Trees

Tree:

- A tree is an abstract data type that stores elements hierarchically
- Stores data nonlinearly
- The collection of nodes has parent-child relationships
- If the tree is non-empty, it has a special node called the ***root***, that has no parents
- Every other node has one parent and some children (0 or more) and it is the parent for each one of its children
- No node can be its own ancestor

Terminology:

- **Root:** The first node or top node. There can only be one root and it has no parents
- **Leaf/leaves:** Nodes that have no children. They are considered external node.
- **Internal node:** Nodes that have at least 1 child
- **Siblings:** Nodes that have the same parents
- **Ancestors:** of a node are nodes reachable by going from child to parent (including the node itself)
- **Descendants:** of a node are nodes reachable by going from parent to child (including the node itself)
- **Subtree:** A tree that consists of a node and its descendants
- **Depth:** The depth of a node is the number of edges between the root and the node
- **Height:** The number of edges in the longest path from root to leaf
- **Ordered Tree:** A tree in which there is some kind of linear ordering of a node children. Such ordering is visualized by arranging the siblings from left to right

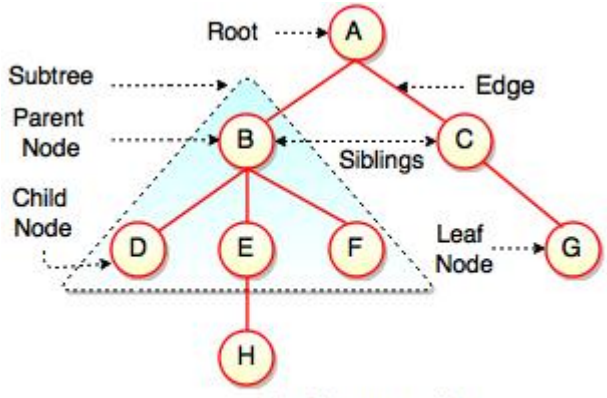
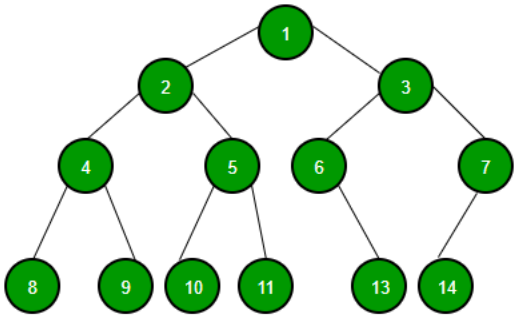


Fig. Structure of Tree

Source: https://www.tutorialride.com/data-structures/trees-in-data-structure.htm

Binary Tree:

- Ordered tree with the following properties:
  - Every node has at most two children
  - Each child is labeled as being either left child or a right child
  - A left child precedes a right child in ordering
- A proper binary tree, is a binary tree that each node has either 0 or 2 children.



Source: https://www.geeksforgeeks.org/binary-tree-data-structure/

Properties of a Binary Tree:

- In a binary tree, level 0 has at most one node, level 1 has at most 2 nodes, level 2 has at most 4 nodes.
    - In general, level d has at most 2^d nodes
- In a proper binary tree, # of leaves = # of internal node +1
    - Proof: We count the edges in the tree.
        - Every internal node has starts 2 edges, one to each of its children.
            - $E = 2*N_I$
        - Every node, except the root, ends 1 edge
            - $E = L + N_I - 1$
        - $2*N_I = L + N_I - 1$
        - $N_I = L - 1$

Tree Traversal:

- A traversal of a tree is a systematic way of accessing or visiting all the nodes of a tree.
- Two ways of traversing the tree, Depth-First (pre-order, post-order, in-order) and Breadth-First (level-order)
- **Pre-order:**
    - The root is visited first and then the subtree rooted at its children are traversed recursively.
    - If the tree is ordered, then the subtrees are traversed according to the order of the children
    - Root, Left Child, Right Child

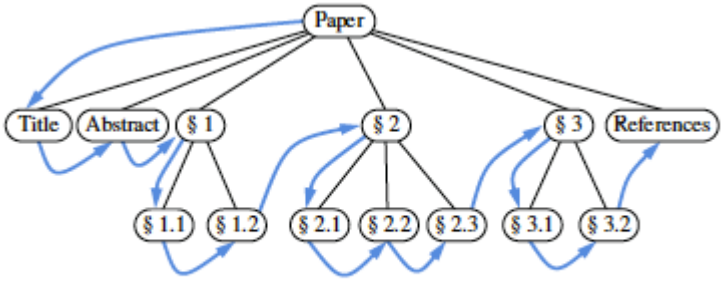


Figure 8.13: Preorder traversal of an ordered tree, where the children of each position are ordered from left to right.

- **Post-order:**
  - Recursively traverse the subtree rooted at the children first and then visits the root
  - This transversal will visit all the descendants of the root before it visits itself
  - Left Child, Right Child, Root

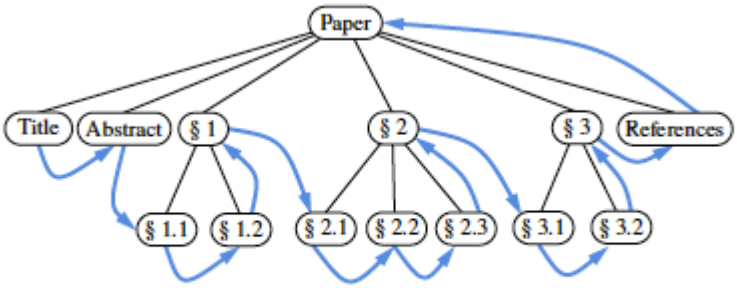


Figure 8.14: Postorder traversal of the ordered tree of Figure 8.13.

- **In-order:**
  - Recursively visit all the left subtree, root, and then recursively visit all the right subtree
  - Left Child, root, Right Child

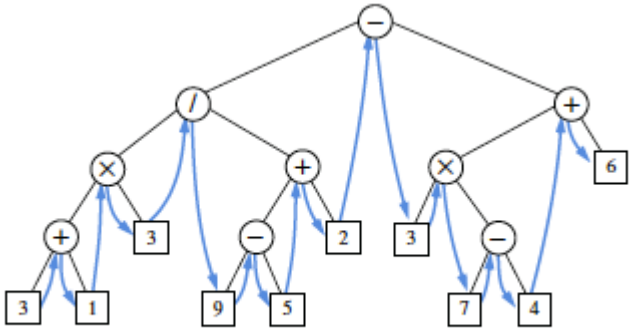


Figure 8.16: Inorder traversal of a binary tree.

- **Level-order:**
  - Visit every node, level by level
  - No recursion involved

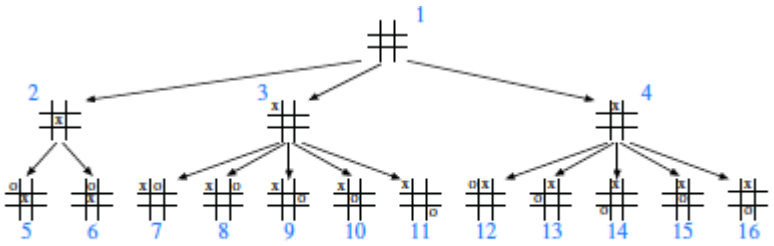


Figure 8.15: Partial game tree for Tic-Tac-Toe when ignoring symmetries; annotations denote the order in which positions are visited in a breadth-first tree traversal.