

|  |
| --- |
| **Assessment unit 2**  *AVL trees and report*  **Harry Fisher**  CMP204: Data structures and algorithms  2021/22 |

Contents

[1 Procedure 1](#_Toc120030729)

[1.1 inserts and complexity 1](#_Toc120030730)

[1.2 sorting algorithm 1](#_Toc120030731)

[1.3 max operation 1](#_Toc120030732)

[1.4 Adelson-Velsky-Landis (AVL) vs Binary-Search-Tree (BST) 1](#_Toc120030733)

[1.5 complexity of rotation 1](#_Toc120030734)

[Appendices 2](#_Toc120030735)

[Appendix A 2](#_Toc120030736)

[Appendix B 2](#_Toc120030737)

[Appendix C 2](#_Toc120030738)

# Procedure

## inserts and complexity

In order to test my input size efficiency, I took timing of my insert function inserting 1000, 10000, 100000. The reason I chose to input in x10 increments from 1000 is because it show the a nice progression of time of O(log n) by starting the time before the for loop to insert and once after then took the time delta of the difference, to see the code please look at [appendix a](#_Appendix_A).

the times I got where:

Text

Description automatically generated

## sorting algorithm

one way to print get the data in ascending order you could you an inOrder search, this runs in O(n) time because it needs to recurse through every element in the tree.

Text

Description automatically generated

## max operation

since ABL trees and balanced in order to find the largest node all we would need to do is recurse down the right side of the tree till we got to the last node which would just be number of nodes on the right side so O(n) where n stands for nodes, or I suppose O(h) for height of the right side

## Adelson-Velsky-Landis (AVL) vs Binary-Search-Tree (BST)

The main difference between a AVL and BST is, in a BST there is no balance factor in a AVL every number has a balance factor either 1, 0 or -1. Because of this Searching is inefficient in BST when there are large number of nodes available. So, if you have a very large data set you would want to use a AVL. But due to the fact AVL trees must be balance and has to make the rotations on insertion and deletion it can take a bit longer than a BST so if you have a very small data set that isn’t going to be around to long, then use a BST

## complexity of rotation

because the height of the tree affects the complexity of all operations, including insertion, deletion, and search and since the AVL tree is self-balancing , in the worst situation, its height is O(log n) the average case is also height of the AVL tree is O(log n) the tree rotations must be balanced, this only takes O(1) time therefore, overall, the complexity is still O(1) but in the worst scenario O(log n)

# Appendices

## Appendix A

Text

Description automatically generated

Text

Description automatically generated