|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Method | BST Time | Big-O BST | Hash Table Time | Big-O Hash Table | AVL Tree Time | Big-O AVL Tree | 2-5 Tree Time | Big-O 2-5 Tree |
| Search | .000133 seconds | O(logn), where logn is height of tree | . 0039817 seconds | O(n), where n is the size of the table | .000199 seconds | O(logn), where logn is height of tree | 0.00028  seconds | O(logn), where logn is height of tree |
| Insert | .00017 seconds | O(logn), where logn is height of tree | .035455 seconds | O(n), where n is the size of the table | .000238 seconds | O(logn), where logn is height of tree | 0.000247  seconds | O(logn), where logn is height of tree |
| Delete | .000138 seconds | O(logn), where logn is height of tree | .011626 seconds | O(n), where n is the size of the table | .000281 seconds | O(logn), where logn is height of tree | .04953 seconds | O(logn), where logn is height of tree |
| Sort | .022591 seconds | O(n), where n is the total number of nodes | .042087 seconds | O(nlogn), where n is the size of the table | .046486 seconds | O(n), where n is the total number of nodes | .048232 seconds | O(n), where n is the total number of nodes |
| Range Query (n=10) | .000017 seconds | O(n), where n is the total number of nodes | .003914 seconds | O(n), where n is number of nodes | 0.000006 seconds | O(n), where n is the total number of nodes | 0.00746 seconds | O(n), where n is the total number of nodes |
| Range Query (n=100) | 0.000169 seconds | O(n), where n is the total number of nodes | .003328 seconds | O(n), where n is number of nodes | 0.000437 seconds | O(n), where n is the total number of nodes | 0.006388 seconds | O(n), where n is the total number of nodes |
| Range Query (n=1000) | .00125 seconds | O(n), where n is the total number of nodes | .00406 seconds | O(n), where n is number of nodes | 0.004184 seconds | O(n), where n is the total number of nodes | 0.005055 seconds | O(n), where n is the total number of nodes |

Analysis

AVL seemed to be Worst case for AVL would be O(logn) (logn is height of tree) for all operations, while worst case for 2-5 Tree is O(logn) (logn is height of tree) for all operations because of the possibility of merging to the root/ splitting to the root. These two data structures, therefore, have the same worst-case scenarios. The time for AVL and 2-5 Tree to perform search, insert, sort, and range search seem to be about the same. The delete operation seemed for 2-5 took much longer time. This may be due to how delete was handled or because of which words were selected for testing.

Overall, it seems that BST has the quickest operations (probably because of the lack of the balancing factor AVL has). However, since this is processing only 100 words for each operation, the run time for a large sample size should result in similar times for BST, AVL, and 2-5.