Trees – Introduction to Trees

1. Introduction

A **Tree** is a **non-linear data structure** used to represent data in a **hierarchical structure**.

Unlike arrays, stacks, queues, or linked lists (which are linear), trees allow one element to be connected to **multiple elements**.

Trees are widely used in **real-world applications** such as file systems, databases, and hierarchical data representation.

---

2. What is a Tree?

A tree consists of:

- **Nodes** – individual elements
- **Edges** – connections between nodes

It has:

- One **root node**
- Zero or more **child nodes**
- No cycles (acyclic structure)

---

3. Basic Terminology of Trees

Term	Description
Node	An element in the tree
Root	Topmost node of the tree
Parent	Node having one or more children
Child	Node connected below a parent
Leaf	Node with no children
Edge	Connection between nodes

Term	Description
Level	Depth position of a node
Height	Longest path from root to a leaf
Subtree	Tree formed from a node and its descendants

4. Structure of a Tree (Conceptual)

```
    A   ← Root
   / \
  B   C
 / \
D   E
```

- A is the root
- B and C are children of A
- D and E are children of B
- D, E, and C are leaf nodes

5. Key Characteristics of Trees

- Hierarchical structure
- Non-linear data organization
- One root node
- Every node (except root) has exactly one parent
- No cycles are present

6. Why Trees Are Needed?

Trees are needed to:

- Represent hierarchical data

- Improve searching efficiency
- Organize data logically
- Support efficient insertions and deletions

Examples:

- Folder structure in computers
- Organization hierarchy
- HTML DOM structure

---

7. Types of Trees (Overview)

Common types of trees include:

1. Binary Tree
2. Binary Search Tree
3. AVL Tree
4. Heap
5. Trie
6. B-Tree

(Each type has different properties and use cases.)

---

8. Tree vs Linear Data Structures

Feature	Linear Structures	Tree
Structure	Sequential	Hierarchical
Access	One-to-one	One-to-many
Complexity	Simple	More complex
Example	Array, Stack	Tree

---

9. Degree of a Node

- **Degree** = Number of children a node has
- Leaf nodes have degree 0
- Root node can have any degree

---

10. Tree Traversal (Overview)

Traversal means visiting all nodes of the tree.

Common traversal methods:

- Preorder
- Inorder
- Postorder
- Level order

(Traversal will be discussed in detail later.)

---

11. Advantages of Trees

- Efficient searching and sorting
- Represents hierarchical relationships naturally
- Faster insertion and deletion than arrays
- Useful for structured data

---

12. Limitations of Trees

- Complex implementation
- Requires more memory
- Difficult to balance
- Traversal logic is complex for beginners

---

13. Applications of Trees

- File system representation
- Database indexing
- Expression parsing
- Search algorithms
- Artificial Intelligence
- Decision-making systems

---

14. Summary

- Tree is a non-linear hierarchical data structure
- Contains nodes connected by edges
- Has a single root and multiple children
- Used to represent structured data
- Widely applied in computer science
