App Academy

Trees

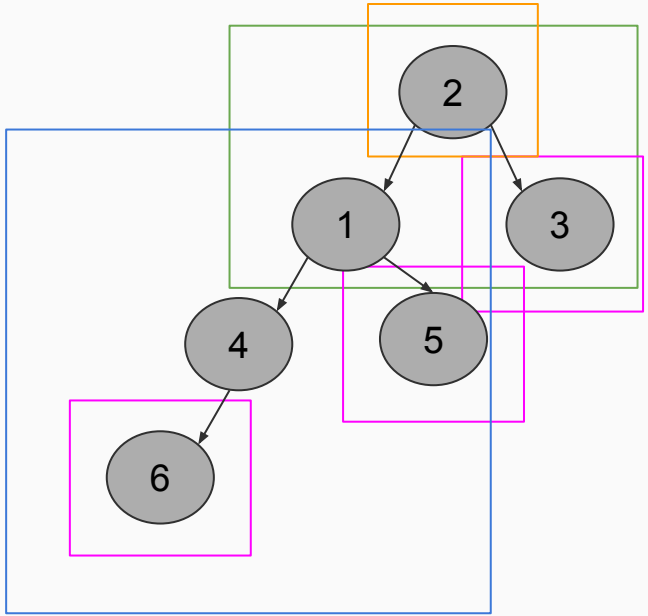
Binary Tree Structure and Definitions

- Similarly to linked lists, trees are made of nodes with pointers.
  - Instead of next or previous pointers, we have left and right pointers.
- The relationship between connected nodes is a "parent-child" relationship.
- A node without a child is considered a "leaf node".
  - Binary trees are always guaranteed to have leaf nodes.
  - Leaf nodes are very good places to look when it comes to bases cases in recursive functions!!
- A node without a parent is considered the "root" of the binary tree.
- If a node is a child or any other node beneath the children, it is considered a "descendant" of another node.
- The height of a node is how far it is from the leaf nodes.
- The depth of a node is how far it is from the root node.
- How height and depth is measured is different in different textbooks, so make sure you define how you measure it during interviews.

Requirements for a Binary Tree

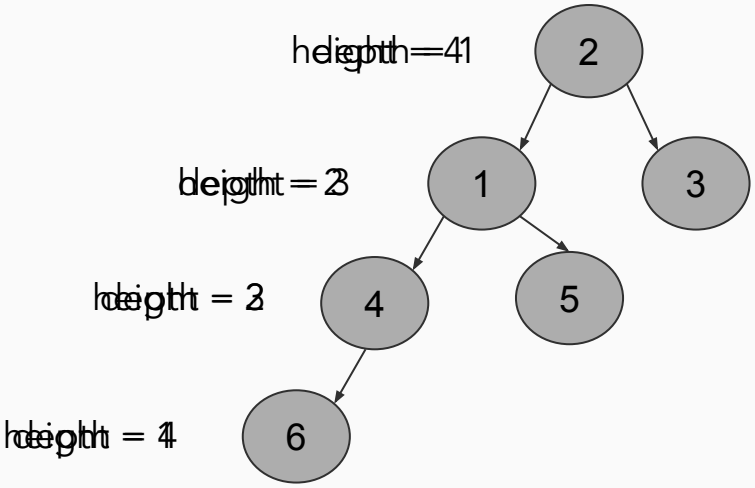
- Has at most one root node.
- Each node has at most two children.
- Each node has at most one parent.
- A binary tree cannot be cyclic.

Binary Tree Structure



- Node 2 is the parent of Node 1 and Node 3.
- Node 1 and Node 3 are the children of Node 2.
- Nodes 3, 5, and 6 are considered leaf nodes
- Node 2 is the root of this binary tree
- Node 4, 5, and 6 are all considered descendants of Node 1
- Node 1 is an ancestor of Nodes 4, 5, and 6.

Measuring a Binary Tree



- Some people prefer to measure height and depth starting from 0 instead of 1.
- Some also prefer to measure by edges instead of nodes.
- Make sure to always define how you're making these measurements to avoid confusion!

Tree node in code

```
class TreeNode(val) {
        this.val = val;
        this.left = null;
        this.right = null;
}
```

Binary Search Trees (BST)

- A few additional properties to regular binary trees
  - Every node on the left subtree must be less than the root value
  - Every node on the right subtree must be greater than the root value
  - BSTs generally do not contain duplicates
  - The above properties are recursive, which means they apply to every subtree and not just the root node
- Why would a BST be useful?
  - We can use binary search to search the tree!
- Demo: Search In a Binary Search Tree

Binary Search Tree (cont.)

```
const searchBST = function(root, val) {
    if (root === null) return null;
    if (val < root.val) {
        return searchBST(root.left, val);
    } else if (val > root.val) {
        return searchBST(root.right, val);
    } else {
        return root
    }
};
```

Questions?