

## Задача А. Снеговика

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	4 секунды
Ограничение по памяти:	64 мегабайта

Зима. 2012 год. На фоне грядущего Апокалипсиса и конца света незамеченной прошла новость об очередном прорыве в областях клонирования и снеговиков: клонирования снеговиков. Вы конечно знаете, но мы вам напомним, что снеговик состоит из нуля или более вертикально поставленных друг на друга шаров, а клонирование — это процесс создания идентичной копии (клона).

В местечке Местячково учитель Андрей Сергеевич Учитель купил через интернет-магазин «Интернет-магазин аппаратов клонирования» аппарат для клонирования снеговиков. Теперь дети могут играть и даже играют во дворе в следующую игру. Время от времени один из них выбирает понравившегося снеговика, клонирует его и:

- либо добавляет ему сверху один шар;
- либо удаляет из него верхний шар (если снеговик не пустой).

Учитель Андрей Сергеевич Учитель записал последовательность действий и теперь хочет узнать суммарную массу всех построенных снеговиков.

### Формат входных данных

Первая строка содержит количество действий  $n$  ( $1 \leq n \leq 200\,000$ ). В строке номер  $i + 1$  содержится описание действия  $i$ :

- $t\ m$  — клонировать снеговика номер  $t$  ( $0 \leq t < i$ ) и добавить сверху шар массой  $m$  ( $0 < m \leq 1000$ );
- $t\ 0$  — клонировать снеговика номер  $t$  ( $0 \leq t < i$ ) и удалить верхний шар. Гарантируется, что снеговик  $t$  не пустой.

В результате действия  $i$ , описанного в строке  $i + 1$  создается снеговик номер  $i$ . Изначально имеется пустой снеговик с номером ноль.

Все числа во входном файле целые.

### Формат выходных данных

Выведите суммарную массу построенных снеговиков.

### Пример

стандартный ввод	стандартный вывод
8	74
0 1	
1 5	
2 4	
3 2	
4 3	
5 0	
6 6	
1 0	

## Задача В. $K$ -я порядковая статистика на отрезке

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

Дан массив из  $N$  неотрицательных чисел, строго меньших  $10^9$ . Вам необходимо ответить на несколько запросов о величине  $k$ -й порядковой статистики на отрезке  $[l, r]$ .

### Формат входных данных

Первая строка содержит число  $N$  ( $1 \leq N \leq 450\,000$ ) — размер массива.

Вторая строка может быть использована для генерации  $a_i$  — начальных значений элементов массива. Она содержит три числа  $a_1, l$  и  $m$  ( $0 \leq a_1, l, m < 10^9$ ); для  $i$  от 2 до  $N$

$$a_i = (a_{i-1} \cdot l + m) \bmod 10^9.$$

В частности,  $0 \leq a_i < 10^9$ .

Третья строка содержит одно целое число  $B$  ( $1 \leq B \leq 1000$ ) — количество групп запросов.

Следующие  $B$  строк описывают одну группу запросов. Каждая группа запросов описывается 10 числами. Первое число  $G$  обозначает количество запросов в группе. Далее следуют числа  $x_1, l_x$  и  $m_x$ , затем  $y_1, l_y$  и  $m_y$ , затем,  $k_1, l_k$  и  $m_k$  ( $1 \leq x_1 \leq y_1 \leq N, 1 \leq k_1 \leq y_1 - x_1 + 1, 0 \leq l_x, m_x, l_y, m_y, l_k, m_k < 10^9$ ). Эти числа используются для генерации вспомогательных последовательностей  $x_g$  и  $y_g$ , а также параметров запросов  $i_g, j_g$  и  $k_g$  ( $1 \leq g \leq G$ )

$$\begin{aligned} x_g &= ((i_{g-1} - 1) \cdot l_x + m_x) \bmod N + 1, & 2 \leq g \leq G \\ y_g &= ((j_{g-1} - 1) \cdot l_y + m_y) \bmod N + 1, & 2 \leq g \leq G \\ i_g &= \min(x_g, y_g), & 1 \leq g \leq G \\ j_g &= \max(x_g, y_g), & 1 \leq g \leq G \\ k_g &= (((k_{g-1} - 1) \cdot l_k + m_k) \bmod (j_g - i_g + 1)) + 1, & 2 \leq g \leq G \end{aligned}$$

Сгенерированные последовательности описывают запросы,  $g$ -й запрос состоит в поиске  $k_g$ -го по величине числа среди элементов отрезка  $[i_g, j_g]$ .

Суммарное количество запросов не превосходит 600 000.

### Формат выходных данных

Выведите единственное число — сумму ответов на запросы.

### Пример

стандартный ввод	стандартный вывод
5	15
1 1 1	
5	
1	
1 0 0 3 0 0 2 0 0	
1	
2 0 0 5 0 0 3 0 0	
1	
1 0 0 5 0 0 5 0 0	
1	
3 0 0 3 0 0 1 0 0	
1	
1 0 0 4 0 0 1 0 0	

## Задача С. Соединение и разъединение

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

Вы когда-нибудь слышали про обход в глубину? Например, используя этот алгоритм, вы можете проверить является ли граф связным за время  $O(E)$ . Вы можете даже посчитать количество компонент связности за то же время.

А вы когда-нибудь слышали про систему непересекающихся множеств? Используя эту структуру, вы можете быстро обрабатывать запросы “Добавить ребро в граф” и “Посчитать количество компонент связности в графе”.

А вы когда-нибудь слышали о *динамической* задаче связности? В этой задаче вам необходимо обрабатывать три типа запросов:

1. Добавить ребро в граф.
2. Удалить ребро из графа.
3. Посчитать количество компонент связности в графе.

Можно считать, что граф является неориентированным. Изначально граф является пустым.

### Формат входных данных

В первой строке находятся два целых числа  $N$  и  $K$  — количество вершин и количество запросов, соответственно ( $1 \leq N \leq 300\,000$ ,  $0 \leq K \leq 300\,000$ ). Следующие  $K$  строк содержат запросы, по одному в строке. Каждый запрос имеет один из трех типов:

1.  $+ u v$ : Добавить ребро между вершинами  $u$  и  $v$ . Гарантируется, что такого ребра нет.
2.  $- u v$ : Удалить ребро между  $u$  и  $v$ . Гарантируется, что такое ребро есть.
3.  $?$ : Посчитать количество компонент связности в графе.

Вершины пронумерованы целыми числами от 1 до  $N$ . Во всех запросах  $u \neq v$ .

### Формат выходных данных

Для каждого запроса типа ‘?’, Выведите количество компонент связности в момент запроса.

### Пример

стандартный ввод	стандартный вывод
5 11	5
?	1
+ 1 2	1
+ 2 3	2
+ 3 4	
+ 4 5	
+ 5 1	
?	
- 2 3	
?	
- 4 5	
?	

## Задача D. River Land

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

Есть страна с большой речной системой. Города в этой стране пронумерованы от 1 до  $N$ ; город № 1 является столицей. Речная система образует корневое дерево, так что столица является корнем. Каждая река соединяет два города и течет к корню. Для каждого некоренного города  $i$  есть река, которая течет из этого города в город  $p_i$ . В каждом городе также есть порт.

В стране есть только два способа путешествовать между городами: на лодке и на пароме. Лодка может двигаться только вниз по течению (то есть к столице), в то время как паром может двигаться в обоих направлениях. Есть прямые паромные маршруты из каждого порта в любой другой порт.

Изначально все порты открыты. Иногда некоторые порты могут закрываться или открываться снова (возможно, несколько раз). Порт в столице всегда открыт. Не разрешается путешествовать на пароме из закрытого порта, но допускается путешествие на пароме из открытого порта в закрытый порт. Путешествие на лодке всегда возможно, даже из закрытого порта.

Иногда вы хотите отправиться в путешествие. Вы уже выбрали город, в котором начнете путешествие. Однако у вас достаточно денег только для одной поездки на пароме. Поэтому вы хотите совершить путешествие следующим образом:

- Выбрать город  $b$  и отправиться в этот город на лодке. Должна быть возможность путешествовать от  $v$  до  $b$  на лодке (в частности,  $v = b$  разрешено).
- Решить либо продолжить движение из города  $b$  на пароме, либо завершить поездку в этом городе.
- Если вы решаете продолжить путь на пароме, вы выбираете город  $f$  и отправляетесь на пароме от  $b$  до  $f$ . Также должна быть возможность путешествовать от  $b$  до  $f$  на пароме: не разрешается путешествовать вдоль любой реки дважды (независимо от направления), т.е., когда вы решаете продолжить путь на пароме, кратчайшие пути между  $(v, b)$  и  $(b, f)$  не должны иметь общих рек.
- Вы любите путешествовать на пароме, поэтому чем больше рек вы посещаете во время путешествия на пароме, тем счастливее вы становитесь. Также в каждом городе есть фиксированная красота; давайте обозначим красоту города  $i$  как  $a_i$ . Затем, для поездки со значением  $T$  (для каждой поездки задано своё значение) определим, что вы получаете счастье  $a_b + D \cdot T$ , где  $D$  — длина маршрута парома (количество рек, посещаемых во время путешествия на пароме; возможно, ноль, если продолжить путешествие на пароме из города  $b$  невозможно).

Вы должны обработать  $Q$  запросов. Есть три типа запросов:

1.  $- v$ : порт в городе  $v$  закрывается. Гарантируется, что порт в городе  $v$  был открыт до этого запроса.
2.  $+ v$ : порт в городе  $v$  вновь открывается. Гарантируется, что порт в городе  $v$  был закрыт до этого запроса.
3.  $? v T$ : вы хотите совершить поездку со значением  $T$  из города  $v$ .

Для каждого запроса третьего типа найдите максимальное счастье, которое вы можете получить от такой поездки.

### Формат входных данных

Первая строка содержит два целых числа  $N$  и  $Q$  ( $2 \leq N \leq 3 \cdot 10^5$ ,  $1 \leq Q \leq 3 \cdot 10^5$ ).

Вторая строка содержит  $N - 1$  целое число  $p_2, p_3, \dots, p_N$  ( $1 \leq p_i < i$ ).

Третья строка содержит  $N$  целых чисел  $a_1, a_2, \dots, a_N$  ( $1 \leq a_i \leq 10^9$ ).

Следующие  $Q$  строк описывают запросы в формате, описанном в условии.  $1 \leq v \leq N$ ,  $1 \leq T \leq 10^9$ .

### Формат выходных данных

Для каждого запроса третьего типа выведите ответ на него в отдельной строке.

### Система оценки

Подзадача	Доп. ограничения	Баллы
1	$N, Q \leq 1000$	30
2	$N, Q \leq 10^5$	35
3	—	35

### Пример

стандартный ввод	стандартный вывод
10 9	57
1 2 3 2 2 6 3 8 8	42
30 20 6 13 8 40 7 9 13 1	33
? 4 11	30
- 4	
? 4 11	
- 7	
? 10 6	
+ 7	
- 6	
- 2	
? 7 4	

### Замечание

В первом запросе вы путешествуете на пароме из города 4 в город 7.

Во втором запросе вы не можете путешествовать на пароме из города 4, поэтому вы должны использовать второй по выгодности маршрут, который идет от города 2 к городу 7.

В третьем запросе вы едете в город 8 на лодке, а затем в город 7 на пароме.

В четвертом запросе вы едете в столицу (город № 1) на лодке и не путешествуете на пароме.

## Задача E. Ciel и гондолы

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

Лиса Ciel зашла в парк аттракционов. И вот, она в очереди на колесо обозрения. В очереди стоит  $n$  людей (хотя нет, скорее лис): мы будем считать, что первая лиса стоит в начале очереди, а  $n$ -я лиса стоит в хвосте очереди.

Всего имеется  $k$  гондол, мы распределяем лис по гондолам следующим образом:

- Когда подплывает первая гондола,  $q_1$  лис переходят из начала очереди в подплывшую гондолу.
- Затем, когда подплывает вторая гондола,  $q_2$  лис из начала оставшейся очереди переходит в эту гондолу.
- ...
- Оставшиеся  $q_k$  лис идут с последней ( $k$ -ю) гондолу.

Обратите внимание, что числа  $q_1, q_2, \dots, q_k$  должны быть положительными. Из условия следует, что  $\sum_{i=1}^k q_i = n$  и  $q_i > 0$ .

Вы знаете как лисам не хочется задерживаться в гондолах с незнакомцами. Итак, Ваша задача — найти оптимальный способ размещения (то есть определить оптимальную последовательность  $q$ ), чтобы угодить всем. Для каждой пары лис  $i$  и  $j$  задано значение  $u_{ij}$ , обозначающее степень незнакомости. Можете считать, что  $u_{ij} = u_{ji}$  для всех  $i, j$  ( $1 \leq i, j \leq n$ ) и что  $u_{ii} = 0$  для всех  $i$  ( $1 \leq i \leq n$ ). Тогда значение незнакомости в гондоле определяется как сумма значений незнакомости между всеми парами лис, которые находятся в этой гондоле. Общее значение незнакомости определяется как сумма значений незнакомости по всем гондолам.

Помогите лисе Ciel найти минимальное возможное значение общей незнакомости при некотором оптимальном распределении лис по гондолам.

### Формат входных данных

В первой строке даны два целых числа  $n$  и  $k$  ( $1 \leq n \leq 4000$  and  $1 \leq k \leq \min(n, 800)$ ) — количество лис в очереди и количество гондол. В следующих  $n$  строках записано по  $n$  целых чисел — матрица  $u$ , ( $0 \leq u_{ij} \leq 9$ ,  $u_{ij} = u_{ji}$  и  $u_{ii} = 0$ ).

Пожалуйста, используйте методы быстрого чтения (например, для Java используйте `BufferedReader` вместо `Scanner`).

### Формат выходных данных

Выведите целое число — минимальное возможное значение общей незнакомости.

## Примеры

стандартный ввод	стандартный вывод
5 2 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0	0
8 3 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0	7

## Замечание

В первом примере можно распределить лис вот так: 1, 2 идут в одну гондолу, 3, 4, 5 идут в другую гондолу.

Во втором примере оптимальное распределение таково: 1, 2, 3 | 4, 5, 6 | 7, 8.

## Задача F. Урны и шары

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

Пусть у вас есть  $n$  урн, в каждой из которых лежит по одному шару. Урна с номером  $i$  содержит шарик под номером  $i$ . У вас есть специальное устройство, которое позволяет перемещать шарики. Им чрезвычайно просто пользоваться: сначала вы выбираете некоторый отрезок последовательных урн. После этого вы выбираете некоторый другой отрезок последовательных урн такой же длины, как и исходный, и затем шарики из урн первого отрезка перемещаются в соответствующие урны второго отрезка.

Дана последовательность перемещений. Установите, в какой урне окажется каждый шарик.

### Формат входных данных

Первая строка входных данных содержит два числа  $n$  и  $m$  — число урн и число перемещений, соответственно ( $1 \leq n \leq 100\,000$ ,  $1 \leq m \leq 50\,000$ ). Каждая из следующих  $m$  строк содержит три числа  $count_i$ ,  $from_i$  и  $to_i$ , которые означают одновременное перемещение всех шариков из урны  $from_i$  в урну  $to_i$ , всех шариков из урны  $from_i + 1$  в урну  $to_i + 1$ , ..., всех шариков из урны  $from_i + count_i - 1$  в урну  $to_i + count_i - 1$  ( $1 \leq count_i, from_i, to_i \leq n$ ,  $\max(from_i, to_i) + count_i \leq n + 1$ ).

### Формат выходных данных

Выведите  $n$  чисел — итоговые позиции каждого шарика.

### Пример

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

## Задача G. Откат

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 1.5 секунд  
Ограничение по памяти: 256 мегабайт

Сергей работает системным администратором в очень крупной компании. Естественно, в круг его обязанностей входит резервное копирование информации, хранящейся на различных серверах и «откат» к предыдущей версии в случае возникновения проблем.

В данный момент Сергей борется с проблемой недостатка места для хранения информации для восстановления. Он решил перенести часть информации на новые сервера. К сожалению, если что-то случится во время переноса, он не сможет произвести откат, поэтому процедура переноса должна быть тщательно спланирована.

На данный момент у Сергея хранятся  $n$  точек восстановления различных серверов, пронумерованных от 1 до  $n$ . Точка восстановления с номером  $i$  позволяет произвести откат для сервера  $a_i$ . Сергей решил разбить перенос на этапы, при этом на каждом этапе в случае возникновения проблем будут доступны точки восстановления с номерами  $l, l + 1, \dots, r$  для некоторых  $l$  и  $r$ .

Для того, чтобы спланировать перенос данных оптимальным образом, Сергею необходимо научиться отвечать на запросы: для заданного  $l$ , при каком минимальном  $r$  в процессе переноса будут доступны точки восстановления не менее чем  $k$  различных серверов.

Помогите Сергею.

### Формат входных данных

Первая строка входного файла содержит два целых числа  $n$  и  $m$ , разделенные пробелами — количество точек восстановления и количество серверов ( $1 \leq n, m \leq 100\,000$ ). Вторая строка содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  — номера серверов, которым соответствуют точки восстановления ( $1 \leq a_i \leq m$ ).

Третья строка входного файла содержит  $q$  — количество запросов, которые необходимо обработать ( $1 \leq q \leq 100\,000$ ). В процессе обработки запросов необходимо поддерживать число  $p$ , исходно оно равно 0. Каждый запрос задается парой чисел  $x_i$  и  $y_i$ , используйте их для получения данных запроса следующим образом:  $l_i = ((x_i + p) \bmod n) + 1$ ,  $k_i = ((y_i + p) \bmod m) + 1$  ( $1 \leq l_i, x_i \leq n$ ,  $1 \leq k_i, y_i \leq m$ ). Пусть ответ на  $i$ -й запрос равен  $r$ . После выполнения этого запроса, следует присвоить  $p$  значение  $r$ .

### Формат выходных данных

На каждый запрос выведите одно число — искомое минимальное  $r$ , либо 0, если такого  $r$  не существует.

### Пример

стандартный ввод	стандартный вывод
7 3	1
1 2 1 3 1 2 1	4
4	0
7 3	6
7 1	
7 1	
2 2	

## Задача H. Intercity Express

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

Андрей разрабатывает систему для продажи железнодорожных билетов. Он собирается протестировать ее на Междугородней Экспресс линии, которая соединяет два больших города и имеет  $n - 2$  промежуточных станций, то есть в итоге есть  $n$  станций, пронумерованных от 1 до  $n$ .

В Междугороднем Экспресс поезде есть  $s$  мест, пронумерованных с 1 до  $s$ . В тестирующем режиме система имеет доступ к базе данных, содержащей проданные билеты в направлении от станции 1 до станции  $n$  и должна отвечать на вопросы, можно ли продать билет от станции  $a$  до станции  $b$ , и если да, нужно найти минимальный номер места, которое свободно на протяжении всего пути между  $a$  и  $b$ .

Изначально система имеет только доступ на чтение, то есть даже если есть свободное место, она должна сообщить об этом, но не должна изменять данные.

Помогите Андрею протестировать его систему написанием программы, которые будет находить ответы на вопросы.

### Формат входных данных

Первая строка содержит число  $n$  — количество станций,  $s$  — количество мест и  $m$  — количество уже проданных билетов ( $2 \leq n \leq 10^9$ ,  $1 \leq s \leq 100\,000$ ,  $0 \leq m \leq 100\,000$ ).

В следующих  $m$  строках описаны билеты, описание каждого билета состоит из трех чисел:  $c_i$ ,  $a_i$  и  $b_i$  — номер места, которое занимает владелец билета, номер станции, с которой продан билет и номер станции, до которой продан билет ( $1 \leq c_i \leq s$ ,  $1 \leq a_i < b_i \leq n$ ).

Следующие строки содержат число  $q$  — количество запросов ( $1 \leq q \leq 100\,000$ ). Специальное значение  $p$  должно поддерживаться в течение считывания запросов. Изначально  $p = 0$ .

Следующие  $2q$  строк описывают запросы. Каждый запрос описывается двумя числами:  $x_i$  и  $y_i$  ( $x_i \leq y_i$ ).

Чтобы получить города  $a$  и  $b$  между которыми нужно проверить наличие места, используется следующая формула:

$a = x_i + p$ ,  $b = y_i + p$ . Ответ на запрос — число 0, если нет места на каждом отрезке между  $a$  и  $b$ , или минимальный номер свободного места.

После ответа на запрос, надо приравнять число  $p$  полученному ответу на запрос.

### Формат выходных данных

Для каждого запроса выведите ответ на него.

## Пример

стандартный ввод	стандартный вывод
5 3 5	1
1 2 5	2
2 1 2	2
2 4 5	3
3 2 3	0
3 3 4	2
10	0
1 2	0
1 2	0
1 2	0
2 3	
-2 0	
2 4	
1 3	
1 4	
2 5	
1 5	

## Замечание

Обратите внимание, что запросы выглядят так: (1, 2), (2, 3), (3, 4), (4, 5), (1, 3), (2, 4), (3, 5), (1, 4), (2, 5), (1, 5).

## Задача I. Персистентная очередь

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 0.5 секунд  
Ограничение по памяти: 256 мегабайт

Реализуйте персистентную очередь.

### Формат входных данных

Первая строка содержит количество действий  $n$  ( $1 \leq n \leq 200\,000$ ). В строке номер  $i + 1$  содержится описание действия  $i$ :

- $1\ t\ m$  — добавить в конец очереди номер  $t$  ( $0 \leq t < i$ ) число  $m$ ;
- $-1\ t$  — удалить из очереди номер  $t$  ( $0 \leq t < i$ ) первый элемент.

В результате действия  $i$ , описанного в строке  $i + 1$  создается очередь номер  $i$ . Изначально имеется пустая очередь с номером ноль.

Все числа во входном файле целые, и помещаются в знаковый 32-битный тип.

### Формат выходных данных

Для каждой операции удаления выведите удаленный элемент на отдельной строке.

### Пример

стандартный ввод	стандартный вывод
10	1
1 0 1	2
1 1 2	3
1 2 3	1
1 2 4	2
-1 3	4
-1 5	
-1 6	
-1 4	
-1 8	
-1 9	

### Замечание

Алгоритм персистентной очереди за  $O(1)$  невероятно простой, там нужно всего 6 стеков, а если очень постараться, то можно обойтись вообще 5 стеками. Я уверен что вы все легко решите эту задачу.