



## Simurgh

De acordo com lendas Persas antigas em Shahnameh, Zal, o lendário herói Persa, está perdidamente apaixonado pela Rudaba, a princesa de Kabul. Quando Zal pediu a mão da Rudaba em casamento, o seu pai deu-lhe um desafio.

Na Pérsia existem  $n$  cidades, numeradas de 0 a  $n - 1$ , e  $m$  estradas bidirecionais, numeradas de 0 a  $m - 1$ . Cada estrada liga um par de cidades distintas. Cada par de cidades é ligado no máximo por uma estrada. Algumas das estradas são *estradas da realeza* utilizadas para viagens da corte real. A tarefa de Zal é determinar quais das estradas são estradas da realeza.

Zal tem um mapa com todas as cidades e estradas da Pérsia. Ele não sabe quais das estradas são da realeza, mas tem a ajuda de Simurgh, o benevolente pássaro mítico que é o protetor de Zal. Porém, Simurgh não quer revelar o conjunto de estradas da realeza diretamente. Em vez disso, ele diz a Zal que o conjunto de todas as estradas da realeza formam um *conjunto de ouro*. Um conjunto é de ouro se e somente se:

- tem *exatamente*  $n - 1$  estradas, e
- para cada par de cidades, é possível ir de uma para a outra utilizando apenas as estradas deste conjunto.

Para além disso, Zal pode perguntar a Simurgh algumas questões. Para cada questão:

1. Zal escolhe um conjunto *de ouro* de estradas, e depois
2. Simurgh diz a Zal quantas das estradas incluídas no conjunto de ouro são estradas da realeza.

O seu programa deve ajudar Zal a encontrar o conjunto de estradas da realeza ao perguntar a Simurgh no máximo  $q$  questões. O corretor fará o papel de Simurgh.

## Detalhes de implementação

Deve implementar a função seguinte:

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

- $n$ : número de cidades,
- $u$  e  $v$ : vetores de comprimento  $m$ . Para todo  $0 \leq i \leq m - 1$ ,  $u[i]$  e  $v[i]$  são cidades ligadas pela estrada  $i$ .
- Esta função deve retornar um vetor de comprimento  $n - 1$  contendo os números que

representam cada estrada da realeza (por ordem arbitrária).

A sua solução pode fazer no máximo  $q$  chamadas à seguinte função do corretor:

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

- $r$ : vetor de comprimento  $n - 1$  contendo os números das estradas num conjunto de ouro (numa ordem arbitrária).
- Esta função retorna o número de estradas da realeza em  $r$ .

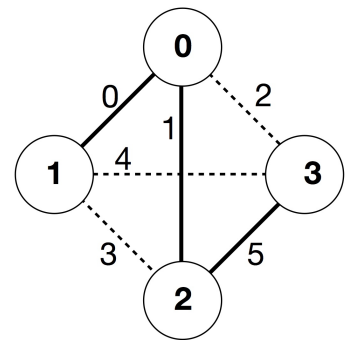
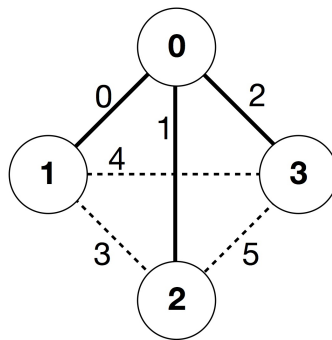
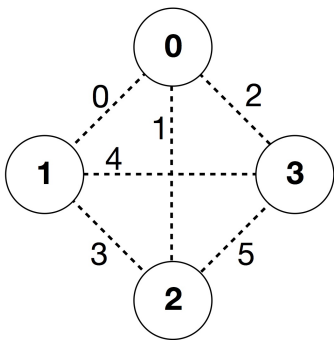
## Exemplo

```
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`



Neste exemplo há 4 cidades e 6 estradas. Denotamos por  $(a, b)$  uma estrada conectando as cidades  $a$  e  $b$ . As estradas são numeradas de 0 a 5 na seguinte ordem:  $(0, 1)$ ,  $(0, 2)$ ,  $(0, 3)$ ,  $(1, 2)$ ,  $(1, 3)$ , e  $(2, 3)$ . Todos os conjuntos de ouro têm  $n - 1 = 3$  estradas.

Assuma que as estradas da realeza são as estradas numeradas por 0, 1 e 5, ou seja, as estradas  $(0, 1)$ ,  $(0, 2)$  e  $(2, 3)$ . Neste caso:

- `count_common_roads([0, 1, 2])` retorna 2. Esta pergunta é relativa às estradas numeradas 0, 1, e 2, ou seja, às estradas  $(0, 1)$ ,  $(0, 2)$  e  $(0, 3)$ . Duas delas são estradas da realeza.
- `count_common_roads([5, 1, 0])` retorna 3. Esta pergunta é relativa ao conjunto de todas as estradas da realeza.

A função `find_roads` deve retornar `[5, 1, 0]` ou qualquer outro vetor de comprimento 3 que contenha estes três elementos.

Note que as seguintes chamadas não são permitidas:

- `count_common_roads([0, 1])`: aqui o comprimento de  $r$  não é 3.
- `count_common_roads([0, 1, 3])`: aqui  $r$  não descreve um conjunto de ouro porque é

impossível chegar da cidade 0 à 3 usando apenas as estradas (0, 1), (0, 2), (1, 2).

## Restrições

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n - 1)/2$
- $0 \leq u[i], v[i] \leq n - 1$  (para todo  $0 \leq i \leq m - 1$ )
- Para todo  $0 \leq i \leq m - 1$ , a estrada  $i$  liga duas cidades diferentes (i.e.,  $u[i] \neq v[i]$ ).
- Existe no máximo uma estrada entre cada par de cidades.
- É possível viajar entre qualquer par de cidades usando as estradas.
- O conjunto de todas as estradas da realza é um conjunto de ouro.
- `find_roads` deve chamar `count_common_roads` no máximo  $q$  vezes. Em cada chamada, o conjunto de estradas especificado por  $r$  deve ser um conjunto de ouro.

## Subtarefas

1. (13 pontos)  $n \leq 7, q = 30\,000$
2. (17 pontos)  $n \leq 50, q = 30\,000$
3. (21 pontos)  $n \leq 240, q = 30\,000$
4. (19 pontos)  $q = 12\,000$  e há uma estrada entre cada par de cidades
5. (30 pontos)  $q = 8000$

## Corretor de exemplo

O corretor de exemplo lê a entrada no seguinte formato:

- linha 1:  $n\ m$
- linha  $2 + i$  (for all  $0 \leq i \leq m - 1$ ):  $u[i]\ v[i]$
- linha  $2 + m$ :  $s[0]\ s[1]\ \dots\ s[n - 2]$

Aqui,  $s[0], s[1], \dots, s[n - 2]$  são os números das estradas da realza.

O corretor de exemplo escreve YES se `find_roads` chamar `count_common_roads` no máximo 30 000 vezes, e retorna o conjunto de estradas da realza correto. Caso contrário, escreve NO.

Atenção que a função `count_common_roads` no corretor de exemplo não verifica se  $r$  tem todas as propriedades de um conjunto de ouro. Ele apenas conta e retorna o número de estradas da realza no vetor  $r$ . Porém, se o programa submetido chamar `count_common_roads` com um conjunto de números que não descreve um conjunto de ouro, o veredicto do corretor será Wrong Answer.

## Nota técnica

A função `count_common_roads` em C++ e Pascal usa o método de *pass by reference* (passagem

por referência) por razões de eficiência. É possível chamar a função da forma usual. É garantido que o corretor não altera o valor de  $r$ .