Dear Reader,

Attached below are the results of an experiment to test three different data structures: Binary Search Trees (BSTs), AVL trees, and Splay trees. The premise of the experiment was to construct a dataset of 1000+ objects, store them in these 3 trees, and see how long it takes to find items in various orders. Being that my dataset was entered *in order*, the differences in these trees has been highlighted even more.

When the data was put into the **BST**, the tree had to have as many levels as there were objects (1048). This meant that to find objects, you would need to traverse an average of 524 levels deep. Another thing to note is that BSTs are not altered by the find method, so the tree remains the same speed no matter what order you look for nodes in. This made the BST underperform substantially when compared to the other two.

When the **AVL** tree is created, unlike the BST, the levels are reorganized so that no two subtrees of any node differ by more than one level of depth. This means that the AVL was much more shallow (the max depth when searching for a node never went above 12 or so). This makes the AVL much faster than the BST. The find method still does not alter the tree, so the order of nodes to search for didn't change the depth much.

The **Splay** tree, unlike the other two, is altered every time the find method is called. The benefits of this are pretty clear. In the first integer file, since the id's are being searched in order, the node that it is searching for is always in the second level, because it's parent node was just splayed to the root when it was found.

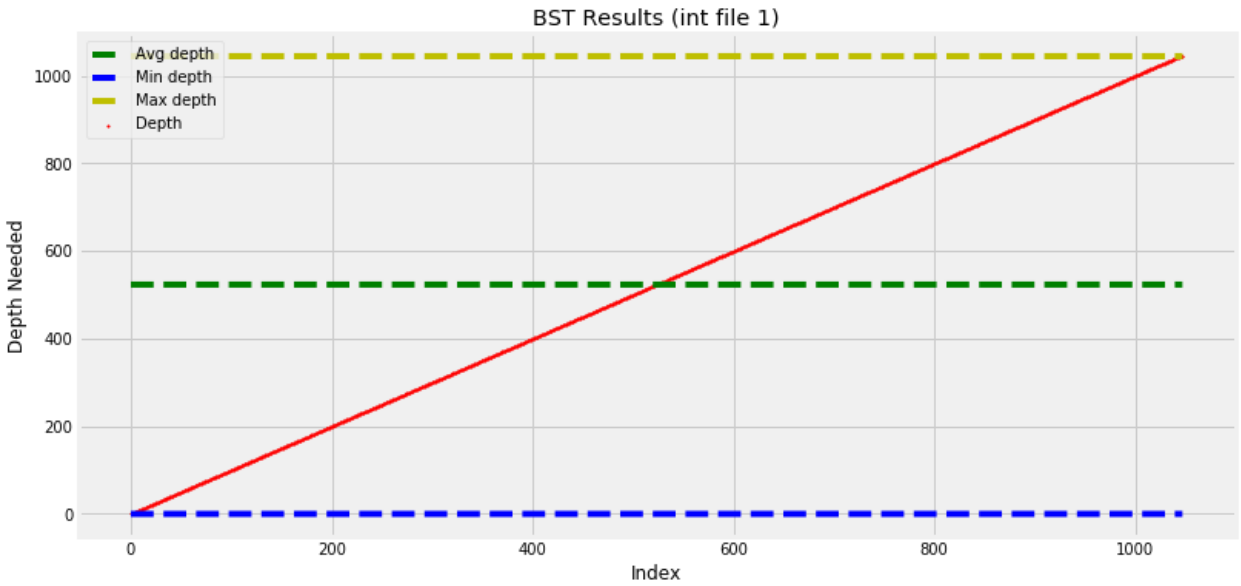
Despite the Splay tree appearing the most efficient from an average depth point of view, the last page of this document illustrates why that might not tell the whole story (due to the added time of having to splay the tree every time the find method is called).

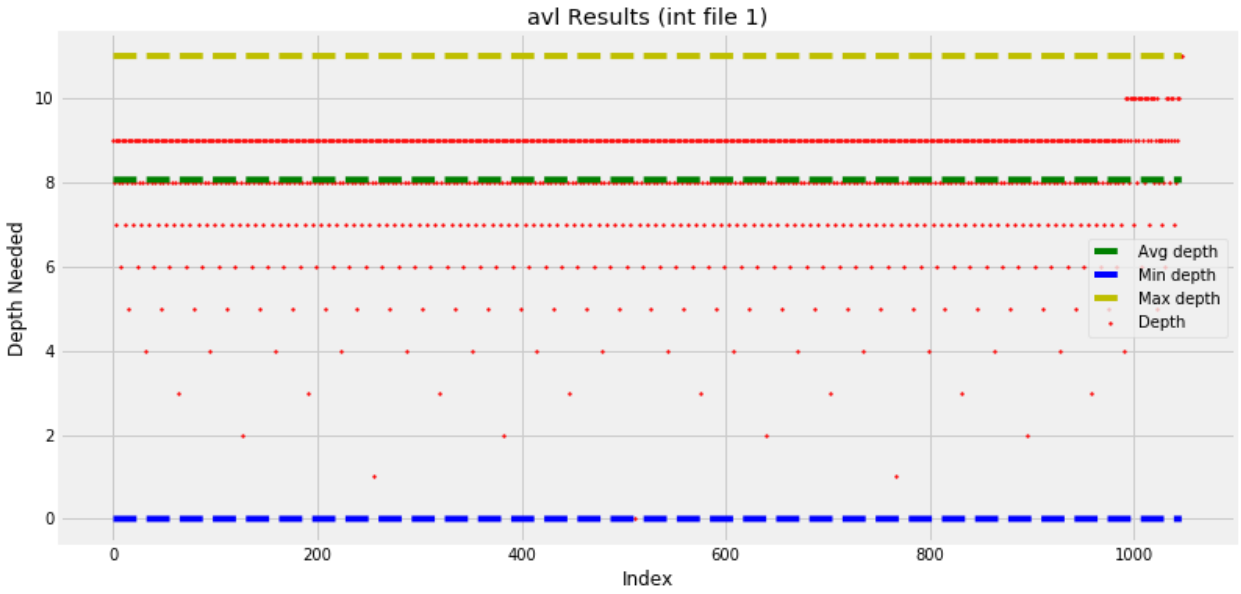
I hope you enjoy the data analysis.

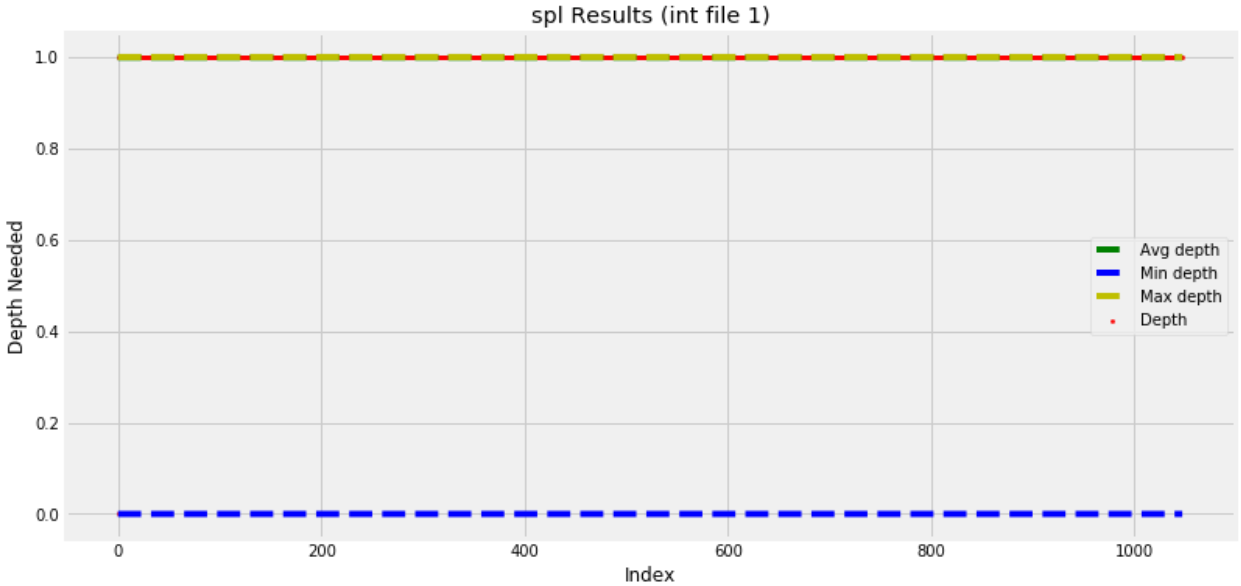
Sincerely,

Tripp

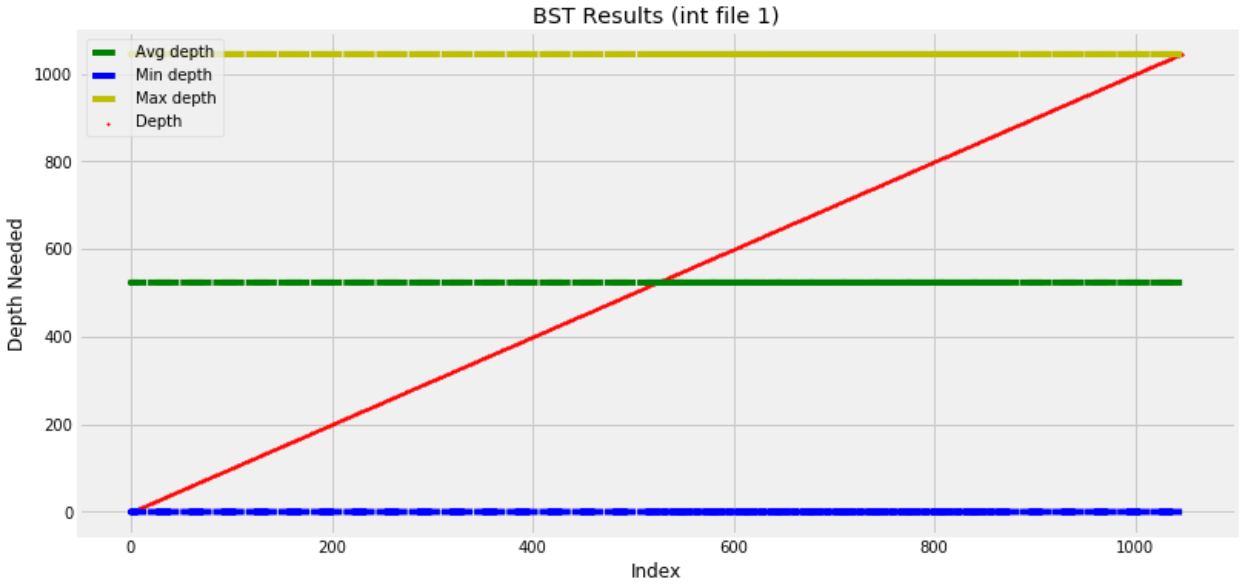
Int File 1 (searching in order)

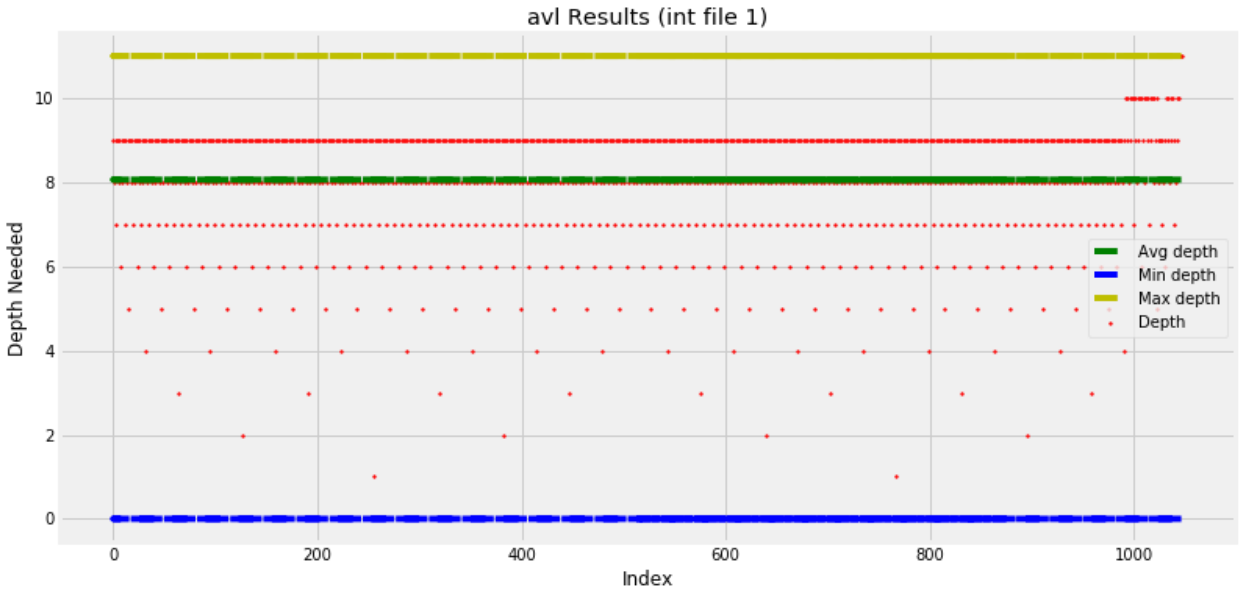


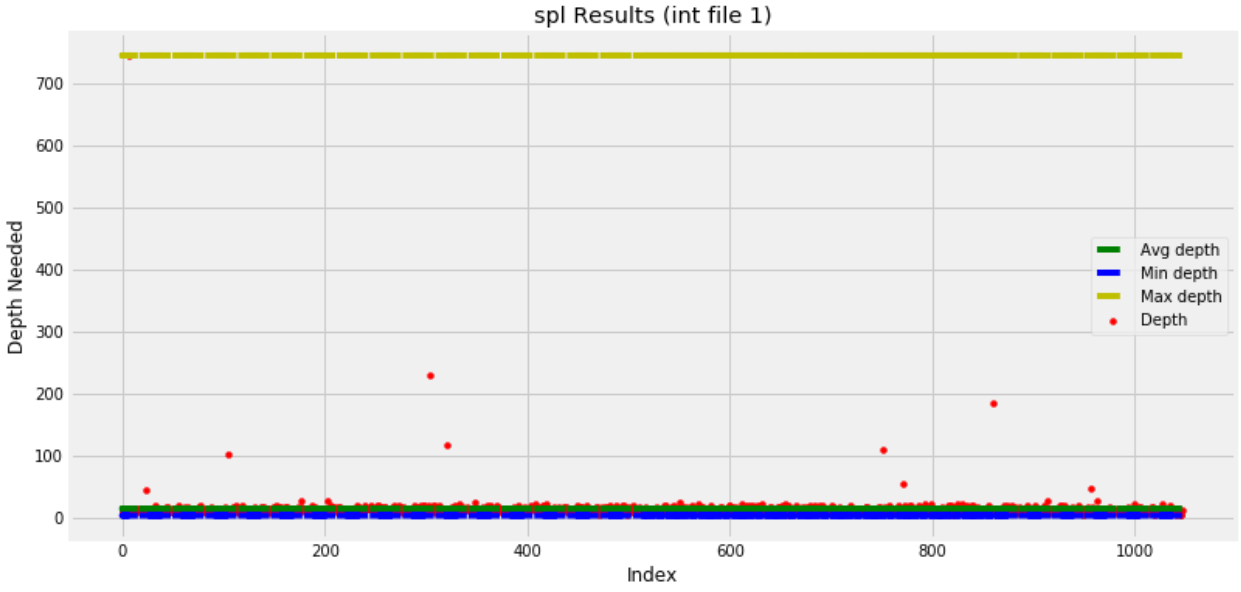




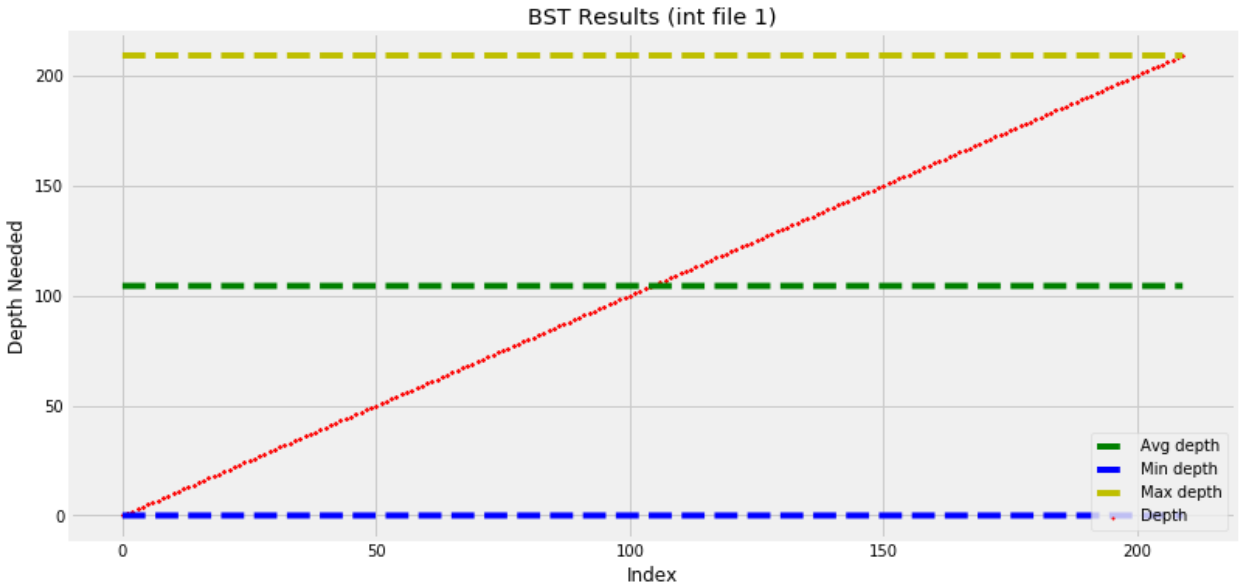
Int File 2 (searching in random order, no repeats)

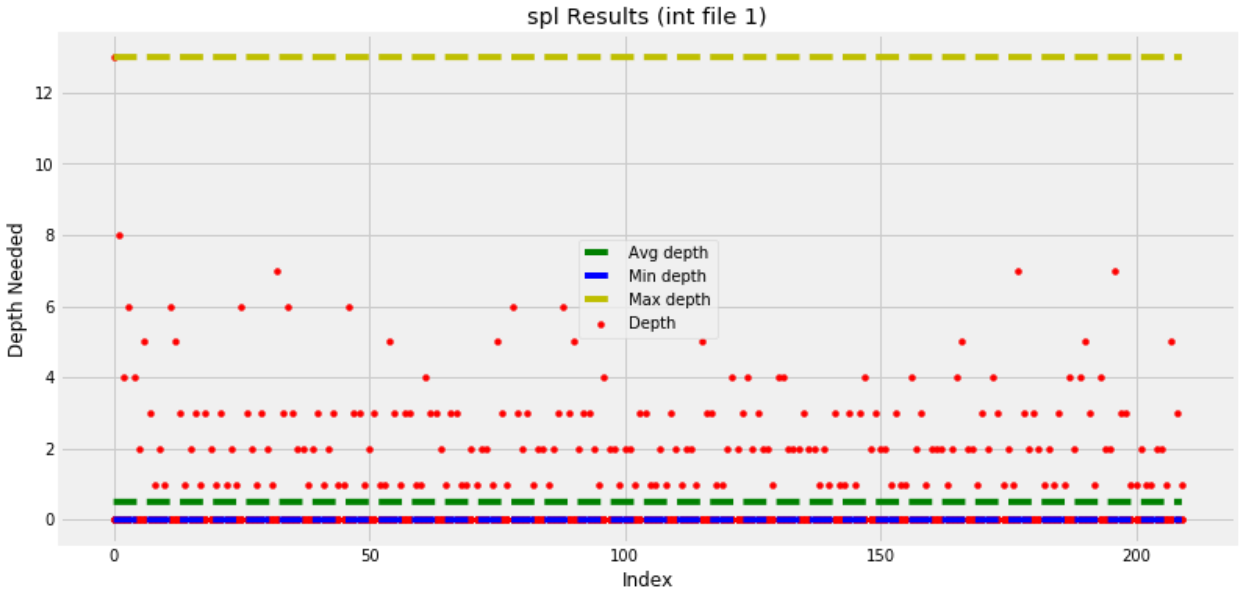






Int File 3 (searching each of the first 200 objects 5 times in a row)





Additional Metrics

In addition to the above analysis, I used timers to record how long it took each tree to find all of the objects (including the time it takes to record the data, however that remains constant). I recorded the results in an additional metrics file (attached). The results were as follows:

BST: **59115** microseconds
AVL: **25648** microseconds
Splay: **22817** microseconds

I have shown a bar graph of these times below, along with a graph of the average depths (from all three int files). As you can see, although the splay tree doesn't have to go quite as deep (on average) as the AVL tree, it actually ends up taking a bit longer anyway because it has to alter the tree every time the find method is called.