

Figure

**What is tree?**

= Tree is a nonlinear data structure with hierarchical relationships between its elements without having any cycle.

**Tree Properties:**

1. Represent hierarchical data

2. Each node has two components: data and a link to it's sub category

3. Base category and sub category under it

**Why a Tree?**

1. It is nonlinear data structure, so time complexity is also nonlinear.

2. Quicker and Easier access to data

3. Store hierarchical data, like folder structure, organization structure

**Tree Terminology:**

Root: top node without parent

Edge: link between parent and child

Leaf: a node which does not have children

Sibling: children of same parent

Ancestor: parent, grandparent, great grandparent of a node

Depth of node: a length of the path from root to node

Height of node: a length of the path from the node to the deepest node

Depth of tree: depth of root node

Height of tree: Height of root node

**Types of Binary Tree:**

1. **Full Binary tree**: Each node has zero or two children.
2. **Perfect Binary tree**: All leaf nodes have two children, and they are at the same depth.
3. Complete Binary tree: In which all levels are filled, except possibly for the last level, which is filled from left to right. In other words, all nodes are as left as possible.
4. Balanced Binary tree: search google

**Tree Traversal:**

1. Pre Order: Root Node > Left Sub Tree > Right Sub Tree  
   example for fig1: N1 > N2 > N4 > N9 > N10 > N5 > N3 > N6 > N7
2. In Order: Left Sub Tree > Root Node > Right Sub Tree  
   example for fig1: N9 > N4 > N10 > N2 > N5 > N1 > N6 > N3 > N7
3. Post Order: Left Sub Tree > Right Sub Tree > Root Node  
   example for fig1: N9 > N10 > N4 > N5 > N2 > N6 > N7 > N3 > N1
4. Level Order: Level by Level, Left to Right  
   example for fig1: N1 > N2 > N3 > N4 > N5 > N6 > N7 > N9 > N10

**Python List vs Linked List:**

Linked is more useful when we don’t know the amount of tree elements.

A screen shot of a computer

Description automatically generated