Indexer

Alison Wong

Patrick Wu

Design:

In this assignment, a prefix tree was used to store the data retrieved from the files. Each Prefix Node whose isWord is true, stores a linked list which holds the file names and the number of occurences for each word/token. There is a struct that contains the root of the prefix tree which has access to the rest of the tree. New nodes are added to the end of the linked list and is sorted after having gone through/read through all the files. The pathnames are used rather than just the files names in order to clarify which file in case their are two files with the same names but within different directories. There is a check if there already exists a file/directory with the same name as the indexer file. The user is prompted whether or not to overwrite the file/directory that has the same name as the indexer file. If a directory is being overwritten, the directory and all its files and sub-directories will also be deleted. There is also a check to see if the given file/directory in the second argument even exists, if not, and error will be given and the program will end. Once the inverted index is made, all the files/directories are closed and all the previously allocated data structures are freed.

Space Usage:

In this assignment, we allocate memory for each node in the prefix tree and the array of 36 for each prefix node that we allocate memory for. We allocate memory for each linked list node within the prefix nodes that are the end of a word. Memory is also allocated for a struct that holds the root of the prefix tree. Two character arrays are also allocated memory in order to concatinate the inverted index output. Another character array has memory allocated in order to print the words/tokens from the prefix tree. Any data structure that had memory allocated is then freed.

Efficiency Analysis: