COMS 3110: Homework 3
Due: July 22^{th}, 11:59pm
Total Points: 110

Late submission policy. Late submissions will not be accepted.

Submission format. Your submission must be a `zip` file consisting of only all necessary source files and any other files explicitly mentioned. You should *not* include compiled files, such as those ending in `.class`, unless otherwise explicitly stated. You should *not* include other files such as project and IDE configurations. Name your submission file: `<Your-net-id>-3110-hw3.zip`. For instance, if your netid is `asterix`, then your submission file will be named `asterix-3110-hw3.zip`. Your submission `zip` should unpack to folder called `asterix-3110-hw3`. Inside that folder should be a `src` folder containing the source code for the assignment. All `java` source files should be in a folder hierarchy that matches the sources' packages. For example, if you are submitting `MyCode.java`, and it is in the package `course.assignment`, your submission should unzip to `./asterix-3110-hw3/src/course/assignment/MyCode.java`. Each student must submit their own assignment. If you discussed the homework or solutions with others, a list of collaborators must be included with each submission. Each of the collaborators has to write the solution(s) themselves (copies are not allowed).

General Requirements

- Programming assignments must be in Java.
- Programming assignment submissions must be properly structured zip files.
- Programming assignment submissions should not include compiled units (e.g. `.class`, `.jar`).
- Programming assignment submissions should not include IDE configuration or other settings files.
- When proofs are required, do your best to make them both clear and rigorous. Even when proofs are not required, you should justify your answers and explain your work.
- When asked to present a construction, you should show the correctness of the construction.

0 Preamble

This description of a programming assignment is not a linear narrative. It may require multiple readings before things start to click. You are encouraged to consult the teaching staff for any questions/clarifications regarding the assignment. Your programs must be in Java.

1 Overview

You've been brought in to manage an internal LAN, consisting of a large number of devices and connections between devices. The devices are responsible for monitoring and reporting incidents at their location to all other devices on the network. All of the hardware is homogeneous, the devices are very powerful, and the links between them are both highly and equally efficient. This means that a message sent between any two directly connected machines is delivered in exactly the same amount of time. Unfortunately, due to various limitations during the construction of the network, some of these connection are simplex, while others are full-duplex. Due to the critical nature of incident reporting, you've been tasked with determining the longest amount of time required for any machine to report an incident to another machine on the network.

2 Graph Implementation

As part of this assignment, you will be implementing a generic graph class. The underlying representation *must be an* ***adjacency list***. In addition to basic graph functionality, you need to implement an efficient algorithm for finding the length of shortest paths in a method on the class.

You are free to use basic data structures (including hash table based ones) implemented in the standard JDK to implement your class.

3 Network Analyzer

You've been assigned to compile a report about a large network of interconnected devices. This network has been expanded over time according to various constraints and limitations, thus precisely which devices are connected may seem to an outsider to be inexplicable. An additional consequence is that some of these connections are unidirectional and others bidirectional; despite this, every device is reachable on the network by every other device. Furthermore, the linked devices are connected with materials and hardware to ensure that sending a message between two directly linked devices has an upper bound of 1 unit of time.

You have one primary and one secondary task to accomplish with respect to the network analysis.

1. When the system a device is monitoring experiences a failure, it broadcasts this information in a message to all other devices on the network so that the other system can adapt. Management has tasked you with determining the maximum time it may take a system failure message to be delivered to every device on the network.

2. Management is also looking ahead, and intends to replace many simplex links from the network. It is important that all devices are pairwise reachable, but each connection is costly. They want to use as few duplex connection as possible to restore connectivity, and they've asked you to determine the minimum number required for this.

Network analysis will provided by the ‘NetworkAnalyzer‘ class and should utilize your ‘Graph‘ class internally.

4 Classes

You may add helper methods and member variables, but they must be private or protected. Your graph class may be tested independently of any other classes using the given public API.

Important classes and some of their notable methods are given below. The methods not listed here but present in the provided skeleton are also required. They must also be implemented correctly.

- `public class Graph<T>`

  A directed graph stored as an adjacency list. In addition to all of the methods providing basic graph functionality, you must implement the following method:

  - `public Map<T, Long> getShortestPaths(T s)`

    : This method computes the shortest path from s to all reachable vertices. Every reachable vertex should be a key in the returned map. The corresponding value should be the length of the shortest path to that vertex. Unreachable vertices should not be present in the map. If the input vertex s is not present in the graph, an empty map should be returned.

- `public class NetworkAnalyzer`

  - `public NetworkAnalyzer(List<String> hosts,`

    `Map<String,String> simplexConnections,`

    `Map<String,String> duplexConnections)`

    : The first parameter is a list of distinct hostnames, each representing a unique device on the network. The second parameter is a collection of simplex connections; a key-value pair of (s, t) means that s is able to send a message to t. The third and final parameter is a collection of duplex connections. For a key-value mapping (u, v), u is able to send a message to v and v is able to send a message to u. Only one key-value pair will be present for any duplex link, i.e. (u, v) or (v, u) will appear in the map but not both.

  - `public long findMaxDeliveryTime()`

    : This method should return the maximum time required for any message to be delivered to another computer on the network.

  - `public int countMinToReconnect(Map<String,String> simplexConnectionsToRemove)`

    : This method is extra credit and thus optional.

5 Important

- Your submission must be *your own work*. The only exceptions are the skeleton provided to you.
- Any code submitted that is not your own *must* be properly attributed (at minimum via a `README` file in your submissions root directory), and it must be in source format (not compiled or otherwise). Any code given to you by us for this assignment is excluded from this requirement.
- Your classes may be tested independently of one another. They will be tested using the specified public API.
- You should document your code where appropriate. This includes inline documentation for long or complex functions.
- You may **not**:
  - Change or alter the public API of any given or required types. This includes *class names, package names, method names, return types, class visibility, method visibility, method arguments, interfaces, etc.*
  - Include any external libraries (`jar` or otherwise). This especially includes graph libraries.
  - Submit any code that is not your own, unless otherwise explicitly allowed. Any code given to you by us with this assignment is excluded from this requirement.