Trees 🌲

Why We Need To Know

We need to know which data structure to utilize within certain use cases so that we may optimize our program's functionality as well as usability. We need to know when to use these structures and how to implement them.

Time Complexity Of Data Structures

Operation	Vector	Linked List	Deque	Tree (Unordered)	Hashtable (Unordered)
Insert front	O(n)	O(1)	O(1)	O(log(n))	NA
Insert Back	O(1)	O(1)	O(1)	O(log(n))	NA
Insert Middle	O(n)	O(1)	O(N)	O(log(n))	O(1)*
Remove Front	O(n)	O(1)	O(1)	O(log(n))	NA
Remove back	O(1)	O(1)	O(1)	O(log(n))	NA
Remove Middle	O(n)	O(1)	O(n)	O(log(n))	O(1)*
Random Access	O(1)	O(n)	O(1)	O(log(n))	NA
Search	O(n)	O(n)	O(n)	O(log(n))	O(1)*

- star is the Average Complexity
- C++ STL ordered map and set: balanced tree
- C++ STL unordered map and set: Hash table

Theory and Terminology

Tree

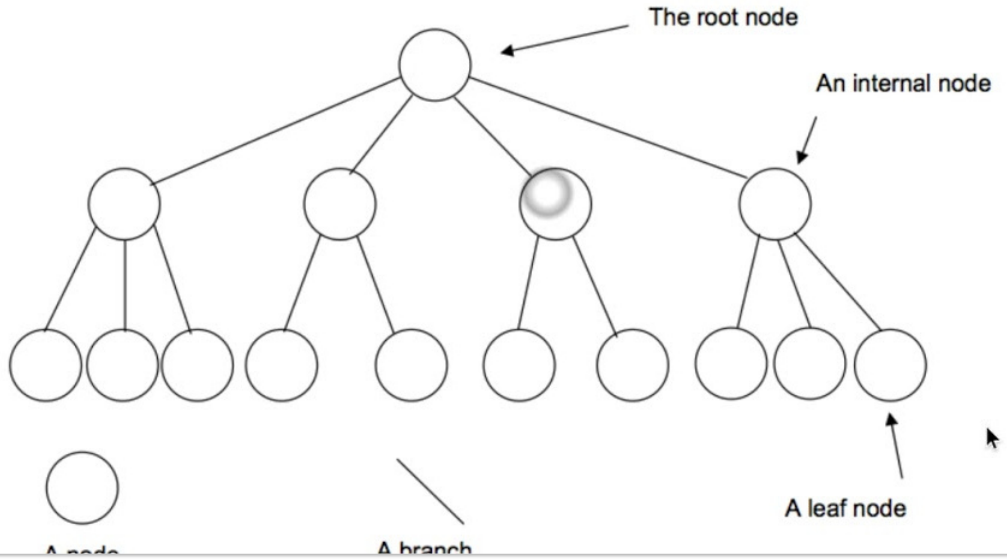
- The tree is a connected graph with no cycles (no circles) - Consequences: - Between any two vertices, there is exactly one unique path

Rooted Tree

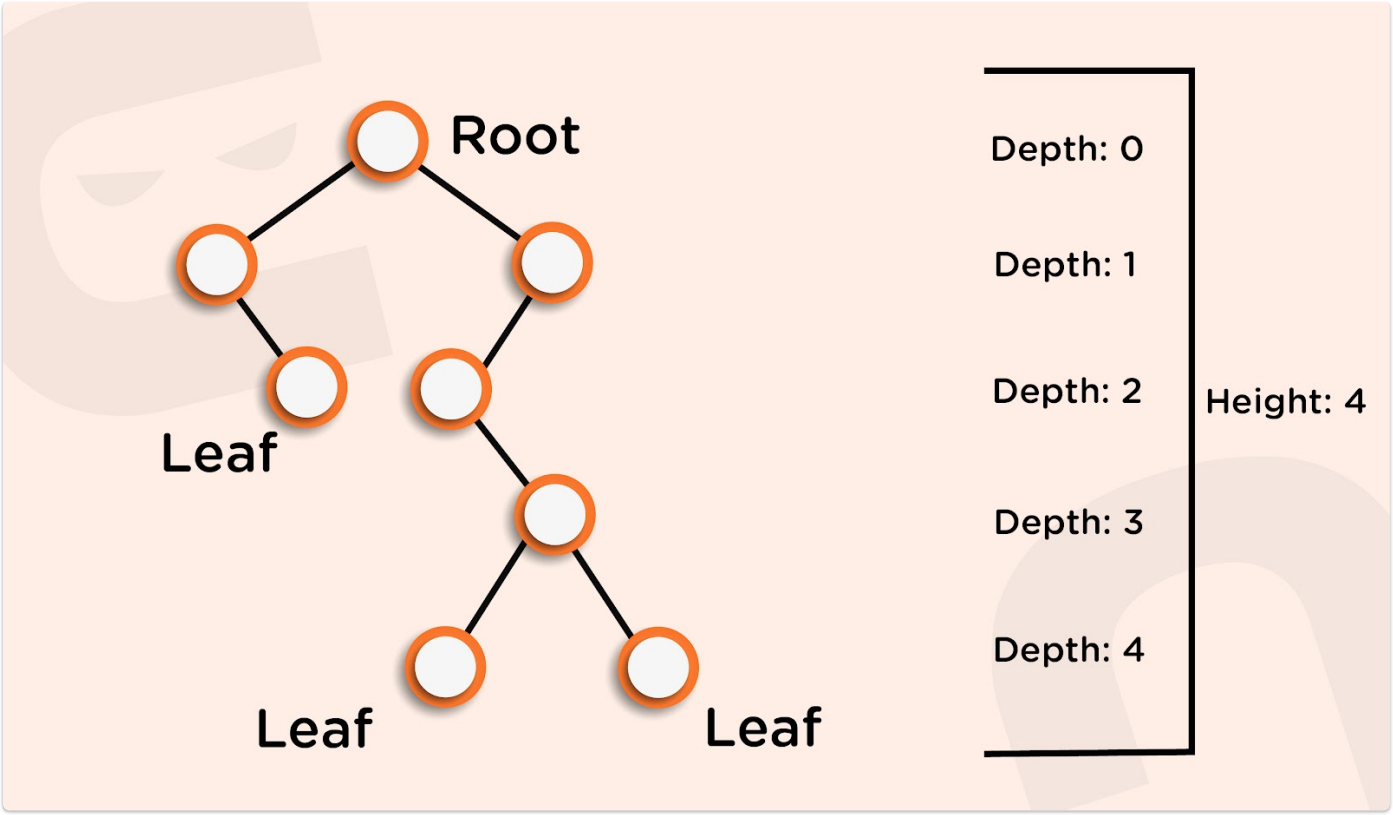
- A rooted tree - is connected - has no cycles - has exactly one vertex called the root of the tree - Consequences - Can be arranged so that the root is at the top - parent vs. child nodes and edges - sibling nodes - Nodes of the same parent nodes - leaf nodes - Nodes without children nodes

Rooted tree

A **routed tree** is a tree in which one node has been designated as the root and every edge is directed away from it.



- A unique path from the root to any vertex is a descending path
- Depth of vertex
  - Length of the unique descending path from the root to v
  - the root is at a depth 0
- The height of a vertex v is the length of the longest path from v to one of its leaves
- The height of a tree is the height of the root
  - Equal to the max depth



Rooted Tree: Recursive Definition

- A graph with N nodes and N-1 edges
- Graph has...
  - one root r
  - Zero or more non-empty subtrees

```
// Tree Node
struct TreeNode{
        Object element;
        TreeNode *firstChild;
        TreeNode *nextSibling;
}
```

Tree Traversal

- Often defined recursively
- Each kind corresponds to an iterator type
- Iterators are implemented non-recursively

Step	Description
1	Go to the root
2	Visit child subtrees

- Depth First Search
  - Begin at root
  - Visit vertex on arrival
- An implementation may be a recursive, stack-based, or nested loop
- For more information on depth-first search visit Stacks notes

Postorder Transversal

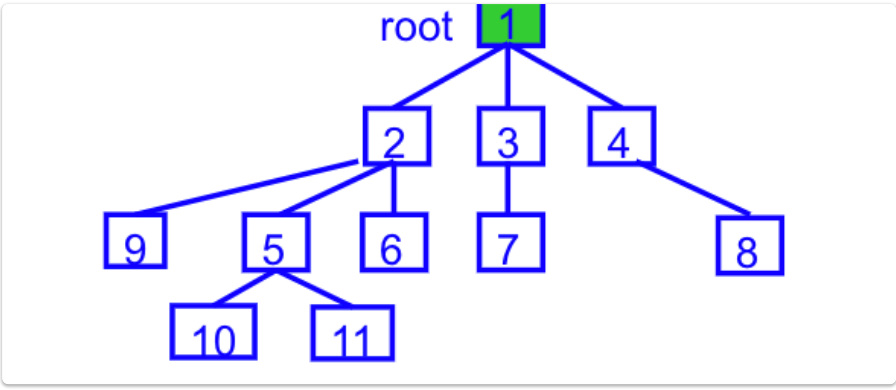
- The left subtree is traversed first
- Then the right subtree is traversed
- Finally, the root node of the subtree is traversed

Inorder Transversal

- The left subtree is traversed first
- Then the root node for that subtree is traversed
- Finally, the right subtree is traversed

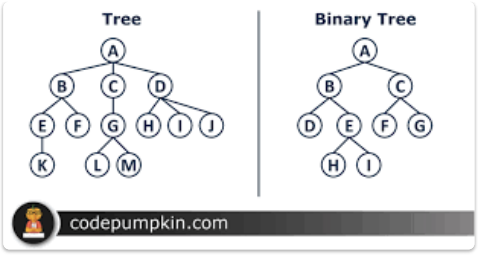
Preorder Transversal

- The root node of the subtree is visited first.
- Then the left subtree is traversed.
- At last, the right subtree is traversed.



Binary Tree

- Each node has at most two children, referred to as the left child and the right child.
- Every layer except maybe the bottom is fully populated with vertices.
- All nodes at the bottom level must occupy the leftmost spots consecutively



A complete binary tree with n vertices and h height satisfies

- $2^H \leq n < 2^{H+1}$
- $H \leq log(n) < H + 1$
- $H = floor(log(n))$