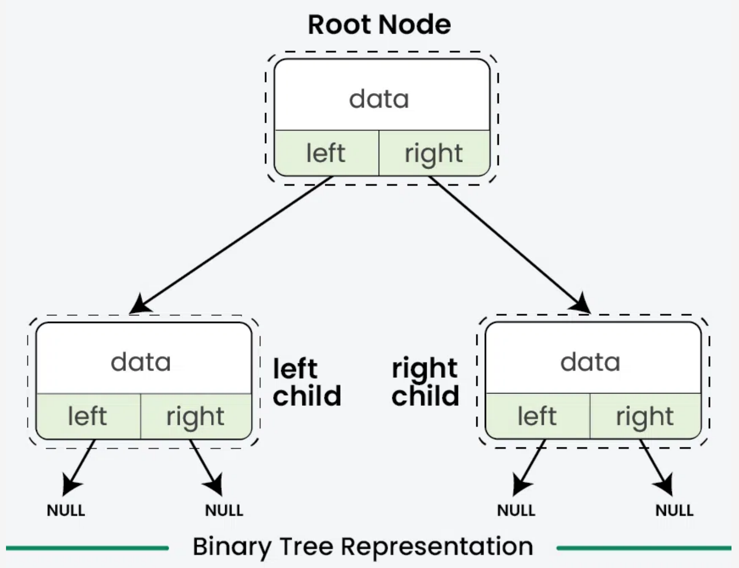
Binary Tree Representation and Properties

Binary Tree Representation

. Binary tree is a tree data structure (non-linear) in which each node can have at most two children which are referred to as the left child and the right child. The topmost node in a binary tree is called the root, and the bottom-most nodes are called leaves. A binary tree can be visualized as a hierarchical structure with the root at the top and the leaves at the bottom.

. Binary trees can be represented in multiple ways, each with its own advantages, depending on the use case. Let's explore the two common methods: linked node representation and array implementation.

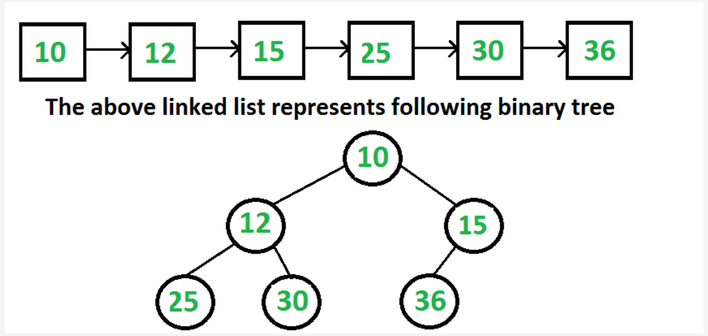


Representation of Binary Trees

1 Linked Node Representation

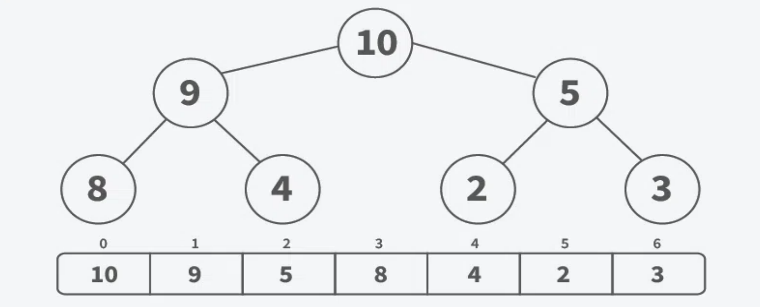
This is the simplest way to represent a binary tree. Each node contains data and pointers to its left and right children.

This representation is mostly used to represent binary tree with multiple advantages..



2 Array Representation

· Array Representation is another way to represent binary trees, especially useful when the tree is complete (all levels are fully filled except possibly the last, which is filled from left to right)



Array Representation of Binary Tree

Advantages and disadvantages of linked node represention

Advantages

1. It can easily grow or shrink as needed, so it uses only the memory it needs.

2. Adding or removing nodes is straightforward and requires only pointer adjustments.

.

3. Only uses memory for the nodes that exist, making it efficient for sparse trees.

Disadvantages

1. Needs extra memory for pointers.

2. Finding a node can take longer because you have to start from the root and follow pointers.

Hint: **Effect Options** gives you even more options for **Morph**.

Advantages and disadvantages of Array representation

Advantages

1. Easy to navigate parent and child nodes using index calculations, which is fast
2. Easier to implement, especially for complete binary trees

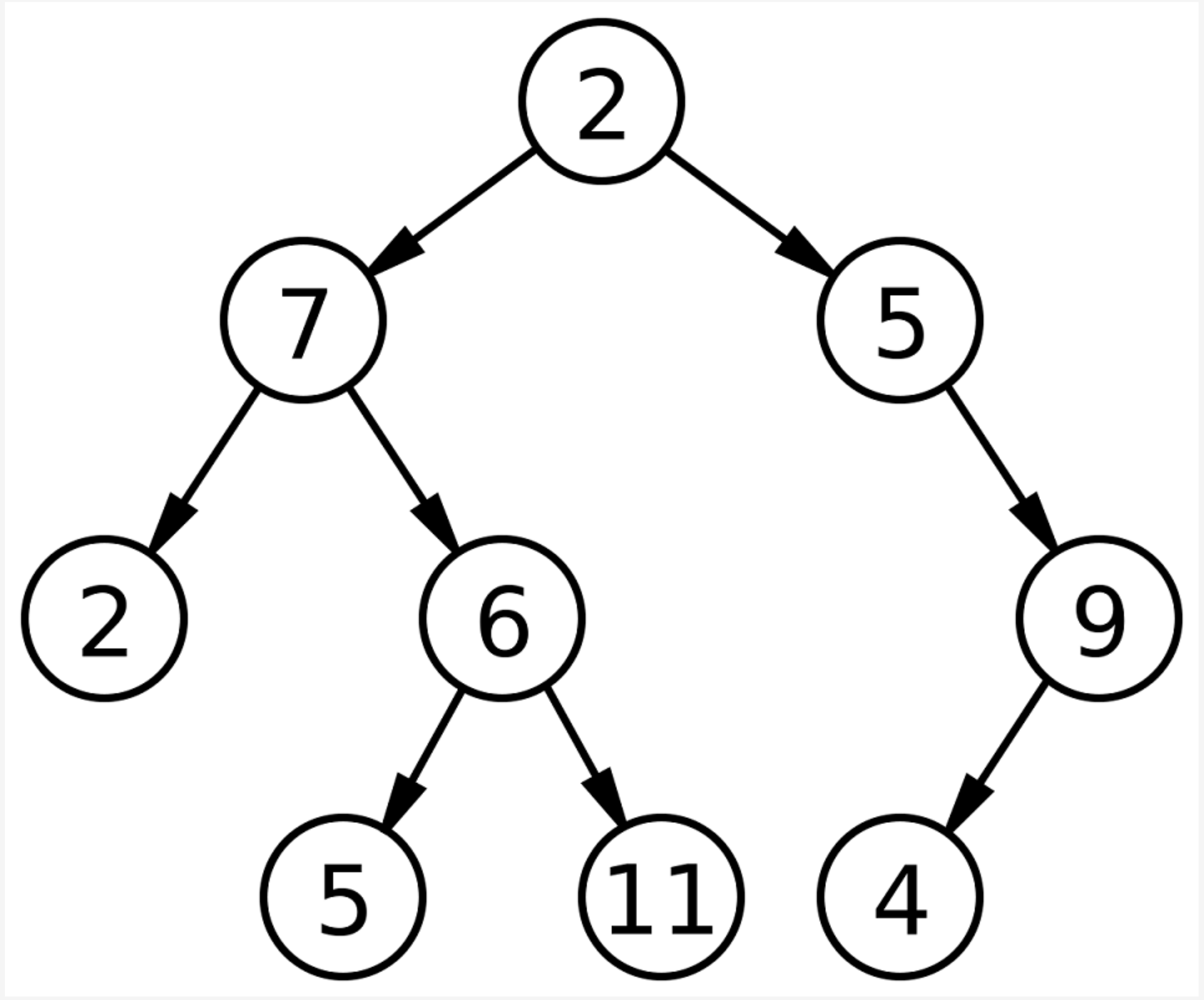
disadvantages

1. You have to set a size in advance, which can lead to wasted space.
2. If the tree is not complete binary tree then then many slots in the array might be empty, this will result in wasting memory

Properties of Binary Tree

The common non-linear data structure known as a tree. A tree illustrates a hierarchical structure in contrast to other data structures such an array, stack, queue, and linked list, which are linear in nature. A tree's ordering information is irrelevant. Two pointers and nodes make up a tree. The parent node's left and right children are represented by these two pointers.

- **Maximum Nodes at Level**
- **Maximum Nodes in a Binary Tree of Height 'h**
- **Minimum Height for 'N' Nodes**
- **Minimum Levels for 'L' Leaves**
- **Nodes with Two Children vs. Leaf Nodes**
- **Total Edges in a Binary Tree**



Properties of Binary Tree

- Maximum Nodes at Level 'l'

A binary tree can have at most $\mathbf{2^l}$ nodes at level **l**.
Level Definition: The number of edges in the path from the root to a node. The root is at level **0**.

- Maximum Nodes in a Binary Tree of Height 'h'

A binary tree of height **h** can have at most $\mathbf{2^{h+1} - 1}$ nodes.
Height Definition: The longest path from the root to a leaf node.
Please note that a tree with only one root node is considered to have height 0 and an empty tree (or root is NULL) is considered to have height "-1"

Alternate Height Convention: Some books consider a tree with only one root node is considered to have height 1 and an empty tree (or root is NULL) is considered to have height 0. making the formula $\mathbf{2^h - 1}$**.**

- Minimum Height for 'N' Nodes

The minimum possible height for **N** nodes is $\lfloor \log_2 N \rfloor$.
Explanation: A binary tree with height **h** can have at most $\mathbf{2^{h+1} - 1}$ nodes.

Properties of Binary Tree

- Minimum Levels for 'L' Leaves

  A binary tree with **L** leaves must have at least **⌈$\log_2 L$⌉** levels.
  Why? A tree has the **maximum number of leaves when all levels are fully filled**.

- Nodes with Two Children vs. Leaf Nodes

  In a **full binary tree** (where every node has either 0 or 2 children),
  the number of **leaf nodes (L) is always one more than the internal nodes (T) with two children**:

- Total Edges in a Binary Tree

  In any **non-empty binary tree** with **n** nodes, the total number of edges is **n - 1**

Key Properties

Node Relationships

Each node has at most **two children**.

- **0 children → Leaf Node**
- **1 child → Unary Node**
- **2 children → Binary Node**

Types of Binary Trees

Full Binary Tree → Every non-leaf node has exactly **two children**.

Complete Binary Tree → All levels are **fully filled except possibly the last**, which is filled from **left to right**.

Perfect Binary Tree → Every level is **completely filled**, and all **leaves are at the same depth**.

Balanced Binary Tree → The **left and right subtrees differ in height by at most 1**.

Tree Traversal Methods

Tree traversal is categorized into **Depth-First Search (DFS)** and **Breadth-First Search (BFS):**

DFS Traversals: Explore one branch fully before backtracking.

- **In-Order (LNR):** Left → Node → Right (retrieves BST elements in sorted order).
- **Pre-Order (NLR):** Node → Left → Right (used for tree reconstruction).
- **Post-Order (LRN):** Left → Right → Node (helps in deleting or evaluating expressions).

BFS Traversals: Visit nodes level by level.

- **Level-Order:** Processes nodes from top to bottom (used in shortest path algorithms).
- **Zig-Zag Traversal:** Alternates left-to-right and right-to-left at each level (used in hierarchical structures)

Thank you