Tausend Inseln

Die *Tausend Inseln* sind eine Inselgruppe in der Javasee, bestehend aus N Inseln, nummeriert von 0 bis $N-1$.

Um zwischen den Inseln zu rudern stehen M Kanus bereit, nummeriert von 0 bis $M-1$.

Für alle i mit $0 \leq i \leq M-1$ kann Kanu i entweder bei Insel $U[i]$ oder $V[i]$ andocken und es kann benutzt werden um zwischen Inseln $U[i]$ und $V[i]$ zu rudern. Konkret, wenn das Kanu bei Insel $U[i]$ angedockt ist, kann es benutzt werden, um von Insel $U[i]$ nach Insel $V[i]$ zu rudern, wonach es bei Insel $V[i]$ angedockt ist. Umgekehrt, wenn das Kanu bei Insel $V[i]$ angedockt ist, kann es benutzt werden, um von Insel $V[i]$ nach Insel $U[i]$ zu rudern, wonach es bei Insel $U[i]$ angedockt ist. Anfangs ist das Kanu bei Insel $U[i]$ angedockt. Es ist möglich, dass es mehrere Kanus gibt, die zwischen dem gleichen Paar von Inseln benutzt werden können. Es ist ebenfalls möglich, dass mehrere Kanus an der selben Insel angedockt sind.

Aus Sicherheitsgründen muss ein Kanu jedesmal gewartet werden, nachdem es benutzt wurde, was es unmöglich macht, ein Kanu zwei Mal hintereinander zu benutzen. Das heißt, nachdem Kanu i benutzt wurde, muss zuerst ein anderes Kanu benutzt werden, bevor wieder Kanu i benutzt werden kann.

Bu Dengklek möchte ihre Reise durch einige der Inseln planen. Die Reise ist **gültig** genau dann, wenn die folgenden Bedingungen erfüllt sind:

- Sie startet und endet ihre Reise bei Insel 0.
- Sie besucht mindestens eine andere Insel abgesehen von Insel 0.
- Am Ende der Reise ist jedes Kanu wieder bei der gleichen Insel angedockt, bei der es anfangs angedockt war. D.h., Kanu i, für jedes i mit $0 \leq i \leq M-1$, muss an Insel $U[i]$ angedockt sein.

Hilf Bu Dengklek, eine gültige Reise zu finden, in welcher sie höchstens $2\,000\,000$ mal rudern muss, oder stelle fest, dass es keine solche Reise gibt. Es lässt sich beweisen, dass sofern unter den gegebenen Bedingungen (siehe Abschnitt Beschränkungen) eine gültige Reise existiert, auch eine gültige Reise existiert, in welcher nicht mehr als $2\,000\,000$ Mal gerudert wird.

Implementierungsdetails

Implementiere folgende Funktion:

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

- N: Die Anzahl an Inseln.
- M: Die Anzahl an Kanus.
- U, V: Arrays der Längen M, welche die Kanus beschreiben.
- Die Funktion soll entweder einen Bool oder ein Array von ganzen Zahlen zurückgeben.
    - Falls keine gültige Reise existiert, gib `false` zurück.
    - Falls eine gültige Reise existiert, dann hast du zwei Optionen:
        - Für volle Punktzahl sollte deine Funktion ein Array mit höchstens $2\,000\,000$ ganzen Zahlen zurückgeben, welches eine gültige Reise beschreibt. Genauer: Die Elemente dieses Arrays sollen die Nummern der Kanus sein, welche in der Reise benutzt werden (in der entsprechenden Reihenfolge).
        - Für Teilpunkte kann deine Funktion auch `true`, ein Array mit mehr als $2\,000\,000$ ganzen Zahlen, oder ein Array, das keine gültige Reise beschreibt, zurückgeben. (Siehe Abschnitt Teilaufgaben für Details.)
- Diese Funktion wird genau einmal aufgerufen.

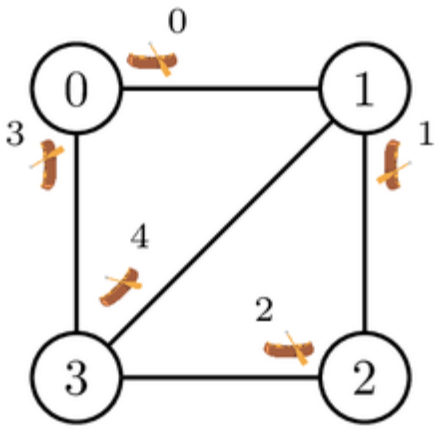
Beispiele

Beispiel 1

Betrachten wir folgenden Funktionsaufruf:

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

Die Inseln und Kanus werden im Bild unten dargestellt.



Eine mögliche Reise ist wie folgt. Bu Dengklek benutzt zuerst Kanus 0, 1, 2 und 4 in dieser Reihenfolge und ist dann auf Insel 1. Danach benutzt Bu Dengklek Kanu 0 ein zweites Mal, da dieses momentan bei Insel 1 angedockt ist und nicht zuletzt Kanu 0 benutzt wurde. Nachdem Bu

Dengklek Kanu 0 wieder benutzt hat, ist sie bei Insel 0. Jedoch sind Kanus 1, 2 und 4 nicht an der selben Insel angedockt wie vor der Reise. Daraufhin setzt Bu Dengklek ihre Reise fort indem sie Kanus 3, 2, 1, 4, und nochmals 3 benutzt. Nun ist Bu Dengklek zurück auf Insel 0 und alle Kanus sind an der selben Insel wie vor Beginn der Reise angedockt.

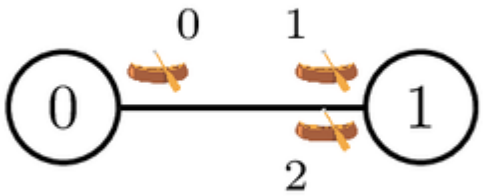
Daher repräsentiert die Rückgabe $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ eine gültige Reise.

Beispiel 2

Betrachten wir den folgenden Funktionsaufruf:

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

Die Inseln und Kanus werden im Bild unten dargestellt.



Bu Dengklek muss als erstes Kanu 0 benutzen. Danach kann sie entweder Kanu 1 oder 2 benutzen, da sie Kanu 0 nicht zweimal hintereinander benutzen kann.

In beiden Fällen ist Bu Dengklek zurück auf Insel 0, jedoch sind die Kanus nicht an den selben Inseln wie vor der Reise angedockt. Außerdem kann Bu Dengklek kein weiteres Kanu benutzen, da sie das einzige Kanu, welches bei Insel 0 angelegt ist, gerade benutzt hat. Deshalb existiert keine gültige Reise und die Funktion sollte `false` zurückgeben.

Beschränkungen

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N - 1$ und $0 \leq V[i] \leq N - 1$ (für alle i mit $0 \leq i \leq M - 1$)
- $U[i] \neq V[i]$ (für alle i mit $0 \leq i \leq M - 1$)

Teilaufgaben

1. (5 Punkte) $N = 2$

2. (5 Punkte) $N \leq 400$. Für jedes Paar von Inseln x und y ($0 \leq x < y \leq N - 1$) gibt es genau zwei Kanus, welche benutzt werden können, um zwischen ihnen hin und her zu rudern. Eines davon ist bei Insel x angedockt und das andere bei Insel y.
3. (21 Punkte) $N \leq 1000$, M ist gerade und für alle **geraden** i mit $0 \leq i \leq M - 1$ können Kanus i und $i + 1$ beide benutzt werden, um zwischen Inseln $U[i]$ und $V[i]$ hin und her zu rudern. Kanu i ist anfangs bei Insel $U[i]$ angedockt und Kanu $i + 1$ ist anfangs bei Insel $V[i]$ angedockt. Formal ausgedrückt: $U[i] = V[i + 1]$ und $V[i] = U[i + 1]$.
4. (24 Punkte) $N \leq 1000$, M ist gerade und für alle **geraden** i mit $0 \leq i \leq M - 1$ und Kanus i und $i + 1$ können beide benutzt werden, um zwischen Inseln $U[i]$ und $V[i]$ hin und her zu rudern. Beide Kanus sind anfangs bei Insel $U[i]$ angelegt. Formal ausgedrückt: $U[i] = U[i + 1]$ und $V[i] = V[i + 1]$.
5. (45 Punkte) Keine weiteren Beschränkungen.

Für jeden Testfall, bei dem eine gültige Reise existiert, erhält deine Lösung:

- Volle Punktzahl, falls sie eine gültige Reise zurückgibt.
- 35% der Punkte, falls sie `true` zurückgibt oder ein Array mit mehr als $2\,000\,000$ ganzen Zahlen, oder ein Array welches keine gültige Reise beschreibt,
- 0 Punkte, andernfalls.

Für jeden Testfall, bei dem keine gültige Reise existiert, erhält deine Lösung:

- Volle Punktzahl, falls sie `false` zurückgibt.
- 0 Punkte, andernfalls.

Die Punktzahl jeder Teilaufgabe ist die minimale Anzahl an Punkten aller Testfälle dieser Teilaufgabe.

Sample-Grader

Der Sample-Grader liest die Eingabe in folgendem Format:

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

Der Sample-Grader gibt deine Antworten in folgendem Format aus:

- Falls `find_journey` einen `bool` zurückgibt:
  - Zeile 1: 0
  - Zeile 2: 0, falls `find_journey` `false` zurückgibt, und ansonsten 1.
- Falls `find_journey` einen `int[]` zurückgibt, dessen Elemente $c[0], c[1], \ldots c[k - 1]$ sind, gibt der Sample-Grader Folgendes aus:
  - Zeile 1: 1
  - Zeile 2: k
  - Zeile 3: $c[0]$ $c[1]$ $\ldots$ $c[k - 1]$