# セリブ諸島

セリブ諸島は美しい島々からなるジャワ海の島嶼群である．この諸島を構成する島は $N$ 島あり，0 から $N-1$ までの番号が付けられている．

島と島との間を航行できる $M$ 艇のカヌーがあり，0 から $M-1$ までの番号が付けられている．$0 \leqq i \leqq M-1$ を満たすそれぞれの $i$ について，カヌー $i$ は島 $U[i]$ と島 $V[i]$ に停泊でき，島 $U[i]$ と島 $V[i]$ の間を航行できる． 具体的には，カヌー $i$ が島 $U[i]$ に停泊しているとき，このカヌーを使って島 $U[i]$ から島 $V[i]$ に航行でき，その後このカヌーは島 $V[i]$ に停泊する． 同様に，カヌー $i$ が島 $V[i]$ に停泊しているとき，このカヌーを使って島 $V[i]$ から島 $U[i]$ に航行でき，その後このカヌーは島 $U[i]$ に停泊する．はじめ，カヌー $i$ は島 $U[i]$ に停泊している． 複数のカヌーが同じ 2 つの島の間を航行していることもある．また，複数のカヌーを同じ島に停泊させることもできる．

安全上の理由で，カヌーは航行するたびにメンテナンスを受ける必要があるため，同じカヌーを 2 回連続で使うことはできない． すなわち，カヌー $i$ を使った後，もう 1 回カヌー $i$ を使う前に他のカヌーを使う必要がある．

Bu Dengklek は島々をいくつか巡る旅を計画することにした． 彼女の旅程が**適切**であるとは，その旅程が以下の条件を満たすことを言う．

- 島 0 で始まり，島 0 で終わる．
- 島 0 以外の島を少なくとも 1 つ訪れる．
- 旅程の終了時，すべてのカヌーは開始前と同じ島に停泊している．すなわち，$0 \leqq i \leqq M-1$ を満たすそれぞれの $i$ について，カヌー $i$ は島 $U[i]$ に停泊している．

Bu Dengklek のために，2 000 000 回以下の航行からなる適切な旅程を見つけるか，適切な旅程が無いことを判定せよ． この問題の制約のもとで (制約の項を参照せよ)，適切な旅程が存在するならば，2 000 000 回以下の航行からなる適切な旅程が存在するということが証明できる．

## 実装の詳細

あなたは，以下の関数を実装する必要がある．

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

- $N$: 島の個数
- $M$: カヌーの個数
- $U, V$: カヌーの情報を表す，長さ $M$ の配列

- この関数は戻り値として，真理値か整数配列を返さなければならない．
  - 適切な旅程が存在しない場合，`false` を返さなければならない．
  - 適切な旅程が存在する場合，2 つの選択肢がある．
    - 満点を得るためには，適切な旅程を表す長さ $2\,000\,000$ 以下の整数配列を返さなければならない．厳密にいうと，この配列の要素は，旅程で使われるカヌーの番号を使われる順に並べたものである必要がある．
    - 部分点を得るためには，`true`，長さ $2\,000\,000$ を超える整数配列，適切な旅程を表さない整数配列のいずれかを返さなければならない．詳細は小課題の項を参照せよ．
- この関数はちょうど 1 回呼び出される．

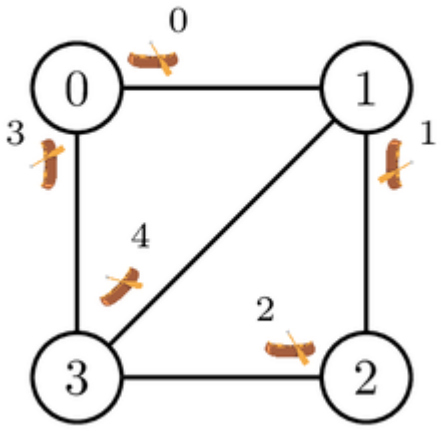
# 入出力例

## 入出力例 1

以下のような関数呼び出しを考える．

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

島とカヌーは以下の図で与えられる．




適切な旅程の一例は以下の通りである．

Bu Dengklek はまず，カヌー $0, 1, 2, 4$ をこの順に使う． この時点で，彼女は島 $1$ にいる． その後，Bu Dengklek はカヌー $0$ を再度使う．カヌー $0$ は島 $1$ に停泊しており，直前に使ったカヌーではないため，ここで使うことができる． この時点で，彼女は島 $0$ にいるが，カヌー $1, 2, 4$ が旅程の開始前と異なる島に停泊している． 彼女はその後，カヌー $3, 2, 1, 4$ をこの順に使い，最後にもう一度カヌー $3$ を使う． これで Bu Dengklek は島 $0$ に帰ってきており，すべてのカヌーは旅程の開始前と同じ島に停泊している．

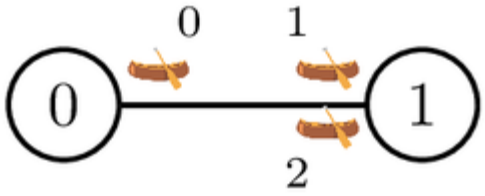
したがって，戻り値 $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ は適切な旅程を表す．

### 入出力例 2

以下のような関数呼び出しを考える．

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

島とカヌーは以下の図で与えられる．




Bu Dengklek が最初に使えるカヌーはカヌー $0$ のみであり，その直後に使えるカヌーはカヌー $1, 2$ のいずれかである． カヌー $0$ を $2$ 回連続で使うことはできないことに注意せよ． いずれの場合も，彼女は島 $0$ に戻る． しかし，カヌーは旅程の開始前と異なる島に停泊している．また，直前に彼女が使ったもの以外に島 $0$ に停泊しているカヌーが存在しないため，その後カヌーを使うことができなくなっている． したがって適切な旅程は存在しないので，この関数は `false` を返さなければならない．

## 制約

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

## 小課題

1. (5 点) $N = 2$
2. (5 点) $N \leqq 400$． どの $2$ つの島の組合せ $x, y$ $(0 \leqq x < y \leqq N-1)$ に対しても，ちょうど $2$ つのカヌーがその間で使える． 片方は島 $x$ に，もう片方は島 $y$ に停泊している．
3. (21 点) $N \leqq 1000$．$M$ は偶数．$0 \leqq i \leqq M-1$ を満たす**偶数** $i$ について，カヌー $i$ とカヌー $i+1$ はともに島 $U[i]$ と島 $V[i]$ の間を航行する. カヌー $i$ ははじめ島 $U[i]$ に，カヌー $i+1$ ははじめ島 $V[i]$ に停泊している． 厳密には，$U[i] = V[i+1]$ および $V[i] = U[i+1]$ が成り立つ.
4. (24 点) $N \leqq 1000$．$M$ は偶数．$0 \leqq i \leqq M-1$ を満たす**偶数** $i$ について，カヌー $i$ とカヌー $i+1$ はともに島 $U[i]$ と島 $V[i]$ の間を航行する． 両方のカヌーははじめ島 $U[i]$ に停泊している． 厳密には，$U[i] = U[i+1]$ および $V[i] = V[i+1]$ が成り立つ．
5. (45 点) 追加の制約はない．

適切な旅程が存在する各テストケースにおいて，

- 戻り値が適切な旅程の場合，満点を得る．
- 戻り値が `true`，長さ $2\,000\,000$ を超える整数配列，適切な旅程を表さない整数配列のいずれかである場合，$35\%$ の得点を得る．
- いずれでもない場合，$0$ 点を得る．

適切な旅程が存在しない各テストケースにおいて，

- 戻り値が `false` の場合，満点を得る．
- そうでない場合，$0$ 点を得る．

各小課題における最終的な得点は，その小課題に含まれるテストケースで得られた得点の最小値によって決まることに注意せよ．

## 採点プログラムのサンプル

採点プログラムのサンプルは次の形式で入力を読み込む．

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

採点プログラムのサンプルは以下の形式であなたの答えを出力する．

- `find_journey` が `bool` を返した場合:
  - $1$ 行目: $0$
  - $2$ 行目: `find_journey` が `false` を返した場合は $0$，そうでない場合は $1$
- `find_journey` が `int[]` を返した場合，その要素を順に $c[0], c[1], \ldots c[k-1]$ とおいて:
  - $1$ 行目: $1$
  - $2$ 行目: $k$
  - $3$ 行目: $c[0]\ c[1]\ \ldots\ c[k-1]$