## Algorithms used

There were no crazy algorithms used in this project. The only recognizable algorithms used are towards the end of the program when we are sorting the linked list.

The merge sort algorithm is applied to sort our linked list in O(N \* LOG N) time. Which is helpful considering we use an algorithm in O(N \* K) time. Where N is the number of words in the text file multiplied by the number of words we've stored in the list already. since we iterate over every word then iterate over every we've stored already in the list to increase it's frequency.

I chose a singly linked list implementation where each node holds a the Word struct. In hindsight it would've been more beneficial to simply use a hashmap. The linked list implementation did have it's own benefits though, seeing as it did not have a fixed length of elements like an array usually would.

### Time complexity

As stated previously, the time complexity of the sorting algorithm is in O(N \* Log N) time, and the time complexity of the algorithm used to count the words and their frequency is in O(N \* K). Where N represents the number of words in the text file, and K represents the number of Nodes in our linked list.

# Implementation status

Complete

# Improvements

A general improvement could be a change of data structures such as a hash map or any other data structure that can be indexed, other than an array. Such as an array list. This would save a few seconds in our sorting process seeing as we are using merge sort and need to find the center node.

# Compatibility

This program is compatible with text files as large as the Alice.txt file provided, however tests on larger files have not been conducted. Presenting an unlikely possibility.