# Approach:

As we are traversing all the words in the file, if no of words are n , complexity is O(n)

Secondly at each step we are searching if word exist in the file , complexity is O(log n) for find in map

else we are inserting word in the string , inserting in map complexity is O(log(n)) n = no of elements in the map

Overall Time Complexity of my algorithm is O(n.log(n)),

Space complexity: O( no of unique words)

# Alternative approach

Another technique can be read each string from the file store it in word, then read all the strings in the file and check how many times that string exist

in the file, similarly loop for all the words in the string.

Time Complexity of that algorithm will be O(n^2) given n is number of words in the file.

My algorithm is better in terms of time complexity then 2nd approach.

We can also use unordered\_map in place of map, map are implemented using Binary Search Tree while unordered map are implemented using hash table.

For unordered map , to find a word in map , best case time complexity is O(1) and to insert in map is also O(1), overall complexity of algorithm then would be

O(n), but worst case time complexity of find in unordered map is O(n ) and also to insert element in map is O(n), which would have given overall complexity to be

O(n^2) . So i have not chosen unordered\_map .

Currently my program is treating mis-spelled word as a different word, if we dont want to treat mis-spelled words as new word, then i should know the criteria , how

words can be mis-spelled. like if hi is mis-spelled as Gi, G being neighbor of H, there was a problem in typing H. I shoould Know this information to not treat

mis-spelled words as new words.