Simurgh

Zgodnie ze starą perską tradycją w Shahnameh, Zal, legendarny perski bohater, jest nieszczęśliwie zakochany w Rudabie, księżniczce Kabulu. Kiedy Zal poprosił Rudabę o rękę, jej ojciec postawił przed nim zadanie.

Jest n miast w Persji, oznaczonych kolejnymi liczbami naturalnymi od 0 do $n-1$ włącznie, oraz m dwukierunkowych dróg, oznaczonych od 0 do $m-1$ włącznie. Każda droga łączy parę różnych miast. Każda para miast połączona jest co najwyżej jedną drogą. Niektóre z dróg są *królewskimi* drogami używanymi tylko dla podróży królewskich i są one tajne. Zadaniem Zala jest ustalić, które drogi są królewskie.

Zal ma mapę ze wszystkimi miastami i drogami w Persji. Nie wie, które drogi są królewskie, ale może zapytać o pomoc Simurgha, życzliwego mitycznego ptaka, który jest opiekunem Zala. Niestety, Simurgh nie chce ujawnić zbioru dróg królewskich bezpośrednio. Zamiast tego, mówi Zalowi, że zbiór wszystkich dróg królewskich jest *złotym* zbiorem. Zbiór dróg jest złotym zbiorem wtedy i tylko wtedy, gdy:

- ma dokładnie $n-1$ dróg,
- dla każdej pary miast, jest możliwe osiągnięcie jednego z drugiego, podróżując tylko po drogach z tego zbioru.

Dodatkowo, Zal może zadawać Simurghowi pytania. Dla każdego pytania:

1. Zal wybiera *złoty* zbiór dróg,
2. wtedy Simurgh oznajmia Zalowi liczbę królewskich dróg z wybranego zbioru.

Twój program powinien pomóc Zalowi znaleźć zbiór dróg królewskich, zadając Simurghowi co najwyżej q pytań. Sprawdzaczka będzie odgrywać rolę Simurgha.

Szczegóły implementacyjne

Należy zaimplementować następującą funkcję:

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

- n: liczba miast,
- u i v: tablice długości m. Dla każdego $0 \leq i \leq m-1$, $u[i]$ oraz $v[i]$ są miastami połączonymi i-tą drogą.
- Funkcja powinna zwrócić tablicę długości $n-1$ zawierającą oznaczenia wszystkich

królewskich dróg (w dowolnej kolejności).

Twoje rozwiązanie może wywołać co najwyżej q-krotnie następującą funkcję sprawdzaczki:

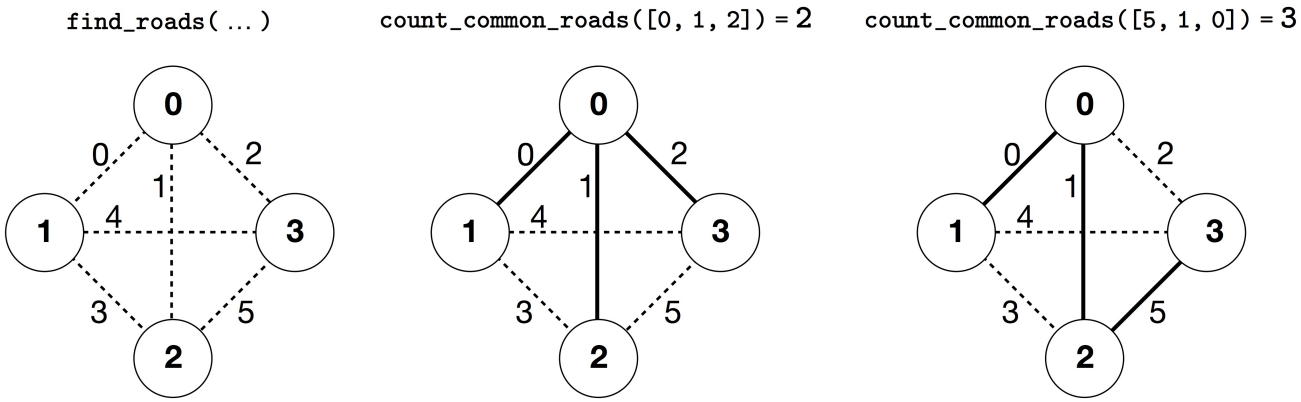
```
int count_common_roads(int[] r)
```

- r: tablica długości $n-1$ zawierająca oznaczenia dróg w złotym zbiorze (w dowolnej kolejności).
- Funkcja zwraca liczbę dróg królewskich wśród dróg ze zbioru r.

Zauważ, że występujące w nazwie funkcji słowo `common` oznacza w tym kontekście *wspólne* drogi zbioru królewskiego oraz r.

Przykład

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



W tym przykładzie są 4 miasta i 6 dróg. Oznaczmy przez (a, b) drogę łączącą miasta a i b. Drogi są numerowane od 0 do 5 w następującej kolejności: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$ i $(2, 3)$. Każdy złoty zbiór składa się z $n - 1 = 3$ dróg.

Załóżmy, że drogi oznaczane 0, 1 oraz 5 (tzn. drogi $(0, 1)$, $(0, 2)$, $(2, 3)$) są królewskie, a program wywoła następujące funkcje:

- `count_common_roads([0, 1, 2])` zwraca 2. To pytanie jest o drogi 0, 1 i 2, to znaczy drogi: $(0, 1)$, $(0, 2)$ oraz $(0, 3)$. Dwie spośród nich są królewskie.
- `count_common_roads([5, 1, 0])` zwraca 3. To pytanie jest o zbiór jedynie dróg królewskich.

Funkcja `find_roads` powinna zwrócić `[5, 1, 0]` lub jakąkolwiek inną tablicę długości 3, która zawiera dokładnie te trzy elementy.

Zauważ, że następujące wywołania nie są dozwolone:

- `count_common_roads([0, 1])`: tutaj długość r nie jest równa 3.

- `count_common_roads([0, 1, 3])`: tutaj r nie opisuje złotego zbioru, ponieważ nie jest możliwe osiągnięcie miasta 3 z miasta 0 jedynie z użyciem dróg $(0, 1)$, $(0, 2)$, $(1, 2)$.

Ograniczenia

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (dla każdego $0 \leq i \leq m - 1$)
- Dla każdego $0 \leq i \leq m - 1$, droga i łączy dwa różne miasta (tzn. $u[i] \neq v[i]$).
- Istnieje co najwyżej jedna droga pomiędzy każdą parą miast.
- Możliwe jest podróżowanie pomiędzy dowolną parą miast z użyciem dróg.
- Zbiór dróg królewskich jest złoty.
- `find_roads` może wywołać funkcję `count_common_roads` co najwyżej q razy. W każdym wywołaniu zbiór dróg r powinien być złotym zbiorem.

Podzadania

1. (13 punktów) $n \leq 7$, $q = 30\,000$
2. (17 punktów) $n \leq 50$, $q = 30\,000$
3. (21 punktów) $n \leq 240$, $q = 30\,000$
4. (19 punktów) $q = 12\,000$ i istnieje droga między każdą parą miast
5. (30 punktów) $q = 8000$

Przykładowa sprawdzaczka

Przykładowa sprawdzaczka odczytuje wejście w następującym formacie:

- wiersz 1: n m
- wiersz $2 + i$ (dla każdego $0 \leq i \leq m - 1$): $u[i]$ $v[i]$
- wiersz $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

W tym przypadku, $s[0], s[1], \ldots, s[n-2]$ są oznaczeniami dróg królewskich.

Przykładowa sprawdzaczka wypisuje `YES` jeśli `find_roads` wywoła `count_common_roads` co najwyżej $30\,000$ razy oraz zwróci właściwy zbiór dróg królewskich. W przeciwnym przypadku, wypisuje `NO`.

Zauważ, że funkcja `count_common_roads` w przykładowej sprawdzaczce nie weryfikuje czy r spełnia wszystkie właściwości złotego zbioru. Zamiast tego, oblicza i zwraca liczbę oznaczeń dróg królewskich wspólnych z oznaczeniami w tablicy r. Natomiast, jeśli nadesłany program wywoła `count_common_roads` ze zbiorem oznaczeń dróg, który nie jest złoty, wynik właściwego sprawdzenia będzie 'Wrong Answer'.

Uwaga techniczna

Funkcja `count_common_roads` w C++ oraz Pascalu używa metody *przekazania wartości przez referencję* ze względów wydajnościowych. Nadal możesz wywołać funkcję w standardowy sposób. Sprawdzaczka nie zmienia wartości r.