IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Data Structures

BINARY TREES AND HASHING

D. A. Ramacharyulu
IARE10859
Mechanical Engineering
Lecture Number - 40
Presentation Date - 04-11-2024

Course Outcomes

At the end of the course, students should be able to:

CO6: Compare various types of data structures in terms of implementation, operations and performance.

UNIT –V
BINARY TREES AND HASHING

Contents

- Binary search trees: Binary search trees, properties and operations; Balanced search trees: AVL trees; Introduction to M- Way search trees, B trees;
- Hashing and collision: Introduction, hash tables, hash functions, collisions, applications of hashing.

Binary Search Tree:

- A **Binary Search Tree (or BST)** is a data structure used in computer science for organizing and storing data in a sorted manner.
- Each node in a **Binary Search Tree** has at most two children, a **left** child and a **right** child, with the **left** child containing values less than the parent node and the **right** child containing values greater than the parent node.
- This hierarchical structure allows for efficient **searching**, **insertion**, and **deletion** operations on the data stored in the tree.

Properties of Binary Search Tree:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree.
- There must be no duplicate nodes(BST may have duplicate values with different handling approaches).

- ***Binary search trees can be used to implement sorted stream of data and doubly ended priority queues.***

-

Insertion in Binary Search Tree:

- Initialize the current node (say, **currNode or node**) with root node
- Compare the **key** with the current node.
- **Move left** if the **key** is less than the current node value.
- **Move right** if the **key** is greater than or equal to current node value.
- Repeat steps 2 and 3 until you reach a leaf node.
- Attach the **new key** as a left or right child based on the comparison with the leaf node's value.

Searching in Binary Search Tree:

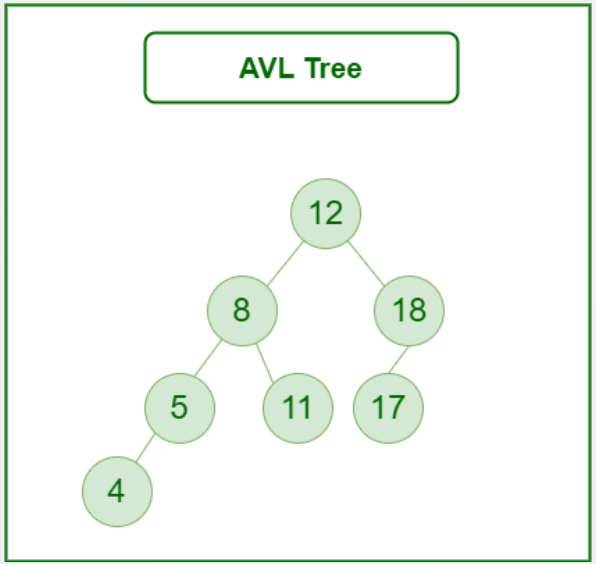
- Compare the value to be searched with the value of the root.
    - If it’s equal we are done with the search if it’s smaller go to the left subtree
    - If it is greater then go to the right subtree.
- Repeat the above step till no more traversal is possible
- If at any iteration, key is found, return True. Else False.

Searching in Binary Search Tree:

- Compare the value to be searched with the value of the root.
    - If it’s equal we are done with the search if it’s smaller go to the left subtree
    - If it is greater then go to the right subtree.
- Repeat the above step till no more traversal is possible
- If at any iteration, key is found, return True. Else False.

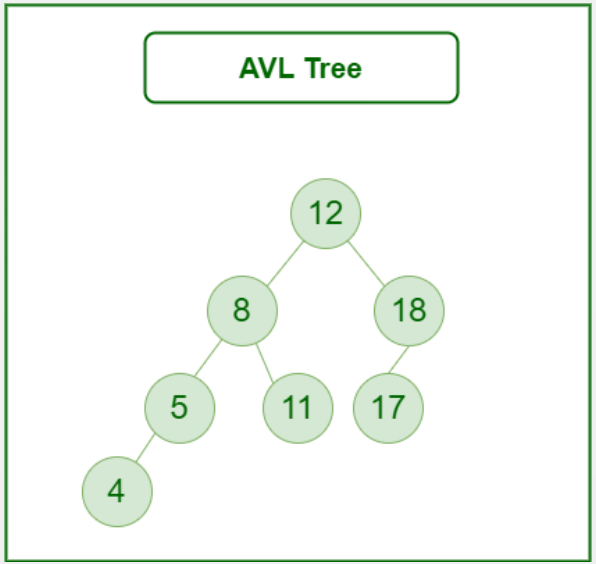
AVL Tree:

*An **AVL tree** defined as a self-balancing **Binary Search Tree** (BST) where the difference between heights of left and right subtrees for any node cannot be more than one.*



AVL Tree:

*An **AVL tree** defined as a self-balancing **Binary Search Tree** (BST) where the difference between heights of left and right subtrees for any node cannot be more than one.*



M-Way Search Tree:

M-way search tree us Multi way search tree, this is similar to Binary search tree.

Binary Search Tree	M-Way search tree
Each node in BST have one key	A search tree in which each node can have m-pointers and (m-1)
Each node can have at-most 2 child	In 3 way search tree, each node can have 2 keys and 3-pointers.

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Thank You