Panda Chess

In the city of Pandaville, the predominant black and white colours of the inhabitants have somehow made chess a very popular game in the area. There is a total of **N** chess players labelled 1 to **N** in the city and they are very competitive.

Recently, the annual Panda Chess Tournament has just ended where a total of **M** matches have been played among the players to decide who is the best chess player. In every match, there was exactly 1 winner and 1 loser with no draws. The same 2 players may play more than 1 match.

Tomorrow, *The Daily Chess* needs to publish the ranking of all the pandas in the chess tournament. They want to publish the ranking in a way such that if a panda won a match against another panda, the panda that won would be of a higher rank. However, this may not always be possible. Furthermore, there might be more than 1 possible ranking that satisfies this property. Time is limited and the publishers are seeking your help to help them find this ranking if it is possible.

In exchange for marks in CS2040, you have volunteered to help the publishers. While you were thinking how about this problem, Rar the Cat suggests that this is a graph problem related to **topological sort**. This can be solved in a manner similar to either Breadth First Search or Depth First Search.

Input
The first line of input will contain two integers, **N** and **M**, the number of chess players and the number of matches played.
The next **M** lines will contain two different integers each. This represents a match between two players where the first integer is the player that won and the second player is the player that lost.

Output
If there is no possible way to rank the pandas so that a panda winning a match will always be ranked higher than the panda that lost, then output “No possible ranking.”
If there is more than 1 possible way to rank the pandas that satisfy the conditions above, then output “Ranking is not unique.”
Otherwise if there is exactly 1 way to rank the pandas according to the conditions above, print **N** lines where each line should contain 1 integer representing a panda. The **N** lines should be printed in order of the ranking where the first line is the top ranking panda and the **N**-th line is the bottom ranking panda.

Limits
- $1 \leq \mathbf{N} \leq 100{,}000$
- $0 \leq \mathbf{M} \leq 200{,}000$

Hint 1: What does it mean if there is no possible ranking? What kind of graphs have no topological sort order?

Hint 2: If there is a unique ranking, the top-ranking panda would have won every other panda directly or indirectly.

Sample Testcases

Sample Input (**pandachess1.in**)	Sample Output (**pandachess1.out**)
2 2 1 2 2 1	No possible ranking.
Sample Input (**pandachess2.in**)	Sample Output (**pandachess2.out**)
2 0	Ranking is not unique.
Sample Input (**pandachess3.in**)	Sample Output (**pandachess3.out**)
2 2 2 1 2 1	2 1

Sample Explanation
In the second sample test case, both 1 2 and 2 1 are possible rankings.

Testing:
You might experience `java.lang.StackOverflowError` while testing your algorithm against your own crafted input. This is due to a low default stack limit of java and it would be increased to 256MB on Codecrunch and when using `check.sh`. To avoid this, please run your program with the following command after compiling instead: `java -Xss256m PandaChess`

Notes:
1. You should develop your program in the subdirectory **ex1** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **2 sec** on Codecrunch.

Skeleton File – PandaChess.java
You are given the skeleton file `PandaChess.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.
You should see the following contents when you open the skeleton file:

```
import java.util.*;

public class PandaChess {
    private void run() {
        //implement your "main" method here
    }
    public static void main(String[] args) {
        PandaChess newPandaChess = new PandaChess();
        newPandaChess.run();
    }
}
```

Source
CS2040 AY2017/18 Semester 2 Makeup Practical Exam