Section 9: Trees

Tree

- Every item in the tree is a node
- The node at the top of the tree is the root
- An edge connects each node
- Every non-root node has one and only one parent
- A leaf node has no children
- A singleton tree has only one node – the root

Binary Tree

- Every node has 0, 1 or 2 children
- Children are referred to as left and right child
- In practice, we use binary search trees
- A binary tree is **complete** if every level, except the last level has two children, and on the last level all the nodes are as left as possible

Binary Search Tree

- Can perform insertions, deletions, and retrievals in **O(log(n))** time
- The left child always has a smaller value than its parent
- The right child always has a larger value than its parent
- This means everything to the left of the root is less than the value of the root, and everything to the right of the root is greater than the value of the root
- Because of that, we can do a binary search

Traversal

- Level – visit nodes on each level
- Pre-order – visit the root of every subtree first
- Post-order – visit the root of every subtree last
- In-order – visit left child, then root, then right child

Delete node with two children

- Need to figure out what the replacement node will be
- Want minimal disruptions to the existing tree structure
- Can't take the replacement node from the deleted node's left subtree or right subtree
- If taking it from the left subtree, we have to take the largest value in the left subtree
- If taking it from the right subtree, we have to take the smallest value in the right subtree
- Choose one and stick to it

TreeMap Class

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/TreeMap.html

TreeSet Class

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/TreeSet.html