Grantige Trainer

An einem schönen Mittwochabend sitzen Flo, Gary und Wolfgang beisammen und sind auf der Suche nach Beispielen für den Wettbewerb am Donnerstag. Es wird immer später und später, und die drei werden immer müder und grantiger. Am Ende sind sie so grantig, dass sie es den Teilnehmern möglichst schwer machen wollen.

Bei einer Aufgabe geht es um ein Spiel aus dem Fernen Japan. In Japan gibt es n Städte (da die japanischen Namen der Städte etwas schwer zu lesen sind, sind sie von 0 bis $n-1$ durchnummeriert), und zwischen manchen dieser Städte verkehren Linienflüge. Einen Linienflug kann man jeweils in beide Richtungen benutzen.

Der Spieler muss nun herausfinden, ob man von jeder der n Städte jede andere erreichen kann (mit einem oder mehreren Flügen), oder ob es Städte gibt, die man von manchen anderen Städten aus nicht mit dem Flugzeug erreichen kann. Dazu kann der Spieler mehrere Fragen stellen. Jede Frage hat die Form „Ist Stadt x durch einen Direktflug mit Stadt y verbunden?", und ein Computerprogramm antwortet automatisch.

Der Spieler fragt nach jeder der $r = n(n-1)/2$ möglichen Flugverbindungen genau einmal. Er gewinnt das Spiel, wenn er bereits nach $r-1$ oder weniger Fragen weiß, ob jedes Paar von Städten miteinander durch Flüge verbunden ist, oder nicht. Weiß er es erst nach der letzten Frage, so hat er verloren.

Um es den Teilnehmern möglichst schwer zu machen, beschließen die übermüdeten, grantigen Trainer, am Wettbewerbs-Server ein Grader-Programm zu installieren, das das Flugnetzwerk erst während des Spiels anhand der Fragen des Spielers erfindet, um es dem Spieler damit möglichst schwer zu machen.

Dann beschließen sie aber, dass es schon viel zu spät in der Nacht ist und die Entwicklung dieses bösartigen Graders um diese Uhrzeit jetzt viel zu mühsam wäre. Jetzt lautet die Aufgabe für die Teilnehmer plötzlich, das Schreiben dieses bösartigen Grader-Programms zu übernehmen.

Beispiel

Es folgen drei Beispiele, um die Spielregeln besser zu erklären. Jedes der Beispiele hat $n = 4$ Städte und daher $r = 6$ Runden.

Im ersten Spiel verliert das bösartige Computerprogramm, da der Spieler nach Runde 4 bereits weiß, dass man zwischen allen Städten mit dem Flugzeug reisen kann:

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 zweiten Spiel kann der Spieler nach Runde 3 beweisen, dass, egal wie der Computer auf

Fragen 4, 5 und 6 antwortet, man nicht mit dem Flugzeug von 0 nach 1 reisen kann, also verliert das bösartige Computerprogramm schon wieder.

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 dritten und letzten Beispiel kann der Spieler nicht feststellen, ob man zwischen allen Städten reisen kann, bis alle 6 Fragen beantwortet wurden, also *gewinnt* das Computerprogramm. Da die Antwort auf die letzte Frage (in der folgenden Tabelle) „ja" war, kann man von jeder Stadt jede andere erreichen. Wäre die Antwort aber „nein" gewesen, so wäre dies 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

Da die Trainer inzwischen eingeschlafen sind, ist es nun Deine Aufgabe, das bösartige Grader-Programm zu schreiben. Dazu musst Du folgende zwei Funktionen implementieren:

- `void initialize(int n)`

  Diese Funktion wird bei Programmstart aufgerufen. Der Parameter n ist die Anzahl der Städte.

- `bool hasEdge(int u, int v)`

  Als nächstes wird die Funktion `hasEdge` r-mal aufgerufen, einmal für jede Frage, die der Spieler stellt. Die Antwort soll `true` sein, wenn es einen direkten Flug zwischen den Städten u und v gibt.

Subtasks

Subtask 1 (15 Punkte): $n = 4$

Subtask 2 (27 Punkte): $4 \leq n \leq 80$

Subtask 3 (58 Punkte): $4 \leq n \leq 1500$

Jeder Subtask besteht aus mehreren Spielen. Du bekommst die Punkte für einen Subtask nur, wenn das bösartige Grader-Programm alle Spiele gewinnt.

Beschränkungen

Zeitlimit: 1.5 s **Speicherlimit:** 256 MB