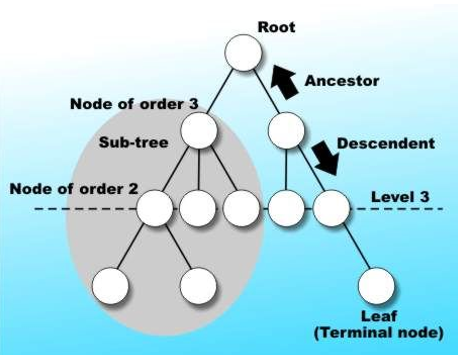
**TREES**

**Introduction**

* Are **hierarchical** in nature.
* Defined as set of **one or more** nodes.

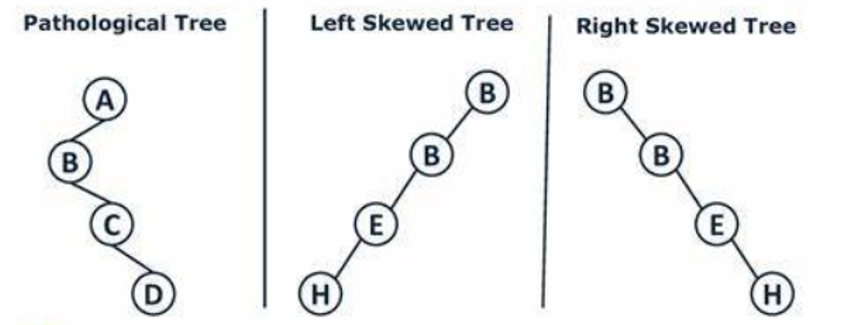
**Structure**

* Root + Sub tree = Tree
* Parent-child relation
* Nodes sharing same parents are **siblings**.
* **Internal node:** Node with at least one child.
* **Leaf node:** Node without children.



**Binary Tree**

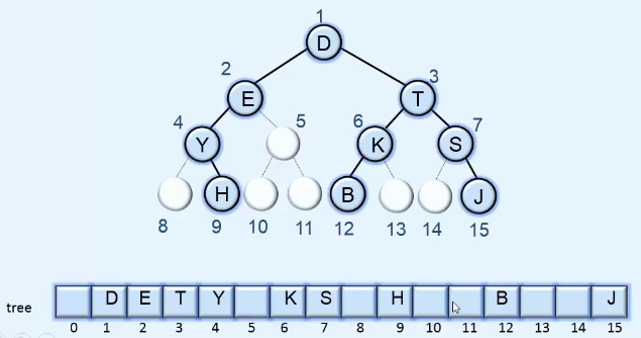
* Children – left & right
* **Usecase:** Application
* **Two types:-**
  + **Full binary tree:**
    - Each node has exactly two child, except leaf nodes.
  + **Complete binary tree:**
    - Each level is filled, except last level.
    - Last level can be filled from left to right only.
* **Perfect binary tree:** Both full and complete.
* **Degenerate (skewed) binary tree:** Each parent with exactly one child (basically linked list).



**Applications of Binary Tree**

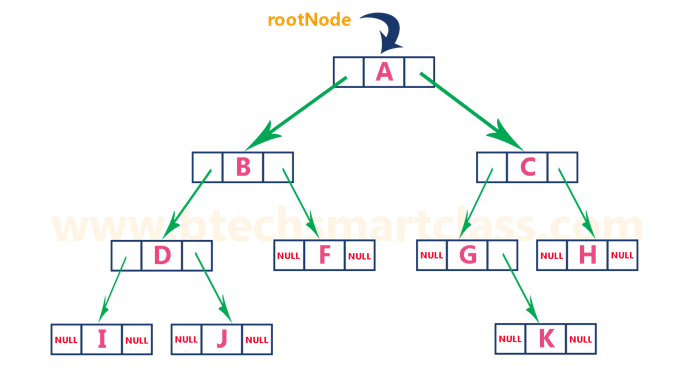
* Syntaxes in compilers
* Natural language processors

**Array Representation of Binary Trees**

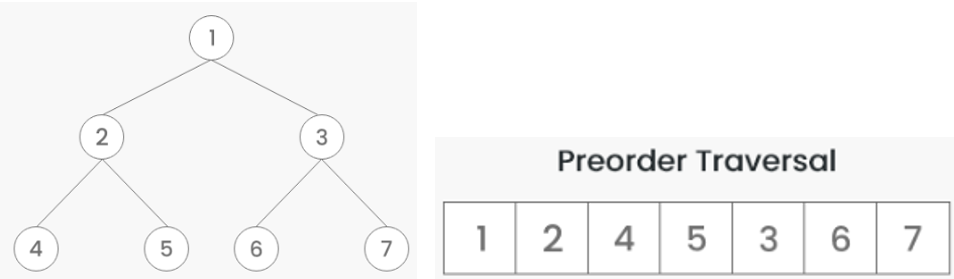


* **Formulae:**
  + If parent is at index **i**, then
    - **Left** child is at index **2i**
    - **Right** child at index **2i + 1**

**Linked List Representation of Binary Tree**



**Traversal**

* **Inorder (LNR):** Left to right traversal (BST ascending order).
* **Preorder (NLR):** 
* **Postorder (LRN):** 
* **Level order traversal (breadth-first traversal):** Travelling level by level.

Red & Black Tree

* Self-balancing tree.
* Properties:
  + Black root.
  + Red nodes can’t have red children.
  + Number of black nodes must same in each root to leaf path.
* Uses rotations and colour changes.
* Height complexity: O(log n)