Estações (stations)

A "Singapore's Internet Backbone" (SIB) consiste em n estações, às quais são dados **índices** de 0 a $n-1$. Existem também $n-1$ ligações bidirecionais, numeradas de 0 a $n-2$. Cada ligação conecta duas estações distintas. Duas estações conectadas por uma única ligação são chamadas de vizinhas.

Um caminho da estação x para a estação y é uma sequência de estações distintas $a_0, a_1, \cdots, a_p$, tal que $a_0 = x$, $a_p = y$ e cada duas estações consecutivas no caminho são vizinhas. Existe **exatamente um** caminho de qualquer estação x para qualquer outra estação y.

Qualquer estação x pode criar um pacote (um pedaço de dados) e enviá-lo para qualquer outra estação y, que é chamada de **alvo** do pacote. Este pacote deve ser enviado através do caminho único desde x até y da seguinte maneira. Considera uma estação z que presentemente detém um pacote cuja estação alvo é y ($z \neq y$). Nesta situação, a estação z:

1. executa um **procedimento de roteamento** que determina o vizinho de z que está no caminho único de z para y, e
2. encaminha o pacote para esse vizinho.

Contudo, as estações têm memória limitada e não conseguem armazenar a lista inteira de ligações da SIB para usarem no seu procedimento de roteamento.

A tua tarefa é implementar o esquema de roteamento da SIB, que consiste em duas funções:

- A primeira função recebe como input n, a lista de ligações da SIB, e um inteiro $k \geq n-1$. Depois, atribui a cada estação um novo índice, uma **etiqueta** que deve ser um inteiro **único** entre 0 e k, inclusive.

- A segunda função é a que faz o roteamento, que é implantada em todas as estações depois de todas as etiquetas terem sido atribuídas. Recebe **apenas** os seguintes inputs:

  - s, a **etiqueta** da estação que presentemente detém um pacote,
  - t, a **etiqueta** da estação alvo do pacote ($t \neq s$),
  - c, a lista de **etiquetas** de todos os vizinhos de s.

  Deve devolver a **etiqueta** do vizinho de s para o qual o pacote deve ser encaminhado.

Adicionalmente, numa das subtarefas, a pontuação da tua solução depende do valor máximo de uma etiqueta atribuída a qualquer estação (no geral, um valor máximo menor é melhor).

Detalhes de implementação

Deves implementar as seguintes funções:

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

- n: número de estações na SIB.
- k: etiqueta máxima que pode ser usada.
- u e v: arrays de tamanho $n-1$ descrevendo as ligações. Para cada i ($0 \leq i \leq n-2$), a ligação i conecta as estações de indíces $u[i]$ e $v[i]$.
- Esta função deve devolver um único array L de tamanho n. Para cada i ($0 \leq i \leq n-1$) $L[i]$ é a etiqueta atribuída à estação com índice i. Todos os elementos do array L devem ser únicos e entre 0 e k, inclusive.

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

- s: etiqueta da estação que detém o pacote.
- t: etiqueta da estação alvo do pacote.
- c: um array dando uma lista das etiquetas de todos os vizinhos de s. O array c vem ordenado de forma crescente.
- Esta função deve devolver a etiqueta do vizinho de s para o qual o pacote deve ser encaminhado.

Cada caso de teste envolve um ou mais cenários independentes (isto é, diferentes descrições de uma SIB). Para um caso de teste envolvendo r cenários, o **programa** que chama as duas funções atrás descritas é executado exatamente duas vezes, da seguinte maneira.

Durante a primeira execução do programa:

- A função `label` é chamada r vezes,
- As etiquetas devolvidas são armazenadas pelo sistema de avaliação e,
- `find_next_station` não é chamada

Durante a segunda execução do progama:

- `find_next_station` pode ser chamada múltiplas vezes. Em cada chamada é escolhido um cenário **arbitrário** e as etiquetas devolvidas pela função `label` nesse cenário são usadas como inputs para as chamadas a `find_next_station`.
- `label` não é chamada.

Em particular, qualquer informação armazenada em variáveis estáticas ou globais na primeira execução do programa não estará disponível na função `find_next_station`.

Exemplo

Considera a seguinte chamada:

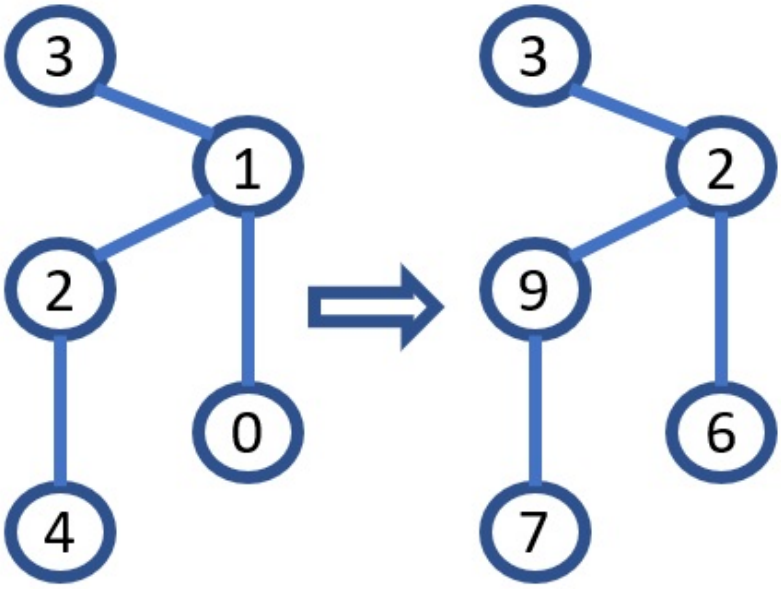
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Existe um total de 5 estações e 4 ligações conectando os pares de estações com índices $(0, 1)$, $(1, 2)$, $(1, 3)$ e $(2, 4)$. Cada etiqueta pode ser um inteiro entre 0 e $k = 10$.

Para reportar a seguinte atribuição de etiquetas:

Índice	Etiqueta
0	6
1	2
2	9
3	3
4	7

a função `label` deve devolver [6, 2, 9, 3, 7]. Os números da figura seguinte mostram os índices (lado esquerdo) e as etiquetas (lado direito).



Assumindo que as etiquetas foram atribuídas como atrás descrito, considera a seguinte chamada:

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

Isto significa que a estação detendo o pacote tem etiqueta 9 e que a estação alvo tem etiqueta 6. As etiquetas das estações no caminho para a estação alvo são $[9, 2, 6]$. Desse modo, a chamada deve devolver 2, que é a etiqueta da estação para a qual o pacote deve ser encaminhado (sendo que essa estação tem índice 1).

Considera outra possível chamada:

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

Esta função deve devolver 3, uma vez que a estação alvo com etiqueta 3 é vizinha da estação com etiqueta 2 e portanto deve receber o pacote de forma directa.

Restrições

- $1 \leq r \leq 10$

Para cada chamada a `label`:

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

Para cada chamada a `find_next_station`, o input vem de uma chamada prévia a `label` escolhida arbitrariamente. Considera as etiquetas aí atribuídas. Então:

- s e t são as etiquetas de duas estações diferentes.
- c é a sequência de todas as etiquetas dos vizinhos da estação com etiqueta s, por ordem crescente.

Para cada caso de teste, o tamanho total dos arrays c passados à função `find_next_station` não excede $100\ 000$, considerando todos os cenários.

Subtarefas

1. (5 pontos) $k = 1000$, nenhuma estação tem mais que 2 vizinhos.
2. (8 pontos) $k = 1000$, a ligação i conecta as estações $i + 1$ e $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 pontos) $k = 1\ 000\ 000$, no máximo uma estação tem mais que 2 vizinhos.
4. (10 pontos) $n \leq 8$, $k = 10^9$
5. (61 pontos) $k = 10^9$

Na subtarefa 5 podes obter pontuação parcial. Seja m a etiqueta máxima devolvida por `label` entre todos os cenários. A tua pontuação para esta subtarefa é calculada de acordo com a tabela seguinte:

Etiqueta máxima	Pontuação
$m \geq 10^9$	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

Avaliador exemplo

O avaliador exemplo lê o input no seguinte formato:

- linha 1: r

Seguem-se r blocos, cada um descrevendo um único cenário. O formato de cada bloco é o seguinte:

- linha 1: n k
- linha $2 + i$ ($0 \le i \le n - 2$): $u[i]$ $v[i]$
- linha $1 + n$: q, o número de chamadas a `find_next_station`.
- linha $2 + n + j$ ($0 \le j \le q - 1$): $z[j]$ $y[j]$ $w[j]$: os **índices** das estações envolvidas na j-ésima chamada a `find_next_station`. A estação $z[j]$ detém o pacote, a estação $y[j]$ é o alvo e a estação $w[j]$ é a estação para a qual o pacote deve ser encaminhado.

O avaliador exemplo escreve o output no seguinte formato:

- linha 1: m

Seguem-se r blocos correspondendo aos cenários consecutivos do input. O formato de cada bloco é o seguinte:

- linha $1 + j$ ($0 \le j \le q - 1$): **índice** da estação cuja **etiqueta** foi devolvida pela j-ésima chamada a `find_next_station` nesse cenário.

Nota que cada execução do avaliador exemplo chama ambas as funções `label` e `find_next_station`.