* 1. 
     1. Preorder: 50, 20, 10, 15, 40, 30, 25, 31, 60, 70, 65, 80, 79
     2. Inorder: 10, 15, 20, 25, 30, 31, 40, 50, 60, 65, 70, 79, 80
     3. Postorder: 15, 10, 25, 31, 30, 40, 20, 65, 79, 80, 70, 60, 50
     4. 
     5. 
  2. struct Node
     1. {
        1. Node \*right;
        2. Node \*left;
        3. Node \*parent;
        4. int val;
     2. }
     3. If the tree is empty
        1. Allocate a new node with the input value
        2. Point root to node and set children/parent to null
     4. Start at root
     5. While insertion has not been completed
        1. If input is equal to current node’s value
           1. Return function (insertion finished)
        2. If input is less than current node’s value
           1. If there is a left child

Go left

* + - * 1. Otherwise

Allocate a new node with the input value

Point parent to the current pointer and the current pointer to the new node

Set children to null

* + - 1. If input is greater than current node’s value
         1. If there is a right child

Go right

* + - * 1. Otherwise

Allocate a new node with the input value

Point parent to the current pointer and the current pointer to the new node

Set children to null

* 1. 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 | 5 | 6 | 4 | 1 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 5 | 2 | 4 | 1 |

* 1. O(C + S)
  2. O(log C + S)
  3. O(log C + log S)
  4. O(log S)
  5. O(1)
  6. O(log C + S)
  7. O(S \* log S)
  8. O(C \* log S)