Stasiun (stations)

Singapore's Internet Backbone (SIB) terdiri dari n buah stasiun, yang diberikan **indeks** dari 0 sampai $n-1$. Ada juga $n-1$ buah penghubung dua arah, yang dinomori dari 0 sampai $n-2$. Setiap penghubung menghubungkan dua stasiun berbeda. Dua stasiun yang terhubung melalui suatu penghubung disebut tetangga.

Suatu jalur dari stasiun x ke stasiun y adalah urutan stasiun-stasiun berbeda $a_0, a_1, \cdots, a_p$, sedemikian sehingga $a_0 = x$, $a_p = y$, dan setiap dua stasiun yang berurutan pada jalur tersebut adalah tetangga. Terdapat **tepat satu** jalur dari stasiun x mana pun ke stasiun y mana pun.

Suatu stasiun x dapat menciptakan paket (sepotong data) dan mengirimkannya ke stasiun y mana pun, yang disebut sebagai **tujuan** paket. Paket ini harus diarahkan sepanjang jalur unik dari x ke y dengan cara sebagai berikut. Misalkan stasiun z yang saat ini sedang memegang paket, yang mana tujuan stasiunnya adalah y ($z \neq y$). Dalam situasi ini, stasiun z:

1. menjalankan **fungsi pengarahan** yang menentukan tetangga z yang berada pada jalur unik dari z ke y, dan
2. melanjutkan paket ke tetangga tersebut.

Namun, stasiun memiliki memori yang terbatas dan tidak menyimpan daftar semua penghubung yang ada di SIB untuk dipergunakan pada fungsi pengarahan.

Tugas Anda adalah melakukan implementasi skema pengarahan untuk SIB, yang terdiri dari dua fungsi.

- Fungsi pertama diberikan n, daftar semua penghubung yang ada di SIB dan sebuah bilangan bulat $k \geq n-1$ sebagai masukan. Fungsi ini memberikan setiap stasiun suatu bilangan bulat **unik** sebagai **label** antara 0 sampai dengan k.

- Fungsi kedua adalah fungsi pengarahan, yang dipasang di setiap stasiun setelah label-label diberikan. Fungsi ini **hanya** diberikan masukan sebagai berikut:

  - s, **label** dari stasiun yang saat ini sedang memegang paket,
  - t, **label** dari stasiun yang adalah tujuan paket ($t \neq s$),
  - c, daftar **label-label** dari stasiun-stasiun tetangga s.

  Fungsi ini harus mengembalikan **label** dari stasiun tetangga s ke mana paket harus diteruskan selanjutnya.

Pada satu subsoal, nilai dari solusi Anda tergantung dari nilai label terbesar yang diberikan pada stasiun-stasiun (secara umum, lebih kecil lebih baik).

Detail Implementasi

Anda harus melakukan implementasi fungsi berikut:

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

- n: banyaknya stasiun di SIB.
- k: label terbesar yang boleh digunakan.
- u dan v: *array* berukuran $n-1$ yang menjelaskan penghubung-penghubung. Untuk setiap i ($0 \leq i \leq n-2$), penghubung i menghubungkan stasiun-stasiun dengan indeks $u[i]$ dan $v[i]$.
- Fungsi ini harus mengembalikan suatu *array* L berukuran n. Untuk setiap i ($0 \leq i \leq n-1$) $L[i]$ adalah label yang diberikan kepada stasiun dengan indeks i. Semua elemen dari *array* L harus unik dan di antara 0 dan k.

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

- s: label dari stasiun yang saat ini sedang memegang paket.
- t: label dari stasiun yang adalah tujuan paket.
- c: suatu *array* yang memberikan daftar label-label dari stasiun-stasiun tetangga s. *Array* c diurutkan dari kecil ke besar.
- Fungsi ini harus mengembalikan label dari stasiun tetangga s ke mana paket harus diteruskan selanjutnya.

Setiap kasus uji melibatkan satu atau lebih skenario independen (i.e., deskripsi SIB yang berbeda). Untuk suatu kasus uji yang melibatkan r buah skenario, suatu **program** yang memanggil fungsi-fungsi di atas akan dijalankan tepat dua kali, sebagai berikut.

Pada eksekusi pertama dari program:

- Fungsi `label` dipanggil r kali,
- label-label yang dikembalikan disimpan oleh sistem *grader*, dan
- `find_next_station` tidak dipanggil.

Pada eksekusi kedua dari program:

- `find_next_station` mungkin dipanggil berulang kali. Pada setiap pemanggilan, suatu skenario **acak** dipilih, dan label-label yang dikembalikan fungsi `label` pada skenario tersebut digunakan sebagai masukan untuk `find_next_station`.
- `label` tidak dipanggil.

Lebih jelasnya, setiap informasi yang disimpan di variabel statis atau global pada eksekusi pertama dari program tidak tersedia pada pemanggilan fungsi `find_next_station`.

Contoh

Perhatikan pemanggilan berikut:

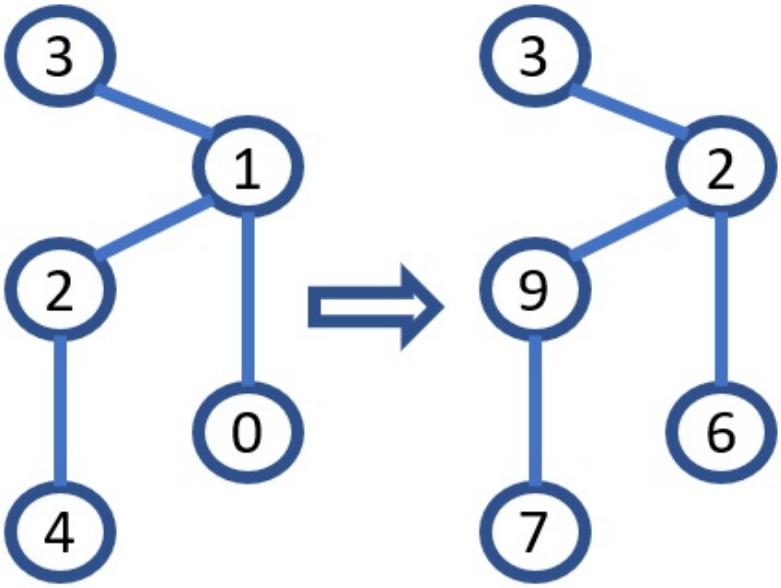
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Terdapat total 5 stasiun, dan 4 penghubung menghubungkan pasangan-pasangan stasiun dengan indeks $(0, 1)$, $(1, 2)$, $(1, 3)$ dan $(2, 4)$. Setiap label boleh merupakan bilangan bulat dari 0 sampai dengan $k = 10$.

Dalam rangka melaporkan pelabelan berikut:

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

fungsi `label` harus mengembalikan $[6, 2, 9, 3, 7]$. Nomor-nomor pada gambar berikut menunjukkan indeks (sebelah kiri) dan label yang diberikan (sebelah kanan).



Misalkan label-label sudah diberikan seperti penjelasan di atas dan perhatikan pemanggilan berikut:

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

Ini berarti stasiun yang saat ini sedang memegang paket adalah stasiun berlabel 9, dan stasiun tujuan adalah stasiun berlabel 6. Label dari stasiun-stasiun sepanjang jalur menuju stasiun tujuan adalah $[9, 2, 6]$. Maka, fungsi harus mengembalikan 2, yang adalah label dari stasiun ke mana paket

harus diteruskan selanjutnya (yang memiliki indeks 1).

Perhatikan pemanggilan lainnya:

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

Fungsi harus mengembalikan 3, karena stasiun tujuan dengan label 3 adalah tetangga dari stasiun dengan label 2, sehingga paket langsung diteruskan ke sana.

Batasan

- $1 \leq r \leq 10$

Untuk setiap pemanggilan `label`:

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

Untuk setiap pemanggilan `find_next_station`, masukan berasal dari pemanggilan `label` sebelumnya yang dipilih secara acak. Perhatikan label-label yang dihasilkan. Maka:

- s dan t adalah label dari dua stasiun berbeda.
- c adalah label dari stasiun-stasiun tetangga dari stasiun berlabel s, diurutkan dari kecil ke besar.

Untuk setiap kasus uji, total panjang dari semua *array* c yang diberikan kepada fungsi `find_next_station` tidak melebihi $100\ 000$ untuk semua skenario jika dijumlahkan.

Subsoal

1. (5 poin) $k = 1000$, tidak ada stasiun yang memiliki lebih dari 2 tetangga.
2. (8 poin) $k = 1000$, penghubung i menghubungkan stasiun $i + 1$ dan $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 poin) $k = 1\ 000\ 000$, maksimal hanya ada satu stasiun yang memiliki lebih dari 2 tetangga.
4. (10 poin) $n \leq 8$, $k = 10^9$
5. (61 poin) $k = 10^9$

Pada subsoal 5 Anda dapet memperoleh nilai parsial. Misalkan m adalah label terbesar yang dikembalikan oleh `label` dari semua skenario. Nilai Anda untuk subsoal tersebut dihitung sesuai dengan tabel berikut:

Label terbesar	Nilai
$m \geq 10^9$	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

Contoh *grader*

Contoh *grader* membaca masukan dalam format sebagai berikut:

- baris 1: r

Diikuti dengan r blok, masing-masing menjelaskan suatu skenario. Format dari setiap blok adalah sebagai berikut:

- baris 1: n k
- baris $2 + i$ ($0 \leq i \leq n - 2$): $u[i]$ $v[i]$
- baris $1 + n$: q: banyaknya pemanggilan `find_next_station`.
- baris $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j]$ $y[j]$ $w[j]$: **indeks-indeks** dari stasiun yang terlibat pada pemanggilan ke-j terhadap `find_next_station`. Stasiun $z[j]$ sedang memegang paket, stasiun $y[j]$ adalah stasiun tujuan, dan stasiun $w[j]$ adalah stasiun ke mana paket harus diteruskan selanjutnya.

Contoh *grader* mencetak keluaran dalam format sebagai berikut:

- baris 1: m

Diikuti dengan r blok sesuai dengan skenario berturutan. Format dari setiap blok adalah sebagai berikut:

- baris $1 + j$ ($0 \leq j \leq q - 1$): **indeks-indeks** dari stasiun, yang **label**nya dikembalikan oleh pemanggilan ke-j-th terhadap `find_next_station` pada skenario tersebut.

Perhatikan bahwa setiap eksekusi dari contoh *grader* memanggil fungsi `label` dan fungsi `find_next_station`.