Vamsi Kalidindi
Dali Xiao

Project 1 Analysis

Method	BST Times (sec)	Worst Case time Complexity of BST	Hash Table Times (sec)	Worst Case Time Complexity of Hash Table
Search (100 searches)	0.0000531	O(n)	0.0012	O(n)
Insert (100 inserts)	0.0079	O(n)	0.00048	O(n)
Delete (100 deletes)	0.0029	O(n)	0.0003	O(n)
Sort	0.010651	O(n)	0.020839	O(nlogn)
Range Query (n=10)	0.001194	O(n)	0.033677	O(n^2)
Range Query (n=100)	0.001531	O(n)	0.036837	O(n^2)
Range Query (n=1000)	0.001165	O(n)	0.035032	O(n^2)

Figure 1

The results we tabulated according to functions for search, insert, delete, sort, and range were analogous to the theoretical worst case results found in real-time applications. For the Binary search tree, typically search, insert, and delete should be O(log n) running time. However, if we were to analyze the worst case, there is a possibility of iterating through every node in the BST in order to perform those operations. Similarly, Sorting the BST and performing a range query would be worst-case O(n) because there is the possibility that all the tree nodes must be iterated through assuming it is not balanced. For example, the range query could require going through the last 1000 nodes in the BST when iterating through (Range query n=1000) and outputting the results. However, this only happens in rare cases.

For the Hash table, range query was a little different compared to that of the BST because we dealt with two for-loops in our range function. Worst-case, these would translate to (n*n) which is n^2. The BST differs from this in that we are only going through if-loops to determine which nodes to iterate through. That is why there is a significant difference in total time. We see O(n) for insertion, deletion, etc because there could be a worst-case of collision within every operation, which would translate the code into a linear search, insertion, deletion. However, this is not generally the case as seen by the results. By using the hashing method, we get much faster times for search, insert, and delete compared to that of the BST because hashing does not require making as many comparisons. Sorting in Hash Table would take O(nlogn) because it take O(n) time to take all the words into a container, and O(n log(n)) time to sort the container, so the overall time is O(nlog(n))