**Run Time Evaluation**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Vector | Hash Table | Binary Tree |
| Load Data | O(1) | O(1)-O(N)  \*Depends on collisions | O(log N) |
| Search | O(n) | O(1) - O(N)  \*Depends on collisions | O(log N) - O(N)  \*depends on if data is already sorted when added to tree |
| Sort and Print | O(N log N)  \*using quick sort | O(N)  \* Table would have to be created in order or find all the keys to sort. | O(log N)  \*Should sort on addition of nodes  \*In order traversal |

**Advantages And Disadvantages**

The advantages for the vector class would be that it is easy to implement and since there are not a large number of classes, the program would not be much slower than the other data structures. It would be a disadvantage if there were a lot more classes.

The advantage for the Hash Table class is that it is very fast for searching large amounts of data but since there are not a lot of classes in this program, I don’t believe this is too much of an advantage. A disadvantage is that I don’t believe it can be sorted outright so we would have to get the keys of all the items and sort those.

The advantages of the Binary Tree are that it is very fast and the data is sorted as it is added to the tree. The disadvantage is that if the data is already sorted when added then the tree will be very large instead of uniform.

**Recommendation**

For this project I would recommend a Binary Search Tree data structure since it is fast to sort through and we need to be able to print out the data in alphanumeric order. Without the alphanumeric order requirement, I would have chosen a Hash Table.