Miles de Islas

Miles de Islas es un grupo de hermosas islas localizadas en el mar de Java. Consta de N islas, numeradas de 0 a $N-1$.

Hay M canoas, numeradas de 0 a $M-1$, que se pueden usar para navegar entre las islas. Para cada i tal que $0 \leq i \leq M-1$, la canoa i puede estar atracada ya sea en la isla $U[i]$ o en la isla $V[i]$, y puede ser usada para navegar entre las islas $U[i]$ y $V[i]$. De manera más específica, si la canoa está atracada en la isla $U[i]$, puede ser usada para navegar de la isla $U[i]$ a la isla $V[i]$, tras lo cual la canoa estará atracada en la isla $V[i]$. De manera similar, cuando la canoa está atracada en la isla $V[i]$, puede ser usada para navegar de la isla $V[i]$ a la isla $U[i]$, tras lo cual la canoa estará atracada en la isla $U[i]$. Inicialmente, la canoa estará atracada en la isla $U[i]$. Es posible que múltiples canoas puedan ser usadas para navegar entre el mismo par de islas. También es posible que múltiples canoas atraquen en la misma isla.

Por razones de seguridad, una canoa necesita recibir mantenimiento cada vez que zarpa, lo que evita que uno pueda usar la misma canoa dos veces seguidas. Esto es, después de usar la canoa i, debes usar otra canoa antes de volver a usar la canoa i.

Bu Dengklek quiere planear un viaje a través de algunas de las islas. Su viaje es **válido** si y sólo si, se cumplen las siguientes condiciones:

- Ella empieza y termina su viaje en la isla 0.
- Ella visita al menos una isla además de la isla 0.
- Después de que su viaje termine, cada canoa debe estar atracada en la misma isla en la que estaba antes de empezar su viaje. Es decir, para cada canoa i tal que $0 \leq i \leq M-1$, debe estar atracada en la isla $U[i]$.

Ayuda a Bu Dengklek a encontrar algún viaje válido que implique navegar a lo más $2\,000\,000$ veces, o determina que no existe ningún viaje válido. Se puede demostrar que bajo las restricciones del problema (ver la sección de Restricciones), si existe un camino válido, entonces también existe un camino válido que no implica navegar más de $2\,000\,000$ veces.

Detalles de Implementación

Debes implementar el siguiente procedimiento:

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

- N: el número de islas.
- M: el número de canoas.
- U, V: arreglos de tamaño M que describen las canoas.
- Este procedimiento debe regresar ya sea un booleano o un arreglo de enteros.
    - Si no existe un viaje válido, el procedimiento debe regresar `false`.
    - Si existe un viaje válido, tienes dos opciones:
        - Para obtener el puntaje completo, el procedimiento debe regresar un arreglo de a lo más $2\ 000\ 000$ enteros representando un camino válido. Más precisamente, los elementos del arreglo deben ser los índices de las canoas que son usadas en el viaje(en el orden en el que son usadas).
        - Para obtener puntos parciales, el procedimiento debe regresar `true`, un arreglo de más de $2\ 000\ 000$ enteros, o un arreglo de enteros que no describen un viaje válido. (ver la sección de Subtareas para más detalles).
- Este procedimiento será llamado exactamente una vez.

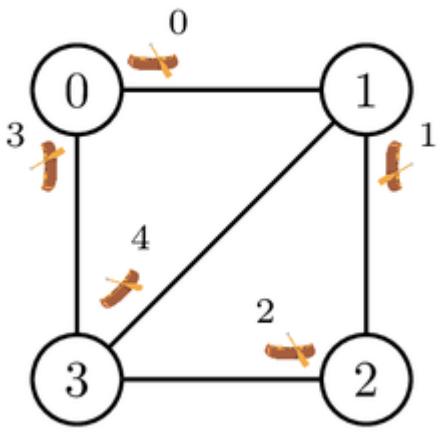
Ejemplos

Ejemplo 1

Considera la siguiente llamada:

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

Las islas y las canoas se muestran en la siguiente imagen.



Un posible viaje válido es el siguiente. Bu Dengklek primero navega en las canoas 0, 1, 2, y 4 en ese orden. Como resultado, ella estará en la isla 1. Después de eso, Bu Dengklek puede volver a navegar en la canoa 0 ya que actualmente está atracada en la isla 1 y la última canoa que usó no es la canoa 0. Después de navegar en la canoa 0 de nuevo, Bu Dengklek está ahora en la isla 0. Sin embargo, las canoas 1, 2 y 4 no están atracadas en las mismas islas en las que estaban antes del

viaje. Bu Dengklek entonces continúa su viaje al navegar en las canoas 3, 2, 1, 4, y 3 de nuevo. Bu Dengklek está de regreso en la isla 0 y todas las canoas están atracadas en la isla en la que estaban antes del viaje.

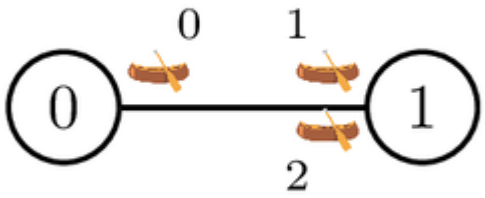
Por lo tanto, el valor regresado $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ representa un viaje válido.

Ejemplo 2

Considera la siguiente llamada:

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

La isla y las canoas se muestran en la siguiente imagen.



Bu Dengklek solo puede empezar a navegar usando la canoa 0, tras lo cual ella puede navegar ya sea en la canoa 1 o 2. Nota que no puede navegar en la canoa 0 dos veces seguidas. En ambos casos, Bu Dengklek regresa a la isla 0. Sin embargo, las canoas no estarán atracadas en la misma isla en la que estaban antes del viaje y Bu Dengklek no puede navegar en ninguna canoa después, ya que la única canoa anclada en la isla 0 es la que acaba de usar. Como no hay ningún viaje válido, el procedimiento debe regresar `false`.

Restricciones

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

Subtareas

1. (5 puntos) $N = 2$
2. (5 puntos) $N \leq 400$. Para cada par de islas distintas x y y ($0 \leq x < y \leq N - 1$), hay exactamente dos canoas que pueden ser usadas para navegar entre ellas. Una de ellas está atracada en la isla x, y la otra está atracada en la isla y.

3. (21 puntos) $N \leq 1000$, M es par, y para cada **par** i tal que $0 \leq i \leq M-1$, las canoas i y $i+1$ pueden ser usadas para navegar entre las islas $U[i]$ y $V[i]$. La canoa i está atracada inicialmente en $U[i]$ y la canoa $i+1$ está inicialmente atracada en la isla $V[i]$. Formalmente, $U[i] = V[i+1]$ y $V[i] = U[i+1]$.
4. (24 puntos) $N \leq 1000$, M es par, y para cada **par** i tal que $0 \leq i \leq M-1$, las canoas i y $i+1$ pueden ser usadas para navegar entre las islas $U[i]$ y $V[i]$. Ambas están atracadas en la isla $U[i]$. Formalmente, $U[i] = U[i+1]$ y $V[i] = V[i+1]$.
5. (45 puntos) Sin consideraciones adicionales.

Para cada caso de prueba en el que existe un viaje válido, tu solución:

- obtiene todos los puntos si regresa un viaje válido,
- obtiene 35% de los puntos si regresa `true`, un arreglo de más de $2\ 000\ 000$ enteros, o un arreglo que no describe un viaje válido,
- obtiene 0 puntos de otra manera.

Para cada caso de prueba en el que no existe un viaje válido, tu solución:

- obtiene puntos completos si regresa `false`,
- obtiene 0 de otra manera.

Nota que el puntaje final de cada subtarea es el mínimo de la cantidad de puntos de los casos de prueba de esa subtarea.

Evaluador de Ejemplo

El evaluador de ejemplo lee la entrada en el siguiente formato:

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

El evaluador de ejemplo imprime tu respuesta en el siguiente formato:

- Si `find_journey` regresa un `bool`:
  - línea 1: 0
  - línea 2: 0 si `find_journey` regreso `false`, o 1 de lo contrario.
- Si `find_journey` regresa un `int[]`, denota los elementos de este arreglo como $c[0], c[1], \ldots c[k-1]$. El evaluador de ejemplo imprime:
  - línea 1: 1
  - línea 2: k
  - línea 3: $c[0]\ c[1]\ \ldots\ c[k-1]$