CS6905 (AGA) Fall 2023 – Assignment 3 (Minor)
Due Wednesday October 4, 2023, by 10pm.

The `DGraphWtAL.java` file (on Desire2Learn) provides the `DGraphWtAL` class, used to build and store a static *directed* graph with edge weights. It uses node indexing (from 0) and adjacency lists (built using an internal class `GNode`) as the underlying data structure, where each node has a list of its outgoing edges (`OutAL[node]`) and another list of its incoming edges (`InAL[node]`). This class also has an array of integers (one for each node) to be used to mark nodes. The two adjacency list entries for each edge (one outgoing, one incoming) are linked, and there is also a mark field on each of them that can be used.

Its constructor sets up the arrays of adjacency lists and array of vertex marks.

`reset()` takes an integer parameter and sets all vertex marks to that integer. This method is to be used to initialize marks to the desired default value.

`addEdge()` takes a pair of indices (x and y) as parameters, and the weight wt and used marker *used*. It inserts the edge $\langle x, y \rangle$ between them into the adjacency list representation, setting the weight to wt and the edge mark to *used*. (Note: the *used* parameter is to enable testing needed by some other algorithms.)

`testEdge()` takes a pair of indices (x and y) as integers, and returns a `boolean` that is **true** if there is an edge in the graph from x to y, **false** otherwise. (Note: you should not use this method in this assignment.)

`toString()` converts the adjacency lists (and current vertex marks) into a string, with one vertex+list per line. The string gives the lists of outgoing edges followed by the lists of incoming edges.

You need to write a `DGraphTopo` class that extends `DGraphWtAL`, to add the following method:

- `topoSort()`: takes no parameter, and returns an array of integers that has the vertices in a topological order, as sorted by the Topological Sort algorithm presented in class. You must implement that linear time topological sort algorithm. You can assume that the graph has already been verified as being a DAG.

Your class will also need a constructor to pass on the size parameter to the `DGraphWtAL` constructor, and should also have private methods and classes as appropriate. Your class should not have any additional variables declared globally for the class.

Ensure that your code works with the provided `DrAGA3.java` code, which will be used to test your submitted solution. Organize and comment your code appropriately.

Submit on D2L: Two separate files: your `DGraphTopo.java` file, and the I/O from one test run of your solution.