Tisočeri otoki

Tisočere otoke imenujemo skupino čudovitih otokov v Javanskem morju. Sestavljeni so iz N otokov, oštevilčenih od 0 do $N-1$.

Na voljo je M kanujev, oštevilčenih od 0 to $M-1$, s katerimi lahko plujete med otoki. Za vsak i, tako da velja $0 \leq i \leq M-1$, je kanu i lahko zasidran na otoku $U[i]$ ali $V[i]$ in se lahko uporablja za plovbo med otokoma $U[i]$ in $V[i]$. Natančneje, ko je kanu zasidran na otoku $U[i]$, ga lahko uporabimo za plovbo od otoka $U[i]$ do otoka $V[i]$, kjer ga nato na otoku $V[i]$ zasidramo. Podobno, ko je kanu zasidran na otoku $V[i]$, ga lahko uporabimo za plovbo od otoka $V[i]$ do otoka $U[i]$, kjer ga nato na otoku $U[i]$ zasidramo. Sprva je kanu zasidran na otoku $U[i]$. Za plutje med istim parom otokov lahko uporabljamo več kanujev. Poleg tega je na istem otoku lahko zasidranih več kanujev.

Zaradi varnosti je kanu po vsakem jadranju podvržen vzdrževanju, kar onemogoča, da bi se s kanujem peljali dvakrat zaporedoma. To pomeni, da moramo po uporabi kanuja i najprej uporabiti neki drug kanu, preden lahko spet uporabimo kanu i.

Ana načrtuje potovanje po nekaterih izmed otokov. Njeno potovanje je **veljavno** samo, če so izpolnjeni naslednji pogoji.

- Svoje potovanje začne in konča na otoku 0.
- Na potovanju poleg otoka 0 obišče vsaj še en otok.a
- Po končanem potovanju je vsak kanu zasidran na istem otoku, kot je bil zasidran pred potovanjem, t.j. kanu i, za vsak i, tako da je $0 \leq i \leq M-1$ mora biti zasidran na otoku $U[i]$.

Pomagajte Ani najti veljavno potovanje, ki vključuje jadranje največ $2\,000\,000$-krat, oziroma ugotovite, da tako veljavno potovanje ne obstaja. Obstaja dokaz, da pod omejitvami, določenimi v tej nalogi (glejte razdelek Omejitve), če obstaja veljavno potovanje, obstaja tudi veljavno potovanje, ki ne vključuje več kot $2\,000\,000$ jadranj.

Podrobnosti izvedbe

Napisati morate naslednjo funkcijo:

```
union(bool, int[]) find_journey(int N, int M, int[] U, int[] V)
```

- N: število otokov.
- M: število kanujev.

- U, V: polji dolžine M, ki določata kanuje.
- Ta funkcija mora vrniti bodisi vrednost tipa boolean bodisi polje celih števil.
  - Če veljavno potovanje ne obstaja, mora funkcija vrniti `false`.
  - Če obstaja veljavno potovanje obstaja, sta dve možnosti:
  - Za polni izkupiček točk, mora funkcija vrniti polje največ $2\,000\,000$ celih števil, ki predstavlja veljavno potovanje. Bolj natančno, vsi elementi tega polja morajo biti številke kanujev, ki so bili uporabljeni na potovanju (po vrstnem redu, kot so se uporabili). Če funkcija bodisi vrne `true`, bodisi vrne polje polje z več kot $2\,000\,000$ celimi števili, bodisi vrne polje celih števil, ki pa ne opisujejo veljavnega potovanja, za rešitev prejmete delni izkupiček točk (glejte razdelek Podnaloge).
- Ta funkcija se kliče natančno enkrat.

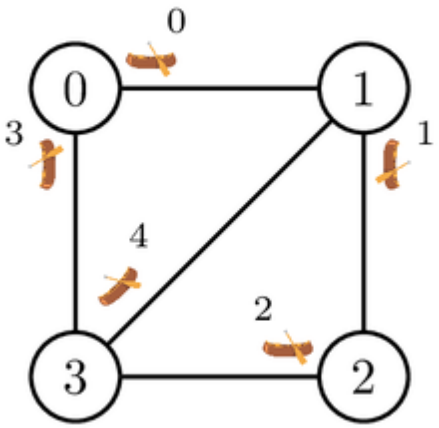
Primeri

Primer 1

Recimo, da kličemo funkcijo na način:

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

Otoki in razporeditev kanujev je prikazana na spodnji sliki.



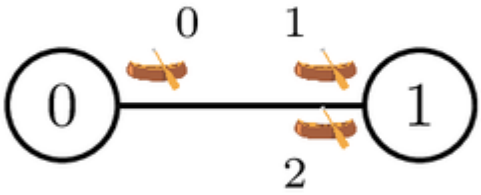
Eden izmed možnih veljavnih načinov potovanja je naslednji. Ana najprej pluje s kanuji 0, 1, 2 in 4. Posledično pripluje do otoka 1. Na tem mestu Ana spet lahko uporabi kanu 0, saj je trenutno zasidran na otoku 1 in zadnji kanu, ki ga je uporabila, ni kanu 0. Po ponovnem plutju s kanujem 0, se Ana sedaj nahaja na otoku 0. Vendar kanuji 1, 2 in 4 niso zasidrani na istih otokih, kot so bili pred potovanjem. Ana nato nadaljuje svojo pot s kanuji 3, 2, 1, 4 in spet 3. Ana se je vrnila na otok 0 in vsi kanuji so zasidrani na istih otokih kot pred potovanjem. To pomeni, da je polje $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ veljavna vrnjena vrednost funkcije.

Primer 2

Recimo, da imamo klic:

```
find_journey(2, 3, [0, 1, 1], [1, 0, 0])
```

Otoki in razporeditev kanujev je prikazana na spodnji sliki.



Ana lahko začne potovanje samo s kanujem 0, potem pa lahko za plutje uporabi bodisi kanu 1 bodisi kanu 2. Upoštevajte, da ne more dvakrat zapored jadrati s kanujem 0. V obeh primerih je Ana spet na otoku 0. Vendar kanuji niso zasidrani na istih otokih, kot so bili pred potovanjem. Poleg tega Ana ne more odpluti z nobenim kanujem, saj je edini kanu, ki je zasidran na otoku 0, tisti, ki ga je pravkar uporabila. Ker ne obstaja veljavno potovanje, mora funkcija vrniti vrednost `false`.

Omejitve

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N-1$ in $0 \leq V[i] \leq N-1$ (za vsak i, tako da je $0 \leq i \leq M-1$)
- $U[i] \neq V[i]$ (za vsak i, tako da je $0 \leq i \leq M-1$)

Podnaloge

1. (5 točk) $N = 2$
2. (5 točk) $N \leq 400$. Za vsak par ločenih (distinct) otokov x and y ($0 \leq x < y \leq N-1$), obstajata natanko dva kanuja, s katerima je mogoče pluti med njimi. Eden izmed njih je zasidran na otoku x, drugi pa na otoku y.
3. (21 točk) $N \leq 1000$, M je sodo, in za vsak **sod** i, tako da je $0 \leq i \leq M-1$, kanuja i in $i+1$ lahko oba uporabljamo za plutje med otokoma $U[i]$ in $V[i]$. Kanu i je prvotno zasidran na otoku $U[i]$ kanu $i+1$ pa je prvotno zasidran na otoku $V[i]$. Formalno velja: $U[i] = V[i+1]$ in $V[i] = U[i+1]$.
4. (24 točk) $N \leq 1000$, M je sodo, in za vsak **sod** i, tako da je $0 \leq i \leq M-1$, kanuja i in $i+1$ lahko oba uporabljamo za plutje med otokoma $U[i]$ in $V[i]$. Oba kanuja sta sprva zasidrana na otoku $U[i]$. Formalno velja: $U[i] = U[i+1]$ in $V[i] = V[i+1]$.

5. (45 točk) Brez dodatnih omejitev.

Za vsak testni primer, v katerem obstaja veljavna pot, vaša rešitev:

- dobi vse točke, če vrne veljavno potovanje,
- dobi 35% točk, če vrne `true`, polje z več kot $2\ 000\ 000$ celimi števili ali polje, ki ne opisuje veljavnega potovanja,
- dobi 0 točk sicer.

Za vsak testni primer, v katerem veljavno potovanje ne obstaja, vaša rešitev:

- dobi vse točke, če vrne `false`,
- dobi 0 točk sicer.

Upoštevajte, da je končni rezultat za vsako podnalogo najmanjše število točk za testne primere v podnalogi.

Vzorčni ocenjevalnik

Vzorčni ocenjevalnik bere vhod v naslednjem formatu:

- vrstica 1: $N\ M$
- vrstica $2 + i$ ($0 \leq i \leq M - 1$): $U[i]\ V[i]$

Vzorčni ocenjevalnik izpiše vaše rezultate v naslednjem formatu

- če `find_journey` vrne `bool`:
  - vrstica 1: 0
  - vrstica 2: 0 če `find_journey` vrne `false`, ali 1 sicer.
- Če `find_journey` vrne `int[]`, označimo elemente polja z $c[0], c[1], \ldots c[k-1]$. Vzorčni ocenjevalnik izpiše:
- If `find_journey` returns a `bool`:
  - vrstica 1: 0
  - vrstica 2: 0 if `find_journey` returns `false`, or 1 otherwise.
- If `find_journey` returns an `int[]`, denote the elements of this array by $c[0], c[1], \ldots c[k-1]$ respectively. Vzorčni ocenjevalnik izpiše:
  - vrstica 1: 1
  - vrstica 2: k
  - vrstica 1: 1
  - vrstica 2: k
  - vrstica 3: $c[0]\ c[1]\ \ldots\ c[k-1]$