הבעיה ריצה מהירה (speedrun)

קובץ header speedrun.h

מרסל (Marcel) התחיל לעשות ריצות מהירות במשחק שבו הוא משחק, בתקווה לקבוע שיא עולמי חדש. לצערו, הוא לא יכול לשלוט בכל האסטרטגיות החדשות, ולכן הוא מנסה להשיג יתרון לא הוגן.

המשחק שהוא מנסה להיות טוב בו נקרא *נשמות העצים 3*, והוא מנסה לקבוע שיא עולמי בקטגוריית *העץ המלא*.

המשחק מתרחש, כמו שהשם רומז, על עץ (גרף עם N קודקודים, ממוספרים מ-1 עד N, ו $N-1$ קשתות, כך שאין מעגלים).

המשחק עובד באופן הבא: הדמות ממוקמת על צומת שרירותי בעץ. מרסל יכול לבחור מספר שלם x, ולנסות ללכת מהצומת שהוא נמצא בו כרגע לצומת x. אם יש קשת בין x לבין הצומת שהוא נמצא בו כרגע, הוא יעבור ל-x, ואחרת, שום דבר לא יקרה. הריצה שלו תקינה אם הוא מבקר בכל צומת לפחות פעם אחת.

עכשיו מגיע החלק שבו הוא מרמה: מאחר והוא לא מכיר את גליטץ'-השתגרות-הקשת, הוא הולך לקבוע את הseed לעולם שלו, ולכן הוא ידע איך בדיוק העץ שלו נראה לפני שהוא מתחיל. אם הוא פשוט יזכור בעל פה את העץ, זה יהיה ברור שהוא מרמה, והוא יורחק לאלתר מקהילת הרצים המהירים של המשחק. לכן, הוא ישים לעצמו רמז בכל צומת (ע"י התעסקות עם קוד המשחק). הרמזים הם מחרוזות בינאריות באורך l (l הוא זהה לכל הרמזים בכל הצמתים). כשהוא יושב על צומת, הוא יכול לקרוא את הרמז ששייך לאותו צומת.

פרוטוקול התקשורת

זוהי שאלה אינטראקטיבית דו-שלבית!
הקוד שלכם ירוץ פעמיים, פעם אחת על האינטראקציה הראשונה (שלב קביעת הרמזים), ופעם שנייה על האינטראקציה השנייה (שלב המשחק).

אתם צריכים לממש את הפונקציות הבאות (ולא את פונקציית main):

```
void assignHints(int subtask, int N, int A[], int B[]);
void speedrun(int subtask, int N, int start);
```

בעזרת שימוש בפונקציות הבאות:

```
void setHintLen(int l);
void setHint(int i, int j, bool b);
int getLength();
bool getHint(int j);
bool goTo(int x);
```

שלב קביעת הרמזים

זוהי ההרצה הראשונה של הקוד שלכם. בשלב זה, קוד הבדיקה יקרא ל $assignHints$ בדיוק פעם אחת. הוא יעביר את ה-subtask, שזה מספר ה-subtask הנוכחי, ו-N, בתור פרמטרים, והוא יעביר את הקשתות של העץ באופן הבא: לכל $i = 1, \dots, N-1$, יש קשת בין $A[i]$ ל-$B[i]$. עליכם לקרוא לפונקציה $setHintLen$ בדיוק פעם אחת, ולהעביר את אורך הרמז שקבעתם כפרמטר. לאחר שקראתם ל-$setHintLen$, אתם יכולים לקרוא לפונקציה $setHint(i,j,b)$ כמה פעמים שתרצו. פונקציה זו קובעת את הביט ה-j של הרמז לצומת i להיות b. הרמזים מאותחלים כמחרוזות של אפסים. בשלב זה אסור לכם לקרוא לפונקציות $getLength, getHint$ או $goTo$. הקוד שלכם צריך לצאת מ$assignHints$ לאחר שסיימתם לקבוע את הרמזים.

שלב המשחק

זוהי ההרצה השנייה של הקוד שלכם. בשלב זה, קוד הבדיקה יקרא ל-$speedrun(subtask, N, start)$ בדיוק פעם אחת, ויעביר את $start$ – הצומת ההתחלתי, ואת N – מספר הקודקודים. בנוסף, תקבלו גם כפרמטר את מספר ה-$subtask$ הנוכחי. לאחר מכן, תוכלו לקרוא לפונקציות: $getLength()$, שתחזיר את l, אורך הרמז שנקבע ע"י התכנית שלכם בשלב הראשון, הפונקציה $getHint(j)$, שתחזיר את הביט ה j של הרמז של הצומת שאתם נמצאים בו כרגע, ופונקציית ה-$goTo$. אם הפרמטר שהועבר ל-$goTo$ הוא מספר של צומת שהוא שכן של הצומת הנוכחי, הפונקציה תחזיר $true$ ואתם תעברו לצומת הזה. אחרת, הפונקציה תחזיר $false$ ואתם תישארו בצומת הנוכחי. בשלב זה אסור לכם לקרוא לפונקציות $setHint$ או $setHintLen$. הקוד שלכם צריך לצאת מפונקציית $speedrun$ לאחר שביקרתם בכל הצמתים בעץ.

מגבלות

- $1 \le N \le 1\,000$
- יהי Q מספר הקריאות ל$goTo$ שהחזירו $false$. בהמשך נקבע את המגבלות ל-l, Q הנדרשות כדי לקבל ניקוד בכל תת משימה.

תת משימה 1 (21 נקודות)

- $l \le N$
- $Q \le 2000$
- הניקוד לתת משימה זו יהיה 21 אם כל הטסטים בו נפתרו, ו0 אחרת.

תת משימה 2 (8 נקודות)

- $l \le 20$
- $Q \le 2000$
- העץ הוא כוכב, כלומר, קיים צומת x ($1 \le x \le N$) שמחובר לכל צומת אחר.
- הניקוד לתת משימה זו יהיה 8 אם כל הטסטים בו נפתרו, ו0 אחרת.

תת משימה 3 (19 נקודות)

- $l \le 20$
- $Q \le 2000$
- הדרגה של כל צומת היא לכל היותר 2.
- הניקוד לתת משימה זו יהיה 19 אם כל הטסטים בו נפתרו, ו0 אחרת.

תת משימה 4 (12 נקודות)

- $l \le 316$
- $Q \le 32000$
- הניקוד לתת משימה זו יהיה 12 אם כל הטסטים בו נפתרו, ו0 אחרת.

תת משימה 5 (40 נקודות)

- $Q \le 2000$
- הניקוד s של כל טסט מחושב באופן הבא: אם $l > 40$, $s = 0$. אם $l \le 20$, $s = 40$. אחרת, $s = 60 - l$. כלומר, אם $l = 40$, תקבלו חצי מהניקוד עבור הטסט, ואם $l \le 20$, תקבלו ניקוד מלא על הטסט. הניקוד של תת המשימה יהיה הניקוד המינימלי מבין כל הטסטים בתת משימה זו.

דוגמה

בשלב הראשון, הפונקציה שלכם תיקרא כך:

```
assignHints(
  /* subtask   = */ 1,
  /* N         = */ 5,
  /* A         = */ {-, 1, 2, 3, 3},
  /* B         = */ {-, 2, 3, 4, 5});
```

הפונקציה שלכם, יכולה לבחור לקבוע $l = 2$, ע"י קריאה:

```
setHintLen( /* l  = */ 2);
```

ואז להשאיר את כל הביטים של הרמזים כ-0, למעט ביט 1 של צומת 2, שאותו יקבע ל-1:

```
setHint(
  /* i  = */ 2,
  /* j  = */ 1,
  /* b  = */ 1);
```

ואז הפונקציה יוצאת. ההרצה הראשונה של התכנית שלכם כעת מסתיימת, ולכן כל מידע ששמור בזיכרון ע"י תכנית זו יימחק. כעת, הרמזים הם "00" לכל צומת, למעט צומת 2, שעבורו הרמז הוא "10".

בהרצה השנייה (שלב המשחק), הפונקציה שלכם תיקרא כך:

```
speedrun(
  /* subtask = */ 1,
  /* N       = */ 5,
  /* start   = */ 1);
```

בתגובה, הפונקציה שלכם תתחיל לרוץ על העץ:

```
getLength();    // = 2
getHint(1);     // = 0
getHint(2);     // = 0
goTo(2);        // = true
getHint(1);     // = 1
getHint(2);     // = 0
```

בשלב זה, הייתם בהתחלה בצומת 1, גיליתם ש $l = 2$ ושהרמז שלו הוא "00", ולאחר מכן עברתם לצומת 2 בהצלחה וגיליתם שהרמז שלו הוא "10". הריצה יכולה להמשיך כך:

goTo(2);	*// = false (אתם לא יכולים לעבור מצומת 2 לעצמו)*
goTo(5);	*// = false (אין קשת מצומת 2 לצומת 5)*
goTo(3);	*// = true*
goTo(4);	*// = true*
goTo(3);	*// = true*
goTo(5);	*// = true*

בשלב זה ביקרתם בכל הצמתים, ולכן הפונקציה $speedrun$ יכולה לסיים את ריצתה. הערך של Q הוא 2, כי היו שתי קריאות ל$goTo$ שהחזירו $false$.