Section 03: Solutions

Binary Search Trees

(a) Write a method `validate` to validate a BST. Although the basic algorithm can be converted to any data structure and work in any format, if it helps, you may write this method for the `IntTree` class:

```
public class IntTree {
    private IntTreeNode overallRoot;

    // constructors and other methods omitted for clarity

    private class IntTreeNode {
        public int data;
        public IntTreeNode left;
        public IntTreeNode right;

        // constructors omitted for clarity
    }
}
```

Solution:

```
public boolean validate() {
    return validate(overallRoot, Integer.MIN_VALUE, Integer.MAX_VALUE);
}

private boolean validate(IntTreeNode root, int min, int max) {
    if (root == null) {
        return true;
    } else if (root.data > max || root.data < min) {
        return false;
    } else {
        return validate(root.left, min, root.data - 1) &&
            validate (root.right, root.data + 1, max);
    }
}
```

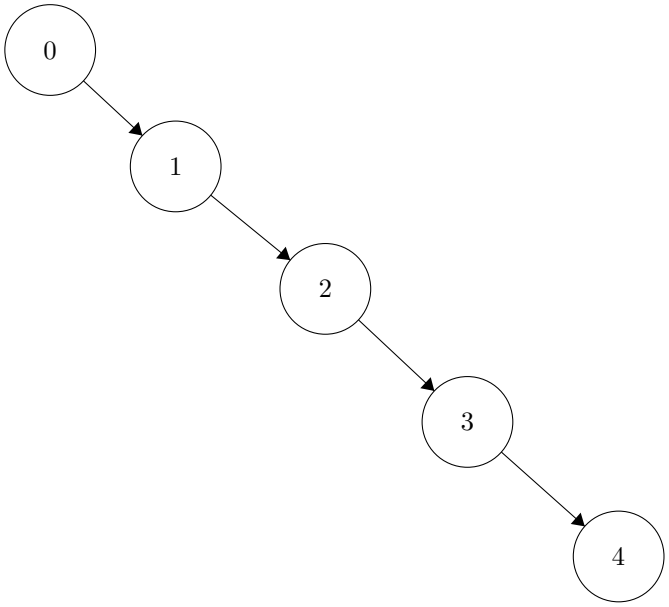
(b) Suppose we want to implement a method `findNode(V value)` that searches a binary search tree with n nodes for a given value.

(i) What is the worst case big-Θ runtime for `findNode`? Draw an example of a binary search tree with up to 4 nodes that would result in this worst-case runtime.

Solution:

The answer is $\Theta(n)$, because we could have a completely unbalanced tree and the value we are looking for could be at the very bottom.

Example `findNode(4)`:

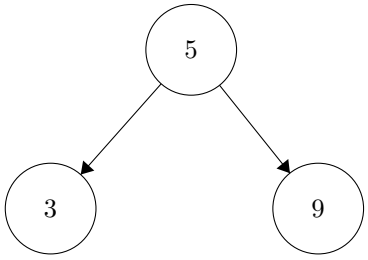


(ii) What is the best case big-Θ runtime for `findNode`? Draw an example of a binary search tree with up to 4 nodes that would result in this best-case runtime.

Solution:

The answer is $\Theta(1)$, because the node we're searching for could be at the root.

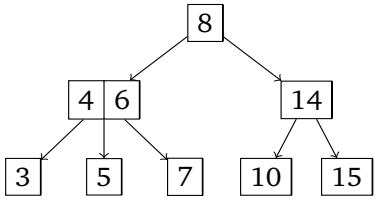
Example: `findNode(5)`:



B-Trees

1. Draw the B-Tree!

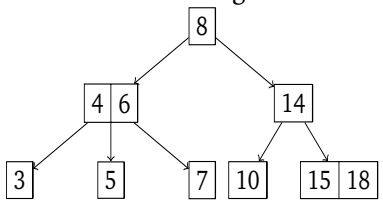
(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



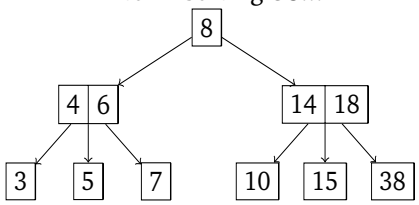
Solution:

The tree is redrawn after each insertion, but only the final tree matters in this case. Remember that doing something like this to show your work can help you earn partial credit in an exam setting!

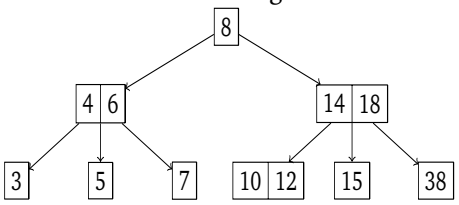
After inserting 18...



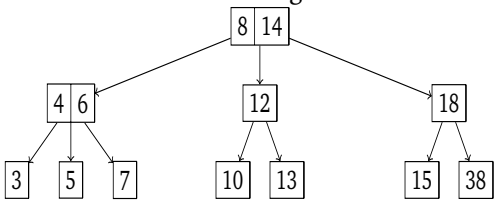
After inserting 38...



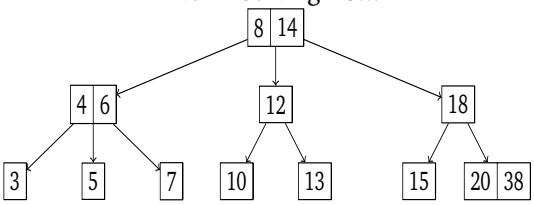
After inserting 12...



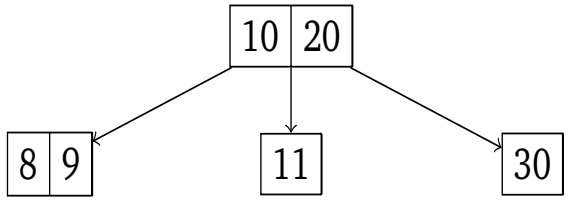
After inserting 13...



After inserting 20...

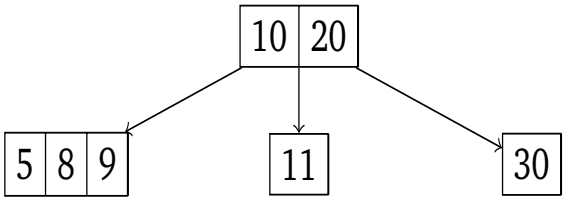


(b) Given the following initial 2-3-4 tree, draw the result of performing each operation.



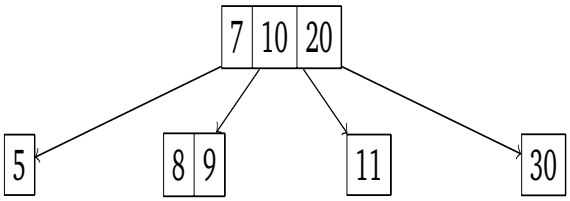
(i) Insert 5 into this tree.

Solution:



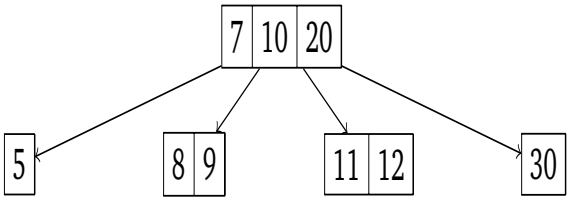
(ii) Insert 7 into the resulting tree.

Solution:



(iii) Insert 12 into the resulting tree.

Solution:



(c) Suppose the keys 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 are inserted sequentially into an initially empty 2-3-4 tree. Which insertions cause a split to take place?

Solution:

4, 6, 8, 10

2. Invariants of the B-Tree

(a) What properties must a B-tree have?

Solution:

(i) B-Tree order property

  (i) Every subtree between keys a and b contains all data x where $a \leq x \leq b$

(ii) B-Tree structure property:

  (i) All leaves must be the same depth from the root.

  (ii) A non-leaf node with K keys must have exactly K + 1 non-null children.