University of Glasgow | School of Computing Science

Level 4 Project - Algorithm Animator
Status Report

Andrei-Mihai Nicolae (2147392)

December 17, 2016

1 Project Description

The project is aimed at students who want to understand algorithms better - it will help them visualize what is going under the hood of some algorithms taught in Algorithmics 3 class. The end product should be a user friendly cross-platform application (i.e. Linux, MacOS, Windows) which should show animations corresponding to steps in performing the algorithm on user-defined input.

2 Progress

After discussing different approaches with Dr. Norman, I have decided to use the Electron framework (i.e. web-based framework built by GitHub; it is used to power their own text editor, Atom, as well as many other desktop applications). This is the core of the project and, as mentioned, it uses web tools: HTML, CSS, JavaScript. I have taught that because web-based apps have such success in the technology world, the multitude of open source libraries and support would provide enough opportunities to create both a user-friendly and efficient application that would run on any major operating system (and feel native at the same time).

In terms of algorithms implemented, we have started with the Huffman Tree Algorithm and I can say that this major step is almost finished. Even though I tried different JavaScript graph representation libraries (i.e. TreantJS, SigmaJS, Vis.js), I decided to use vis.js in the end as it provides a huge amount of features and very well written documentation. The user has the chance of choosing how to input the text: input it manually in a text box, randomly generated, as well as reading from a file.

After this algorithm is fully implemented (which is aimed to be done before Christmas), I will proceed with other algorithms as well (after discussions with Dr. Norman, we have thought of maybe doing some graph).

3 Plan

The AuctionParticipant interface (with its implementation AuctionParticipantImpl) is defining a user of the system. When the client starts, it will also create a new user with the name parsed by a scanner. Then, I decided to use a rather simple interface for it, giving it only 3 methods: `getName()`, `getId()` and `notify(String)`. The last method is the most important as it is used by the server to notify the user when specific events occur (e.g. the user won an auction, his/her item was sold for x pounds).

4 Problems

I tried to make the client as user-friendly as possible, implementing features such as: asking the user for name, the ability to always type help to see the list of available commands if they were forgotten, as well as letting him now of a variety of things through the notification mechanism I mentioned above.

It will go through and loop until user has either typed in quit or the server has been shut down/quit unexpectedly. I have also implemented easy to understand errors informing the user of any error such

as typing a wrong command, placing a bid smaller than the item's current value, displaying an auction that hasn't been created yet, as well as inputting the closing time for the auction in an invalid format.

I have created also created a network failure detector which is a single thread running continuously that will call a dummy method on the server and, if the timeout reaches 5 seconds, it will assume that the server is down and it will stop the client automatically.

Coming back to the core functionality of the client, I have made it look-up the address and port specified by the user as arguments when running the class. It will try to see if there is an Auction Manager bound to that address (using `Naming.lookup()` method) and, if so, it will start waiting for commands from the user. As stated before, the user can choose to do one of the following: bid, create auction, display specific auction, display all auctions, save state, quit the client as well as display all possible commands via help.