Estaciones (stations)

La red de Internet de Singapur (SIB: *Singapore's Internet Backbone*) está formada por n estaciones de comunicación, que tienen asignados **índices** desde 0 hasta $n-1$. También hay $n-1$ enlaces bidireccionales, numerados desde 0 hasta $n-2$. Cada enlace conecta dos estaciones diferentes. Si dos estaciones están conectadas por algún enlace, se dice que son vecinas.

Un camino desde la estación x hasta la estación y es una secuencia de estaciones diferentes $a_0, a_1, \cdots, a_p$, tal que $a_0 = x$, $a_p = y$, y cada par de estaciones consecutivas en el camino son vecinas. Existe **exactamente un** camino desde cualquier estación x hasta cualquier otra estación y.

Una estación x cualquiera puede crear un paquete (que contiene datos) y enviarlo a cualquier otra estación y, la cual se denomina **destino** del paquete. Este paquete debe ser guiado ("enrutado") a través del único camino desde x hasta y de la siguiente manera. Supongamos que una estación z tiene actualmente un paquete, cuya estación de destino es y ($z \neq y$). En esta situación la estación z:

1. ejecuta una **función de ruteo** que determina aquel vecino de z que forma parte del único camino desde z hasta y, y luego
2. redirige el paquete a este vecino.

Sin embargo, la capacidad de memoria en las estaciones es limitada, y no almacenan la totalidad de la lista de enlaces del SIB para utilizar en la función de ruteo.

Tu tarea consiste en implementar un esquema de ruteo para el SIB, que consta de dos funciones.

- La primera función recibe n, la lista de enlaces del SIB y un entero $k \geq n-1$ como parámetros. La función le asigna **a cada estación un entero diferente**, que es una **etiqueta** entre 0 y k, inclusive.

- La segunda función es la función de ruteo, que se instala en todas las estaciones luego de que se hayan asignado las etiquetas. Recibe **solamente** los siguientes parámetros:

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

  Debe retornar la **etiqueta** del vecino de s al cual se debe redirigir el paquete.

En una de las subtareas, el puntaje obtenido depende del valor de la máxima etiqueta asignada a una estación (en general, cuanto menor sea, mejor).

Detalles de implementación

Debes implementar las siguientes funciones:

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

- n: cantidad de estaciones del SIB.
- k: máxima etiqueta que se puede utilizar.
- u y v: arreglos de longitud $n-1$ que describen los enlaces. Para cada i ($0 \leq i \leq n-2$), el enlace i conecta las estaciones con índices $u[i]$ y $v[i]$.
- Esta función debe retornar un único arreglo L de longitud 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 diferentes entre sí, y estar entre 0 y k, inclusive.

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

- s: etiqueta de la estación que tiene el paquete.
- t: etiqueta de la estación destino del paquete.
- c: un arreglo que indica las etiquetas de todos los vecinos de s. El arreglo c está ordenado ascendentemente.
- Esta función debe retornar la etiqueta del vecino de s al cual el paquete debe ser redirigido.

Cada caso de prueba involucra uno o más escenarios independientes (es decir, diferentes descripciones del SIB). Para un caso de prueba que involucra r escenarios, un **programa** que llama a las funciones anteriores se ejecuta exactamente dos veces, de la siguiente manera.

Durante la primera ejecución del programa:

- `label` se llama r veces,
- las etiquetas retornadas se almacenan en el sistema de evaluación, y
- `find_next_station` no se llama.

Durante la segunda ejecución del programa:

- `find_next_station` se llama múltiples veces. En cada llamada, se elige un escenario **arbitrario**, y se usan las etiquetas retornadas por la llamada a la función `label` para ese escenario como parámetros para la función `find_next_station`.
- `label` no se llama.

En particular, ningún tipo de información almacenada en variables globales o "static" durante la primera ejecución del programa estará disponible para usar en la función `find_next_station`.

Ejemplo

Considere la siguiente llamada:

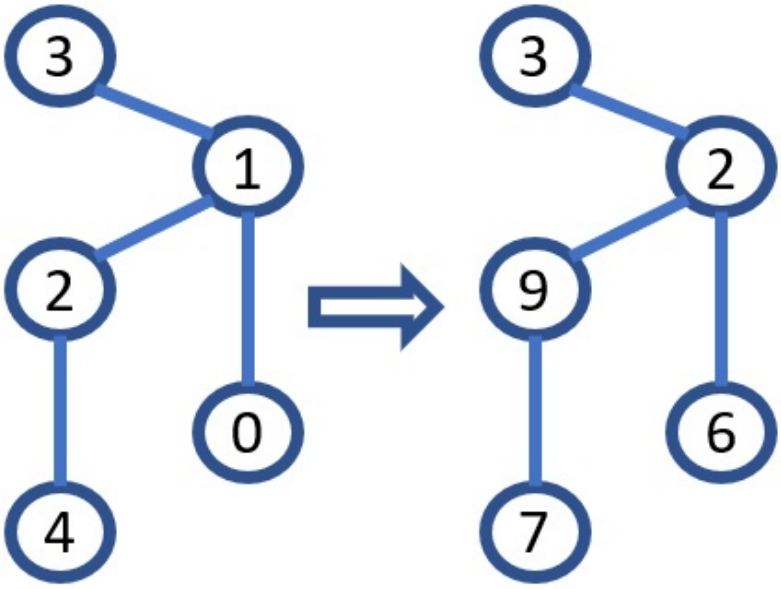
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Hay en total 5 estaciones, y 4 enlaces que conectan pares de estaciones con índices $(0, 1)$, $(1, 2)$, $(1, 3)$ y $(2, 4)$. Cada etiqueta puede ser un entero desde 0 hasta $k = 10$.

Para informar la siguiente asignación de etiquetas:

Índice	Etiqueta
0	6
1	2
2	9
3	3
4	7

la función `label` debe retornar $[6, 2, 9, 3, 7]$. Los números de la siguiente figura indican los índices (panel izquierdo) y las etiquetas asignadas (panel derecho).



Supongamos que las etiquetas se han asignado de la forma explicada arriba, y consideremos 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 hasta la estación destino son $[9, 2, 6]$. Por lo tanto, la llamada debe retornar 2, que es la etiqueta de la estación a la cual el paquete debe ser redirigido (la cual tiene índice 1).

Consideremos otra posible llamada:

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

La función debe retornar 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.

Cotas

- $1 \leq r \leq 10$

En cada llamada a `label`:

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

En cada llamada a `find_next_station`, la entrada proviene de una llamada previa a `label` elegida arbitrariamente.
Consideremos las etiquetas generadas por esa llamada. Entonces:

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

En cada caso de prueba, la longitud total de todos los arreglos c pasados a la función `find_next_station` nunca excede $100\ 000$, considerando 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 la estación $i + 1$ con la $\left\lfloor \frac{i}{2} \right\rfloor$.
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 puntaje parcial. Sea m el valor de la etiqueta máxima retornada por `label` entre todos los escenarios. Tu puntaje en 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 local

El evaluador local lee la entrada con el siguiente formato:

- línea 1: r

A continuación r bloques, cada uno de los cuales describe 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: la cantidad 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 $z[j]$ tiene el paquete, la estación $y[j]$ es el destino del paquete, y la estación $w[j]$ es aquella a la cual el paquete debe ser redirigido.

El evaluador local escribe la salida con el siguiente formato:

- línea 1: m

A continuación r bloques que se corresponden con los escenarios de la entrada. El formato de cada bloque es el siguiente:

- 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.

Notar que cada ejecución del evaluador local llama tanto a `label` como a `find_next_station`.