Spiel

Jian-Jia ist ein junger Knabe, der Spiele mag. Wenn man ihm eine Frage stellt, zieht er es vor Spiele zu spielen, anstatt die Frage direkt zu beantworten. Jian-Jia hat seine Freundin Mei-Yu getroffen und hat ihr vom Flugnetzwerk in Taiwan erzählt. Es gibt n Städte in Taiwan (nummeriert von 0 bis $n-1$), wobei einige davon durch Flugstrecken verbunden sind. Jede Flugstrecke verbindet zwei Städte und kann in beide Richtungen geflogen werden.

Mei-Yu fragte Jian-Jia, ob es möglich ist von jeder Stadt zu jeder anderen zu fliegen (entweder direkt oder indirekt). Jian-Jia wollte die Antwort nicht verraten, aber schlug vor, stattdessen ein Spiel zu spielen. Mei-Yu kann ihm Fragen der Form "Sind die Städte x und y mit einem *direkten* Flug verbunden?" stellen und Jian-Jia wird solche Fragen umgehend beantworten. Mei-Yu wird nach jedem Paar von Städten genau einmal fragen, was insgesamt in $r = n(n-1)/2$ Fragen resultiert. Mei-Yu gewinnt das Spiel, falls sie mit den Antworten zu den ersten i Fragen, für ein $i < r$, schliessen kann, ob das Netzwerk zusammenhängend ist, das heisst, ob es möglich ist zwischen jedem Paar von Städten zu reisen oder nicht (direkt oder indirekt). Sonst (falls sie alle r Fragen benötigt) ist Jian-Jia der Gewinner.

Damit das Spiel Jian-Jia mehr Spass macht, haben die beiden Freunde abgemacht, dass er das reale Taiwanesische Flugnetzwerk vergessen und das Netzwerk im Laufe des Spieles nach und nach erfinden darf. Dabei kann er seine Antworten basierend auf Mei-Yus bisherigen Fragen wählen. Deine Aufgabe ist es Jian-Jia zu helfen, das Spiel zu gewinnen, indem du entscheidest, wie er die Fragen beantworten soll.

Beispiele

Wir erklären die Spielregeln anhand dreier Beispiele. Jedes Beispiel hat $n = 4$ Städte und $r = 6$ Runden von Fragen und Antworten.

Im ersten Beispiel (der untenstehenden Tabelle), *verliert* Jian-Jia, da Mei-Yu nach vier Runden sicher ist, dass man zwischen jedem Paar von Städten fliegen kann, egal wie Jian-Jia die Fragen 5 oder 6 beantwortet.

Runde	Frage	Antwort
1	0, 1	Ja
2	3, 0	Ja
3	1, 2	Nein
4	0, 2	Ja
-----	--------	------
5	3, 1	Nein
6	2, 3	Nein

Im nächsten Beispiel kann Mei-Yu nach der dritten Runde beweisen, dass man *nicht* zwischen den Städten 0 und 1 per Flugzeug reisen kann, egal wie Jian-Jia die Fragen 4, 5 oder 6 beantwortet. Also verliert Jian-Jia erneut.

Runde	Frage	Antwort
1	0, 3	Nein
2	2, 0	Nein
3	0, 1	Nein
-----	--------	------
4	1, 2	Ja
5	1, 3	Ja
6	2, 3	Ja

Im letzten Beispiel kann Mei-Yu nicht bestimmen, ob man zwischen jedem Paar von Städten fliegen kann, bis alle sechs Fragen beantwortet sind, also *gewinnt* Jian-Jia das Spiel. Genauer, weil Jian-Jia die letzte Frage mit *Ja* beantwortete (in der Tabelle unten) ist es möglich zwischen jedem Städte-Paar zu reisen. Doch falls Jian-Jia die letzte Frage mit *Nein* beantwortet hätte, wäre es unmöglich.

Runde	Frage	Antwort
1	0, 3	Nein
2	1, 0	Ja
3	0, 2	Nein
4	3, 1	Ja
5	1, 2	Nein
6	2, 3	Ja

Aufgabe

Bitte schreibe ein Programm, welches Jian-Jia hilft, das Spiel zu gewinnen. Beachte, dass weder Mei-Yu noch Jian-Jia die Strategie des anderen kennt. Mei-Yu kann Fragen zu Städte-Paaren in einer beliebigen Reihenfolge stellen und Jian-Jia muss diese umgehend beantworten, ohne die zukünftigen Fragen zu kennen. Du musst die folgenden zwei Funktionen implementieren:

- `initialize(n)` -- Wir rufen deine `initialize` Funktion zuerst auf. Der Parameter n ist die Anzahl der Städte.
- `hasEdge(u, v)` -- Danach rufen wir die `hasEdge` Funktion $r = n(n-1)/2$ Mal auf. Diese Aufrufe stellen Mei-Yus Fragen in der von ihr gewählten Reihenfolge dar. Du musst antworten, ob es einen Direktflug zwischen den Städten u und v gibt. Genauer, der Rückgabewert soll 1 sein, falls es einen Direktflug gibt und sonst 0.

Subtasks

Jeder Subtask besteht aus mehreren Spielen. Du bekommst nur Punkte für einen Subtask, falls dein

Programm alle Spiele für Jian-Jia gewinnt.

Subtask	Punkte	n
1	15	$n = 4$
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Implementierungsdetails

Du darfst genau eine Datei einsenden, welche `game.c`, `game.cpp` oder `game.pas` heisst. Diese Datei muss mit den folgenden Signaturen die oben beschriebenen Teilprogramme implementieren:

Programme in C/C++

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

Programme in Pascal

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

Sample-Grader

Der Sample-Grader liest die Eingabe in folgendem Format:

- Erste Zeile: `n`
- r folgende Zeilen: jede Zeile enthält zwei Ganzzahlen `u` und `v`, die eine Frage zu den Städten u und v beschreiben.