Gra (Game)

Jian-Jia jest chłopakiem, który uwielbia grać w gry. Zapytany o coś, zamiast odpowiedzieć bezpośrednio woli rozpocząć grę ze swoim rozmówcą. Pewnego dnia Jian-Jia spotkał swoją przyjaciółkę Mei-Yu i opowiedział jej o sieci połączeń lotniczych na Tajwanie. Na Tajwanie jest n miast ponumerowanych 0, ..., $n-1$. Między niektórymi miastami istnieją bezpośrednie połączenia lotnicze. Każde takie połączenie jest dwukierunkowe.

Mei-Yu była ciekawa, czy za pomocą samolotów można przemieścić się pomiędzy każdymi dwoma miastami (bezpośrednio lub pośrednio). Jian-Jia nie odpowiedział na pytanie przyjaciółki, ale zaproponował jej pewną grę. Mei-Yu może w niej zadawać Jian-Jia pytania postaci „Czy istnieje *bezpośrednie* połączenie lotnicze pomiędzy miastami x oraz y?". Na każde takie pytanie Jian-Jia natychmiast odpowiada. Mei-Yu zada łącznie $r = n(n-1)/2$ pytań, pytając o każde połączenie dokładnie raz. Mei-Yu zwycięża w grze, jeśli po zadaniu i pytań, dla pewnego $i < r$, może stwierdzić, czy sieć jest spójna, tzn. czy da się podróżować samolotami pomiędzy każdymi dwoma miastami (bezpośrednio lub pośrednio). Jeśli do stwierdzenia, czy sieć jest spójna czy nie, potrzebuje ona r pytań, wówczas zwycięzcą w grze jest Jian-Jia.

Żeby gra była ciekawsza dla Jian-Jia, jego przyjaciółka zgodziła się, aby gra nie odnosiła się do rzeczywistej sieci lotniczej na Tajwanie. Zamiast tego Jian-Jia może tworzyć strukturę sieci na bieżąco w trakcie gry, biorąc pod uwagę swoje wcześniejsze odpowiedzi na pytania Mei-Yu. Twoim zadaniem jest pomóc Jian-Jia zwyciężyć, podpowiadając mu odpowiedzi na zadawane pytania.

Przykłady

Reguły gry zostaną objaśnione na trzech przykładach. W każdym przykładzie liczba miast to $n = 4$, natomiast liczba rund pytań i odpowiedzi to $r = 6$.

W pierwszym przykładzie (tabela poniżej) Jian-Jia *przegrywa*, ponieważ po rundzie 4 Mei-Yu wie na pewno, że można przelecieć samolotami pomiędzy każdymi dwoma miastami, niezależnie od odpowiedzi Jian-Jia na pytania 5 i 6.

runda	pytanie	odpowiedź
1	0, 1	tak
2	3, 0	tak
3	1, 2	nie
4	0, 2	tak
-----	--------	------
5	3, 1	nie
6	2, 3	nie

W kolejnym przykładzie Mei-Yu może wykazać już po 3 rundzie, że *nie można* przelecieć samolotami

pomiędzy miastami 0 i 1, niezależnie od tego, co Jian-Jia odpowie na pytania 4, 5 i 6. Jian-Jia znowu przegrywa.

runda	pytanie	odpowiedź
1	0, 3	nie
2	2, 0	nie
3	0, 1	nie
-----	--------	------
4	1, 2	tak
5	1, 3	tak
6	2, 3	tak

W ostatnim przykładzie Mei-Yu nie może stwierdzić, czy pomiędzy każdymi dwoma miastami da się przelecieć, czy nie, aż do momentu, gdy wszystkie 6 pytań zostanie zadane. Jian-Jia *wygrywa* grę. Jest tak dlatego, gdyż jeśli Jian-Jia odpowie *tak* na ostatnie pytanie (patrz tabela poniżej), to podróż samolotami pomiędzy każdą parą miast jest możliwa, natomiast jeśli Jian-Jia odpowie *nie* na ostatnie pytanie, wówczas istnieje para miast, pomiędzy którymi podróż lotnicza nie jest możliwa.

runda	pytanie	odpowiedź
1	0, 3	nie
2	1, 0	tak
3	0, 2	nie
4	3, 1	tak
5	1, 2	nie
6	2, 3	tak

Zadanie

Napisz program, który pomoże Jian-Jia zwyciężyć w grze. Zauważ, że ani Mei-Yu, ani Jian-Jia nie znają strategii przeciwnika. Mei-Yu może pytać o połączenia pomiędzy parami miast w dowolnej kolejności, a Jian-Jia musi odpowiadać natychmiast na każde pytanie, nie znając kolejnych pytań. Twoje zadanie polega na zaimplementowaniu dwóch następujących funkcji.

- `initialize(n)` -- Funkcja `initialize` będzie wywołana jako pierwsza. Parametr n oznacza liczbę miast.
- `hasEdge(u, v)` -- Następnie będzie wywoływana $r = n(n-1)/2$ razy funkcja `hasEdge`. Te wywołania reprezentują pytania Mei-Yu i są wykonywane w kolejności ich zadawania. Na każde z nich należy odpowiedzieć, czy istnieje bezpośrednie połączenie lotnicze pomiędzy miastami u i v. Jeśli takie bezpośrednie połączenie istnieje, wówczas wynikiem wywołania funkcji powinno być 1, a w przeciwnym przypadku powinno to być 0.

Podzadania

Każde podzadanie składa się z pewnej liczby gier. Twój program otrzyma punkty za dane podzadanie, tylko jeśli w imieniu Jian-Jia wygra wszystkie gry.

podzadanie	liczba punktów	n
1	15	$n = 4$
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Implementacja

Powinieneś zgłosić dokładnie jeden plik o nazwie `game.c`, `game.cpp` lub `game.pas`. W tym pliku powinna znaleźć się implementacja funkcji opisanych powyżej, o następujących sygnaturach.

Programy w C/C++

```
void initialize(int n);
int hasEdge(int u, int v);
```

Programy w Pascalu

```
procedure initialize(n: longint);
function hasEdge(u, v: longint): longint;
```

Przykładowy program sprawdzający

Przykładowy program sprawdzający wczytuje dane w następującym formacie:

- wiersz 1: `n`
- kolejne r wierszy: każdy wiersz zawiera dwie liczby całkowite `u` i `v`, które opisują pytanie odnoszące się do miast u i v.