CS 245 - Practice Assignment 09 Spring, 2020

Practice Assignment 09 - Basic Graphs & Topological Sort

The goal of this assignment is to practice one implementation of the graph data type and one algorithm: topological sort. For this assignment, the correctness of the data structures is measured by how well your implementation conforms to the given specification.

Background

In class, we discussed the graph data structure as one which contains vertices and edges connecting pairs of vertices. Of the implementations we discussed, this assignment requires an unweighted, directed graph. Only three graph functions are required, as shown described in Table 1 below and defined by the Graph.java interface. While the size of the graphs used in this assignment may suggest the use of an adjacency matrix, you are free to implement your graph using an adjacency list if you wish.

Function Signature	**Usage / Notes**
<<constructor>> (int vertices)	Constructs and returns a graph with the number of vertices passed as the argument. Vertices have IDs, numbered 0, 1, …, vertices-1. No edges exist between vertices at instantiation.
void addEdge (int src, int tar)	Adds a directed edge between two vertices from src to tar.
List<Integer> neighbors (int vertex)	Returns a List of vertex IDs, with each ID representing a vertex which is the destination of the edge originating from the source vertex, passed as the argument.
List<Integer> topologicalSort ()	Prints (to console) one ordering of vertices such that each directed edge (v1, v2) from vertex v1 to vertex v2, v1 comes before v2 in the ordering. If such an ordering is not possible (due to cycles, for example), this function must indicate so, though it may print a partial solution in so doing.

Table 1: Graph Functions

Requirements (Process)

Requirement 1: Get the files you need from GitHub Classroom:

1. Log into GitHub.
2. Point your browser to the URL https://classroom.github.com/a/npOTGNnO.

3. Select your name and GitHub ID from the list.
4. Accept the assignment by selecting the appropriate button.

When successful, this procedure gives you a (private) with two (2) Java files, `PracticeTest.java`, which contains the main function, and `Graph.java`, an interface for the Graph class.

Requirement 2: Add to the code in order to make it run. Specifically, your implementation must be performed in a class called `GraphImplementation.java`, which must implement `Graph.java`. This implementation must contain at least the functions listed in Table 1.

Grading

The assignment generates text ("`Starting point for this assignment`") based on tests of the functions priority queue functions. Assuming your implementation of the topological sort also produces the correct output, your starting point for the grade is as indicated. Your final grade may be adjusted based on additional items including, but not limited to, code quality and adherence to the above requirements.

Submission

You are required to submit one class for this assignment: GraphImplementation.java. Use GitHub to check in the class required to complete the implementation. On Canvas, submit the URL for your GitHub repository.