Тисячі островів

Тисяча островів - це група красивих островів, розташованих у морі Java. Вона складається з N островів, пронумерованих від 0 до $N-1$.

Є M каное з номерами від 0 до $M-1$, якими можна плисти між островами. Для кожного i такого, що $0 \leq i \leq M-1$, каное i може пришвартуватися на острові $U[i]$ або $V[i]$, і використовувати для плавання між островами $U[i]$ та $V[i]$. Зокрема, коли каное пришвартовано біля острова $U[i]$, його можна використати для плавання з острова $U[i]$ на острів $V[i]$, після чого каное пришвартується до острова $V[i]$. Подібним чином, коли каное пришвартовано біля острова $V[i]$, його можна використати для плавання з острова $V[i]$ на острів $U[i]$, після чого каное пришвартується до острова $U[i]$. Спочатку каное пришвартовано на острові $U[i]$. Цілком можливо, що для плавання між однією парою островів можна використовувати кілька каное. Також можливо, що на одному острові пришвартовано кілька каное.

З міркувань безпеки, каное потрібно обслуговувати після кожного плавання, що забороняє плисти на одному каное два рази поспіль. Тобто після використання одного каное i, потрібно використати інше каное, перш ніж каное i можна буде використовувати знову.

Бу Денгклек хоче спланувати подорож через деякі з островів. Його подорож є **валідною** тоді і тільки тоді, коли виконуються наступні умови.

- Він починає і закінчує свою подорож на острові 0.
- Він відвідує принаймні один острів, не враховуючи острів 0.
- Після закінчення подорожі кожне каное пришвартовується на тому самому острові, де було до подорожі. Тобто каное i, для кожного i такого, що $0 \leq i \leq M-1$, має бути пришвартовано до острова $U[i]$.

Допоможіть Бу Денгклеку знайти будь-яку валідну подорож, яка передбачає плавання щонайбільше $2\,000\,000$ разів, або визначте, що такої валідної подорожі не існує. Можна довести, що згідно з обмеженнями, визначеними в цьому завданні (див. розділ «Обмеження»), якщо існує валідна подорож, існує також валідна подорож, яка не передбачає плавання більше ніж $2\,000\,000$ разів.

Деталі реалізації

Ви повинні реалізувати таку функцію:

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

- N: кількість островів.
- M: кількість каное.
- U, V: масиви, довжини M, що описують каное.
- Ця процедура має повертати або логічне значення (boolean), або масив цілих чисел.
  - Якщо валідної подорожі не існує, процедура має повернути `false`.
  - Якщо валідна подорож існує, у вас є два варіанти:
    - Щоб отримати повний бал, процедура має повернути масив із щонайбільше $2\,000\,000$ цілих чисел, що представляють валідну подорож. Точніше, елементами цього масиву повинні бути номери каное, які використовуються в подорожі (у порядку їх використання).
    - Щоб отримати частковий бал, процедура має повернути `true`, масив із понад $2\,000\,000$ цілих чисел або масив цілих чисел, які не описують валідну подорож. (Додаткову інформацію див. у розділі Підзадачі).
- Ця процедура викликається рівно один раз.

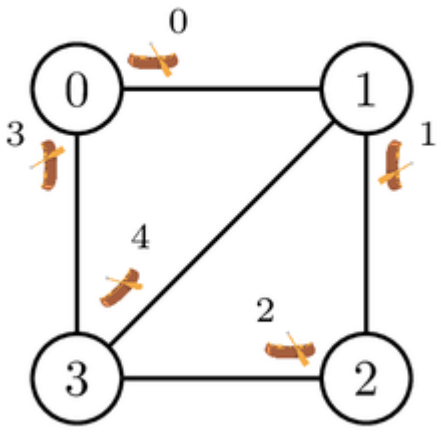
Приклади

Приклад 1

Розглянемо такий виклик:

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

Острови та каное показані на малюнку нижче.



Одна з можливих валідних подорожей виглядає наступним чином. Бу Денгклек спочатку пливе на каное 0, 1, 2, і 4 відповідно. У результаті він опинився на острові 1. Після цього Бу Денгклек знову зможе плисти на каное 0, оскільки воно зараз пришвартовано на острові 1, і останнє каное,

яке він використовував, не є каное 0. Знову попливши на каное 0, Бу Денгклек тепер на острові 0. Однак каное 1, 2 і 4 не пришвартовані на тих самих островах, де вони були до подорожі. Бу Денгклек продовжує свою подорож на каное 3, 2, 1, 4, і 3. Бу Денгклек повернувся на острів 0 і усі каное пришвартовані на тих самих островах, що й до подорожі.

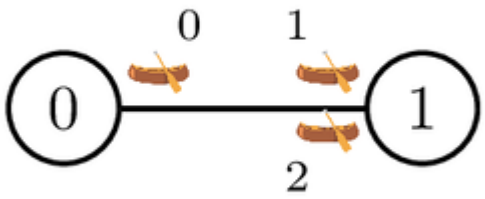
Таким чином, $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ описує валідну подорож.

Приклад 2

Розглянемо такий виклик:

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

Острови та каное показані на малюнку нижче.



Бу Денгклек може почати плисти на каное 0, після чого він може плисти на каное 1 або 2. Зверніть увагу, що він не може плисти на каное 0 двічі поспіль. В обох випадках Бу Денгклек повертається на острів 0. Однак каное не пришвартовано на тих самих островах, де вони були до подорожі, і Бу Денгклек не зможе плисти на жодному каное після цього, оскільки єдине каное, пришвартоване на острові 0, це те, яким він щойно скористався. Оскільки валідної подорожі немає, процедура має повернути `false`.

Обмеження

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N - 1$ і $0 \leq V[i] \leq N - 1$ (для кожного i такого, що $0 \leq i \leq M - 1$)
- $U[i] \neq V[i]$ (для кожного i такого, що $0 \leq i \leq M - 1$)

Підзадачі

1. (5 балів) $N = 2$
2. (5 балів) $N \leq 400$. Для кожної пари різних островів x та y ($0 \leq x < y \leq N - 1$), існує рівно два каное, які можна використовувати для плавання між ними. Один з них

пришвартований на острові x, а інший - на острові y.
3. (21 бал) $N \leq 1000$, M парне, і для кожного **парного** i такого, що $0 \leq i \leq M-1$, каное i та $i+1$ можна використовувати для плавання між островами $U[i]$ та $V[i]$. Каное i спочатку пришвартовано на острові $U[i]$, а каное $i+1$ спочатку пришвартовано на острові $V[i]$. Формально, $U[i] = V[i+1]$ і $V[i] = U[i+1]$.
4. (24 бали) $N \leq 1000$, M парне, і для кожного **парного** i такого, що $0 \leq i \leq M-1$, каное i та $i+1$ можна використовувати для плавання між островами $U[i]$ і $V[i]$. Обидва каное спочатку пришвартовані на острові $U[i]$. Формально, $U[i] = U[i+1]$ і $V[i] = V[i+1]$.
5. (45 балів) Без додаткових обмежень.

Для кожного тесту, в якому існує валідна подорож, ваше рішення:

- отримує повний бал, якщо повертає валідну подорож,
- отримує 35% балів, якщо повертає `true`, масив із понад $2\ 000\ 000$ цілих чисел або масив, який не описує валідну подорож
- інакше отримує 0 балів.

Для кожного тесту, у якому не існує валідної подорожі, ваше рішення:

- отримує повну кількість балів, якщо повертає `false`,
- інакше отримує 0 балів.

Зауважте, що кінцева оцінка для кожної підзадачі є мінімальною кількістю балів за усі тести в підзадачі.

Приклад градера

Градер зчитує вхідні дані в такому форматі:

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

Градер виводить ваші відповіді в такому форматі:

- Якщо `find_journey` повертає `bool`:
  - 1-й рядок: 0
  - 2-й рядок: 0 якщо `find_journey` повертає `false`, або 1 в іншому випадку.
- Якщо `find_journey` повертає `int[]`, позначте елементи цього масиву $c[0], c[1], \ldots c[k-1]$, відповідно. Градер виводить:
  - 1-й рядок: 1
  - 2-й рядок: k
  - 3-й рядок: $c[0]\ c[1]\ \ldots\ c[k-1]$