ETH Zurich
Institut für Theoretische Informatik
Prof. Dr. Angelika Steger

HS 2015
Exercise Sheet 1

ACM Programming Challenges Lab

Exercise 1 – *Important Bridges*

Scatteredalot is a small country, that consists of many inhabited islands connected with bidirectional bridges, so that you can get from any island to any other (possibly using several bridges). Every bridge connects two different islands and for any pair of islands there is at most one bridge that connects them.

Of course it happens from time to time that one of these bridges has to be repaired and the Scatterlotian people cannot use it during this time. Usually this is not a problem because there are so many bridges that most of the times they can use another path leading to the same destination they are heading to. Unfortunately this is not always the case. If repairing a single bridge causes some two islands to be inaccesible from each other, we will call such a bridge *critical*.

Scatteredalot's prime minister, who would like to be re-elected, asks you to find all critical bridges.

Can you help him?

Input The first line of the input file will contain an integer giving the number $c \leq 25$ of test cases following.

Each test case starts with a line containing n, the number of islands and m, the number of bridges. ($0 \leq n \leq 1000, 0 \leq m \leq n^2$.) After that you'll find m lines, the i-th of them being $1 \leq e_{i_1}, e_{i_2} \leq n, e_{i_1} \neq e_{i_2}$, the numbers of the islands the i-th bridge connects. You'll never find two bridges connecting the same pair of islands.

Output For each test case write first k, the amount of critical bridges, in a single line, followed by k lines containing the island numbers $e_{i_1} e_{i_2}$ i-th critical bridge connects.

Please make sure that the list of bridges is ordered in such a way that e_{i_1} is always smaller then e_{i_2}, and that for all i, j with $i < j$ holds that $e_{i_1} \leq e_{j_1}$ and if $e_{i_1} = e_{j_1}$ then also $e_{i_2} < e_{j_2}$. (Which means that the output should be ordered *lexicographically*.) In this way it will be easier for the prime minister to check the list you've created.

Sample Input

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

Sample Output

```
0
2
1 2
1 3
```