CS 361 – Lab #11 – Directed Graphs

If we have a directed acyclic graph (DAG), then there is a way to appropriately number the vertices. Formally, we say that we are creating a “topological ordering” of the vertices. In a topological ordering we have the following property: for every (directed) edge going from vertex #i to vertex #j, the number i must be less than j.

In today’s lab, you will implement an algorithm to determine a topological ordering of vertices in a directed acyclic graph. Here is the pseudocode based on the constructive proof we saw:

- Let v_1 be a vertex with in-degree 0. We are guaranteed to have such a vertex.
- Remove v_1 and its out edges. The resulting graph is also acyclic.
- Similarly, we can now find a vertex v_2 with in-degree 0. Remove v_2 and its out edges.
- And so on for vertices v_3 through v_n.

Create a directory called lab11 on your account. Please copy the source files Graph.java and Driver.java from my folder lab11 on the class Web site. You will be modifying these source files.

Graph.java is a rudimentary implementation of a directed graph class. Several instance methods have already been created for your convenience. You need to implement:

- removeVertex()
- topOrdering() – because this returns void, do your output from this function. Alternatively, you may change its return type to String, and return a string containing a list of the vertices listed in correct numerical sequence.

Driver.java contains the main() function. Here, you need to create your own input data for the directed graph. Use the methods addVertex() and addEdge() as appropriate. What sort of graph should you create? One possible example is the pre-requisite structure of the Furman CS major. ☺