Stanice (stations)

Singapore's Internet Backbone (SIB) se sastoji od n stanica, kojima su dodijeljeni **indeksi** od 0 do $n-1$. Takođe postoji $n-1$ dvosmjernih veza, numerisanih od 0 do $n-2$. Svaka veza povezuje dvije različite stanice. Dvije povezane stanice nazivaju se susjedima.

Put od stanice x do stanice y je sekvenca različitih stanica $a_0, a_1, \cdots, a_p$, tako da $a_0 = x$, $a_p = y$, i sve uzastopne stanice su susjedi. Postoji **tačno jedan** put od bilo koje stanice x do bilo koje druge stanice y.

Svaka stanica x može stvoriti paket (dio podataka) i poslati u bilo koju drugu stanicu y, koja se naziva **ciljna destinacija** paketa. Ovaj paket se mora usmjeravati jedinstvenim putem od x do y kao što dalje slijedi. Posmatrajmo stanicu z u kojoj se trenutno nalazi paket čija je ciljna destinacija y ($z \neq y$). U ovoj situaciji stanica z:

1. izvršava **postupak usmjeravanja** koji nalazi susjeda stanice z koji se nalazi na jedinstvenom putu od z do y, i
2. prosljeđuje paket tom susjedu.

Međutim, stanice imaju ograničenu memoriju i ne čuvaju kompletnu listu veza u SIB koju bi koristili u procesu usmjeravanja.

Tvoj zadatak je da implementiraš šemu usmjeravanja za SIB, koja se sastoji od dvije procedure.

- Prva procedura dobija n, listu veza u SIB-u i cijeli broj $k \geq n-1$ kao ulaz. ona dodjeljuje svakoj stanici **unikatan** cijeli broj u **segmentu** između 0 i k, uključivo.
- Druga procedura je procedura usmjeravanja,koja se raspoređuje na sve stanice nakon dodjele oznaka. Ova procedura dobija **samo** sljedeće ulaze:
    - s, **oznaka** stanice u kojoj se trenutno nalazi paket,
    - t, **oznaka** ciljne destinacije paketa ($t \neq s$),
    - c, lista **oznaka** svih susjeda stanice s.

    Ona treba vratiti **oznaku** jednog susjeda stanice s kojem treba dalje proslijediti paket.

U jednom podzadataku, rezultat vašeg rješenja zavisi od vrijednosti maksimalne oznake dodijeljene bilo kojoj stanici (uopšteno, manja je bolja).

Detalji implementacije

Trebate implementirati sljedeće procedure:

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

- n: broj stanica u SIB-u.
- k: maksimalna oznaka koja se smije koristiti.
- u i v: nizovi veličine $n-1$ koji opisuju veze. Za svako i ($0 \leq i \leq n-2$), veza i povezuje stanice označene sa $u[i]$ i $v[i]$.
- Ova procedura treba vratiti niz L veličine n. Za svako i ($0 \leq i \leq n-1$) $L[i]$ je oznaka dodijeljena stanici indeksiranoj sa i. Svi elementi niza L moraju biti jedinstveni i nalaziti se u intervalu od 0 do k, uključivo.

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

- s: oznaka stanice u kojoj se trenutno nalazi paket.
- t: oznaka finalalne destinacije paketa.
- c: niz koji sadrži listu oznaka svih susjeda stanice s. Niz c je sortiran u rastućem redoslijedu.
- Ova procedura treba vratiti oznaku susjeda stanice s u koju paket mora biti proslijeđen.

Svaki testni slučaj uključuje jedan ili više nezavisnih scenarija (tj. različitih SIB opise).

Za test slučaj koji uključuje r različitih scenarija, **program** koji poziva gornje procedure izvodi se tačno dva puta, kao što slijedi.

Tokom prvog pokretanja programa:

- procedura `label` je pozvana r puta,
- vraćene oznake se čuvaju od strane sistema za ocjenjivanje, i
- `find_next_station` se ne poziva.

Tokom drugog pokretanja programa:

- `find_next_station` se može pozvati više puta. U svakom pozivu, bira se **proizvoljan** scenarij, i oznake koje su vraćene iz procedure `label` u tom scenariju se koriste kao ulaz za `find_next_station`.
- `label` se ne poziva.

Posebno treba naglasit da nikoja informacija sačuvana u statičnim ili globalnim varijablama u prvom pokretanju programa nije dostupna unutar `find_next_station` procedure.

Primjer

Posmatrajmo sljedeći poziv:

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

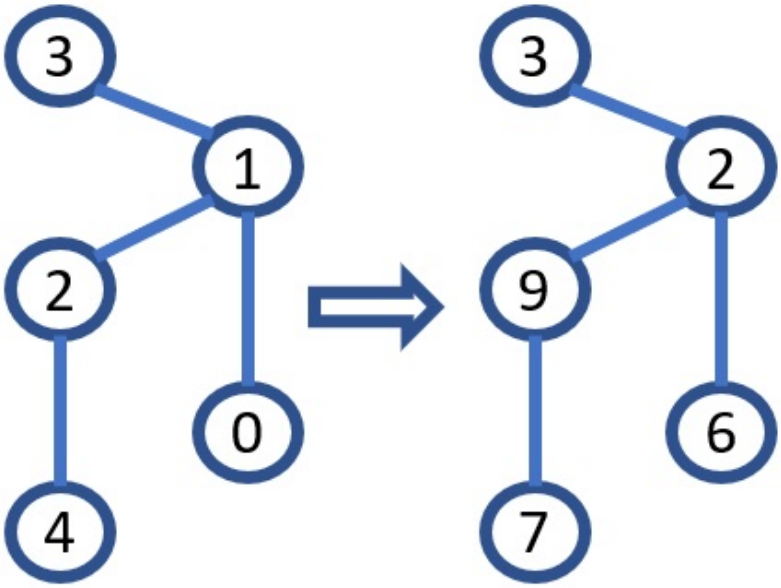
Postoje 5 stanica, i 4 veze između stanica sa indeksima $(0, 1)$, $(1, 2)$, $(1, 3)$ i $(2, 4)$. Svaka oznaka

može biti cijeli broj između 0 i $k = 10$.

Kako bi se prijavilo sljedeće označavanje:

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

procedura `label` treba vratiti $[6, 2, 9, 3, 7]$. Vrijednosti indeksa na prvom grafu (lijeva slika) i dodijeljene oznake (desna slika).



Pretpostavimo da su oznake dodijeljene kako je gore opisano i razmotrite sljedeći poziv:

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

Ovo znači da stanica koja sadrži paket ima oznaku 9, i finalna destinacija ima oznaku 6. Oznake stanica na putu do finalne destinacije su $[9, 2, 6]$. Dakle, procedura treba vratiti 2, oznaku stanice u koju paket mora biti proslijeđen (indeksirana sa 1).

Razmotrimo još jedan mogući poziv:

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

Procedura treba vratiti 3, pošto je stanica sa oznakom 3 susjed stanice sa oznakom 2, i stoga treba

direktno prihvatiti paket.

Ograničenja

- $1 \leq r \leq 10$

Za svaki poziv procedure `label`:

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

Za svaki poziv procedure `find_next_station`, ulaz dolazi iz proizvoljno odabranog prethodnog poziva procedure `label`. Posmatrajmo oznake koje je proizvela. Onda:

- s i t su oznake od dvije različite stanice.
- c je sekvenca svih oznaka susjeda stanice sa oznakom s, u rastućem redoslijedu.

Za svaki testni slučaj, ukupna dužina svih nizova c proslijeđenih proceduri `find_next_station` neće prelaziti $100\ 000$ za sve scenarije zajedno.

Podzadaci

1. (5 bodova) $k = 1000$, ni jedna stanica nema više od 2 susjeda.
2. (8 bodova) $k = 1000$, veza i povezuje stanice $i + 1$ i $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 bodova) $k = 1\ 000\ 000$, najviše jedna stanica ima preko 2 susjeda.
4. (10 bodova) $n \leq 8$, $k = 10^9$
5. (61 bod) $k = 10^9$

U podzadatku 5 možete dobiti parcijalne bodove. Neka je m maksimalna oznaka vraćena iz procedure `label` kroz sve scenarije. Tvoji bodovi za ovaj podzadatak se računaju po sljedećoj tabeli:

Maksimalna oznaka	Bodovi
$m \geq 10^9$	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

Sample grader

Sample grader čita ulaz u sljedećem formatu:

- linija 1: r

slijedi r blokova, gdje svaki opisuje jedan scenario. Format svakog bloka je sljedeći:

- linija 1: $n \ \ k$
- linija $2 + i$ ($0 \leq i \leq n - 2$): $u[i] \ \ v[i]$
- linija $1 + n$: q: broj poziva procedure `find_next_station`.
- linija $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j] \ \ y[j] \ \ w[j]$: **indeksi** stanica uključenih u j-ti poziv procedure `find_next_station`. U stanici $z[j]$ se trenutno nalazi palet, stanica $y[j]$ je finalna destinacija paketa, i u stanicu $w[j]$ se treba proslijediti paket.

Sample grader ispisuje rezultate u sljedećem formatu:

- linija 1: m

r blokova koji odgovaraju uzastopnim scenarijima u ulazu. Format svakog bloka je sljedeći:

- linija $1 + j$ ($0 \leq j \leq q - 1$): **indeks** stanice, čija je **oznaka** vraćena u j-tom pozivu procedure `find_next_station` u ovom scenariju.

Imajte na umu da svako pokretanje sample gradera poziva i `label` i `find_next_station` proceduru.