# Game (Juego)

Jian-Jia es un joven al que le gusta jugar a los acertijos. Cuando le hacen una pregunta, prefiere jugar a contestar de manera directa. Un día Jian-Jia está platicando con su amiga Mei-Yu sobre las líneas aéreas de Taiwan. Hay $n$ ciudades en Taiwan (las cuales se numeran 0, ..., $n-1$), algunas de las cuales están conectadas por vuelos directos. Cada vuelo conecta dos ciudades y puede tomarse en cualquier dirección.

Mei-Yu le pregunta a Jian-Jia si es posible viajar por avión entre cualquier pareja de ciudades (puede ser de forma directa o con escalas). Jian-Jia no quiere dar una respuesta directa, sugiere jugar un juego. Mei-Yu le puede hacer preguntas como la que sigue: "¿Están las ciudades $x$ y $y$ conectadas de manera *directa* por un vuelo?" a estas preguntas Jian-Jia debe contestar de manera inmediata. Mei-Yu va a preguntarle por todas las parejas posibles de ciudades, exactamente una vez por cada pareja, en total $r = n(n-1)/2$ preguntas. Mei-Yu ganará el juego si después de haber hecho las primeras $i$ preguntas para una $i < r$, ella puede inferir si la red es conexa, es decir, si es posible viajar entre cualquier pareja de ciudades mediante vuelos (ya sea directamente o con escalas). Por el contrario, si requiere de las $r$ preguntas para determinarlo, el ganador será Jian-Jia.

Para hacer el juego más divertido, ambos amigos acuerdan ignorar la red de vuelos real de Taiwan y deciden que Jian-Jia puede contestar las preguntas como él quiera decidiendo sus respuestas en base a lo que ha preguntado Mei-Yu. Debes ayudar a Jian-Jia a ganar el juego, decidiendo de qué forma debe contestar las preguntas.

# Ejemplos

Explicaremos el juego con tres ejemplos. En cada ejemplo hay $n = 4$ ciudades y $r = 6$ preguntas para contestar.

En el primer ejemplo (siguiente tabla), Jian-Jia *pierde* porque tras 4 pregutnas, Mei-Yu sabe sin lugar a dudas que es posible viajar entre cualesquiera dos ciudades. No importa de qué forma conteste Jian-Jia las preguntas 5 y 6.

| ronda | pregunta | respuesta |
|---|---|---|
| 1 | 0, 1 | si |
| 2 | 3, 0 | si |
| 3 | 1, 2 | no |
| 4 | 0, 2 | si |
| ----- | -------- | ------ |
| 5 | 3, 1 | no |
| 6 | 2, 3 | no |

En el siguiente ejemplo Mei-Yu está segura, después de 3 preguntas, sin importar como Jian-Jia

conteste las preguntas 4, 5 y 6 que *no es posible* viajar entre las ciudades 0 y 1 mediante vuelos. Por lo tanto Jian-Jia pierde de nuevo.

| ronda | pregunta | respuesta |
|---|---|---|
| 1 | 0, 3 | no |
| 2 | 2, 0 | no |
| 3 | 0, 1 | no |
| ----- | -------- | ------ |
| 4 | 1, 2 | si |
| 5 | 1, 3 | si |
| 6 | 2, 3 | si |

En el último ejemplo Mei-Yu no puede determinar si es posible viajar entre cualquier pareja de ciudades hasta que se contestan las seis preguntas, en este ejemplo Jian-Jia *gana* el juego. Específicamente, debido a que Jian-Jia contestó *si* a la última pregunta (en la tabla que sigue), es posible viajar entre cualquier pareja de ciudades. Sin embargo, si Jian-Jia hubiera contestado *no* a la última pregunta entonces hubiera sido imposible.

| ronda | pregunta | respuesta |
|---|---|---|
| 1 | 0, 3 | no |
| 2 | 1, 0 | si |
| 3 | 0, 2 | no |
| 4 | 3, 1 | si |
| 5 | 1, 2 | no |
| 6 | 2, 3 | si |

# Problema

Escribe un programa que le ayude a Jian-Jia a ganar el juego. Observa que ni Mei-Yu ni Jian-Jia conocen la estrategia del otro. Mei-Yu puede preguntar por las parejas de ciudades en cualquier orden y Jian-Jia debe contestar de manera inmediata, sin saber cuales serán las siguientes preguntas. Debes implementar las siguientes dos funciones.

- `initialize(n)` -- Se ejecutará tu función `initialize` al inicio. El parámetro $n$ representa el número de ciudades.
- `hasEdge(u, v)` -- Después se llamará `hasEdge` un número $r = n(n-1)/2$ de veces. Estas llamadas representan las preguntas de Mei-Yu en el orden en el que ella las hace. Tu programa debe contestar si existe un vuelo directo entre las ciudades $u$ y $v$. Específicamente, el valor de retorno debe ser 1 si hay un vuelo directo o 0 si no es así.

# Subproblemas

Cada subproblema consiste de varios juegos. Tu programa obtendrá puntos para un subproblema sólo

si tu programa gana todos los juegos para Jian-Jia.

| subproblema | puntos | $n$ |
|---|---|---|
| 1 | 15 | $n = 4$ |
| 2 | 27 | $4 \leq n \leq 80$ |
| 3 | 58 | $4 \leq n \leq 1500$ |

# Detalles de implementación

Debes enviar exactamente un archivo llamado `game.c`, `game.cpp` o `game.pas`. Este archivo debe contener la implementación de las funciones arriba descritas de acuerdo a los siguientes prototipos.

## programas en C/C++

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

## programas en Pascal

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

## Evaluador de prueba

El evaluador de prueba espera una entrada en el siguiente formato:

- línea 1: `n`
- siguientes $r$ líneas: cada línea contiene dos enteros `u` y `v` que describen una pregunta correspondiente a las ciudades $u$ y $v$ respectivamente.