Stacje (stations)

Singapurska Sieć Szkieletowa (SSS) składa się z n stacji o **indeksach** od 0 do $n-1$. Sieć zawiera też $n-1$ dwukierunkowych połączeń ponumerowanych od 0 do $n-2$. Każde połączenie łączy dwie różne stacje. Każde dwie połączone stacje nazywamy sąsiadami.

Ścieżka ze stacji x do stacji y to ciąg różnych stacji $a_0, a_1, \cdots, a_p$ taki, że $a_0 = x$, $a_p = y$ oraz każde dwie sąsiednie stacje znajdujące się na ścieżce są sąsiadami. Dla każdej pary różnych stacji x, y, istnieje **dokładnie jedna** ścieżka ze stacji x do stacji y.

Dowolna stacja x może stworzyć pakiet (informację) i przesłać go do dowolnej innej stacji y, którą nazwiemy **stacją docelową** pakietu. Pakiet musi być przesłany wzdłuż jedynej ścieżki ze stacji do x do y w następujący sposób. Rozważmy stację z, która obecnie posiada pakiet, którego stacją docelową jest y ($z \neq y$). Wtedy stacja z:

1. Wykonuje **procedurę trasowania**, która wyznacza tego sąsiada stacji z, który leży na jedynej ścieżce z z do y.
2. Następnie, przekazuje pakiet wyznaczonej stacji.

Niestety, stacje mają ograniczoną pamięć operacyjną i nie mogą zapamiętać wszystkich połączeń zdefiniowanych w SSS.

Implementujesz schemat trasowania dla SSS, który będzie składać się z dwóch funkcji:

- Pierwsza funkcja dostaje jako parametry liczbę n, listę połączeń w SSS oraz liczbę naturalną $k \geq n-1$. Funkcja ta przydziela wszystkim stacjom **różne etykiety** pomiędzy 0 i k, włącznie.
- Druga funkcja to procedura trasowania, która zostanie umieszczona na wszystkich stacjach po przypisaniu etykiet stacjom. Funkcja otrzymuje **jedynie** poniższe parametry:
  - s: **etykietę** stacji, która w danym momencie posiada pakiet;
  - t: **etykietę** stacji docelowej pakietu ($t \neq s$);
  - c: listę **etykiet** wszystkich sąsiadów stacji s.

  Funkcja powinna zwrócić **etykietę** sąsiada s, któremu powinien zostać przekazany pakiet.

W jednym podzadaniu wynik Twojego zgłoszenia będzie zależał od maksymalnej wartości etykiety przypisanej do którejkolwiek stacji; mniejsze wartości etykiet są lepsze.

Szczegóły implementacji

Zaimplementuj poniższe funkcje:

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

- n: liczba stacji w SSS.
- k: maksymalna wartość etykiety, której możesz użyć.
- u, v: tablice rozmiaru $n-1$ opisujące połączenia. Dla każdego i ($0 \leq i \leq n-2$), połączenie i łączy stacje o indeksach $u[i]$ oraz $v[i]$.
- Ta funkcja powinna zwrócić pojedynczą tablicę L rozmiaru n. Dla każdego i ($0 \leq i \leq n-1$), $L[i]$ powinno być etykietą przypisaną stacji o indeksie i. Wszystkie elementy tablicy L muszą być różne oraz równe co najmniej 0 i co najwyżej k.

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

- s: etykieta stacji, która w danym momencie posiada pakiet.
- t: etykieta stacji docelowej pakietu.
- c: tablica zawierająca listę etykiet wszystkich sąsiadów s. Ta tablica jest uporządkowana rosnąco.
- Ta funkcja powinna zwrócić etykietę sąsiada s, któremu powinien zostać przekazany pakiet.

Każdy zestaw testowy składa się z jednego lub więcej niezależnych scenariuszy testowych (to jest, różnych opisów SSS). Dla zestawu testowego składającego się z r scenariuszy testowych, **program** uruchamiający powyższe procedury jest uruchomiony dwukrotnie, w sposób opisany poniżej.

W trakcie pierwszego uruchomienia programu:

- Funkcja `label` jest wywołana r-krotnie.
- Etykiety zwrócone przez wywołania funkcji `label` są zapisywane w systemie sprawdzającym.
- Funkcja `find_next_station` nie jest wywoływana.

W trakcie drugiego uruchomienia programu:

- Funkcja `find_next_station` może zostać uruchomiona wielokrotnie. W każdym wywołaniu tej funkcji, program sprawdzający wybiera **dowolny** scenariusz testowy i używa etykiet zwróconych przez funkcję `label` w tym scenariuszu jako argumentów funkcji `find_next_station`.
- Funkcja `label` nie jest wywoływana.

Zwróć uwagę na to, że informacje zapisane w zmiennych statycznych lub zmiennych globalnych w trakcie pierwszego wywołania programu nie będę dostępne w trakcie żadnego wywołania funkcji `find_next_station`.

Przykład

Rozważmy następujące wywołanie funkcji:

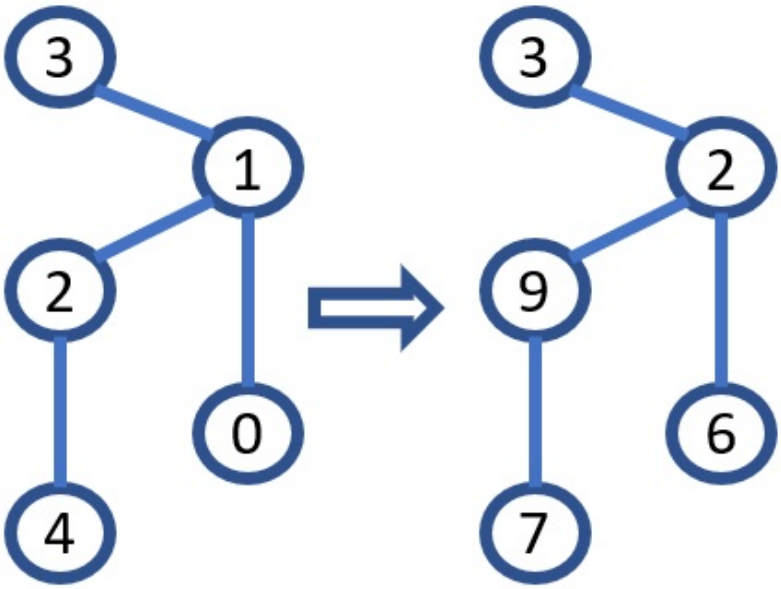
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

SSS zawiera 5 stacji oraz 4 dwukierunkowe połączenia łączące pary stacji o indeksach: $(0, 1)$, $(1, 2)$, $(1, 3)$ i $(2, 4)$. Funkcja `label` może przypisywać stacjom indeksy z przedziału od 0 do $k = 10$ włącznie.

Przypuśćmy, że chcemy zwrócić poniższe etykietowanie stacji:

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

W takiej sytuacji funkcja `label` powinna zwrócić $[6, 2, 9, 3, 7]$. Liczby na schemacie poniżej opisują indeksy (po lewej) i etykiety (po prawej) stacji SSS:



Przypuśćmy, że etykiety zostały przypisane serwerom w sposób opisany powyżej. Rozważmy teraz następujące wywołanie funkcji:

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

Oznacza to, że stacja, która posiada obecnie pakiet, ma etykietę 9, zaś docelowa stacja dla pakietu ma etykietę 6. Etykiety stacji na ścieżce do stacji docelowej to $[9, 2, 6]$. Wobec tego, funkcja powinna zwrócić wartość 2, która jest etykietą stacji, do której powinien zostać przekazany pakiet.

(Stacja o etykiecie 2 ma indeks 1.)

Rozważmy inne możliwe wywołanie funkcji:

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

Tutaj funkcja powinna zwrócić 3, gdyż stacja docelowa o etykiecie 3 sąsiaduje ze stacją o indeksie 2. Dlatego też stacja docelowa może otrzymać pakiet bezpośrednio od stacji o indeksie 2.

Ograniczenia

- $1 \leq r \leq 10$

Dla każdego wywołania funkcji `label`:

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

Dla każdego wywołania funkcji `find_next_station`, argumenty pochodzą z wywołania funkcji `label` w dowolnie wybranym scenariuszu testowym. Rozważmy etykiety wyprodukowane przez funkcję `label`. Wtedy:

- s oraz t są etykietami dwóch różnych stacji.
- c jest ciągiem wszystkich etykiet sąsiadów stacji o etykiecie s, uporządkowanym rosnąco.

W żadnym zestawie testowym łączna długość wszystkich tablic c przekazanych do funkcji `find_next_station` nie przekroczy $100\ 000$ (dla wszystkich scenariuszy testowych w zestawie łącznie).

Podzadania

1. (5 punktów) $k = 1000$, każda stacja ma co najwyżej 2 sąsiadów.
2. (8 punktów) $k = 1000$, połączenie o numerze i łączy stacje o indeksach $i + 1$ oraz $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 punktów) $k = 1\ 000\ 000$, co najwyżej jedna stacja ma więcej niż 2 sąsiadów.
4. (10 punktów) $n \leq 8$, $k = 10^9$
5. (61 punktów) $k = 10^9$

Twoje zgłoszenie może otrzymać częściowy wynik za podzadanie 5. Niech m będzie maksymalną etykietą zwróconą przez `label` we wszystkich scenariuszach testowych w tym podzadaniu. Twój wynik za podzadanie zostanie obliczony zgodnie z poniższą tabelką:

Maksymalna etykieta	Wynik
$m \geq 10^9$	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

Przykładowy program oceniający

Przykładowy program oceniający wczytuje wejście w następującym formacie:

- wiersz 1: r

Po nim następuje r bloków opisujących kolejne scenariusze testowe. Każdy blok ma następujący format:

- wiersz 1: $n \ \ k$
- wiersze $2 + i$ $(0 \leq i \leq n - 2)$: $u[i] \ \ v[i]$
- wiersz $1 + n$: q: liczba wywołań funkcji `find_next_station` w tym scenariuszu testowym.
- wiersze $2 + n + j$ $(0 \leq j \leq q - 1)$: $z[j] \ \ y[j] \ \ w[j]$: **indeksy** stacji w j-tym wywołaniu funkcji `find_next_station`. Stacja o indeksie $z[j]$ obecnie posiada pakiet, którego stacja docelowa ma indeks $y[j]$. W tym zapytaniu, stacja $z[j]$ powinna przekazać pakiet stacji o indeksie $w[j]$.

Przykładowy program oceniający wypisuje wyjście w następującym formacie:

- wiersz 1: m

Po nim następuje r bloków odpowiadających kolejnym scenariuszom testowym. Każdy blok ma następujący format:

- wiersze $1 + j$ $(0 \leq j \leq q - 1)$: **indeks** stacji, którego **etykieta** została zwrócona przez j-te wywołanie funkcji `find_next_station` w tym scenariuszu testowym.

Zwrć uwagę na to, że przykładowy program oceniający wywołuje zarówno funkcję `label`, jak i `find_next_station`.