Simurgh

Şahname'deki eski Fars efsanesine göre, kahraman Zal Kabil prensesi Rudaba'ya aşıktır. Rudaba'nın babası Zal'a Rudaba ile evlenebilmesi için zorlu bir görev verir.

İran'da 0'dan $n-1$ kadar numaralandırılmış n şehir vardır. Herbiri farklı iki şehri birbirine bağlayan, 0'dan $m-1$'e kadar numaralandırılmış m tane çift-yönlü yol vardır. Bu yollardan bazıları *kraliyet yolu* olup sadece kraliyet üyeleri tarafından kullanılabilir. Kraliyet yollarının hangi yollar olduğu gizlidir ve Zal'ın görevi bu yolların hangileri olduğunu bulmaktır.

Zal'ın elinde İran'daki bütün şehirleri ve yolları gösteren bir harita vardır. Zal hangi yolların kraliyet yolu olduğunu bilmemektedir ancak bunun için efsanevi kuş Simurgh'dan yardım alacaktır. Simurgh hangi yolların kraliyet yolu olduğunu Zal'a direk olarak söylememektedir. Ancak kraliyet yollarının bir *altın küme* olduğu bilgisini vermektedir. Yollardan oluşan bir kümenin altın küme olması aşağıdaki gibi tanımlanmıştır:

- Bir altın kümede *tam olarak* $n-1$ yol vardır, ve
- Her şehir çifti için, şehirlerin birinden diğerine sadece bu altın kümedeki yollar kullanılarak erişilebilir.

İlaveten, Zal Simurgh'e bazı sorular sorabilir. Her soru için:

1. Önce Zal bir altın küme seçer, sonra
2. Simurgh Zal'ın seçtiği altın kümedeki yollardan kaç tanesinin kraliyet yolu olduğunu söyler.

Programınız Zal'a kraliyet yollarını bulmakta yardım edecektir. Bunu yaparken Simurgh'a en fazla q soru sorabilir. Değerlendirici Simurgh'ün rolünü üstlenecektir.

Gerçekleştirim Detayları

Aşağıdaki prosedürü kodlayınız:

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

- n: şehir sayısı,
- u ve v: m uzunluğunda diziler. $0 \leq i \leq m-1$ için; $u[i]$ ve $v[i]$, i'inci yol tarafından bağlanan şehirlerdir.
- Bu prosedür $n-1$ uzunluğunda bir dizi döndürmelidir. Bu dizide $n-1$ kraliyet yolunun numaraları (herhangi bir sırada) bulunmalıdır.

Çözümünüz aşağıdaki değerlendirici prosedürüne en fazla q çağrı (call) yapabilir:

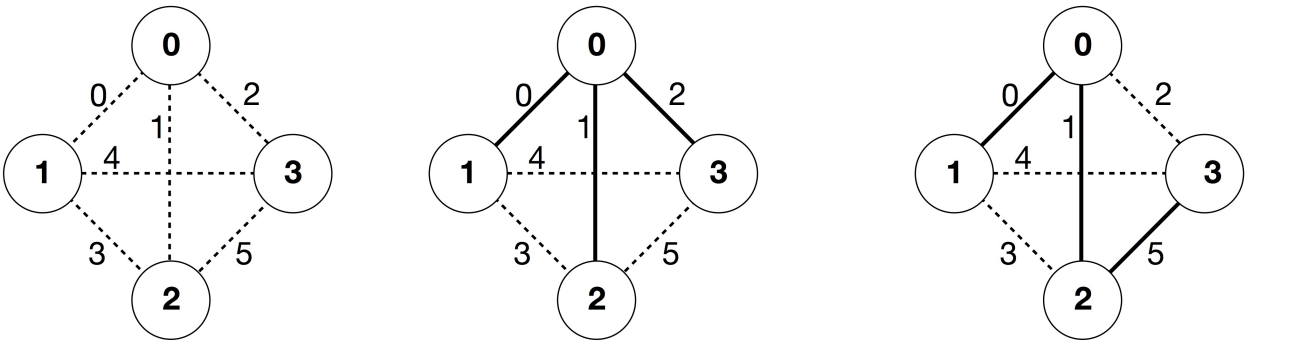
```
int count_common_roads(int[] r)
```

- r: $n-1$ uzunluğunda bir dizi olup, bir altın kümedeki yolların numaralarından oluşmaktadır (herhangi bir sırada).
- Bu prosedür r'deki kraliyet yollarının sayısını döndürmektedir.

Örnek

```
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`



Bu örnekte 4 şehir ve 6 yol vardır. (a, b) ile a ve b şehirlerini bağlayan yolu gösteriyoruz. Yollar 0'dan 5'e kadar şu sırada numaralandırılmıştır: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, ve $(2, 3)$. Her altın kümede $n - 1 = 3$ yol vardır.

Kraliyet yolları 0, 1, ve 5 numaralı yollar olsun. Başka bir deyişle $(0, 1)$, $(0, 2)$, ve $(2, 3)$ kraliyet yolları olsun. Programın aşağıdaki çağrıları yaptığını varsayın:

- `count_common_roads([0, 1, 2])` cevap olarak 2 döndürür. Bu sorguda parametre olarak $0, 1$, ve 2 numaralı yollar, yani $(0, 1)$, $(0, 2)$ ve $(0, 3)$ yolları verilmiştir. Bu yollardan 2 tanesi kraliyet yoludur.
- `count_common_roads([5, 1, 0])` cevap olarak 3 döndürür. Bu sorguda parametre olarak verilen bütün yollar kraliyet yoludur.

`find_roads` prosedürü `[5, 1, 0]` dizisini yada bu dizideki elemanlardan oluşan herhangi bir 3 uzunluğundaki diziyi döndürmelidir.

Aşağıdaki tarzda çağrılara izin verilmediğine dikkat ediniz:

- `count_common_roads([0, 1])`: Bu örnekte r'nin uzunluğu 3 değil.
- `count_common_roads([0, 1, 3])`: Bu örnekte r bir altın küme değil, çünkü sadece $(0, 1)$, $(0, 2)$, $(1, 2)$ yollarını kullanarak 0 numaralı şehirden 3 numaralı şehire ulaşmak mümkün değil.

Kısıtlar

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n-1)/2$
- $0 \leq i \leq m - 1$ aralığındaki bütün i değerleri için $0 \leq u[i], v[i] \leq n - 1$'dir.
- $0 \leq i \leq m - 1$ aralığındaki bütün i değerleri için, i'inci yol iki farklı şehri bağlamaktadır, yani $u[i] \neq v[i]$.
- Her şehir ikilisi arasında en fazla bir tane yol vardır.
- Yolları kullanarak herhangi iki şehir arasında seyahat etmek mümkündür.
- Bütün kraliyet yollarının oluşturduğu küme bir altın kümedir.
- `find_roads` prosedürü `count_common_roads` prosedürünü en fazla q defa çağırmalıdır. Her çağrıda r dizisindeki yollar bir altın küme olmalıdır.

Alt Görevler

1. (13 puan) $n \leq 7$, $q = 30\,000$
2. (17 puan) $n \leq 50$, $q = 30\,000$
3. (21 puan) $n \leq 240$, $q = 30\,000$
4. (19 puan) $q = 12000$'dir ve her farklı şehir çifti arasında bir yol vardır.
5. (30 puan) $q = 8000$

Örnek Değerlendirici

Örnek değerlendirici girdiyi aşağıdaki formatta okumaktadır:

- satır 1: n m
- satır $2 + i$ ($0 \leq i \leq m - 1$ aralığındaki her i değeri için): $u[i]$ $v[i]$
- satır $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

Buradaki $s[0], s[1], \ldots, s[n-2]$ kraliyet yollarının numaralarıdır.

Eğer `find_roads` prosedürü `count_common_roads` prosedürünü en fazla $30\,000$ defa çağırırsa ve bütün kraliyet yollarını doğru olarak döndürürse örnek değerlendirici `YES` çıktısı verir. Aksi taktirde `NO` çıktısı verir.

Örnek değerlendiricideki `count_common_roads` prosedürünün r'nin bir altın küme olup olmadığını kontrol etmediğine dikkat ediniz. `count_common_roads` prosedürü verilen r dizisindeki kraliyet yollarını sayıp, bu sayıyı döndürmektedir. Ancak, gönderdiğiniz program `count_common_roads` prosedürünü altın kümeye karşılık gelmeyen bir diziyle çağırırsa değerlendirici kararı 'Wrong Answer' olacaktır.

Teknik Not

`count_common_roads` prosedürü C++ ve Pascal'da verimlilik gerekçesiyle *referans ile*

gönderme(pass by reference) yöntemini kullanmaktadır. İsterseniz prosedürü her zamanki şekilde çağırabilirsiniz. Değerlendirici r'nin değerini değiştirmemeyi garanti eder.