1 2-3 Tree Insertion

Given the following 2-3 tree, draw what the tree would look like after inserting 18.

Original tree: After inserting 18:

```
[8]
/  \
[6]  [14]
/ \  /  \  
[3] [7] [10] [15, 16]
```

2 BSTs and Balance

Given the following binary trees, determine if each is a BST, and whether it has minimum-BST-height (circle the correct answer). By minimum-BST-height, we mean that the height of the tree is the same as the height of the optimal binary search tree containing the given elements.

```
6 10 9
/  /  /  \
5 9 6 12 7 11
/  /  /  /  /  \
2 7 10 3 11 5 8 13
```

Valid: T F Valid: T F Valid: T F
Min-Height: T F Min-Height: T F Min-Height: T F

Suppose we know the height H and number of nodes N of a BST. Can we determine whether or not this BST is minimum-BST-height without having to check the values of each node? If so, how? If not, why not?

3 Binary Tree Creation

Implement a function that, given a sorted array of integers of length \( N = 2^k - 1 \), creates and returns a BST of minimum height. You can assume you have a method `slice` that takes in an integer array and two indices to slice between (inclusive of the first index, exclusive of the second):

```
slice([1, 2, 3], 0, 1) // Returns [1]
slice([1, 2, 3], 1, 3) // Returns [2, 3]
```

Use the following definition of a Binary Search Tree Node (`BSTNode`):
public class BSTNode {
    public BSTNode left, right;
    public int value;

    public BSTNode(int n) {
        value = n;
    }
}

public BSTNode makeBST(int[] nums) {

}  

Runtime recap: What is the runtime of makeBST()?

4 Common Ancestor

Challenge Problem: Implement a function that, given a valid BST and two integers, returns the BSTNode X that is the deepest common ancestor of the two integers. By deepest, we mean that its distance from the root is maximized. By common ancestor, we mean that n1 <= X.val and n2 >= X.val. You may assume that n1 < n2. If no such node exists, return null.

/** Returns the BSTNode that is the shortest common ancestor of n1 and n2. */
public BSTNode commonAncestor(BSTNode root, int n1, int n2) {

}

Runtime recap: What is the runtime of commonAncestor()?