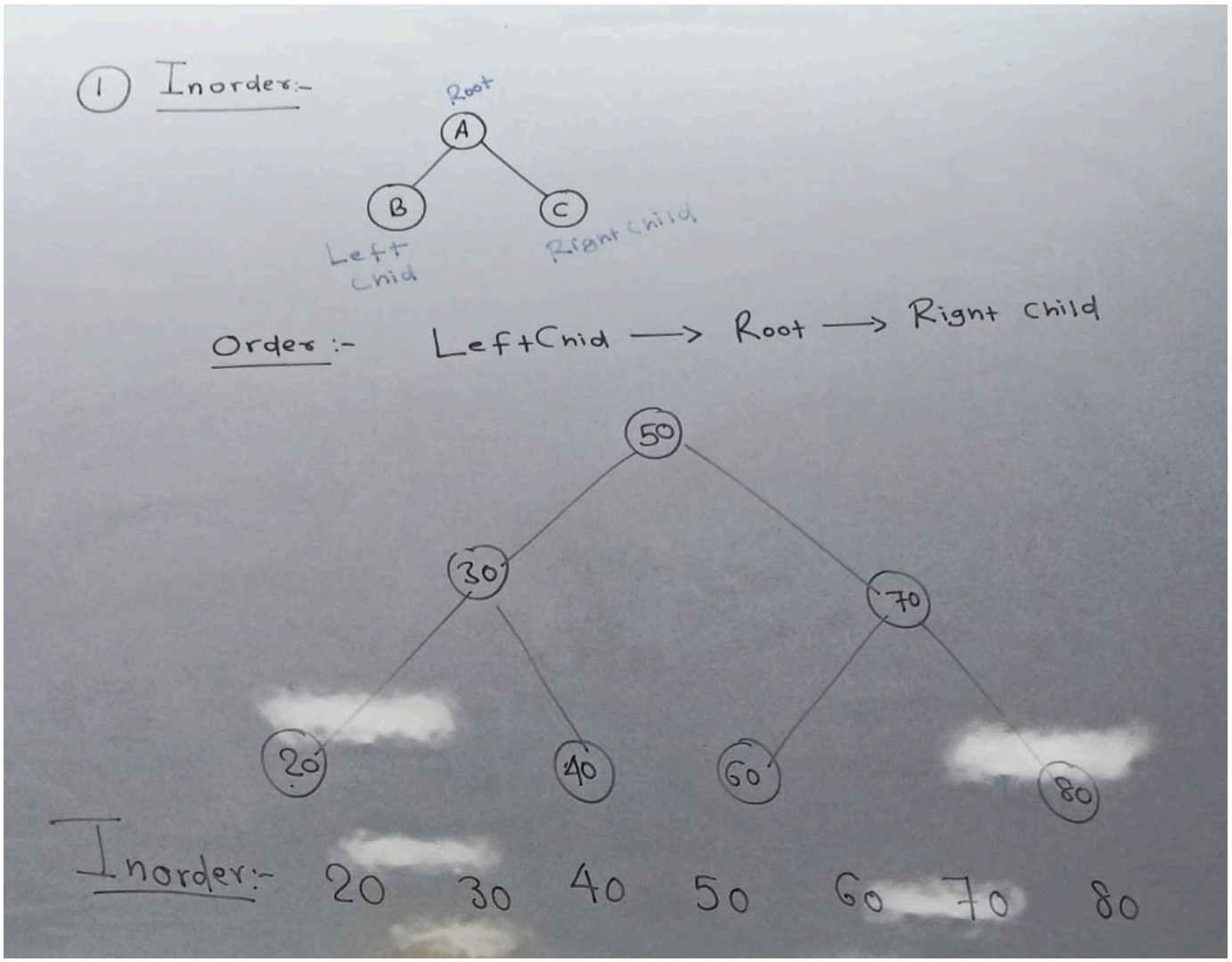
Tree Traversal

Accessing elements of a tree one by one is called **tree traversal**.

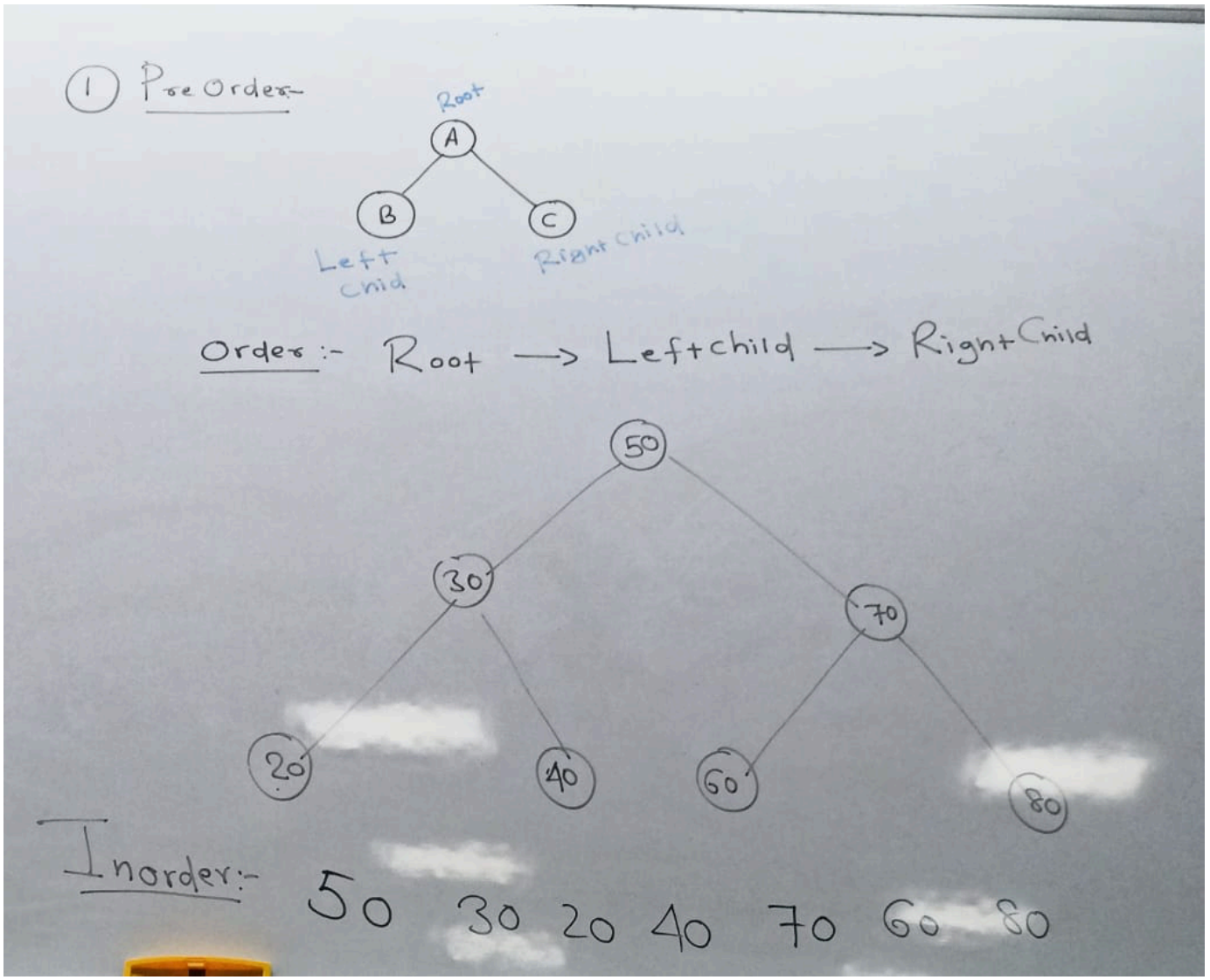
It is of two types:

1. Depth First Search (DFS)

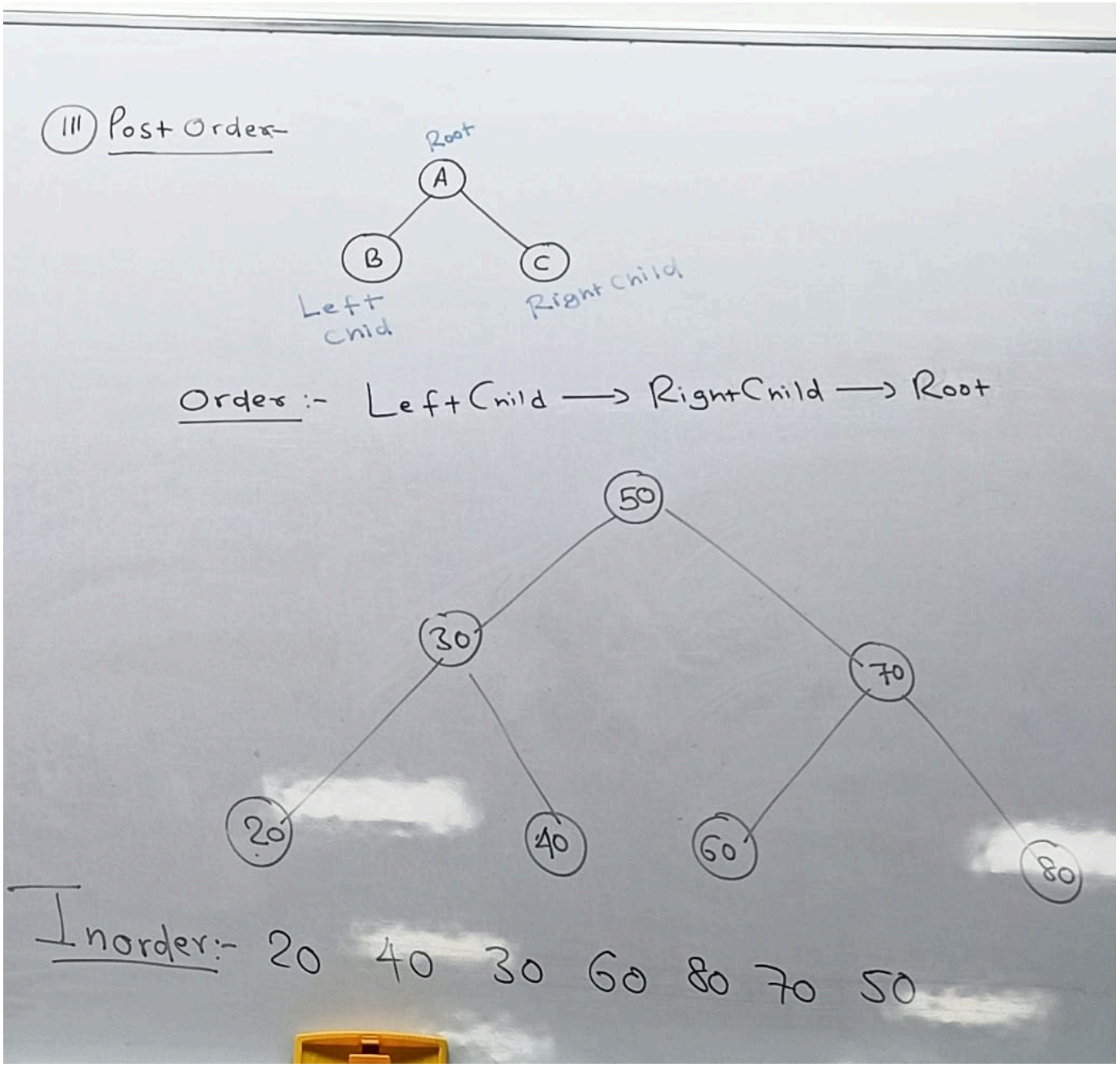
- **Inorder (Left → Root → Right)**



- **Preorder (Root → Left → Right)**



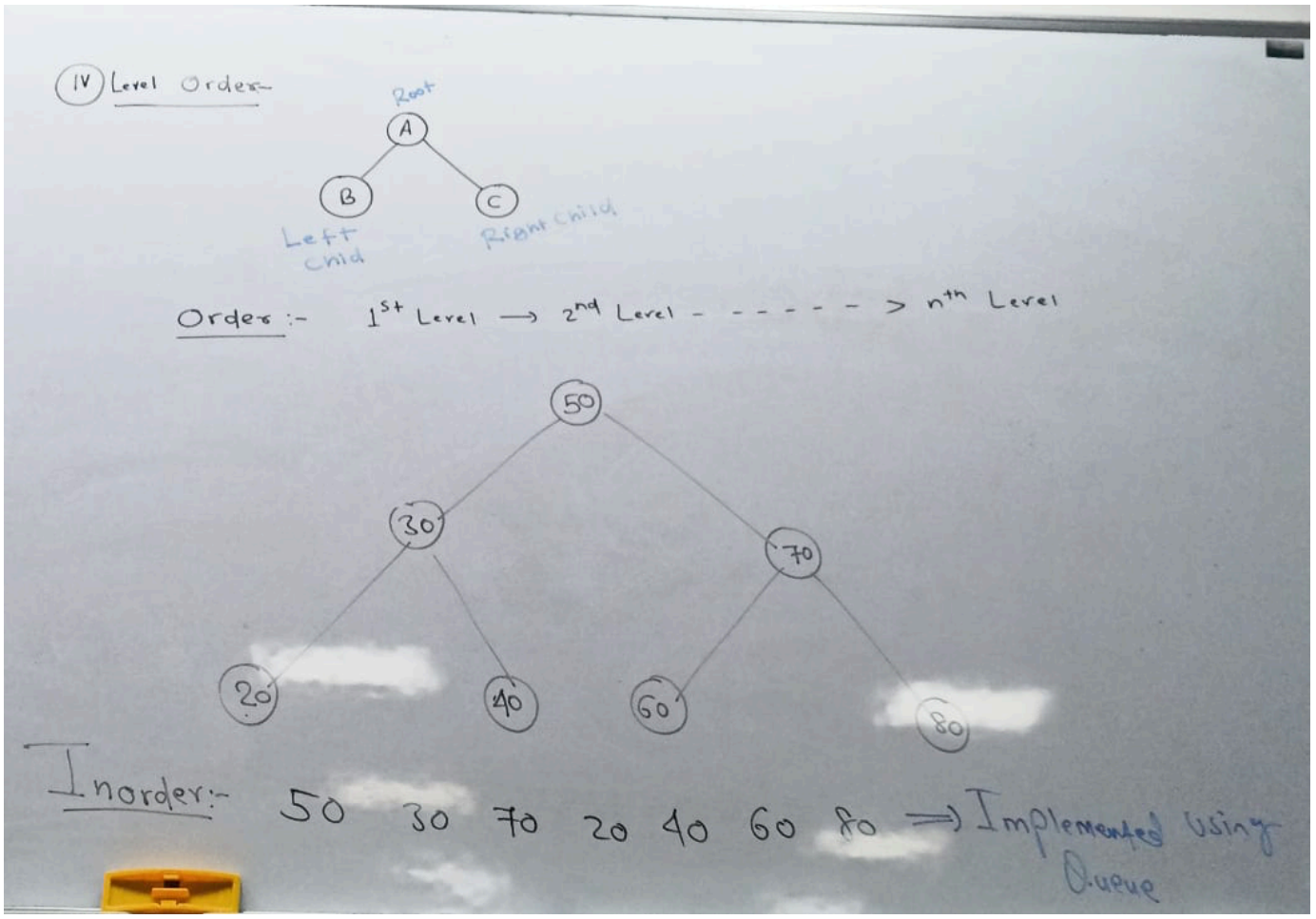
- **Postorder (Left → Right → Root)**



2. Breadth First Search (BFS)

- **Level Order**

  Access nodes level by level. Implemented using a **queue**.



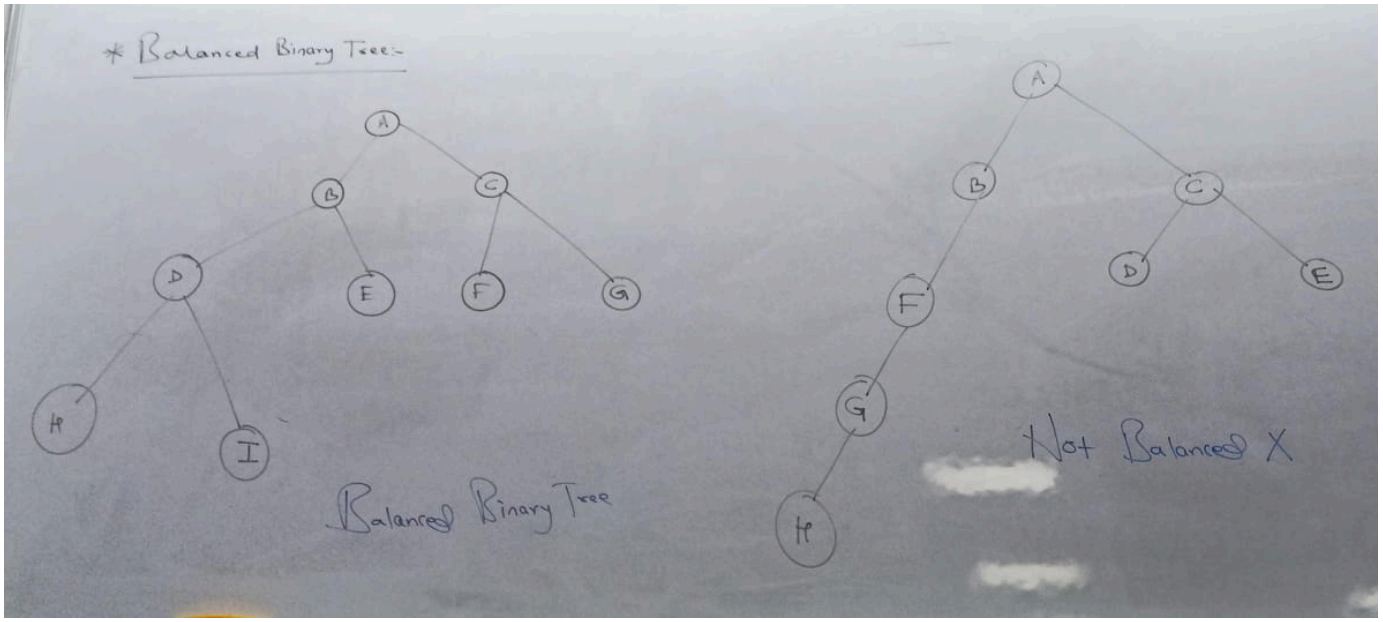
Special Types of Trees

1. Balanced Binary Tree

A binary tree where the height difference between the left subtree and right subtree of any node is at most 1.

2. Non-Balanced Binary Tree

A binary tree where the height difference between the left subtree and right subtree of any node is more than 1.

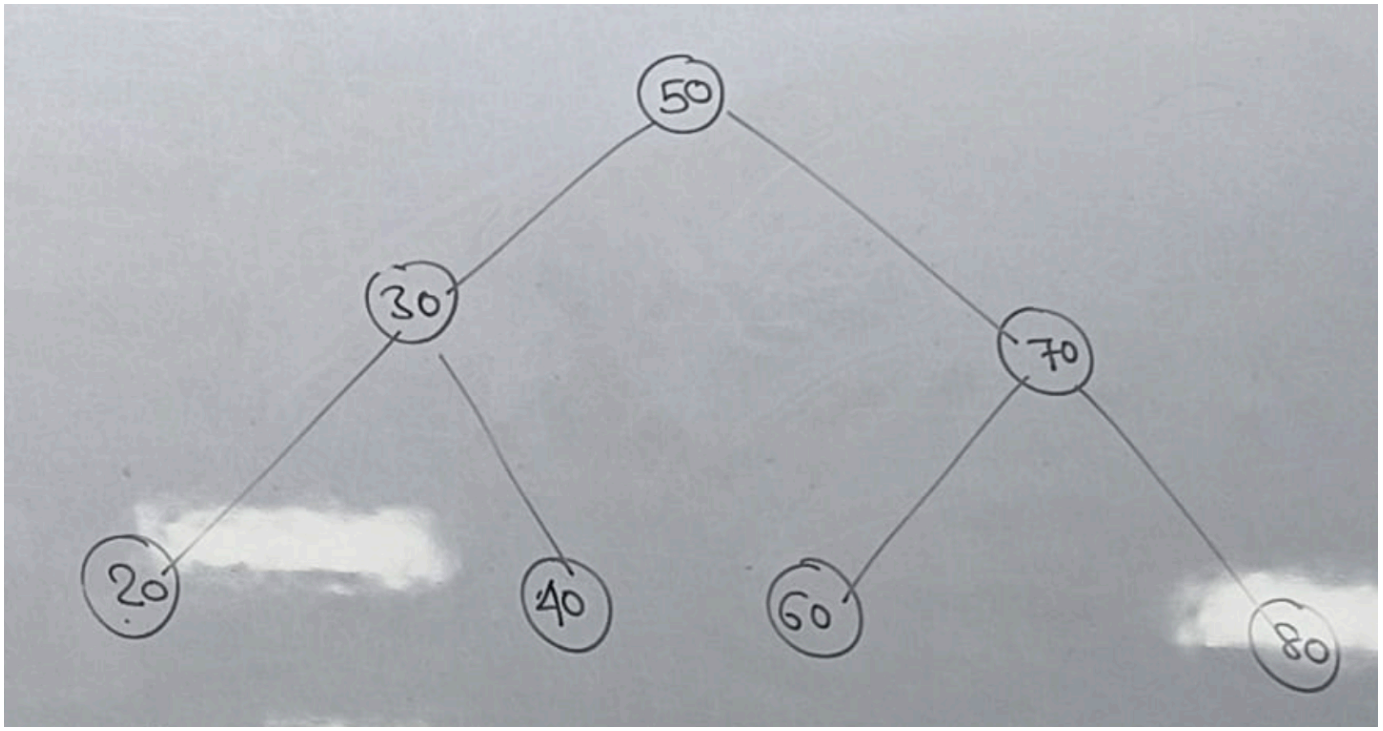


3. Binary Search Tree (BST)

A tree where:

- Left child value < Root value
- Right child value > Root value

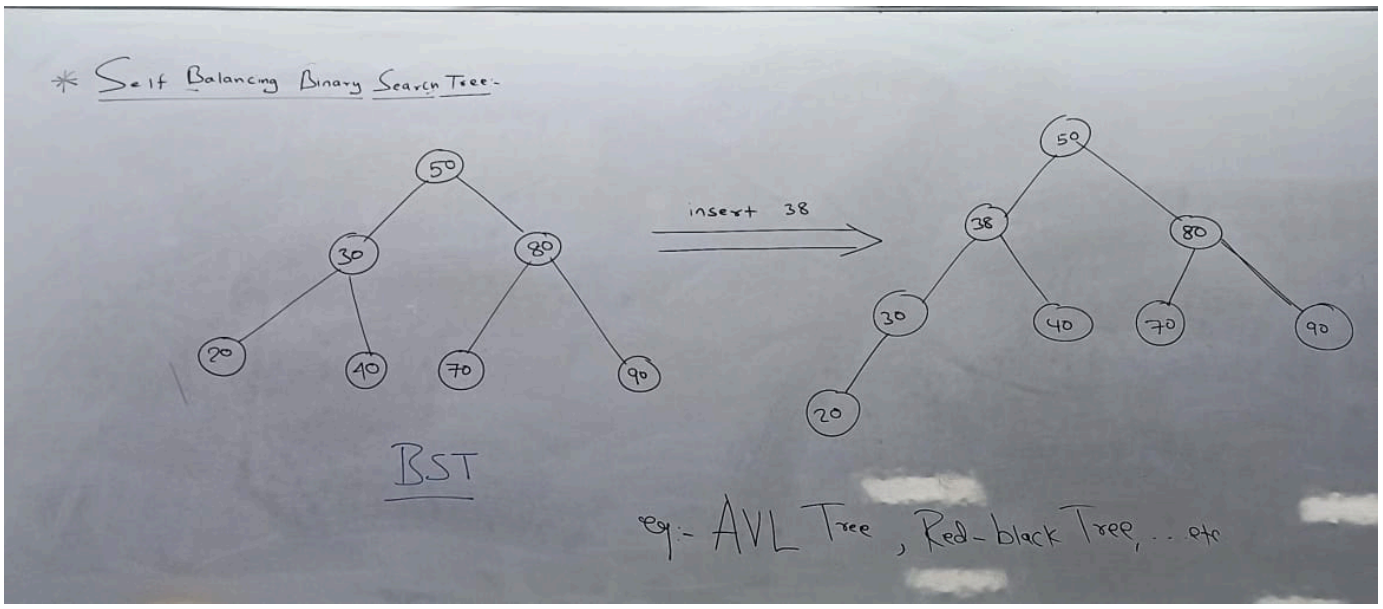
BST is very useful for achieving fast search, insertion, and deletion operations.



4. Self-Balancing Binary Search Tree

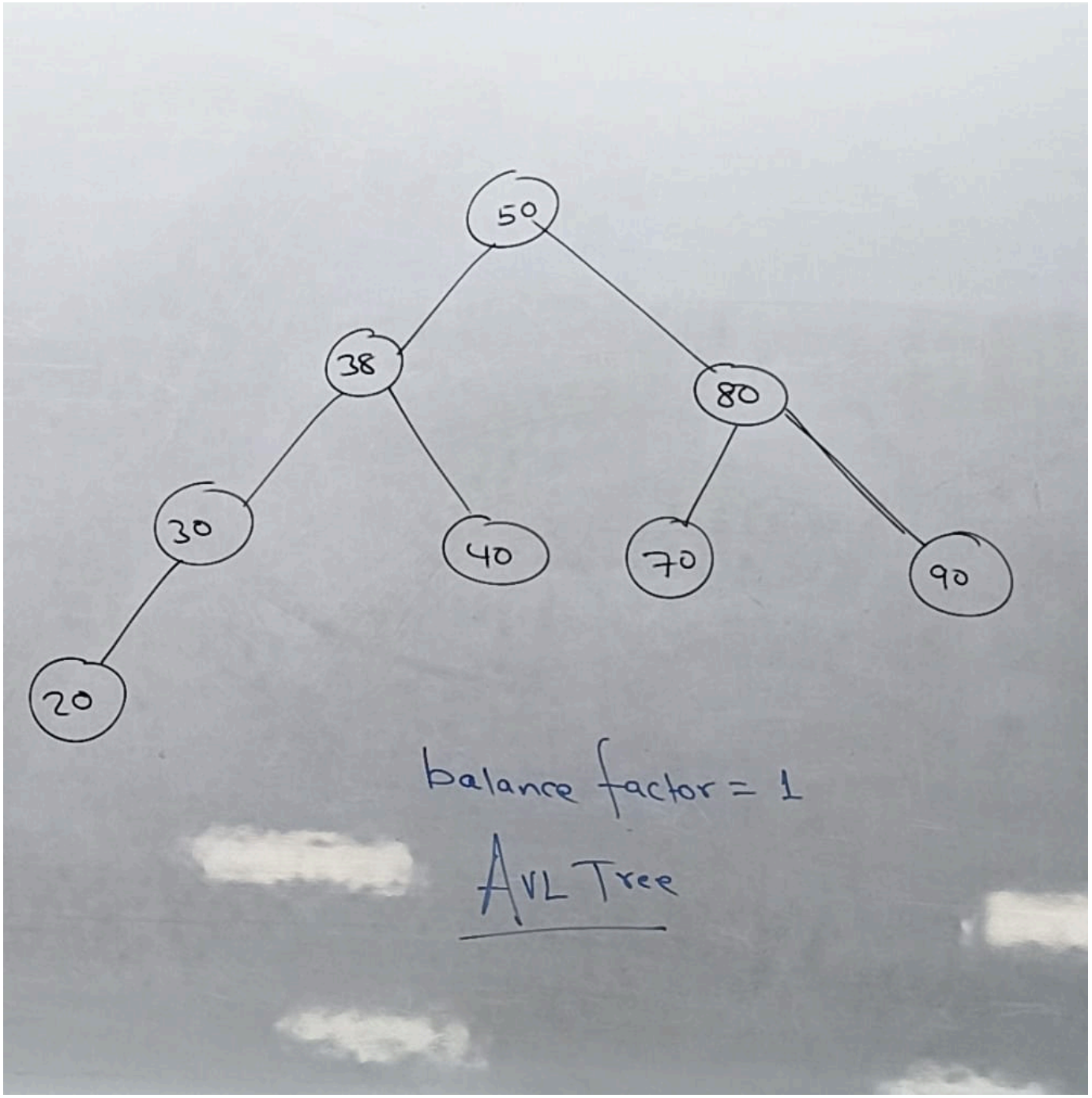
A binary search tree that automatically restructures itself whenever an insertion or deletion operation is performed.

Examples: AVL Tree, Red-Black Tree



5. AVL Tree

AVL (Adelson-Velsky-Landis) is the first self-balancing binary search tree where the balance factor is strictly maintained as `0`, `-1`, or `1`.



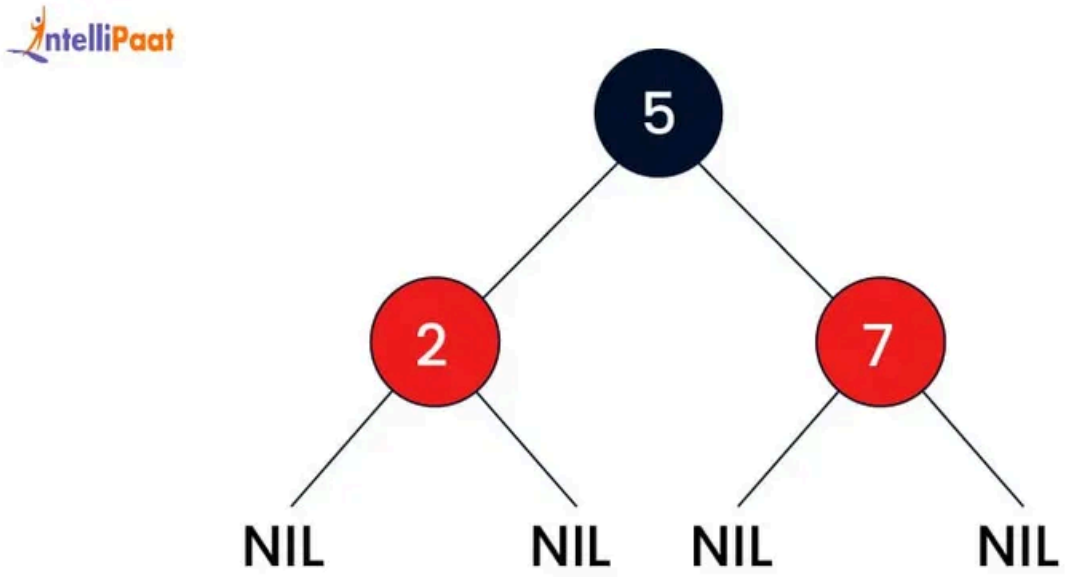
6. Red-Black Tree

A self-balancing tree that maintains strict color-coding rules.

Rules:

- Root must be **black**
- Two consecutive nodes must not be **red**
- Leaf or null nodes should be **black**
- Each path from root to leaf must have the same number of black nodes

It is highly efficient in searching, insertion, and deletion, so it is widely used in built-in tree implementations such as **TreeSet**, **TreeMap**, **HashMap**, etc.

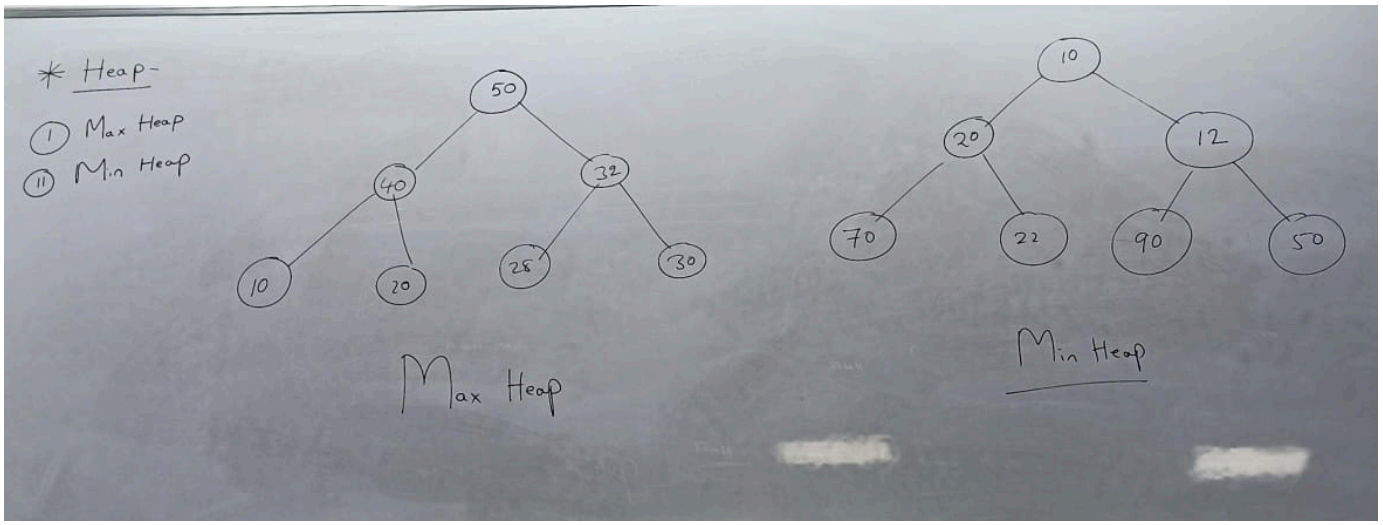


7. Heap

A heap is a structured representation of data that follows certain rules.

Types:

- **Max Heap**
  - A heap where the root is always greater than its children.
- **Min Heap**
  - A heap where the root is always smaller than its children.



Tree Representation of Array

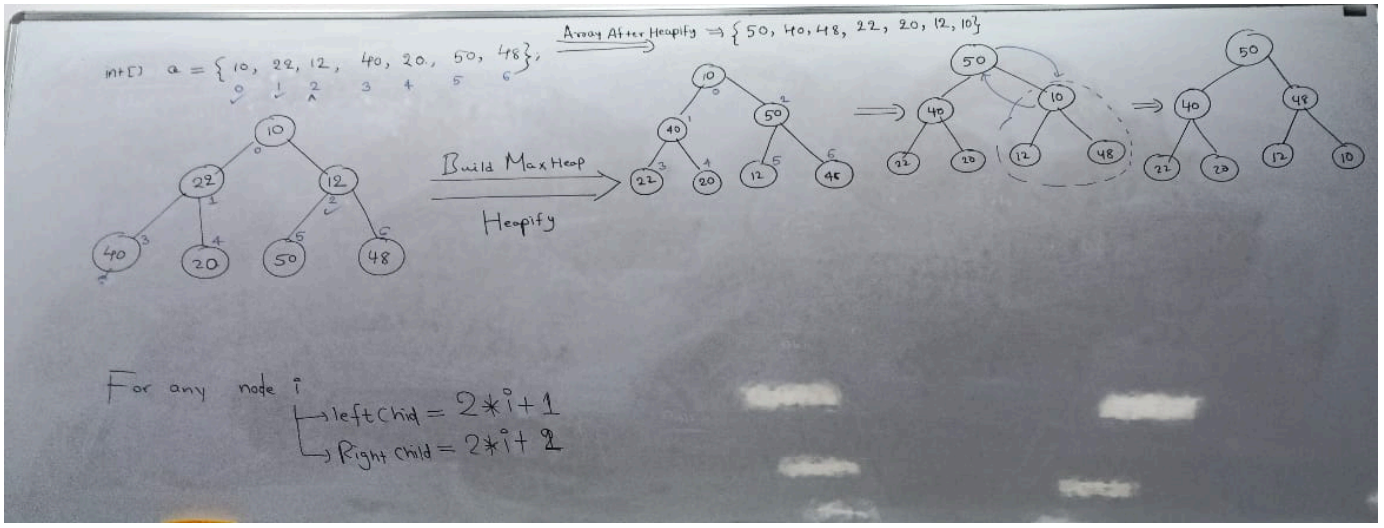
```
int[] a = {10, 20, 12, 40, 22, 55, 48, 30};
```

Heapify

The process of converting an array or tree into a **min heap** or **max heap** is called **Heapify**.

Starting point for Heapify:

```
int n = a.length;
// i (the point from where Heapify process will start) = (n/2) - 1
```



Array Index to Tree Node Mapping:

Array Index	Tree Node
i	Current Node
2i + 1	Left Child
2i + 2	Right Child