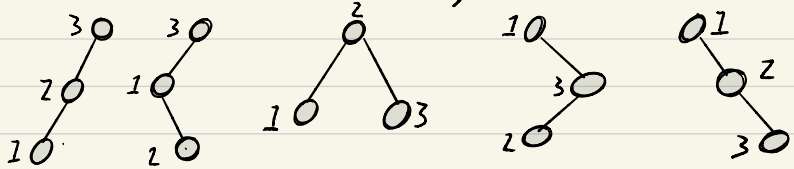
A) Binary Search Trees

- 5 distinct binary search trees on 3 nodes



- Rooted Binary tree
  - Either empty or consists of a node called the root, together with 2 rooted binary trees called the left and right subtrees
  - The order among subtrees matter so we can distinguish b/t the left & right nodes

- Binary Search Tree
  - labels each node in a binary tree with a single key such that for any node labeled x, all nodes in the left subtree of x have keys $< x$ while all nodes in the right subtree of of x have keys $> x$

B) Implementing Binary search Trees
- Binary tree nodes have left & right pointer fields, an optional parent pointer, & a data field

- Basic Operations
  i) Searching
    - Start at the the root
    - Unless it contains the query key x, proceed left or right depending on whether x occurs before or after the root key
    - O(h) where h = height of tree

  ii) Finding minimum & maximum
    - Minimum
      - By definition the smallest key must reside in the left subtree of the root since all keys in the left subtree have values less than that of the root
      - Therefore, the min element must be the leftmost descendent of the root
    - Maximum
      - Maximum element must be the right most descendent of the root by the definition of minimum.

iii) Traversal in a Tree
- Each item is processed once
- Runs in $O(n)$ where n = # of nodes in the tree

iv) Insertion
- Only 1 place to insert an item x into a binary search tree
- Must replace the NULL pointer found in T after an unsuccessful query for the key K
- The implementation uses recursion to combine the search and node insertion stages of key insertion
- After the search is implemented, allocating the node & linking it to the tree is a constant-time operation

C) Balanced Search Tree
- Red-black trees and splay trees