



Problem Speedrun

C++ header speedrun.h

Марсель почав займатися спідранінгом, сподіваючись встановити новий Ср (Світовий рекорд). На жаль, він не може оволодіти новими стратегіями, отже він повинен отримати нечесну перевагу.

Гра, у якій він намагається навчитися називається *Tree Souls III*, і він пробує встановити а Ср в категорії *повного дерева*%.

Гра відбувається, як слідує з назви, на дереві (тобто граф з N вершин, понумерованих від 1 до N , і $N - 1$ ребер, без циклів).

Гра працює так: персонаж розташований в довільній вершині в дереві. Він може вибрати x , і спробувати піти з вершини, у якій він знаходиться зараз до вершини x . Якщо існує ребро між вершиною x і вершиною, у якій він знаходиться, він піде до x , інакше нічого не відбудеться. Його спідран є дійсним, якщо він відвідає кожну вершину хоча б раз.

Тепер ми підійшли до частини з обманом: оскільки він не знає глюк-з-телепортацією-ребер, він збирається встановити сід для його світу, таким чином він знатиме, як виглядає дерево перед початком. Якщо він збереже дерево, це буде дуже очевидно, що він обманює і його виключать зі змагання, отже він залишить підказки у вершинах (підроблюючи код). Підказка, це двійкова строка довжини l (l такий самий для кожної підказки у кожній вершині). Коли він знаходиться у вершині, може прочитати підказку у цій вершині.

Interaction Protocol

Це двухфазова інтерактивна задача!

Ваш код запустять два рази, перший для першої інтеракції (*фаза встановки підказок*), і один раз для другої ітерації (*the фаза спідранінгу*).

Ви маєте реалізувати наступні функції (не main).

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

Використовуючи наступні функції виклику:

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

Фаза встановки підказок. Це перший запуск вашого коду. В цій фазі, код журі запутить функцію `assignHints` рівно один раз. Воно видасть йому *subtask*, яка є індексом підзадачі, і N яко параметри, і передасть ребра дерева в A і B в наступний спосіб: для кожного $i = 1, \dots, N-1$, буде ребро між $A[i]$ і $B[i]$. Ви маєте викликати функцію `setHintLen` рівно один раз, передаючи їх довжину строк, які ви вирішили використовувати. Після виклику `setHintLen`, ви можете викликати `setHint(i, j, b)` кілька разів. Вона встановлює біт j підказки для вершини i на b . Біти індексовані від 1 до l . Підказки на початку заповнені нулями. В цій фазі ви не маєте викликати функції `getLength`, `getHint` чи `goTo`. Ваш код має вийти з `assignHints` коли ви закінчили присвоювати підказки.



Фаза спідранінгу. Це другий запуск вашого коду. В цій фазі, код журі викличе функцію `speedrun(subtask, N, start)` рівно один раз, передаючи їй вершину `start` в якій герой починає, і вартість N . Також ви можете отримати індекс підзадачі `subtask`. Далі ви можете викликати функцію `getLength()` котра повертає l , довжина підказок, встановлена вашою програмою у першій фазі, функцію `getHint(j)` котра вам повертає біт j у вершині, в якій персонаж зараз є, і функцію `goTo`. Якщо параметр, переданий `goTo` це індекс вершини з ребром від поточної вершини, функція повертає `true` і ви рухаєтесь до цієї вершини. Інакше, вона повертає `false` і ви залишаєтесь у вершині. В цій фазі ви не маєте викликати функції `setHint` або `setHintLen`. Ваш код має вийти з функції `speedrun` після того, як ви відвідали всі вершини дерева.

Restrictions

- $1 \leq N \leq 1\,000$
- Нехай Q буде числом викликів `goTo` які повернули `false`. Далі ми окреслюємо обмеження для l і Q , які мають виконуватися, щоб отримати бали за підзадачу.

Subtask 1 (21 points)

- $l \leq N$
- $Q \leq 2000$
- Пункти за підзадачу рівняються 21, якщо усі тести пройдено, і 0 в іншому випадку.

Subtask 2 (8 points)

- $l \leq 20$
- $Q \leq 2000$
- Дерево має вигляд зірки; тобто існує вершина x ($1 \leq x \leq N$), яка з'єднана з усіма іншими вершинами.
- Пункти за підзадачу рівняються 8, якщо усі тести пройдено, і 0 в іншому випадку.

Subtask 3 (19 points)

- $l \leq 20$
- $Q \leq 2000$
- Степінь кожної вершини не більше 2.
- Пункти за підзадачу рівняються 19, якщо усі тести пройдено, і 0 в іншому випадку.

Subtask 4 (12 points)

- $l \leq 316$
- $Q \leq 32000$
- Пункти за підзадачу рівняються 12, якщо усі тести пройдено, і 0 в іншому випадку.

Subtask 5 (40 points)

- $Q \leq 2000$
- Результат s кожного тесту рахується так: якщо $l > 40$, то $s = 0$. Якщо $l \leq 20$, то $s = 40$. Інакше, $s = 60 - l$. Тобто, якщо $l = 40$, ви отримаєте половину балів за тест, а якщо $l \leq 20$, ви отримаєте повний бал за цей тест. Бал за підзадачу буде рахуватись як мінімум з усіх значень s з усіх тестів цієї підзадачі.

Interaction example

В першій фазі (Фаза встановки підказок), функція учасника буде викликатись так:



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

Ця функція може обрати, щоб встановити $l = 2$:

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

І тоді щоб залишити усі біти підказок рівними 0, окрім для біту 1 вершини 2, що встановлюється рівним 1:

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

Далі, він виходить. Першу роботу програми учасника буде припинено, отже, будь-які дані, що зберігаються в програмі буде видалено. На даний момент підказки "00" для кожної вершини, окрім вершини 2, що має підказку "10".

В другій фазі (Фаза спідранінгу), функція учасника буде викликана:

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

У відповідь, функція починає оброблювати дерево:

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

В цей момент, ви спочатку знаходитесь в вершині 1, дізнаєтесь, що $l = 2$ та підказку вершини 1 рівну "00", далі успішно переміщаєтесь в вершину 2 і дізнаєтесь підказку вершини 2 рівну "10". Далі може продовжуватись наприклад так:

```
goTo(2);      // = false (you can not go from node 2 back to itself)  
goTo(5);      // = false (there is no edge from node 2 to node 5)  
goTo(3);      // = true  
goTo(4);      // = true  
goTo(3);      // = true  
goTo(5);      // = true
```

В цей момент усі вершини будуть відвідані, тож speedrun функція може завершуватись. Значення $Q = 2$, тому що 2 виклики goTo повернули false.