# Topological Sort

This is a warm up for you to be familiar with the format and requirements of programming including the basics of Java (open and read text files and use basic data structures) and Unix commands, reports, and submission.

You are asked to write a java program that can read in a text file that contains multiple directed graphs and find a topological order for each graph, if possible. You should name your program as topSort.java (note that, the first letter of the file name is little case “t”. Unix is case sensitive). I will compile your program straightforwardly as follows without any option from the directory where topSort.java is saved.

javac topSort.java

If your program fails to compile, you will get 0 point. If succeeds, I will run your program’s class file (byte code) as follows:

java topSort graphs.txt

where graphs.txt is the input file that contains multiple graphs. graphs.txt is given in my Public directory (will be explained later) as the required input for the assignment, but I may run your program on different graph files for testing. The graph files will be prepared in the same format as follows:

50 graphs for testing acyclic.

\*\* G1: V = { 0 1 2 3 4} (u, v) E = {

( 0, 1)

( 0, 3)

( 0, 4)

( 3, 1) }

......

......

\*\* G50: V = { 0 1 2 3 4 49}

(u, v) E = {

...... }

The first line indicates the number of graphs in the file. In graphs.txt, there are 50 graphs, where the first graph contain 5 vertices while the last contains 50 vertices. Each graph *G* : (*V, E*) is started with \*\* and ended with a dash line. V is the index set of vertices, e.g., if |*V* | = 5, then V = { 0 1 2 3 4}. It is self-explained that E is the set of directed edges of the graph, one line for each edge. The input graph file should be provided through the command line for topSort to test. You should not hard-code the file name in your program as I may test your program on some different graph files. Note that, the topological order may not be unique. You don’t have to find all of them. Any topological order fulfils the request.

The output should be printed on the terminal screen as follows:

Topological Orders:

G1: 0 2 3 1 4

G2: 1 2 4 0 3

......

G7: 7 -> no more in-degree 0 vertex; not an acyclic graph.

......

G50: No in-degree 0 vertex; not an acyclic graph.

## There are two parts of submission: Programs (60%) and Reports (40%) Important!! You will lose significant points if you fail to follow the rules.

* Programs 60%, submission on Linux server. The score is based on the correctness and documentation of your programs.

1. Also, at the beginning of **every** method, class, or function that you developed on your own, put a few lines of comments with your name and date/time when you develop the code.
2. Do not declare any package for your programs. You can use any Java IDE on your local computer, but you have to transfer all programs and data files needed to our Linux server. Note that, some IDE may automatically add your programs/classes to some default package. You have to remove the package declaration after you transfer the program to a different location, otherwise, the grading script will fail to handle the special need and run into compilation problems.

* Report 40%. You have to write up a report and prepare it in pdf format. The report should include the following items:

1. 10%. The program code is not required in the report. Instead, the *direct output* of your programs on the required input, which is graphs.txt, should be included immediately after the cover page. You can use Unix redirect command, >, to redirect your program’s output to a text file, then copy and paste to your report. Don’t fetch the screen shot. The output on the report must be consistent with the result I get when I run your program on graphs.txt.
2. 10%. A summary of the methods, algorithms and data structures, and the difficulties, if any, the project has faced and how to solve them.
3. 10%. Both time and space efficiency should be analyzed in terms of big-O notations.