



Problem Speedrun

C++ header `speedrun.h`

Марсель занялся спидраном (проходом игры на скорость), в надежде побить мировой рекорд WR (World Record). К сожалению, он пока не достиг совершенства, поэтому он хочет найти способ получить несправедливое преимущество.

Игра, в которой он пытается преуспеть, называется *Души деревьев III*, и он пытается побить WR в категории *полное-дерево*.

Игра происходит, как следует из названия, на дереве (то есть на графе с N вершинами, пронумерованными от 1 до N , и $N - 1$ ребрами, причем в графе нет циклов).

Игра работает следующим образом: персонаж помещается в произвольную вершину дерева. Он может выбрать целое число x и попытаться переместиться из вершины, в которой он находится, в вершину x . Если есть ребро между x и вершиной, в которой он находится, он перемещается в x , иначе ничего не происходит. Спидран считается корректным, если персонаж посетил каждую вершину хотя бы один раз.

Теперь начинается мошенническая часть: Марсель собирается взломать генератор случайных карт, и таким образом узнать, как выглядит дерево, прежде чем начнет спидран. Но если он просто запомнит дерево, то будет слишком очевидно, что он жульничает, что приведет к немедленному бану в сообществе, поэтому он просто поместит по подсказке в каждую вершину. Подсказка — это двоичная строка размера l (каждая вершина содержит подсказку одной и той же длины l). Когда он находится в вершине, он может прочитать подсказку, находящуюся в этой вершине.

Interaction Protocol

Это интерактивная задача с двойным запуском!

Ваш код будет запущен дважды, один раз для первого интерактивного взаимодействия с грейдером (*фаза формирования подсказок*), и один раз для второго интерактивного взаимодействия с грейдером (*фаза спидрана*).

Вам следует реализовать следующие функции (но не следует реализовывать функцию `main`).

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

При этом вы можете вызывать следующие функции:

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

Фаза формирования подсказок. Происходит первый запуск вашего кода, на этой фазе грейдер жюри вызовет функцию `assignHints` ровно один раз. Он передаст в качестве параметра `subtask` номер текущей подзадачи, а также значение N и описание дерева в массивах A и B следующим образом: для каждого $i = 1, \dots, N - 1$, в дереве присутствует ребро между $A[i]$ и $B[i]$ (обратите внимание, $A[0]$ и $B[0]$ не используются и содержат мусор). Вы должны ровно один раз вызвать функцию `setHintLen`, передав ей длину подсказки, которую вы решили сформировать, в качестве параметра. После вызова `setHintLen` вы можете вызвать функцию `setHint(i, j, b)` несколько раз. Такой вызов устанавливает бит j подсказки для вершины i в b . Биты нумеруются от 1 до l . Подсказки исходно заполнены нулями. На этой фазе не разрешается делать вызовы `getLength`, `getHint` или `goTo`. Ваш код должен выйти из функции `assignHints` после того, как установка подсказок завершена.



Фаза спидрана. Происходит второй запуск вашего кода. На этой фазе грейдер жюри вызовет функцию `speedrun(subtask, N, start)` ровно один раз. Он передаст номер вершины, где начинается спидран, `start`, а также значение N . Дополнительно вы получите номер подзадачи в параметре `subtask`. Вы можете вызвать функцию `getLength()`, которая вернет l , длину подсказок, установленных вашей программой на первой фазе, функцию `getHint(j)`, которая возвращает вам бит j в подсказке в текущей вершине, а также функцию `goTo`. Если параметр, переданный функции `goTo`, является номером вершины, которая соединена ребром с текущей вершиной, функция возвращает `true`, и вы перемещаетесь в эту вершину. Иначе она возвращает `false`, и вы остаетесь в текущей вершине. На этой фазе запрещается вызывать функции `setHint` и `setHintLen`. Ваш код должен в итоге выйти из функции `speedrun`, после того как вы посетили все вершины дерева.

Restrictions

- $1 \leq N \leq 1000$
- Пусть Q равен числу вызовов `goTo`, которые вернули значение `false`. В описании подзадач указаны ограничения для l и Q , которые должны выполняться, чтобы получить баллы за каждую из подзадач.

Подзадача 1 (21 баллов)

- $l \leq N$
- $Q \leq 2000$
- Баллы за подзадачу равны 21, если все тесты решены корректно, иначе вы получаете 0.

Подзадача 2 (8 баллов)

- $l \leq 20$
- $Q \leq 2000$
- Дерево имеет форму звезды, то есть одна из вершин x ($1 \leq x \leq N$) соединена ребром с любой другой вершиной.
- Баллы за подзадачу равны 8, если все тесты решены корректно, иначе вы получаете 0.

Подзадача 3 (19 баллов)

- $l \leq 20$
- $Q \leq 2000$
- Степень каждой вершины не больше 2.
- Баллы за подзадачу равны 19, если все тесты решены корректно, иначе вы получаете 0.

Подзадача 4 (12 баллов)

- $l \leq 316$
- $Q \leq 32000$
- Баллы за подзадачу равны 12, если все тесты решены корректно, иначе вы получаете 0.

Подзадача 5 (до 40 баллов)

- $Q \leq 2000$
- Баллы за тест s вычисляются следующим образом. Если $l > 40$, то $s = 0$. Если $l \leq 20$, то $s = 40$. Иначе, $s = 60 - l$. А именно, если $l = 40$, вы получите половину баллов за тест, а если $l \leq 20$, вы получите полный балл за тест. Баллы за подзадачу равны минимальному значению s по всем тестам этой подзадачи.



Пример взаимодействия

Первый запуск — фаза установки подсказок. Функция в решении участника вызвана со следующими параметрами:

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

Решение участника могло, например, решить установить значение $l = 2$:

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

И затем оставить все биты подсказки равными 0, кроме бита 1 в вершине 2, который устанавливается в 1:

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

Затем решение участника выходит из функции. На этом первый запуск закончен, программа завершается, вся информация в памяти уничтожается. Подсказки во всех вершинах дерева — "00", кроме вершины 2, в которой подсказка "10".

Второй запуск — фаза спидрана. Функция в решении участника вызвана со следующими параметрами:

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

Функция начинает выполнять спидран по дереву:

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

В этом примере вы исходно были в вершине 1, выяснили, что $l = 2$, и что подсказка в вершине 1 равна "00", затем перешли в вершину 2 и выяснили, что подсказка в вершине 2 равна "10". Выполнение может продолжиться следующим образом:

```
goTo(2);      // = false (нет ребра из 2 в себя)  
goTo(5);      // = false (нет ребра из 2 в 5)  
goTo(3);      // = true  
goTo(4);      // = true  
goTo(3);      // = true  
goTo(5);      // = true
```

Теперь все вершины дерева посещены и `speedrun` может выйти. Значение $Q = 2$, поскольку 2 вызова функции `goTo` вернули `false`.