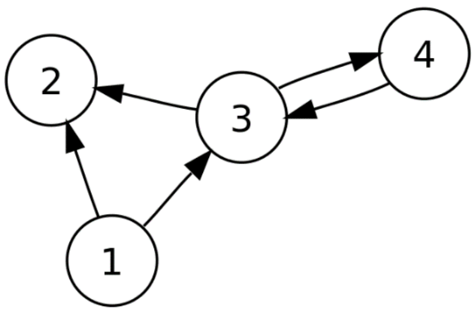
EECS 660 Homework 2: Strongly Connected Components

The goal of this assignment is to implement an algorithm to find all Strongly Connected Components (SCC) in a given graph. Find the algorithm on Page 98 of the book *Algorithm Design*. You may also need the DFS/BFS algorithm on page 90-93.

For a directed graph, a strongly connected component is defined as a subgraph that for any of its two nodes u and v, there exists a path from u to v and a path from v to u. Note that a single node itself can be a valid SCC because, by definition, it can always go to itself.

For example, the graph below exists a path from node 3 to node 4 and also a path from node 4 to node 3, then nodes 3 and 4 are in the same SCC. There is a path from node 4 to node 2, but a 2-4 path does not exist. So, node 2 and 4 are not in the same SCC.



Your program should read in an input file which has the following format:

For example,

```
4    // there are 4 nodes in the graph
     // an empty line, separating the graph size field and the graph field
2 3  // node 1 has two outgoing edges, one connects node 2 and the other connects 3
     // an empty line, indicating node 2 has no outgoing edge
2 4
3    // do NOT include newline here
```

The first argument represents the number of nodes in the graph. In this case, there are 4 nodes. The second argument represents the directed connected nodes to each node. In this example, the first node connects to node 2 and node 3, the second node has no node to connect, the third node connects to node 2 and node 4, and the last node connects to node 3.

1. Create a python file named "scc_username", with **username** corresponding to your KU online username with numbers and your initials. For example, for a student with an online username of j0861791, his/her python file should name "scc_j0861791".

2. Your program should be able to run from the console using the command:

   python scc_username.py input.txt

   where arguments are your program and the input file.

3. Your program should write to **stdout**, not an output file. **python 3** will be used to grade your submissions.

4. Write all of your SCCs in separated lines, one SCC per line. The SCCs need to be sorted based on their smallest node ID, and all nodes within each SCC need to be sorted based on their IDs. In each line, the node IDs should be separated by single spaces.

   For example, the SSCs of the given example should be output as:

   ```
   1     // an SCC containing a single node
   2
   3 4   // do not include newline here
   ```

5. If you think your submission received low score because of formatting issue, you can contact the grader. If we did not specify the format clearly, your score will be revised without penalty. If we did specify it clearly and the reason for the penalty was indeed a formatting issue, you can resubmit within the same day when the grader replied you. **Each resubmission will cost you 10pts (total 100pts).** So, make sure you read the input and output specification carefully.

6. If your program's wall-clock running time is 10X slower than the median of all submissions, it implies that your program may have an unnecessarily high time complexity. **In this case, you will be deducted for 50pts (total 100pts).**

7. Please submit your assignments through Blackboard before **Mar 6th Friday 11:59PM**. If your submission is late and Blackboard is closed, please directly email me with a title "**EECS 660 Spring 2020 HW2 Late Submission [username]**". For example, "**EECS 660 Spring 2020 HW2 Late Submission j0861791**". Submissions not titled properly will not be considered. If you have any questions, please email the grader Jinke Li at j0861791@ku.edu.