

## Станици (stations)

Singapore's Internet Backbone (SIB) се состои од  $n$  станици, означени со **индекси** од 0 до  $n - 1$ . Исто така, постојат  $n - 1$  двонасочни врски, нумерирани од 0 до  $n - 2$ . Секоја врска поврзува две различни станици. Две станици поврзани со една врска се нарекуваат соседи.

Патека од станицата  $x$  до станицата  $y$  е низа од различни станици  $a_0, a_1, \dots, a_p$ , така што  $a_0 = x$ ,  $a_p = y$ , и секои две последователни станици во патеката се соседи. Постои **точно една** патека од било која станица  $x$  до било која друга станица  $y$ .

Секоја станица  $x$  може да создаде пакет (дел од податоците) и може да го испрати до било која друга станица  $y$ , која се нарекува **цел (мета)** на пакетот. Овој пакет мора да биде насочен (рутиран) по единствената патека од  $x$  до  $y$ , како што е опишано подолу. Земете ја за пример станицата  $z$  која што во моментот чува пакет, а чија што целна станица е  $y$  ( $z \neq y$ ). Во оваа ситуација станицата  $z$ :

1. извршува **рутирачка процедура** што го одредува соседот на  $z$  кој што се наоѓа на единствената патека од  $z$  до  $y$ , и
2. го препраќа пакетот на овој сосед.

Сепак, станиците имаат ограничена меморија и не ја зачувуваат целата листа на врски во SIB за да ја користат во рутирачката процедура.

Ваша задача е да имплементирате рутирачка шема за SIB, која што се состои од две процедури.

- Како влез на првата процедура се дадени бројот  $n$ , листата на врски во SIB и цел број  $k \geq n - 1$ . Процедурата доделува на секоја станица **единствена ознака** (цел број) помеѓу 0 и  $k$ , вклучувајќи ги и 0 и  $k$ .
- Втората процедура е рутирачката процедура, која е распоредена на сите станици откако ќе бидат доделени ознаките. Ги има **само** следните влезни параметри:
  - $s$ , **ознаката** на станицата која што во моментот го чува пакетот,
  - $t$ , **ознаката** на целната станица на пакетот ( $t \neq s$ ),
  - $c$ , листа од **ознаки** на сите соседи на  $s$ .

Треба да ја врати **ознаката** на соседот на  $s$  до кој што треба да биде препратен пакетот.

Во една подзадача, резултатот од вашето решение зависи од вредноста на максималната ознака доделена на која било станица (генерално, помала ознака е подобра).

## Имплементациски детали

Треба да ги имплементирате следните процедури:

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

- $n$ : број на станици во SIB.
- $k$ : максималната ознака која може да се користи.
- $u$  и  $v$ : низи со големина  $n - 1$  кои ги опишуваат врските. За секое  $i$  ( $0 \leq i \leq n - 2$ ), врската  $i$  ги поврзува станиците со индекси  $u[i]$  и  $v[i]$ .
- Оваа процедура треба да врати една низа  $L$  со големина  $n$ . За секое  $i$  ( $0 \leq i \leq n - 1$ )  $L[i]$  е ознаката доделена на станицата со индекс  $i$ . Сите елементи на низата  $L$  мора да бидат единствени и да бидат помеѓу 0 и  $k$ , вклучувајќи ги и 0 и  $k$ .

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

- $s$ : ознака на станицата која чува пакет.
- $t$ : ознака на целната станица на пакетот.
- $c$ : низа која содржи листа на ознаките на сите соседи на  $s$ . Низата  $c$  е сортирана во растечки редослед.
- Оваа процедура треба да ја врати ознаката на соседот на  $s$  до кој треба да биде препратен пакетот.

Секој тест случај вклучува едно или повеќе независни сценарија (т.е. различни описи на SIB). За тест случај кој што вклучува  $r$  сценарија, **програма** која што ги повикува процедурите опишани погоре се извршува точно два пати, како што е опишано подолу.

За време на првото извршување на програмата:

- процедурата `label` се повикува  $r$  пати,
- Вратените ознаки се зачувани од системот кој оценува, и
- `find_next_station` не се повикува.

За време на второто извршување на програмата:

- `find_next_station` може да биде повикана повеќе пати. Во секој повик, се избира **произволно** сценарио, и ознаките вратени од повикот на процедурата `label` во тоа сценарио се користат како влезови на `find_next_station`.
- `label` не се повикува.

Поконкретно, секоја информација зачувана во статички или глобални променливи при првото извршување на програмата не е достапна во процедурата `find_next_station`.

## Пример

Земете го за пример следниот повик:

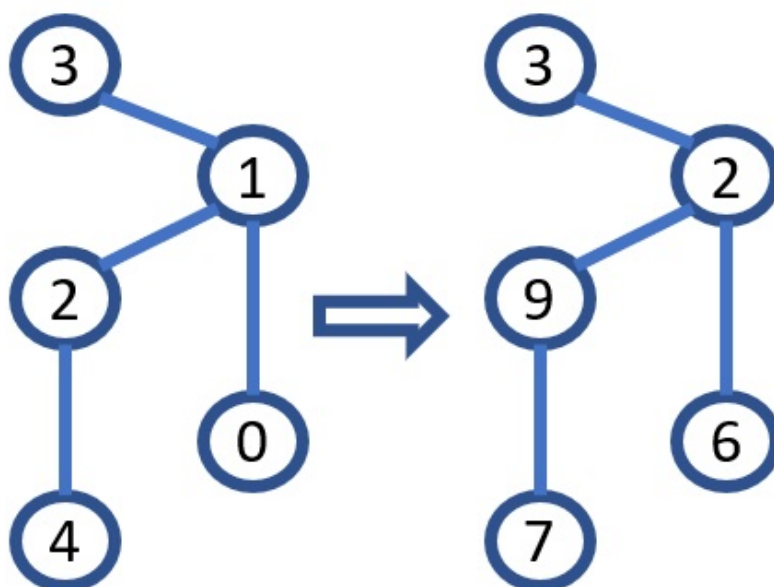
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Постојат вкупно 5 станици, и 4 врски кои што ги поврзуваат паровите од станици со следните индекси: (0, 1), (1, 2), (1, 3) и (2, 4). Секоја ознака може да биде цел број од 0 до  $k = 10$ .

За да го пријавите следново означување:

Индекс	Ознака
0	6
1	2
2	9
3	3
4	7

процедурата `label` треба да врати [6, 2, 9, 3, 7]. Броевите на сликата подолу ги прикажуваат индексите (лев панел) и доделените ознаки (десен панел).



Претпоставете дека ознаките се доделени како што е опишано погоре и разгледајте го следниот повик:

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

Ова значи дека станицата која го чува пакетот има ознака 9, а целната станица има ознака 6. Ознаките на станиците на патеката до целната станица се [9, 2, 6]. Оттука, повикот треба да врати 2, што претставува и ознаката на станицата до која што треба да биде препратен

пакетот (а која има индекс 1).

Да разгледаме друг можен повик:

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

Процедурата треба да врати 3, бидејќи целната станица со ознака 3 е сосед на станицата со ознака 2, и оттука треба да го прими пакетот директно.

## Ограничувања

- $1 \leq r \leq 10$

За секој повик до `label`:

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

За секој повик до `find_next_station`, влезот доаѓа од произволно избран претходен повик до `label`. Разгледајте ги ознаките што повикот ги создал. Потоа:

- $s$  и  $t$  се ознаки на две различни станици.
- $c$  е низа од сите ознаки на соседите на станицата со ознака  $s$ , во растечки редослед.

За секој тест случај, вкупната должина на сите низи  $c$  предадени на процедурата `find_next_station` не надминува 100 000 за сите сценарија заедно.

## Подзадачи

1. (5 поени)  $k = 1000$ , нема станица со повеќе од 2 соседи.
2. (8 поени)  $k = 1000$ , врската  $i$  ги поврзува станиците  $i + 1$  и  $\lfloor \frac{i}{2} \rfloor$ .
3. (16 поени)  $k = 1\,000\,000$ , најмногу една станица има повеќе од 2 соседи.
4. (10 поени)  $n \leq 8$ ,  $k = 10^9$
5. (61 поени)  $k = 10^9$

Во подзадачата 5 може да добиете делумни бодови. Нека  $m$  е максималната ознака вратена од `label` низ сите сценарија. Вашиот резултат за оваа подзадача се добива според следната табела:

Максимална ознака	Бодови
$m \geq 10^9$	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

## Пример оценувач

Пример оценувачот го чита влезот во следниот формат:

- линија 1:  $r$

Следуваат  $r$  блокови, од кои секој опишува едно сценарио. Форматот на секој блок е даден во продолжение:

- линија 1:  $n \ k$
- линии  $2 + i$  ( $0 \leq i \leq n - 2$ ):  $u[i] \ v[i]$
- линија  $1 + n$ :  $q$ : бројот на повици до `find_next_station`.
- линии  $2 + n + j$  ( $0 \leq j \leq q - 1$ ):  $z[j] \ y[j] \ w[j]$ : **индекси** на станиците вклучени во  $j$ -тиот повик до `find_next_station`. Станицата  $z[j]$  го чува пакетот, станицата  $y[j]$  е целната станица на пакетот, и станицата  $w[j]$  е станицата до која пакетот треба да биде препратен.

Пример оценувачот го печати резултатот во следниот формат:

- линија 1:  $m$

Следуваат  $r$  блокови што одговараат на последователните сценарија од влезот. Форматот на секој блок е следниот:

- линии  $1 + j$  ( $0 \leq j \leq q - 1$ ): **индекс** на станицата, чија **ознака** беше вратена од  $j$ -тиот повик на `find_next_station` во ова сценарио.

Забележете дека секое извршување на пример оценувачот ги повикува и двете процедури, `label` и `find_next_station`.