Estaciones (stations)

La Infraestructura de Internet de Singapur (IIS) consiste de n estaciones, a las que se le asignan **indices** del 0 al $n-1$. También hay $n-1$ enlaces bidireccionales, numerados de 0 a $n-2$. Cada enlace conecta dos estaciones distintas. Dos estaciones conectadas por un mismo enlace son llamadas 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 dos estaciones consecutivas en el camino son vecinas. Hay **exactamente un** camino de cualquier estación x a cualquier otra estación y.

Cualquier estación x puede crear un paquete (una pieza de información) y enviarlo a cualquier otra estación y, a la que le llamamos el **objetivo** del paquete. Este paquete debe ser enrutado a lo largo del camino único de x a y de la siguiente manera. Considera una estación z que actualmente contiene a un paquete cuya estación objetivo es y ($z \neq y$). En esta situación, la estación z:

1. ejecuta un **procedimiento de enrutamiento** que determina el vecino de z que se encuentra en el camino único de z a y, y
2. envía el paquete a ese vecino.

Sin embargo, las estaciones tienen memoria limitada y no almacenan la lista entera de enlaces en la IIS para utilizarlo en el procedimiento de enrutamiento.

Tu tarea es implementar un esquema de enrutamiento para la IIS, que consiste en dos procedimientos.

- El primer procedimiento recibe n, la lista de enlaces en la IIS y un entero $k \geq n-1$ como entradas. Este asigna a cada estación una **etiqueta** que es un número entero **único** entre 0 y k, inclusive.
- El segundo procedimiento es el procedimiento de enrutamiento, el cual es enviado a todas las estaciones después que las etiquetas son asignadas. Le son dadas **únicamente** las siguientes entradas:
  - s, la **etiqueta** de la estación que actualmente contiene el paquete.
  - t, la **etiqueta** de la estación objetivo del paquete ($t \neq s$),
  - c, la lista de las **etiquetas** de todos los vecinos de s.

Debe retornar la **etiqueta** del vecino de s al que debe enviarse el paquete.

En una subtarea, la puntuación de tu solución depende del valor de la máxima etiqueta asignada a cualquier estación (en general, más pequeña es mejor).

Detalles de implementación

Debes implementar los siguientes procedimientos (funciones):

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

- n: número de estaciones en la IIS.
- k: etiqueta máxima que puede ser usada.
- 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 $v[i]$.
- Este procedimiento debe retornar un único arreglo 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 deben estar entre 0 y k, inclusive.

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

- s: etiqueta de la estación que actualmente contiene al paquete.
- t: etiqueta de la estación objetivo del paquete.
- c: un arreglo que contiene la lista de las etiquetas de todos los vecinos de s. El arreglo c está ordenado ascendentemente.
- Este procedimiento debe retornar la etiqueta del vecino de s al que debe enviarse el paquete.

Cada caso de prueba involucra uno o más escenarios independientes (es decir, diferentes descripciones de la IIS). Para un caso de prueba que involucre r escenarios, un **programa** que llama a los procedimientos antes mencionados es ejecutado exactamente dos veces, como se describe a continuación.

Durante la primera ejecución del programa:

- El procedimiento `label` es llamado r veces,
- los valores de retorno son almacenados por el sistema evaluador, y
- `find_next_station` no es llamada.

Durante la segunda ejecución del programa:

- `find_next_station` puede ser llamada múltiples veces. En cada llamada, un escenario **cualquiera** es seleccionado, y las etiquetas retornadas por la llamada a `label` en ese escenario son usadas como entrada para `find_next_station`.
- `label` no es llamada.

En particular, cualquier información guardada en variables estáticas o globales en la primera ejecución del programa no estarán disponibles dentro del procedimiento `find_next_station`.

Ejemplo

Considere la siguiente llamada:

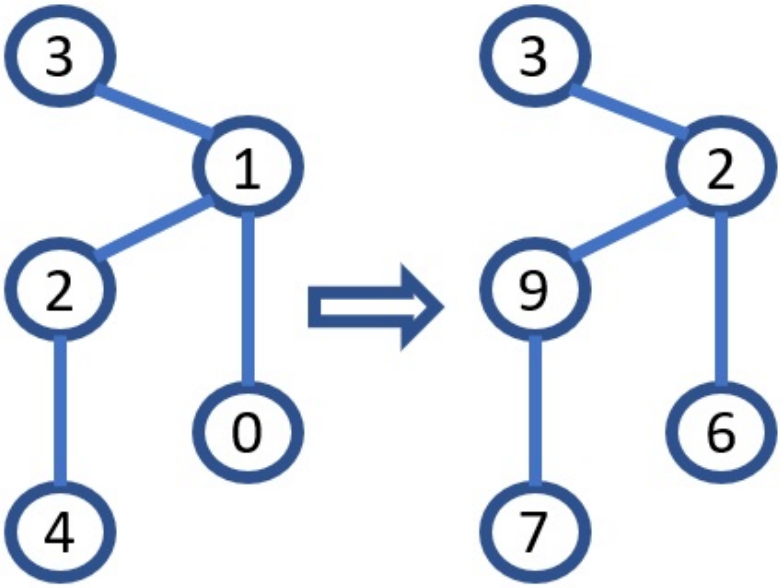
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

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

Para reportar la siguiente asignación de etiquetas:

Index	Label
0	6
1	2
2	9
3	3
4	7

el procedimiento `label` debe retornar $[6, 2, 9, 3, 7]$. Los números en la siguiente figura muestran los índices (imágen de la izquierda) y las etiquetas asignadas (imágen derecha).



Asume que las etiquetas han sido asignadas como se describió anteriormente y considera la llamada siguiente:

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

Esto significa que la estación conteniendo el paquete tiene como etiqueta el 9, y la estación objetivo del paquete tiene etiqueta 6. Las etiquetas de las estaciones en el camino hacia la estación objetivo son $[9, 2, 6]$. Por lo tanto, el procedimiento debe retornar 2, que es la etiqueta de la estación hacia

la cual debe enviarse el paquete (esa estación tiene índice 1).

Considere otra llamada posible:

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

El procedimiento debe retornar 3 ya que la estación objetivo con etiqueta 3 es una 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 es tomada de una llamada previa cualquiera de las llamadas a `label`. Considera las etiquetas producidas en dicha llamada. Entonces:

- s y t son etiquetas de dos estaciones diferentes.
- c es la secuencia de todas las etiquetas de las estaciones vecinas a la estación s, en orden ascendente.

Para cada caso de prueba, el largo total de los arreglos c dados al procedimiento `find_next_station` no excede $100\ 000$ para todos los escenarios combinados.

Subtareas

1. (5 puntos) $k = 1000$, y 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 sumo 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 una puntuación parcial. Sea m la máxima etiqueta retornada por `label` tomando en cuenta todos los escenarios. Tu puntuación para esta subtarea es calculada de acuerdo a la siguiente tabla:

Etiqueta máxima	Puntuación
$m \geq 10^9$	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

Grader de ejemplo

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

- línea 1: r

Siguen r bloques, cada uno describiendo un único escenario. El formato de cada bloque es:

- 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 $2 + n + j$ $(0 \leq j \leq q - 1)$: $z[j]$ $y[j]$ $w[j]$: **índices** de las estaciones involucradas en la j-ésima llamada a `find_next_station`. La estación $z[j]$ contiene al paquete, la estación $y[j]$ es el objetivo del paquete, y la estación $w[j]$ es la estación a la cual debe enviarse el paquete.

El grader de ejemplo imprime el resultado en el siguiente formato:

- línea 1: m

Seguidos de r bloques correspondientes a los escenarios consecutivos. El formato de cada bloque es:

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

Nota que cada ejecución del grader de ejemplo llama tanto a `label` como a `find_next_station`.