جزر الألفية

جزر الألفية هي مجموعة من الجزر الجميلة في بحر جافا. تتكون جزر الألفية من N جزيرة، مرقمة من 0 إلى $N-1$.

هنالك M زورقاً، مرقمة من 0 إلى $M-1$، تستخدم للإبحار بين الجزر.

لكل $0 \leq i \leq M-1$، يرسو الزورق رقم i في إحدى الجزيرتين $U[i]$ أو $V[i]$، ويستخدم للإبحار بين الجزيرتين $U[i]$ و $V[i]$. بشكل أكثر دقة، عندما يرسو الزورق في الجزيرة $U[i]$، يمكن استخدام الزورق للإبحار من الجزيرة $U[i]$ إلى الجزيرة $V[i]$، بعد ذلك يصبح الزورق راسياً في الجزيرة $V[i]$. وبشكل مشابه، عندما يرسو الزورق في الجزيرة $V[i]$، يمكن استخدام الزورق للإبحار من الجزيرة $V[i]$ إلى الجزيرة $U[i]$، بعد ذلك يصبح الزورق راسياً في الجزيرة $U[i]$. بدايةً، يكون الزورق راسياً في الجزيرة $U[i]$.

من الممكن أن يبحر أكثر من زورق بين نفس الزوج من الجزر. كما أنه من الممكن أن ترسو عدة زوارق في نفس الجزيرة.

لأسباب متعلقة بالسلامة، يجب القيام بعمليات صيانة للزورق بعد كل عملية إبحار، مما يمنع استخدام نفس الزورق مرتين متتاليتين. بمعنى آخر، بعد استخدام الزورق رقم i، يجب استخدام زورق آخر قبل إعادة استخدام الزورق رقم i.

ترغب سارة بالتخطيط لرحلة لزيارة بعض الجزر. تعتبر الرحلة **صحيحة** (valid) إذا وفقط إذا تحققت الشروط التالية:

- تنطلق الرحلة وتنتهي في الجزيرة رقم 0.
- تزور سارة جزيرة واحدة على الأقل غير الجزيرة رقم 0.
- بعد انتهاء الرحلة، يكون كل زورق راسياً في نفس الجزيرة التي كان راسياً فيها قبل بدء الرحلة. أي، يكون الزورق $0 \leq i \leq M-1$ راسياً في الجزيرة $U[i]$.

ساعد سارة على ايجاد رحلة صحيحة تتضمن على الأكثر $2,000,000$ عملية إبحار، أو قرر أنه لا توجد رحلة صحيحة ممكنة.

يمكن البرهان أنه ضمن القيود المذكورة في نص المسألة (انظر Constraints)، إذا كان هنالك رحلة صحيحة ممكنة، يكون هنالك أيضاً رحلة صحيحة ممكنة لا تتضمن أكثر من $2,000,000$ عملية إبحار.

تفاصيل التحقيق

عليك تحقيق التابع التالي:

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

- N: عدد الجزر.
- M: عدد الزوارق.
- U ,V: مصفوفات من طول M تعبر عن الزوارق.
- يجب أن يرد التابع إما قيمة بوليانية أو مصفوفة أعداد صحيحة.
  - إذا لم يكن هنالك أي رحلة صحيحة ممكنة، يجب أن يرد التابع `false`.
  - إذا كان هنالك رحلة صحيحة ممكنة، لديك خيارين:
    - لنيل العلامة الكاملة، يجب أن يرد التابع مصفوفة لا تتجاوز $2,000,000$ عدداً صحيحاً تمثل رحلة صحيحة ممكنة. بشكل أدق، تعبر عناصرالمصفوفة عن أرقام الزوارق التي يتم استخدامها في الرحلة (وفق ترتيب استخدامها).
    - لنيل جزء من العلامة، يجب أن يرد التابع القيمة `true`، أو مصفوفة تتجاوز $2,000,000$ عدداً صحيحاً، أو مصفوفة من الأعداد الصحيحة لا تعبر عن رحلة صحيحة. (انظر Subtasks للتفاصيل).
- يتم استدعاء هذا التابع مرة واحدة فقط.

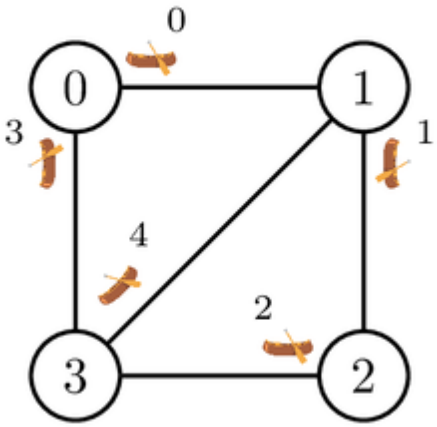
أمثلة

المثال 1

ليكن الاستدعاء التالي:

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

تعبر الصورة التالية عن الجزر والزوارق.



تكون واحدة من الرحلات الصحيحة الممكنة على الشكل التالي: تستخدم سارة الزوارق 0 ثم 1 ثم 2 ثم 4 وبذلك تكون قد وصلت إلى الجزيرة 1. بعد ذلك، يمكن لسارة استخدام الزورق 0 مرة أخرى حيث يكون راسياً في الجزيرة 1 وهو ليس آخر زورقاً تم استخدامه. بعد استخدام الزورق 0 هذه المرة، تصل سارة إلى الجزيرة 0 مرة ثانية. ولكن، لا تكن الزوارق 1 و2 و4 راسية في الجزر التي كانت راسية فيها عند بدء الرحلة. تكمل سارة رحلتها باستخدام الزوارق 3 ثم 2 ثم 1 ثم 4 ثم 3 مرة أخرى. تكون سارة قد عادت إلى الجزيرة 0 وتكون جميع الزوارق راسية في نفس الجزر التي كانت راسية فيها قبل بدء الرحلة.

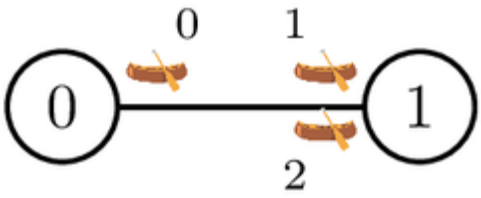
أي، يجب ان يرد التابع القيمة $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ والتي تعبر عن رحلة صحيحة.

المثال 2

ليكن الاستدعاء التالي:

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

تعبر الصورة التالية عن الجزر والزوارق.



يمكن لسارة أن تبدأ رحلتها باستخدام الزورق رقم 0 فقط. بعدها يمكن لسارة استخدام أحد الزورقين 1 أو 2. لاحظ أنها لن تستطيع الابحار باستخدام الزورق رقم 0 مرتين متتاليتين. في كلتا الحالتين، تعود سارة إلى الجزيرة 0. ولكن، لا تكن الزوارق راسية في الجزر التي كانت راسية فيها عند بدء الرحلة، ولن تتمكن سارة من الابحار باستخدام أي زورق بعد ذلك لأن الزورق الوحيد الراسي في الجزيرة 0 هو الزورق الذي تم استخدامه في آخر عملية ابحار.

وبما أنه لا يوجد رحلة صحيحة ممكنة، يجب أن يرد التابع القيمة `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 $c[0], c[1], \ldots c[k - 1]$. The sample grader prints:
  - line 1: 1
  - line 2: k
  - line 3: $c[0]$ $c[1]$ $\ldots$ $c[k - 1]$