Lab 3: Binary Trees

Kimberly Arenas

**Introduction:**

Binary trees are a data structure used to search through data in a specific order, for example, a binary search tree (BST) has the rule that any node left subtree key is always less than or equal to the node’s key while the right subtree key is greater than or equal to the node key. Having this in mind in this lab we implement two specific trees: AVL Tree (BST where the difference between heights of left and right subtrees cannot be more than one for all nodes) and Red and Black Tree (every node has a color either red or black,root of tree is always black. there are no two adjacent red nodes). After making a two separate class implementation of both AVL and Red and Black Trees we will then read through a text file and keep count of the anagrams of a specific word chosen by the user and the max number of anagrams by that word will be returned.

**Solution design and implementation:**

Firstly, we were given permission to modify the pseudocode for the AVL Tree and Red and Black Tree via Zybooks by our professor. Therefore I believe it be convenient to focus on how the main method was created of course this included integrating the text file into the program. In order to access the text file previously mentioned was done by first asking the user what tree they would like to use. Once chosen by specific user input of either “a” for AVL tree or by “b” for Red and Black Tree the text file would follow path and be implemented accordingly to the data structure chosen.

In order to continue I have to mention that three different files were created: the main execute file, AVL tree implementation, and the Red and Black tree implementation. The reasoning behind this is due to the fact that the tree implementations node functions varied. Both tree implementations were imported onto the main execute file to be manipulated in there.

More specifically, the method that reads the text file to search for anagrams was created by having in mind to be able to insert it by whoever tree would be picked. Then the print\_anagrams method was used to find the solution with strings. Then to find count\_anagrams was as simple as indicated which was to return the number of anagrams that the word chosen by the user has. Lastly maximum\_anagrams method reads another text file and compares to the previous and returns the word with the greatest number of anagrams. In order to do that the user is asked if they would like to open a new file for a new word and if done then that new word is inserted in either AVL or Red and Black Tree then compared to the previous word chosen.

**Experimental Results:**

The user is asked to insert a text file and for testing purposes I made small text files of words with the amount of anagrams that were quick to test instead of the big one provided to us. For example:

A screenshot of a cell phone

Description automatically generatedA picture containing screenshot

Description automatically generated

Then after the text file is picked the user is prompt to then choose the tree they would like the words to be manipulated in. These tree implementations were simple to complete since we had to translate the pseudocode from zybooks to python. For the count anagrams method the “print anagrams” method had to be modified and that was by not exactly printing the anagram for the word inputted by the user, but instead keeping a counter that returns the integer. Lastly the method “maximum\_anagrams” is used to allow the user input a new word and compare it to the words previously inputted and keep track of what word contains the maximum anagrams.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Conclusion:

Lab 3 encouraged me to practice python skills in a good way since I had to translate the pseudocode and modify it. I believe that I could have made the program better by prompting the user to input another word after loading another file instead of just taking in the first word of the second file.

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provide inappropriate assistance to any student in the class.