Simurgh

In einer der Legenden des persischen "Königsbuchs" Shahnameh ist der Held Zal unsterblich in Prinzessin Rudaba verliebt. Bevor er Rudaba heiraten darf, muss Zal jedoch zuerst das folgende Rätsel lösen.

In Persien gibt es n Städte (nummeriert von 0 bis $n-1$) und m Straßen (nummeriert von 0 bis $m-1$). Jede Straße verbindet zwei verschiedene Städte miteinander und kann in beiden Richtungen benutzt werden. Jeweils zwei Städte sind durch höchstens eine Straße miteinander verbunden. Einige der Straßen sind *Königsstraßen*, die nur die Königsfamilie benutzen darf. Zal soll herausfinden, welche Straßen Königsstraßen sind.

Zal hat einen Plan der Städte und Straßen. Noch weiß er nicht, welche Straßen die Königsstraßen sind. Doch Simurgh, der weise Märchenvogel, ist Zal wohlgesonnen und wird ihm helfen. Zwar darf Simurgh die Königsstraßen nicht einfach verraten. Er kann aber sagen, dass die Menge der Königsstraßen genau die Eigenschaften einer *goldenen Menge* hat, nämlich:

- Sie besteht aus *genau* $n-1$ Straßen, und
- von jeder Stadt aus ist jede andere Stadt über die Straßen der Menge erreichbar.

Außerdem ist Simurgh bereit, Zal bestimmte Fragen zu beantworten. In jeder Frage muss Zal eine *goldene Menge* von Straßen wählen. Dann verrät Simurgh die Anzahl der Königsstraßen innerhalb dieser goldenen Menge.

Hilf Zal mit einem Programm, die Menge der Königsstraßen zu finden. Dein Programm darf dazu höchstens q Fragen stellen. Der Grader übernimmt Simurghs Part.

Implementierungsdetails

Du sollst folgende Funktion implementieren.

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

- n: Anzahl der Städte,
- u und v: Arrays der Länge m. Für alle $0 \leq i \leq m-1$ sind $u[i]$ und $v[i]$ die Städte, die durch die Straße i verbunden sind.
- Diese Funktion soll ein Array der Länge $n-1$ mit den Nummern der Königsstraßen zurückgeben (in beliebiger Reihenfolge).

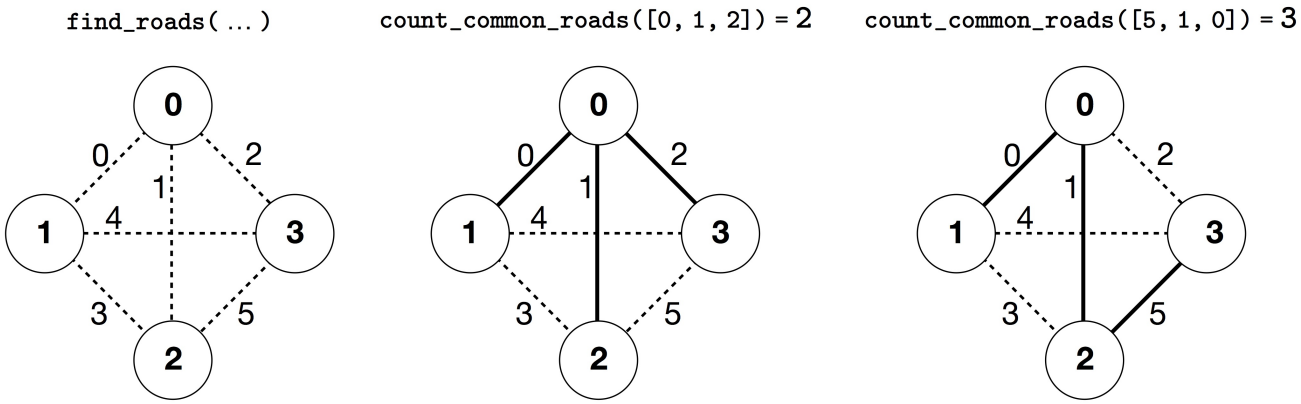
Deine Lösung kann die folgende Graderfunktion höchstens q-mal aufrufen:

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

- r: Array der Länge $n-1$ mit den Nummern der Straßen aus einer goldenen Menge (in beliebiger Reihenfolge).
- Diese Funktion gibt die Anzahl der Königsstraßen in r zurück.

Beispiel

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



In diesem Beispiel gibt es 4 Städte und 6 Straßen. Mit (a, b) benennen wir eine Straße, welche die Städte a und b verbindet. Die Straßen sind nummeriert von 0 bis 5 in der folgenden Reihenfolge: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$ und $(2, 3)$. Jede goldene Menge hat $n - 1 = 3$ Straßen.

Gehe davon aus, dass die Straßen mit den Nummern 0, 1 und 5 – also die Straßen $(0, 1)$, $(0, 2)$ und $(2, 3)$ – die Königsstraßen sind. Das Programm führt folgende Aufrufe aus:

- `count_common_roads([0, 1, 2])` gibt den Wert 2 zurück. Diese Anfrage betrifft die Straßen mit den Nummern $0, 1$ und 2, also die Straßen $(0, 1)$, $(0, 2)$ und $(0, 3)$. Zwei davon sind Königsstraßen.
- `count_common_roads([5, 1, 0])` gibt den Wert 3 zurück. Diese Anfrage betrifft die Menge aller Königsstraßen.

Die Funktion `find_roads` sollte `[5, 1, 0]` zurückgeben oder ein beliebiges anderes Array der Länge 3, das diese drei Elemente enthält.

Beachte, dass die folgenden Aufrufe nicht erlaubt sind:

- `count_common_roads([0, 1])`: Hier ist die Länge von r nicht 3.
- `count_common_roads([0, 1, 3])`: Hier beschreibt r keine goldene Menge, weil es nicht möglich ist, von der Stadt 0 zur Stadt 3 nur über die Straßen $(0, 1)$, $(0, 2)$, $(1, 2)$ zu gelangen.

Beschränkungen

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (für alle $0 \leq i \leq m - 1$)
- Für alle $0 \leq i \leq m - 1$ verbindet die Straße i zwei verschiedene Städte (d.h. $u[i] \neq v[i]$).
- Zwischen jeweils zwei Städten gibt es höchstens eine Straße.
- Von jeder Stadt aus ist jede andere Stadt über Straßen erreichbar.
- Die Menge aller Königsstraßen ist eine goldene Menge.
- `find_roads` sollte `count_common_roads` höchstens q-mal aufrufen. Bei jedem Aufruf sollte die Menge der Straßen aus r eine goldene Menge sein.

Subtasks

1. (13 Punkte) $n \leq 7$, $q = 30\,000$
2. (17 Punkte) $n \leq 50$, $q = 30\,000$
3. (21 Punkte) $n \leq 240$, $q = 30\,000$
4. (19 Punkte) $q = 12\,000$, und es gibt von jeder Stadt zu jeder anderen Stadt eine Straße.
5. (30 Punkte) $q = 8\,000$

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe in folgendem Format:

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

Hier sind $s[0], s[1], \ldots, s[n-2]$ die Nummern der Königsstraßen.

Der Beispiel-Grader gibt `YES` aus, wenn `find_roads` die Funktion `count_common_roads` höchstens $30\,000$ Mal aufruft und die korrekte Menge an Königsstraßen zurückgibt. Andernfalls gibt er `NO` aus.

Beachte, dass die Funktion `count_common_roads` im Beispiel-Grader nicht kontrolliert, ob r alle Eigenschaften einer goldenen Menge hat. Stattdessen zählt er die Anzahl der Nummern von Königsstraßen in dem Array r und gibt diesen Wert zurück.

Das Grading-System wird hingegen "Wrong Answer" ausgeben, wenn r keine goldene Menge ist.

Technischer Hinweis

Die Funktion `count_common_roads` in C++ (bzw. die Prozedur in Pascal) verwendet aus Effizienzgründen *pass by reference*. Du kannst die Funktion aber wie gewohnt aufrufen. Es ist garantiert, dass der Grader den Wert von r nicht verändert.