Social Engineering

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

O rețea socială constă dintr-un graf neorientat conex cu n vârfuri și m muchii, unde fiecare vârf este o persoană și două persoane sunt prietene dacă există o muchie între ele.

Maria este membră a acestei rețele sociale. Ei îi place să își provoace prietenii să facă diferite lucruri. Aceasta înseamnă că ea efectuează mai întâi o sarcină simplă și apoi provoacă pe unul dintre prietenii săi să facă la fel. Acest prieten va provoca apoi pe unul dintre prietenii săi, care va provoca pe unul dintre prietenii săi și așa mai departe. Se poate întâmpla ca aceeași persoană să fie provocată de mai multe ori, dar fiecare pereche neordonată de prieteni va lua parte la provocare cel mult o dată. (Odată ce o persoană A provoacă o persoană B, nici persoana A nici persoana B nu o poate provoca din nou pe cealaltă.) Cu alte cuvinte, provocările vor fi o plimbare în graf care nu utilizează aceeași muchie de mai multe ori.

O persoană pierde provocarea dacă este rândul ei și ea nu mai poate provoca pe niciunul dintre prietenii săi. Provocările sunt întodeauna începute de Maria și ea pierde rareori. Acum celelalte $n - 1$ persoane au decis să colaboreze pentru a o face pe Maria să piardă următoarea provocare și este sarcina ta să coordonezi acest efort.

Implementare

Tu trebuie să implementezi o funcție:

```
void SocialEngineering(int n, int m, vector<pair<int,int>> edges);
```

care joacă acest joc pe un graf cu n vârfuri și m muchii. Această funcție va fi apelată o dată de către grader. Lista `edges` va conține exact m perechi de întregi (u, v), semnificând că există o muchie între vârful u și vârful v. Vârfurile sunt numerotate de la 1 la n. Maria este întotdeauna vârful 1. Funcția ta poate apela următoarele funcții:

```
int GetMove();
```

Această metodă trebuie să fie apelată ori de câte ori este rândul Mariei, cum ar fi la începutul jocului. Dacă tu apelezi această metodă când nu este rândul Mariei, vei primi verdictul `Wrong Answer`. Metoda poate returna una dintre următoarele valori:

- un întreg v, unde $2 \le v \le n$. Aceasta înseamnă că Maria provoacă persoana cu indicele v. Aceasta va fi întotdeauna o mutare legală.
- 0, dacă Maria renunță la joc. Maria va renunța întotdeauna dacă nu mai are mutări legale. Când acest lucru se întâmplă programul tău trebuie să lase funcția `SocialEngineering` să revină, și vei primi verdictul `Accepted`.

```
void MakeMove(int v);
```

Această metodă trebuie să fie apelată ori de câte ori nu este rândul Mariei. Aceasta înseamnă că persoana care este la rând provoacă persoana v. Dacă nu este o mutare legală sau dacă este rândul Mariei în momentul apelului atunci vei obține verdictul `Wrong Answer`.
Dacă Maria are o strategie de câștig la începutul jocului, programul tău trebuie să lase `SocialEngineering` să revină *înainte* de primul apel al funcției `GetMove()`. Atunci tu vei obține verdictul `Accepted`.

Restricții

- $2 \le n \le 2 \cdot 10^5$.
- $1 \le m \le 4 \cdot 10^5$.
- Graful este conex. Fiecare pereche neordonată de vârfuri va apărea cel mult o dată ca muchie și fiecare muchie va fi între două vârfuri diferite.

Subtasks

Maria va juca întotdeauna perfect, în sensul că ea va face mutări câștigătoare ori de câte ori ea are o strategie de câștig. Dacă ea nu are strategie de câștig, atunci ea va încerca să ademenească programul tău să facă o greșeală în diferite moduri istețe. Ea va renunța doar dacă nu mai are mutări legale, cu excepția Subtaskului 3.

1. (15 puncte) $n, m \le 10$.
2. (15 puncte) Toată lumea cu excepția Mariei are cel mult 2 prieteni.
3. (20 puncte) Maria va renunța imediat, dacă nu are strategie de câștig.
4. (25 puncte) $n, m \le 100$.

5. (25 puncte) Nu există restricții suplimentare.

Exemplu de interacțiune

Acțiunea ta	Acțiunea Grader-ului	Explicație
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering` este apelată cu un graf cu 5 vârfuri și 6 muchii.
`GetMove()`	`Returnează 4`	Maria provoacă persoana 4.
`MakeMove(2)`	-	Persoana 4 provoacă persoana 2.
`MakeMove(5)`	-	Persoana 2 provoacă persoana 5.
`MakeMove(1)`	-	Persoana 5 o provoacă pe Maria.
`GetMove()`	`Returnează 0`	Maria nu are mutări legale, deci ea renunță.
`Revine`	-	Tu ai câștigat jocul și trebuie să lași `SocialEngineering` să revină.

Acțiunea ta	Acțiunea Grader-ului	Explicație
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` este apelată cu un graf cu 2 vârfuri și 1 muchie.
`Revine`	-	Maria are strategie de câștig pe acest graf, deci tu trebuie să revii pentru a renunța fără a face niciun apel al `GetMove()`.

Grader exemplu

Grader-ul furnizat ca exemplu, `grader.cpp`, în atașamentul la problemă `SocialEngineering.zip`, citește intrarea de la intrarea standard în următorul format:

- Prima linie conține numărul de vârfuri *n* și numărul de muchii *m* din graf.
- Următoarele *m* linii conțin doi întregi *u* și *v*, indicând că există o muchie între *u* și *v*.

Grader-ul exemplu citește intrarea și apelează funcția `SocialEngineering` în soluția utilizatorului. Observați că grader-ul exemplu nu implementează strategia de câștig a Mariei și este furnizat doar pentru exemplificarea interacțiunii.
Pentru a compila grader-ul exemplu cu soluția ta, tu poți utiliza următoarea comandă la

prompterul terminalului:

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

unde `solution.cpp` este fișierul cu soluția ta care va fi trimis pe CMS. Pentru a executa programul cu input-ul exemplu furnizat în atașament tastați următoarea comandă la prompterul terminalului:

```
./solution < input.txt
```