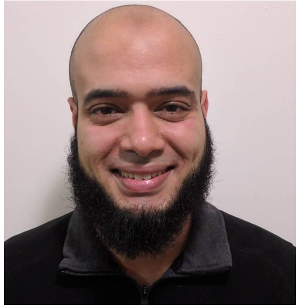
Data Structures

Trees

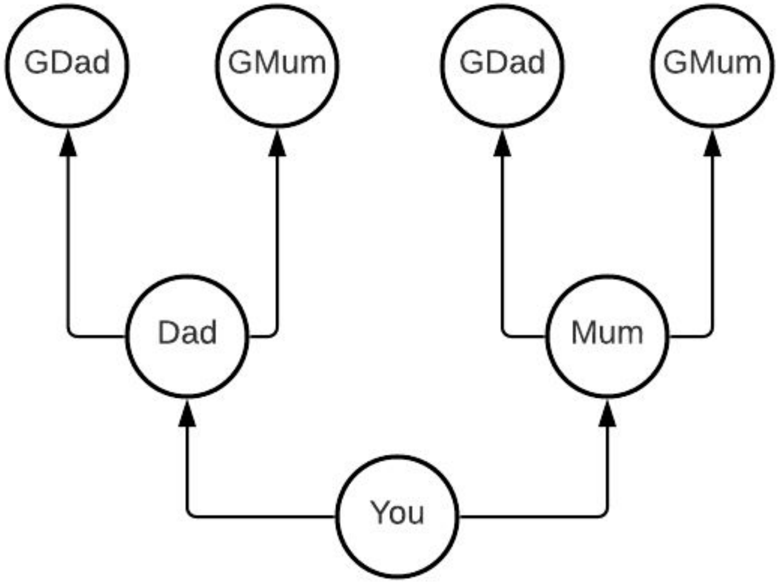
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



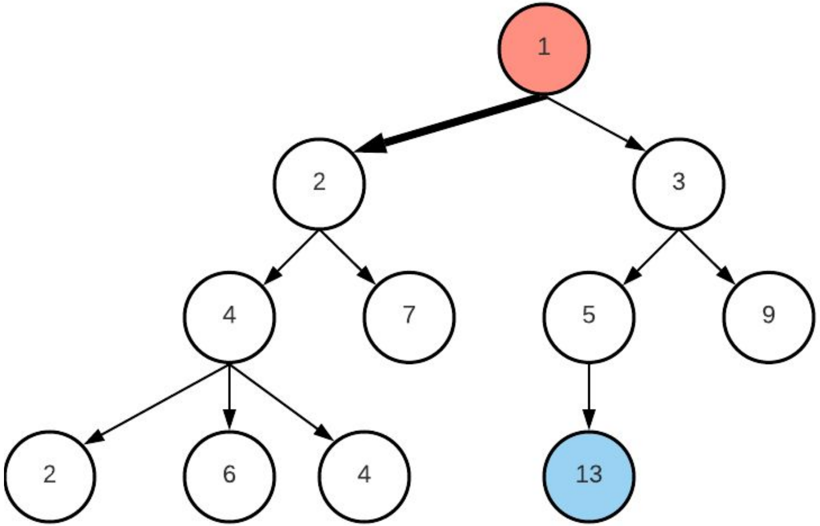
Your family tree

- Have you ever constructed your wider family tree?
- We know a tree has
  - a **Root**: **You** in this case
  - Branches (**edges**) such as **You⇒Dad**
  - **Leaves,** such as Granddad
- How could we represent this information using a computer?
  - Imagine a deep tree!
  - Recall that the stack/queue/linked list are **linear**



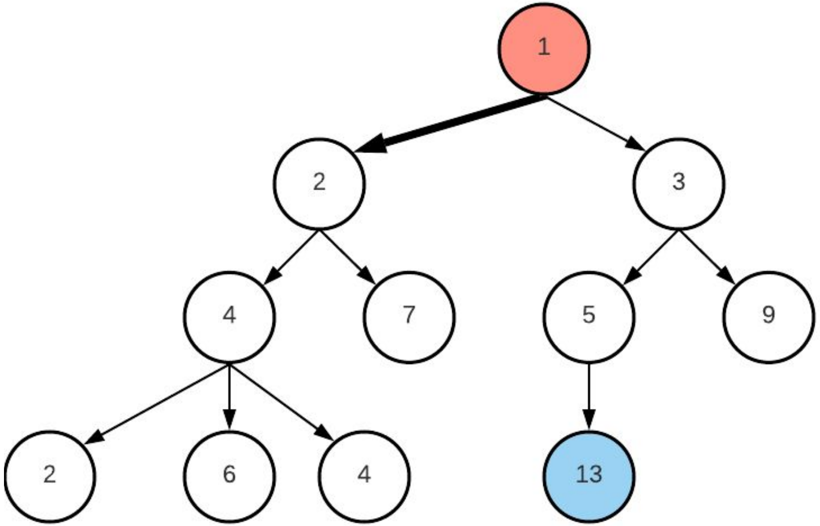
Tree Data Structure

- The tree data structure is used to represent information in an easily searchable hierarchical format
  - The tree is usually **upside-down**
- Each circle is called a node (or vertex)
  - A node may contain values (numbers, letters, strings, objects, whatever)
  - Here, the node with value (1) is called the root
  - Node(1) has 2 children: node(2) and node(3)
  - Node(4) has 3 children
  - Node(13) has no children. We call it a **leaf**
- The link between 2 nodes is an **edge**
  - Edges can sometimes have values
  - E.g. road length between 2 cities



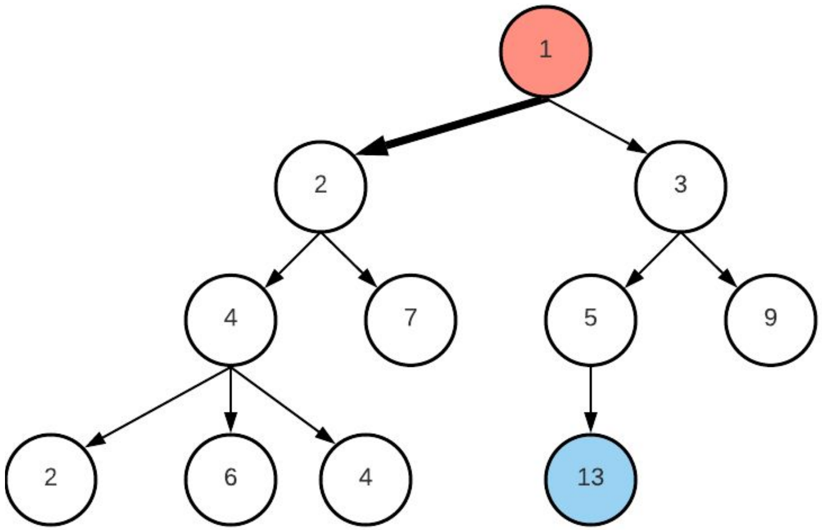
Tree Data Structure: Relations & Levels

- Node(1) has 2 **children**: 2 and 3
- The **parent** of node(7) is node(2)
- Nodes {5, 9} are siblings (brothers)
    - Same for {2, 6, 4} with common parent 4
- This tree has 4 levels:
    - **Level 0** has the nodes: 1
    - Level 1 has the nodes: 2, 3
    - Level 2 has the nodes: 4, 7, 5, 9
    - Level 3 has the nodes: 2, 6, 4, 13



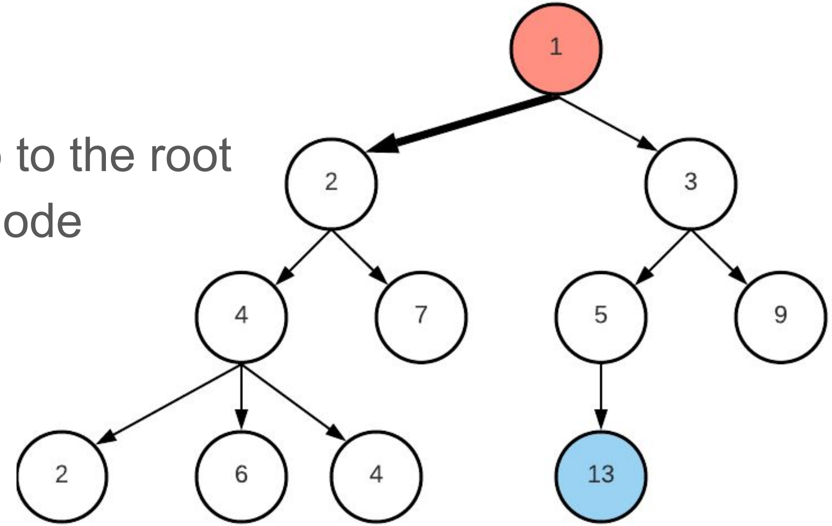
Tree Data Structure: Height

- Each tree has height: the **number of edges** on the **longest** downward path between the root and a **leaf**
    - Height of node (1) = 3
        - Longest path is: 1 ⇒ 2 ⇒ 4 ⇒ 6
    - Height of node (3) = 2
    - Height of node (4) = 1
    - Height of node (13) = 0
- Tree of N levels has N-1 height
    - We refer to it as h (**height of root**)



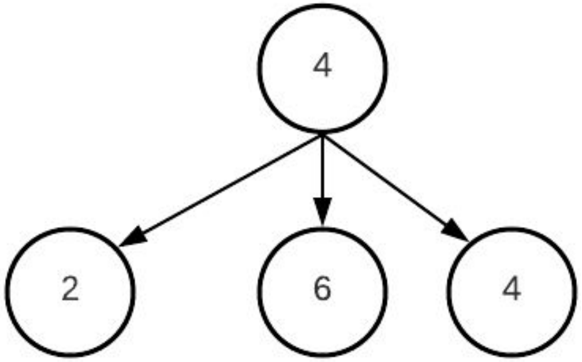
Tree Data Structure: Depth

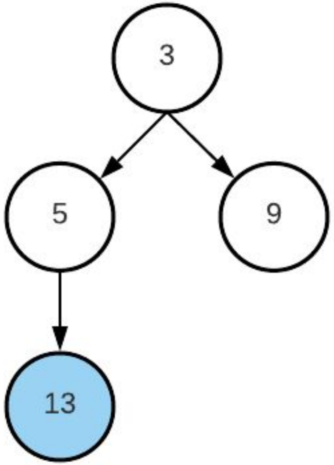
- Node’s Depth = the number of edges from the node **to the root node**.
  - Depth(root) = 0
  - Depth(4) = 2: 1 ⇒ 2 ⇒ 4 [2 edges]
  - Depth(6) = 3: 1 ⇒ 2 ⇒ 4 ⇒ 6 [3 edges]
  - Height(6) = 0
- So, depth is about going from the node up to the root
- And height is about going down to a leaf node

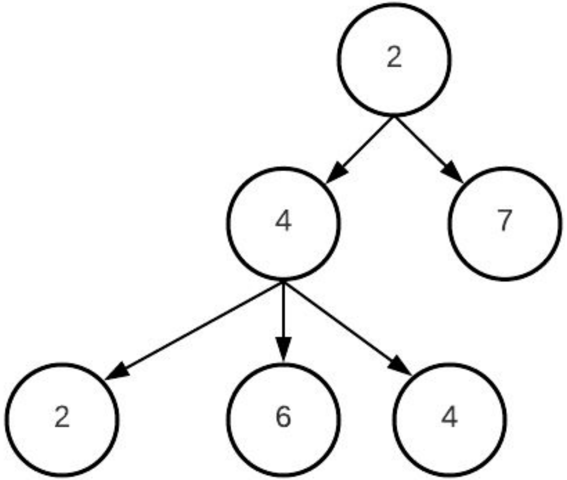


Tree Data Structure: Subtrees

- A node, along with **all** nodes below it that are linked to it, is called a subtree: e.g. in the middle tree below, we have a subtree of {5, 13}, or also a subtree of {3, 5, 9, 13}
- This is recursive in nature. That's why **recursion** is very common in trees

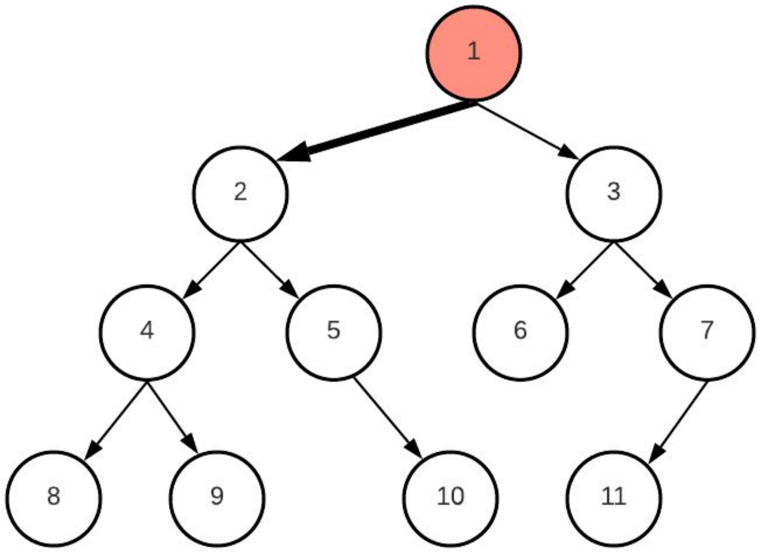






Binary Tree

- A tree in which each node has at **most two children: left** and **right** nodes
- **Left** of node(1) is node(2)
    - **Right** of node(1) is node(3)
- Left of node(4) is node(8)
    - Right of node(4) is node(9)
- Right of node(5) is node(10)
    - But it **doesn't** have a **left** node
- Left of node(7) is node(11)
    - But it **doesn't** have a **right** node
- Node (10) is a **leaf** ⇒ **No** children



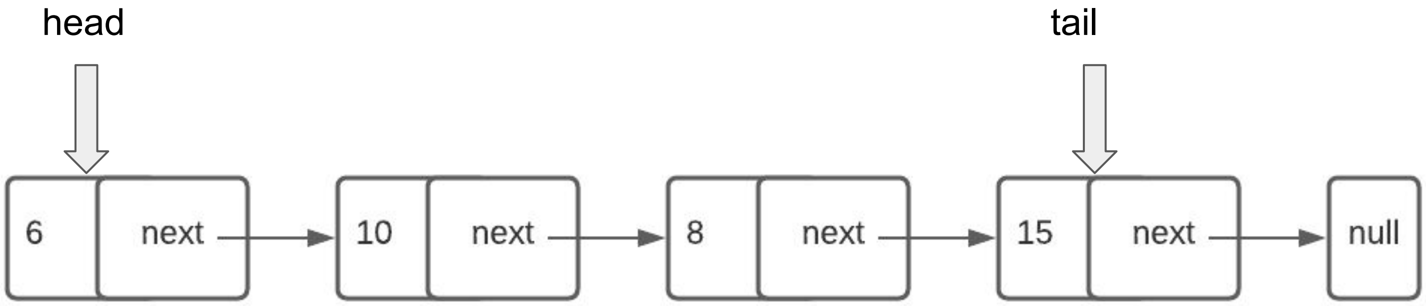
Binary Tree types

- There are many types of **binary trees**. They share some properties, but each also has its own design or characteristics
    - We will study:
    - Binary **Search** tree
    - **Balanced** Trees (AVL)
- **Trie**: General Tree

- AA tree
- AVL tree
- Binary search tree
- Binary tree
- Cartesian tree
- Conc-tree list
- Left-child right-sibling binary tree
- Order statistic tree
- Pagoda
- Randomized binary search tree
- Red–black tree
- Rope
- Scapegoat tree
- Self-balancing binary search tree
- Splay tree
- T-tree
- Tango tree
- Threaded binary tree
- Top tree
- Treap
- WAVL tree
- Weight-balanced tree

From linked list to binary tree

- You can think of a linked list as being a **special case** of binary tree
  - In an SLL, each node has a single child alone (.next),
    except the last has nothing (.next = None)
  - **Head** is the **root** node and tail is a leaf node!
- To code a binary tree, we simply extend it to have 2 children (2 next)



Your Turn: Try To Code A Binary Tree

- Similarly to a linked list, we need:
  - A struct for the node content
  - The BinaryTree class itself
- Try to design their attributes!

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."