CS 008
Lecture notes
4/11/24

1 Outline

- Binary tree (recap)
- Data structures (Sequences vs Sets)
- Balanced (AVL) Trees (Subset of Binary Trees) (Skew & Rotate)

2 Binary Tree

Traversal Order (In-order) goes from Left-Subtree → Parent → Right-Subtree.

Operations for Binary Trees:

- first_in_subtree(node): find first node in the subtree defined by node
- last_in_subtree(node)
- successor(node): find the next node in traversal order after node
- predecessor(node)
- insert_after(existing_node, new_node): insert new_node in traversal order
- insert_before(existing_node, new_node)
- delete(node): remove node from the tree while maintaining traversal order

Details on implementation would've been added but they've already been mentioned multiple times before in lectures and can be googled at this point if it's not intuitive.

3 Data structure theory

Insert table

Notice how with some functions of certain implementations, we see a 1(a) time complexity. The *a* stands for the amortized (aka: average) time because for example, if we had a dynamic array, there is a chance we need to extend the dynamic array.

3.1 Set Binary Tree (BST)

One property of sets is the ability to also have keys. A binary search tree is considered a 'keyed' set where it is sorted by the key / value to determine it's position in the tree. Using a binary search tree, you can implement an entire set interface by just adding a key.

3.2 Sequence Binary Tree

Similarly to how a BST can be used to implement a set, the traversal order of the binary tree is also the sequence order. In summary, for BST Trees, we can implement a sequence or a set with the BST Tree because of the properties the BST Tree has that can be used for these purposes. Using specific properties for a BST Tree, we can implement certain abstract data types.

4 AVL Trees

AVL Trees are a specific type of binary trees and have a specific property where it "self-balances" itself. Sometimes, trees will have an issue where