|  |  |  |  |
| --- | --- | --- | --- |
|  | Vector | Hash Table | Binary Tree |
| Loading Data | O(1) | O(1) – O(N) | O(log N) |
| Search | O(N) | O(1) – O(N) | O(log N) – O(N) |
| Sort/Print | O(N log N) | O(N) | O(N) |

All of the data structures have advantages and disadvantages. A hash table could always operate at O(1) if it were large enough to prevent any collisions. This would require a near infinite table which is not possible so the hash table would be between O(1) – O(N). The binary tree will be near O(log N) pretty consistently. This would change dependent on how the data is loaded in and whether it is sorted. This would change the tree and cause it to become unbalanced, thus slowing it down. The vector is fairly quick and efficient, but searching items becomes more difficult depending on the size of the list.

I would recommend the hash table since the data is going to be searched more often than anything else. However, the table size would become an issue in terms of collisions so it would be just a little bit slower to accommodate for that.