Tysiące wysp

Tysiące wysp, to archipelag pięknych wysp na Morzu Jawajskim. Składa się z N wysp ponumerowanych od 0 do $N-1$.

Między wyspami można przeprawiać się za pomocą M łódek, ponumerowanych od 0 do $M-1$. Dla każdego i takiego że $0 \leq i \leq M-1$, łódka i może kotwiczyć na wyspie $U[i]$ lub $V[i]$ i można jej używać do kursowania tylko między wyspami $U[i]$ oraz $V[i]$. Dokładniej, jeśli łódka jest zakotwiczona na wyspie $U[i]$, można się nią przeprawić z wyspy $U[i]$ na wyspę $V[i]$, po czym łódka pozostaje zakotwiczona na wyspie $V[i]$. Podobnie, jeśli łódka jest zakotwiczona na wyspie $V[i]$, można się nią przeprawić z wyspy $V[i]$ na wyspę $U[i]$, po czym łódka pozostaje zakotwiczona na wyspie $U[i]$. Początkowo każda łódka jest zakotwiczona na wyspie $U[i]$. Między dwoma wyspami może kursować więcej niż jedna łódka. Możliwe jest też kotwiczenie wielu łódek na tej samej wyspie.

Ze względów bezpieczeństwa, każda łódka musi przejść przegląd po każdej przeprawie i nie można jej użyć dwa razy pod rząd. Inaczej mówiąc, po przeprawie za pomocą łódki i, do kolejnej przeprawy trzeba użyć innej łódki niż i.

Bu Dengklek chciałaby zaplanować podróż po niektórych wyspach. Jej podróż będzie **poprawna** wtedy i tylko wtedy gdy będą spełnione następujące warunki:

- Bu zaczyna i kończy podróż na wyspie 0.
- Odwiedza przynajmniej jedną wyspę różną od 0.
- Po zakończeniu podróży każda łódka jest zakotwiczona na tej samej wyspie, co na początku podróży. Inaczej mówiąc, dla każdego i, takiego że $0 \leq i \leq M-1$, łódka i musi być zakotwiczona na wyspie $U[i]$.

Pomóż Bu Dengklek znaleźć jakąkolwiek poprawną podróż wymagającą nie więcej niż $2\,000\,000$ przepraw lub stwierdź, że taka poprawna podróż nie istnieje. Można dowieść, że przy ograniczeniach dla tego zadania (patrz rozdział Ograniczenia), jeśli poprawna podróż istnieje, to nie wymaga więcej niż $2\,000\,000$ przepraw.

Szczegóły implementacyjne

Powinieneś zaimplementować następującą funkcję:

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

- N: liczba wysp.
- M: liczba łódek.
- U, V: tablice długości M definiujące łódki.
- Wynikiem tej funkcji powinna być wartość logiczna lub tablica liczb całkowitych.
    - Jeśli nie istnieje poprawna podróż, to wynikiem funkcji powinno być `false`.
    - Jeśli poprawna podróż istnieje, to masz dwie możliwości:
        - Aby uzyskać maksymalny wynik, wynikiem funkcji powinna być tablica długości nieprzekraczającej $2\,000\,000$ złożona z liczb całkowitych definiujących poprawną podróż. Dokładniej, elementy tej tablicy powinny być kolejnymi numerami łódek, które zostały użyte do podróży (w kolejności ich użycia).
        - Aby uzyskać częściowy wynik, wynikiem funkcji powinno być `true` lub tablica zawierająca więcej niż $2\,000\,000$ liczb całkowitych lub tablica nieopisująca poprawnej podróży. Patrz rodział Podzadania.
- Ta funkcja jest wywoływana tylko raz.

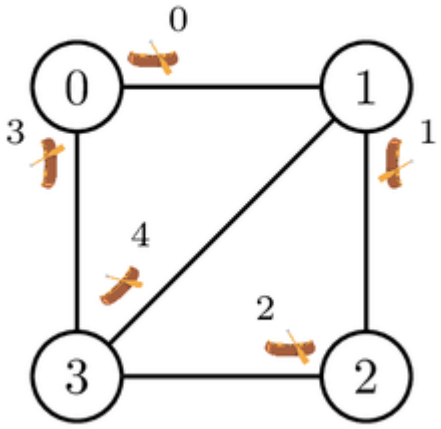
Przykłady

Przykład 1

Rozważ następujące wywołanie:

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

Wyspy i łódki są przedstawione na poniższym rysunku.



Jedna z możliwych poprawnych podróży jest następująca: najpierw Bu Dengklek kolejno płynie łódkami 0, 1, 2, i 4. W rezultacie znajduje się na wyspie 1. Następnie Bu Dengklek może użyć łódki 0 powtórnie, gdyż jest ona zakotwiczona na wyspie 1 i łódka 0 nie jest łódką z ostatniej przeprawy. Po powtórnej przeprawie łódką 0 Bu Dengklek jest teraz na wyspie 0. Niestety łódki 1, 2 i 4 nie są zakotwiczone na tych samych wyspach, co na początku, więc Bu Dengklek kontynuuje podróż

przeprawiając się kolejno łódkami 3, 2, 1, 4 i powtórnie 3. Bu Dengklek jest teraz z powrotem na wyspie 0, a wszystkie łódki są na początkowych pozycjach.

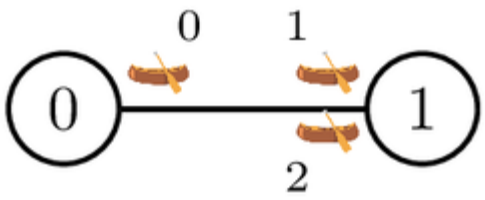
Zatem $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ jest poprawnym wynikiem wywołania funkcji.

Przykład 2

Rozważ następujące wywołanie

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

Wyspy i łódki są pokazane na poniższym rysunku:



Bu Dengklek może zacząć tylko za pomocą łódki 0, po czym może wrócić albo łódką 1 albo 2. Zwróć uwagę, że nie może powtórnie użyć łódki 0. W obu przypadkach Bu Dengklek jest z powrotem na wyspie 0. Niestety łódki nie znajdują się teraz na wyjściowych wyspach i Bu Dengklek nie może dalej wykonać żadnej przeprawy, bo właśnie przypłynęła jedyną dostępną łódką. Ze względu na to, że nie istnieje żadna poprawna podróż, wynikiem funkcji powinno być `false`.

Ograniczenia

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N - 1$ oraz $0 \leq V[i] \leq N - 1$ (dla każdego i takiego, że $0 \leq i \leq M - 1$)
- $U[i] \neq V[i]$ (dla każdego i takiego, że $0 \leq i \leq M - 1$)

Podzadania

1. (5 punktów) $N = 2$
2. (5 punktów) $N \leq 400$. Dla każdej pary różnych wysp x oraz y ($0 \leq x < y \leq N - 1$), są dokładnie dwie łódki kursujące między nimi. Jedna z nich jest zakotwiczona na wyspie x, druga na wyspie y.
3. (21 punktów) $N \leq 1000$, M jest parzyste, dla każdego **parzystego** i takiego, że $0 \leq i \leq M - 1$, łódki i oraz $i + 1$ pływają między wyspami $U[i]$ oaz $V[i]$. Łódka i jest

początkowo zakotwiczona na wyspie $U[i]$, a łódka $i+1$ jest zakotwiczona początkowo na wyspie $V[i]$. Formalnie: $U[i] = V[i+1]$ oraz $V[i] = U[i+1]$.
4. (24 punkty) $N \leq 1000$, M jest parzyste i dla każdego **parzystego** i takiego, że $0 \leq i \leq M-1$, łódki i oraz $i+1$ mogą być używane do przepraw między wyspami $U[i]$ oraz $V[i]$. Obie łódki są początkowo zaotwiczone na wsypie $U[i]$. Formalnie: $U[i] = U[i+1]$ i $V[i] = V[i+1]$.
5. (45 punktów) Brak dodatkowych ograniczeń.

Dla każdego z przypadków testowych, dla których istnieje poprawna podróż, Twoje rozwiązanie

- otrzymuje komplet punktów, jeśli wynikiem jest poprawna podróż
- otrzymuje 35% punktów, jeśli wynikiem jest `true`, tablica mająca więcej niż $2\ 000\ 000$ liczb całkowitych lub tablica nie opisująca poprawnej podróży.
- otrzymuje 0 punktów w przeciwnym razie.

Dla każdego z przypadków testowych, dla których nie istnieje poprawna podróż, Twoje rozwiązanie

- otrzymuje komplet punktów, jeśli wynikiem jest `false',
- otrzymuje 0 punktów w przeciwnym razie.

Musisz też wiedzieć, że wynik każdego z podzadań, to minimum z punktów spośród wszystkich przypadków testowych dla tego podzadania.

Przykładowy program oceniający

Przykładowy program oceniający czyta dane z wejścia w następującym formacie:

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

Przykładowy program oceniający wypisuje Twoje odpowiedzi w następującym formacie:

- Jeśli wynikiem `find_journey` jest `bool`:
  - wiersz 1: 0
  - wiersz 2: 0 jeśli wynikiem `find_journey` jest `false`, lub 1 w przeciwnym razie.
- Jeśli `find_journey` przekazuje `int[]`; niech elementy tablicy to odpowiednio $c[0], c[1], \ldots c[k-1]$. Przykładowy program oceniający wypisuje:
  - wiersz 1: 1
  - wiersz 2: k
  - wiersz 3: $c[0]\ c[1]\ \ldots\ c[k-1]$