# Kanalizaciona mreža (simurgh)

Parsian Evin hotel u Teheranu je poznat po kvalitetnoj kanalizacionoj mreži. Mreža se sastoji od $n$ čvorišta, označenih brojevima od $0$ do $n-1$, i $m$ dvosmernih cevi, označenih brojevima od $0$ do $m-1$. Svaka cev povezuje par različitih čvorišta, a svaka dva čvorišta su povezana preko najviše jedne cevi.

Talentovani ali buntovni Filipd je posumnjao u vrhunski kvalitet ove kanalizacione mreže, i odmah je ličnim primerom pokazao kako je moguće zapušiti neke cevi. Cevi koje je Filipd zapušio zvaćemo *zapušene* cevi. Menadžment hotela bi želeo da pronađe koje su cevi *zapušene*, i za tu priliku su unajmili Vladada, poznatog kanalizacionog detektiva, da im pomogne.

Vladad je specijalista za *braon skupove*. Neki skup cevi se naziva *braon skup* ako i samo ako:

1. skup ima tačno $n-1$ cev
2. za svaki par čvorišta, nešto može doplutati iz jednog do drugog čvorišta plutajući samo kroz cevi iz ovog skupa.

Vladad poseduje kompletnu mapu kanalizacione mreže. Nije mu poznato koje su cevi zapušene, ali zna da skup svih zapušenih cevi čini braon skup.

U toku svog istraživanja, Vladad može raditi testove opterećenja kanalizacije. Svaki test opterećenja se sastoji od sledećeg:

1. Vladad bira *braon skup* cevi nad kojim radi dodatno opterećenje, i zatim
2. na osnovu visine vode u wc šolji utvrđuje koliko ima *zapušenih* cevi (koje je Filipd zapušio) u *braon skupu* koji je Vladad izabrao.

Vaš program treba da pomogne Vladadu da pronađe sve cevi *zapušene* Filipdovom majstorijom, vršeći najviše $q$ testova opterećenja.

## Detalji implementacije

Implementirajte sledeću funkciju:

```
int[] find_roads(int n, int[] u, int[] v)
```

- $n$: broj čvorišta
- $u$ i $v$: nizovi dužine $m$. Za svako $0 \leq i \leq m-1$, $u[i]$ i $v[i]$ su čvorišta koje povezuje cev $i$.
- Ova funkcija vraća niz dužine $n-1$ koji sadrži oznake zapušenih cevi (u proizvoljnom poretku).

Vaše rešenje može najviše $q$ puta pozvati sledeću funkciju grejdera:

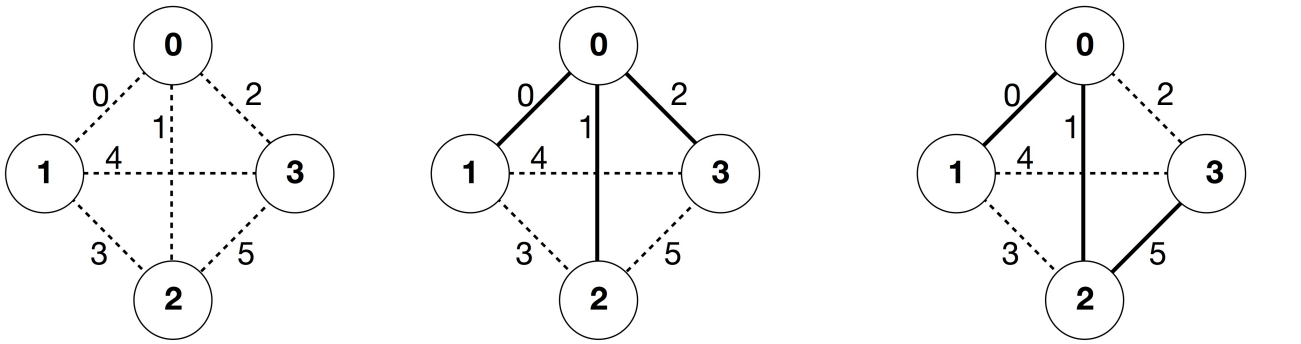
```
int[] count_common_roads(int[] r)
```

- $r$: niz dužine $n - 1$ koji sadrži oznake cevi u *braon skupu* (u proizvoljnom poretku) za koji se radi test opterećenja.
- Ova funkcija vraća broj *zapušenih cevi* u skupu predstavljenim nizom $r$

# Primer

```
find_roads(4, [0, 0, 0, 1, 1, 2], [1, 2, 3, 2, 3, 3])
```

`find_roads( ... )` `count_common_roads([0, 1, 2]) = 2` `count_common_roads([5, 1, 0]) = 3`




U ovom primeru postoje $4$ čvorišta i $6$ cevi. Označimo sa $(a, b)$ cev koje povezuje čvorišta $a$ i $b$. Cevi su numerisane od $0$ do $5$ u sledećem poretku: $(0, 1)$, $(0, 2)$, $(0, 3)$, $(1, 2)$, $(1, 3)$ i $(2, 3)$. Svaki *braon skup* ima $n - 1 = 3$ cevi.

Prepostavimo da su cevi $0$, $1$ i $5$ (tj. cevi $(0, 1)$, $(0, 2)$ i $(2, 3)$) *zapušene*, i da program izvršava sledeće pozive:

- `count_common_roads([0, 1, 2])` vraća $2$. Ovaj test opterećenja je nad cevima čije su oznake $0$, $1$ i $2$ (tj. cevi $(0, 1)$, $(0, 2)$ i $(0, 3)$). Dve od njih su *zapušene*.
- `count_common_roads([0, 1, 2])` vraća $3$. Ovaj test opterećenja se vrši baš na svim *zapušenim* cevima.

Funkcija `find_roads` treba da vrati `[5, 1, 0]` ili bilo koji drugi niz dužine $3$ koji sadrži ta tri elementa.

Obratite pažnju da sledeći pozivi nisu dopušteni:

- `count_common_roads([0, 1])`: dužina niza $r$ nije $3$.
- `count_common_roads([0, 1, 3])`: u ovom slučaju niz $r$ ne opisuje *braon skup*, jer nije moguće da nešto dopluta iz čvorišta $0$ do čvorišta $3$ koristeći samo cevi $(0, 1)$, $(0, 2)$ i $(1, 2)$

# Ograničenja

- $2 \leq n \leq 500$.
- $n-1 \leq m \leq n(n-1)/2$.
- $0 \leq u[i], v[i] \leq n-1$, za svako $0 \leq i \leq m-1$)
- Za svako $0 \leq i \leq m-1$, cev $i$ povezuje dva različita čvorišta (tj. $u[i] \neq v[i]$).
- Postoji najviše jedna cev između bilo kog para čvorišta.
- Moguće je da nešto dopluta od bilo kog čvorišta do bilo kog drugog čvorišta koristeći date cevi.
- Skup svih zapušenih cevi jeste braon skup.
- `find_roads` može pozivati funkciju `find_common_roads` najviše $q$ puta. Pri svakom pozivu, skup opisan nizom $r$ mora biti braon skup.

## Podzadaci

1. (13 bodova) $n \leq 7$, $q = 30\,000$
2. (17 bodova) $n \leq 50$, $q = 30\,000$
3. (21 bod) $n \leq 240$, $q = 30\,000$
4. (19 bodova) $q = 12\,000$ i postoji cev između svaka dva čvorišta.
5. (30 bodova) $q = 8000$

## Primer grejdera

Primer grejdera čita ulazne podatke u sledećem formatu:

- red $1$: $n$ $m$
- red $2 + i$ (za svako $0 \leq i \leq m-1$): $u[i]$ $v[i]$
- red $2 + m$: $s[0]$ $s[1]$ $\ldots$ $s[n-2]$

Ovde su $s[0], s[1], \ldots, s[n-2]$ oznake zapušenih cevi.

Primer grejdera daje izlaz `YES` ako funkcija `find_roads` poziva `count_common_roads` najviše $30\,000$ puta i vraća tačan skup zapušenih cevi. U suprotnom, grejder daje izlaz `NO`.

Obratite pažnju da `count_common_roads` u primeru grejdera ne proverava da li $r$ opisuje braon skup. Umesto toga, samo prebrojava koliko ima zapušenih cevi u nizu $r$ i vraća njihov broj. Međutim, ako vaš program poziva `count_common_roads` sa skupom oznaka koji ne opisuje braon skup, CMS će vam vratiti 'Wrong Answer'.

## Tehnička napomena

Funkcija `count_common_roads` u C++ i Pascal-u radi efikasnosti koristi *prosleđivanje parametara po referenci* (engl. *call by reference*). Vi je možete pozivati na uobičajeni način. Garantuje se da grejder neće menjati vrednosti u nizu $r$.