**Problem Description:**

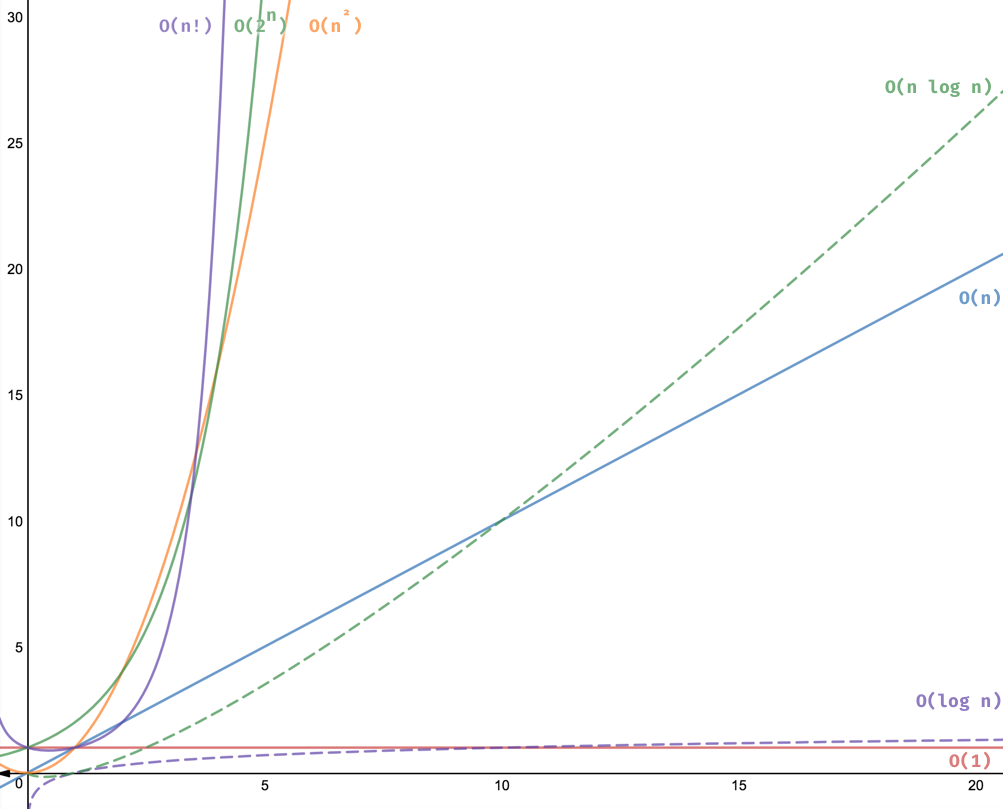
We have to read a file of over 466k thousand words and create a data structure called english\_words using a binary search tree (AVL and Red Black) that stores all of the words.

**Solution design and implementation**

All of the pseudocode for AVL and Red Black trees were already given to us on zybooks. The only challenge was to implement it all in Python and make it work together. I put both classes in the same file since the program could use any of them based on what the user chose. Apart from the given implementations, the only thing the code does is ask the user for what type of tree they want to use and then they insert all of the words in the given text file into that chosen tree. The print anagrams function was also given to us by Diego so we just had to make sure everything worked together.

**Experimental results**

I implemented a counter to give myself and idea of how fast it is inserting words. I used 2 different text files, one with the entire given set of words which was 466552 words and another one with just 1000 words. I then tested each file with each type of tree and compared how long it took to insert all of the words. When testing the small file, both trees seemed to insert pretty much instantly. When testing the larger file however, you can instantly notice a difference. The AVL tree inserts in linear time while the Red Black inserts in exponential time. I didn’t even wait to see how long RB would take because it definitely took more than 10 minutes on my machine while the AVL tree took less than 30 seconds.



Red Black Tree

AVL Tree

**Conclusion**

AVL trees are way faster than RB trees when it comes to insertion. This lab helped me realize how two similar data structures can have such different time complexities when it comes to different functions

**Source code**

<https://github.com/lebriones/CS2302/blob/master/Lab3.py>