In this problem, you will be given a directed forest... wait what? A directed forest? Does that even exist?

Well, here in programming world, everything is possible. So let me describe what is meant by a directed forest first. A directed forest is just a set of one or more directed trees, and, a directed tree is just like a normal tree, except the edges are directed. Oh well, we call that a DAG (Directed Acyclic Graph), you'd say, but, I'm not sure if both are same. But I can say this, a directed tree is a DAG whose underlying undirected graph is a tree.

Now, come back to what I was saying earlier, you will be given a directed forest, and you have to make sets of nodes. But there is a restriction, if node **A** is an ancestor of node **B** in the given forest, then **A** and **B** cannot be in the same set. If you do not know what is an ancestor, if there is a directed path from node **A** to **B**, then **A** is the ancestor of **B**. Can you find out what would be the minimum number of such sets to contain all of the nodes?

Input

Input starts with an integer T ($T \leq 100$), the number of test cases. For each case, there will be two integers N and E, the number of nodes and number of edges respectively. Nodes are numbered from 1 to N. Then, there are E pairs of integers (u, v), each denoting a directed edge from u to v. Here you can assume, $1 \leq N \leq 10^5$, $0 \leq E < N$, and $1 \leq u, v \leq N$.

There is a blank line before every case.

Output

For each test case, first print a line of the format '`Case` X`:` Y', without the quotes of course, where X is the test case number starting from 1, and Y is the required answer. Please check sample input and output for more details.

Sample Input

```
3

4 2
1 2
3 4

4 3
1 2
2 3
4 1

7 3
3 6
3 7
1 5
```

Sample Output

```
Case 1: 2
Case 2: 4
Case 3: 2
```