Задача A. Снеговики

Имя входного файла:	`snowmen.in`
Имя выходного файла:	`snowmen.out`
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 записал последовательность действий и теперь хочет узнать суммарную массу всех построенных снеговиков.

Формат входных данных

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

- `t m` — клонировать снеговика номер t ($0 \leqslant t < i$) и добавить сверху шар массой m ($0 < m \leqslant 1000$);
- `t 0` — клонировать снеговика номер t ($0 \leqslant t < i$) 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.

Примеры

snowmen.in	snowmen.out
8 0 1 1 5 2 4 3 2 4 3 5 0 6 6 1 0	74

Задача B. НВП на дереве

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

Дано дерево на N вершинах, у которого i-е ребро соединяет вершину u_i и вершину v_i. На вершине с номером i записано целое число a_i. Для каждого целого числа k от 1 до N решите следующую задачу:

Составим последовательность, выписав целые числа на вершинах, вдоль кратчайшего пути от вершины 1 до вершины k, в том порядке, в котором они появляются. Найдите длину наибольшей возрастающей подпоследовательности этой последовательности.

Формат входных данных

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

Во второй строке через пробел задаются числа a_i ($1 \leqslant a_i \leqslant 10^9$) — числа, записанные на вершинах.

В каждой из следующих $n-1$-й строке вводятся пары v_i, u_i — рёбра дерева ($1 \leqslant v_i \neq u_i \leqslant n$). Гарантируется, чтоданный набор рёбер образует дерево.

Формат выходных данных

Выведите N строк. В k-й строке выведите длину наибольшей возрастающей подпоследовательности последовательности, полученной по кратчайшему пути из вершины 1 в вершину k.

Примеры

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

Задача C. K-я порядковая статистика на отрезке

Имя входного файла:	`kth.in`
Имя выходного файла:	`kth.out`
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 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 ($1 \leqslant N \leqslant 450\,000$) — размер массива.

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

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

В частности, $0 \leqslant a_i < 10^9$.

Третья строка содержит одно целое число B ($1 \leqslant B \leqslant 1000$) — 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 ($1 \leqslant x_1 \leqslant y_1 \leqslant N$, $1 \leqslant k_1 \leqslant y_1 - x_1 + 1$, $0 \leqslant l_x, m_x, l_y, m_y, l_k, m_k < 10^9$). Эти числа используются для генерации вспомогательных последовательностей x_g и y_g, а также параметров запросов i_g, j_g и k_g ($1 \leqslant g \leqslant G$)

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

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.

Примеры

kth.in	kth.out
5 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	15

Задача D. Менеджер памяти

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	8 секунды
Ограничение по памяти:	512 мегабайт

Одно из главных нововведений новейшей операционной системы Indows 7 — новый менеджер памяти. Он работает с массивом длины N и позволяет выполнять три самые современные операции:

- `copy(a, b, l)` — скопировать отрезок длины $[a, a + l - 1]$ в $[b, b + l - 1]$
- `sum(l, r)` — посчитать сумму элементов массива на отрезке $[l, r]$
- `print(l, r)` — напечатать элементы с l по r, включительно

Вы являетесь разработчиком своей операционной системы, и Вы, безусловно, не можете обойтись без инновационных технологий. Вам необходимо реализовать точно такой же менеджер памяти.

Формат входных данных

Первая строка входного файла содержит целое число N ($1 \leqslant N \leqslant 1\,000\,000$) — размер массива, с которым будет работать Ваш менеджер памяти.

Во второй строке содержатся четыре числа $1 \leqslant X_1, A, B, M \leqslant 10^9 + 10$. С помощью них можно сгенерировать исходный массив чисел $X_1, X_2, \cdots, X_N$. $X_{i+1} = (A * X_i + B) \mod M$

Следующая строка входного файла содержит целое число K ($1 \leqslant K \leqslant 200\,000$) — количество запросов, которые необходимо выполнить Вашему менеджеру памяти.

Далее в K строках содержится описание запросов. Запросы заданы в формате:

- `cpy` a b l — для операции `copy`
- `sum` l r — для операции `sum` ($l \leqslant r$)
- `out` l r — для операции `print` ($l \leqslant r$)

Гарантируется, что суммарная длина запросов `print` не превышает 3 000. Также гарантируется, что все запросы корректны.

Формат выходных данных

Для каждого запроса *sum* или *print* выведите в выходной файл на отдельной строке результат запроса.

Примеры

stdin	stdout
6 1 4 5 7 7 out 1 6 cpy 1 3 2 out 1 6 sum 1 4 cpy 1 2 4 out 1 6 sum 1 6	1 2 6 1 2 6 1 2 1 2 2 6 6 1 1 2 1 2 6 13

Задача E. Испытание силомера

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Сайтама выполняет последовательные удары по силомеру. Силомер представляет из себя массив целых чисел длины n. Изначально i-е число массива равно a_i для всех i.

Вам необходимо обработать q событий, происходящих с силомером. Событие номер i может быть одного из трех типов:

1. подходит наблюдатель и просит посчитать сумму чисел массива на отрезке $[l_i; r_i]$, то есть величину $a_{l_i} + a_{l_i+1} + \ldots + a_{r_i}$;
2. Сайтама наносит обычный удар силы x_i по отрезку $[l_i; r_i]$: всем элементам массива на позициях от l_i до r_i включительно присваивается значение x_i
3. Сайтама наносит сильный удар по отрезку $[l_i; r_i]$: для всех j от l_i до r_i включительно происходит присваивание $a_j \leftarrow \texttt{popcount}(a_j)$.

Здесь $\texttt{popcount}(x)$ — это количество единичных бит в двоичной записи числа x. Иными словами, при событии третьего типа каждое число на отрезке события заменяется на количество своих единичных бит.

На каждый подход наблюдателя, то есть событие первого типа, сообщите ему интересующую его сумму.

Формат входных данных

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

Во второй строке через пробел записаны n целых чисел a_1, ..., a_n — изначальные элементы массива силомера ($0 \leqslant a_i \leqslant 10^9$).

Следующие q строк описывают события. Первое число t_i в описании события — тип события ($1 \leqslant t \leqslant 3$). Следующие два заданные через пробел числа — это границы отрезка l_i и r_i ($1 \leqslant l_i \leqslant r_i \leqslant n$). Если это событие второго типа, то есть $t_i = 2$, далее следует число x_i, обозначающее, что надо выполнить присваивания $a_j \leftarrow x_i$ для всех $l_i \leqslant j \leqslant r_i$ ($0 \leqslant x_i \leqslant 10^9$).

Формат выходных данных

Для каждого события первого типа выведите в отдельной строке сумму элементов массива на отрезке, заданном этим событием.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты этой подзадачи и необходимых подзадач, а также тесты из условия успешно пройдены.

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи	Информация о проверке
1	12	$n, q \leqslant 5000$	–	полная
2	7	все a_i и x_i — степени двойки	–	полная
3	4	все a_i и x_i — степени двойки или нули	2	полная
4	9	$t_i \neq 2$ для всех i	–	полная
5	5	для всех событий второго типа $l_i = r_i$	4	полная
6	12	$a_i, x_i \leqslant 20$	–	полная
7	15	$n \leqslant 10^5$ и $q \leqslant 10^4$	1	полная
8	16	все события первого типа следуют строго позже событий второго и третьего типов	–	полная
9	20	нет	1 – 8	первая ошибка

Примеры

стандартный ввод	стандартный вывод
6 6 9 1 3 1 6 3 2 2 3 9 2 4 5 10 3 2 4 1 2 4 1 2 6 3 2 5	6 19
5 6 0 3 4 1 1 1 2 5 2 2 3 8 3 1 4 3 2 2 1 3 4 2 3 5 5	9 2

Задача F. Минимизируй!

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

Дан массив целых чисел a длины n. Поступает q запросов двух типов:

- $1\ l\ r\ x$. Для каждого i на отрезке от l до r включительно нужно заменить a_i на $\min(a_i, x)$.
- $2\ l\ r$. Необходимо вывести сумму элементов массива a на отрезке от l до r включительно.

Формат входных данных

В первой строке входных данных дано целое число n ($1 \leqslant n \leqslant 300\,000$) — количество элементов массива a.

Во второй строке даны n целых чисел a_i ($1 \leqslant a_i \leqslant 10^9$) — элементы массива a.

В третьей строке дано целое число q ($1 \leqslant q \leqslant 300\,000$) — количество запросов.

В последующих q строках даны запросы по одному в строке.

Запрос первого типа задается так: $1\ l\ r\ x$

Где $1 \leqslant l \leqslant r \leqslant n$, $1 \leqslant x \leqslant 10^9$ — целые числа. Это означает, что все элементы массива a на отрезке от l до r нужно заменить на минимум из текущего значения и x.

Запрос второго типа задается так: $2\ l\ r$

Где $1 \leqslant l \leqslant r \leqslant n$ — целые числа. Это означает, что нужно вывести сумму элементов массива a на отрезке от l до r.

Формат выходных данных

Для каждого запроса второго типа выведите в отдельной строке сумму элементов на соответствующем отрезке.

Примеры

стандартный ввод	стандартный вывод
3 1 4 2 5 2 1 3 1 1 3 3 2 1 3 1 1 3 1 2 1 3	7 6 3
7 1 7 2 4 8 4 100 7 1 3 6 3 2 2 7 1 2 3 5 2 1 7 1 1 7 3 2 1 4 2 2 7	118 117 9 17

Задача G. Исторический максимум

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

Дан массив a. Поступают запросы прибавления на отрезке и поиска максимума исторических максимумов на отрезке. Необходимо их обрабатывать.

Формат входных данных

В первой строке дано число n ($1 \leqslant n \leqslant 300\,000$) — количество элементов массива a.

Во второй строке даны n чисел — элементы массива a ($-10^9 \leqslant a_i \leqslant 10^9$).

В третьей строке дано число q ($1 \leqslant q \leqslant 300\,000$) — количество запросов.

В последующих q строках даны запросы.

Запрос прибавления на отрезке задается следующим образом: 1 ql qr x

Это означает, что на отрезке от ql до qr включительно ($1 \leqslant ql \leqslant qr \leqslant n$) нужно прибавить ко всем числам x ($-10^9 \leqslant x \leqslant 10^9$).

Запрос поиска исторического максимума задается следующим образом: 2 ql qr

Это означает, что нужно найти максимум исторических максимумов на отрезке от ql до qr включительно ($1 \leqslant ql \leqslant qr \leqslant n$).

Формат выходных данных

Для каждого запроса второго типа выведите ответ в отдельной строке.

Пример

стандартный ввод	стандартный вывод
5 1 2 3 4 5 5 1 1 4 2 2 2 5 1 3 5 -5 2 4 5 2 1 5	6 6 6

Задача H. Комбокамень

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

Одна большая компания разрабатывает компьютерную игру «Комбокамень», которая должна мигом перевернуть всю индустрию. Правила игры достаточно сложные, и на реализацию серверного движка, моделирующего ход игры, был объявлен конкурс. От вас требуется реализовать подобный серверный движок.

Суть игры — игроки умеют призывать на арену и усиливать существ, используя заклинания, а также заставлять их сражаться друг с другом. Каждое существо имеет два параметра — численное значение атаки a и численное значение оставшегося здоровья h. Для краткости будем обозначать параметры существа как (a, h). Исходно на арене нет существ.

Игроку доступны следующие заклинания:

- *Призыв существа*: Призвать новое существо с характеристиками $(1, 1)$. Если уже в игру было введено k существ, то новое существо получает номер $k + 1$.
- *Благословение силы*: Удвоить атаку выбранного существа. Если до применения этого заклинания оно имело характеристики (a, h), то после этого действия оно будет иметь характеристики $(2a, h)$.
- *Божественный дух*: Удвоить здоровье выбранного существа. Если до применения этого заклинания оно имело характеристики (a, h), то после этого действия оно будет иметь характеристики $(a, 2h)$.
- *Копия из лавы*: Призвать новое существо, которое будет иметь такие же характеристики, как и выбранное заклинанием существо. Если уже в игру было введено k существ, то новое существо получает номер $k + 1$.
- *Сражайся!*: Заставить двух различных существ сразиться. Во время сражения оба существа одновременно наносят друг другу по одному удару, уменьшая количество здоровья соперника на значение своей атаки. Так, если сражаются два существа с характеристиками (a_1, h_1) и (a_2, h_2), то после сражения они будут иметь характеристики $(a_1, h_1 - a_2)$ и $(a_2, h_2 - a_1)$, соответственно. Если после сражения у существа остается 0 или меньше единиц здоровья, оно умирает и больше не может участвовать в игре.

От серверного движка, на реализацию которого объявлен конкурс, требуется способность промоделировать все события и для каждого созданного во время игры существа вывести номер хода, на котором оно погибло, либо определить, что оно осталось живо к концу игры.

Кроме того, движок должен корректно обрабатывать случаи, когда игрок пытается взаимодействовать с мертвыми по мнению сервера существами: если заклинание *Благословение силы*, *Божественный дух* или *Сражайся!* обращено к уже мертвому существу, то не должно произойти ничего. Если заклинание *Копия из лавы* применено к мертвому существу, создается его мертвая копия с такими же характеристиками, но умершая на текущем ходу, в момент копирования.

Формат входных данных

В первой строке дано число n — количество совершенных ходов ($1 \leqslant n \leqslant 250\,000$).

В следующих n строках даны ходы, пришедшие к серверному движку, в следующем формате:

- 1 — применить заклинание *Призыв существа*;
- 2 i — применить заклинание *Благословение силы* к существу с номером i;
- 3 i — применить заклинание *Божественный дух* к существу с номером i;
- 4 i — применить заклинание *Копия из лавы* к существу с номером i;
- 5 i j — применить заклинание *Сражайся!* к существам с номерами i и j.

Гарантируется, что любые упомянутые в запросах существа к моменту запроса уже были призваны, но, возможно, могут уже быть мертвы.

Формат выходных данных

В первой строке выведите одно целое число k — количество существ, призванных за время игры.

В следующей строке выведите k целых чисел $t_1, t_2, \ldots, t_k$ — если существо с номером i осталось живо к концу игры, то t_i должно быть равно -1, иначе t_i должно быть равно номеру хода, на котором оно погибло.

Пример

стандартный ввод	стандартный вывод
16 1 2 1 3 1 1 5 1 2 3 1 1 3 3 3 3 4 1 5 1 3 3 3 5 1 3 5 4 3 5 4 3 4 1	5 13 5 14 -1 16

Замечание

В таблице можно увидеть, как изменялись характеристики существ в первом примере.

ход	1	2	3	4	5
0	-	-	-	-	-
1	(1, 1)	-	-	-	-
2	(2, 1)	-	-	-	-
3	(2, 2)	-	-	-	-
4	(2, 2)	(1, 1)	-	-	-
5	(2, 1)	мертво	-	-	-
6	(2, 2)	мертво	-	-	-
7	(2, 2)	мертво	(1, 1)	-	-
8	(2, 2)	мертво	(1, 2)	-	-
9	(2, 2)	мертво	(1, 4)	-	-
10	(2, 2)	мертво	(1, 4)	(2, 2)	-
11	(2, 1)	мертво	(1, 2)	(2, 2)	-
12	(2, 1)	мертво	(1, 4)	(2, 2)	-
13	мертво	мертво	(1, 2)	(2, 2)	-
14	мертво	мертво	мертво	(2, 1)	-
15	мертво	мертво	мертво	(2, 1)	-
16	мертво	мертво	мертво	(2, 1)	мертво

Задача I. Урны и шары

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

Пусть у вас есть n урн, в каждой из которых лежит по одному шарику. 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 ($1 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 50\,000$). 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 $to_i + count_i - 1$ ($1 \leqslant count_i, from_i, to_i \leqslant n$, $\max(from_i, to_i) + count_i \leqslant n + 1$).

Формат выходных данных

Выведите n чисел — итоговые позиции каждого шарика.

Примеры

balls.in	balls.out
2 3 1 1 2 1 2 1 1 2 1	1 1