



## Werewolf

במחוז Ibaraki ביפן, יש  $N$  ערים ו- $M$  כבישים. הערים ממוספרות מ-0 עד  $N - 1$  בסדר עולה לפי כמות התושבים. כל כביש מחבר זוג ערים שונות וניתן לנסוע בו בשני הכיוונים. ניתן להגיע מכל עיר לכל עיר אחרת דרך כביש אחד או יותר.

תכנתם  $Q$  נסיעות, ממוספרות מ-0 עד  $Q - 1$ . הנסיעה ה- $i$  ( $0 \leq i \leq Q - 1$ ) מתחילה בעיר  $S_i$  ומסתיימת בעיר  $E_i$ .

אתם איש זאב. יש לכם שתי צורות: **צורת אדם וצורת זאב**. אתם מתחילים כל נסיעה בצורת אדם. בסוף כל נסיעה אתם חייבים להיות בצורת זאב. במהלך הנסיעה עליכם **לשנות צורה** (מצורת אדם לצורת זאב) בדיוק פעם אחת. ניתן לשנות צורה רק כאשר אתם נמצאים בעיר כלשהי (אפשר גם ב- $S_i$  או ב- $E_i$ ).

לא קל להיות איש זאב. עליכם להימנע מערים קטנות כאשר אתם בצורת אדם ולהימנע מערים גדולות כאשר אתם בצורת זאב. לכל נסיעה  $i$  ( $0 \leq i \leq Q - 1$ ), נתונים שני חסמים:  $L_i$  ו- $R_i$  ( $0 \leq L_i \leq R_i \leq N - 1$ ) שמגדירים מאילו ערים יש להימנע. ליתר דיוק, אסור לכם להיות בערים  $0, 1, \dots, L_i - 1$ , כאשר אתם בצורת אדם, ואסור לכם להיות בערים  $R_i + 1, R_i + 2, \dots, N - 1$  כאשר אתם בצורת זאב. בפרט זה אומר שבנסיעה  $i$ , תוכלו לשנות צורה רק באחת מהערים  $L_i, L_i + 1, \dots, R_i$ .

עבור כל נסיעה, מטרתכם היא לקבוע האם ניתן להגיע מהעיר  $S_i$  לעיר  $E_i$  תחת המגבלות שצוינו. אין מגבלה על אורך הנסיעה.

### פרטי מימוש

עליכם לממש את הפונקציה הבאה:

```
int[] check_validity(int N, int[] X, int[] Y, int[] S, int[] E,  
int[] L, int[] R)
```

- $N$ : מספר הערים.
- $X$  ו- $Y$ : מערכים באורך  $M$ . לכל  $j$  ( $0 \leq j \leq M - 1$ ), העיר  $X[j]$  מחוברת ישירות לעיר  $Y[j]$  בכביש.
- $S, E, L, R$ : מערכים באורך  $Q$ , שמייצגים את הנסיעות.

שימו לב שהערכים  $M$  ו- $Q$  הם אורכי המערכים וניתן לקבלם כפי שמתואר ב-implementation notice.

הפונקציה `check_validity` נקראת בדיוק פעם אחת בכל `test case`. הפונקציה צריכה להחזיר מערך  $A$  של מספרים שלמים באורך  $Q$ . הערך  $A_i$  ( $0 \leq i \leq Q - 1$ ) צריך להיות 1 אם הנסיעה  $i$  אפשרית לפי התנאים שתוארו לעיל, או 0 אחרת.

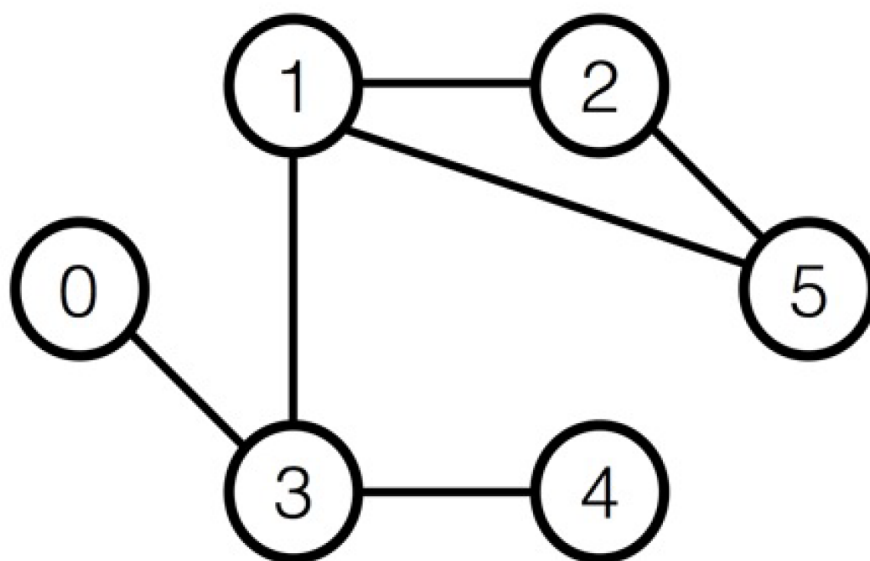
## דוגמה

נתונים:

$N = 6, M = 6, Q = 3, X = [5, 1, 1, 3, 3, 5], Y = [1, 2, 3, 4, 0, 2], S = [4, 4, 5],$   
 $E = [2, 2, 4], L = [1, 2, 3], R = [2, 2, 4]$

הגרייידר קורא ל-

```
Check_validity(6, [5, 1, 1, 3, 3, 5], [1, 2, 3, 4, 0, 2],  
[4, 4, 5], [2, 2, 4], [1, 2, 3], [2, 2, 4])
```



את נסיעה מספר 0, מהעיר 4 לעיר 2 ניתן לבצע באופן הבא:

- מתחילים בעיר 4 (בצורת אדם).
- מתקדמים לעיר 3 (בצורת אדם).
- מתקדמים לעיר 1 (בצורת אדם).
- משנים צורה לזאב (כעת אתם בצורת זאב).
- מתקדמים לעיר 2 (בצורת זאב).

את נסיעות מספר 1 ו-2, לא ניתן לבצע.

לכן, התוכנית צריכה להחזיר את המערך  $[1, 0, 0]$ .

הקבצים sample-01-in.txt ו-sample-01-out.txt בקובץ ה-`zip` המצורף, מתאימים לדוגמה זו. קובץ ה-`zip` מכיל זוג נוסף של קובץ קלט וקובץ פלט.

## מגבלות

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$
- $1 \leq Q \leq 200\,000$
- לכל  $0 \leq j \leq M - 1$
- $0 \leq X_j \leq N - 1$
- $0 \leq Y_j \leq N - 1$
- $X_j \neq Y_j$
- ניתן להגיע מכל עיר לכל עיר אחרת דרך הכבישים.
- כל זוג ערים מחובר בכביש אחד לכל היותר.
- במילים אחרות, לכל  $0 \leq j < k \leq M - 1$  וגם  $(X_j, Y_j) \neq (X_k, Y_k)$
- לכל  $0 \leq i \leq Q - 1$
- $0 \leq L_i \leq S_i \leq N - 1$
- $0 \leq E_i \leq R_i \leq N - 1$
- $S_i \neq E_i$
- $L_i \leq R_i$

## תת משימות

1. (7 נקודות):  $N \leq 100, M \leq 200, Q \leq 100$
2. (8 נקודות):  $N \leq 3\,000, M \leq 6\,000, Q \leq 3\,000$
3. (34 נקודות):  $M = N - 1$  וכל עיר מחוברת לשני כבישים לכל היותר (הערים מחוברות במסילה)
4. (51 נקודות): ללא מגבלות נוספות.

## גריידר לדוגמה (Sample grader)

הגריידר לדוגמה קורא קלט בפורמט הבא:

- שורה ראשונה:  $N$  ואחריו  $M$  ואחריו  $Q$ .
- שורה מספר  $2 + j$  ( $0 \leq j \leq M - 1$ ):  $X_j$  ואחריו  $Y_j$
- שורה מספר  $2 + M + i$  ( $0 \leq i \leq Q - 1$ ):  $S_i$ , אחריו  $E_i$ , אחריו  $L_i$  ואחריו  $R_i$ .

הגריידר לדוגמה מדפיס את הערך המוחזר מ `check_validity` בפורמט הבא:

- שורה מספר  $1 + i$  ( $0 \leq i \leq Q - 1$ ):  $A_i$