In computational complexity theory, polynomial-time reduction is an important concept.

If the existence of a polynomial-time algorithm for problem B implies that problem A also has a polynomial-time algorithm, we say that problem A has a polynomial-time reduction to problem B. The relation of reduction is transitive, i.e. if A has a reduction to B and B has a reduction to C, then A has a reduction to C.

Theoretical computer science researchers have found hundreds of polynomial-time reductions between problems, which build a large network of reductions in computational complexity theory.

In this network, some reductions are explicitly presented and others exist implicitly. For example, if the reductions from A to B and from B to C are explicitly presented and the reduction from A to C is not explicitly presented, we say that the reduction from A to C exists implicitly.

In fact, some reductions are not necessary to present explicitly in the network. For example, if reductions from A to B and from B to C exist explicitly or implicitly in the network, then the reduction from A to C is not necessary to present explicitly. By this fact, it's possible to simplify the network of reductions.

Your task is just to simplify the network of reductions such that the number of reductions explicitly presented in the simplified network is minimized and reduction from problem A to problem B exists explicitly or implicitly in the simplified network if and only if it exists in the original network explicitly or implicitly.

Input

The input consists of multiple test cases. Each test case starts with a line containing two integers N and M ($1 \le N \le 100, 0 \le M \le 10000$), which are the number of problems and the number of explicitly presented reductions in the network. The problems are numbered from 1 to N.

Each of the following M lines contains two integers A and B ($1 \le A \le N$, $1 \le B \le N$, $A \ne B$), which means a polynomial-time reduction from problem A to problem B is explicitly presented in the network. The last test case is followed by a line containing two zeros.

Output

For each test case, print a line containing the test case number (beginning with 1) followed by a integer which is the number of explicitly presented reductions in the simplified network.

Sample Input

```
3 3
1 2
2 3
1 3
4 6
1 2
2 1
2 3
3 2
3 4
4 3
0 0
```

Sample Output

```
Case 1: 2
Case 2: 4
```