Social Engineering

Nom du problème	Social Engineering
Fichier d'entrée	Sujet interactif
Fichier de sortie	Sujet interactif
Limite de temps	5 secondes
Limite de mémoire	256 megaoctets

Un réseau social est constitué d'un graphe connexe non orienté avec n noeuds et m arêtes, où chaque noeud est une personne, et où deux personnes sont amies s'il y a une arête entre elles.

Maria est membre de ce réseau social. Elle aime lancer des défis en tout genre à ses amies. Cela signifie qu'elle fait une action simple, puis défie l'une de ses amies de faire la même action. Cette amie va alors défier l'une de ses amies, qui défiera l'une des amies, et ainsi de suite. Il peut arriver que la même personne soit défiée plus d'une fois, mais chaque paire d'amies ne peut participer au défi qu'une fois au maximum. (Une fois qu'une personne A a défié une personne B, alors ni la personne A, ni la personne B ne peut lancer ce défi à l'autre plus tard). Autrement dit, les défis correspondent à un chemin dans le graphe, qui n'utilise jamais une même arête plus d'une fois.

Une personne perd le défi si c'est son tour et qu'elle ne peut défier aucune de ses amies. Un défi est toujours commencé par Maria, et elle perd rarement. Les n - 1 autres personnes ont décidé de collaborer pour faire perdre le prochain défi à Maria, et votre rôle est de coordonner cet effort.

Implémentation

Vous devez programmer une fonction :

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

qui joue au jeu dans un graphe constitué de n noeuds et m arêtes. Cette fonction sera appelée une fois par l'évaluateur. La liste `edges` contiendra exactement m paires d'entiers (u, v), ce qui signifie qu'une arête se trouve entre le noeud u et le noeud v. Les noeuds sont numérotés de 1 à n. Maria est toujours le noeud 1. Votre fonction peut appeler les fonctions suivantes :

`int GetMove();`

Cette fonction doit être appelée lorsque c'est le tour de Maria, par exemple au tout début du jeu. Si vous appelez cette fonction lorsque ce n'est pas le tour de Maria, vous obtiendrez le résultat `Wrong Answer`. La fonction peut retourner l'une des valeurs suivantes :

- un entier v, où $2 \le v \le n$. Cela signifie que Maria défie la personne de numéro v. Il s'agit toujours d'un coup valide.
- 0, si Maria abandonne le jeu. Maria abandonne toujours si elle ne peut jouer aucun coup valide. Lorsque cela se produit, votre programme doit faire un return pour sortir de la fonction `SocialEngineering`, et vous obtiendrez le résultat `Accepted`.

`void MakeMove(int v);`

Cette fonction doit être appelée lorsque ce n'est pas le tour de Maria. Cela signifie que la personne dont c'est le tour défie la personne v. Si ce n'est pas un coup valide, ou si c'est le tour de Maria au moment de l'appel, alors vous obtiendrez le résultat `Wrong Answer`.

Si Maria a une stratégie gagnante dès le début du jeu, alors votre programme doit faire un return pour sortir de la fonction `SocialEngineering`, *avant* le premier appel à `GetMove()`. Vous obtiendrez alors le résultat `Accepted`.

Contraintes

- $2 \le n \le 2 \cdot 10^5$.
- $1 \le m \le 4 \cdot 10^5$.
- Le graphe est connexe. Chaque paire non ordonnée de noeuds apparaîtra au maximum une fois parmi les arêtes, et toute arête relie deux noeuds différents.

Sous-tâches

Maria joue toujours parfaitement, dans le sens où elle joue des coups gagnants dès qu'elle a une stratégie gagnante. Si elle n'a pas une stratégie gagnante, alors elle essaiera de faire faire une erreur à votre programme, de différentes manières intelligentes. Elle n'abandonnera que si elle n'a aucun coup valide à jouer, sauf dans la Sous-tâche 3.

1. (15 points) $n, m \le 10$.
2. (15 points) Tout le monde sauf Maria a au maximum 2 amies.
3. (20 points) Maria abandonne immédiatement, sauf si elle dispose d'une stratégie gagnante.

4. (25 points) $n, m \leq 100$.

5. (25 points) Pas de contrainte supplémentaire.

Exemple d'interaction

Votre action	Action de l'évaluateur	Explication
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering` est appelé avec un graphe contenant 5 noeuds et 6 arêtes.
`GetMove()`	`Retourne 4`	Maria défie la personne 4.
`MakeMove(2)`	-	La personne 4 défie la personne 2.
`MakeMove(5)`	-	La personne 2 défie la personne 5.
`MakeMove(1)`	-	La personne 5 défie Maria.
`GetMove()`	`Retourne 0`	Maria n'a aucun coup valide possible, donc elle abandonne.
`Return`	-	Vous avez gagné le jeu, et devez effectuer un return pour quitter la fonction `SocialEngineering`.

Votre action	Action de l'évaluateur	Explication
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` est appelé avec un graphe contenant 2 noeuds et 1 arête.
`Return`	-	Maria a une stratégie gagnante avec ce graphe, donc vous devez quitter la fonction pour abandonner sans effectuer aucun appel à `GetMove()`.

Évaluateur d'exemple.

L'évaluateur d'exemple, `grader.cpp`, dans le fichier `SocialEngineering.zip` attaché au sujet, lit l'entrée depuis l'entrée standard, au format suivant :

- La première ligne contient le nombre de noeuds n, et le nombre d'arêtes m du graphe.
- Les m lignes suivantes contiennent deux entiers u et v, indiquant qu'il y a une arête entre u et v.

L'évaluateur d'exemple lit l'entrée et appelle la fonction `SocialEngineering` dans la solution de l'utilisateur. Notez que l'évaluateur d'exemple n'implémente pas la stratégie gagnante de Maria, et est fourni uniquement pour tester l'interaction.

Pour compiler l'évaluateur d'exemple avec votre solution, utilisez la commande suivante dans votre terminal : `g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp`
où `solution.cpp` est le fichier de votre solution, à transmettre à CMS. Pour exécuter le programme avec l'entrée d'exemple fournie dans le fichier attaché, lancez la commande suivante dans votre terminal : `./solution < input.txt`