Social Engineering

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

Een sociaal netwerk bestaat uit een undirected connected graaf met n knopen en m lijnen, waar elke knoop een persoon voorstelt. Twee personen zijn vrienden als er een lijn tussen hun knopen zit.

Maria is lid van dit sociale netwerk. Ze daagt haar vrienden graag uit voor challenges. Dit betekent dat zij eerst een simpele taak doet en dan een van haar vrienden vraagt om hetzelfde te doen. Deze vriend daagt dan een vriend uit, die weer een van zijn vrienden uitnodigt, enzovoort. Het kan gebeuren dat dezelfde persoon meerdere keren uitgedaagd wordt, maar elk paar vrienden kan elkaar maar één keer uitdagen. (Als een persoon A persoon B uitdaagt, dan kan noch persoon A noch persoon B de ander nog een keer uitdagen). In andere woorden, de challenges lopen door het netwerk zonder een lijn twee keer te gebruiken.

Een persoon verliest de challenge als het zijn beurt is en hij geen van zijn vrienden kan uitdagen. De challenges worden altijd gestart door Maria en ze verliest bijna nooit. Daarom hebben de overige n - 1 personen besloten om samen te werken om Maria de volgende opdracht te laten verliezen. Het is jouw taak om deze uitdaging te coördineren.

Implementatie

Implementeer de volgende functies:

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

dat het spel speelt op een graaf met n knopen en m lijnen. Deze functie wordt exact 1 keer aangeroepen door de grader. De lijst `edges` bevat precies m paren van integers (u, v), die aangeven dat er een lijn is tussen knoop u en knoop v. Knopen zijn genummerd van 1 tot en met n. Maria is altijd knoop 1. Jouw functie kan calls maken naar de volgende functies: `int GetMove();` Iedere keer wanneer Maria aan de beurt is moet deze functie aangeroepen worden, zoals in het

begin van het spel. Als je deze functie aanroept wanneer Maria niet aan de beurt is, krijg je `Wrong Answer`. De functie kan één van de volgende waarden teruggeven:

- een integer v, met $2 \leq v \leq n$. Dit betekent dat Maria de persoon met index v uitdaagt. Dit is altijd een geldige zet.
- 0, als Maria opgeeft. Maria zal altijd opgeven, wanneer ze geen geldige zetten meer heeft. Wanneer dit gebeurt, moet jouw programma de functie `SocialEngineering` laten eindigen, en dan krijg je `Accepted`.

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

Deze functie moet aangeroepen worden, wanneer Maria niet aan de beurt is. Dit betekent dat de persoon die aan de beurt is persoon v uitdaagt. Als dit geen geldige zet is of als Maria aan de beurt is bij het maken van de call, dan krijg je `Wrong Answer`. Als Maria een winnende strategie heeft bij de start van de wedstrijd, dan moet je programma de functie `SocialEngineering` beëindigen *voor* de eerste call naar `GetMove()`. Je krijgt dan `Accepted`.

Constraints

- $2 \leq n \leq 2 \cdot 10^5$.
- $1 \leq m \leq 4 \cdot 10^5$.
- De graaf is verbonden. Er is maximaal één rechtstreekse lijn tussen elk paar knopen, en elke lijn zit tussen twee verschillende knopen.

Subtasks

Maria speelt altijd perfect, in de zin dat zij altijd winnende zetten zet wanneer ze een winnende strategie heeft. Als ze geen winnende strategie heeft, dan zal ze proberen het programma een fout te laten maken op verschillende slimme manieren. Ze zal enkel opgeven als ze geen geldige zetten kan doen, muv in Subtaak 3.

1. (15 punten) $n, m \leq 10$.
2. (15 punten) Iedereen behalve Maria heeft maximaal 2 vrienden.
3. (20 punten) Maria geeft onmiddelijk op, tenzij ze zeker weet dat ze kan winnen.
4. (25 punten) $n, m \leq 100$.
5. (25 punten) Geen verdere randvoowaarden.

Voorbeeld Interactie

Jouw actie	Grader actie	Uitleg
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering` wordt aangeroepen met een graaf met 5 knopen and 6 lijnen.
`GetMove()`	`Returns 4`	Maria daagt persoon 4 uit.
`MakeMove(2)`	-	Persoon 4 daagt persoon 2 uit.
`MakeMove(5)`	-	Persoon 2 daagt persoon 5 uit.
`MakeMove(1)`	-	Persoon 5 daagt Maria uit.
`GetMove()`	`Returns 0`	Maria heeft geen geldige zetten, dus ze geeft op.
`Returns`	-	Je hebt gewonnen en `SocialEngineering` moet stoppen.

Jouw actie	Grader actie	Uitleg
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` wordt aangeroepen met een graaf met 2 knopen en 1 lijn.
`Returns`	-	Maria heeft een winnende strategie voor deze graaf, dus je geeft op door je programma te beëindigen zonder `GetMove()` aan te roepen.

Voorbeeld Grader

De beschikbare sample grader, `grader.cpp`, in de attachment van de opdracht `SocialEngineering.zip`, leest de input van de standaard input in het volgende formaat:

- De eerste regel bevat het aantal knopen *n* en het aantal lijnen *m* in de graaf.
- De volgende *m* lijnen bevatten twee integers *u* and *v* die weergeven dat er een lijn is tussen *u* en *v*..

De Sample grader leest de input en callt de `SocialEngineering` functie in de oplossing van de gebruiker. Let op, de sample grader implementeert de winnende strategie van Marie niet en wordt alleen voor voorbeeld-interactie ter beschikking gesteld.

Om de sample grader te compilen met jouw oplossing, kun je het volgende commando gebruiken op de terminal prompt:

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

waar `solution.cpp` jouw oplossing is, die je wilt gaan submitten to CMS. Om het programme te runnen met de voorbeeld input in de bijlage, type het volgende commando in de terminal prompt:

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