Estaciones (stations)

La conexión a la red troncal de Internet de Singapur (SIB, Singapore's Internet Backbone) consiste de n estaciones, a las cuales se les asigna **índices** de 0 a $n-1$. Hay además $n-1$ enlaces bidireccionales, numerados de 0 a $n-2$. Cada enlace conecta dos estaciones distintas. Dos estaciones conectadas con un simple enlace son llamadas vecinas.

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

Cualquier estación x puede crear un paquete (un conjunto de datos) y enviar este a cualquier otra estación y, la cual es llamada el **objetivo** del paquete. Este paquete debe ser ruteado a travez de un único camino de x a y como sigue. Considere una estación z que actualmente contiene un paquete, cuya estación objetivo es y ($z \neq y$). La estación

1. ejecuta un **procedimiento de enrutamiento** que determina el vecino de z el cual está en el camino único de z a y, y
2. reenvía el paquete a su vecino.

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

Tú estás implementando un esquema de enrutamiento para SIB, el cual consiste de dos procedimientos.

- El primer procedimiento es, dado n la lista de los enlaces en SIB y un entero $k \geq n-1$ como las entradas. Este asigna a cada estación un único entero como **etiqueta** entre 0 y k, inclusive.

- El segundo procedimiento es el procedimiento de enrutamiento, y es desplegado a todas las estaciones después que sus etiquetas sean asignadas. Este es, dadas **solo** las siguientes entradas:

    - s, la **etiqueta** de la estación que actualmente contiene un 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.

    Este debe devolver la **etiqueta** del vecino de s al que se debe reenviar el paquete.

Adicionalmente, el puntaje de tu solución depende del valor de la máxima etiqueta asignada a cualquier estación (en general, mientras más pequeña mejor).

Detalles de implementación

Debes implementar los siguientes procedimientos:

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

- n: número de estaciones en SIB.
- k: máxima etiqueta que puede ser usada.
- u y v: arreglos de tamaño $n-1$ describiendo los enlaces. Por cada i ($0 \leq i \leq n-2$), el enlace i conecta las estaciones con índices $u[i]$ y $v[i]$.
- Este procedimiento debe retornar un arreglo simple 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 arrelgo L deben ser únicos y estar 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 dando la lista de etiquetas de todos los vecinos de s. El arreglo c es guardado en orden ascendente.
- Este procedimiento debe retornar la etiqueta de un vecino de s al que se debe reenviar el paquete.

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

Durante la primera corrida del programa:

- el procedimiento `label` es llamado r veces,
- las etiquetas retornadas son guardadas por el sistema de evaluación, y
- `find_next_station` no es llamada.

Durante la segunda corrida del programa:

- `find_next_station` puede ser llamada multiples veces,
- las etiquetas dadas a cada llamada a `find_next_station` son las etiquetas producidas por una llamada a `label` para un escenario **arbitrariamente tomado** de la primera corrida, y
- `label` no es llamada.

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

Ejemplo

Considere la siguiente llamada:

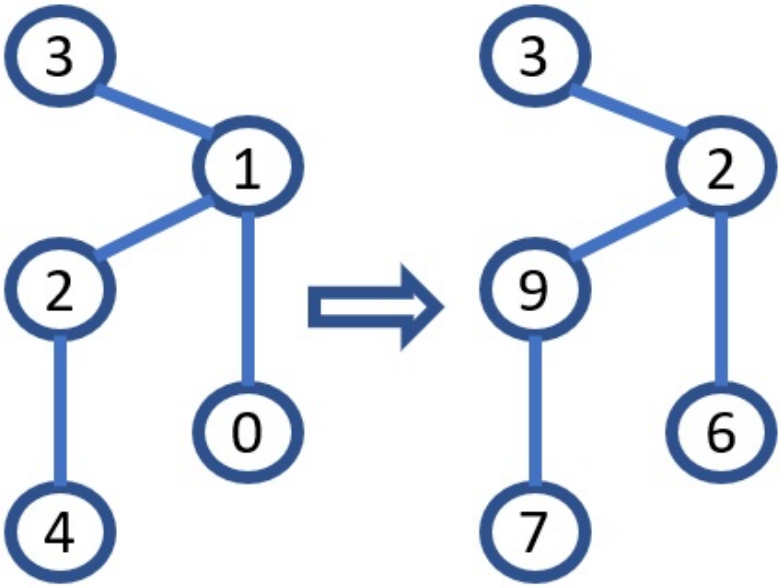
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

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

A fin de reportar 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]. En la siguiente figura, el panel izquierdo muestra los índices, y el panel derecho muestra las etiquetas asignadas a cada estación.



Asuma que las etiquetas han sido asignadas como se describe arriba y considere la siguiente llamada:

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

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

paquete (la cual tiene un índice de 1).

considere otra posible llamada:

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

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

Límites

- $1 \leq r \leq 10$

Por cada llamada a `label`:

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

Por cada llamada a `find_next_station`, asumiendo que d es el tamaño del arreglo c:

- $1 \leq d \leq n - 1$
- $0 \leq s, t \leq k$
- $s \neq t$
- $0 \leq c[i] \leq k$ (para cada $0 \leq i \leq d - 1$)
- $c[i - 1] < c[i]$ (para cada $1 \leq i \leq d - 1$)
- El tamaño total de todos los arreglos c pasados al procedimiento `find_next_station` no exceden $100\ 000$ para todos los escenarios combinados.

Subtareas

1. (5 puntos) $k = 1000$, ninguna estación tiene más de 2 vecinos.
2. (8 puntos) $k = 1000$, el enlace i conecta las estaciones $i + 1$ y $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 puntos) $k = 1\ 000\ 000$, a lo mucho una estación tiene mas 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 con máximo valor retornada por `label` a travez de todos los escenarios. Tu puntaje para esta subtarea es calculada de acuerdo a la siguiente tabla:

Máxima etiqueta	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 ejemplo

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

- línea 1: r

r blocks follow, each describing a single scenario. The format of each block is as follows:

- línea 1: n k
- línea $2 + i$ $(0 \leq i \leq n - 2)$: $u[i]$ $v[i]$
- línea $1 + n$: el número de llamadas a `find_next_station`, q.
- línea $2 + n + j$ $(0 \leq j \leq q - 1)$: $z[j]$ $y[j]$ $w[j]$: **índices** de estaciones involucradas en la j-ésima 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 que sigue $z[j]$ en el único camino de $z[j]$ a $y[j]$.

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

- línea 1: m

r bloques correspondientes a los escenarios consecutivos en la entrada siguen. El formato de cada bloque es como sigue:

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

Note que cada corrida de el evaluador de ejemplo llama a ambos `label` y `find_next_station`.