CSCI 2300 Lab 5:

Strongly Connected Components

In this lab you will implement the linear time strongly connected components algorithm described in Sec 3.4.2, as described on page 94. Given a directed graph G you first create the reverse graph G_R in linear time. Next, you run the undirected connected components method from sec 3.2.3, processing the vertices in decreasing *post* order. You must make sure that both steps can be carried out in $O(|V|+|E|)$ time. In more detail, carry out the following steps:

1. Create G_R
2. Compute DFS interval on G_R for each node
3. Iteratively select the next unvisited node uu with highest post(u)post(u):

   For each such u, use DFS on G to visit all nodes reachable from u and output this set as a connected component.

Your python script should read the graph file name from the command line. Each line of the graph file will have two integers representing a directed edge from u to v.

You code should output all strongly connected components.

You can try your code on the graph in Fig 3.9a, i.e., on the EXAMPLE.TXT dataset below where each letter of the alphabet has been converted to an integer. The correct output should be similar to:
7 8 9 10 11 12
4
3 6
2 5
1
Note that 4 can also come before 7 8 9 10 11 12.

Finally, report your output on the LAB6TESTSET.TXT (posted in Piazza)

You cannot use any external python graph library. Do not implement an elaborate graph class either. Just use the adjacency list format to store the input graph. Make sure not to use sorting, otherwise the time will not be linear. You may need to increase the maximum recursion depth via the **sys.setrecursionlimit** call.

EXAMPLE.TXT

```
1 2
2 3
2 4
```

2 5
3 6
5 2
5 6
5 7
6 3
6 8
7 8
7 10
8 11
9 7
10 9
11 12
12 10
---------- end ----------