

A. Сумма простая

1 секунда, 512 мегабайт

Вам нужно научиться отвечать на запрос «сумма чисел на отрезке».

Массив не меняется. Запросов много. Отвечать на каждый запрос следует за $O(1)$.

Входные данные

Размер массива — n и числа x, y, a_0 , порождающие массив a :

$$a_i = (x \cdot a_{i-1} + y) \bmod 2^{16}$$

Далее следует количество запросов m и числа z, t, b_0 , порождающие массив b : $b_i = (z \cdot b_{i-1} + t) \bmod 2^{30}$.

Массив c строится следующим образом: $c_i = b_i \bmod n$.

Запросы: i -й из них — найти сумму на отрезке от $\min(c_{2i}, c_{2i+1})$ до $\max(c_{2i}, c_{2i+1})$ в массиве a .

Ограничения: $1 \leq n \leq 10^7$, $0 \leq m \leq 10^7$. Все числа целые от 0 до 2^{16} . t может быть равно -1 .

Выходные данные

Выведите сумму всех сумм.

входные данные
3 1 2 3 3 1 -1 4
выходные данные
23

$$a = \{3, 5, 7\}, b = \{4, 3, 2, 1, 0, 2^{30} - 1\}, c = \{1, 0, 2, 1, 0, 0\},$$

запросы = $\{\{0, 1\}, \{1, 2\}, \{0, 0\}\}$, суммы = $\{8, 12, 3\}$.

B. RSQ

2 секунды, 256 мегабайт

Входные данные

В первой строке находится число n — размер массива.

($1 \leq n \leq 500\,000$) Во второй строке находится n чисел a_i — элементы массива. Далее содержится описание операций, их количество не превышает $1\,000\,000$. В каждой строке находится одна из следующих операций:

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

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

Выходные данные

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

входные данные

```
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
```

выходные данные

```
14
15
10
9
12
22
20
10
```

C. RMQ2

2 секунды, 256 мегабайт

Входные данные

В первой строке находится число n — размер массива. ($1 \leq n \leq 10^5$) Во второй строке находится n чисел a_i — элементы массива. Далее содержится описание операций, их количество не превышает $2 \cdot 10^5$. В каждой строке находится одна из следующих операций:

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

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

Выходные данные

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

входные данные

```
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
```

выходные данные

```
2
1
1
2
5
5
8
8
```

D. Художник

2 секунды, 256 мегабайт

Итальянский художник-абстракционист Ф. Мандарино увлекся рисованием одномерных черно-белых картин. Он пытается найти оптимальное местоположение и количество черных участков картины. Для этого он проводит на прямой белые и черные отрезки, и после каждой из таких операций хочет знать количество черных отрезков на получившейся картине и их суммарную длину.

Изначально прямая — белая. Ваша задача — написать программу, которая после каждой из таких операций выводит в выходной файл интересные художника данные.

Входные данные

В первой строке входного файла содержится общее количество нарисованных отрезков ($1 \leq n \leq 100\,000$). В последующих n строках содержится описание операций. Каждая операция описывается строкой вида $c\ x\ l$, где c — цвет отрезка (W для белых отрезков, B для черных), а сам отрезок имеет вид $[x; x + l)$, причем координаты обоих концов — целые числа, не превосходящие по модулю $500\,000$. Длина задается положительным целым числом.

Выходные данные

После выполнения каждой из операций необходимо вывести в выходной файл на отдельной строке количество черных отрезков на картине и их суммарную длину, разделенные одним пробелом.

входные данные

```
7
W 2 3
B 2 2
B 4 2
B 3 2
B 7 2
W 3 1
W 0 10
```

выходные данные

```
0 0
1 2
1 4
1 4
2 6
3 5
0 0
```

E. Криптография

2 секунды, 256 мегабайт

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

Входные данные

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

Выходные данные

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

входные данные

```
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
```

выходные данные

```
0 2
0 0

0 2
0 1

0 1
0 0

2 1
1 2
```

F. Разрезанные таблицы

2 секунды, 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.

Входные данные

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

Для того, размер ввода был небольшой, массив и запросы генерируются.

Элементы a_2, a_3, \dots, 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:

$$u_{i+1} = ((17 \cdot u_i + 751 + r_i + 2i) \bmod n) + 1,$$

$$v_{i+1} = ((13 \cdot v_i + 593 + r_i + 5i) \bmod n) + 1,$$

где r_i — ответ на запрос номер i .

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

Выходные данные

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

входные данные
10 8 12345 3 9
выходные данные
5 3 1565158

Можно заметить, что массивы u , v и r можно не сохранять в памяти полностью.

Запросы и ответы на них выглядят следующим образом:

i	u_i	v_i	r_i
1	3	9	570265
2	10	1	12345
3	1	2	12345
4	10	10	1325095
5	5	9	570265
6	2	1	12345
7	3	2	305498
8	5	3	1565158

Эта задача скорее всего не решается стандартными интерпретаторами Python 2 и Python 3. Используйте соответствующие компиляторы PyPy.

G. Окна

2 секунды, 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.

Входные данные

В первой строке входного файла записано число окон n ($1 \leq n \leq 50000$). 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 $2 \cdot 10^5$.

Выходные данные

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

входные данные
2 0 0 3 3 1 1 4 4

выходные данные

2
1 3

входные данные

1
0 0 1 1

выходные данные

1
0 1

H. RMQ наоборот

2 секунды, 256 мегабайт

Рассмотрим массив $a[1..n]$. Пусть $Q(i, j)$ — ответ на запрос о нахождении минимума среди чисел $a[i], \dots, a[j]$. Вам даны несколько запросов и ответы на них. Восстановите исходный массив.

Входные данные

Первая строка входного файла содержит число n — размер массива, и m — число запросов ($1 \leq n, m \leq 100\,000$). Следующие m строк содержат по три целых числа i, j и q , означающих, что $Q(i, j) = q$ ($1 \leq i \leq j \leq n$, $-2^{31} \leq q \leq 2^{31} - 1$).

Выходные данные

Если искомого массива не существует, выведите строку «inconsistent».

В противном случае в первую строку выходного файла выведите «consistent». Во вторую строку выходного файла выведите элементы массива. Элементами массива должны быть целые числа в интервале от -2^{31} до $2^{31} - 1$ включительно. Если решений несколько, выведите любое.

входные данные

3 2
1 2 1
2 3 2

выходные данные

consistent
1 2 2

входные данные

3 3
1 2 1
1 1 2
2 3 2

выходные данные

inconsistent

I. Горы

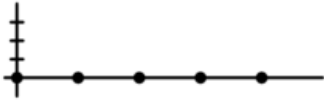
2 секунды, 256 мегабайт

В парке развлечений «Ай-ой-ай» открылся новейший аттракцион: польские горки. Трек состоит из n рельс, присоединенных одна к концу другой. Начало первой рельсы находится на высоте 0. Оператор Петя может конфигурировать аттракцион, изменяя по своему желанию подъём нескольких последовательных рельс. При этом подъём всех остальных рельс не изменяется. При каждом изменении конфигурации рельс положение следующих за изменяемыми подбирается таким образом, чтобы весь трек оставался связным.

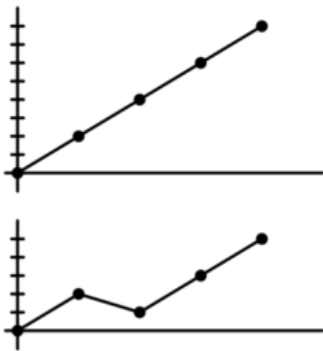
Каждый запуск вагонетки осуществляется с энергией, достаточной для достижения высоты h . Это значит, что вагонетка будет двигаться до тех пор, пока высота не превысит h , либо пока не закончится трек.

По записям о всех изменениях конфигурации рельс и временах запусков вагонетки для каждого запуска определите, сколько рельс вагонетка проедет до остановки.

Трек можно представить как последовательность n подъемов d_i , по одному на рельс. Изначально рельсы горизонтальны, то есть $d_i = 0$ для всех i .



Каждое изменение конфигурации определяется числами a , b и D : все рельсы с a -й по b -ю включительно после этого действия имеют подъем, равный D .



Каждый запуск вагонетки определяется единственным целым числом h — максимальной высотой, на которую способна подняться вагонетка.

Входные данные

В первой строке записано целое число n ($1 \leq n \leq 10^9$) — число рельс. Следующие строки содержат запросы трех видов:

- $I a b D$ — изменение конфигурации. Рельсы с a -й по b -ю включительно после выполнения запроса имеют подъем, равный D .
- $Q h$ — запуск вагонетки. Требуется найти число рельс, которое проедет вагонетка, которая способна подняться на высоту h .
- E — конец ввода. Этот запрос встретится ровно один раз в конце файла.

В любой момент времени высота любой точки трека лежит в промежутке от 0 до 10^9 . Во вводе не более 100 000 строк.

Выходные данные

Для каждого запроса Q выведите единственное целое число — количество рельс, которое проедет вагонетка.

входные данные
4
Q 1
I 1 4 2
Q 3
Q 1
I 2 2 -1
Q 3
E

выходные данные
4
1
0
3

Ж. Великая Китайская Стена

2 секунды, 256 мегабайт

В этой задаче мы проследим альтернативную историю Великой Китайской Стены.

Великая Китайская Стена состоит из n метровых участков, пронумерованных по порядку целыми числами от 1 до n . Каждый участок характеризуется своей высотой в метрах — целым неотрицательным числом. До начала нашей истории Стена ещё не построена, поэтому высота каждого участка равна нулю.

Происходят события двух видов.

1. *Укрепление Стены* (запись: «defend $a b c$ »). Император вызывает к себе вассалов из приграничных провинций и велит им сделать так, чтобы промежуток Стены, охватывающий участки от a до b включительно, имел высоту не менее c метров. Это значит, что все участки меньшей высоты на этом промежутке нужно достроить до высоты c , а остальные оставить нетронутыми. Приказ императора выполняется немедленно, то есть до наступления следующего события.
2. *Нападение варваров* (запись: «attack $d e$ »). Варвары подходят к Стене снаружи и занимают позиции напротив промежутка Стены, охватывающего участки от d до e включительно. После этого они находят такой участок на этом промежутке, у которого высота как можно меньше, и пытаются через него проникнуть на территорию Китая. Нападение также происходит немедленно, до наступления следующего события.

Для восстановления достоверной альтернативно-исторической картины не хватает одного: для каждого нападения варваров указать минимальную высоту Стены на соответствующем промежутке, а также какой-нибудь участок из этого промежутка с такой высотой. По заданной последовательности событий найдите эти числа.

Входные данные

В первой строке заданы через пробел два целых числа n и m — длина Стены в метрах и количество событий соответственно ($1 \leq n \leq 10^6$, $0 \leq m \leq 10^5$). В следующих m строках описаны события в порядке их следования. Если событие описывает укрепление Стены, оно задано в форме «defend $a b c$ » ($1 \leq a \leq b \leq n$, $1 \leq c \leq 10^7$). Если же событие описывает нападение варваров, оно задано в форме «attack $d e$ » ($1 \leq d \leq e \leq n$).

Выходные данные

В ответ на каждое нападение варваров выведите строку, содержащую два числа, разделённые пробелом. Первое из этих чисел — минимальная высота Стены на соответствующем промежутке. Второе — номер любого метрового участка Стены на этом промежутке, имеющего такую высоту.

входные данные
5 4
defend 1 3 10
attack 1 4
attack 2 3
attack 1 2

выходные данные

```
0 4
10 2
10 1
```

К. Парковка

2 секунды, 256 мегабайт

На кольцевой парковке есть n мест пронумерованных 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.

Входные данные

В первой строке входного файла находится два числа n и m — размер парковки и количество запросов ($1 \leq n, m \leq 100000$). 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.

Выходные данные

Выведите последовательно результаты выполнения всех операций enter.

входные данные

```
3 5
enter 1
enter 1
exit 1
enter 2
enter 2
```

выходные данные

```
1
2
3
1
```

Л. Звезды

2 секунды, 256 мегабайт

Вася любит наблюдать за звездами. Но следить за всем небом сразу ему тяжело. Поэтому он наблюдает только за частью пространства, ограниченной кубом размером $n \times n \times n$. Этот куб поделен на маленькие кубики размером $1 \times 1 \times 1$. Во время его наблюдений могут происходить следующие события:

1. В каком-то кубике появляются или исчезают несколько звезд.
2. К нему может заглянуть его друг Петя и поинтересоваться, сколько видно звезд в части пространства, состоящей из нескольких кубиков.

Входные данные

Первая строка входного файла содержит натуральное число $1 \leq n \leq 128$. Координаты кубиков — целые числа от 0 до $n - 1$. Далее следуют записи о происходивших событиях по одной в строке. В начале строки записано число m . Если m равно:

- 1, то за ним следуют 4 числа — x, y, z ($0 \leq x, y, z < N$) и k ($-20000 \leq k \leq 20000$) — координаты кубика и величина, на которую в нем изменилось количество видимых звезд;
- 2, то за ним следуют 6 чисел — $x_1, y_1, z_1, x_2, y_2, z_2$ ($0 \leq x_1 \leq x_2 < N, 0 \leq y_1 \leq y_2 < N, 0 \leq z_1 \leq z_2 < N$), которые означают, что Петя попросил подсчитать количество звезд в кубиках (x, y, z) из области: $x_1 \leq x \leq x_2, y_1 \leq y \leq y_2, z_1 \leq z \leq z_2$;
- 3, то это означает, что Васе надоело наблюдать за звездами и отвечать на вопросы Пети. Эта запись встречается во входном файле только один раз и будет последней.

Количество записей во входном файле не больше 100 002.

Выходные данные

Для каждого Петиного вопроса выведите искомое количество звезд.

входные данные

```
2
2 1 1 1 1 1 1
1 0 0 0 1
1 0 1 0 3
2 0 0 0 0 0 0
2 0 0 0 0 1 0
1 0 1 0 -2
2 0 0 0 1 1 1
3
```

выходные данные

```
0
1
4
2
```