Цикъл в компоненти

Намерете броя на свързаните компоненти в графа, в които има цикъл.

Графът ще бъде **ненасочен**. За да се установи цикъл в ненасочен граф, трябва да се спази следното правило, за да се избегне тривиалния цикъл - нямаме право да се връщаме във възела, от който сме дошли. Тоест, ако във възел 1 сме стигнали от възел 0, то нямаме право да се върнем във възел 0.

Никога няма да има повече от 1 ребро между 2 възела.

Спрямо тези правила, свързан граф с 2 върха никога няма да има цикъл.

Input Format

На първия ред ще получите едно число T - броя на заявките

За всяка заявка ще получите следното:

На първия ред ще получите две числа разделени с интервал - V и E - броя на възлите и ребрата в графа.

На следващите E на брой реда ще получите по 2 числа, разделени с интервал - $from$ и to, между кои два възела има ребро.

Constraints

$1 \leq T \leq 10$

$1 \leq V \leq 10^6$

$1 \leq E \leq 2 * V$

$0 \leq from, to < V$

$1 \leq$ *Брой Свързани Компоненти* ≤ 500

Output Format

За всяка заявка T_i изведете едно число на нов ред - броя на свързани компоненти в графа, в които имат цикъл.

Sample Input 0

```
1
3 2
0 1
1 2
```

Sample Output 0

```
0
```

Explanation 0

В този случай имаме 1 свързана компонента, в която няма цикъл. https://ibb.co/6RLnbVj

Sample Input 1

```
1
3 3
0 1
1 2
2 0
```

Sample Output 1

```
1
```

Explanation 1

В този случай имаме 1 компонента, в която обаче има цикъл - например 0 -> 1 -> 2 -> 0. https://ibb.co/jWH9hvg

Sample Input 2

```
1
11 9
0 1
1 2
2 0
3 4
4 5
6 7
7 9
9 10
10 7
```

Sample Output 2

```
2
```

Explanation 2

Тук, имаме 3 свързани компоненти, като само 2 от тях имат цикъл. Забележете, че, обхаждайки графа, нямаме право да се връщаме към предходния възел, от който сме дошли - например от 4 като стигнем в 3, нямаме право да се върнем обратно в 4. https://ibb.co/rFjnvGX

Sample Input 3

```
1
8 10
0 1
1 2
```

```
2 3
3 4
4 5
5 6
6 7
2 5
7 1
5 5
```

Sample Output 3

```
1
```