Stanice (stations)

Singapurské metro má n stanic, které jsou **očíslovány** od 0 do $n-1$. Mezi nimi vede $n-1$ obousměrných kolejí očíslovaných od 0 do $n-2$. Každá kolej spojuje dvě různé stanice a stanicím, které jsou přímo spojeny kolejemi, říkáme sousední.

Cesta ze stanice x do stanice y je posloupnost navzájem různých stanic $a_0, a_1, \cdots, a_p$ taková, že $a_0 = x$, $a_p = y$ a každé dvě v ní po sobě následující stanice jsou sousední. V Singapurském metru vede **právě jedna** cesta mezi každými dvěma stanicemi.

Vlak ze stanice x do cílové stanice y musí jet po této jednoznačné cestě. Strojvedoucí si ale cestu nepamatují, proto jim na každé stanici musí napovídat výpravčí. Tedy je-li právě ve stanici z vlak jedoucí do cílové stanice y ($z \neq y$), pak

1. výpravčí určí sousední stanici na jednoznačné cestě ze stanice z do stanice y, poradí ji strojvedoucímu a
2. vlak přejede do této sousední stanice.

Výpravčí ale také maji jen omezenou paměť a nemohou si tedy pamatovat celou síť metra. Napište program, který jim pomůže. Tento program musí implementovat dvě funkce:

- První z nich dostane zadáno n, seznam kolejí a celé číslo $k \geq n-1$. Tato funkce musí každé stanici přiřadit jednoznačný **identifikátor**, což je celé číslo mezi 0 a k (včetně).
- Druhá funkce simuluje práci výpravčího a dostane **pouze** následující vstupy:
  - s: **identifikátor** aktuální stanice,
  - t: **identifikátor** cílové stanice ($t \neq s$),
  - c: seznam **identifikátorů** všech stanic sousedících se stanicí s.

  Tato funkce musí vrátit **identifikátor** souseda s na cestě z s do t.

V jedné z podúloh bude vaše skóre záviset i na velikosti použitých identifikátorů (čím menší, tím lepší).

Implementační detaily

Implementujte následující funkce:

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

- n: počet stanic.
- k: největší možný identifikátor.
- u a v: pole velikosti $n-1$ popisující koleje. Pro každé i ($0 \leq i \leq n-2$), kolej číslo i spojuje stanice číslo $u[i]$ a $v[i]$.
- Tato funkce musí vrátit pole L velikosti n. Pro každé i ($0 \leq i \leq n-1$), $L[i]$ je identifikátor přiřazený stanici číslo i. Prvky pole L musí být navzájem různé a v rozmezí od 0 do k (včetně).

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

- s: identifikátor aktuální stanice.
- t: identifikátor cílové stanice.
- c: pole obsahující identifikátory stanic sousedících se stanicí s. Pole c je setříděné v rostoucím pořadí.
- Tato funkce musí vrátit identifikátor stanice sousedící se stanicí s na cestě ze stanice s do stanice t.

Každý test se skládá z jednoho či více scénářů s různými sítěmi metra. Pro test skládající se z r scénářů bude vyhodnocovač volán právě dvakrát.

V prvním volání vyhodnocovače:

- funkce `label` je volána r-krát,
- vyhodnocovací systém uloží navrácené identifikátory a
- funkce `find_next_station` není volána.

V druhém volání vyhodnocovače:

- funkce `find_next_station` může být volána libovolněkrát. V každém volání zvlášť se vybere **libovolný** z r scénářů a jako vstupy funkce `find_next_station` jsou použity identifikátory vrácené funkcí `label` pro tento scénář.
- Funkce `label` není volána.

Z toho plyne, že ve druhém volání vyhodnocovače nejsou k dispozici hodnoty globálních či statických proměnných z prvního volání.

Příklad

Uvažujme následující volání:

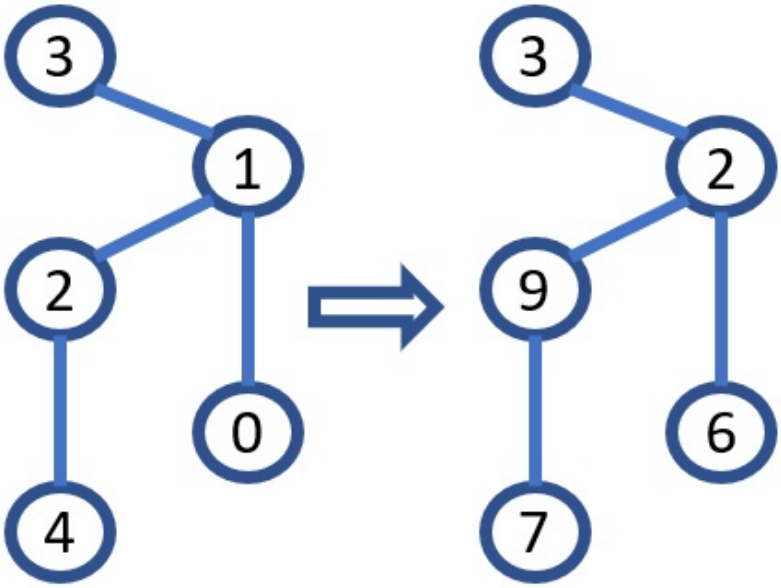
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Mezi 5 stanicemi vedou 4 koleje, spojující dvojice stanic $(0, 1)$, $(1, 2)$, $(1, 3)$ a $(2, 4)$. Jako identifikátory lze použít celá čísla mezi 0 a $k = 10$.

Aby nahlásila následující přiřazení identifikátorů:

Číslo stanice	Identifikátor
0	6
1	2
2	9
3	3
4	7

funkce `label` vrátí pole [6, 2, 9, 3, 7]. Na následujícím obrázku jsou v levé části čísla stanic, v pravé části odpovídající identifikátory.



Nechť je pro výše popsané přiřazení identifikátorů volána funkce `find_next_station` s následujícími parametry:

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

Jedeme tedy ze stanice s identifikátorem 9 do cílové stanice s identifikátorem 6. Identifikátory na cestě mezi těmito dvěma stanicemi jsou $[9, 2, 6]$. Musíme tedy vrátit 2, což je identifikátor následující stanice na cestě (její číslo je 1).

Pro volání

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

funkce musí vrátit 3, jelikož cílová stanice s identifikátorem 3 sousedí s aktuální stanicí s identifikátorem 2.

Omezení

- $1 \leq r \leq 10$

V každém volání funkce `label`:

- $2 \leq n \leq 1000$
- $k \geq n - 1$
- $0 \leq u[i], v[i] \leq n - 1$ (pro všechna i tž. $0 \leq i \leq n - 2$)

V každém volání funkce `find_next_station` je vstup zvolen na základě libovolného z předchozích volání funkce `label`.

- s a t jsou identifikátory navzájem různých stanic.
- c je posloupnost identifikátorů stanic sousedících se stanicí s identifikátorem s, v rostoucím pořadí.

Pro každý test platí, že součet délek všech polí c, pro nějž je v rámci tohoto testu volána funkce `find_next_station`, nepřesahuje $100\ 000$.

Podúlohy

1. (5 bodů) $k = 1000$ a každá stanice má nejvýše 2 sousedy.
2. (8 bodů) $k = 1000$ a pro každé i tž. $0 \leq i \leq n - 2$, kolej číslo i spojuje stanice číslo $i + 1$ a $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 bodů) $k = 1\ 000\ 000$ a nejvýše jedna stanice má více než 2 sousedy.
4. (10 bodů) $n \leq 8$, $k = 10^9$
5. (61 bodů) $k = 10^9$

V podúloze 5 můžete získat i část z bodů. Nechť m je maximum z identifikátorů vrácených funkcí `label` ve všech scénářích z této podúlohy. Vaše skóre bude:

Maximum z identifikátorů	Počet bodů
$m \geq 10^9$	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

Ukázkový vyhodnocovač

Ukázkový vyhodnocovač načítá vstup v následujícím formátu:

- řádek 1: r

Následuje r bloků, z nichž každý popisuje jeden scénář. Formát každého bloku je následující:

- řádek 1: n k
- řádek $2 + i$ ($0 \leq i \leq n - 2$): $u[i]$ $v[i]$
- řádek $1 + n$: q: počet volání funkce `find_next_station`.
- řádek $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j]$ $y[j]$ $w[j]$: **čísla** stanic pro j-té volání funkce `find_next_station`. Aktuální stanice je číslo $z[j]$, cílová stanice je číslo $y[j]$ a následující stanice na cestě z aktuální do cílové stanice má číslo $w[j]$.

Ukázkový vyhodnocovač vypisuje výstup v následujícím formátu:

- řádek 1: m

Následuje r bloků odpovídajících jednotlivým scénářům. Formát každého bloku je následující:

- řádek $1 + j$ ($0 \leq j \leq q - 1$): **číslo** stanice, jejíž **identifikátor** byl vrácen j-tým voláním funkce `find_next_station` v tomto scénáři.

Upozorňujeme, že na rozdíl od skutečného vyhodnocovače proběhnou všechna volání funkcí `label` a `find_next_station` v rámci jednoho spuštění programu.