

15CSE374  
INTRODUCTION TO DATA STRUCTURES  
AND ALGORITHMS

*Sarath tv*

# Last Lecture

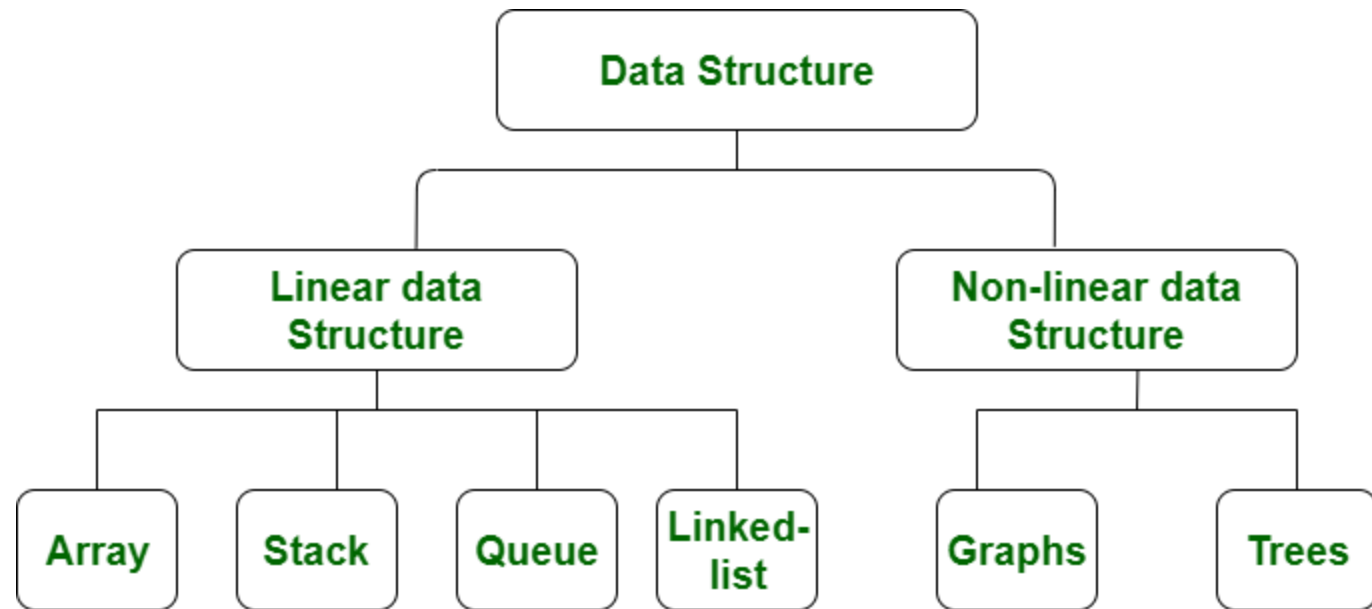
- Stack and queue using Linked List.

# Linear DS

- Data structure where data elements are arranged **sequentially** or **linearly** where the elements are attached to its **previous** and **next** adjacent in what is called a linear data structure.
- In linear data structure, **single level** is involved.
- Traverse all the elements in **single run** only.
- Its examples are **array, stack, queue, linked list**, etc.

# Non Linear DS

- Data structures where data elements are **not arranged sequentially** or linearly are called **non-linear data structures**.
- In a non-linear data structure, **single level is not** involved.
- Therefore, we **can't traverse** all the elements in **single run** only.
- Non-linear data structures are **not easy** to implement in comparison to linear data structure.
- Examples are **trees and graphs**.



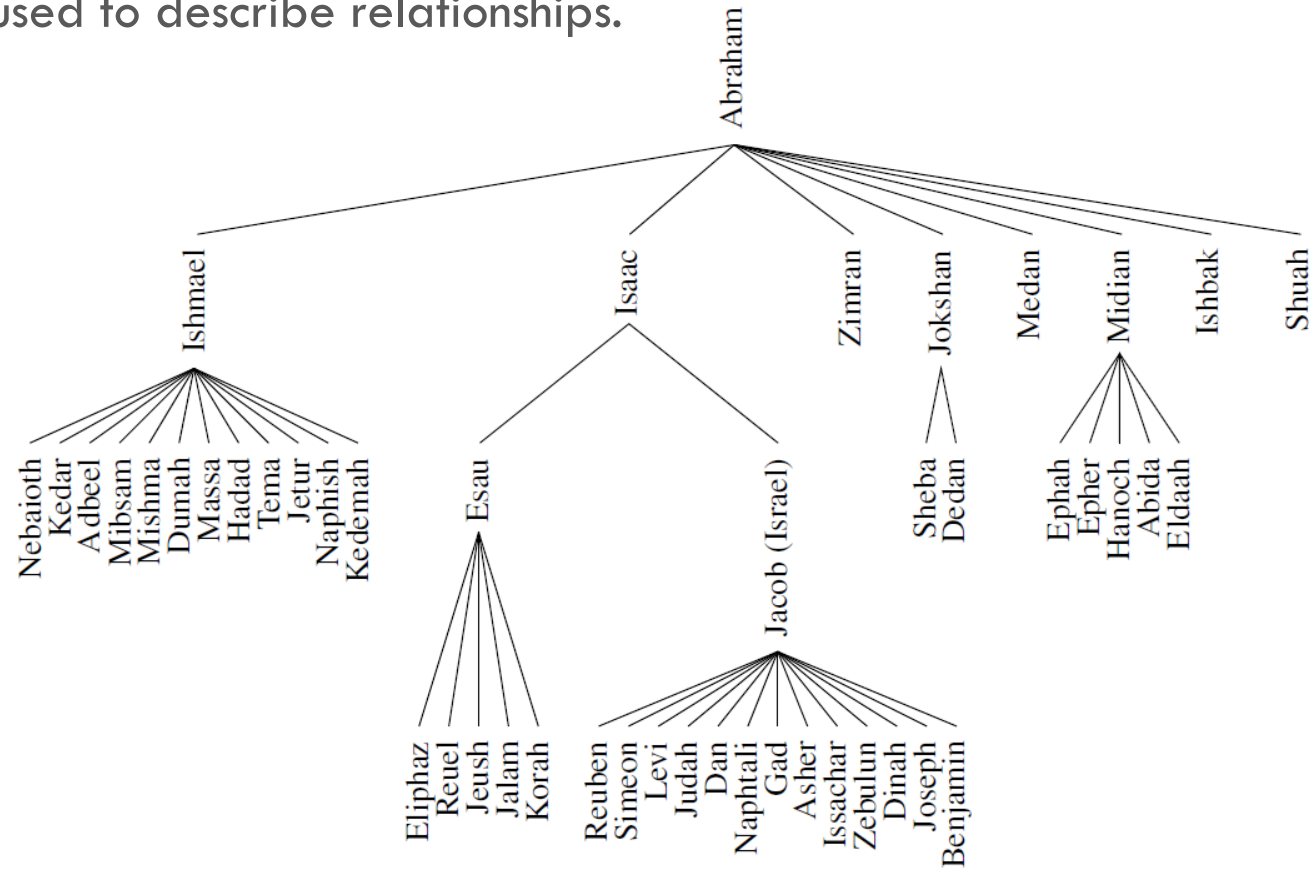
S.No	Linear data structure	Non-linear data structure
1.	Data elements are arranged in a linear order where each and every elements are attached to its previous and next adjacent.	Data elements are attached in hierarchically manner.
2.	Single level is involved.	Multiple levels are involved.
3.	Its implementation is easy	While its implementation is complex
4.	Data elements can be traversed in a single run only.	Data elements can't be traversed in a single run only.
6.	Examples are: array, stack, queue, linked list	Trees and graphs.
6.	Applications of linear data structures are mainly in application software development.	Applications of non-linear data structures are in artificial intelligence and image processing.

# Non-linear data structure

- **Non-linear data structure** does not arrange the data consecutively rather it is **arranged in sorted order**.
- In this, the data elements can be attached to more than one element **exhibiting the hierarchical relationship** which involves the relationship between the child, parent, and grandparent.
- In the non-linear data structure, the traversal of data elements and insertion or deletion are not done sequentially.
- A **tree** data structure organizes and stores the data elements in a **hierarchical relationship**.
- A **data elements** of the non linear data structure could be **connected** to **more than one elements** to reflect a **special relationship** among them.

# Tree – Basic Representation

- A tree is collection of nodes where these nodes are arranged hierarchically and **form a parent child relationships**.
- An organizational relationship that is richer than the simple “before” and “after” relationships between objects in sequences.
- The main terminology for tree data structures comes from family trees, with the terms “**parent,**” “**child,**” “**ancestor,**” and “**descendant**” being the most common words used to describe relationships.





# Definitions

- A tree is an abstract data type that stores elements hierarchically.
- With the exception of the top element,
  - *each element in a tree has a parent element and zero or more children elements.*
- Call the top element the **root** of the tree,
  - it is drawn as the highest element, with the other elements being connected below
- Two nodes that are children of the same parent are *siblings*.
- **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**

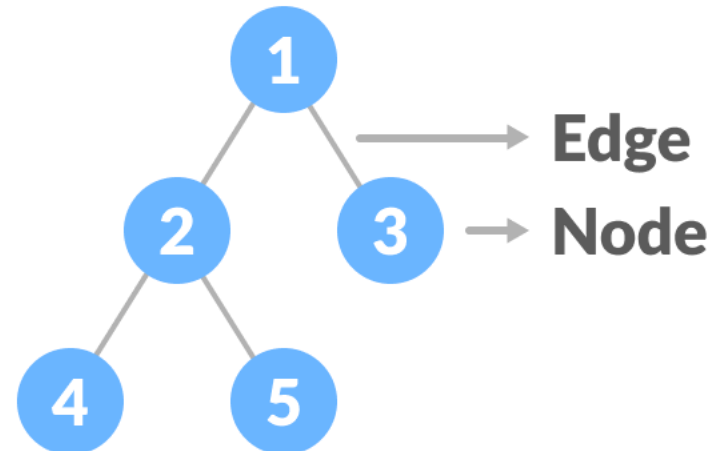
- It is the link between any two nodes

- **Path**

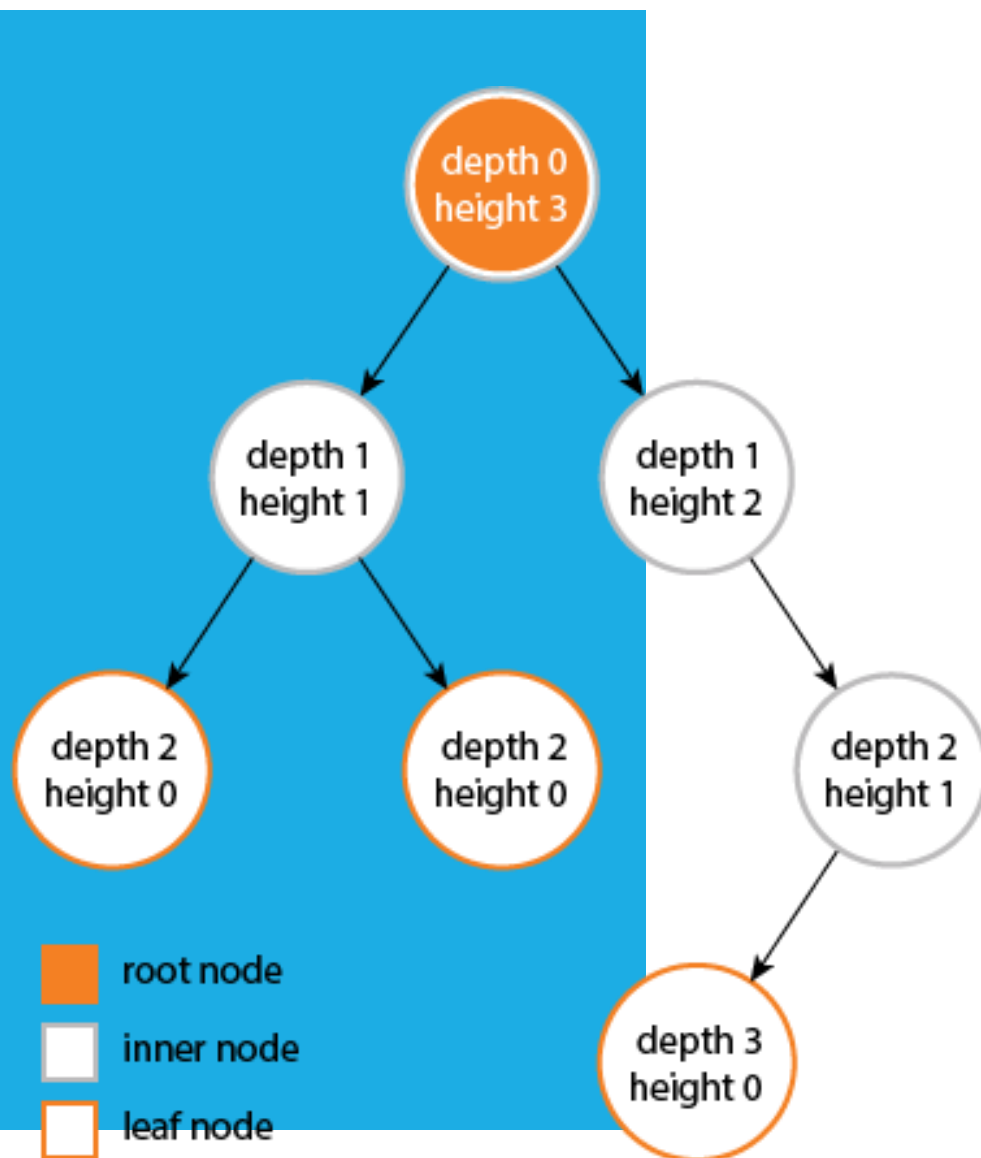
- a sequence of nodes such that any two consecutive nodes in the sequence form an edge

- **Leaves**

- The last nodes on a tree. They are nodes without children.



- **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.
- **Degree of a Node**
  - The degree of a node is the total number of branches of that node.
  - Equal to the number of children that a node has.



- 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.

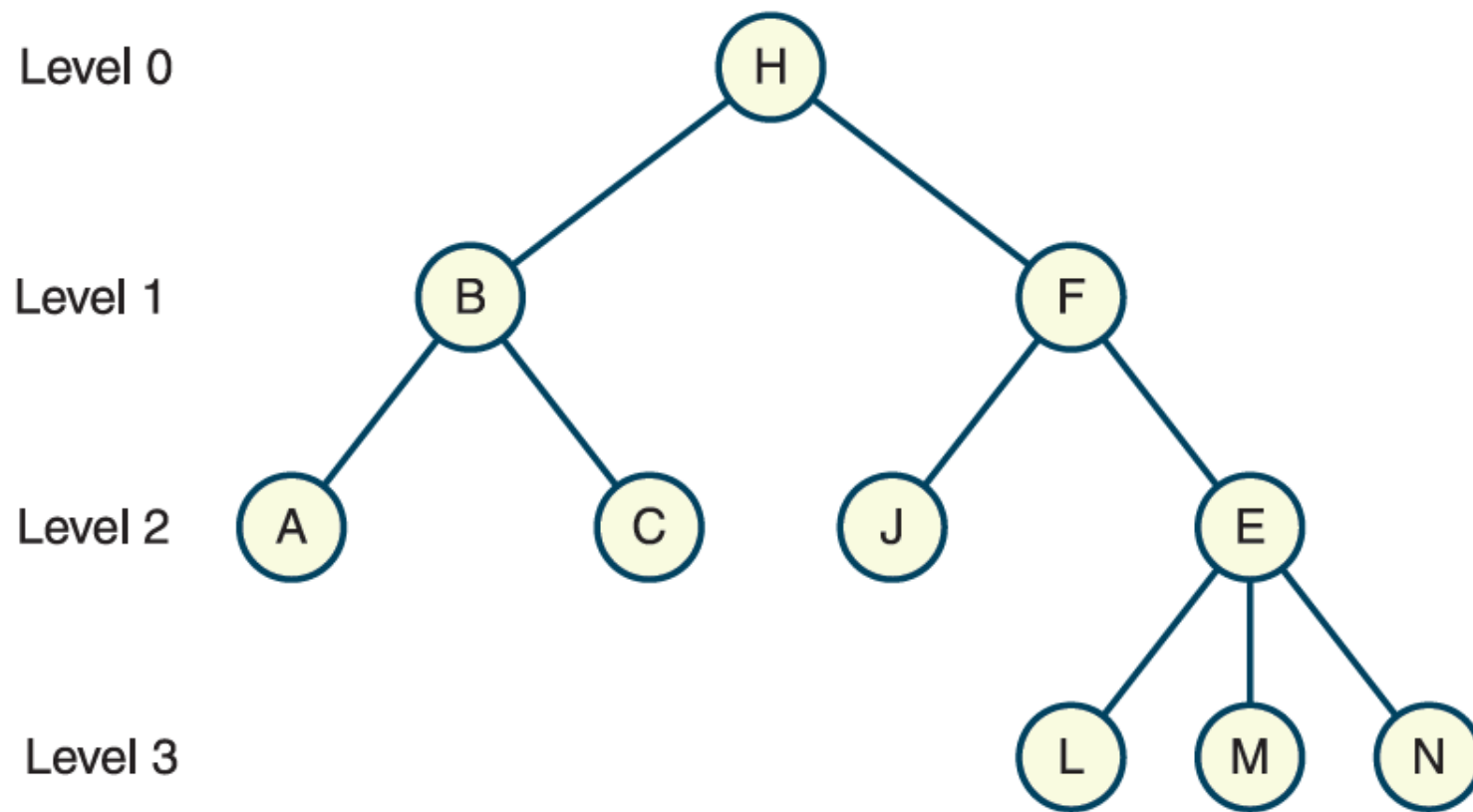
- The **height** of a node is the number of edges on the **longest path** from the **node** to a **leaf**.

A leaf node will have a height of 0.

# Terminology summary

Term	Definition
Node	An item stored in a tree.
Root	The topmost node in a tree. It is the only node without a parent.
Child	A node immediately below and directly connected to a given node. A node can have more than one child, and its children are viewed as organized in left-to-right order. The leftmost child is called the first child, and the rightmost is called the last child.
Parent	A node immediately above and directly connected to a given node. A node can have only one parent.
Siblings	The children of a common parent.
Leaf	A node that has no children.
Interior node	A node that has at least one child.

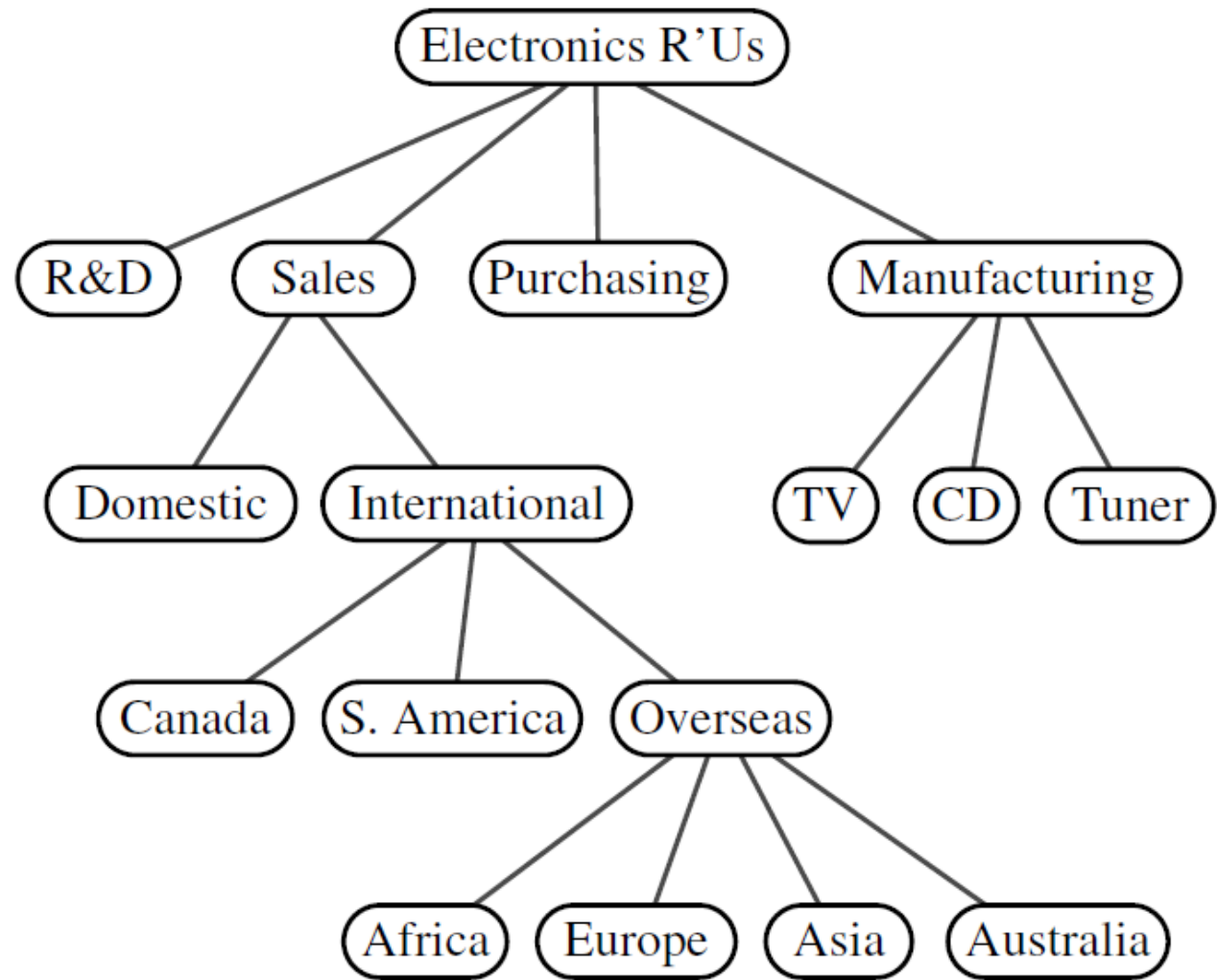
Edge/Branch/Link	The line that connects a parent to its child.
Descendant	A node's children, its children's children, and so on, down to the leaves.
Ancestor	A node's parent, its parent's parent, and so on, up to the root.
Path	The sequence of edges that connect a node and one of its descendants.
Path length	The number of edges in a path.
Depth or level	The depth or level of a node equals the length of the path connecting it to the root. Thus, the root depth or level of the root is 0. Its children are at level 1, and so on.
Height	The length of the longest path in the tree; put differently, the maximum level number among leaves in the tree.



<b>PROPERTY</b>		<b>VALUE</b>
Number of nodes		10
Height		3
Root node		H
Leaves		A, C, J, L, M, N
Interior nodes		H, B, F, E
Nodes at level 2		A, C, J, E
Ancestors of E		F, H
Descendants of F		J, E, L, M, N

# Organization of a fictitious corporation

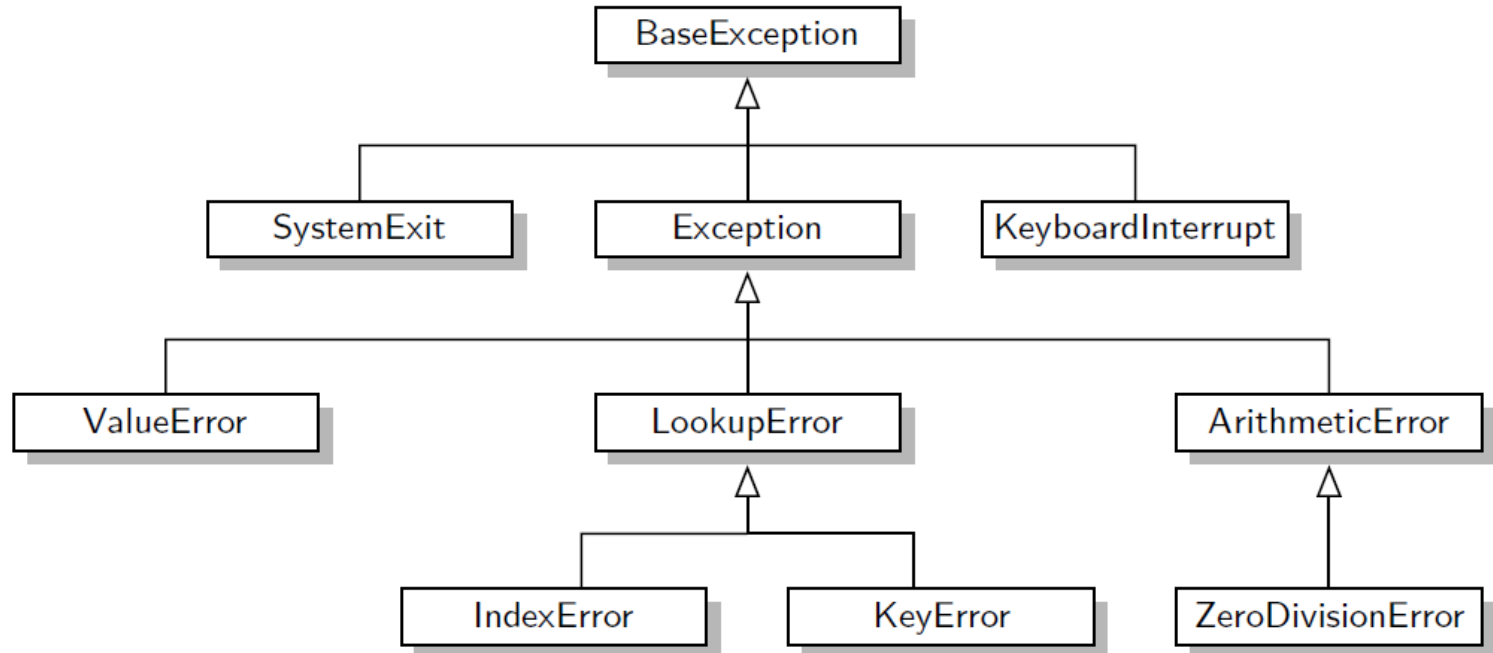
Practice :: Try to find the depth and height of each node in this tree!!!



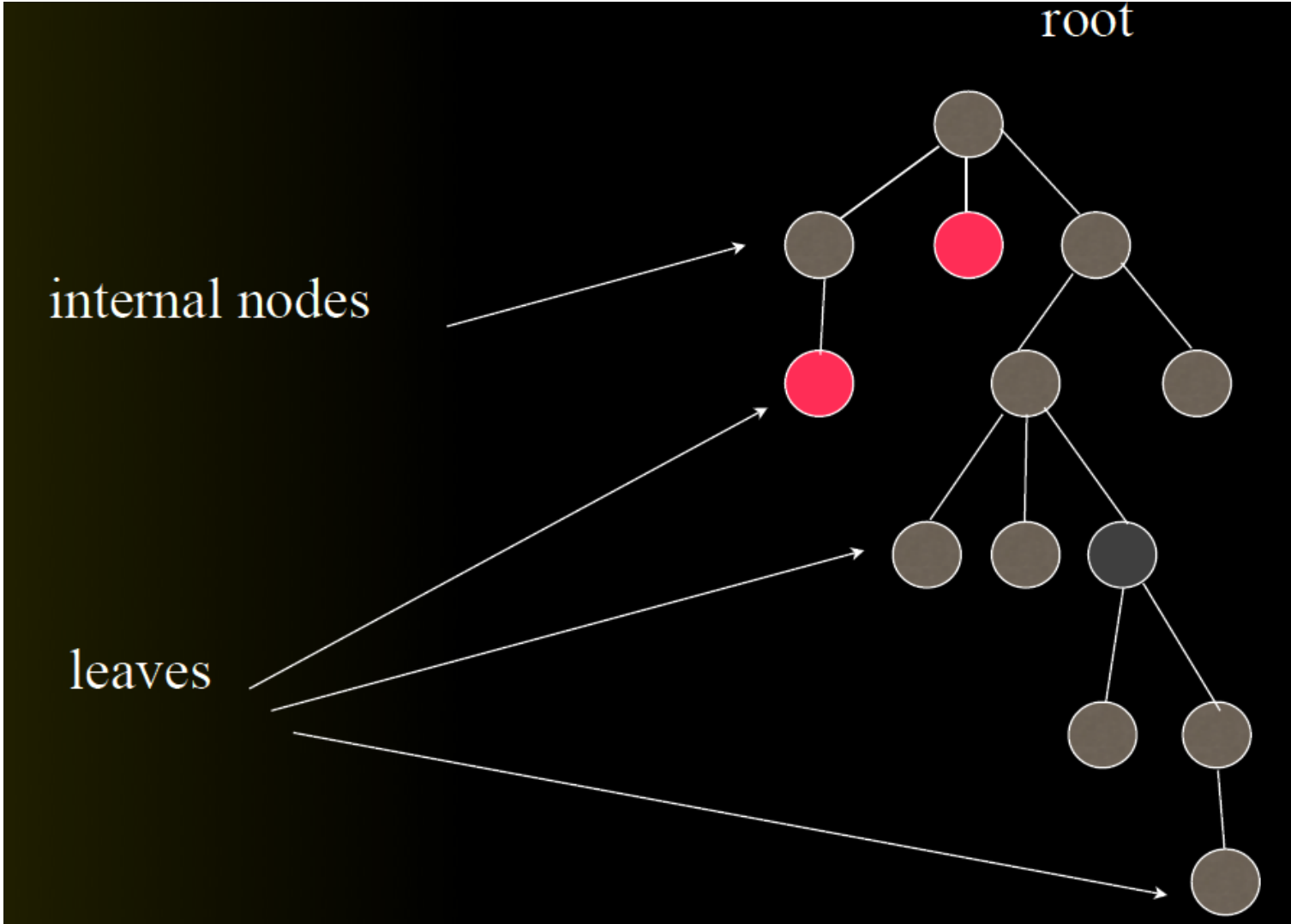


Practice :: Try to find the depth and height of each node in this tree!!!

# Python's hierarchy of exception types

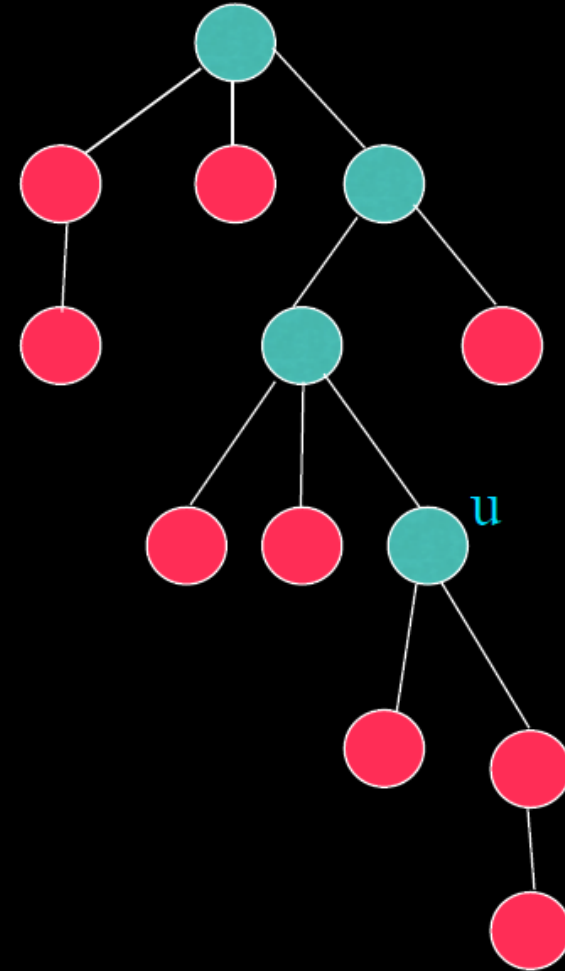


# Summary



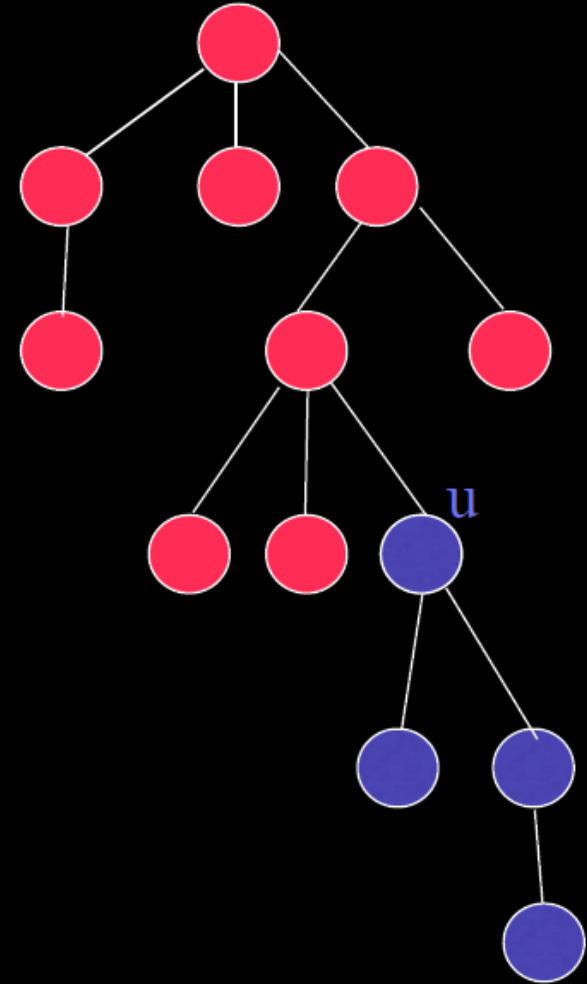
# Summary

ancestors of  $u$

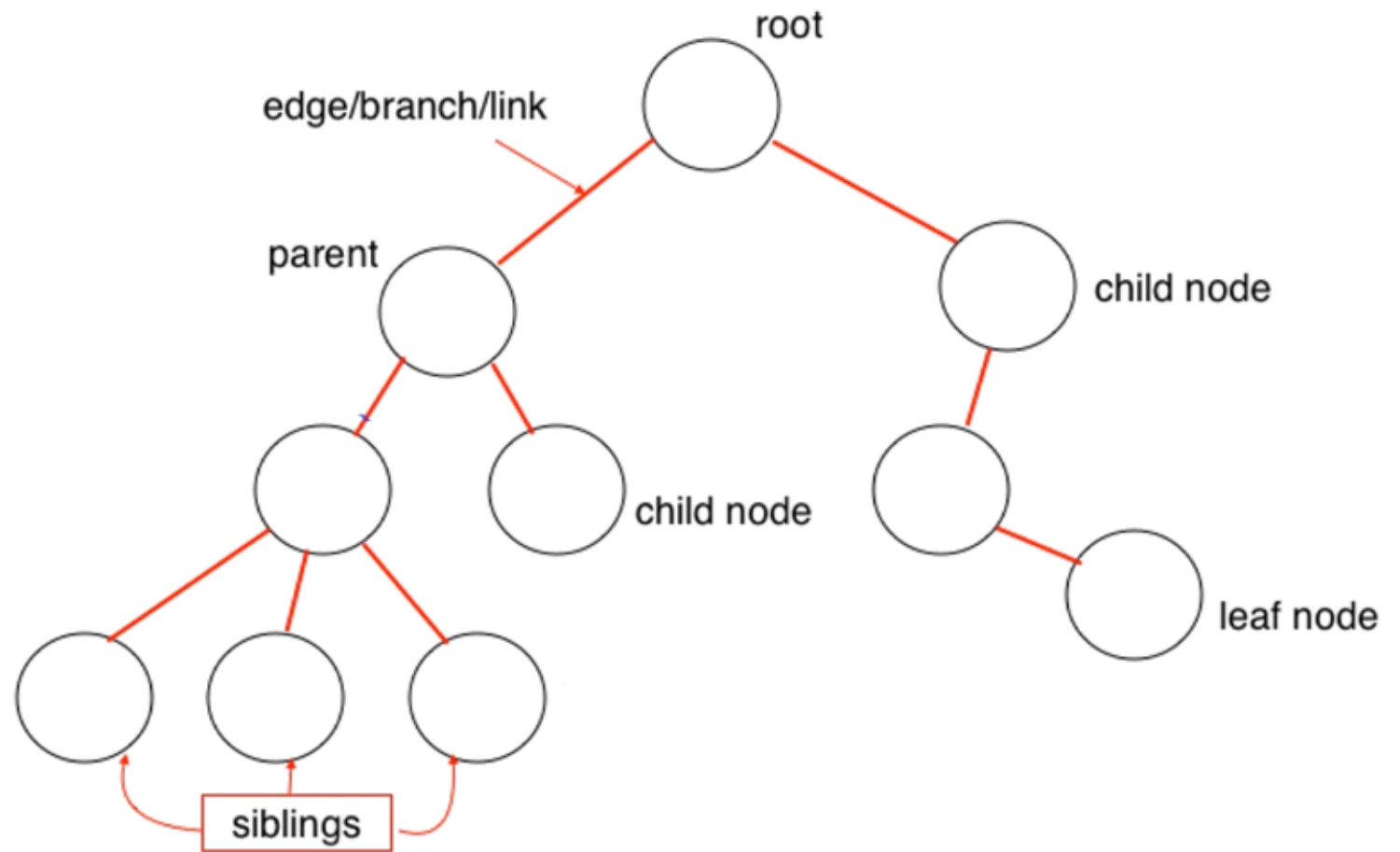


# Summary

descendants of  $u$



# Summary

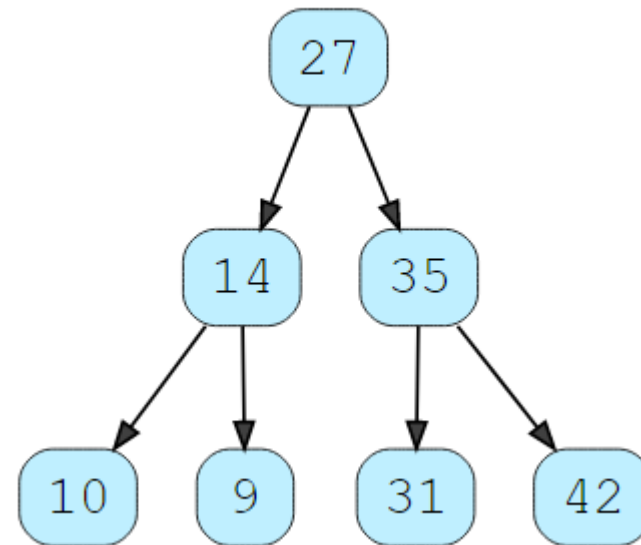


# Application of Tree

- Search trees store data in a way that makes an efficient search algorithm possible via tree traversal
- Represent organization
- Represent computer file systems
- Networks to find best path in the Internet

# Binary tree

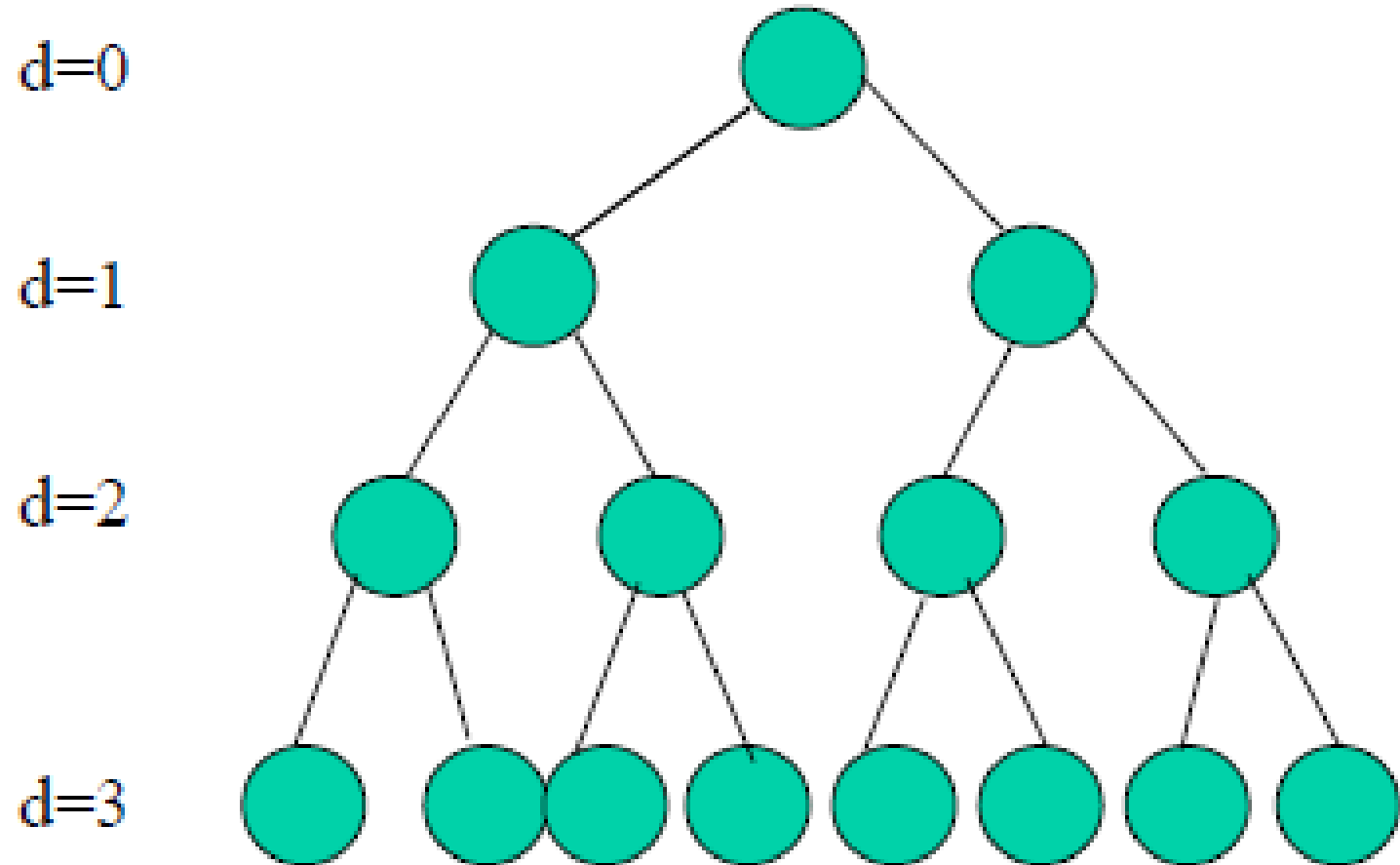
- A tree whose elements have at most two children is called a **binary tree**. Each element in a binary tree can have only two children.



## Properties of binary trees

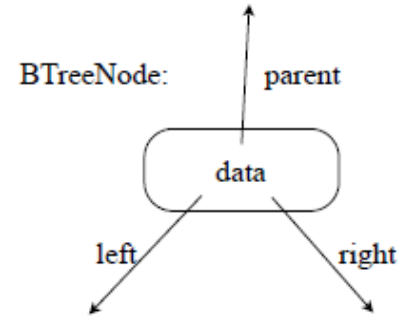
In a binary tree

- level 0 has  $\leq 1$  node
- level 1 has  $\leq 2$  nodes
- level 2 has  $\leq 4$  nodes
- ...
- level  $i$  has  $\leq 2^i$  nodes

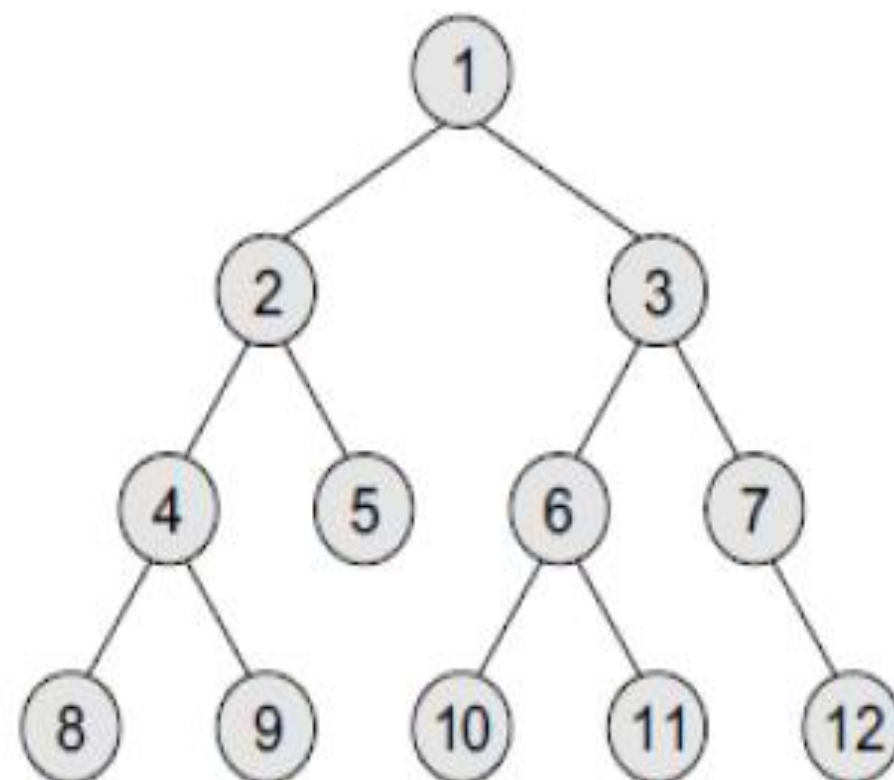
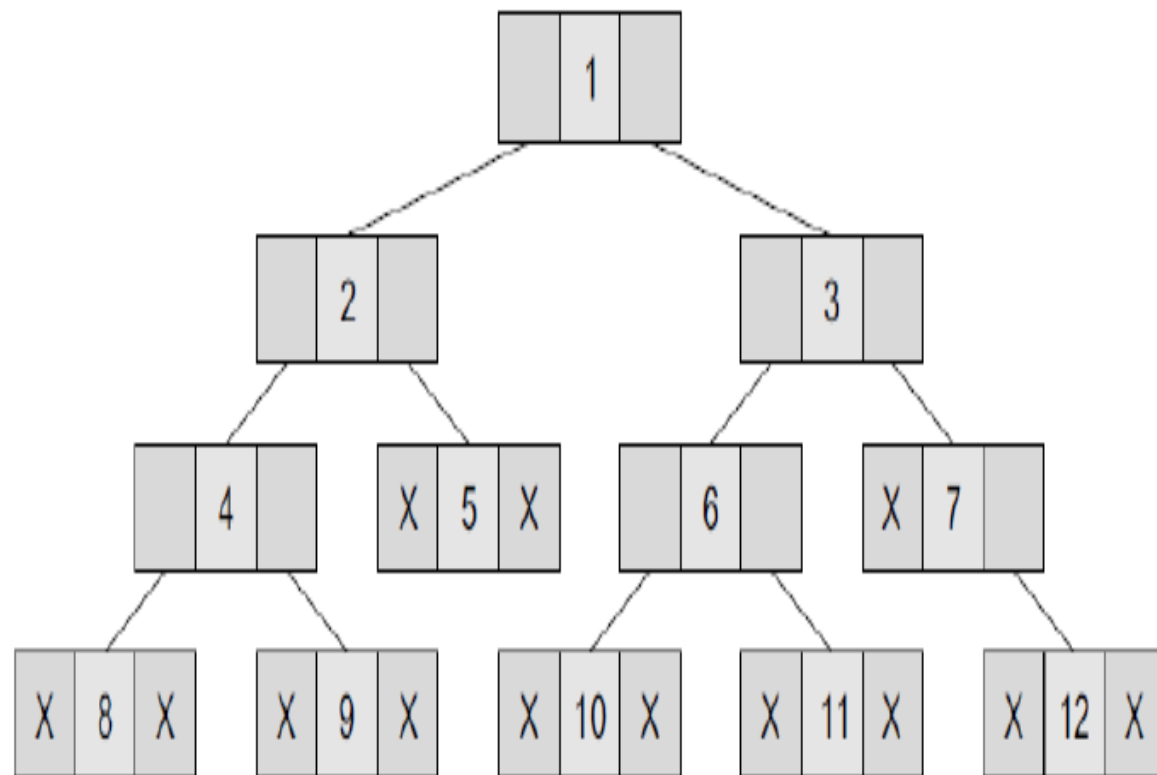





# Binary tree implementation



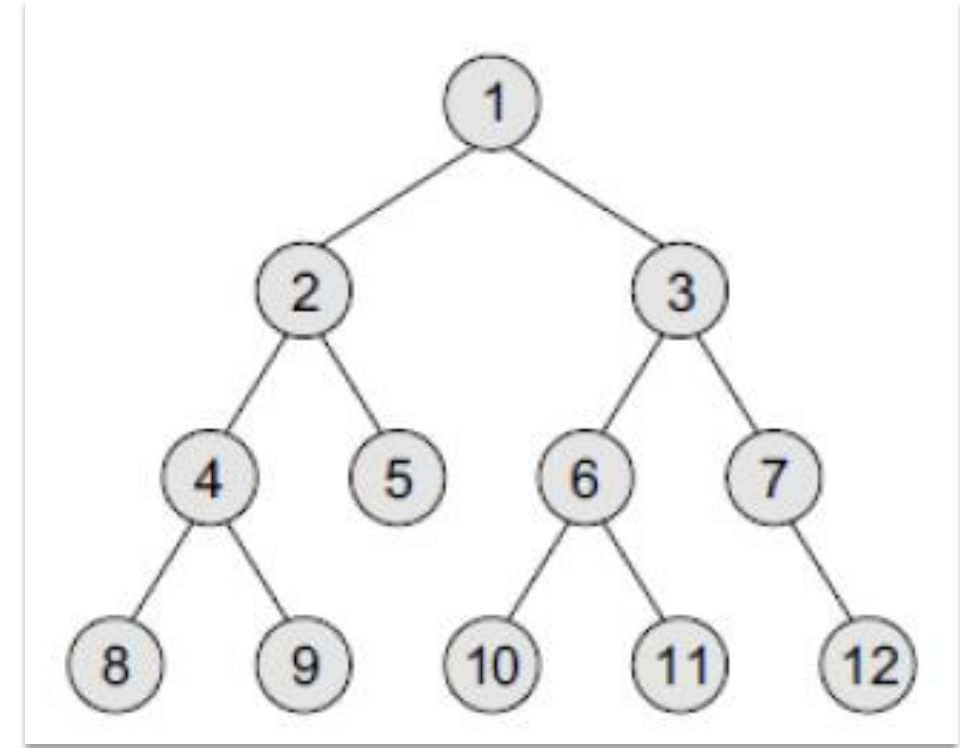
- each node points to its left and right children ;
- the tree class stores the root node and the size of the tree

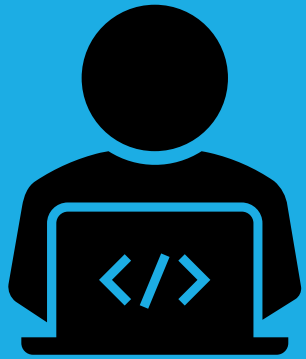


ROOT  
3



Mem location	LEFT	DATA	RIGHT
1	NONE	8	NONE
2	NONE	10	NONE
3	5	1	8
4			
5	9	2	14
6			
7			
8	20	3	11
9	1	4	12
10			
11	NONE	7	18
12	NONE	9	NONE
13			
14	NONE	5	NONE
15			
16	NONE	11	NONE
17			
18	NONE	12	NONE
19			
20	2	6	16





**THANK YOU!!!!!!**