Задача B. Мутанты

Имя входного файла:	`mutants.in`
Имя выходного файла:	`mutants.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	128 мегабайта

Уже долгое время в Институте Искусств, Мутантов и Информационных Технологий разводят милых разноцветных зверюшек. Для удобства каждый цвет обозначен своим номером, всего цветов не более 10^9. В один из прекрасных дней в питомнике случилось чудо: все зверюшки выстроились в ряд в порядке возрастания цветов. Пользуясь случаем, лаборанты решили посчитать, сколько зверюшек разных цветов живет в питомнике, и, по закону жанра, попросили вас написать программу, которая поможет им в решении этой нелегкой задачи.

Формат входных данных

В первой строке входного файла содержится единственное число N ($0 \leqslant N \leqslant 10^5$) — количество зверюшек в Институте. В следующей строке находятся N упорядоченных по неубыванию неотрицательных целых чисел, не превосходящих 10^9 и разделенных пробелами — их цвета. В третьей строке файла записано число M ($1 \leqslant M \leqslant 100\,000$) — количество запросов вашей программе, в следующей строке через пробел записаны M целых неотрицательных чисел (не превышающих $10^9 + 1$).

Формат выходных данных

Выходной файл должен содержать M строчек. Для каждого запроса выведите число зверюшек заданного цвета в питомнике.

Примеры

mutants.in	mutants.out
10 1 1 3 3 5 7 9 18 18 57 5 57 3 9 1 179	1 2 1 2 0

Задача B. Топологическая сортировка

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

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входных данных

В первой строке входного файла даны два целых числа N и M ($1 \leqslant N \leqslant 100\,000, 0 \leqslant M \leqslant 100\,000$) — количества вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходных данных

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, вывести «`-1`».

Примеры

topsort.in	topsort.out
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5

Задача C. Слонёнок

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

Шёл 2034 год. ЛКШ продолжалась вечно. Тигрёнок (Илья Матюхин) вырос и привёз в ЛКШ своего сына, которого здесь все называют Слонёнком. Конечно же, он живёт в ГК и каждый день ест вечёрку.

Исторически в ЛКШ сложилась следующая традиция поедания вечёрки в ГК:

- По заранее заказанному списку, который определяется один раз на всю смену, все напитки выставляются в ряд слева направо. К 2034 году помимо привычных вам снежка, молочка, ряженки и кефирчика появились другие, не менее вкусные напитки.
- Преподаватели берут напитки с правого края и не берут очередной напиток, если ещё можно допить один из стоящих справа.
- Школьники действуют аналогично, но выбирают напитки слева направо.

Когда все школьники уходят спать, а преподаватели начинают их укладывать, в ГК появляется Слонёнок и наслаждается оставшейся вечёркой.

Так как школьники употребляют напитки слева направо, а преподаватели справа налево, то для Слонёнка остаётся какой-то подотрезок напитков.

От каждой вечёрки Слонёнок получает определённое количество радости. Радость, полученная им от вечёрки в определённый день может быть вычислена по следующей формуле:

$$H = \sum_{p=1}^{10\,000} p \cdot cnt(p)$$

Здесь p — вид вечёрки (кефирчик, ряженка, сок, чай и другие), а $cnt(p)$ — количество напитка p среди тех, которые остались для Слонёнка. Как видно из формулы, удовольствие получаемое Слонёнком от данного вида вечёрки пропорционально его номеру.

Смена в ЛКШ длится t дней. В день с номером i Слонёнку оставят напитки с номерами от l_i до r_i включительно.

От вас требуется посчитать, сколько радости получит Слонёнок от вечёрки в каждый из дней.

Формат входных данных

В первой строке ввода записаны два числа n и t ($1 \leqslant n, t \leqslant 100\,000$) — количество напитков на вечёрке в ГК и количество дней в смене соответственно.

Во второй строке ввода содержатся n чисел. i-е число описывает вид напитка в вечёрке на позиции i. Все числа в этой строке целые положительные и не превосходят 10 000.

В следующих t строках записаны пары чисел, описывающие оставшуюся вечёрку в каждый из дней смены. В i-й из данных строк содержится пара чисел l_i и r_i, которая означает, что в день с номером i Слонёнку оставят напитки из вечёрки с номерами от l_i до r_i ($1 \leqslant l_i \leqslant r_i \leqslant n$).

Формат выходных данных

Выведите t строк. В i-й из них выведите количество радости, полученной Слонёнком от вечёрки в день с номером i.

Примеры

elephant-sum.in	elephant-sum.out
19 2 3 3 1 9 3 3 1 5 5 5 5 9 9 1 3 9 5 1 5 2 7 10 10	20 5

Замечание

В первый день Слонёнку достанутся 6 напитков, среди которых будут присутствовать виды 1, 3 и 9, встречающиеся 2, 3 и 1 раз соответственно. Удовольствие Слонёнка будет равно $1 \cdot 2 + 3 \cdot 3 + 9 \cdot 1 = 20$.

Во второй день Слонёнок сможет насладиться только одним напитком типа 5. Удовольствие Слонёнка будет равно $5 \cdot 1 = 5$.

Задача D. Задача Иосифа

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

N мальчиков стоят по кругу. Они начинают считать себя по часовой стрелке, счет ведется с единицы. Как только количество посчитанных достигает p, последний посчитанный (p-й) мальчик покидает круг, а процесс счета начинается со следующего за ним мальчика и вновь ведется с единицы.

Последний оставшийся в кругу выигрывает.

Можете ли вы посчитать, номер выигрывшего мальчика в исходном кругу? (мальчики нумеруются числами от 1 до N по часовой стрелке, начиная с того самого мальчика, с которого начинался счет).

Формат входных данных

Во входном файле два целых числа — N и P ($1 \leqslant N, P \leqslant 10^5$).

Формат выходных данных

Выведите номер выигравшего мальчика.

Примеры

joseph.in	joseph.out
3 4	2

Задача E. Сумма подмножества

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

Дано множество из N натуральных чисел. Существует ли в нем такое подмножество, сумма элементов в котором равна S?

Формат входных данных

В первой строке вводятся числа N (натуральное, не превышает 12) и S (натуральное, не превосходит 10000), в следующей строке вводятся элементы исходного множества — натуральные числа, не превосходят 100.

Формат выходных данных

Выведите слово `'YES'`, если существует такое подмножество, сумма элементов которого равна S, или вывести `'NO'` в противном случае.

Примеры

subset-sum.in	subset-sum.out
1 5 10	NO
3 8 5 4 3	YES

Задача F. Транзитивное замыкание

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

Невзвешенный ориентированный граф задан своей матрицей смежности. Требуется построить его транзитивное замыкание, то есть матрицу, в которой в i-й строке и j-м столбце находится 1, если от вершины i можно добраться до вершины j, и 0 — иначе.

Формат входных данных

В первой строке дано число N ($1 \leqslant N \leqslant 100$) — число вершин в графе. Далее задана матрица смежности графа: в N строках даны по N чисел 0 или 1 в каждой. i-е число в i-й строке всегда равно 1.

Формат выходных данных

Необходимо вывести матрицу транзитивного замыкания графа в формате, аналогичным формату матрицы смежности.

Примеры

closure.in	closure.out
4 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 1	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1

Задача A. Предок

Имя входного файла:	`ancestor.in`
Имя выходного файла:	`ancestor.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 100\,000$) — количество вершин в дереве. Во второй строке находятся n чисел, i-е из которых определяет номер непосредственного родителя вершины с номером i. Если это число равно нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leqslant m \leqslant 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

Формат выходных данных

Для каждого из m запросов выведите на отдельной строке число `1`, если вершина a является одним из предков вершины b, и `0` в противном случае.

Примеры

ancestor.in	ancestor.out
6 0 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

Задача B. Мосты

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

Дан неориентированный граф, не обязательно связный, но не содержащий петель и кратных рёбер. Требуется найти все мосты в нём.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 20\,000$, $1 \leqslant m \leqslant 200\,000$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \leqslant b_i, e_i \leqslant n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество мостов в заданном графе. На следующей строке выведите b целых чисел — номера рёбер, которые являются мостами, в возрастающем порядке. Рёбра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Примеры

bridges.in	bridges.out
6 7 1 2 2 3 3 4 1 3 4 5 4 6 5 6	1 3

Задача C. Точки сочленения

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

Дан неориентированный граф. Требуется найти все точки сочленения в нём.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 20\,000$, $1 \leqslant m \leqslant 200\,000$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \leqslant b_i, e_i \leqslant n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество точек сочленения в заданном графе. На следующей строке выведите b целых чисел — номера вершин, которые являются точками сочленения, в возрастающем порядке.

Примеры

points.in	points.out
6 7 1 2 2 3 2 4 2 5 4 5 1 3 3 6	2 2 3

Задача D. Конденсация графа

Имя входного файла:	`condense2.in`
Имя выходного файла:	`condense2.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Требуется найти количество рёбер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных рёбер и петель.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и рёбер графа соответственно ($n \leqslant 10\,000, m \leqslant 100\,000$). Следующие m строк содержат описание рёбер, по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — началом и концом ребра соответственно ($1 \leqslant b_i, e_i \leqslant n$). В графе могут присутствовать кратные рёбра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество рёбер в конденсации графа.

Примеры

condense2.in	condense2.out
4 4 2 1 3 2 2 3 4 3	2

Задача E. Противопожарная безопасность

Имя входного файла:	`firesafe.in`
Имя выходного файла:	`firesafe.out`
Ограничение по времени:	0.5 секунда
Ограничение по памяти:	256 мегабайт

В Судиславле n домов. Некоторые из них соединены дорогами с односторонним движением.

В последнее время в Судиславле участились случаи пожаров. В связи с этим жители решили построить в посёлке несколько пожарных станций. Но возникла проблема: едущая по вызову пожарная машина, конечно, может игнорировать направление движения текущей дороги, однако возвращающаяся с задания машина обязана следовать правилам дорожного движения (жители Судиславля свято чтут эти правила!).

Ясно, что, где бы ни оказалась пожарная машина, у неё должна быть возможность вернуться на ту пожарную станцию, с которой она выехала. Но строительство станций стоит больших денег, поэтому на совете посёлка было решено построить минимальное количество станций таким образом, чтобы это условие выполнялось. Кроме того, для экономии было решено строить станции в виде пристроек к уже существующим домам.

Ваша задача — написать программу, рассчитывающую оптимальное положение станций.

Формат входных данных

В первой строке входного файла задано число n ($1 \leqslant n \leqslant 3\,000$). Во второй строке записано количество дорог m ($1 \leqslant m \leqslant 100\,000$). Далее следует описание дорог в формате a_i b_i, означающее, что по i-й дороге разрешается движение автотранспорта от дома a_i к дому b_i ($1 \leqslant a_i, b_i \leqslant n$).

Формат выходных данных

В первой строке выведите минимальное количество пожарных станций K, которое необходимо построить. Во второй строке выведите K чисел в произвольном порядке — дома, к которым необходимо пристроить станции. Если оптимальных решений несколько, выведите любое.

Примеры

firesafe.in	firesafe.out
5 7 1 2 2 3 3 1 2 1 2 3 3 4 2 5	2 4 5

Задача 3. Размещение данных

Имя входного файла:	`data.in`
Имя выходного файла:	`data.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Телекоммуникационная сеть крупной IT-компании содержит n серверов, пронумерованных от 1 до n. Некоторые пары серверов соединены двусторонними каналами связи, всего в сети m каналов. Гарантируется, что сеть серверов устроена таким образом, что по каналам связи можно передавать данные с любого сервера на любой другой сервер, возможно с использованием одного или нескольких промежуточных серверов.

Множество серверов A называется *отказоустойчивым*, если при недоступности любого канала связи выполнено следующее условие. Для любого не входящего в это множество сервера X существует способ передать данные по остальным каналам на сервер X хотя бы от одного сервера из множества A.

На рис. 1 показан пример сети и отказоустойчивого множества из серверов с номерами 1 и 4. Данные на сервер 2 можно передать следующим образом. При недоступности канала между серверами 1 и 2 — с сервера 4, при недоступности канала между серверами 2 и 3 — с сервера 1. На серверы 3 и 5 при недоступности любого канала связи можно по другим каналам передать данные с сервера 4.

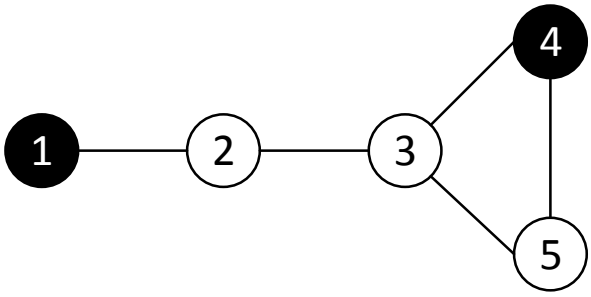

Рис. 1. Пример сети и отказоустойчивого множества серверов.

В рамках проекта группе разработчиков компании необходимо разместить свои данные в сети. Для повышения доступности данных и устойчивости к авариям разработчики хотят продублировать свои данные, разместив их одновременно на нескольких серверах, образующих отказоустойчивое множество. Чтобы минимизировать издержки, необходимо выбрать минимальное по количеству серверов отказоустойчивое множество. Кроме того, чтобы узнать, насколько гибко устроена сеть, необходимо подсчитать количество способов выбора такого множества, и поскольку это количество способов может быть большим, необходимо найти остаток от деления этого количества способов на число $10^9 + 7$.

Требуется написать программу, которая по заданному описанию сети определяет следующие числа: k — минимальное количество серверов в отказоустойчивом множестве серверов, c — остаток от деления количества способов выбора отказоустойчивого множества из k серверов на число $10^9 + 7$

Формат входного файла

Первая строка входного файла содержит целые числа n и m — количество серверов и количество каналов связи соответственно ($2 \le n \le 200\,000$, $1 \le m \le 200\,000$).

Следующие m строк содержат по два целых числа и описывают каналы связи между серверами. Каждый канал связи задается двумя целыми числами: номерами серверов, которые он соединяет.

Гарантируется, что любые два сервера соединены напрямую не более чем одним каналом связи, никакой канал не соединяет сервер сам с собой, и для любой пары серверов существует способ передачи данных с одного из них на другой, возможно с использованием одного или нескольких промежуточных серверов.

Формат выходного файла

Выведите два целых числа, разделенных пробелом: k — минимальное число серверов в отказоустойчивом множестве серверов, c — количество способов выбора отказоустойчивого множества из k серверов, взятое по модулю $10^9 + 7$

Пример входных и выходных файлов

data.in	data.out
5 5 1 2 2 3 3 4 3 5 4 5	2 3

Пояснение к примеру

В приведенном примере отказоустойчивыми являются следующие множества из двух серверов: {1, 3}, {1, 4}, {1, 5}.

Описание подзадач и системы оценивания

Баллы за каждую подзадачу начисляются только, если все тесты для этой подзадачи и всех необходимых подзадач успешно пройдены.

Подзадача	Баллы	Дополнительные ограничения		Необходимые подзадачи
		N	m	
1	25	$2 \le n \le 10$	$1 \le m \le 45$	
2	27	$2 \le n \le 200\,000$	$m = n - 1$	
3	28	$2 \le n \le 1000$	$1 \le m \le 5000$	1
4	21	$2 \le n \le 200\,000$	$1 \le m \le 200\,000$	1, 2, 3

Получение информации о результатах окончательной проверки

По запросу сообщается результат окончательной проверки на каждом тесте.