполнения всех операций `enter`.

Пример

parking.in	parking.out
3 5 enter 1 enter 1 exit 1 enter 2 enter 2	1 2 3 1

Задача I. Функция на отрезке

Имя входного файла:	`fun.in`
Имя выходного файла:	`fun.out`
Ограничение по времени:	3 seconds
Ограничение по памяти:	256 megabytes

Вам задан массив натуральных чисел $a_1, a_2, \ldots, a_n$.

Вам нужно ответить на несколько запросов посчитать функцию

$$f(l, r) = \sum_{x \in \mathbb{N}} K_x^2 \cdot x,$$

где K_x — число вхождений числа x в отрезок $a[l \ldots r]$.

Формат входных данных

В первой строке заданы числа n и m — размер массива и число запросов ($1 \leqslant n, m \leqslant 200\,000$). Во второй строке находится n чисел a_i — элементы массива ($1 \leqslant a_i \leqslant 10^6$). Далее содержится описание запросов: в каждой строке заданы два натуральных числа l и r — отрезок, на котором нужно посчитать функцию f ($1 \leqslant l \leqslant r \leqslant n$).

Формат выходных данных

Выведите m строк — ответы на запросы.

Примеры

fun.in	fun.out
3 2 4 7 4 2 3 1 3	11 23
8 4 3 3 6 6 3 7 3 3 2 3 1 4 2 2 1 8	9 36 3 106

Задача J. Окна

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

На экране расположены прямоугольные окна, каким-то образом перекрывающиеся (со сторонами, параллельными осям координат). Вам необходимо найти точку, которая покрыта наибольшим числом из них.

Формат входных данных

В первой строке входного файла записано число окон n ($1 \leqslant n \leqslant 50\,000$). Следующие n строк содержат координаты окон $x_{(1,i)}$ $y_{(1,i)}$ $x_{(2,i)}$ $y_{(2,i)}$, где $(x_{(1,i)}, y_{(1,i)})$ — координаты левого верхнего угла i-го окна, а $(x_{(2,i)}, y_{(2,i)})$ — правого нижнего (на экране компьютера y растет сверху вниз, а x — слева направо). Все координаты — целые числа, по модулю не превосходящие 10^6.

Формат выходных данных

В первой строке выходного файла выведите максимальное число окон, покрывающих какую-либо из точек в данной конфигурации. Во второй строке выведите два целых числа, разделенных пробелом — координаты точки, покрытой максимальным числом окон. Окна считаются замкнутыми, т. е. покрывающими свои граничные точки.

Примеры

windows.in	windows.out
2 0 0 3 3 1 1 4 4	2 1 3
1 0 0 1 1	1 0 1
4 0 0 1 1 0 1 1 2 1 0 2 1 1 1 2 2	4 1 1
5 0 0 1 1 0 1 1 2 0 0 2 2 1 0 2 1 1 1 2 2	5 1 1

Задача K. LCA offline

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

Изначально имеется дерево состоящее только из корня (вершина с номером 1). Требуется научиться отвечать на следующие запросы:

- ADD a b — подвесить вершину b за вершину a (гарантируется, что вершина a уже существует).
- GET a b — вернуть LCA вершин a и b.

Все номера вершин от 1 до N.
В каждый момент времени у нас есть одно дерево.

Формат входных данных

В первой строке входного файла содержится число k — количество запросов. Следующие k строк содержат сами запросы. Гарантируется, что число запросов каждого из типов не превосходит 500 000.

Формат выходных данных

Для каждого запроса типа GET выведите в отдельную строку одно целое число — ответ на соответствующий запрос.

Примеры

lca.in	lca.out
9 ADD 1 2 ADD 1 3 ADD 2 4 GET 1 3 GET 2 3 GET 3 4 ADD 2 5 GET 4 5 GET 5 5	1 1 1 2 5

Задача L. Двумерные запросы

Имя входного файла:	`find2d.in`
Имя выходного файла:	`find2d.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам задан массив размера 2^{17}. Требуется ответить на запросы: сколько есть элементов $f[i]$ таких, что $l \leqslant i \leqslant r$ и $x \leqslant f[i] \leqslant y$.

Формат входных данных

На первой строке число q ($1 \leqslant q \leqslant 2^{17}$). На второй строке пара целых чисел a, b от 1 до 10^9, используемая в генераторе случайных чисел.

```
0. unsigned int a, b; // даны во входных данных
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand17() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur » 15; // число от 0 до 2^17 - 1.
5. }
6. unsigned int nextRand24() {
7.     cur = cur * a + b; // вычисляется с переполнениями
8.     return cur » 8; // число от 0 до 2^24 - 1.
9. }
```

Сначала массив генерируется следующим образом:

```
1. for (int i = 0; i < 1 « 17; i++)
2.     f[i] = nextRand24();
```

Потом генерируются запросы следующим образом:

```
1. l = nextRand17();
2. r = nextRand17();
3. if (l > r) swap(l, r); // получили отрезок [l..r]
4. x = nextRand24();
5. y = nextRand24();
6. if (x > y) swap(x, y); // получили отрезок [x..y]
7. b += c; // c -- ответ на данный запрос, для ответа на запросы в online
```

Формат выходных данных

Выведите сумму ответов на все запросы второго типа по модулю 2^{32}.

Примеры

find2d.in	find2d.out
5 13 239	111139

Задача M. LCA Problem Revisited

Имя входного файла:	`lca_rmq.in`
Имя выходного файла:	`lca_rmq.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	64 мегабайта

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

lca_rmq.in	lca_rmq.out
3 2 0 1 2 1 1 1 0	2

Задача N. Железная дорога

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 megabytes

Вера очень любит поезда и железные дороги. Ее домашняя коллекция железных дорог настолько большая, что она построила сеть железных дорог, которая соединяет n железнодорожных станций. Каждая дорога соединяет две станции. Сеть она построила так, чтобы от каждой станции до любой другой можно было доехать на поезде по дорогам. Вера не любит делать ничего лишнего и любит, когда все сделано оптимально, поэтому проехать между любыми двумя станциями можно ровно одним способом.

Вера запрограммировала m поездов: i-й из поездов следует от станции s_i до станции f_i. Поезда всегда ездят по кратчайшему пути.

Теперь Вера задумалась, есть ли такие дороги, по которым не проезжает ни один поезд. Помогите ей понять это.

Формат входных данных

Первая строка содержит одно целое число n — число станций в железнодорожной сети ($2 \leqslant n \leqslant 100\,000$).

Следующие $n-1$ строк описывают дороги: i-я из этих строк содержит два целых числа v_i и u_i — номера станций, соединенных i-й дорогой ($1 \leqslant v_i, u_i \leqslant n$).

Гарантируется, что в сети между каждой парой станций есть единственный путь.

В следующей строке содержится целое число m — число запрограммированных Верой поездов ($0 \leqslant m \leqslant 100\,000$). В i-й из m следующих строк содержится два целых числа s_i и f_i — номера станций, между которыми курсирует i-й поезд ($1 \leqslant s_i, f_i \leqslant n$).

Формат выходных данных

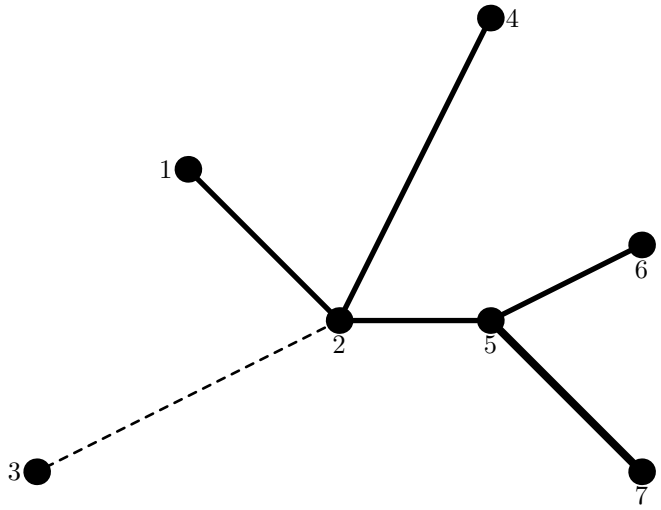
Выведите число дорог, по которым не проезжает ни один запрограммированный Верой поезд.

Примеры

standard input	standard output
4 1 2 1 3 1 4 0	3
7 1 2 2 3 2 4 5 2 5 6 7 5 3 1 7 2 4 7 6	1

Замечание

Иллюстрация ко второму примеру.



Пунктирной линией обозначена дорога, по которой не проходит ни один поезд.

Задача O. Самое дешевое ребро

Имя входного файла:	`minonpath.in`
Имя выходного файла:	`minonpath.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Дано подвешенное дерево с корнем в первой вершине. Все ребра имеют веса (стоимости). Вам нужно ответить на M запросов вида “найти у двух вершин минимум среди стоимостей ребер пути между ними”.

Формат входных данных

В первой строке задано целое число n — число вершин в дереве ($1 \leqslant n \leqslant 2 \cdot 10^5$).

В следующих $n-1$ строках записаны два целых числа x и y. Число x на строке i означает, что x — предок вершины i, y задает стоимость ребра ($x < i$; $|y| \leqslant 10^6$).

Далее заданы m ($0 \leqslant m \leqslant 5 \cdot 10^5$) запросов вида (x, y) — найти минимум на пути из x в y ($1 \leqslant x, y \leqslant n$; $x \neq y$).

Формат выходных данных

Выведите ответы на запросы.

Примеры

minonpath.in	minonpath.out
5 1 2 1 3 2 5 3 2 2 2 3 4 5	2 2
5 1 1 1 2 2 3 3 4 2 1 4 3 2	1 1

Задача P. Прибавление на пути

Имя входного файла:	`treepathadd.in`
Имя выходного файла:	`treepathadd.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Задано дерево. В каждой вершине есть значение, изначально все значения равны нулю. Требуется обработать запрос прибавления на пути и запрос значения в вершине.

Формат входных данных

В первой строке задано целое число n — число вершин в дереве ($1 \leqslant n \leqslant 3 \cdot 10^5$).

В следующих $n - 1$ строках заданы ребра дерева: по два целых числа v и u в строке — номера вершин, соединенных ребром ($1 \leqslant v, u \leqslant n$).

В следующей строке задано целое число m — число запросов ($1 \leqslant m \leqslant 5 \cdot 10^5$).

Следующие m строк содержат запросы в одном из двух форматов:

- `+ v u d` — прибавить число d во все значения в вершинах на пути от v до u ($1 \leqslant v, u \leqslant n$; $1 \leqslant d \leqslant 10^9$);
- `? v` — вывести значение в вершине v ($1 \leqslant v \leqslant n$).

Формат выходных данных

Выведите ответы на все запросы.

Примеры

treepathadd.in	treepathadd.out
5 1 2 1 3 3 4 3 5 5 + 2 5 1 ? 3 + 1 1 2 ? 1 ? 3	1 3 1

Задача Q. Пещеры и туннели

Имя входного файла:	`caves.in`
Имя выходного файла:	`caves.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

После посадки на Марс учёные нашли странную систему пещер, соединённых туннелями. И учёные начали исследовать эту систему, используя управляемых роботов. Было обнаружено, что существует ровно один путь между каждой парой пещер. Но потом учёные обнаружили специфическую проблему. Иногда в пещерах происходят небольшие взрывы. Они вызывают выброс радиоактивных изотопов и увеличивают уровень радиации в пещере. К сожалению, роботы плохо выдерживают радиацию. Но для исследования они должны переместиться из одной пещеры в другую. Учёные поместили в каждую пещеру сенсор для мониторинга уровня радиации. Теперь они каждый раз при движении робота хотят знать максимальный уровень радиации, с которым придётся столкнуться роботу во время его перемещения. Как вы уже догадались, программу, которая это делает, будете писать вы.

Формат входных данных

Первая строка входного файла содержит одно целое число N ($1 \leqslant N \leqslant 100\,000$) — количество пещер. Следующие $N - 1$ строк описывают туннели. Каждая из этих строк содержит два целых числа — a_i и b_i ($1 \leqslant a_i, b_i \leqslant N$), описывающие туннель из пещеры с номером a_i в пещеру с номером b_i. Следующая строка содержит целое число Q ($1 \leqslant Q \leqslant 100\,000$), означающее количество запросов. Далее идут Q запросов, по одному на строку. Каждый запрос имеет вид «C U V», где C — символ «`I`» либо «`G`», означающие тип запроса (кавычки только для ясности). В случае запроса «`I`» уровень радиации в U-й пещере ($1 \leqslant U \leqslant N$) увеличивается на V ($0 \leqslant V \leqslant 10\,000$). В случае запроса «`G`» ваша программа должна вывести максимальный уровень радиации на пути между пещерами с номерами U и V ($1 \leqslant U, V \leqslant N$) после всех увеличений радиации (запросов «`I`»), указанных ранее. Предполагается, что изначальный уровень радиации равен 0 во всех пещерах, и он никогда не уменьшается со временем (потому что период полураспада изотопов много больше времени наблюдения).

Формат выходных данных

Для каждого запроса «`G`» выведите одну строку, содержащую максимальный уровень радиации.

Примеры

caves.in	caves.out
4 1 2 2 3 2 4 6 I 1 1 G 1 1 G 3 4 I 2 3 G 1 1 G 3 4	1 0 1 3