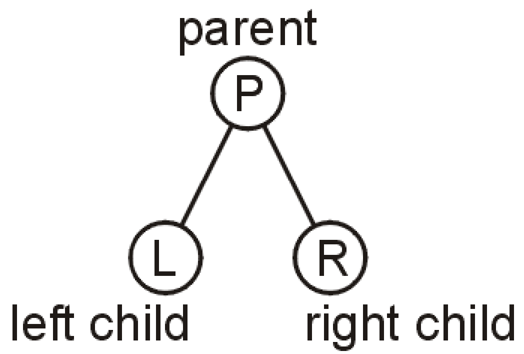
Data Structures

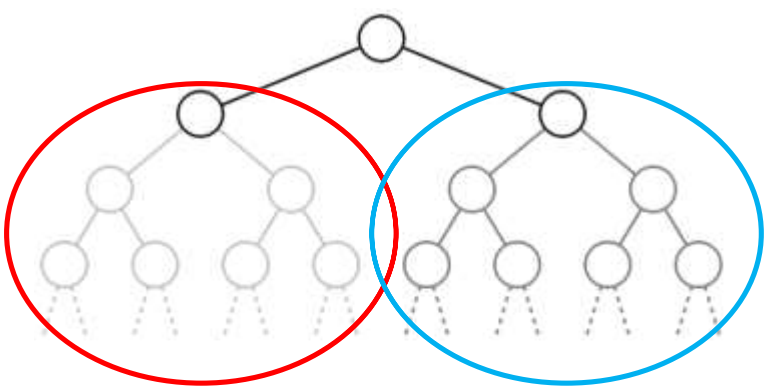
16. Binary Tree

Binary Tree

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

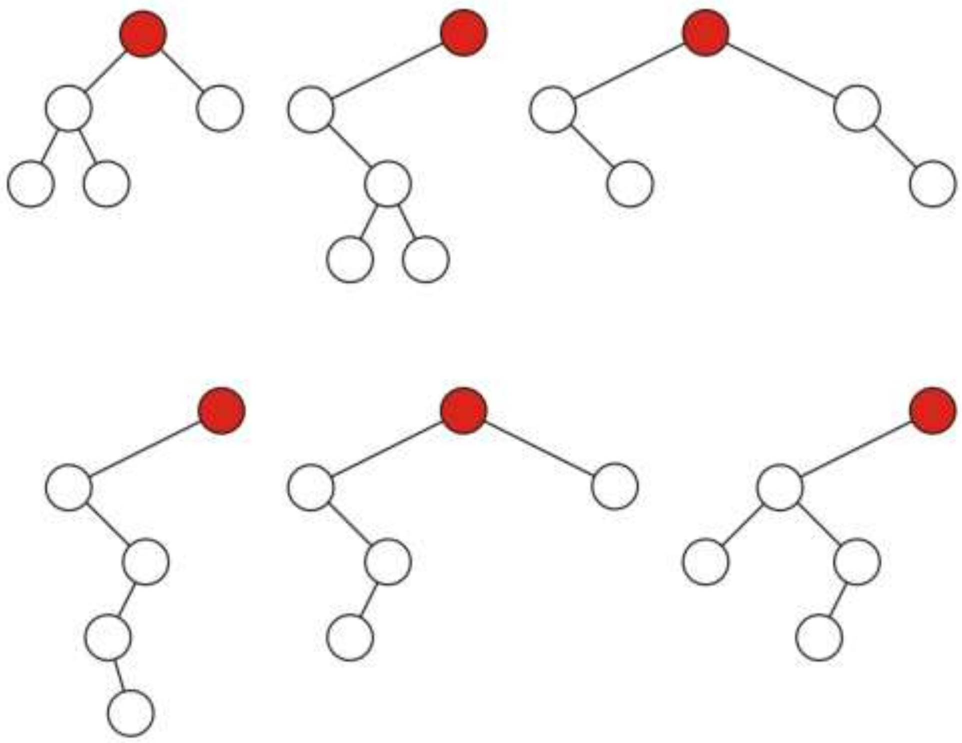


- Likewise, the two sub-trees are referred as
  - Left-hand subtree
  - Right-hand subtree



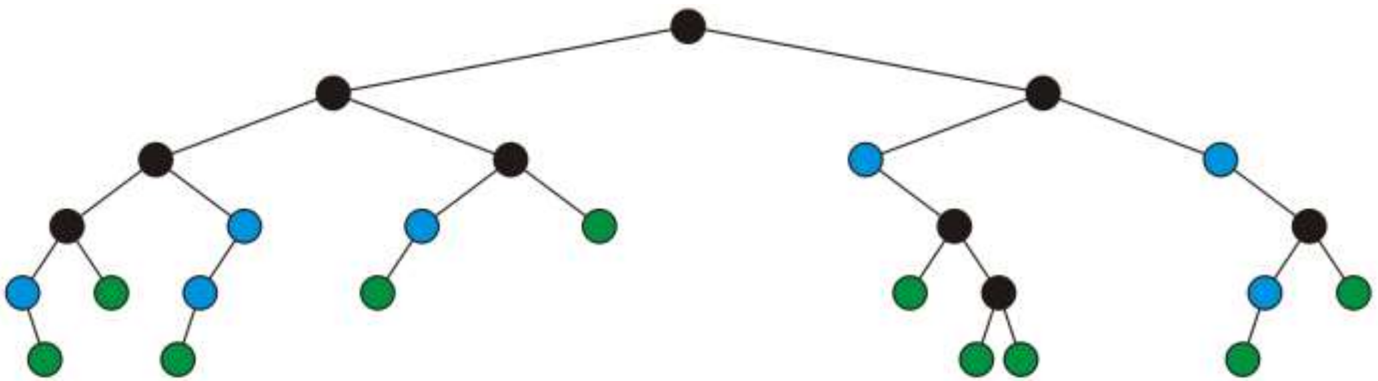
Binary Tree: Example

- Some variations on binary trees with five nodes



Binary Tree: Full Node

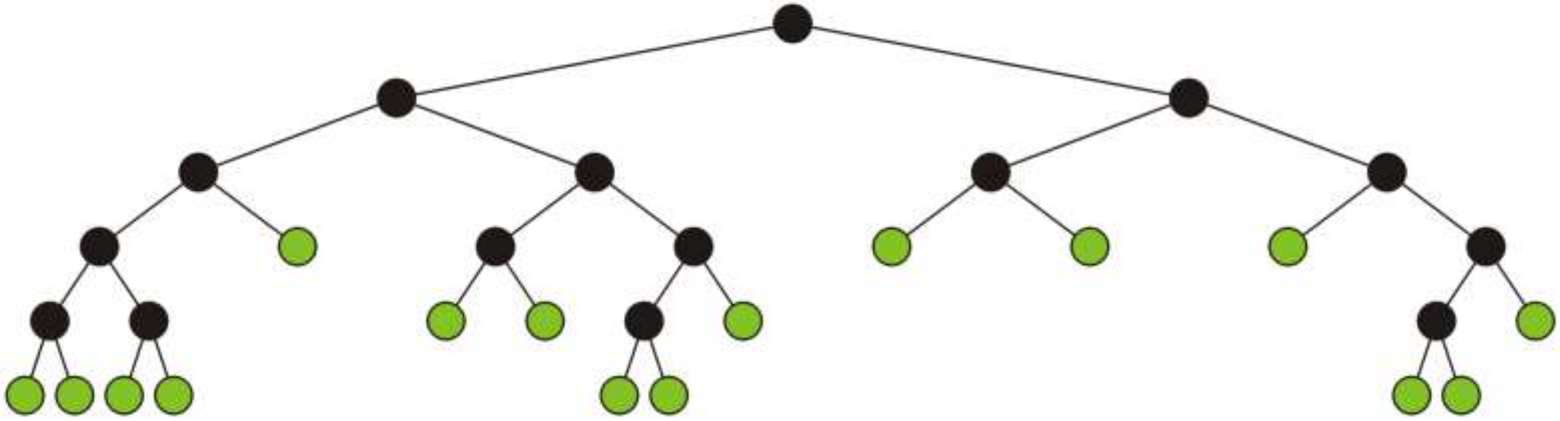
- A full node is a node where both the left and right sub-trees are non-empty trees



full nodes   neither   leaf nodes

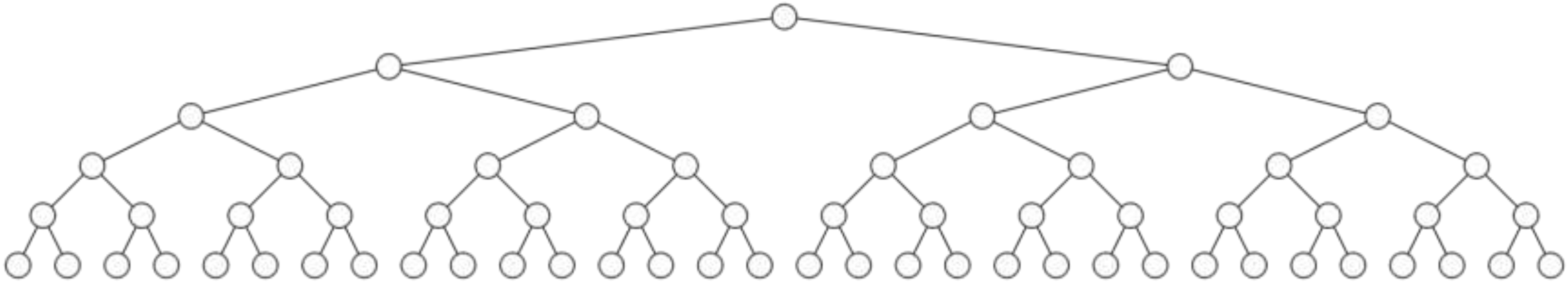
Full Binary Tree

- 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



Complete (Or 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

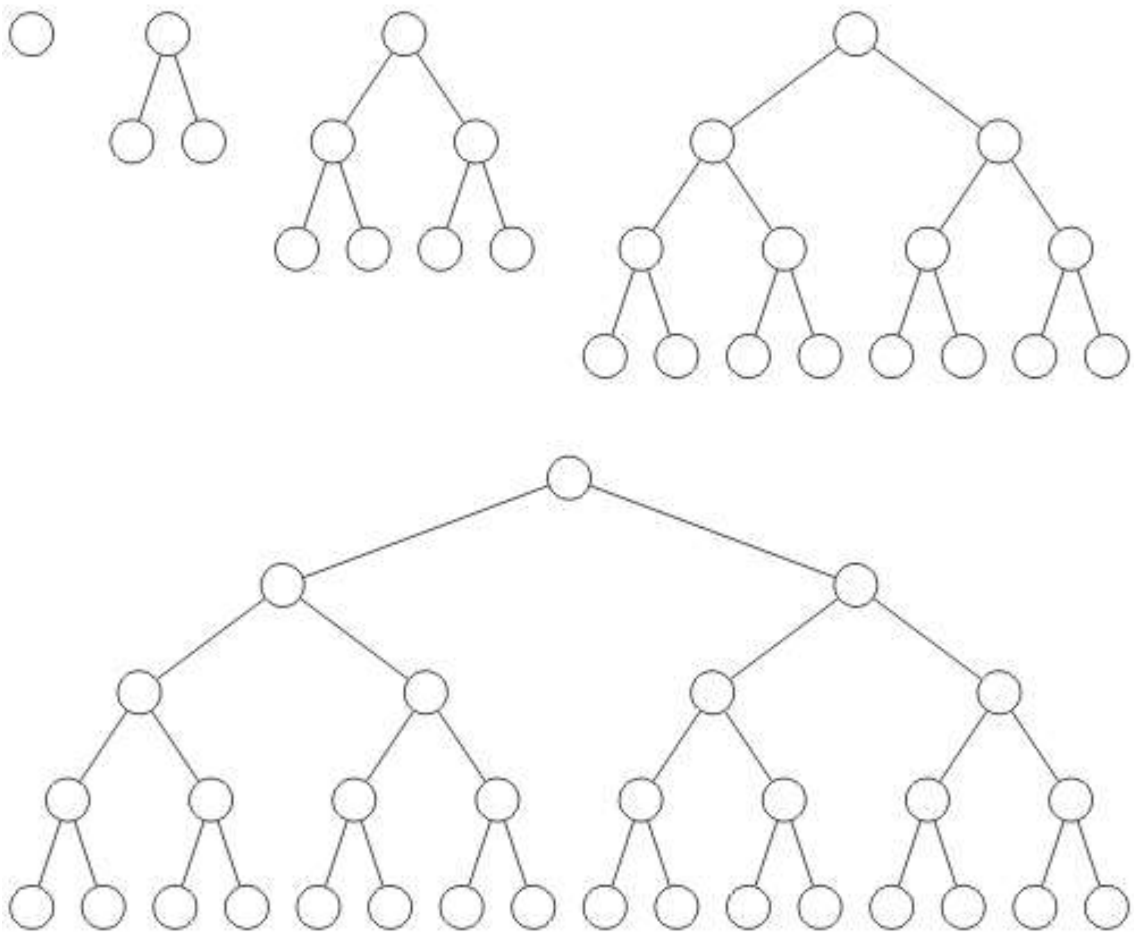


Complete Binary Tree: Recursive Definition

- A binary tree of height h = 0 is perfect
- A binary tree with height h > 0 is perfect
  - If both sub-trees are prefect binary trees of height h – 1

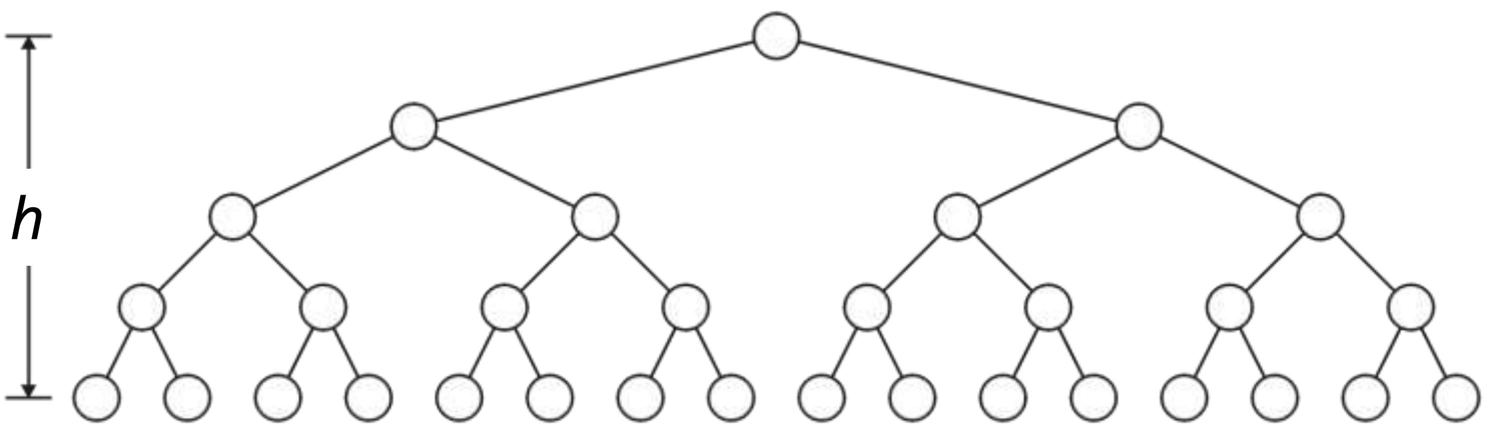
Complete Binary Tree: Example

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



Binary Tree: Properties (1)

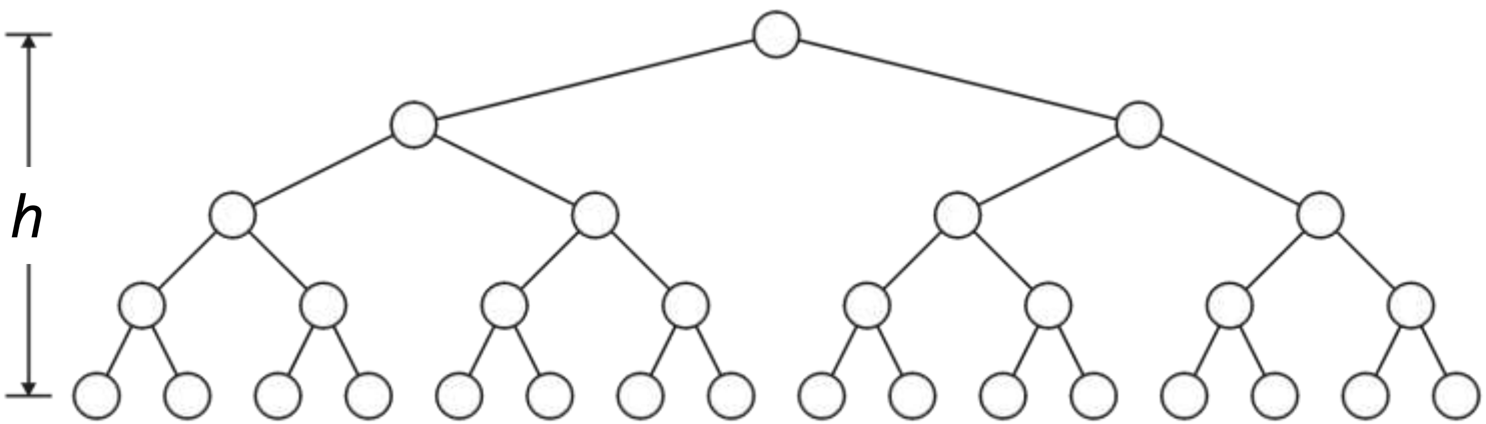
- A complete binary tree with height h has ___ leaf nodes
- Figure out the answer.
- The number of nodes in a complete binary tree with height h is: ________



Binary Tree: Properties (2)

- 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

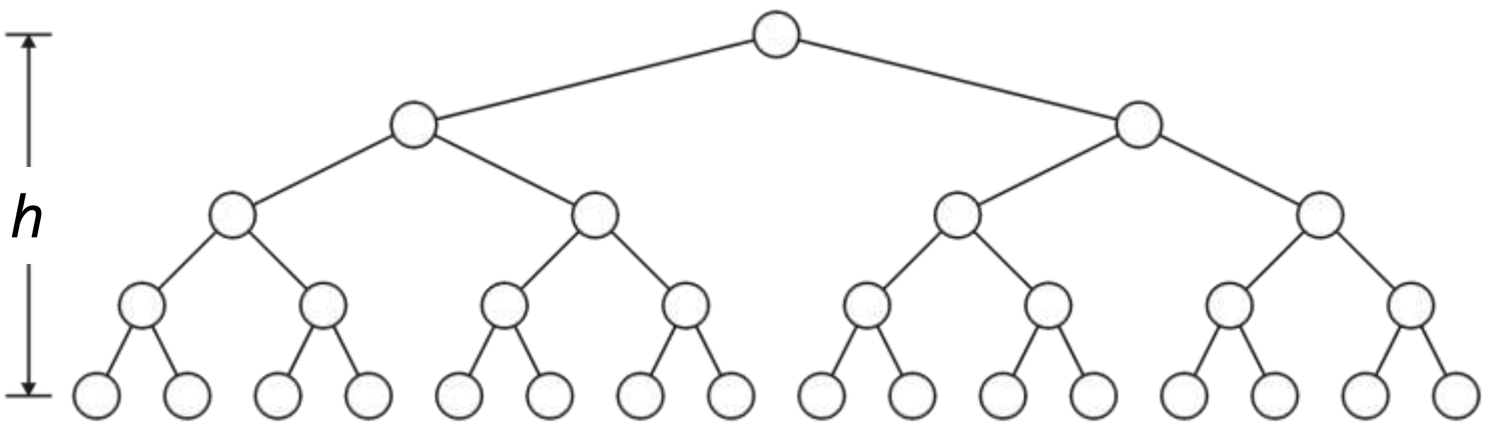
$$n = 2^0 + 2^1 + 2^2 + ... + 2^h = \sum_{j=0}^{h} 2^j = 2^{h+1} - 1$$



Binary Tree: Properties (3)

- 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: $2L-1 = 2^{h+1} - 1$



Binary Tree: Properties (4)

- 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: $2L-1 = 2^{h+1} - 1$

- A complete binary tree with n nodes has height $\log_2(n + 1) - 1$

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

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

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

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

Binary Tree: Properties (4)

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

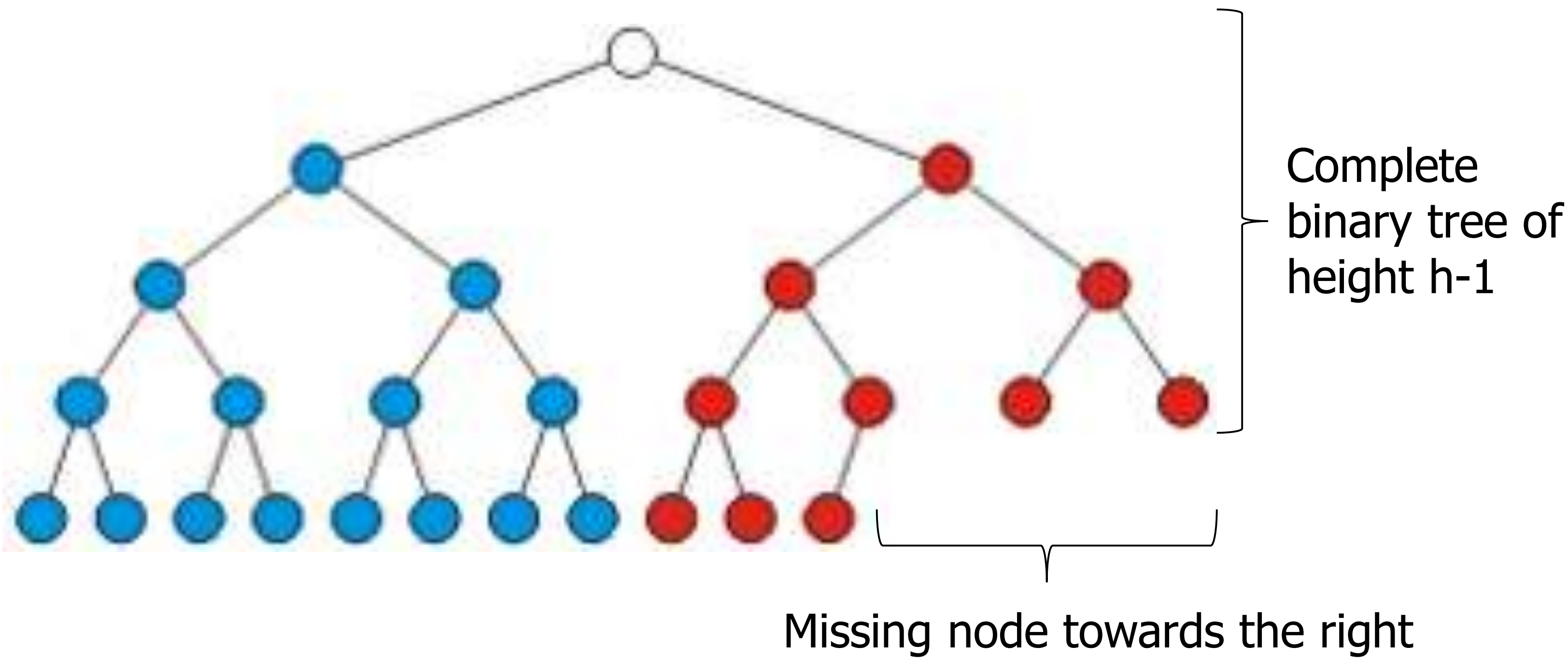
- A complete binary tree of height h has $2^{h+1} - 1$ nodes
  - Number of leaf nodes: $L = 2^h$
  - Number of internal nodes: $2^h - 1$
  - Total number of nodes: $2L-1 = 2^{h+1} - 1$

- A complete binary tree with n nodes has height $\log_2(n + 1) - 1$

- Number n of nodes in a binary tree of height h is at least $h+1$ and at most $2^{h+1} - 1$

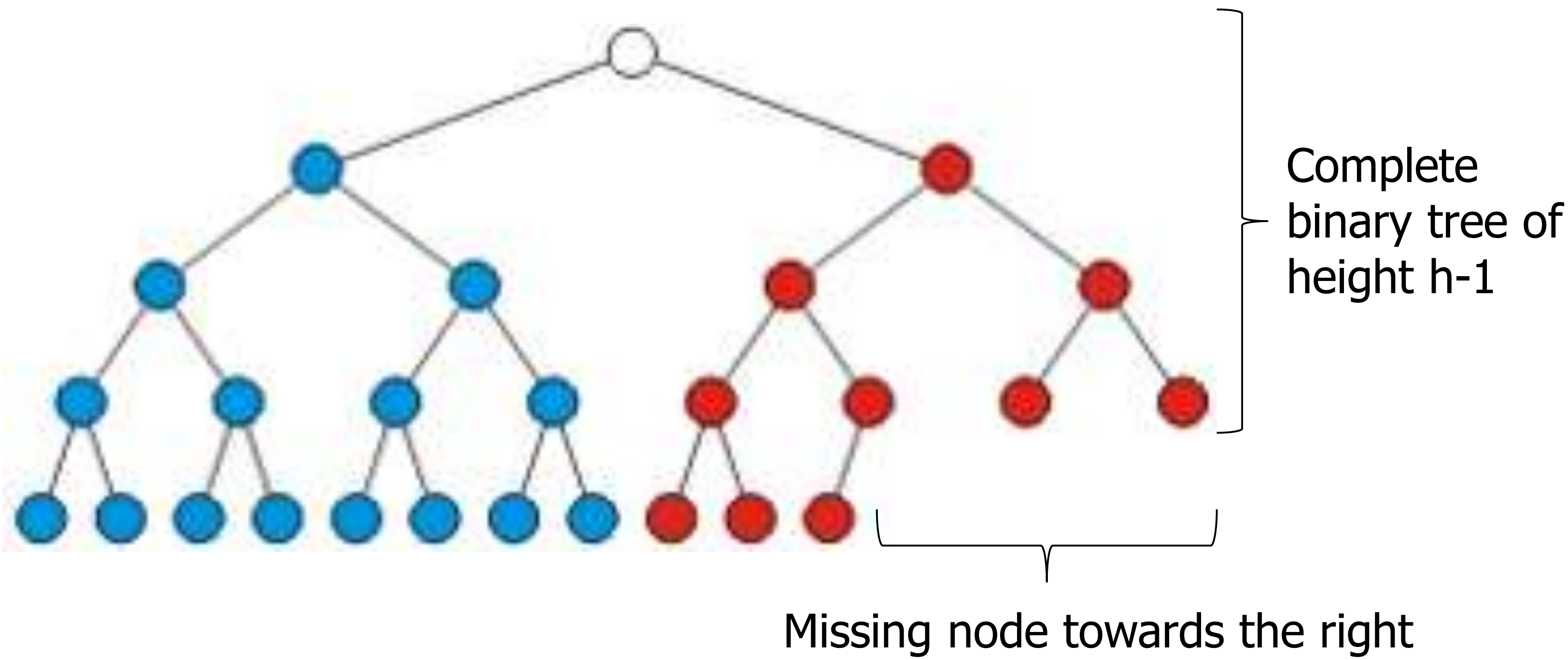
Almost (or Nearly) Complete Binary Tree

- 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,...,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 (or Nearly) Complete Binary Tree

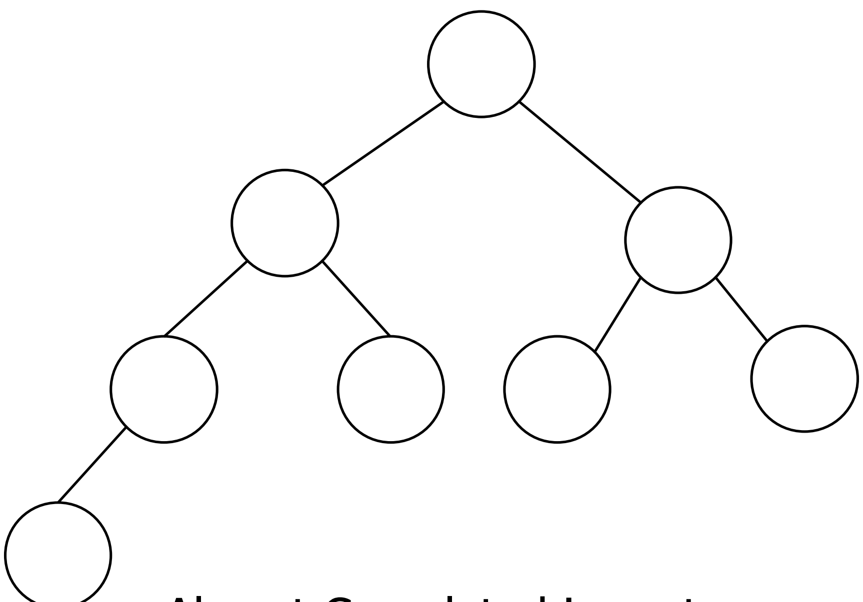
- Almost complete binary tree of height h is a binary tree in which
  1. There are 2^d nodes at depth `d` for `d = 1,2,...,h-1`
     - Each leaf in the tree is either at level `h` or at level `h- 1`
  2. **The nodes at depth h are as far left as possible (Formal ?)**



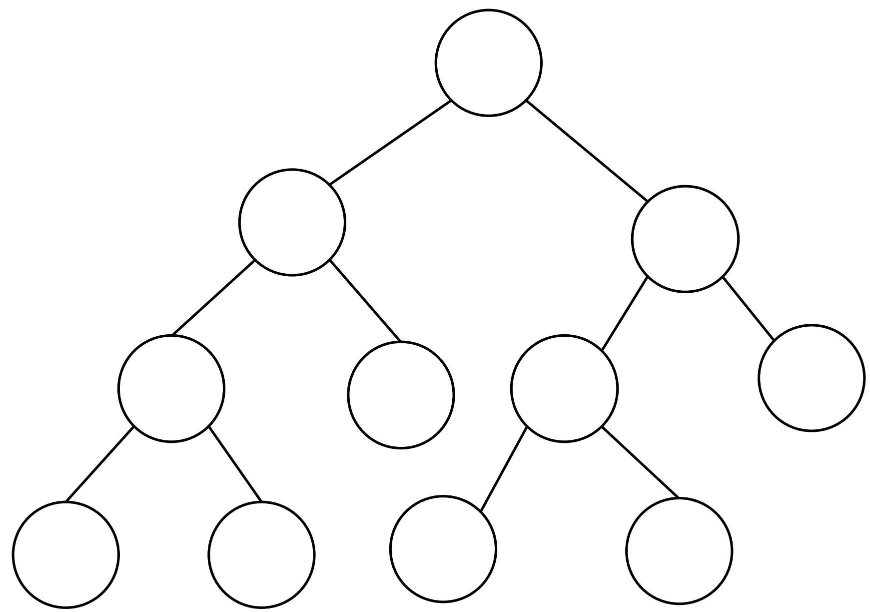
Almost (or Nearly) Complete Binary Tree

Condition 2: 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



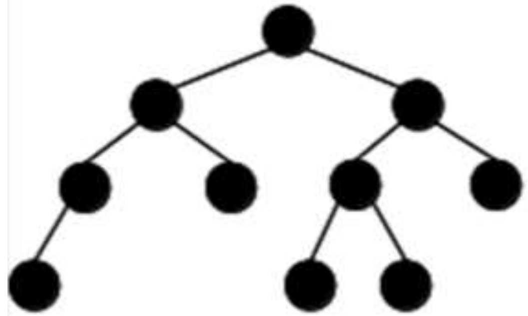
Almost Complete binary tree



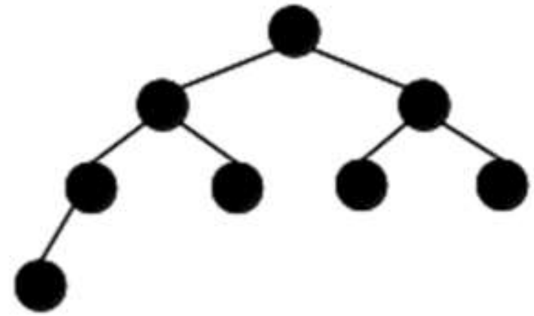
Not Almost Complete binary tree
(condition 2 violated)

Full vs. Almost Complete Binary Tree

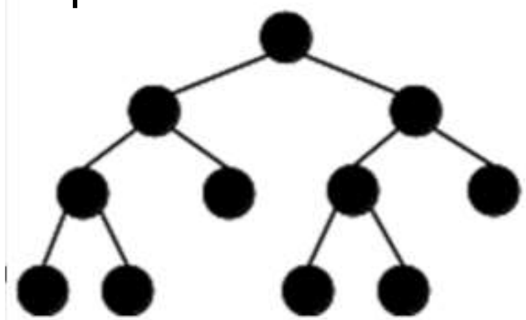
Neither Full nor Complete



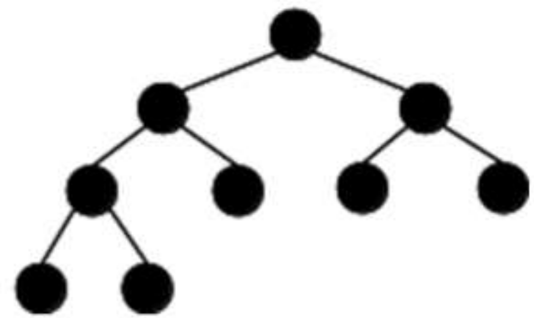
Almost Complete but not full



neither Complete nor almost Complete but Full



Full and Almost Complete



Almost Complete Binary Tree: Properties

- Total number of nodes `n` are between
  - Complete binary tree of height `h-1`, i.e., 2^h nodes
  - Complete binary tree of height `h`, i.e., $2^{h+1} - 1$ nodes

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

(Completely) Balanced Binary Tree

- Balanced binary tree
  - For each node, the difference in height of the right and left sub-trees is no more than one

- Completely balance binary tree
  - Left and right sub-trees of every node have the same height

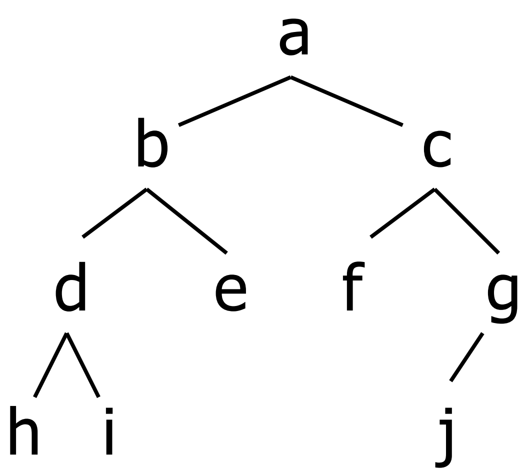
(Completely) Balanced Binary Tree

- Balanced binary tree
  - For each node, the difference in height of the right and left sub-trees is no more than one

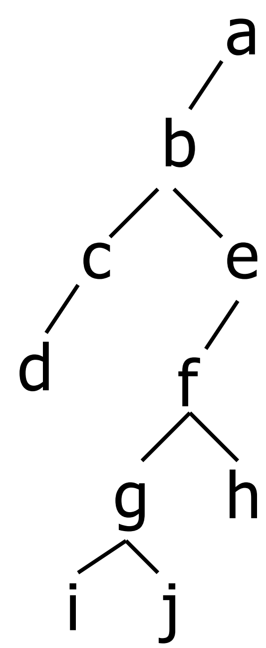
- When a binary tree is balanced?
  - If every level above the lowest is “complete”

- **Completely balance binary tree**
  - Left and right sub-trees of every node have the same height

Balanced Binary Tree: Example



A balanced binary tree



An unbalanced binary tree

Examples

- **Full & balanced** - All nodes have 0 or 2 children, `level 3 - level 2 <= 1`, (**Not complete** - last level nodes are not as far left as possible)

```
      1               --- LEVEL 0
    /   \
   1      1           --- LEVEL 1
  /\      /\
 1  1    1  1         --- LEVEL 2
 -  /\   -  -
   1  1               --- LEVEL 3
x x -  -
```

- **Full, balanced & complete** - All nodes have 0 or 2 children, `3 - 2 <= 1`, last level nodes are as far left as possible:

```
      1               --- LEVEL 0
    /   \
   1      1           --- LEVEL 1
  /\      /\
 1  1    1  1         --- LEVEL 2
/\  -    -  -
1  1                  --- LEVEL 3
-  -
```

Examples

- **Full** - All nodes have 0 or 2 children (**Unbalanced** - `3 - 1 > 1`, **Not complete** - level 1 has a node with 0 children):

```
       1              --- LEVEL 0
      / \
     1   1            --- LEVEL 1
    / \  -
   1   1              --- LEVEL 2
  / \  - x x
 1   1                --- LEVEL 3
 -   -
```

- **Complete & balanced** - Last level nodes are as far left as possible, `3 - 3 <= 1` (**Not full** - there is a level 2 node with 1 child):

```
        1             --- LEVEL 0
      /    \
     1      1         --- LEVEL 1
    /\      /\
   1  1    1  1       --- LEVEL 2
  /\  /\  /\  /x
 1 1 1  11 1 1        --- LEVEL 3
 - - -  -- - -
```

Examples

- **Balanced** - `3 - 3 <= 1`, (**Not full** - there is a level 2 node with 1 child, **Not complete** - last level nodes are not as far left as possible)

```
       1             --- LEVEL 0
     /   \
    1     1          --- LEVEL 1
   /\     /\
  1  1   1  1        --- LEVEL 2
 /\ /\   /x  /\
1 11  11   1  1      --- LEVEL 3
- --  -- x -  -
```

Any Question So Far?