6894 Algorist Club

The pride of the Algorist Club is that its members really know one another well. In May of every year when it is time for new members to come in, they have a chicken and beer party where the new members are acquainted with the current members and also with one another. The rule is that, if two members $\mathcal{A}$ and $\mathcal{B}$ do not know each other then they have to have a talk session for 15 minutes to really get to know each other. If a member $\mathcal{A}$ does not know members $\mathcal{B}$ and $\mathcal{C}$ then $\mathcal{A}$ must have two separate sessions.

Of course the party has to start as soon as possible and the leaders of Algorist Club have decided to plan the get-to-know-each-other sessions. The time for the sessions will be divided into 15-minute slots. An unlimited number of sessions can go on simultaneously in one slot. However, it is clear that one person can be in only one session in a slot. Let's say that $\mathcal{A}$ is the one who does not know the largest number of people in the Club and k is the number of people that $\mathcal{A}$ does not know. It is obvious that at least k slots are needed. But we really do not know if exactly k slots will be enough. So the leaders have decided to allow $k + 1$ slots for the sessions.

Given the pairs of people who do not know each other, write a program to find a schedule for the sessions which has $k + 1$ slots, where k is defined as above.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing two integers n and m, where n is the number of people in the club and m is the number of pairs of people who do not know each other $(2 \le n \le 400,\ 1 \le m \le (n(n-1))/2)$. The people are numbered from 1 to n. In each of the next m lines, a pair of integers a and b is given, indicating a pair of people who do not know each other. The pairs are given in lexicographical order. That is, they are given in such way that $a < b$ is always true, pairs with smaller a are given earlier, and within the pairs with the same value for a, those with smaller b's are given earlier.

Output

Your program is to write to standard output. For each test case, you should print the pairs given in the input *exactly as they were given*, with the slot number added as the third integer for each pair. That is, for each of the pair given in the input, you should print the pair and the slot that the pair is assigned to. The slots are numbered from 1 to $k + 1$. If it is not possible to finish all the sessions using $k + 1$ slots, print a '`0`' for the slots of *all* pairs.

The following shows sample input and output for two test cases.

Sample Input

```
2
3 3
1 2
1 3
2 3
4 5
1 2
1 4
```

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

Sample Output

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