

# **Programming Assignment #5: TopoSort**

**COP 3503, Fall 2013**

**Due:** Friday, October 25, 11:59 PM via Webcourses@UCF

## **Abstract**

In this assignment, you will determine whether an arbitrary directed graph has a valid topological sort in which some vertex,  $x$ , comes before some other vertex,  $y$ .

You will gain experience reading graphs from an input file, representing them computationally, and writing graph theory algorithms. You will also solidify your understanding of topological sorts, sharpen your problem solving skills, and get some practice at being clever, because your solution to this problem must be  $O(n^2)$ .

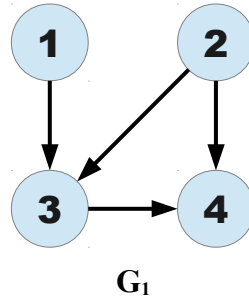
If you use any code that I have given you so far in class, you should probably include a comment to give me credit. The intellectually curious student will, of course, try to write the whole program from scratch.

## **Deliverables**

TopoSort.java

## 1. Problem Statement

Given a directed graph,  $G$ , and two integers,  $x$  and  $y$ , determine whether  $G$  has a valid topological sort in which vertex  $x$  comes before vertex  $y$ . (Notice that the problem does **not** ask whether  $x$  comes *directly* before  $y$ .) For example:



In  $G_1$ , there is a valid topological sort in which vertex 2 comes before vertex 1 (**2, 1, 3, 4**). There is also a valid topological sort in which vertex 1 comes before vertex 2 (**1, 2, 3, 4**). Both of those are also valid topological sorts in which vertex 2 comes before vertex 4. (Notice that vertex 2 does not have to come *directly* before vertex 4.) However, there is no valid topological sort in which vertex 4 comes before vertex 1.

## 2. Input File Format

Each input file contains a single digraph. The first line contains a single integer,  $n \geq 2$ , indicating the number of vertices in the graph. (Vertices in these graphs are numbered 1 through  $n$ .) The following  $n$  lines are the adjacency lists for each successive vertex in the graph. Each adjacency list begins with a single non-negative integer,  $k$ , indicating the number of vertices that follow. The list of vertices that follows will contain  $k$  distinct integers (i.e., no repeats) on the range 1 through  $n$ . For example, the following text file corresponds to the graph  $G_1$  that is pictured above:

*g1.txt*

```
4
1 3
2 3 4
1 4
0
```

## 3. Method and Class Requirements

Implement the following methods in a class named `TopoSort`.

```
public TopoSort(String filename)
```

This constructor opens the file named *filename* and reads the graph it contains into either an adjacency matrix or adjacency list. We will process multiple  $xy$  queries for this graph, but we only want to load it into memory once. This method should throw exceptions as necessary.

```
public boolean hasTopoSort(int x, int y)
```

Given integers  $x$  and  $y$  such that  $1 \leq x \leq n$  and  $1 \leq y \leq n$ , if this graph has a valid topological sort in which vertex  $x$  precedes vertex  $y$ , return *true*. Otherwise, return *false*. Do this in  $\mathbf{O}(n^2)$  time.

```
public static double difficultyRating()
```

Return a double on the range 1.0 (ridiculously easy) to 5.0 (insanely difficult).

```
public static double hoursSpent()
```

Return an estimate (greater than zero) of the number of hours you spent on this assignment.

## 4. Runtime Requirements

Implement a solution that is  $\mathbf{O}(n^2)$ , where  $n = |V|$ . Recall from our formal definition of Big-Oh that a faster solution is still considered  $\mathbf{O}(n^2)$ .

## 5. Grading Criteria and Miscellaneous Requirements

The *tentative* scoring breakdown (not set in stone) for this programming assignment is:

80%	program passes test cases
10%	<code>difficultyRating()</code> and <code>hoursSpent()</code> return doubles in the specified ranges
10%	adequate comments and whitespace

**Programs that do not compile will receive zero credit.** Please be sure to submit your `.java` file, not a `.class` file (and certainly not a `.doc` or `.pdf` file). Your best bet is to submit your program in advance of the deadline, then download the source code from Webcourses, re-compile, and re-test your code in order to ensure that you uploaded the correct version of your source code.

**NEW! Please remove `main()` before submitting.** A lot of `main()` methods are causing compilation issues because they include references to home-brewed classes that are not submitted with the assignment. Please remove.

**Your program should not print anything to the screen.** Extraneous output is disruptive to the TAs' grading process and will result in severe point deductions. Please do not print to the screen.

**Please do not create a java package.** Articulating a `package` in your source code could be disruptive to the grading process and will result in severe point deductions.

**Name your source file, class(es), and method(s) correctly.** Minor errors in spelling and/or capitalization could be hugely disruptive to the grading process and may result in severe point deductions. Please double check your work!

**Test your code thoroughly.** Please be sure to create your own test cases and thoroughly test your code.

*Start early! Work hard! Ask questions! Good luck!*