Game

Jian-Jia este un tânăr căruia îi plac jocurile. Când i se pune o întrebare, el preferă să joace jocuri în loc să răspundă direct. Jian-Jia a întâlnit-o pe prietena sa Mei-Yu și i-a povestit despre rețeaua aeriană din Taiwan. Există n orașe în Taiwan (numerotate 0, ..., $n-1$), dintre care unele sunt conectate prin zboruri directe. Fiecare zbor conectează două orașe și poate fi folosit în ambele direcții.

Mei-Yu l-a întrebat pe Jian-Jia dacă este posibil să călătorească cu avionul între oricare două orașe (fie direct, fie indirect). Jian-Jia nu a vrut să răspundă, dar în schimb, a sugerat să joace un joc. Mei-Yu poate să îi pună întrebări de forma: "Există un zbor *direct* între orașele x și y ? ", și Jian-Jia va răspunde acestor întrebări imediat. Mei-Yu va întreba despre fiecare pereche de orașe o singură dată, punând $r = n(n-1)/2$ întrebări în total. Mei-Yu câștigă jocul dacă, după ce obține răspunsurile la primele i întrebări cu $i < r$, poate afirma cu certitudine dacă rețeaua de zbor este conexă, adică dacă este posibil zborul (direct sau indirect) între oricare două orașe. Altfel, adică dacă are nevoie de toate cele r întrebări, învingător este considerat Jian-Jia.

Pentru a face jocul mai amuzant pentru Jian-Jia, cei doi au convenit că el poate răspunde după cum dorește, fără a ține cont de rețeaua aeriană reală din Taiwan, inventând rețeaua pe măsură ce jocul progresează, alegându-și răspunsurile în funcție de întrebările puse anterior de Mei-Yu. Sarcina ta este să-l ajuți pe Jian-Jia să câștige jocul, decizând, care ar trebui să fie răspunsurile lui.

Exemple

Vom explica regulile jocului prin următoarele trei exemple. Fiecare exemplu are $n = 4$ orașe și $r = 6$ runde de întrebări și răspunsuri.

În primul exemplu (din următorul tabel), Jian-Jia *pierde* deoarece după *runda* 4, Mei-Yu știe cu certitudine că se poate zbura între oricare două orașe, indiferent de răspunsurile lui Jian-Jia la întrebările 5 sau 6.

rundă	întrebare	răspuns
1	0, 1	yes
2	3, 0	yes
3	1, 2	no
4	0, 2	yes
-----	--------	------
5	3, 1	no
6	2, 3	no

În următorul exemplu Mei-Yu poate demonstra după runda a treia, că indiferent cum va răspunde Jian-Jia la întrebările 4, 5, sau 6, *nu se poate* călători între orașele 0 și 1 prin zboruri, deci Jian-Jia pierde din nou.

rundă	întrebare	răspuns
1	0, 3	no
2	2, 0	no
3	0, 1	no
-----	--------	------
4	1, 2	yes
5	1, 3	yes
6	2, 3	yes

În exemplul final Mei-Yu nu poate determina dacă se poate călători între oricare două orașe prin zboruri, până când Jian-Jia nu va răspunde la toate cele șase întrebări, astfel Jian-Jia *câștigă* jocul. Mai exact, deoarece Jian-Jia a răspuns *yes* la ultima întrebare (din următorul tabel), este posibilă călătoria între oricare două orașe. Totuși, dacă Jian-Jia ar fi răspuns *no* la ultima întrebare acest lucru nu ar fi fost posibil.

rundă	întrebare	răspuns
1	0, 3	no
2	1, 0	yes
3	0, 2	no
4	3, 1	yes
5	1, 2	no
6	2, 3	yes

Cerință

Scrieți un program care îl ajută pe Jian-Jia să câștige jocul. Se știe că nici Mei-Yu nici Jian-Jia nu cunosc unul strategia celuilalt. Mei-Yu poate întreba despre perechile de orașe în orice ordine, iar Jian-Jia trebuie să răspundă la ele imediat, fără a cunoaște următoarele întrebări. Voi trebuie să implementați următoarele două funcții.

- `initialize(n)` -- Noi vă vom apela funcția `initialize` o singură dată, la început. Parametrul n reprezintă numărul de orașe.
- `hasEdge(u, v)` -- Apoi vă vom apela funcția `hasEdge` de $r = n(n-1)/2$ ori. Aceste apeluri reprezintă întrebările domnișoarei Mei-Yu în ordinea în care acestea sunt puse. Voi trebuie să răspundeți dacă există sau nu un zbor direct între orașele u și v. Mai exact, valoarea returnată trebuie să fie 1 dacă există un zbor direct și 0 în caz contrar.

Subprobleme

Fiecare subproblemă (subtask) constă din mai multe jocuri. Veți primi puncte pentru o subproblemă dacă programul vostru câștigă toate jocurile pentru Jian-Jia.

subproblemă	puncte	n
1	15	$n = 4$
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Detalii de implementare

Trebuie să încărcați exact un fișier, numit `game.c`, `game.cpp` sau `game.pas`. În acest fișier vor fi implementate cele două funcții descrise mai sus, utilizând următoarele antete.

pentru programele C/C++

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

pentru programele Pascal

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

Grader-ul de pe computerul vostru

Grader-ul de pe computerul vostru citește datele de intrare în următorul format:

- linia 1: `n`
- următoarele r linii: fiecare linie conține două numere întregi `u` și `v` care descriu întrebarea referitoare la orașele u și v.