Inżynieria społeczna

Nazwa zadania	Inżynieria społeczna
Wejście	Zadanie interaktywne
Wyjście	Zadanie interaktywne
Limit czasu	5 sekund
Limit pamięci	256 MB

Sieć społeczna jest nieskierowanym spójnym grafem o n wierzchołkach i m krawędziach, w ktorym każdy wierzchołek jest osobą i dwie osoby się znają, jeśli jest między nimi krawędź. Ala jest członkiem sieci społecznej. Lubi rzucać swoim znajomym przeróżne wyzwania. To oznacza, że najpierw wykonuje pewne proste zadanie, a potem prosi jednego ze swoich znajomych, by zrobił to samo. Później to wyzwanie jest przekazywane dalej w sieci. Może się zdarzyć, że jedna osoba zostaje wyzwana więcej niż raz, ale każda nieuporządkowana para znajomych może wziąć udział w wyzwaniu co najwyżej raz (jeśli osoba A rzuci wyzwanie osobie B, to ani osoba A nie może już wyzwać osoby B, ani osoba B nie może już wyzwać osoby A). Innymi słowy, droga przekazywania wyzwania w sieci społecznej będzie ścieżką w grafie, która nie odwiedza żadnej krawędzi więcej niż raz, ale może wielokrotnie odwiedzać wierzchołki. Członek sieci przegrywa tę zabawę, jeśli jest jego kolej i nie może rzucić wyzwania żadnemu ze swoich znajomych. Wyzwania zawsze są rozpoczynane przez Alę i, co interesujące, Ala rzadko przegrywa. Teraz pozostałe n-1 osób postanowiło współpracować i sprawić, by Ala przegrała kolejną rozgrywkę, a Twoim zadaniem jest zarządzanie tym sojuszem.

Implementacja

Powinnaś zaimplementować funkcję: `void SocialEngineering(int n, int m, vector<pair<int,int>> edges);`
która przeprowadza opisaną wyżej rozgrywkę na grafie o n wierzchołkach i m krawędziach. Ta funkcja będzie wywołana raz przez sprawdzaczkę. Wektor `edges` zawiera dokładnie m par liczb całkowitych (u, v) oznaczających, że pomiędzy wierzchołkami u i v istnieje krawędź. Wierzchołki są ponumerowane od 1 do n. Ala jest zawsze wierzchołkiem 1. Twoja funkcja może wywoływać następujące funkcje:

`int GetMove();`

Ta funkcja powinna być wywołana wtedy, gdy jest kolej Ali (w szczególności na początku rozgrywki). Jeśli wywołasz tę funkcję w momencie, gdy jest kolej kogoś innego, otrzymasz werdykt `Wrong Answer`. Funkcja zwraca jedną z następujących wartości:

- liczbę całkowitą v, gdzie $2 \le v \le n$. To oznacza, że Ala rzuca wyzwanie osobie o numerze v. Ten ruch będzie zawsze dozwolony.
- 0, jeśli Ala zrezygnuje z gry. Ala zrezygnuje, gdy nie będzie mogła wykonać żadnego dozwolonego ruchu. Gdy to nastąpi, Twój program powinien przerwać wykonywanie funkcji `SocialEngineering` i otrzymasz werdykt `Accepted`.

`void MakeMove(int v);`

Ta funkcja powinna być wywołana w momencie, gdy jest kolej kogoś innego niż Ala. To oznacza, że osoba, której jest kolej, wyzywa osobę v. Jeśli to nie jest dozwolony ruch lub jeśli w rzeczywistości jest kolej Ali, otrzymasz werdykt `Wrong Answer`. Jeśli Ala ma strategię wygrywającą na początku gry, Twój program powienien zakończyć wywołanie funkcji `SocialEngineering` *przed* pierwszym wywołaniem `GetMove`. Wtedy otrzymasz werdykt `Accepted`.

Ograniczenia

- $2 \le n \le 2 \cdot 10^5$.
- $1 \le m \le 4 \cdot 10^5$.
- Graf jest spójny. Każda nieuporządkowana para wierzchołków pojawi się co najwyżej raz jako krawędź i każda krawędź będzie biegła pomiędzy dwoma różnymi wierzchołkami.

Podzadania

Ala zawsze będzie grała optymalnie w następującym znaczeniu: będzie wykonywać ruchy wygrywające, gdy ma strategię wygrywającą. Jeśli nie ma strategii wygrywającej, spróbuje omamić Twój program, aby popełnił błąd - a ma na to wiele różnych, podstępnych sposobów. Zrezygnuje z gry wtedy i tylko wtedy, gdy nie będzie miała żadnych dozwolonych ruchów, z wyjątkiem Podzadania 3.

Podzadanie 1 (15 punktów): $n, m \le 10$.

Podzadanie 2 (15 punktów): Każdy poza Alą ma co najwyżej 2 znajomych.

Podzadanie 3 (20 punktów): Ala zrezygnuje natychmiast, chyba że ma strategię wygrywającą.

Podzadanie 4 (25 punktów): $n, m \le 100$.

Podzadanie 5 (25 punktów): Brak dodatkowych ograniczeń.

Przykłady

Akcja zawodniczki	Akcja sprawdzaczki	Wyjaśnienie
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering` jest wywołane z grafem o 5 wierzchołkach i 6 krawędziach.
`GetMove()`	`Zwraca 4`	Ala rzuca wyzwanie osobie o numerze 4.
`MakeMove(2)`	-	Osoba 4 rzuca wyzwanie osobie 2.
`MakeMove(5)`	-	Osoba 2 rzuca wyzwanie osobie 5.
`MakeMove(1)`	-	Osoba 5 wyzywa Alę.
`GetMove()`	`Zwraca 0`	Ala nie ma dozwolonych ruchów, więc rezygnuje.
`Kończy wywołanie funkcji`	-	Zawodniczka wygrała grę i powinna pozwolić, by funkcja `SocialEngineering` zakończyła swoje wywołanie.

Akcja zawodniczki	Akcja sprawdzaczki	Wyjaśnienie
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` jest wywołane z grafem o 2 wierzchołkach i 1 krawędzią.
`Kończy wywołanie funkcji`	-	Ala ma strategię wygrywającą dla tego grafu, więc zawodniczka powinna zakończyć wywołanie funkcji bez wywoływania funkcji `GetMove()`.

Sprawdzaczka pomocnicza

Udostępniona sprawdzaczka pomocnicza, `grader.cpp`, w załączniku `SocialEngineering.zip`, czyta dane ze standardowego wejścia w następującym formacie:

- Pierwsza linia zawiera liczbę wierzchołków, *n*, i liczbę krawędzi *m* w grafie.
- Kolejne *m* linii zawiera dwie liczby całkowite *u* i *v* oznaczające, że pomiędzy *u* i *v* istnieje krawędź.

Sprawdzaczka pomocnicza czyta wejście i wywołuje funkcję `SocialEngineering` z rozwiązania zawodniczki. Zauważ, że sprawdzaczka nie sprawdza, czy Ala ma strategię wygrywającą i jest przeznaczona jedynie do przykładowej interakcji. Aby skompilować sprawdzaczkę pomocniczą ze swoim rozwiązaniem, możesz użyć następującej komendy w terminalu: `g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp`
gdzie `solution.cpp` jest plikiem z rozwiązaniem, który wyślesz do CMS. Aby uruchomić program z przykładowym wejściem z załącznika, wpisz następującą komendę w terminalu: `./solution < input.txt`.