Social Engineering

Задача	Social Engineering
Вхідні дані	Інтерактивна задача
Вихідні дані	Інтерактивна задача
Ліміт часу	5 секунд
Ліміт пам'яті	256 МБ

Соціальна мережа це неорієнтований зв'язний граф з n вершин і m ребер, де кожна вершина є людиною, а дві людини є друзями, якщо між ними є ребро. Марія є учасником цієї соціальної мережі. Вона любить кидати друзям виклики на різні речі. Це означає, що вона спочатку виконує якесь просте завдання, а потім викликає одного зі своїх друзів зробити те саме. Потім цей друг кине виклик одному зі своїх друзів, той кине виклик одному зі своїх друзів і так далі. Друзі це називають грою. Може статися, що одна й та сама особа отримує виклик кілька разів, але кожна невпорядкована пара друзів може брати участь у виклику лише один раз. (Якщо особа A кидає виклик особі B, тоді ні особа A, ні особа B не можуть знову кинути виклик один одному.) Іншими словами, виклики будуть прогулянкою по графу, яка ніколи не проходить через якесь ребро більше одного разу.

Людина програє гру, якщо на своєму ході вона не може кинути виклик жодному зі своїх друзів. Гру завжди починає Марія, і вона дуже рідко програє. Тепер решта $n-1$ людей вирішили скооперуватись, щоб змусити Марію програти, і ваша робота — координувати ці зусилля.

Реалізація

Вам потрібно реалізувати функцію:

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

яка запускає гру на графі з n вершин та m ребер. Ця функція буде викликана одноразово грейдером. Вектор `edges` міститиме рівно m пар цілих чисел (u, v), що означає існує ребро, яке проходить між вершиною u і вершиною v. Вершини пронумеровані від 1 до n. Марія завжди є вершиною 1. Ваша функція може викликати такі інші функції:

`int GetMove();`

Цю функцію слід викликати щоразу, коли настає черга Марії, наприклад, на початку гри. Якщо ви викличете її, коли не черга Марії, ви отримаєте вердикт «Неправильна відповідь». Функція може повертати одне з таких значень:

- Ціле число v, де $2 \le v \le n$. Це означає, що Марія кидає виклик людині з індексом v. Це завжди буде правильним викликом.
- 0, якщо Марія здасться. Марія завжди здасться, якщо у неї немає правильних ходів. Коли це станеться, ваша програма має завершити виконання функції `SocialEngineering`, і ви отримаєте вердикт «Прийнято».

`void MakeMove(int v);`

Цю функцію слід викликати під час ходу всіх інших гравців. Це означає, що особа, яка має хід, кидає виклик людині v. Якщо це неправильний хід або якщо цей хід є ходом Марії, ви отримаєте вердикт «Неправильна відповідь».

Якщо Марія має виграшну стратегію на початку гри, ваша програма має завершити виконання функції `SocialEngineering` *перед* першим викликом `GetMove()`. Після цього ви отримаєте вердикт «Прийнято».

Обмеження

- $2 \le n \le 2 \cdot 10^5$.
- $1 \le m \le 4 \cdot 10^5$.
- Граф зв'язний. В списку ребер, кожна невпорядкована пара вершин з’явиться не більше одного разу, і кожне ребро буде проходити між двома різними вершинами.

Оцінювання

Марія завжди гратиме ідеально в тому сенсі, що вона робитиме виграшні ходи, коли у неї буде виграшна стратегія. Якщо у неї немає виграшної стратегії, вона намагатиметься змусити вашу програму зробити помилку різними хитрими способами. Вона здасться, лише якщо не матиме жодних правильних ходів (це не стосується підзадачі 3).

1. (15 балів) $n, m \le 10$.
2. (15 балів) Будь-хто, окрім Марії, має не більше 2-х друзів.
3. (20 балів) Марія здасться одразу, якщо вона не має виграшної стратегії.

4. (25 балів) $n, m \leq 100$.

5. (25 балів) Без додаткових обмежень.

Приклад взаємодії

Ваша дія	Дія грейдера	Пояснення
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering` викликане на графі з 5 вершин і 6 ребер.
`GetMove()`	`Повертає 4`	Марія кидає виклик людині з номером 4.
`MakeMove(2)`	-	Людина з номером 4 кидає виклик людині з номером 2.
`MakeMove(5)`	-	Людина з номером 2 кидає виклик людині з номером 5.
`MakeMove(1)`	-	Людина з номером 5 кидає виклик Марії.
`GetMove()`	`Повертає 0`	Марія не може здійснити правильного ходу, отже вона здається.
`Завершує виконання функції (return)`	-	Ви виграли гру, і завершили виконання функції `SocialEngineering` за допомогою return.

Ваша дія	Дія грейдера	Пояснення
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` є викликана з графом з 2-х вершин і 1-го ребра.
`Завершує виконання функції (return)`	-	Марія має виграшну стратегію на цьому графі, тому ви повинні завершити виконання функції до першого виклику `GetMove()`.

Приклад грейдера

Наданий зразок грейдера, `grader.cpp` у файлі `SocialEngineering.zip`, зчитує вхідні дані зі стандартного вводу в такому форматі:

- Перший рядок містить кількість вершин n і кількість ребер m в графі.
- Наступні m рядків містять ребра як пари вершин: числа u та v означають що існує ребро між вершинами u та v.

Зразок грейдера зчитує вхідні дані та викликає функцію `SocialEngineering` у рішенні учасника. Зауважте, що зразок грейдера не реалізовує виграшну стратегію Марії та надається лише для зразка взаємодії.
Щоб скомпілювати зразок грейдера з вашим рішенням, ви можете використати наступну команду в терміналі:

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

де `solution.cpp` це ваше рішення, яке ви відправляєте в CMS. Щоб запустити цю програму достатньо виконати в терміналі:

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