Method	BST Time	Big –O BST	HT Time	Big-0 HT	AVL Tree Time	Big-O AVL Time	2-5 Time	Big-O 2-5 Time
Search	171	O(n)	63	O(n)	176	O(logn)	159	O(logn)
Insert	93	O(n)	30	O(n)	91	O(logn)	223	O(logn)
Delete	167	O(n)	57	O(n)	211	O(logn)	264	O(logn)
Sort	63739	O(n)	447300	O(nlogn)	62439	O(logn)	62477	O(logn)
Range Query (n=10)	71	O(n)	4283	O(n)	108	O(n)	341	O(n)
Range Query (n=100)	52	O(n)	4172	O(n)	53	O(n)	314	O(n)
Range Query (n=1000)	190	O(n)	4179	O(n)	211	O(n)	264	O(n)

All times are in microseconds
Everything was run on our personal machines

We found that the hash table is really efficient with searching, inserting and deleting. However the BST is more efficient for ranged searching and sorting. This is because the BST stores all nodes in a sorted manner, whereas hash tables focus on fast and immediate entry over organization. AVL trees are comparable to BSTs but are slightly faster when compared to 2-5 trees because they don't lose as much time in memory allocation (except for search). Search is 2-5 trees is faster because it doesn't have to traverse as many "levels" as the AVL tree has.

The range queries can be made more efficient in BSTs because we only look for nodes based on whether they're inside the range. The hash table, on the other hand, has to iterate through all values. The AVL tree and BST have comparable running times because they use the same algorithm. The 2-5 tree is slower because it has to iterate through the individual values in the node which can cause it to slow down.

Anu Polisetty and Lawrence Lim