Задача A. Два измерения

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

Ученые планируют провести важный эксперимент с использованием исследовательского модуля на планете X-2019. В процессе эксперимента будет проведено два измерения: основное и контрольное. Каждое измерение занимает ровно один час и должно начинаться спустя целое число часов после начала работы исследовательского модуля.

Данные эксперимента планируется немедленно передать на орбитальную станцию. Канал связи с орбитальной станцией будет установлен с l-го по r-й час от начала работы исследовательского модуля, включительно. Кроме того, согласно плану эксперимента между измерениями планета должна совершить целое число оборотов вокруг своей оси. Планета X-2019 осуществляет оборот вокруг своей оси за a часов.

Таким образом, если измерения осуществляются на i-м и j-м часу, то должно выполняться неравенство $l \leqslant i < j \leqslant r$, а величина $(j - i)$ должна быть кратна a. Теперь учёным необходимо понять, сколько существует различных способов провести измерения.

Требуется написать программу, которая по заданным границам времени измерений l и r и периоду обращения планеты вокруг своей оси a определяет количество возможных способов провести измерения: количество пар целых чисел i и j, таких что $l \leqslant i < j \leqslant r$, и величина $(j - i)$ кратна a.

Формат входных данных

Входные данные содержат три целых числа, по одному на строке: l, r и a $(1 \leqslant l < r \leqslant 10^9, 1 \leqslant a \leqslant 10^9)$.

Формат выходных данных

Выведите одно целое число: количество способов провести измерения.

Пример

стандартный ввод	стандартный вывод
1 5 2	4
4 9 6	0

Задача B. Полные квадраты

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

С целью поиска закономерностей иногда полезно сгенерировать длинную последовательность по определенным правилам. Известно, например, что последовательность 0, 0 + 1, 0 + 1 + 3, 0 + 1 + 3 + 5, ... , 0 + 1 + 3 + ... + $(2n-1)$, ..., составленная из сумм нескольких первых нечетных натуральных чисел, состоит из квадратов целых чисел: 0, 1, 4, 9, ..., n^2, ...

Обобщим эту последовательность следующим образом: будем использовать вместо начального значения не ноль, а число k. Получим последовательность: $k, k+1, k+1+3, k+1+3+5, ..., k+1+3+...+(2n-1), ...$. В отличие от случая $k = 0$, в этой последовательности могут встречаться не только полные квадраты. Необходимо найти минимальное целое неотрицательное число, квадрат которого встречается в этой последовательности.

Требуется написать программу, которая по заданному целому числу k определяет, квадрат какого минимального неотрицательного целого числа встречается в описанной последовательности, либо выясняет, что в ней вообще не встречается полных квадратов.

Формат входных данных

В единственной строке содержится целое число k — начальное число в последовательности ($-10^{12} \leqslant k \leqslant 10^{12}$).

Обратите внимание, что для считывания и хранения такого большого числа необходимо использовать 64-битный тип данных.

Формат выходных данных

Выведите минимальное неотрицательное целое число, квадрат которого встречается в описанной последовательности. Если в последовательности не встречается квадратов целых чисел, выведите «none».

Пример

стандартный ввод	стандартный вывод
0	0
-5	2
2	none

Задача C. Автоматизация склада

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

Компания занимается автоматизацией склада. На складе хранятся n видов товаров, пронумерованных от 1 до n, каждый вид товара хранится в своём помещении. Товар вида i хранится в помещении с номером i.

Специальный робот обслуживает запросы по получению товаров со склада. Для доступа в помещения склада робот использует специальные электронные карты. Карты у робота хранятся в специальном отсеке, из которого он может вынуть верхнюю карту. Вынутую карту робот может вернуть в отсек на любое место: на верхнюю позицию, между любыми двумя картами или на самую нижнюю позицию.

Чтобы открыть помещение, робот действует следующим образом. Он вынимает карты из отсека для их хранения и возвращает их обратно в отсек, пока на верхней позиции не окажется карта от помещения, которое ему необходимо открыть. После этого, вынув эту карту, робот использует её, чтобы открыть помещение, и затем также возвращает в отсек для хранения карт. Если суммарно роботу потребовалось вынуть из отсека x карт, включая ту, которой он в итоге открыл помещение, будем говорить, что для открытия помещения робот совершил x действий.

В начале рабочего дня роботу поступил заказ на выдачу m товаров: $a_1, a_2, ..., a_m$. Робот должен выдать товары именно в этом порядке. Для этого он последовательно выполняет следующие действия: открывает помещение, в котором лежит очередной товар, берет товар, закрывает помещение и выдаёт товар клиенту. После этого робот переходит к выдаче следующего товара.

Исходно электронные карты лежат в отсеке в следующем порядке, от верхней к нижней: $b_1, b_2, ..., b_n$. Для каждого помещения в отсеке лежит ровно одна карта.

Время выдачи товаров со склада зависит от того, сколько раз суммарно роботу придётся вынимать верхнюю карту из отсека для их хранения, чтобы найти карту от очередного помещения. Необходимо таким образом выбрать места, куда робот должен возвращать вынутые карты, чтобы минимизировать суммарное количество действий робота для открытия помещений.

Требуется написать программу, которая по заданным целым числам n и m, последовательности выдаваемых товаров $a_1, a_2, ..., a_m$ и начальному положению карт в отсеке для хранения $b_1, b_2, ..., b_n$ определяет, какое минимальное количество действий придется совершить роботу, чтобы открыть все помещения в необходимом порядке. Для каждой вынутой карты необходимо также указать позицию, на которую её необходимо вернуть, чтобы добиться оптимального количества действий.

Формат входных данных

Первая строка входных данных содержит два целых числа n и m ($1 \leqslant n, m \leqslant 310^5$) — количество видов товаров и количество товаров, которые необходимо выдать со склада.

Вторая строка содержит m целых чисел $a_1, a_2, ..., a_m$ ($1 \leqslant a_i \leqslant n$) — типы товаров, которые необходимо выдать со склада, перечисленные в том порядке, в котором это необходимо сделать.

Третья строка содержит n различных целых чисел $b_1, b_2, ..., b_n$ ($1 \leqslant b_i \leqslant n$) — порядок, в котором карты исходно находятся в отсеке для их хранения, перечисленные от верхней к

нижней.

Формат выходных данных

Первая строка должна содержать число k — минимальное количество действий, которое потребуется совершить роботу, чтобы выдать товары в заданном порядке.

Далее выведите k чисел. Для каждого действия робота выведите одно число: позицию, на которую ему следует вернуть вынутую карту в отсек для хранения. Если карта возвращается на самую верхнюю позицию, следует вывести 1, если после одной карты, 2, и так далее, для последней позиции следует вывести n.

Если существует несколько способов минимизировать суммарное число действий, выведите любой из них.

Пример

стандартный ввод	стандартный вывод
1 1 1 1	1 1
4 5 4 1 2 4 4 4 3 2 1	7 4 4 2 4 4 1 4
2 2 1 2 2 1	3 2 2 2

Задача D. Машинное обучение

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

В лаборатории искусственного интеллекта разработали новый метод машинного обучения. В процессе обучения программы используется n итераций. Каждая итерация заключается в том, что обучаемая программа запускается на некотором обучающем наборе.

Были подготовлены обучающие наборы сложности от 0 до k. План обучения задаётся массивом целых чисел $[a_1, a_2, ..., a_n]$, где a_i задаёт сложность набора, используемого на i-й итерации обучения. Для всех i от 1 до n должно выполняться неравенство $0 \leqslant a_i \leqslant k$.

Выяснилось, что эффективность плана обучения зависит от битовых представлений сложностей обучающих наборов. Для того, чтобы план был эффективным, необходимо, чтобы для любых двух значений i и j, где $1 \leqslant i < j \leqslant n$, выполнялось $(a_i and a_j) = a_i$. Напомним, что побитовое «и» (and) двух целых неотрицательных чисел устроено следующим образом: запишем оба числа в двоичной системе счисления, i-й двоичный разряд результата равен 1, если у обоих аргументов он равен 1. Например, $(14 and 7) = (1110_2 and 0111_2) = 110_2 = 6$. Эта операция реализована во всех современных языках программирования, в языках C++, Java и Python она записывается как «&», в Паскале как «and».

Однако постоянное использование наборов одной сложности не даёт прогресса в обучении. Чтобы этого избежать, для плана обучения должны быть выполнены m требований следующего вида. Каждое требование задаётся двумя числами l_i и r_i и означает, что $a_{l_i} \neq a_{r_i}$.

Сотрудники лаборатории хотят найти количество эффективных планов, которые удовлетворяют всем требованиям. Так как это число может быть очень большим, нужно найти его остаток от деления на $10^9 + 7$.

Требуется написать программу, которая по заданным целым числам n и k, а также m требованиям вида l_i, r_i определяет количество эффективных планов, которые удовлетворяют всем требованиям, и выводит остаток от деления этого количества на число $10^9 + 7$.

Формат входных данных

Первая строка входных данных содержит три целых числа n, m и k — количество итераций обучения, количество требований и максимальную сложность обучающего набора ($1 \leqslant n \leqslant 3 \times 10^5, 0 \leqslant m \leqslant 3 \times 10^5, 0 \leqslant k \leqslant 10^{18}$).

Следующие m строк описывают требования, i-я строка содержит два целых числа l_i, r_i, которые означают, что $a_{l_i} \neq a_{r_i}$ ($1 \leqslant l_i < r_i \leqslant n$). Гарантируется, что все требования различны.

Формат выходных данных

Выведите одно целое число — остаток от деления количества эффективных планов, удовлетворяющих всем требованям, на число $10^9 + 7$.

Пример

стандартный ввод	стандартный вывод
2 0 3	9
3 1 2 1 2	2