Trees – Height of Tree

1. Introduction

The **Height of a Tree** is an important property used to measure how **deep** a tree is.

It represents the **maximum number of edges (or nodes)** from the **root node to the farthest leaf node**.

Tree height is commonly used to analyze **performance, balance, and efficiency** of tree-based algorithms.

---

2. What is Height of a Tree?

The **height of a tree** is defined as:

- The **longest path from the root to any leaf node**
- Measured in **edges** (most common) or **nodes** (sometimes specified)

> Convention may vary, but
>
> **edge-based height**

---

3. Height vs Depth

Term	Meaning
Height	Longest path from root to a leaf
Depth	Distance of a node from the root

- Height applies to the **entire tree**
- Depth applies to **individual nodes**

---

4. Example of Tree Height

Tree:

```
    A
   / \
  B   C
 /
D
```

Paths:

- A → B → D (length = 2 edges)
- A → C (length = 1 edge)

👉 **Height of tree = 2**

---

5. Height of Empty Tree

- An **empty tree** has height **−1** (edge-based)
- A **single-node tree** has height **0**

---

6. Height of Tree Using Nodes

Sometimes height is defined as **number of nodes** in the longest path.

Example:

```
A → B → D
```

- Node-based height = 3
- Edge-based height = 2

👉 Always clarify which definition is being used.

---

7. Logic to Find Height of a Tree (Plain English)

1. If the tree is empty, return −1
2. Find the height of the left subtree

3. Find the height of the right subtree
4. Take the maximum of both heights
5. Add 1 to it

```
Height = 1 + max(leftHeight, rightHeight)
```

---

8. Recursive Nature of Height Calculation

- Each subtree is itself a tree
- Height calculation is naturally recursive
- Base case occurs when node is NULL

---

9. Time and Space Complexity

Aspect	Complexity
Time Complexity	O(n)
Space Complexity	O(h)

Where:

- n = number of nodes
- h = height of the tree (recursive stack)

---

10. Why Height is Important?

Tree height affects:

- Search performance
- Traversal efficiency
- Balance of the tree
- Time complexity of operations

---

11. Height in Balanced vs Skewed Trees

Tree Type	Height
Balanced Tree	O(log n)
Skewed Tree	O(n)

Lower height → Better performance.

---

12. Applications of Tree Height

- Checking tree balance
- Optimizing search trees
- Performance analysis
- Path finding
- Tree restructuring (AVL, Red-Black Trees)

---

13. Common Mistakes

- Confusing height with depth
- Mixing node-based and edge-based definitions
- Forgetting base case for empty tree

---

14. Summary

- Height measures the longest root-to-leaf path
- Usually measured in edges
- Calculated using recursion
- Time complexity is O(n)
- Critical for tree efficiency analysis
