

## Задача А. Мосты

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
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

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

## Задача В. Точки сочленения

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
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

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

## Задача С. Магнитные подушки

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
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

city.in	city.out
3 1 1 2 3	1 1
3 2 1 2 3 3 2 1	0
5 4 1 2 3 2 4 3 1 2 4 3 5 1	1 4

## Задача D. Компоненты вершинной двусвязности

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 мегабайта

Компонентой вершинной двусвязности графа  $\langle V, E \rangle$  называется максимальный по включению подграф (состоящий из вершин и ребер), такой что любые два ребра из него лежат на вершинно простом цикле.

Дан неориентированный граф без петель. Требуется выделить компоненты вершинной двусвязности в нем.

### Формат входных данных

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

Следующие  $m$  строк содержат описание ребер 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

В первой строке выходного файла выведите целое число  $k$  — количество компонент вершинной двусвязности графа. Во второй строке выведите  $m$  натуральных чисел  $a_1, a_2, \dots, a_m$ , не превосходящих  $k$ , где  $a_i$  — номер компоненты вершинной двусвязности, которой принадлежит  $i$ -е ребро. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

### Примеры

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

## Задача E. Эйлеров путь

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Дан неориентированный связный граф, не более трех вершин имеет нечетную степень. Требуется определить, существует ли в нем путь, проходящий по всем ребрам.

Если такой путь существует, необходимо его вывести.

### Формат входных данных

Первая строка входного файла содержит натуральное число  $n$  — количество вершин графа ( $1 \leq n \leq 100\,000$ ). Далее следуют  $n$  строк, задающих ребра. В  $i$ -й из этих строк находится число  $m_i$  — количество ребер, инцидентных вершине  $i$ . Далее следуют  $m_i$  натуральных чисел — номера вершин, в которые ведут ребро из  $i$ -й вершины.

Граф может содержать кратные ребра, но не содержит петель.

Граф содержит не более 300 000 ребер.

### Формат выходных данных

Если решение существует, то в первую строку выходного файла выведите одно число  $k$  — количество ребер в искомом маршруте, а во вторую  $k + 1$  число — номера вершин в порядке их посещения.

Если решений нет, выведите в выходной файл одно число -1.

Если решений несколько, выведите любое.

### Примеры

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

## Задача F. Таня и пароль

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Пока папа был на работе, маленькая девочка Таня решила поиграть с папиным паролем к секретной базе данных. Папин пароль представляет собой строку, состоящую из  $n + 2$  символов. Она выписала все возможные  $n$  трёхбуквенных подстрок пароля на бумажки, по одной на каждую бумажку, а сам пароль выкинула. Каждая трёхбуквенная подстрока была выписана на бумажки столько раз, сколько она встречалась в пароле. Таким образом, в итоге у Тани оказалось  $n$  бумажек.

Потом Таня поняла, что папа расстроится, если узнает о ее игре, и решила восстановить пароль или, по крайней мере, хотя бы какую-то строку, соответствующую получившемуся набору трёхбуквенных строк. Вам предстоит помочь ей в этой непростой задаче. Известно, что папин пароль состоял из строчных и заглавных букв латинского алфавита, а также из цифр. Строчные и заглавные буквы латинского алфавита считаются различными.

### Формат входных данных

В первой строке следует целое число  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ), количество трёхбуквенных подстрок, которые получились у Тани.

Следующие  $n$  строк каждая содержат по три буквы, образующие подстроку пароля папы. Каждый символ во вводе — строчная или заглавная буква латинского алфавита или цифра.

### Формат выходных данных

Если во время игры Таня что-то напутала, и строк, соответствующих данному набору подстрок, не существует, то выведите «NO».

Если же возможно восстановить строку, соответствующую данному набору подстрок, то выведите «YES», а затем любой подходящий вариант пароля.

### Примеры

password.in	password.out
5 aca aba aba cab bac	YES abacaba
4 abc bCb cb1 b13	NO
7 aaa aaa aaa aaa aaa aaa aaa	YES aaaaaaaaa

## Задача G. 2-SAT

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Формулировка 2-SAT: нужно подобрать значения  $n$  булевых переменных так, чтобы все  $m$  утверждений вида  $x_{i_1} = e_1 \vee x_{i_2} = e_2$  обратились в истину. В данной задаче вам гарантируется, что решение существует.

### Формат входных данных

Входной файл состоит из одного или нескольких тестов.

Каждый тест описывается следующим образом. На первой строке число переменных  $n$  и число утверждений  $m$ . Каждая из следующих  $m$  строк содержит числа  $i_1, e_1, i_2, e_2$ , задает утверждение  $x_{i_1} = e_1 \vee x_{i_2} = e_2$  ( $0 \leq i_j < n$ ,  $0 \leq e_j \leq 1$ ). Ограничения: сумма всех  $n$  не больше 100 000, сумма всех  $m$  не больше 300 000.

### Формат выходных данных

Для каждого теста выведите строку из  $n$  нулей и единиц — значения переменных. Если у данной задачи 2-SAT есть несколько решений, выведите любое.

### Примеры

2sat.in	2sat.out
1 0	0
2 2	01
0 0 1 0	000
0 1 1 1	
3 4	
0 1 1 0	
0 0 2 1	
1 1 2 0	
0 0 0 1	

## Задача Н. Раскраска в три цвета

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Петя нарисовал на бумаге  $n$  кружков и соединил некоторые пары кружков линиями. После этого он раскрасил каждый кружок в один из трех цветов — красный, синий или зеленый.

Теперь Петя хочет изменить их раскраску. А именно — он хочет перекрасить каждый кружок в некоторый другой цвет так, чтобы никакие два кружка одного цвета не были соединены линией. При этом он хочет обязательно перекрасить каждый кружок, а перекрашивать кружок в тот же цвет, в который он был раскрашен исходно, не разрешается.

Помогите Пете решить, в какие цвета следует перекрасить кружки, чтобы выполнялось указанное условие.

### Формат входных данных

Первая строка содержит два целых числа  $n$  и  $m$  — количество кружков и количество линий, которые нарисовал Петя, соответственно ( $1 \leq n \leq 1000$ ,  $0 \leq m \leq 20\,000$ ).

Следующая строка содержит  $n$  символов из множества  $\{‘R’, ‘G’, ‘B’\}$  —  $i$ -й из этих символов означает цвет, в который раскрашен  $i$ -й кружок (‘R’ — красный, ‘G’ — зеленый, ‘B’ — синий).

Следующие  $m$  строк содержат по два целых числа — пары кружков, соединенных отрезками.

### Формат выходных данных

Выведите в выходной файл одну строку, состоящую из  $n$  символов из множества  $\{‘R’, ‘G’, ‘B’\}$  — цвета кружков после перекраски. Если решений несколько, выведите любое.

Если решения не существует, выведите в выходной файл слово “Impossible”.

### Примеры

color.in	color.out
4 5 RRRG 1 3 1 4 3 4 2 4 2 3	GGBR
4 5 RGRR 1 3 1 4 3 4 2 4 2 3	Impossible