En vänlig, mytisk fågel

Enligt lastgamla persiska legender i ett enormt poetiskt epos så är Zal, den legendariska persiska hjälten, blixtkär i Raduba, prinsessan av Kabul. När Zal bad om Radubas hand gav hennes far Zal en utmaning.

Det finns n städer i Persien, etiketterade 0 till och med $n-1$, och m dubbelriktade vägar, etiketterade från 0 till och med $m-1$. Varje väg sammanbinder ett par av parvis olika städer. Varje par av städer sammanbinds av högst en väg. Vissa av vägarna är kongliga vägar som används för resor av de kongliga. Zals uppgift är att avgöra vilka av vägarna som är kongliga.

Zal har en karta med alla städer och vägar i Persien. Han vet inte vilka vägar som är kongliga, men han kan få hjälp av en vänlig, mytisk fågel, den vänliga, mytiska fågeln som är Zals beskyddare. Den vänliga, mytiska fågeln vill dock inte berätta för Zal vilka de kongliga vägarna är rakt av. Istället berättar hon för Zal att de kongliga vägarna bildar en gyllene mängd. En mängd vägar är en gylleme mängd om och endast om

- den innehåller exakt $n-1$ vägar, och
- för varje par av städer är det möjligt att resa mellan dem genom att enbart resa längs kanter i mängden.

Dessutom kan Zal fråga den vänliga, mytiska fågeln några frågor. För varje fråga så:

1. Väljer Zal en gyllene mängd av vägar
2. Berättar den vänliga, mytiska fågeln för Zal hur många vägar i den valda gyllene vägen som är kongliga.

Ditt program ska hjälpa Zal att hitta alla kongliga vägar genom att fråga den vänliga, mytiska fågeln högst q frågor. Domaren kommer spela den vänliga, mytiska fågelns roll.

Implementation details

You should implement the following procedure:

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

- n: number of cities,
- u and v: arrays of length m. For all $0 \leq i \leq m-1$, $u[i]$ and $v[i]$ are the cities connected by road i.
- This procedure should return an array of length $n-1$ containing the labels of the royal roads (in an arbitrary order).

Your solution can make at most q calls to the following grader procedure:

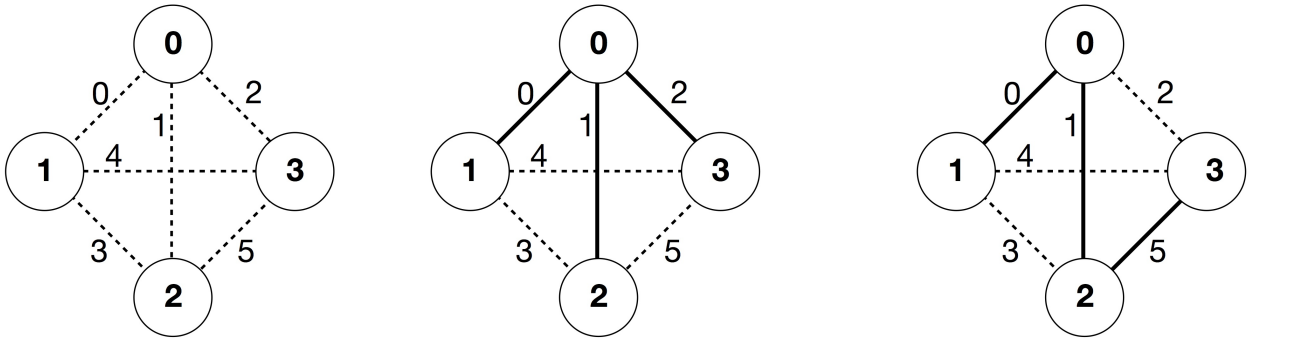
```
int count_common_roads(int[] r)
```

- r: array of length $n-1$ containing the labels of roads in a golden set (in an arbitrary order).
- This procedure returns the number of royal roads in r.

Example

```
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



In this example there are 4 cities and 6 roads. We denote by (a, b) a road connecting cities a and b. The roads are labeled from 0 to 5 in the following order: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, and $(2, 3)$. Every golden set has $n-1 = 3$ roads.

Assume that the royal roads are the roads labeled 0, 1, and 5, that is, the roads $(0, 1)$, $(0, 2)$, and $(2, 3)$. Then:

- `count_common_roads([0, 1, 2])` returns 2. This query is about roads labeled $0, 1$, and 2, that is, roads $(0, 1)$, $(0, 2)$ and $(0, 3)$. Two of them are royal roads.
- `count_common_roads([5, 1, 0])` returns 3. This query is about the set of all royal roads.

The procedure `find_roads` should return `[5, 1, 0]` or any other array of

length 3 that contains these three elements.

Note that the following calls are not allowed:

- `count_common_roads([0, 1])`: here the length of r is not 3.
- `count_common_roads([0, 1, 3])`: here r does not describe a golden set, because it is impossible to travel from city 0 to 3 only using the roads $(0, 1)$, $(0, 2)$, $(1, 2)$.

Constraints

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (for all $0 \leq i \leq m - 1$)
- For all $0 \leq i \leq m - 1$, road i connects two different cities (i.e., $u[i] \neq v[i]$).
- There is at most one road between each pair of cities.
- It is possible to travel between any pair of cities through the roads.
- The set of all royal roads is a golden set.
- `find_roads` should call `count_common_roads` at most q times. In each call, the set of roads specified by r should be a golden set.

Subtasks

1. (13 points) $n \leq 7$, $q = 30\,000$
2. (17 points) $n \leq 50$, $q = 30\,000$
3. (21 points) $n \leq 240$, $q = 30\,000$
4. (19 points) $q = 12\,000$ and there is a road between every pair of cities
5. (30 points) $q = 8000$

Sample grader

The sample grader reads the input in the following format:

- line 1: n m
- line $2 + i$ (for all $0 \leq i \leq m - 1$): $u[i]$ $v[i]$
- line $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

Here, $s[0], s[1], \ldots, s[n-2]$ are the labels of the royal roads.

The sample grader outputs `YES`, if `find_roads` calls `count_common_roads` at most $30\,000$ times, and returns the correct set of royal roads. Otherwise, it outputs `NO`.

Beware that the procedure `count_common_roads` in the sample grader does

not check whether r has all properties of a golden set. Instead, it counts and returns the number of labels of royal roads in the array r. However, if the program you submit calls `count_common_roads` with a set of labels that does not describe a golden set, the grading verdict will be 'Wrong Answer'.

Technical note

The procedure `count_common_roads` in C++ and Pascal uses the pass by reference method for efficiency reasons. You can still call the procedure in the usual way. The grader is guaranteed not to change the value of r.