Game

Jian-Jia is a young boy who loves playing games. When he is asked a question, he prefers playing games rather than answering directly. Jian-Jia met his friend Mei-Yu and told her about the flight network in Taiwan. There are n cities in Taiwan (numbered 0, ..., $n-1$), some of which are connected by flights. Each flight connects two cities and can be taken in both directions.

Mei-Yu asked Jian-Jia whether it is possible to go between any two cities by plane (either directly or indirectly). Jian-Jia did not want to reveal the answer, but instead suggested to play a game. Mei-Yu can ask him questions of the form "Are cities x and y *directly* connected with a flight?", and Jian-Jia will answer such questions immediately. Mei-Yu will ask about every pair of cities exactly once, giving $r = n(n-1)/2$ questions in total. Mei-Yu wins the game if, after obtaining the answers to the first i questions for some $i < r$, she can infer whether or not it is possible to travel between every pair of cities by flights (either directly or indirectly). Otherwise, that is, if she needs all r questions, then the winner is Jian-Jia.

In order for the game to be more fun for Jian-Jia, the friends agreed that he may forget about the real Taiwanese flight network, and invent the network as the game progresses, choosing his answers based on Mei-Yu's previous questions. Your task is to help Jian-Jia win the game, by deciding how he should answer the questions.

Examples

We explain the game rules with three examples. Each example has $n = 4$ cities and $r = 6$ rounds of question and answer.

In the first example (the following table), Jian-Jia *loses* because after round 4, Mei-Yu knows for certain that one can travel between any two cities by flights, no matter how Jian-Jia answers questions 5 or 6.

round	question	answer
1	0, 1	yes
2	3, 0	yes
3	1, 2	no
4	0, 2	yes
-----	--------	------
5	3, 1	no
6	2, 3	no

In the next example Mei-Yu can prove after round 3 that no matter how Jian-Jia answers questions 4, 5, or 6, one *cannot* travel between cities 0 and 1 by flights, so Jian-Jia loses again.

round	question	answer
1	0, 3	no
2	2, 0	no
3	0, 1	no
-----	--------	------
4	1, 2	yes
5	1, 3	yes
6	2, 3	yes

In the final example Mei-Yu cannot determine whether one can travel between any two cities by flights until all six questions are answered, so Jian-Jia *wins* the game. Specifically, because Jian-Jia answered *yes* to the last question (in the following table), then it is possible to travel between any pair of cities. However, if Jian-Jia had answered *no* to the last question instead then it would be impossible.

round	question	answer
1	0, 3	no
2	1, 0	yes
3	0, 2	no
4	3, 1	yes
5	1, 2	no
6	2, 3	yes

Task

Please write a program that helps Jian-Jia win the game. Note that neither Mei-Yu nor Jian-Jia knows the strategy of each other. Mei-Yu can ask about pairs of cities in any order, and Jian-Jia must answer them immediately without knowing the future questions. You need to implement the following two functions.

- `initialize(n)` -- We will call your `initialize` first. The parameter n is the number of cities.
- `hasEdge(u, v)` -- Then we will call `hasEdge` for $r = n(n-1)/2$ times. These calls represent Mei-Yu's questions, in the order that she asks them. You must answer whether there is a direct flight between cities u and v. Specifically, the return value should be 1 if there is a direct flight, or 0 otherwise.

Subtasks

Each subtask consists of several games. You will only get points for a subtask if your program wins all of the games for Jian-Jia.

subtask	points	n
1	15	$n = 4$

subtask	points	n
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Implementation details

You have to submit exactly one file, called `game.c`, `game.cpp` or `game.pas`. This file implements the subprograms described above using the following signatures.

C/C++ programs

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

Pascal programs

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

Sample grader

The sample grader reads the input in the following format:

- line 1: `n`
- the following r lines: each line contains two integers `u` and `v` that describe a question regarding cities u and v.