Adjacency Lists

Input File	Output File	Time Limit	Memory Limit
standard input	standard output	1 second	512 MiB

This is an easy problem which exists so you can practice reading graphs from input and using vectors in C++.

You are given an undirected graph with N vertices (numbered from 1 to N) and M edges. The input format is an edge list. That is, the first line of input contains N and M and the following M lines each contain a_i and b_i, meaning there is an edge between vertex a_i and vertex b_i. For each vertex, you should output a list of all its edges.

Constraints

For all cases:

- $1 \leq N, M \leq 200\,000$.
- $a_i \neq b_i$ for all i. That is, there are no self loops in the graph.
- $(a_i, b_i) \neq (a_j, b_j)$ and $(a_i, b_i) \neq (b_j, a_j)$ for all $i \neq j$. That is, all edges in the input are different.

There are no subtasks for this problem.

Input

- The first line of input contains two integers N and M, the number of vertices and edges.
- The next M lines each contain two integers a_i and b_i, which describe one edge.

Output

You should output N lines, one for each vertex.

The i-th line should begin with the integer d_i, the number of edges which have one end at vertex i (this is called the degree of vertex i). The following d_i integers should contain the other end of all the edges. You may output the edges in any order.

A note about input and output

In C++, there are two main ways to handle input and output. The first is `scanf` and `printf`, and the second is `cin` and `cout`.

By default, `cin` and `cout` are slower than `scanf` and `printf`. To prevent this, if you use `cin` and `cout`, you should put the following two lines at the very beginning of your main function:

```
cin.sync_with_stdio(false);
cin.tie(0);
```

Below is a description of what they do, but it is fine if you don't understand it.

The first line unsyncs iostream (`cin`/`cout`) from cstdio (`scanf`/`printf`). This will make `cin` and `cout` faster, but **you must not use this line if you are using both scanf/printf and cin/cout in your code** as this will lead to errors.

The second line stops cout from flushing whenever cin is used (you can read about what flushing is here).

Additionally, you should use `'\n'` instead of `std::endl` for printing new lines. For example, you should use:

```
cout << '\n';
```

instead of:

```
cout << std:endl;
```

This is because `endl` flushes the output, which can slow down your program when you are printing many lines.

If you are using `scanf` and `printf`, you do not need to worry about any of this.

Sample Input

```
7 5
1 3
6 2
4 3
3 5
5 1
```

Sample Output

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

Explanation

The input corresponds to the graph in the image below.

- Vertex 1 has an edge to vertex 3 and vertex 5.
- Vertex 2 has an edge to vertex 6.
- Vertex 3 has an edge to vertex 1, vertex 4 and vertex 5.
- Vertex 4 has an edge to vertex 3.
- Vertex 5 has an edge to vertex 1 and vertex 3.
- Vertex 6 has an edge to vertex 2.
- Vertex 7 has no edges.

Please note that the order of the edges does not matter, and so there are other possible outputs.

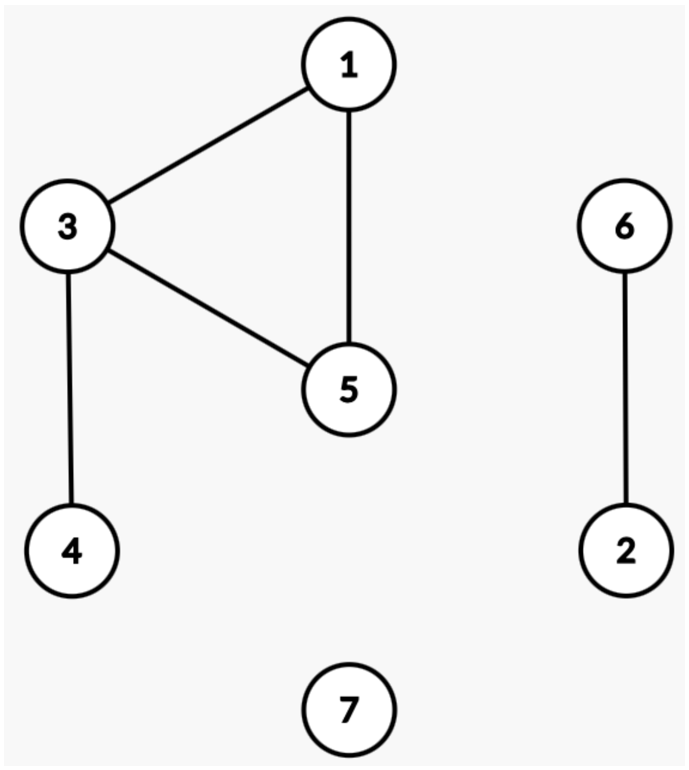

Figure 1: Sample Input