

## Задача А. Разрезание графа

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

Дан неориентированный граф. Над ним в заданном порядке производят операции следующих двух типов:

- `cut` — разрезать граф, то есть удалить из него ребро;
- `ask` — проверить, лежат ли две вершины графа в одной компоненте связности.

Известно, что после выполнения всех операций типа `cut` рёбер в графе не осталось. Найдите результат выполнения каждой из операций типа `ask`.

### Формат входных данных

Первая строка ввода содержит три целых числа, разделённых пробелами — количество вершин графа  $n$ , количество рёбер  $m$  и количество операций  $k$  ( $1 \leq n \leq 50\,000$ ,  $0 \leq m \leq 100\,000$ ,  $m \leq k \leq 150\,000$ ).

Следующие  $m$  строк задают рёбра графа;  $i$ -ая из этих строк содержит два числа  $u_i$  и  $v_i$  ( $1 \leq u_i, v_i \leq n$ ), разделённые пробелами — номера концов  $i$ -го ребра. Вершины нумеруются с единицы; граф не содержит петель и кратных рёбер.

Далее следуют  $k$  строк, описывающих операции. Операция типа `cut` задаётся строкой «`cut u v`» ( $1 \leq u, v \leq n$ ), которая означает, что из графа удаляют ребро между вершинами  $u$  и  $v$ . Операция типа `ask` задаётся строкой «`ask u v`» ( $1 \leq u, v \leq n$ ), которая означает, что необходимо узнать, лежат ли в данный момент вершины  $u$  и  $v$  в одной компоненте связности. Гарантируется, что каждое ребро графа встретится в операциях типа `cut` ровно один раз.

### Формат выходных данных

Для каждой операции `ask` выведите в отдельной строке слово «YES», если две указанные вершины лежат в одной компоненте связности, и «NO» в противном случае. Порядок ответов должен соответствовать порядку операций `ask` во вводе.

## Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
3 3 7	YES
1 2	YES
2 3	NO
3 1	NO
ask 3 3	
cut 1 2	
ask 1 2	
cut 1 3	
ask 2 1	
cut 2 3	
ask 3 1	

## Задача В. Рёбра добавляются, граф растёт

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

В неориентированный граф последовательно добавляются новые рёбра. Изначально граф пустой. После каждого добавления нужно говорить, является ли текущий граф двудольным.

### Формат входных данных

На первой строке  $n$  — количество вершин,  $m$  — количество операций «добавить ребро» ( $1 \leq n, m \leq 300\,000$ ). Следующие  $m$  строк содержат пары чисел от 1 до  $n$  — описания добавляемых рёбер.

### Формат выходных данных

Выведите в строчку  $m$  нулей и единиц:  $i$ -й символ должен быть равен единице, если граф, состоящий из первых  $i$  рёбер, является двудольным.

### Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
3 3 1 2 2 3 3 1	110

Задача С. Система непересекающихся множеств

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

Система непересекающихся множеств — структура данных, хранящая для некоторого множества его разбиение на подмножества в каждый момент времени. Она поддерживает две операции: слияние двух подмножеств и проверку принадлежности двух элементов одному подмножеству.

В этой задаче структура используется так. Дан граф из  $n$  вершин, изначально не имеющий рёбер. В граф последовательно предлагаются рёбра для добавления. Каждый раз, когда предлагается ребро между какими-то вершинами  $u$  и  $v$  веса  $c$ , следует проверить, лежат ли на этот момент  $u$  и  $v$  в одной компоненте связности. Если да, ребро игнорируется. Если нет, ребро добавляется в граф.

Найдите суммарный вес рёбер, добавленных в граф после всех указанных операций.

Формат входных данных

В первой строке заданы через пробел целые числа  $n$  и  $k$  — количество вершин в графе и количество строк, описывающих рёбра ( $1 \leq n \leq 10^7$ ,  $0 \leq k \leq 10^5$ ).

В следующих  $k$  строках заданы рёбра. Каждая из них имеет вид  $u\ v\ c\ \Delta u\ \Delta v\ \Delta c\ m$  ( $0 \leq u, v < n$ ,  $0 \leq c < 10^9$ ,  $|\Delta u|, |\Delta v|, |\Delta c| < 10^9$ ,  $1 \leq m \leq 10^7$ , все числа в строке целые). Такая строка означает, что последовательно поступают предложения о добавлении в граф  $m$  рёбер. Первое из них — ребро между вершинами  $u$  и  $v$ , имеющее вес  $c$ . Второе — ребро между  $(u + \Delta u) \bmod n$  и  $(v + \Delta v) \bmod n$ , имеющее вес  $(c + \Delta c) \bmod 10^9$ , и так далее. Последнее из этих  $m$  рёбер соединяет вершины  $(u + (m - 1) \cdot \Delta u) \bmod n$  и  $(v + (m - 1) \cdot \Delta v) \bmod n$ , а его вес равен  $(c + (m - 1) \cdot \Delta c) \bmod 10^9$ . Напомним, что  $x \bmod y$  — это наименьшее неотрицательное число  $z$  такое, что величина  $z - x$  делится нацело на  $y$ .

Общее количество предлагаемых рёбер, равное сумме чисел  $m$  во всех строках, не превосходит  $10^7$ . Среди предлагаемых рёбер могут быть кратные рёбра и петли. Обратите внимание на то, что вершины в графе нумеруются с нуля.

Формат выходных данных

В первой строке выведите одно число — суммарный вес рёбер, добавлен-

ных в граф после всех указанных операций.

Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
8 3 0 1 1 2 2 0 4 0 1 2 1 1 0 5 0 3 6 9 12 15 2	29
4 1 1 2 1 1 1 1 2	3

Задача D. Всем чмоки в этом чатике!

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 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 ( $1 \leq n \leq 10^5$ ). 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 входных данных записаны натуральные числа  $n$  ( $1 \leq n \leq 10^5$ ) — число пользователей социальной сети. и  $m$  ( $1 \leq m \leq 3 \cdot 10^5$ ) — число событий, произошедших за день. В следующих  $m$  строках содержится описание событий. Первое целое число в строке означает тип события  $t$  ( $1 \leq t \leq 3$ ). Если  $t = 1$ , далее следует число  $i$  ( $0 \leq i < n$ ), по которому можно вычислить, какой участник послал сообщение. Если  $t = 2$ , далее следуют числа  $i$  и  $j$  ( $0 \leq i, j < n$ ), по которым можно вычислить номера участников,

чаты с которыми должны объединиться. Если  $t = 3$ , далее следует число  $i$  ( $0 \leq i < n$ ), 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.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>	Пояснение
4 10	1	4 10
1 0	1	1 0
1 2	2	1 1
1 1		1 2
1 2		1 3
3 1		3 0
2 1 2		2 0 1
1 3		1 1
3 3		3 0
2 3 2		2 2 1
3 2		3 1

Замечание

Справа указаны номера участников в запросах после декодирования.

Задача Е. СНМ и персистентность

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мебибайт

Ваша задача — реализовать **Persistent Disjoint Set Union** (персистентную систему непересекающихся множеств). Что это значит?

Про **Disjoint Set Union**:

Изначально у вас есть  $n$  элементов, пронумерованных целыми числами от 1 до  $n$  и лежащих каждый в своём множестве. Нужно научиться обрабатывать два типа запросов.

- $+ \ a \ b$  — объединить множества, в которых лежат элементы  $a$  и  $b$
- $? \ a \ b$  — сказать, лежат ли элементы  $a$  и  $b$  сейчас в одном множестве

Про **Persistent**:

Теперь у нас будет несколько копий (версий) структуры данных **Disjoint Set Union**.

Запросы будут выглядеть так:

- $+ \ i \ a \ b$  — запрос к  $i$ -й структуре: объединить множества, в которых лежат элементы  $a$  и  $b$ . При этом  $i$ -я структура остаётся неизменной, создаётся новая версия, ей присваивается новый номер (какой? читайте дальше)
- $? \ i \ a \ b$  — запрос к  $i$ -й структуре: сказать, лежат ли в ней элементы  $a$  и  $b$  в одном множестве

Формат входных данных

В первой строке заданы два целых числа: число элементов в структуре  $N$  и число запросов  $K$  ( $1 \leq N \leq 10^5$ ,  $0 \leq K \leq 10^5$ ). Изначальная копия (версия) структуры имеет номер 0.

Далее следуют  $K$  строк, в каждой из которых записан очередной запрос. Формат запросов описан выше. Запросы нумеруются целыми числами от 1 до  $K$ . При обработке  $j$ -го запроса, если он имеет вид « $+ \ i \ a \ b$ », новая версия получит номер  $j$ .

Формат выходных данных

Для каждого запроса вида « $? \ i \ a \ b$ » в отдельной строке нужно вывести «YES» или «NO».

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
4 7	NO
+ 0 1 2	YES
? 0 1 2	YES
? 1 1 2	YES
+ 1 2 3	NO
? 4 3 1	
? 0 4 4	
? 4 1 4	

## Задача F. Гармонический ряд

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

Дано целое число  $n$ . Выведите число, которое получится в  $r$  после выполнения следующей программы на языке, аналогичном C++:

```
r = 0;
for (i = 1; i <= n; i++)
    for (j = i; j <= n; j += i)
        r += 1;
```

Все переменные — 64-битные целые числа со знаком.

### Формат входных данных

В первой строке записано целое число  $n$  ( $1 \leq n \leq 10^{14}$ ).

### Формат выходных данных

В первой строке выведите одно целое число  $r$  — значение результата после выполнения программы.

### Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
3	5
10	27

Задача G. Список степеней

Можно доказать, что в этой последовательности встречаются лишь числа 1 и 4. Поэтому список в ответе будет пустым.

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 3.5 секунды  
Ограничение по памяти: 512 мегабайт

Пусть задано простое число  $p$  и число  $a$  такое, что  $0 < a < p$ . Рассмотрим все числа от  $l$  до  $r$  включительно, представимые в виде  $a^k \bmod p$  для какого-то целого неотрицательного числа  $k$ . Пусть известно, что таких чисел не более 100. Выведите все эти числа в порядке возрастания.

Формат входных данных

В единственной строке заданы через пробел четыре целых числа  $p, a, l$  и  $r$  ( $0 < a < p \leq 10^9$ ,  $p$  простое,  $0 \leq l \leq r < p$ ).

Формат выходных данных

Выведите все числа от  $l$  до  $r$  включительно, представимые в виде  $a^k \bmod p$  для какого-то целого неотрицательного числа  $k$ , в порядке возрастания, разделяя соседние числа пробелом. Гарантируется, что входные данные таковы, что в правильном ответе не более 100 чисел.

Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
5 3 0 3	1 2 3
5 4 2 3	

Пояснения к примерам

В первом примере требуется найти все числа от  $l = 0$  до  $r = 3$  включительно, которые представимы в виде  $3^k \bmod 5$  для некоторого целого  $k \geq 0$ . Это числа  $3^0 \bmod 5 = 1$ ,  $3^1 \bmod 5 = 3$  и  $3^3 \bmod 5 = 27 \bmod 5 = 2$ . Число 0 не представимо в таком виде, поскольку  $3^k$  не делится на 5 ни для какого целого  $k \geq 0$ . Значит, следует вывести числа 1, 2 и 3 в порядке возрастания.

Во втором примере требуется найти все числа от  $l = 2$  до  $r = 3$  включительно, которые представимы в виде  $4^k \bmod 5$  для некоторого целого  $k \geq 0$ . Выпишем первые несколько чисел такого вида:

$4^0 \bmod 5 = 1$ ,  
 $4^1 \bmod 5 = 4$ ,  
 $4^2 \bmod 5 = 16 \bmod 5 = 1$ ,  
 $4^3 \bmod 5 = 64 \bmod 5 = 4$ ,  
 $4^4 \bmod 5 = 256 \bmod 5 = 1$ , ...

## Задача Н. Поиск суммы на отрезке

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

Задана последовательность целых чисел  $A$  длины  $N$ . Для каждой заданной пары чисел  $(k, l)$  требуется найти значение суммы элементов в последовательности  $A$ , начиная с индекса  $k$  и заканчивая индексом  $l$ .

### Формат входных данных

В первой строке входных данных содержатся два числа  $N$  и  $M$  — количество элементов последовательности  $1 \leq N \leq 100\,000$  и количество запросов  $1 \leq M \leq 100\,000$ . В следующей строке через пробел перечислены элементы последовательности  $A$ . Все числа не превышают границ 32-битного числа со знаком. Последующие  $M$  строк содержат по два числа  $(k, l)$  — начало и конец отрезков, на которых требуется найти сумму элементов.

### Формат выходных данных

Выведите суммы элементов последовательности  $A$  на заданных отрезках для всех запросов.

### Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
6 3	28
1 8 4 5 3 7	8
1 6	12
2 2	
3 5	



## Задача I. Поиск минимума на отрезке

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

Задана последовательность целых чисел  $A$  длины  $N$ . Для каждой заданной пары чисел  $(k, l)$  требуется найти значение индекса минимального элемента в последовательности  $A$ , начиная с индекса  $k$  и заканчивая индексом  $l$ .

### Формат входных данных

В первой строке входных данных содержатся два числа  $N$  и  $M$  — количество элементов последовательности  $1 \leq N \leq 100\,000$  и количество запросов  $1 \leq M \leq 100\,000$ . В следующей строке через пробел перечислены элементы последовательности  $A$ . Все числа не превышают границ 32-битного числа со знаком. Последующие  $M$  строк содержат по два числа  $(k, l)$  — начало и конец отрезков, на которых требуется найти индекс минимального элемента.

### Формат выходных данных

Выведите индексы минимальных элементов последовательности  $A$  на заданных отрезках для всех запросов.

### Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
6 3	1
1 8 4 5 3 7	2
1 6	5
2 2	
3 5	

Задача J. Range Variation Query

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мебибайт

В начальный момент времени последовательность  $a_n$  задана следующей формулой:  $a_n = n^2 \bmod 12\,345 + n^3 \bmod 23\,456$ .

Требуется много раз обрабатывать запросы следующего вида:

- найти разность между максимальным и минимальным значением среди элементов  $a_i, a_{i+1}, \dots, a_j$ ;
- присвоить элементу  $a_i$  значение  $j$ .

Формат входных данных

Первая строка содержит целое число  $k$  — количество запросов ( $1 \leq k \leq 100\,000$ ). Следующие  $k$  строк содержат запросы, по одному на строке. Запрос номер  $i$  описывается двумя целыми числами  $x_i, y_i$ .

Если  $x_i > 0$ , то требуется найти разность между максимальным и минимальным значением среди элементов  $a_{x_i} \dots a_{y_i}$ . При этом  $1 \leq x_i \leq y_i \leq 100\,000$ .

Если  $x_i < 0$ , то требуется присвоить элементу  $a_{|x_i|}$  значение  $y_i$ . При этом  $-100\,000 \leq x_i \leq -1$  и  $|y_i| \leq 100\,000$ .

Формат выходных данных

Для каждого запроса первого типа выведите одну строку, содержащую разность между максимальным и минимальным значением на соответствующем отрезке.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
7	34
1 3	68
2 4	250
-2 -100	234
1 5	1
8 9	
-3 -101	
2 3	

Задача К. Числа на отрезке

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 512 мегабайт

Вова нарисовал на доске горизонтальную прямую, отметил на ней  $N$  точек и пронумеровал их слева направо натуральными числами от 1 до  $N$ . После этого он стал обводить некоторые точки кружочками. Время от времени Паша, чтобы оторвать его от этого занятия, спрашивает его, сколько точек на отрезке от  $A$  до  $B$ , включительно, Вова уже обвёл кружочками. Ответьте Паше на все его вопросы, чтобы не отвлекать Вову.

Формат входных данных

В первой строке входных данных записаны два целых числа  $N$  и  $K$  — количество точек на отрезке и количество событий, соответственно ( $1 \leq N \leq 1\,000\,000$ ,  $1 \leq K \leq 100\,000$ ).

В следующих  $K$  строках заданы события в порядке, в котором они случались. Каждая из этих строк либо содержит целое число  $C$  от 1 до  $N$ , включительно, которое означает, что Вова обвёл кружочком точку с номером  $C$ , либо имеет вид «0  $A$   $B$ », где  $1 \leq A \leq B \leq N$ , что означает, что Паша спросил, сколько точек на отрезке от  $A$  до  $B$ , включительно, уже обведено кружочками.

Вова обводит каждую точку не более одного раза.

Формат выходных данных

Выведите ответ на каждый вопрос Паши на отдельной строке в том порядке, в котором эти вопросы даны во входных данных.

Примеры

<i>стандартный ввод</i>	<i>стандартный вывод</i>
3 4 1 0 1 1 2 0 1 3	1 2
10 6 0 1 10 6 1 4 0 2 9 8	0 2

Задача L. Добавление и GCD

Имя входного файла: *стандартный ввод*  
Имя выходного файла: *стандартный вывод*  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 512 мегабайт

Дан массив из  $n$  целых чисел:  $a_1, a_2, \dots, a_n$ . Поступают  $q$  запросов двух типов:

- «1  $\ell$   $r$   $x$ » — ко всем элементам массива с индексами от  $\ell$  до  $r$  включительно прибавить число  $x$ ;
- «2  $\ell$   $r$ » — вывести  $\gcd(a_\ell, a_{\ell+1}, \dots, a_r)$ .

Здесь  $\gcd(S)$  обозначает наибольший общий делитель множества чисел  $S$ .

Формат входных данных

В первой строке заданы два целых числа  $n$  и  $q$  ( $1 \leq n, q \leq 2 \cdot 10^5$ ) — длина массива и количество запросов.

Во второй строке заданы  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^9$ ) — элементы массива  $a$ .

Каждая из следующих  $q$  строк описывает запрос в формате, указанном в условии ( $1 \leq \ell \leq r \leq n, 1 \leq x \leq 10^9$ ).

Гарантируется, что хотя бы один запрос имеет тип 2.

Формат выходных данных

На каждый запрос второго типа выведите  $\gcd$  всех чисел из соответствующего отрезка.

Пример

<i>стандартный ввод</i>	<i>стандартный вывод</i>
6 8	3
10 6 15 12 18 30	6
2 2 6	3
1 1 1 8	3
2 1 2	3
2 1 4	3
1 3 4 3	
2 1 6	
2 2 5	
2 3 4	