Stazioni di routing (stations)

Il *Singapore's Internet Backbone* (SIB) consiste di n stazioni (numerate da 0 a $n-1$) collegate ad albero da $n-1$ *link* bidirezionali (numerati da 0 a $n-2$). Due stazioni collegate da un link sono dette *vicine*. Il percorso da x a y è l'**unica** sequenza di stazioni distinte che sono vicine a coppie e che inizia con x e finisce con y.

Una stazione x può creare un pacchetto e mandarlo ad un'altra stazione y detta *target*. Per fare il *routing* di questo pacchetto sull'unico percorso tra x e y, ogni volta che il pacchetto si trova in una stazione z ($z \neq y$), la stazione z esegue una *procedura di routing* per determinare quale dei suoi vicini è sull'unico percorso tra z e y, e poi gli inoltra il pacchetto. Purtroppo, la memoria delle stazioni del SIB è molto limitata! Devi quindi implementare un sistema di routing in due fasi:

1. Dati n, la lista dei link del SIB e un intero $k \geq n-1$, una funzione `label` deve assegnare ad ogni stazione un'**unica** etichetta intera a scelta tra 0 e k (inclusi).

2. Poi, la procedura di routing `find_next_station` avrà a disposizione **solamente**:

   - s, l'etichetta della stazione corrente,
   - t, l'etichetta della stazione target ($t \neq s$),
   - c, la lista delle etichette dei vicini di s,

   per calcolare l'etichetta del vicino di s a cui il pacchetto deve essere inoltrato. In particolare, durante la procedura la lista degli archi del SIB **non** è disponibile.

In un subtask, il punteggio della tua soluzione dipenderà anche dalla massima etichetta assegnata ad una qualche stazione (quindi meglio se usi etichette più piccole).

Note di implementazione

Devi implementare le seguenti funzioni:

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

- n: il numero di stazioni nel SIB.
- k: la massima etichetta che può essere usata.
- u e v: array di lunghezza $n-1$ per cui l'i-esimo link ($i = 0 \ldots n-2$) collega $u[i]$ e $v[i]$.
- La funzione deve restituire un singolo array L di lunghezza n, contenente le etichette $L[i]$ assegnate alle stazioni $i = 0 \ldots n-1$. Gli elementi di L devono essere tutti distinti e compresi tra 0 e k inclusi.

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

- s: l'etichetta della stazione in cui si trova il pacchetto.
- t: l'etichetta della stazione target del pacchetto.
- c: un array con la lista delle etichette dei vicini di s, in ordine crescente.
- La funzione deve restituire l'etichetta del vicino di s a cui il pacchetto deve essere inoltrato.

Ogni caso di test è composto da r scenari indipendenti, e la valutazione viene effettuata in due esecuzioni distinte. Durante la prima esecuzione:

- la funzione `label` è chiamata r volte, mentre `find_next_station` non viene chiamata;
- le etichette restituite sono memorizzate dal grader.

Durante la seconda esecuzione:

- `label` non viene chiamata, mentre `find_next_station` viene chiamata più volte, ogni volta rispetto a uno **qualunque** degli r scenari, usando le etichette calcolate durante la prima esecuzione.

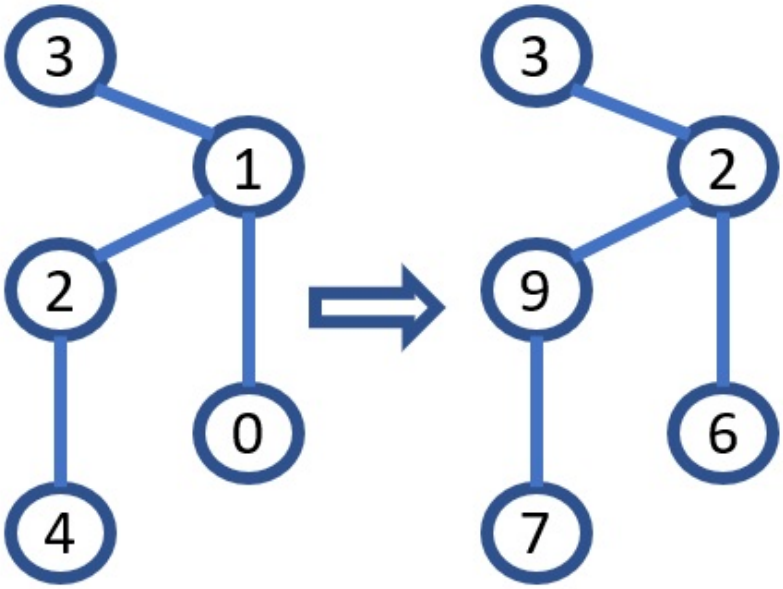
Le informazioni salvate nella prima esecuzione **non** sono disponibili durante la seconda esecuzione.

Esempio

Considera la seguente chiamata:

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

Ci sono 5 stazioni e 4 link: $(0, 1)$, $(1, 2)$, $(1, 3)$, $(2, 4)$. Le etichette devono essere comprese tra 0 e $k = 10$. Per restituire la seguente etichettatura, la funzione `label` deve restituire $[6, 2, 9, 3, 7]$.



Indice	Etichetta
0	6
1	2
2	9
3	3
4	7

Durante la seconda esecuzione, consideriamo la seguente chiamata:

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

Questo significa che nella stazione di etichetta 9 si trova un pacchetto destinato alla stazione di etichetta 6. Dato che le etichette delle stazioni nel percorso sono $[9, 2, 6]$, la funzione deve restituire 2 (corrispondente all'indice 1).

Considerando poi la seguente chiamata:

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

La funzione deve restituire 3, dato che il target di etichetta 3 è un vicino della stazione di etichetta 2 e può ricevere il pacchetto direttamente.

Assunzioni

- $1 \leq r \leq 10$

Per ogni chiamata a `label`:

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

Per ogni chiamata a `find_next_station`, l'input è generato a partire dalle etichette prodotte da un'arbitraria chiamata precedente a `label`, e:

- s e t sono le etichette di due diverse stazioni.
- c è la sequenza di tutte le etichette dei vicini della stazione di etichetta s, in ordine crescente.

In ogni caso di test, la somma delle lunghezze degli array c passati alle chiamate a `find_next_station` non supera $100\ 000$ in totale per tutti gli scenari.

Subtask

1. (5 punti) $k = 1000$, nessuna stazione ha più di 2 vicini.
2. (8 punti) $k = 1000$, e il link i collega le stazioni $i + 1$ e $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 punti) $k = 1\ 000\ 000$, e al massimo una stazione ha più di 2 vicini.
4. (10 punti) $n \leq 8$, $k = 10^9$
5. (61 punti) $k = 10^9$

Unicamente nel subtask 5, il tuo punteggio ottenuto dipenderà dalla massima etichetta m prodotta da `label` tra tutti gli scenari di tutti i test case, secondo la seguente tabella:

Massima etichetta	Punteggio
$m \geq 10^9$	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

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

- riga 1: r

Seguono r blocchi di righe che descrivono uno scenario ciascuno, secondo il seguente formato:

- riga 1: n k
- righe $2 + i$ $(0 \leq i \leq n - 2)$: $u[i]$ $v[i]$
- riga $1 + n$: q: il numero di chiamate a `find_next_station`.
- righe $2 + n + j$ $(0 \leq j \leq q - 1)$: $z[j]$ $y[j]$ $w[j]$: gli **indici** delle stazioni corrispondenti alla j-esima chiamata a `find_next_station`. Il pacchetto si trova in $z[j]$, ha target $y[j]$, e $w[j]$ è la stazione a cui dovrebbe essere inoltrato il pacchetto.

Il grader di esempio stampa l'output nel seguente formato:

- riga 1: m

Seguono r blocchi di righe corrispondenti a uno scenario ciascuno, secondo il seguente formato:

- righe $1 + j$ $(0 \leq j \leq q - 1)$: l'**indice** della stazione la cui **etichetta** è stata restituita dalla chiamata j-esima e `find_next_station` per questo scenario.

Nota che ogni esecuzione del grader di esempio chiama sia `label` che `find_next_station`.