Sangini Shah and Rob Williams

CS214 Programming Assignment 3

Indexer- Readme

In order to implement our indexer we used a multiple layered data structure. The larger structure is an array of 26 SortedListPtrs, one list for each letter of the alphabet. This structure is referred to as the Indexer and works like a hashmap, where a token can be hashed based upon its first letter, in which the list will then insert it in its appropriate lexicographical location. The Indexer's SortedLists hold another structure referred to as TokenNode. This TokenNode holds a string which is the token and a pointer to another SortedList which contains a list of FileNodes. These FileNodes have two elements, a string which holds the file name and an integer that counts the frequency of the given token in that file. The FileNodes are sorted based upon frequency and then their file names.

Due to the multiple insertions and iterations required to populate the Indexer and then produce JSON from it, the runtime of this program is not very efficient. Assuming there are f files with n unique tokens with each token being in m files, the insert would be O(m) time for the insert of the FileNode multiplied by O(n) time for the insert of the TokenNode multiplied with O(f) for iteration over all f files. Assuming n = m = f, the runtime of this program would be quadratic $O(n^3)$. Once the Indexer has been filled, it needs to be iterated through to print out the JSON. Again, each list and its sublist must be iterated, giving a runtime of $O(n^2)$. Summing these two values with an O(n) runtime for tokenizing gives us a total runtime of $O(n^3)$. This would prove bad for situations where we have multiple and large files, but is fast enough to run on small directories and files.