Ostrovy

V Jávském moři je N ostrovů očíslovaných od 0 do $N-1$.

Pro plavbu mezi ostrovy se používá M kánoí očíslovaných od 0 do $M-1$. Kánoe i (kde $0 \leq i \leq M-1$) se používá pro plavbu mezi ostrovy $U[i]$ a $V[i]$. Může být zakotvena buď u ostrova $U[i]$, nebo u ostrova $V[i]$. Když je kánoe zrovna zakotvena u ostrova $U[i]$, můžeme se s ní přeplavit k ostrovu $V[i]$, kde tato kánoe následně bude zakotvena. Podobně když je kánoe zrovna zakotvena u ostrova $V[i]$, můžeme se s ní přeplavit k ostrovu $U[i]$, kde tato kánoe následně bude zakotvena. Kánoe je na začátku zakotvena u ostrova $U[i]$. K plavbě mezi jednou dvojicí ostrovů může být používáno více různých kánoí. U jednoho ostrova může najednou kotvit několik kánoí.

Z bezpečnostích důvodů se po každé plavbě musí provést údržba kánoe, takže nikdy není možné použít stejnou kánoi dvakrát po sobě. Jestliže tedy kánoe i právě byla použita k plavbě, musí být použita nějaká jiná kánoe $j \neq i$, než se znovu použije kánoe i.

Bu Dengklek by chtěla naplánovat výpravu po ostrovech. Její výprava bude **úspěšná** právě tehdy, když splňuje všechny násldující podmínky.

- Výprava začíná a končí na ostrově 0.
- Výprava vede přes aspoň jeden ostrov různý od ostrova 0.
- Na konci výpravy jsou všchny kánoe zakotveny tam, kde byly na začátku. Tedy pro každé i splňující $0 \leq i \leq M-1$ je kánoe i zakotvena u ostrova $U[i]$.

Pomozte *Bu Dengklek* najít libovolnou úspěšnou výpravu, pří které provede nějvýše $2\,000\,000$ plaveb (přesunů mezi ostrovy za pomoci kánoe), nebo rozhodněte, že zádná úspěšná výprava neexistuje. Za daných omezení (specifikovaných sekci Omezení) je možné dokázat, že pokud úspěšná výprava existuje, potom také existuje úspěčná výprava obsahující nejvýše $2\,000\,000$ plaveb.

Implementační detaily

Implementujte následující funkci:

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

- N: počet ostrovů.
- M: počet kánoí.

- U, V: pole délky M popisující kánoe.
- Tato funkce by měla vrátit buď bool, nebo pole celých čísel.
    - Pokud neexistuje žádná úspěšná výprava, funkce by měla vrátit `false`.
    - Pokud existuje úspěšná výprava, máte dvě možnosti:
        - K získání plného počtu bodů by funkce měla vrátit pole celých čísel délky nejvýše $2\,000\,000$ reprezentující úspěšnou výpravu. Vrácené pole by mělo obsahovat čísla kánoí, které jsou na této výpravě použity (a to v pořadí, ve kterém jsou použity).
        - Částečné body můžete získat, pokud funkce vrátí `true`, pole o více než $2\,000\,000$ prvcích, nebo pole nepopisující úspěšnou výpravu (podrobnosti naleznete v sekci Podúlohy).
- Tato funkce bude zavolána právě jednou.

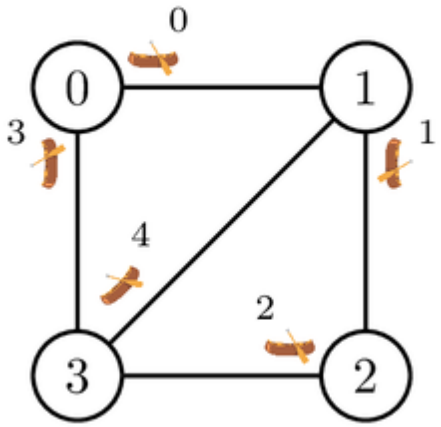
Příklady

Příklad 1

Uvažte následující volání:

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

Ostrovy a kánoe jsou zobrazeny na ilustraci níže.



Jedna možná úspěšná výprava vypadá následovně. *Bu Dengklek* nejdříve postupně použije kánoe 0, 1, 2, a 4, načež se nachází na ostrově 1. *Bu Dengklek* následně může znovu použít kánoi 0, protože je momentálně zakotvena u ostrova 1 a poslední použitá kánoe nebyla kánoe 0. Po druhém použití kánoe 0 se *Bu Dengklek* nachází zpátky na ostrově 0. Kánoe 1, 2 a 4 ovšem nejsou zakotveny u ostrovů, kde začínaly. *Bu Dengklek* pokračuje za pomoci kánoí 3, 2, 1, 4 a nakonec znovu 3. *Bu Dengklek* se nyní nachází na ostrově 0 a všechny kánoe jsou zakotveny u ostrovů, kde začínaly.

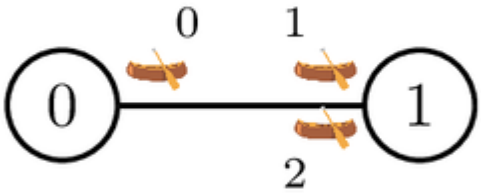
Jednou ze správných návratových hodnost je tedy pole $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$.

Příklad 2

Uvažujte následující volání:

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

Ostrovy a kánoe jsou zobrazeny na ilustraci níže.



Jediná kánoe, kterou může *Bu Dengklek* výpravu začít, je kánoe 0. Následně může použít kánoi 1 nebo 2. Všimněte si, že nemůže použít kánoi 0 dvakrát po sobě. V obou případech bude *Bu Dengklek* zpět na ostrově 0. Některé kánoe teď ale nejsou zakotveny u strovů, kde začínaly, a *Bu Dengklek* nyní nemůže použít žádnou kánoi. Jediná kánoe kotvící u ostrova 0 je ta, kterou právě použila. Protože neexistuje žádná úspěšná výprava, funkce by měla vrátit `false`.

Omezení

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N-1$ a $0 \leq V[i] \leq N-1$ (pro i splňující $0 \leq i \leq M-1$)
- $U[i] \neq V[i]$ (pro i splňující $0 \leq i \leq M-1$)

Podúlohy

1. (5 bodů) $N = 2$
2. (5 bodů) $N \leq 400$. Pro každou dvojici různých ostrovů x a y ($0 \leq x < y \leq N-1$) existují právě dvě kánoe, které se mezi nimi plaví. Jedna z nich je na začátku zakotvana u ostrova x, druhá je na začátku zakotvena u ostrova y.
3. (21 bodů) $N \leq 1000$, M je sudé, a pro každé **sudé** i splňující $0 \leq i \leq M-1$ se kánoe i a $i+1$ obě plaví mezi ostrovy $U[i]$ a $V[i]$. Kánoe i na začátku kotví u ostrova $U[i]$ a kánoe $i+1$ na začátku kotví u ostrova $V[i]$. Formálně, $U[i] = V[i+1]$ a $V[i] = U[i+1]$.
4. (21 bodů) $N \leq 1000$, M je sudé, a pro každé **sudé** i splňující $0 \leq i \leq M-1$ se kánoe i a $i+1$ obě plaví mezi ostrovy $U[i]$ a $V[i]$. Obě kánoe na začátku kotví u ostrova $U[i]$. Formálně, $U[i] = U[i+1]$ a $V[i] = V[i+1]$.
5. (45 bodů) Žádná další omezení.

Pro každý vstup, kde existuje úspěšná výprava, vaše řešení:

- získá plný počet bodů, pokud vrátí úspěšnou výpravu,
- získá 35% bodů, pokud vrátí `true`, pole o více než $2\ 000\ 000$ prvcích, nebo pole nepopisující úspěšnou výpravu,
- nezíská žádné body v ostatních případech.

Pro každý vstup, kde neexistuje úspěšná výprava, vaše řešení:

- získá plný počet bodů, pokud vrátí `false`,
- nezíská žádné body v ostatních případech.

Počet bodů získaných za podúlohu je minumum z počtů bodů získaných za všechny vstupy, které tato podúloha obsahuje.

Ukázkový grader

Ukázkový grader načítá vstup v následujícím formátu:

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

Ukázkový grader vypisuje odpověd v následujícím formátu:

- Pokud `find_journey` vrátí `bool`:
  - řádek 1: 0
  - řádek 2: 0 pokud `find_journey` vrátí `false`, nebo 1 ostatních případech.
- Pokud `find_journey` vrátí `int[]`, označme prvky tohoto pole postupně $c[0], c[1], \ldots c[k-1]$. Ukázkový grader vypíše:
  - řádek 1: 1
  - řádek 2: k
  - řádek 3: $c[0]\ c[1]\ \ldots\ c[k-1]$