SocialEngineering

Problem Name	Social Engineering
Input file	Interactive task
Output file	Interactive task
Time limit	5 seconds
Memory limit	256 megabytes

Социальная сеть представляет собой неориентированный связный граф с n вершинами и m ребрами, каждая вершина соответствует участнику, два участника сети друзья, если между ними есть ребро.

Мария — участница этой социальной сети. Она любит устраивать челленджи для своих друзей. Это означает, что она выполняет какое-то задание, а затем вызывает одного из своих друзей на челлендж — предлагает ему сделать то же самое. Выполнивший задание участник затем вызывает на челлендж своего друга, и так далее. Задание, таким образом, начинает перемещаться по сети. Может случиться, что кому-то придется выполнить задание более одного раза, но для каждой неупорядоченной пары друзей вызов на челлендж может произойти только один раз. (После того, как участник A вызывает на челлендж участника B, ни A не может больше вызывать на челлендж B, ни B не может вызывать на челлендж A.) Другими словами, задание челленджа перемещается по пути в графе, который не проходит по одному ребру дважды.

Участник проигрывает в челлендже, если выполнив задание, он не может вызвать на челлендж ни одного из своих друзей. Мария всегда начинает челленджи и часто выигрывает. Теперь остальные n – 1 участников социальной сети решили объединить усилия и добиться, чтобы Мария проиграла в следующем челлендже. Ваша задача — скоординировать их действия.

Implementation

Вам необходимо реализовать функцию

`void SocialEngineering(int n, int m, vector<pair<int,int>> edges);` которая соответствует запуску челленджа на графе с n вершинами и m ребрами. Эта функция будет

вызвана грейдером ровно один раз. Вектор `edges` будет содержать ровно m пар целых чисел (u, v), такая пара задает ребро между вершинами u и v. Вершины пронумерованы от 1 до n. Мария соответствует вершине номер 1.

Эта функция может вызывать следующие функции:

```
int GetMove();
```
Эта функция должна вызываться каждый раз, когда Мария выполняет задание, в частности, в начале челленджа. Если вы вызовете эту функцию, когда задание выполняет не Мария, вы получите вердикт `Wrong Answer`. Функция возвращает одно из следующих значений:

- целое число v, где $2 \le v \le n$. Это означает, что Мария вызывает на челлендж участника номер v. Это всегда будет корректный вызов.
- 0, если Мария сдается, в частности, если она проиграла и не может вызывать никого на челлендж. После этого ваша программа должна выйти из функции `SocialEngineering` и получит вердикт `Accepted`.

```
void MakeMove(int v);
```
Эта функция должна быть вызвана, если сейчас не Мария выполняет задание. Вызов этой функции означает, что выполняющий сейчас задание вызывает на челлендж участника номер v. Если это некорректный вызов или сейчас задание должна выполнять Мария, то вы получите вердикт `Wrong Answer`.

Если у Марии на заданном графе есть выигрышная стратегия в начале челленджа, то ваша программа должна выйти из функции `SocialEngineering` *до* первого вызова `GetMove`. В этом случае вы получите вердикт `Accepted`.

Constraints

- $2 \le n \le 2 \cdot 10^5$.
- $1 \le m \le 4 \cdot 10^5$.
- Заданный граф связен. Каждая неупорядоченная пара вершин соединена не более чем одним ребром, каждое ребро соединяет две различные вершины.

Subtasks

Мария всегда будет играть оптимально в том смысле, что если она может выиграть, она будет это делать. Если у нее нет выигрышной стратегии, она будет пытаться заставить вашу

программу ошибиться разными способами. Она будет сдаваться только, если у нее нет возможности вызвать кого-нибудь на челлендж, за исключением подзадачи 3.

1. (15 баллов) $n, m \leq 10$.
2. (15 баллов) У всех, кроме Марии, не более двух друзей.
3. (20 баллов) Мария сдаётся во время первого вызова `GetMove` всегда, за исключением случая, когда у нее есть выигрышная стратегия.
4. (25 баллов) $n, m \leq 100$.
5. (25 баллов) Нет дополнительных ограничений.

Sample Interaction

Действия решения	Действия грейдера	Пояснение
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering` вызвана на графе с 5 вершинами и 6 ребрами.
`GetMove()`	`Возвращает 4`	Мария вызывает на челлендж участника с номером 4.
`MakeMove(2)`	-	Участник номер 4 вызывает на челлендж участника номер 2.
`MakeMove(5)`	-	Участник номер 2 вызывает на челлендж участника номер 5.
`MakeMove(1)`	-	Участник номер 5 вызывает на челлендж Марию.
`GetMove()`	`Возвращает 0`	Мария не может вызвать никого на челлендж, она сдается.
`Выход из функции`	-	Мария проиграла, необходимо выйти из функции `SocialEngineering`.

Contestant action	Grader action	Explanation
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` вызывается на графе с 2 вершинами и 1 ребром.

Contestant action	Grader action	Explanation
`Выход из функции`	-	У Марии есть выигрышная стратегия, поэтому необходимо выйти из функции.

Sample Grader

Пример грейдера `grader.cpp`, приведенный в файле `SocialEngineering.zip`, считывает входные данные со стандартного потока ввода в следующем формате:

- Первая строка содержит количество вершин *n* и число ребер *m* в графе.
- Следующие *m* строк содержат описание ребер, каждое ребро задается парой вершин, которые оно соединяет.

Пример грейдера считывает входные данные и вызывает функцию `SocialEngineering` в вашем решении. Обратите внимание, что пример грейдера не реализует оптимальную стратегию Марии, а просто демонстрирует, как устроено взаимодействие.
Чтобы скомпилировать пример грейдера с вашим решением, используйте следующую команду:

```
g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp
```

где `solution.cpp` это файл с решением, который вы планируете отправить на провеку. Для запуска программы с примером ввода, можно использовать команду: `./solution < input.txt`