Milhares de Ilhas

O Arquipélago das Milhares de Ilhas é um grupo de belas ilhas localizadas no Mar de Java. Ele consiste em N ilhas, numeradas de 0 a $N-1$.

Existem M canoas, numeradas de 0 a $M-1$, que podem ser usadas para navegar entre as ilhas. Para cada i, tal que $0 \leq i \leq M-1$, a canoa i pode estar ancorada ou na ilha $U[i]$ ou na ilha $V[i]$, e pode ser usada para navegar entre as ilhas $U[i]$ e $V[i]$. Especificamente, quando a canoa está ancorada na ilha $U[i]$, ela pode ser usada para navegar da ilha $U[i]$ para a ilha $V[i]$, e, ao chegar, a canoa é ancorada na ilha $V[i]$. Similarmente, quando a canoa está ancorada na ilha $V[i]$, ela pode ser usada para navegar da ilha $V[i]$ para a ilha $U[i]$, e, ao chegar, a canoa é ancorada na ilha $U[i]$. Inicialmente, a canoa está ancorada na ilha $U[i]$. É possível que várias canoas possam ser usadas para navegar entre o mesmo par de ilhas. Também é possível que várias canoas estejam ancoradas na mesma ilha.

Por razões de segurança, uma canoa precisa passar por uma manutenção depois de cada vez que for navegada, o que proíbe que a mesma canoa seja navegada duas vezes seguidas. Isto é, depois de usar alguma canoa i, outra canoa deve ser usada antes que a canoa i possa ser usada novamente.

Bu Dengklek quer planejar uma viagem através de algumas das ilhas. Sua viagem é **válida** se e somente se as seguintes condições forem satisfeitas.

- Ela começa e termina sua viagem na ilha 0.
- Ela visita pelo menos uma ilha que não seja a ilha 0.
- Após o término da viagem, cada canoa está ancorada na mesma ilha que estava antes da viagem. Isto é, a canoa i, para cada i tal que $0 \leq i \leq M-1$, deve terminar ancorada na ilha $U[i]$.

Ajude Bu Dengklek a encontrar qualquer viagem válida que navega no máximo $2\,000\,000$ vezes, ou determine que não existe tal viagem válida. Pode-se provar que sob as restrições especificadas nesta tarefa (ver seção Restrições), se existe uma viagem válida, também existe uma viagem válida que não navega mais de $2\,000\,000$ vezes.

Detalhes de Implementação

Você deve implementar o seguinte procedimento:

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

- N: o número de ilhas.
- M: o número de canoas.
- U, V: vetores de tamanho M descrevendo as canoas.
- Este procedimento deve retornar ou um booleano ou um vetor de inteiros.
    - Se não houver uma viagem válida, o procedimento deve retornar `false`.
    - Se existir uma viagem válida, você tem duas opções:
        - Para receber a pontuação completa, o procedimento deve retornar um vetor de no máximo $2\,000\,000$ inteiros representando uma jornada válida. Mais precisamente, os elementos deste vetor devem ser os números das canoas que são utilizadas na viagem (na ordem em que são utilizadas).
        - Para receber uma pontuação parcial, o procedimento deve ou retornar `true`, ou um vetor de mais de $2\,000\,000$ inteiros, ou um vetor de inteiros que não descrevam uma viagem válida. (Consulte a seção Subtarefas para obter mais detalhes).
- Este procedimento é chamado exatamente uma vez.

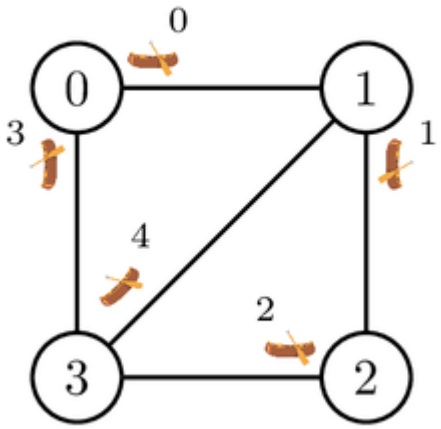
Exemplos

Exemplo 1

Considere a seguinte chamada:

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

As ilhas e canoas são mostradas na figura abaixo.



Uma possível viagem válida é descrita a seguir. Bu Dengklek primeiro navega as canoas 0, 1, 2 e 4, nessa ordem. Como resultado, ela está na ilha 1. Depois disso, a Bu Dengklek pode navegar a

canoa 0 novamente, já que ela está atualmente ancorada na ilha 1 e a última canoa que ela usou não foi a canoa 0. Depois de navegar a canoa 0 novamente, Bu Dengklek está agora na ilha 0. Entretanto, as canoas 1, 2 e 4 não estão ancoradas nas mesmas ilhas que estavam antes da viagem. Bu Dengklek então continua sua viagem navegando as canoas 3, 2, 1, 4 e 3 novamente. Bu Dengklek está de volta à ilha 0 e todas as canoas estão ancoradas nas mesmas ilhas que estavam antes da viagem.

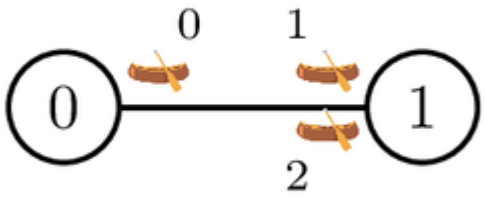
Portanto, o valor retornado $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ representa uma viagem válida.

Exemplo 2

Considere a seguinte chamada:

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

As ilhas e canoas são mostradas na figura abaixo.



Bu Dengklek só pode começar navegando a canoa 0, e após isso ela pode navegar a canoa 1 ou 2. Note que ela não pode navegar a canoa 0 duas vezes seguidas. Em ambos os casos, a Bu Dengklek está de volta à ilha 0. Entretanto, as canoas não estão ancoradas nas mesmas ilhas que estavam antes da viagem, e Bu Dengklek não pode navegar nenhuma canoa depois, pois a única canoa ancorada na ilha 0 é a que ela acabou de usar. Como não há uma viagem válida, o procedimento deve retornar `false`.

Restrições

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N - 1$ e $0 \leq V[i] \leq N - 1$ (para cada i tal que $0 \leq i \leq M - 1$)
- $U[i] \neq V[i]$ (para cada i tal que $0 \leq i \leq M - 1$)

Subtarefas

1. (5 pontos) $N = 2$

2. (5 pontos) $N \leq 400$. Para cada par de ilhas distintas x e y ($0 \leq x < y \leq N-1$), existem exatamente duas canoas que podem ser usadas para navegar entre esse par de ilhas. Uma delas está ancorada na ilha x, e a outra está ancorada na ilha y.
3. (21 pontos) $N \leq 1000$, M é um número par, e para cada i **par** tal que $0 \leq i \leq M-1$, as canoas i e $i+1$ podem ambas ser usadas para navegar entre as ilhas $U[i]$ e $V[i]$. A canoa i está inicialmente ancorada na ilha $U[i]$ e a canoa $i+1$ está inicialmente ancorada na ilha $V[i]$. Formalmente, $U[i] = V[i+1]$ and $V[i] = U[i+1]$.
4. (24 pontos) $N \leq 1000$, M é um número par, e para cada i **par** tal que $0 \leq i \leq M-1$, as canoas i e $i+1$ podem ambas ser usadas para navegar entre as ilhas $U[i]$ e $V[i]$. Ambas as canoas estão inicialmente ancoradas na ilha $U[i]$. Formalmente, $U[i] = U[i+1]$ e $V[i] = V[i+1]$.
5. (45 pontos) Nenhuma restrição adicional.

Para cada caso de teste em que existe uma viagem válida, sua solução:

- recebe pontos completos se retornar uma viagem válida,
- recebe 35% dos pontos se retornar `true` (verdadeiro), um vetor de mais de $2\,000\,000$ de inteiros ou um vetor que não descreve uma viagem válida,
- recebe 0 pontos caso contrário.

Para cada caso de teste em que não existe uma viagem válida, sua solução:

- recebe pontos completos se retornar `false` (falso),
- recebe 0 pontos caso contrário.

Observe que a pontuação final para cada subtarefa é o mínimo dos pontos para os casos de teste na subtarefa.

Corretor Exemplo

O corretor exemplo lê a entrada no seguinte formato:

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

O corretor exemplo imprime a sua resposta no seguinte formato:

- Se `find_journey` returna um `bool`:
  - linha 1: 0
  - linha 2: 0 se `find_journey` returna `false`, ou 1 caso contrário.
- Se `find_journey` returna um `int[]`, vamos denotar por $c[0], c[1], \ldots c[k-1]$, respectivamente, os elementos desse vetor. O corretor exemplo imprime:
  - linha 1: 1
  - linha 2: k
  - linha 3: $c[0]$ $c[1]$ $\ldots$ $c[k-1]$