DSA Assignment

Documentation & REPORT

George Aziz - 19765453

2019

# DOCUMENTATION:

**A description of any classes you have, you need to let us know not only what the purpose of that class is but why you chose to create it. As part of this, also identify and justify any places where it was possibly useful to create a new class but you chose not to, especially when it comes to inheritance.**

**Classes in my program including their private inner classes:**

1. SocialSim (Class with my main)
2. DSAGraph
   1. User
   2. Post
3. DSALinkedList
   1. DSAListNode
4. DSAQueue
5. UserInterface
6. FileManager
7. SimulationMode

The **socialSim** class is the class that contains my main. It is the class that you run the program with and calls either UserInterface or SimulationMode depending on the mode the user wants to run the program with.

The **DSAGraph** class could be regarded as the main file as it contains all the main network/graph functionalities. Anything such as adding or removing a user, post, like or a follow is within the DSAGraph class. In addition to adding and removing, the DSAGraph class also contains anything related to displaying the network such as an adjacency list or as exporting as a network file to be saved later. The class itself does not actually display through System.out.print but uses DSAQueue to enqueue anything that I would want to be displayed and then the actual display happens in UserInterface by iterating over the queue. This class also contains sorting the users and posts in terms of popularity as it is required in the assignment specification sheet for the display of statistics . The reason I created a DSAGraph class is to be able to have a social network and connect users with each other.

Within the DSAGraph class, there are two private inner classes. These classes are **User** and **Post**. For User, it was originally a vertex but I decided to change the name to make it easier to understand the code but also to make it have more sense for the class to contain the classfields that have been included. The user class is responsible for everything related to the user which is to have a user name, followers and following list, a list of posts and a count for how much follows and followers. Each of these classfields have accessors and mutators such as addFollow for example. My second private inner class is Post. I decided to create a Post class quite late into the assignment, but it has helped me a lot and made my code a lot shorter and easier to understand. This class is responsible for everything related to a post such as the post itself, a list of likes and a like count, and in addition the name of the poster. This class is very important since makes the adding of likes and removal of likes much easier but also is stored in each User’s posts linkedlist which makes accessing a post very easy.

The **DSALinkedList** class is very important for this program as most of my private class fields in either DSAGraph, User or Post utilise this class a lot. I used it since I am most familiar with this abstract data type/structure but also since it makes the most sense to me to use a linked list within a graph. Any adding or removing requires DSALinkedList in my program since that’s the main structure that is used.

**DSAQueue** was a class I added towards the end when I realised for my writeFile method in FileManager it will have to be able to write in many formats depending on my usage. For example, exportNetwork in DSAGraph is a method that takes in the whole graph as a network file format and then that gets used for when I want to save my network. But in simulation mode I also need to write to file but in a format as an adjacency list and the statistics. In order to overcome this problem, I decided to use a queue where everything that will be displayed either to the terminal for the user or to a file will be enqueued in the format that is set to and then can be used for both displaying to the user or file output.

The **UserInterface** class is responsible for displaying to the user a main menu and also all the inputs for everything such as follows or a user. This class is also responsible for validating user input whether that’s a user or an integer or even a post. It is very important to have this class for my program since the mainMenu submodule inside of it connects all the functionality of DSAGraph together as well as FileManager.

The **FileManager** class is self-explanatory with its name but basically manages reading and outputting from/to a file. It is able to read both event and network files and also output a graph in the same format as a network file or output statistics of the network as well.

**SimulationMode** is a class that is responsible to run the sim mode of the program. It runs reads both event and networkfiles and then goes through a loop of time steps until the user stops the loop. Just like the UserInterface class is mainly for interactive mode, this class is mainly for sim mode. I decided to create this class since it made my main a lot cleaner. Rather than having everything for sim mode in main which is possible and would’ve worked it makes more sense to have everything for the simulation mode to be in one class and for it to be called in one line in the main.

It was possible for me to create a new class called Network and shorten the DSAGraph class by a little bit but I didn’t really see a point since all the methods that are in DSAGraph currently relate to the graph’s functionality or display of it but also if I did create a Network class, then I would’ve had to make my User and Post class public and a separate class to DSAGraph which doesn’t really fit my design as I like to keep everything for a graph, inside of DSAGraph.

**Justification of all major decisions made. In particular, when you choose an ADT, underlying data structure or an algorithm, you need to justify why you chose that one and not one of the alternatives. These decisions are going to be of extreme importance in this assignment.**

For the data structures, I used DSAGraph since that is required of us from the assignment specification, but also because it is the data structure that is used to create the social network itself. Without the graph data structure, there is no social network.

Another data structure that I use is a linked list which is in the DSALinkedList class and I’ve used it since it is much easier to iterate over it in comparison to other data structures such as Hash Tables and it is much easier to use since I’ve used it many times before. The time complexity of a LinkedList at its worst is of O(N) whilst the best case is either when the node is in the head or tail and that is O(1).

The ADT I use is a queue. The queue is in the DSAQueue class and is used mainly for outputting to a file or to the user since many outputs to files or to the user are required in different formats and the easiest way is to keep the format in a queue and then iterate over it when its time for output. I chose a queue over a stack since a stack follows the FILO structure which isn’t useful for when displaying unless I push everything in reverse order which is not ideal and makes the program much more complicated than needed.

# REPORT:

**Abstract: Explain the purpose of the report and state what you are investigating, and the outcomes/recommendations.**

In this report I will be investigating the time complexities of the linked list

**Background: Discuss your approach to developing the simulation code, and the aspects of the simulation that you will be investigating.**

**Methodology: Describe how you have chosen to profile and compare multiple runs of the simulation, and why. You should give some prediction of the expected “performance” of the aspects of your code you are investigating – this includes time complexity and/or memory usage. Include the commands, input files, outputs – anything needed to reproduce your results.**

**Results: Present the results of your simulations, and what you discovered.**

**Conclusion and Future Work: Give your conclusions and what further investigations could follow. Do the actual results match your prediction?**