Game

Jian-Jia est un jeune garçon très joueur. Lorsqu'on lui pose une question, il préfère jouer à un jeu plutôt que de répondre directement. Jian-Jia a rencontré son amie Mei-Yu et lui a parlé du réseau aérien de Taiwan. Il y a n villes à Taiwan (numérotées 0, ..., $n-1$), certaines étant reliées par des vols. Chaque vol relie deux villes et peut être emprunté dans les deux sens.

Mei-Yu a demandé à Jian-Jia s'il est possible de passer de toute ville à toute autre en voyageant en avion (soit directement, soit indirectement). Jian-Jia n'a pas souhaité révéler la réponse, mais a plutôt proposé de jouer à un jeu. Mei-Yu peut lui poser des questions de la forme "Est-ce que les villes x et y sont *directement* reliées par un vol ?", et Jian-Jia répondra immédiatement à ces questions. Mei-Yu va lui poser la question pour chaque paire de villes exactement une fois, ce qui donne au total $r = n(n-1)/2$ questions. Mei-Yu gagne le jeu si, après avoir obtenu les réponses aux i premières questions pour un certain $i < r$, elle peut déduire si le réseau est connecté ou non, c'est-à-dire s'il est possible ou non de voyager entre toute paire de villes en empruntant des vols (soit directs soit indirects). Sinon, si elle a besoin de toutes les r questions, le gagnant sera Jian-Jia.

Pour que le jeu soit plus amusant pour Jian-Jia, les amis se sont mis d'accord sur le fait que celui-ci peut ignorer la structure du véritable réseau aérien Taiwanais, et inventer son propre réseau au fur et à mesure que le jeu progresse, en choisissant ses réponses en fonction des questions précédentes de Mei-Yu. Votre objectif est d'aider Jian-Jia à gagner le jeu, en décidant comment il devrait répondre aux questions.

Exemples

Nous expliquons les règles du jeu à partir de trois exemples. Chaque exemple a $n = 4$ villes et $r = 6$ tours avec une question et sa réponse.

Dans le premier exemple (la table ci-dessous), Jian-Jia *perd* car après le tour 4, Mei-Yu sait avec certitude que l'on peut voyager entre deux villes quelconques par une suite de vols, quelle que soit la réponse de Jian-Jia aux questions 5 ou 6.

tour	question	réponse
1	0, 1	oui
2	3, 0	oui
3	1, 2	non
4	0, 2	oui
-----	--------	------
5	3, 1	non
6	2, 3	non

Dans l'exemple suivant, Mei-Yu peut prouver après le tour 3 que, quelles que soient les réponses de

Jian-Jia aux questions 4, 5 ou 6, il sera *impossible* de voyager entre les villes 0 et 1 par une succession de vols. Donc Jian-Jia perd à nouveau.

tour	question	réponse
1	0, 3	non
2	2, 0	non
3	0, 1	non
-----	--------	------
4	1, 2	oui
5	1, 3	oui
6	2, 3	oui

Dans l'exemple final, Mei-Yu ne peut pas déterminer s'il est possible de voyager entre deux villes quelconques par une suite de vols tant qu'elle n'a pas obtenu de réponse à ses 6 questions. Jian-Jia *gagne* donc le jeu. Plus précisément, comme Jian-Jia a répondu *oui* à la dernière question (dans la table ci-dessous), on peut déduire qu'il est possible de voyager entre toute paire de villes. Cependant, si au lieu de cela, Jian-Jia avait répondu *non* à la dernière question, alors cela serait impossible.

tour	question	réponse
1	0, 3	non
2	1, 0	oui
3	0, 2	non
4	3, 1	oui
5	1, 2	non
6	2, 3	oui

Tâche

Veuillez écrire un programme qui aide Jian-Jia à gagner le jeu. Notez que ni Mei-Yu ni Jian-Jia ne connaissent la stratégie l'un de l'autre. Mei-Yu peut poser des questions sur des paires de villes dans n'importe-quel ordre, tandis que Jian-Jia doit y répondre immédiatement sans connaître les questions futures. Vous devez implémenter les deux fonctions suivantes.

- `initialize(n)` -- Nous appellerons votre fonction `initialize` en premier. Le paramètre n est le nombre de villes.
- `hasEdge(u, v)` -- Nous appellerons ensuite `hasEdge` $r = n(n-1)/2$ fois. Ces appels représentent les questions de Mei-Yu, dans l'ordre dans lequel elle les pose. Vous devez répondre s'il y a un vol direct entre les villes u et v ou pas. Plus précisément, la valeur de retour doit être 1 s'il y a un vol direct, et 0 sinon.

Sous-tâches

Chaque sous-tâche est constituée de plusieurs jeux, Vous n'aurez les points d'une sous-tâche que si

votre programme gagne tous les jeux pour Jian-Jia.

sous-tâche	points	n
1	15	$n = 4$
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Détails d'implémentation

Vous devez soumettre un seul fichier, appelé `game.c`, `game.cpp` ou `game.pas`. Ce fichier implémente les fonctions décrites plus haut en utilisant les signatures suivantes.

Programmes C/C++

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

Programmes Pascal

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

Évaluateur

L'évaluateur fourni lit l'entrée au format suivant :

- ligne 1 : `n`
- les r lignes suivantes : chaque ligne contient deux entiers `u` et `v` qui décrivent une question concernant les villes u et v.