სათამაშოს კონსტრუქცია

Problem Name	ToyDesign
Input File	Interactive Task
Output File	Interactive Task
Time limit	1 second
Memory limit	256 megabytes

თქვენ მუშაობთ კომპანიაში, რომელიც ჰქმნის სათამაშოებს. ახალი სათამაშო მუშაობს შემდეგი პრინციპით: ყუთზე დამაგრებულია n ქინძისთავი, რომლებიც გადანომრილია 1-დან n-მდე. ქინძისთავების ზოგიერთი წყვილი დაკავშირებულია სადენებით ყუთის შიგნით (სხვა სიტყვებით, ქინძისთავები და სადენები ჰქმნიან არაორიენტირებულ გრაფს, სადაც ქინძისთავები წვეროებია, ხოლო სადენები - წიბოები). სადენები გარედან არ ჩანს და ერთადერთი საშუალება მათ შესახებ რაიმე ინფორმაცია მივიღოთ არის **ტესტერის** გამოყენება ქინძისთავებზე: ჩვენ შეგვიძლია ავირჩიოთ ორი i და j ისეთი ქინძისთავი, სადაც $i \neq j$, და ტესტერი გვეტყვის, არის თუ არა დაკავშირებული ერთმანეთთან ეს ორი ქინძისთავი ყუთის შიგნით უშუალოდ ან სხვების გავლით (სხვაგვარად რომ ვთქვათ, ტესტერი შეგვატყობინებს, არსებობს თუ არა გზა ორ წვეროს შორის გრაფში).

კავშირების სიმრავლეს ყუთის შიგნით ვუწოდოთ სათამაშოს **კონსტრუქცია**.

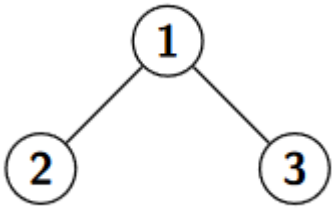
თქვენ იყენებთ სპეციალურ პროგრამულ უზრუნველყოფას შეკითხვებისათვის და კონსტრუქციის შესაქმნელად. ეს პროგრამული უზრუნველყოფა მუშაობს შემდეგნაირად: ის იწყებს სათამაშოს რომელიღაც კონსტრუქციიდან, რომელსაც თქვენ უწოდებთ "კონსტრუქცია 0"-ს. ის არ გვიჩვენებს შეერთებებს ამ კონსტრუქციისთვის ყუთის შიგნით. ამის ნაცვლად, თქვენ მრავალჯერადად შეგიძლიათ შეასრულოთ შემდეგი ოპერაციები:

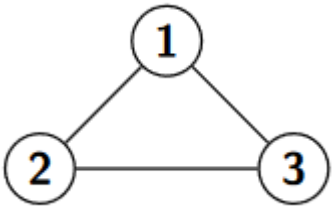
- თქვენ ირჩევთ კონსტრუქციის a ნომერს და ორ i და j ნომრის მქონე ისეთ ქინძისთავს, სადაც $i \neq j$.
- პროგრამული უზრუნველყოფა გვატყობინებს, თუ რა მოხდება ტესტერის გამოყენებისას ამ ორ ქინძისთავზე. სხვა სიტყვებით, ის შეგვატყობინებს, არიან თუ არა i და j ქინძისთავები დაკავშირებული (უშუალოდ ან სხვების გავლით) a ნომერ კონსტრუქციაში.
- გარდა ამისა, თუ ქინძისთავები უშუალოდ ან სხვების გავლით არ არიან დაკავშირებული a კონსტრუქციაში, იქმნება ახალი კონსტრუქცია, რომელსაც აქვს ყველა კავშირი a

კონსტრუქციიდან და პლიუს ერთი დამატებითი პირდაპირი კავშირი i და j წვეროებს შორის. ამ კონსტრუქციას მიენიჭება კონსტრუქციათა მორიგი, ჯერ არგამოყენებული ნომერი. (ანუ ამგვარად შექმნილ პირველ კონსტრუქციას მიენიჭება ნომერი 1, შემდეგ ნომერი 2 და ა. შ.). შევნიშნოთ, რომ ამ დროს არ იცვლება კონსტრუქცია a, არამედ უბრალოდ იქმნება ახალი კონსტრუქცია დამატებითი კავშირით.

თქვენი მიზანია, რაც შეიძლება მეტი გაიგოთ ინსტრუქცია 0-ის შესახებ ამ ოპერაციით.

მიაქციეთ ყურადღება, რომ ყოველთვის არაა შესაძლებელი განვსაზღვროთ კავშირების ზუსტი სიმრავლე კონსტრუქცია 0-სათვის, რადგან შეუძლებელია განვასხვავოთ უშუალო და სხვების გავლით არსებული კავშირები. მაგალითად, განვიხილოთ შემდეგი ორი კონსტრუქცია $n = 3$-სათვის:





ტესტერი ორივე შემთხვევაში შეგატყობინებთ, რომ ქინძისთავების ყველა წყვილი შეერთებულია, ამიტომ ვერ შევძლებთ, რომ ისინი განვასხვავოთ ზემოთ აღწერილი პროგრამული უზრუნველყოფით.

თქვენი ამოცანაა განსაზღვროთ ნებისმიერი კონსტრუქცია, რომელიც კონსტრუქცია 0-ის ექვივალენტურია. ორი კონსტრუქცია **ექვივალენტურია**, თუ ტესტერი ორივე კონსტრუქციაში იძლევა ერთსა და იმავე შედეგს ქინძისთავთა ნებისმიერი წყვილისათვის.

იმპლემენტაცია

ეს ინტერაქტიული ამოცანაა. თქვენ უნდა მოახდინოთ შემდეგი ფუნქციის იმპლემენტაცია:

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

რომელიც განსაზღვრავს კონსტრუქცია 0-ის *ექვივალენტურ* კონსტრუქციას. თქვენმა იმპლემენტაციამ უნდა მიაღწიოს ამ მიზანს ორი კონკრეტული ოპერაციის 0-ჯერ ან მეტჯერ შესრულებით. ერთ ოპერაციაზე თქვენმა პროგრამამ უნდა გამოიძახოს ფუნქცია:

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

სადაც $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, და a-მ არ უნდა გადააჭარბოს მოცემული მომენტისთვის შემნილ კონსტრუქციათა რაოდენობას. თუ i და j ქინძისთავები დაკავშირებული არიან უშუალოდ ან სხვების გავლით a კონსტრუქციაში, მაშინ ის დააბრუნებს a-ს. წინააღმდეგ შემთხვევაში დაბრუნდება მოცემული მომენტისთვის შექმნილი კონსტრუქციათა რაოდენობა პლიუს 1, რომელიც გახდება

ახალი კონსტრუქციისთვის მინიჭებული ნომერი. ამ ახალ კონსტრუქციაში იარსებებს ყველა კავშირი, რომელიც არსებობდა a კონსტრუქციაში პლიუს კავშირი i-სა და j-ს შორის. ფუნქცია `Connected` შეიძლება გამოძახებულ იქნას არაუმეტეს `max_ops`-ჯერ.

როცა თქვენი პროგრამა დაასრულებს მუშაობას `Connected` ოპერაციებთან, მან უნდა აღწეროს კონსტრუქცია 0-ის ექვივალენტური სქემა და კორექტულად დაასრულოს მუშაობა. კონსტრუქციის აღწერისათვის პროგრამამ უნდა გამოიძახოს: `void DescribeDesign(std::vector<std::pair<int,int>> result);`

`result`-ის პარამეტრი უნდა იყოს მთელი რიცხვების წყვილებისაგან შედგენილი ვექტორი, რომელიც აღწერს უშუალო კავშირებს ქინძისთავებს შორის. ყოველი წყვილი შეესაბამება ერთ კავშირს და უნდა შეიცავდეს ორი დაკავშირებული ქინძისთავის ნომრებს. ქინძისთავების ყოველ (დაულაგებელ) წყვილს შორის უნდა იყოს მხოლოდ ერთი უშუალო კავშირი და არ უნდა არსებობდეს ქინძისთავის უშუალო კავშირი საკუთარ თავთან.

შეზღუდვები

- $2 \leq n \leq 200$

შეფასება

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

Sample Interaction

Contestant action	Grader action	Explanation
	`ToyDesign(4, 5)`	სათამაშოში 4 ქინძისთავია. თქვენ უნდა განსაზღვროთ კონსტრუქცია 0-ის ექვივალენტური კონსტრუქცია ფუნქცია `Connected`-ის არაუმეტეს 5 გამოძახებით.
`Connect(0, 1, 2)`	`Returns 1.`	1 და 2 ქინძისთავები არ არიან დაკავშირებული უშუალოდ ან სხვების გავლით კონსტრუქცია 0-ში. იქმნება ახალი კონსტრუქცია 1.
`Connected(1, 3, 2)`	`Returns 2.`	3 და 2 ქინძისთავები არ

		არიან დაკავშირებული უშუალოდ ან სხვების გავლით კონსტრუქცია 1-ში. ახალი კონსტრუქცია არ იქმნება.
`Connected(0, 3, 4)`	`Returns 0.`	3 და 4 ქინძისთავები არიან დაკავშირებული უშუალოდ ან სხვების გავლით კონსტრუქცია 0-ში. იქმნება ახალი კონსტრუქცია 2.
`DescribeDesign({{3, 4}})`	-	ჩვენ აღვწერთ კონსტრუქციას, რომელსაც აქვს მხოლოდ ერთი კავშირი: ქინძისთავები 3 და 4.

სანიმუშო გრადერი

წარმოდგენილი სანიმუშო გრადერი, `grader.cpp`, მოცემულია ამოცანის ბმულში `ToyDesign.zip`. იგი კითხულობს სტანდარტულ შემოტანას შემდეგი ფორმატით:

- პირველი სტრიქონი შეიცავს წვეროების რაოდენობა n-ს, წიბოების რაოდენობა m-ს და *max_ops*-ს.
- მომდევნო m სტრიქონი შეიცავს წვეროთა წყვილებით წარმოდგენილ წიბოებს.

სანიმუშო გრადერი კითხულობს შესატან მონაცემებს და იძახებს `ToyDesign` ფუნქციას მომხმარებლის ამოხსნაში.

სანიმუშო გრადერის თქვენს პროგრამასთან ერთად დასაკომპილირებლად თქვენ უნდა გაუშვათ ბრძანება ტერმინალის ბრძანებათა სტრიქონიდან:

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

სადაც `solution.cpp` თქვენი ამოხსნაა, რომელიც CMS-ზე უნდა გაუშვათ. იმისათვის, რომ გაუშვათ პროგრამა შესატანი მონაცემების მაგალითთან ერთად, ტერმინალიდან გაუშვით:

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