Симурх

Съгласно древните Персийски легенди в поемата Шахнаме, легендарният персийски герой Зал е лудо влюбен в принцесата на Кабул Рудаба. Когато Зал поискал ръката на Рудаба, нейният баща му поставил следната задача. В Персия съществуват n града, номерирани с числата от 0 до $n-1$, и m двупосочни преки пътища, номерирани с числата от 0 до $m-1$. Всеки пряк път свързва два различни града. Всяка двойка градове е свързана най-много с един пряк път. Някои от преките пътища са *царски пътища*, използвани за пътуванията на царете, и, естествено, те са секретни. Задачата на Зал е да определи кои от преките пътища са царски. Зал разполага с карта, съдържаща всички градове и преки пътища в Персия. Той не знае кои от преките пътища са царски, но може да получи помощ от великодушната митична птица Симурх, която е негов покровител. Все пак, Симурх не желае направо да разкрие множеството от царски пътища. Вместо това, тя казва на Зал, че множеството от царски преки пътища е *златно множество*. Едно множество от преки пътища е златно тогава и само тогава, когато:

- то съдържа *точно* $n-1$ преки пътища и
- за всяка двойка градове е възможно да се достигне от единия до другия, използвайки само преки пътища от това множество.

Освен това, Зал може да задава определени въпроси на Симурх. При всеки въпрос:

- Зал избира *златно* множество от преки пътища;
- Симурх казва на Зал колко от преките пътища в това златно множество са царски.

Вашата програма трябва да помогне на Зал да определи множеството от царски пътища, като зададе на Симурх най-много q въпроса. Грейдерът ще играе ролята на птицата Симурх.

Детайли по реализацията

Вие трябва да създадете следната процедура:

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

- n: брой на градовете;
- u и v: масиви с дължина m. За всяко i $(0 \leq i \leq m-1)$ елементите $u[i]$ и $v[i]$ са номерата на градовете, свързани с пряк път i.
- Тази процедура трябва да върне масив с дължина $n-1$, съдържащ номерата на царските преки пътища (в произволен ред).
- За означението `int[]` вижте първата таблица в Notice

Вашето решение трябва да прави максимум q извиквания на следната процедура на грейдера:

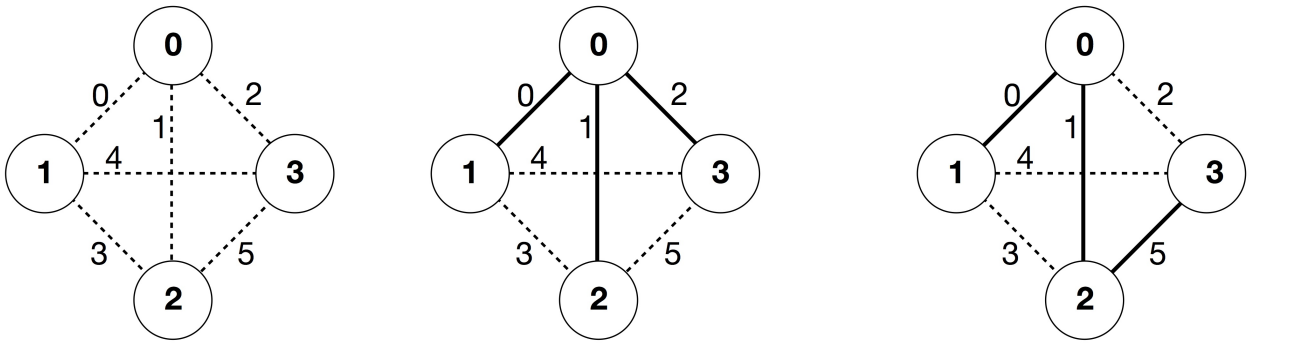
```
int count_common_roads(int[] r)
```

- r: масив с дължина $n-1$, съдържащ номерата на преките пътища в някакво златно множество (в произволен ред).
- Тази процедура връща броя на царските пътища в 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



В този пример има 4 града и 6 шосета. Ние означаваме с (a, b) пътя, свързващ градовете a и b. Шосетата са номерирани от 0 до 5 в следния ред: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, и $(2, 3)$. Всяко златно множество съдържа $n - 1 = 3$ шосета.

Да предположим, че кралските пътища са номерирани с 0, 1, and 5, т.е. това са пътищата $(0, 1)$, $(0, 2)$, и $(2, 3)$, и програмата прави следните извиквания:

- `count_common_roads([0, 1, 2])` връща 2. Това запитване е относително множеството от пътища $0, 1$ и 2, т.е. пътища $(0, 1)$, $(0, 2)$ и $(0, 3)$. Два от тях са царски.
- `count_common_roads([5, 1, 0])` връща 3. Това запитване е относително цялото множество от царски пътища.

Процедурата `find_roads` трябва да върне `[5, 1, 0]` или какъвто и да е друг масив с дължина 3 който съдържа тези три елемента.

Забележете, че следните извиквания не са позволени:

- `count_common_roads([0, 1])`: тук дължината на r не е 3.
- `count_common_roads([0, 1, 3])`: тук r не описва златно множество, защото е невъзможно да се отиде от град 0 до град 3 използвайки само преки пътища $(0, 1)$, $(0, 2)$, $(1, 2)$.

Ограничения

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n - 1$ (за всички $0 \leq i \leq m - 1$)
- За всички $0 \leq i \leq m - 1$, пряк път i свързва два различни града (т.е., $u[i] \neq v[i]$).
- Съществува най-много един пряк път между всяка двойка градове.
- Възможно е да се достигне от всеки до всеки град, използвайки системата от преки пътища.
- Множеството от всички царски пътища е златно.
- `find_roads` може да вика `count_common_roads` най-много q пъти. При всяко извикване множеството от преки пътища, зададено в r, трябва да бъде златно.

Подзадачи

1. (13 точки) $n \leq 7$, $q = 30\,000$
2. (17 точки) $n \leq 50$, $q = 30\,000$
3. (21 точки) $n \leq 240$, $q = 30\,000$
4. (19 точки) $q = 12\,000$ и съществува пряк път между всяки два града.
5. (30 точки) $q = 8\,000$

Примерен грейдер

Примерният грейдер чете вход в следния формат:

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

Тук, $s[0], s[1], \ldots, s[n-2]$ са номерата на царските пътища.

Примерният грейдер извежда `YES`, ако `find_roads` вика `count_common_roads` максимум $30\,000$ пъти и връща правилното множество от царски пътища. В противен случай извежда `NO`.

Обърнете внимание, че процедурата `count_common_roads` в примерния грейдер не проверява дали r притежава всички свойства на златното множество. Вместо това, тя преброява и връща броя на номерата на царски пътища в масива r. Обаче, ако програмата, която пращате в системата, извика `count_common_roads` с множество от преки пътища, което не е златно, съобщението от грейдера ще бъде 'Wrong Answer'.

Техническа бележка

Процедурата `count_common_roads` в C++ и Pascal използва *pass by reference* метод от съображения за ефективност. Вие трябва да извиквате процедурата по обичайния начин. Гарантирано е, че грейдерът няма да променя стойностите на r.