სადგურები (stations)

Singapore's Internet Backbone (SIB) შედგება n სადგურისგან, რომელთაც მინიჭებული აქვთ **ინდექსები** 0-დან $(n-1)$-მდე. ასევე არსებობს $n-1$ ორმხრივი ბმული, დანომრილი 0-დან $(n-2)$-მდე. თითოეული ბმული აერთებს ორ სხვადასხვა სადგურს. ორ სადგურს რომელთა შორისაც ბმულია მეზობლები ეწოდებათ.

x სადგურიდან y სადგურამდე გზა ეწოდება განსხვავებული $a_0, a_1, \cdots, a_p$ სადგურების მიმდევრობას, სადაც $a_0 = x$, $a_p = y$ და გზაში ყოველი ორი მომდევნო სადგური მეზობლები არიან. არსებობს **ზუსტად ერთი** გზა ყოველი x სადგურიდან სხვა y სადგურამდე.

რომელიმე x სადგურმა შეიძლება შექმნას პაკეტი (მონაცემების ნაწილი) და გაუგზავნოს სხვა y სადგურს, რომელსაც პაკეტის **სამიზნე** დავარქვათ. უნდა მოხდეს პაკეტის მარშრუტიზაცია x-დან y-სკენ უნიკალური გზის მიმართულებით შემდეგნაირად: დავუშვათ, ამჟამად z სადგურში იმყოფება პაკეტი, რომლის სამიზნეცაა y ($z \neq y$) სადგური. ამ სიტუაციაში სადგური z:

1. ასრულებს **მარშრუტიზაციის პროცედურას**, რომელიც ადგენს z-ის მეზობელს, რომელიც არის უნიკალურ გზაზე z-დან y-მდე და
2. გადაგზავნის პაკეტს ამ მეზობელში.

თუმცა, სადგურებს აქვთ შეზღუდული მეხსიერება და ვერ ინახავენ SIB-ის ბმულების სრულ სიას მარშრუტიზაციის პროცედურაში გამოსაყენებლად.

თქვენი ამოცანაა შექმნათ მარშრუტიზაციის სქემა SIB-ისთვის, რომელიც შედგება ორი პროცედურისგან.

- პირველ პროცედურას გადაეცემა n, SIB-ის ბმულების სია და მთელი $k \geq n-1$ რიცხვი. ის ანიჭებს თითოეულ სადგურს **უნიკალურ** მთელ **იარლიყს** 0-დან k-ს ჩათვლით.

- მეორე პროცედურა არის მარშრუტიზაციის პროცედურა, რომელიც ჩატვირთულია ყველა სადგურზე იარლიყების მინიჭების შემდეგ. მას გადაეცემა **მხოლოდ** შემდეგი მონაცემები:

  - s, **იარლიყი** სადგურისა, რომელშიც ამჟამად იმყოფება პაკეტი;
  - t, **იარლიყი** პაკეტის სამიზნე სადგურის ($t \neq s$);
  - c, **იარლიყების** სია s-ის მეზობლებისა.

  მან უნდა დააბრუნოს s-ის იმ მეზობლის **იარლიყი**,კ რომელშიც პაკეტი უნდა გადაიგზავნოს.

ყოველ ქვეამოცანაში თქვენი ამოხსნის შეფასება დამოკიდებულია სადგურებისთვის მინიჭებული იარლიყების მნიშვნელობების მაქსიმუმზე (ზოგადად, რაც ნაკლები მით უკეთესი).

იმპლემენტაციის დეტალები

თქვენ უნდა მოახდინოთ შემდეგი პროცედურების იმპლემენტაცია:

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

- n: სადგურების რაოდენობა SIB-ში;
- k: მაქსიმალური იარლიყი, რომლის გამოყენებაც შეიძლება;
- u და v: $n-1$ ზომის მასივები, რომლებიც აღწერენ ბმულებს. ყოველი i-თვის ($0 \leq i \leq n-2$), ბმული i აერთებს სადგურებს ინდექსებით $u[i]$ და $v[i]$;
- ამ პროცედურამ უნდა დააბრუნოს ერთი L მასივი ზომით n. ყოველი i-თვის ($0 \leq i \leq n-1$) $L[i]$ არის იარლიყი, რომელიც ენიჭება სადგურს i. L-ის ელემენტები უნდა იყოს განსხვავებული და 0-დან k-ს ჩათვლით.

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

- s: სადგურის იარლიყი, სადაც იმყოფება პაკეტი;
- t: სამიზნე სადგურის იარლიყი;
- c: მასივი, რომელიც აღწერს s სადგურის მეზობლების სიას. c მასივი დალაგებულია ზრდადობით;
- ამ პროცედურამ უნდა დააბრუნოს s-ის მეზობელის იარლიყი, რომელშიც პაკეტი უნდა გადაიგზავნოს.

თითოეული ტესტი შეიცავს ერთ ან მეტ დამოუკიდებელ სცენარს (ანუ სხვადასხვა SIB-ის აღწერას). თუ ტესტში r სცენარია, **პროგრამა**, რომელიც იძახებს ზემოთ აღწერილ პროცედურებს, გაეშვება ორჯერ შემდეგნაირად:

პროგრამის პირველი გაშვებისას:

- `label` პროცედურა გამოძახებულ იქნება r-ჯერ;
- დაბრუნებული იარლიყები შეინახება გრადერის სისტემაში და
- `find_next_station` არ იქნება გამოძახებული.

პროგრამის მეორე გაშვებისას:

- `find_next_station` შეიძლება გამოძახებულ იქნას მრავალჯერ. ყოველ გამოძახებაში **ნებისმიერი** სცენარი ირჩევა და ამ სცენარისთვის `label`-ის მიერ დაბრუნებული იარლიყები გამოიყენება შესატან მონაცემებად `find_next_station`-თვის.
- `label` არ იქნება გამოძახებული.

კერძოდ, პროგრამის პირველი გაშვებისას შენახული სტატიკური თუ გლობალური ცვლადები ვერ იქნება გამოყენებული `find_next_station` პროცედურაში.

მაგალითი

განვიხილოთ შემდეგი გამოძახება:

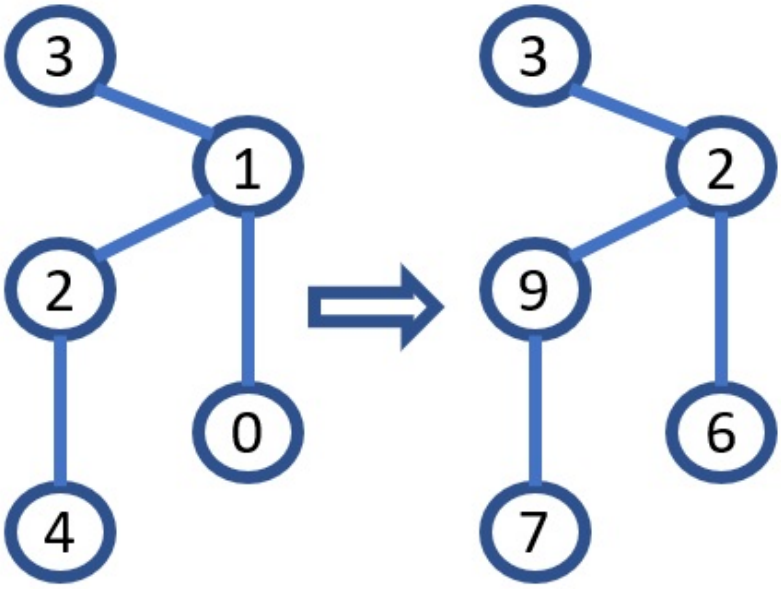
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

სულ არის 5 სადგური და 4 ბმული, რომლებიც აერთებს სადგურებს ინდექსებით $(0, 1)$, $(1, 2)$, $(1, 3)$ და $(2, 4)$. თითოეული იარლიყი უნდა იყოს მთელი რიცხვი 0-დან $k = 10$-მდე.

შემდეგი იარლიყების მისანიჭებლად:

ინდექსი	იარლიყი
0	6
1	2
2	9
3	3
4	7

`label` პროცედურამ უნდა დააბრუნოს [6, 2, 9, 3, 7]. შემდეგ სურათზე რიცხვები ასახავს ინდექსებს (მარცხენა მხარეს) და მინიჭებულ იარლიყებს (მარჯვენა მხარეს).



დავუშვათ, რომ იარლიყები იქნა მინიჭებული ისე, როგორც ზემოთ აღვწერეთ და განვიხილოთ შემდეგი გამოძახება:

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

ეს ნიშნავს, რომ სადგურს, სადაც არის პაკეტი აქვს იარლიყი 9 და სამიზნე სადგურს აქვს იარლიყი 6. იარლიყები სამიზნე სადგურისკენ გზაზე არის: $[9, 2, 6]$. შესაბამისად, გამოძახებამ უნდა დააბრუნოს 2, რომელიც არის იმ სადგურის იარლიყი, სადაც პაკეტი უნდა გადაიგზავნოს (რომელსაც აქვს ინდექსი 1).

განვიხილოთ სხვა შესაძლო გამოძახება:

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

პროცედურამ უნდა დააბრუნოს 3, რადგან სამიზნე სადგური იარლიყით 3 არის მეზობელი სადგურისა იარლიყით 2 და, შესაბამისად, პაკეტს მიიღებს პირდაპირ.

შეზღუდვები

- $1 \leq r \leq 10$

`label`-ის ყოველი გამოძახებისას:

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

`find_next_station`-ის ყოველი გამოძახებისას შესატანი მონაცემები მომდინარეობს `label`-ის რომელიმე წინა გამოძახებიდან. განვიხილოთ იარლიყები, რომელიც მან დააბრუნა. მაშინ:

- s და t არის ორი სხვადასხვა სადგურის იარლიყები;
- c არის s-ის ყველა მეზობელი სადგურის ზრდადობით დალაგებული იარლიყების მიმდევრობა.

ყოველ ტესტში c მასივების ჯამური სიგრძე, რომლებიც გადაეცემა `find_next_station` პროცედურას, არ აღემატება 100 000-ს ყველა სცენარში ერთად.

ქვეამოცანები

1. (5 ქულა) $k = 1000$, არცერთ სადგურს არ ჰყავს 2-ზე მეტი მეზობელი.
2. (8 ქულა) $k = 1000$, ბმული i აერთებს სადგურებს $i + 1$ და $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 ქულა) $k = 1\ 000\ 000$, არაუმეტეს ერთ სადგურს ჰყავს 2-ზე მეტი მეზობელი.
4. (10 ქულა) $n \leq 8$, $k = 10^9$
5. (61 ქულა) $k = 10^9$

ქვეამოცანაში 5 თქვენ შეგიძლიათ მიიღოთ ნაწილობრივი შეფასება. ვთქვათ m არის `label`-ის მიერ დაბრუნებული იარლიყის მაქსიმალური მნიშვნელობა ყველა სცენარს

შორის. ამ ქვეამოცანაში თქვენი ქულა დაითვლება შემდეგი ცხრილის მიხედვით:

მაქსიმალური იარლიყი	ქულა
$m \geq 10^9$	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

სანიმუშო გრადერი

სანიმუშო გრადერს შეაქვს მონაცემები შემდეგი ფორმატით:

- სტრიქონი 1: r

მას მოსდევს r ბლოკი, თითოეული სცენარის აღწერით. ყოველი ბლოკის ფორმატი შემდეგია:

- სტრიქონი 1: n k;
- სტრიქონი $2 + i$ ($0 \leq i \leq n - 2$): $u[i]$ $v[i]$;
- სტრიქონი $1 + n$: q: `find_next_station`-ის გამოძახებების რაოდენობა;
- სტრიქონი $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j]$ $y[j]$ $w[j]$: `find_next_station`-ის j-ურ გამოძახებაში მონაწილე სადგურების **ინდექსები**. სადგურს $z[j]$ აქვს პაკეტი, სადგური $y[j]$ არის სამიზნე და სადგური $w[j]$ არის სადგური სადაც პაკეტი უნდა გადაიგზავნოს.

სანიმუშო გრადერს გამოაქვს შედეგი შემდეგი ფორმატით:

- სტრიქონი 1: m

r რაოდენობის ბლოკი, რომლებიც შეესაბამება ზემოთ აღწერილ თანმიმდევრულ სცენარებს. თითოეული ბლოკის ფორმატი შემდეგია:

- სტრიქონი $1 + j$ ($0 \leq j \leq q - 1$): სადგურის **ინდექსი**, რომლის **იარლიყიც** იქნა დაბრუნებული `find_next_station`-ის j-ურ გამოძახებაზე ამ სცენარში.

შევნიშნოთ, რომ სანიმუშო გრადერის ყოველი გაშვება იძახებს როგორც `label`-ს, ასევე `find_next_station`-ს.