– Beehives –

Bees are one of the most industrious insects. Since they collect nectar and pollen from flowers, they have to rely on the trees in the forest. For simplicity they numbered the n trees from 0 to $n-1$. Instead of roaming around all over the forest, they use a particular list of paths. A path is based on two trees, and they can move either way i.e. from one tree to another in straight line. They don't use paths that are not in their list.

As technology has been improved a lot, they also changed their working strategy. Instead of hovering over all the trees in the forest, they are targeting particular trees, mainly trees with lots of flowers. So, they planned that they will build some new hives in some targeted trees. After that they will only collect their foods from these trees. They will also remove some paths from their list so that they don't have to go to a tree with no hive in it.

Now, they want to build the hives such that if one of the paths in their new list go down (some birds or animals disturbs them in that path) it's still possible to go from any hive to another using the existing paths.

They don't want to choose less than two trees and as hive-building requires a lot of work, they need to keep the number of hives as low as possible. Now you are given the trees with the paths they use, your task is to propose a new bee hive colony for them.

INPUT

Input starts with an integer T ($T \leq 50$), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers n ($2 \leq n \leq 500$) and m ($0 \leq m \leq 20000$), where n denotes the number of trees and m denotes the number of paths. Each of the next m lines contains two integers u v ($0 \leq u, v < n, u \neq v$) meaning that there is a path between tree u and v. Assume that there can be at most one path between tree u to v, and needless to say that a path will not be given more than once in the input.

OUTPUT

For each case, print the case number and the number of beehives in the proposed colony or '**impossible**' if its impossible to find such a colony.

NOTE

Dataset is huge. Use faster I/O methods.

INPUT EXAMPLE

```
3

3 3
0 1
1 2
2 0

2 1
0 1

5 6
0 1
1 2
1 3
2 3
0 4
3 4
```

OUTPUT EXAMPLE

```
Case 1: 3
Case 2: impossible
Case 3: 3
```