Name: Jennifer Phuc Tran
Date: 10/22/2021
Exercise 1 Report

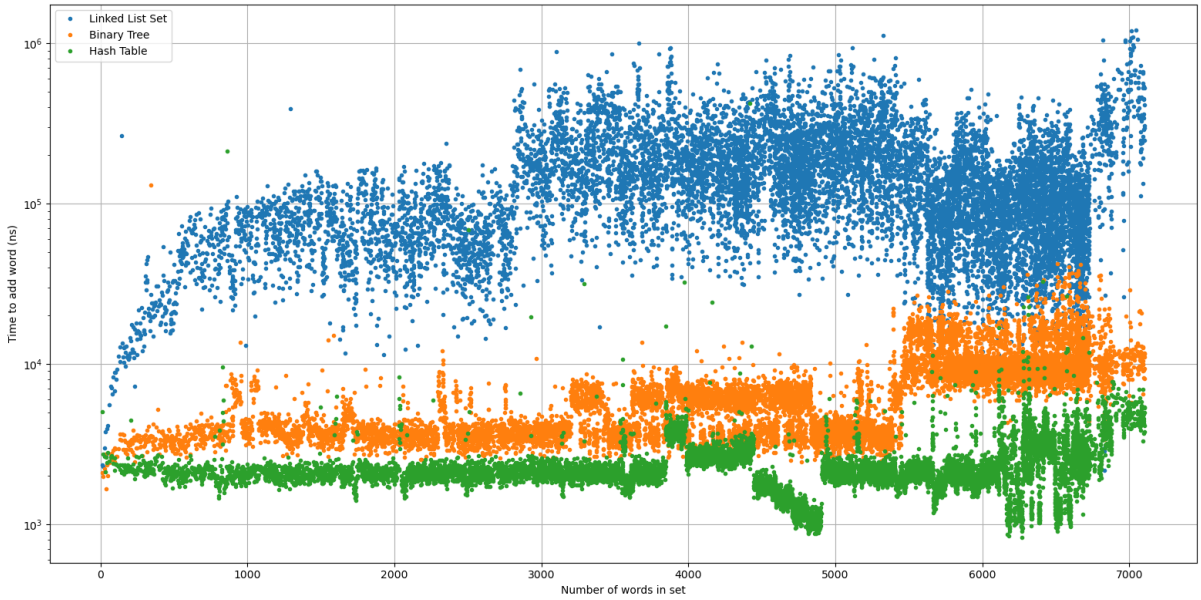
EXERCISE 1 REPORT

**My findings:** After inserting words into sets using the Boolean add (String word) method, the number of words that are present in the sets is **7106.** The same count is found for each set generated using Linked List, Binary Tree and Hash Table data structures. Then, using Boolean contains(String word) method with the implementation of the 3 data structures again, the number of words do not exist in the set is **5**.

Conduct experiments to assess the relative performance of the set implementations:

**Experiment 1:** Measure how much time (in nanoseconds) it takes to insert each word in the sets. Matplot lib in Python was utilized to plot a graph that shows as independent variable (x-axis) the number of words already inserted, and as dependent variable (y-axis) the time it took to insert it.

The experiments were run 10 times using Python code to output data to a .txt file, and manually copied to excel sheets Linked List, Binary Tree, and Hash Table in an Excel workbook.




*Graph shows number of words inserted in the set vs time in nanosecond to add a word to the set*

Based on the graph, as the number of words in the set increased on the x-axis, Hash Table (green data points) proved to be the most efficient algorithm time wise that took the least amount of time to insert a word in the sets. Binary Tree (orange data points) proved to be the second most effective algorithm, and Linked List data structure (blue data points) took the most time to insert a word in the sets.

**Experiment 2:** Measure how much time it takes to search for a word. You should detect the worst case, the best case, and the average of all words.

Again, the code was run 10 times to produces the average time to search for a word, average worst case search time, and average best case search time for each data structure.

| | Average time to search for a word | Worst case average search time | Best case average search time |
|---|---|---|---|
| Linked List | 434861.296 | 4795710 | 590 |
| Binary Tree | 12367.37989 | 91480 | 1360 |

| | | | |
|---|---|---|---|
| Hash Table | 3561.592527 | 42740 | 4050 |

Based on the data collected, Hash Table is the fastest algorithm to search for a word and Linked List is the slowest of the 3 algorithms implemented.

**COMPLEXITY ANALYSIS:**

The data matched with the theoretical analysis of each data structure's time complexity:

**Linked List:** O(n). To insert a word, we need to go through the entire set of linked nodes to check if the word exists, then add the word to the list if it is not already in the list. As the list gets longer, it takes more time to add a word because there are more nodes to check.

**Binary Tree:** O(logn). To insert a word, we need to check if the word is there, then compare it to the parent node to see if we put the word in the left or the right of the parent. Since we don't need to visit every node, but only left or right of parent, this data structure will take less time than Linked List theoretically. The data showed that this data structure took less time than the Linked List.

**Hash Table:** O(1). For this data structure, what needed to be done was to find the key, and then look up the value in the hash table. This will take O(1) time. The data showed that this data structure took the least amount of time out of all 3 data structures implemented.

Since O(n) > O(logn) > O(1) by theoretical analysis, and the graph shows the time accordingly, the theory matched with the actual data.