Trees

In the world of data structures, trees stand out as a *non-linear structure*. They are similar to real-life trees; the only difference is they are inverted. That means the root of this data structure is at the top then the branches and then comes leaves. Trees are especially useful for representing *hierarchical data*. Moreover, we unknowingly use the tree data structure almost every day on our computer as all the files on our computer are organized in the tree data structure only. Now, if you are a Windows user, just open your command prompt, type in the command "tree" and see what happens.

Anatomy of Trees:

- Root Node: The starting point of the tree, which has no parent.
- Child Nodes: Nodes directly connected to another node as you move away from the root.
- Parent Nodes: Nodes directly connected to another node as you move towards the root.
- Leaf Nodes: Nodes with no children and representing the endpoints or edges of the tree.
- Binary Tree: A type of tree where each node has at most two children, called the left and right children.
- Subtree: Any node and its descendants can be considered a subtree of the main tree.

Tree Traversal Methods:

1. Level-order Traversal: Visits nodes level by level from top to bottom, starting from the root.
2. Depth-First Traversal:
    a. Pre-order Traversal: Visit the root, then the left subtree, followed by the right subtree. (Root → Left → Right)
    b. In-order Traversal: Visit the left subtree, then the root, followed by the right subtree. (Left → Root → Right). This will give elements in a sorted way in increasing order.
    c. Post-order Traversal: Visit the left subtree, then the right subtree, and finally the root. (Left → Right → Root)

This uses O(n) time complexity and space complexity.

Morris Traversal does it in O(1) space complexity and O(n) time complexity.

AVL Trees and Red-black trees are special self-balanced binary trees that use the time complexity of O(log n).