5. Binary Search Trees, BST Sort

Link: https://www.youtube.com/watch?v=9Jry5-82I68

Runway Reservation System

Airport with a single runway:

- Certain data structures are no good for accomplishing O(logn) time for reserving landing
    - Unsorted arrays require linear time to search, match, and insert landing time
        - No binary search since it's unsorted
    - Sorted array
        - Log n to search
        - Constant for comparison (comparing R[1] and R[i-1])
        - Actual insertion requires shifting O(n) time
    - Sorted list
        - You can't do binary search on a list
    - Heaps: min/max
        - Has a weak invariant
        - Will take O(n) times
        - In a min/max heap, you have to check both sides
- Reservations for future landings
    - Reserve, request specifies landing time t
    - Add t to the set R if no other landings are scheduled within k minutes
- Remove from set R after plane lands

Binary Search Trees

- Unlike a heap, it requires pointers to parent, left child, and right child
- Invariant: for all nodes x, if y is in the left subtree of x... key(y) <= key(x), y is the right….. key(y) >= key(x)
- To find the lowest and highest numbers in BST, it will require O(heightoftree)

New Requirement (Rank)

- Rank(t): how many planes are scheduled to land at times <= t?
- We will have to augment the BST structure
    - We can add a size number to each node stating how many children nodes in contains
- What lands before t?
    - Walk down tree to find desired time
    - Add in the nodes that are smaller
    - Add in the subtree size to the left
-
- If we placed the array 43,46,48,50 into a BST, it will appear as a list. Searching will take O(n) equivalent to a linked list which is no good. We need it to be balanced so searching will equal to O(logn)