

### Задача А. Предок

Имя входного файла: `ancestor.in`  
Имя выходного файла: `ancestor.out`

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

#### Формат входного файла

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 выходного файла

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
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: `cycle2.in`  
Имя выходного файла: `cycle2.out`

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

#### Формат входного файла

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

#### Формат выходного файла

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

#### Примеры

cycle2.in	cycle2.out
2 2 1 2 2 1	YES 1 2
2 1 1 2	NO

### Задача С. Мосты

Имя входного файла: `bridges.in`  
Имя выходного файла: `bridges.out`

Дан неориентированный граф. Требуется найти все мосты в нём.

#### Формат входного файла

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 выходного файла

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 D. Точки сочленения

Имя входного файла: `points.in`  
Имя выходного файла: `points.out`

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

#### Формат входного файла

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 выходного файла

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	2
1 2	2
2 3	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`

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

### Формат входного файла

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 выходного файла

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

### Примеры

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

### Задача F. Противопожарная безопасность

Имя входного файла: `firesafe.in`

Имя выходного файла: `firesafe.out`

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 входного файла

В первой строке входного файла задано число  $n$  ( $1 \leq n \leq 3\,000$ ). Во второй строке записано количество дорог  $m$  ( $1 \leq m \leq 100\,000$ ). 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  $b_i$  ( $1 \leq a_i, b_i \leq n$ ).

### Формат выходного файла

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	2
7	4 5
1 2	
2 3	
3 1	
2 1	
2 3	
3 4	
2 5	