Game

Jian-Jia è un ragazzo appassionato di giochi. Quando riceve una domanda, preferisce rispondere attraverso un gioco piuttosto che fornire una risposta diretta. Jian-Jia incontra la sua amica Mei-Yu e le parla della rete delle linee aree in Taiwan. Ci sono n città in Taiwan (numerate da 0 a $n-1$), alcune delle quali sono connesse direttamente da voli. Ciascun volo connette due città e può essere preso in entrambe le direzioni.

Mei-Yu chiede a Jian-Jia se ogni coppia di città è collegata (direttamente o indirettamente), oppure no. Jian-Jia non vuole rivelare subito la risposta ma invece propone un gioco. Mei-Yu può chiedergli domande del tipo "le città x e y sono *direttamente* collegate da un volo?", e Jian-Jia risponde immediatamente a tali domande. Mei-Yu, per ogni coppia di città, formula esattamente una volta la domanda per quella coppia, per un totale di $r = n(n-1)/2$ domande. Mei-Yu vince il gioco se, dopo aver ottenuto le risposte per le prime i domande per un qualche $i < r$, può già inferire se la rete di linee aeree è connessa o meno, cioè se ogni coppia di città è collegata o meno (direttamente o indirettamente). Altrimenti, se Mei-Yu ha bisogno di tutte le r domande, il vincitore è Jian-Jia.

Per rendere il gioco più divertente, i due amici concordano che Jian-Jia non deve considerare la vera rete di Taiwan, ma può inventarsi la rete via via che il gioco procede, scegliendo le risposte da fornire in base alle domande formulate da Mei-Yu fino a quel momento. Il tuo compito è quello di aiutare Jian-Jia a vincere, decidendo quali risposte fornire alle domande.

Esempi

Utilizziamo tre esempi per illustrare le regole del gioco. Ciascun esempio ha $n = 4$ città e $r = 6$ iterazioni di domanda/risposta.

Nel primo esempio (nella tabella seguente), Jian-Jia *perde* perché dopo l'iterazione 4, Mei-Yu deduce con certezza che ogni coppia di città è collegata (direttamente o indirettamente), indipendentemente dalle risposte fornite da Jian-Jia alle iterazioni 5 e 6.

iterazione	domanda	risposta
1	0, 1	sì
2	3, 0	sì
3	1, 2	no
4	0, 2	sì
-----	--------	------
5	3, 1	no
6	2, 3	no

Nel secondo esempio, Mei-Yu può dedurre dopo l'iterazione 3 che (indipendemente dalle risposte di Jian-Jia alle iterazioni 4, 5 e 6) le città 0 e 1 *non* sono collegate da voli (direttamente o indirettamente), e così Jian-Jia perde nuovamente.

iterazione	domanda	risposta
1	0, 3	no
2	2, 0	no
3	0, 1	no
-----	--------	------
4	1, 2	sì
5	1, 3	sì
6	2, 3	sì

Nell'ultimo esempio, Mei-Yu non riesce a determinare se ogni coppia di città è collegata da voli (direttamente o indirettamente) fino al momento in cui tutte le 6 iterazioni sono svolte, e quindi Jian-Jia *vince*. Specificatamente, poiché Jian-Jia risponde *sì* nell'ultima iterazione (nella tabella seguente), ogni coppia di città è collegata da voli (direttamente o indirettamente). Se invece rispondesse *no* nell'ultima iterazione, questo non potrebbe accadere.

iterazione	domanda	risposta
1	0, 3	no
2	1, 0	sì
3	0, 2	no
4	3, 1	sì
5	1, 2	no
6	2, 3	sì

Descrizione del problema

Devi scrivere un programma che aiuti Jian-Jia a vincere il gioco. Notare che né Mei-Yu né Jian-Jia conoscono l'una la strategia dell'altro. Mei-Yu può chiedere informazioni sulle coppie di città in qualunque ordine, e Jian-Jia deve rispondere immediatamente senza conoscere le domande future. Devi implementare le seguenti due funzioni.

- `initialize(n)` -- la funzione `initialize` verrà chiamata per prima nell'esecuzione del programma. Il parametro n è il numero di città.
- `hasEdge(u, v)` -- successivamente verrà chiamata `hasEdge` per $r = n(n-1)/2$ volte. Queste chiamate rappresentano le domande di Mei-Yu, nell'ordine in cui vengono poste. Devi rispondere se è presente o meno un volo diretto tra le città u e v. In particolare, il valore restituito deve essere 1 se c'è un volo diretto, 0 altrimenti.

Subtask

Ogni subtask consiste di diverse partite. Prenderai punti per un subtask solo se il tuo programma vince tutte le partite per Jian-Jia.

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

Dettagli di implementazione

Devi sottoporre esattamente un file, di nome `game.c`, `game.cpp` or `game.pas`. Il file deve implementare le procedure descritte sopra utilizzando le intestazioni seguenti.

Linguaggio C/C++

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

Linguaggio Pascal

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

Grader di esempio

Il grader di esempio legge l'input nel formato seguente:

- riga 1: `n`
- seguenti r righe: ognuna contenente due interi `u` e `v` che descrivono la domanda riguardante le città u e v.