**Week-2 Documentation**

**1. Introduction to Binary Trees**

A **Binary Tree** is a hierarchical data structure in which each node has at most two children, referred to as the left child and the right child. It is a foundational structure used in various algorithms and applications like expression trees, heaps, and more.

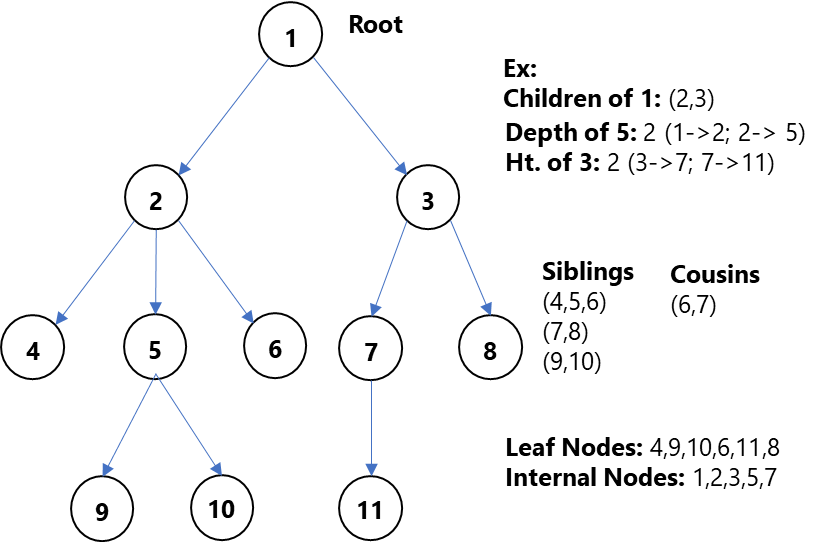
**Properties of a Binary Tree**

**Maximum Two Children**: Each node can have **0, 1, or 2** children.

**Subtree Structure**: Each node’s children are also binary trees (recursive structure).

**Types of Binary Trees**:

* + **Full Binary Tree**: A binary tree in which every node has either **0** or **2** children.
  + **Complete Binary Tree**: A binary tree that is filled except possibly for the last level, which is filled from left to right.
  + **Perfect Binary Tree**: A binary tree in which all interior nodes have two children, and all leaves are at the same level.
  + **Balanced Binary Tree**: A binary tree in which the height of the left and right subtrees of any node differs by no more than one.
  + **Degenerate Tree (Skewed Tree)**: A tree where each parent node has only one child. This creates a structure that resemble es a linked list.



**Common Operations on Binary Trees:**

**Insertion**: Add a new node to the tree following specific rules (depending on the type of binary tree).

**Creation of a node, Insertion Method.**

A screen shot of a computer program

Description automatically generated

**Traversal**: Visiting all the nodes in some order:

* + **In-order Traversal**: Left subtree → Root → Right subtree.
  + **Pre-order Traversal**: Root → Left subtree → Right subtree.
  + **Post-order Traversal**: Left subtree → Right subtree → Root.

1. **Searching**: Finding a node with a specific value (no order, so typically a linear search).
2. **Deletion**: Removing a node from the tree while maintaining the tree structure.

**A screen shot of a computer program

Description automatically generated**

**Inorder Traversal, Pre-order Traversal, Post-Order traversal.**

**A screen shot of a computer program

Description automatically generated**

**Deletion of a node, Recursive method, Finding minimum value node.**

**A screen shot of a computer program

Description automatically generated**

**Depth First Search**

**A screen shot of a computer program

Description automatically generated**

**Breath First Search**

**LEETCODE:**

**Depth of a binary tree**

**A screenshot of a computer program

Description automatically generated**

**Subtree of a Binary Tree:**

**A diagram of a root and subroot

Description automatically generated**

**Question:**

Given the roots of two binary trees root and subRoot, return true if there is a subtree of root with the same structure and node values of subRoot and false otherwise.

1. A subtree of a binary tree tree is a tree that consists of a node in tree and all of this node's descendants. The tree tree could also be considered as a subtree of itself.

**A screenshot of a computer code

Description automatically generated**

**A computer screen shot of a code

Description automatically generated**

**2. Introduction to Binary Search Trees (BST)**

A Binary Search Tree (BST) is a special type of binary tree that follows an additional property, which makes search operations more efficient.

**Properties of a Binary Search Tree**

**1. Binary Tree Properties:** The BST is a binary tree, meaning each node has at most two children.

**2. BST Search Property:**

* + The left subtree of a node contains only nodes with values less than the node's value.
  + The right subtree of a node contains only nodes with values greater than the node's value.
  + No duplicate values are allowed in a standard BST.

**Time Complexity of Operations in a BST**

* Best-case time complexity (Balanced BST): O (log n) for search, insert, and delete operations.
* Worst-case time complexity (Skewed BST): O(n) for search, insert, and delete operations.

**Advantages of BST**

* Fast lookups and deletions, particularly when the tree is balanced.
* Ordered structure, which makes it useful for operations that require sorted data.

**Key Operations on Binary Search Trees (BST)**

**Insertion in a BST**

* + Start from the root.
  + If the new value is less than the root, move to the left subtree; if greater, move to the right subtree.
  + Repeat the process recursively until an appropriate null spot is found and insert the new node there.

**Searching in a BST**

* + Start from the root.
  + If the value matches the root, return the node.
  + If the value is less than the root, move to the left subtree; if greater, move to the right subtree.
  + Continue recursively until the value is found or the search reaches a null node.

**Time Complexity:**

* Balanced BST: O(log n)
* Skewed BST: O(n)

**Deletion in a BST**

Deletion in a BST requires special handling to maintain the BST properties.

**There are three cases:**

Leaf Node: Simply remove the node.

Node with One Child: Replace the node with its child.

Node with Two Children: Find the node’s in-order successor (the smallest node in its right subtree), replace the node with the in-order successor, and then delete the in-order successor.

**Traversal Techniques in Binary Trees and BSTs**

In-order Traversal

* Procedure: Visit the left subtree, root, and then the right subtree.
* BST Property: In a BST, an in-order traversal will always result in a sorted sequence of values.

**Pre-order Traversal**

**Procedure:** Visit the root first, then the left subtree, and finally the right subtree.

**Post-order Traversal**

**Procedure:** Visit the left subtree, right subtree, and then the root.

**Implementation of a BST:**

**Inserting a Node, Recursive Function.**

A screen shot of a computer program

Description automatically generated

**Traversal Techniques:**

**A screenshot of a computer program

Description automatically generated**

**Deleting A Node:**

**A computer screen shot of a program code

Description automatically generated**

**Leetcode:**

**Search in a Binary Search Tree:**

**A screenshot of a computer

Description automatically generated**

**Delete a Node in BST:**

**A screenshot of a computer

Description automatically generated**

**3. Dynamic Programming (DP)** is a powerful algorithmic technique used to solve problems by breaking them down into simpler overlapping subproblems, solving each subproblem just once, and storing their solutions for future reuse.

**Key Concepts of Dynamic Programming:**

**Overlapping Subproblems:**

* + DP is useful when the problem can be broken down into subproblems that are reused multiple times.
  + Example: Fibonacci sequence – calculating Fibonacci (5) requires calculating Fibonacci (4) and Fibonacci (3), and so on.

**Optimal Substructure:**

* + A problem exhibits optimal substructure if an optimal solution to the problem can be constructed efficiently from optimal solutions to its subproblems.
  + Example: Shortest path in a graph – if you have the shortest path to node A, you can use it to build the shortest path to node B.

**Memoization vs. Tabulation:**

* + Memoization (Top-Down Approach): Solves the problem recursively and stores the result of each subproblem to avoid redundant work.
  + Tabulation (Bottom-Up Approach): Solves the problem iteratively by building up solutions to subproblems from smaller problems.

**Steps to Solve Problems Using DP:**

Identify if the problem can be solved using DP:

* + Look for overlapping subproblems and optimal substructure.

Define the DP array:

* + Decide what each entry in the DP array represents. For example, dp[i] might represent the minimum cost to reach the ith step.

**Formulate the recurrence relation:**

* Define how each subproblem relates to others. For example, for Fibonacci, the relation is dp[n] = dp[n-1] + dp[n-2].

**Initialize base cases:**

* + Define the known solutions to the smallest subproblems. For example, dp[0] = 0 and dp[1] = 1 in the Fibonacci sequence.

Iterate or recurse to fill the DP array:

* + Solve the subproblems in order, ensuring each subproblem is solved only once.

**LeetCode:**

**A screenshot of a computer program

Description automatically generated**