Thousands Islands

Thousands Islands este un grup de insule deosebit de frumoase situat în Marea Java.

Acesta se compune din N insule, numerotate de la 0 la $N-1$.

Există M canoe, numerotate de la 0 la $M-1$, care pot fi folosite pentru a naviga între insule. Pentru fiecare i astfel încât $0 \leq i \leq M-1$, canoea i poate fi ancorată fie la insula $U[i]$, fie la insula $V[i]$, și poate fi folosită pentru a naviga între insulele $U[i]$ și $V[i]$. Mai exact, atunci când canoea este ancorată la insula $U[i]$, aceasta poate fi folosită pentru a naviga de la insula $U[i]$ către insula $V[i]$, dupa care canoea devine ancorată la insula $V[i]$. Similar, atunci cand canoea este ancorata la insula $V[i]$, aceasta poate fi folosită pentru a naviga de la insula $V[i]$ către insula $U[i]$, dupa care canoea devine ancorată la insula $U[i]$. Inițial, canoea este ancorată la insula $U[i]$. Este posibil ca mai multe canoe să poată fi utilizate pentru a naviga între aceeași pereche de insule. Este, de asemenea, posibil ca mai multe canoe să fie ancorate la aceeași insulă.

Din motive de siguranță, unei canoe trebuie să i se realizeze mentenanță de fiecare dată când se navighează cu aceasta, lucru care împiedică astfel folosirea aceleiași canoe de două ori la rând. Adică, după folosirea unei canoe i, o canoe alta decât i trebuie folosită înainte de a putea folosi canoea i din nou.

Bu Dengklek vrea să își planifice o călătorie prin unele insule. Călătoria ei este **validă** dacă și numai dacă următoarele condiții sunt satisfăcute.

- Ea începe și își termină călătoria în insula 0.
- Ea vizitează cel puțin o insulă, alta decât insula 0.
- După ce călătoria ia sfârșit, toate canoele sunt ancorate la aceeași insulă ca înainte de începerea călătoriei. Adică, canoea i, pentru fiecare i astfel încat $0 \leq i \leq M-1$ trebuie să fie ancorată la insula $U[i]$.

Ajutați-o pe Bu Dengklek să găsească orice călătorie validă, navigând de cel mult $2\,000\,000$ de ori, sau determinați faptul că o astfel de călătorie nu există. Se poate demonstra că, presupunând restricțiile specificate în enunț (vezi secțiunea Restricții), dacă o călătorie validă există, atunci există o astfel de călătorie care nu presupune mai mult de $2\,000\,000$ de navigări.

Detalii de implementare

Veți implementa urmatoarea procedură:

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

- N: numărul de insule.
- M: numărul de canoe.
- U, V: tablouri unidimensionale de lungime M, descriind canoele.
- Procedura returnează fie o valoare booleană, fie un tablou unidimensional de tip întreg.
    - Dacă nu există nicio călătorie validă, procedura returnează `false`.
    - Altfel, dacă există o călătorie validă, aveți două opțiuni:
        - Pentru a obține punctajul integral, procedura trebuie să returneze un tablou unidimensional de întregi de lungime cel mult $2\,000\,000$, reprezentând o călătorie validă. Mai exact, elementele tabloului trebuie să reprezinte indicii canoelor folosite într-o călătorie validă (în ordinea folosirii acestora).
        - Pentru a obține un punctaj parțial, procedura trebuie să returneze `true`, un tablou unidimensional de lungime mai mare ca $2\,000\,000$, sau un tablou unidimensional care nu descrie o călătorie validă (vezi secțiunea Subtask-uri pentru mai multe detalii).
- Această procedură se va apela exact o singură dată.

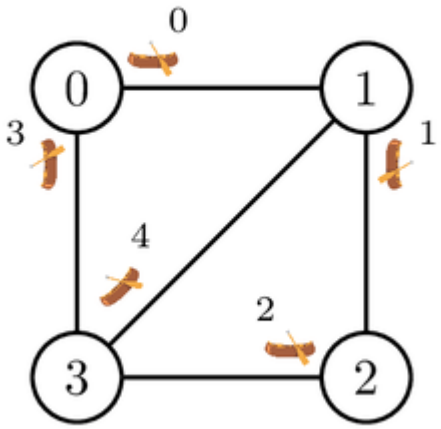
Exemple

Exemplul 1

Considerăm următorul apel:

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

Insulele și canoele sunt ilustrate mai jos.



O călătorie validă posibilă este următoarea. Mai întâi Bu Dengklek navighează folosind canoele 0, 1, 2, și 4, în ordinea dată, ajungând astfel pe insula 1. Apoi, Bu Dengklek poate naviga folosind

canoea 0 din nou, deoarece aceasta este ancorată la insula 1 și ultima canoe folosită de ea nu este canoea 0. După ce navighează folosind canoea 0 din nou, Bu Dengklek ajunge pe insula 0. Totuși, canoele 1, 2 și 4 nu sunt ancorate la insulele unde erau ancorate înainte de călătorie. Bu Dengklek își continuă călătoria, folosind canoele 3, 2, 1, 4, și 3 din nou. Bu Dengklek este iarăși pe insula 0 și toate canoele sunt ancorate la aceleași insule la care erau ancorate înainte de începerea călătoriei.

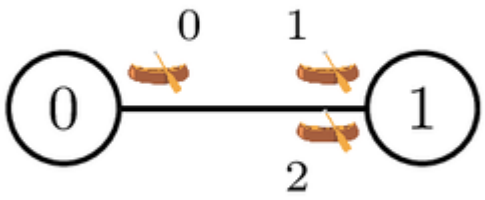
Așadar, valoarea returnată $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ reprezintă o călătorie validă.

Exemplul 2

Considerăm următorul apel:

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

Insulele și canoele sunt ilustrate mai jos.



Bu Dengklek poate începe doar alegând să navigheze cu canoea 0, dupa care ea poate naviga folosind fie canoea 1, fie canoea 2. De notat faptul că ea nu poate naviga folosind canoea 0 de două ori la rând. În ambele cazuri, Bu Dengklek ajunge înapoi pe insula 0. Totuși, canoele nu sunt ancorate la aceleași insule ca înainte de efectuarea călătoriei, iar Bu Dengklek nu mai poate naviga cu nicio canoe în acest moment, deoarece singura canoe ancorată la insula 0 a fost tocmai folosită. Deoarece nu există nicio calatorià validă, procedura returneaza `false`.

Restricții

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N - 1$ si $0 \leq V[i] \leq N - 1$ (pentru fiecare i astfel încât $0 \leq i \leq M - 1$)
- $U[i] \neq V[i]$ (pentru fiecare i astfel încât $0 \leq i \leq M - 1$)

Subtask-uri

1. (5 puncte) $N = 2$

2. (5 puncte) $N \leq 400$. Pentru fiecare pereche de insule distincte x și y ($0 \leq x < y \leq N-1$), există exact două canoe care pot fi utilizate pentru a naviga între ele. Una dintre canoe este ancorată la insula x, iar cealaltă este ancorată la insula y.
3. (21 puncte) $N \leq 1000$, M este par, și pentru fiecare număr **par** i astfel încât $0 \leq i \leq M-1$, canoele i si $i+1$ pot fi folosite ambele pentru a naviga între insulele $U[i]$ și $V[i]$. Canoea i este inițial ancorată la insula $U[i]$, iar canoea $i+1$ este inițial ancorată în insula $V[i]$. Formal, $U[i] = V[i+1]$ si $V[i] = U[i+1]$.
4. (24 puncte) $N \leq 1000$, M este par, și pentru fiecare număr **par** i astfel încât $0 \leq i \leq M-1$, canoele i și $i+1$ pot fi ambele folosite pentru a naviga între insulele $U[i]$ și $V[i]$. Ambele canoe sunt initial ancorate în insula $U[i]$. Formal, $U[i] = U[i+1]$ și $V[i] = V[i+1]$.
5. (45 puncte) Fără alte restricții.

Pentu fiecare test în care o călătorie validă există, soluției voastre i se acorda:

- punctaj maxim dacă returnează o calatorie validă,
- 35% din punctaj daca returnează `true`, un tablou unidimensional de întregi de lungime mai mare ca $2\ 000\ 000$, sau un tablou unidimensional care nu descrie o călătorie validă,
- 0 puncte altfel.

Pentru fiecare test în care nu există o călătorie validă, soluției voastre i se acordă:

- punctaj maxim dacă returnează `false`,
- 0 puncte altfel.

De notat faptul că punctajul acordat pentru un subtask este cel mai mic dintre numerele de puncte acordate fiecărui test din subtask.

Grader-ul local

Grader-ul local citește intrarea în următorul format:

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

Grader-ul local afisează răspunsurile voastre în următorul format:

- Daca `find_journey` returnează un `bool`:
  - linia 1: 0
  - linia 2: 0 dacă `find_journey` returnează `false`, si 1 altfel.
- Dacă `find_journey` returnează un `int[]`, să notăm elementele acestui tablou unidimensional cu $c[0], c[1], \ldots c[k-1]$. Atunci, grader-ul local afisează:
  - linia 1: 1
  - linia 2: k
  - linia 3: $c[0]$ $c[1]$ $\ldots$ $c[k-1]$