Estaciones (stations)

La red troncal de Internet de Singapur (SIB) consiste en n, estaciones a las que se les asignan **índices** de 0 a $n-1$. También hay $n-1$ enlaces bidireccionales, numerados desde 0 hasta $n-2$. Cada enlace conecta dos estaciones distintas. Dos estaciones conectadas con un solo enlace se llaman vecinas.

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

Cualquier estación x puede crear un paquete (una parte de los datos) y enviarlo a cualquier otra estación y, que es llamado **objetivo** del paquete. Este paquete debe ser enrutado a lo largo del único camino desde x hasta y de la siguiente manera.

Considere una estación z que actualmente contiene 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 está en el camino único de z a y, y
2. envía el paquete a este vecino.

Sin embargo, las estaciones tienen una memoria limitada y no almacenan la lista completa de los enlaces en SIB para usarla en el procedimiento de enrutamiento.

Su tarea es implementar un esquema de enrutamiento para SIB, que consiste en dos procedimientos.

- El primer procedimiento se le da n, la lista de los enlaces en el SIB y un entero $k \geq n-1$ como las entradas. Le asigna a cada estación un **unico** entero **etiqueta** entre 0 y k, inclusive.

- El segundo procedimiento es el de enrutamiento, que se despliega en todas las estaciones después de que se asignen las etiquetas. Se le dan **sólo** las siguientes entradas:

    - 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 las **etiquetas** de todos los vecinos de s.

    Debería retornar la **etiqueta** del vecino s a la que el paquete debería ser enviado.

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

Detalles de implementación

Debería implementar los siguientes procedimientos:

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

- n: número de estaciones en el SIB.
- k: etiqueta máxima que puede ser usada.
- u y v: arreglos de tamaño $n-1$ que describen los enlaces. Por cada i ($0 \leq i \leq n-2$), el enlace i conecta las estaciones con los índices $u[i]$ y $v[i]$.
- Este procedimiento debería retornar un único arreglo L de tamaño n. Por cada i ($0 \leq i \leq n-1$) $L[i]$ es la etiqueta asignada a la estación con el índice i. Todos los elementos del arreglo L deben ser únicos y entre 0 y k, inclusive.

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

- s: etiqueta de la estación que contiene un paquete.
- t: etiqueta de la estación objetivo del paquete.
- c: un arreglo que da la lista de las etiquetas de todos los vecinos de s. El arreglo c está ordenada en orden ascendente.
- Este procedimiento debería retornar la etiqueta de un vecino de s al que el paquete debería ser enviado.

Cada caso de prueba implica uno o más escenarios independientes (es decir, diferentes descripciones de SIB). Para un caso de prueba que involucra r escenarios, un **programa** que llama a los procedimientos anteriores se ejecuta exactamente dos veces, como sigue.

Durante la primera ejecución del programa:

- `label` el procedimiento se llama r veces,
- las etiquetas retornadas son almacenadas por el sistema grader, y
- `find_next_station` no se llama.

Durante la segunda ejecución del programa:

- `find_next_station` puede ser llamado varias veces. En la llamada de búsqueda, se elige un escenario **arbitrario**, y las etiquetas devueltas por la llamada al procedimiento `label` en que se usan como entradas, a `find_next_station`.
- `label` no se llama

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

Ejemplo

Considere la siguiente llamada:

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

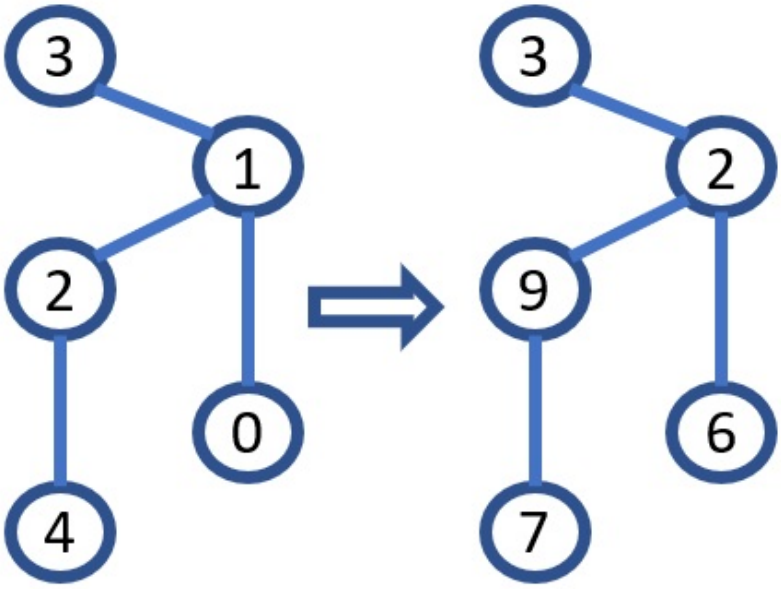
There are a total of 5 stations, and 4 links connecting pairs of stations with indices $(0, 1)$, $(1, 2)$, $(1, 3)$ and $(2, 4)$. Each label can be an integer from 0 to $k = 10$.

In order to report the following labelling: Hay un total de 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 número entero desde 0 hasta $k = 10$.

El orden del reporte sobre el siguiente etiquetado:

Índice	Etiqueta
0	6
1	2
2	9
3	3
4	7

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



Supongamos que las etiquetas se han asignado como se ha descrito anteriormente y consideremos la siguiente llamada:

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

Esto significa que la estación que sostiene el paquete tiene la etiqueta 9, y la estación objetivo tiene la etiqueta 6. Las etiquetas de las estaciones en el camino hacia la estación objetivo son $[9, 2, 6]$. Por lo tanto, la llamada debe retornar 2, que es la etiqueta de la estación a la que se debe reenviar el paquete (que tiene el índice 1).

Considere otra posible llamada:

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

El procedimiento debe devolver 3, ya que la estación objetivo con la etiqueta 3 es vecina de la estación con la etiqueta 2, y por lo tanto debe recibir el paquete directamente.

Restricciones

- $1 \leq r \leq 10$

Para cada llamada a `label`:

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

Para cada llamada a `find_next_station`, la entrada proviene de una llamada previa a `label`. elegida arbitrariamente. Considere las etiquetas que produjo. Luego:

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

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

Subtareas

1. (5 puntos) $k = 1000$, ninguna estación tiene más de 2 vecinos.
2. (8 puntos) $k = 1000$, i enlaces conectan 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 usted puede obtener una puntuación parcial. Dejemos que m sea el valor máximo de la etiqueta retornada por `label` en todos los escenarios. Tu puntuación para esta subtarea se calcula según 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 ejemplos

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

- línea 1: r

Siguen r,bloques, cada uno describiendo un simple 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 $2 + n + j$ $(0 \leq j \leq q - 1)$: $z[j] \ \ y[j] \ \ w[j]$: **ndices** de las estaciones involucradas en la j-th llamada a `find_next_station`. La estación $z[j]$ contiene el paquete, la estación $y[j]$ es el objetivo del paquete, y la estación $w[j]$ es la estación a la que el paquete debe ser reenviado.

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

- line 1: m

r bloques correspondientes a los escenarios consecutivos en la siguiente entrada siguen. 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-th llamada a `find_next_station` en este escenario.

Observe que cada ejecución del clasificador de muestras llama tanto a `label` como a `find_next_station`.