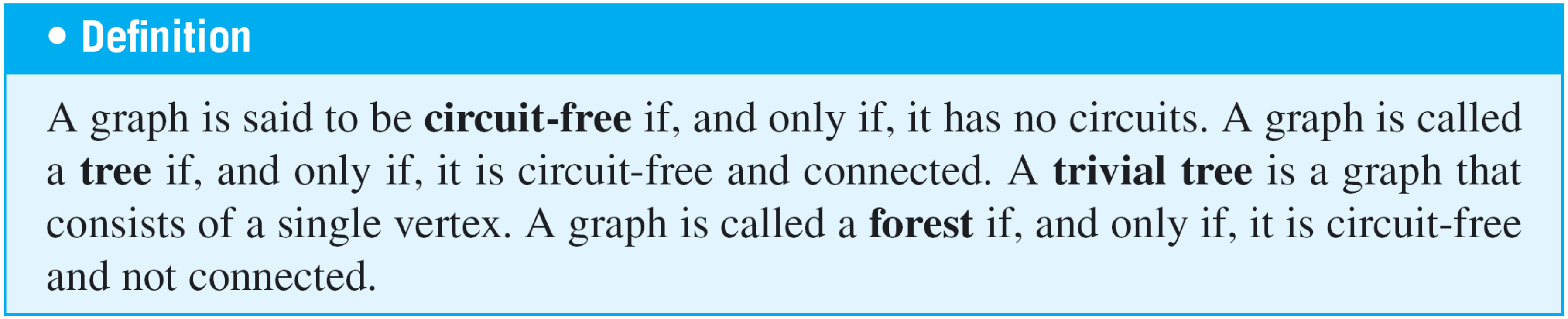
CMSC 207- Lecture 29
CHAPTER 10: Graphs And Trees
(10.5 and 10.6 - Trees)

Dr. Ahmed Tarek

Trees

- A tree is a connected graph that does not contain any circuits.
- Mathematical trees are similar in certain ways to their botanical namesakes.



• **Definition**

A graph is said to be **circuit-free** if, and only if, it has no circuits. A graph is called a **tree** if, and only if, it is circuit-free and connected. A **trivial tree** is a graph that consists of a single vertex. A graph is called a **forest** if, and only if, it is circuit-free and not connected.

Characterizing Trees

- If ***n*** is a positive integer, then any tree with ***n* vertices** has ***n* – 1 edges**.
- Any *connected* graph with ***n* vertices** and ***n* – 1 edges** is **a tree**.
- If **one new edge (but no new vertex)** is added to a tree, the resulting graph must contain a circuit.

Characterizing Trees

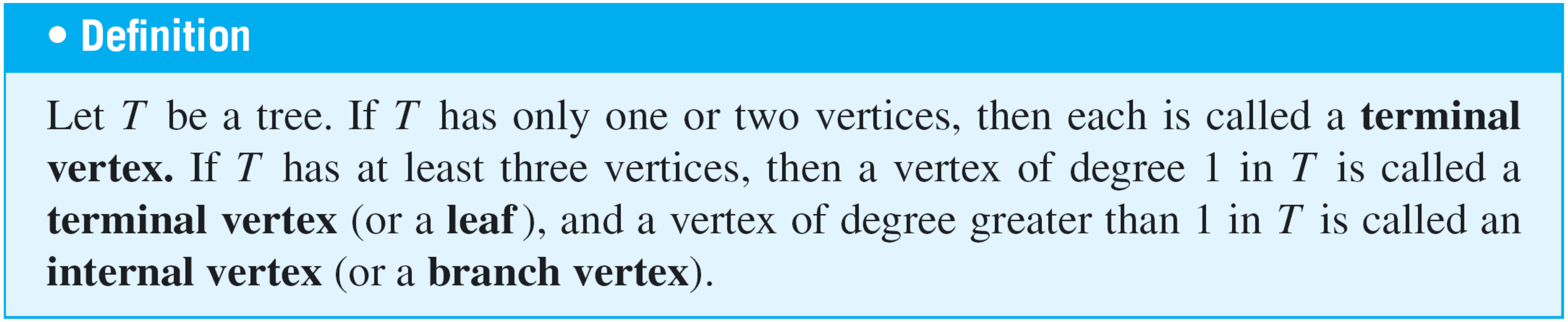
- Also, from the fact that removing an edge from a circuit does not disconnect a graph, it can be shown that every connected graph has a subgraph that is a tree.
- It follows that if *n* is a positive integer, any graph with *n* vertices and *fewer* than *n* – 1 edges is not connected.
- Any nontrivial tree must have at least one vertex of degree 1.

Lemma 10.5.1

Any tree that has more than one vertex has at least one vertex of degree 1.

Characterizing Trees

- Using Lemma 10.5.1 it is not difficult to show that, in fact, any tree that has more than one vertex has at least *two* vertices of degree 1.

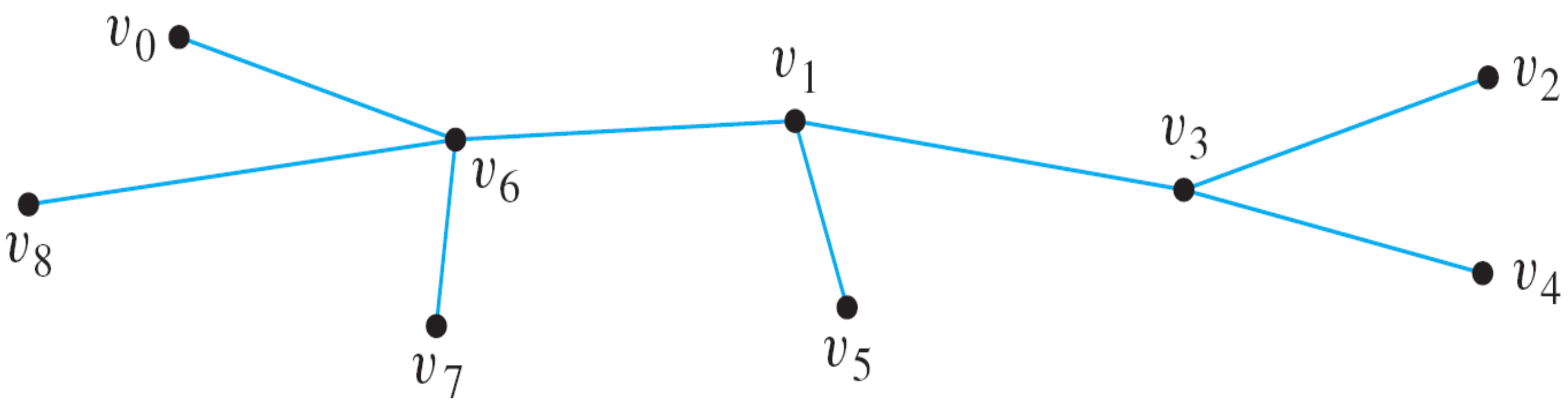


- **Definition**

Let T be a tree. If T has only one or two vertices, then each is called a **terminal vertex.** If T has at least three vertices, then a vertex of degree 1 in T is called a **terminal vertex** (or a **leaf**), and a vertex of degree greater than 1 in T is called an **internal vertex** (or a **branch vertex**).

Example – *Terminal and Internal Vertices*

- Find all terminal vertices and all internal vertices in the following tree:



- **Solution:**

- The terminal vertices are v_0, v_2, v_4, v_5, v_7, and v_8.
- The internal vertices are v_6, v_1, and v_3.

Characterizing Trees

Theorem 10.5.2

For any positive integer n, any tree with n vertices has $n - 1$ edges.

Two graphs that are the same except for the labeling of their vertices and edges are called *isomorphic*.

- **Definition**

Let G and G' be graphs with vertex sets $V(G)$ and $V(G')$ and edge sets $E(G)$ and $E(G')$, respectively. **G is isomorphic to G'** if, and only if, there exist one-to-one correspondences $g\colon V(G) \to V(G')$ and $h\colon E(G) \to E(G')$ that preserve the edge-endpoint functions of G and G' in the sense that for all $v \in V(G)$ and $e \in E(G)$,

$$v \text{ is an endpoint of } e \quad \Leftrightarrow \quad g(v) \text{ is an endpoint of } h(e). \qquad 10.4.1$$

Example – *Finding Trees Satisfying Given Conditions*

- Find all **nonisomorphic trees** with four vertices.

•Solution:

- By Theorem 10.5.2, any tree with four vertices has three edges. Thus the total degree of a tree with four vertices must be **6**.

Theorem 10.5.2

For any positive integer n, any tree with n vertices has $n - 1$ edges.

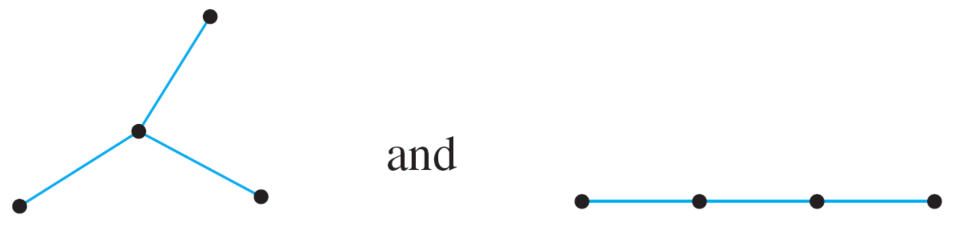
- Also, every tree with more than one vertex has at least two vertices of degree 1.

Example – *Solution*

•Thus the following combinations of degrees for the vertices are the only ones possible:

1, 1, 1, 3 and **1, 1, 2, 2.**

•There are **two nonisomorphic trees** corresponding to both of these possibilities, as shown below.



Characterizing Trees

Lemma 10.5.3

If G is any connected graph, C is any circuit in G, and any one of the edges of C is removed from G, then the graph that remains is connected.

- Lemma 10.5.3 is true as any two vertices in a circuit are connected by two distinct paths.
- It is possible to draw the graph so that one of these goes "**clockwise**" and the other goes "**counterclockwise**" around the circuit.

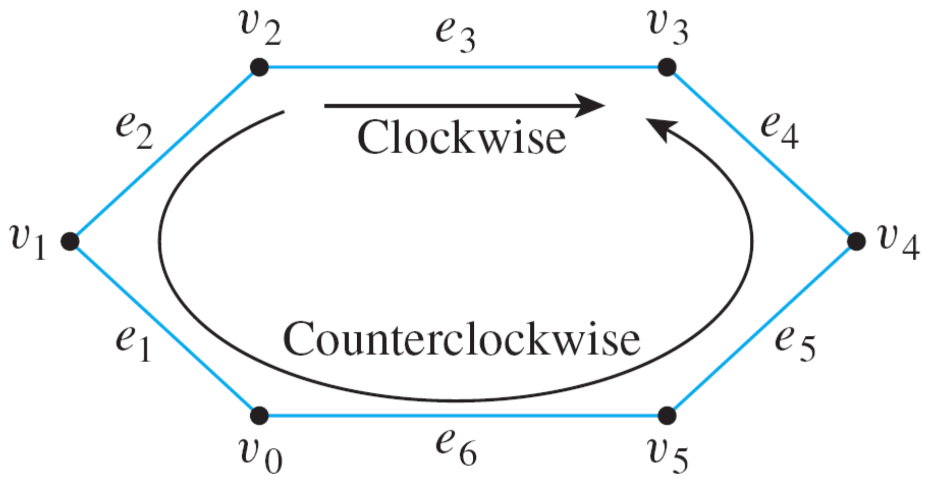
Characterizing Trees

- For example, in the circuit shown below, the clockwise path from v_2 to v_3 is

$$v_2 e_3 v_3$$

- and the counterclockwise path from v_2 to v_3 is

$$v_2 e_2 v_1 e_1 v_0 e_6 v_5 e_5 v_4 e_4 v_3 .$$



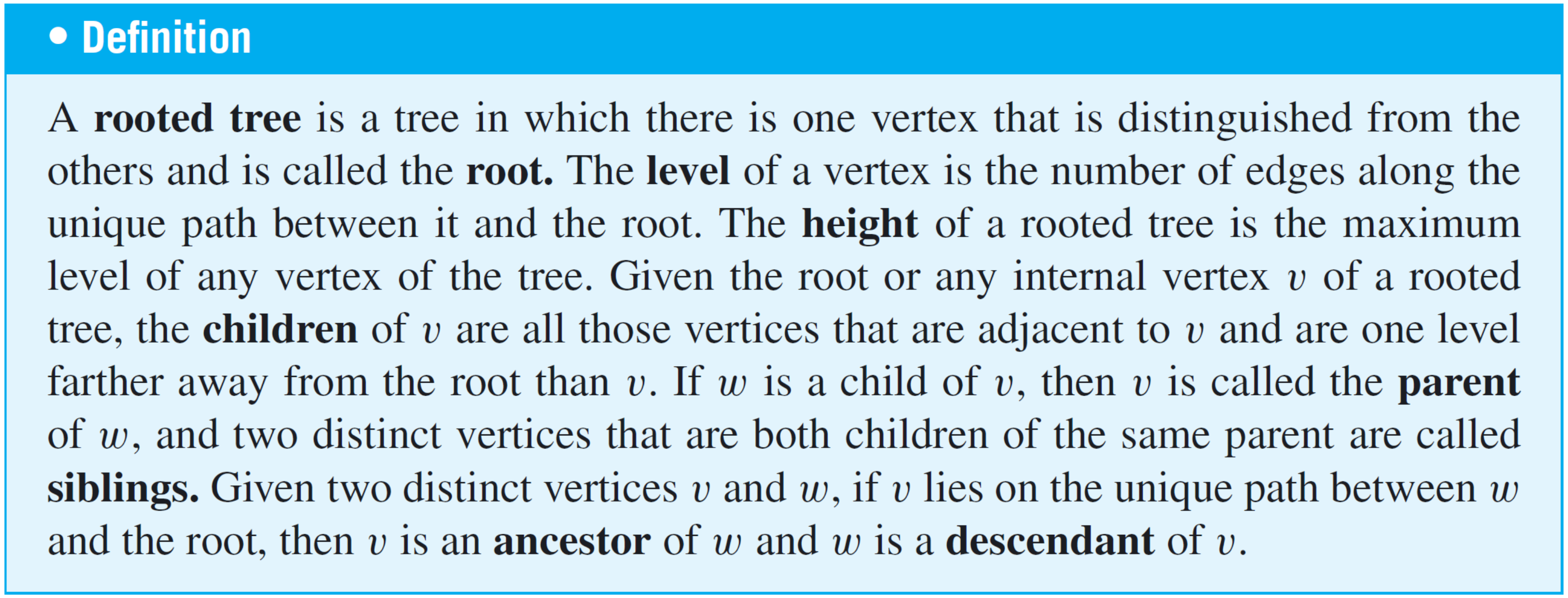
Rooted Trees

- In mathematics, a rooted tree is a tree in which one vertex has been distinguished from the others and is designated the *root*.

- Given any other vertex *v* in the tree, there is a unique path from the root to *v*. (If there were two distinct paths, a circuit could be constructed.)

Rooted Trees

- The number of edges in such a path is called the *level* of *v*, and the *height* of the tree is the length of the longest such path. It is traditional in drawing rooted trees to place the root at the top and show the branches descending from it.

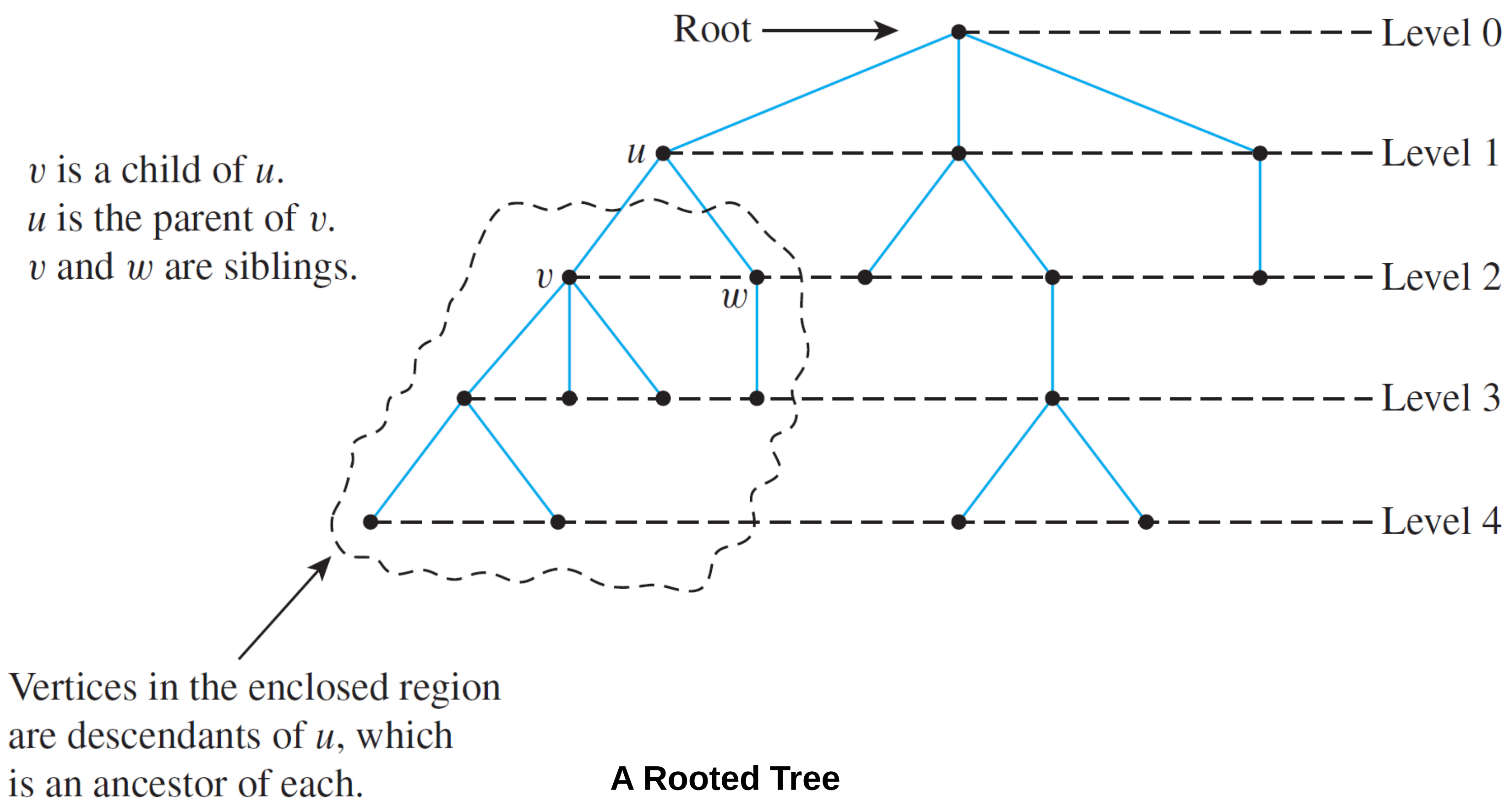


• Definition

A **rooted tree** is a tree in which there is one vertex that is distinguished from the others and is called the **root.** The **level** of a vertex is the number of edges along the unique path between it and the root. The **height** of a rooted tree is the maximum level of any vertex of the tree. Given the root or any internal vertex v of a rooted tree, the **children** of v are all those vertices that are adjacent to v and are one level farther away from the root than v. If w is a child of v, then v is called the **parent** of w, and two distinct vertices that are both children of the same parent are called **siblings.** Given two distinct vertices v and w, if v lies on the unique path between w and the root, then v is an **ancestor** of w and w is a **descendant** of v.

Rooted Trees

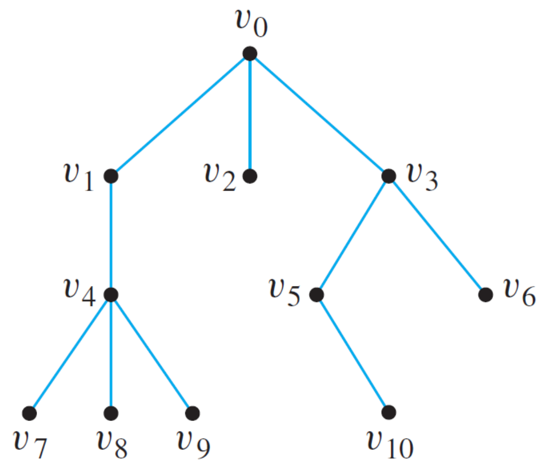
- These terms are illustrated in Figure 10.6.1.



A Rooted Tree

Example – *Rooted Trees*

- Consider the tree with root v_0 shown on the right.
- **a.** What is the level of v_5?
- **b.** What is the level of v_0?
- **c.** What is the height of this rooted tree?
- **d.** What are the children of v_3?
- **e.** What is the parent of v_2?
- **f.** What are the siblings of v_8?
- **g.** What are the descendants of v_3?

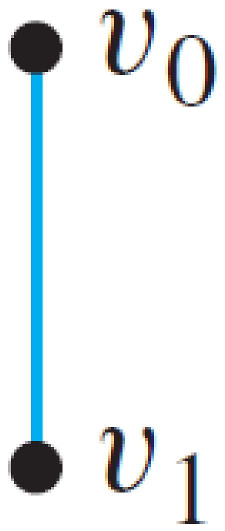


Example – *Solution*

- **a.** 2
- **b.** 0
- **c.** 3
- **d.** v_5 and v_6
- **e.** v_0
- **f.** v_7 and v_9
- **g.** v_5, v_6, v_{10}

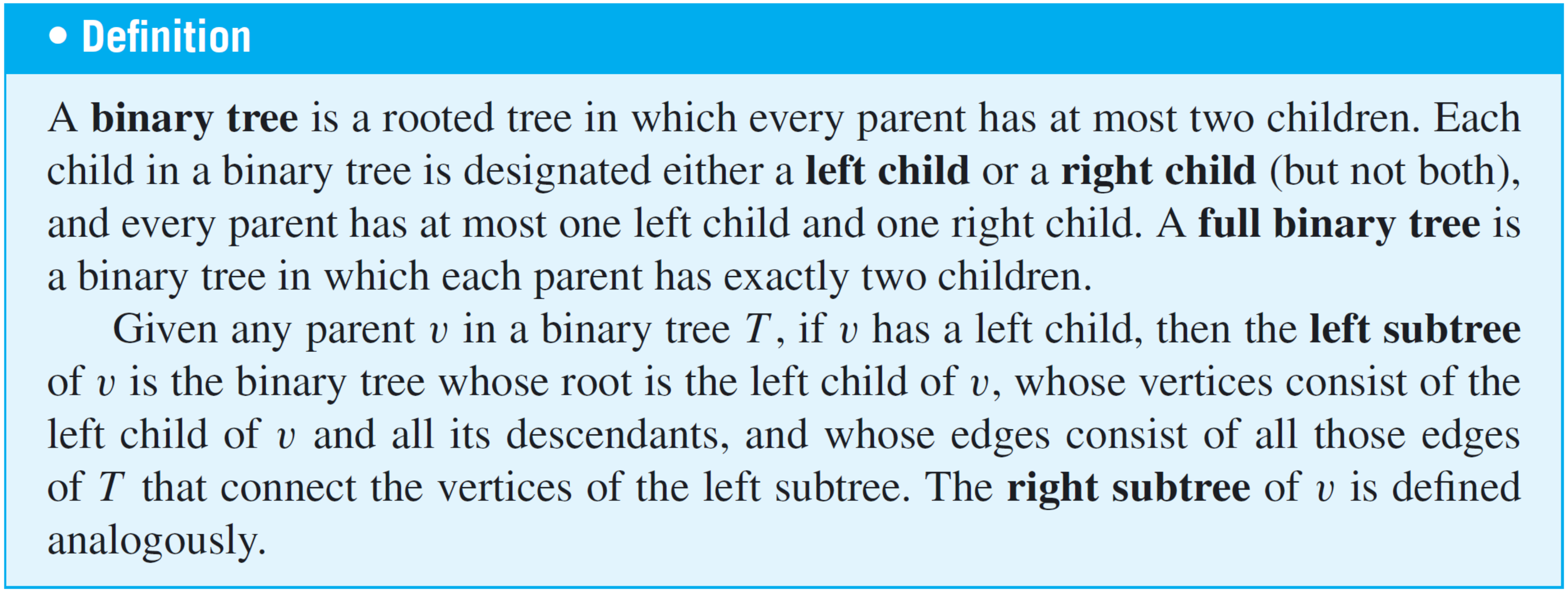
Rooted Trees

- In the tree with root ***v_0*** shown below, ***v_1*** has **level 1** and is the child of v_0, and both v_0 and v_1 are terminal vertices.



Binary Trees

- **When every vertex in a rooted tree has at most two children and each child is designated either the (unique) left child or the (unique) right child, the result is a *binary tree*.**



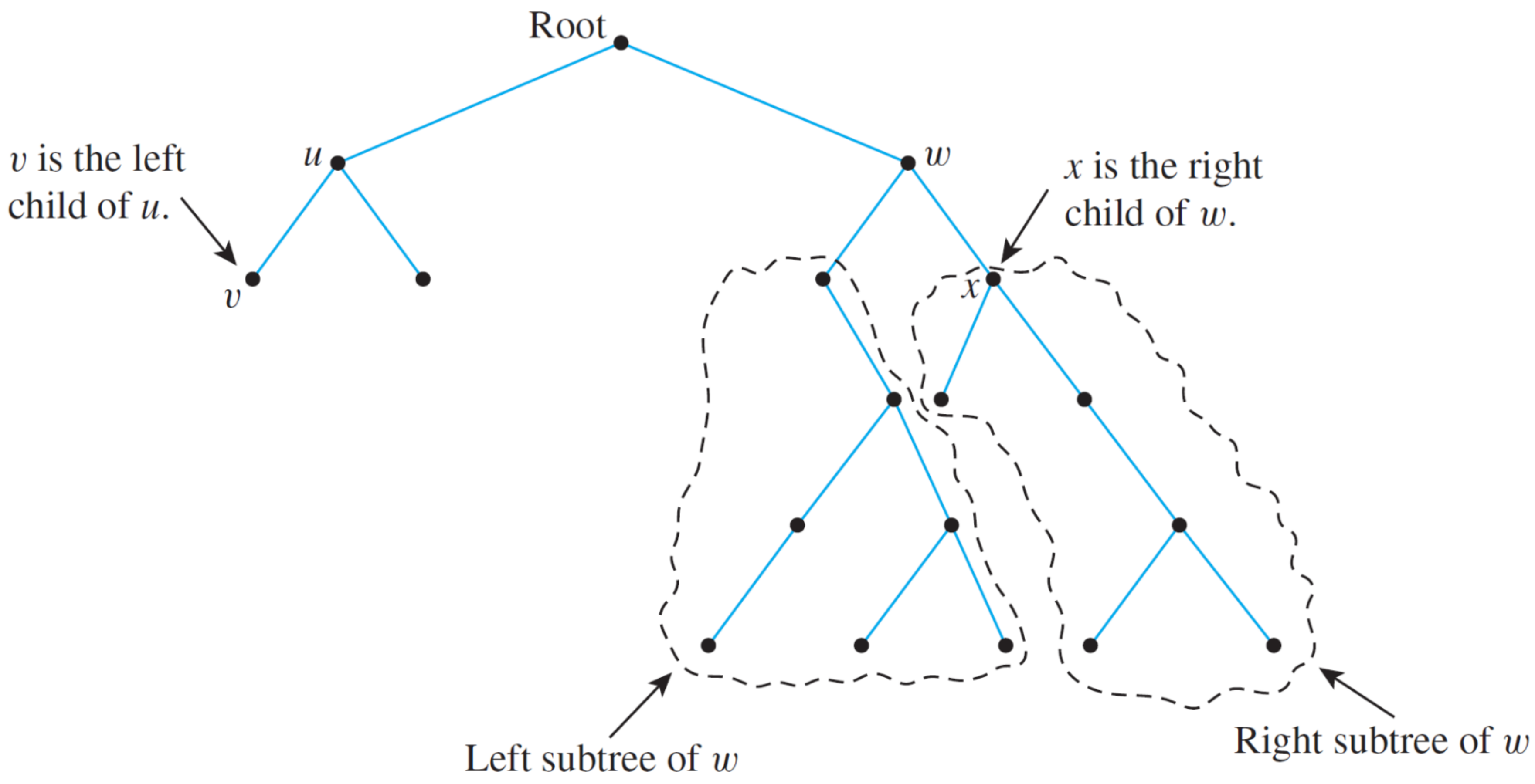
• **Definition**

A **binary tree** is a rooted tree in which every parent has at most two children. Each child in a binary tree is designated either a **left child** or a **right child** (but not both), and every parent has at most one left child and one right child. A **full binary tree** is a binary tree in which each parent has exactly two children.

Given any parent v in a binary tree T, if v has a left child, then the **left subtree** of v is the binary tree whose root is the left child of v, whose vertices consist of the left child of v and all its descendants, and whose edges consist of all those edges of T that connect the vertices of the left subtree. The **right subtree** of v is defined analogously.

Binary Trees

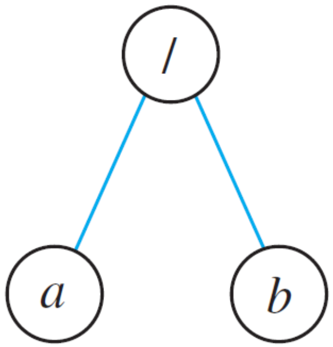
- These terms are illustrated in Figure.



A Binary Tree

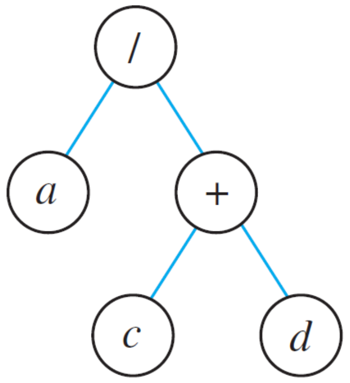
Example – *Representation of Algebraic Expressions*

- Binary trees are used in many ways in computer science. One use is to represent algebraic expressions with arbitrary nesting of balanced parentheses. For instance, the following (labeled) binary tree represents the expression *a*/*b*: The operator is at the root and acts on the left and right children of the root in left-right order.



Example – *Representation of Algebraic Expressions*

- More generally, the binary tree shown below represents the expression *a*/(*c* + *d*). In such a representation, the internal vertices are arithmetic operators, the terminal vertices are variables, and the operator at each vertex acts on its left and right subtrees in left-right order.



In-class Assignment #1

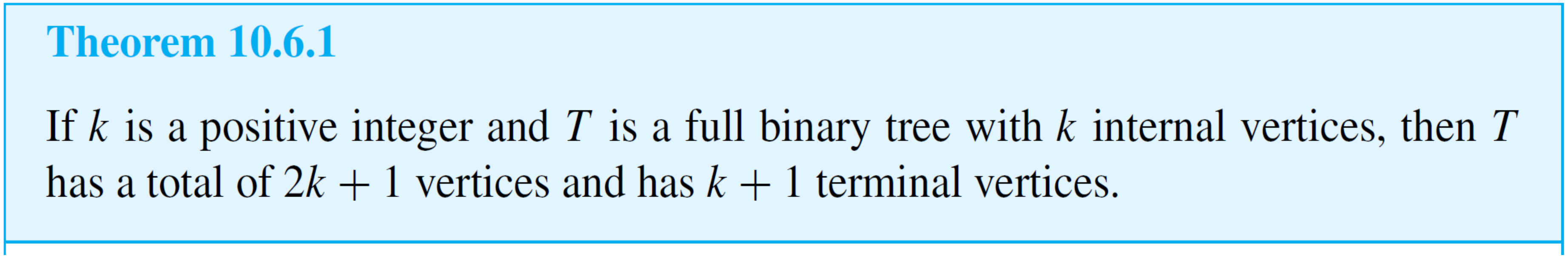
Draw the binary tree to represent the expression

((*a* – *b*) • *c*) + (*d*/*e*).

Hints: Start with the innermost parentheses, and then extend outward.

Binary Trees

•An interesting theorem about binary trees says that if you know the **number of internal vertices** of a **full binary tree**, then you can calculate both the total number of vertices and the number of terminal vertices, and conversely. For a full binary tree with ***k* internal vertices** has a total of **2*k* + 1 vertices** of which ***k* + 1 are terminal vertices**.



Theorem 10.6.1

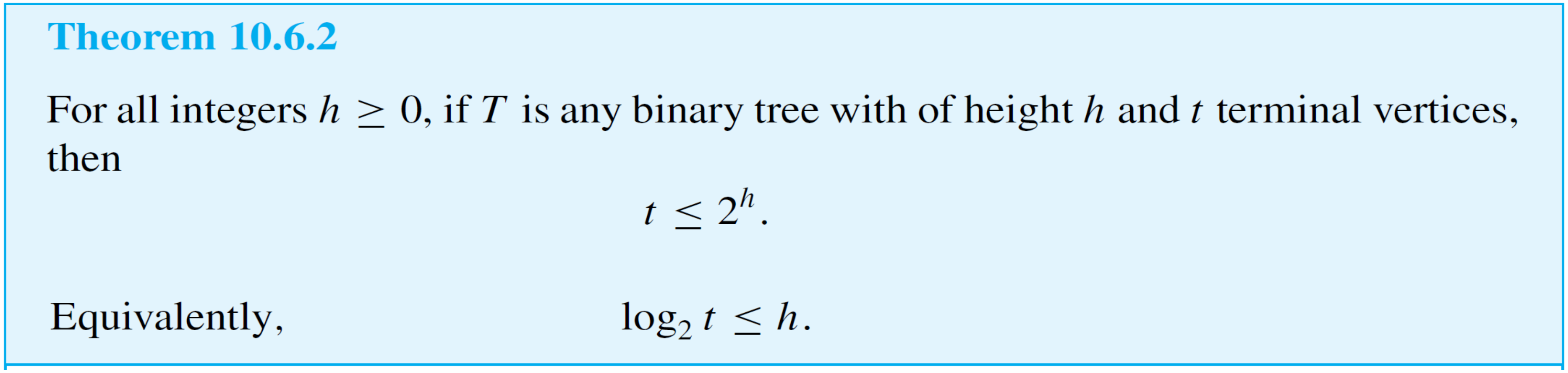
If k is a positive integer and T is a full binary tree with k internal vertices, then T has a total of $2k + 1$ vertices and has $k + 1$ terminal vertices.

Example – *Determining Whether a Certain Full Binary Tree Exists*

- Is there a full binary tree that has 10 internal vertices and 13 terminal vertices?
- **Solution:**
- **No.** By **Theorem 10.6**.1, a full binary tree with 10 internal vertices has 10 + 1 = 11 terminal vertices, not 13.

Binary Trees

- Another interesting theorem about binary trees specifies the maximum number of terminal vertices of a binary tree of a given height. Specifically, **the maximum number of terminal vertices of a binary tree of height *h* is 2^h**. Another way to say this is that a binary tree with *t* terminal vertices has height of at least $\mathbf{\log_2 t}$.



Theorem 10.6.2

For all integers $h \geq 0$, if T is any binary tree with of height h and t terminal vertices, then

$$t \leq 2^h.$$

Equivalently, $$\log_2 t \leq h.$$

Example – ***Determining Whether a Certain Binary Tree Exists***

- Is there a binary tree that has height 5 and 38 terminal vertices?
- **Solution:**
- No. By **Theorem 10.6.2**, any binary tree ***T*** with **height 5** has at most $\mathbf{2^5 = 32}$ terminal vertices. So such a tree cannot have 38 terminal vertices.

In-class Assignment #2

- Draw a Full Binary Tree with **five internal vertices**.
- Draw a Full Binary Tree with **nine vertices in total**.