Binary Tree Data Structure

261217 Data Structures for Computer Engineers

Patiwet Wuttisarnwattana, Ph.D.

patiwet@eng.cmu.ac.th

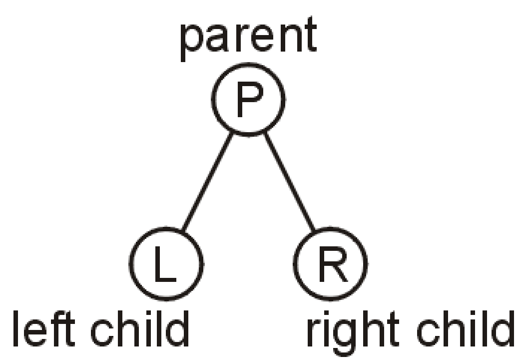
Computer Engineering, Chiang Mai University

What is Binary Tree?

Definition

A binary tree is a restriction where each node has at most two children:

- Each child is either empty or another binary tree
- This restriction allows us to label the children as *left* and *right* subtrees



Implementation is super easy

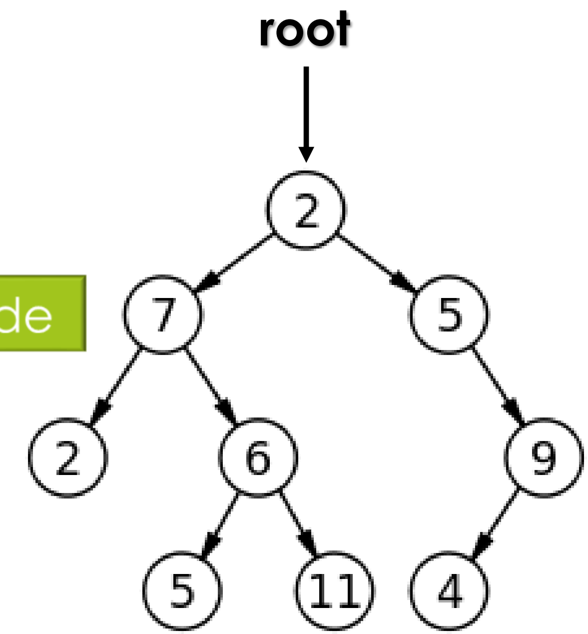
Binary Tree Implementation

class Node

Object Key;
Node left;
Node right;

Node parent; // optional

Node



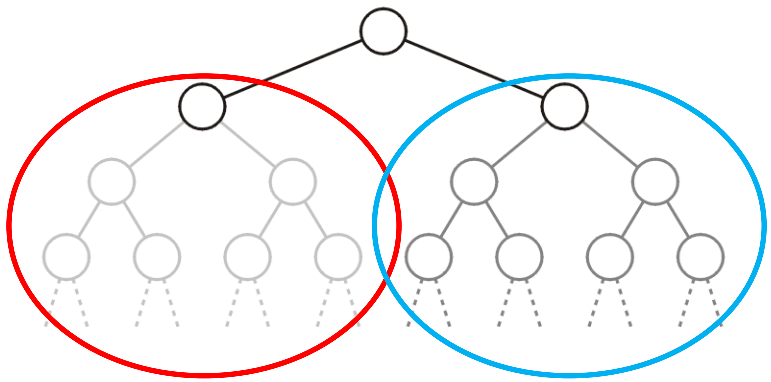
Binary Tree data structure

```
class Tree{
Node root
}
```

Sub-trees

A binary tree can have two sub-trees:

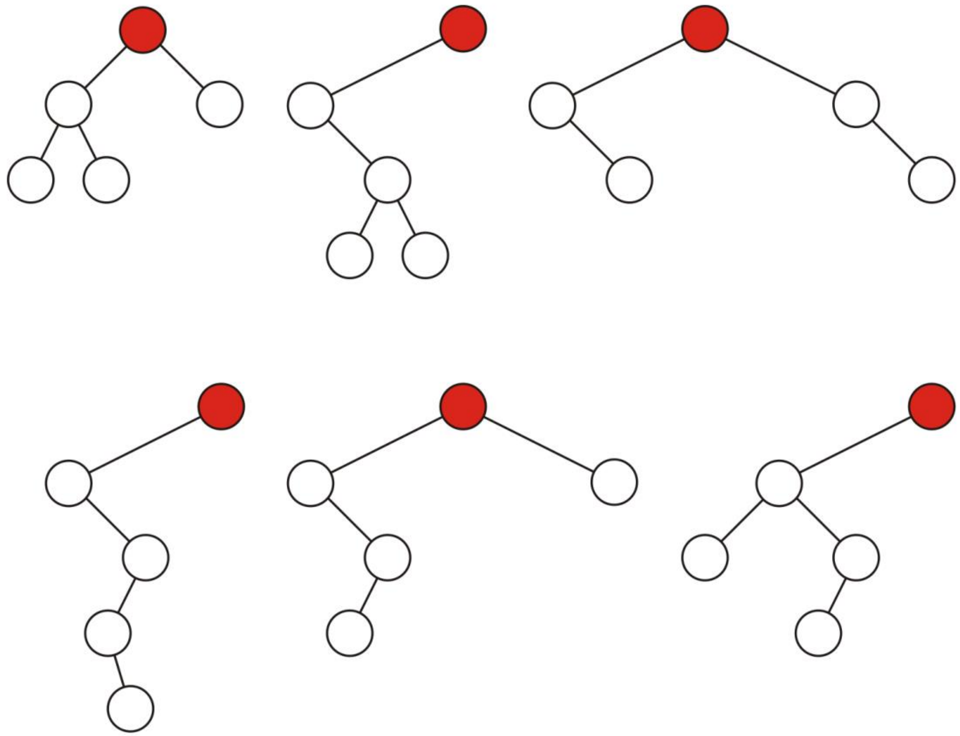
- The left-hand sub-tree, and
- The right-hand sub-tree



Binary Trees with 5 nodes

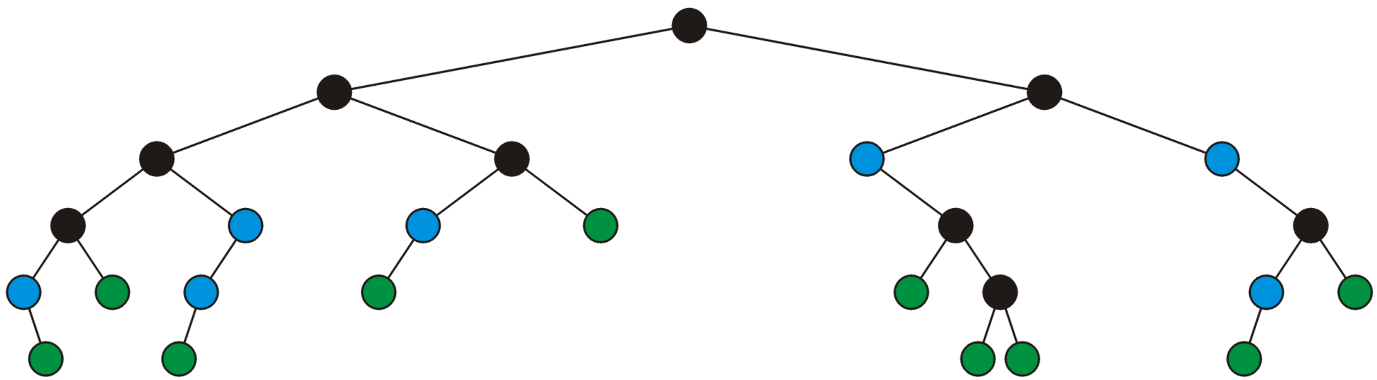
Sample variations on binary trees with five nodes:

Root node -> red node



Full nodes

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

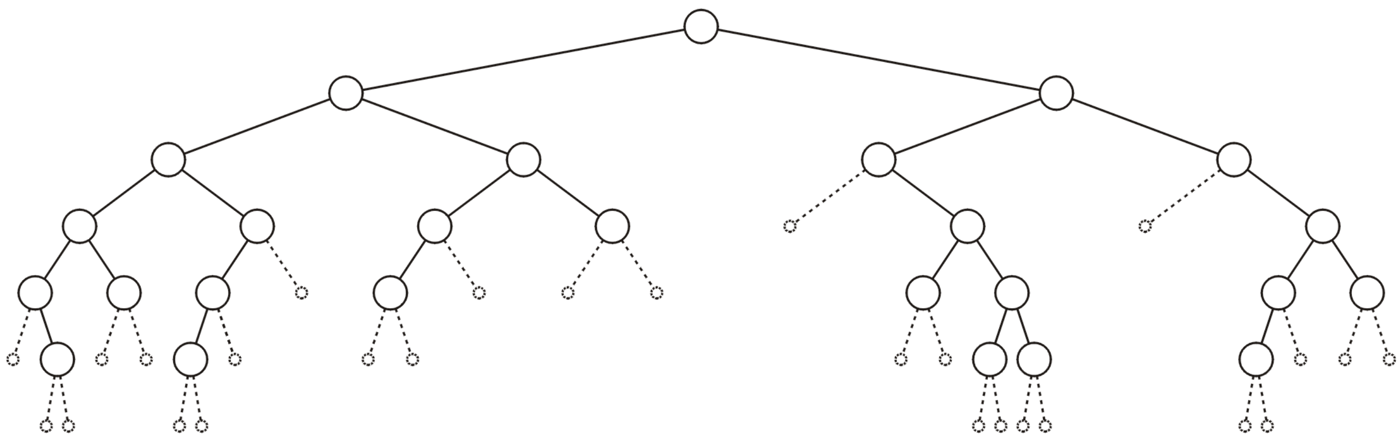


Legend:

full nodes ● neither ● leaf nodes ●

Empty node

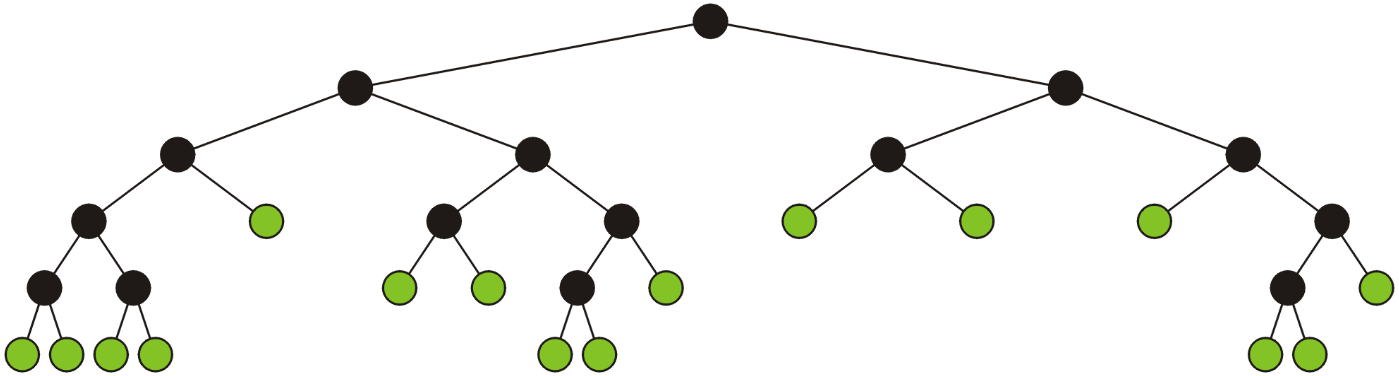
An *empty node* or a *null sub-tree* is any location where a new leaf node could be appended



Full binary tree

A *full binary tree* is where each node is:

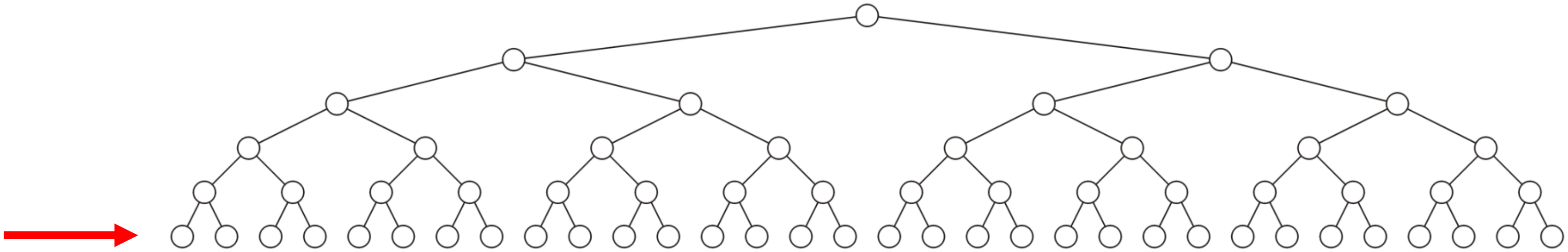
- A full node, or
- A leaf node



Perfect Binary Tree

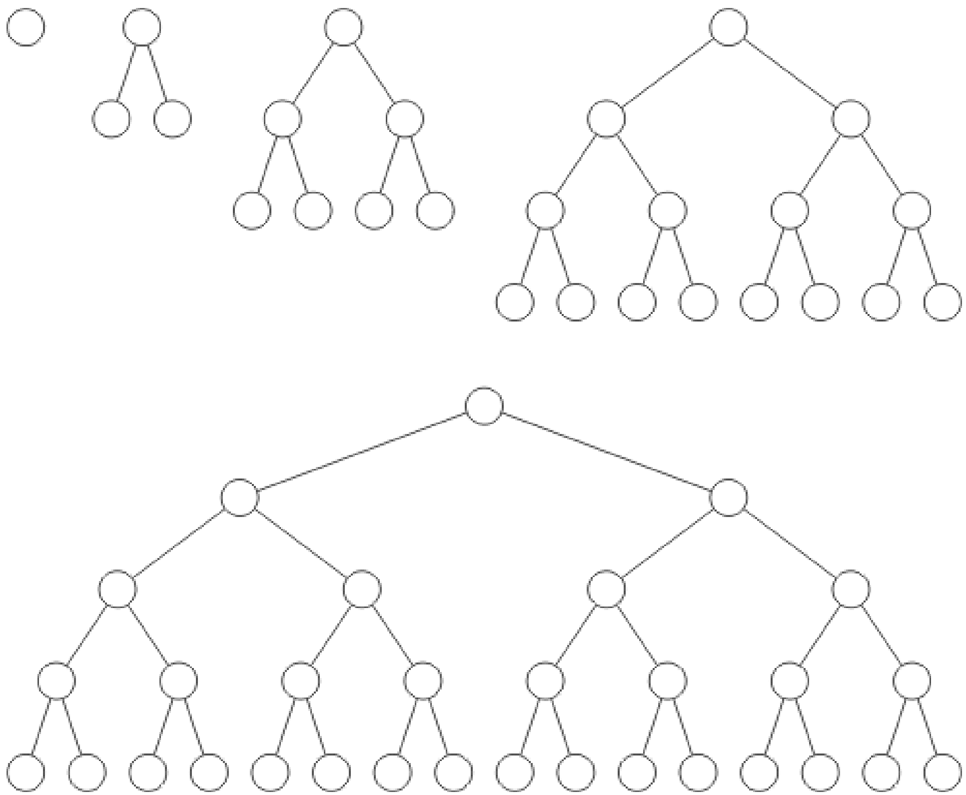
Standard definition:

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



Examples

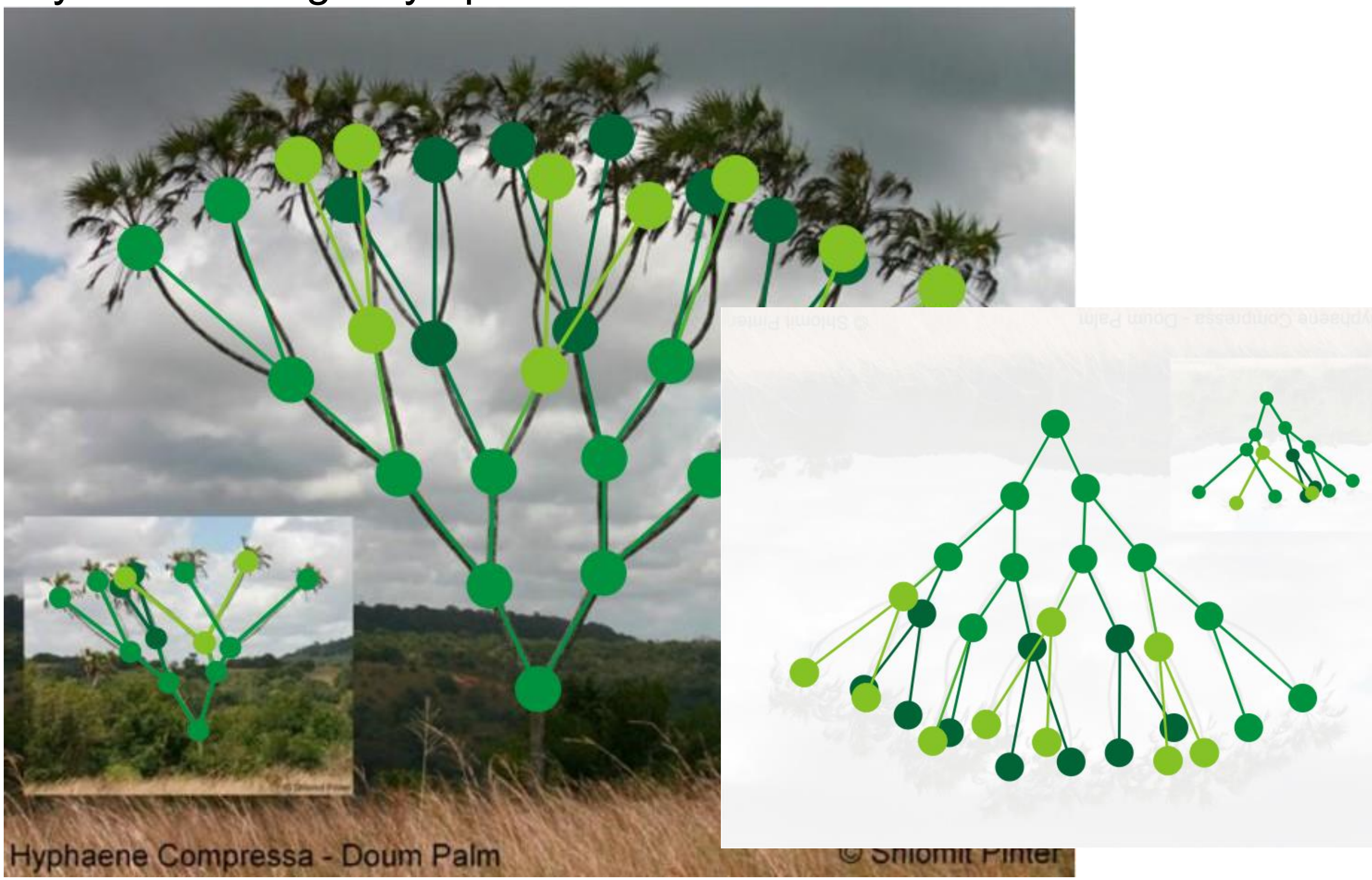
Perfect binary trees of height $h = 0, 1, 2, 3$ and 4



Examples

Perfect binary trees of height $h = 3$ and $h = 4$

– Note they're the wrong-way up…



Properties of Perfect Binary Trees

We will now look at four theorems that describe the properties of perfect binary trees:

- A perfect tree with the height h, will have $\mathbf{2^{h+1} - 1}$ nodes
- A perfect tree with n nodes, will have height $\mathbf{\log_2(n+1) - 1}$
  - A perfect tree has height $\mathbf{\Theta(\ln(n))}$
- A perfect tree with the height h, will have $\mathbf{2^h}$ leaf nodes
- The average depth of a node is $\Theta(\ln(n))$

The results of these theorems will allow us to determine the optimal run-time properties of operations on binary trees

Logarithmic Height Proof

Theorem

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

Proof

Solving $n = 2^{h + 1} - 1$ for h:

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

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

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

Logarithmic Height Proof

Lemma

$\lg(n + 1) - 1 = \Theta(\ln(n))$

Proof

$$\lim_{n\to\infty}\frac{\lg(n+1)-1}{\ln(n)}=\lim_{n\to\infty}\frac{\dfrac{1}{(n+1)\ln(2)}}{\dfrac{1}{n}}=\lim_{n\to\infty}\frac{n}{(n+1)\ln(2)}=\lim_{n\to\infty}\frac{1}{\ln(2)}=\frac{1}{\ln(2)}$$

Perfect binary tree is not practical

Searching for a key in a “binary search tree” has O(tree depth)

Thus, a perfect binary search tree guarantee to have runtime searching of $\Theta(\log_2 N)$

A perfect binary tree has ideal properties but restricted in the number of nodes: $n = 2^h - 1$

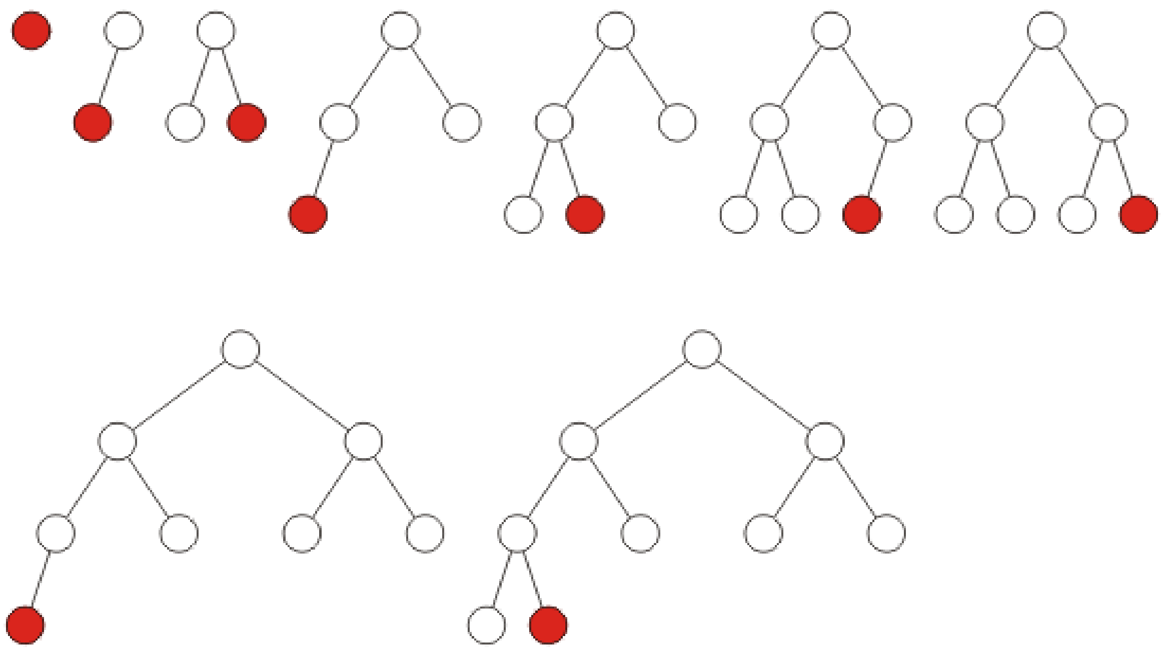
1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, ...

A perfect binary tree is not practical

Complete Binary Tree

A complete binary tree is a binary tree in which every level, except the last, is completely filled, and all nodes are as far left as possible.

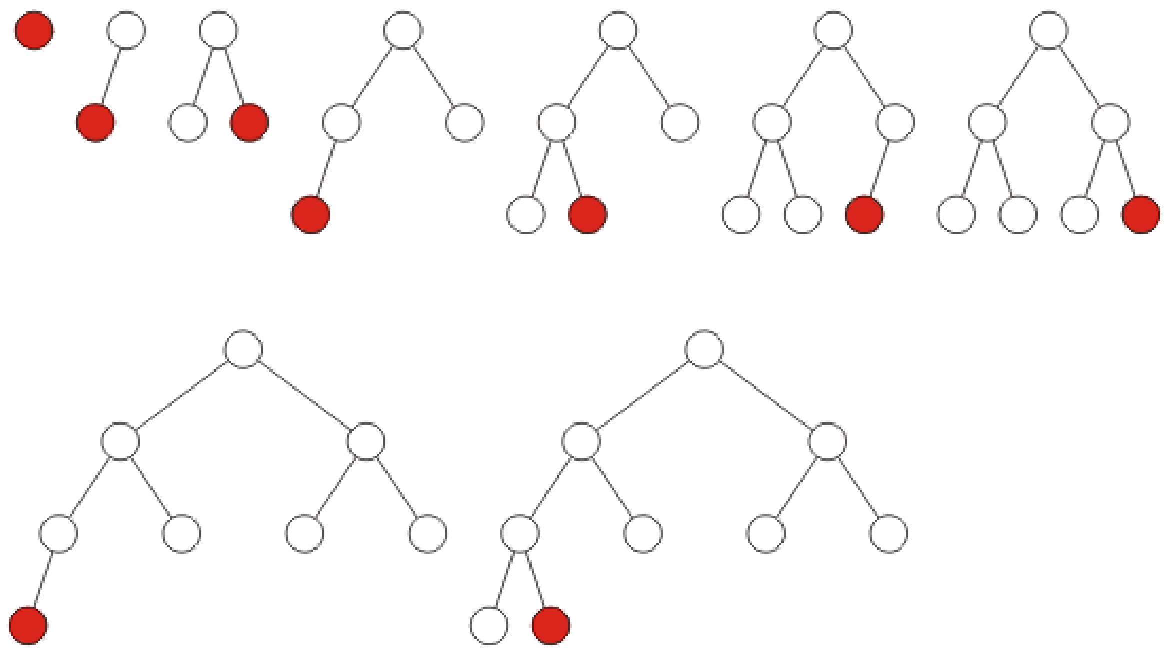
A complete binary tree filled at each depth from left to right:
(The order is identical to that of a breadth-first traversal)



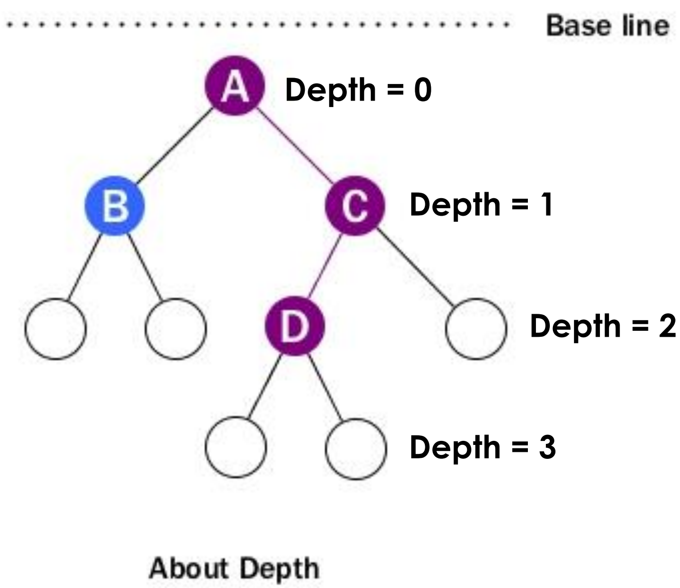
Height of Complete Binary Tree

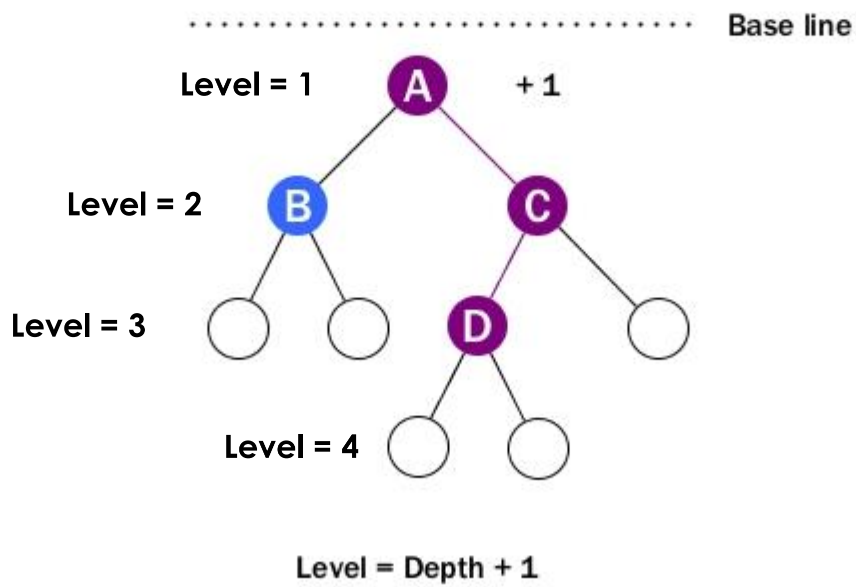
- What is the height of Complete Binary Tree with n nodes?

- $\lfloor \log_2(n) \rfloor$



Level vs Depth





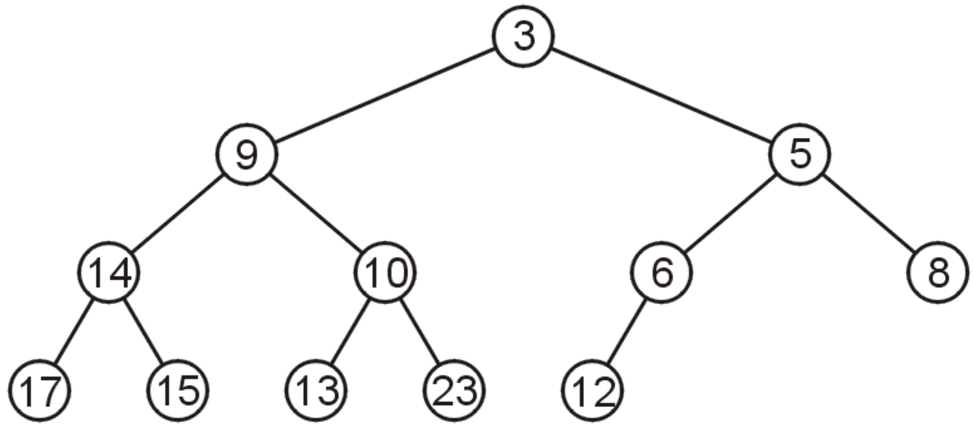
Review: What are Perfect Binary Tree, Complete Binary Tree, Full Binary Tree?

- Full Binary Tree:
  - A BT with every node is either full node or leaf node
  - Full node = a node with max children (2 children)
- Perfect Binary Tree:
  - A BT with all leave nodes have the same depth AND all the internal nodes are full
- Complete Binary Tree:
  - A BT with which every level, except the last, is completely filled, and all nodes are as far left as possible.

Array Implementation for Complete Binary Tree

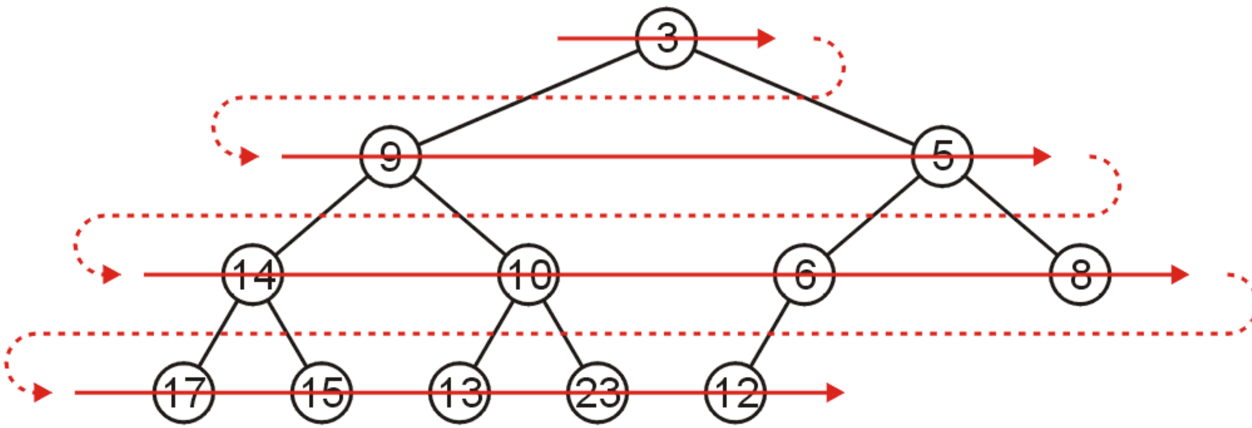
We are able to store a complete tree as an array

- Traverse the tree in breadth-first order, placing the entries into the array



Array Implementation for Complete Binary Tree

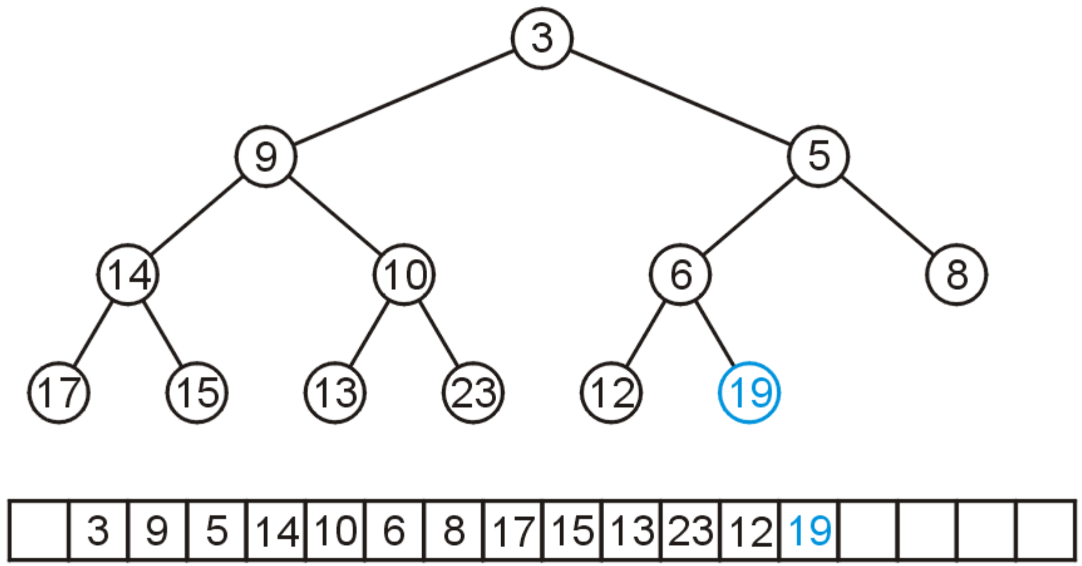
We can store this in an array after a quick traversal:



	3	9	5	14	10	6	8	17	15	13	23	12					

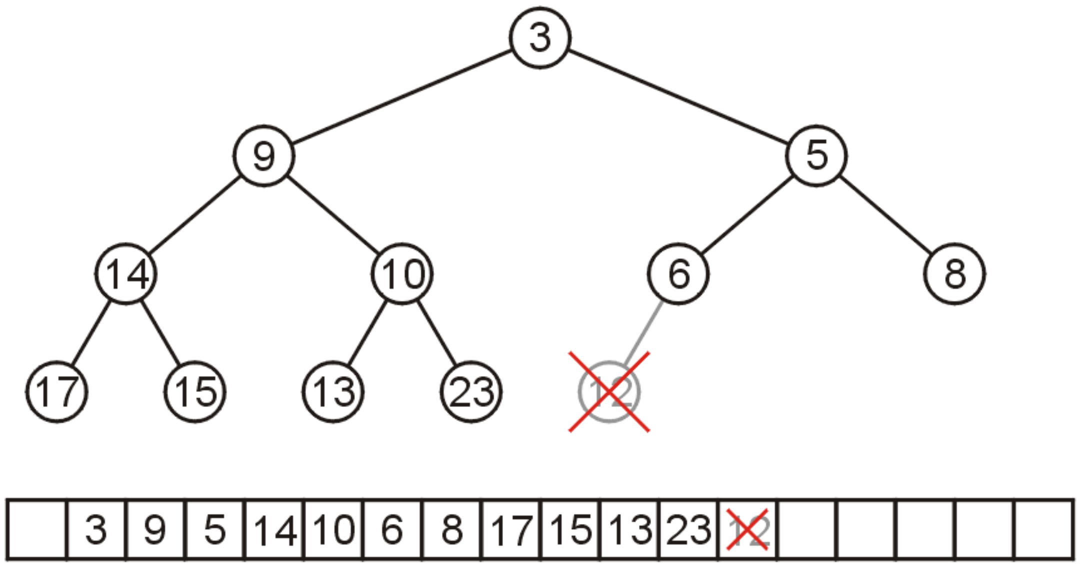
Array Implementation for Complete Binary Tree

To insert another node while maintaining the complete-binary-tree structure, we must insert into the next array location



Array Implementation for Complete Binary Tree

To remove a node while keeping the complete-tree structure, we must remove the last element in the array



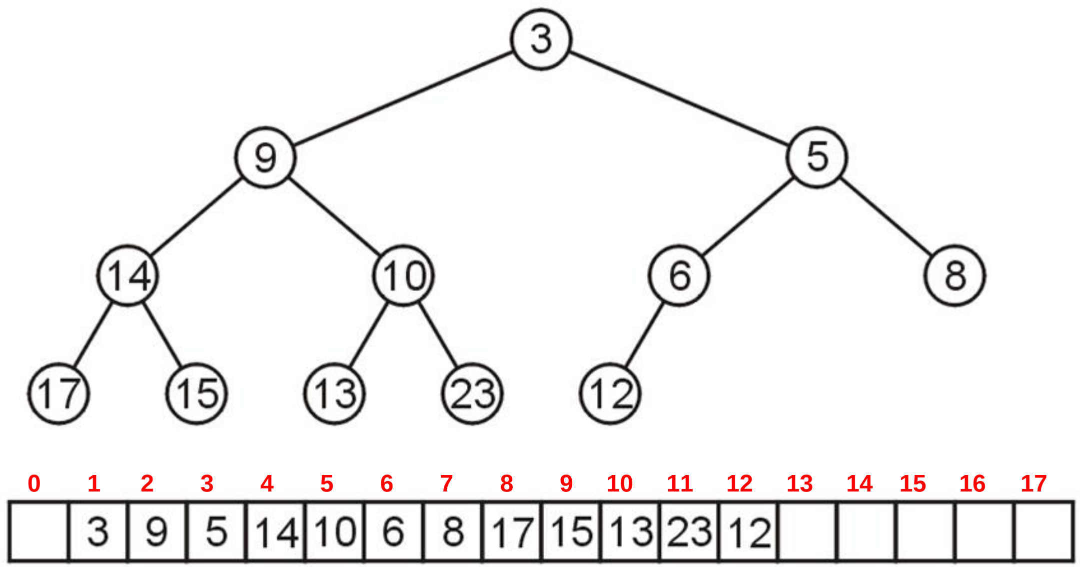
Array Implementation for Complete Binary Tree

Leaving the first entry blank yields a bonus

– The left child of a node with index k is indexed at $2k$

– The right child is indexed at $2k + 1$

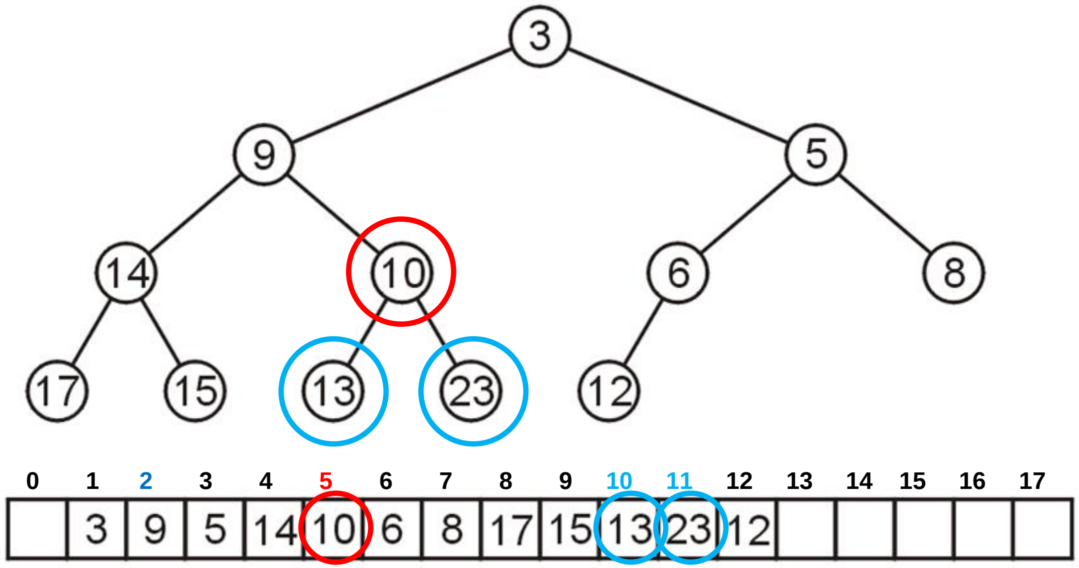
– The parent is indexed at $\text{floor}(k \div 2)$



Array Implementation for Complete Binary Tree

For example, node 10 has index 5:

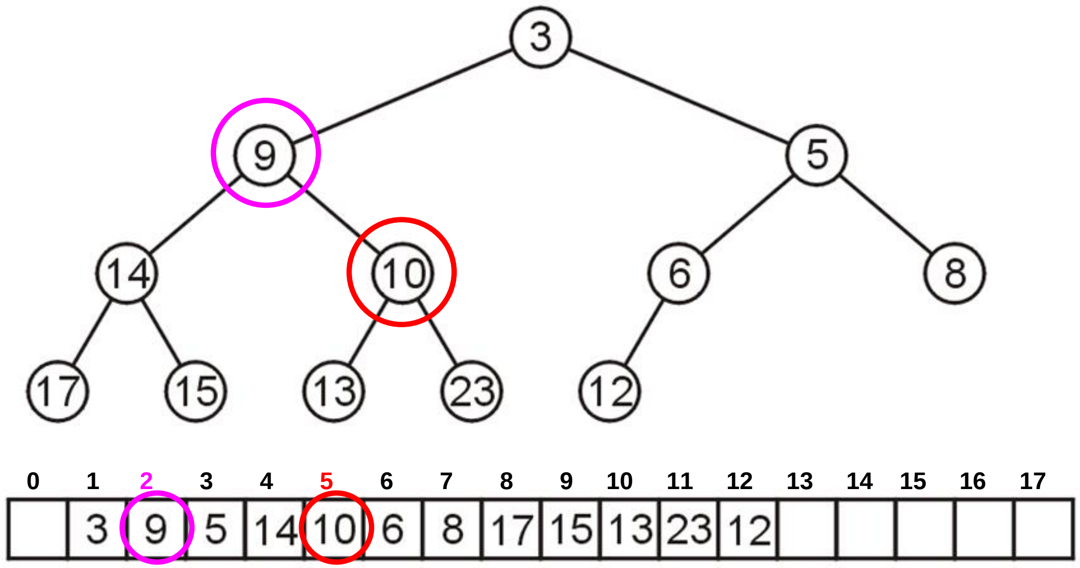
- Its children 13 and 23 have indices 10 and 11, respectively



Array Implementation for Complete Binary Tree

For example, node 10 has index 5:

- Its children 13 and 23 have indices 10 and 11, respectively
- Its parent is node 9 with index 5/2 = **2**



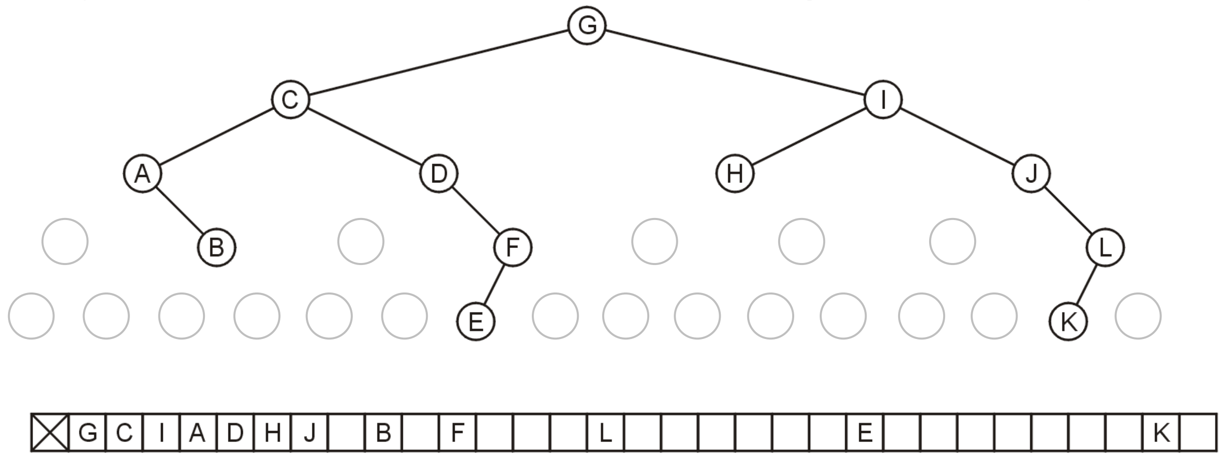
Array Implementation for Any Tree?

Question: why not store any tree as an array using breadth-first traversals?

- There is a significant potential for a lot of wasted memory

Consider this tree with 12 nodes would require an array of size 32

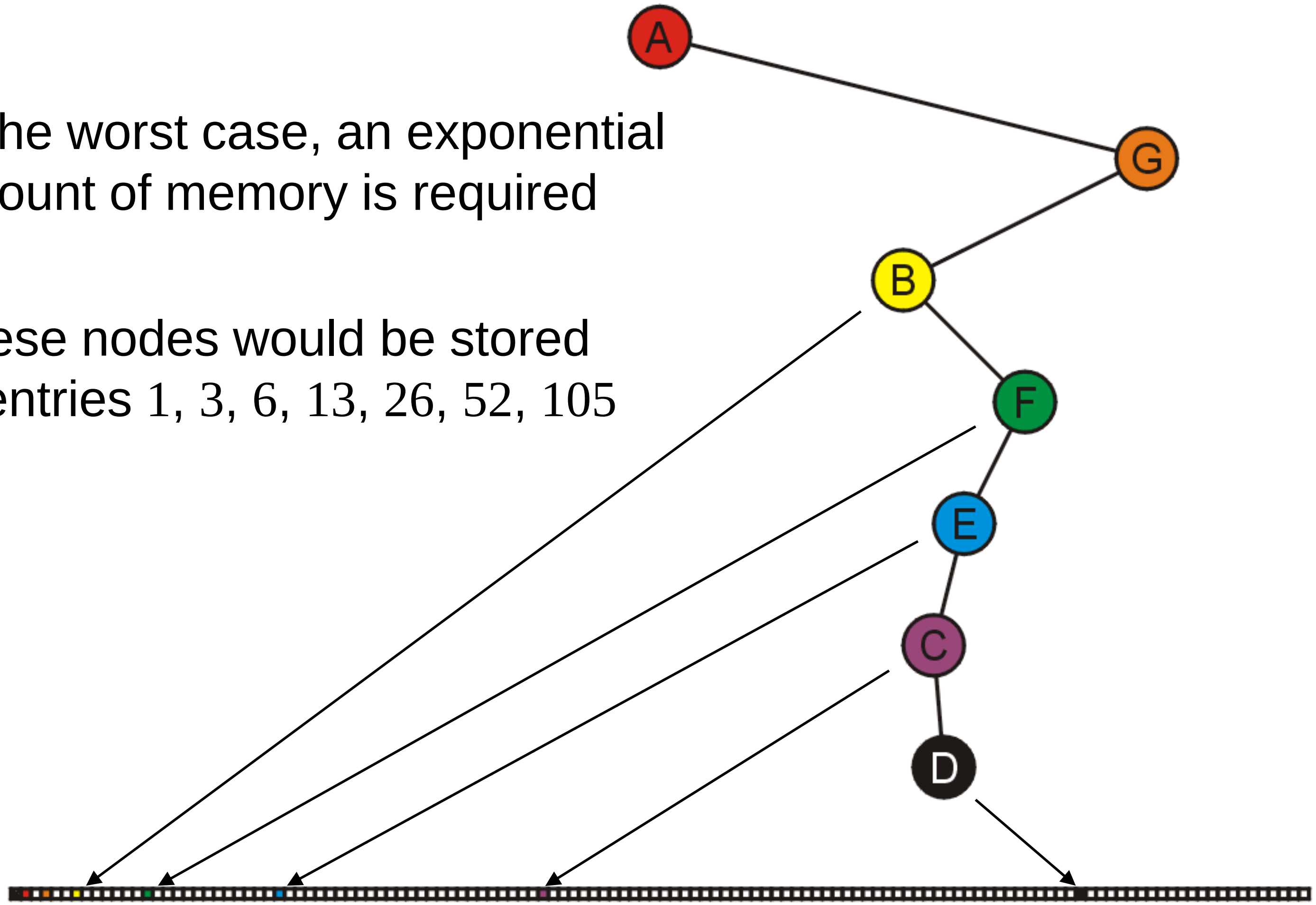
- Adding a child to node K doubles the required memory



Array Implementation for Any Tree?

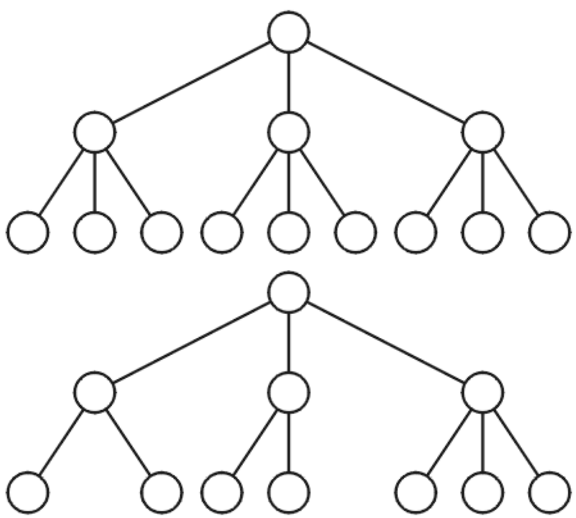
In the worst case, an exponential amount of memory is required

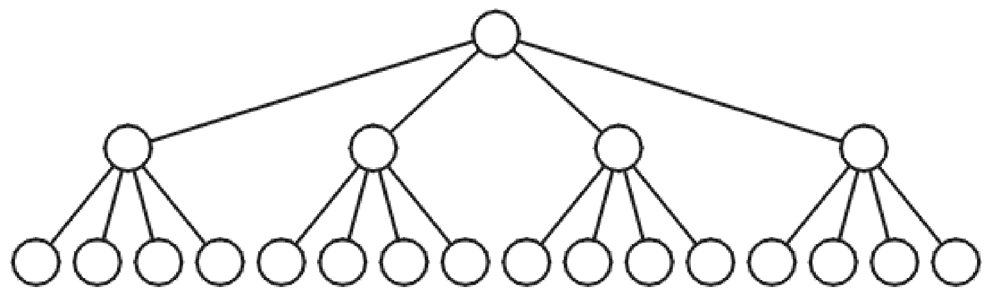
These nodes would be stored in entries 1, 3, 6, 13, 26, 52, 105



N-ary Trees

- N-aray tree is a tree that a node can have at most N children
- Examples of a ternary (3-ary) trees and quaternary tree (4-ary) tree





Perfect N-ary Trees

Each node can have N children

The number of nodes in a perfect N-ary tree of height h is

$$1 + N + N^2 + N^3 \cdots + N^h$$

This is a geometric sum, and therefore, the number of

nodes is $$n = \sum_{k=0}^{h} N^k = \frac{N^{h+1} - 1}{N - 1}$$

Solving this equation for h, a perfect N-ary tree with n nodes has a height given by

$$h = \log_N\left(n(N-1)+1\right) - 1$$

Complete N-ary Trees

A complete N-ary tree with n nodes has height

$$h = \left\lfloor \log_N \left((N-1)n \right) \right\rfloor$$

Like complete binary trees, complete N-ary trees can also be stored efficiently using an array:

- Assume the root is at index 0
- The parent of a node with index k is at location $\left\lfloor \frac{k-1}{N} \right\rfloor$
- The children of a node with index k are at locations

$$kN + j$$

for $j = 1, \ldots, N$