

## Задача А. Сумма на отрезке

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

Дан массив из  $N$  элементов, нужно научиться находить сумму чисел на отрезке.

### Формат входных данных

Первая строка входного файла содержит два целых числа  $N$  и  $K$  — количество чисел в массиве и количество запросов ( $1 \leq N \leq 100\,000$ ,  $0 \leq K \leq 100\,000$ ). Следующие  $K$  строк содержат следующие запросы:

- A  $i$   $x$  — присвоить  $i$ -му элементу массива значение  $x$  ( $1 \leq i \leq n$ ,  $0 \leq x \leq 10^9$ );
- Q  $l$   $r$  — найти сумму чисел в массиве на позициях от  $l$  до  $r$  ( $1 \leq l \leq r \leq n$ ).

Изначально в массиве живут нули.

### Формат выходных данных

На каждый запрос вида Q  $l$   $r$  нужно вывести единственное число — сумму на отрезке.

### Примеры

<code>sum.in</code>	<code>sum.out</code>
5 9	0
A 2 2	2
A 3 1	1
A 4 2	2
Q 1 1	0
Q 2 2	5
Q 3 3	
Q 4 4	
Q 5 5	
Q 1 5	

### Замечание

TL для Python 4 секунды

## Задача В. Range Variation Query

Имя входного файла: rvq.in  
Имя выходного файла: rvq.out  
Ограничение по времени: 0.5 секунда  
Ограничение по памяти: 64 мегабайта

В начальный момент времени последовательность  $a_n$  задана следующей формулой:  $a_n = n^2 \bmod 12345 + n^3 \bmod 23456$ .

Требуется много раз отвечать на запросы следующего вида:

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

### Формат входных данных

Первая строка входного файла содержит натуральное 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 значениями 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  $y_i$ . В этом случае  $-100\,000 \leq x_i \leq -1$  и  $|y_i| \leq 100\,000$ .

### Формат выходных данных

Для каждого запроса первого типа в выходной файл требуется вывести одну строку, содержащую разность между максимальным и минимальным значениями на соответствующем отрезке.

### Примеры

rvq.in	rvq.out
7	34
1 3	68
2 4	250
-2 -100	234
1 5	1
8 9	
-3 -101	
2 3	

## Задача С. Дерево отрезков с операцией на отрезке

Имя входного файла: `segment-tree.in`  
Имя выходного файла: `segment-tree.out`  
Ограничение по времени: 0.5 секунд  
Ограничение по памяти: 256 мегабайт

Реализуйте эффективную структуру данных для хранения элементов и увеличения нескольких подряд идущих элементов на одно и то же число.

### Формат входных данных

В первой строке вводится одно натуральное число  $N$  ( $1 \leq N \leq 100\,000$ ) — количество чисел в массиве.

Во второй строке вводятся  $N$  чисел от 0 до 100 000 — элементы массива.

В третьей строке вводится одно натуральное число  $M$  ( $1 \leq M \leq 30\,000$ ) — количество запросов.

Каждая из следующих  $M$  строк представляет собой описание запроса. Сначала вводится одна буква, кодирующая вид запроса ('g' — получить текущее значение элемента по его номеру, 'a' — увеличить все элементы на отрезке).

Следом за 'g' вводится одно число — номер элемента.

Следом за 'a' вводится три числа — левый и правый концы отрезка и число  $add$ , на которое нужно увеличить все элементы данного отрезка массива ( $0 \leq add \leq 100\,000$ ).

### Формат выходных данных

Выведите в одну строку через пробел ответы на каждый запрос 'g'.

### Примеры

segment-tree.in	segment-tree.out
5	4
2 4 3 5 2	2
5	14
g 2	5
g 5	
a 1 3 10	
g 2	
g 4	

## Задача D. Знакочередование

Имя входного файла: `signchange.in`  
Имя выходного файла: `signchange.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 64 мегабайта

Реализуйте структуру данных из  $n$  элементов  $a_1, a_2, \dots, a_n$ , поддерживающую следующие операции:

- присвоить элементу  $a_i$  значение  $j$ ;
- найти знакопередающую сумму на отрезке от  $l$  до  $r$  включительно, т. е.  $(a_l - a_{l+1} + a_{l+2} - \dots - a_r)$ .

### Формат входных данных

В первой строке входного файла содержится натуральное число  $n$  ( $1 \leq n \leq 10^5$ ) — длина массива. Во второй строке записаны начальные значения элементов — неотрицательные целые числа, не превосходящие  $10^4$ .

В третьей строке находится натуральное число  $m$  ( $1 \leq m \leq 10^5$ ) — количество операций. В последующих  $m$  строках записаны операции:

- операция первого типа задаётся тремя числами  $0 \ i \ j$  ( $1 \leq i \leq n, 1 \leq j \leq 10^4$ ).
- операция второго типа задаётся тремя числами  $1 \ l \ r$  ( $1 \leq l \leq r \leq n$ ).

### Формат выходных данных

Для каждой операции второго типа выведите на отдельной строке соответствующую знакопередающую сумму.

### Пример

<code>signchange.in</code>	<code>signchange.out</code>
3	-1
1 2 3	2
5	-1
1 1 2	3
1 1 3	
1 2 3	
0 2 1	
1 1 3	

### Замечание

TL для Python 4 секунды

## Задача Е. 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 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.

### Формат входных данных

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.

### Формат выходных данных

Выведите последовательно результаты выполнения всех операций enter.

### Примеры

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

## Задача F. Диаграмма Юнга и Серёжа

Имя входного файла: `maxsum.in`  
Имя выходного файла: `maxsum.out`  
Ограничение по времени: 0.5 секунд  
Ограничение по памяти: 256 мегабайт

Посмотрим на диаграмму Юнга, состоящую из  $n$  столбцов. Немногие знают, что можно делать не только положительные, но и отрицательные слагаемые. Тогда столбцы в диаграмме просто растут не вверх, а вниз.

Серёжа изучает диаграммы Юнга. Сегодня он рассматривает некоторые подотрезки столбцов и ищет на этих отрезках подотрезки с максимальной площадью.

Помогите Серёже, а то он хочет спать и ему не думается.

### Формат входных данных

Входные данные содержат один или несколько тестовых примеров. Описание каждого из них начинается с двух чисел  $n$  и  $m$  — число столбцов диаграммы и количество интересующих Серёжу подотрезков.

В следующей строке содержится  $n$  чисел — высоты столбцов. Каждое из этих чисел по абсолютной величине не превосходит  $10^4$ .

Далее следуют описания подотрезков, каждое описание состоит из двух чисел  $l$  и  $r$ , обозначающих левый и правый конец подотрезка ( $1 \leq l \leq r \leq n$ ).

Суммарная длина всех диаграмм, а также суммарное число подотрезков не превосходит  $10^5$ .

### Формат выходных данных

Для каждого из тестовых примеров выведите  $m$  чисел: искомую максимальную площадь для каждого из подотрезков.

### Примеры

<code>maxsum.in</code>	<code>maxsum.out</code>
10 3	50
-100 1 2 3 4 -10 50 -100 -1 2	10
1 10	-1
1 5	3
9 9	3
5 2	
-1 2 -1 2 -1	
1 5	
2 4	

## Задача G. Ближайшее большее число справа

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

Дан массив  $a$  из  $n$  чисел. Нужно обрабатывать запросы:

0. `set(i, x)` — присвоить новое значение элементу массива  $a[i] = x$ ;
1. `get(i, x)` — найти  $\min k: k \geq i$  и  $a_k \geq x$ .

### Формат входных данных

Первая строка входных данных содержит два числа: длину массива  $n$  и количество запросов  $m$  ( $1 \leq n, m \leq 200\,000$ ).

Во второй строке записаны  $n$  целых чисел — элементы массива  $a$  ( $0 \leq a_i \leq 200\,000$ ).

Следующие  $m$  строк содержат запросы, каждый запрос содержит три числа  $t, i, x$ . Первое число  $t$  равно 0 или 1 — тип запроса.  $t = 0$  означает запрос типа `set`,  $t = 1$  соответствует запросу типа `get`,  $1 \leq i \leq n$ ,  $0 \leq x \leq 200\,000$ . Элементы массива нумеруются с единицы.

### Формат выходных данных

На каждый запрос типа `get` на отдельной строке выведите соответствующее значение  $k$ . Если такого  $k$  не существует, выведите  $-1$ .

### Примеры

<code>nearandmore.in</code>	<code>nearandmore.out</code>
4 5	1
1 2 3 4	3
1 1 1	-1
1 1 3	2
1 1 5	
0 2 3	
1 1 3	

### Замечание

TL для Python 8 секунд

## Задача Н. Катый ноль

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

Реализуйте эффективную структуру данных, позволяющую изменять элементы массива и вычислять индекс  $k$ -го слева нуля на данном отрезке в массиве.

### Формат входных данных

В первой строке вводится одно натуральное число  $N$  ( $1 \leq N \leq 200\,000$ ) — количество чисел в массиве. Во второй строке вводятся  $N$  чисел от 0 до 100 000 — элементы массива. В третьей строке вводится одно натуральное число  $M$  ( $1 \leq M \leq 200\,000$ ) — количество запросов. Каждая из следующих  $M$  строк представляет собой описание запроса. Сначала вводится одна буква, кодирующая вид запроса (`s` — вычислить индекс  $k$ -го нуля, `u` — обновить значение элемента). Следом за `s` вводится три числа — левый и правый концы отрезка и число  $k$  ( $1 \leq k \leq N$ ). Следом за `u` вводятся два числа — номер элемента и его новое значение.

### Формат выходных данных

Для каждого запроса `s` выведите результат. Все числа выводите в одну строку через пробел. Если нужного числа нулей на запрашиваемом отрезке нет, выводите `-1` для данного запроса.

### Примеры

<code>kthzero.in</code>	<code>kthzero.out</code>
5	4
0 0 3 0 2	
3	
u 1 5	
u 1 0	
s 1 5 3	

## Задача I. Число возрастающих подпоследовательностей

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

Задана последовательность из  $n$  чисел  $a_1, a_2, \dots, a_n$ . Подпоследовательностью длины  $k$  этой последовательности называется набор индексов  $i_1, i_2, \dots, i_k$ , удовлетворяющий неравенствам  $1 \leq i_1 < i_2 < \dots < i_k \leq n$ . Подпоследовательность называется возрастающей, если выполняются неравенства  $a_{i_1} < a_{i_2} < \dots < a_{i_k}$ .

Необходимо найти число возрастающих подпоследовательностей наибольшей длины заданной последовательности  $a_1, \dots, a_n$ . Так как это число может быть достаточно большим, необходимо найти остаток от его деления на  $10^9 + 7$ .

### Формат входных данных

Первая строка входного файла содержит целое число  $n$  ( $1 \leq n \leq 100000$ ). Вторая строка входного файла содержит  $n$  целых чисел:  $a_1, a_2, \dots, a_n$ . Все  $a_i$  не превосходят  $10^9$  по абсолютной величине.

### Формат выходных данных

В выходной файл выведите ответ на задачу.

### Примеры

subseq.in	subseq.out
5 1 2 3 4 5	1
6 1 1 2 2 3 3	8

## Задача J. Количество инверсий

Имя входного файла: `stdin`  
Имя выходного файла: `stdout`  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 256 мегабайта

Напишите программу, которая для заданного массива  $A = \langle a_1, a_2, \dots, a_n \rangle$  находит количество пар  $(i, j)$  таких, что  $i < j$  и  $a_i > a_j$ . Обратите внимание на то, что ответ может не влезать в `int`.

### Формат входных данных

Первая строка входного файла содержит натуральное число  $n$  ( $1 \leq n \leq 100\,000$ ) — количество элементов массива. Вторая строка содержит  $n$  попарно различных элементов массива  $A$  — целых неотрицательных чисел, не превосходящих  $10^9$ .

### Формат выходных данных

В выходной файл выведите одно число — количество инверсий в массиве.

### Примеры

<code>stdin</code>	<code>stdout</code>
5 6 11 18 28 31	0
8 999994 999989 999982 999972 999969 999961 999954 999950	28

## Задача К. Задача Иосифа

Имя входного файла: joseph.in  
Имя выходного файла: joseph.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 мегабайта

$N$  мальчиков стоят по кругу. Они начинают считать себя по часовой стрелке, счет ведется с единицы. Как только количество посчитанных достигает  $p$ , последний посчитанный ( $p$ -й) мальчик покидает круг, а процесс счета начинается со следующего за ним мальчика и вновь ведется с единицы.

Последний оставшийся в кругу выигрывает.

Можете ли вы посчитать, номер выигравшего мальчика в исходном кругу? (мальчики нумеруются числами от 1 до  $N$  по часовой стрелке, начиная с того самого мальчика, с которого начинался счет).

### Формат входных данных

Во входном файле два целых числа —  $N$  и  $P$  ( $1 \leq N, P \leq 10^5$ ).

### Формат выходных данных

Выведите номер выигравшего мальчика.

### Примеры

joseph.in	joseph.out
3 4	2