SocialEngineering

Nome	Social Engineering
File di input	Problema interattivo
File di output	Problema interattivo
Limite di tempo	5 secondi
Limite di memoria	256 megaottetti

Un social network consiste di un grafo connesso non diretto con n vertici e m archi, dove ogni vertice è una persona. Se due persone sono amiche allora c'è un arco tra di loro.

Maria è un membro di questo social network e a lei piace sfidare i suoi amici. Questo vuol dire che prima fa un semplice task, e poi sfida uno dei suoi amici a fare lo stesso, che a sua volta sfiderà un suo amico, e così via. Può capitare che la stessa persona debba svolgere il compito più di una volta, ma ogni coppia non ordinata di amici può sfidarsi al più una volta (quando una persona A sfida una persona B, poi né A né B può sfidare l'altro di nuovo). In altre parole, la sfida è un cammino all'interno del grafo che non usa lo stesso arco più di una volta.

Una persona perde se è il suo turno e non può sfidare uno dei suoi amici. Le sfide sono sempre iniziate da Maria e lei perde molto di rado.

Adesso le n - 1 persone hanno deciso di collaborare per far perdere Maria, ed è il tuo compito coordinare questo compito.

Dettagli di implementazione

Devi implementare una funzione:

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

che gioca in un grafo di n vertici e m archi. Questa funzione verrà chiamata una volta dal grader. La lista *edges* contiene esattamente m coppie di interi (u, v), che indicano che c'è un arco tra il vertice u e il vertice v. I vertici sono numerati da 1 a n.

La tua funzione può fare chiamate alle seguenti funzioni:

```
int GetMove();
```

Questa funzione deve essere chiamata quando è il turno di Maria, come ad esempio all'inizio del gioco. Se chiami questa funzione quando non è il turno di Maria otterrai il verdetto `Wrong Answer`. Questa funzione può restituire uno dei seguenti valori:

- un intero v, con $2 \le v \le n$. Questo vuol dire che Maria sfida la persona con indice v. Questa è sempre una mossa valida.
- 0 se Maria si ritira. Maria si ritirerà sempre se non ha mosse possibili. Quando questo succede, devi uscire dalla funzione `SocialEngineering`, e otterrai il verdetto `Accepted`.

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

Questa funzione deve essere chiamata ogni volta che non è il turno di Maria. Vuol dire che la persona del turno corrente sfiderà la persona v. Se questa non è una mossa valida, oppure è il turno di Maria, otterrai il verdetto `Wrong Answer`.

Se Maria ha una strategia vincente all'inizio del gioco, devi uscire da `SocialEnginnering` *prima* della prima chiamata a `GetMove`. In questo caso otterrai il verdetto `Accepted`.

Assunzioni

- $2 \le n \le 2 \cdot 10^5$.
- $1 \le m \le 4 \cdot 10^5$.
- Il grafo è connesso. Ogni coppia di vertici appare nel grafo al più una volta. Ogni arco collega due vertici distinti.

Sottoproblemi

Maria giocherà sempre in modo ottimo, nel senso che giocherà con una stragia vincente se questa esiste. Se non esiste una strategia vincente, allora cercherà di indurre la soluzione a fare degli errori, in molti modi furbi. Maria si ritirerà solo se non ha mosse valide, tranne nel subtask 3.

1. (15 punti) $n, m \le 10$.
2. (15 punti) Tutti tranne Maria hanno al più due amici.
3. (20 punti) Maria si ritirerà immediatamente se non ha una strategia vincente.
4. (25 punti) $n, m \le 100$.
5. (25 punti) Nessuna limitazione aggiuntiva.

Sample Interaction

Azione della soluzione	Azione del grader	Spiegazione
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering()` è chiamata con un grafo di 5 vertici e 6 archi.
`GetMove()`	`Restituisce 4`	Maria sfida la persona 4.
`MakeMove(2)`	-	La persona 4 sfida la persona 2.
`MakeMove(5)`	-	La persona 2 sfida la persona 5.
`MakeMove(1)`	-	La persona 5 sfida Maria.
`GetMove()`	`Restituisce 0`	Maria non ha mosse legali, quindi si ritira.
`Esce da SocialEngineering`	-	La soluzione ha vinto, quindi la funzione `SocialEngineering()` esce.

Azione della soluzione	Azione del grader	Spiegazione
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` è chiamata con un grafo di 2 vertici e 1 arco.
`Esce da SocialEngineering`	-	Maria ha una strategia vincente in questo grafo, quindi la soluzione deve ritirarsi ritornando senza mai chiamare `GetMove`.

Grader di esempio

Il grader di esempio, `grader.cpp` all'interno dell'allegato `SocialEngineering.zip`, legge l'input dallo standard input secondo il formato:

- La prima riga contiene il numero di vertici *n*, e il numero di archi *m* del grafo.
- Le seguenti *m* righe contengono ciascuna due interi *a* e *b*, che indicano che c'è un arco tra *a* e *b*.

Il grader d'esempio legge l'input e chiama `SocialEngineering`. Nota che il grader d'esempio non implementa la strategia ottima per Maria ed è solo un esempio di una possibile interazione.

Per compilare il grader di esempio con la tua soluzione, puoi usare questo comando nel terminale:

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

dove `solution.cpp` è il file della tua soluzione che dovrai inviare a CMS. Per eseguire il tuo programma con l'input di esempio allegato, esegui questo comando nel terminale:

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