

Задача А. 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 ($1 \leq n \leq 20\,000$, $1 \leq m \leq 200\,000$).

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

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

<code>bridges.in</code>	<code>bridges.out</code>
6 7	1
1 2	3
2 3	
3 4	
1 3	
4 5	
4 6	
5 6	

Задача В. Конденсация графа

Имя входного файла: `condense2.in`
Имя выходного файла: `condense2.out`
Ограничение по времени: 0.65 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 ($n \leq 10\,000, m \leq 100\,000$). 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 ($1 \leq b_i, e_i \leq n$). 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

<code>condense2.in</code>	<code>condense2.out</code>
4 4 2 1 3 2 2 3 4 3	2

Задача С. 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 целое число n ($1 \leq n \leq 100\,000$) — 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 ($1 \leq m \leq 100\,000$) — количество запросов.

Каждая из следующих m строк содержит два различных числа a и b ($1 \leq a, b \leq n$).

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

<code>ancestor.in</code>	<code>ancestor.out</code>
6	0
0 1 1 2 3 3	1
5	1
4 1	0
1 4	0
3 6	
2 6	
6 5	

Задача D. Горилла и граф

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

Однажды молодая горилла нашла граф, в котором n вершин и m ребер. В вершине i находится a_i бананов, которые горилла может собрать, пройдя через эту вершину. Помогите горилле найти максимальное количество бананов, которое она может собрать, самостоятельно выбрав стартовую и конечную вершину своего маршрута.

Формат входных данных

В первой строке даны два целых числа n ($1 \leq n \leq 10^5$) и m ($1 \leq m \leq 2 \times 10^5$). Во второй строке даны n целых чисел a_i ($1 \leq a_i \leq 10^9$). Затем даны m строк, описывающих граф, в каждой строке даны два целых числа v, u ($1 \leq v, u \leq n$), символизирующие наличие одностороннего ребра из v в u .

Формат выходных данных

В единственной строке выведите одно целое число - ответ на задачу.

Пример

стандартный ввод	стандартный вывод
4 4 4 5 2 7 1 2 2 1 1 3 2 4	16

Задача Е. Сигнализация

Имя входного файла: `alarm.in`
Имя выходного файла: `alarm.out`
Ограничение по времени: 4 секунды
Ограничение по памяти: 512 мегабайт

Подземный бункер состоит из n комнат, соединённых $n - 1$ коридорами. Каждый коридор соединяет две различные комнаты и имеет определённую длину. Бункер устроен таким образом, что из любой комнаты i можно прийти в любую другую комнату j . Заметим, что существует единственный такой путь, не проходящий по одному и тому же коридору дважды. Сумма длин коридоров, составляющих этот путь, называется расстоянием между комнатами i и j и обозначается $\rho(i, j)$.

Каждая комната бункера оборудована звуковой сигнализацией, состоящей из сирены и датчика звука, который её включает. Сирена, включённая в комнате i , активирует датчик звука в каждой комнате, расстояние до которой не превосходит расстояние d_i , определяемое мощностью этой сирены. Другими словами, включение сирены в комнате i автоматически включает сирену во всех комнатах j , таких что $\rho(i, j) \leq d_i$. Эта сирена, в свою очередь, может вызвать автоматическое включение других сирен и так далее.

В случае возникновения чрезвычайной ситуации некоторые сирены необходимо включить вручную, после чего звук от них автоматически включит сирены в других комнатах. Правила безопасности предписывают выбор такого набора сирен для ручного включения, который в конце концов приведёт к автоматическому включению сирен во всех комнатах.

Требуется написать программу, которая определяет минимальное количество сирен в наборе, удовлетворяющем правилам безопасности.

Формат входных данных

Первая строка входных данных содержит единственное число n — количество комнат ($1 \leq n \leq 3000$).

Вторая строка содержит последовательность из n целых чисел d_i , i -е из них равно максимальному расстоянию, на котором расположенная в комнате i сирена активирует датчики ($0 \leq d_i \leq 10^9$).

Последующие $n - 1$ строк описывают коридоры бункера. В i -й из них находятся три целых числа: u_i, v_i, l_i , где u_i, v_i — номера различных комнат, соединённых коридором i , а l_i — длина этого коридора ($1 \leq u_i, v_i \leq n; 1 \leq l_i \leq 10^9$).

Формат выходных данных

Выходные данные должны состоять из единственного числа — минимального количества сирен, которые необходимо включить вручную.

Примеры

<code>alarm.in</code>	<code>alarm.out</code>
10 1 2 2 2 6 3 4 5 4 3 1 2 5 2 3 1 2 4 5 4 5 2 4 6 4 4 7 3 1 8 1 8 9 5 8 10 4	3

Задача F. Размещение данных

Имя входного файла: `data.in`
Имя выходного файла: `data.out`
Ограничение по времени: 2 секунды
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 . В условиях 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.

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 входных данных

Первая строка входного файла содержит целые числа n и m — количество серверов и количество каналов связи соответственно ($2 \leq n \leq 200000$, $1 \leq m \leq 200000$). 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 выходных данных

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 $10^9 + 7$

Примеры

<code>data.in</code>	<code>data.out</code>
5 5 1 2 2 3 3 4 3 5 4 5	2 3

Замечание

В приведенном примере отказоустойчивыми являются следующие множества $\{1, 3\}$, $\{1, 4\}$, $\{1, 5\}$.