Here are some common types of **tree data structures**:

1. **Binary Tree**:
   * Each node has at most two children.
   * It’s the basic structure from which other tree types are derived.
2. **Binary Search Tree (BST)**:
   * A binary tree where each node follows the "left child is smaller, right child is greater" property.
   * Used for efficient searching, insertion, and deletion operations.
3. **AVL Tree**:
   * A self-balancing binary search tree.
   * The height difference (balance factor) between left and right subtrees of any node is at most 1.
   * Provides O(log n) time complexity for search, insert, and delete operations.
4. **Red-Black Tree**:
   * Another self-balancing binary search tree.
   * Each node has an extra bit for determining its color (red or black) which helps maintain balance during insertion and deletion operations.
5. **Heap (Binary Heap)**:
   * A complete binary tree where the parent nodes are ordered with respect to their children.
     + **Max-Heap**: Parent nodes are greater than or equal to their children.
     + **Min-Heap**: Parent nodes are smaller than or equal to their children.
   * Used for priority queues.
6. **Trie (Prefix Tree)**:
   * A tree used for storing strings where each node represents a character of the string.
   * Efficient for tasks like autocomplete and spell-checking.
7. **B-Tree**:
   * A balanced tree used in databases and file systems.
   * Nodes can have more than two children, and the tree is balanced by keeping the number of elements in each node within a specified range.
8. **B+ Tree**:
   * A variation of the B-tree, often used in databases and file systems.
   * All leaf nodes are linked together in a linked list, which allows range queries to be faster.
9. **Segment Tree**:
   * A binary tree used for storing intervals or segments.
   * Allows efficient querying and updating of range-based data, like finding the sum of elements in a range.
10. **Fenwick Tree (Binary Indexed Tree)**:
    * A data structure that provides efficient methods for cumulative frequency tables.
    * Supports prefix sum and update queries in logarithmic time.
11. **N-ary Tree**:
    * A tree where each node can have at most **N** children.
    * Useful for representing hierarchical structures, such as file systems.
12. **Expression Tree**:
    * A binary tree used to represent expressions where internal nodes represent operators and leaf nodes represent operands.
    * Used in evaluating expressions or compiling.
13. **Cartesian Tree**:
    * A binary tree that maintains the heap property on one axis and the binary search tree property on another.
14. **Quad Tree**:
    * A tree used for partitioning a two-dimensional space by recursively subdividing it into four quadrants or regions.
    * Used in image processing, spatial indexing, and geographic information systems (GIS).
15. **Octree**:
    * A tree used for partitioning a three-dimensional space.
    * Similar to a quad tree, but divides the space into eight regions.

Each of these trees serves different purposes and has different use cases, so the choice of which to use depends on the problem at hand.