משחק

לאחר שגילו n פלנטות, הממוספרות מ-0 עד $n-1$, הפרעונים התחילו לבנות מערכת תחבורה של **משגרים חד סטריים** ביניהם. לכל משגר פלנטת התחלה ופלנטת סוף. כשתייר משתמש במשגר בפלנטת ההתחלה, התייר משתגר לפלנטת הסוף. שימו לב שפלנטת ההתחלה והסוף של משגר יכולות להיות זהות. משגר עם פלנטת התחלה u ופלנטת סוף v מסומן על ידי (u, v).

כדי לעודד שימוש נרחב במערכת המשגרים, הפרעונים יצרו משחק שהתיירים יכולים לשחק בו בזמן שהם מטיילים עם מערכת המשגרים. תייר יכול להתחיל את המשחק בכל פלנטה. הפלנטות $0, 1, \ldots, k-1$ $(k \leq n)$ נקראות **פלנטות מיוחדות**. בכל פעם שבה תייר נכנס לפלנטה מיוחדת, התייר מקבל חותמת.

כרגע, לכל i $(0 \leq i \leq k-2)$, קיים משגר $(i, i+1)$. $k-1$ המשגרים הללו נקראים **משגרים התחלתיים**.

משגרים חדשים מתווספים אחד אחד. כשמשגרים חדשים מתווספים, זה יכול להפך לאפשרי עבור תייר לקבל מספר אינסופי של חותמות. ליתר דיוק, זה קורה כאשר קיימת סדרה של פלנטות $w[0], w[1], \ldots, w[t]$ המקיימת את התנאים הבאים:

- $1 \leq t$
- $0 \leq w[0] \leq k-1$
- $w[t] = w[0]$
- לכל i $(0 \leq i \leq t-1)$, קיים משגר $(w[i], w[i+1])$.

שימו לב שתייר יכול להשתמש במשגרים ההתחלתיים **ובכל** משגר שהתווסף עד כה.

המשימה שלכם היא לעזור לפרעונים לבדוק, לאחר ההוספה של כל משגר, האם תייר יכול לקבל מספר אינסופי של חותמות או לא.

פרטי מימוש

עליכם לממש את הפונקציות הבאות:

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

- n: מספר הפלנטות.
- k: מספר הפלנטות המיוחדות.
- פונקציה זו נקראת בדיוק פעם אחת, לפני כל הקריאות ל-`add_teleporter`.

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

- u ו-v: פלנטת ההתחלה והסוף של המשגר שנוסף.
- פונקציה זו נקראת לכל היותר m פעמים (לערך של m, ראו את המגבלות).
- על פונקציה זו להחזיר 1 אם, לאחר שהמשגר (u, v) נוסף, התייר יכול לקבל מספר אינסופי של חותמות. אחרת, על פונקציה זו להחזיר 0.
- ברגע שפונקציה זו תחזיר 1, ריצת התוכנית שלכם תסתיים.

דוגמאות

דוגמה 1

הביטו בקריאה הבאה:

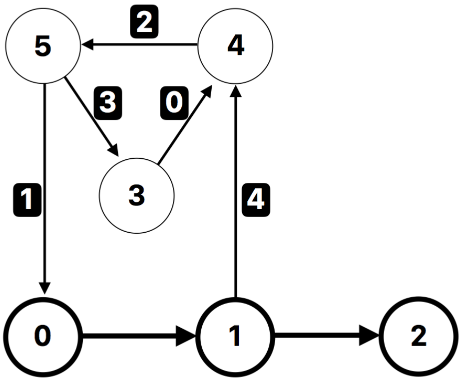
```
init(6, 3)
```

בדוגמה זו, ישנן 6 פלנטות ו-3 פלנטות מיוחדות. הפלנטות 0, 1, ו-2 הן פלנטות מיוחדות. המשגרים ההתחלתיים הם $(0, 1)$ ו-$(1, 2)$.

הניחו שהגריידר מבצע את הקריאות:

- (0) `add_teleporter(3, 4)`: עליכם להחזיר 0.
- (1) `add_teleporter(5, 0)`: עליכם להחזיר 0.
- (2) `add_teleporter(4, 5)`: עליכם להחזיר 0.
- (3) `add_teleporter(5, 3)`: עליכם להחזיר 0.
- (4) `add_teleporter(1, 4)`: בנקודה זו ניתן לקבל מספר אינסופי של חותמות. למשל, התייר מתחיל בפלנטה 0, ועובר לפלנטות $1, 4, 5, 0, 1, 4, 5, 0, \ldots$ בסדר זה. לכן, עליכם להחזיר 1, וריצת התוכנית שלכם תסתיים.

האיור הבא ממחיש דוגמה זו. הפלנטות המיוחדות והמשגרים ההתחלתיים מודגשים. המשגרים שנוספו על ידי `add_teleporter` מסומנים מ-0 עד 4, לפי הסדר.



דוגמה 2

הביטו בקריאה הבאה:

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

בדוגמה זו, ישנן 4 פלנטות ו-2 פלנטות מיוחדות. הפלנטות 0 ו-1 הן פלנטות מיוחדות. המשגר ההתחלתי הוא $(0, 1)$.

הניחו שהגריידר מבצע את הקריאה:

- `add_teleporter(1, 1)`: לאחר הוספת המשגר $(1, 1)$, ניתן לקבל מספר אינסופי של חותמות. למשל, התייר מתחיל בפלנטה 1, ונכנס לפלנטה 1 מספר פעמים אינסופי באמצעות המשגר $(1, 1)$. לכן, עליכם להחזיר 1, וריצת התוכנית שלכם תסתיים.

דוגמת קלט/פלט נוספת זמינה גם בחבילת הקבצים המצורפים.

מגבלות

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

עבור כל קריאה לפונקציה `add_teleporter`:

- $0 \leq u \leq n - 1$ וגם $0 \leq v \leq n - 1$
- לא קיים משגר מהפלנטה u לפלנטה v לפני הוספת המשגר (u, v).

תת משימות

1. (2 נקודות) $n = k$, $n \leq 100$, $m \leq 300$
2. (10 נקודות) $n \leq 100$, $m \leq 300$
3. (18 נקודות) $n \leq 1\,000$, $m \leq 5\,000$
4. (30 נקודות) $n \leq 30\,000$, $m \leq 50\,000$, $k \leq 1\,000$
5. (40 נקודות) ללא מגבלות נוספות.

גריידר לדוגמה

הגריידר לדוגמה קורא את הקלט בפורמט הבא:

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

הגריידר לדוגמה תחילה קורא ל-`init`, ואז ל-`add_teleporter` עם $u = u[i]$ ו-$v = v[i]$ עבור $i = 0, 1, \ldots, m - 1$ לפי הסדר.

הוא מדפיס את מספר הקריאה הראשונה ל-`add_teleporter` שמחזירה 1 (שבין 0 ל-$m - 1$, כולל), או m אם כל הקריאות ל-`add_teleporter` מחזירות 0.

אם קריאה כלשהי ל-`add_teleporter` מחזירה מספר שלם ששונה מ-0 או מ-1, הגריידר לדוגמה מדפיס -1 וריצת התוכנית שלכם תופסק באופן מיידי.