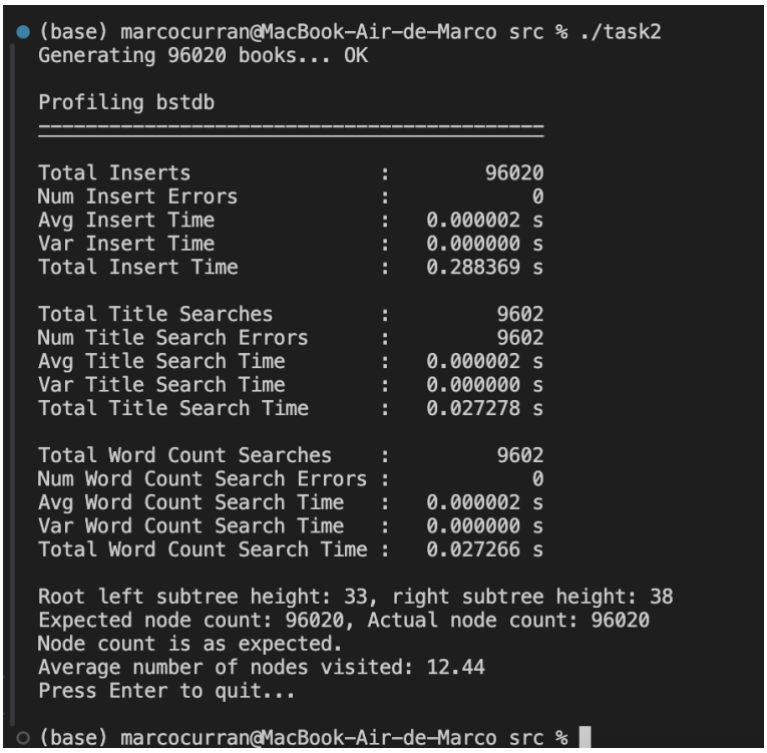
Report Assignment 4

Part 2

For this assignment in part 2, we were asked to implement a BST that would accumulate nodes from a structure of books with their titles. Then, with the use of other files and a separate main function, we would run some parameters that will tell us how our BST is working. Some of these are: total word count searches, total title searches, or total title search errors. After completing the program, this is the output I got:



Here, we can see how the BST implementation worked and our program is running successfully. Keeping the tree balanced was the most difficult part of this assignment, as there are many ways to do this but some can be very complicated. A more complex approach would have been for example using an AVL which balances by itself, but has more complicated code. After trying several approaches, like calculating the height and a midpoint to give the doc_id a specific value. The approach that is more simple and ended up working for me was using the rand() function. As we know, this function gives a random value to our variable. This would ensure our tree wasn't just a one sided tree and resulted quicker than a linked list. To ensure that we had no repeated values, I added a separate if statement to change the doc_id until it gives a number that hasn't been used. As we can see in my result, the resulting BST is pretty balanced and would work faster than a linked list.

Lastly, we have a specific function in our code that adds as much extra information about our tree as we want. My choice, was to add a few functions, that would allow us to calculate: the height of the tree on each side, if the expected and actual node count are equal, and the average number of nodes visited. The results to this function are seen in the paragraph under the table. Where we can see the comparison of node count, the height of the tree on each side, and the average number of nodes visited.