Game

Jian-Jia houdt van spelletjes. Als je hem een vraag stelt, maakt hij er liever een spelletje van, dan dat hij meteen antwoord geeft. Jian-Jia treft zijn vriendin Mei-Yu en heeft het met haar over het netwerk van luchtvaartverbindingen in Taiwan. Er zijn n steden in Taiwan (van 0 tot en met $n-1$ genummerd), sommige steden zijn verbonden door een rechtstreekse vlucht. Elke vlucht verbindt twee steden en kan beide kanten op worden genomen.

Mei-Yu vraagt Jian-Jia of elk paar steden verbonden is door een luchtvaartverbinding (rechtstreeks of door over te stappen). Jian-Jia wil dat niet vertellen, maar stelt voor om er een spelletje van te maken. Mei-Yu kan hem vragen stellen van het type: "Is er een rechtstreekse vlucht tussen de steden x en y ?", en Jian-Jia zal zulke vragen meteen beantwoorden. Mei-Yu zal dat voor ieder paar steden precies één keer vragen, dat betekent in totaal $r = n(n-1)/2$ vragen. Mei-Yu wint het spel als ze, wanneer ze antwoord heeft gekregen op de eerste i vragen, voor een $i < r$, kan beredeneren of het netwerk verbonden is, dat wil zeggen dat alle paren steden onderling verbonden zijn, al dan niet door een rechtstreekse vlucht. Als ze dat pas weet na alle r vragen, dan heeft Jian-Jia gewonnen.

Om er een leuker spel van te maken voor Jian-Jia, spreken ze af dat hij het feitelijke Taiwanese luchtvaartsysteem mag vergeten. Hij mag de vragen beantwoorden zoals hij wil en zo het netwerk verzinnen, afhankelijk van de vragen die Mei-Yu hem stelt. Het is jouw opdracht om Jian-Jia te helpen het spel te winnen, door te beslissen hoe hij de vragen moet beantwoorden.

Voorbeelden

We leggen de spelregels uit aan de hand van drie voorbeelden. In elk voorbeeld zijn er $n = 4$ steden en $r = 6$ vraag- en antwoordrondes.

In het eerste voorbeeld (zie onderstaande tabel), *verliest* Jian-Jia omdat Mei-Yu na vier rondes met vraag en antwoord zeker weet dat alle steden verbonden zijn, hoe Jian-Jia de vragen 5 en 6 ook zal beantwoorden.

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 al bewijzen dat, hoe Jian-Jia de vragen 4, 5, en 6 ook beantwoord, het *niet mogelijk* is om tussen de steden 0 en 1 te vliegen, en dus verliest Jian-Jia 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 de vraag of alle steden onderling verbonden zijn niet beantwoorden voordat alle zes vragen zijn beantwoord. Dus *wint* Jian-Jia het spel. Als Jian-Jia met *ja* op de laatste vraag antwoordt (in de tabel hieronder), dan zijn alle steden met elkaar verbonden. Maar als Jian-Jia met *nee* op deze laatste vraag antwoordt, dan zal dat niet het geval 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

Opdracht

Schrijf een programma dat Jian-Jia helpt om dit spel te winnen. Bedenk dat zowel Mei-Yu als Jian-Jia de strategie van de ander niet kent. Mei-Yu kan de vragen over paren steden in willekeurige volgorde stellen, en Jian-Jia moet deze vragen meteen beantwoorden, zonder iets te weten over de volgende vragen. Je moet twee functies implementeren.

- `initialize(n)` -- De functie `initialize` wordt als eerste aangeroepen. De parameter n is het aantal steden.
- `hasEdge(u, v)` -- Daarna wordt `hasEdge` $r = n(n-1)/2$ keer aangeroepen. Deze aanroepen horen bij de vragen van Mei-Yu, in de volgorde waarin ze die heeft gesteld. Jij moet aangeven of er een rechtstreekse vlucht is tussen de steden u en v. Let op, de return waarde moet een 1 zijn als er een rechtstreekse vlucht is, of een 0 als dat niet het geval is.

Subtasks

Elke subtask bestaat uit verschillende spelletjes. Je kunt alleen maar punten voor een subtask krijgen als je programma alle spelletjes voor Jian-Jia wint.

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

Implementatie details

Je moet precies één bestand submitten, dat `game.c`, `game.cpp` of `game.pas` heet. Dit bestand implementeert de subprogramma's die hierboven beschreven zijn, met de volgende syntax:

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;
```

Voorbeeld grader

De voorbeeld grader leest de invoer in het volgende formaat:

- regel 1: `n`
- de volgende r regels: iedere regel bevat twee integers `u` en `v` die aangeven dat er gevraagd wordt naar een rechtstreekse verbinding tussen de steden u en v.