Systems Programming Project 3 Readme (Daniel Schley and Ryan Dunn)

With this project, we were assigned to create a program that took a directory or file as input and returned a list, in JSON format, of how many times each word appeared in each file. I believe we have done that very successfully. When given the directory, the program recursively searches it. Each file that it reaches is tokenized one token at a time. That token, once received, is inserted into our data structure (a list of “word” structs, each pointing to a list of “occurrence” structs). The insert method goes through the list of words until it either finds the token, in which case it adds a new occurrence, or has passed where it would have been, at which case it creates the word and then adds the occurrence. The runtime of this application, given n total words, is $O(n^2)$. This is because the worst case for each new word is that it is either a new word at the end of the existing words alphabetically or it is the last word in the last possible occurrence. Because the amount of files doesn’t change how many structs we have to pass though, this becomes the longest process and in turn the big O.

In terms of space usage, we still have a maximum of n^2 structs. With the words being character arrays, we can’t necessarily say how many bytes each will be, but the other two pointers will be 8 bytes each, and taking a rough estimate at an average of 6 letters per word (plus the end character) that gives us $23n^2$ bytes of data usage maximum, which isn’t too bad.

A further note, we implemented two possible ways to store the words from the different files. In our alternative method, files with the same name in different directories will be treated as separate files and the name that is outputted will be the path name instead. If you, the grader, would like to consider this alternative method for extra credit, uncommenting line 293 and commenting line 292 of our indexer.c file will give you this result.