Симург

В национальном персидском эпосе Шахнаме есть легенда, повествующая о легендарном персидском герое Зале, который влюблён в Рудабу, принцессу Кабула. Когда Заль попросил руки Рудабы, её отец решил дать жениху задание.

В Персии есть n городов, пронумерованных от 0 до $n-1$, которые соединены m двусторонними дорогами, пронумерованными от 0 до $m-1$. Каждая дорога соединяет пару различных городов. Каждую пару городов соединяет не более одной дороги. Некоторые дороги называются *шахскими* и используются для передвижения членов шахской семьи. Задача Заля состоит в том, чтобы определить, какие дороги являются шахскими.

У Заля есть карта, на которой отмечены все города и дороги в Персии. Он не знает, какие дороги являются шахскими, но он может попросить помощи у Симург, легендарной птицы, покровительствующей Залю. Однако, Симург не хочет просто так сообщить Залю, какие дороги являются шахскими. Вместо этого Симург сообщила, что набор из шахских дорог является *золотым*. Набор дорог является золотым тогда и только тогда, когда:

- он состоит *ровно* из $n-1$ дорог;
- для каждой пары городов возможно добраться от одного из них до другого, используя для перемещения только дороги из набора.

Теперь Заль может задавать Симург вопросы. Каждый вопрос устроен следующим образом:

1. Заль выбирает *золотой* набор дорог;
2. Симург сообщает Залю, сколько из выбранных им дорог являются шахскими.

Ваша программа должна помочь Залю найти, какие дороги являются шахскими, задав Симург не более q вопросов. Проверяющий модуль будет играть роль Симург.

Детали реализации

Вы должны реализовать следующую функцию (метод):

```
int[] find_roads(int n, int[] u, int[] v)
```

- n: количество городов,
- u и v: массивы длины m. Для каждого $0 \leq i \leq m-1$ верно, что $u[i]$ и $v[i]$ это города, соединённые дорогой i.
- Функция должна вернуть массив длины $n-1$, содержащий номера шахских дорог (в

произвольном порядке).

Ваше решение может произвести не более q вызовов следующей функции проверяющего модуля:

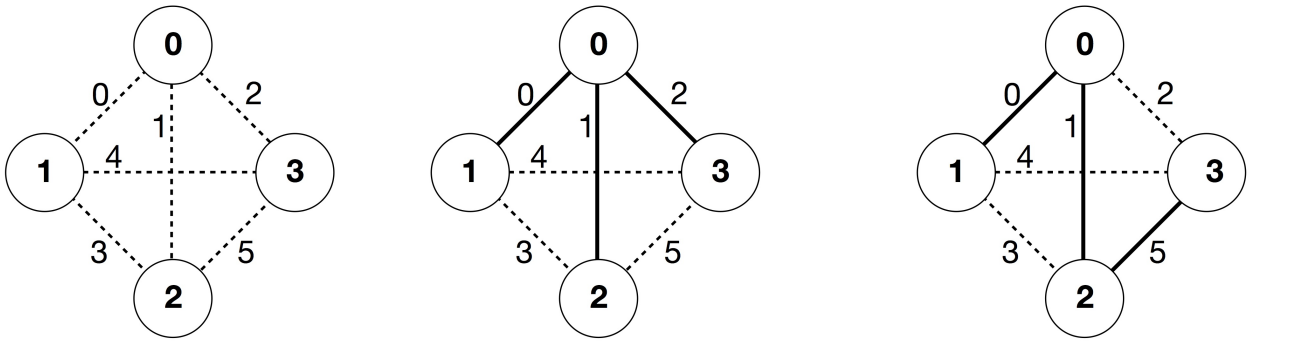
```
int count_common_roads(int[] r)
```

- r: массив длины $n-1$, содержащий номера дорог в золотом наборе дорог (в произвольном порядке).
- Функция возвращает число шахских дорог в массиве r.

Пример

```
find_roads(4, [0, 0, 0, 1, 1, 2], [1, 2, 3, 2, 3, 3])
```

`find_roads( ... )` `count_common_roads([0, 1, 2])` = 2 `count_common_roads([5, 1, 0])` = 3



В данном примере есть 4 города и 6 дорог. Обозначим за (a, b) дорогу, соединяющую города a и b. Дороги пронумерованы от 0 до 5 в следующем порядке: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$ и $(2, 3)$. Каждый набор золотых дорог содержит ровно $n-1$ дорог, то есть 3 дороги.

Предположим, что шахскими являются дороги 0, 1 и 5, то есть, дороги $(0, 1)$, $(0, 2)$ и $(2, 3)$, а программа совершает следующие вызовы:

- `count_common_roads([0, 1, 2])`, который возвращает 2. В данном запросе рассматриваются дороги с номерами 0, 1, и 2, то есть, дороги $(0, 1)$, $(0, 2)$ и $(0, 3)$. Две из них являются шахскими.
- `count_common_roads([5, 1, 0])`, который возвращает 3. В данном запросе рассматриваются все дороги, принадлежащие шахскому набору дорог.

Функция `find_roads` должна вернуть `[5, 1, 0]` или любой другой массив длины 3, который содержит три данных элемента.

Обратите внимание, что следующие вызовы не являются допустимыми.

- `count_common_roads([0, 1])`: длина массива r не равна 3.

- `count_common_roads([0, 1, 3])`: массив r не описывает золотой набор дорог, так как невозможно добраться из города 0 в город 3, используя только дороги $(0, 1)$, $(0, 2)$, $(1, 2)$.

Ограничения

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (для всех $0 \leq i \leq m - 1$)
- Для всех $0 \leq i \leq m - 1$ дорога i соединяет два различных города (то есть $u[i] \neq v[i]$).
- Между каждой парой городов есть не более одной дороги.
- Используя дороги, возможно проехать между каждой парой городов.
- Набор всех шахских дорог является золотым.
- `find_roads` должен вызвать `count_common_roads` не более q раз. В каждом вызове набор дорог, задаваемых массивом r, должен являться золотым.

Система оценивания

1. (13 баллов) $n \leq 7$, $q = 30\,000$
2. (17 баллов) $n \leq 50$, $q = 30\,000$
3. (21 балл) $n \leq 240$, $q = 30\,000$
4. (19 баллов) $q = 12\,000$, а также верно, что между каждой парой городов есть дорога
5. (30 баллов) $q = 8000$

Пример проверяющего модуля

Пример проверяющего модуля читает входные данные в следующем формате:

- Строка 1: n m
- Строка $2 + i$ (для всех $0 \leq i \leq m - 1$): $u[i]$ $v[i]$
- Строка $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

Здесь $s[0], s[1], \ldots, s[n-2]$ обозначают номера шахских дорог.

Пример проверяющего модуля выводит `YES`, если `find_roads` произведёт не более $30\,000$ вызовов `count_common_roads` и вернёт правильный набор шахских дорог. В противном случае, вывод будет `NO`.

Обратите внимание, что функция `count_common_roads` в примере проверяющего модуля не проверяет, обладает ли r свойствами золотого набора дорог. Пример проверяющего модуля только определяет количество шахских дорог в наборе дорог r. Если вы отправите на проверку программу, которая вызовет функцию `count_common_roads` от набора дорог, не являющегося золотым, вердикт будет 'Wrong Answer'.

Техническое замечание

Функция `count_common_roads` в C++ и Pascal использует *передачу аргументов по ссылке* в целях эффективности. Это не влияет на то, как вы должны вызывать процедуру. Гарантируется, что проверяющий модуль не меняет значения в r.