Estaciones (stations)

El Internet Backbone de Singapur (SIB por sus siglas en inglés) consiste de n estaciones, a las que se les asignan **índices** del 0 a $n-1$. También tiene $n-1$ enlaces bidireccionales, numerados del 0 al $n-2$. Cada enlace conecta dos estaciones diferentes. Decimos que dos estaciones conectadas por un mismo enlace son vecinas.

Un camino de la estación x a la estación y es una secuencia de estaciones distintas $a_0, a_1, \cdots, a_p$, tal que $a_0 = x$, $a_p = y$, y cada par de estaciones consecutivas son vecinas. Hay **exactamente un** camino de cada estación x a cada otra estación y.

Cualquier estación x puede crear un paquete (una unidad de información) y mandársela a cualquier otra estación y, que la llamamos el **destino** del paquete. Este paquete debe ser enviado a través del camino único de x a y de la siguiente manera. Considera una estación z que en este momento tiene un paquete cuya estación destino es y ($z \neq y$). En esta situación, la estación z:

1. ejecuta un **procedimiento de enrutamiento** que determina el vecino de z que está en el camino único de z a y, y
2. le pasa el paquete a ese vecino.

Sin embargo, las estaciones tienen una cantidad limitada de memoria y no guardan la lista entera de enlaces en el SIB para usarlos en el procedimiento de enrutamiento.

Tu tarea es implementar un esquema de enrutamiento para el SIB, que tiene dos funciones.

- La primer función recibe n, la lista de los enlaces en el SIB y un entero $k \geq n-1$ como entradas.La función le asigna a cada estación una **etiqueta** entera **única** entre 0 y k, inclusive.

- La segunda función es el procedimiento de enrutamiento, que es enviado a todas las estaciones después de que las etiquetas fueron asignadas. Se le da **únicamente** los siguientes datos de entrada:

  - s, la **etiqueta** de la estación que en este momento tiene un paquete,
  - t, la **etiqueta** de la estación de destino del paquete ($t \neq s$),
  - c, la lista de las **etiquetas** de todos los vecinos de s.

  Debe regresar la **etiqueta** del vecino de s al que se le debería enviar el paquete.

En una subtarea, el puntaje de tu solución depende del valor máximo de etiqueta que le hayas asignado a cualquier estación. (En general, entre más pequeño es mejor.)

Detalles de implementación

Debes implementar las siguentes funciones:

```
int[] label(int n, int k, int[] u, int[] v)
```

- n: número de estaciones en el SIB.
- k: máximo número de etiqueta que se puede usar.
- u y v: arreglos de tamaño $n-1$ describiendo los enlaces. Para cada i ($0 \leq i \leq n-2$), el enlace i conecta a las estaciones con índices $u[i]$ y $v[i]$.
- Esta función debe regresar un único entero L de tamaño n. Para cada i ($0 \leq i \leq n-1$) $L[i]$ es la etiqueta asignada a la estación con índice i. Todos los elementos del arreglo L deben ser únicos y estar entre 0 y k, inclusive.

```
int find_next_station(int s, int t, int[] c)
```

- s: la etiqueta de la estación que tiene un paquete.
- t: la etiqueta de la estación destino del paquete.
- c: un arreglo con la lista de todas las etiquetas de todos los vecinos de s. El arreglo c está ordenado de manera ascendiente.
- Esta función debe regresar la etiqueta del vecino de s al que se debería enviar el paquete.

Cada caso de prueba tiene uno o más escenarios independientes (es decir, diferentes descripciones del SIB). Para un caso de prueba con r escenarios, un **programa** que llama las funciones descritas arriba es ejecutado exactamente dos veces, de la siguiente manera.

Durante la primera ejecución del programa:

- La función `label` es llamada r veces,
- las etiquetas devueltas son guardadas por el sistema de evaluación, y
- no se llama a `find_next_station`.

Durante la segunda ejecución del programa:

- `find_next_station` puede ser llamada varias veces. En cada llamada, un escenario **arbitrario** es seleccionado, y las etiquetas regresadas por la llamada a la función a `label` en ese escenario son usadas como input de `find_next_station`.
- `label` no se manda llamar.

En particular, cualquier información guardado en variables estáticas o globales de la primera ejecución del programa no estará disponible en la función `find_next_station`.

Ejemplo

Considera la siguiente llamada:

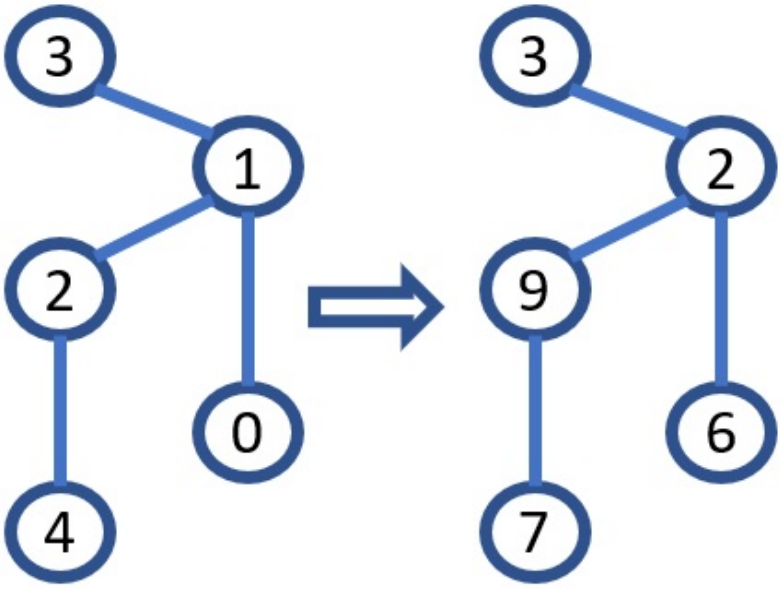
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Hay 5 estaciones en total, y 4 enlaces conectando a las estaciones con índices $(0, 1)$, $(1, 2)$, $(1, 3)$ y $(2, 4)$. Cada etiqueta puede ser un entero entre 0 y $k = 10$.

Para reportar el siguiente etiquetado:

Índice	Etiqueta
0	6
1	2
2	9
3	3
4	7

la función `label` debe regresar el arreglo $[6, 2, 9, 3, 7]$. Los números en la siguiente figura muestra los índices (panel izquierdo) y las etiquetas asignadas (panel derecho).



Supón que las etiquetas fueron asignadas de la manera descrita anteriormente y considera la siguiente llamada:

```
find_next_station(9, 6, [2, 7])
```

Esto significa que la estación que tiene el paquete tiene etiqueta 9, y la estación destino tiene etiqueta 6. Las etiquetas de las estaciones en el camino a la estación destino son $[9, 2, 6]$. Por lo tanto, la llamada debería regresar 2, que es la etiqueta de la estación a la que el paquete se debería enviar (que tiene índice 2).

Considera otra llamada posible:

```
find_next_station(2, 3, [3, 6, 9])
```

La función debería regresar 3, ya que la estación destino con etiqueta 3 es vecina de la estación con etiqueta 2, y por lo tanto debe recibir el paquete directamente.

Límites

- $1 \leq r \leq 10$

Para cada llamada a `label`:

- $2 \leq n \leq 1000$
- $k \geq n - 1$
- $0 \leq u[i], v[i] \leq n - 1$ (para toda $0 \leq i \leq n - 2$)

Para cada llamada a `find_next_station`, la entrada viene de una llamada previa a `label`, arbitrariamente seleccionada : Considera las etiquetas producidas, entonces:

- s y t son etiquetas de dos estaciones distintas.
- c es la secuencia de todas las etiquetas de vecinos de la estación con etiqueta s, en orden ascendente.

Para cada caso de prueba, la longitud total de todos los arreglos c que se mandan a la función `find_next_station` no excede $100\ 000$ entre todos los escenarios combinados.

Subtareas

1. (5 puntos) $k = 1000$, ninguna estación tiene más de 2 vecinos.
2. (8 puntos) $k = 1000$, el enlace i conecta a las estaciones $i + 1$ y $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 puntos) $k = 1\ 000\ 000$, a lo más una estación tiene más de 2 vecinos.
4. (10 puntos) $n \leq 8$, $k = 10^9$
5. (61 puntos) $k = 10^9$

En la subtarea 5 puedes obtener un puntaje parcial. Sea m la etiqueta máxima regresada por `label` entre todos los escenarios. Tu puntaje para esta subtarea se calcula de acuerdo a la siguiente tabla:

Etiqueta máxima	Puntaje
$m \geq 10^9$	0
$2000 \leq m < 10^9$	$50 \cdot \log_{5 \cdot 10^5}(\frac{10^9}{m})$
$1000 < m < 2000$	50
$m \leq 1000$	61

Evaluador de prueba

El evaluador de prueba lee la entrada con el siguiente formato:

- línea 1: r

Después hay r bloques, cada uno describiendo un único escenario. El formato de cada bloque es el siguiente:

- línea 1: $n \ \ k$
- línea $2 + i$ $(0 \leq i \leq n - 2)$: $u[i] \ \ v[i]$
- línea $1 + n$: q: el número de llamadas a `find_next_station`.
- línea $2 + n + j$ $(0 \leq j \leq q - 1)$: $z[j] \ \ y[j] \ \ w[j]$: los **índices** de las estaciones involucradas en la $j$ésima llamada a `find_next_station`. La estación $z[j]$ tiene el paquete, la estación $y[j]$ es el destino, y la estación $w[j]$ es la estación del paquete que debe ser enviado.

El evaluador de prueba imprime el resultado con el siguiente formato:

- línea 1: m

Después r bloques correspondiendo a los escenarios de la entrada. El formato de cada bloque es el siguiente:

- línea $1 + j$ $(0 \leq j \leq q - 1)$: el **índice** de la estación cuya **etiqueta** fue devuelta por la $j$ésima llamada a `find_next_station` en este escenario.

Nota que cada ejecución del evaluador de prueba llama tanto a `label` como a `find_next_station`.