Simurgh

De acuerdo a antiguas leyendas Persas en Shahnameh, Zal, el legendario héroe Persa, está perdidamente enamorado de Rudaba, la princesa de Kabul. Cuando Zal pidió la mano de Rudaba en matrimonio, el padre de ella le dio un desafío.

En Persia hay n ciudades, etiquetadas de 0 a $n-1$, y m vías de doble sentido etiquetadas de 0 a $m-1$. Cada vía conecta un par de ciudades distintas. Cada par de ciudades está conectado por a lo sumo una vía. Algunas vías son *vías reales* usadas para viajes de la realeza. La tarea de Zal es determinar cuáles vías son vías reales.

Zal tiene un mapa con todas las ciudades y vías en Persia. Él no sabe cuáles vías son reales, pero puede conseguir ayuda de Simurgh, la benevolente ave mítica que protege a Zal. Sin embargo, Simurgh no quiere revelar el conjunto de vías reales directamente. En cambio, Simurgh le ha dicho a Zal que las vías reales forman un *conjunto dorado*. Un conjunto de vías es un conjunto dorado si y sólo si:

- tiene exactamente $n-1$ vías, y
- para cada par de ciudades, es posible llegar a una desde la otra viajando sólo a través de las vías de este conjunto.

Más aún, Zal puede hacer preguntas a Simurgh. Por cada pregunta:

1. Zal escoge un conjunto *dorado* de vías, y luego
2. Simurgh le dice a Zal cuántas de las vías en el conjunto dorado elegido son vías reales.

Su programa debe ayudar a Zal a encontrar el conjunto de vías reales haciendo a lo sumo q preguntas a Simurgh. El calificador jugará el rol de Simurgh.

Detalles de implementación

Usted debe implementar la siguiente función:

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

- n: cantidad de ciudades,
- u y v: arreglos de longitud m. Para todo $0 \leq i \leq m-1$, $u[i]$ y $v[i]$ son las ciudades conectadas por el camino i.
- Esta función debe retornar un arreglo de longitud $n-1$ que contenga las etiquetas de las vías reales (en orden arbitrario).

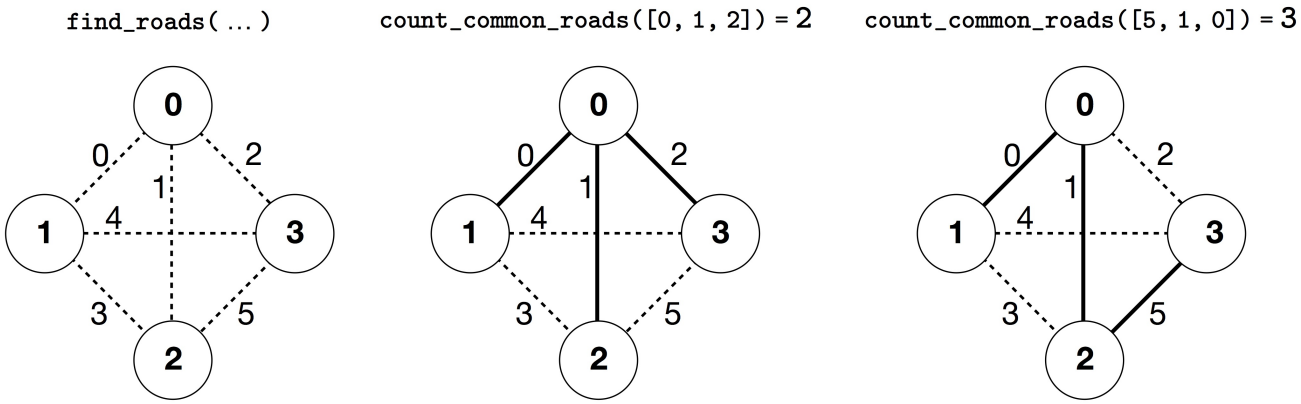
Su solución puede realizar a lo sumo q llamadas a la siguiente función:

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

- r: arreglo de longitud $n-1$ que contiene las etiquetas de vías en un conjunto dorado (en orden arbitrario).
- Esta función retorna la cantidad de vías reales en r.

Ejemplo

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



En este ejemplo hay 4 ciudades y 6 vías. Denotamos por (a, b) una vía que conecta las ciudades a y b. Las vías están etiquetadas de 0 a 5 en el siguiente orden: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$, y $(2, 3)$. Cada conjunto dorado tiene $n - 1 = 3$ vías.

Asuma que las vías reales son las etiquetadas con 0, 1, y 5, es decir, las vías $(0, 1)$, $(0, 2)$, y $(2, 3)$. Entonces:

- `count_common_roads([0, 1, 2])` retorna 2. Esta consulta pregunta por las vías etiquetadas 0, 1, y 2, es decir, las vías $(0, 1)$, $(0, 2)$, y $(0, 3)$. Dos de ellas son vías reales.
- `count_common_roads([5, 1, 0])` retorna 3. Esta consulta pregunta por el conjunto de todas las vías reales.

La función `find_roads` debería retornar `[5, 1, 0]` o cualquier otro arreglo de longitud 3 que contenga a estos tres elementos.

Note que las siguientes llamadas no están permitidas:

- `count_common_roads([0, 1])`: aquí la longitud de r no es 3.
- `count_common_roads([0, 1, 3])`: aquí r no describe un conjunto dorado, porque es imposible viajar desde la ciudad 0 a la ciudad 3 siguiendo solo las vías $(0, 1)$, $(0, 2)$, y $(1, 2)$.

Restricciones

- $2 \leq n \leq 500$
- $n-1 \leq m \leq n(n-1)/2$
- $0 \leq u[i], v[i] \leq n-1$ (para todo $0 \leq i \leq m-1$)
- para todo $0 \leq i \leq m-1$, la vía i conecta dos ciudades diferentes (es decir, $u[i] \neq v[i]$).
- Hay a lo sumo una vía entre cada par de ciudades.
- Es posible viajar entre cualquier par de ciudades a través de las vías
- El conjunto de todas las vías reales es un conjunto dorado.
- `find_roads` debe llamar a `count_common_roads` a lo sumo q veces. En cada llamada, el conjunto de vías especificado por r debe ser un conjunto dorado.

Subtareas

1. (13 puntos) $n \leq 7$, $q = 30000$
2. (17 puntos) $n \leq 50$, $q = 30000$
3. (21 puntos) $n \leq 240$, $q = 30000$
4. (19 puntos) $q = 12000$ y hay una vía entre cada par de ciudades
5. (30 puntos) $q = 8000$

Calificador de ejemplo

El calificador de ejemplo lee la entrada en el siguiente formato:

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

Aquí, $s[0], s[1], \ldots, s[n-2]$ son las etiquetas de los caminos reales.

El calificador de ejemplo responde `YES`, si `find_roads` llama a `count_common_roads` a lo sumo 30000 veces, y retorna el conjunto correcto de vías reales. De lo contrario responde `NO`.

Considere que la función `count_common_roads` en el calificador de ejemplo no revisa si r tiene todas las propiedades de un conjunto dorado. Por el contrario, cuenta y retorna la cantidad de etiquetas de vías reales en el arreglo r. Sin embargo, si el programa que usted genere llama a `count_common_roads` con un conjunto de etiquetas que no describe un conjunto dorado, el veredicto del CMS será 'Wrong Answer'.

Nota técnica

La función `count_common_roads` en C++ y Pascal usa el método de *pasar por referencia* por razones de eficiencia. De todas maneras usted puede llamar a la función en la forma usual. Se garantiza que el calificador no cambiará el valor de r.