



## De tusen öarna

De tusen öarna är en grupp vackra öar nära Javasjön. Ögruppen består av  $N$  öar, numrerade mellan 0 och  $N - 1$ .

Öarna har  $N$  kanoter, numrerade mellan 0 och  $M - 1$ , som kan användas för att färdas mellan öarna.

För varje  $j$ , där  $0 \leq j \leq M - 1$ , så kan kanot nummer  $j$  ankras dels på kanten av ön  $U[j]$ , och dels på kanten av  $V[j]$ . Kanoten kan användas för att färdas mellan  $U[j]$  och  $V[j]$ . När kanoten är ankrad nära ön  $U[j]$  kan den användas för att färdas från  $U[j]$  mot  $V[j]$ , och därefter kommer den vara ankrad nära ön  $V[j]$ . När kanoten är ankrad nära  $V[j]$  så kan den på samma sätt användas för att färdas från  $V[j]$  mot  $U[j]$ , och kommer därefter vara ankrad nära  $U[j]$ . Först så är kanoten ankrad nära  $U[j]$ . Det kan vara så att flera kanoter kan användas för att färdas mellan samma par av öar. Det kan också hända att flera kanoter ankras på samma ö.

Av säkerhetsskäl måste en kanot underhållas efter varje gång den använts. Därför går det ej att använda samma kanot två gånger på raken. Med andra ord, efter att kanot  $j$  använts måste en annan kanot användas före kanot  $j$  åter kan användas.

Bea Klämkäck planerar att färdas mellan några av öarna. Hennes resa är **bra** om och endast om följande krav är uppfyllda.

- Hon börjar och slutar resan på ön 0.
- Hon besöker mer än noll andra öar utom ön 0.
- Efter att resan slutar måste varje kanot vara ankrad på samma ö som före resan. Med andra ord ska kanot  $j$ , för alla  $j$  så att  $0 \leq j \leq M - 1$ , vara ankrad på ön  $U[j]$ .

Hjälp Bea Klämkäck att konstruera en bra resa som har som mest 2 000 000 kanotfärder, eller bestäm att det ej kan göras. Det garanteras att om en bra resa kan konstrueras, så kan en annan även konstrueras med som mest 2 000 000 kanotfärder.

## Implementation Details

You should implement the following procedure:

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

- $N$ : the number of islands.

- $M$ : the number of canoes.
- $U, V$ : arrays of length  $M$  describing the canoes.
- This procedure should return either a boolean or an array of integers.
  - If no valid journey exists, the procedure should return `false`.
  - If a valid journey exists, you have two options:
    - To be awarded the full score, the procedure should return an array of at most 2 000 000 integers representing a valid journey. More precisely, the elements of this array should be the numbers of the canoes that are used in the journey (in the order they are used).
    - To be awarded a partial score, the procedure should return `true`, an array of more than 2 000 000 integers, or an array of integers not describing a valid journey. (See the Subtasks section for more details.)
- This procedure is called exactly once.

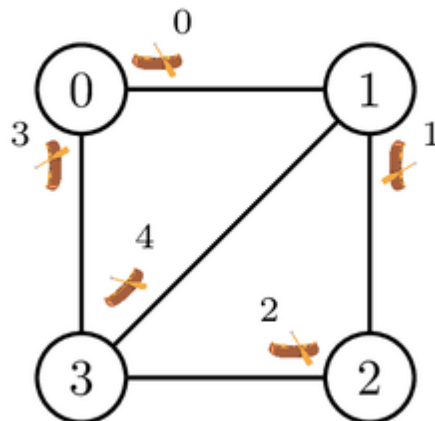
## Examples

### Example 1

Consider the following call:

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

The islands and canoes are shown in the picture below.



One possible valid journey is as follows. Bu Dengklek first sails canoes 0, 1, 2, and 4 in that order. As a result, she is at island 1. After that, Bu Dengklek can sail canoe 0 again as it is currently docked at island 1 and the last canoe she used is not canoe 0. After sailing canoe 0 again, Bu Dengklek is now at island 0. However, canoes 1, 2 and 4 are not docked at the same islands as they were before the journey. Bu Dengklek then continues her journey by sailing canoes 3, 2, 1, 4, and 3 again. Bu

Dengklek is back at island 0 and all the canoes are docked at the same islands as before the journey.

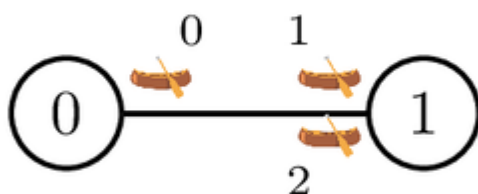
Therefore, the returned value  $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$  represents a valid journey.

## Example 2

Consider the following call:

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

The islands and canoes are shown in the picture below.



Bu Dengklek can only start by sailing canoe 0, after which she can sail either canoe 1 or 2. Note that she cannot sail canoe 0 twice in a row. In both cases, Bu Dengklek is back at island 0. However, the canoes are not docked at the same islands as they were before the journey, and Bu Dengklek cannot sail any canoe afterwards, as the only canoe docked at island 0 is the one she has just used. As there is no valid journey, the procedure should return `false`.

## Constraints

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N - 1$  and  $0 \leq V[i] \leq N - 1$  (for each  $i$  such that  $0 \leq i \leq M - 1$ )
- $U[i] \neq V[i]$  (for each  $i$  such that  $0 \leq i \leq M - 1$ )

## Subtasks

1. (5 points)  $N = 2$
2. (5 points)  $N \leq 400$ . For each pair of distinct islands  $x$  and  $y$  ( $0 \leq x < y \leq N - 1$ ), there are exactly two canoes that can be used to sail between them. One of them is docked at island  $x$ , and the other one is docked at island  $y$ .
3. (21 points)  $N \leq 1000$ ,  $M$  is even, and for each **even**  $i$  such that  $0 \leq i \leq M - 1$ , canoes  $i$  and  $i + 1$  can both be used to sail between islands  $U[i]$  and  $V[i]$ . Canoe  $i$  is initially docked at

island  $U[i]$  and canoe  $i + 1$  is initially docked at island  $V[i]$ . Formally,  $U[i] = V[i + 1]$  and  $V[i] = U[i + 1]$ .

4. (24 points)  $N \leq 1000$ ,  $M$  is even, and for each **even**  $i$  such that  $0 \leq i \leq M - 1$ , canoes  $i$  and  $i + 1$  can both be used to sail between islands  $U[i]$  and  $V[i]$ . Both canoes are initially docked at island  $U[i]$ . Formally,  $U[i] = U[i + 1]$  and  $V[i] = V[i + 1]$ .
5. (45 points) No additional constraints.

For each test case in which a valid journey exists, your solution:

- gets full points if it returns a valid journey,
- gets 35% of the points if it returns `true`, an array of more than 2 000 000 integers, or an array that does not describe a valid journey,
- gets 0 points otherwise.

For each test case in which a valid journey does not exist, your solution:

- gets full points if it returns `false`,
- gets 0 points otherwise.

Note that the final score for each subtask is the minimum of the points for the test cases in the subtask.

## Sample Grader

The sample grader reads the input in the following format:

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

The sample grader prints your answers in the following format:

- If `find_journey` returns a `bool`:
  - line 1: 0
  - line 2: 0 if `find_journey` returns `false`, or 1 otherwise.
- If `find_journey` returns an `int[]`, denote the elements of this array by  $c[0], c[1], \dots, c[k - 1]$ . The sample grader prints:
  - line 1: 1
  - line 2:  $k$
  - line 3:  $c[0] c[1] \dots c[k - 1]$