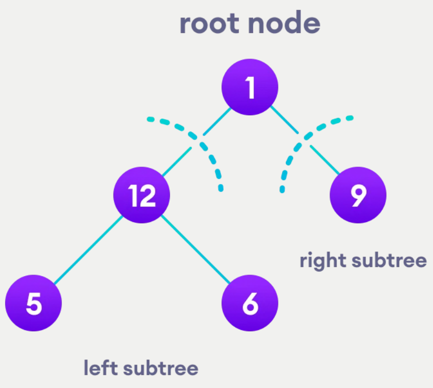
Tree

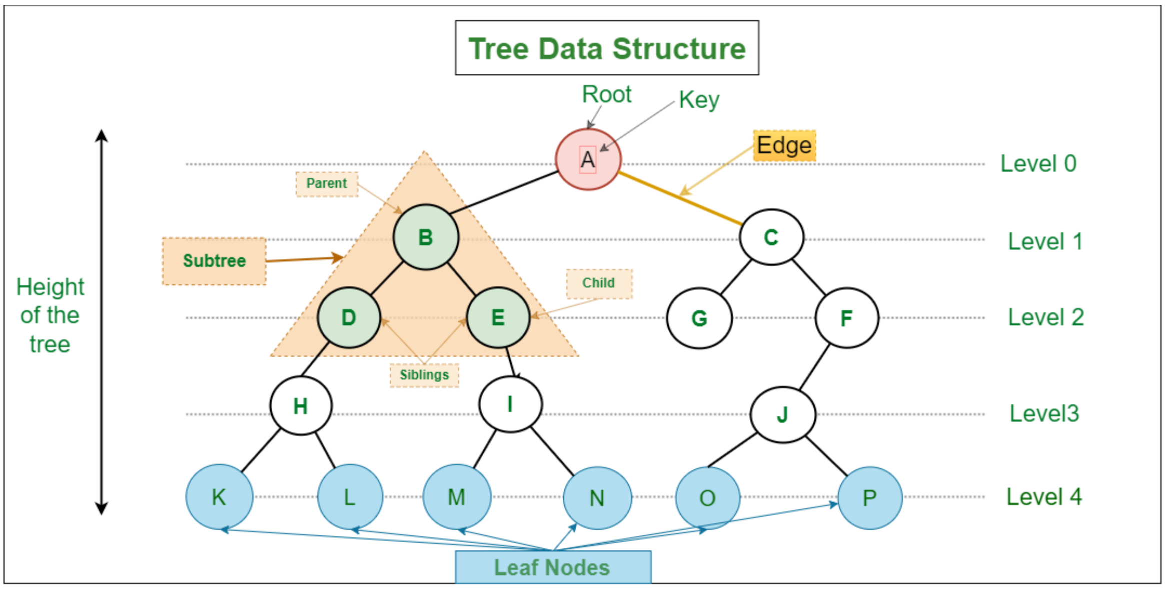
💡 *A tree data structure is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search. It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes.*

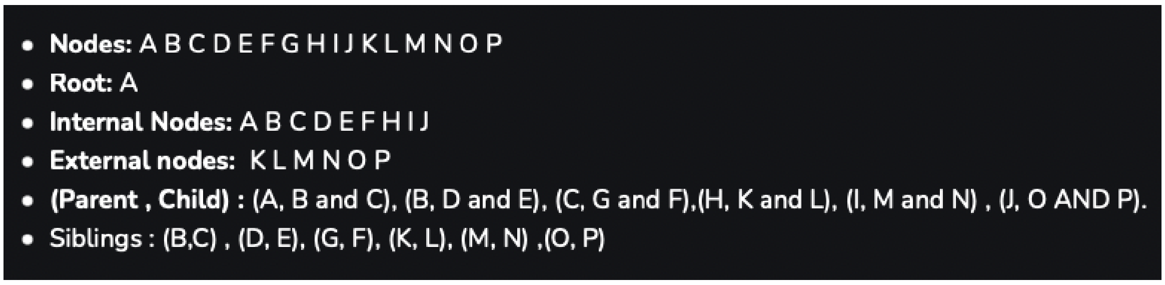


Definitions Used in Tree

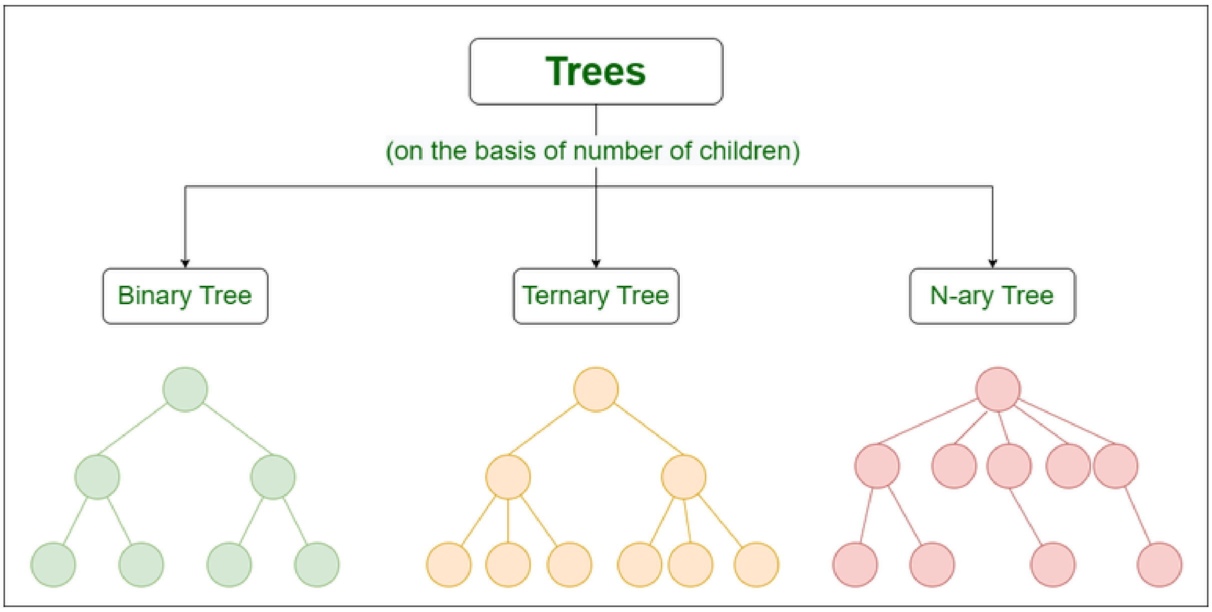
- Parent Node: The node which is a predecessor of a node is called the parent node of that node.
- Child Node: The node which is the immediate successor of a node is called the child node of that node.

- Root Node: The topmost node of a tree or the node which does not have any parent node is called the root node.
- Leaf Node or External Node: The nodes which do not have any child nodes are called leaf nodes.
- Ancestor of a Node: Any parent of given node or parent of parent nodes on the path of the root to that node are called Ancestors of that node.
- Descendant: Any successor node on the path from the leaf node to that node. {E,I} are the descendants of the node {B}.
- Level of a node: The count of edges on the path from the root node to that node. The root node has level 0.

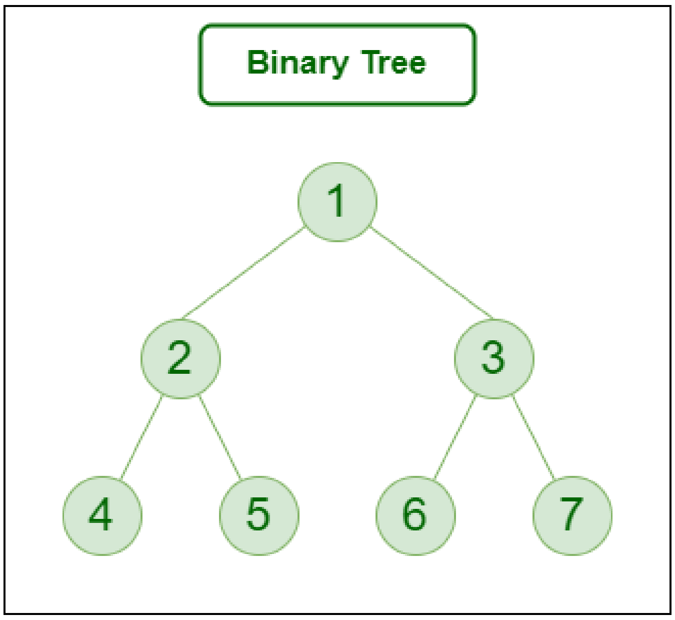



- **Nodes:** A B C D E F G H I J K L M N O P
- **Root:** A
- **Internal Nodes:** A B C D E F H I J
- **External nodes:** K L M N O P
- **(Parent , Child) :** (A, B and C), (B, D and E), (C, G and F),(H, K and L), (I, M and N) , (J, O AND P).
- Siblings : (B,C) , (D, E), (G, F), (K, L), (M, N) ,(O, P)

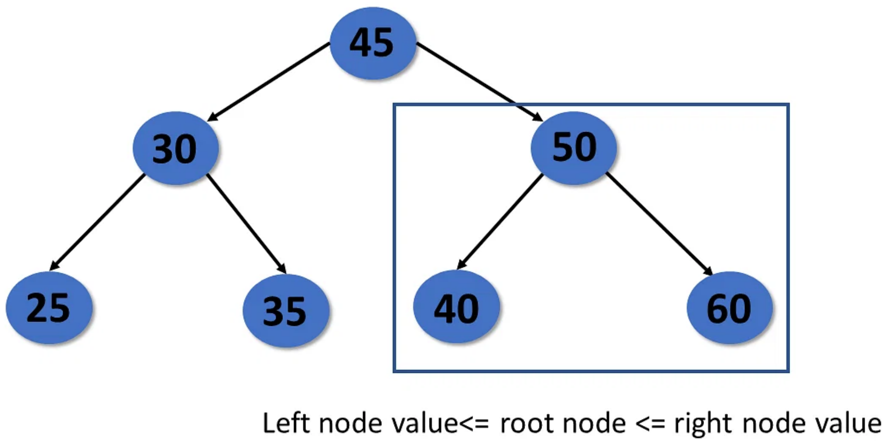
Types of Tree



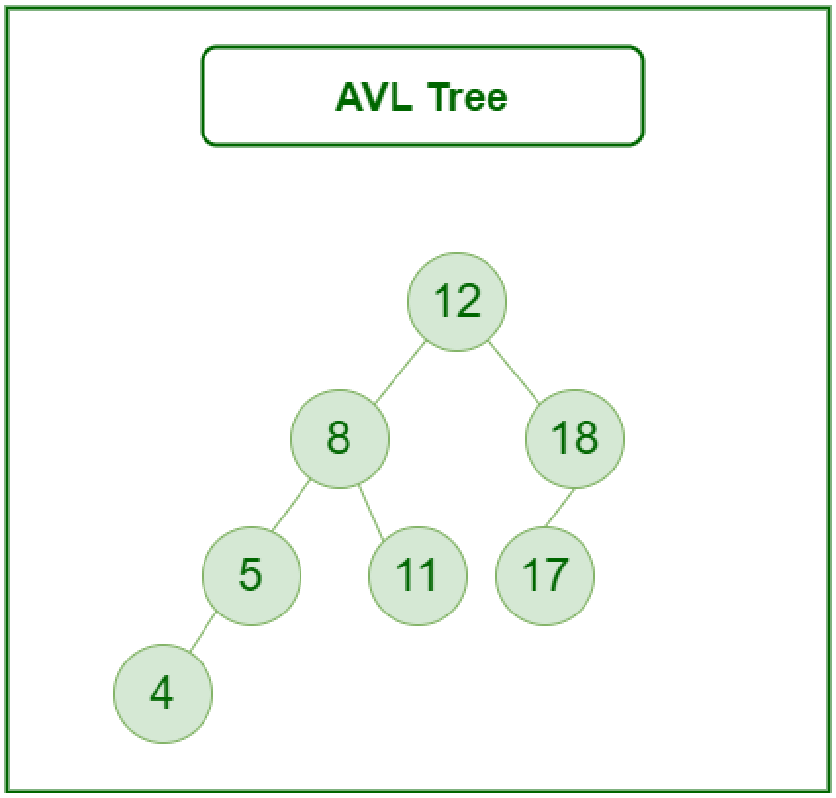
- Binary Tree: In a binary tree, each node can have a maximum of two children linked to it. W*e typically name them the left and right child.* Some common types of binary trees include full binary trees, complete binary trees, balanced binary trees, and degenerate or pathological binary trees.



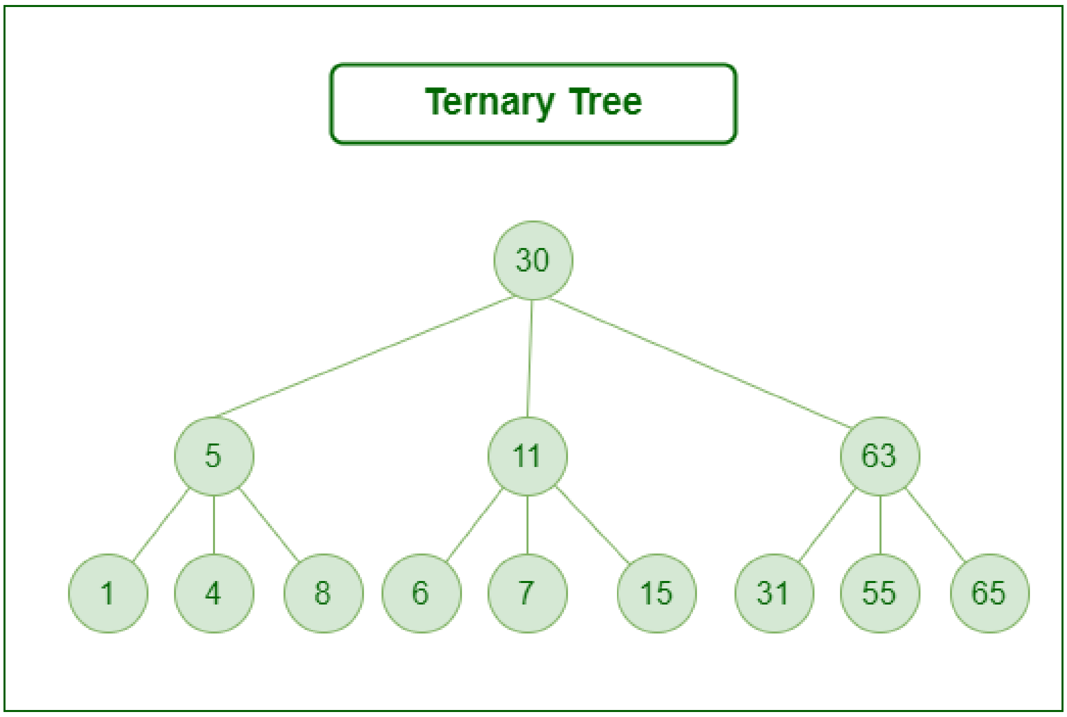
  - Binary Search Tree: A binary Search Tree is a node-based binary tree data structure with left subtree of a node contains only nodes with keys lesser than the node’s key, right subtree of a node contains only nodes with keys greater than the node’s key. The left and right subtree each must also be a binary search tree.



- AVL Tree : *It is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees for any node cannot be more than one.*

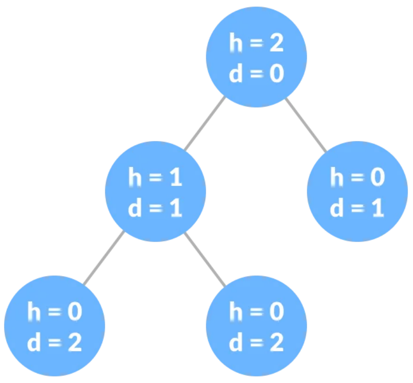


- Tenary Tree: A Ternary Tree is a tree data structure in which each node has at most three child nodes, usually distinguished as “left”, “mid” and “right”.



- N-arry or Generic Tree: Generic trees are a collection of nodes where each node is a data structure that consists of records and a list of references to its children(duplicate references are not allowed). Unlike the linked list, each node stores the address of multiple nodes. (Many children at every node, the number of nodes for each node is not known in advance)

Properties of Tree



- Number of edges: An edge can be defined as the connection between two nodes. If a tree has N nodes then it will have (N-1) edges. There is only one path from each node to any other node of the tree.
- Depth of a node: The depth of a node is defined as the length of the path from the root to that node. Each edge adds 1 unit of length to the path. So, it can also be

defined as the number of edges in the path from the root of the tree to the node.

- Height of a node: The height of a node can be defined as the length of the longest path from the node to a leaf node of the tree.
- Height of the Tree: The height of a tree is the length of the longest path from the root of the tree to a leaf node of the tree.
- Degree of a Node: The total count of subtrees attached to that node is called the degree of the node. The degree of a leaf node must be 0. The degree of a tree is the maximum degree of a node among all the nodes in the tree.

📌 **SUMMARY:** Traversing a tree means visiting every node in the tree. Tree can be traversed in different ways.

Post-Order Traversal

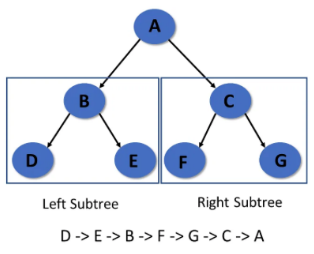
It visits the left subtree first in post-order traversal, then the right subtree, and finally the root node.

Algorithm:

Step 1- Recursively traverse the left subtree

Step 2- Visit root node

Step 3- Recursively traverse right subtree



In-Order Traversal

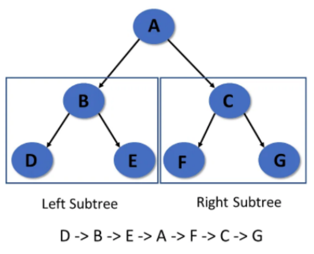
In the in-order traversal, the left subtree is visited first, then the root, and later the right subtree.

Algorithm:

Step 1- Recursively traverse the left subtree

Step 2- Visit root node

Step 3- Recursively traverse right subtree



Pre-Order Traversal

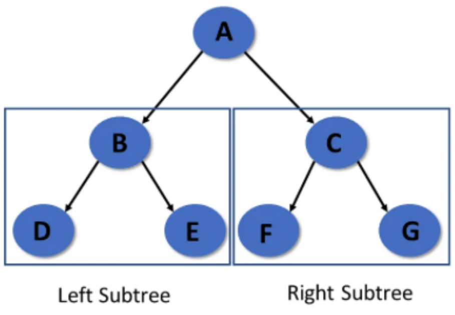
In pre-order traversal, it visits the root node first, then the left subtree, and lastly right subtree.

Algorithm:

Step 1- Visit root node

Step 2- Recursively traverse the left subtree

Step 3- Recursively traverse right subtree



A -> B -> D -> E -> C -> F -> G

Advantages

- **Efficient searching:** Trees are particularly efficient for searching and retrieving data. The time complexity of searching in a tree is typically O(log n), which means that it is very fast even for very large data sets.
- **Flexible size:** Trees can grow or shrink dynamically depending on the number of nodes that are added or removed. This makes them particularly useful for applications where the data size may change over time.

Disadvantages

- **Memory overhead**: Trees can require a significant amount of memory to store, especially if they are very large. This can be a problem for applications that have limited memory resources.
- **Imbalanced trees:** If a tree is not balanced, it can result in uneven search times. This can be a problem in applications where speed is critical.
- **Complexity**: Trees can be complex data structures, and they can be difficult to understand and implement correctly. This can be a problem for developers who are not familiar with them.
- **Limited flexibility**: While trees are flexible in terms of size and structure, they

- **Easy to traverse:** Traversing a tree is a simple operation, and it can be done in several different ways depending on the requirements of the application. This makes it easy to retrieve and process data from a tree structure.
- **Easy to maintain:** Trees are easy to maintain because they enforce a strict hierarchy and relationship between nodes. This makes it easy to add, remove, or modify nodes without affecting the rest of the tree structure.
- **Natural organization:** Trees have a natural hierarchical organization that can be used to represent many types of relationships. This makes them particularly useful for representing things like file systems, organizational structures, and taxonomies.
- **Fast insertion and deletion**: Inserting and deleting nodes in a tree can be done in O(log n) time, which means that it is very fast even for very large trees.

are not as flexible as other data structures like hash tables. This can be a problem in applications where the data size may change frequently.

- **Inefficient for certain operations:** While trees are very efficient for searching and retrieving data, they are not as efficient for other operations like sorting or grouping. For these types of operations, other data structures may be more appropriate.

Applications of Tree

1. Store hierarchical data, like folder structure, organization structure, XML/HTML data.
2. Binary Search Tree is a tree that allows fast search, insert, delete on a sorted data. It also allows finding closest item

3. Heap is a tree data structure which is implemented using arrays and used to implement priority queues.
4. Syntax Tree : Scanning, parsing , generation of code and evaluation of arithmetic expressions in Compiler design.
5. K-D Tree : A space partitioning tree used to organize points in K dimensional space.
6. Trie : Used to implement dictionaries with prefix lookup.
7. Suffix Tree : For quick pattern searching in a fixed text.
8. Spanning Trees and shortest path trees are used in routers and bridges respectively in computer networks
9. As a workflow for compositing digital images for visual effects.
10. Decision trees.
11. Organization chart of a large organization.
12. In XML parser.
13. Machine learning algorithm.
14. For indexing in database.
15. IN server like DNS (Domain Name Server)
16. In Computer Graphics.
17. To evaluate an expression.
18. In chess game to store defense moves of player.
19. In java virtual machine.
20. Tree data structures are used to organize and manage files and directories in a file system. Each file and directory is represented as a node in the tree, with parent-child relationships indicating the hierarchical structure of the file system.
21. Tree data structures are often used in parsing, such as in compilers and interpreters, to represent the structure of a program or a document.
22. Tree data structures, such as binary search trees, are commonly used to implement efficient searching and sorting algorithms.
23. Graphics and UI design
24. Tree data structures are commonly used in decision-making algorithms in artificial intelligence, such as game-playing algorithms, expert systems, and decision trees.

25. Tree data structures can be used to represent the topology of a network and to calculate routing tables for efficient data transmission.

📚 References

GeeksforGeeks | A computer science portal for geeks

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive programming/company

https://www.geeksforgeeks.org/

Programiz: Learn to Code for Free

Learn to code in Python, C/C++, Java, and other popular programming languages with our easy to follow tutorials, examples, online compiler and references.

https://www.programiz.com

Simplilearn | Online Courses - Bootcamp & Certification Platform

Simplilearn is the popular online Bootcamp & online courses learning platform that offers the industry's best PGPs, Master's, and Live Training. Start upskilling!

https://www.simplilearn.com/

✍🏼 Author → Serhat Kumas

https://www.linkedin.com/in/serhatkumas/

SerhatKumas - Overview

Computer engineering student who loves coding in different fields instead of focusing on a one spesific area. - SerhatKumas

https://github.com/SerhatKumas