Assignment: Capstone warm up

✓ Passed and verified · 20/20 points

Deadline The assignment was due on March 27, 11:59 PM PDT
You can still pass this assignment before the course ends.

Instructions (/learn/intermediate-programming-capstone/programming/wkgsS/capstone-warm-up)

My submission (/learn/intermediate-programming-capstone/programming/wkgsS/capstone-warm-up/submission)

Discussions (/learn/intermediate-programming-capstone/programming/wkgsS/capstone-warm-up/discussions)

Warm Up Assignment Instructions

Although the capstone main project is open-ended, this first assignment is a bit more prescriptive to help get you warmed up.

In this first assignment, you'll be working with some social network data. We are providing you with two sets of data to start: social network data from Facebook and communication (tweet) data from Twitter. In this first assignment you will:

- Design and implement a set of classes to store this data as a graph (this graph must implement our provided interface, and the class that implements this interface must be named CapGraph)
- Use our provided files to load the data into your graph
- Write a method that extracts a subgraph from the Facebook data called an “egonet”. Egonets are explored in more detail below.
- Write a method that finds the strongly connected components in a directed graph.

You can extend your work on Egonets or strongly connected components to build up your capstone project in future weeks. Or, you can choose a completely new avenue for your self-directed project.

Starter code, data and setup

As in our previous courses, we are providing an eclipse package to start you off with this assignment. Because you've worked through those courses, we assume that you are familiar with the steps for setting up eclipse and importing an existing Java project. If not, you can refer back to our earlier setup guides (e.g. here is the guide from course 3 (https://www.coursera.org/learn/advanced-data-structures/supplement/cREMI/setting-up-java-eclipse-and-the-starter-code)).

You can download the starter code and some data (Eclipse project, as usual) here:

UCSDCapstoneCode.zip (https://d18ky98rnyall9.cloudfront.net/_742091298e79df199...

(Last uploaded 3/29/2016, 3:36pm PDT)

Changes since 3/24/16, 3:28pm PDT

- Fixed score calculation in SCCGrader

Changes since 3/14/16, 3:44pm PDT (first uploaded starter files):

- On printing an "incorrect" message, the correct "expected" value is now shown.
- The grader now accepts egonets that include the center vertex (data files describing egonets did not include the center vertex, as all vertices were implicitly connected to it, so the grader originally expected this (and will still accept it as a valid answer)).

In the starter code you will find the following directories and files:

- **src/graph**: This package contains two files. Graph.java is the graph interface you must implement for this assignment. CapGraph.java is a stub class definition of a class that implements the Graph interface. You should use this class in your implementation by filling in the methods. Our graders will create instances of CapGraph to test your implementation so your graph class must be named CapGraph. You are free and encouraged to add other files and classes to the src/graph package. You should not add files or code to any other package in this warm up assignment (though you're welcome to build on this warm up starter code for your main project). You will submit all of the files in this package for grading this week.
- **src/graph/grader**: The graph.grader package contains the code we use to grade your CapGraph implementation when you submit it. If you get any errors when you submit your files, you can run these graders from your own computer to help you figure out which tests are failing.
- **src/util**: The util package contains only one file: GraphLoader.java. This is a utility class that has a method to load the data files we provide into a Graph object that implements the Graph interface.
- **data**: The root data folder contains a number of files that you might find useful. The data/data_source.txt gives the source of the data files, and the data/README file provides more information about the files in that folder. You should not modify any of the files in that folder as many are used for grading (so if you modify them, the graders we provide you might not give correct results). But you are welcome to make copies and modify those copies. And once you have passed this assignment you can edit any of the files freely.

Part 1: Graph Representation

Your first task is to design a graph representation in Java. This involves creating a class or set of classes to implement the Graph interface specified in the file Graph.java using the class CapGraph. How you design this graph is entirely up to you, however. You should think about the data you need to represent and design the graph accordingly. You might find it useful to refer back to Course 3 where we talked about graph representation. https://www.coursera.org/learn/advanced-data-structures/home/week/1 (https://www.coursera.org/learn/advanced-data-structures/home/week/1) (See lesson 2)

Implement at least the addVertex and addEdge methods in this part, and ensure that your graphs are being created correctly, using small, made-up data. You will also need to add some additional methods to your graph class (e.g. some print methods) and/or implement the exportGraph method to ensure that your graph was in fact loaded correctly. Alternatively, we encourage you create JUnit tests to ensure that your graph is being loaded correctly.

Once you are sure your methods are working, you can use the GraphLoader class we provide to load the Facebook and Twitter data we provide into your graph representation. After you've satisfied yourself that the graph is loading correctly, go on to the next part of this week's assignment.

Part 2: Two graph algorithms

In this part of the assignment, you will complete the implementation of the Graph interface by implementing the getEgonet and getSCC methods. You will also need to have implemented the exportGraph method which returns the data in your graph in a format that we can grade it.

public Graph getEgonet(int center)

This method takes an int which is the node/user at the center of the desired egonet, and returns that user's egonet as a graph. The returned graph should not share any objects with the original graph. E.g. if your vertices or edges are represented using objects, the returned graph should contain copies of all of the vertex and edge objects.

An egonet is a subgraph that includes the vertex center and all of the vertices, v_i, that are directly connected by an edge from center to v_i and all of the edges between these vertices that are present in the original graph. Examples of egonets are given in the lectures in this module.

If the vertex center is not present in the original graph, this method should return an empty Graph.

public List<Graph> getSCC()

This method returns all of the strongly connected components in the Graph as a list of subgraphs. As with getEgonet, the returned graphs should not share any objects with the original graph.

The lectures present details about strongly connected components and an algorithm for finding them.

public HashMap<Integer, HashSet<Integer>> exportGraph()

This method returns the nodes and edges in your graph in a format suitable for grading. It should return a HashMap where the keys in the HashMap are all the vertices in the graph, and the values are the Set of vertices that are reachable from the vertex key via a directed edge. The returned representation ignores edge weights and multi-edges, but it's sufficient for our grading of the two main methods on this assignment.

Submission

To submit both parts of this assignment, zip all the files from the graph package that you use into a file called warmup.zip. Include only your .java files that you added to the graph package, and not the graph folder itself. Do not include files from any other folder (you should not have added files to any other folder) and do not include data files. Use this file as your submission.

How to submit

When you're ready to submit, you can upload files for each part of the assignment on the "My submission" tab.