Stations (stations)

Singapore's Internet Backbone (SIB) bestaat uit n locaties, die allemaal een **index** hebben van 0 tot en met $n-1$. Er zijn ook $n-1$ bidirectionele links, genummerd van 0 tot en met $n-2$. Elke link verbindt twee aparte locaties. Twee locaties die verbonden zijn door een link noemen we buren.

Een pad van locatie x naar locatie y is een rij verschillende locaties $a_0, a_1, \cdots, a_p$, zodat $a_0 = x$, $a_p = y$, en iedere twee opvolgende locaties zijn buren. Er is **precies één** pad van iedere locatie x naar iedere andere locatie y.

Elke locatie x kan een pakket maken (een stukje data) en het naar iedere andere locatie y sturen, wat we het **doel** van het pakket noemen. Dit pakket wordt als volgt langs het unieke pad van x naar y gestuurd. Stel dat locatie z een pakket heeft met doel y ($z \neq y$). De locatie z doet dan het volgende:

1. Voert een **routefunctie** uit die bepaalt welke buurman van z op het unieke pad van z naar y ligt.
2. Stuurt het pakket door naar deze buurman.

Maar, de locaties hebben maar een gelimiteerde hoeveelheid geheugen en kunnen niet de hele lijst met links in SIB opslaan om te gebruiken in de routefunctie.

Jij moet een routestelsel voor SIB implementeren, dat uit twee functies bestaat:

- De eerste functie ontvangt n, een lijst van de links in SIB en een integer $k \geq n-1$. Die wijst iedere locatie een **uniek** integer **label** toe van 0 tot en met k.
- De tweede functie is de routefunctie, die wordt uitgerold op alle locaties nadat de labels toegewezen zijn. De routefunctie ontvangt **alleen** de volgende waarden:
  - s, het **label** van de locatie waar het pakket momenteel is.
  - t, het **label** van het doel van het pakket ($t \neq s$).
  - c, de lijst van **labels** van alle buren van s.

Het moet het **label** teruggeven van een buurman van s waar het pakket naar verzonden moet worden.

In een subtask hangt de score van je oplossing af van de waarde van het maximale label dat is toegewezen aan de locaties (lager is beter).

Implementatiedetails

Je moet de volgende functies implementeren:

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

- n: het aantal locaties in het SIB.
- k: het maximale label dat kan worden gebruikt.
- u en v: arrays van lengte $n-1$ die de links beschrijven. Voor iedere i ($0 \leq i \leq n-2$), verbindt link i locaties met indices $u[i]$ en $v[i]$.
- Deze functie moet een array L van lengte n teruggeven. Voor iedere i ($0 \leq i \leq n-1$) is $L[i]$ het label toegewezen aan de locatie met index i. Alle waarden in array L moeten uniek en van 0 tot en met k zijn.

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

- s: het label van de locatie waar het pakket is.
- t: het label van de doellocatie van het pakket.
- c: een array van labels van alle buren van s. De array c is oplopend gesorteerd.
- Deze functie moet het label van een buurman van s teruggeven, waar het pakket naar doorgestuurd moet worden.

Ieder testgeval gaat over één of meedere onafhankelijke scenario's (dus andere SIB omschrijvingen). Voor een testgeval met r scenario's draait een **programma** precies twee maal dat de bovenstaande functies als volgt aanroept.

Tijdens de eerste keer dat het programma draait:

- Wordt `label` r keer aangeroepen.
- Worden de teruggegeven labels opgeslagen in het gradersysteem.
- `find_next_station` wordt niet aangeroepen.

De tweede keer dat het programma draait:

- Kan `find_next_station` meerdere keren aangeroepen worden. Iedere keer wordt een **willekeurig** scenario gekozen, en de labels teruggegeven van de `label` functie voor dat scenario worden gebruikt als invoer voor `find_next_station`.
- `label` wordt niet aangeroepen.

In het bijzonder is informatie, opgeslagen in static of global variabelen tijdens de eerste keer dat het programma draaide, niet beschikbaar in de `find_next_station` functie.

Voorbeeld

Neem de volgende aanroep:

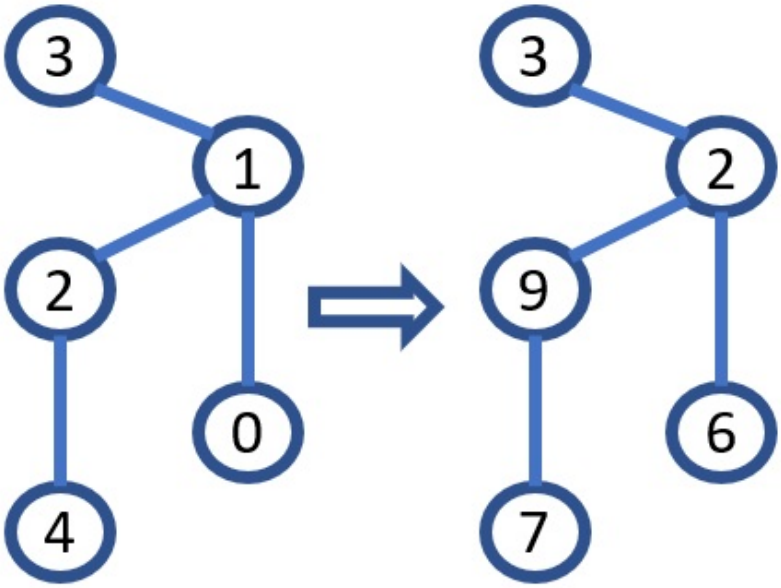
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Er zijn in totaal 5 locaties, en 4 links tussen paren van locaties met indices $(0, 1)$, $(1, 2)$, $(1, 3)$ en $(2, 4)$. Ieder label kan een integer van 0 tot en met $k = 10$ zijn.

Om de volgende labels aan te geven:

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

moet de `label` functie $[6, 2, 9, 3, 7]$ teruggeven. In de volgende afbeelding laat de linkerkant de indices zien, en de rechterkant de labels die zijn aangewezen aan elke locatie.



Neem aan dat de labels aangewezen zijn zoals boven omschreven en beschouw de volgende aanroep:

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

Dit betekent dat het pakket op de locatie met label 9 is, en de doellocatie het label 6 heeft. De labels van de locaties op het pad naar het doel zijn $[9, 2, 6]$.

De functie moet dus 2 teruggeven, wat het label is van de locatie waar het pakket naar verzonden moet worden (die index 1 heeft).

Neem de volgende andere aanroep:

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

De functie moet 3 teruggeven, omdat de doellocatie met label 3 een buurman is van de locatie met label 2, en daarom het pakket direct ontvangt.

Randvoorwaarden

- $1 \leq r \leq 10$

Voor iedere aanroep van `label`:

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

Voor iedere aanroep van `find_next_station`, komt de invoer uit een willekeurig gekozen eerdere aanroep naar `label`. Beschouw de labels uit dit scenario. Dan:

- s en t zijn labels van twee verschillende locaties.
- c is de rij van labels van buren van de locatie met label s, in oplopende volgorde.

Voor elke testcase, is de totale lengte van alle arrays c die aan `find_next_station` worden gegeven niet meer dan $100\ 000$ voor alle scenario's samen.

Subtasks

1. (5 punten) $k = 1000$, geen locatie heeft meer dan 2 buren.
2. (8 punten) $k = 1000$, link i verbindt locatie $i + 1$ en $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 punten) $k = 1\ 000\ 000$, maximaal één locatie heeft meer dan 2 buren.
4. (10 punten) $n \leq 8$, $k = 10^9$
5. (61 punten) $k = 10^9$

In subtask 5 kan je een deelscore krijgen. Neem m als het maximale label teruggegeven van de functie `label` in alle scenarios. Je score voor deze subtask wordt berekend volgens de volgende tabel:

Maximale label	Score
$m \geq 10^9$	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

Voorbeeldgrader

De voorbeeldgrader leest de invoer in het volgende formaat:

- regel 1: r

Hierna volgen r blokken die ieder een scenario omschrijven. Het formaat van ieder blok is als volgt:

- regel 1: n k
- regel $2 + i$ ($0 \leq i \leq n - 2$): $u[i]$ $v[i]$
- regel $1 + n$: q:
  het aantal aanroepen van `find_next_station`
- regel $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j]$ $y[j]$ $w[j]$: **indices** van locaties die te maken hebben met de j-de aanroep van `find_next_station`: de locatie $z[j]$ heeft het pakket, de locatie $y[j]$ is het doel, en locatie $w[j]$ is de locatie die $z[j]$ volgt op het unieke pad van $z[j]$ naar $y[j]$.

De voorbeeldgrader schrijft de resultaten in het volgende formaat weg:

- regel 1: m

r blokken die overeenkomen met de scenario's in de invoer volgen. Het formaat van ieder blok is als volgt:

- regel $1 + j$ ($0 \leq j \leq q - 1$): **index** van de locatie, wiens **label** teruggeven was bij de j-de aanroep van `find_next_station` in dit scenario.

Let op dat iedere keer dat de voorbeeldgrader draait, de functies `label` en `find_next_station` allebei aangeroepen worden.