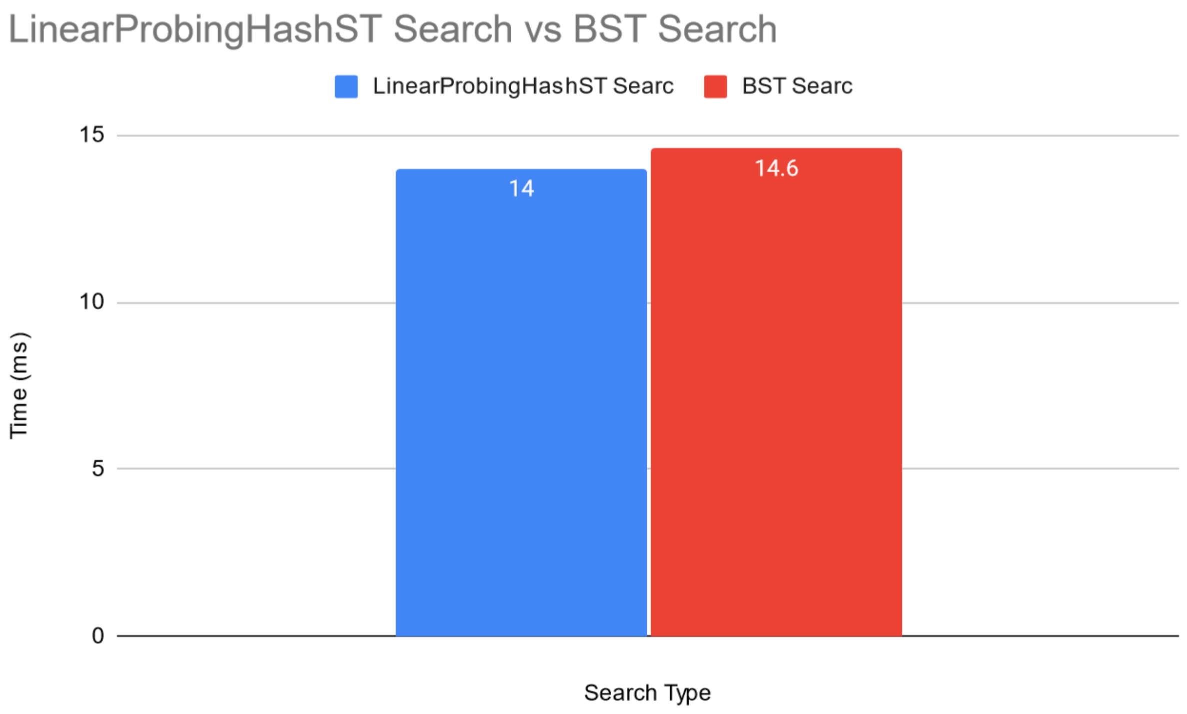
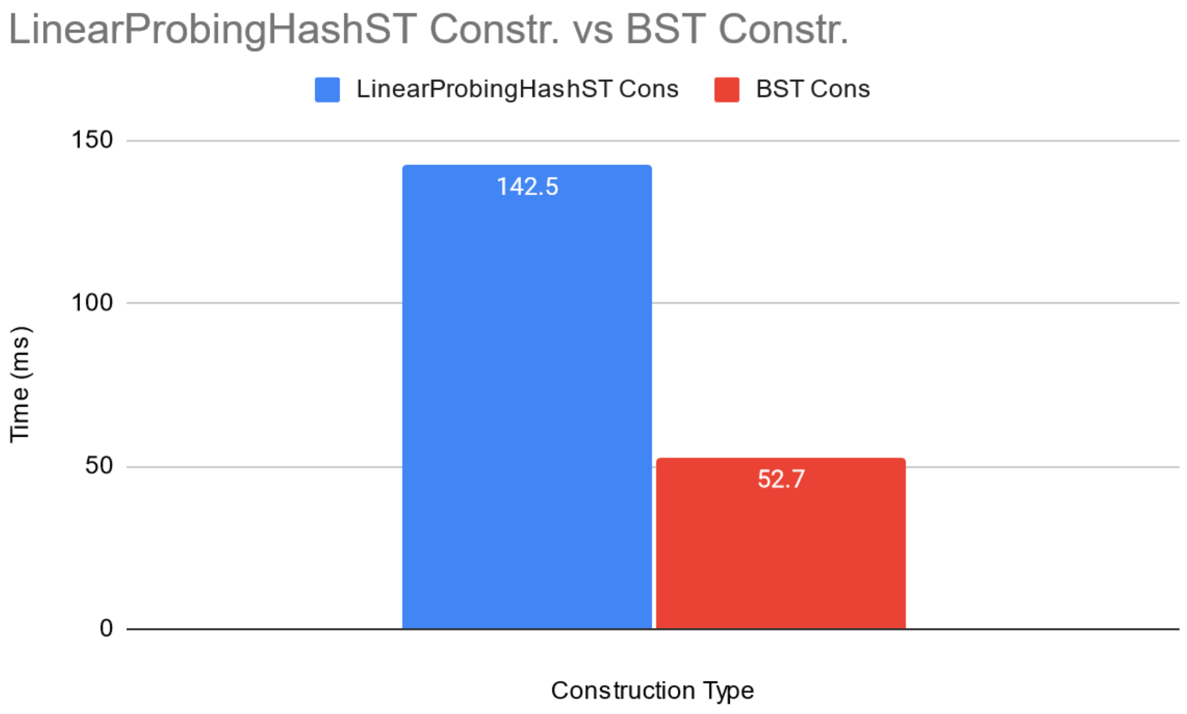
Conner McKevitt





LinearProbingHashST Search		LinearProbingHashST Cons		BST Search	BST Cons
13		137		14	56
15		138		16	53
14		133		13	48
12		137		14	50
16		161		15	54
15		145		16	54
14		137		16	56
14		136		15	51
13		145		14	49
14		156		13	56
LinearProbingHashST Search Avg		LinearProbingHashST Cons Avg		BST Search Avg	BST Cons Avg
14		142.5		14.6	52.7

Data was collected for searching by running search with an input of 1/10 n random strings and repeating that 10 times. Data for construction was collected by construction the nodes 10 times for each algorithm

In terms of methods LinearProbingHashST uses a hash system with extra space to help deal with collisions while the BinarySearchTree puts the values into nodes then places them properly into their place in a binary tree.

Searchwise both algorithms appear to be around the same with a difference in search of .6 ms. The construction of the Binary Search Tree on average takes 89.8 ms less than the LinearProbingHashST. This would be very important for a larger data set/

For my company I think BST would be the best algorithm to use because it is already ordered with the highest score in node 1 so retrieving the most unique word (highest TD-IDF score) will be much faster than using a hash. Since hash is not in any specific order another sorting algorithm will be used to find the most unique words. BST also constructs much faster which will aid in searching through larger plays and works. The company is attempting to find the most unique words so that they can save money so BST seems to be the best option. Have a faster construction time will be a good selling point for the algorithm to potential buyers as well.