Tūkstančiai salų

Tūkstančiai salų yra nuostabių Javos jūroje esančių salų grupė. Ją sudaro N salų, sunumeruotų nuo 0 iki $N-1$.

Iš vienos salos į kitą plaukiama kanojomis. Yra M kanojų, sunumeruotų nuo 0 iki $M-1$. Kiekvienam i ($0 \leq i \leq M-1$), i-oji kanoja gali būti prišvartuota arba $U[i]$-oje, arba $V[i]$-oje saloje, ir ja galima plaukioti tarp $U[i]$-osios ir$V[i]$-osios salų. Kitaip sakant, jei kanoja prišvartuota $U[i]$-oje saloje, ja galima nuplaukti iš $U[i]$-osios salos į $V[i]$-ąją salą. Nuplaukus, kanoja prišvartuojama $V[i]$-oje saloje. Analogiškai, kai kanoja prišvartuota $V[i]$-oje saloje, ja galima nuplaukti iš $V[i]$-osios salos į $U[i]$-ąją salą, o nuplaukus, kanoja prišvartuojama $U[i]$-oje saloje. Pradiniu momentu kanoja prišvartuota $U[i]$-oje saloje. Tarp tos pačios salų poros gali plaukioti daug kanojų. Taip vienoje saloje gali būti prišvartuota daug kanojų.

Po kiekvieno plaukimo būtina atlikti kanojos techninę apžiūrą, todėl kanoja negali plaukti du kartus iš eilės. Tai reiškia, kad nuplaukus kuria nors kanoja i, būtina plaukti kita kanoja prieš vėl plaukiant kanoja i.

Bu Dengklek planuoja savo maršrutą per salas. Jos maršrutas yra **korektiškas** tada ir tik tada, kai tenkinamos šios sąlygos:

- Maršrutas prasideda ir baigiasi 0-inėje saloje.
- Aplankoma nors viena kita sala be 0-inės.
- Kelionei pasibaigus kiekviena kanoja lieka prišvartuota ten, kuri ji buvo prišvartuota prieš kelionę. T.y., i-oji kanoja kiekvienam i tokiam, kad $0 \leq i \leq M-1$, turi būti prišvartuota $U[i]$-oje saloje.

Suplanuokite Bu Dengklek korektišką maršrutą, tokį, kad jį sudarytų ne daugiau $2\,000\,000$ perplaukimų, arba nustatykite, kad korektiško maršruto sudaryti neįmanoma.

Galima įrodyti, kad esant *Ribojimų* skyrelyje nurodytiems ribojimams, jei apskritai egzistuoja korektiškas maršrutas, tuomet egzistuoja toks korektiškas maršrutas, kurį sudarytų ne daugiau $2\,000\,000$ perplaukimų.

Realizacija

Parašykite tokią funkciją:

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

- N: salų skaičius.
- M: kanojų skaičius
- U, V: kanojas nusakantys M ilgio masyvai.

- Funkcija turi grąžinti arba loginę reikšmę, arba sveikųjų skaičių masyvą.
    - Jei korektiškas maršrutas neegzistuoja, funkcija turi grąžinti `false`.
    - Jei korektiškas maršrutas egzistuoja, turite du pasirinkimus:
        - norint gauti visus taškus, funkcija turi grąžinti sveikųjų skaičių masyvą, ne ilgesnį nei $2\ 000\ 000$, nusakantį korektišką maršrutą. Kitaip sakant, masyvą turi sudaryti kanojų numeriai, pateikti ta tvarka, kuria buvo plaukiama.
        - norint gauti dalinius taškus, funkcija turi grąžinti `true`, masyvą, ilgesnį nei $2\ 000\ 000$, arba masyvą, nenusakantį korektiško maršruto, (žr. skyrelį *Dalinės užduotys*).
- Ši funkcija iškviečiama lygiai vieną kartą.

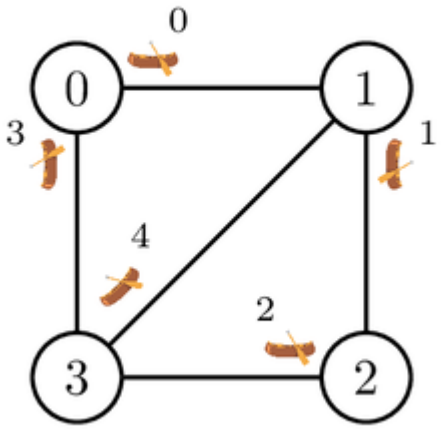
Pavyzdžiai

Pavyzdys nr. 1

Panagrinėkime tokį pavyzdį:

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

Žemiau pavaizduotos salos ir kanojos.



Galimas toks sprendinys. Bu Dengklek plaukia kanojomis, kurių numeriai iš eilės 0, 1, 2, ir 4. Ji atsiduria 1-oje saloje. Tuomet Bu Dengklekg gali vėl plaukti 0-ine kanoja, nes ši prišvartuota 1-oje saloje, o paskutinį kartą ji plaukė ne 0-ine kanoja. Nuplaukusi 0-ine kanoja, Bu Dengklek atsiduria 0-inėje saloje. Tačiau kanojos, kurių numeriai 1, 2 ir 4 nėra prišvartuotos tose pačiose salose, kuriose buvo pradiniu momentu. Todėl Bu Dengklek tęsia kelionę ir plaukia kanojomis 3, 2, 1, 4, ir vėl 3. Bu Dengklek sugrįžta į 0-inę salą, o visos kanojos yra prišvartuotos tose salose, kuriose jos buvo prieš kelionę.

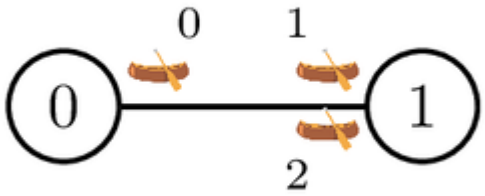
Taigi, grąžinamas masyvas $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ nusako korektišką maršrutą.

Pavyzdys nr. 2

Panagrinėkime tokį iškvietimą:

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

Salos ir kanojos pavaizduotos žemiau esančiame paveikslėlyje.



Bu Dengklek gali pradėti keliauti tik 0-ine kanoja, o tuomet ji gali plaukti arba 1-aja, arba 2-aja kanoja. Atkreipkite dėmesį, kad ji negali du kartus iš eilės plaukti 0-ine kanoja. Abiem atvejais Bu Dengklek grįžta į 0-inę salą. Tačiau kanojos nėra prišvartuotos salose, kuriose buvo pradiniu momentu ir toliau Bu Dengklek negali plaukti jokia kanoja, nes ką tik atplaukė vienintele 0-inėje saloje esančia kanoja. Kadangi neįmanoma suplanuoti korektiško maršruto, funkcija turi grąžinti `false`.

Ribojimai

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N - 1$ ir $0 \leq V[i] \leq N - 1$ (kiekvienam i tokiam, kad $0 \leq i \leq M - 1$)
- $U[i] \neq V[i]$ (kiekvienam i tokiam, kad $0 \leq i \leq M - 1$)

Dalinės užduotys

1. (5 taškai) $N = 2$
2. (5 taškai) $N \leq 400$. Kiekvienai skirtingų salų porai x ir y ($0 \leq x < y \leq N - 1$), yra lygiai dvi kanojos, kuriomis galima plaukioti tarp šių salų. Viena jų prišvartuota saloje x, kita - saloje y.
3. (21 taškas) $N \leq 1000$, M yra lyginis, ir kiekvienam **lyginiam** i tokiam, kad $0 \leq i \leq M - 1$, i-ąja ir $(i + 1)$-ąja kanojomis plaukiojama tarp $U[i]$-osios ir $V[i]$-osios salų. i-oji kanoja pradiniu momentu prišvartuota $U[i]$-oje saloje, o $(i + 1)$-oji kanoja pradiniu momentu prišvartuota $V[i]$-oje saloje. Kitaip sakant, $U[i] = V[i + 1]$ ir $V[i] = U[i + 1]$.
4. (24 taškai) $N \leq 1000$, M yra lyginis, ir kiekvienam **lyginiam** i tokiam, kad $0 \leq i \leq M - 1$, i-ąja ir $(i + 1)$-ąja kanojomis galima plaukti tarp $U[i]$-osios ir $V[i]$-osios salų. Abi kanojos pradiniu momentu prišvartuotos $U[i]$-oje saloje. Kitaip sakant, $U[i] = U[i + 1]$ ir $V[i] = V[i + 1]$.
5. (45 taškai) Papildomų ribojimų nėra.

Kiekvienam testui, kuriam egzistuoja korektiškas maršrutas, jūsų sprendimas:

- gauna visus taškus, jei grąžina korektišką maršrutą,
- gauna 35% taškų, jei grąžina `true`, sveikųjų skaičių masyvą, ilgesnį nei $2\,000\,000$, arba masyvą, kuris nenusako korektiško maršruto,
- gauna 0 taškų priešingu atveju.

Kiekvienam testui, kuriam neegzistuoja korektiškas maršrutas, jūsų sprendimas:

- gauna visus taškus, jei grąžina `false`,
- gauna 0 taškų kitu atveju.

Atkreipkite dėmesį, kad taškų skaičius skiriamas už dalinę užduotį lygus minimaliam taškų skaičiui, gautam už tos dalinės užduoties testus.

Pavyzdinė vertinimo programa

Pavyzdinė vertinimo programa skaito duomenis tokiu formatu:

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

Pavyzdinė vertinimo programa išveda duomenis tokiu formatu:

- Jei `find_journey` grąžina `bool`:
  - 1-oji eilutė: 0
  - 2-oji eilutė: 0 jei `find_journey` grąžina `false`, ar 1 kitu atveju.
- Jei `find_journey` grąžina `int[]`, o masyvo elementus pažymėsime $c[0], c[1], \ldots c[k - 1]$, pavyzdinė vertinimo programa išveda:
  - 1-oji eilutė: 1
  - 2-oji eilutė: k
  - 3-ioji eilutė: $c[0]$ $c[1]$ $\ldots$ $c[k - 1]$