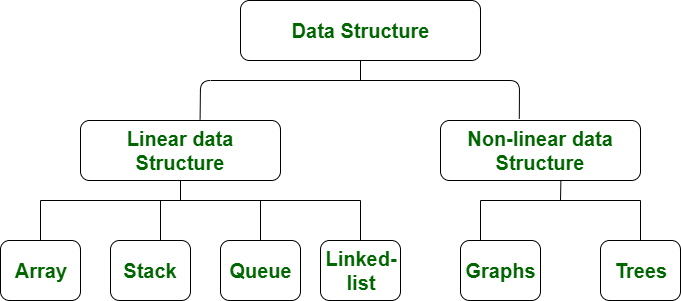
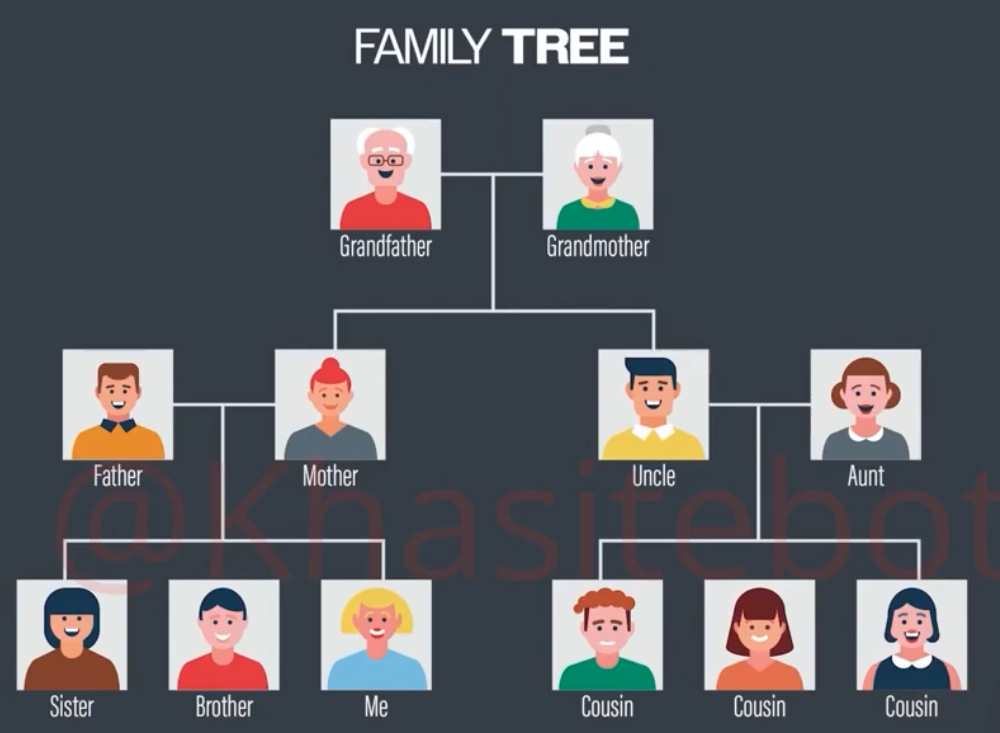
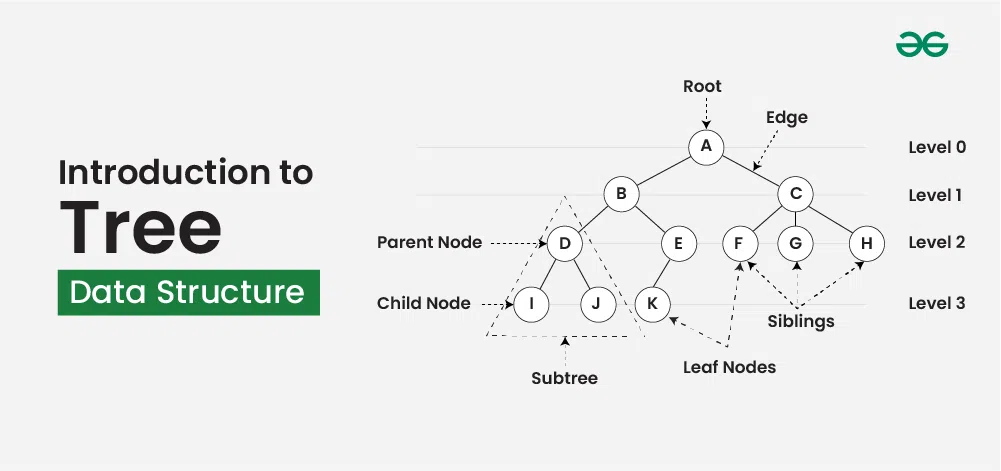
**What is Tree Data Structure?**

**Tree data structure** is a hierarchical data structure **(information/ data are stored in form of hierarchy). 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.

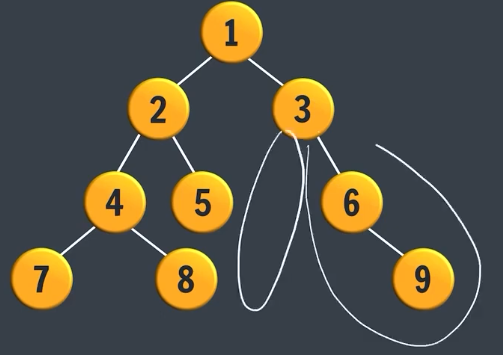
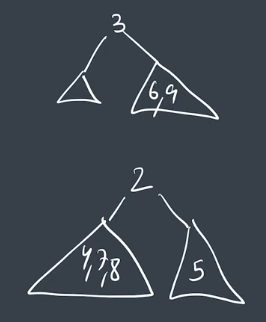
The topmost node of the tree is called the **root**, and the **nodes below it are called the child nodes**. Each node can have multiple child nodes, and these child nodes can also have their own child nodes, forming a recursive structure.  **Why Tree is considered a non-linear data structure?**

The data in a tree are not stored in a sequential manner i.e., they are not stored linearly. Instead, they are arranged on multiple levels or we can say it is a hierarchical structure. For this reason, the tree is considered to be a [non-linear data structure](https://www.geeksforgeeks.org/difference-between-linear-and-non-linear-data-structures).



 **Basic Terminologies in Tree Data Structure:**

* **Parent Node:** The node which is a predecessor of a node is called the parent node of that node.**{B}** is the parent node of **{D, E}**.
* **Child Node:** The node which is the immediate successor of a node is called the child node of that node. Examples: **{D, E}** are the child nodes of **{B}.**
* **Root Node:** The topmost node of a tree or the node which does not have any parent node is called the root node. {A**}** is the root node of the tree. A non-empty tree must contain exactly one root node and exactly one path from the root to all other nodes of the tree.
* **Leaf Node or External Node:** The nodes which do not have any child nodes are called leaf nodes. **{I, J, K, F, G, H}** are the leaf nodes of the tree.
* **Ancestor of a Node:** Any predecessor nodes on the path of the root to that node are called Ancestors of that node.**{A,B}** are the ancestor nodes of the node**{E}**
* **Descendant:** A node x is a descendant of another node y if and only if y is an ancestor of x.
* **Sibling:** Children of the same parent node are called siblings.**{D,E}** are called siblings.
* **Level of a node:** The count of edges on the path from the root node to that node. The root node has level **0**.
* **Internal node:** A node with at least one child is called Internal Node.
* **Neighbour of a Node:** Parent or child nodes of that node are called neighbors of that node.
* **Subtree**: Any node of the tree along with its descendant.

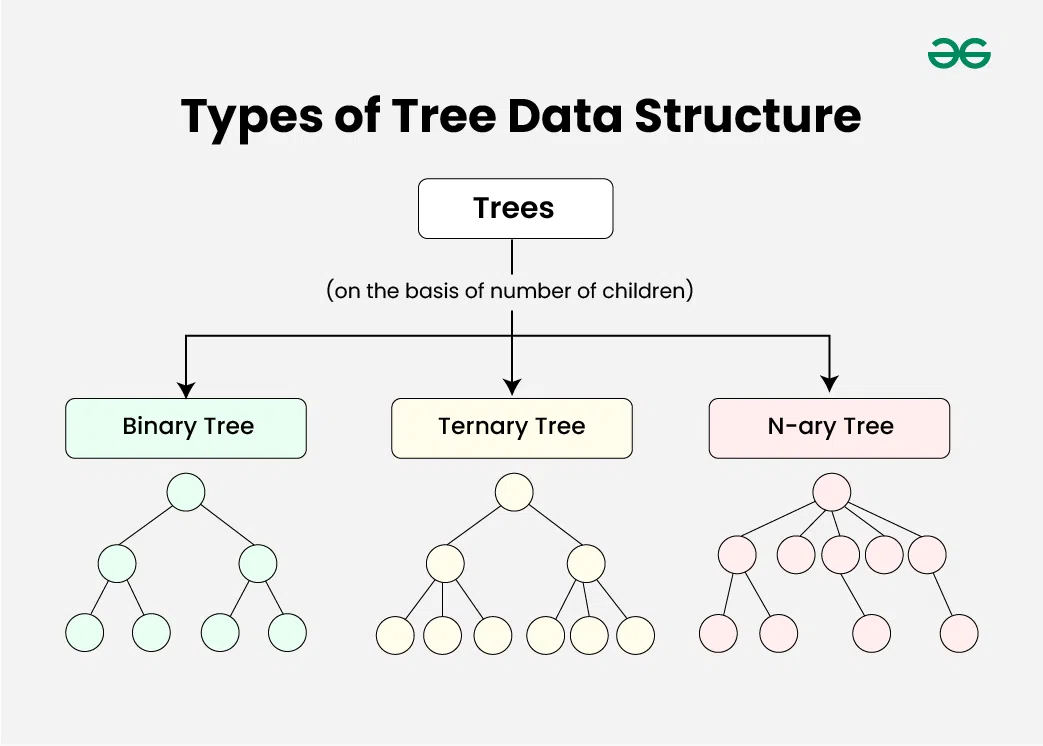
Examples above show subtrees of node 3 and 2

Leaf node have left and right subtree as null

**Note:** Total no. of children in above figure is 4 (7, 8,5,9)

Ancesors of 8 are {4, 2, 1}, 5 is not an ancestor of 8

[**Types of Tree data structures:**](https://www.geeksforgeeks.org/types-of-trees-in-data-structures)



Tree data structure can be classified into three types based upon the number of children each node of the tree can have. The types are:

* [**Binary tree**](https://www.geeksforgeeks.org/types-of-trees-in-data-structures)**:** In a binary tree, each node can have a maximum of two children linked to it. A binary search tree, also known as ordered binary tree, is a variant of binary tree in which nodes are arranged in an order.
* [**Ternary Tree**](https://www.geeksforgeeks.org/ternary-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-ary Tree or Generic Tree**](https://www.geeksforgeeks.org/generic-treesn-array-trees)

**Properties of Tree Data Structure:**

* **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 because every node except the root node is connected to its parent via an edge.
* **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** (number of edges) 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** (number of edges) from the root of the tree to a leaf node of the tree.

**The root node will be at level zero that means if the root node doesn't have any of the child nodes connected to it then the height or depth of the particular tree is said to be zero.**

* **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.

[**Applications of Tree Data Structure:**](https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-tree)

* **File System:** This allows for efficient navigation and organization of files.
* **Compiler Design:** In compiler design, a syntax tree is used to represent the structure of a program.
* **Database Indexing**: B-trees and other tree structures are used in database indexing to efficiently search for and retrieve data.

[**Advantages of Tree Data Structure:**](https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-tree)

* Tree offer **Efficient Searching** Depending on the type of tree, with average search times of O(log n) for balanced trees like AVL.
* Trees provide a hierarchical representation of data, making it**easy to organize and navigate**large amounts of information.
* The recursive nature of trees makes them **easy to traverse and manipulate** using recursive algorithms.

[**Disadvantages of Tree Data Structure:**](https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-tree)

* **Unbalanced Trees**, meaning that the height of the tree is skewed towards one side, which can lead to **inefficient search times.**
* Trees demand**more memory space requirements** than some other data structures like arrays and linked lists, especially if the tree is very large.
* The implementation and **manipulation of trees can be complex**and require a good understanding of the algorithms.

