Maršrutētāji (stations)

Singapūras Interneta Maģistrālē (*Singapore's Internet Backbone*, SIB) ietilpst n maršrutētāji, kuriem ir piešķirti **indeksi** no 0 līdz $n-1$. Šos maršrutētājus savieno $n-1$ divvirzienu sakaru kanāli, kas sanumurēti no 0 līdz $n-2$. Katrs sakaru kanāls savieno divus atšķirīgus maršrutētājus. Divus maršrutētājus, kurus savieno sakaru kanāls, sauc par kaimiņiem.

Ceļu no maršrutētāja x līdz maršrutētājam y veido tādu atšķirīgu maršrutētāju virkne $a_0, a_1, \cdots, a_p$, ka $a_0 = x$, $a_p = y$, un katri divi secīgi maršrutētāji ir kaimiņi. No jebkura maršrutētāja x līdz jebkuram citam maršrutētājam y ir **tieši viens** ceļš.

Jebkurš maršrutētājs x var izveidot datu paketi un nosūtīt to jebkuram citam maršrutētājam y, kuru sauc par paketes **adresātu**. Šī pakete jāpārsūta pa vienīgo ceļu no x uz y kā aprakstīts tālāk. Pieņemsim, ka šobrīd pakete atrodas maršrutētājā z, bet tās adresāts ir maršrutētājs y ($z \neq y$). Šajā situācijā maršrutētājs z:

1. izpilda **maršrutēšanas procedūru**, kas nosaka, kurš no z kaimiņiem atrodas uz vienīgā ceļa no z uz y, un
2. pārsūta paketi šim kaimiņam.

Diemžēl maršrutētājiem ir ierobežots atmiņas apjoms, tāpēc tie neglabā pilnu SIB sakaru kanālu sarakstu izmantošanai maršrutēšanas procedūrā.

Jums jāimplementē SIB maršrutēšanas shēma, kura sastāvēs no divām procedūrām.

- Pirmai procedūrai kā ievaddati tiks dota n vērtība, SIB sakaru kanālu saraksts un vesela skaitļa $k \geq n-1$ vērtība. Šai procedūrai katram maršrutētājam jāpiešķir **unikāla** skaitliska **iezīme** — vesels skaitlis robežās no 0 līdz k (ieskaitot).

- Otra procedūra ir maršrutēšanas procedūra, kas pēc iezīmju piešķiršanas tiks izmantota visos maršrutētājos. Tai tiks padoti **tikai** šādi ievaddati:

    - s: maršrutētāja, kurā šobrīd atrodas pakete, **iezīme**,
    - t: paketes adresāta **iezīme** ($t \neq s$),
    - c: maršrutētāja s kaimiņu **iezīmju** saraksts.

    Šai procedūrai jāatgriež tā s kaimiņa **iezīme**, kuram tālāk jāpārsūta pakete.

Vienā apakšuzdevumā jūsu risinājuma rezultāts būs atkarīgs no lielākās kādam maršrutētājam piešķirtās iezīmes skaitliskās vērtības (parasti — jo tā mazāka, jo labāk).

Implementēšanas detaļas

Jums jāimplementē šādas funkcijas:

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

- n: SIB maršrutētāju skaits.
- k: Lielākais skaitlis, kuru drīkst izmantot kā iezīmi.
- u un v: masīvi ar $n-1$ elementu, kas apraksta sakaru kanālus. Katram i ($0 \leq i \leq n-2$), sakaru kanāls i savieno maršrutētājus, kuru indeksi ir $u[i]$ un $v[i]$.
- Funkcijai jāatgriež masīvs L ar n elementiem. Katram i ($0 \leq i \leq n-1$) $L[i]$ ir iezīme, kas piešķirta maršrutētājam ar indeksu i. Visiem masīva L elementiem jābūt savā starpā atšķirīgiem un robežās starp 0 un k (ieskaitot).

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

- s: maršrutētāja, kurā atrodas pakete, iezīme.
- t: paketes adresāta iezīme.
- c: masīvs ar visu s kaimiņu iezīmēm augošā secībā.
- Funkcijai jāatgriež tā s kaimiņa iezīme, kuram jāpārsūta pakete.

Katrs tests ietver vienu vai vairākus neatkarīgus scenārijus (atšķirīgus SIB aprakstus). Katram testam ar r scenārijiem, **programma**, kas izsauc iepriekš aprakstītās funkcijas, tiek palaista tieši divreiz kā aprakstīts tālāk.

Pirmajā programmas palaišanas reizē:

- funkcija `label` tiek izsaukta r reizes,
- atgrieztās iezīmes tiek saglabātas testēšanas sistēmā un
- funkcija `find_next_station` netiek izsaukta.

Otrajā programmas palaišanas reizē:

- funkcija `find_next_station` var tikt izsaukta vairākkārt. Katrā izsaukumā tiek izvēlēts **patvaļīgs** scenārijs un iezīmes, kuras šajā scenārijā atgrieza funkcijas `label` izsaukums, tiks izmantotas kā funkcijas `find_next_station` ievaddati.
- funkcija `label` izsaukta netiek.

Jo īpaši, jebkura informācija, kas saglabāta statiskos vai globālos mainīgajos programmas pirmajā palaišanas reizē, nebūs pieejama funkcijā `find_next_station`.

Piemērs

Aplūkosim funkcijas izsaukumu:

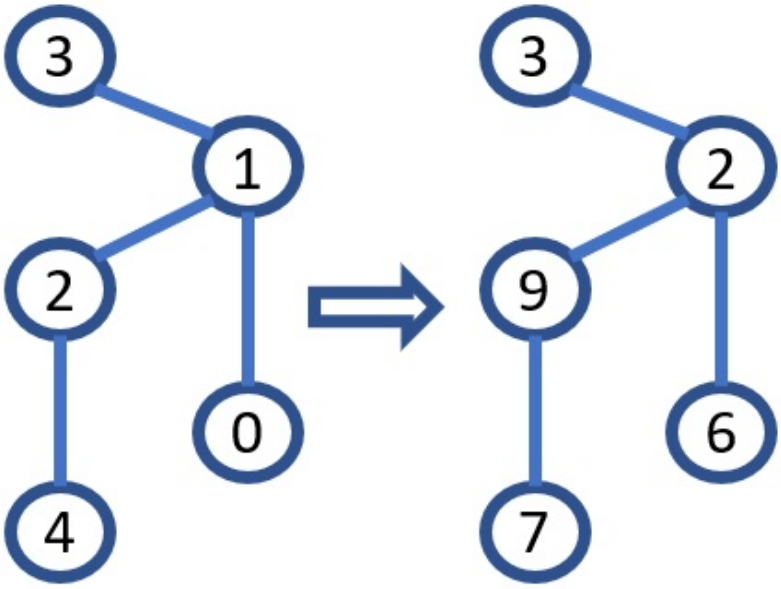
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Pavisam ir 5 maršrutētāji un 4 sakaru kanāli, kas savieno maršrutētāju pārus ar indeksiem $(0, 1)$, $(1, 2)$, $(1, 3)$ un $(2, 4)$. Katra iezīme var būt vesels skaitlis robežās no 0 līdz $k = 10$.

Ja maršrutētājiem iezīmes ir piešķirtas šādi:

Indekss	Iezīme
0	6
1	2
2	9
3	3
4	7

tad funcijai `label` jāatgriež $[6, 2, 9, 3, 7]$. Nākamajā zīmējumā skaitļi pa kreisi attēlo indeksus, bet pa labi — iezīmes.



Tālāk šim iezīmju piešķiršanas variantam aplūkosim funkcijas izsaukumu:

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

Tas nozīmē, ka pakete atrodas maršrutētājā 9 un tās adresāts ir maršrutētājs 6. Ceļu līdz adresātam veido maršrutētāji ar iezīmēm $[9, 2, 6]$. Tādejādi funkcijai jāatgriež 2, kas ir nākamā maršrutētāja ceļā uz adresātu iezīme (šī maršrutētāja indekss ir 1).

Cits šīs funkcijas izsaukums:

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

Funkcijai jāatgriež vērtība 3, jo adresāts ar iezīmi 3 ir maršrutētāja ar iezīmi 2 kaimiņš un tam pakete jānogādā uzreiz (jāpārsūta tieši pa sakaru kanālu).

Ierobežojumi

- $1 \leq r \leq 10$

Katram funkcijas `label` izsaukumam:

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

Katram funkcijas `find_next_station` izsaukumam ievaddati tiek veidoti no patvaļīga iepriekš veikta funkcijas `label` izsaukuma izvaddatiem. Ja aplūko piešķirtās iezīmes, tad:

- s un t ir divu atšķirīgu maršrutētāju iezīmes.
- c ir visu maršrutētāja s kaimiņu iezīmes augošā secībā.

Katram testpiemēram visu funkcijai `find_next_station` padoto c masīvu kopgarums kopā visiem scenārijiem nepārsniegs $100\ 000$.

Apakšuzdevumi

1. (5 punkti) $k = 1000$, nevienam maršrutētājam nav vairāk par diviem kaimiņiem.
2. (8 punkti) $k = 1000$, sakaru kanāls i savieno maršrutētājus $i + 1$ un $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 punkti) $k = 1\ 000\ 000$, ne vairāk kā vienam maršrutētājam ir vairāk nekā divi kaimiņi.
4. (10 punkti) $n \leq 8$, $k = 10^9$
5. (61 punkts) $k = 10^9$

Pēdējā (5.) apakšuzdevumā jūs varat iegūt daļēju vērtējumu. Ja m ir lielākā iezīmes vērtība, ko funkcija `label` atgriezusi kopumā visos scenārijos, piešķirto punktu skaits par apakšuzdevumu tiks aprēķināts pēc šādas tabulas:

Lielākā iezīmes vērtība	Punkti
$m \geq 10^9$	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

Paraugvērtētājs

Paraugvērtētājs ielasa datus šādā formātā:

- 1. rinda: r

Tam seko r bloki, kur katrs bloks apraksta vienu scenāriju. Katra bloka formāts ir šāds:

- 1. rinda: n k
- $(2+i)$-tā rinda ($0 \leq i \leq n-2$): $u[i]$ $v[i]$
- $(1+n)$-tā rinda: q: funkcijas `find_next_station` izsaukumu skaits.
- $(2+n+j)$-tā rinda ($0 \leq j \leq q-1$): $z[j]$ $y[j]$ $w[j]$: maršrutētāju **indeksi**, kas iesaistīti j-tajā funkcijas `find_next_station` izsaukumā. Maršrutētājā $z[j]$ atrodas pakete, maršrutētājs $y[j]$ ir paketes adresāts, un pakete tālāk jāsūta maršrutētājam $w[j]$.

Paraugvērtētājs rezultātu izvada šādā formātā:

- 1. rinda: m

Seko r bloki, kas atbilst secīgiem scenārijiem ievadā. Katra bloka formāts:

- $(1+j)$-tā rinda ($0 \leq j \leq q-1$): tā maršrutētāja **indekss**, kura **iezīme** šajā scenārijā tika atgriezta j-tajā funkcijas `find_next_station` izsaukumā.

Ievērojiet, ka katrā paraugvērtētāja darbības reizē tas izsauc gan funkciju `label`, gan funkciju `find_next_station`.