Miles Islas

Miles islas es un grupo de hermosas islas ubicadas en el Mar de Java. Consiste de N islas, numeradas desde 0 a $N-1$.

Hay M canoas, numeradas desde 0 a $M-1$, que pueden ser usadas para navegar entre islas. Para cada i tal que $0 \leq i \leq M-1$, la canoa i se puede atracar en la isla $U[i]$ o $V[i]$, y puede ser usada para navegar entre las islas $U[i]$ y $V[i]$. Específicamente, cuando la canoa está atracada en la isla $U[i]$, puede ser usada para navegar desde la isla $U[i]$ a la isla $V[i]$, después de lo cual la canoa se atraca en la isla $V[i]$. Similarmente, cuando la canoa esta atracada en la isla $V[i]$, puede ser usada para navegar desde la isla $V[i]$ a la isla $U[i]$, después de lo cual la canoa se atraca en la isla $U[i]$. Inicialmente, la canoa está 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 multiples canoas se encuentre atracadas en la misma isla.

Por razones de seguridad, una canoa necesita ser mantenida luego de cada vez que navega, lo que prohibe que la misma canoa sea usada para navegar dos veces seguidas. Es decir, después de usar alguna canoa i, se debe usar otra canoa antes de poder usar la canoa i nuevamente.

Bu Dengklek quiere planificar un viaje a través de algunas de las islas. Su viaje es **válido** sí y solo sí las siguientes condiciones son satisfechas.

- Ella comienza y termina su viaje en la isla 0.
- Ella visita al menos una isla que no sea la isla 0.
- Luego de que su viaje termina, cada canoa se encuentra atracada en la misma isla que estaba antes de su viaje. Es decir, la canoa i, para cada i tal que $0 \leq i \leq M-1$, debe estar acoplada en la isla $U[i]$.

Ayuda a Bu Dengklek a encontrar cualquier viaje válido que involucre navegar a lo sumo $2\,000\,000$ veces, o determina que tal viaje no existe. Se puede probar que bajo las restricciones especificadas en esta tarea (ver la sección Restricciones), sí un viaje válido existe, también existe un viaje válido que no involucra navegar más de $2\,000\,000$ veces.

Detalles de Implementación

Tu 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 longitud M describiendo las canoas.
- Este procedimiento debe retornar un booleano o un arreglo de enteros.
    - Sí no existe un viaje válido, el procedimiento debe retornar `false`.
    - Si existe un viaje válido, usted tiene dos opciones:
        - Para obtener el puntaje completo el procedimiento debe retornar un arreglo de a lo sumo $2\,000\,000$ enteros representando un viaje válido. Más específicamente, los elementos de este arreglo deben ser los números de las canoas que son usados en el viaje (en el orden que son usadas).
        - Para obtener un puntaje parcial el procedimiento debe retornar `true`, un arreglo con más de $2\,000\,000$ enteros, o un arreglo de enteros que no describe un viaje válido. (Ver la sección Subtareas para más detalles)
    - Este procedimiennto es llamado exáctamente una vez.

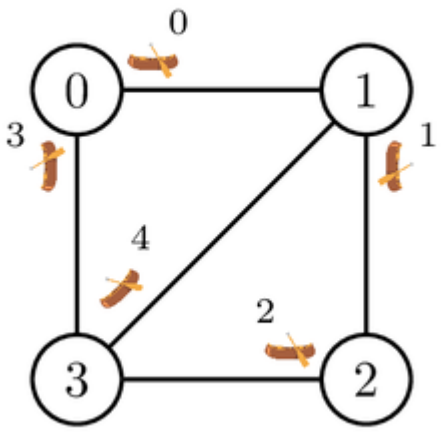
Ejemplos

Ejemplo 1

Considere la siguiente llamada:

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

Las islas y las canoas son mostradas en la imagen que sigue.



Un posible viaje es como sigue. Bu Dengklek primero navega con las canoas 0, 1, 2 y 4 en ese orden. Como resultado, ella está en la isla 1. Luego, Bu Dengklek puede navegar la canoa 0 otra vez ya que se encuentra atracada en la isla 1 y la última canoa que usó no es la canoa 0. Luego de navegar con la canoa 0 de nuevo, Bu Dengklek está ahora en la isla 0. Sin embargo, las canoas 1, 2 y 4 no se encuentran atracadas en las mismas islas que estaban antes de su viaje. Bu Dengklek

entonces continúa su viaje navegando con las canoas 3, 2, 1, 4, y 3 de nuevo. Bu Dengklek está de vuelta en la isla 0 y todas las canoas están atracadas en las mismas islas que estaban antes de su viaje.

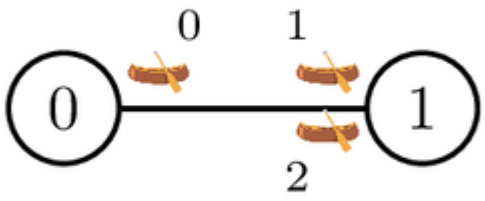
Entonces el valor retornado por el procedimiento $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ representa un viaje válido.

Ejemplo 2

Considere la siguiente llamada:

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

Las islas y las canoas son mostradas en la imagen que sigue.



Bu Dengklek sólo puede comenzar navegando la canoa 0, y luego ella puede usar la canoa 1 o la 2. Nota que ella no puede usar la canoa 0 dos veces seguidas. En ambos casos, Bu Dengklek regresa a la isla 0. Sin embargo, las canoas no se encuentran en las mismas islas que estaban antes de su viaje, y Bu Dengklek no puede navegar ninguna canoa luego, ya que la única canoa atracada en la isla 0 es la canoa que acaba de usar. Como no hay un viaje válido, el procedimiento debe retornar `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) $N \leq 400$. Por cada par de islas distintas x y y ($0 \leq x < y \leq N - 1$), hay exáctamente dos canoas que pueden ser usadas para navegar entre ellas. Una de ellas está

atracada en la isla x, y la otra en la isla y
3. (21 puntos) $N \leq 1000$, M es par, y por cada i **par** tal que $0 \leq i \leq M-1$, ambas canoas i y $i+1$ pueden ser usadas para navegar entre las islas $U[i]$ y $V[i]$. La canoa i está inicialmente atracada en la isla $U[i]$ y la canoa $i+1$ 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 por cada i **par** tal que $0 \leq i \leq M-1$, ambas canoas i y $i+1$ pueden ser usadas para navegar entre las islas $U[i]$ y $V[i]$. Ambas canoas están inicialmente atracadas en la isla $U[i]$. Formalmente $U[i] = U[i+1]$ y $V[i] = V[i+1]$.
5. (45 puntos) No hay restricciones adicionales.

Por cada caso de prueba en el que haya un viaje válido, tu solución:

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

Por cada caso de prueba en el que no exista una solución valida, tu solución:

- recibe todos los puntos si retorna `false`,
- recibe 0 puntos de otra manera.

Tenga en cuenta que la puntuación final de cada subtarea es el mínimo de los puntos de los casos de prueba de la 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 tus respuestas en el siguiente formato:

- Sí `find_journey` retorna un `bool`:
  - línea 1: 0
  - línea 2: 0 sí `find_journey` retorna `false`, o 1 de otra manera.
- Sí `find_journey` retorna un `int[]`, denota los elementos de este arreglo por $c[0], c[1], \ldots c[k-1]$. El evaluador de ejemplo imprime:
  - línea 1: 1
  - línea 2: k
  - línea 3: $c[0]$ $c[1]$ $\ldots$ $c[k-1]$