

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. И снова сумма...

Имя входного файла: `sum2.in`
Имя выходного файла: `sum2.out`
Ограничение по времени: 3 секунды
Ограничение по памяти: 256 мегабайт

Реализуйте структуру данных, которая поддерживает множество S целых чисел, с которым разрешается производить следующие операции:

- $add(i)$ — добавить в множество S число i (если он там уже есть, то множество не меняется);
- $sum(l, r)$ — вывести сумму всех элементов x из S , которые удовлетворяют неравенству $l \leq x \leq r$.

Формат входных данных

Исходно множество S пусто. Первая строка входного файла содержит n — количество операций ($1 \leq n \leq 300\,000$). Следующие n строк содержат операции. Каждая операция имеет вид либо «+ i », либо «? l r ». Операция «? l r » задает запрос $sum(l, r)$.

Если операция «+ i » идет во входном файле в начале или после другой операции «+», то она задает операцию $add(i)$. Если же она идет после запроса «?», и результат этого запроса был y , то выполняется операция $add((i + y) \bmod 10^9)$.

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до 10^9 .

Формат выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

Примеры

<code>sum2.in</code>	<code>sum2.out</code>
6	3
+ 1	7
+ 3	
+ 3	
? 2 4	
+ 1	
? 2 4	

Задача С. Варенье

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

Мальш и Карлсон решили пойти на прогулку. Они знают, что прогулка будет совсем скучной, если перед ней не опустошить несколько банок варенья.

Мальш достал из кладовки N банок варенья и выставил их в ряд. В банке номер i содержится ровно a_i грамм варенья. Карлсон немного подумал и решил, что в некоторых банках недостаточно варенья, и что в банке номер i должно быть хотя бы b_i грамм варенья.

Выходить из этой ситуации Карлсон хочет в M этапов. На каждом этапе он выбирает числа l , r , x и y , а затем выполняет следующие операции: в банку номер l он добавляет x грамм варенья, в банку номер $l + 1 - x + y$ грамм варенья, в банку номер $l + 2 - x + 2 \cdot y$, и так далее. В банку номер r наш герой добавит $x + y \cdot (r - l)$ грамм варенья.

Мальшу хочется определить для каждой банки i наименьший номер операции, после которой в ней станет хотя бы b_i грамм варенья. Помогите Мальшу: найдите соответствующее число для каждой банки.

Формат входных данных

В первой строке входного файла задано одно число N ($1 \leq n \leq 10^5$) — количество банок. Во второй строке заданы N чисел a_i ($0 \leq a_i \leq 2 \cdot 10^9$) — изначальное количество варенья в банке номер i . В третьей строке заданы N чисел b_i ($0 \leq b_i \leq 2 \cdot 10^9$) — минимальное количество варенья, которое должно быть в банке номер i .

В четвертой строке задано M ($0 \leq M \leq 10^5$) — число этапов добавления варенья в банки, которые выполнит Карлсон. В следующих M строках описаны сами этапы в хронологическом порядке. Каждый этап задан четырьмя числами l, r, x и y ($1 \leq l \leq r \leq N, 0 \leq x, y \leq 3 \cdot 10^5$).

Формат выходных данных

Выведите N чисел в одной строке, разделенные пробелом. Число номер i должно быть равно нулю, если в банке номер i изначально было достаточно варенья, номеру этапа, после которого в ней станет хотя бы b_i варенья, или -1 , если даже после выполнения всех этапов, в этой банке будет недостаточно варенья. Этапы нумеруются с единицы.

Примеры

jam.in	jam.out
5	1 2 0 3 -1
5 4 4 2 1	
7 7 4 7 7	
3	
1 2 2 0	
2 5 1 1	
3 4 2 2	

Задача D. Фаброзавры-дизайнеры

Имя входного файла: `fabro.in`
Имя выходного файла: `fabro.out`
Ограничение по времени: 3 секунды
Ограничение по памяти: 64 мегабайта

Фаброзавры известны своим тонким художественным вкусом и увлечением ландшафтным дизайном. Они живут около очень живописной реки и то и дело перестраивают тропинку, идущую вдоль реки: либо насыпают дополнительной земли, либо срывают то, что есть. Для того, чтобы упростить эти работы, они поделили всю тропинку на горизонтальные участки, пронумерованные от 1 до N , и их переделки устроены всегда одинаково: они выбирают часть дороги от L -ого до R -ого участка (включительно) и изменяют (увеличивают или уменьшают) высоту на всех этих участках на одну и ту же величину (если до начала переделки высоты были разными, то и после переделки они останутся разными).

Поскольку, как уже говорилось, у фаброзавров тонкий художественный вкус, каждый из них считает, что их река лучше всего выглядит с определенной высоты. Поэтому им хочется знать, есть ли поблизости от их дома место на тропинке, где высота на их взгляд оптимальна. Помогите им в этом разобраться.

Формат входных данных

Первая строка входного файла содержит два числа N и M — длину дороги и количество запросов соответственно ($1 \leq N, M \leq 10^5$). На второй строке содержатся N чисел, разделенных пробелами — начальные высоты соответствующих частей дороги; высоты не превосходят 10^4 по модулю. В следующих M строках содержатся запросы по одному на строке.

Запрос $+ L R X$ означает, что высоту частей дороги от L -ой до R -ой (включительно) нужно изменить на X . При этом $1 \leq L \leq R \leq N$, а $|X| \leq 10^4$.

Запрос $? L R X$ означает, что нужно проверить, есть ли между L -ым и R -ым участками (включая эти участки) участок, где дорога проходит точно на высоте X . Гарантируется, что $1 \leq L \leq R \leq N$, а $|X| \leq 10^9$.

Формат выходных данных

На каждый запрос второго типа нужно вывести в выходной файл на отдельной строке одно слово «YES» (без кавычек), если нужный участок существует, и «NO» в противном случае.

Примеры

<code>fabro.in</code>	<code>fabro.out</code>
10 5	NO
0 1 1 3 3 3 2 0 0 1	YES
? 3 5 2	YES
+ 1 4 1	
? 3 5 2	
+ 7 10 2	
? 9 10 3	

Задача Е. Мощные юнги

Имя входного файла: `power.in`
Имя выходного файла: `power.out`
Ограничение по времени: 5 секунд
Ограничение по памяти: 256 мегабайт

Имеется список из n юнг, для каждого из которых известен его рост a_1, a_2, \dots, a_n . Рассмотрим некоторый его подсписок a_l, a_{l+1}, \dots, a_r , где $1 \leq l \leq r \leq n$, и для каждого натурального числа s обозначим через K_s число юнг с ростом s в этом подсписке. Назовем *мощностью* подсписка сумму произведений $K_s \cdot K_s \cdot s$ по всем различным натуральным s . Так как количество различных чисел в массиве конечно, сумма содержит лишь конечное число ненулевых слагаемых.

Необходимо вычислить мощности каждого из t заданных подсписков.

Формат входных данных

Первая строка содержит два целых числа n и t ($1 \leq n, t \leq 200000$) — длина списка и количество запросов соответственно.

Вторая строка содержит n натуральных чисел a_i ($1 \leq a_i \leq 10^6$) — рост юнг.

Следующие t строк содержат по два натуральных числа l и r ($1 \leq l \leq r \leq n$) — индексы левого и правого концов соответствующего подсписка.

Формат выходных данных

Выведите t строк, где i -ая строка содержит единственное натуральное число — мощность подсписка i -го запроса.

Примеры

<code>power.in</code>	<code>power.out</code>
3 2	3
1 2 1	6
1 2	
1 3	
8 3	20
1 1 2 2 1 3 1 1	20
2 7	20
1 6	
2 7	

Задача J. Соединение и разъединение

Имя входного файла: `connect.in`
Имя выходного файла: `connect.out`
Ограничение по времени: 3 секунды
Ограничение по памяти: 512 мегабайт

Вы когда-нибудь слышали про обход в глубину? Например, используя этот алгоритм, вы можете проверить является ли граф связным за время $O(E)$. Вы можете даже посчитать количество компонент связности за то же время.

А вы когда-нибудь слышали про систему непересекающихся множеств? Используя эту структуру, вы можете быстро обрабатывать запросы “Добавить ребро в граф” и “Посчитать количество компонент связности в графе”.

А вы когда-нибудь слышали о *динамической* задаче связности? В этой задаче вам необходимо обрабатывать три типа запросов:

1. Добавить ребро в граф.
2. Удалить ребро из графа.
3. Посчитать количество компонент связности в графе.

Можно считать, что граф является неориентированным. Изначально граф является пустым.

Формат входных данных

В первой строке находятся два целых числа N и K — количество вершин и количество запросов, соответственно ($1 \leq N \leq 300\,000$, $0 \leq K \leq 300\,000$). Следующие K строк содержат запросы, по одному в строке. Каждый запрос имеет один из трех типов:

1. $+ u v$: Добавить ребро между вершинами u и v . Гарантируется, что такого ребра нет.
2. $- u v$: Удалить ребро между u и v . Гарантируется, что такое ребро есть.
3. $?$: Посчитать количество компонент связности в графе.

Вершины пронумерованы целыми числами от 1 до N . Во всех запросах $u \neq v$.

Формат выходных данных

Для каждого запроса типа ‘?’, Выведите количество компонент связности в момент запроса.

Примеры

<code>connect.in</code>	<code>connect.out</code>
5 11	5
?	1
+ 1 2	1
+ 2 3	2
+ 3 4	
+ 4 5	
+ 5 1	
?	
- 2 3	
?	
- 4 5	
?	