Section 03: BSTs and B-Trees

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
    }
}
```

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

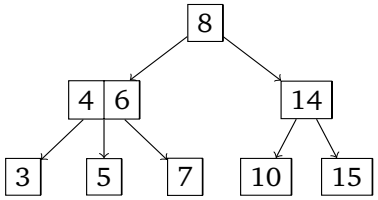
(i) What is the worst case big-Θ runtime for `findNode`? Draw an example of a binary search tree with up to 4 nodes that would result in this worst-case runtime.

(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.

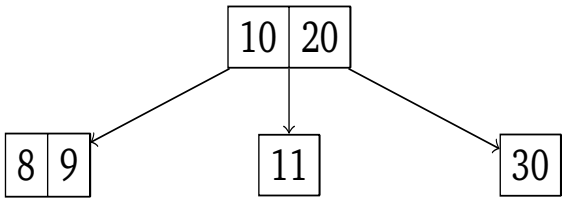
B-Trees

1. Draw the B-Tree!

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



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



(i) Insert 5 into this tree.

(ii) Insert 7 into the resulting tree.

(iii) Insert 12 into the resulting tree.

(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?

2. Invariants of the B-Tree

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