Spēle

Jian-Jia ir jauns puisis, kuram patīk spēlēt spēles. Kad viņam uzdod jautājumu, viņam labāk patīk atbildēt netieši nekā sniegt precīzu atbildi. Jian-Jia satika draudzeni Mei-Yu un izstāstīja viņai par Taivānas lidojumu tīklu. Taivānā ir n pilsētas (sanumurētas 0, ..., $n-1$), dažas no tām ir savienotas ar lidojumiem. Katrs no lidojumiem savieno tieši divas pilsētas un var tikt izmantots jebkurā no virzieniem.

Mei-Yu prasīja Jian-Jia-m, vai, izmantojot lidojumus, ir iespējams nokļūt no jebkuras uz jebkuru pilsētu (ar tiešo lidojumu vai ar pārsēšanos). Jian-Jia nevēlas uzreiz atbildēt, bet ierosina spēlēt spēli. Mei-Yu var viņam uzdot jautājumus formā "Vai pilsētas x un y savieno *tiešs lidojums*?", un Jian-Jia uzreiz atbildēs. Mei-Yu par katru pilsētu pāri prasīs tieši vienu reizi, prasot pavisam $r = n(n-1)/2$ jautājumus. Mei-Yu uzvar spēlē, ja pēc pirmo i jautājumu uzprasīšanas, kur $i < r$, viņa var izsecināt, vai lidojumu tīkls ir saistīts vai nav - t.i., ar lidojumiem ir iespējams nokļūt no katras uz katru pilsētu (ar tiešo lidojumu vai ar pārsēšanos) vai nav. Pretējā gadījumā - t.i., ja viņai vajag izmantot visus r jautājumus, tad uzvar Jian-Jia.

Lai spēle Jian-Jia-am būtu interesantāka, draugi nolēma, ka viņš spēles gaitā var aizmirst īsto Taivānas lidojumu tīklu un, ņemot vērā Mei-Yu uzdotos jautājumus, izdomāt lidojumu tīklu spēles gaitā un sniegt atbildes par to. Jūsu uzdevums ir palīdzēt Jian-Jia-am uzvarēt spēlē, izdomājot atbildes uz jautājumiem.

Piemēri

Spēles noteikumi tiks izskaidroti ar trim piemēriem. Katrā piemērā ir $n = 4$ pilsētas un $r = 6$ jautājumu un atbilžu kārtas.

Pirmajā piemērā (sekojošajā tabulā), Jian-Jia *zaudē*, jo pēc ceturtās kārtas Mei-Yu droši zina, ka starp katrām divām pilsētām ir iespējams aizlidot, neatkarīgi no tā, kādas būs Jian-Jia atbildes uz piekto un sesto jautājumu.

kārta	jautājums	atbilde
1	0, 1	jā
2	3, 0	jā
3	1, 2	nē
4	0, 2	jā
-----	--------	------
5	3, 1	nē
6	2, 3	nē

Nākamajā piemērā Mei-Yu pēc trešās kārtas var pierādīt, ka neatkarīgi no nākamajām Jian-Jia atbildēm uz atlikušajiem jautājumiem no nultās uz pirmo pilsētu *nebūs iespējams* nokļūt, tātad Jian-Jia atkal zaudē.

kārta	jautājums	atbilde
1	0, 3	nē
2	2, 0	nē
3	0, 1	nē
-----	--------	------
4	1, 2	jā
5	1, 3	jā
6	2, 3	jā

Pēdējā piemērā Mei-Yu nevar noteikt, vai no katras uz katru pilsētu iespējams aizlidot, neuzdodot visus sešus jautājumus, tāpēc Jian-Jia *uzvar*. Tieši tāpēc, ka Jian-Jia atbildēja ar *jā* uz pēdējo jautājumu, varēja noskaidrot, ka ir iespējams ar lidojumiem nokļūt no jebkuras pilsētas jebkurā citā. Taču, ja Jian-Jia uz pēdējo jautājumu būtu atbildējis *nē*, tad nebūtu iespējams nokļūt no jebkuras pilsētas jebkurā citā, izmantojot lidojumus.

kārta	jautājums	atbilde
1	0, 3	nē
2	1, 0	jā
3	0, 2	nē
4	3, 1	jā
5	1, 2	nē
6	2, 3	jā

Uzdevums

Lūdzu uzrakstiet programmu, kas palīdzētu Jian-Jia-am uzvarēt spēlē. Ievērojiet, ka ne Mei-Yu, ne Jian-Jia nezina viens otra stratēģiju. Mei-Yu var prasīt jautājumus par pilsētu pāriem jebkurā secībā un Jian-Jia jāatbild uz tiem nekavējoties, nesagaidot nākamos jautājumus. Jums jārealizē sekojošas divas funkcijas:

- `initialize(n)` -- Jūsu `initialize` tiks izsaukta pašā sākumā ar parametru n, kas apraksta pilsētu skaitu.
- `hasEdge(u, v)` -- Pēc tam $r = n(n-1)/2$ reizes tiks izsaukta funkcija `hasEdge`. Šie izsaukumi atbildīs Mei-Yu uzdotajiem jautājiem to izdarīšanas secībā. Jums jāatbild, vai starp pilsētām u un v ir tiešais lidojums. Precīzāk, atgrieziet 1, ja ir tiešais lidojums, bet citādāk - 0.

Apakšuzdevumi

Katrs apakšuzdevums sastāv no vairākām spēlēm. Par apakšuzdevumu Jūs saņemsiet punktus tikai tad, ja Jūsu programma uzvar visas spēles par labu Jian-Jia-am.

apakšuzdevums	punkti	n
1	15	$n = 4$
2	27	$4 \leq n \leq 80$
3	58	$4 \leq n \leq 1500$

Realizācijas detaļas

Jums jāiesūta tieši viens fails ar nosaukumu `game.c`, `game.cpp` vai `game.pas`. Šajā failā jārealizē apakšprogrammas, kas aprakstītas iepriekš ar norādīto signatūru.

C/C++ programma

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

Pascal programma

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

Paraugtestētājs

Paraugtestētājs lasa ievadu šādā formātā:

- rinda 1: `n`
- nākamās r rindas: katra rinda satur divus veselus skaitļus `u` un `v`, kas apraksta jautājumu par pilsētām u un v.