Trees

Chapter No.	14
Status	Completed

- A tree is a non-linear, hierarchical data structure

▼ Tree Terminology

▼ Node

- A node is represented by an oval and each node stores a value

▼ Edge

- An edge is represented by an arrow and each edge connects two nodes to show the relationship between them

▼ Root

- The root is the only node that has no incoming edge

▼ Parent, Child & Sibling

- A parent node is connected to its child node through an edge
- The children nodes of the same parent node are sibling nodes

▼ Subtree (Branch)

- A subtree or branch is a set of nodes and edges comprising a parent node and all the descendant nodes of that parent node

▼ Leaf

- A leaf node is a node that has no children

▼ Path

- A path is an ordered list of nodes that are connected by edges

▼ Level

- The level of a node refers to its depth in a tree
- By definition, the level of the root node is 0

- Height
  - The height of the tree refers to how deep the entire tree is
- Size
  - The size of the tree refers to the total number of nodes present

- Binary Tree
  - A binary tree is a tree where each node has at most two children
  - The two children of a node are commonly referred to as the left child and the right child of the node

- Tree Traversal
  - Pre-Order Traversal
    - Root → Left Subtree → Right Subtree
  - In-Order Traversal
    - Left Subtree → Root → Right Subtree
  - Post-Order Traversal
    - Left Subtree → Right Subtree → Root

- Binary Search Tree
  - A binary search tree is a binary tree in which all values smaller than a node are stored in that node's left subtree and all values larger than a node are stored in that node's right subtree
  - Common Binary Search Tree Operations
    - create
      - Create a new binary search tree with a root node of a given value without any children
    - edit
      - Change the value of a node in the binary search tree
    - search

        - Search a given value in the binary search tree

    ▼ insert

        - Insert a given value into the binary search tree

    ▼ delete

        - Delete a node of a given value in the binary search tree and possibly rearrange the tree
        - If a node to be deleted has no children (leaf node), there is no need to shift any of the remaining nodes
        - If the node to be deleted has one child, the child node will take the deleted node's place

        ▼ If the node to be deleted has two children, there are two possibilities

            - The largest node in the left subtree takes the deleted node's place
            - The smallest node in the right subtree takes the deleted node's place

- In-order traversal of a binary search tree returns the values sorted in increasing order

▼ Rebalancing Binary Search Trees

- A binary search tree is unbalanced when one side has many more element than the other side
- This makes binary search of the binary search tree inefficient because the time complexity of binary search becomes almost O(n) instead of O(lg n)
- In this case, the binary search tree needs to be re-created or rebalanced with the same items to make it balanced
- This distributes the items more evenly between the left and right subtrees, ensuring that the time complexity of binary search is O(lg n)

▼ Advantages of Binary Search Trees

- Binary search can be used to search for items instead of less efficient searching algorithms, such as linear search
- Insertion and deletion of items involves the updating of pointers, similar to linked lists, instead of rearranging elements