Problem E. Iterative Connected Components

Input file:	`iterative.in`
Output file:	`iterative.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider the following algorithm that finds connected components in an undirected graph:

- Define $c_i = i$ for each vertex i (vertices are numbered from 1 to n).
- Repeat the following many times:
  - Iterate over all edges (a, b) in the graph, and set $c_a = c_b = min(c_a, c_b)$.
  - If the previous iteration over all edges didn't change any number c_i, stop the algorithm.

It's not hard to see that the algorithm will terminate after at most n iterations of the main loop. However, the order of vertices and edges matters. You are given a graph, and need to renumber its vertices and reorder its edges in such a way that this algorithm takes as many iterations of the main loop as possible.

Input

The first line of the input file contains two integers n and m ($1 \le n \le 100$, $1 \le m \le 4950$), denoting the number of vertices and edges in the graph. The next m lines describe the edges, each containing two integers a and b ($1 \le a, b \le n$, $a \ne b$) — the numbers of the vertices connected by the edge. At most one edge will connect any pair of vertices, no edge will connect a vertex to itself.

Output

On the first line of the output file print the maximum number of iterations of the main loop of the above algorithm.

On the second line of the output file print n space-separated distinct integers between 1 and n, describing the vertices *in your order*: the first number should be the number of the vertex in the input file that you declare to be vertex 1, the second number should be the number of the vertex in the input file that you declare to be vertex 2, and so on.

On the third line of the output file print m space-separated distinct integers between 1 and m, describing the edges *in your order*: the first number should be the number of the edge in the input file that you declare to be edge 1, the second number should be the number of the edge in the input file that you declare to be edge 2, and so on. The algorithm will iterate over the edges in your order.

If there are several orderings that yield the maximum number of iterations of the main loop, output any.

Examples

iterative.in	iterative.out
3 2 1 2 1 3	3 2 1 3 2 1

Note

Here's what happens in the example case. We've renumbered the old vertex number 2 to be number 1, old vertex number 2 to be number 1, and left vertex number 3 as number 3. Our edges now connect vertex

number 1 with vertex number 2 and vertex number 2 with vertex number 3, and we process the second edge before the first edge.

We start with the following values of c_i: 1 2 3.

On the first iteration of the main loop, we first process the edge connecting vertex 2 with vertex 3, and c_i become 1 2 2, then we process the edge connecting vertex 1 with vertex 2, and c_i become 1 1 2.

On the second iteration of the main loop, we first process the edge connecting vertex 2 with vertex 3, and c_i become 1 1 1, then we process the edge connecting vertex 1 with vertex 2, and c_i stays 1 1 1.

On the third iteration of the main loop, nothing changes, and the algorithm stops.