Stationen (stations)

Singapurs Internet-Backbone (SIB) besteht aus n Stationen mit zugewiesenen **Indizes** von 0 bis $n-1$. Außerdem gibt es $n-1$ ungerichtete Verbindungen mit Nummern von 0 bis $n-2$. Jede davon verbindet zwei verschiedene Stationen. Zwei auf diese Weise verbundene Stationen nennen wir Nachbarn.

Ein Pfad von Station x zu Station y ist eine Folge verschiedener Stationen $a_0, a_1, \cdots, a_p$, sodass $a_0 = x$ und $a_p = y$ gilt und zwei aufeinanderfolgende Stationen in dem Pfad stets Nachbarn sind. Es gibt **genau einen** Pfad von jeder Station x zu jeder anderen Station y.

Jede Station x kann ein (Daten-)Paket erzeugen und an eine andere Station y senden, welche wir das **Ziel** des Pakets nennen. Dieses Paket muss entlang des eindeutigen Pfades von x nach y wie folgt geleitet ("*geroutet*") werden. Betrachte eine Station z, die gerade ein Paket besitzt, dessen Zielstation y ($z \neq y$) ist. In dieser Situation tut Station z das Folgende:

1. Sie führt eine **Routing-Funktion** aus, die den Nachbarn von z bestimmt, der auf dem eindeutigen Pfad von z nach y liegt, und
2. leitet das Paket an diesen Nachbarn weiter.

Allerdings haben Stationen beschränkten Speicher und speichern nicht die gesamte Liste aller Verbindungen im SIB ab, um sie für die Routing-Funktion zu benutzen.

Deine Aufgabe ist es, ein Routing-Verfahren zu implementieren, das aus den folgenden zwei Funktionen besteht.

- Der ersten Funktion werden n, die Liste mit den Verbindungen im SIB und eine Ganzzahl $k \geq n-1$ als Eingabe gegeben. Sie weist jeder Station ein **eindeutiges** Ganzzahl-**Label** zwischen 0 und k, inklusive zu.

- Die zweite Funktion ist die Routing-Funktion, die an alle Stationen ausgeliefert wird, sobald die Labels zugewiesen sind. Ihr werden **nur** die folgenden Eingaben gegeben:

  - s, das **Label** der Station, das gerade ein Paket besitzt,
  - t, das **Label** der Zielstation des Pakets ($t \neq s$),
  - c, die Liste der **Label** aller Nachbarn von s.

  Sie soll das **Label** des Nachbarn von s zurückgeben, an den das Paket weitergeleitet werden soll.

In einer der Teilaufgaben hängt die Punktzahl Deiner Lösung vom Wert des maximalen einer Station zugewiesenen Labels ab (grundsätzlich ist kleiner besser).

Implementierungsdetails

Du sollst die folgenden Funktionen implementieren:

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

- n: die Anzahl der Stationen im SIB.
- k: das maximale Label, das benutzt werden kann.
- u und v: Arrays der Größe $n-1$, die die Verbindungen beschreiben. Für jedes i ($0 \leq i \leq n-2$) besteht die Verbindung i zwischen den Stationen mit den Indizes $u[i]$ und $v[i]$.
- Diese Funktion soll ein einziges Array L der Größe n zurückgeben. Für jedes i ($0 \leq i \leq n-1$) ist $L[i]$ das Label, das der Station mit Index i zugewiesen wird. Die Einträge des Arrays L müssen verschieden sein und zwischen 0 und k, inklusive liegen.

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

- s: das Label der Station, die ein Paket besitzt.
- t: das Label der Zielstation des Pakets.
- c: ein Array, das die Liste der Labels aller Nachbarn von s angibt. Das Array c ist aufsteigend sortiert.
- Diese Funktion soll das Label eines Nachbarn von s zurückgeben, an den das Paket weitergeleitet werden soll.

Jeder Testfall umfasst ein oder mehrere Szenarios (d.h. Beschreibungen verschiedener SIBs). Für einen r Szenarios umfassenden Testfall wird ein **Programm**, das die obigen Funktionen aufruft, genau zwei Mal ausgeführt, wie im Folgenden beschrieben.

Während der ersten Ausführung des Programms:

- Die `label`-Funktion wird r Mal aufgerufen,
- die zurückgegebenen Labels werden vom Grading-System gespeichert und
- `find_next_station` wird nicht aufgerufen.

Während der zweiten Ausführung des Programms:

- `find_next_station` kann mehrmals aufgerufen werden. Bei jedem Aufruf wird ein **beliebiges** Szenario ausgewählt und die Labels, die vom Aufruf der `label`-Funktion in diesem Szenario zurückgegeben werden, werden als Eingabe für `find_next_station` verwendet.
- `label` wird nicht aufgerufen.

Insbesondere ist jegliche Information, die bei der ersten Ausführung des Programms in statischen oder globalen Variablen gespeichert wird, für die `find_next_station`-Funktion nicht verfügbar.

Beispiel

Betrachte den folgenden Aufruf:

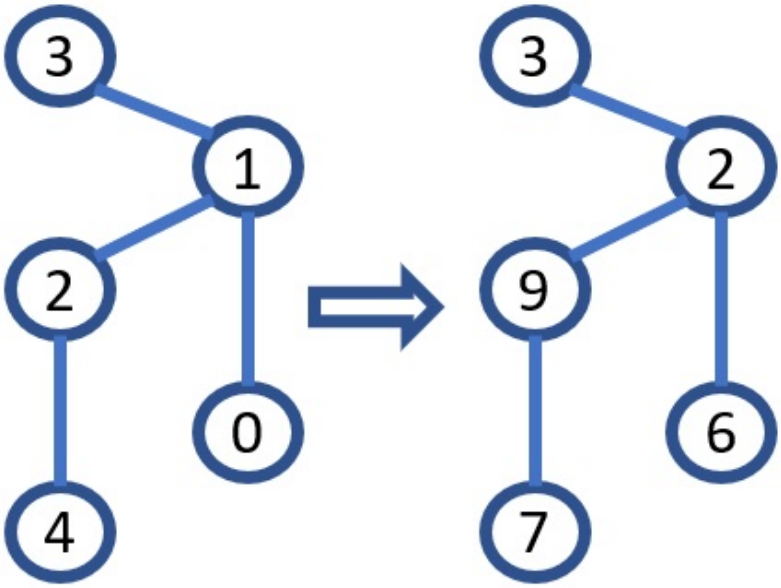
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Es gibt insgesamt 5 Stationen und 4 Verbindungen unter den Paaren von Stationen mit den Indizes $(0, 1)$, $(1, 2)$, $(1, 3)$ und $(2, 4)$. Jedes Label kann eine Ganzzahl von 0 bis $k = 10$ sein.

Um das folgende Labelling anzugeben:

Index	Label
0	6
1	2
2	9
3	3
4	7

soll die `label`-Funktion das Array $[6, 2, 9, 3, 7]$ zurückgeben. Die Zahlen in der folgenden Abbildung zeigen die Indizes (linkes Bild) und die zugewiesenen Label (rechtes Bild).



Nehmen wir an, dass die Label wie oben beschrieben zugewiesen wurden. Betrachte den folgenden Aufruf:

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

Das bedeutet, dass die Station, die das Paket besitzt, das Label 9 und die Zielstation das Label 6 hat. Die Label der Stationen auf dem Pfad zur Zielstation sind $[9, 2, 6]$. Daher soll der Aufruf 2 zurückgeben, was das Label der Station ist, an die das Paket weitergeleitet werden soll (welche

Index 1 hat).

Betrachte einen anderen möglichen Aufruf:

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

Die Funktion soll 3 zurückgeben, da die Zielstation mit Label 3 ein Nachbar der Station mit Label 2 ist und das Paket daher direkt erhalten soll.

Beschränkungen

- $1 \leq r \leq 10$

Für jeden Aufruf von `label`:

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

Für jeden Aufruf von `find_next_station` stammt die Eingabe von einem beliebig gewählten vorherigen Aufruf von `label`. Betrachte die Labels, die dieser erzeugt hat. Dann gilt:

- s und t sind Labels zweier verschiedener Stationen.
- c ist eine Folge aller Labels von Nachbarn der Station mit Label s in aufsteigender Reihenfolge.

Für jeden Testfall ist die Gesamtlänge aller Arrays c, die an `find_next_station` übergeben werden, nicht größer als $100\ 000$ für alle Szenarios zusammen.

Teilaufgaben

1. (5 points) $k = 1000$, keine Station hat mehr als 2 Nachbarn.
2. (8 points) $k = 1000$, Verbindung i verbindet $i + 1$ und $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 points) $k = 1\ 000\ 000$, höchstens eine Station hat mehr als 2 Nachbarn.
4. (10 points) $n \leq 8$, $k = 10^9$
5. (61 points) $k = 10^9$

In Teilaufgabe 5 kannst Du Teilpunkte erzielen. Sei m der maximale Wert aller von `label` zurückgegebenen Label unter allen Szenarien. Deine Punktzahl für diese Teilaufgabe wird nach der folgenden Tabelle berechnet:

Maximales Label	Punktzahl
$m \geq 10^9$	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

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

- Zeile 1: r

r Abschnitte folgen, von denen jeder ein einzelnes Szenario beschreibt. Das Format jedes Blocks ist wie folgt:

- Zeile 1: n k
- Zeile $2 + i$ ($0 \leq i \leq n - 2$): $u[i]$ $v[i]$
- Zeile $1 + n$: q: die Anzahl der Aufrufe von `find_next_station`.
- Zeile $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j]$ $y[j]$ $w[j]$: **Indizes** der Stationen, die Teil des $j$ten Aufrufs von `find_next_station` sind. Die Station $z[j]$ besitzt das Paket, die Station $y[j]$ ist das Paketziel und die Station $w[j]$ ist die Station, an die das Paket weitergeleitet werden soll.

Der Beispiel-Grader gibt das Ergebnis im folgenden Format aus:

- Zeile 1: m

r Abschnitte folgen, die zu den aufeinanderfolgenden Szenarien der Eingabe gehören. Das Format jedes Abschnitts ist wie folgt:

- Zeile $1 + j$ ($0 \leq j \leq q - 1$): **Index** der Station, deren **Label** vom $j$ten Aufruf von `find_next_station` in diesem Szenario zurückgegeben wurde.

Beachte, dass jede Ausführung des Beispiel-Graders sowohl `label` als auch `find_next_station` aufruft.