AKTU · 2ND YEAR · SEMESTER 3 · BCS301

Data Structure

Unit 4 — Trees

- Basic terminology used with Tree
- Binary Trees, Binary Tree Representation: Array Representation and Pointer (Linked List) Representation
- Binary Search Tree, Strictly Binary Tree, Complete Binary Tree, Extended Binary Trees
- Tree Traversal algorithms: Inorder, Preorder and Postorder
- Constructing Binary Tree from given Tree Traversal



DECODED BY

Chirag Jain

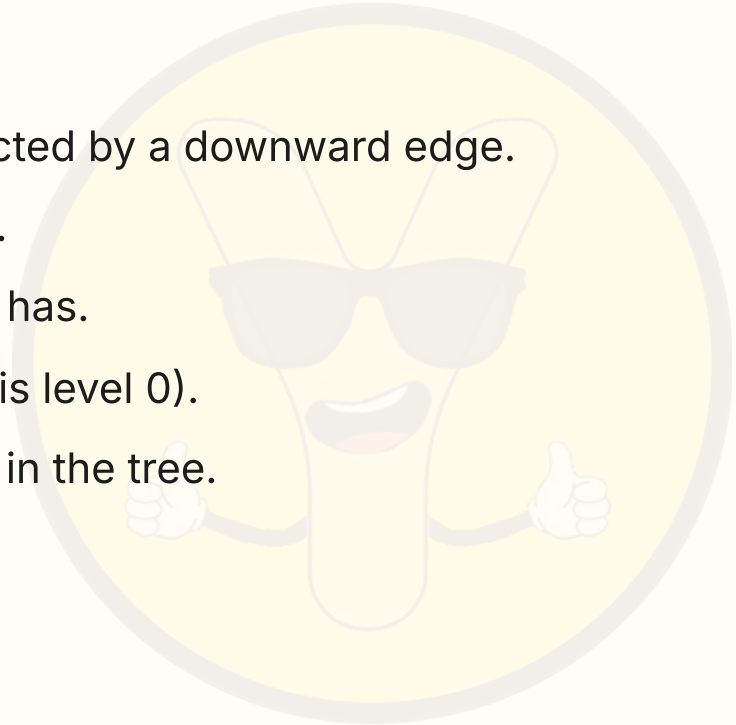
Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

theycotes.com

Introduction to Trees

A tree is a non-linear hierarchical data structure consisting of nodes connected by edges.

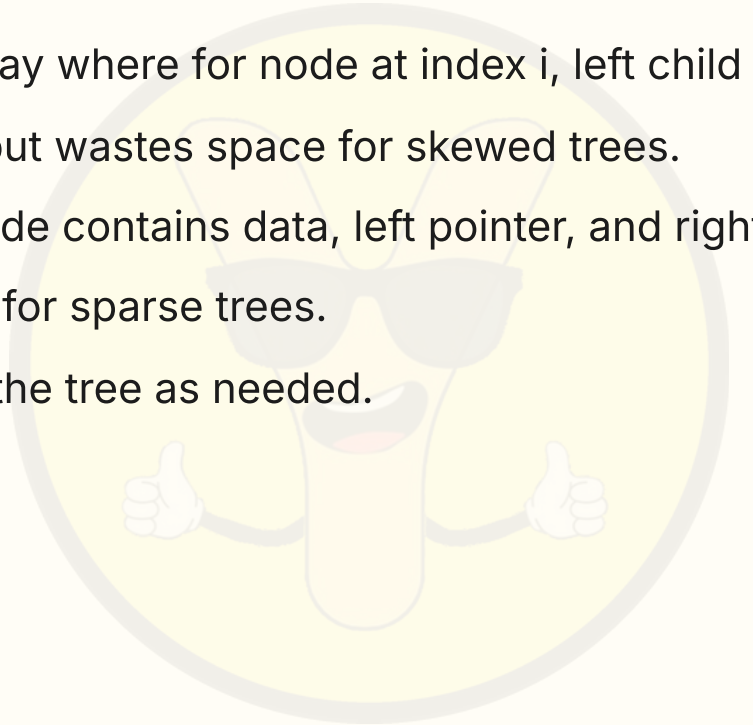
- Root: The topmost node of the tree.
- Parent/Child: Nodes directly connected by a downward edge.
- Leaf Node: A node with no children.
- Degree: Number of children a node has.
- Level: Distance from the root (Root is level 0).
- Height: Maximum level of any node in the tree.



Binary Tree Representation

Binary trees can be stored in memory using two primary methods.

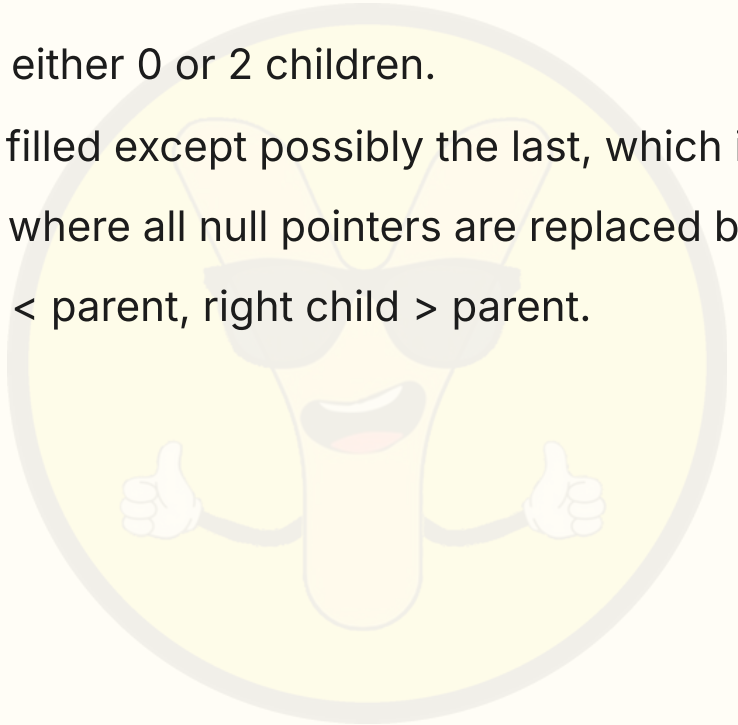
- Array Representation: Uses a 1D array where for node at index i, left child is at 2i+1 and right child at 2i+2.
- Efficient for complete binary trees but wastes space for skewed trees.
- Linked List Representation: Each node contains data, left pointer, and right pointer.
- More flexible and memory-efficient for sparse trees.
- Dynamic allocation allows growing the tree as needed.



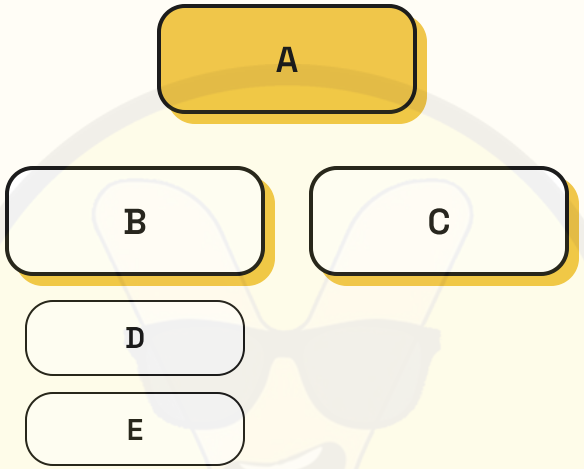
Types of Binary Trees

Specialized binary trees defined by their structural properties.

- Strictly Binary Tree: Every node has either 0 or 2 children.
- Complete Binary Tree: All levels are filled except possibly the last, which is filled left-to-right.
- Extended Binary Tree: A binary tree where all null pointers are replaced by special 'external' nodes.
- Binary Search Tree (BST): Left child < parent, right child > parent.



Binary Tree Structure

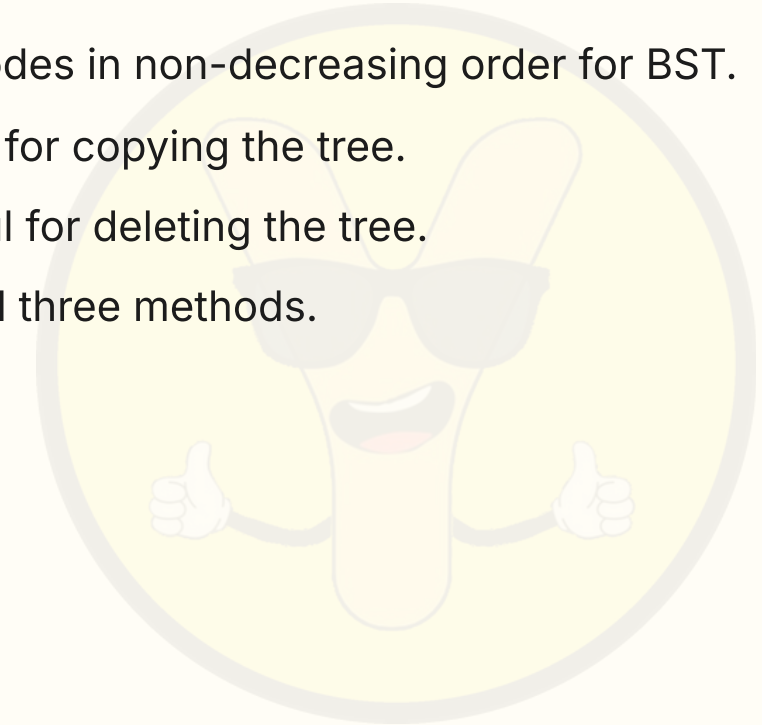


A simple binary tree with root A and children B and C.

Tree Traversal Algorithms

Traversals visit every node in the tree exactly once.

- Inorder (Left-Root-Right): Visits nodes in non-decreasing order for BST.
- Preorder (Root-Left-Right): Useful for copying the tree.
- Postorder (Left-Right-Root): Useful for deleting the tree.
- Algorithm complexity is O(n) for all three methods.



Constructing Binary Tree

Construct a tree given Inorder: [D, B, E, A, F, C] and Preorder: [A, B, D, E, C, F]

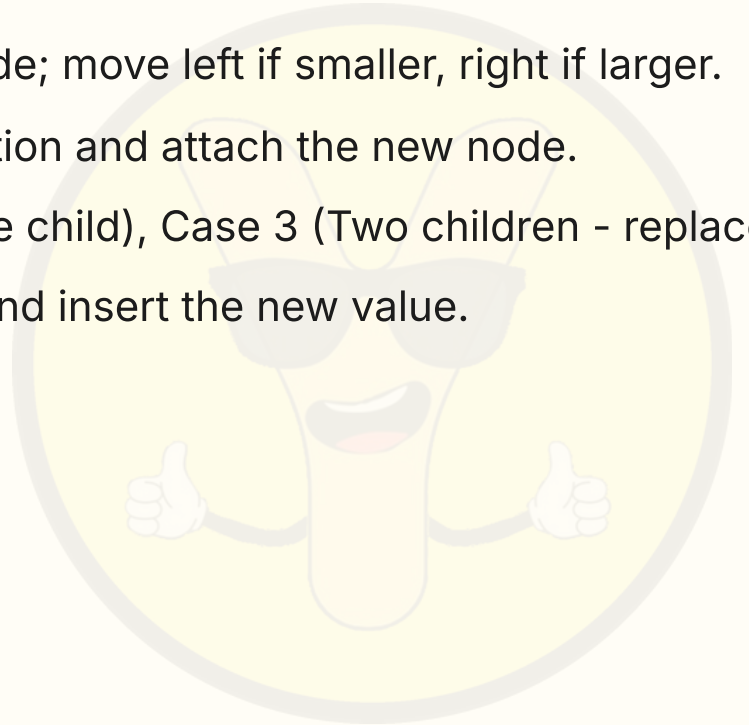
1. Use Preorder to identify the root (First element is A).
2. Locate A in Inorder to split into Left (D, B, E) and Right (F, C) subtrees.
3. Recursively apply the process for subtrees using the next elements in Preorder.

A tree with root A, left child B, and right child C.

Binary Search Tree (BST) Operations

BST allows efficient searching, insertion, and deletion.

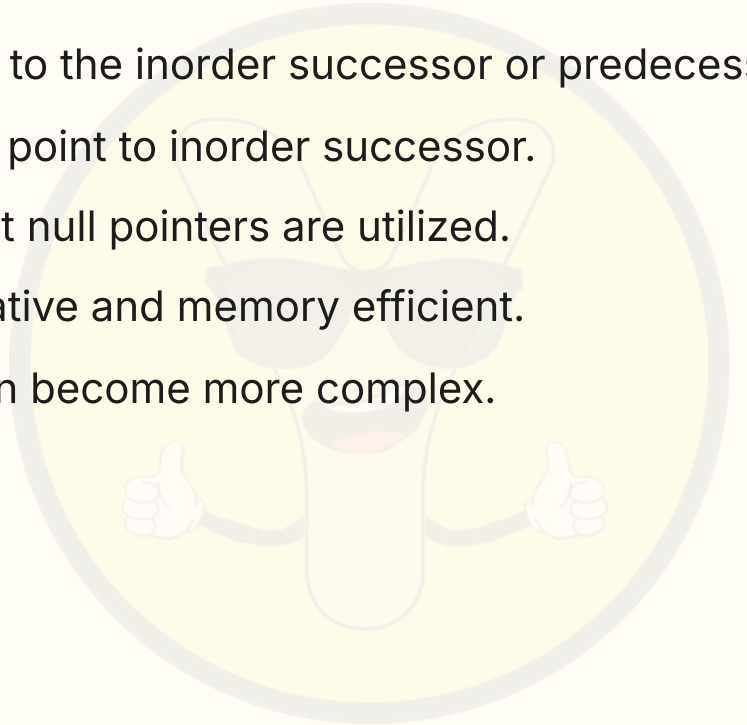
- Searching: Compare target with node; move left if smaller, right if larger.
- Insertion: Find the correct leaf position and attach the new node.
- Deletion: Case 1 (Leaf), Case 2 (One child), Case 3 (Two children - replace with inorder successor).
- Modification: Delete the old value and insert the new value.



Threaded Binary Trees

A technique to traverse a tree without recursion or stack.

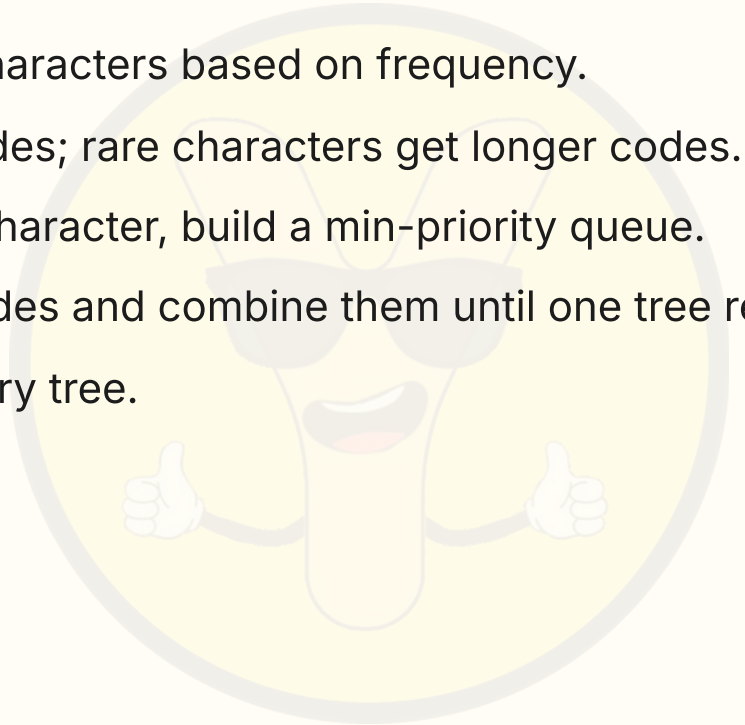
- Uses null pointers to store 'threads' to the inorder successor or predecessor.
- Single Threaded: Null right pointers point to inorder successor.
- Double Threaded: Both left and right null pointers are utilized.
- Advantage: Traversal becomes iterative and memory efficient.
- Disadvantage: Insertion and deletion become more complex.



Huffman Coding

A greedy algorithm for lossless data compression.

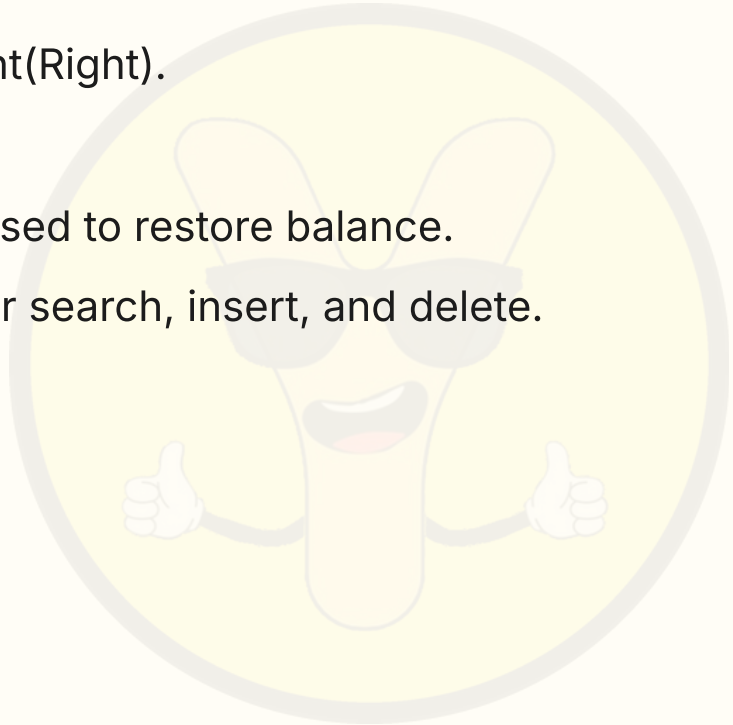
- Assigns variable-length codes to characters based on frequency.
- Frequent characters get shorter codes; rare characters get longer codes.
- Steps: Create leaf nodes for each character, build a min-priority queue.
- Repeatedly extract two smallest nodes and combine them until one tree remains.
- Result is an optimal prefix-free binary tree.



AVL Trees

A self-balancing Binary Search Tree.

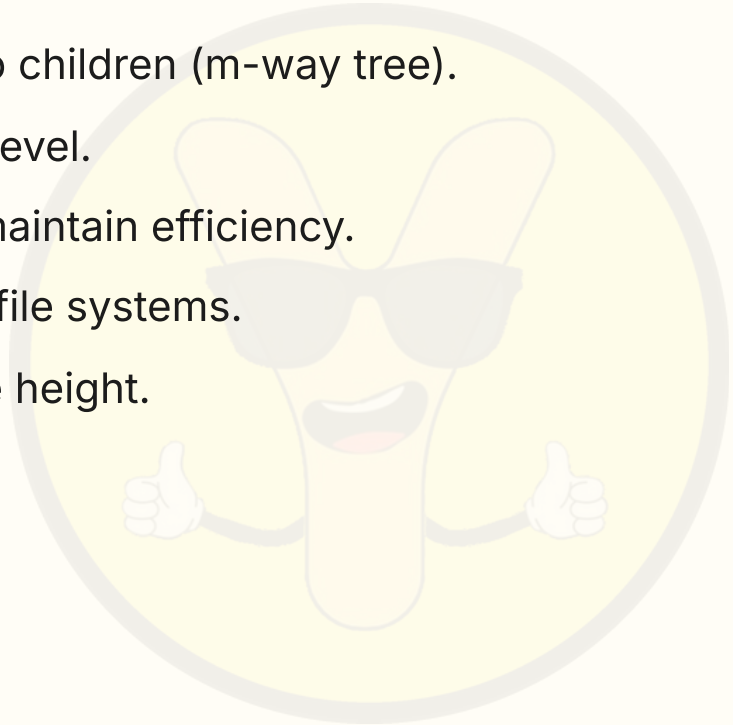
- Balance Factor: Height(Left) - Height(Right).
- Allowed Balance Factors: -1, 0, +1.
- Rotations: LL, RR, LR, RL rotations used to restore balance.
- Ensures O(log n) time complexity for search, insert, and delete.
- Ideal for read-heavy applications.



B-Trees

A multi-way balanced search tree optimized for disk storage.

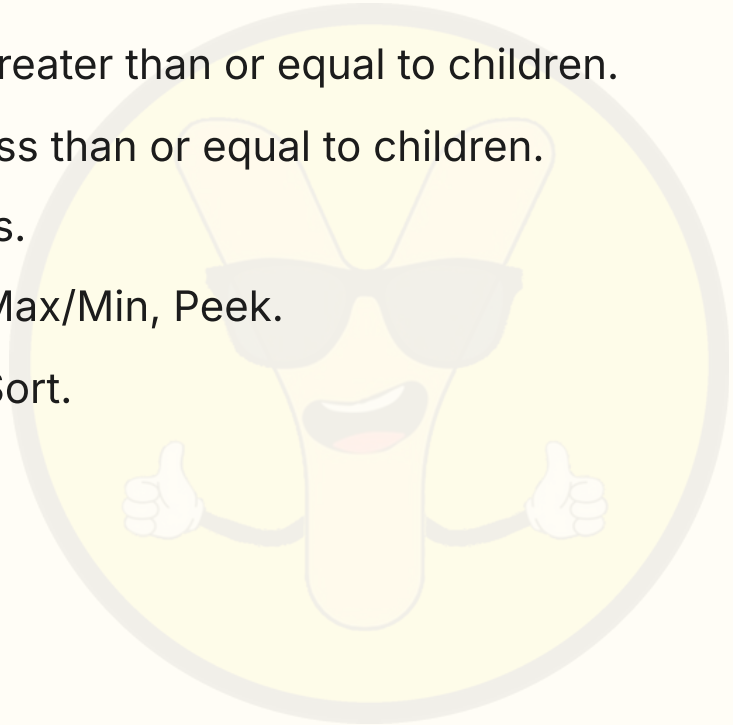
- Every node can have more than two children (m-way tree).
- All leaf nodes must be at the same level.
- Nodes are kept at least half full to maintain efficiency.
- Used extensively in databases and file systems.
- Reduces disk I/O by minimizing tree height.



Binary Heaps

A complete binary tree that satisfies the heap property.

- Max-Heap: Parent node is always greater than or equal to children.
- Min-Heap: Parent node is always less than or equal to children.
- Implemented efficiently using arrays.
- Primary operations: Insert, Delete-Max/Min, Peek.
- Used in Priority Queues and Heap Sort.



Exam corner

- **Q1** Differentiate between a Binary Tree and a Binary Search Tree.
- **Q2** Explain the three types of tree traversals with algorithms.
- **Q3** How do you delete a node with two children in a BST?
- **Q4** What is a Threaded Binary Tree? Explain its advantages.
- **Q5** Construct a Huffman tree for a given set of character frequencies.
- **Q6** Explain the need for AVL trees and the four types of rotations.
- **Q7** Compare B-Trees and Binary Search Trees in terms of disk storage.

Unit 4 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 5 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

