Tree

Wednesday, 11 September 2024 9:54 pm

10.4 Trees: Examples and Basic Properties

Screen clipping taken: 11/09/2024 9:55 pm

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.

Screen clipping taken: 11/09/2024 9:55 pm

Example 10.4.1 Trees and Non-trees

All the graphs shown in Figure 10.4.1 are trees, whereas those in Figure 10.4.2 are not.

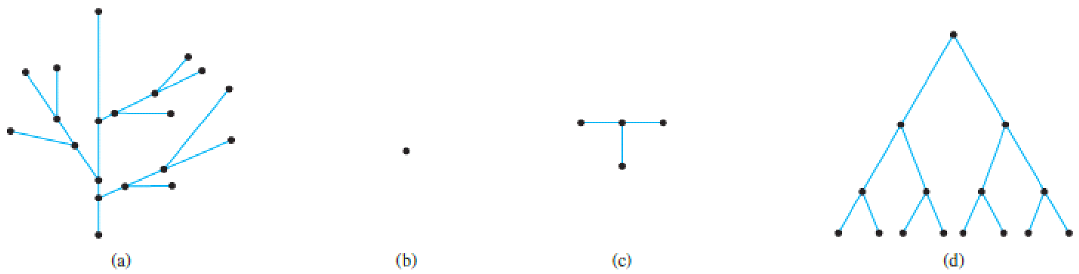


FIGURE 10.4.1 Trees. All the graphs in (a)–(d) are connected and circuit-free.

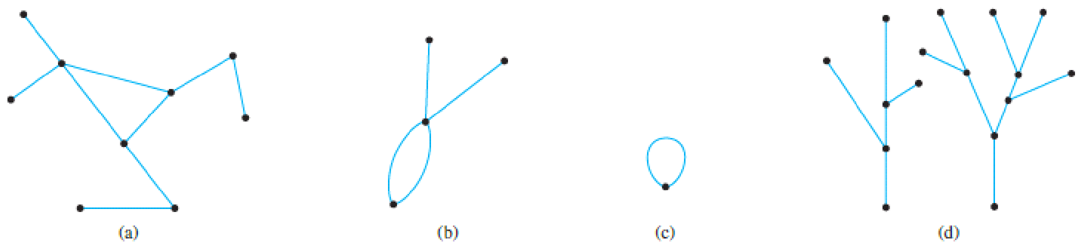


FIGURE 10.4.2 Non-trees. The graphs in (a), (b), and (c) all have circuits, and the graph in (d) is not connected. ■

Screen clipping taken: 11/09/2024 9:55 pm

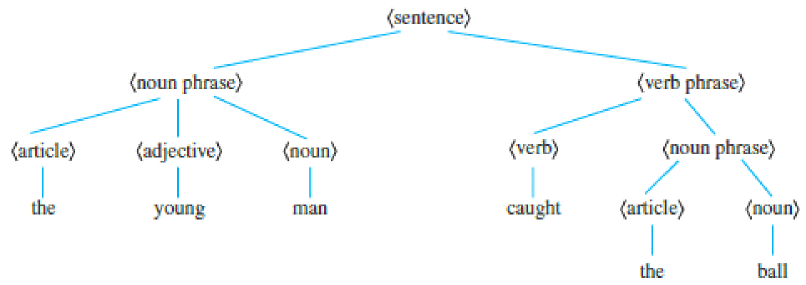
Example 10.4.2 A Decision Tree

Screen clipping taken: 11/09/2024 9:57 pm

Example 10.4.3 A Parse Tree

Screen clipping taken: 11/09/2024 9:57 pm

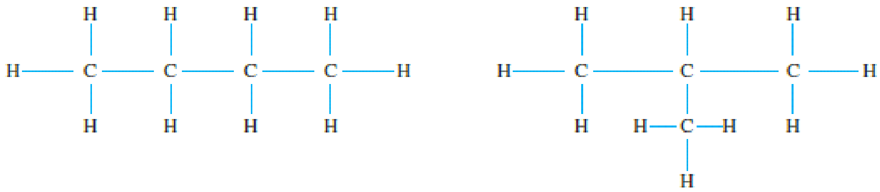
The derivation of the sentence "The young man caught the ball" from the above rules is described by the tree shown below.



Screen clipping taken: 11/09/2024 9:57 pm

Example 10.4.4 **Structure of Hydrocarbon Molecules**

Screen clipping taken: 11/09/2024 9:58 pm



Butane   Isobutane

Screen clipping taken: 11/09/2024 9:58 pm

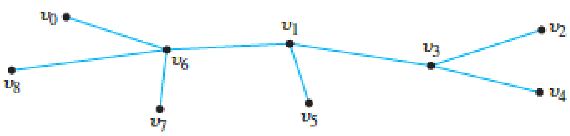
Definition

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

Screen clipping taken: 11/09/2024 10:03 pm

Example 10.4.5 **Leaves and Internal Vertices in Trees**

Find all leaves (or terminal vertices) and all internal (or branch) vertices in the following tree:



Solution The leaves (or terminal vertices) are v_0, v_2, v_4, v_5, v_7, and v_8. The internal (or branch) vertices are v_6, v_1, and v_3. ■

Screen clipping taken: 11/09/2024 10:04 pm

Theorem 10.4.2

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

Screen clipping taken: 11/09/2024 10:05 pm

10.5 Rooted Trees

Screen clipping taken: 11/09/2024 10:19 pm

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.

Screen clipping taken: 11/09/2024 10:07 pm

These terms are illustrated in Figure 10.5.1.

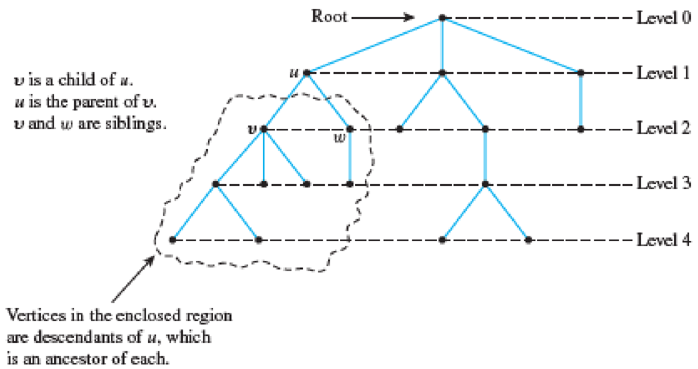


FIGURE 10.5.1 A Rooted Tree

Example 10.5.1 **Rooted Trees**

Consider the tree with root v_0 shown below.

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?

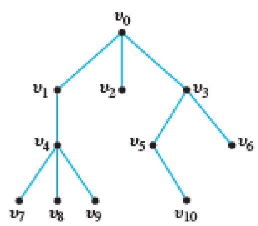
d. What are the children of v_3?

e. What is the parent of v_2?

f. What are the siblings of v_8?

g. What are the descendants of v_3?

h. How many leaves (terminal vertices) are on the tree?



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} h. 6

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.

These terms are illustrated in Figure 10.5.2.

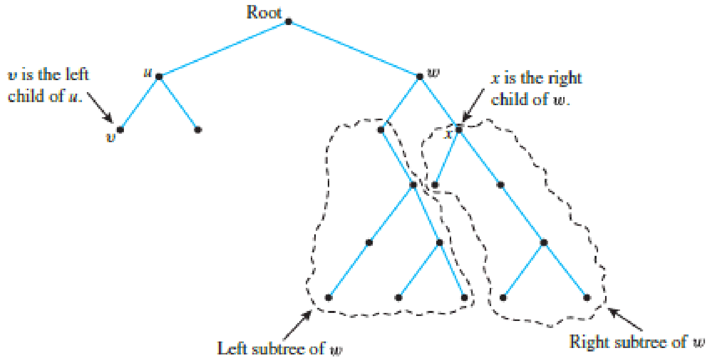


FIGURE 10.5.2 A Binary Tree

Spanning Trees and a Shortest Path Algorithm

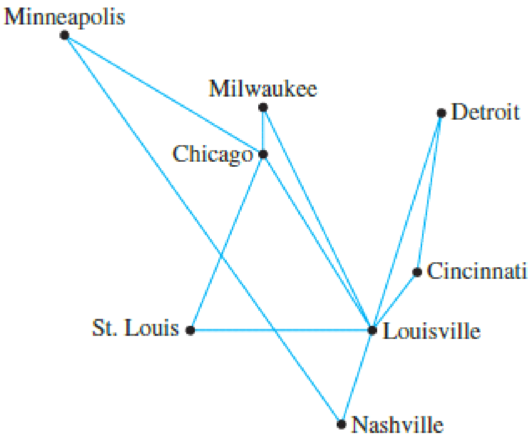


FIGURE 10.6.1

Screen clipping taken: 11/09/2024 11:13 pm

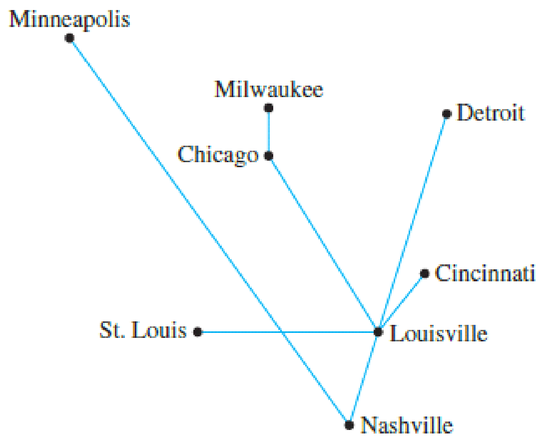


FIGURE 10.6.2

Screen clipping taken: 11/09/2024 11:14 pm

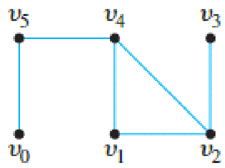
Definition

A **spanning tree** for a graph G is a subgraph of G that contains every vertex of G and is a tree.

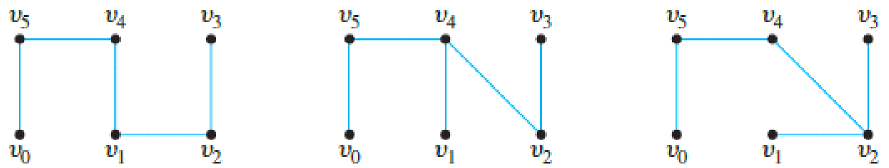
Screen clipping taken: 11/09/2024 11:14 pm

Spanning Trees

Find all spanning trees for the graph G pictured below.



Solution The graph G has one circuit $v_2v_1v_4v_2$, and removing any edge of the circuit gives a tree. Thus, as shown below, there are three spanning trees for G.



Screen clipping taken: 11/09/2024 11:14 pm

Definition and Notation

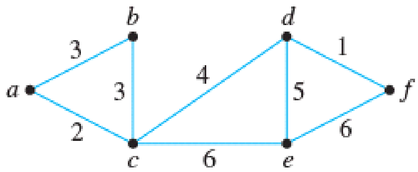
A **weighted graph** is a graph for which each edge has an associated positive real number **weight**. The sum of the weights of all the edges is the **total weight** of the graph. A **minimum spanning tree** for a connected, weighted graph is a spanning tree that has the least possible total weight compared to all other spanning trees for the graph.

If G is a weighed graph and e is an edge of G, then $w(e)$ denotes the weight of e and $w(G)$ denotes the total weight of G.

Screen clipping taken: 11/09/2024 11:15 pm

Example 10.6.4 Finding Minimum Spanning Trees

Find all minimum spanning trees for the following graph. Use Kruskal's algorithm and Prim's algorithm starting at vertex a. Indicate the order in which edges are added to form each tree.



Screen clipping taken: 11/09/2024 11:18 pm

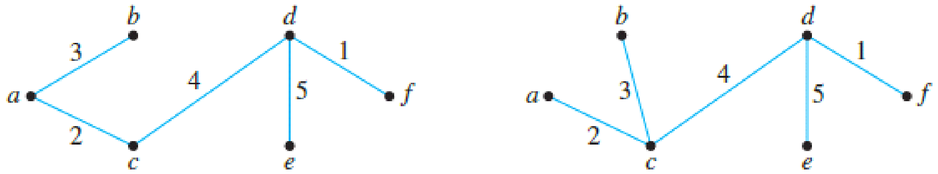
Solution When Kruskal's algorithm is applied, edges are added in one of the following two orders:

1. $\{d, f\}$, $\{a, c\}$, $\{a, b\}$, $\{c, d\}$, $\{d, e\}$
2. $\{d, f\}$, $\{a, c\}$, $\{b, c\}$, $\{c, d\}$, $\{d, e\}$

When Prim's algorithm is applied starting at a, edges are added in one of the following two orders:

1. $\{a, c\}$, $\{a, b\}$, $\{c, d\}$, $\{d, f\}$, $\{d, e\}$
2. $\{a, c\}$, $\{b, c\}$, $\{c, d\}$, $\{d, f\}$, $\{d, e\}$

Thus, as shown below, there are two distinct minimum spanning trees for this graph.



(a) (b)

Screen clipping taken: 11/09/2024 11:18 pm

Algorithm 10.6.1 Kruskal

Input: *G [a connected, weighted graph with n vertices, where n is a positive integer]*

Algorithm Body:

[Build a subgraph T of G to consist of all the vertices of G with edges added in order of increasing weight. At each stage, let m be the number of edges of T.]

Screen clipping taken: 11/09/2024 11:23 pm

Algorithm 10.6.2

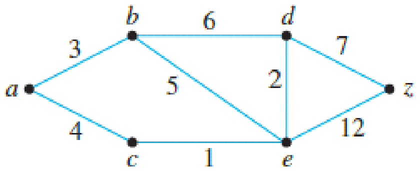
Input: *G [a connected, weighted graph with n vertices where n is a positive integer]*

Algorithm Body:

[Build a subgraph T of G by starting with any vertex v of G and attaching edges (with their endpoints) one by one to an as-yet-unconnected vertex of G, each time choosing an edge of least weight that is adjacent to a vertex of T.]

Action of Dijkstra's Algorithm

Show the steps in the execution of Dijkstra's shortest path algorithm for the graph shown below with starting vertex a and ending vertex z.



Screen clipping taken: 11/09/2024 11:26 pm