

## Задача А. 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  $n$  ( $1 \leq n \leq 100\,000$ ) — 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 В. Поиск цикла

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

Дан ориентированный невзвешенный граф. Необходимо определить есть ли в нём циклы, и если есть, то вывести любой из них.

### Формат входных данных

В первой строке входного файла находятся два натуральных числа  $N$  и  $M$  ( $1 \leq N \leq 100\,000$ ,  $M \leq 100\,000$ ) — количество 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

Если в графе нет цикла, то вывести «NO», иначе — «YES» и затем перечислить все вершины в порядке обхода цикла.

### Примеры

| <code>cycle.in</code> | <code>cycle.out</code> |
|-----------------------|------------------------|
| 2 2<br>1 2<br>2 1     | YES<br>2 1             |
| 2 2<br>1 2<br>1 2     | NO                     |

## Задача С. 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 D. 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 ( $1 \leq n \leq 20\,000$ ,  $1 \leq m \leq 200\,000$ ).

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 точек сочленения в заданном графе. На следующей строке выведите  $b$  целых чисел — номера вершин, которые являются точками сочленения, в возрастающем порядке.

### Примеры

| <code>points.in</code> | <code>points.out</code> |
|------------------------|-------------------------|
| 6 7                    | 2                       |
| 1 2                    | 2 3                     |
| 2 3                    |                         |
| 2 4                    |                         |
| 2 5                    |                         |
| 4 5                    |                         |
| 1 3                    |                         |
| 3 6                    |                         |

## Задача Е. 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 ( $n \leq 10\,000, m \leq 100\,000$ ). 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<br>2 1<br>3 2<br>2 3<br>4 3 | 2                          |

## Задача F. Магнитные подушки

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

Город будущего застроен небоскребами, для передвижения между которыми и парковки транспорта многие тройки небоскребов соединены треугольной подушкой из однополярных магнитов. Каждая подушка соединяет ровно 3 небоскреба и вид сверху на нее представляет собой треугольник, с вершинами в небоскребах. Это позволяет беспрепятственно передвигаться между соответствующими небоскребами. Подушки можно делать на разных уровнях, поэтому один небоскреб может быть соединен различными подушками с парами других, причем два небоскреба могут соединять несколько подушек (как с разными третьими небоскребами, так и с одинаковым). Например, возможны две подушки на разных уровнях между небоскребами 1, 2 и 3, и, кроме того, магнитная подушка между 1, 2, 5.

Система магнитных подушек организована так, что с их помощью можно добраться от одного небоскреба, до любого другого в этом городе (с одной подушки на другую можно перемещаться внутри небоскреба), но поддержание каждой из них требует больших затрат энергии.

Требуется написать программу, которая определит, какие из магнитных подушек нельзя удалять из подушечной системы города, так как удаление даже только этой подушки может привести к тому, что найдутся небоскребы из которых теперь нельзя добраться до некоторых других небоскребов, и жителям станет очень грустно.

### Формат входных данных

В первой строке входного файла находятся числа  $N$  и  $M$  — количество небоскребов в городе и количество работающих магнитных подушек соответственно ( $3 \leq N \leq 100000$ ,  $1 \leq M \leq 100000$ ). В каждой из следующих  $M$  строк через пробел записаны три числа — номера небоскребов, соединенных подушкой. Небоскребы пронумерованы от 1 до  $N$ . Гарантируется, что имеющиеся магнитные подушки позволяют перемещаться от одного небоскреба до любого другого.

### Формат выходных данных

Выведите в выходной файл сначала количество тех магнитных подушек, отключение которых невозможно без нарушения сообщения в городе, а потом их номера. Нумерация должна соответствовать тому порядку, в котором подушки перечислены во входном файле. Нумерация начинается с единицы.

### Примеры

| <code>city.in</code>                    | <code>city.out</code> |
|---|-----------------------|
| 3 1<br>1 2 3                            | 1<br>1                |
| 3 2<br>1 2 3<br>3 2 1                   | 0                     |
| 5 4<br>1 2 3<br>2 4 3<br>1 2 4<br>3 5 1 | 1<br>4                |

## Задача G. Пожарные депо

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

В городе есть  $n$  перекрестков, некоторые из которых соединены односторонними дорогами. Всего в городе  $m$  дорог.

Мэр решил построить несколько пожарных депо. Для этого ему надо выбрать, на каких перекрестках их построить. При этом когда пожарная машина едет на пожар, она может ехать как по направлению движения по дороге, так и против этого направления. А вот при возвращении в депо срочности нет и ехать «по встречке» уже нельзя.

Помогите мэру построить минимальное число депо, чтобы при пожаре дома на любом перекрестке пожарная машина из одного из депо могла доехать до этого перекрестка (возможно против движения), а затем могла вернуться в свое депо (уже двигаясь только по направлению дорог).

### Формат входных данных

В первой строке входного файла находятся два натуральных числа  $n$  и  $m$  ( $1 \leq n \leq 20\,000, 1 \leq m \leq 200\,000$ ) — количество вершин и улиц. Далее в  $m$  строках перечислены улицы, каждая улица задаётся парой чисел — номерами начального и конечного перекрестка.

### Формат выходных данных

Выведите одно число — минимальное число депо, которые необходимо построить.

### Примеры

| <code>firesafe.in</code> | <code>firesafe.out</code> |
|--------------------------|---------------------------|
| 6 7                      | 2                         |
| 1 2                      |                           |
| 2 3                      |                           |
| 3 1                      |                           |
| 1 4                      |                           |
| 1 5                      |                           |
| 4 6                      |                           |
| 6 4                      |                           |

### Замечание

В приведенном примере можно построить депо, например, на перекрестках 4 и 5.

## Задача Н. 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<br>1 2 2 2 6 3 4 5 4 3<br>1 2 5<br>2 3 1<br>2 4 5<br>4 5 2<br>4 6 4<br>4 7 3<br>1 8 1<br>8 9 5<br>8 10 4 | 3                      |