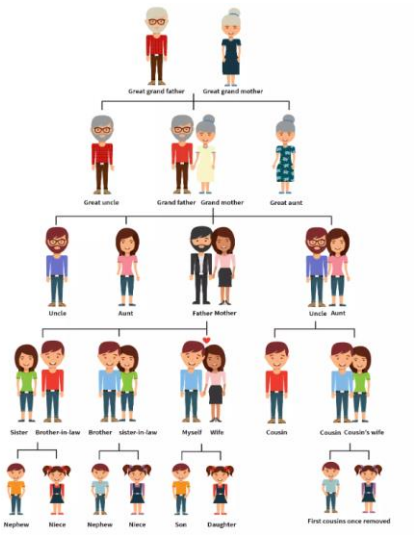
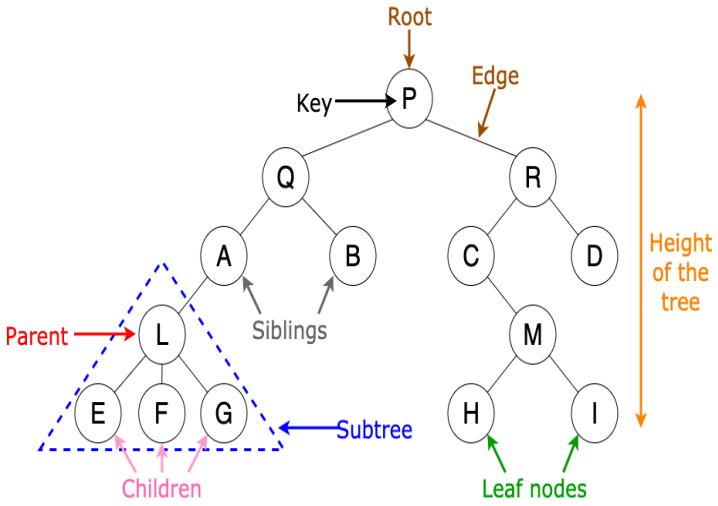
Tree Data Structure

A Tree is a Data structure in which data items are connected using references in a hierarchical manner. Each Tree consists of a root node from which we can access each element of the tree. Starting from the root node, each node contains zero or more nodes connected to it as children.





A few terminologies associated with trees in data structure are:

- **Node:** the node is an entity in a tree data structure that contains a key or a value and pointers to its child nodes.
- **Child node:** A child node is the descendant of any node.
- **Leaf nodes:** The nodes which don't have any child nodes and are the bottom-most node in a tree. They are also called the external nodes.
- **Parent node:** This node is located directly above a node. It is the principal in the hierarchy. For example, 50 is the parent node of 60, 70, and 80.
- **Root:** It is the topmost point of a tree.
- **Internal node:** The node having at least one child node.
- **Edge:** An edge refers to the connection between any two nodes in a tree.
- **Height of a node:** Number of edges from the node to the deepest leaf.
- **Depth of a node:** Number of edges from the root to the node. The root node's depth is zero. Any other node's depth is the number of edges it derives from the root node.
- **Degree of a tree:** It denotes the maximum of all the probable degrees of its nodes.
- **Number of edges:** If a tree has n nodes, the number of edges would be n-1. Excluding the root node, all other nodes in the tree possess a minimum of one incoming edge.

- **Height of a tree:** Height of the root node. It is the path length from a specific node to the extreme leaf node. It is always calculated from bottom to top. Hence, each leaf's height is 0.
- **Degree of a node:** Total number of branches to that node.
- **Forest:** A collection of disjoint trees.
- **Subtree:** It denotes children in the tree. Every child is either a leaf node or a subtree.
- **Generation:** Nodes existing at the same level of the tree are called a generation. For instance, 60, 70, and 80 belong to the same generation.
- **Ancestor:** Suppose there are two nodes, A and B, and there is a path from A to B, then A is the ancestor of B. It is common terminology used in various trees in data structure.
- **Descendent:** It is the reverse of an ancestor.
- **Sibling:** Two nodes with the same parent are called siblings. Suppose 70, 80, and 90 are siblings and 100 is the parent.
- **Subtree:** It denotes children in the tree. Every child is either a leaf node or a subtree.
- **Generation:** Nodes existing at the same level of the tree are called a generation. For instance, 60, 70, and 80 belong to the same generation.
- **Ancestor:** Suppose there are two nodes, A and B, and there is a path from A to B, then A is the ancestor of B. It is common terminology used in various trees in data structure.
- **Descendent:** It is the reverse of an ancestor.
- **Sibling:** Two nodes with the same parent are called siblings. Suppose 70, 80, and 90 are siblings and 100 is the parent.
- **Levels:** The root node exists at level zero, and its children exist at level one. Level one's children exist at level two.

Lets us discuss the few types of trees

1. **General tree:** A general tree is a tree data structure where there are no constraints on the hierarchical structure.
2. **Binary tree:** node can have at most two child nodes (children). These two child nodes are known as the *left child* and *right child*.
3. **Binary search tree:** the value of the left child of a given node should be less than or equal to the parent value and the value of the right child should be greater than or equal to the parent value.
4. **AVL tree:** Each node stores a value called a balance factor which is the difference in height between its left subtree and right subtree. All the nodes must have a balance factor of -1, 0 or 1.