Trees Quickstart Guide

In this mini-workbook, we'll get you up and running with trees. These are one of the most commonly used data structures in coding interviews so it is critical that you have a good handle on them.

In this guide, we will quickly highlight the key concepts that you need to know for your interview. If you're unfamiliar with any of these, I highly recommend taking some time to review them.

Then, I've provided you with a series of daily exercises to help you get up and running quickly. I recommend using these similarly to how you use your daily workbook. Do one set of exercises each day and be sure to log them in your tracker. This will help you ensure that you've reviewed all the most important points.

Key Tree Terminology

- **Binary Tree**
  The core data structure.
- **Binary Search Tree (BST)**
  A binary tree where all of the nodes are in sorted order.
- **N-ary Tree**
  A tree where each node has at most N children (N can be specified, ie binary tree, ternary tree, quad tree, etc).
- **Complete Binary Tree**
  A binary tree where all levels except the bottom level are completely filled.
- **Balanced Binary Tree**
  There are two conflicting definitions here. Either no two leaf nodes differ by more than 1 in height or no two subtrees differ by 1 in height.
- **Tree Traversal**
  Any algorithm used to visit the nodes of a tree

Key Tree Patterns

- **Basic Tree Operations**
  - Construct a tree
  - Insert a node
  - Remove a node
  - Find a value
- **Traversals**
  - Inorder
  - Preorder
  - Postorder
  - Level order
- **Searching**
  - DFS
  - BFS
- **Divide and Conquer**

Daily Exercises

Day 1

Implement a `Node` and `BinarySearchTree` class. You can assume that node values are just integers. Implement the following methods for the tree class:

- Constructor
- `insert(int n)`
  Inserts the value n into the tree
- `delete(int n)`
  Removes the value n from the tree
- `contains(int n)`
  Return true if the tree contains the given value
- `height()`
  Return the height of the tree
- `numberOfNodes()`
  Return the number of nodes in the tree

Day 2

Implement the following tree traversals. I would highly encourage you to attempt solving each both recursively and iteratively. Each has a natural approach that lends itself, but practicing both approaches will help you develop a much deeper understanding.

- Inorder Traversal (Leetcode)
- Preorder Traversal (Leetcode)
- Postorder Traversal (Leetcode)
- Level order Traversal (Leetcode)

Day 3

Today, we're going to start applying our patterns to some common interview questions. As you go through these questions, make sure that you follow your standard problem solving framework.

You will also want to look at what patterns that we've learned might apply to these different problems. If you're getting stuck, think about how you can break down the problem into the basic patterns.

1. Serialize Binary Tree
2. Max Binary Tree Depth (Leetcode)
3. Lowest Common Ancestor (Leetcode)

Day 4

Today, we're going to continue applying our patterns to some common interview questions. As you go through these questions, make sure that you follow your standard problem solving framework.

You will also want to look at what patterns that we've learned might apply to these different problems. If you're getting stuck, think about how you can break down the problem into the basic patterns.

1. Balanced Binary Tree (Leetcode)
2. Merge Binary Trees (Leetcode)
3. Invert Binary Tree (Leetcode)

Day 5

Today, we're going to continue applying our patterns to some common interview questions. As you go through these questions, make sure that you follow your standard problem solving framework.

You will also want to look at what patterns that we've learned might apply to these different problems. If you're getting stuck, think about how you can break down the problem into the basic patterns.

1. Diameter of a Binary Tree (Leetcode)
2. Tree to Linked List (Leetcode)
3. Maximum Path Sum (Leetcode)

Exercises Solutions

Coming soon!