Simurgh

Volgens een oude Perzische legende in Shahnameh, is Zal, de legendarische Perzische held, verschrikkelijk verliefd op Rudaba, de prinses van Kabul. Toen Zal Rudaba ten huwelijk vroeg, gaf haar vader hem een uitdaging.

In Perzië zijn n steden, gelabeld van 0 tot en met $n-1$, en m tweerichtingsverkeer wegen, gelabeld van 0 tot en met $m-1$. Elke weg verbindt twee verschillende steden. Elk paar steden is verbonden door maximaal één weg. Sommige wegen zijn *koninklijke wegen* die gebruikt worden door de koninklijke familie. Zal moet bepalen welke wegen koninklijk zijn.

Zal heeft een plattegrond met alle steden en wegen in Perzië. Hij weet niet welke wegen koninklijk zijn, maar Simurgh, de mytische beschermvogel van Zal, kan hem helpen. Simurgh wil niet direct onthullen welke wegen koninklijk zijn. In plaats daarvan vertelt hij Zal dat de set van alle koninklijke wegen een *gouden set* is. Een set van wegen is een gouden set wanneer:

- deze uit *precies* $n-1$ wegen bestaat
- voor elk tweetal steden het mogelijk is om vanuit de ene stad de andere te bereiken langs alleen wegen die in de set zitten.

Daarnaast mag Zal aan Simurgh enkele vragen stellen. Voor elke vraag:

1. kiest Zal een *gouden* set van wegen
2. vertelt Simurgh aan Zal hoeveel wegen in deze gouden set koninklijke wegen zijn.

Jouw programma moet Zal helpen de set van gouden wegen te vinden door Simurgh maximaal q vragen te stellen. De grader speelt de rol van Simurgh.

Implementatiedetails

Implementeer de volgende functie:

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

- n: aantal steden,
- u en v: arrays van lengte m. Voor alle $0 \leq i \leq m-1$, geldt dat steden $u[i]$ en $v[i]$ verbonden zijn door weg i.
- Je functie moet een array van lengte $n-1$ teruggeven met de labels van de koninklijke wegen (in willekeurige volgorde).

Je oplossing mag maximaal q aanroepen doen van de volgende functie van de grader:

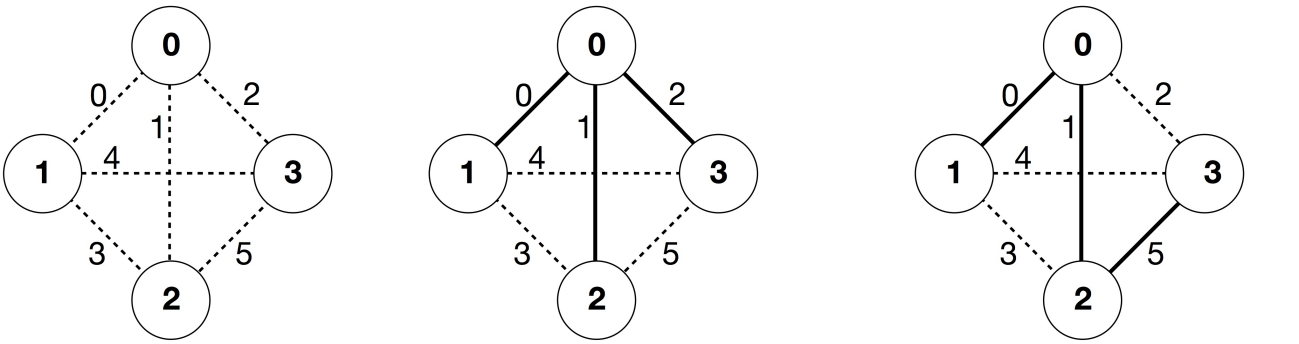
```
int count_common_roads(int[] r)
```

- r: array van lengte $n-1$ met de labels van de wegen die een gouden set vormen (in willekeurige volgorde).
- De functie levert als resultaat hoeveel koninklijke wegen er in r zitten.

Voorbeeld

```
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 dit voorbeeld zijn er 4 steden en 6 wegen. Met (a, b) geven we de weg aan die de steden a en b met elkaar verbindt. De wegen zijn gelabeled van 0 tot en met 5 in de volgende volgorde: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$ en $(2, 3)$. Elke gouden set heeft $n-1=3$ wegen.

Stel dat de koninklijke wegen 0, 1, en 5 zijn, dat zijn de wegen $(0, 1)$, $(0, 2)$ en $(2, 3)$. Je programma kan nu bijvoorbeeld de volgende aanroepen doen:

- `count_common_roads([0, 1, 2])` levert als resultaat 2. Deze vraag gaat over de wegen $0, 1$ en 2, dat zijn de wegen $(0, 1)$, $(0, 2)$ en $(0, 3)$. Twee van deze wegen zijn koninklijk.
- `count_common_roads([5, 1, 0])` levert als resultaat 3. Deze vraag gaat over alle wegen in de koninklijke set.

De functie `find_roads` moet `[5, 1, 0]` als resultaat leveren, of elke andere array van lengte 3 die deze drie elementen bevat.

Merk op dat de volgende aanroepen niet toegestaan zijn:

- `count_common_roads([0, 1])`: de lengte van r is niet 3.
- `count_common_roads([0, 1, 3])`: hier beschrijft r geen gouden set omdat het onmogelijk is om van stad 0 naar stad 3 te gaan over de wegen $(0, 1)$, $(0, 2)$, $(1, 2)$.

Randvoorwaarden

- $2 \leq n \leq 500$
- $n-1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n-1$ (voor alle $0 \leq i \leq m-1$)
- Voor alle $0 \leq i \leq m-1$ geldt dat weg i twee verschillende steden verbindt (dus $u[i] \neq v[i]$).
- Er is maximaal één weg tussen elk tweetal steden.
- Het is mogelijk om tussen elk tweetal steden te reizen over de wegen.
- De set van alle koninklijke wegen is een gouden set.
- `find_roads` mag `count_common_roads` maximaal q maal aanroepen. In elke aanroep moet de set van wegen in r een gouden set zijn.

Deeltaken

1. (13 punten) $n \leq 7$, $q = 30\,000$
2. (17 punten) $n \leq 50$, $q = 30\,000$
3. (21 punten) $n \leq 240$, $q = 30\,000$
4. (19 punten) $q = 12\,000$ en er is een weg tussen elk tweetal steden
5. (30 punten) $q = 8000$

Voorbeeldgrader

De voorbeeldgrader leest de invoer in het volgende formaat:

- regel 1: n m
- regel $2+i$ (voor alle $0 \leq i \leq m-1$): $u[i]$ $v[i]$
- regel $2+m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

In dit geval zijn, $s[0], s[1], \ldots, s[n-2]$ de labels van de koninklijke wegen.

De voorbeeldgrader geeft `YES` als uitvoer indien `find_roads` maximaal $30\,000$ keer `count_common_roads` aanroept en ook de juiste set koninklijke wegen oplevert. Anders geeft het `NO` als uitvoer.

Let op dat de functie `count_common_roads` van de voorbeeldgrader niet controleert of r aan alle eigenschappen van een gouden set voldoet. De grader telt louter het aantal labels van koninklijke wegen in r en geeft dit terug. Als jouw programma `count_common_roads` aanroept met een set van labels die geen gouden set vormen, dan krijg je `Wrong Answer` terug.

Technische aanwijzing

De functie `count_common_roads` in C++ en Pascal gebruikt *pass by reference* om efficiënt te zijn. Je kunt de procedure op de normale manier aanroepen. De grader past de waarde van r gegarandeerd niet aan.