เกม (Game)

หลังจากได้ค้นพบดาวเคราะห์จำนวน n ดวง เรียกเป็นหมายเลข 0 ถึง $n-1$ เหล่าฟาโรห์ได้เริ่มสร้างระบบขนส่งระหว่างดาวเคราะห์ด้วย **อุปกรณ์เคลื่อนย้ายมวลสารแบบทิศทางเดียว** (one-way teleporters) อุปกรณ์เคลื่อนย้ายมวลสารแต่ละเครื่องจะมีดาวเคราะห์เริ่มต้นและดาวเคราะห์ปลายทาง เมื่อนักท่องเที่ยวใช้อุปกรณ์เคลื่อนย้ายมวลสารที่ดาวเคราะห์เริ่มต้น นักท่องเที่ยวจะถูกย้ายมวลสารไปยังดาวเคราะห์ปลายทาง เป็นไปได้ที่ดาวเคราะห์เริ่มต้นและดาวเคราะห์ปลายทางของอุปกรณ์เคลื่อนย้ายมวลสารอาจจะเป็นดาวเคราะห์ดวงเดียวกัน อุปกรณ์เคลื่อนย้ายมวลสารที่มีดาวเคราะห์เริ่มต้นเป็น u และดาวเคราะห์ปลายทางเป็น v จะเขียนแทนด้วย (u, v)

เพื่อจะกระตุ้นการใช้งานระบบเคลื่อนย้ายมวลสาร ฟาโรห์จึงได้สร้างเกมที่นักท่องเที่ยวสามารถเล่นได้ระหว่างที่เดินทางในระบบ นักท่องเที่ยวจะเริ่มเล่นเกมที่ดาวเคราะห์ใดก็ได้ ดาวเคราะห์ที่ $0, 1, \ldots, k-1$ ($k \leq n$) จะเรียกว่า **ดาวเคราะห์พิเศษ** เมื่อใดก็ตามที่นักท่องเที่ยวเดินทางถึงดาวเคราะห์พิเศษ นักท่องเที่ยวจะได้ประทับตราหนึ่งครั้ง

ในปัจจุบัน สำหรับค่า i ต่าง ๆ ที่ $0 \leq i \leq k-2$ จะมีอุปกรณ์เคลื่อนย้ายมวลสาร $(i, i+1)$ อยู่ อุปกรณ์เคลื่อนย้ายมวลสารจำนวน $k-1$ เครื่องเหล่านี้ จะเรียกว่า **อุปกรณ์เคลื่อนย้ายมวลสารเริ่มต้น**

อุปกรณ์เคลื่อนย้ายมวลสารใหม่ถูกเพิ่มเข้ามาทีละเครื่อง เมื่อมีเครื่องเคลื่อนย้ายมวลสารใหม่เข้ามา ก็อาจจะเป็นไปได้ที่นักท่องเที่ยวจะสามารถได้รับการประทับตราเป็นจำนวนไม่จำกัดครั้ง เพื่อความชัดเจน เหตุการณ์ดังกล่าวจะเกิดขึ้นเมื่อมีลำดับของดาวเคราะห์ $w[0], w[1], \ldots, w[t]$ ที่สอดคล้องกับเงื่อนไขดังนี้:

- $1 \leq t$
- $0 \leq w[0] \leq k-1$
- $w[t] = w[0]$
- สำหรับแต่ละค่า i ($0 \leq i \leq t-1$) มีอุปกรณ์เคลื่อนย้ายมวลสาร $(w[i], w[i+1])$ ในระบบ

สังเกตว่านักท่องเที่ยวสามารถใช้อุปกรณ์เคลื่อนย้ายมวลสารเริ่มต้นและอุปกรณ์เคลื่อนย้ายมวลสาร **อื่น ๆ** ที่ได้เพิ่มเข้ามาในระบบแล้ว

งานของคุณก็คือช่วยตรวจสอบยืนยันกับฟาโรห์ว่า ภายหลังมีการเพิ่มอุปกรณ์เคลื่อนย้ายมวลสารแต่ละเครื่อง เป็นไปได้หรือไม่ที่นักท่องเที่ยวจะสามารถได้รับการประทับตราเป็นจำนวนไม่จำกัดครั้ง

รายละเอียดการเขียนโปรแกรม

คุณจะต้องเขียนฟังก์ชันต่อไปนี้:

```
init(int n, int k)
```

- n: จำนวนดาวเคราะห์
- k: จำนวนดาวเคราะห์พิเศษ

- ฟังก์ชันนี้จะถูกเรียกหนึ่งครั้ง ก่อนการเรียกฟังก์ชัน `add_teleporter` ทั้งหมด

```
int add_teleporter(int u, int v)
```

- u และ v: ดาวเคราะห์เริ่มต้นและดาวเคราะห์ปลายทางของอุปกรณ์เคลื่อนย้ายมวลสารที่เพิ่มเข้าในระบบ
- ฟังก์ชันจะถูกเรียกไม่เกิน m ครั้ง (สำหรับค่าของ m ให้ดูในเงื่อนไข)
- ฟังก์ชันจะต้องคืนค่า 1 ถ้าหลังจากการเพิ่มอุปกรณ์เคลื่อนย้ายมวลสาร (u, v) แล้ว นักท่องเที่ยวสามารถได้ตราประทับไม่จำกัดจำนวนครั้ง ถ้าไม่เช่นนั้นจะต้องคืนค่า 0
- เมื่อใดก็ตามที่ฟังก์ชันนี้คืนค่า 1 โปรแกรมของคุณจะจบการทำงาน

ตัวอย่าง

ตัวอย่าง 1

พิจารณาการเรียกดังนี้:

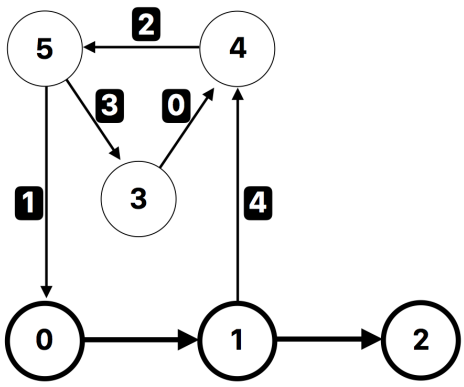
```
init(6, 3)
```

ในตัวอย่างนี้ มีดาวเคราะห์ 6 ดวง และมี 3 ดวงเป็นดาวเคราะห์พิเศษ ดาวเคราะห์ 0, 1, และ 2 เป็นดาวเคราะห์พิเศษ อุปกรณ์เคลื่อนย้ายมวลสารเริ่มต้นคือ $(0, 1)$ และ $(1, 2)$.

สมมติว่าเกรดเดอร์เรียกดังนี้:

- (0) `add_teleporter(3, 4)`: จะต้องคืนค่า 0.
- (1) `add_teleporter(5, 0)`: จะต้องคืนค่า 0.
- (2) `add_teleporter(4, 5)`: จะต้องคืนค่า 0.
- (3) `add_teleporter(5, 3)`: จะต้องคืนค่า 0.
- (4) `add_teleporter(1, 4)`: เมื่อเพิ่มอุปกรณ์นี้เข้าไป นักท่องเที่ยวจะสามารถได้รับการประทับตราเป็นจำนวนไม่จำกัดครั้ง ยกตัวอย่างเช่น นักท่องเที่ยวอาจจะเริ่มที่ดาวเคราะห์ 0 ไปยังดาวเคราะห์ $1, 4, 5, 0, 1, 4, 5, 0, \ldots$ ตามลำดับ ดังนั้นคุณจะต้องคืนค่า 1 และโปรแกรมของคุณจะถูกจบการทำงาน

รูปด้านล่างแสดงตัวอย่างนี้ ดาวเคราะห์พิเศษและอุปกรณ์เคลื่อนย้ายมวลสารเริ่มต้นแสดงด้วยเส้นหนา อุปกรณ์เคลื่อนย้ายมวลสารที่เพิ่มโดยฟังก์ชัน `add_teleporter` มีป้ายแสดงหมายเลข 0 ถึง 4 ตามลำดับ



ตัวอย่าง 2

พิจารณาการเรียกต่อไปนี้:

```
init(4, 2)
```

ในตัวอย่างนี้ มีดาวเคราะห์ 4 ดวงและมีดาวเคราะห์พิเศษ 2 ดวง ดาวเคราะห์ 0 และ 1 เป็นดาวเคราะห์พิเศษ อุปกรณ์เคลื่อนย้ายมวลสารเริ่มต้นคือ $(0, 1)$

สมมติว่าเกรดเดอร์เรียก:

- `add_teleporter(1, 1)`: หลังจากการเพิ่มอุปกรณ์เคลื่อนย้ายมวลสาร $(1, 1)$ แล้ว นักท่องเที่ยวสามารถได้รับการประทับตราจำนวนครั้งไม่จำกัด ยกตัวอย่างเช่น นักท่องเที่ยวอาจจะเริ่มที่ดาวเคราะห์ 1 และเดินทางไปยังดาวเคราะห์ 1 เป็นจำนวนไม่จำกัดครั้งโดยใช้อุปกรณ์เคลื่อนย้ายมวลสาร $(1, 1)$ ดังนั้น คุณจะต้องคืนค่า 1 และโปรแกรมของคุณจะถูกจบการทำงาน

ตัวอย่างข้อมูลนำเข้าและส่งออกมีใน attachment package.

เงื่อนไข

- $1 \leq n \leq 300\,000$
- $1 \leq m \leq 500\,000$
- $1 \leq k \leq n$

ในการเรียกฟังก์ชัน `add_teleporter` แต่ละครั้ง:

- $0 \leq u \leq n - 1$ และ $0 \leq v \leq n - 1$
- จะไม่มีอุปกรณ์เคลื่อนย้ายมวลสารจากดาวเคราะห์ u ไปยังดาวเคราะห์ v ก่อนที่จะมีการเพิ่มอุปกรณ์เคลื่อนย้ายมวลสาร (u, v)

ปัญหาย่อย

1. (2 points) $n = k, n \leq 100, m \leq 300$
2. (10 points) $n \leq 100, m \leq 300$
3. (18 points) $n \leq 1\,000, m \leq 5\,000$
4. (30 points) $n \leq 30\,000, m \leq 50\,000, k \leq 1\,000$
5. (40 points) ไม่มีเงื่อนไขเพิ่มเติมอื่น ๆ

เกรดเดอร์ตัวอย่าง

เกรดเดอร์ตัวอย่างอ่านข้อมูลนำเข้าในรูปแบบต่อไปนี้:

- บรรทัดที่ 1: $n\ m\ k$
- บรรทัดที่ $2 + i$ ($0 \leq i \leq m - 1$): $u[i]\ v[i]$

เกรดเดอร์ตัวอย่างจะเริ่มโดยเรียก `init` และเรียก `add_teleporter` โดยให้ $u = u[i]$ และ $v = v[i]$ สำหรับค่า $i = 0, 1, \ldots, m-1$ ตามลำดับ

เกรดเดอร์จะพิมพ์หมายเลขของการเรียก `add_teleporter` ครั้งแรกที่คืนค่า 1 (ซึ่งจะมีค่าระหว่าง 0 และ $m-1$ รวม 0 และ $m-1$ ด้วย) หรือ m ถ้าทุก ๆ การเรียก `add_teleporter` คืนค่า 0.

ถ้าในบางการเรียกใช้ ฟังก์ชัน `add_teleporter` คืนจำนวนเต็มอื่นนอกจาก 0 หรือ 1 เกรดเดอร์ตัวอย่างจะพิมพ์ -1 และโปรแกรมของคุณจะจบการทำงานทันที