Adrian Sanchez

Prof: Diego Aguirre

Lab 4

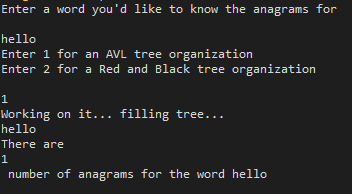
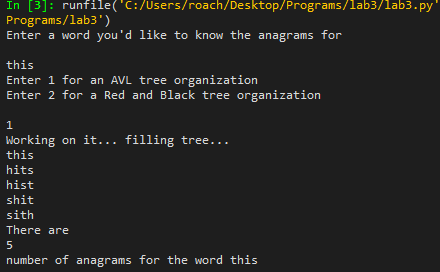
Report

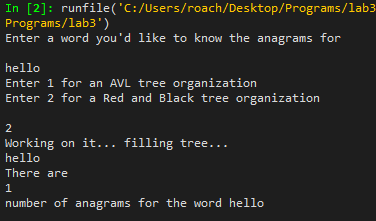
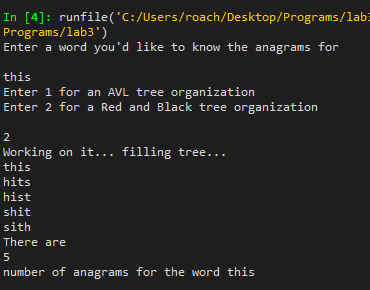
Introduction: The problem that I am trying to solve is that of adding all words of a given list into either an AVL search tree, a Red and Black search tree or a B-Tree which then is used to get all the possible anagrams from any given word

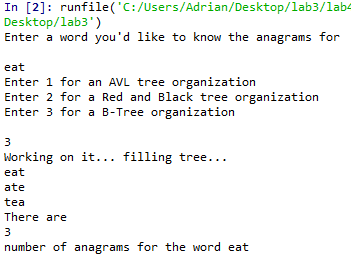
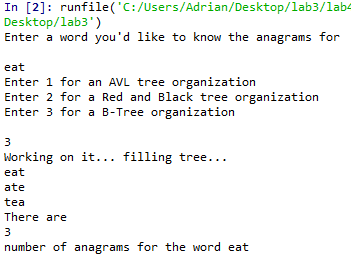
Proposed Solution(s):

The proposed solution is to create the two different data structures, one being an AVL tree as well as a RnB tree and a B-Tree. All trees would be created with all the necessary methods in order to be able add as well as a few other methods in order to function in a complete and thorough fashion. The more important methods that must be in either data type are the insertion and rotation methods, since those are the ones that balance and organize the tree after adding on the extra node. After the data types are created, the words are filled into a list, which is then used to fill the respective tree based on the decision of the user. After the user has chosen what type of tree to use, the word that the user wants to find the anagram for is searched though the tree, which if found is then put into a method that creates its anagrams. If the anagrams are within the tree that was chosen, they are added into a count that keeps check of the total number of anagrams for the word.

Experimental Results:



Conclusion:

While both trees are effective in holding and organizing data in their own unique way, it most certainly depends on what one would feel is more useful when it comes down to the organization of the data. The Black and Red tree is effective in a situation where one would want to keep the main roots as well as subroots of a tree, maybe storing a main source of data. The AVL tree is simply useful in the sense that it is a balanced tree that organizes data in a rather simple and neat way. A real neat and useful data structure