Nancy Hernandez

Diego Aguirre

Lab 4 B Report

**Introduction**

The purpose of this lab was to gain experience working with B-Trees.

**Proposed Solution**

In order to build the B-Tree, I copied and pasted the methods that were provided on blackboard. These methods included find\_child, insert\_internal, split, insert\_leaf, leaves, insert, height, print, search and a few others that I did not use.

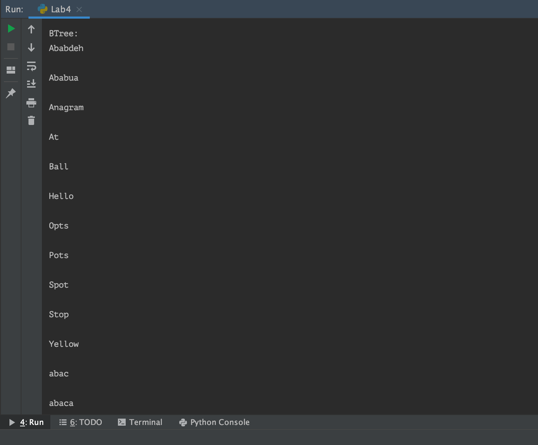
Like the AVL and Red Black Trees, I created the methods print\_anagrams, count anagrams, biggest anagram, and a read\_file. For the print anagrams I used the code provided in the lab 3 instructions. For the count\_anagrms method I copied the whole print\_anagrams method except instead of printing the anagrams I used a counter and returned it.

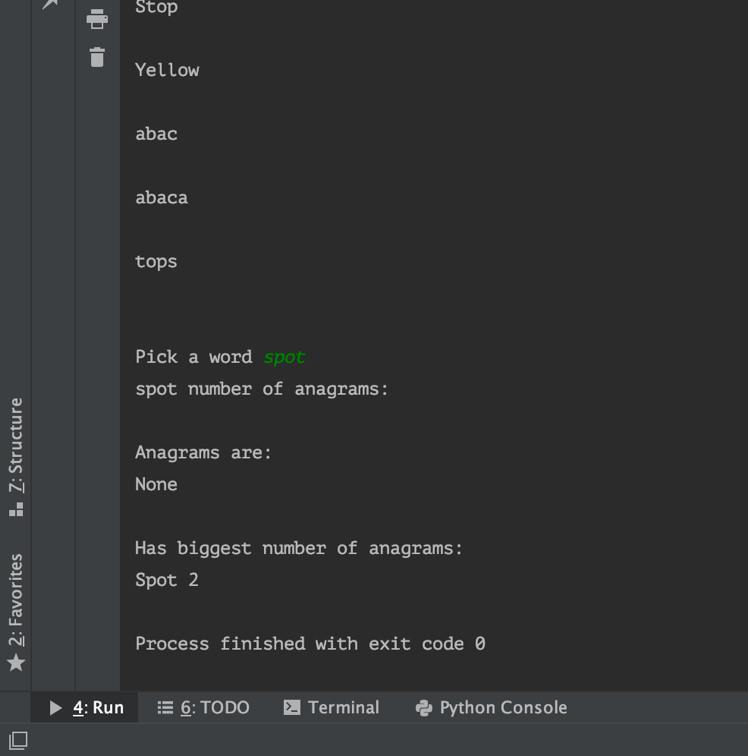
Next I created the biggest\_anagram which first begins with reading the file. I created a variable “biggest” which will be a place holder to compare in the future. I also created a variable called word that is currently just empty. I then carried on with a for loop that goes through the whole file and inside I call the count\_anagrams method that return the count. Once I have that value for the anagrams I begin comparing to find the largest number. Once found, I print the word that corresponds with the largest amount of anagrams.

In the read\_file method, like the biggest\_anagram, I also begin by reading the file. This method is in charge of not only reading the file but also actually creating the B-Tree. I created a variable “btree” and its set equal to the class BTree. Now that that is declared I created a for loop which goes through every line in the file and inserts every word into my variable “btree”. Once every word has been inserted into the tree, I return the final “btree”.

Finally, in my main method I ask the user to pick a word so that I can apply the methods that I created. In this method I will call the count\_anagrams, print\_anagrams, and biggest\_anagram methods.

**Results**

****

****

**Test cases**:

I used the word “spot” to test the methods print\_anagram, count\_anagram and greatest\_angram. I did not use the original file with 10,000 words as it took too long to run. Because of this I used a text file with less words in it.

**Time Complexities:**

AVL - O(log n)

Red Black - O(log n)

B-Tree – O(log n)

**Conclusion**

From this lab I learned how to implement B-trees, how to use them and also when to use them. I would use a B-Tree instead of an AVL tree or RB tree when I want to insert things faster or look for them faster.

**Appendix**

def print\_anagrams(word, prefix=""):

if len(word) <= 1:

str = prefix + word

if str in engish\_words:

print(prefix + word)

else:

for i in range(len(word)):

cur = word[i: i + 1]

before = word[0: i] # letters before cur

after = word[i + 1:] # letters after cur

**Academic Dishonesty Statement**

I, Nancy Hernandez, was not involved in any copying from or providing information to another student, possessing unauthorized materials during a test, or falsifying data in laboratory reports. Neither did I participate in any type of collusion involving collaboration with another person to commit an academically dishonesty.