Divide and Conquer Algorithms: Binary Tree Traversals

- Read section 5.3 (pages 182-185)
- Why is divide-and-conquer a natural approach to solving problems involving binary trees?
- Study the algorithm to compute the height of a binary tree.
  - What is the basic operation?
  - Why is -1 the correct value to return for an empty tree?
  - What nodes do we call *external*? What nodes do we call *internal*?
    - What is the relationship between the number of external nodes and the number of internal nodes?
  - What is the time efficiency class of the height algorithm? How is that related to the number of external and internal nodes?
  - Is the height algorithm's running time affected by how balanced the tree is?
  - Can we perform a similar algorithm for a rooted ordered tree (i.e. allowing any number of children)?
    - Is the relation between external and internal nodes the same for such a tree?
    - Can you come up with at least an upper bound on how many external nodes a tree with n internal nodes can have?
- What are the three traversal orders for binary trees?
  - Write pseudocode for each traversal order.
  - Work out by hand the three orders in Figure 5.6
- Practice problems: 5.3.1, 5.5.5, 5.5.7