Задача A. Транзитивное замыкание

Имя входного файла:	`floyd32.in`
Имя выходного файла:	`floyd32.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Дан ориентированный граф. Найдите его транзитивное замыкание, то есть для каждой пары вершин a, b определите, есть ли путь из a в b.

Формат входных данных

На первой строке число вершин n ($1 \leqslant n \leqslant 1\,000$). Следующие n строк имеют длину n, состоят из нулей и единиц и задают матрицу смежности графа. Единица в i-й строке, j-м столбце обозначает ребро из i в j.

Формат выходных данных

Выведите матрицу смежности транзитивного замыкания данного графа.

Примеры

floyd32.in	floyd32.out
3 010 001 000	011 001 000

Задача B. Функция gorilla

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

SSE инструкции, SSE инструкции...

```
__m128i gorilla(int *arr, int *brr, int *crr) {
  __m128i rega = _mm_loadu_si128((__m128i *)arr);
  __m128i regb = _mm_loadu_si128((__m128i *)brr);
  __m128i regc = _mm_loadu_si128((__m128i *)crr);
  return _mm_add_epi64(
      _mm_and_si128(_mm_xor_si128(_mm_shuffle_epi32(rega, 0b10001101),
                                  _mm_or_si128(_mm_sub_epi16(regb, regc),
                                               _mm_setzero_si128())),
                    _mm_set1_epi8(-1)),
      _mm_max_epu32(rega, regc));
}
```

Формат входных данных

В этой задаче ровно один тест.

Даны 4 целых числа типа `int` — результат вызова функции `gorilla`.

Формат выходных данных

В первой строке выведите 4 целых числа — массив `arr`.

Во второй строке выведите 4 целых числа — массив `brr`.

В третьей строке выведите 4 целых числа — массив `crr`.

Если ответов несколько, можно вывести любой.

Пример

стандартный ввод	стандартный вывод
42 1984 42 1	YOUR OUTPUT HERE

Задача C. Поиск подстроки

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

Найти позиции всех вхождений строки T в строку S.

Формат входных данных

Первые две строки входных данных содержат строки S и T, соответственно. Длины строк больше 0 и меньше 50000, строки содержат только латинские буквы.

Формат выходных данных

Выведите номера символов, начиная с которых строка T входит в строку S, в порядке возрастания.

Примеры

стандартный ввод	стандартный вывод
ababbababa aba	0 5 7

Задача D. 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 E. Фонари

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

В детском лагере «Берендеевы поляны» вряд ли найдётся кто-нибудь, кто не побоялся бы сходить ночью на хутор. В связи с этим администрация лагеря установила вдоль дороги N фонарей. Лампочки в фонарях могут перегорать, и, хотя в «Берендеевых полянах» есть электрик, он может заболеть и не успеть заменить перегоревшую лампочку на новую до приезда очередной комиссии.

Каждая посещающая лагерь комиссия считает своим долгом проинспектировать состояние фонарей, освещающих дорогу на хутор, но дорога эта очень длинная, поэтому каждый раз членов комиссии подвозят на машине к какому-нибудь фонарю, они выходят, идут пешком до другого фонаря, осматривая по пути все фонари, и там садятся в машину и уезжают.

Если они замечают хотя бы один перегоревший фонарь, в «Берендеевых полянах» устраивается штрафное посвящение для преподавателей.

По информации о происходящих событиях для каждого приезда комиссии выведите результат осмотра фонарей.

Формат входных данных

В первой строке входного файла записаны два числа N и k ($1 \leqslant N, k \leqslant 100\,000$) — количество фонарей и количество событий. Далее следуют k строк, описывающих события. События описываются следующим образом:

- перегорание фонаря задаётся двумя числами, первое из которых равно -1, а второе задаёт номер перегоревшего фонаря (от 1 до N);
- замена лампочки электриком задаётся двумя числами, первое из которых равно 1, а второе задаёт номер заменённой лампочки (от 1 до N);
- приезд комиссии задается тремя числами, первое из которых равно 0, а два других (a и b) задают отрезок дороги, по которому проходит комиссия. А именно, члены комиссии видят все фонари с номерами от a до b включительно и только их. Гарантируется, что $1 \leqslant a \leqslant b \leqslant N$.

Изначально все фонари исправны.

Формат выходных данных

Для каждого приезда комиссии выведите в выходной файл одну строку. А именно, если члены комиссии не заметят ни одного перегоревшего фонаря, то выведите «`PASSED`», иначе — «`PENALTY`».

Пример

streetlamps.in	streetlamps.out
5 6 -1 2 0 1 1 0 2 5 -1 4 1 2 0 2 3	PASSED PENALTY PASSED

Задача F. 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 ($1 \leqslant n, m \leqslant 100000$) — размер массива и число операций. Далее следует описание операций. Описание каждой операции имеет следущий вид:

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

Формат выходных данных

Для каждой операции третьего типа выведите соответствующее значение.

Примеры

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

Задача G. RMQ

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

Дан массив `a[1..n]`. Требуется написать программу, обрабатывающую два типа запросов.

- Запрос "`max` l r". Требуется найти максимум в массиве `a` от l-ой ячейки до r-ой включительно.
- Запрос "`add` l r v". Требуется прибавить значение v к каждой ячейке массива `a` от l-ой до r-ой включительно.

Формат входных данных

Первая строка содержит два целых числа n и q ($1 \leqslant n, q \leqslant 10^5$) – длина массива и число запросов соответственно. Вторая строка содержит n целых чисел $a_1, \ldots, a_n$ ($|a_i| \leqslant 10^5$), задающих соответствующие значения массива. Следующие q строк содержат запросы.

В зависимости от типа запрос может иметь вид либо "`max` l r", либо "`add` l r v". При этом $1 \leqslant l \leqslant r \leqslant n$, $|v| \leqslant 10^5$.

Формат выходных данных

Для каждого запроса вида "`max` l r" требуется в отдельной строке выдать значение соответствующего максимума.

Примеры

rmq.in	rmq.out
5 3 1 2 3 4 -5 max 1 3 add 1 2 5 max 1 3	3 7

Задача H. Перестановки strike back

Имя входного файла:	`permutation2.in`
Имя выходного файла:	`permutation2.out`
Ограничение по времени:	5 секунд
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. Иногда он стирает какое-то число и записывает на его место другое. Количество чисел, выписанных Васей, оказалось довольно большим, поэтому Вася не может окинуть взглядом все числа. Однако ему надо всё-таки представлять эту последовательность, поэтому он написал программу, которая в любой момент 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 — $1 \leqslant N \leqslant 100\,000$ — количество чисел, которые выписал Вася и $1 \leqslant M \leqslant 100\,000$ — суммарное количество вопросов и изменений сделанных Васей. Во второй строке дано N чисел — последовательность чисел, выписанных Васей. Далее в M строках находятся описания вопросов. Каждый запрос на изменение числа в некоторой позиции начинается со слова `SET` и имеет вид `SET a b` ($1 \leqslant a \leqslant N$, $1 \leqslant b \leqslant N$). Это означает, что Вася изменил число, записанное в позиции a на число b. Каждый Васин вопрос начинается со слова `GET` и имеет вид `GET x y k l` ($1 \leqslant x \leqslant y \leqslant N$, $1 \leqslant k \leqslant l \leqslant N$).

Формат выходных данных

Для каждого Васиного вопроса выведите единственное число — ответ на Васин вопрос.

Примеры

permutation2.in	permutation2.out
4 4 1 2 3 4 GET 1 2 2 3 GET 1 3 1 3 SET 1 4 GET 1 3 1 3	1 3 2

Задача I. 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 ($1 \leqslant N, M \leqslant 10^5$). 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 $1 \leqslant L \leqslant R \leqslant N$, а $|X| \leqslant 10^4$.

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 $1 \leqslant L \leqslant R \leqslant N$, а $|X| \leqslant 10^9$.

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

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

Задача J. Прибавление арифметической прогрессии

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	10 секунд
Ограничение по памяти:	256 мегабайт

В этой задаче вам надо написать не целую программу, а две функции:

1. `unsigned GetMin(const unsigned *a, int n);`

   Эта функция считает минимум в массиве a длиной n, то есть считает минимум от элементов $a[0], a[1], a[2], \ldots, a[n-1]$. Например `GetMin({2, 1, 3}, 3)` $= 1$.

2. `void AddProg(unsigned *a, int n, unsigned k);`

   Эта функция прибавляет к массиву a длиной n арифметическую прогрессию с коэффициентом k. То есть к $a[0]$ прибавляется k, к $a[1]$ прибавляется $k \cdot 2$, к $a[2]$ прибавляется $k \cdot 3$, и так далее, к $a[n-1]$ прибавляется $k \cdot n$. Функция должна модифицировать сам массив a. Прибавления делаются по модулю 2^{32}. Например $AddProg(\{1, 2, 3\}, 3, 1)$ сделает массив равным $\{2, 3, 4\}$

Функции суммарно вызовутся не более 200 000 раз, n не превышает 200 000.

Замечание

Про то как работает AVX и как его примерно использовать написано здесь:
`https://www.codeproject.com/Articles/874396/Crunching-Numbers-with-AVX-and-AVX`
Все инструкции AVX написанны здесь:
`https://www.laruence.com/sse/#techs=AVX2`
Не забудьте в начале программы написать

```
#include <immintrin.h>
```

Никакие прагмы писать не надо, они работать всё равно не будут.

Задача K. Магистр Горилл

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Сложнейшая задача на AVX, это она есть. Функцию горилла описать должны вы.

```
int gorilla(const int *a, int n, const int *b, int m)
```

Две **строго возрастающих** последовательности целых чисел принимает она. Последовательности чисел a и b, длинами n и m даны.

Сколько чисел как в a, так и в b есть, возвращать горилла должна.

Формат входных данных

Функция ваша вызываться будет много раз на последовательностях длины не больше 200.

Формат выходных данных