Toy Design

Nome	ToyDesign
File di input	Problema interattivo
File di output	Problema interattivo
Limite di tempo	1 secondo
Limite di memoria	256 megaottetti

Lavori per un'azienda che progetta giocattoli. Quando un gioco viene creato, il prototipo funziona così: ci sono n *pin*, numerati da 1 a n, che escono dal gioco; alcune coppie di questi sono connessi con dei cavi all'interno del giocattolo. In altre parole, i pin e i cavi formano un grafo non diretto, dove i pin sono i nodi e i cavi sono gli archi.

I cavi non sono visibili dall'esterno, e l'unico modo per capire qualcosa su di loro è di usare un **tester** sui pin: possiamo prendere due pin i e j (con $i \neq j$), e il tester dice se questi sono connessi all'interno del giocattolo, direttamente o indirettamente. Quindi il tester dice se c'è un cammino tra questa coppia di nodi nel grafo.

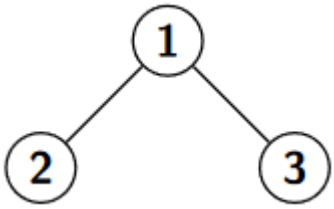
Chiamiamo l'insieme delle connessioni all'interno del giocattolo: **progetto** del giocattolo.

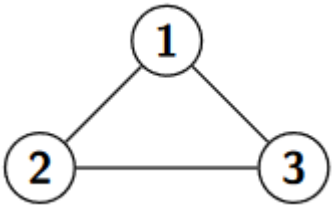
Stai usando un particolare software per studiare e creare questi progetti. Questo software funziona così: si inizia con un qualche progetto del giocattolo, chiamato "progetto 0". Non puoi vedere le connessioni interne, ma puoi eseguire ripetutamente la seguente operazione in tre passaggi:

1. Scegli a, il numero di un progetto, e i e j, i numeri di due pin (tali che $i \neq j$).
2. Il software ti dice cosa succederebbe se usassimo il tester su questi due pin. In altre parole, ti dice se i pin i e j sono connessi (direttamente o indirettamente) all'interno del progetto a.
3. In più, se i due pin non sono connessi (direttamente o indirettamente) nel progetto a, viene creato un nuovo progetto che ha tutte le connessioni del progetto a, più una connessione aggiuntiva tra i e j. A questo progetto viene assegnato il primo numero di progetto libero. Quindi il primo progetto creato avrà numero 1, poi 2, e così via. Nota che questa operazione non modifica il progetto a, ma semplicemente crea un nuovo progetto con la connessione aggiuntiva.

Il tuo obiettivo è di capire più informazioni possibili riguardo al progetto 0.

Nota che non è sempre possibili determinare l'esatto insieme di connessioni del progetto 0, perché non c'è modo di distinguere le connessioni dirette e indirette. Per esempio, considera i seguenti progetti con $n = 3$:





Per ciascuna coppia di pin, in entrambi i progetti il tester dirà che sono connessi, quindi non c'è modo di distinguere i due progetti.

Il tuo obiettivo è di trovare un progetto equivalente al progetto 0. Due progetti sono **equivalenti** se il tester produce lo stesso risultato su entrambi i progetti per ogni coppia di pin.

Implementazione

Questo è un problema interattivo. Devi implementare la funzione

```
void ToyDesign(int n, int max_ops);
```

che cerca un progetto che sia *equivalente* al progetto 0. Per farlo può chiamare la funzione:

```
int Connected(int a, int i, int j);
```

dove $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$ e a non può superare il numero di progetti creati fino a quel momento. Se i pin i e j sono connessi (direttamente o indirettamente) nel progetto a, questa funzione restituirà a. Altrimenti restituirà il numero di progetti creati, più uno. Questo sarà il numero del nuovo progetto creato (che contiene tutte le connessioni di a, più la connessione tra i e j). La funzione `Connected` può essere chiamata al più `max_ops` volte.

Al termine dell'esecuzione, per descrivere un progetto equivalente al progetto 0, devi chiamare questa funzione:

```
void DescribeDesign(std::vector<std::pair<int,int>> result);
```

Il parametro `result` è un vettore di coppie di interi che descrivono le connessioni tra le coppie dei pin. Ogni coppia corrisponde ad una connessione e deve contenere i numeri dei pin collegati. Deve esserci al più una connessione tra ogni coppia (non ordinata) di pin, e nessuna connessione tra un pin e sé stesso. Chiamare questa funzione terminerà l'esecuzione del programma.

Assunzioni

- $2 \leq n \leq 200$.

Sottoproblemi

- Subtask 1 (10 punti): $n \leq 200$, *max_ops* $= 20\,000$.
- Subtask 2 (20 punti): $n \leq 8$, *max_ops* $= 20$.
- Subtask 3 (35 punti): $n \leq 200$, *max_ops* $= 2\,000$.
- Subtask 4 (35 punti): $n \leq 200$, *max_ops* $= 1\,350$.

Interazione di esempio

Azione della soluzione	Azione del grader	Spiegazione
	`ToyDesign(4, 20)`	Il giocattolo ha 4 pin. Devi determinare un progetto equivalente al progetto 0 chiamando `Connected` al più 20 volte.
`Connected(0, 1, 2)`	`Restituisce 1.`	I pin 1 e 2 non sono connessi (direttamente o indirettamente) nel progetto 0. Il progetto 1 viene creato.
`Connected(1, 3, 2)`	`Restituisce 2.`	I pin 3 e 2 non sono connessi (direttamente o indirettamente) nel progetto 1. Il progetto 2 viene creato.
`Connected(0, 3, 4)`	`Restituisce 0.`	I pin 3 e 4 sono connessi (direttamente o indirettamente) nel progetto 0. Nessun progetto nuovo viene creato.
`DescribeDesign({{3, 4}})`	-	Descriviamo un progetto con una sola connessione: pin 3 e 4.

Grader di esempio

Il grader d'esempio fornito, `grader.cpp`, all'interno dell'allegato `ToyDesign.zip`, legge l'input dallo standard input nel seguente formato:

- La prima riga contiene il numero di nodi n, il numero di archi m, e *max_ops*
- Le seguenti m righe contengono gli archi come coppie di nodi.

Il grader di esempio legge l'input e chiama `ToyDesign`. Il grader di esempio stamperà uno dei seguenti messaggi, in base al comportamento della tua soluzione:

- *"Wrong answer: Number of operations exceeds the limit."* se il numero di chiamate a `Connected` supera *max_ops*.
- *"Wrong answer: Wrong design id."* se il parametro a di una chiamata a `Connected` è il numero di un progetto non ancora esistente nel momento della chiamata alla funzione.
- *"Wrong answer: Incorrect design."* se il progetto descritto con `DescribeDesign` non è equivalente al progetto 0.
- *"OK!"* se il progetto descritto da `DescribeDesign` è equivalente al progetto 0.

Per compilare il grader d'esempio con la tua soluzione, puoi usare questo comando nel terminale:

```
g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp
```

dove `solution.cpp` è il file della tua soluzione che invierai su CMS. Per eseguire il programma con l'input di esempio allegato, esegui questo comando nel terminale:

```
./solution < input.txt
```