Data Structure and Algorithms – Tree

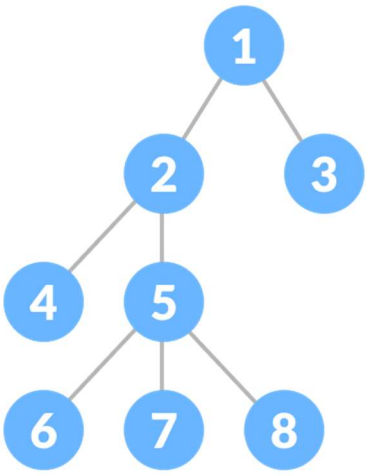
Tree represents the nodes connected by edges.

Definition

In computer science, a **tree** is a widely used abstract data type that simulates a hierarchical tree structure, with a root value and subtrees of children with a parent node, represented as a set of linked nodes.

- A tree data structure can be defined recursively as a collection of nodes (starting at a root node), where each node is a data structure consisting of a value, together with a list of references to nodes (the "children"), with the constraints that no reference is duplicated, and none points to the root.
- Alternatively, a tree can be defined abstractly as a whole (globally) as an ordered tree, with a value assigned to each node.
- Both these perspectives are useful: while a tree can be analyzed mathematically as a whole, when actually represented as a data structure it is usually represented and worked with separately by node (rather than as a set of nodes and an adjacency list of edges between nodes, as one may represent a digraph, for instance).
- For example, looking at a tree as a whole, one can talk about "the parent node" of a given node, but in general, as a data structure, a given node only contains the list of its children but does not contain a reference to its parent (if any).

A tree is a nonlinear hierarchical data structure that consists of nodes connected by edges.



A Tree

Why Tree Data Structure?

- Other data structures such as arrays, linked list, stack, and queue are linear data structures that store data sequentially.
- In order to perform any operation in a **linear data structure, the time complexity increases with the increase in the data size**.
- But, it is not acceptable in today's computational world.

Different tree data structures allow quicker and easier access to the data as it is a non-linear data structure.

Tree Terminologies

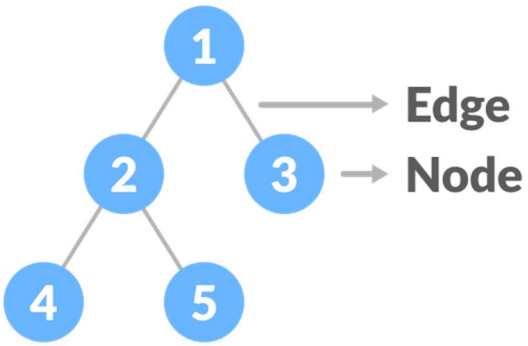
Node

- A node is an entity that contains a key or value and pointers to its child nodes.
- The last nodes of each path are called **leaf nodes or external nodes** that do not contain a link/pointer to child nodes.

- The node having at least a child node is called an **internal node**.

Edge

It is the link between any two nodes.



Nodes and edges of a tree

Root

It is the topmost node of a tree.

Height of a Node

The height of a node is the number of edges from the node to the deepest leaf (ie. the longest path from the node to a leaf node).

Depth of a Node

The depth of a node is the number of edges from the root to the node.

Height of a Tree

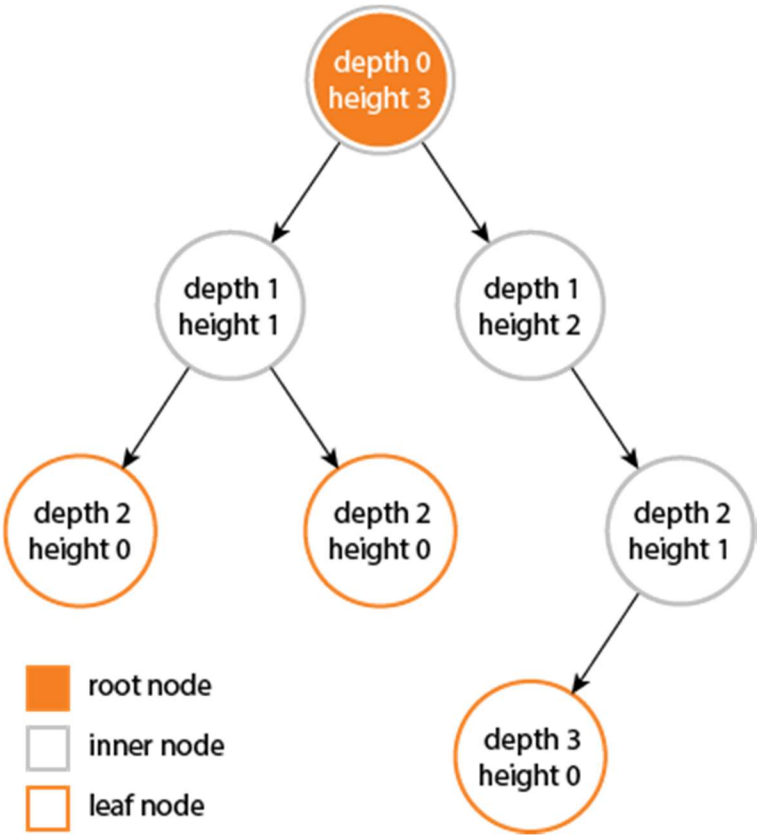
The height of a Tree is the height of the root node or the depth of the deepest node.

The depth and height are properties of a *node*:

- The **depth** of a node is the number of edges from the node to the tree's root node.
- A root node will have a depth of 0.
- The **height** of a node is the number of edges on the *longest path* from the node to a leaf.
- A leaf node will have a height of 0.

Properties of a *tree*:

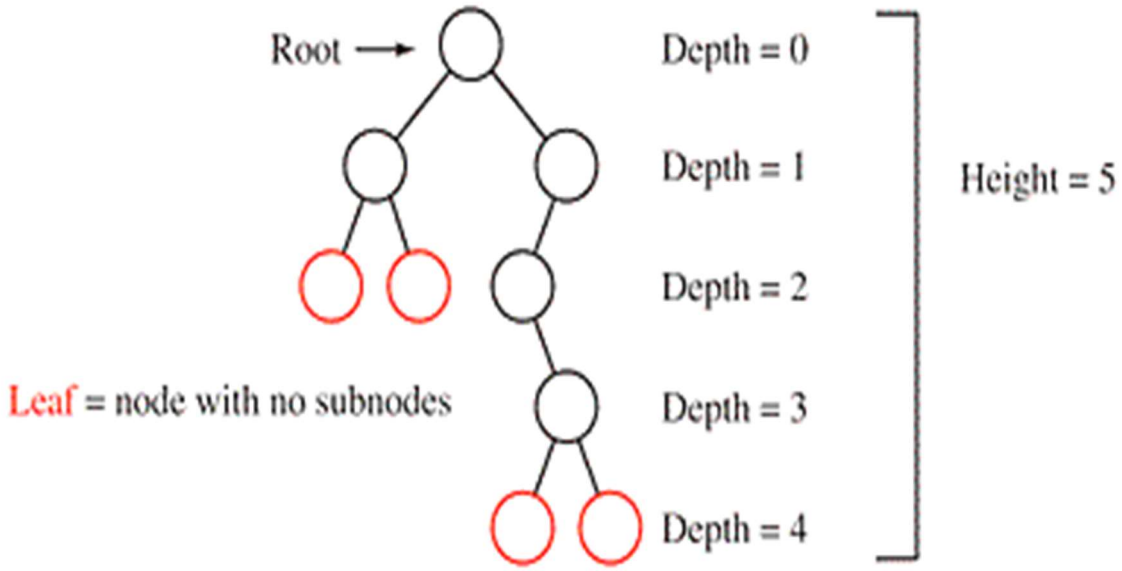
- The **height** of a tree would be the height of its root node, or equivalently, the depth of its deepest node.
- The **diameter** (or **width**) of a tree is the number of *nodes* on the longest path between any two leaf nodes. The tree below has a diameter of 6 nodes.



It means height == max depth

- 6
- The **Height (or depth) of a tree is defined to be the maximum level of any node in the tree**. Some authors define depth of a node to be the length of the longest path from the root node to that node, which yields the relation:
- **Depth of the tree = Height of the tree - 1**

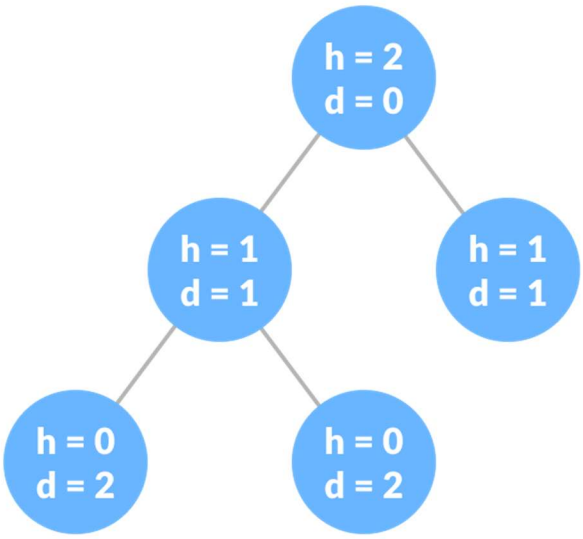
Depth:
1. **Tree**: **Number of edges/arc** f



rom the root node to the leaf node of the tree is called as the Depth of the Tree.

2. **Node**: **Number of edges/arc** from the root node to that node is called as the Depth of that node.

- **Another example:**
-
- Another way to understand those concept is as follow:
- **Depth:** Draw a **horizontal line at the root position** and treat this line as ground. So the depth of the root is 0, and all its children are grow downward so each level of nodes has the current depth + 1.
- **Height: Same horizontal line but this time the ground position** is external nodes, which is the leaf of tree and count upward.



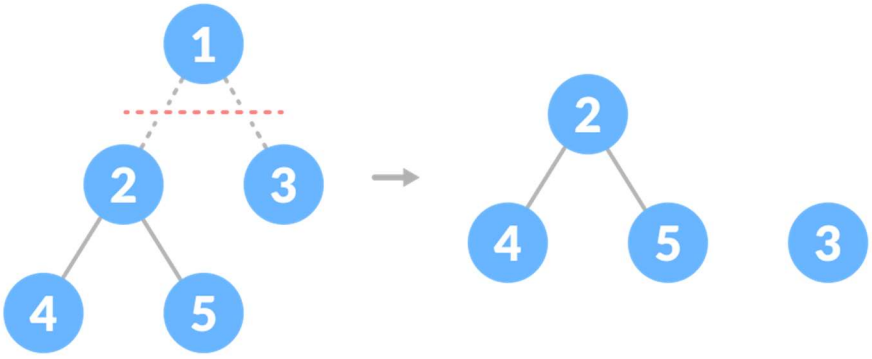
Height and depth of each node in a tree

Degree of a Node

The degree of a node is the total number of branches of that node.

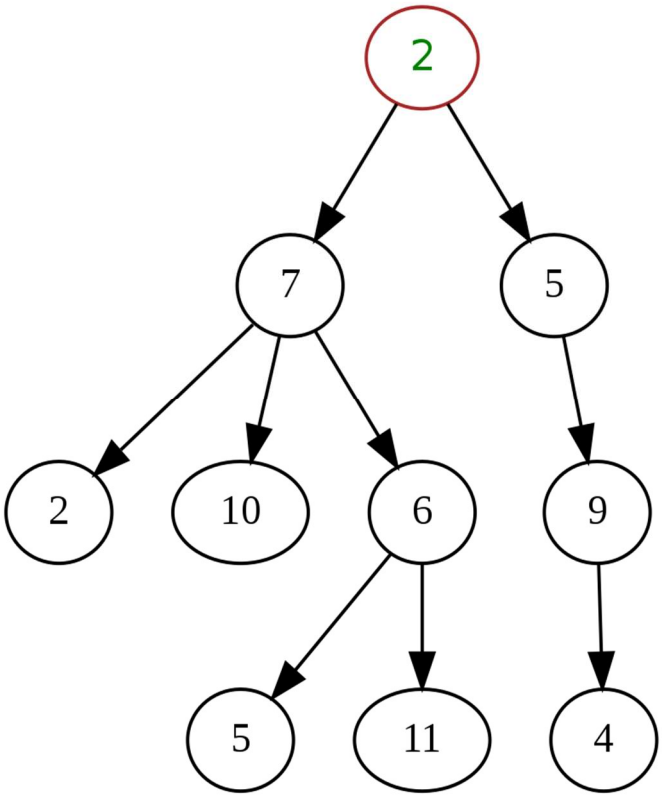
Forest

A collection of disjoint trees is called a forest.

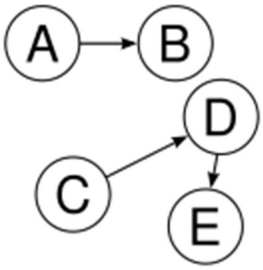


Creating forest from a tree

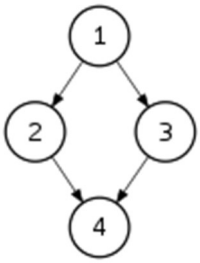
You can create a forest by cutting the root of a tree.



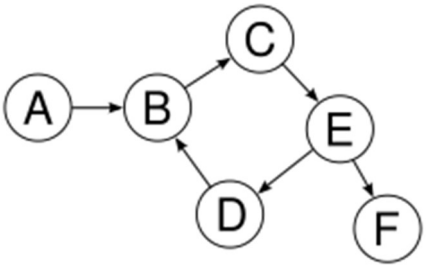
Preliminary definition



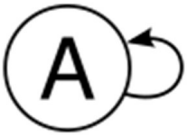
Not a tree: two non-connected parts, A→B and C→D→E. There is more than one root.



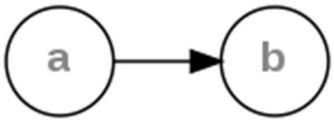
Not a tree: undirected cycle 1-2-4-3. 4 has more than one parent (inbound edge).



Not a tree: cycle B→C→E→D→B. B has more than one parent (inbound edge).



Not a tree: cycle A→A. A is the root but it also has a parent.



Each linear list is trivially a tree

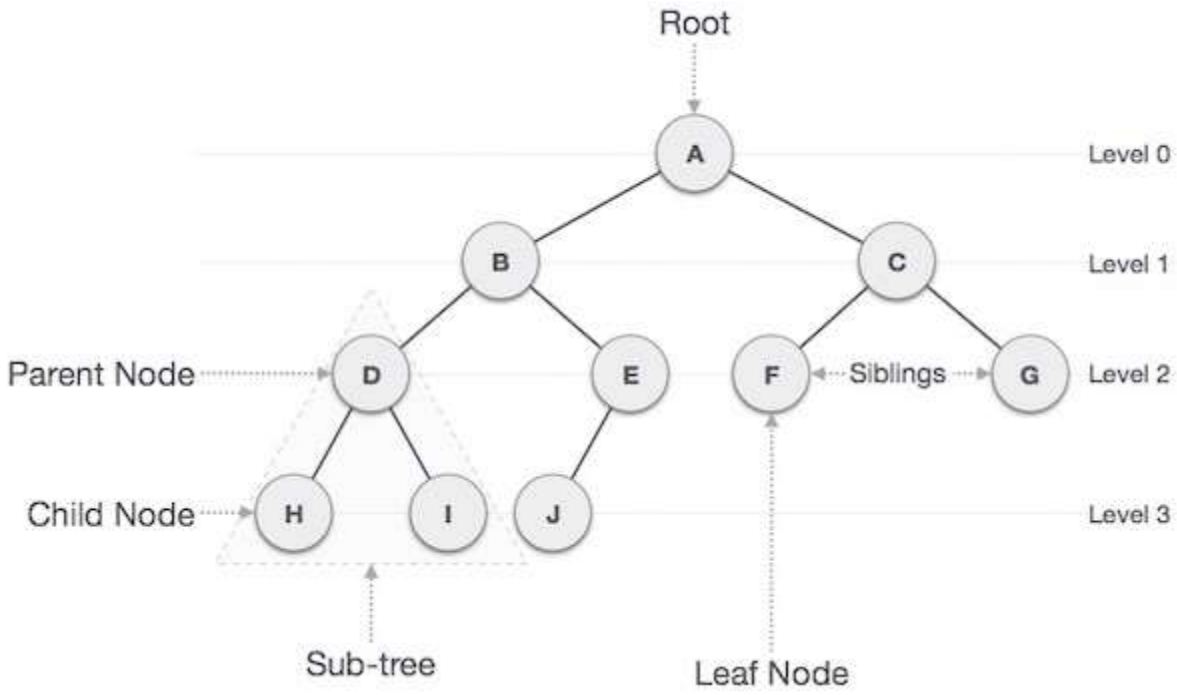
- A tree is a nonlinear data structure, compared to arrays, linked lists, stacks and queues which are linear data structures.
- A tree can be empty with no nodes or a tree is a structure consisting of one node called the root and zero or one or more subtrees.

Common uses

- **Representing hierarchical data** such as:
  - Abstract syntax trees for computer languages
  - Parse trees for human languages
  - Document Object Models of XML and HTML documents
  - JSON and YAML documents being processed
- **Search trees store data** in a way that makes an efficient search algorithm possible via tree traversal
  - A binary search tree is a type of binary tree
- **Representing sorted lists of data**
- As a workflow for compositing digital images for visual effects
- Storing Barnes-Hut trees used to simulate galaxies

We will discuss binary tree or binary search tree specifically.

- Binary Tree is a special data structure used for data storage purposes.
- A binary tree has a special condition that each node can have a maximum of two children.
- A binary tree has the benefits of both an ordered array and a linked list as search is as quick as in a sorted array and insertion or deletion operation are as fast as in linked list.



Important Terms

Following are the important terms with respect to tree.

- **Path** − Path refers to the sequence of nodes along the edges of a tree.
- **Root** − The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.
- **Parent** − Any node except the root node has one edge upward to a node called parent.
- **Child** − The node below a given node connected by its edge downward is called its child node.
- **Leaf** − The node which does not have any child node is called the leaf node.
- **Subtree** − Subtree represents the descendants of a node.
- **Visiting** − Visiting refers to checking the value of a node when control is on the node.
- **Traversing** − Traversing means passing through nodes in a specific order.

- **Levels** − Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.
- **keys** − Key represents a value of a node based on which a search operation is to be carried out for a node.

Terminology

- A **node** is a structure which may contain a value or condition, or represent a separate data structure (which could be a tree of its own).
- Each node in a tree has zero or more **child nodes**, which are below it in the tree (by convention, trees are drawn growing downwards).
- A node that has a child is called the child's **parent node** (or superior).
- A node has at most one parent, but possibly many *ancestor nodes*, such as the parent's parent. Child nodes with the same parent are **sibling nodes**.
- An **internal node** (also known as an **inner node**, **inode** for short, or **branch node**) is any node of a tree that has child nodes.
- Similarly, an **external node** (also known as an **outer node**, **leaf node**, or **terminal node**) is any node that does not have child nodes.
- The topmost node in a tree is called the **root node**. Depending on the definition, a tree may be required to have a root node (in which case all trees are non-empty), or may be allowed to be empty, in which case it does not necessarily have a root node.
- Being the topmost node, the root node will not have a parent.
- It is the node at which algorithms on the tree begin, since as a data structure, one can only pass from parents to children.
- Note that some algorithms (such as post-order depth-first search) begin at the root, but first visit leaf nodes (access the

value of leaf nodes), only visit the root last (i.e., they first access the children of the root, but only access the value of the root last).

- All other nodes can be reached from it by following **edges** or **links**.
- (In the formal definition, each such path is also unique.)
- In diagrams, the root node is conventionally drawn at the top. In some trees, such as heaps, the root node has special properties. Every node in a tree can be seen as the root node of the subtree rooted at that node.
- The **height** of a node is the length of the longest downward path to a leaf from that node.
- The height of the root is the height of the tree.
- The **depth** of a node is the length of the path to its root (i.e., its *root path*).
- This is commonly needed in the manipulation of the various self-balancing trees, AVL Trees in particular.
- The root node has depth zero, leaf nodes have height zero, and a tree with only a single node (hence both a root and leaf) has depth and height zero. Conventionally, an empty tree (tree with no nodes, if such are allowed) has height −1.
- A **subtree** of a tree *T* is a tree consisting of a node in *T* and all of its descendants in *T*.
- Nodes thus correspond to subtrees (each node corresponds to the subtree of itself and all its descendants) – the subtree corresponding to the root node is the entire tree, and each node is the root node of the subtree it determines; the subtree corresponding to any other node is called a **proper subtree** (by analogy to a proper subset).

Other terms used with trees:

Neighbor

Parent or child.

Ancestor

A node reachable by repeated proceeding from child to parent.

Descendant

A node reachable by repeated proceeding from parent to child. Also known as *subchild*.

Branch node

Internal node

A node with at least one child.

Degree

For a given node, its number of children. A leaf has necessarily degree zero.

Degree of tree

The degree of a tree is the maximum degree of a node in the tree.

Distance

The number of edges along the shortest path between two nodes.

Level

1 + the number of edges between a node and the root, i.e. (depth + 1)

Width

The number of nodes in a level.

Breadth

The number of leaves.

Forest

A set of $n \geq 0$ disjoint trees.

Ordered tree

A rooted tree in which an ordering is specified for the children of each vertex.

Size of a tree

Number of nodes in the tree.

•