Spel

Jian-Jia heeft een voorliefde voor het spelen van spellen. Wanneer hij een vraag krijgt, zal hij eerder een spelletje spelen dan direct te antwoorden. Jian-Jia vertelde zijn vriendin Mei-Yu bij hun laatste ontmoeting over het luchtvaartnetwerk in Taiwan. Er zijn n steden in Taiwan (genummerd 0, ..., $n-1$), waarvan sommigen zijn verbonden met een vlucht. Elke vlucht verbindt twee steden en kan in beide richtingen worden genomen.

Mei-Yu vroeg Jian-Jia of het mogelijk is om tussen eender welke twee steden te reizen per vliegtuig (direct of indirect). Jian-Jia wou het antwoord niet onmiddellijk onthullen, en stelde voor om in de plaats een spel te spelen. Mei-Yu kan hem vragen stellen van de vorm: "Zijn steden x en y *direct* verbonden met een vlucht?", en Jian-Jia zal zulke vragen onmiddellijk beantwoorden. Mei-Yu zal over elk paar steden exact éénmaal die vraag stellen, dat geeft in totaal $r = n(n-1)/2$ vragen. Mei-Yu wint het spel als ze na het krijgen van de antwoorden op de eerste i vragen voor een zekere $i < r$, kan afleiden of het netwerk geconnecteerd is, d.w.z. of het mogelijk is tussen elk paar steden te reizen per vliegtuig (direct of indirect). In het andere geval, dus wanneer ze alle r vragen nodig heeft, is Jian-Jia de winnaar.

Om het spel wat leuker te maken voor Jian-Jia, zijn de vrienden overeen gekomen dat hij het echte Taiwanese luchtvaartnetwerk mag vergeten. Hij mag zelf het netwerk verzinnen terwijl het spel vordert, en zijn antwoorden kiezen gebaseerd op Mei-Yu's vorige vragen. Jouw taak is om Jian-Jia te helpen het spel te winnen, door te beslissen hoe hij de vragen moet beantwoorden.

Voorbeelden

We leggen de spelregels uit met drie voorbeelden Elk voorbeeld heeft $n = 4$ steden en $r = 6$ rondjes vraag en antwoord.

In het eerste voorbeeld (de volgende tabel), *verliest* Jian-Jia omdat na ronde 4, Mei-Yu al zeker weet dat men tussen eender welke twee steden kan vliegen, ongeacht hoe Jian-Jia vragen 5 of 6 beantwoordt.

ronde	vraag	antwoord
1	0, 1	ja
2	3, 0	ja
3	1, 2	nee
4	0, 2	ja
-----	--------	------
5	3, 1	nee
6	2, 3	nee

In het volgende voorbeeld kan Mei-Yu na ronde 3 bewijzen dat, hoe Jian-Jia vragen 4, 5 of 6 ook beantwoordt, men *niet* tussen steden 0 en 1 kan vliegen, dus Jian-Jia verliest opnieuw.

ronde	vraag	antwoord
1	0, 3	nee
2	2, 0	nee
3	0, 1	nee
-----	--------	------
4	1, 2	ja
5	1, 3	ja
6	2, 3	ja

In het laatste voorbeeld kan Mei-Yu niet bepalen of men tussen eender welke twee steden kan vliegen totdat alle 6 vragen zijn beantwoord, dus Jian-Jia *wint* het spel. Meer specifiek, net omdat Jian-Jia *ja* heeft geantwoord op de laatste vraag (in de volgende tabel), is het mogelijk tussen eender welk paar steden te vliegen. Als Jian-Jia *nee* geantwoord zou hebben op de laatste vraag, zou dat onmogelijk zijn.

ronde	vraag	antwoord
1	0, 3	nee
2	1, 0	ja
3	0, 2	nee
4	3, 1	ja
5	1, 2	nee
6	2, 3	ja

Taak

Schrijf een programma dat Jian-Jia helpt om het spel te winnen. Let op dat Mei-Yu noch Jian-Jia elkaars strategie kennen. Mei-Yu kan vragen stellen over paren van steden in eender welke volgorde, en Jian-Jia moet onmiddellijk een antwoord geven zonder de volgende vragen te kennen. Je moet de volgende twee functies implementeren.

- `initialize(n)` -- We zullen jouw functie `initialize` eerst aanroepen. De parameter n is het aantal steden.
- `hasEdge(u, v)` -- Daarna roepen we $r = n(n-1)/2$ keer jouw functie `hasEdge` aan. Deze functie-calls stellen Mei-Yu's vragen voor, in de volgorde waarin zij ze stelt. Je moet antwoorden of er een directe vlucht is tussen steden u en v: de teruggegeven waarde moet 1 zijn als er een directe vlucht is, en 0 indien niet.

Subtaken

Elke subtaak bevat verschillende spelletjes. Je krijgt enkel punten voor een subtaak als jouw programma alle spelletjes kan winnen voor Jian-Jia.

subtaak	punten	n
1	15	$n = 4$
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Implementatiedetails

Je moet exact één bestand indienen, genaamd `game.c`, `game.cpp` of `game.pas`. Dit bestand implementeert de hierboven beschreven subroutines volgens de volgende declaraties.

C/C++ programma's

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

Pascal programma's

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

Voorbeeldgrader

De voorbeeldgrader leest input in het volgende formaat:

- lijn 1: `n`
- de volgende r lijnen: elke lijn bevat twee integers `u` en `v` die een vraag beschrijven over de steden u and v.