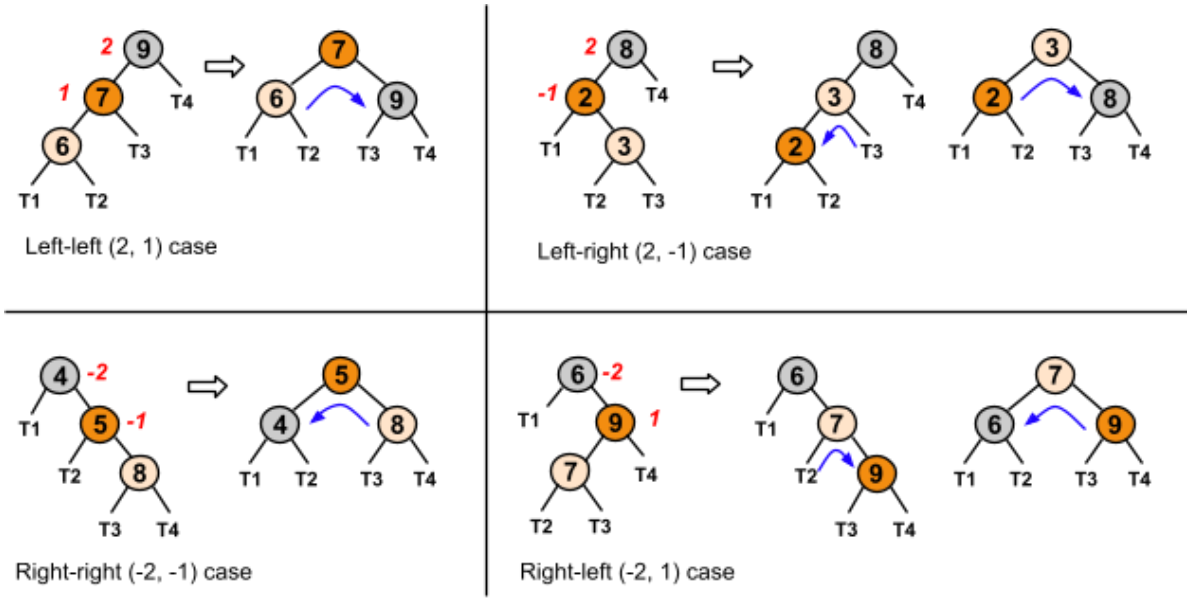
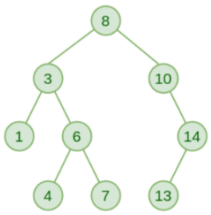


- Recursion
- Binary Trees - Everything on left is smaller than parent, everything on right is bigger
- Binary Search Trees
    - Types of traversals
    - Insertion - traversing through and place accordingly
    - Deletion: find node
        - If # of children:
0 - Delete node
1 - Replace node with child
2- Replace node with largest node in left subtree or replace with small in right subtree
            - Indicate if swapping with predecessor (left BIG) or successor (right small)
- Heaps
    - Removal
        - Remove root and replace with rightmost node on bottom layer + make swaps
            - min heap: swap with least child
            - max heap: swap with greatest child
    - Insertion
        - Insert into leftmost free space on the bottom layer + do swapping for heap property
- Sorting Algorithms (Selection, Bubble, Insertion, Mergesort and Heapsort, excluding ShellSort)
    - Selection - finding minimum elements one by one and swapping : O(n^2) with O(n) exchanges
    - Bubble - compare elements next to each other and bubble up max : exchange/complexity is O(n^2)
    - Insertion - compare elements next to each other : O(n^2)
    - Mergesort - splitting and combining (nlogn) : logn is time to get to greatest depth when splitting, n because merge and comparison happens for each element
    - Heapsort - make max heaps, remove max and place at the end of array (nlogn, insert is logn)
- Sets and Maps
    - Sets
        - Not indexed - no Set.get method (this means iterating gives you arbitrary order)
        - Must contain unique elements - Set.add returns false if you attempt to insert a duplicate item)
    - Maps - keys need to be unique
- Hashing
    - Load factor = number of full / total spots in table
    - Number of comparisons = C = ½ (1 + 1/(1 - L))
    - Number of chaining comparisons = 1 + L/2

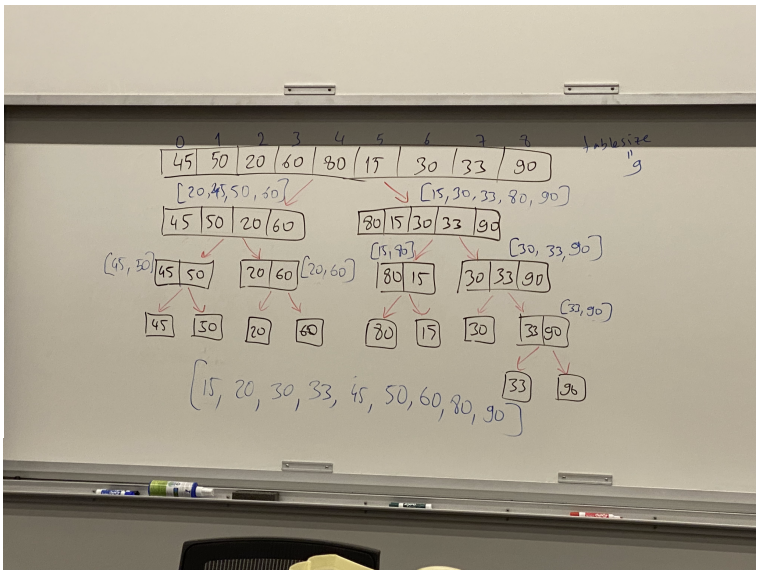
    - ASCII- UPPERCASE A starts at 65
    - UPPERCASE Z ends at 90
    - lowercase a starts at 97
    - lowercase z ends at 122
- AVL Trees - used because runtime (unbalanced can have a max n, balanced gives max logn)
    - Diagram + understand
    - Height of node = height of right - height of left
    - Removal and insertion are the same as BST - perform balancing operations to balance :)

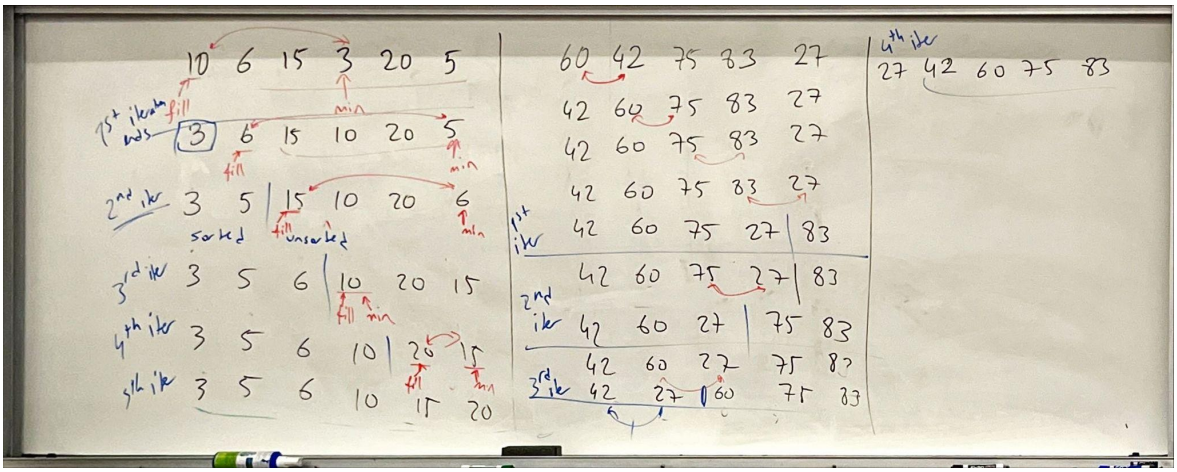
Traversals

Pre-order: Node - Left Subtree - Right Subtree
In-order: Left Subtree - Node - Right Subtree
Post-order: Left Subtree - Right Subtree - Node



Pre-order: 8 3 1 6 4 7 10 14 13
In-order: 1 3 4 6 7 8 10 13 14
Post-order: 1 4 7 6 3 13 14 10 8





Right is bubble, left is selection