DSA Lab 10 Set 1 | StartingTasks

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

A job order given by a client consist of N tasks, numbered $1, 2, 3, ..., N$. However, some tasks may depend on other tasks. For instance, given 9 tasks, if task 4 depends on task 6, task 6 depends on task 8, and task 8 depends on task 5 and no other dependencies are there, then tasks could be carried out in one possible order 1, 2, 3, 5, 7, 8, 9, 6, 4. However, other orders may also exist. We need to order the execution of these tasks based on the dependencies, that is, a task is done after all its dependencies are executed. However, there could be peculiar situations such as task 3 depending on 8, task 8 depending on task 7 and task 7 depending on task 3. This is called a deadlock situation, where the dependencies form a loop, and it is not possible to execute the set of tasks.
Given a set of tasks, and a set of dependencies, your program should determine if this is going to result in a deadlock. If there is no possibility of a deadlock then the program should print all possible tasks that could be executed first in any valid ordering of tasks.
If there is a deadlock for the given set of tasks, output -1. In a case of no deadlock, Output a list of all possible tasks, in ascending order, that could be initiated at the beginning.(Refer to the sample for clarity)

Input

The first line contains two single space separated integers N and D, where N is the number of tasks and D is the number of dependencies.
Next D lines contain two single space separated integers x and y where x and y are task IDs (1-based) and task y is dependent on task x.
Constraints:
$1 \leq D \leq N(N-1)$
$1 \leq x, y \leq N, x \neq y$
Easy: $1 \leq N \leq 10$
Advanced: $1 \leq N \leq 500$

Output

Print -1 if a deadlock is possible. Else, print space separated IDs of tasks that can be executed first in the ascending order.

Examples

standard input	standard output

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

6 6
1 2
1 5
3 4
4 3
5 6
6 1
	-1

Note

Sample 1: No deadlock exists and tasks 1 and 3 could be scheduled first as they have no dependency. One possible order of execution could be 1 3 2 5 4, and another would be 3 1 2 5 4.
Sample 2: Two possible loops are present for the given dependencies for the set of tasks $\{1, 5, 6\}$ and $\{3, 4\}$. Hence, deadlocks exist.