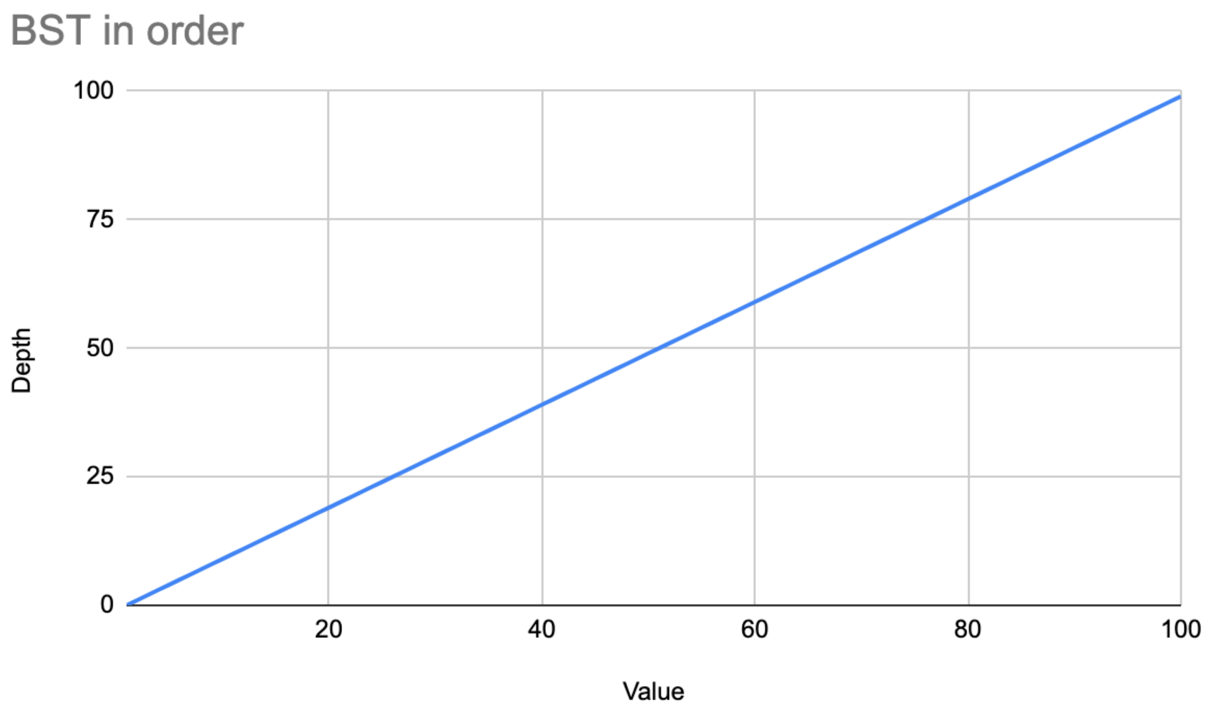
Report Project 3
Anna Dysinger
3/8/2023

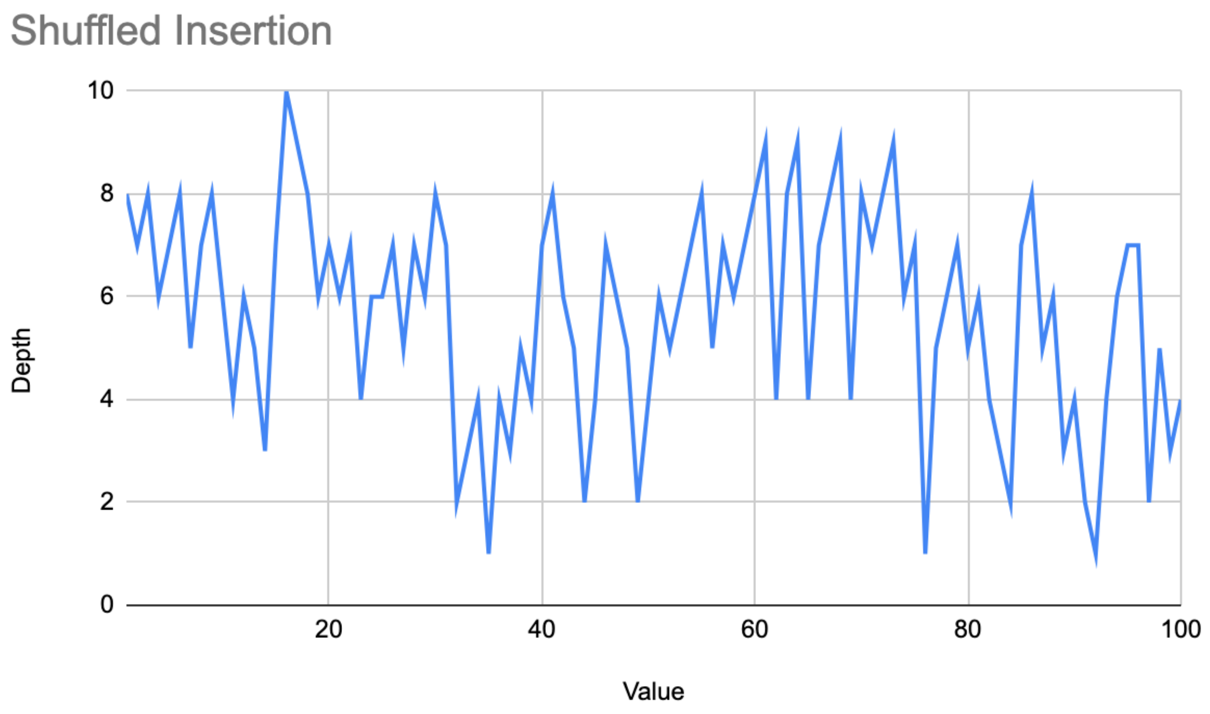
The data I used for this project included information about the top 10,000 streamed songs on Spotify. I narrowed the data down to the top 5000 streamed songs. I found the data at https://www.kaggle.com/datasets/rakkesharv/spotify-top-10000-streamed-songs?resource=download and I thought it would be good to use for this project because it has several data types. There are 9 fields in this Dataset. By default the entries are ordered by position, artist name, song name, days, top ten, peak position, peak position(# of times), peak streams, and total streams. Fields that have a unique record for each row include the Position and the Total Streams. The attributes include:

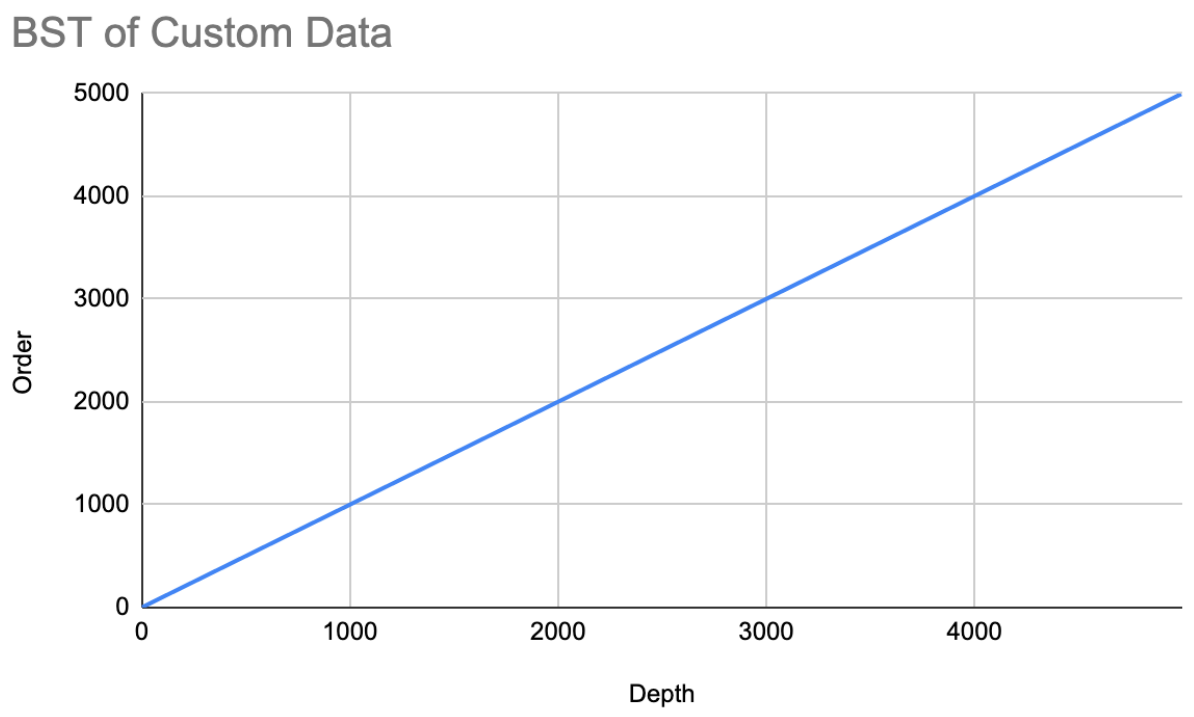
- Position (int)- The Spotify position
- Artist Name (string) - Name of the Artist or Band.
- Song Name (string) - Name of song.
- Days (int) - Number of days on Spotify.
- Top Ten (double) - Number of times the song hit the top ten.
- Peak Position (int) - Peak position attained.
- Peak Position (xTimes) (string)- Number of times peak position has been attained.
- Peak Streams (int) - total streams while in peak position.
- Total Streams (int) - Total number of streams.

Search for each number from 1 through 100, and record the depth of each find operation to a file. What do you notice? What is the value of the root node? What is the depth of the tree?
When shuffled the depths were much more scattered but when they were inserted in order the depths were constantly increasing. The root node is equal to 1 and the depth of the tree is 100. The time complexity for searching, deleting, and finding in BST is O(h), where h is the height of the tree.
Now instantiate a BST to hold objects of your custom data type. Read your objects into a vector (as you have done in earlier projects) and then insert them into the BST. Make sure you have at least 1000 objects in your BST. Now search for each of your objects and record its depth in a file. What do you notice? What is the value of the root node? What is the depth of your tree?
The custom data typed also depths that were constantly increasing.

Prepare three separate plots, one for each of these tests: BST of ints with ordered insertion, BST of ints with shuffled insertion, and BST of your custom data type.





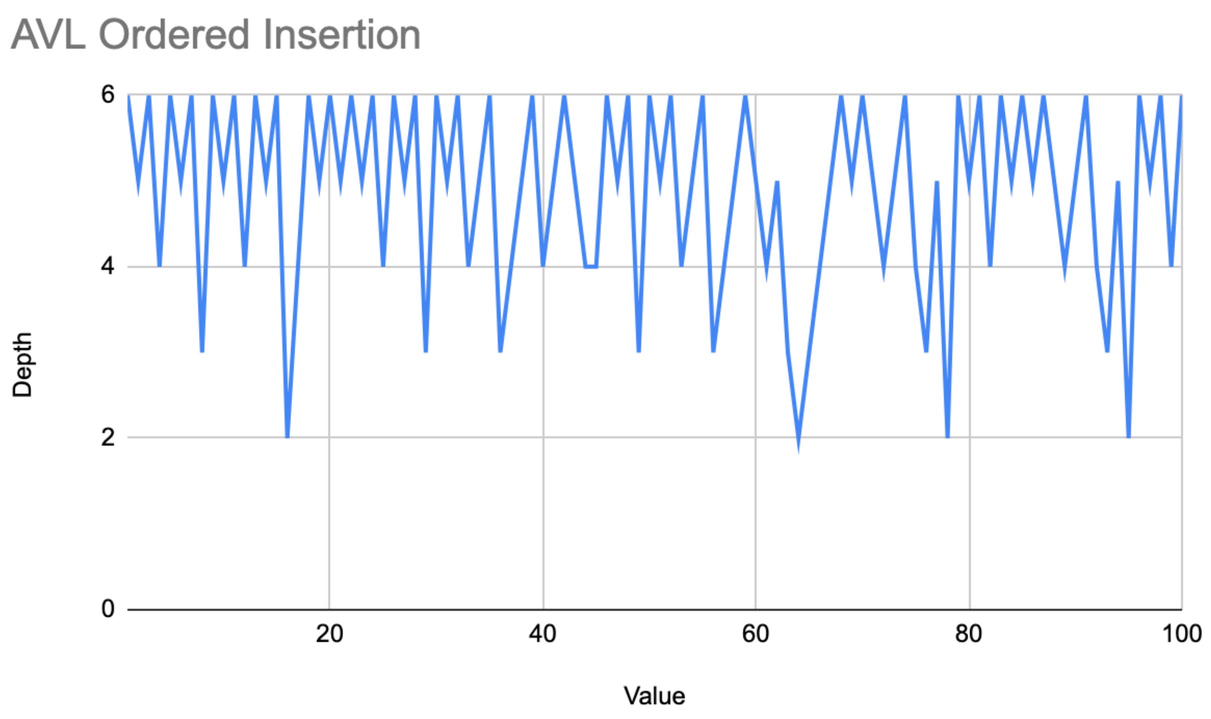


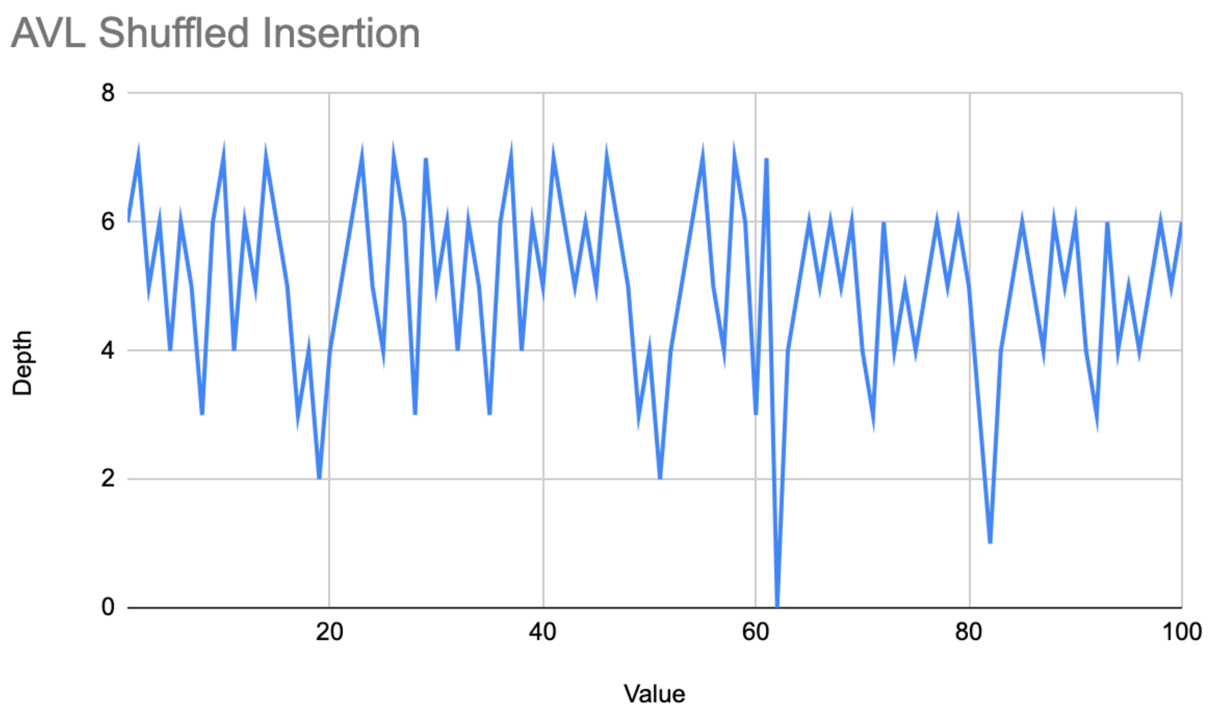
Perform the same experiments as you did for binary search trees, except now, all your trees are AVL trees. Record and analyze your results as above. Generate plots for your results as above.

[redacted]

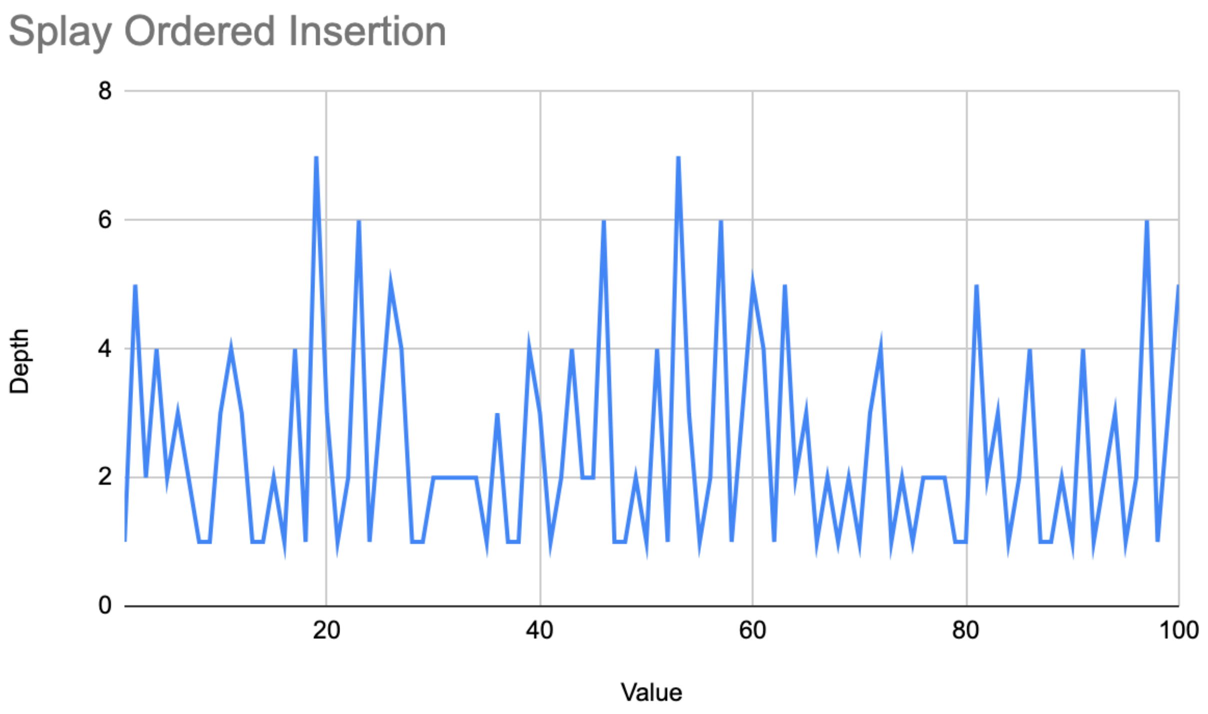
How do these depths compare with your results for binary search trees? Why?

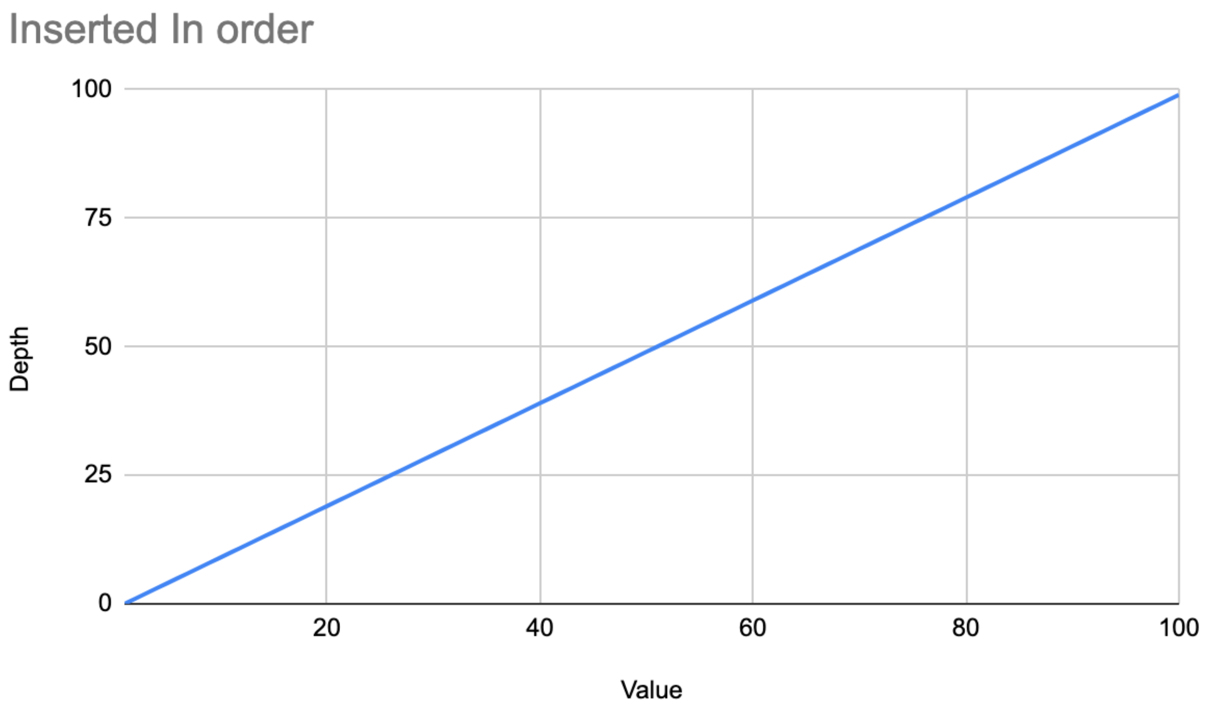
[redacted]





Shuffled insertion and ordered insertion have similar graphs, with depths wavering between 0 and 8. Ordered insertion had an average depth of about 6. AVL tree is a binary search tree with an additional property that the difference between the height of the left sub-tree and the right sub-tree of any node can't be more than 1. When traversing the worst case time complexity is *O(log*$_2$*n).*

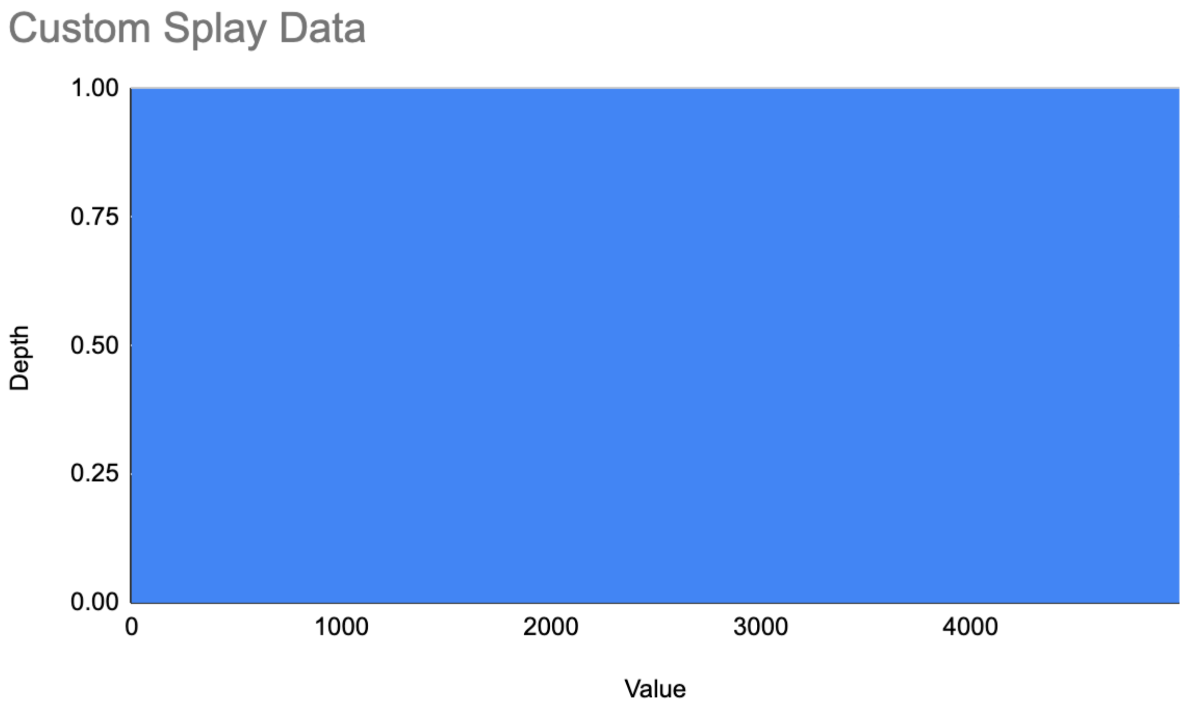




The splay tree acted similarly to BST, where the ordered insertion was constantly increasing and shuffled insertion had more scattered depths.

Now perform an experiment where you populate a splay tree with integers from 1 to 100, and then search for integers in random order. But in this case, search for each integer five separate times in succession, and record the depth in each case. What do you notice? Why do these depths make sense? (You do not need to generate a plot from these data.)

Here the depth was constant with all of the values. The worst case complexity of the splay tree's operations is $O(n)$. Since all operations also splay the tree on the node, the tree ends up roughly balancing itself, this results in a $O(\log n)$ worst case time complexity for all operations.


Custom Splay Data
Depth
Value