ค่ายกลแสนเกาะ

หมู่เกาะเทาซันด์เป็นหมู่เกาะที่ประกอบด้วยเกาะอันสวยงามมากมายในทะเลชวา หมู่เกาะนี้ประกอบด้วยเกาะจำนวน N เกาะ หมายเลข 0 ถึง $N-1$

มีเรือแคนูจำนวน M ลำหมายเลข 0 ถึง $M-1$ ที่ใช้สำหรับการเดินทางระหว่างหมู่เกาะ สำหรับค่า i ($0 \leq i \leq M-1$) เรือแคนู i สามารถจอดได้ที่เกาะ $U[i]$ หรือ $V[i]$ เท่านั้นและสามารถใช้เดินทางไปมาระหว่างเกาะ $U[i]$ และ $V[i]$ กล่าวคือ เรือแคนูที่จอดอยู่ที่เกาะ $U[i]$ สามารถใช้สำหรับเดินทางจากเกาะ $U[i]$ ไปยังเกาะ $V[i]$ ได้ และหลังจากนั้นเรือแคนูจะจอดที่เกาะ $V[i]$ ทำนองเดียวกันเรือแคนูที่จอดอยู่ที่เกาะ $V[i]$ สามารถใช้สำหรับเดินทางจากเกาะ $V[i]$ ไปยังเกาะ $U[i]$ ได้และหลังจากนั้นเรือแคนูจะจอดอยู่ที่เกาะ $U[i]$ โดยตอนเริ่มต้นเรือแคนูทั้งหมดจอดอยู่ที่เกาะ $U[i]$ เป็นไปได้ว่าเรือแคนูหลายลำจะใช้เดินทางระหว่างคู่ของเกาะคู่เดียวกัน และเป็นไปได้ว่าเรือแคนูหลายลำจะจอดอยู่ที่เกาะเดียวกัน

ด้วยเหตุผลทางด้านความปลอดภัย หลังจากการใช้งานเรือแคนูแต่ละเที่ยว เรือแคนูจะได้รับการซ่อมบำรุงเสมอ ดังนั้นเรือแคนูจะไม่สามารถถูกใช้งานสองเที่ยวต่อติดกันได้ กล่าวคือหลังจากการใช้เรือแคนู i แล้วจะต้องใช้เรือแคนูลำอื่นก่อนจึงจะสามารถใช้เรือแคนู i ได้อีกครั้ง

คุณบู่ เดงเกล็กต้องการวางแผนเส้นทางการท่องเที่ยวไปยังเกาะต่าง ๆ บางเกาะ เส้นทางการท่องเที่ยวของเธอจะ**ถูกต้อง**ก็ต่อเมื่อเงื่อนไขทั้งหมดด้านล่างเป็นจริง

- เธอเริ่มต้นและสิ้นสุดการเดินทางที่เกาะ 0
- เธอจะต้องเดินทางผ่านเกาะอื่นอย่างน้อยหนึ่งเกาะที่ไม่ใช่เกาะ 0
- หลังจากการเดินทางเสร็จสิ้น เรือแคนูแต่ละลำจะต้องจอดไว้ที่เกาะเดียวกันกับตอนก่อนเริ่มการเดินทาง กล่าวคือ เรือแคนู i สำหรับทุก i ($0 \leq i \leq M-1$) จะต้องจอดที่เกาะ $U[i]$

ให้คุณช่วยคุณบู่ เดงเกล็กหาเส้นทางการท่องเที่ยวที่ถูกต้องมาหนึ่งเส้นทาง ที่มีจำนวนเดินทางไม่เกิน $2\,000\,000$ เที่ยว หรือ ระบุว่าไม่มีเส้นทางที่ถูกต้องอยู่ นอกจากนี้มีการพิสูจน์มาแล้วว่า ภายใต้ข้อกำหนดสำหรับโจทย์ข้อนี้ (ดูรายละเอียดในข้อกำหนด) หากมีเส้นทางที่ถูกต้องอยู่ จะมีเส้นทางที่ถูกต้องที่ใช้การเดินทางไม่เกิน $2\,000\,000$ เที่ยว

รายละเอียดการเขียนโปรแกรม

คุณต้องเขียนฟังก์ชันด้านล่าง

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

- N: จำนวนเกาะ
- M: จำนวนเรือแคนู
- U, V: อาร์เรย์ความยาว M อธิบายเรือแคนู

- ฟังก์ชันนี้จะคืนค่าบูลีนหรืออาร์เรย์ของจำนวนเต็ม เพียงอย่างใดอย่างหนึ่ง
  - หากไม่มีเส้นทางที่ถูกต้อง ให้คืนค่า `false`
  - หากมีเส้นทางที่ถูกต้อง จะเป็นไปได้สองแบบ
    - คุณจะได้รับคะแนนเต็ม หากฟังก์ชันนี้คืนค่าอาร์เรย์ที่ประกอบด้วยจำนวนเต็มไม่เกิน $2\,000\,000$ จำนวน ที่แสดงถึงเส้นทางการเดินทางที่ถูกต้อง กล่าวให้ชัดเจนคือคุณต้องระบุหมายเลขเรือแคนูได้อย่างถูกต้อง (ตามลำดับที่ใช้)
    - คุณจะได้คะแนนบางส่วน หากฟังก์ชันนี้คืนค่า `true` หรือ อาร์เรย์ประกอบด้วยจำนวนมากกว่า $2\,000\,000$ จำนวน หรือ อาร์เรย์ระบุเส้นทางที่ไม่ถูกต้อง (ดูรายละเอียดในปัญหาย่อย)
  - ฟังก์ชันนี้ถูกเรียกใช้งานเพียงหนึ่งครั้ง

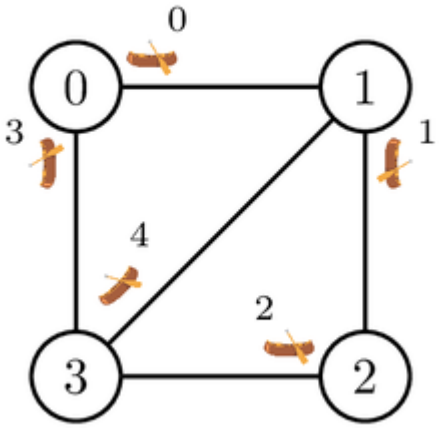
ตัวอย่าง

ตัวอย่างที่ 1

พิจารณาการเรียกฟังก์ชัน:

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

เกาะและเรือแคนูทั้งหมดแสดงดังรูปด้านล่าง



เส้นทางที่ถูกต้องเส้นทางหนึ่งเป็นดังนี้ คุณบู่ เดงเกล็กเริ่มต้นจากแคนู 0 จากนั้น 1, 2, และ 4 ตามลำดับ ผลที่เกิดขึ้นคือเธอจะอยู่ที่เกาะหมายเลข 1 จากนั้นคุณบู่ เดงเกล็กจะใช้เรือแคนู 0 อีกครั้ง ตอนนี้จอดอยู่ที่เกาะหมายเลข 1 และเรือแคนูลำสุดท้ายที่เธอใช้ไม่ใช่เรือแคนูหมายเลข 0
ภายหลังจากการใช้เรือแคนู 0 เธอจะกลับไปที่เกาะหมายเลข 0
แต่อย่างไรก็ตามเรือแคนู 1, 2 และ 4 ยังไม่ได้กลับไปจอดที่เกาะเริ่มต้นก่อนเริ่มการเดินทาง คุณบู่ เดงเกล็กจึงเดินทางต่อโดยใช้เรือแคนู 3, 2, 1, 4, และ 3 อีกครั้ง ตอนนี้คุณบู่ เดงเกล็กจะกลับมาที่เกาะหมายเลข 0 และเรือแคนูทั้งหมดก็กลับไปจอดที่เกาะเริ่มต้นก่อนการเดินทาง

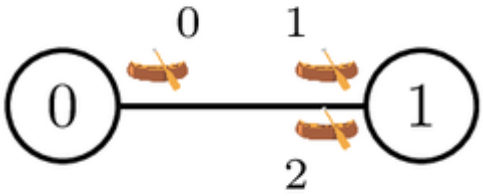
ดังนั้น $[0, 1, 2, 4, 0, 3, 2, 1, 4, 3]$ เป็นคำตอบหนึ่งที่ถูกต้องสำหรับฟังก์ชันนี้

ตัวอย่างที่ 2

พิจารณาการเรียกฟังก์ชัน:

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

เกาะและเรือแคนูทั้งหมดแสดงดังรูปด้านล่าง



คุณบู่ เดงเกล็กสามารถเริ่มต้นโดยใช้เรือแคนู 0 เท่านั้น จากนั้นเธอสามารถเลือกที่จะใช้เรือแคนู 1 หรือ 2
เนื่องจากเธอไม่สามารถใช้เรือแคนู 0 สองเที่ยวต่อติดกันได้ ทั้งสองกรณีเธอจะกลับมาอยู่ที่เกาะหมายเลข 0 อย่างไรก็ตาม เรือแคนูบางลำไม่ได้จอดอยู่ที่เกาะเริ่มต้น และคุณบู่ เดงเกล็กไม่มาสามารถใช้เรือแคนูลำอื่นใดได้อีกด้วย เนื่องจากตอนนี้เรือที่จอดอยู่ที่เกาะหมายเลข 0 มีเพียงลำเดียวเท่านั้นคือลำที่เธอเพิ่งใช้เดินทาง

ดังนั้นไม่มีเส้นทางที่ถูกต้อง ฟังก์ชันนี้จะต้องคืนค่า `false`

ข้อจำกัด

- $2 \leq N \leq 100\,000$
- $1 \leq M \leq 200\,000$
- $0 \leq U[i] \leq N-1$ และ $0 \leq V[i] \leq N-1$ (สำหรับทุก i ที่ $0 \leq i \leq M-1$)
- $U[i] \neq V[i]$ (สำหรับทุก i ที่ $0 \leq i \leq M-1$)

ปัญหาย่อย

1. (5 คะแนน) $N = 2$
2. (5 คะแนน) $N \leq 400$ สำหรับแต่ละคู่ของเกาะที่แตกต่างกัน x และ y ($0 \leq x < y \leq N-1$) จะมีเรือแคนูจำนวนเพียง 2 ลำเท่านั้นที่สามารถเดินทางระหว่างคู่เกาะนี้ โดยที่เรือลำหนึ่งจะจอดอยู่ที่เกาะ x และอีกลำจะจอดอยู่ที่เกาะ y
3. (21 คะแนน) $N \leq 1000$, M เป็นจำนวนคู่ และสำหรับทุก**จำนวนคู่** i ที่ $0 \leq i \leq M-1$, เรือแคนู i และ $i+1$ ทั้งคู่จะใช้สำหรับการเดินทางระหว่างเกาะ $U[i]$ และ $V[i]$ โดยที่เรือแคนู i จะเริ่มต้นจอดอยู่ที่เกาะ $U[i]$ และเรือแคนู $i+1$ จะเริ่มต้นจอดอยู่ที่เกาะ $V[i]$ นั่นคือ $U[i] = V[i+1]$ และ $V[i] = U[i+1]$
4. (24 คะแนน) $N \leq 1000$, M เป็นจำนวนคู่ และสำหรับทุก**จำนวนคู่** i ที่ $0 \leq i \leq M-1$, เรือแคนู i และ $i+1$ ทั้งคู่จะใช้สำหรับการเดินทางระหว่างเกาะ $U[i]$ และ $V[i]$ โดยที่เรือแคนูทั้งคู่จะเริ่มต้นจอดอยู่ที่เกาะ $U[i]$ นั่นคือ $U[i] = U[i+1]$ และ $V[i] = V[i+1]$
5. (45 คะแนน) ไม่มีข้อกำหนดเพิ่มเติม

สำหรับแต่ละทดสอบที่มีเส้นทางการเดินทางที่ถูกต้อง:

- คุณจะได้รับคะแนนเต็ม ถ้าฟังก์ชันของคุณคืนคำตอบเส้นทางการเดินทางที่ถูกต้อง
- คุณจะได้รับ 35% ถ้าฟังก์ชันของคุณคืนค่า `true` หรือ อาร์เรย์ประกอบด้วยจำนวนมากกว่า $2\,000\,000$ จำนวน หรือ อาร์เรย์ระบุเส้นทางการเดินทางที่ไม่ถูกต้อง
- คุณจะได้รับ 0 คะแนนสำหรับกรณีอื่น ๆ

สำหรับแต่ละทดสอบที่ไม่มีเส้นทางการเดินทางที่ถูกต้อง:

- คุณจะได้รับคะแนนเต็ม ถ้าฟังก์ชันของคุณคืนค่า `false`
- คุณจะได้รับ 0 คะแนนสำหรับกรณีอื่น ๆ

หมายเหตุ คะแนนสุดท้ายของแต่ละปัญหาย่อยจะเป็นคะแนนน้อยที่สุดของชุดทดสอบในปัญหาย่อยนั้น

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้:

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

เกรดเดอร์ตัวอย่างจะแสดงคำตอบของคุณในรูปแบบต่อไปนี้:

- ถ้าหาก `find_journey` คืนค่า `bool`:
  - บรรทัดที่ 1: 0
  - บรรทัดที่ 2: 0 ถ้า `find_journey` คืนค่า `false`, หรือ 1 สำหรับกรณีอื่น ๆ
- ถ้าหาก `find_journey` คืนค่า `int[]` ที่นิยามโดย $c[0], c[1], \ldots c[k-1]$ เกรดเดอร์ตัวอย่างจะแสดง:
  - บรรทัดที่ 1: 1
  - บรรทัดที่ 2: k
  - บรรทัดที่ 3: $c[0]$ $c[1]$ $\ldots$ $c[k-1]$