Algorithms & Data Structures I
CSC 225

Ali Mashreghi

Fall 2018

Department of Computer Science, University of Victoria

Data Structure

Data Structure

A **data structure** is a **way** to **store** and **organize data** in order to facilitate **access** and **modifications**.

- Each data structure has its own strength and weaknesses, and there is no data structure that is good for all purposes.

Arrays

- An **array** is a data structure which **stores data** as a number of elements in a specific order.
- Using an **index**, each element can be **accessed** and **modified** in **constant time**.
- Usually, arrays **allocate contiguous memory** words for the elements of arrays which allows for **fast processing time** of successive elements.

Arrays

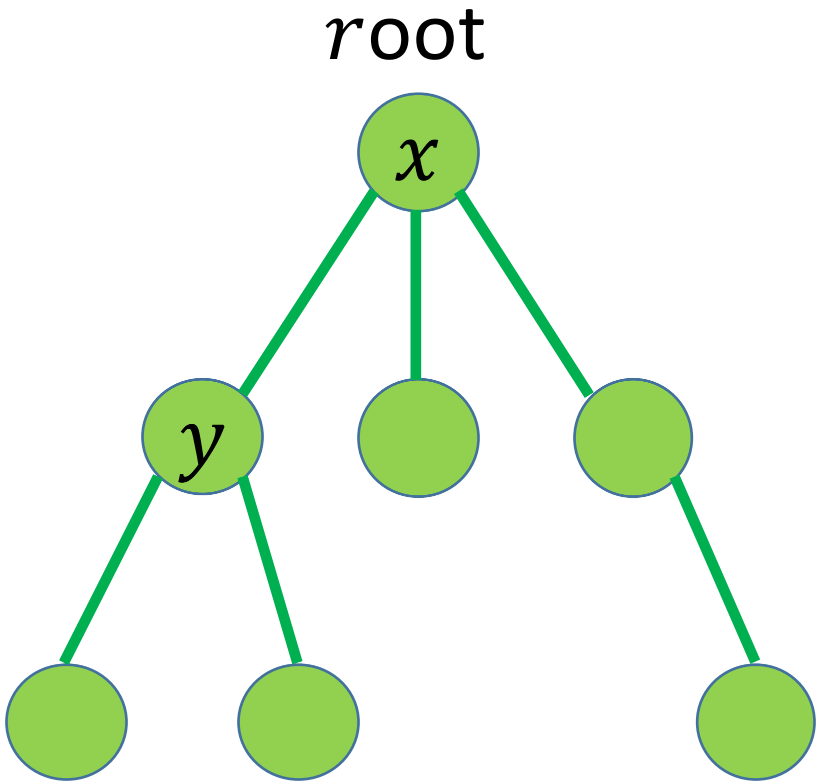
- The downside is that we **cannot adjust their size** in the middle of the execution.

- Another is that if the elements are being dynamically updated **keeping track of max and min is hard**. (needs $\boldsymbol{O(n)}$ time)

Trees

- A **(rooted) tree** T is a data structure which **stores data** in a **parent-child relationship**

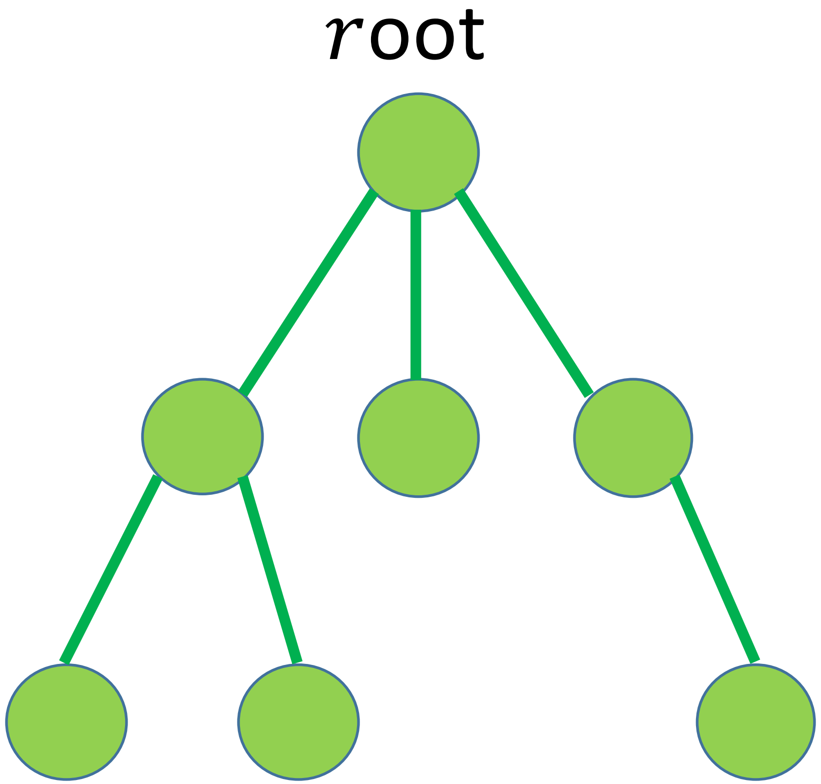
- If x is a parent of y, y is a child of x



Trees

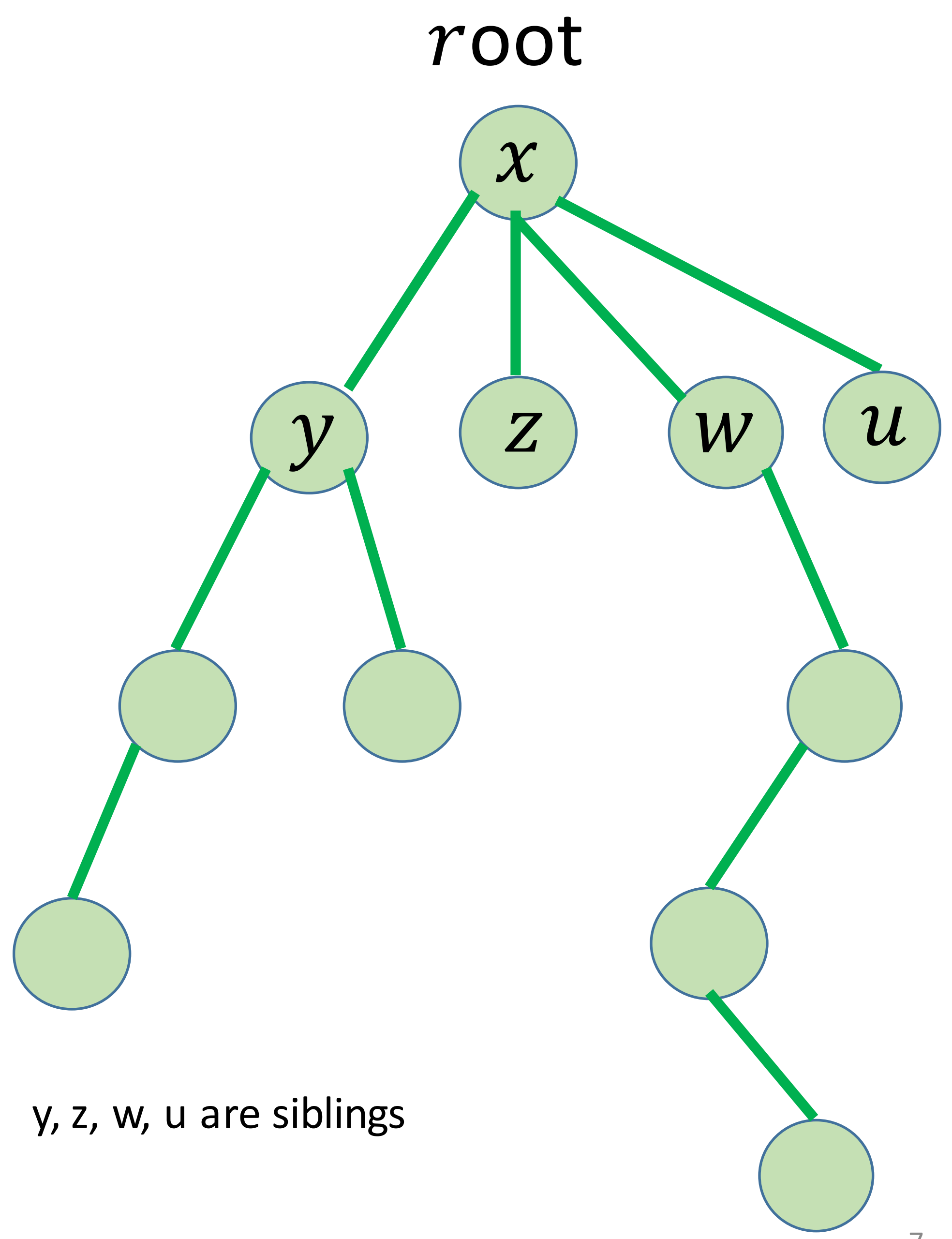
- T has a special node called the **root** of T

- Root has no parent, and every other node has **exactly one parent**



Trees

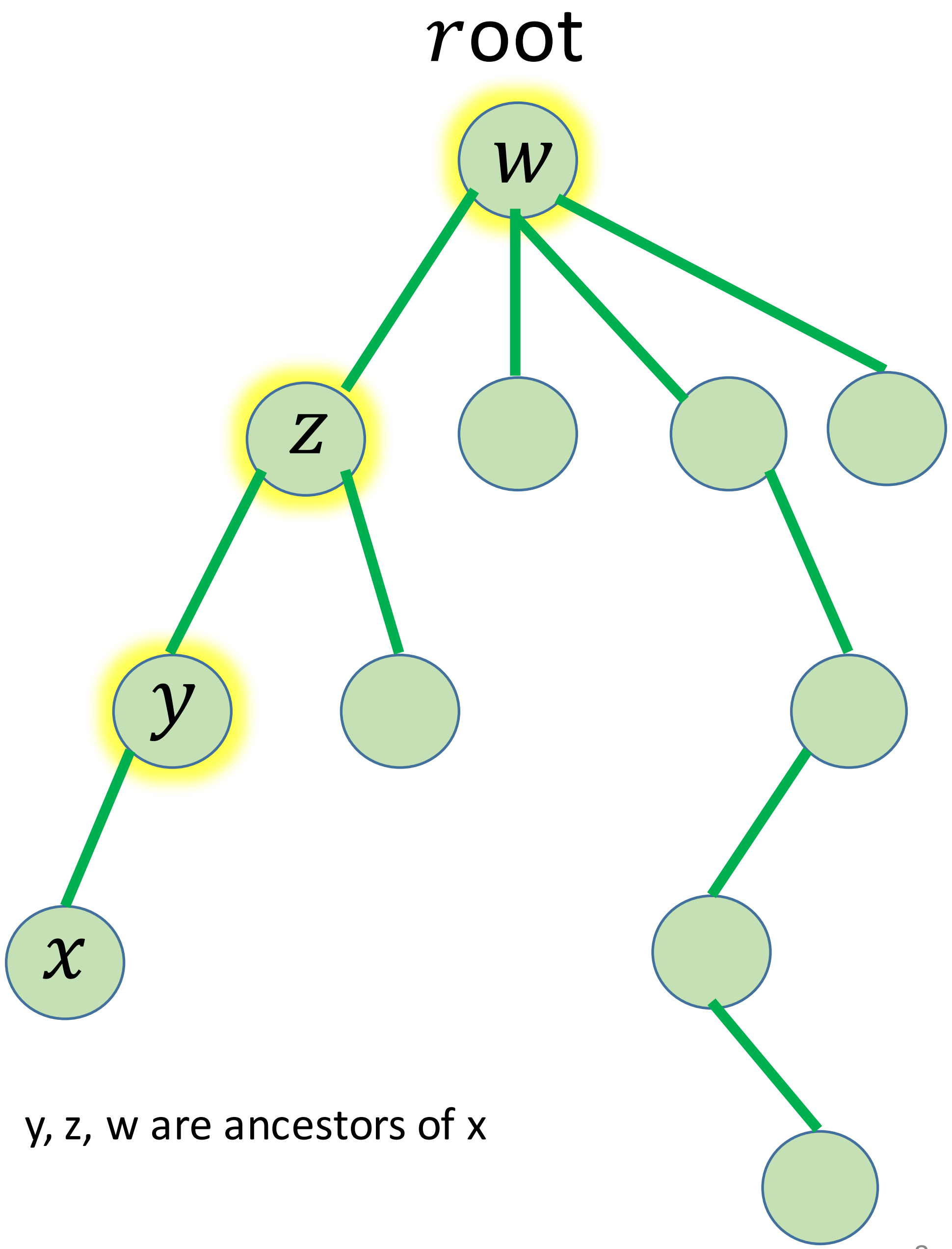
Siblings: Nodes that have the **same parent**



y, z, w, u are siblings

Trees

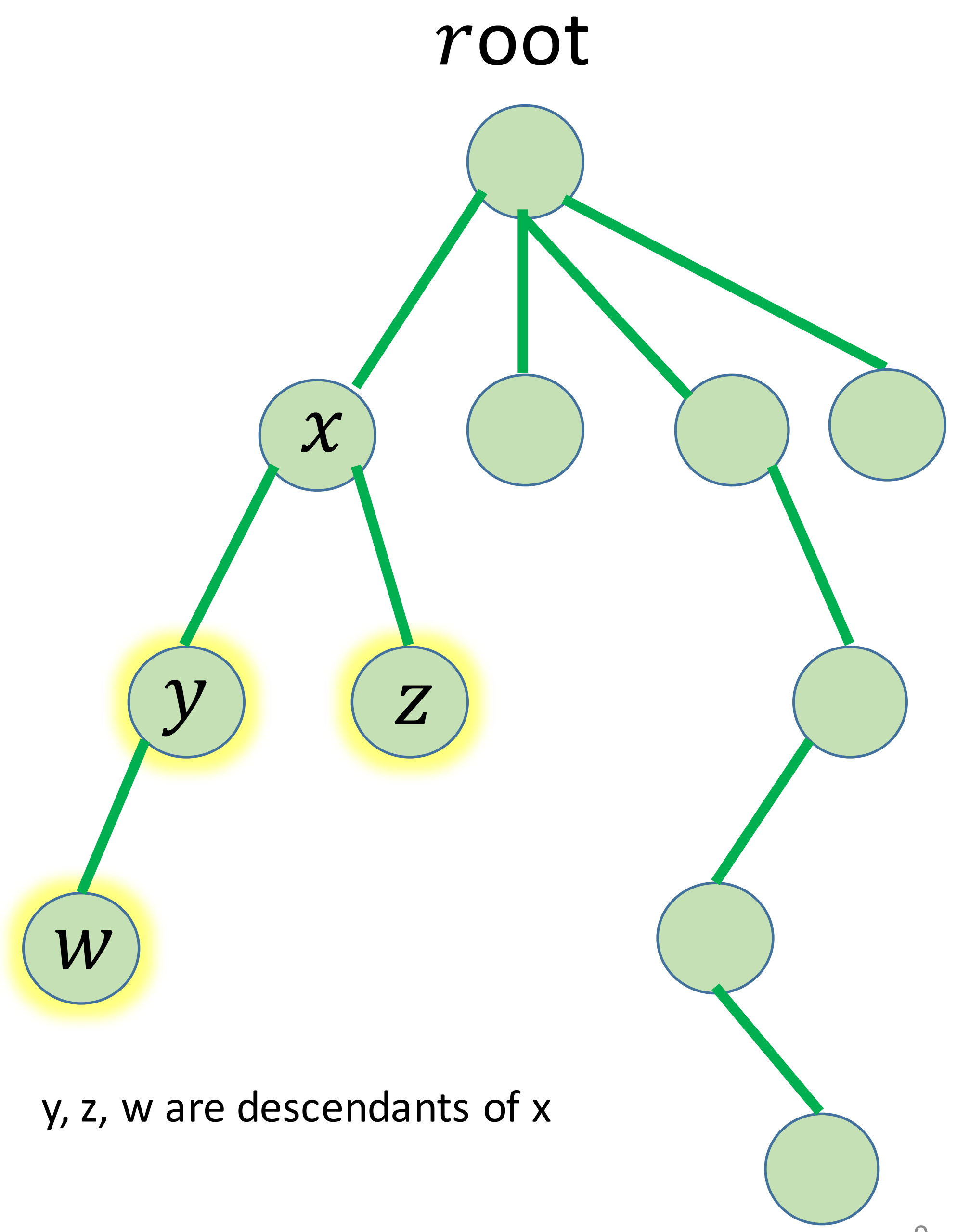
Ancestors: Ancestors of a node x are nodes on the path from x to root (excluding x)



y, z, w are ancestors of x

Trees

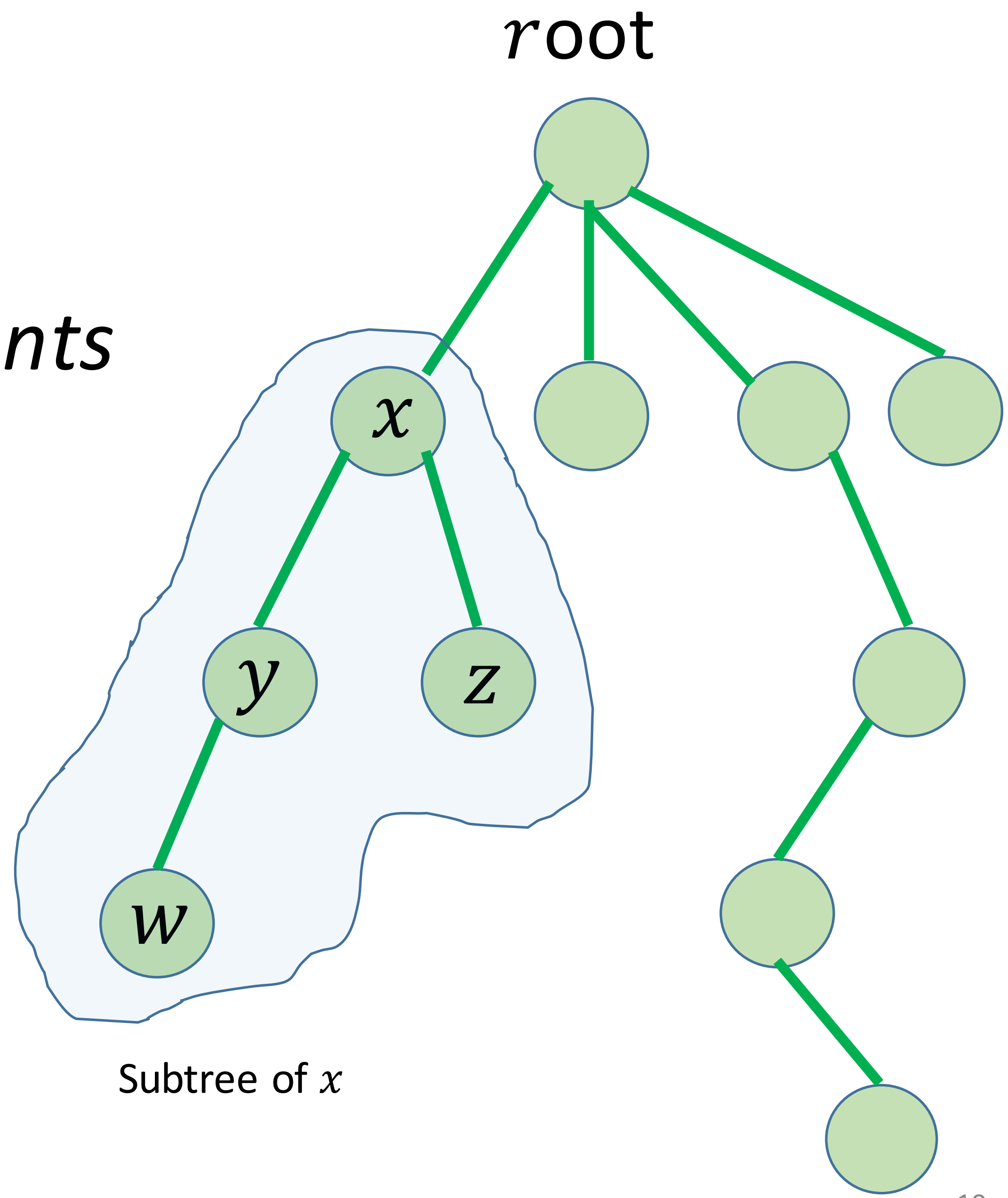
Descendants: Descendants of a node x are all nodes in the subtrees of x's children



Trees

Subtree of a node x:

Node x *and all of its descendants*

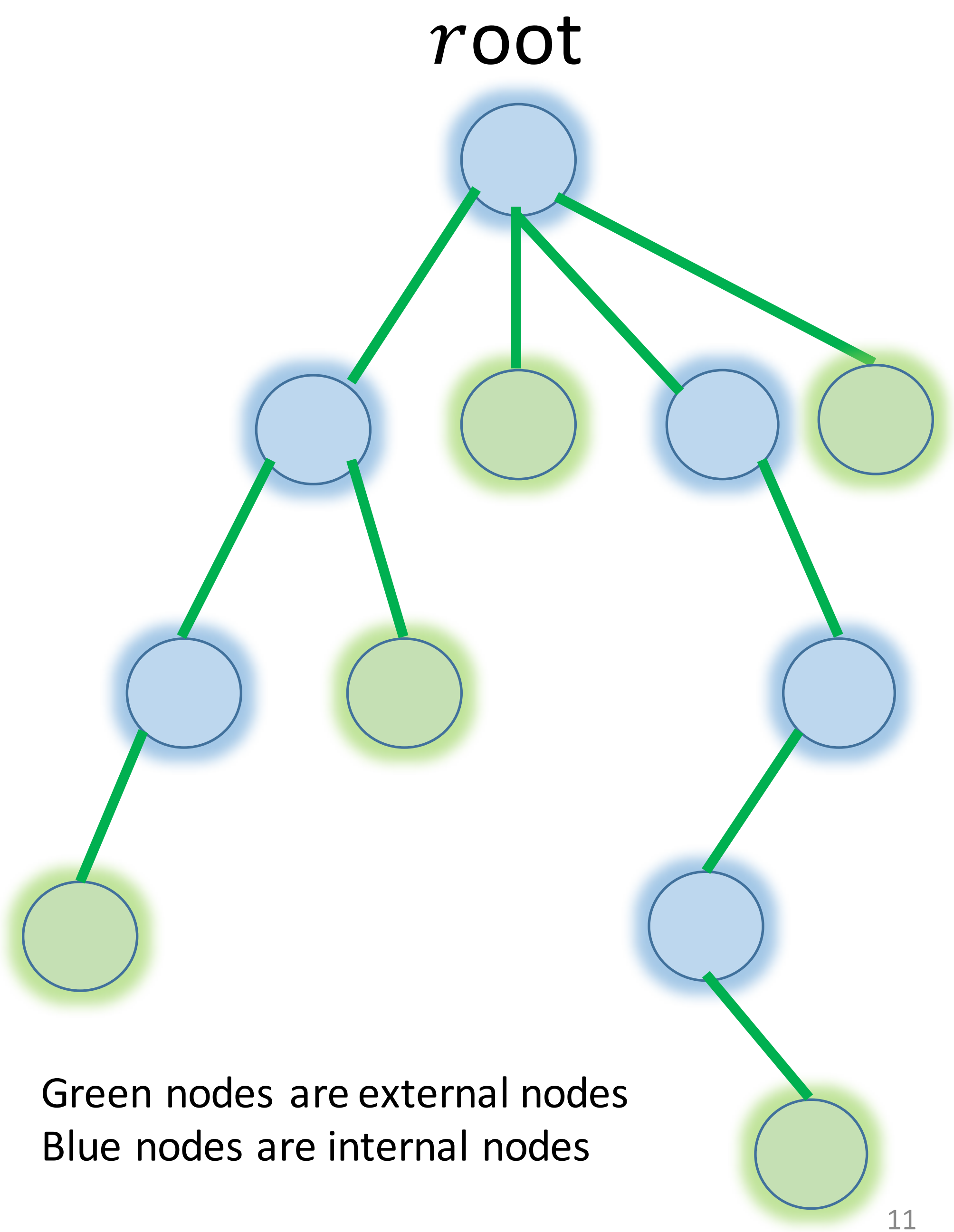


Subtree of x

Trees

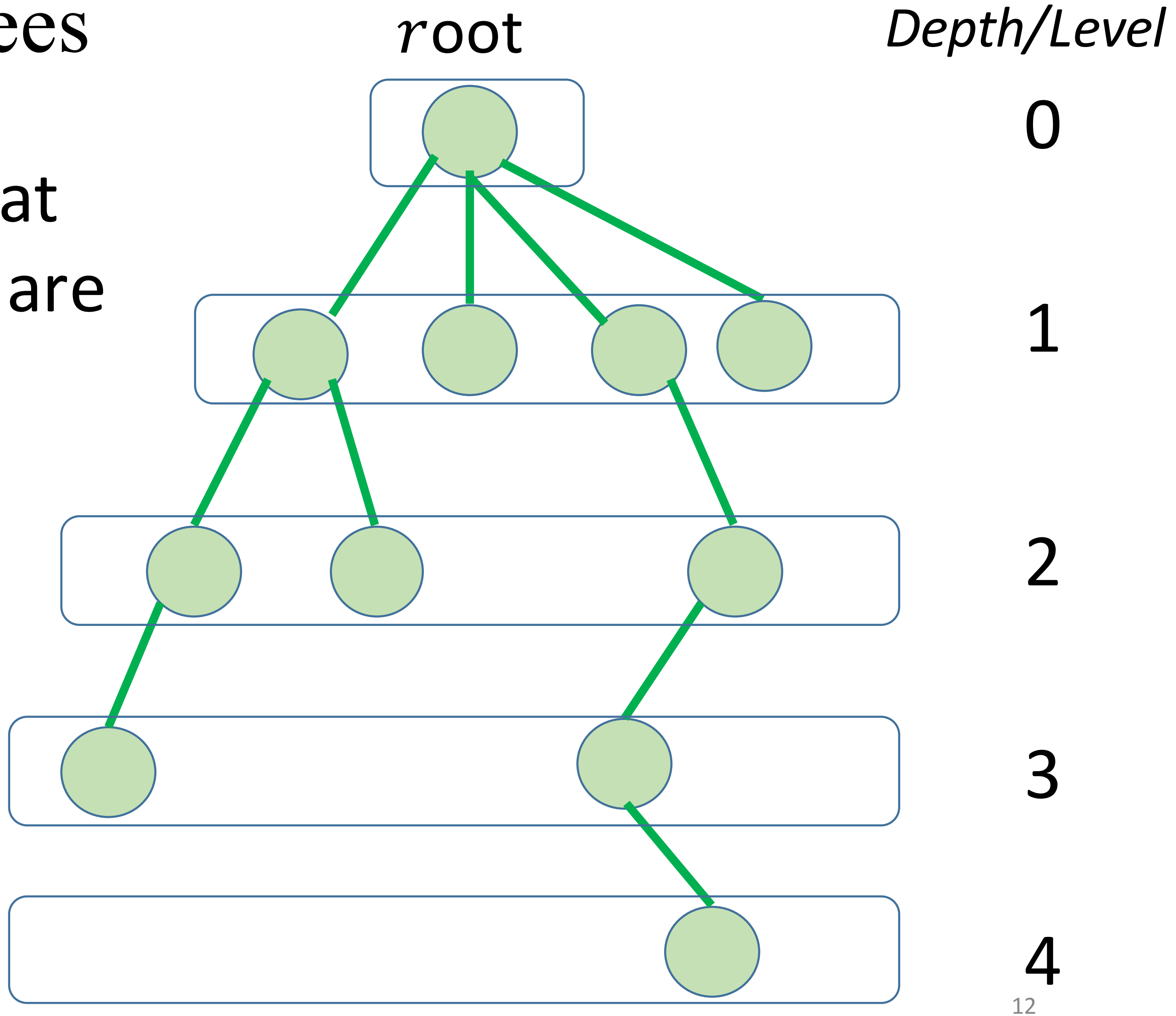
Leaf (external node): A node that has no child

Non-leaf (internal node): A node with at least one child



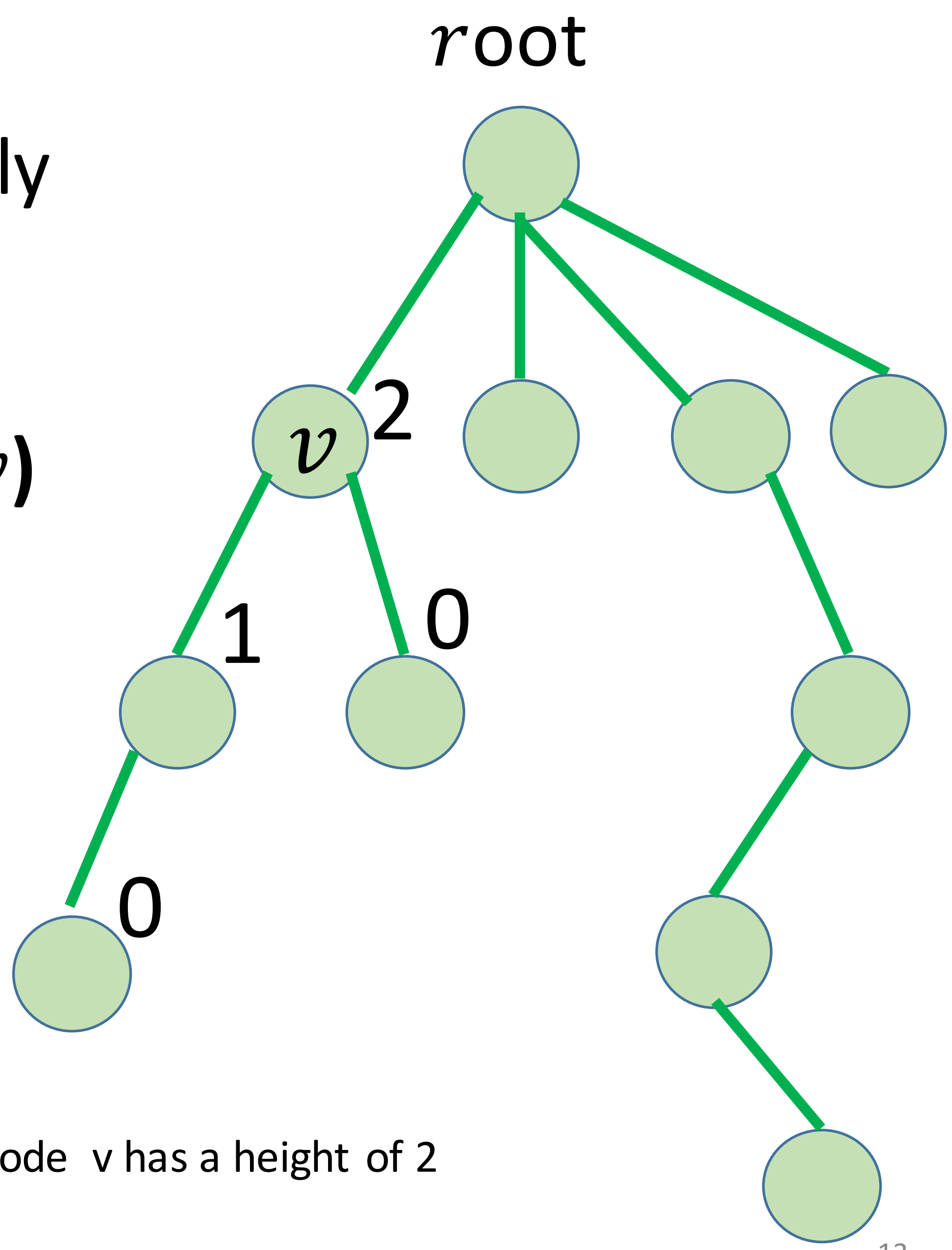
Depth and level in trees

All nodes of the tree that have the same *depth d* are at *level d* of the tree



Height of a tree

- The *height* of a node v is recursively defined to be:
  1. **0** if v is a leaf node.
  2. **1 + max(**height of any child of v**)**

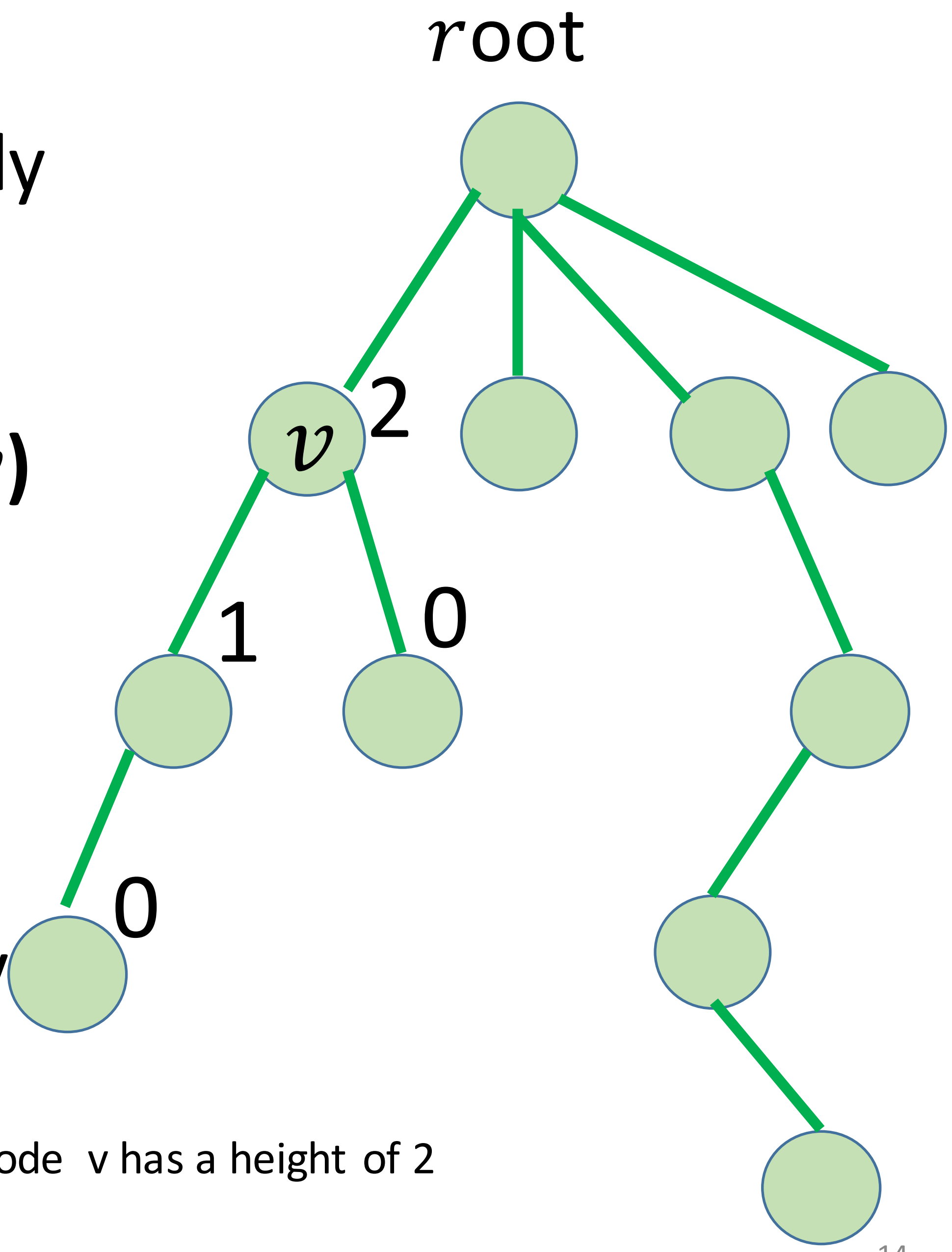


Node v has a height of 2

Height of a tree

- The *height* of a node v is recursively defined to be:
  1. **0** if v is a leaf node.
  2. **1 + max(**height of any child of v**)**

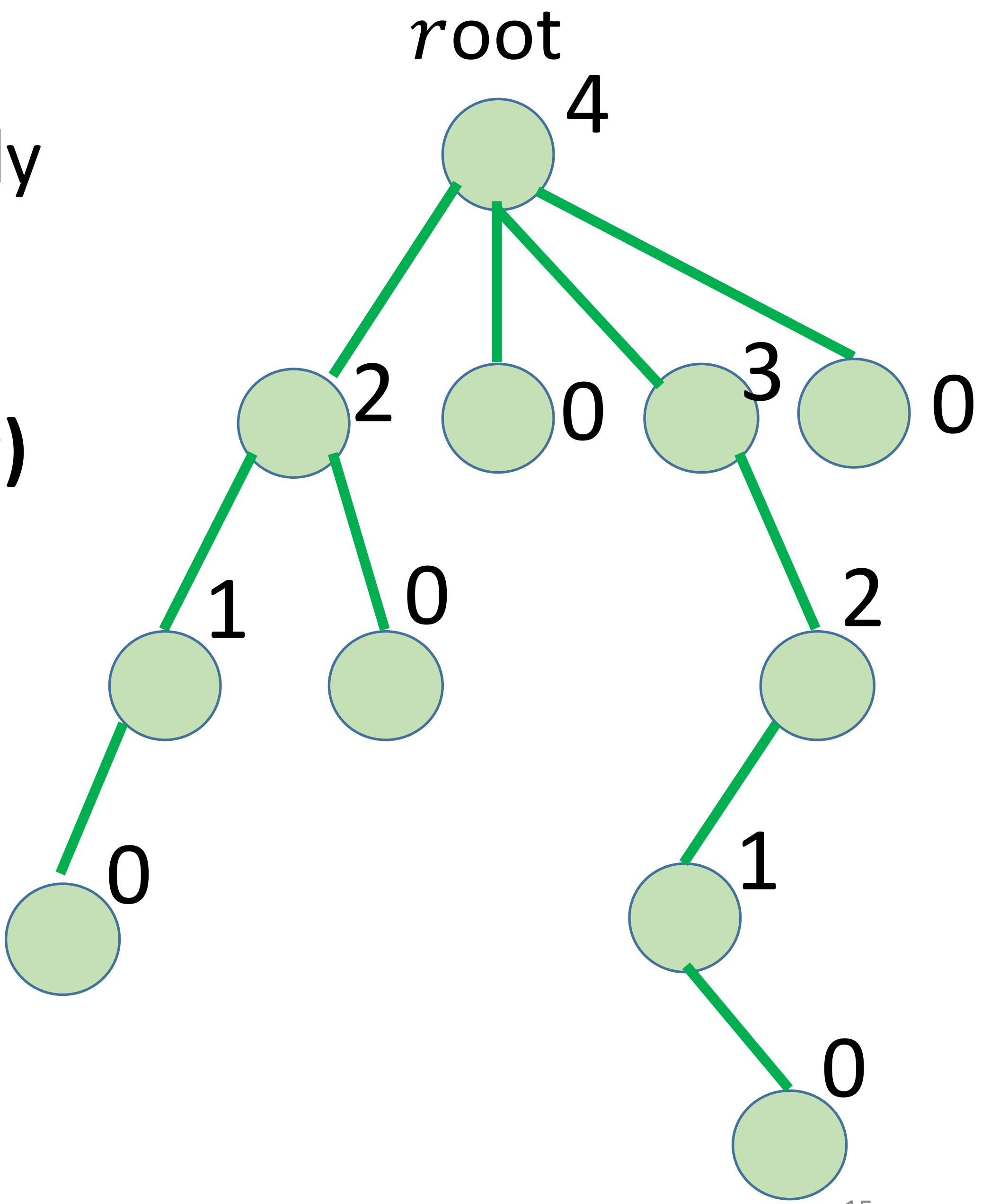
- An intuition could be separating the subtree of the desired node v and putting it on the ground, then height determines how many nodes v is away from the ground



Node v has a height of 2

Height of a tree

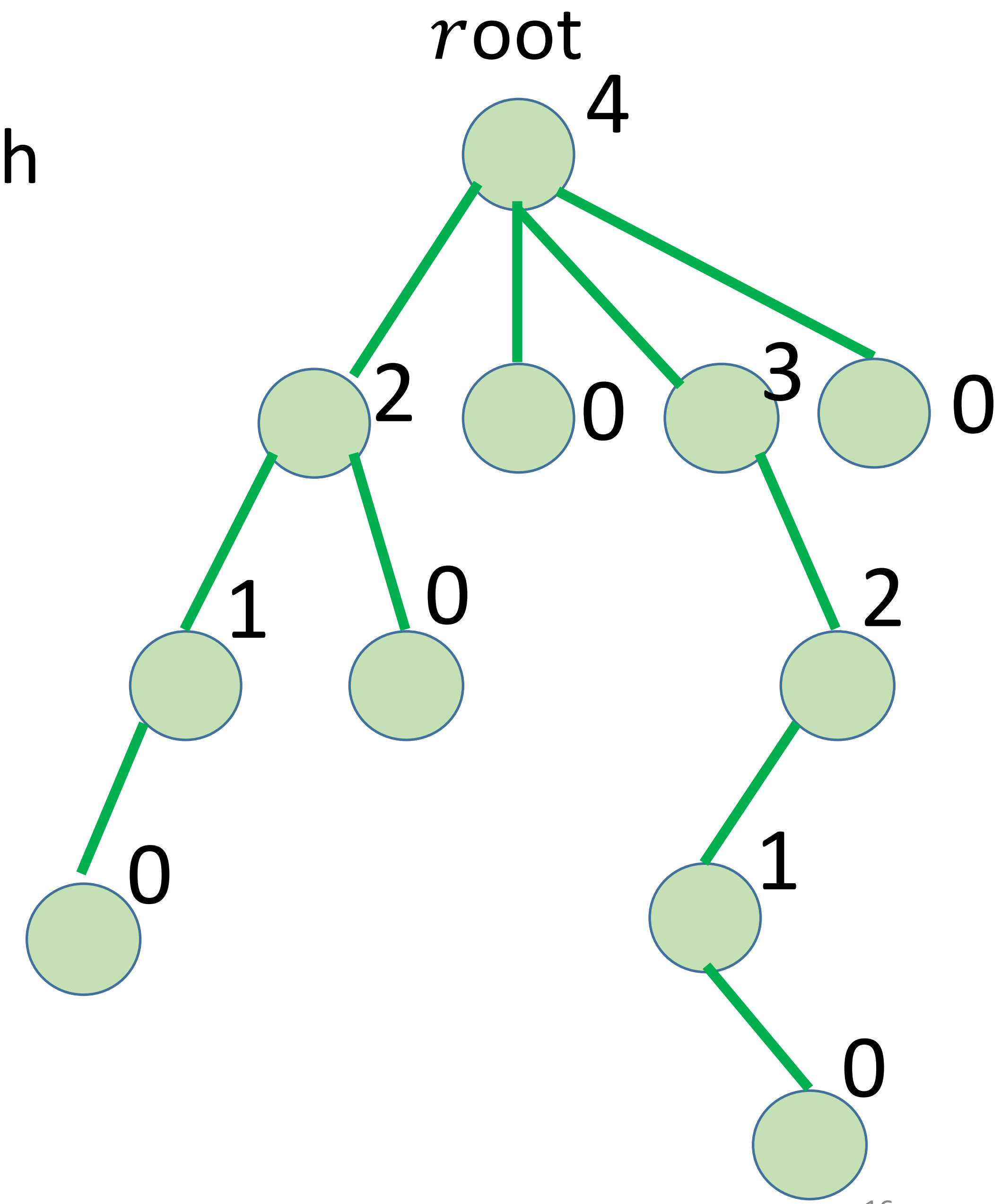
- The *height* of a node v is recursively defined to be:
  1. **0** if v is a leaf node.
  2. **1 + max(**height of any child of v**)**



Height of a tree

- It can also be defined as the length of the maximum path from v to a leaf in v's subtree.

- Height of a tree is height of the root node.

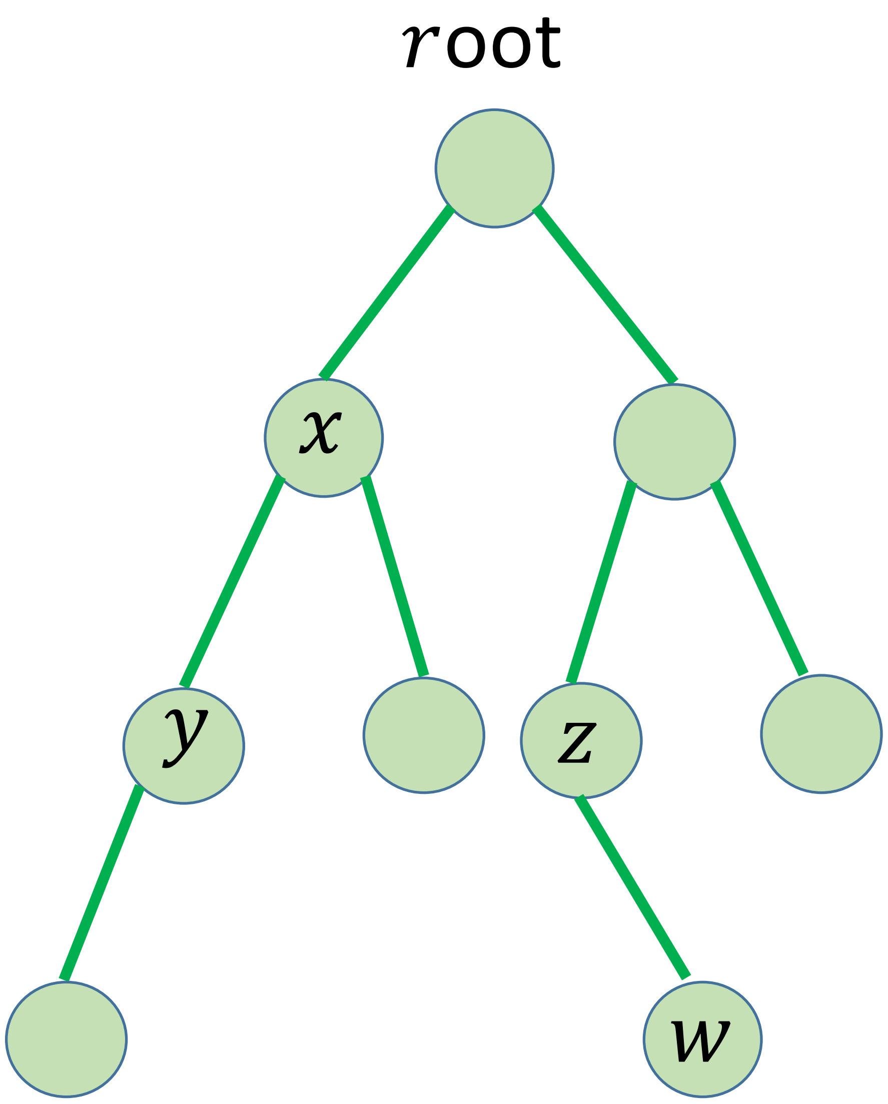


Binary tree

Definition: A binary tree is a rooted tree in which each node has **at most 2 children**

We refer to these children by **left child** and **right child**.

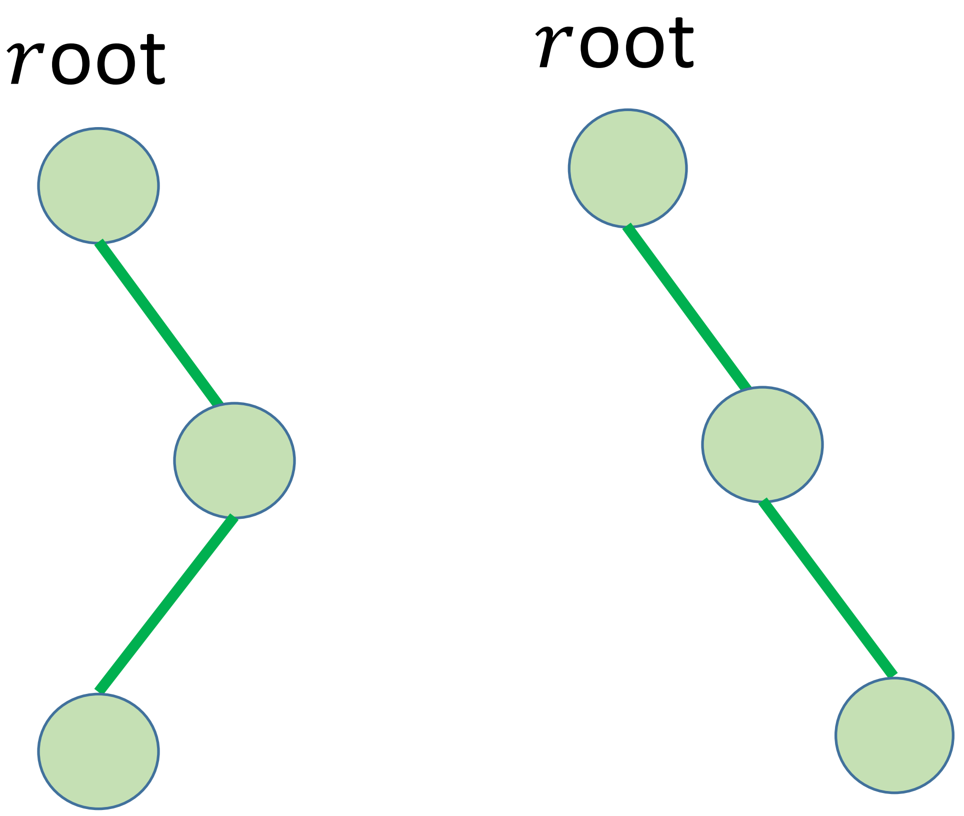
Note: It is possible that a node has only the left child or only the right child



y is the left child of x
w is the right child of z

Binary tree

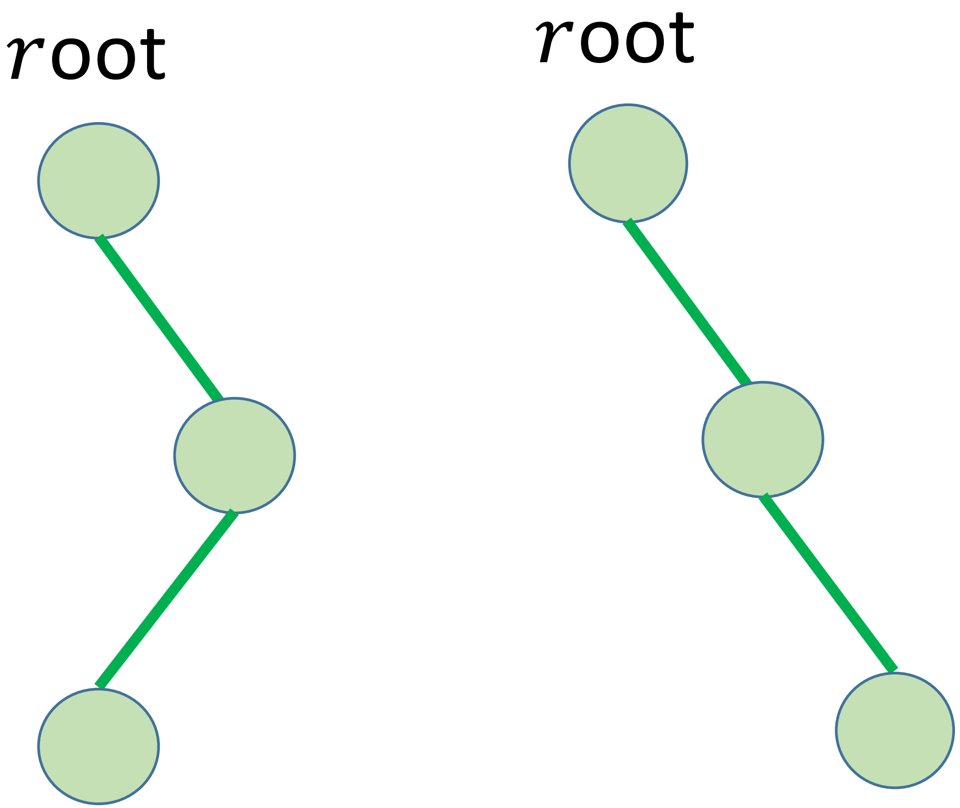
Question: Are these binary trees the same?



Binary tree

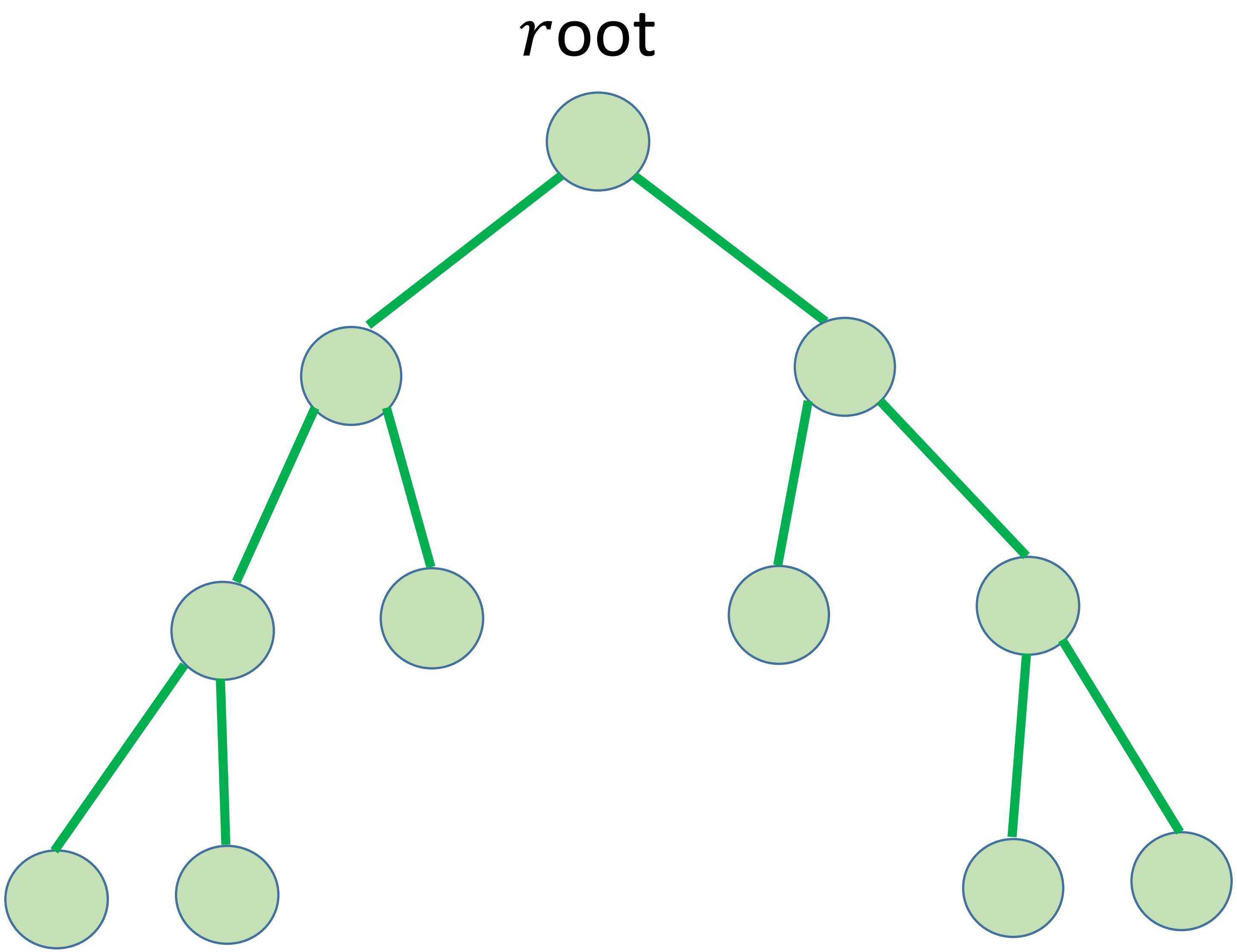
Question: Are these binary trees the same?

Answer: No :)



Full binary tree

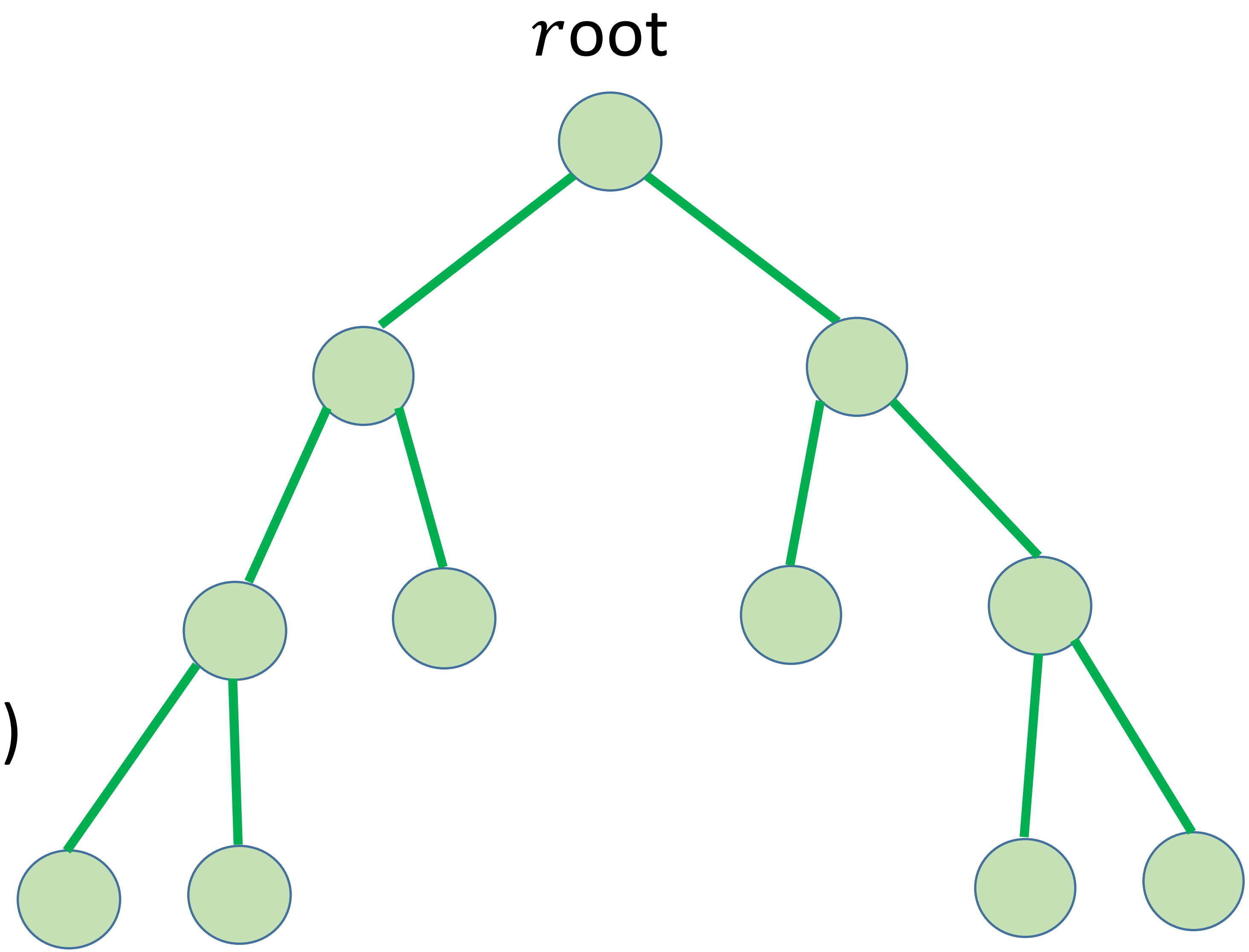
Definition: In a full binary tree each non-leaf node has **exactly 2 children**



Full binary tree

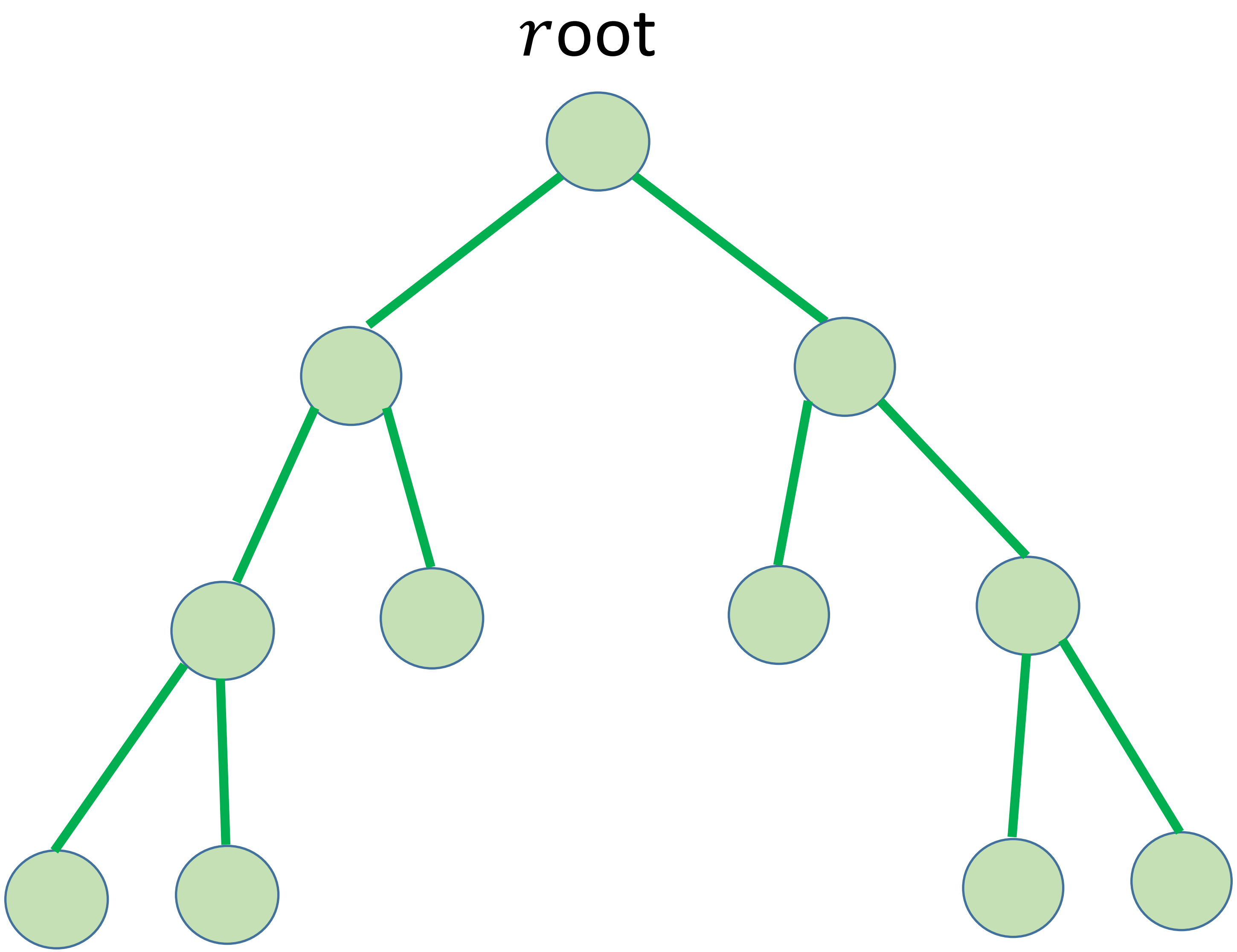
Definition: In a full binary tree each non-leaf node has **exactly 2 children**

The **maximum** number of nodes at level d is 2^d (this is also true for binary trees)



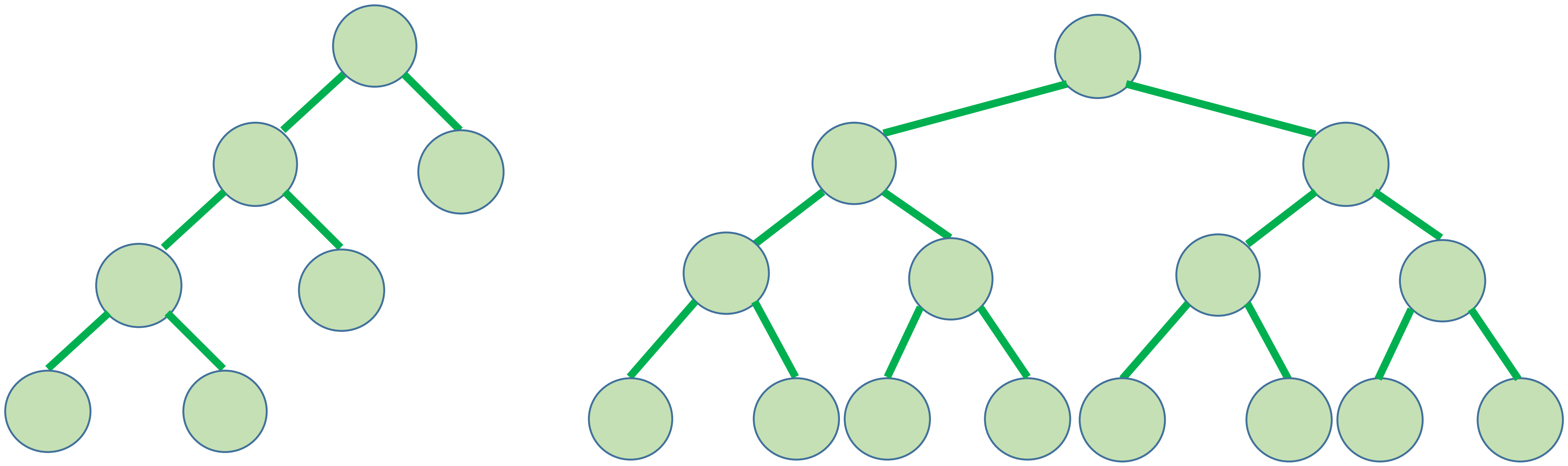
Full binary tree

Question: What is the minimum and the maximum **number of nodes** in a full binary tree with height $\boldsymbol{h}$?



Full binary tree

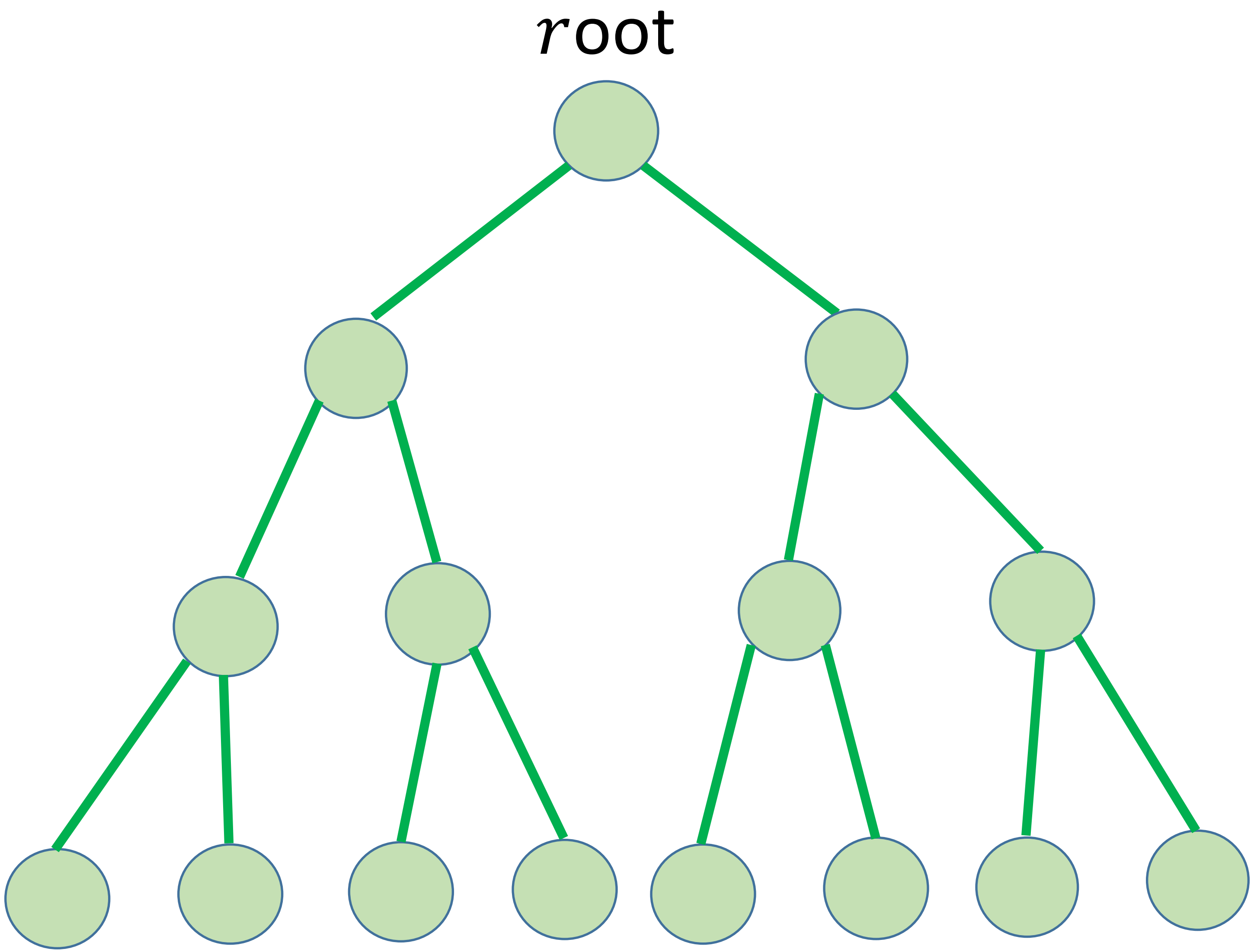
Answer: $2h + 1$ is the minimum, and $2^{h+1} - 1$ is the maximum



Complete binary tree

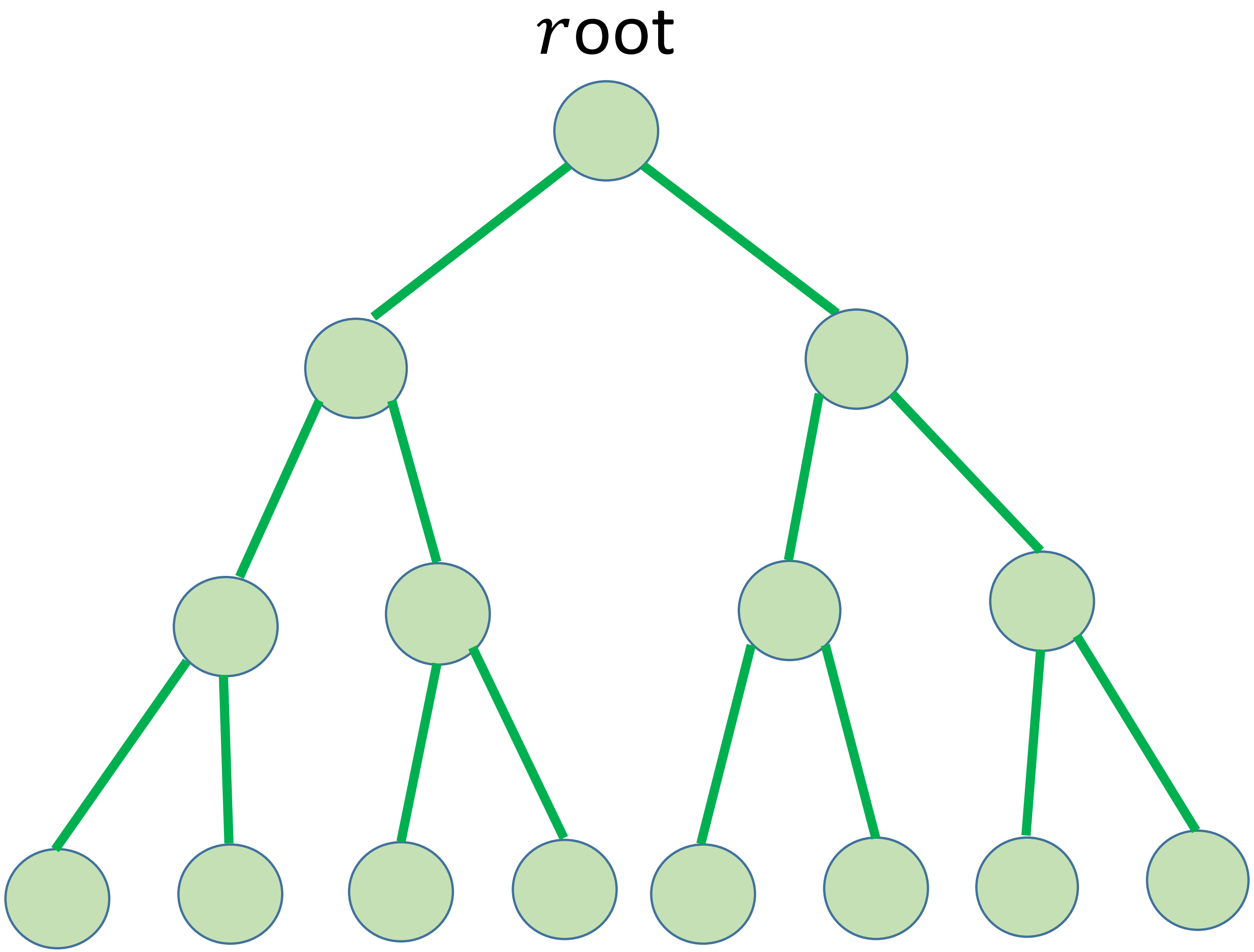
Definition: A complete binary tree is one in which all leaves have the same depth.

In other words, all levels are **complete**.



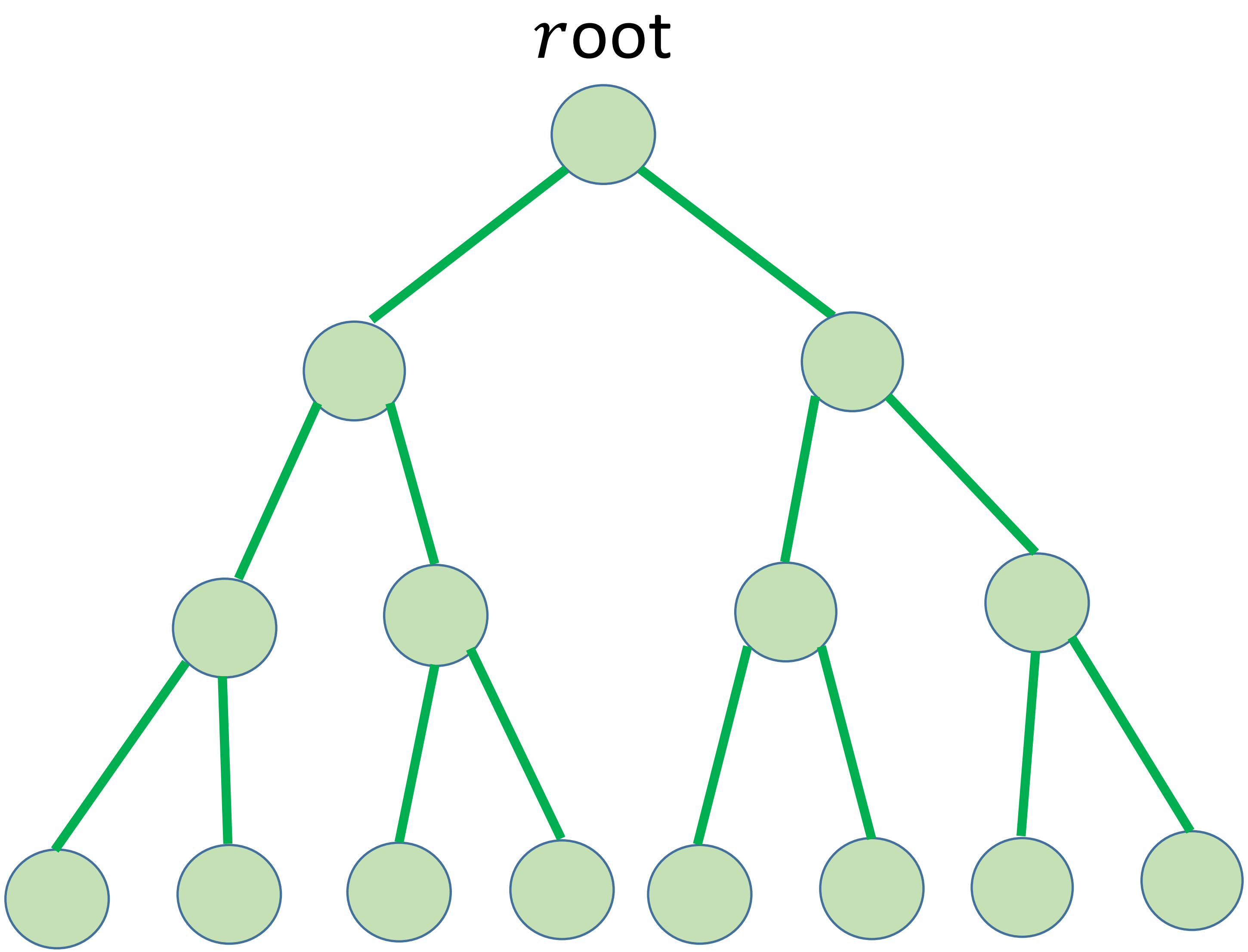
Complete binary tree

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



Complete binary tree

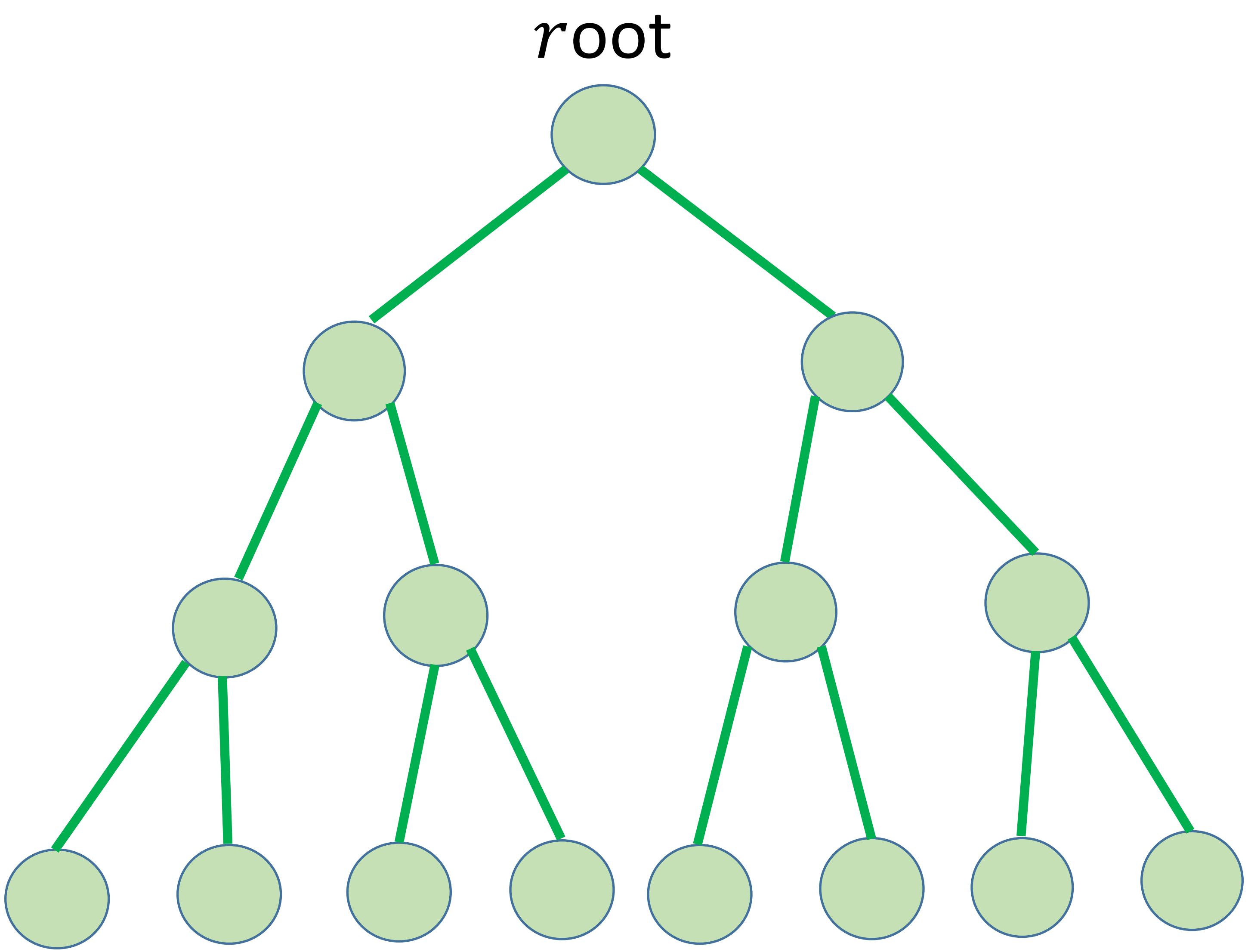
Question: What is the relation between number of leaves and the number of non-leaf nodes in a completer binary tree?



Complete binary tree

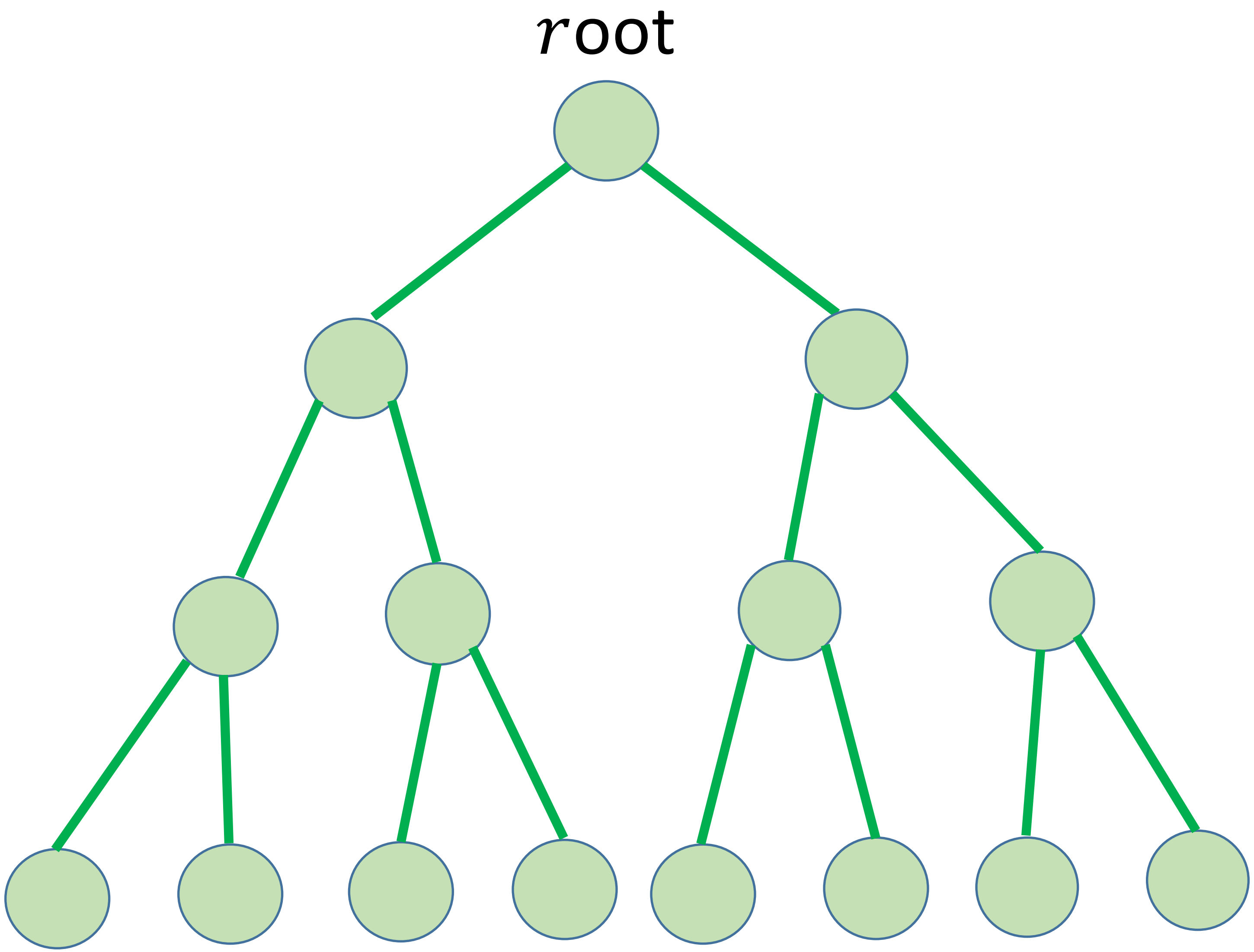
Answer: If we have x internal nodes, we are going to have $x + 1$ external nodes.

Can be proved by induction **on the height** of the complete binary tree.



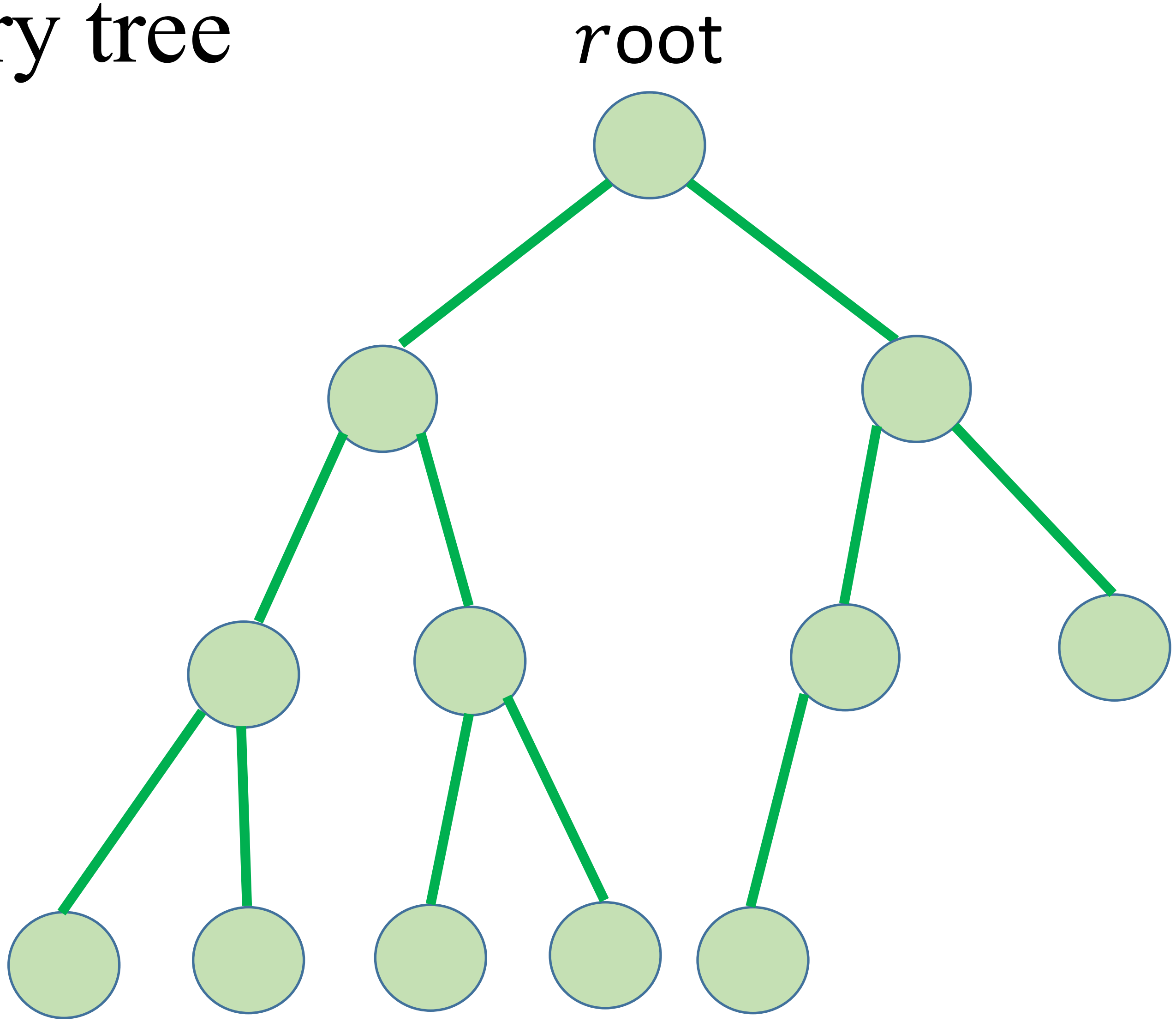
Complete binary tree

This is very useful since the number of leaves can be used as a constant approximation on n, and vice versa. (n is the number of nodes)



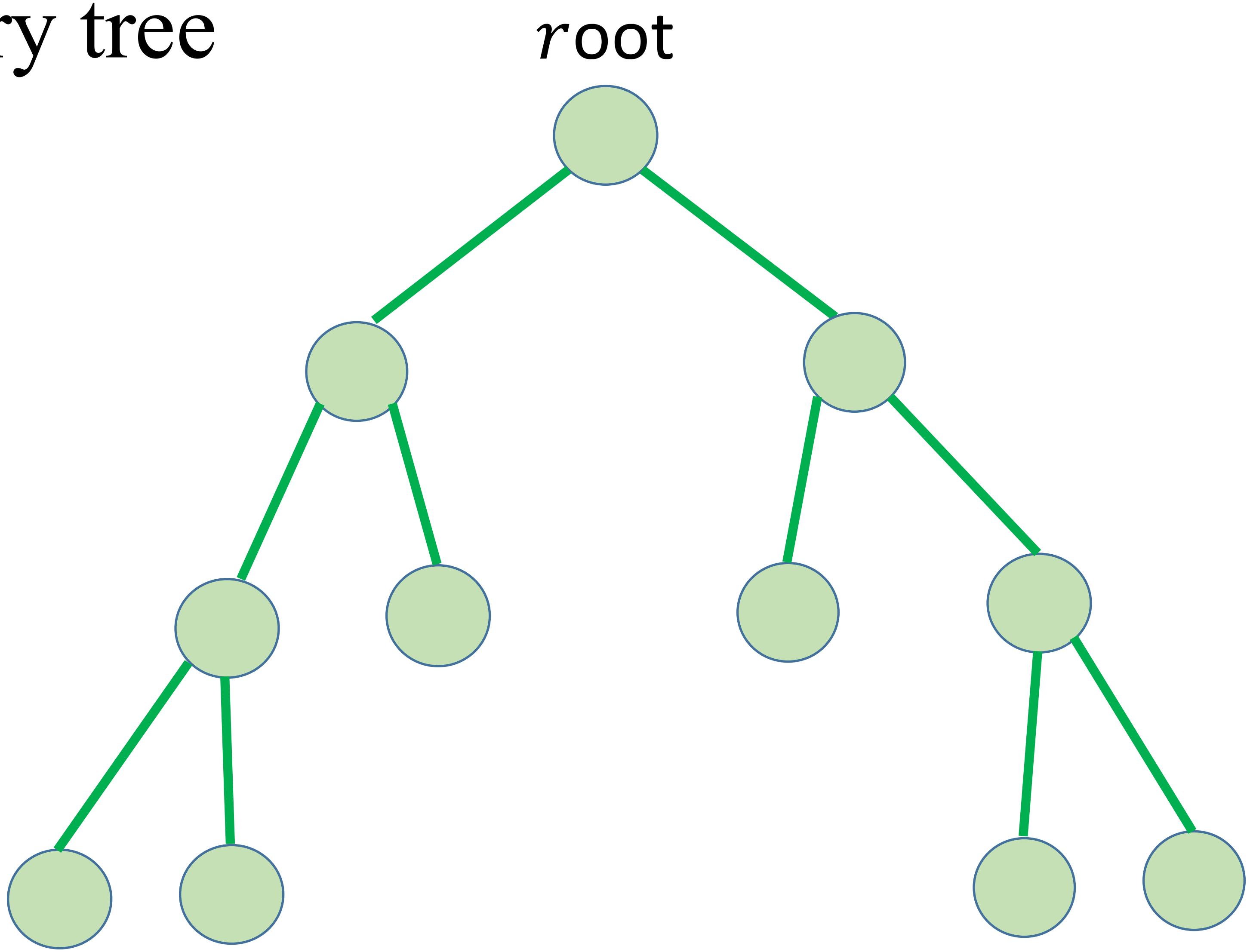
Nearly complete binary tree

Definition: In a **nearly complete binary tree** all levels are complete except possibly the last level which **is filled from left up to a point**.



Nearly complete binary tree

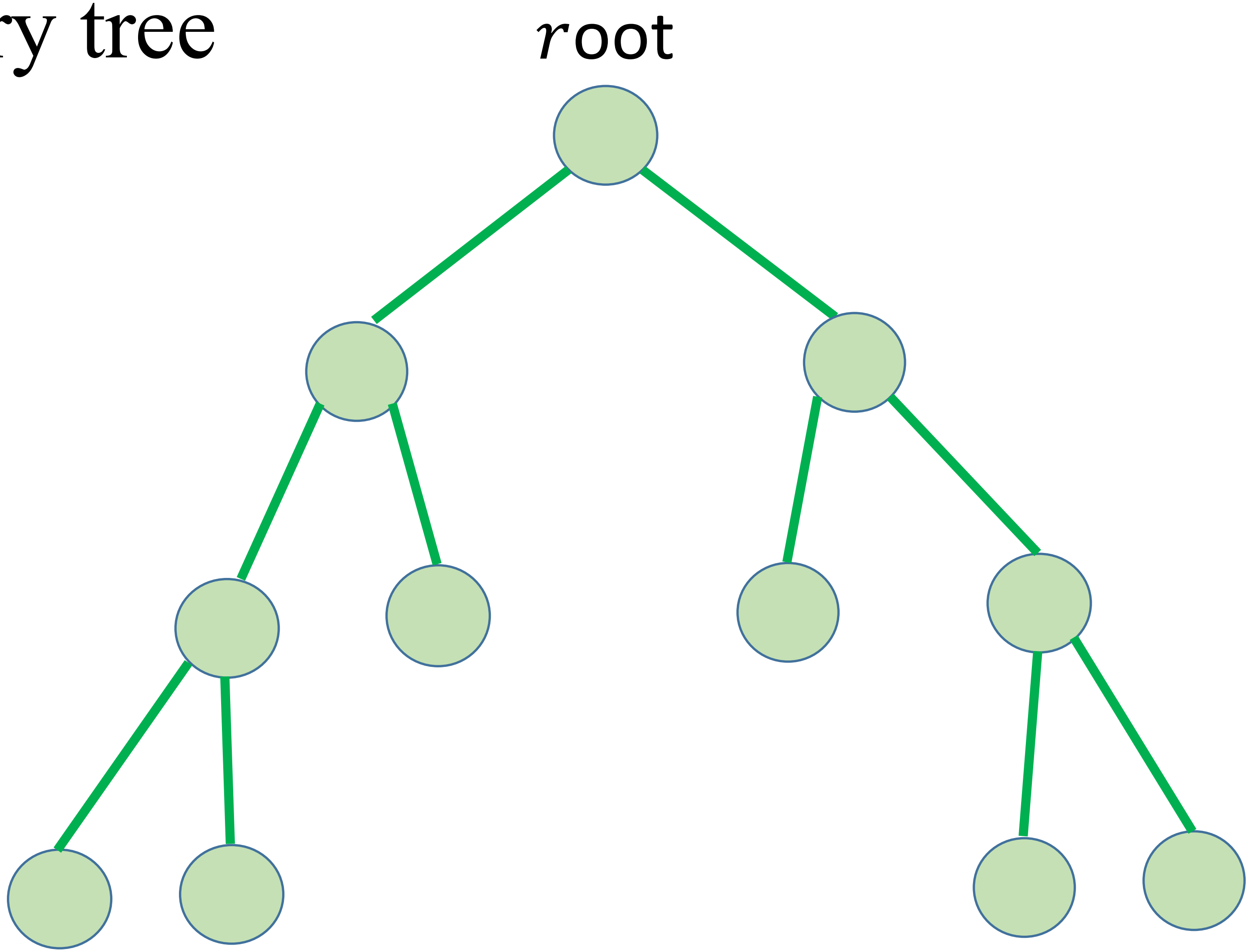
Question: Is this a nearly complete binary tree?



Nearly complete binary tree

Question: Is this a nearly complete binary tree?

Answer: No



Nearly complete binary tree

Exercise: If n is the number of nodes, what is the **range of n** in a nearly complete binary tree of height h?