Задача A. 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 ($1 \leqslant N \leqslant 100\,000$, $0 \leqslant K \leqslant 100\,000$). Следующие K строк содержат следующие запросы:

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

Изначально в массиве живут нули.

Формат выходных данных

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

Примеры

sum.in	sum.out
5 9 A 2 2 A 3 1 A 4 2 Q 1 1 Q 2 2 Q 3 3 Q 4 4 Q 5 5 Q 1 5	0 2 1 2 0 5

Замечание

TL для Python 4 секунды

Задача B. Поиск максимума

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

Реализуйте структуру данных для эффективного вычисления номера максимального из нескольких подряд идущих элементов массива.

Формат входных данных

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

Во второй строке вводятся N чисел от 1 до 100 000 — элементы массива.

В третьей строке вводится одно натуральное число K ($1 \leqslant K \leqslant 30\,000$) — количество запросов на вычисление максимума.

В следующих K строках вводится по два числа — номера левого и правого элементов отрезка массива (считается, что элементы массива нумеруются с единицы).

Формат выходных данных

Для каждого запроса выведите индекс максимального элемента на указанном отрезке массива. Если максимальных элементов несколько, выведите любой их них.

Числа выводите в одну строку через пробел.

Примеры

index-max.in	index-max.out
5 2 2 2 1 5 2 2 3 2 5	3 5

Замечание

TL для Python 2 секунды

Задача C. 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 $a_1, a_2, \ldots, a_n$, поддерживающую следующие операции:

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

Формат входных данных

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

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

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

Формат выходных данных

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

Пример

signchange.in	signchange.out
3 1 2 3 5 1 1 2 1 1 3 1 2 3 0 2 1 1 1 3	-1 2 -1 3

Замечание

TL для Python 4 секунды

Задача D. 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 $\min k$: $k \geqslant i$ и $a_k \geqslant x$.

Формат входных данных

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

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

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`, $1 \leqslant i \leqslant n$, $0 \leqslant x \leqslant 200\,000$. Элементы массива нумеруются с единицы.

Формат выходных данных

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

Примеры

nearandmore.in	nearandmore.out
4 5 1 2 3 4 1 1 1 1 1 3 1 1 5 0 2 3 1 1 3	1 3 -1 2

Замечание

TL для Python 8 секунд

Задача E. 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 ($1 \leqslant N \leqslant 200\,000$) — 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 ($1 \leqslant M \leqslant 200\,000$) — 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 ($1 \leqslant k \leqslant N$). 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

kthzero.in	kthzero.out
5 0 0 3 0 2 3 u 1 5 u 1 0 s 1 5 3	4

Замечание

TL для Python 10 секунд