CS-2001
DATA STRUCTURE

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

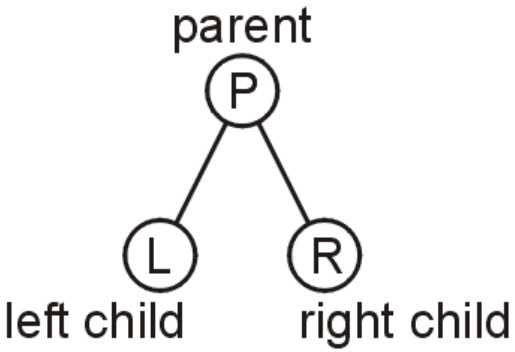
Faisalabad, Pakistan.

BINARY TREE

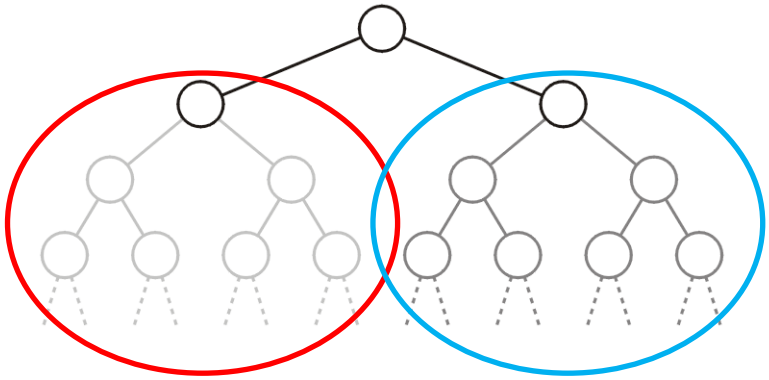
Binary Trees

Binary Tree

- In a binary tree, each node has at most two children
  - Allows to label the children as left and right



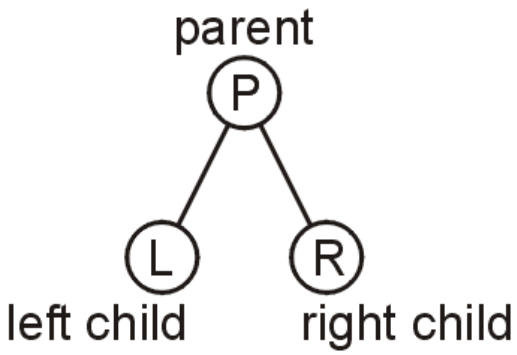
- Likewise, the two sub-trees are referred as
  - Left sub-tree
  - Right sub-tree



Binary Trees

Binary Tree

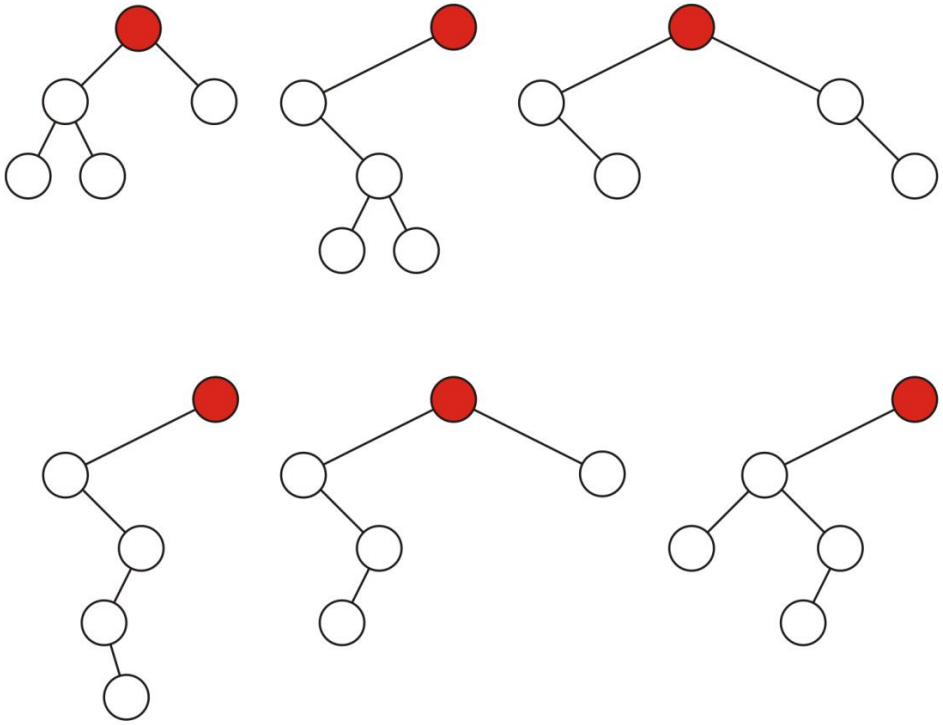
The mathematical definition of a binary tree is
"A binary tree is a finite set of elements that is either empty or is partitioned into three disjoint subsets. The first subset contains a single element called the root of the tree. The other two subsets are themselves binary trees called the left and right sub-trees".
Each element of a binary tree is called a node of the tree.
Following figure shows a binary tree.



Binary Trees

Binary Tree

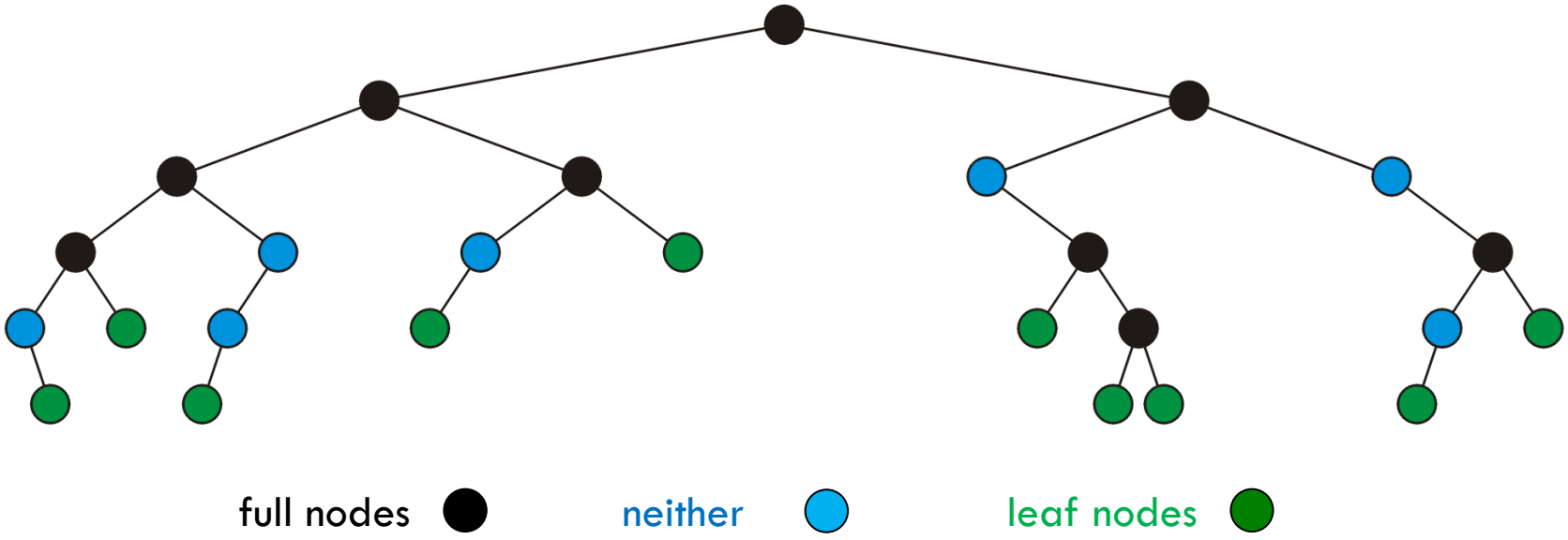
Some variations on binary trees with five nodes



Binary Trees

Full Node

A **full node** is a node where both the left and right sub-trees are non-empty trees

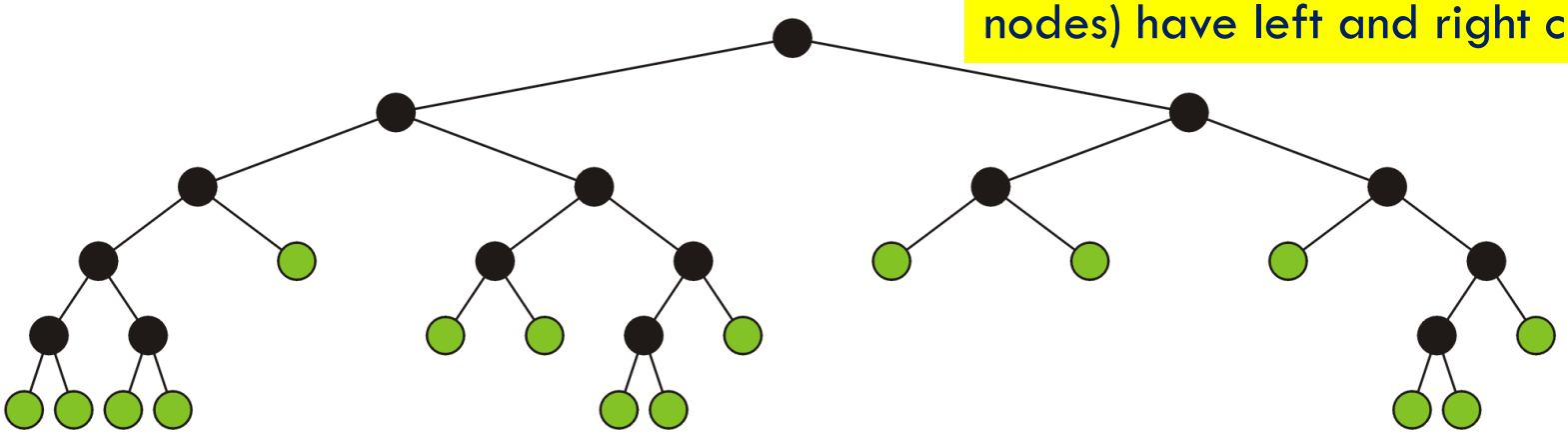


Full Binary Trees

Full Binary Tree

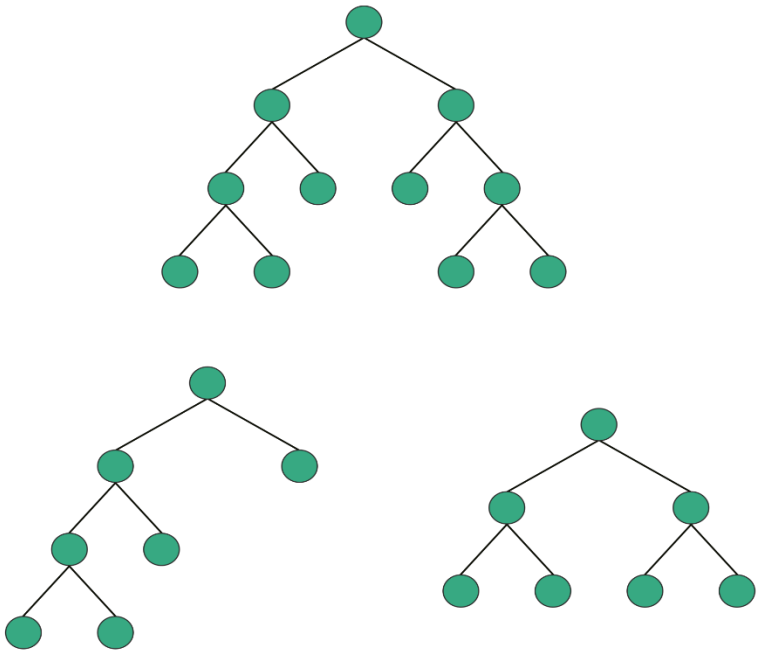
- A full binary tree is where each node is:
  - a full node, or
  - a leaf node.
- Full binary tree is also called **proper binary tree**, **strictly binary tree** or **2-tree**

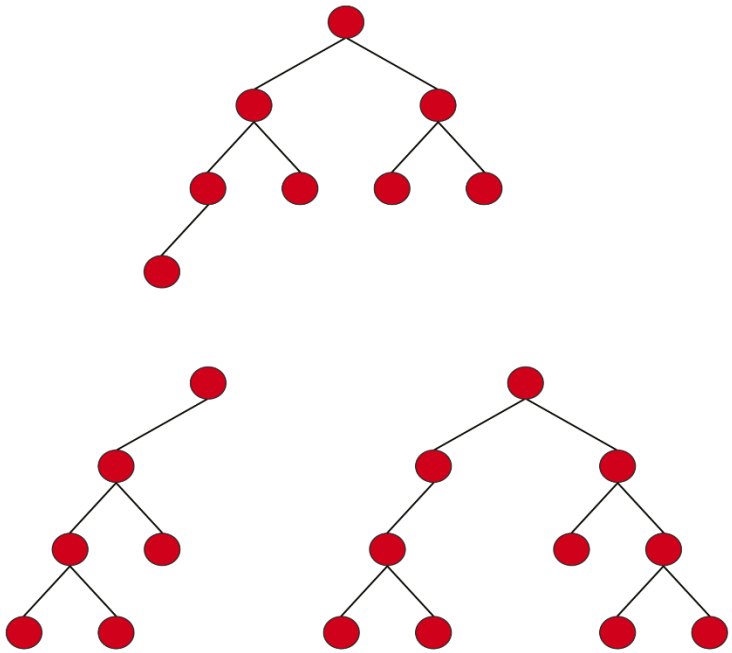
All the nodes (other than leaf nodes) have left and right child



Full Binary Trees

Full Binary Tree



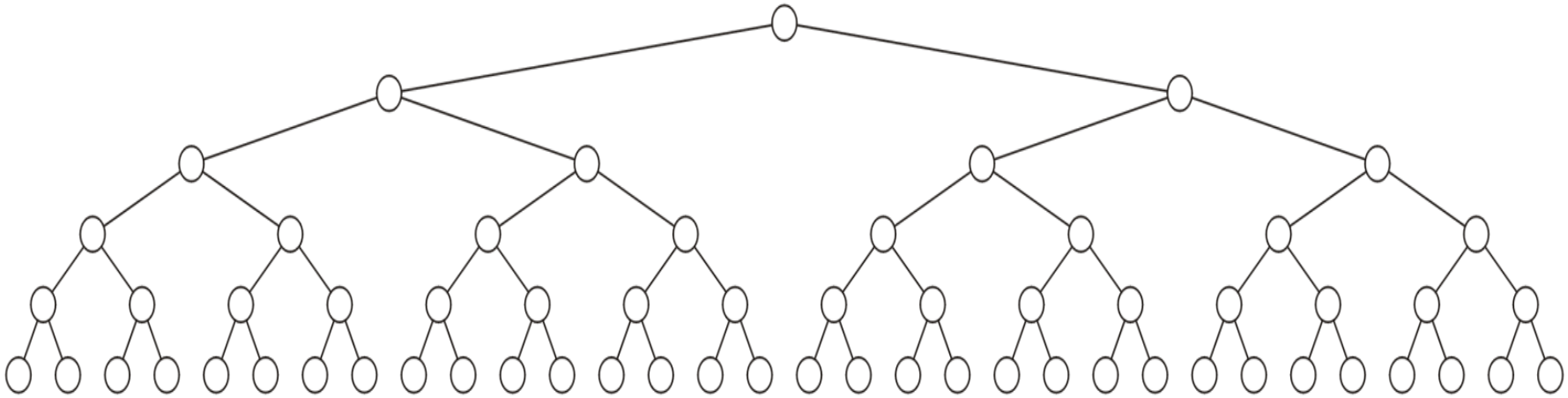


Valid and Invalid Structure of Full Binary Tree

Complete Binary Trees

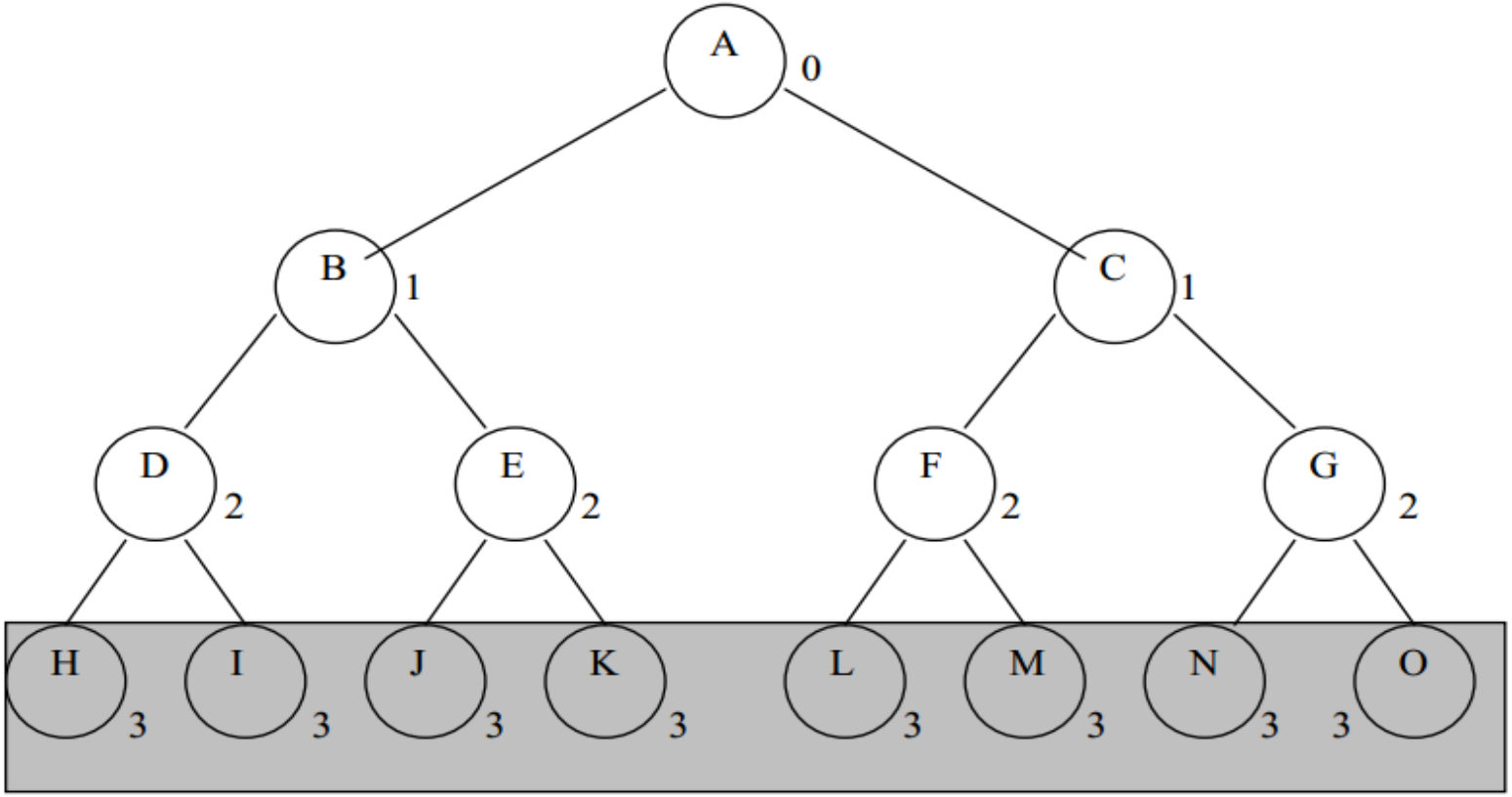
Complete (Perfect) Binary Tree

- A complete binary tree of height h is a binary tree, where,
  - All leaf nodes have the same depth h
  - All other nodes are **full** nodes



Complete Binary Trees

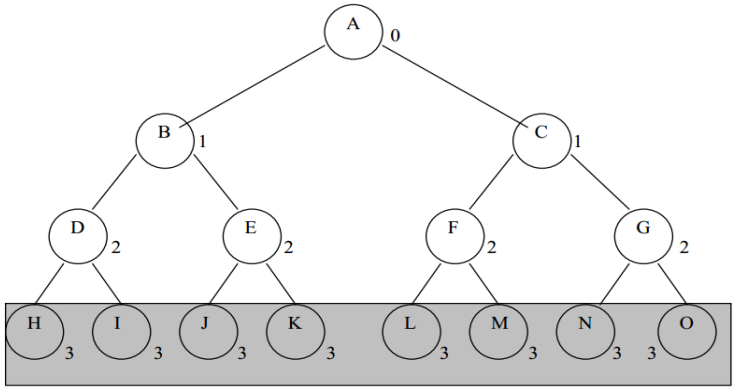
Complete (Perfect) Binary Tree … Examples



Complete Binary Trees

Complete (Perfect) Binary Tree … Examples

The definition of the complete binary tree is
"A complete binary tree of depth d is the strictly binary tree all of whose leaves are at level d".
Now look at the tree, the leaf nodes of the tree are at level 3 and are H, I, J, K, L, M, N and O. There is no such a leaf node that is at some level other than the depth level *d* i.e. 3. All the leaf nodes of this tree are at level 3 (which is the depth of the tree i.e. d). So this is a complete binary tree. In the figure, all the nodes at level 3 are highlighted.

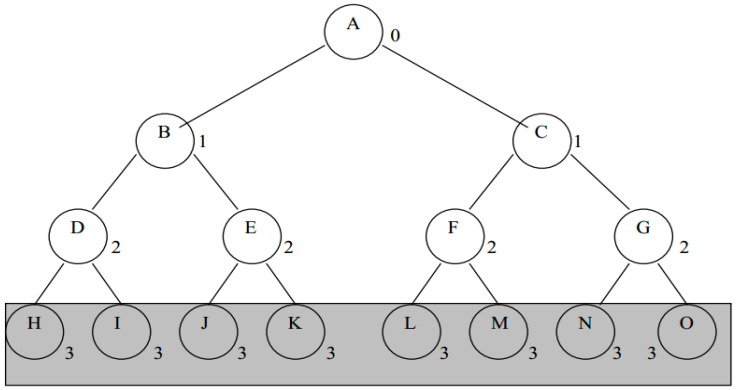


Complete Binary Trees

Complete (Perfect) Binary Tree … Examples

The definition of the complete binary tree is
"A complete binary tree of depth d is the strictly binary tree all of whose leaves are at level d".
Now look at the tree, the leaf nodes of the tree are at level 3 and are H, I, J, K, L, M, N and O. There is no such a leaf node that is at some level other than the depth level *d* i.e. 3. All the leaf nodes of this tree are at level 3 (which is the depth of the tree i.e. d). So this is a complete binary tree. In the figure, all the nodes at level 3 are highlighted.

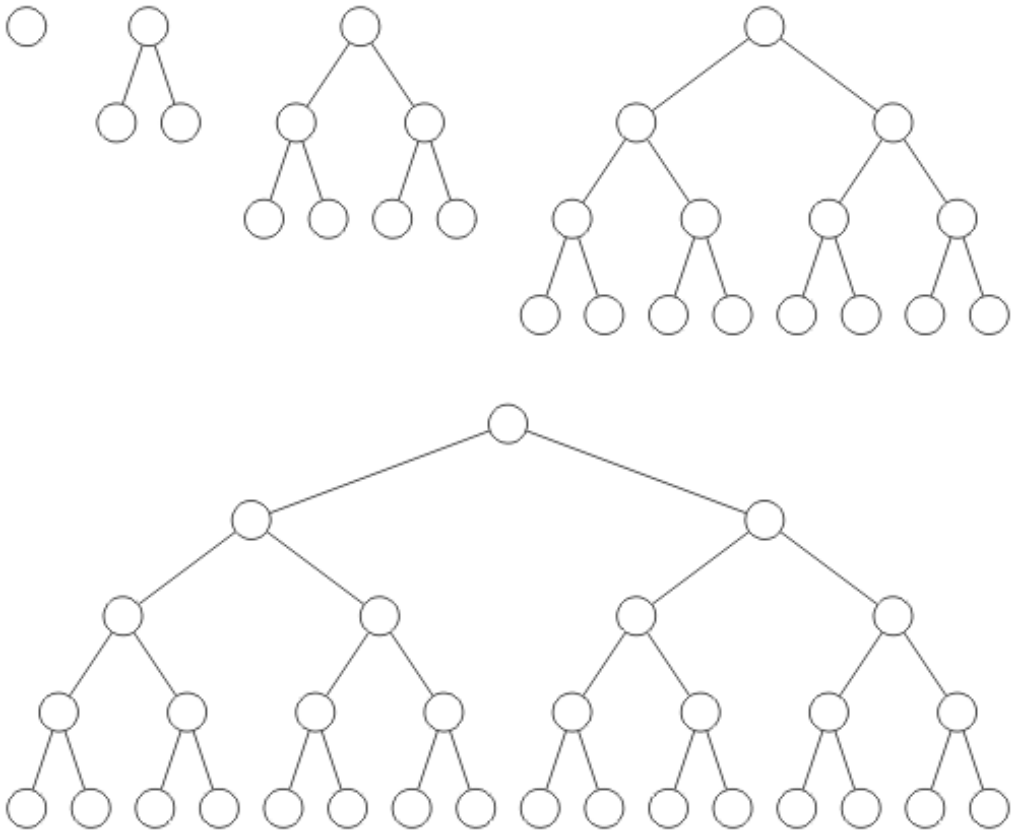
We can say every Strictly Binary Tree may or may not be a complete binary tree but every complete or perfect binary tree will always Strictly as well



Complete Binary Trees

Complete (Perfect) Binary Tree … Examples

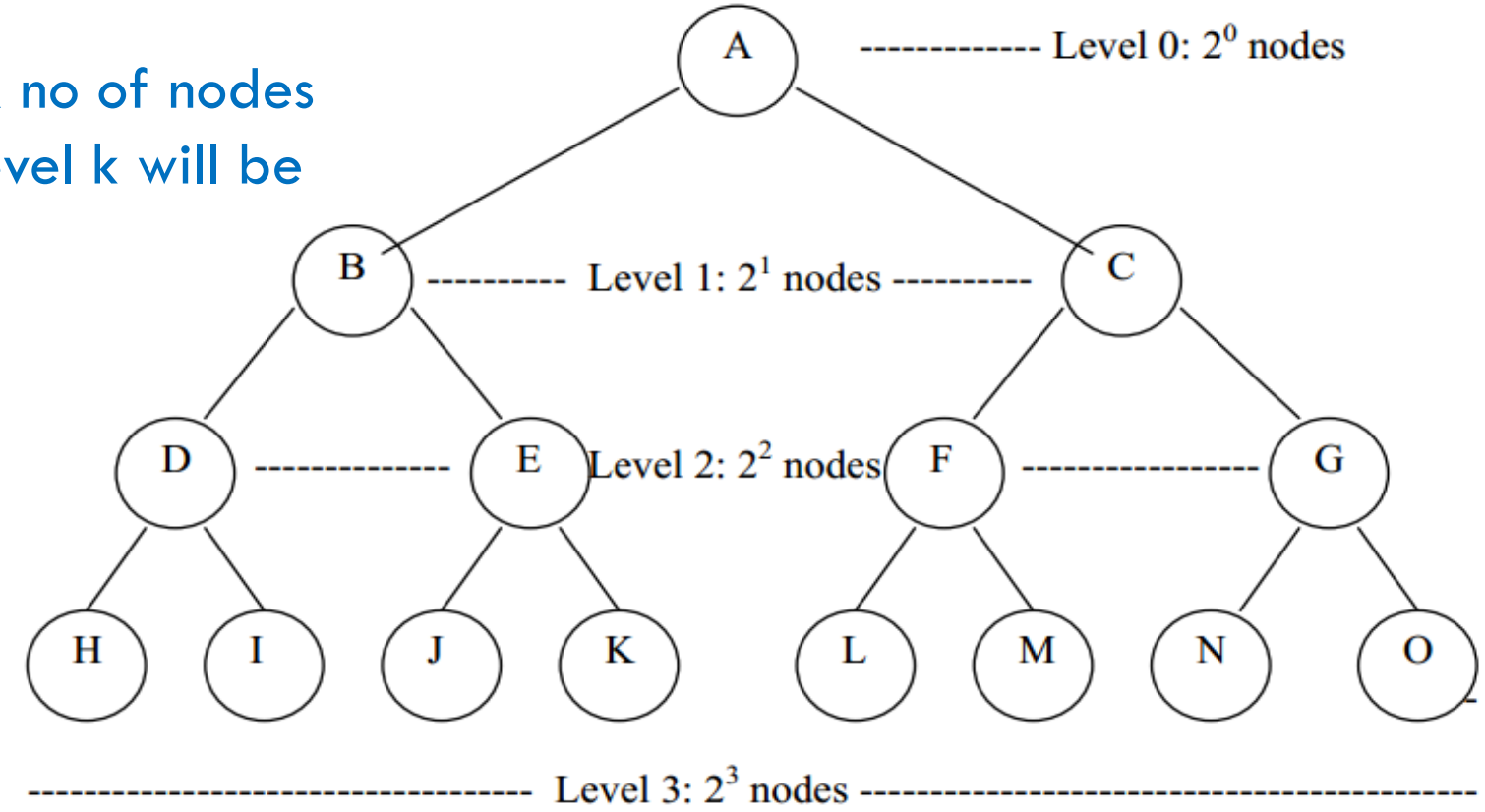
Complete binary trees of height *h = 0, 1, 2, 3 and 4*



Complete Binary Trees

Complete (Perfect) Binary Tree … Examples

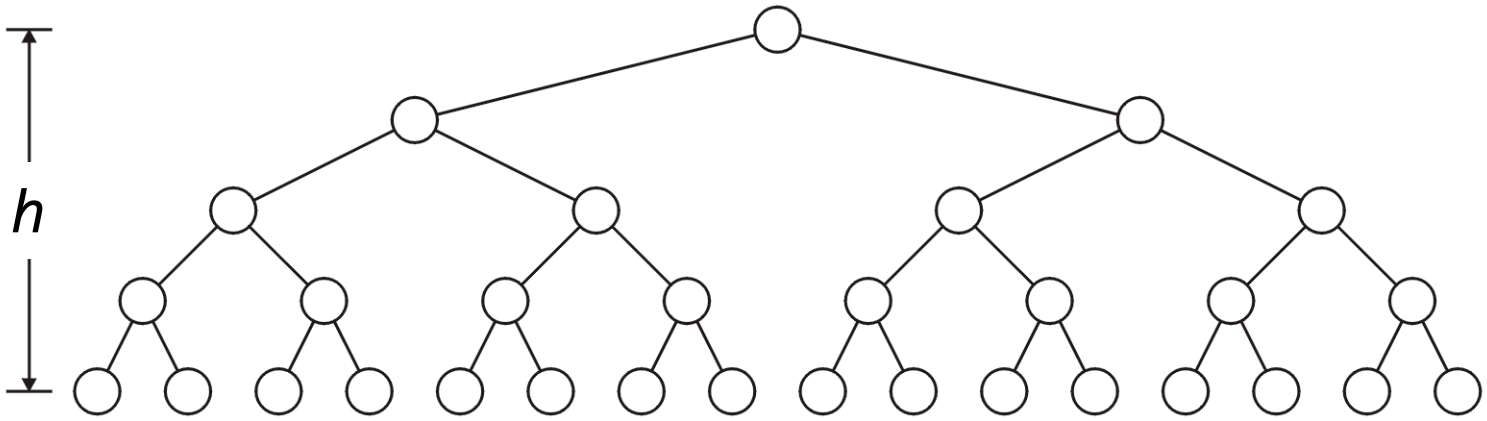
Max no of nodes at level k will be 2^k



Complete Binary Trees

Complete (Perfect) Binary Tree … Properties

- A complete binary tree with height h has 2^h leaf nodes



Complete Binary Trees

Complete (Perfect) Binary Tree … Properties

Depth d	# nodes at depth d	# of child nodes
0	1 = 2^0	2 (each node has 2 children)
1	2 = 2^1	4 (each node has 2 children)
2	4 = 2^2	8 (each node has 2 children)
...		

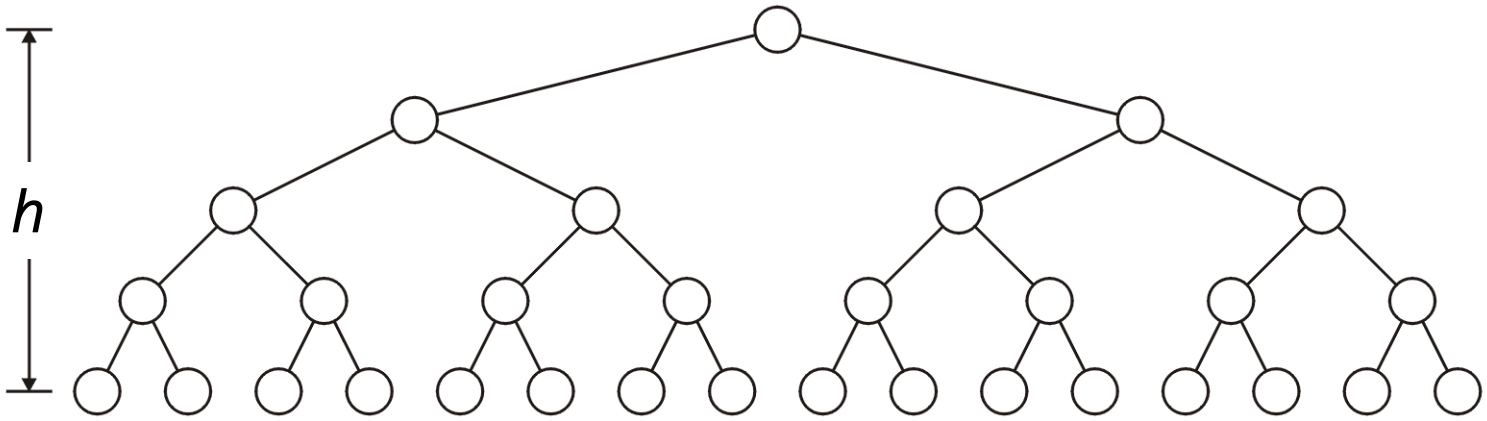
- The **number** of nodes **doubles** every time the **depth** increases by **1** !

$$\text{\# nodes at depth } d = 2^d$$

Complete Binary Trees

Complete (Perfect) Binary Tree … Properties

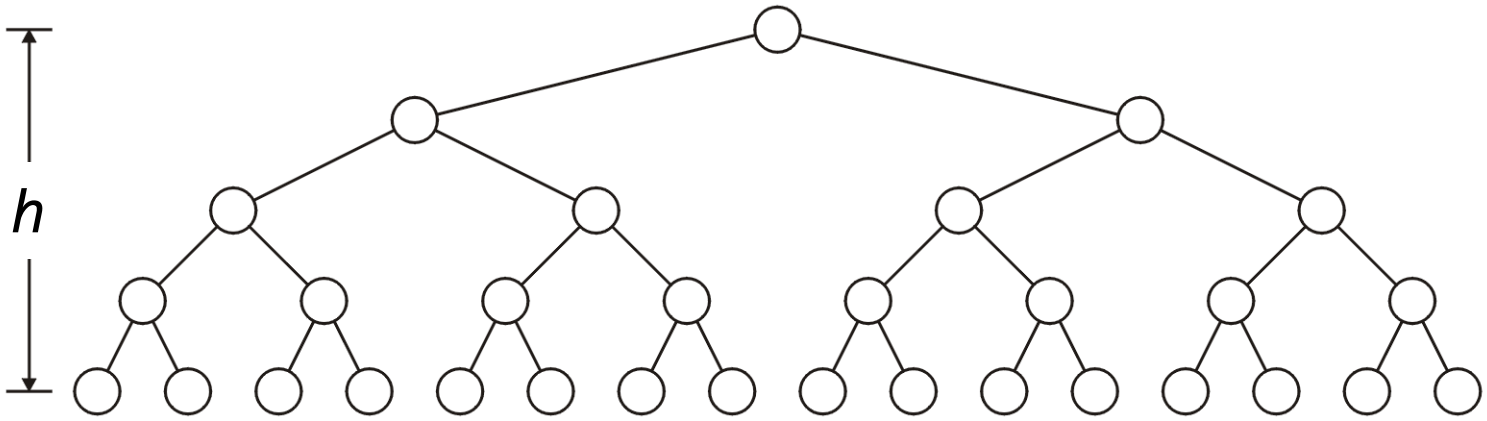
- A complete binary tree with height h has 2^h leaf nodes
- A complete binary tree of height h has $2^{h+1} - 1$ nodes



Complete Binary Trees

Complete (Perfect) Binary Tree … Properties

- A complete binary tree with height h has 2^h leaf nodes
- A complete binary tree of height h has $2^{h+1} - 1$ nodes
  - Number of leaf nodes: $L = 2^h$
  - Number of internal nodes: $2^h - 1$
  - Total number of nodes: $2^{h+1} - 1$



Complete Binary Trees

Complete (Perfect) Binary Tree … Properties

- A complete binary tree with height h has 2^h leaf nodes
- A complete binary tree of height h has $2^{h+1} - 1$ nodes
  - Number of leaf nodes: $L = 2^h$
  - Number of internal nodes: $2^h - 1$
  - Total number of nodes: $2^{h+1} - 1$
- A complete binary tree with n nodes has height $\log_2(n + 1) - 1$

$$n = 2^{h+1} - 1$$

$$2^{h+1} = n + 1$$

$$h + 1 = \log_2(n + 1)$$

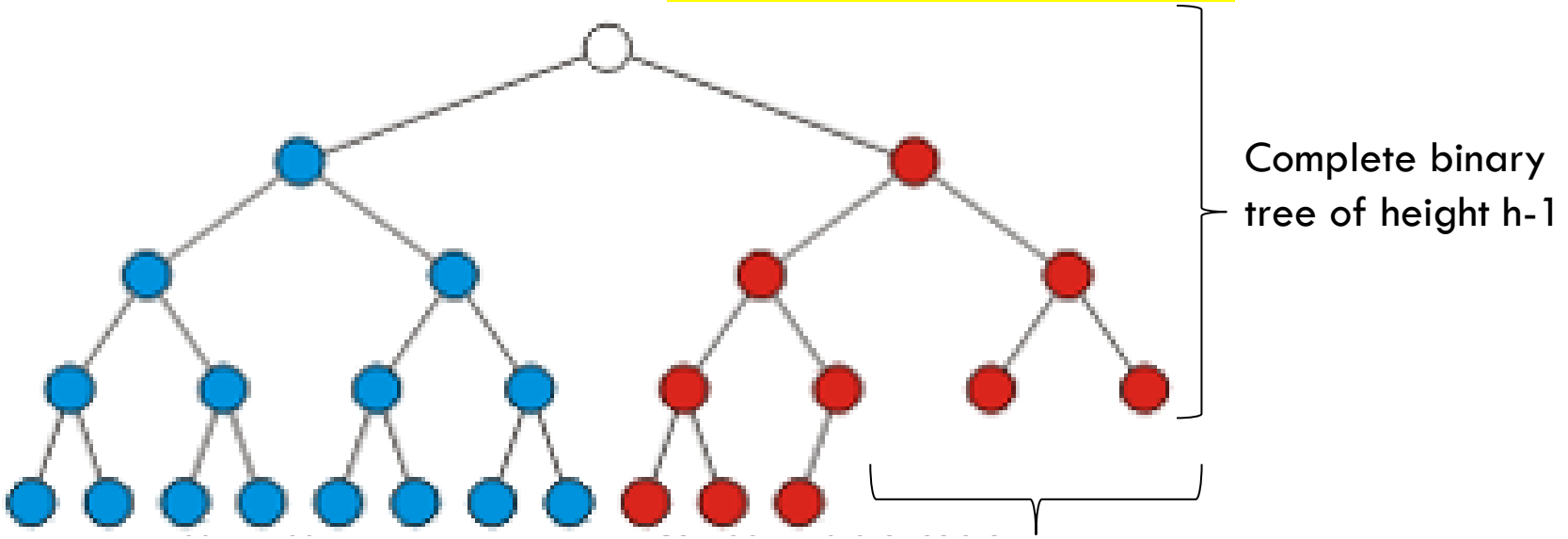
$$\Rightarrow h = \log_2(n + 1) - 1$$

Almost Complete Binary Trees

Almost Complete Binary Tree

Almost complete binary tree of height h is a binary tree in which

1. There are 2^d nodes at depth d for $d = 1, 2, \ldots, h-1$.
   Each leaf in the tree is either at level h or at level h–1.
2. ***The nodes at depth h are as far left as possible***

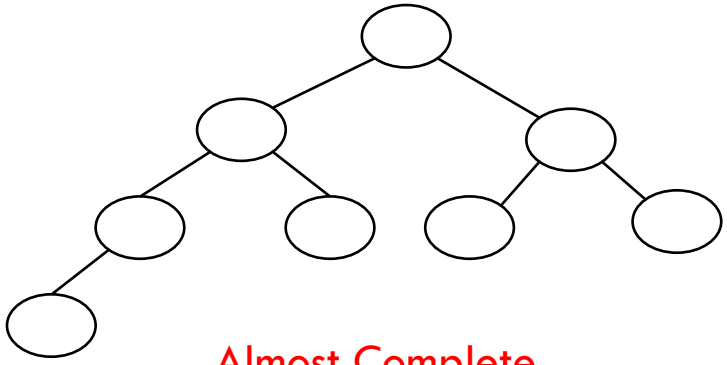


Almost Complete Binary Trees

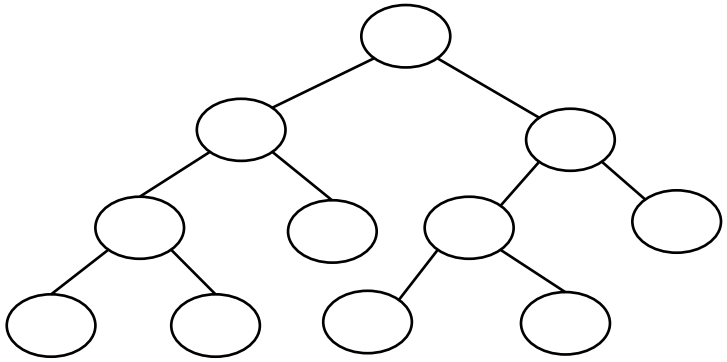
Almost Complete Binary Tree

The nodes at depth h are as far left as possible

- If a node p at depth h−1 has a left child
  - Every node at depth h−1 to the left of p has 2 children
- If a node at depth h−1 has a right child
  - It also has a left child



Almost Complete binary tree



Not Almost Complete binary tree (condition 2 violated)

Almost Complete Binary Trees

Almost Complete Binary Tree … Properties

Total number of nodes n is between

A complete binary tree of height h-1, i.e., 2^h nodes

A complete binary tree of height h, i.e., $2^{h+1} - 1$ nodes

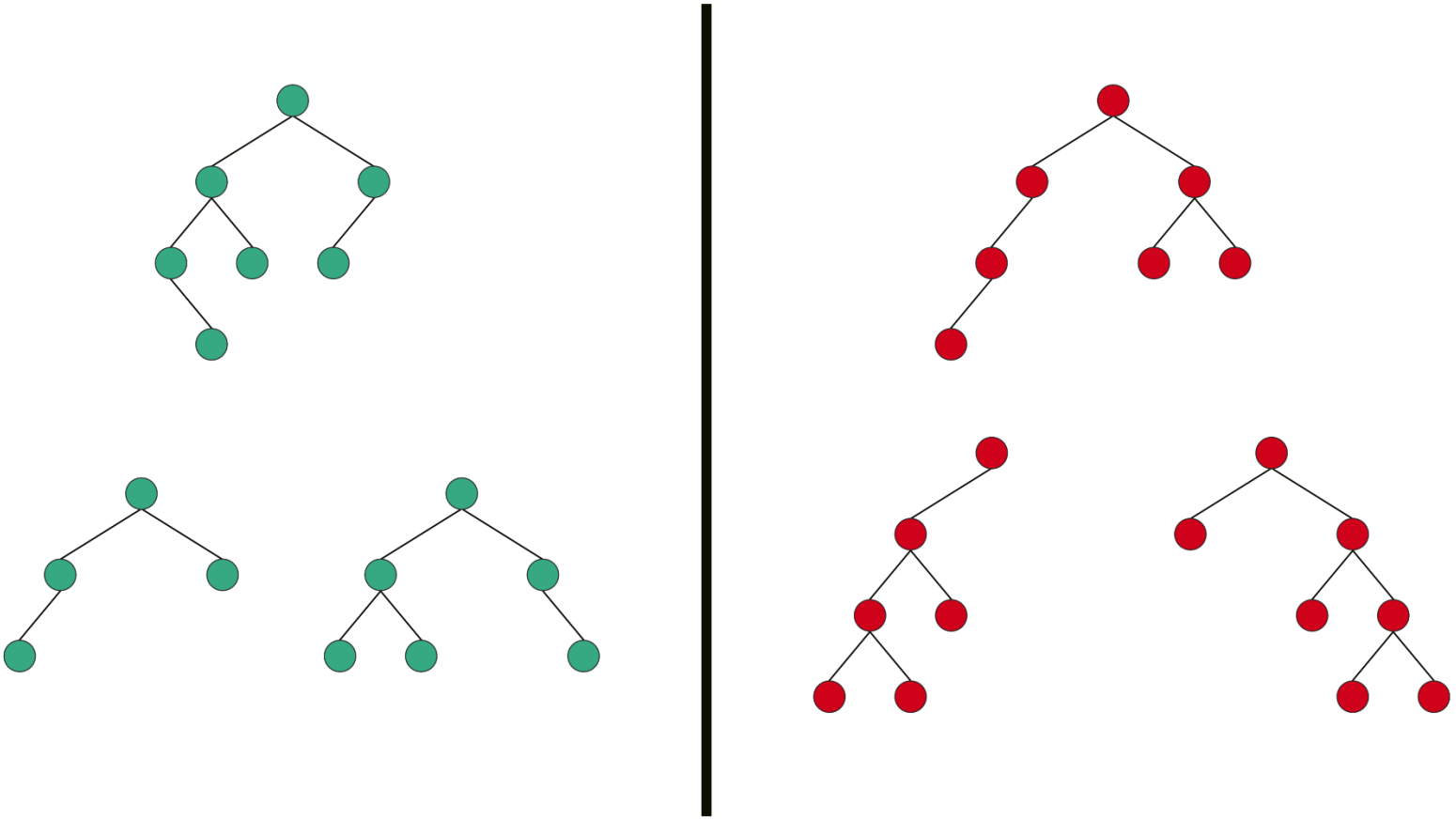
Height h is the largest integer less than or equal to $\log_2(n)$

Balanced Binary Trees

Balanced Binary tree

- is a Binary tree in which height of the left and the right sub-trees of every node may differ by at most 1

Balanced Binary Trees



Valid and Invalid Structure of Balanced Binary Tree

BINARY TREE IMPLEMENTATION

Binary Tree … Implementation

- Binary Tree … Implementation
  - Array based implementation
  - Linked List based implementation

Binary Tree … Implementation

- Objects: any type of objects can be stored in a tree data structure
- Accessor methods
  - root() – return the root of the tree
  - parent(p) – return the parent of a node
  - children(p) – returns the children of a node

Binary Tree … Implementation

- query methods
  - size() – returns the number of nodes in the tree
  - isEmpty() - returns true if the tree is empty
  - elements() – returns all elements
  - isRoot(p), isInternal(p), isExternal(p)
- other methods
  - Tree traversal, Node addition/deletion, create/destroy

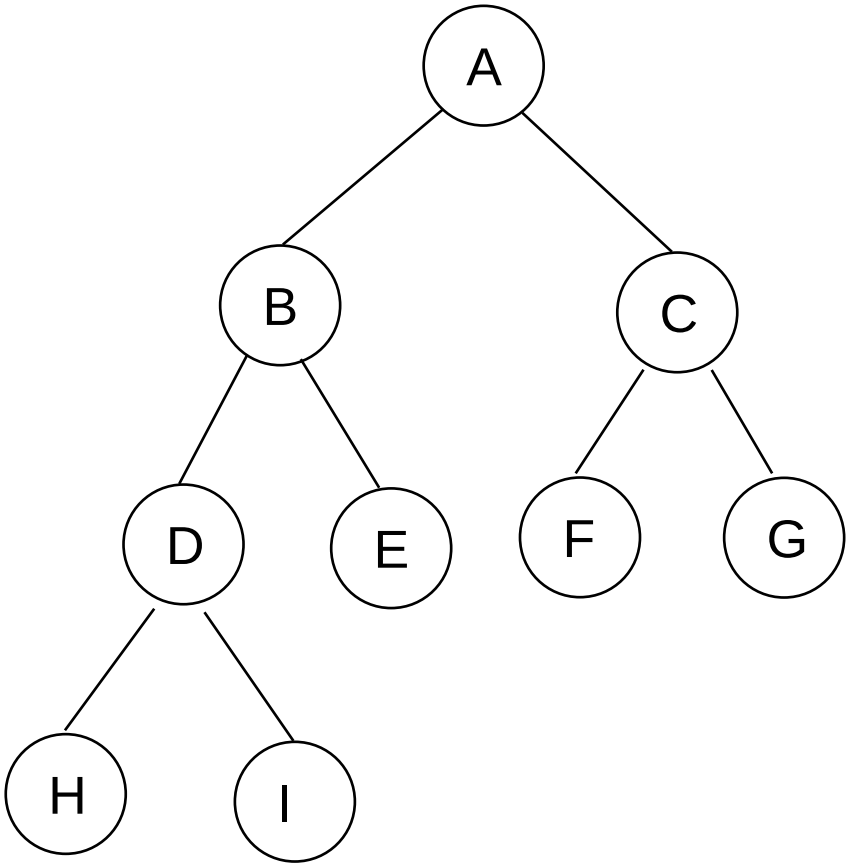
Binary Tree … Implementation

Array-based Implementation:

- Value in root node stored first, followed by left child, then right child
- Each successive level in the tree ***stored left to right***; unused nodes in tree represented by a bit pattern to indicate nothing stored there
- Children of any given node *n* is stored in cells *2n* and *2n + 1* (If array index starts at 1)
- *Storage allocated as for full tree, even if many nodes empty*
- For a tree of depth *h* we need array of $2^{h+1}-1$ cells

Binary Tree … Implementation

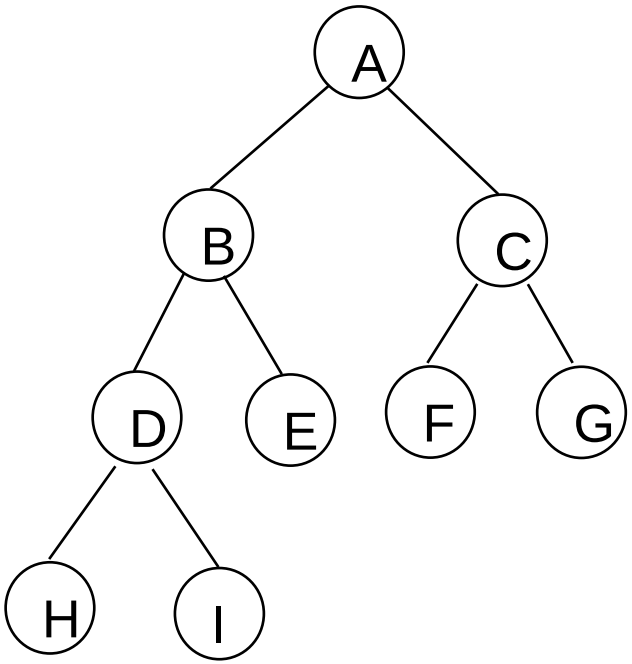
Array-based Implementation:

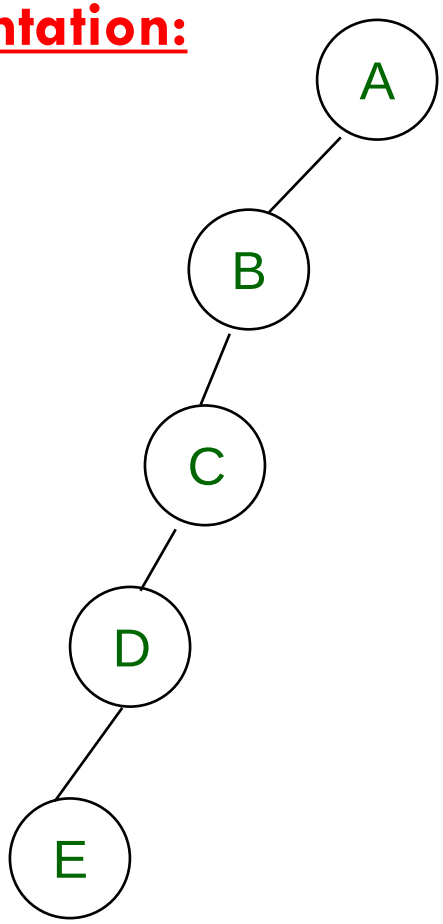


[1]	A
[2]	B
[3]	C
[4]	D
[5]	E
[6]	F
[7]	G
[8]	H
[9]	I

Binary Tree … Implementation

Array-based Implementation:

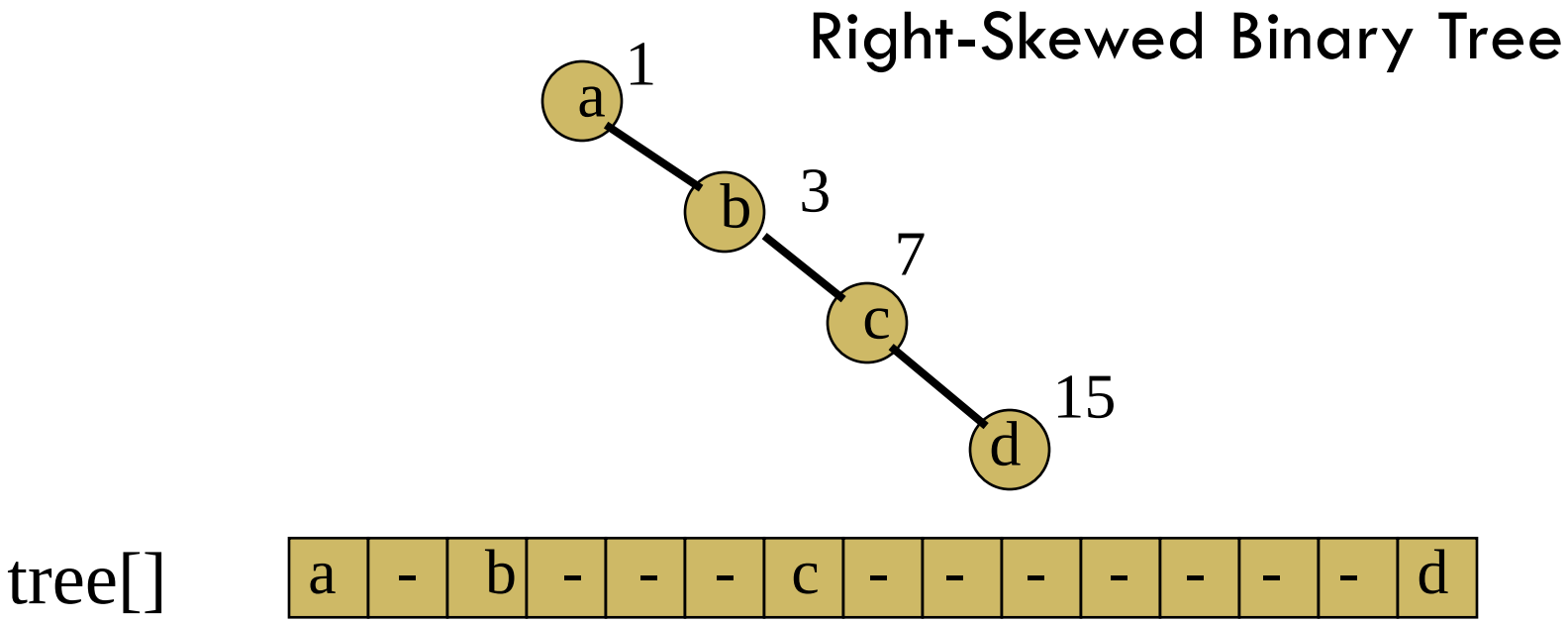




[1]	A
[2]	B
[3]	--
[4]	C
[5]	--
[6]	--
[7]	--
[8]	D
[9]	--
.	.
[16]	E

Binary Tree … Implementation

Array-based Implementation:



Binary Tree … Implementation

Linked List-based Implementation:

- Implementation of Binary tree can be done using dynamic creation of node having data and two pointers.

```
Struct Node {
    int data;
    Node* left;
    Node * right;
    }
```

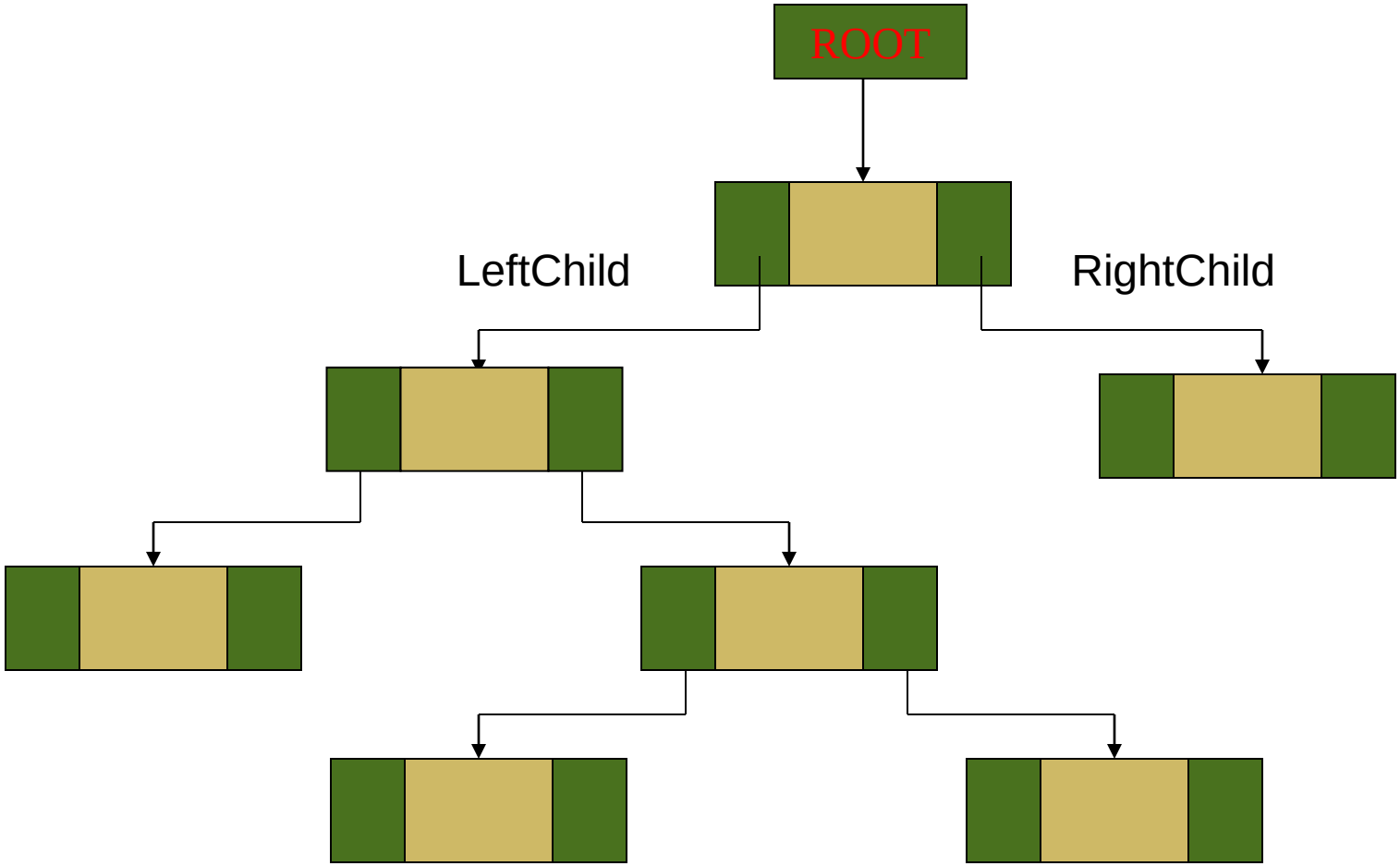
Binary Tree … Implementation

- A binary tree is composed of zero or more nodes
- Each node contains:
  - A value (some sort of data item)
  - A reference or pointer to a left child (may be null), and
  - A reference or pointer to a right child (may be null)
- A binary tree may be *empty* (contain no nodes)

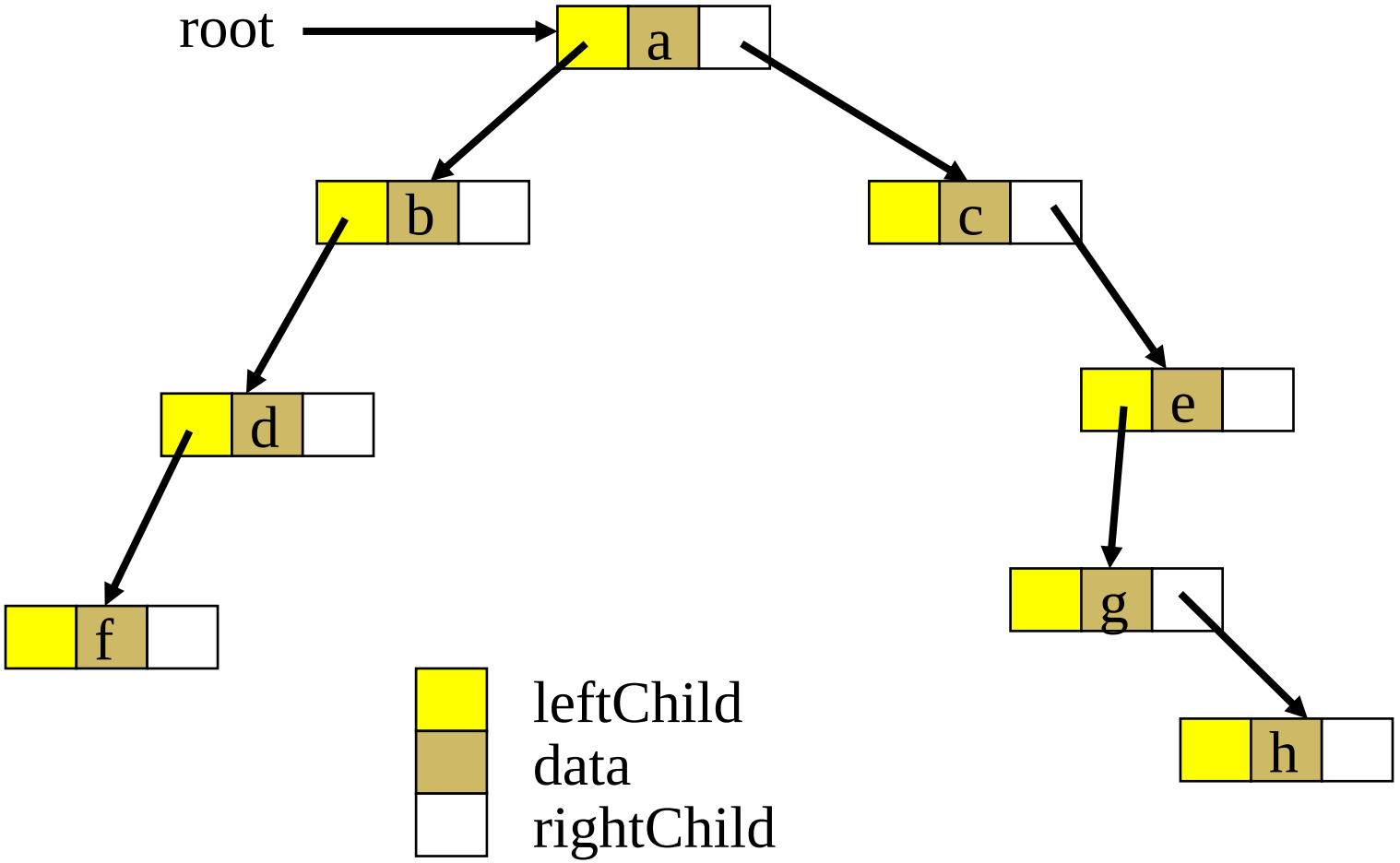
Binary Tree … Implementation

- If not empty, a binary tree has a root node
  - Every node in the binary tree is reachable from the root node by a ***unique*** path
- A node with neither a left child nor a right child is called a **leaf**
  - In some binary trees, only the leaves contain a value

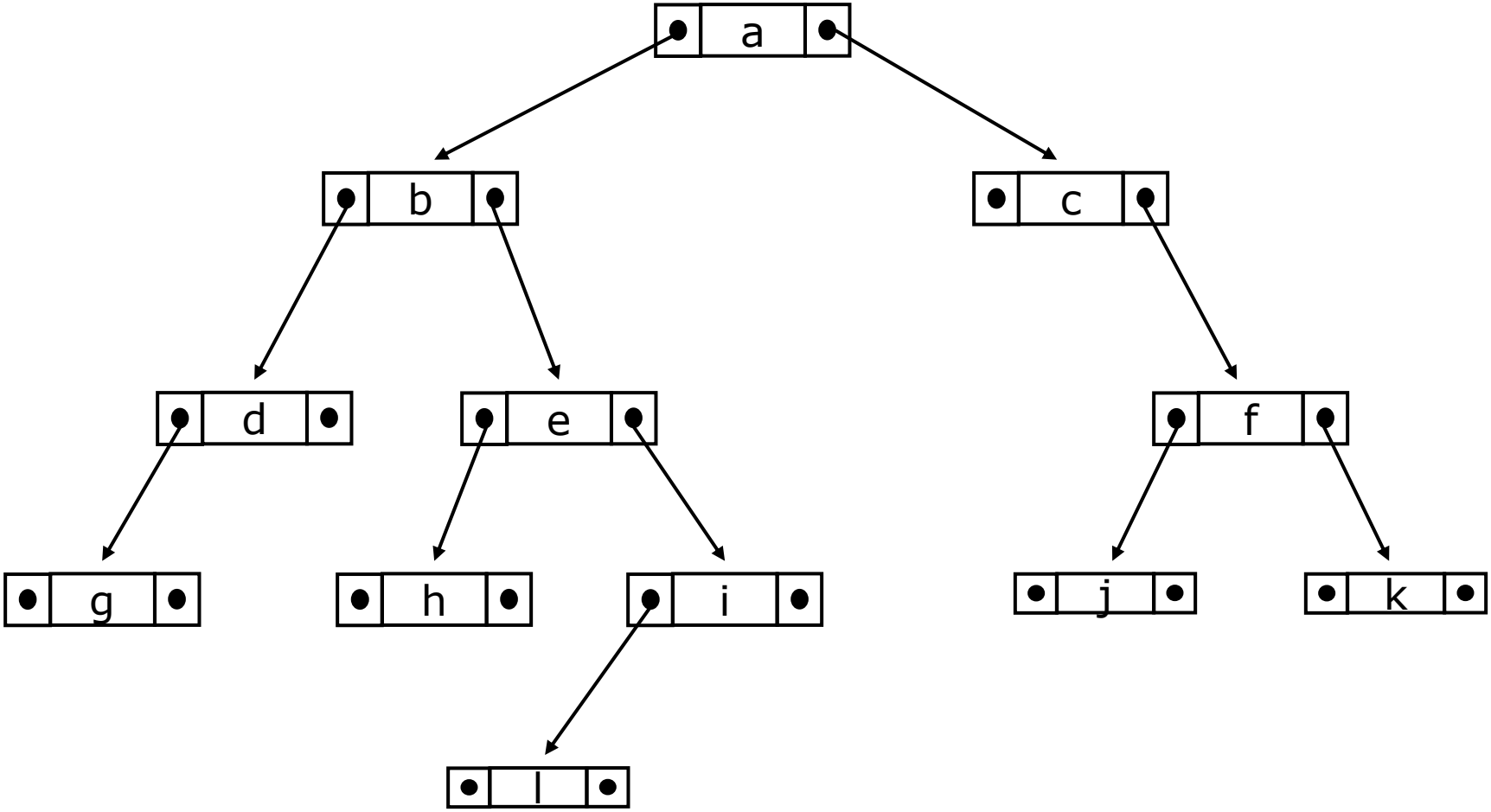
Binary Tree … Implementation



Binary Tree … Implementation



Binary Tree … Implementation



Binary Tree … Implementation

Implementation: with Linked List

- **Each node** in the tree consists of:
  - The data, or value contained in the element
  - A left child pointer (pointer to first child)
  - A right child pointer (pointer to second child)
- A **root pointer** points to the root node
  - Follow pointers to find every other element in the tree
- *Add and remove nodes by manipulating pointers*
- **Leaf nodes** have child pointers set to null

Binary Tree … Implementation

- **Left(node):** Gives index/pointer of left child
- **Right(node):** Gives index/pointer of right child
- **Parent(node):** Returns index/pointer of parent
- **Brother(node):** Returns index/pointer of brother
- **Root:** Gives index/pointer of root node
- **Info(Node):** Data/Info stored at node
- **IsLeft(node):** Is node left child? Yes/No
- **IsRight(node):** Is node right child? Yes/No

Reading Materials

- Nell Dale Chapter#8
- Schaum's Outlines Chapter#7
- D. S. Malik Chapter#11