Estações (stations)

O Backbone de Internet de Singapura (BIS) consiste de n estações, as quais são atribuídos **índices** de 0 a $n-1$. Há também $n-1$ ligações bidirecionais, numeradss 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 todo par de estações consecutivas no caminho são vizinhas. Há **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, a qual é chamada de **destino** do pacote. Esse pacote deve ser encaminhado pelo único caminho de x a y como segue. Considere a estação z que atualmente detém o pacote, cuja estação destino é y ($z \neq y$). Nessa situação a estação z:

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

Contudo, estações possuem memória limitada e não armazenam a lista inteira das ligações em BIS para usar no procedimento routing.

Sua tarefa é implementar um esquema de routing para BIS, o qual consiste de dois procedimentos.

- Ao primeiro procedimento é dado n, a lista de ligações no BIS e um inteiro $k \geq n-1$ como entradas. Ele atribui a cada estação um **único** inteiro **rótulo** entre 0 e k, inclusive.

- O segundo procedimento é o procedimento routing, o qual é implantado em todas as estações após os rótulos serem atribuídos. A ele é dado **apenas** as seguintes entradas:

    - s, o **rótulo** da estação que atualmente detém um pacote,
    - t, o **rótulo** da estação de destino do pacote ($t \neq s$),
    - c, a lista dos **rótulos** de todos os vizinhos de s.

    Ele deve retornar o **rótulo** do vizinho de s para o qual o pacote deve ser encaminhado.

Em uma subtarefa, a pontuação de sua solução depende do valor máximo de um rótulo atribuído a alguma estação (em geral, menor é melhor).

Detalhes de Implementação

Você deve implementar os seguintes procedimentos:

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

- n: número de estações no BIS.
- k: rótulo máximo que pode ser usado.
- 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 com índices $u[i]$ e $v[i]$.
- Esse procedimento deve retornar um único array L de tamanho n. Para cada i ($0 \leq i \leq n-1$) $L[i]$ é o rótulo atribuído a 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: rótulo da estação que detém um pacote.
- t: rótulo da estação de destino do pacote.
- c: um array contendo a lista de rótulos de todos os vizinhos de s. O array c é ordenado em ordem crescente.
- Esse procedimento deve retornar o rótulo do vizinho de s ao qual o pacote deve ser encaminhado.

Cada caso de teste envolve um ou mais cenários independentes (i.e., descrições de diferentes BIS). Para um caso de teste envolvendo r cenários, um **programa** que chama o procedimento acima é rodado exatamente duas vezes, como segue.

Durante a primeira rodada do programa:

- procedimento `label` é chamado r vezes,
- os rótulos retornados são armazenados pelo sistema de correção, e
- `find_next_station` não é chamado.

Durante a segunda rodada do programa:

- `find_next_station` pode ser chamado muitas vezes. Em cada chamada, um cenário **arbitrário** é escolhido, e os rótulos retornados pela chamada ao procedimento `label` nesse cenário são usados como entradas para `find_next_station`.
- `label` não é chamado.

Em particular, qualquer informação salva em variáveis estáticas ou globais na primeira rodada do programa não está disponível dentro do procedimento `find_next_station`.

Exemplo

Considere a seguinte chamada:

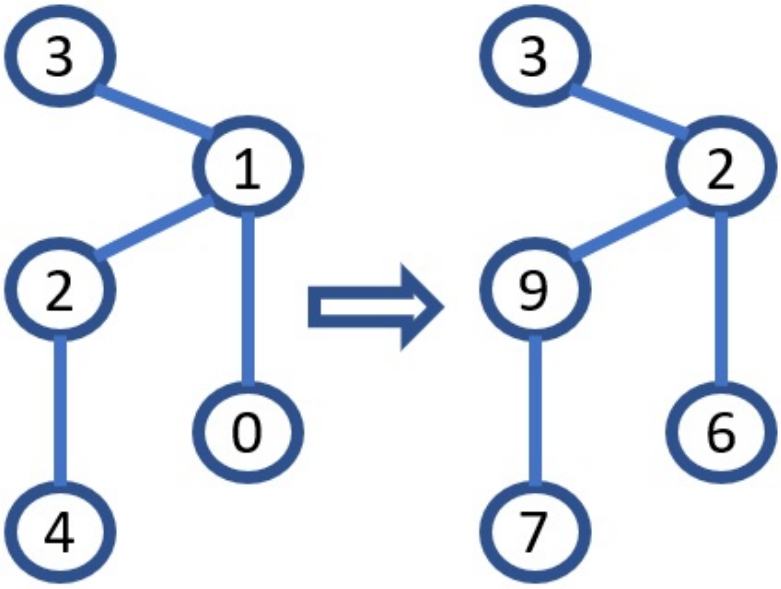
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Há um total de 5 estaçõea, e 4 ligações conectando pares de estações com índices $(0, 1)$, $(1, 2)$, $(1, 3)$ e $(2, 4)$. Cada rótulo pode ser um inteiro de 0 a $k = 10$.

Para reportar a seguinte atribuição de rótulos:

Índice	Rótulo
0	6
1	2
2	9
3	3
4	7

o procedimento `label` deve retornar [6, 2, 9, 3, 7]. Os números na figura a seguir mostram os índices (painel esquerdo) e rótulos atribuídos (painel direito).



Assuma que os rótulos tenham sido atribuídos como descritos acima e considere a seguinte chamada:

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

Isso significa que a estação que detém o pacote possui rótulo 9, e a estação de destino possui rótulo 6. Os rótulos das estações no caminho para a estação de destino são $[9, 2, 6]$. Então, a chamada deve retornar 2, que é o rótulo da estação que o pacote deve ser encaminhado (o qual possui índice 1).

Considere outra possível chamada:

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

O procedimento deve retornar 3, pois a estação de destino com rótulo 3 é vizinha da estação com rótulo 2, e então deve receber o pacote diretamente.

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 $0 \leq i \leq n - 2$)

Para cada chamada a `find_next_station`, a entrada vem de uma chamada prévia a `label`, arbitrariamente escolhida. Considere os rótulos que ela produziu. Então:

- s e t são rótulos de duas estações diferentes.
- c é a sequência de todos os rótulos de vizinhos da estação com rótulo s, em ordem crescente.

Para cada caso de teste, o tamanho total de todos os arrays c passados para o procedimento `find_next_station` não excede $100\ 000$ para todos os cenários combinados.

Subtarefas

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

Na subtarefa 5 você pode obter uma pontuação parcial. Seja m o valor de rótulo máximo retornado por `label` entre todos os cenários. Sua pontuação para essa subtarefa é calculada de acordo com a seguinte tabela:

Rótulo máximo	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

Corretor exemplo

O corretor exemplo lê da entrada no seguinte formato:

- linha 1: r

seguem 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 $2 + n + j$ $(0 \le j \le q - 1)$: $z[j]$ $y[j]$ $w[j]$: **índices** de estações envolvidas na j-ésima chamada a `find_next_station`. A estação $z[j]$ detém o pacote, a estação $y[j]$ é o destino do pacote e a estação $w[j]$ é a estação para a qual o pacote deve ser encaminhado.

O corretor padrão escreve o resultado no seguinte formato:

- linha 1: m

seguem r blocos correspondentes aos cenários consecutivos. O formato de cada bloco é o seguinte:

- linha $1 + j$ $(0 \le j \le q - 1)$: **índice** da estação, cujo **rótulo** foi retornado pela j-ésima chamada a `find_next_station` neste cenário.

Note que cada execução do corretor exemplo chama ambos os procedimentos `label` e `find_next_station`.