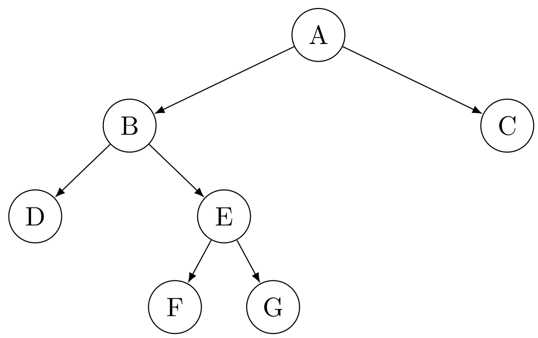
Induction Proofs over Binary Trees

Daniel Bauer

February 2026

1 Full Binary Tree Theorem

A *full binary tree* (also called a *proper binary tree*) is a binary tree in which every internal node has exactly 2 children.



Full Binary Tree Theorem: A full binary tree with I internal nodes has $L = I + 1$ leaves.
Corollary: since the total number of nodes $N = I + L$

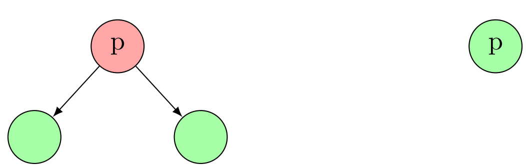
- $N = 2I + 1$
- $I = (N–1)/2$
- $L = (N + 1)/2$
- $N = 2L–1$
- $I = L–1$

Proof by Induction (technique 1)

Base case: A tree consisting of a single node has one leaf and no internal nodes: 1 = 0+1.

Inductive step (over the number of leaves): Assume, every binary tree with I internal nodes has $L = I + 1$ leaves. We need to show that any tree with $L + 1$ leaves has $I + 2$ internal nodes.

Assume you have a tree with $L + 1$ leaves. Choose a leaf of maximum depth. Its sibling must also be a leaf (otherwise there would be a leaf at greater depth), so these two leaves share a common parent p. Removing both leaves turns p from an internal node into a leaf. Thus, if we remove both leaves, we get a tree with $L + 1 - 2 + 1 = L$ leaves. **By the inductive hypothesis**, the smaller tree with L leaves has $I + 1$ internal nodes. The larger tree (before to removing the leaves) had one additional internal node (node p), for a total of $I + 1 + 1 = I + 2$ internal nodes. □

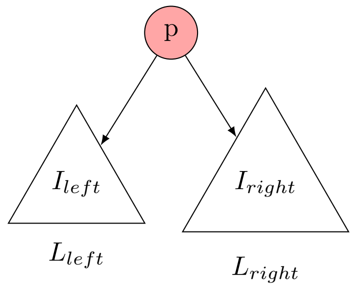


Proof by Structural Induction (technique 2)

Base case: A tree consisting of a single node has one leaf and no internal nodes: 1 = 0+1.

Inductive step: Assume that every full binary tree with I internal nodes has $L = I + 1$ leaves.

Now consider a tree T with $I + 1$ internal nodes. Any full binary tree, other than the base case, consists of a root node p and two non-empty subtrees.



Let the number of internal nodes in the left subtree I_{left} and the number of internal nodes in the right subtree be I_{right}. The full tree has $I + 1 = I_{left} + I_{right} + 1$ internal nodes (including p). **By the inductive hypothesis**, the left subtree has $L_{left} = I_{left} + 1$ leaves and the right subtree has $L_{right} = I_{right} + 1$ leaves. The number of leaves in the full tree is

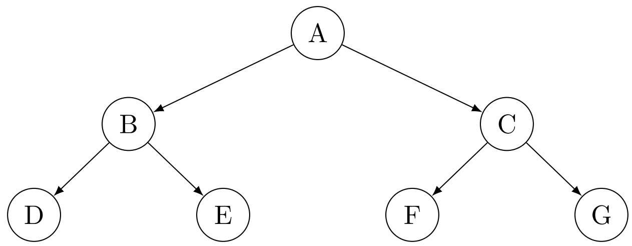
$$L = L_{left} + L_{right} = I_{left} + 1 + I_{right} + 1 = I + 2.$$

$\square$

Exercise: Try to prove some of the corollary equations above directly using structural induction.

2 Perfect Binary Trees

A *perfect binary tree* is a binary tree where the number of nodes at each depth k is 2^k. In other words, all levels of the tree are completely filled with no gaps.



Number of Nodes in a Perfect Binary Tree: The number of nodes in a perfect binary tree of height h is $N = 2^{h+1} - 1$. Note, that all leaves in a perfect binary tree are at the same level.

Proof by Structural Induction

Base case: A binary tree of height $h = 0$ has a single node. $2^{0+1} - 1 = 1$.

Inductive step: Assume that any perfect binary tree of height h has $2^{h+1} - 1$ nodes. Now consider a perfect binary tree of height $h + 1$. We need to show that such a tree has $2^{(h+1)+1} - 1 = 2^{h+2} - 1$ nodes. The tree consists of a root node with a non-empty left and right subtree. Because all leaves are at the same level, these subtrees each have height h. **By the inductive hypothesis**, each subtree has $2^{h+1} - 1$ nodes. The total number of nodes in the full tree of height $h + 1$, including the root node, is

$$2 \cdot (2^{h+1} - 1) + 1 = 2^{h+2} - 2 + 1 = 2^{h+2} - 1$$

□

Corollary: Height of a Perfect Binary Tree

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

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

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

$$h = \log_2(N + 1) - 1 = O(\log N)$$