Կայաններ (stations)

Singapore's Internet Backbone (SIB) բաղկացած է n կայաններից, որոնք նշանակված են **ինդեքսներով** 0-ից մինչև $n-1$: Նաև կան $n-1$ երկկողմանի կողեր, համարակալված 0-ից մինչև $n-2$: Յուրաքանչյուր կող միացնում է երկու տարբեր կայաններ: Մեկ կողով միացված երկու կայանները կոչվում են հարևան:

x կայանից դեպի y կայան տանող ճանապարհը իրարից տարբեր կայանների հաջորդականություն է $a_0, a_1, \cdots, a_p$, այնպես որ $a_0 = x$, $a_p = y$, և յուրաքանչյուր երկու հաջորդական կայաններ ճանապարհի մեջ հարևաններ են: Կա **ճիշտ մեկ** ճանապարհ կամայական x կայանից դեպի կամայական y կայան:

Կամայական x կայան կարող է ստեղծել փաթեթ (հիշողության կտոր) և ուղարկել դա դեպի կամայական ուրիշ y կայան, որը կոչվում է փաթեթի **նպատակակետ**: Այս փաթեթը պետք է լինի ուղղորդված x-ից դեպի y տանող միարժեք ճանապարհի երկայնքով հետևյալ կերպ: Դիտարկենք կայան z-ը, որն այս պահին պարունակում է փաթեթը, որի նպատակակետ կանգառը y-ն է ($z \neq y$): Այս պարագայում կայան z-ը.

1. կատարում է **ուղղորդման ֆունկցիա**, որը կորոշի z-ի այն հարևանին, որը գտնվում է z-ից դեպի y տանող միարժեք ճանապարհի վրա, և
2. ուղարկում է փաթեթը այդ հարևանին:

Այնուամենայնիվ, կայանները ունեն սահմանափակ հիշողություն և չեն կարող պահել SIB-ի հղումների ամբողջ ցուցակը օգտագործելով դա ուղղորդման ֆունկցիայի մեջ:

Ձեր խնդիրն է իրականացնել ուղղորդման սխեմա SIB-ի համար, որը բաղկացած է երկու ֆունկցիաներից:

- Առաջին ֆունկցիայում տրված են n-ը, SIB-ում հղումների ցուցակը և $k \geq n-1$ ամբողջ թիվը, որպես մուտքային տվյալներ: Ֆունկցիան վերագրում է յուրաքանչյուր կայանին **միարժեք** ամբողջ **պիտակ** 0-ից k միջակայքում, ներառյալ:

- Երկրորդ ֆունկցիան ուղղորդման ֆունկցիան է, որը տեղակայված է բոլոր կայաններում պիտակների վերագրումից հետո: Տրված են **միայն** հետևյալ մուտքային տվյալները.

  - s, այն կայանի **պիտակ**-ը, որն այս պահին պարունակում է փաթեթը,
  - t, փաթեթի նպատակակետ կայանի **պիտակ**-ը ($t \neq s$),
  - c, s-ի բոլոր հարևան կայանների **պիտակների** ցուցակը:

Վերադարձնում է s-ի այն հարևանի կայանի **պիտակ**-ը, ուր փաթեթը պետք է ուղարկվի:

Ենթախնդիրներից մեկում, ձեր լուծման միավորը կախված կլինի կայաններին վերագրված պիտակների առավելագույն արժեքից (ընդհանրապես, ավելի փոքրը ավելի լավ է):

Իրականացման մանրամասներ

Դուք պետք է իրականացնեք հետևյալ ֆունկցիաները.

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

- n. SIB-ում կայանների քանակը:
- k. մաքսիմալ պիտակը, որը կարող է օգտագործվել:
- u և v. կողերը նկարագրող $n-1$-ի չափսի զանգվածներ: Յուրաքանչյուր i-ի համար $(0 \leq i \leq n-2)$, i-րդ կողը միացնում է $u[i]$ և $v[i]$ ինդեքսներով կայանները:
- Այս ֆունկցիան պետք է վերադարձնի n չափի մեկ L զանգված: Յուրաքանչյուր i-ի համար $(0 \leq i \leq n-1)$ $L[i]$-ն պիտակն է վերագրված i ինդեքսով կայանին: L զանգվածի բոլոր էլեմենտները պետք է լինեն միարժեք և 0-ից k միջակայքում, ներառյալ:

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

- s. փաթեթը պարունակող կայանի պիտակը:
- t. փաթեթի նպատակակետ կայանի պիտակը:
- c. զանգված պարունակող s-ի բոլոր հարևան կայանների պիտակների ցուցակը: c զանգված է սորտավորված է աճման կարգով:
- Այս ֆունկցիան պետք է վերադարձնի s-ի այն հարևան կայանի պիտակը, որտեղ փաթեթը պետք է ուղարկվի:

Յուրաքանչյուր թեստ ներգրավում է մեկ կամ ավել անկախ սցենարներ (այն է, որ տարբեր SIB-ի նկարագրություններ): r սցենարներ ներգրաված թեստի համար, **ծրագիր**-ը, որը կանչում է վերը նշված ֆունկցիաները աշխատում է ճիշտ երկու անգամ, հետևյալ կերպով:

Ծրագրի առաջին աշխատանքի ընթացքում.

- `label` ֆունկցիան կանչվում է r անգամ,
- վերադարձված պիտակները պահվում են գնահատման սիստեմի կողմից, և
- `find_next_station`-ը չի կանչվում:

Ծրագրի երկրորդ աշխատանքի ընթացքում.

- `find_next_station`-ը կարող է կանչվել մի քանի անգամ: Յուրաքանչյուր կանչի ժամանակ, ընտրվում է **կամայական** սցենար, և `label` ֆունկցիայի կողմից վերադարձված պիտակները այդ սցենարում օգտագործվում են, որպես մուտքային տվյալներ `find_next_station`-ի համար:
- `label`-ը չի կանչվում:

Մասնավորապես, ծրագրի առաջին կանչում ստատիկ կամ գլոբալ փոփոխականներում պահված կամայական ինֆորմացիա հասանելի չէ `find_next_station` ֆունկցիայի ներսում:

Օրինակ

Դիտարկենք հետևյալ կանչը.

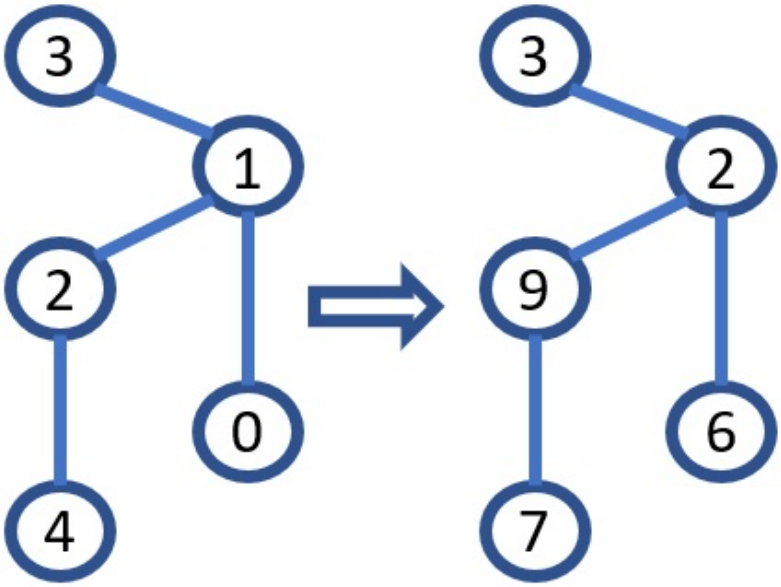
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Կան ընդհանուր 5 կայաններ, և 4 կողեր՝ միացնող կայանների ինդեքսների զույգերը $(0, 1)$, $(1, 2)$, $(1, 3)$ և $(2, 4)$. Յուրաքանչյուր պիտակ կարող է լինել ամբողջ թիվ 0-ից մինչև $k = 10$:

Հետևյալ պիտակավորման մասին զեկուցելու համար.

Ինդեքս	Պիտակ
0	6
1	2
2	9
3	3
4	7

`label` ֆունկցիան պետք է վերադարձնի $[6, 2, 9, 3, 7]$: Հետևյալ նկարում թվերը ցույց են տալիս ինդեքսները (ձախ հատված) և վերագրված պիտակները (աջ հատված):



Ենթադրենք պիտակները վերագրվել են այնպես ինչպես նկարագրված է վերևում և դիտարկենք հետևյալ կանչը.

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

Սա նշանակում է, որ փաթեթը պահող կայանն ունի 9 արժեքով պիտակ, իսկ նպատակակետ կայանը՝ 6 արժեքով պիտակ: Դեպի նպատակակետ տանող ճանապարհի կայանների պիտակներն են՝ $[9, 2, 6]$: Այսպիսով, կանչը պետք է վերադարձնի 2, որն այն կայանի պիտակն է, որտեղ փաթեթը պետք է ուղարկվի (որն ունի 1 ինդեքս):

Դիտարկենք մեկ այլ հնարավոր կանչ.

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

Ֆունկցիան պետք է վերադարձնի 3, քանի որ նպատակակետ կայանը՝ 3 արժեքով պիտակը, 2 արժեքով պիտակի կայանի հարևան է, և այսպիսով պետք է ստանա փաթեթը միանգամից:

Սահմանափակումներ

- $1 \leq r \leq 10$

Յուրաքանչյուր `label` կանչի համար.

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

Յուրաքանչյուր `find_next_station` կանչի համար, մուտքային տվյալները գալիս են

նախորդ `label` կանչերից կամայական ընտրվածից: Դիտարկենք առաջացած պիտակները: Ապա.

- s-ը և t-ն երկու տարբեր կայանների պիտակներ են:
- c-ն s պիտակով կայանի հարևան կայանների պիտակների հաջորդականությունն է, դասավորված աճման կարգով:

Յուրաքանչյուր թեստի համար, բոլոր c զանգվածների գումարային երկարությունը, փոխանցված `find_next_station` ֆունկցիային, չի գերազանցում 100 000-ը բոլոր սցենարների համար:

Ենթախնդիրներ

1. (5 միավոր) $k = 1000$, ոչ մի կայան չունի 2-ից ավել հարևան:
2. (8 միավոր) $k = 1000$, i-րդ կողը միացնում է $i + 1$-րդ կայանը և $\lfloor \frac{i}{2} \rfloor$-րդին:
3. (16 միավոր) $k = 1\ 000\ 000$, ամենաշատը մեկ կայան ունի 2-ից ավել հարևան:
4. (10 միավոր) $n \leq 8$, $k = 10^9$
5. (61 միավոր) $k = 10^9$

5-րդ ենթախնդրում դուք կարող եք հավաքել մասնակի միավոր: Ենթադրենք m-ը be մաքսիմում պիտակի արժեքն է վերադարձված `label`-ի կողմից բոլոր սցենարներով: Ձեր միավորը այս ենթախնդրի համար կհաշվվի ըստ հետևյալ աղյուսակի.

Մաքսիմալ պիտակ	Միավոր
$m \geq 10^9$	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

Գնահատող ծրագրի նմուշ

Գնահատող ծրագրի նմուշը կարդում է մուտքային տվյալները հետևյալ ձևաչափով.

- տող 1: r

հետևում է r բլոկներ, յուրաքանչյուրը մեկ սցենար ներկայացնող: Յուրաքանչյուր բլոկի ձևաչափը հետևյալն է.

- տող 1: n k
- տող $2 + i$ ($0 \leq i \leq n - 2$): $u[i]$ $v[i]$
- տող $1 + n$: q: `find_next_station` կանչերի քանակը:
- տող $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j]$ $y[j]$ $w[j]$: կայանների **ինդեքսներներ**-ն է ներգրավված `find_next_station`-ի j-րդ կանչի մեջ: $z[j]$-րդ կայանը

պարունակում է փաթեթը, $y[j]$-րդ կայանը փաթեթի նպատակակետն է և $w[j]$-րդ կայանը, թե որտեղ փաթեթը պետք է ուղարկվի:

Գնահատող ծրագրի նմուշը տպում է արդյունքը հետևյալ ձևաչափով.

- տող 1: m

r բլոկներ՝ համապատասխանող մուտքային տվյալներում տրված սցենարներին: Յուրաքանչյուր բլոկի ձևաչափը հետևյալն է.

- տող $1 + j$ $(0 \leq j \leq q - 1)$: կայանի **ինդեքսը**, ում **պիտակը** վերադարձվել է `find_next_station`-ի j-րդ կանչի կողմից այս սցենարում:

Նկատեք, որ յուրաքանչյուր գնահատող ծրագրի նմուշը աշխատացնելուց կանչվում է և `label` -ը և `find_next_station`-ը: