Binary Tree

By Mani Abedii

A **binary tree** is a tree-like data structure in which each node has at most two children (branching factor of 2). These children are referred to as left child & right child.

Binary trees are used to store data in a hierarchical manner, allowing for efficient searching & sorting. They are used in various applications, such as search trees, expression trees, and Huffman trees.

Don’t Confuse These!

- A **full binary tree** is a binary tree in which every node has either 0 or 2 children.
- A **complete binary tree** is a binary tree in which every level, except possibly the last level, is completely filled, and all nodes are as far left as possible.
- A **perfect binary tree** is a full & complete binary tree.
- A **balanced binary tree** is a binary tree in which the height of the left & right subtrees of every node differ by at most one.

Minimum & Maximum Number of Nodes of a Binary Tree:

- The **maximum** number of nodes for height h = $2^{h+1} - 1$
- The **minimum** number of nodes for height h = h + 1

Minimum & Maximum Height of a Binary Tree:

- The **minimum** height of a binary tree occurs when the tree is as balanced as possible, with nodes evenly distributed across levels → height = $[\log_2(n)]$
- The **maximum** height occurs when the tree is completely unbalanced, essentially forming a linked list. → height = n - 1

→ When the number of nodes is maximized for a given height h, the tree becomes complete, resulting in the minimum height.

→ When the number of nodes is minimized for a given height h, the tree becomes skewed, resulting in a linear structure.

Implementation:

Trees can be implemented using arrays or linked lists. Arrays are most efficient for trees that are complete or full & less likely to change size.

When implementing using an array, if a node is in the i-th index:

- Left child = (2 * i) + 1
- Right child = (2 * i) + 2
- Parent node = [(i - 1) / 2]

However, if the tree has many missing nodes, linked lists save memory by allocating space only for existing nodes. Also, when the tree's size changes frequently, linked lists handle these operations efficiently without resizing.

Although binary trees have several advantages –such as efficient searching and sorting and use in various applications– they also have some disadvantages, including their limited height and the potential to become unbalanced. As we will see in the future, many trees have been created to overcome these problems.