Anthony Gatte

Dr. Timmerman

Computer Science 220

14 November, 2017

Implementation/ Designs Paper

When first starting out with the project I was heavily considering using an AVL tree since we talked about it in class and how it should work with practically any information. However, I ended up using a Binary Search Tree since I am very comfortable with binary search, I like the tree structures of them, and I had coded a binary search tree in a previous class. We first must search the root, if it’s the root we return that, if not we search lower values on left, and higher on the right, then we just keep going until we find the value.

I needed an insert and a remove method for this, both of which are in O(log n), since they’re binary search trees they keep dividing the list into halves until it reaches the node it needs to either insert to or delete. The remove function was the one I found most difficult and had to rework several times before it worked. I initially was trying to remove from ifStream but that wasn’t working so I went to what I currently have. I used these methods to remove and insert the words into different trees which would have courses from coursesNeeded, the courses from coursesTaken, or both. I also struggled with how I wanted to print this out and and the ifstream, however was able to go back and look and notes from the previous course and was able to find information on this. This has a lot of opening and closing of files, and is also where I then call insert and remove in order to place them in the appropriate tree, so I am not certain about what this would do to time complexity, if anything at all. If I were to guess, I don’t think it would do that much, it would only really affect the amount of time if there were say hundreds of files we were opening and closing, and inserting/removing words from.

This is the most effective way I could think of since an array would take Big O(n) and that would be my next best option.