Проектиране на играчки

Име на задачата	Проектиране на играчки
Вход	Интерактивна задача
Изход	Интерактивна задача
Ограничение по време	1 секунда
Ограничение по памет	256 MB

Вие работите за компания, която проектира играчки. Нова играчка, която се създава, работи по следния начин: има n топлийки, номерирани от 1 до n, стърчащи от кутия. Някои двойки топлийки са свързани с проводници вътре в кутията. (С други думи, топлийките и проводниците образуват неориентиран граф, където топлийките са върховете, а проводниците са ребрата.) Проводниците не се виждат отвън и единственият начин да разберете нещо за тях е да използвате **тестер** на топлийки: Можем да изберем две топлийки i и j, така че $i \neq j$, и тестерът ще каже дали тези две топлийки са свързани вътре в кутията, директно или индиректно. (Така тестерът казва дали има път между тези топлийки в графа.)

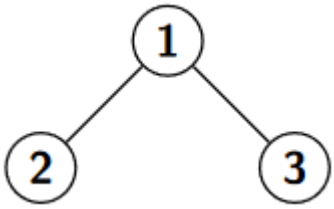
Ще наречем набора от връзки вътре в кутията **проект** на играчката.

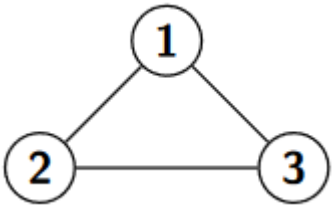
Използвате специализиран софтуер за запитване и създаване на тези проекти. Този софтуер работи по следния начин: Започва с някакъв проект на играчката, който обозначаваме като "проект 0". Не ви показва връзките вътре в кутията за този проект. Вместо това можете многократно да извършвате следната 3 стъпкова операция:

1. Избирате номер на проект a и два номера на топлийки i и j, така че $i \neq j$.
2. Софтуерът ви казва какво ще се случи, ако използваме тестера на тези две топлийки. С други думи, той ви казва дали топлийките i и j са (директно или косвено) свързани в проект a.
3. Освен това, ако топлийките не са били пряко или непряко свързани в проекта a, тогава се създава нов проект, който има всички връзки от проекта a плюс една допълнителна директна връзка между i и j. Този проект получава следващия наличен номер на проект. (Така че първият проект, създаден по този начин, ще има номер 1, след това номер 2 и т.н.) Имайте предвид, че това не променя проекта a, а просто създава нов проект, който има допълнителна връзка.

Вашата цел е да научите възможно най-много за проект 0, като използвате тази операция.

Имайте предвид, че не винаги е възможно да се определи точният набор от връзки за проект 0, тъй като няма начин да се разграничат преки и непреки връзки. Например, разгледайте следните два проекта с $n = 3$:





Тестерът ще докладва всеки чифт топлийки като свързани и за двата проекта, така че няма да можем да ги различим с помощта на описания по-горе софтуер.

Вашата цел е да определите всеки проект, който е еквивалентен на проект 0. Два проекта са **еквивалентни**, ако тестерът отчете един и същ резултат и в двата проекта за всички двойки топлийки.

Имплементация

Това е една интерактивна задача. Вие трябва да разработите функцията

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

която определя проект, който е *еквивалентен* на проект 0. Вашата имплементация трябва да постигне тази цел чрез извикване на две функции, както е описано по-долу. Първата функция, която можете да извикате, е:

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

където $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, и a не трябва да надвишава броя на проектите, създадени до момента. Ако топлийките i и j са (пряко или непряко) свързани в проект a, тогава той ще върне a обратно. В противен случай ще върне броя на проектите, създадени досега плюс един, което става числото, присвоено на новия проект, който има всички връзки на проект a плюс връзката между i и j. Функцията `Connected` може да се извика най-много `max_ops` пъти.

Когато вашата програма приключи с операциите `Connected`, тя трябва да опише проект, който е еквивалентен на проект 0. За да опише дизайн, програмата трябва да извика:

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

Параметърът `result` е вектор от двойки цели числа, описващи преките връзки между топлийките. Всяка двойка отговаря на една връзка и трябва да съдържа двата номера на топлийките от връзката. Трябва да има най-много една директна връзка между всяка (неподредена) двойка топлийки и никакви директни връзки между топлийки и себе си. Извикването на тази функция прекратява изпълнението на вашата програма.

Ограничения

- $2 \leq n \leq 200$

Оценяване

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

Примерно взаимодействие

Действия на състезател	Действия на грейдъра	Обяснение
	`ToyDesign(4, 5)`	В играчката има 4 топлийки. Трябва да определите всеки проект, който е еквивалентен на проект 0, като извикате `Connected` най-много 5 пъти.
`Connected(0, 1, 2)`	`Returns 1.`	Топлийки 1 и 2 не са свързани пряко или непряко в проект 0. Създава се нов проект 1.
`Connected(1, 3, 2)`	`Returns 2.`	Топлийки 3 и 2 не са свързани пряко или непряко в проект 1. Създава се нов проект 2.
`Connected(0, 3, 4)`	`Returns 0.`	Топлийки 3 и 4 са свързани пряко или непряко в проект 0. Не се създава нов проект.
`DescribeDesign({{3, 4}})`	-	Ние описваме проект, който има само една връзка: топлийки 3 и 4.

Примерен грейдър

Предоставеният примерен грейдър, `grader.cpp`, в прикачения файл `ToyDesign.zip`, чете от стандартния вход в следния формат:

- Първият ред съдържа броя на възлите n, броя на ребрата m и *max_ops*
- Следващите m реда съдържат ребрата като двойки от възли.

Примерният грейдър чете входа и извиква функцията `ToyDesign` във вашето решение.

За да компилирате примерния грейдър с вашето решение, можете да използвате следната команда в терминала:

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

където `solution.cpp` е вашият файл с решение, който трябва да бъде изпратен в CMS. За да стартирате програмата с примерния вход, предоставен в прикачения файл, въведете следната команда от терминала:

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