Simurgh

叙事詩「シャー・ナーメ」によれば, 古代ペルシアの英雄 Zal は, カブールの王女 Rudaba を溺愛したという. Zal が Rudaba にプロポーズすると, Rudaba の父は Zal に試練を与えた.

ペルシアには順に 0 から $n-1$ まで番号づけられた n 個の都市と, 順に 0 から $m-1$ まで番号づけられた m 個の両方向に通行可能な道路がある. 各々の道路は異なる都市を結んでおり, どの 2 つの都市の間にも 2 本以上の道路はない. いくつかの道路は **王立道路** であり, 王族の通行に使われる. どの道路が王立道路であるかは秘匿されている. Zal に与えられた試練は, すべての王立道路を特定することである.

Zal はペルシアのすべての都市とすべての道路の記載された地図を持っている. Zal はどの道路が王立道路であるかを知らないが, 心優しい伝説の鳥, Zal の守護鳥 Simurgh の助けを借りることができる. Simurgh は王立道路の集合を直接明かさない代わりに, 王立道路全体が **黄金の道路網** を成すということを Zal に伝えた.

道路からなる集合は, 以下の条件を満たすとき, またそのときに限り, 黄金の道路網と呼ばれる.

- 集合は, ちょうど $n-1$ 本の道路からなる.
- すべての 2 都市間を, その集合に属する道路のみを用いて行き来することができる.

Zal は Simurgh に質問をすることができる. 各々の質問に対し,

1. Zal は黄金の道路網を指定し,
2. Simurgh は Zal に, Zal が指定した黄金の道路網に含まれる王立道路の本数を答える.

Simurgh に最大 q 回の質問をすることで, すべての王立道路を特定するプログラムを書いて欲しい. なお, 採点プログラムが Simurgh の役割を果たす.

実装の詳細

あなたは, 以下のプロシージャを実装する必要がある.

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

- n: 都市の個数である.
- u と v: 長さ m の配列である. すべての $0 \leqq i \leqq m-1$ に対し, 道路 i は 都市 $u[i]$ と $v[i]$ を結んでいる.
- このプロシージャは, 王立道路の番号すべてを, 任意の順番で並べた長さ $n-1$ の配列を返さなければならない.

あなたのプログラムは, 以下のプロシージャを最大で q 回まで呼び出すことができる.

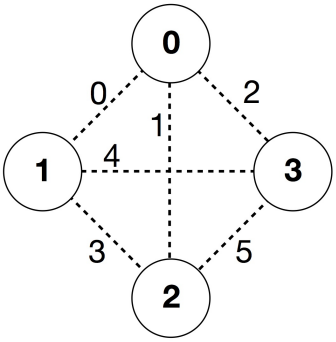
```
int count_common_roads(int[] r)
```

- r: 黄金の道路網を成す道路の番号を, 任意の順番で並べた $n-1$ の配列である.
- このプロシージャは, r に属する王立道路の本数を返す.

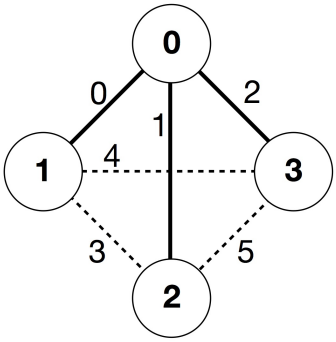
入出力例

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

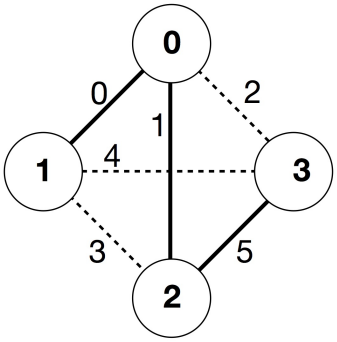
find_roads(...)



count_common_roads([0, 1, 2]) = 2



count_common_roads([5, 1, 0]) = 3



この入力例には, 4 個の都市と 6 本の道路がある. (a, b) で都市 a と b を結ぶ道路を表す. 道路は, $(0,1), (0,2), (0,3), (1,2), (1,3), (2,3)$ の順に 0 から 5 まで番号づけられている. すべての黄金の道路網は, $n-1=3$ 本の道路からなる.

道路 $0, 1, 5$, すなわち道路 $(0,1), (0,2), (2,3)$ が王立道路であり, プログラムが以下のような呼び出しを行ったとする.

- `count_common_roads([0, 1, 2])` は 2 を返す. 番号 $0, 1, 2$ の道路, すなわち道路 $(0,1), (0,2), (0,3)$ のうち, 2 本の道路が王立道路である.
- `count_common_roads([5, 1, 0])` は 3 を返す. 3 本すべての道路が王立道路である.

プロシージャ `find_roads` は, $[5, 1, 0]$ またはその並べ替えを返す必要がある.

以下の呼び出しは不正であることに注意せよ.

- `count_common_roads([0, 1])`: r の長さが 3 でない.
- `count_common_roads([0, 1, 3])`: 道路 $(0,1), (0,2), (1,2)$ のみを用いて都市 0 から 都市 3 へ行くことは不可能なため, r は黄金の道路網を表さない.

制約

- $2 \leqq n \leqq 500$.
- $n-1 \leqq m \leqq n(n-1)/2$.
- すべての $0 \leqq i \leqq m-1$ に対し, 道路 i は異なる 2 都市間を結ぶ. すなわち, $u[i] \neq v[i]$ である.

- どの 2 つの都市の間にも 2 本以上の道はない.
- すべての 2 都市間を, 道路を通って行き来することができる.
- 王立道路全体は, 黄金の道路網を成す.
- `find_roads` は `count_common_roads` を最大で q 回まで呼び出すことができる. 各々の呼び出しにおいて, r は黄金の道路網を表さなければならない.

小課題

1. (13 点) $n \leqq 7, q = 30\,000$.
2. (17 点) $n \leqq 50, q = 30\,000$.
3. (21 点) $n \leqq 240, q = 30\,000$.
4. (19 点) $q = 12000$ であり, どの 2 つの都市の間にも道路が存在する.
5. (30 点) $8000 \leqq q \leqq 30\,000$.

採点プログラムのサンプル

採点プログラムのサンプルは以下の書式で入力を読み込む.

- 1 行目: n m
- $2 + i$ 行目 ($0 \leqq i \leqq m - 1$): $u[i]$ $v[i]$
- $2 + m$ 行目: $s[0]$ $s[1]$ ... $s[n - 2]$

$s[0], s[1], ..., s[n - 2]$ は王立道路の番号を表す.

`find_roads` が `count_common_roads` の $30\,000$ 回以下の呼び出しによって王立道路の集合を正しく出力したとき, 採点プログラムのサンプルは `YES` を返す. そうでない場合, `NO` を返す.

採点プログラムのサンプルの `count_common_roads` プロシージャは, r が黄金の道路網を表すかどうかを確かめないことに注意せよ. このプロシージャは, 配列 r に含まれる王立道路の本数を返す. もちろん, 提出されたプログラムが `count_common_roads` を黄金の道路網を表さないような引数で呼び出したならば, 採点結果は `Wrong Answer` となる.

技術情報

C++ または Pascal において, 高速化のために `count_common_roads` プロシージャは **参照渡し** で実装されている. これについて, 特に留意する必要はない. 採点プログラムは r の値を変更しないことが保障されている.