DATASTRUCTURES AND ALGORITHMS

COURSEWORK 1

Mark Gordon

Program Specification

Create a simple spell-checker program. It should take in two files, a dictionary txt filled with the correct spelling of words and another txt file to correct the words if necessary. The program should read the words of both into two separate maps. The checkmap should then be compared to the dictionarymap and with each word compared there should be several checks to provide corrections if the word has been misspelled. These checks are:

- Letter substitution – go over all characters in the word replacing the current character with all characters in the alphabet one at a time and then do a comparison to the dictionary map to see if there are any matches.
  Example : dat -> cat
- Letter Omission – go over all characters in the word omitting one each time and comparing the new word to the dictionary map to see if there are any matches.
  Example : cpat -> cat
- Letter Insertion – for each letter of the alphabet add to each index one at a time and compare new word to dictionary map to see if there are any matches.
  Example : ct -> cat
- Letter Reversal – Swap adjacent characters for all possibilities and check new word to the dictionary to see if there are any matches.
  Example : cta -> cat

If any matches are found during the modifications of the word, then print out the original word and how its modified spelling is.

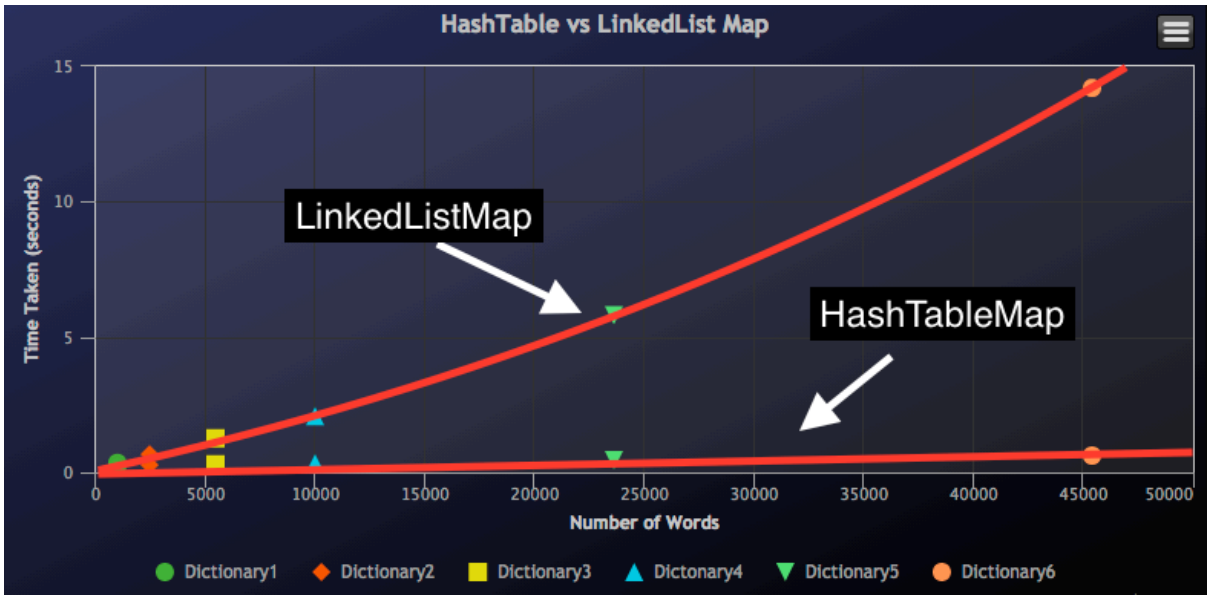
IDE USED

IDE used was Mars Release (4.5.0).

Design Choices and Limitations

- Program passes provided test
- Program passes self done tests with d1-d6.
- HashTableMap and LinkedListMap follows expected time complexities
- Program passes 5 parameters to the word modification methods, this is high coupling and there should be a better way of doing this.
- Could not get HashTableMap to work when I left the array empty, I had to fill with "-1" by default and when a value is deleted it has to be filled with the string "deleted". When I used .isEmpty() instead this caused problems and would get stuck in the loop for reasons beyond me.
- I've also had to change IHashCode to take in an integer, which I pass the array size to. I tried to avoid this issue by making a getter for arraySize and also making arraySize and public variable and just calling it instead but I couldn't get either to work. This was the only fix I could get to work.

HashTableMap vs LinkedListMap



- LinkedListMap follows quadratic time as expected
- HashTableMap follows linear time.
- LinkedListMap isn't a problem at low word count but the greater the word count the efficiency greatly decreases. It can enter the words into the map quickly, however as it needs to search the array by each index this is very slow.
- HashTableMap enters the words fast but also finds the word very quickly as well, this is due to performing a calculation on what's stored and using that as the array index value so when finding a work you will do the calculation on that word and search in that array index, if its not there then you simple need to increase the array by the declared step distance until you find it or the array index has a "-1" in it meaning the word is not in the list.
- Overall HashTableMap is a much better choice for this example due to its greater efficiency as the word count increases.