Problem C: Pipe Hype

Time limit: 5 seconds

As everyone knows, *Flubbyplaxmol* is this incredible slimy fluid that carries stunning energetic properties and that might fuel in a near future the voyage of a human spacecraft beyond our solar system to colonize new Goldilocks planets. Your friend from the Terrifically Powerful Transports Laboratories (or simply TPT Labs) is working on a *Flubbyendoslasher*, which is a quite simple device that routes Flubbyplaxmol between n entry gates and n exit gates. In a Flubbyendoslasher, the Flubbyplaxmol enters by different gates. It flows through pipes and finally exits by zero, one, or many gates. As Flubbyplaxmol does not mix well, no flows in a Flubbyendoslasher can be mixed. In particular, two different entry gates are never connected to the same exit gate.

Your friend is a specialist of *t-folded Flubbyendoslashers*, or t-fFs for short: they are a revolutionary multilayered stack of t identical Flubbyendoslashers, where the number of copies t can be *extremely* large. In a t-fF, the Flubbyplaxmol that exits by the i^{th} exit gate of the j^{th} Flubbyendoslasher enters immediately into the i^{th} entry gate of the $(j+1)^{\text{th}}$ Flubbyendoslasher. Note that some exit gates may not be connected to an entry gate, and vice-versa: in that case, no Flubbyplaxmol enters or exits by these gates.

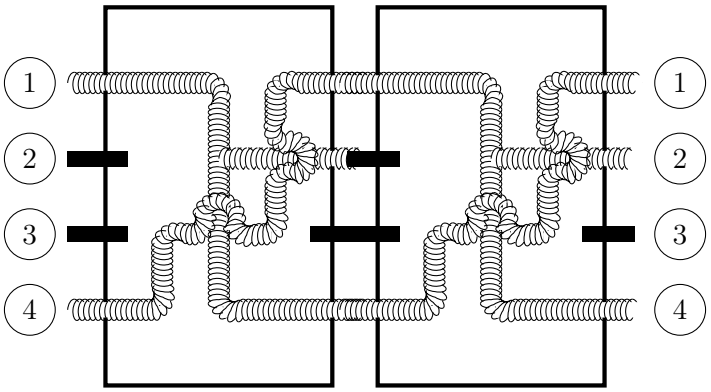

Figure 2: Illustration of a 2-folded Flubbyendoslasher

After much research, your friend has come up with a remarkable observation: *a t-fF is always equivalent to a single Flubbyendoslasher*! This is an incredible opportunity to make considerable plumbing optimizations in the race for the stars! However, to put this result to industrial uses, your friend needs your help. You are given the value of the integer t, and the description of the initial Flubbyendoslasher that is used to build the t-fF. Your goal is to compute a description of a Flubbyendoslasher that is equivalent to the t-fF.

Input

The input file consists of multiple test cases. The first line of the input file consists of a single integer indicating the number of test cases. Each test case follows. The first line of a test case consists of three integers n, m, and t, each separated by a single space:

- $1 \leq n \leq 10^6$ is the number of gates in the Flubbyendoslasher;
- $1 \leq m \leq n$ is the number of pipes between an entry gate and an exit gate in the Flubbyendoslasher;

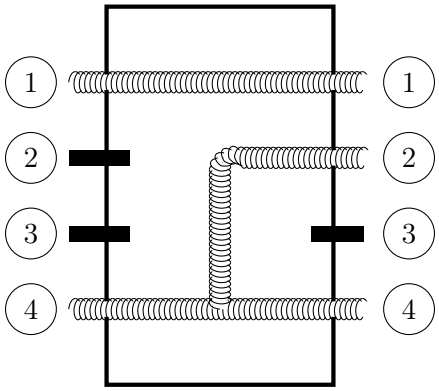

Figure 3: Flubbyendoslasher which is equivalent to the 2-fF in Figure 2

- $1 \leq t \leq 10^{15}$ is the number of copies of the Flubbyendoslasher that have been stacked in the folded Flubbyendoslasher.

The rest of the test case consists of m lines. For $1 \leq i \leq m$, the i-th line describes the i-th pipe of the Flubbyendoslasher, and consists of two integers a_i and b_i separated by a single space: it indicates that Flubbyplaxmol introduced in the entry gate $1 \leq a_i \leq n$ of the Flubbyendoslasher propagates until its exit gate $1 \leq b_i \leq n$. The pipes are given in an arbitrary order. It may be the case that the same entry gate occurs multiple times, but each exit gate occurs at most once.

Output

For each test case in the input, your program should produce a description of the Flubbyendoslasher to which the t-folded Flubbyendoslasher is equivalent. The output of your program for each test case should start with one first line consisting of one integer r that is the number of pipes. The rest of the output of your program for that test case should consist of r lines. For $1 \leq j \leq r$, the j-th line should consist of two integers x_j and y_j (with $1 \leq x_j \leq n$ and $1 \leq y_j \leq n$) separated by a single space, indicating that Flubbyplaxmol introduced in the entry gate x_j of the t-folded Flubbyendoslasher propagates until its exit gate y_j. These r lines must all be distinct and must be given in lexicographical order, i.e., they should be sorted first by the value of the entry gate, and second by the value of the exit gate. There should be no blank lines in your output.

Sample Input

```
2
4 3 2
1 2
1 4
4 1
10 3 8
1 3
3 5
5 7
```

Sample Output

```
3
1 1
4 2
4 4
0
```