Fuheng Zhao, Runyu Gao

CS 130A

Project 1 Part 2

Method	**BST time**	**Worst case time complexity of BST**	**Hash Table time**	**Worst case time complexity of HT**
Search	1.1e^(-5) sec	O(n)	7e^(-6) sec	O(n),
Insert	1.6e^(-5) sec	O(n)	2e^(-6)sec	O(n),
Delete	1.2e^(-5) sec	O(n)	4e^(-6)sec	O(n)
Sort	0.0175 sec	O(n)	0.0280 sec	O(nlogn)
Range query (n=10)	5.8e^(-5)sec	O(n)	0.0064 sec	O(n)
Range query (n=100)	0.000328 sec	O(n)	.00568 sec	O(n)
Range query (n=1000)	0.0045sec	O(n)	.0103 sec	O(n)

Analysis:

In this project out implementation of BST did not ensure a balanced BST, and thus in the worst case, the BST can be very skewed and looks like an array, in which you need to look all data for search, insert, and delete. In hash table in the worst case, all data may be clustered at a single part of the table. Therefore, you might need to look at all data to do the search, insert, and delete.

From this table, we can clearly see that there are both advantages and disadvantages in both data structure. BST is quick at range query, since it doesn't need to look to the subtrees which cannot contain any possible string. Hash table is not a well data structure for sort, since you need to first get all elements and then do a quick sort, or other sort algorithm. However, hash table, may be good for search in theory. With a good hash function, search and insert can be very fast.

Besides these analyses based on the table, we also need to consider the memory. BST is memory-efficient in which it uses n nodes compare to that of the hash table.