

Задача А. 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 знакочередующуюся 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 знакочередующуюся сумму.

Пример

signchange.in	signchange.out
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 В. 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 0 0 3 0 2 3 u 1 5 u 1 0 s 1 5 3	4

Задача С. Прибавление и максимум

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

Реализуйте эффективную структуру данных для хранения массива и выполнения следующих операций: увеличение всех элементов данного интервала на одно и то же число; поиск максимума на интервале.

Формат входных данных

В первой строке вводится одно натуральное число N ($1 \leq N \leq 100000$) – количество чисел в массиве.

Во второй строке вводятся N чисел от 0 до 100000 – элементы массива.

В третьей строке вводится одно натуральное число M ($1 \leq M \leq 30000$) – количество запросов.

Каждая из следующих M строк представляет собой описание запроса. Сначала вводится одна буква, кодирующая вид запроса (m – найти максимум на отрезке (оба конца включительно), a – увеличить все элементы на отрезке, то есть все элементы с l -го по r -й).

Следом за m вводятся два числа – левая и правая граница отрезка.

Следом за a вводятся три числа – левый и правый концы отрезка и число add , на которое нужно увеличить все элементы данного отрезка массива ($0 \leq add \leq 100000$).

Формат выходных данных

Выведите в одну строку через пробел ответы на каждый запрос m .

Примеры

<code>addandmax.in</code>	<code>addandmax.out</code>
5	4
2 4 3 1 5	104
5	104
m 1 3	
a 2 4 100	
m 1 3	
a 5 5 10	
m 1 5	

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 $\min k: k \geq i$ 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 каждой 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 Е. 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 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. Марио и трубы

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Марио собирается проходить уровень, состоящий из N последовательно расположенных труб, высота i -й трубы — a_i . Он еще не знает, где он будет располагаться изначально, и куда ему надо добраться, поэтому хочет рассмотреть несколько вариантов.

Находясь на трубе, Марио может переместиться только на соседние трубы слева и справа (если они существуют). Спускаться он может с любой высоты, также он может перемещаться между одинаковыми трубами. Подниматься Марио может только на трубу, высота которой больше высоты текущей на 1. Более формально, Марио может переместиться с трубы i на трубу j , если $|i - j| = 1$ и $a_j - a_i \leq 1$.

Однако злой динозавр Боузер хочет помешать Марио пройти уровень, для чего иногда увеличивает высоту нескольких подряд идущих труб на одно число k . Теперь Марио не может понять, удастся ли ему пройти уровень и поэтому просит вас обрабатывать два типа запросов — Боузер изменяет высоту некоторых труб, и Марио пытается пройти от одной трубы до другой.

Формат входных данных

В первой строке заданы два целых числа N и M — число труб и число запросов соответственно ($2 \leq N \leq 3 \cdot 10^5, 1 \leq M \leq 10^6$).

Следующая строка содержит N целых чисел a_i — высоты труб на уровне ($1 \leq a_i \leq 10^9$).

Далее идут M строк, содержащие описание запросов. Каждая строка имеет вид:

- $1 \ x \ y$ — может ли Марио пройти от трубы с номером x до трубы с номером y ($1 \leq x, y \leq N$). Гарантируется, что номера x и y не совпадают.
- $2 \ l \ r \ d$ — Боузер увеличивает высоты труб с l -й до r -й на величину d ($1 \leq l \leq r \leq N, -10^9 \leq d \leq 10^9$).

Формат выходных данных

Для каждого запроса первого типа нужно на отдельной строке вывести «YES», если Марио может дойти от одной трубы до другой и «NO» в противном случае (без кавычек).

Примеры

стандартный ввод	стандартный вывод
5 7	YES
1 2 3 4 5	NO
1 5 1	NO
2 2 4 3	YES
1 5 4	NO
1 1 3	
2 2 3 3	
1 2 4	
1 1 3	

Задача Н. Блин, че-то я тупанул, конечно

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

Письмо на Балабановскую спичечную фабрику: «Я 11 лет считаю спички у вас в коробках — их то 59, то 60, а иногда и 58. Вы там сумасшедшие что ли все???»

Гриша очень педантичный препод, который любит спорт. Поэтому он записал количество голов, которые он, *по его словам*, забил за каждый футбольный матч (может даже отрицательное, если он пропускал голы).

Еще Гриша считает, что главное в спорте — прогресс. Но прогресс должен быть плавным. Поэтому он хочет для разных последовательностей дней узнавать, насколько же прогресс был плавным. Мера плавности — длина максимального подотрезка вида $(x, x + 1, \dots, x + l)$, где l — длина такого отрезка.

Сокомандник Гриши, Женя, завидует результативности Гриши. Поэтому иногда он убавляет количество голов в блокнотике Гриши за некоторые матчи (иногда даже добавляет, чтобы ему не так обидно было).

А еще Гриша старший преподаватель, поэтому у него нет времени че-то там считать, поэтому вам дали данную задачу.

Формат входных данных

В первой строке находится одно целое число n — количество матчей по футболу за смену, в которых Гриша принял участие. $1 \leq n \leq 10^5$ Во второй строке находятся n чисел q_i ($|q_i| \leq 10^9$) — количество голов, который Гриша забил в каждом матче. В третьей строке находится одно целое число m ($0 \leq m \leq 100\,000$) — количество действий с блокнотиком Гриши. В следующих m строках содержится описание эксперимента.

- $+ l_i r_i d_i$ — Женя добавил d_i голов в матчах с l_i -го по r_i -й. ($1 \leq l_i \leq r_i \leq n$, $|d_i| \leq 10^9$)
- $? l_i r_i$ — Гриша попросил Вас узнать насколько у него был плавный прогресс в матчах с l_i -го по r_i -й. ($1 \leq l_i \leq r_i \leq n$)

Формат выходных данных

Для каждого действия второго типа выведите в новой строке одно число — меру плавности.

Примеры

<code>atoms.in</code>	<code>atoms.out</code>
6	3
2 3 4 3 4 4	3
5	5
? 1 6	
+ 6 6 1	
? 2 6	
+ 4 6 2	
? 1 5	

Задача I. Присваивание, прибавление и сумма

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 1024 мегабайта

Есть массив из n элементов, изначально заполненный нулями. Вам нужно написать структуру данных, которая обрабатывает три вида запросов:

- присвоить всем элементам на отрезке от l до $r - 1$ значение v ,
- прибавить ко всем элементам на отрезке от l до $r - 1$ число v ,
- узнать сумму на отрезке от l до $r - 1$.

Формат входных данных

Первая строка содержит два числа n и m ($1 \leq n, m \leq 100000$) — размер массива и число операций. Далее следует описание операций. Описание каждой операции имеет следующий вид:

- $1\ l\ r\ v$ — присвоить всем элементам на отрезке от l до $r - 1$ значение v ($0 \leq l < r \leq n$, $0 \leq v \leq 10^5$).
- $2\ l\ r\ v$ — прибавить ко всем элементам на отрезке от l до $r - 1$ число v ($0 \leq l < r \leq n$, $0 \leq v \leq 10^5$).
- $3\ l\ r$ — узнать сумму на отрезке от l до $r - 1$ ($0 \leq l < r \leq n$).

Формат выходных данных

Для каждой операции третьего типа выведите соответствующее значение.

Примеры

стандартный ввод	стандартный вывод
5 7	8
1 0 3 3	10
2 2 4 2	4
3 1 3	
2 1 5 1	
1 0 2 2	
3 0 3	
3 3 5	

Задача J. Перестановки

Имя входного файла: `permutation.in`
Имя выходного файла: `permutation.out`
Ограничение по времени: 1.5 секунды
Ограничение по памяти: 256 мегабайт

Вася выписал на доске в каком-то порядке все числа от 1 по N , каждое число ровно по одному разу. Количество чисел оказалось довольно большим, поэтому Вася не может окинуть взглядом все числа. Однако ему надо всё-таки представлять эту последовательность, поэтому он написал программу, которая отвечает на вопрос — сколько среди чисел, стоящих на позициях с x по y , по величине лежат в интервале от k до l . Сделайте то же самое.

Формат входных данных

В первой строке лежит два натуральных числа — $1 \leq N \leq 100\,000$ — количество чисел, которые выписал Вася и $1 \leq M \leq 100\,000$ — количество вопросов, которые Вася хочет задать программе. Во второй строке дано N чисел — последовательность чисел, выписанных Васей. Далее в M строках находятся описания вопросов. Каждая строка содержит четыре целых числа $1 \leq x \leq y \leq N$ и $1 \leq k \leq l \leq N$.

Формат выходных данных

Выведите M строк, каждая должна содержать единственное число — ответ на Васин вопрос.

Примеры

permutation.in	permutation.out
4 2	1
1 2 3 4	3
1 2 2 3	
1 3 1 3	

Задача К. Кислотные дожди

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Для сборки лаборатории-поселения на Венеру доставлены n блоков. Блоки расположены в ряд, i -й блок имеет высоту h_i .

Сборку будет осуществлять специальный робот. В процессе сборки последовательные сегменты блоков будут постепенно объединяться. При этом порядок блоков в ряду не будет меняться.

Исходно каждый блок представляет собой отдельный сегмент, сегменты пронумерованы от 1 до n в том же порядке, что и блоки. Если есть два соседних сегмента, составленных из блоков: сегмент из блоков $A = [i, i+1, \dots, i+p-1]$ и сегмент из блоков $B = [i+p, i+p+1, \dots, i+p+q-1]$, то после их объединения в один получается сегмент $AB = [i, i+1, \dots, i+p-1, i+p, i+p+1, \dots, i+p+q-1]$.

Инструкция по сборке состоит из $n-1$ инструкций. Каждая инструкция характеризуется одним числом, j -я инструкция характеризуется числом k_j . После выполнения этой инструкции сегменты с номерами k_j и k_j+1 объединяются в один, получившийся сегмент занимает место в последовательности сегментов на месте двух объединенных сегментов, и вводится новая нумерация на сегментах в том порядке, в котором они расположены — номера сегментов, начиная с k_j+2 , уменьшаются на один. После выполнения всех инструкций все сегменты окажутся объединены в один общий сегмент.

На Венере постоянно идут кислотные дожди, поэтому в процессе сборки важно для каждого сегмента блоков понимать, сколько жидкости может скопиться в этом сегменте. Пусть сегмент состоит из блоков высотой $h_l, h_{l+1}, \dots, h_r$. Для p , где $l \leq p \leq r$ определим *глубину* блока с высотой h_p в этом сегменте следующим образом. Посчитаем величины $l_p = \max\{h_l, \dots, h_p\}$, $r_p = \max\{h_p, \dots, h_r\}$. Это самые высокие блоки в сегменте слева и справа от p -го. Тогда глубина блока p в его сегменте равна $d_p = \min(l_p, r_p) - h_p$, заметим, что $d_p \geq 0$. *Емкостью* сегмента будем называть сумму глубин блоков этого сегмента, то есть $w = d_l + d_{l+1} + \dots + d_r$.

Задана последовательность объединений сегментов. После каждого объединения выведите емкость получившегося сегмента.

Рисунок на следующей странице показывает процесс выполнения инструкции из примера, над каждым блоком указана его глубина, а для нового сегмента показана его емкость.

Формат входных данных

Первая строка содержит одно целое число n — количество блоков ($2 \leq n \leq 10^5$).

Во второй строке записано n чисел $h_1, \dots, h_n$ ($1 \leq h_i \leq 10^9$).

В третьей строке записаны $n-1$ чисел — инструкции по объединению сегментов. Каждая инструкция характеризуется одним числом k_j ($1 \leq k_j \leq n-j$).

Формат выходных данных

Выведите $n-1$ чисел — после каждого объединения сегментов выведите емкость получившегося объединенного сегмента.

Пример

стандартный ввод	стандартный вывод
8	0
9 1 8 1 5 2 3 6	4
3 3 1 3 3 2 1	0
	0
	0
	13
	20

