Les Îles Seribu

Les Îles Seribu sont un archipel de belles îles situées dans la mer de Java. Il est formé de N îles, numérotées de 0 à $N-1$.

Il y a M canoës, numérotés de 0 à $M-1$, qui peuvent être utilisés pour naviguer entre les îles. Pour chaque i tel que $0 \leq i \leq M-1$, le canoë i peut être à quai sur l'île $U[i]$ ou $V[i]$, et peut être utilisé pour naviguer entre les îles $U[i]$ et $V[i]$. Spécifiquement, quand le canoë est à quai sur l'île $U[i]$, il peut être utilisé pour naviguer de l'île $U[i]$ à l'île $V[i]$, après quoi le canoë est à quai sur l'île $V[i]$. Similairement, quand le canoë est à quai sur l'île $V[i]$, il peut être utilisé pour naviquer de l'île $V[i]$ à l'île $U[i]$, après quoi le canoë est à quai sur l'île $U[i]$. Initialement, le canoë est à quai sur l'île $U[i]$. Il est possible que plusieurs canoës soit utilisés pour naviguer entre la même paire d'îles. Il est également possible que plusieurs canoës soient à quai sur la même île.

Pour des raisons de sécurité, un canoë doit être inspecté après chaque fois qu'il a navigué, ce qui interdit au même canoë de naviguer deux fois de suite. C'est-à-dire qu'après avoir utilisé le canoë i, un autre canoë doit être utilisé avant que le canoë i puisse être utilisé à nouveau.

Bu Dengklek veut planifier son voyage à travers les îles. Son voyage est **valide** si et seulement si les conditions suivantes sont satisfaites.

- Elle commence et termine son voyage sur l'île 0.
- Elle visite au moins une autre île que l'île 0.
- Après la fin du voyage, chaque canoë est à quai sur la même île qu'avant le voyage. C'est-à-dire que, pour tout i tel que $0 \leq i \leq M-1$, le canoë i doit être à quai sur l'île $U[i]$.

Aidez Bu Dengklek à trouver n'importe quel voyage valide avec au plus $2\ 000\ 000$ navigations, ou déterminez qu'il n'existe pas de voyage valide. Il peut être prouvé que sous les contraintes spécifiées dans ce sujet (voir la section Contraintes), si un voyage valide existe, il existe aussi un voyage valide qui utilise moins de $2\ 000\ 000$ navigations.

Détails d'implémentation

Vous devez implémenter la fonction suivante :

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

- N : le nombre d'îles.

- M : le nombre de canoës.
- U, V : tableaux de longueur M décrivant les canoës.
- Cette fonction doit renvoyer soit un booléen, soit un tableau d'entiers.
    - S'il n'existe pas de voyage valide, la fonction doit renvoyer `false`.
    - S'il existe un voyage valide, vous avez deux options :
        - Pour obtenir un score complet, la fonction doit renvoyer un tableau d'au plus $2\,000\,000$ entiers représentant un voyage valide. Plus précisément, les éléments de ce tableau doivent être les nombres des canoës qui sont utilisés par le voyage (dans l'ordre dans lequel ils sont utilisés).
        - Pour obtenir un score partiel, la fonction doit renvoyer `true`, un tableau de plus de $2\,000\,000$ entiers, ou un tableau d'entiers ne décrivant pas un voyage valide. (Voir la section Sous-tâches pour plus de détails.)
- Cette fonction est appelée exactement une fois.

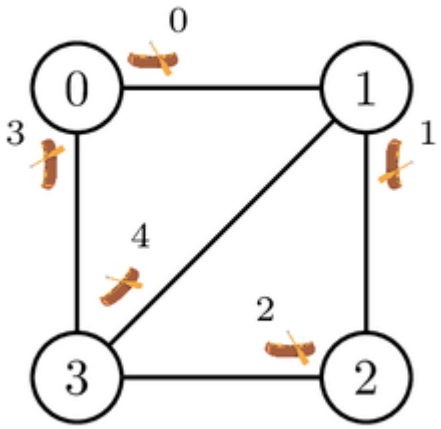
Exemples

Exemple 1

Considérez l'appel suivant :

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

Les îles et les canoës sont illustrés par l'image ci-dessous.



Un voyage valide possible est le suivant. Bu Dengklek navigue d'abord avec les canoës 0, 1, 2 et 4 dans cet ordre. Ainsi, elle se trouve sur l'île 1. Après cela, Bu Dengklek peut naviguer à nouveau avec le canoë 0 puisqu'il est à ce moment à quai sur l'île 1 et le dernier canoë qu'elle a utilisé n'est pas le canoë 0. Après avoir navigué le canoë 0 à nouveau, Bu Dengklek se trouve sur l'île 0. Toutefois, les canoës 1, 2 et 4 ne sont pas à quai sur les mêmes îles qu'au début du voyage. Bu

Dengklek continue alors son voyage en naviguant les canoës 3, 2, 1, 4 et 3 à nouveau. Bu Dengklek est de retour à l'île 0 et tous les canoës sont à quai sur les mêmes îles qu'avant le voyage.

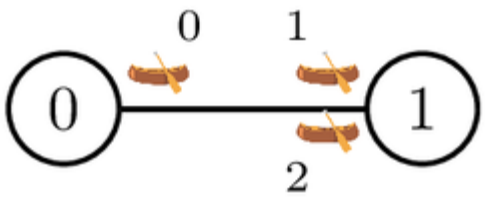
Par conséquent, la valeur de retour $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ représente un voyage valide.

Exemple 2

Considérez l'appel suivant :

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

Les îles et canoës sont illustrés par l'image ci-dessous.



Bu Dengklek peut commencer seulement en naviguant le canoë 0, après lequel elle peut naviguer le canoë 1 ou 2. Remarquez qu'elle ne peut pas naviguer le canoë deux fois de suite. Dans les deux cas, Bu Dengklek est de retour sur l'île 0. Toutefois, les canoës ne sont pas à quai sur les mêmes îles qu'au début du voyage, et Bu Dengklek ne peut pas naviguer un canoë après cela, puisque le seul canoë à quai sur l'île 0 est celui qu'elle vient juste d'utiliser. Comme il n'y a pas de voyage valide, la fonction doit renvoyer `false`.

Contraintes

- $2 \leq N \leq 100\ 000$
- $1 \leq M \leq 200\ 000$
- $0 \leq U[i] \leq N - 1$ et $0 \leq V[i] \leq N - 1$ (pour chaque i tel que $0 \leq i \leq M - 1$)
- $U[i] \neq V[i]$ (pour chaque i tel que $0 \leq i \leq M - 1$)

Sous-tâches

1. (5 points) $N = 2$
2. (5 points) $N \leq 400$. Pour chaque paire d'îles distinctes x et y ($0 \leq x < y \leq N - 1$), il y a exactement deux canoës qui peuvent être utilisés pour naviguer entre elles. L'un d'entre eux est à quai sur l'île x, et l'autre est à quai sur l'île y.

3. (21 points) $N \leq 1000$, M est pair, et pour chaque i **pair** tel que $0 \leq i \leq M-1$, les canoës i et $i+1$ peuvent tous les deux être utilisés pour naviguer entre les îles $U[i]$ et $V[i]$. Le canoë i est initialement à quai sur l'île $U[i]$ et le canoë $i+1$ est initialement à quai sur l'île $V[i]$. Formellement, $U[i] = V[i+1]$ et $V[i] = U[i+1]$.
4. (24 points) $N \leq 1000$, M est pair, et pour chaque i **pair** tel que $0 \leq i \leq M-1$, les canoës i et $i+1$ peuvent tous les deux être utilisés pour naviguer entre les îles $U[i]$ et $V[i]$. Les deux canoës sont initialement à quai sur l'île $U[i]$. Formellement, $U[i] = U[i+1]$ et $V[i] = V[i+1]$.
5. (45 points) Pas de contrainte supplémentaire.

Pour chaque test pour lequel un voyage valide existe, votre solution :

- obtient un score complet s'il renvoie un voyage valide,
- obtient 35% des points s'il renvoie `vrai`, un tableau de plus que $2\,000\,000$ entiers, ou un tableau qui ne décrit pas un voyage valide,
- obtient 0 point sinon.

Pour chaque test pour lequel il n'existe pas de voyage valide, votre solution :

- obtient un score complet si elle renvoie `false`,
- obtient 0 point sinon.

Notez que le score final pour chaque sous-tâche est le minimum de points obtenus parmi les tests de cette sous-tâche.

Évaluateur d'exemple

L'évaluateur d'exemple lit l'entrée au format suivant :

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

L'évaluateur d'exemple affiche vos réponses au format suivant :

- Si `find_journey` renvoie un `bool` :
  - ligne 1 : 0
  - ligne 2 : 0 si `find_journey` renvoie `false`, ou 1 sinon.
- Si `find_journey` renvoie un `int[]`, en notant les éléments de ce tableau $c[0], c[1], \ldots c[k-1]$, l'évaluateur d'exemple affiche :
  - ligne 1 : 1
  - ligne 2 : k
  - ligne 3 : $c[0]$ $c[1]$ $\ldots$ $c[k-1]$