Benson Chu

2/17/18

CS 130-A

Programming Assignment #1 Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **BST** | **Worst case time complexity of BST** | **Hash Table** | **Worst case time complexity of HT** |
| Search | <time taken to perform 100 searches>  .000133 seconds | O(logn), where logn is the height of the tree | <time taken to perform 100 searches>  . 0039817 seconds | O(n), where n is the size of the table |
| Insert | <time taken to perform 100 inserts>  .00017 seconds | O(logn) , where logn is the height of the tree | <time taken to perform 100 inserts>  .035455 seconds | O(n), where n is the size of the table |
| Delete | <time taken to perform 100 deletes>  .000138 seconds | O(logn) , where logn is the height of the tree | <time taken to perform 100 deletes>  .011626 seconds | O(n), where n is the size of the table |
| Sort | <time taken to sort all the words>  .022591 seconds | O(n), where n is the total number of nodes | <time taken to sort all the words>  .042087 seconds | O(nlogn), where n is the size of the table |
| Range query  (n=10) | <time taken to range query n words>  .000017 seconds | O(n), where n is number of nodes | <time taken to range query n words>  .003914 seconds | O(n), where n is number of nodes |
| Range query  (n=100) | <time taken to range query n words>  0.000169 seconds | O(n), where n is number of nodes | <time taken to range query n words>  .003328 seconds | O(n), where n is number of nodes |
| Range query  (n=1000) | <time taken to range query n words>  .00125 seconds | O(n), where n is number of nodes | <time taken to range query n words>  .00406 seconds | O(n), where is number of nodes |

**Analysis:**

In every case, the BST was quicker than the hash table. Perhaps the hash function was not implemented efficiently enough, or the functions themselves were not implemented efficiently enough. However, considering the load factor being 2/3, multiple collisions were bound to occur, with a total count of 570,000 words. After implementing both data structures and comparing their functions’ time complexities, it is suffice to say that much further study is needed in optimizing the hash table’s well-known O(1) time complexity for insert, delete, and search in order to surpass BST’s functions. One way of doing this is to create a more effective method of resolving collisions as well as create a more effective hash function.