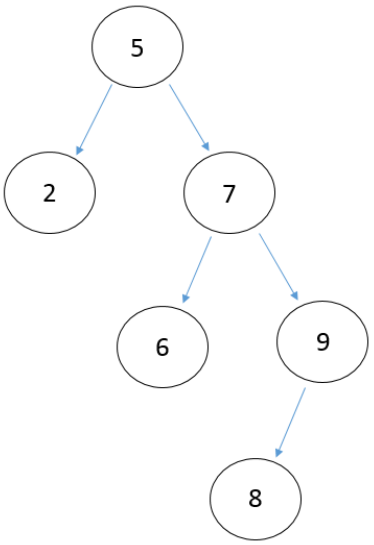
Exercise – Trees

Exercises:



1. Implement a Binary Search Tree. This is a method for storing integers that is very efficient when searching for a value or displaying them in numerical order. It has the following properties:
    a. Each node stores an integer value.
    b. Each node has at most two child nodes: usually named left and right.
    c. The value of the left child will always be less than the value of the parent node.
    d. The value of the right node will always be greater than the value of the parent node.
    e. The tree cannot contain duplicate values. If a duplicate is added, discard it.

    First, create a Node class containing:
    a. An integer value to be stored
    b. A pointer to a left node.
    c. A pointer to a right node.

    Create a Binary Tree class containing:
    a. A pointer to the root (top-most) node. This will start null but be filled by the first node added to the tree.
    b. Function: IsEmpty() – Return true if the tree is empty.
    c. Function: Insert(int value) – Take in a value and add it to the tree. Start at the root node and search through the tree until you find a valid place to insert the node. You don't need to move any nodes to do this, no overwriting values or rearranging anything. Just put it in the first empty spot you find but remember the rules: lower values must go to the left of the parent, higher values must go to the right, discard duplicates.
    d. Function: Find(int value) – Search through the nodes for the value, return true if you find it in the tree, false if you don't. Remember that you don't need to check every node, search left for lower values and right for higher ones.

Continued next page…

2. CHALLENGE: Add a remove function to delete a node of a specific value from the tree. There are three possibilities:
    a. The target node has no children: Simply delete it and make sure its parent's reference to it is set to null.
    b. The target node has one child: Delete the node, then make its parent's reference to it instead point to its child.
    c. The target node has two children: This is tricky, rearranging the nodes is too difficult so instead we do some tricky copying.
        i. Find the lowest valued node below the target's right child (from the target, go right once, then go left as far as you can).
        ii. Copy the value of this lowest node into the target node.
        iii. Call this remove function on that lowest node to delete it and go through this process again so we keep any children it may have intact.