LinkedListSet

- Insertion & Search
  - Time Complexity: O(n)
  - To insert a new word, we have to visit every existed element and add the new val to the end of the linked list. But if the word exists, we can stop searching when we find it.
  - If the val we are searching does not exist, we have to visited all of the elements

BinarySearchTreeSet

- Insertion & Search
  - Time Complexity: O(logn)
  - We need to traversal to the leave node of the BST to insert a new word. But if the word exists, we can stop searching when we find it.
  - If the val we are searching does not exist, we have to traversal to the leave node.

HashTableSet

- Insertion & Search
  - Time Complexity: O(1)
  - The size of slots is 10000, which means there's hardly any element in the same slot. As a result, both the insertion and the search operation only takes constant time.

Summary

There are 7106 unique words in the novel, and 140 words in the words-shuffled (without deleting signal '_') did not exist in the set.

The data obtained from the experiment roughly match the theoretical analysis. Because the amount of data is big enough. As shown in the graph: the points in the graph of LinkedListSet is approximately linearly distributed, and those of the graph of BSTSet is approximately logarithmical. While insertion time of HashTableSet shows no obvious relation with number of elements already inserted

However, because of the repeated appearance of words like "a", "I", "is", "he", "she", "and", etc., the already inserted words in an article may be much more than the new ones (In our sample, there're 120k words and 7k unique ones). So that the Insertion time of BinarySearchTreeSet and LinkedListSet may be shorter than expected.