CSCI 3104: Algorithms

Lecture 2: Proof by Induction

Rachel Cox

Department of Computer Science

Introduction & Logistics

Course Platforms:

Canvas - All links to Zoom Lectures, HW assignments, Quizzes, Grades

Piazza - Q&A from classmates, TAs, CAs, and Instructors

Gradescope - Submission of HW and Exams

What will we learn today?

- ❑ Proof by Induction

Induction

Let $P(n)$ be the property that we're trying to prove.

An inductive argument goes as follows:

Base Case: Verify that $P(0)$ holds

Induction Step: For $k \geq 0$, If $P(k)$, then $\mathrm{P(k+1)}$

Conclusion: $(\forall n \geq 0)\ P(n)$

Induction

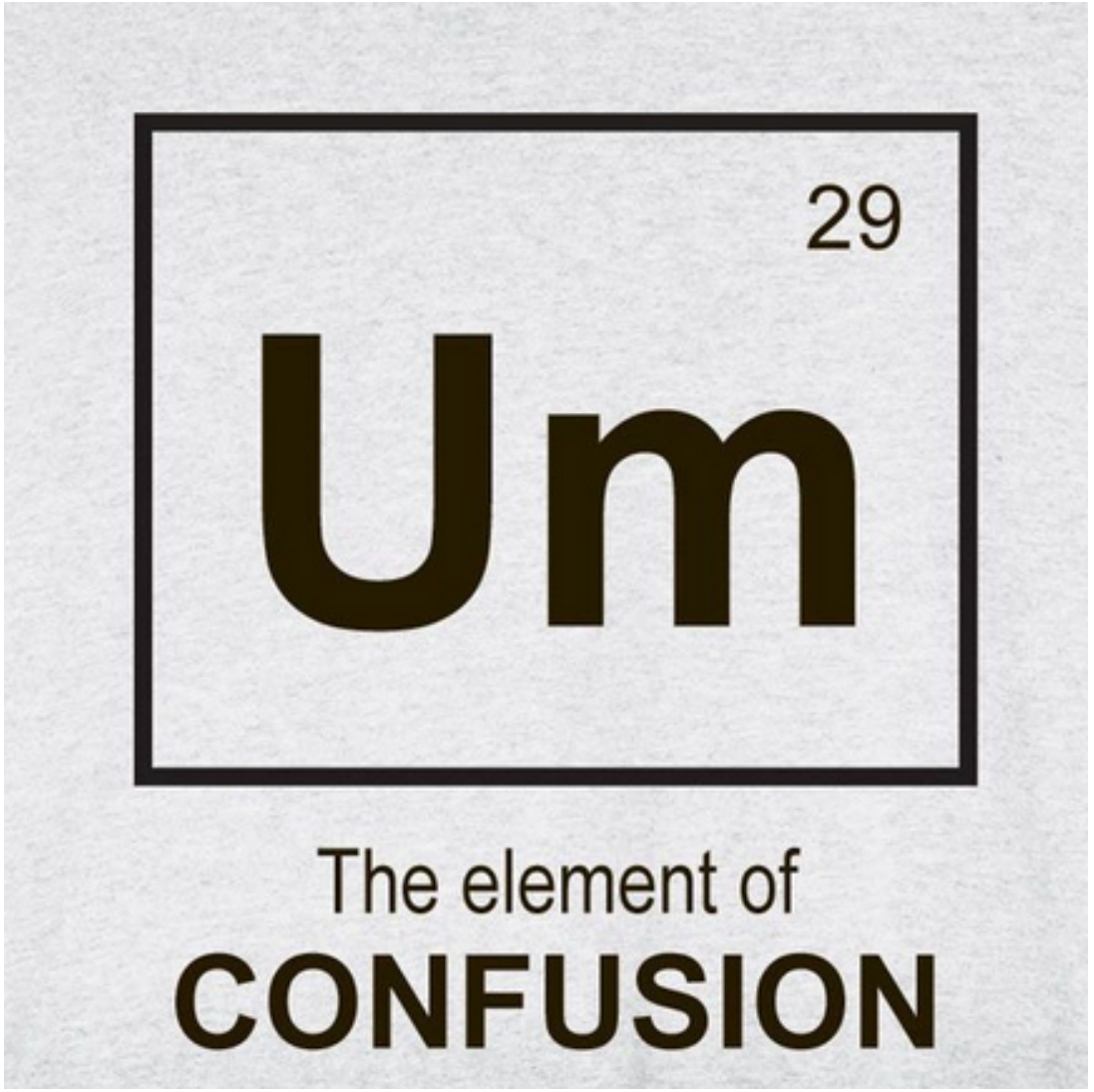
Weak Induction:

- Verify that $P(1)$ is true.
- Assume $P(k)$ is true and show that $P(k+1)$ is true.

Key difference is in the inductive hypothesis

Strong Induction:

- Verify that $P(1)$ is true.
- Assume $P(m)$ for all $1 \leq m \leq k$ and show $P(k+1)$



Induction

In both strong and weak induction, you must prove that the first domino in the line falls. i.e. the first logical proposition is true – base case.

Analogy 1: To prove that ALL the other dominoes fall, you either show why (1) each falling domino by itself causes the next one to fall, or (2) all the dominoes that have fallen up to a point will cause the next one to fall. Tactic 1 is called weak induction, tactic 2 is called strong induction.

Analogy 2: Weak induction only cares about the ladder rung you are currently standing on. As long as that one exists, you know that you can step up to the next rung. For strong induction, you need all the previous rungs to still exist before you can safely move up to the next rung.

Induction

Example: Prove the claim below using induction.

$$\forall n \geq 1 \left[\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \right]$$

Proof: We proceed with induction; specifically weak induction.

Base Case: $n = 1$

$$\sum_{i=1}^{1} i = 1$$

$$\frac{1(1+1)}{2} = \frac{2}{2} = 1 \qquad 1 = 1 \ \checkmark$$

Inductive Step: Assume that $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$ for some $k \geq 1$ (Induction Hypothesis)

$$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + k+1 \qquad \text{By properties of sums.}$$

$$= \frac{k(k+1)}{2} + k+1 \qquad \text{By our inductive hypothesis}$$

Induction

Example: Prove the claim below using induction.

$$\forall n \geq 1 \quad \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

(continued)

$$\sum_{i=1}^{k+1} i = \frac{k(k+1)}{2} + k+1$$

$$= \frac{k(k+1)}{2} + \frac{2(k+1)}{2}$$

$$= \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{(k+1)[k+2]}{2}$$

$$= \frac{(k+1)(k+1+1)}{2}$$

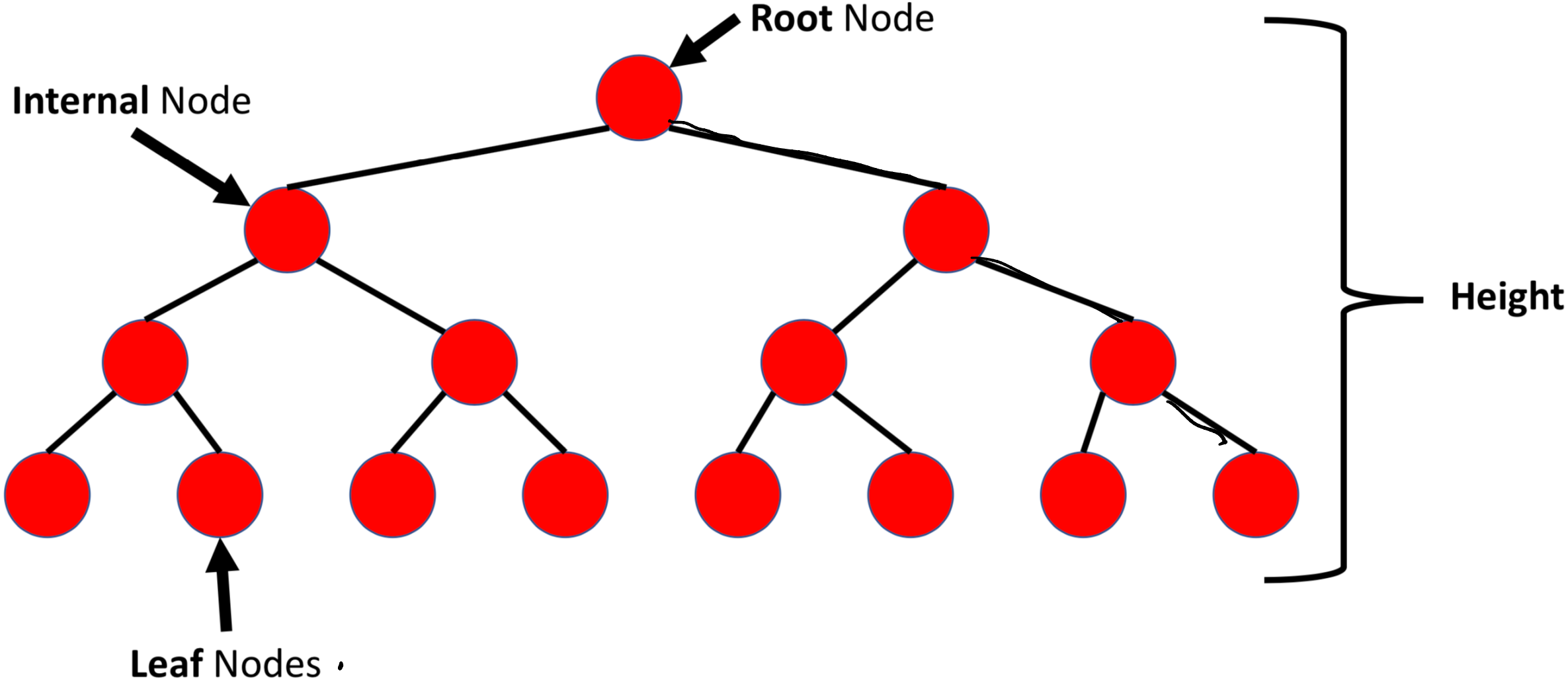
Thus by weak induction, we've shown that

$\forall n \geq 1, \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

■

Trees

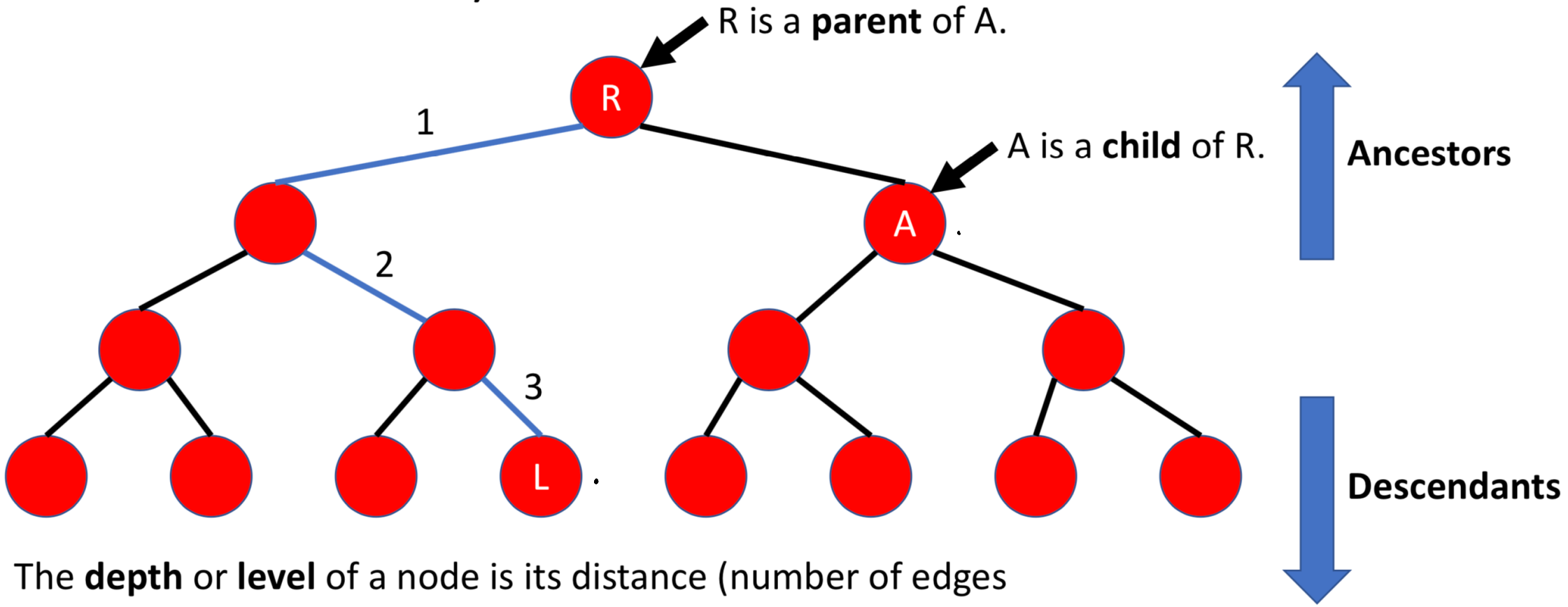
Rooted Binary Trees



Trees

A **rooted tree** is **binary** if every node has at most 2 children.

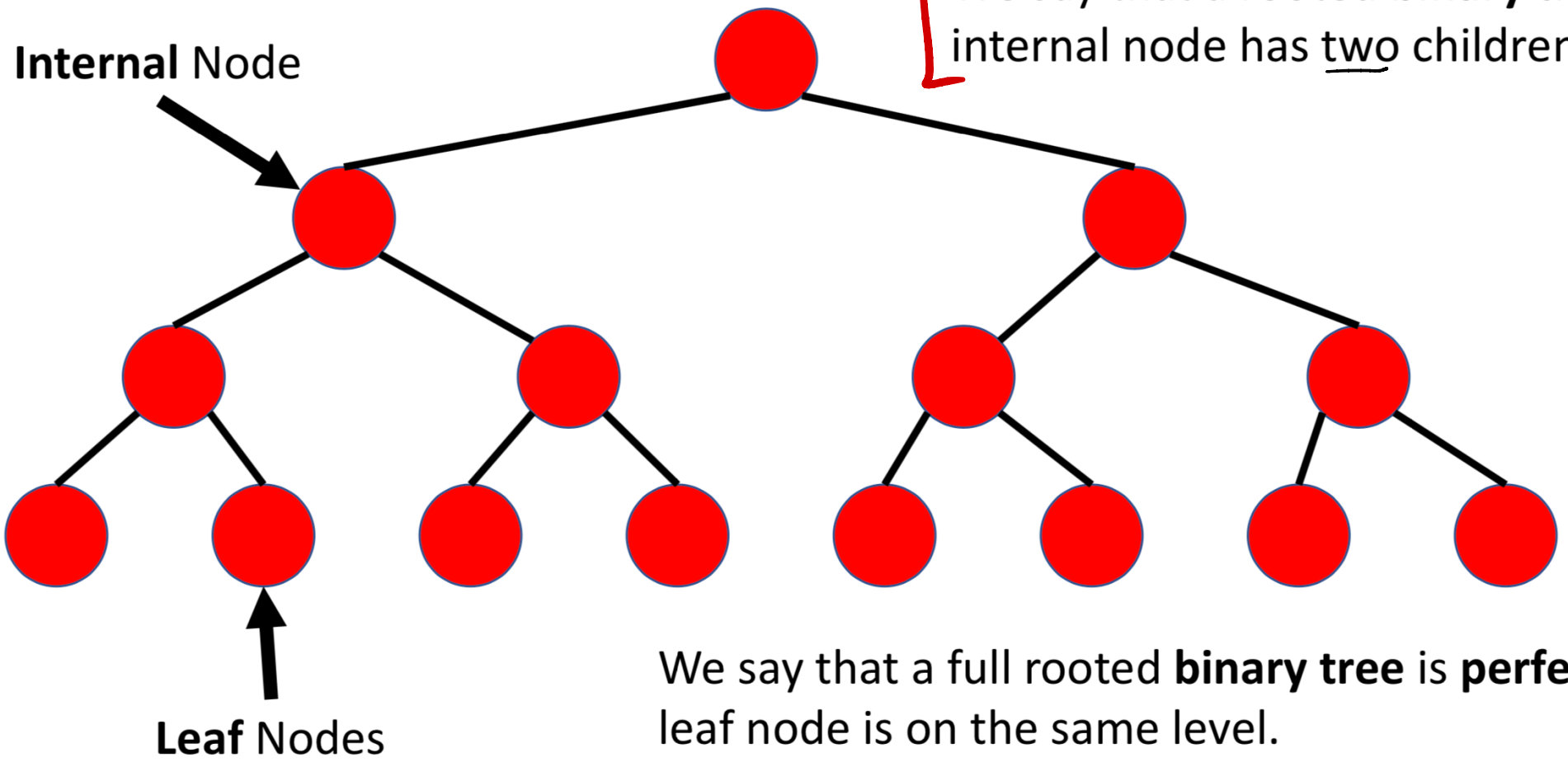
Rooted Binary Trees



The **depth** or **level** of a node is its distance (number of edges traversed in the shortest path) from the root. For example, we say L is at depth 3 and A is at level 1.

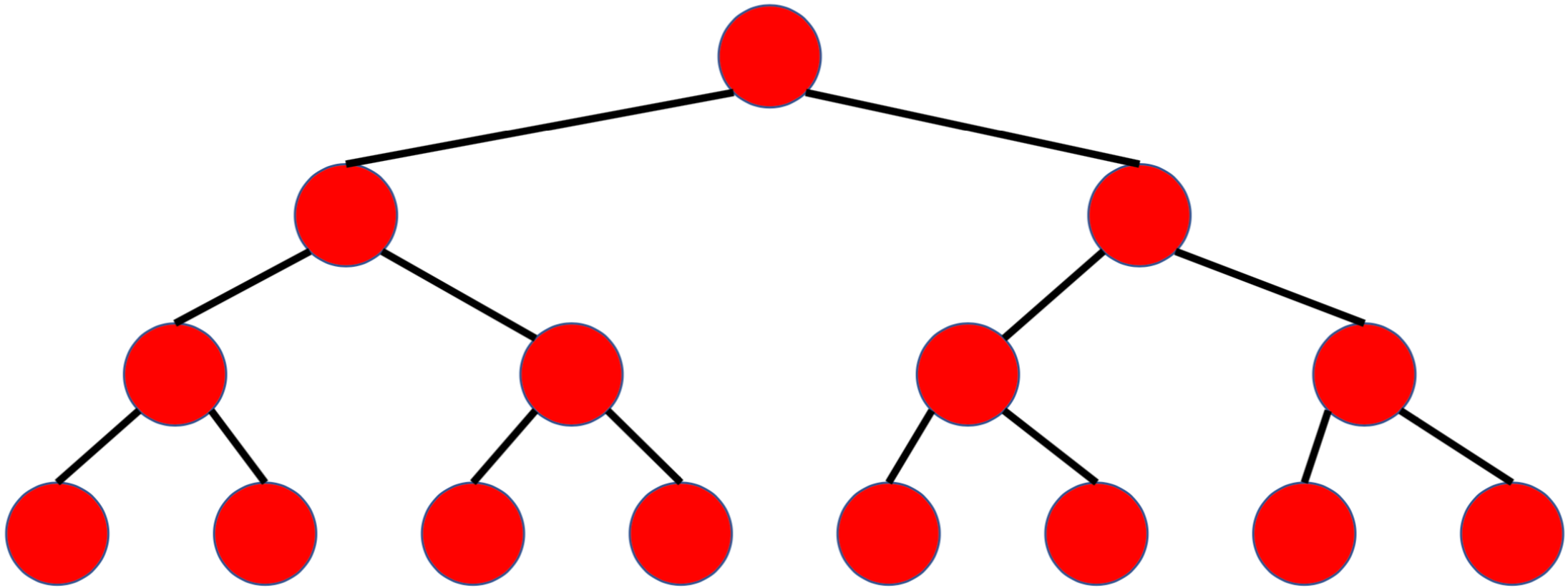
Rooted Binary Trees

We say that a rooted **binary tree** is **full** if every internal node has two children.

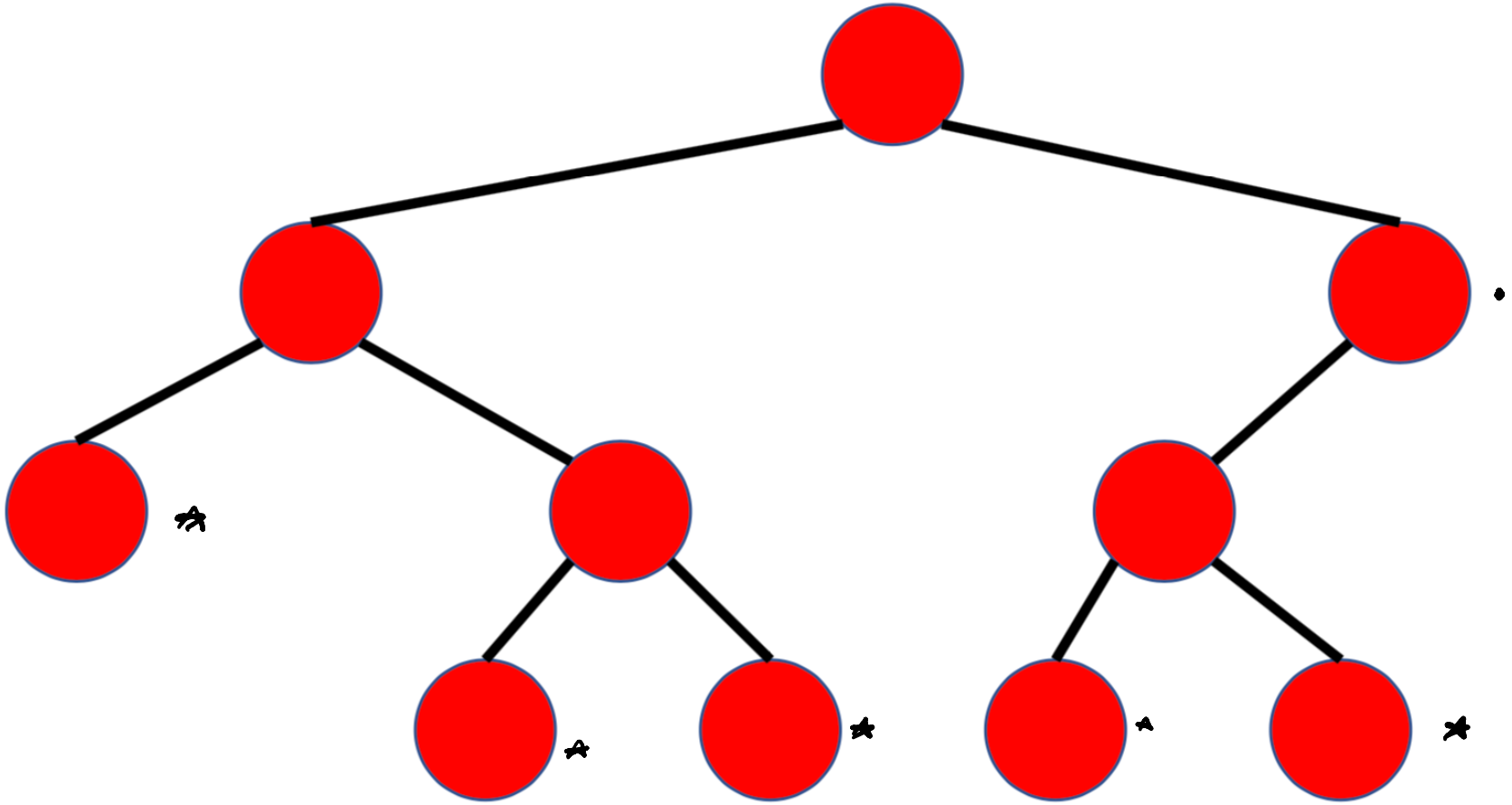


We say that a full rooted **binary tree** is **perfect** if every leaf node is on the same level.

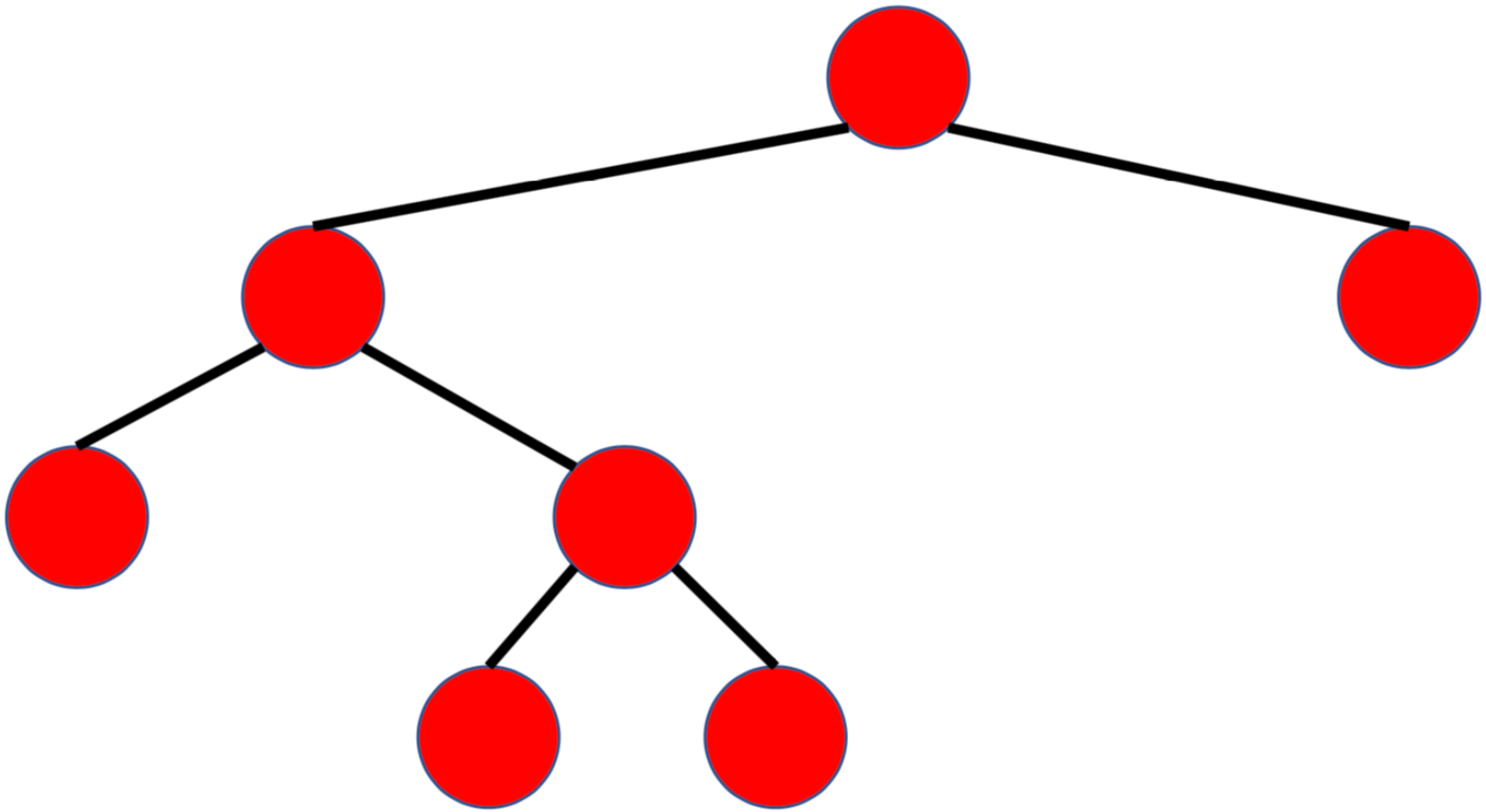
Binary Trees (perfect)



Rooted Binary Trees (not perfect or full)



Rooted Binary Trees (not perfect but full)



Induction

Example: Prove the claim below using induction.
$\forall n \geq 1$ All trees with n vertices have exactly $n - 1$ edges.

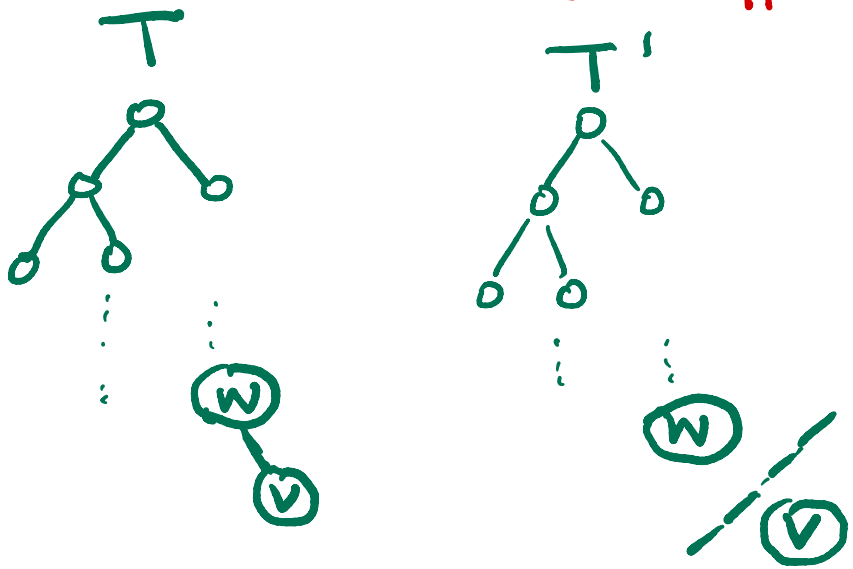
Base Case Tree

Proof: The proof is by induction.

Base case: $n = 1$. A tree with one vertex, has $1 - 1 = 0$ edges. ✓

Induction Step: Assume that every tree with k vertices has $k-1$ edges. } Inductive Hypothesis

- Let T be a tree with $K+1$ vertices
- Let v be a leaf of T.
- Let w be the parent of v.
- Let T' be the tree obtained by Removing v and the edge (w, v).



Induction

Example: Prove the claim below using induction.

$\forall n \geq 1$ All trees with n vertices have exactly $n - 1$ edges.

(continued)

$\Rightarrow$ T' has k vertices (we just removed a vertex from T and we assumed T had $k+1$ vert.)

By the inductive hypothesis, T' has $k-1$ edges.

$\Rightarrow$ Thus, T has $k+1$ vertices and k edges because it has one more edge than T'.

Since k was arbitrary, we've shown that all trees with n vertices have $n-1$ edges. ∎

Recap: Components of Induction

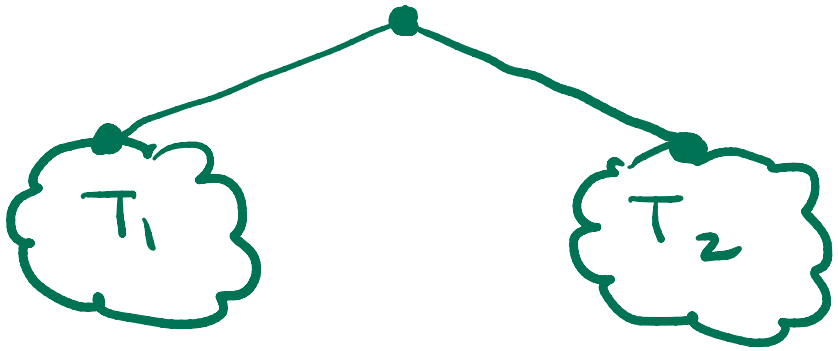
1. Statement of Inductive intent
2. Base Case
3. Inductive Hypothesis
4. Derive (K+1) from K
5. Concluding statement extrapolating to all n

Recursive Definition of Binary Trees

Example Create the set of all full binary trees by recursion

Basis Step: There is a full binary tree consisting of just a root r

Recursive Step: If T_1 and T_2 are full binary trees, then there exists a full binary tree, denoted $T_1 \cdot T_2$, created by a root vertex connected to the roots of T_1 and T_2



T

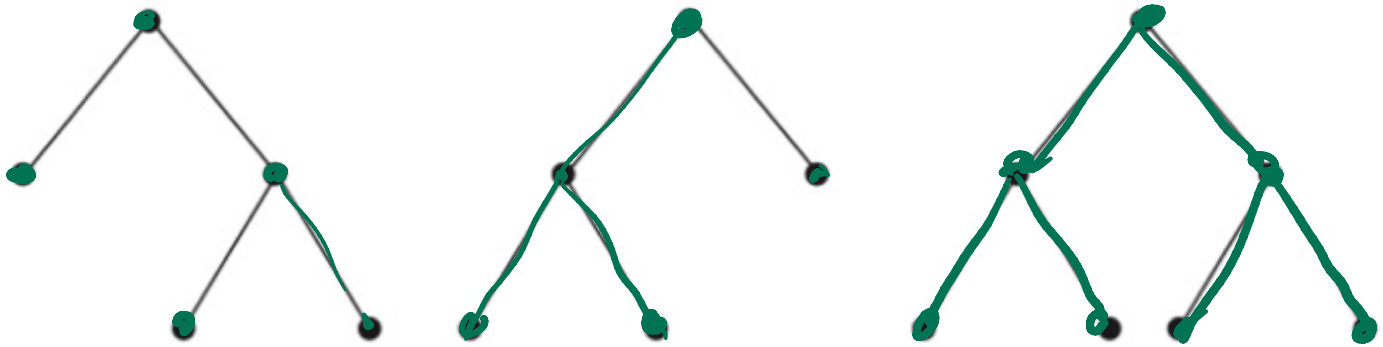
Induction

Example Create the set of all full binary trees by recursion

Basis step

Step 1

Step 2



Induction

In mathematical induction (strong or weak) we prove that a proposition P indexed by a natural number n holds for all values of n

P(n)

When proving things about objects defined recursively, it's typically more convenient to use a proof technique called **structural induction**.

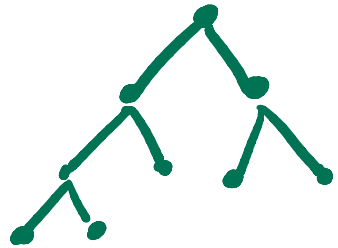
A **structural induction** proof has two steps:

Basis Step: Show that the result holds for all elements defined in the basis step of the recursive definition.

Recursive Step: Show that if the result holds for all elements used to construct new elements in the recursive step in the recursive definition, then it holds for the new element.

Induction

Example: Prove the claim below using structural induction.
If T is a full binary tree, then T has an odd number of vertices.



Let $n(T)$ Represent the number of vertices for a tree T

Proof: We will use induction to prove this claim; specifically structural induction.

Basis Step: Let T be a full binary tree comprised of a single Root r.

$\Rightarrow$ $n(T) = 1$ which is odd ✓

Induction

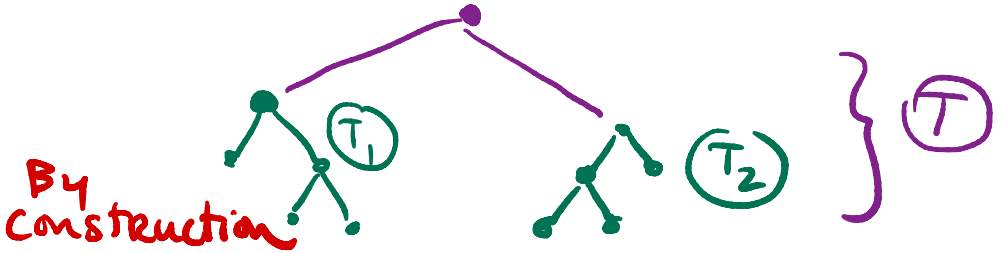
(continued)

Example: Prove the claim below using structural induction.
If T is a full binary tree, then T has an odd number of vertices.

Recursive Step: For the inductive hypothesis, assume that T_1 and T_2 are full binary trees.

$n(T_1) = 2p+1$ and $n(T_2) = 2q+1$ for some nonnegative integers p, q

Consider $T = T_1 \cdot T_2$



By construction

$\Rightarrow n(T) = n(T_1) + n(T_2) + 1$

By Inductive Hypothesis

$= 2p+1 + 2q+1 + 1$

$= 2p + 2q + 2 + 1$

$= 2(p+q+1) + 1$

$\Rightarrow n(T)$ is odd!

Thus if T is an FBT, then $n(T)$ is odd. ■

Induction

Example: Prove the claim below using induction.
If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

[Here, $n(T)$ represents the number of vertices in tree T. And $h(T)$ represents the height of tree T. The height of a tree is defined to be the length of the longest path from the root to a leaf.

Proof: We prove by structural induction.

Basis Step: Let T be a Full Binary tree comprised of a single root r.

$\Rightarrow n(T) = 1$

$h(T) = 0$

$\Rightarrow n(T) = 1 \leq 2^{0+1} - 1$

$= 2^1 - 1$

$= 1$

$1 \leq 1$ ✓

Induction

(continued)

Example: Prove the claim below using induction.

If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

[Here, $n(T)$ represents the number of vertices in tree T. And $h(T)$ represents the height of tree T. The height of a tree is defined to be the length of the longest path from the root to a leaf.

Recursive Step: For the inductive hypothesis, assume that whenever T_1 and T_2 are FBTs, $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$

Recall that if $T = T_1 \cdot T_2$ then $n(T) = n(T_1) + n(T_2) + 1$

$h(T) = 1 + \max(h(T_1), h(T_2))$

$n(T) = n(T_1) + n(T_2) + 1$

$\leq (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) + 1$ By the inductive hypothesis. →

Induction

(continued)

Example: Prove the claim below using induction.
If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

[Here, $n(T)$ represents the number of vertices in tree T. And $h(T)$ represents the height of tree T. The height of a tree is defined to be the length of the longest path from the root to a leaf.

$$n(T) \leq 2^{h(T_1)+1} + 2^{h(T_2)+1} - 1$$

$$\leq 2 \cdot \max\left(2^{h(T_1)+1}, 2^{h(T_2)+1}\right) - 1$$

$$= 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1$$

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

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

$\Rightarrow$ Thus $n(T) \leq 2^{h(T)+1} - 1$ as desired. ■

Next Time:

- Asymptotic Analysis