Симург

Според старата персиска легенда во Шахнамех, Зал, легендарен персиски херој, е лудо вљубен во принцезата Рудаба. Кога Зал ја запросил, нејзиниот татко му задал предизвик.

Во Персија има n градови, означени со броевите од 0 до $n-1$, и m двонасочни улици обележани од 0 до $m-1$. Секоја улица поврзува пар од различни градови. Секој пар од градови е поврзан со најмногу една улица. Некои од улиците се *кралски улици* кои се користат за патување на кралевите. Задачата на Зал е да определи кои од улиците се кралски улици.

Зал има мапа со сите градови и улици во Персија. Тој не знае кои од улиците се кралски, ама може да добие помош од Симург, добронамерната мистична птица која е заштитник на Зал. Но, Симург не сака да му го открие множеството на кралски улици директно. Наместо тоа тој му кажал на Зал дека множеството од кралски улици е *златно множество*.

Едно множество од улици е златно множество ако и само ако:

- тоа има точно $n-1$ улици, и
- за секој пар на градови можно е да се стигне од едниот до другиот со движење само по улиците од ова множество.

Дополнително, Зал може да му постави на Симург прашања. За секое прашање:

1. Зал избира *златно* множество на улици, и потоа
2. Симург му кажува на Зал колку од улиците од одбраното златно множество се кралски улици.

Вашата програма треба да му помогне на Зал да го најде множеството од кралски улици поставувајќи му на Симург најмногу q прашања. Оценувачот ќе ја има улогата на Симург.

Детали за имплементација

Треба да ја имплементирате следнава процедура:

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

- n: број на градови,
- u и v: низи со должина m. За секое $0 \leq i \leq m-1$, $u[i]$ и $v[i]$ се градовите поврзани со улицата i.
- Оваа процедура треба да врати низа со должина $n-1$ која ги содржи ознаките на кралските улици (во произволен редослед).

Вашето решение може да направи најмногу q повици на следната процедура на оценувачот:

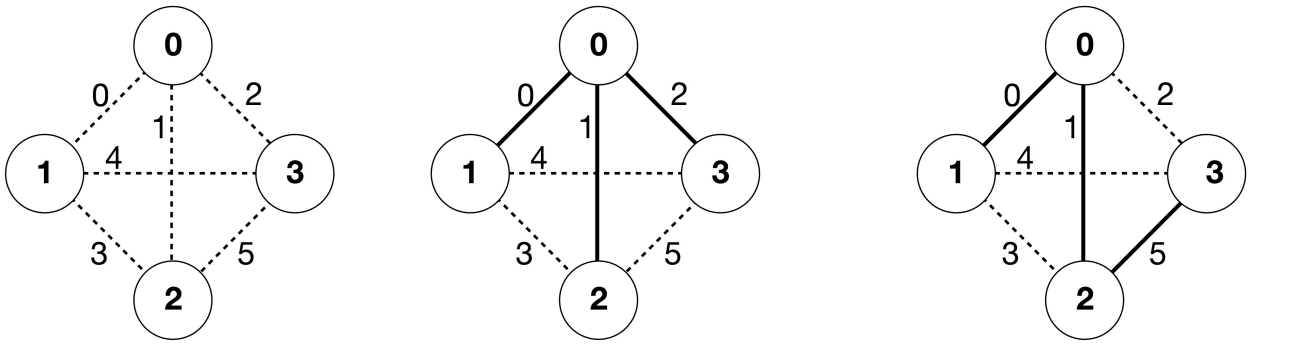
```
int count_common_roads(int[] r)
```

- r: низа со должина $n-1$ која ги содржи ознаките на улиците во златното множество (во произволен редослед).
- Оваа процедура го враќа бројот на кралски улици во r.

Пример

```
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



Во примерот има 4 градови и 6 улици. Со (a, b) ја обележуваме улицата која ги поврзува градовите a и b. Улиците се означени со броевите од 0 до 5 по следниов редослед: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, и $(2, 3)$. Секое златно множество има $n - 1 = 3$ улици.

Да претпоставиме дека кралските улици се означени со броевите 0, 1 и 5, т.е. улиците $(0, 1)$, $(0, 2)$, и $(2, 3)$. Тогаш:

- `count_common_roads([0, 1, 2])` враќа 2. Ова прашање се однесува на улиците означени со $0, 1$, и 2, а тоа се улиците $(0, 1)$, $(0, 2)$ и $(0, 3)$. Две од нив се кралски улици.
- `count_common_roads([5, 1, 0])` враќа 3. Ова прашање се однесува на множеството од сите кралски улици.

Процедурата `find_roads` треба да врати `[5, 1, 0]` или која било друга низа со должина 3 која ги содржи овие три елементи.

Забележете дека следниве повици не се дозволени:

- `count_common_roads([0, 1])`: должината на низата r не е 3.
- `count_common_roads([0, 1, 3])`: овде r не се однесува на златно множество, бидејќи од градот 0 до градот 3 не може да се стигне користејќи ги единствено улиците $(0, 1)$, $(0, 2)$, $(1, 2)$.

Ограничувања

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (за секое $0 \leq i \leq m - 1$)
- За секое $0 \leq i \leq m - 1$, улицата i поврзува два различни града (т.е., $u[i] \neq v[i]$).
- Постои најмногу една улица помеѓу секој пар од градови.
- Возможно е да се патува помеѓу секој пар од градови користејќи ги улиците.
- Множеството од сите кралски улици е златно множество.
- Процедурата `find_roads` треба да ја повика `count_common_roads` најмногу q пати. Во секој повик, множеството од улици специфицирано со r треба да биде златно множество.

Подзадачи

1. (13 поени) $n \leq 7$, $q = 30\,000$
2. (17 поени) $n \leq 50$, $q = 30\,000$
3. (21 поени) $n \leq 240$, $q = 30\,000$
4. (19 поени) $q = 12\,000$ и постои улица помеѓу секој пар од градови
5. (30 поени) $q = 8000$

Пример оценувач

Пример оценувачот го чита влезот во следниот формат:

- линија 1: n m
- линија $2 + i$ (за секое $0 \leq i \leq m - 1$): $u[i]$ $v[i]$
- линија $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

Овде, $s[0], s[1], \ldots, s[n-2]$ се ознаките на кралските улици.

Пример оценувачот печати `YES`, ако `find_roads` ја повика `count_common_roads` најмногу $30\,000$ пати, и притоа го врати точното множество од кралски улици. Инаку, тој печати `NO`.

Обрнете внимание на фактот дека процедурата `count_common_roads` во пример оценувачот не проверува дали r ги има сите особини на златно множество. Наместо тоа, таа го брои и враќа бројот на ознаки на кралски улици во низата r. Како и да е, ако програмата што ќе ја прикачите ја повика процедурата `count_common_roads` со множество од ознаки што не опишува златно множество, оценката што ќе ја добиете ќе биде 'Wrong Answer'.

Техничка забелешка

Процедурата `count_common_roads` во C++ и Pascal користи „*пренос по референца*“ за да се добие на ефикасност. Вие сепак можете да ја повикате процедурата на вообичаениот начин. Се гарантира дека оценувачот нема да ја промени вредноста на r.