Henry Wandover
Data Structures

Lab 7: Frequency Counting

For this lab I was given a sequential search algorithm that added words to a circular linked list, and then would comb through that list to increase the value (amount of times a certain word appears) or in the end print out the results. The skeleton of the self organizing alternative is based upon the same principle of getting item values and putting keys into nodes in the list, however, self organizing is based upon a heuristic. In both the put and get methods, the combing through the list is based on recently accessed items are at the front of the list. Where the former algorithm, is based on calling of methods rather than any conscious ordering of the list as seen in the self organizing that has been written.

Looking at the two tests that I have included in both algorithms I can confidently say that the self organizing symbol table approach is more effective at it's job. For the tests I used the novel *Ulysses* by James Joyce, which clocks in at around 265,000 words and cyphered through the text with both algorithms. Looking at runtime SeqSearch took a quite cumbersome 142,563,178,300 nano seconds or 142.56 seconds, over two minutes. While SelfOrganizing took only 27,030,383,700 nano seconds or 27.03 seconds, a starch difference. There was some problems with using another method (I originally used a separate method to find the previous node which took much more time, making SelfOrganzing slower) but now it operates a satisfactory speed. For the other test, I ran a comparisons analysis. SeqSearch made 12,595,034,346 comparisons while SelfOrganizing made 1,771,530,860. Obviously a tenth of the comparisons made is something that's sought after, and that also contributes to why this algorithm is preferred when using symbol tables to find the most frequent words in a text corpus.