Simurgh

O legendă veche din Shahnameh spune că Zal, legendarul erou persan s-a îndrăgostit de Rubada, prințesa de Kabul. Când Zal a cerut-o pe Rubada în căsătorie, tatăl ei i-a cerut să îndeplinească mai întâi o misiune.

În Persia sunt n orașe marcate de la 0 la $n-1$, și m drumuri bidirecționale marcate de la 0 la $m-1$. Fiecare drum conectează o pereche de orașe distincte. Fiecare pereche de orașe este conectată cu cel mult un drum. Unele drumuri sunt *drumuri regale* folosite în călătorii de către familia regală. Zal trebuie să determine drumurile care sunt drumuri regale.

Zal are o hartă cu toate orașele și drumurile din Persia. El nu știe care sunt drumurile regale, dar poate apela la ajutorul lui Simurgh, o pasăre binevoitoare mitică - protectoarea lui Zal. Totuși, Simurgh nu vrea să îi dezvăluie lui Zal setul de drumuri regale în mod direct. În schimb, pasărea i-a șoptit lui Zal că setul de drumuri regale formează un *set de aur*. Un set de drumuri este un set de aur, dacă și numai dacă:

- conține *exact* $n-1$ drumuri și
- pentru oricare pereche de orașe, este posibilă deplasarea din unul în celălalt folosind doar drumurile din acest set.

Mai mult, Zal poate adresa lui Simurgh unele întrebări. Pentru fiecare întrebare:

1. Zal selectează un *set de aur*, iar mai apoi
2. Simurgh îi spune lui Zal câte din drumurile setului selectat sunt drumuri regale.

Programul vostru trebuie să-l ajute pe Zal să găsească setul de drumuri regale, adresându-i lui Simurgh cel mult q întrebări. Evaluatorul va juca rolul lui Simurgh.

Detalii de implementare

Trebuie să implementați următoarea procedură:

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

- n: numărul de orașe,
- u și v: șiruri de lungime m. Pentru toate $0 \leq i \leq m-1$, $u[i]$ și $v[i]$ sunt orașele conectate de drumul i.
- Procedura trebuie să returneze un șir de lungime $n-1$ conținând indicii drumurilor regale (în ordine arbitrară).

Soluția voastră poate efectua cel mult q apeluri către următoarea procedură a evaluatorului:

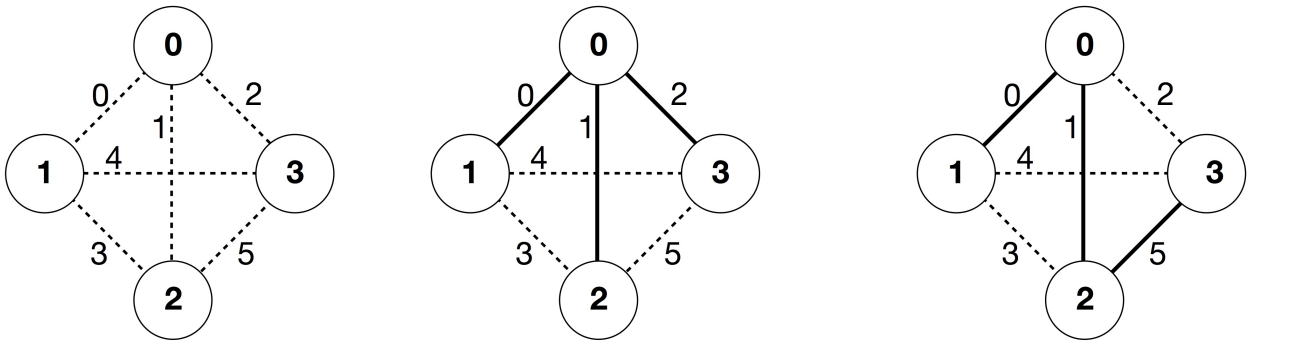
```
int count_common_roads(int[] r)
```

- r: un șir de lungime $n-1$ conținând indicii drumurilor din setul de aur (în ordine arbitrară).
- Această procedură returnează numărul de drumuri regale în r.

Exemplu

```
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`



În acest exemplu sunt 4 orașe și 6 drumuri. Notăm prin (a, b) drumul care conectează orașele a și b. Drumurile sunt numerotate de la 0 la 5 în ordinea următoare: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, și $(2, 3)$. Fiecare set de aur conține $n - 1 = 3$ drumuri.

Fie setul regal format din drumurile cu indicii 0, 1, și 5, adică drumurile care conectează orașele $(0, 1)$, $(0, 2)$, și $(2, 3)$. Atunci:

- `count_common_roads([0, 1, 2])` returnează 2. Această interogare se referă la drumurile cu indicii $0, 1$, și 2, adică, la drumurile care conectează orașele $(0, 1)$, $(0, 2)$ și $(0, 3)$. Două dintre acestea sunt drumuri regale.
- `count_common_roads([5, 1, 0])` returnează 3. Această interogare se referă la setul integral de drumuri regale.

Procedura `find_roads` ar trebui să returneze `[5, 1, 0]` sau oricare alt șir de lungime 3 care va conține aceste trei elemente.

De remarcat, că următoarele apeluri nu sunt permise:

- `count_common_roads([0, 1])`: aici lungimea lui r nu este 3.
- `count_common_roads([0, 1, 3])`: aici r nu descrie un set de aur, deoarece este imposibil de a călători din orașul 0 în 3 doar folosind drumurile $(0, 1)$, $(0, 2)$, $(1, 2)$.

Restricții și precizări

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (pentru toate $0 \leq i \leq m - 1$)
- Pentru toate $0 \leq i \leq m - 1$, drumul i conectează două orașe distincte ($u[i] \neq v[i]$).
- Fiecare pereche de orașe este conectată cu cel mult un drum.
- Cu ajutorul drumurilor date este posibilă deplasarea între oricare două orașe.
- Setul compus din toate drumurile regale este un set de aur.
- `find_roads` trebuie să apeleze `count_common_roads` de cel mult q ori. În fiecare apel, setul de drumuri specificat de r trebuie să fie un set de aur.

Subtask-uri

1. (13 puncte) $n \leq 7$, $q = 30\,000$
2. (17 puncte) $n \leq 50$, $q = 30\,000$
3. (21 puncte) $n \leq 240$, $q = 30\,000$
4. (19 puncte) $q = 12\,000$ și există un drum între fiecare pereche de orașe
5. (30 puncte) $q = 8000$

Evaluator local

Evaluatorul local citește datele din input în următorul format:

- linia 1: n m
- linia $2 + i$ (pentru toate $0 \leq i \leq m - 1$): $u[i]$ $v[i]$
- linia $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

Aici, $s[0], s[1], \ldots, s[n-2]$ sunt indicii drumurilor regale.

Evaluatorul local returnează `YES`, dacă `find_roads` apelează `count_common_roads` cel mult $30\,000$ ori, și returnează setul corect de drumuri regale. În caz contrar, este returnat `NO`.

Atenție! - procedura `count_common_roads` în evaluatorul local nu verifică dacă r posedă toate proprietățile unui set de aur. În schimb, ea numără și returnează numărul de indici a drumurilor regale în șirul r. Cu toate acestea, în cazul în care programul submitat apelează `count_common_roads` cu un set de indici care nu descriu un set de aur, verdictul graderului va fi 'Wrong Answer'.

Observație tehnică

Procedura `count_common_roads` în C++ și Pascal folosește metoda *pass by reference* din considerente de eficiență. Puteți apela procedura în modul obișnuit. Se garantează că evaluatorul nu va modifica valoarea r.