Alex is a developer at the *Formal Methods Inc.* central office. Everyday Alex is challenged with new practical problems related to automated reasoning. Along with her team, Alex is currently working on new features for a computational theorem prover called "Prove Them All" (or PTA for short). The PTA inference engine is based, mainly, on the *modus ponens* inference rule:

$$\frac{\psi, \quad (\psi \rightarrow \phi)}{\therefore \phi}$$

This rule is commonly used in the following way: for any pair of formulae ϕ and ψ, if there is a proof of the formula ψ and a proof of the logical implication $(\psi \rightarrow \phi)$, then there is a proof of ϕ. In other words, if ψ and $(\psi \rightarrow \phi)$ are theorems, then ϕ is a theorem too.

Today's challenge for Alex and her team is as follows: given a collection of formulae Γ and some relationships among them in the form of logical implication, what is the minimum number of formulae in Γ that need to be proven (outside of PTA) so that the rest of formulae in Γ can be proven automatically using only modus ponens?

Input

The input consists of several test cases. The first line of the input contains a non-negative integer indicating the number of test cases. Each test case begins with a line containing two blank-separated integers m and n ($1 \leq m \leq 10000$ and $0 \leq n \leq 100000$), where m is the number of formulae in Γ of the form ϕ_a and n the number of logical implications which have been proven between some of these formulae. The next n lines contain each two blank-separated integers a and b ($1 \leq a, b \leq m$), indicating that $(\phi_a \rightarrow \phi_b)$ is a proven logical implication. Each test case in the input is followed by a blank line.

Output

For each test case, output one line with the format '`Case` k`:` c' where k is the case number starting with 1 and c is the minimum number of formulae in Γ that need to be proven outside of PTA so that the rest of the formulae in Γ can be proven automatically using only modus ponens.

Sample Input

```
1
4 4
1 2
1 3
4 2
4 3
```

Sample Output

```
Case 1: 2
```