■ Binary Trees (Array & Linked Representation)

1. Binary Tree Basics

A Binary Tree is a tree data structure where each node has at most two children, called Left Child and Right Child.

2. Array Representation of Binary Tree

A binary tree can be represented using a 1-D array (level-order representation). The root is stored at index 1.

Index Rules (root at index 1):

- Left Child → 2 * i

- Right Child → 2 * i + 1

- Parent → i // 2

Advantages: Simple, compact storage; easy parent/child access.

Disadvantages: Wastes space in skewed trees; difficult for dynamic changes.

Example:

Tree: A / \ B C / \ D E Array Representation (1-based index): Index: 1 2 3 4 5 Value: A B C D E

3. Linked Representation of Binary Tree

Each node has: Data, Left pointer (child), Right pointer (child).

Node Structure (C-like pseudocode):

struct Node { int data; struct Node* left; struct Node* right; };

Advantages: Efficient memory, dynamic insertion/deletion.

Disadvantages: Requires extra memory for pointers; parent access not direct.

4. Comparison Table

Feature	Array Representation	Linked Representation
Storage	Continuous memory (1-D array)	Dynamic nodes with pointers
Parent/Child Access	Easy using formulas (2i, 2i+1)	Requires pointers
Space Utilization	Poor for sparse/skewed trees	Efficient (only memory for nodes)
Insertion/Deletion	Difficult (shifting needed)	Easy (update pointers)
Traversal	Possible but less flexible	Natural using pointers