READ ME- SEARCH

Rahul Purwah
Gregg Benjamin

In order to successfully code this assignment we used a hash table which is composed of a special struct. We called this struct Information. This struct has 4 parameters. The parameters are and int, char*, SortedList*, and a UT_hash_handle struct which makes the item hashable. The int parameter is used as a key for the hashtable. The char* is the actual word which was found in the file. This word is what the searches are looking for. The SortedList holds all the file names which were found In the directories. The last parameter is used to make the item hashable.

MEMORY ANALYSIS

The slowest part of our code is the implementation of the Hash Table into a sorted list. The run time for this can said to be O(number_of_files*number_of_words_in_file). The SEARCH AND first looks up the entire list of file names which takes O(number_of_files*number_of_words_in_file). This list is then compared with the same list for another word which doubles the run time. Therefore it is O(number_of_files*number_of_words_in_file)^2. The SEARCH OR only looks for the list for all the names once. OR does not have to check for the common names between 2 lists but it does have to check for duplicates therefore in worst case it ends up being O(number_of_files^2* number_of_words_in_file^2)

Index File Format

<list> 1820
long.txt 1
</list>
<list> a
long.txt 56 derp 3 baa 1 boo 1
</list>
<list> abilities
long.txt 1
</list>