Simurgh

Menurut legenda Persia kuno di Shahnameh, Zal, sang pahlawan legendaris Persia, sangat mencintai Rudaba, sang permaisuri Kabul. Ketika Zal mengajak Rudaba menikah, ayahnya memberikannya sebuah tantangan.

Di Persia terdapat n kota, dinomori dari 0 hingga $n-1$, dan m jalan dua arah, dinomori dari 0 hingga $m-1$. Setiap jalan menghubungkan sepasang kota yang berbeda. Setiap pasang kota dihubungkan oleh paling banyak satu jalan. Beberapa jalan adalah *jalan kerajaan* yang digunakan para raja untuk berkelana. Tugas Zal adalah menentukan jalan-jalan mana saja yang merupakan jalan kerajaan.

Zal memiliki sebuah peta yang mengandung seluruh kota dan jalan di Persia. Ia tidak tahu jalan-jalan mana saja yang merupakan jalan kerajaan, tetapi ia dapat meminta bantuan Simurgh, seekor burung mistis penuh kebajikan yang merupakan pelindung Zal. Namun, Simurgh tidak ingin langsung begitu saja memberi tahu himpunan jalan kerajaan tersebut. Akan tetapi, ia memberi tahu Zal bahwa himpunan semua jalan kerajaan adalah *himpunan keemasan*. Sebuah himpunan jalan adalah sebuah himpunan keemasan jika dan hanya jika:

- terdiri atas *tepat* $n-1$ jalan, dan
- untuk setiap pasang kota, masing-masing dari keduanya dapat dicapai dari yang lain dengan hanya menelusuri jalan-jalan pada himpunan ini.

Zal juga dapat bertanya kepada Simurgh beberapa pertanyaan. Untuk setiap pertanyaan:

1. Zal memilih sebuah himpunan jalan *keemasan*, lalu
2. Simurgh memberi tahu Zal berapa banyak jalan yang terdapat pada himpunan keemasan tersebut yang merupakan jalan kerajaan.

Program Anda harus membantu Zal menentukan himpunan jalan kerajaan dengan bertanya kepada Simurgh paling banyak q pertanyaan. *Grader* akan berperan sebagai Simurgh.

Detil implementasi

Anda harus mengimplementasikan prosedur berikut ini:

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

- n: banyaknya kota,
- u dan v: *array-array* sepanjang m. Untuk setiap $0 \leq i \leq m-1$, $u[i]$ dan $v[i]$ adalah

sepasang kota yang dihubungkan oleh jalan i.
- Prosedur ini harus mengembalikan sebuah *array* sepanjang $n - 1$ yang berisi nomor-nomor jalan kerajaan (dalam urutan mana saja).

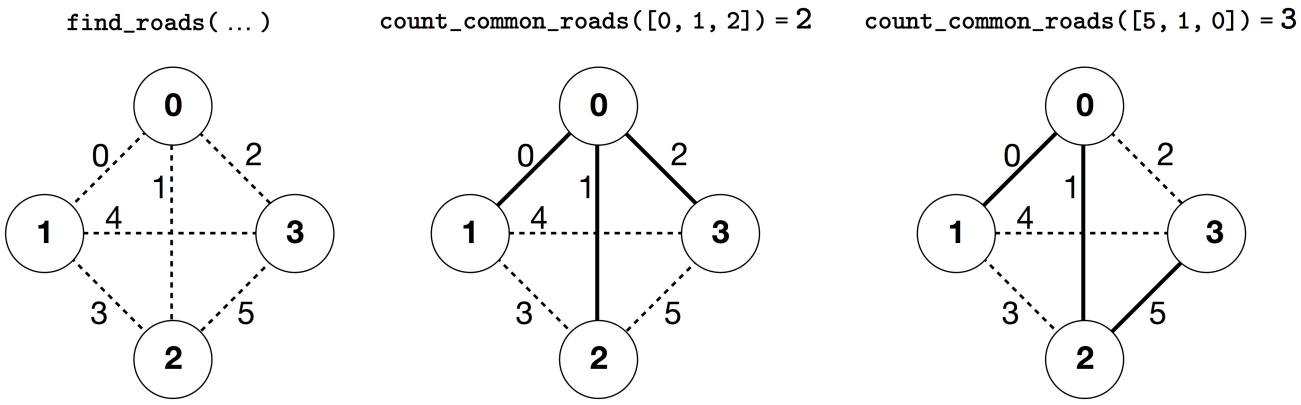
Solusi Anda dapat memanggil prosedur *grader* berikut ini paling banyak q kali:

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

- r: *array* sepanjang $n - 1$ yang berisi nomor-nomor jalan pada sebuah himpunan keemasan (dalam urutan mana saja).
- Prosedur ini mengembalikan banyaknya jalan kerajaan pada r.

Contoh

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



Pada contoh ini, terdapat 4 kota dan 6 jalan. Di sini, (a, b) menyatakan sebuah jalan yang menghubungkan kota-kota a dan b. Jalan-jalan dinomori dari 0 hingga 5 dalam urutan berikut ini: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, dan $(2, 3)$. Setiap himpunan keemasan memiliki $n - 1 = 3$ jalan.

Anggap bahwa jalan-jalan kerajaan adalah jalan-jalan bernomor 0, 1, dan 5; yakni, jalan-jalan $(0, 1)$, $(0, 2)$, dan $(2, 3)$. Kemudian:

- `count_common_roads([0, 1, 2])` mengembalikan 2. Pertanyaan ini mengenai jalan-jalan bernomor $0, 1$, dan 2; yakni, jalan-jalan $(0, 1)$, $(0, 2)$ dan $(0, 3)$. Dua di antaranya adalah jalan-jalan kerajaan.
- `count_common_roads([5, 1, 0])` mengembalikan 3. Pertanyaan ini mengenai himpunan seluruh jalan kerajaan.

Prosedur `find_roads` harus mengembalikan `[5, 1, 0]` atau *array* lain apa saja sepanjang 3 yang berisi ketiga elemen ini.

Perhatikan bahwa pemanggilan-pemanggilan berikut ini tidak diperbolehkan:

- `count_common_roads([0, 1])`: di sini, panjang r bukan 3.
- `count_common_roads([0, 1, 3])`: di sini, r tidak mendeskripsikan sebuah himpunan keemasan, karena kota 3 tidak mungkin dicapai dari kota 0 dengan hanya menelusuri jalan-jalan $(0, 1)$, $(0, 2)$, $(1, 2)$.

Batasan

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (untuk setiap $0 \leq i \leq m - 1$)
- Untuk setiap $0 \leq i \leq m - 1$, jalan i menghubungkan dua kota yang berbeda (yakni, $u[i] \neq v[i]$).
- Terdapat paling banyak satu jalan yang menghubungkan setiap pasang kota.
- Setiap kota dapat dicapai dari semua kota lainnya dengan menelusuri jalan-jalan yang ada.
- Himpunan seluruh jalan kerajaan adalah himpunan keemasan.
- `find_roads` harus memanggil `count_common_roads` paling banyak q kali. Pada setiap pemanggilan, himpunan jalan yang dinyatakan oleh r harus merupakan sebuah himpunan keemasan.

Subsoal

1. (13 poin) $n \leq 7$, $q = 30\,000$
2. (17 poin) $n \leq 50$, $q = 30\,000$
3. (21 poin) $n \leq 240$, $q = 30\,000$
4. (19 poin) $q = 12\,000$ dan terdapat sebuah jalan yang menghubungkan setiap pasang kota
5. (30 poin) $q = 8000$

Grader contoh

Grader contoh membaca masukan dengan format berikut ini:

- baris 1: n m
- baris $2 + i$ (untuk setiap $0 \leq i \leq m - 1$): $u[i]$ $v[i]$
- baris $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

Di sini, $s[0], s[1], \ldots, s[n-2]$ adalah nomor-nomor jalan kerajaan.

Grader contoh mencetak `YES`, jika `find_roads` memanggil `count_common_roads` paling banyak $30\,000$ kali, dan mengembalikan himpunan jalan kerajaan yang benar. Jika tidak, *grader* contoh mencetak `NO`.

Berhati-hatilah karena prosedur `count_common_roads` pada *grader* contoh tidak memeriksa apakah r memenuhi seluruh syarat himpunan keemasan. Akan tetapi, prosedur tersebut menghitung dan mengembalikan banyaknya nomor jalan kerajaan pada *array* r. Namun, jika

program yang Anda kumpulkan memanggil `count_common_roads` dengan sebuah himpunan nomor yang tidak mendeskripsikan sebuah himpunan keemasan, *verdict grading*-nya akan berupa 'Wrong Answer'.

Catatan teknis

Prosedur `count_common_roads` pada C++ dan Pascal menggunakan metode *pass by reference* untuk alasan efisiensi. Anda masih dapat memanggil prosedur tersebut seperti biasanya. *Grader* dijamin tidak mengubah nilai r.