Trees

0 Contents

1 Introduction to Trees

2 Applications of Trees

3 Tree Traversal

4 Spanning Trees

5 Minimum Spanning Trees

1 Introduction to Trees

DEFINITION

A *tree* is a connected undirected graph with no simple circuits.

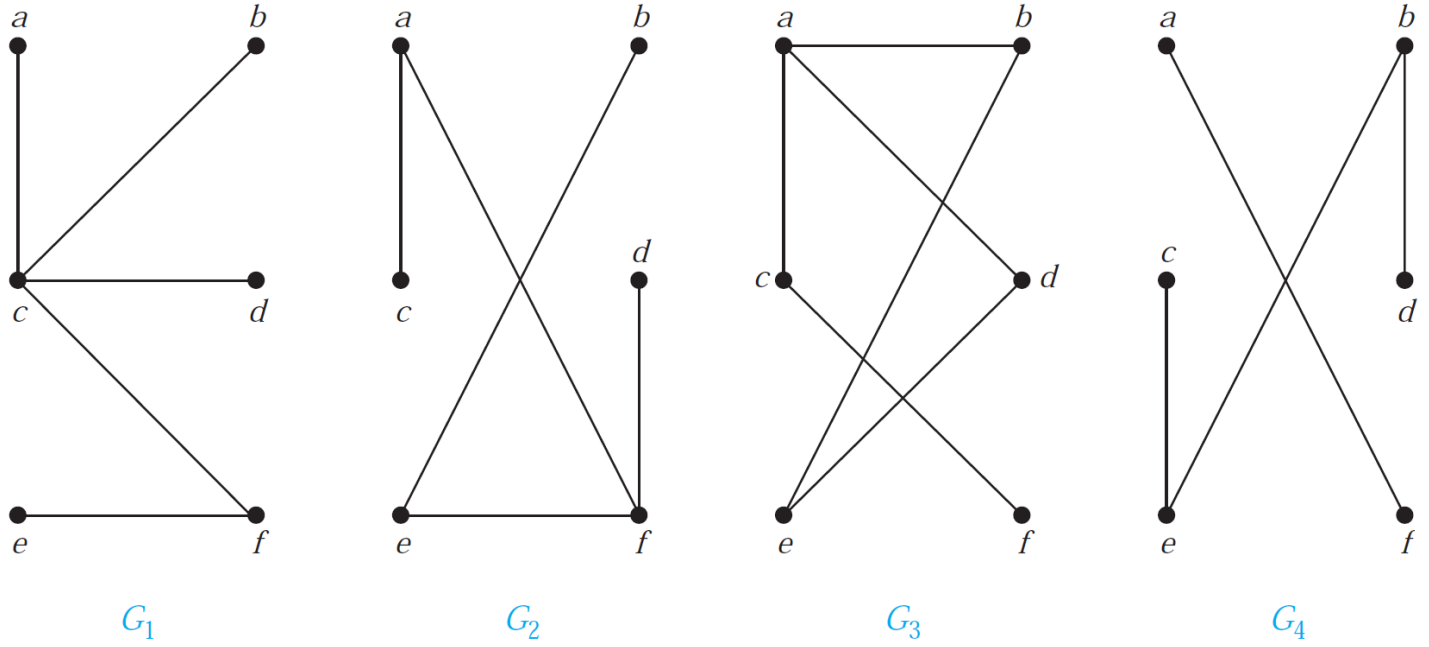


FIGURE 2 **Examples of Trees and Graphs That Are Not Trees.**

1 Introduction to Trees – Forests

DEFINITION

Graphs containing no simple circuits that are not connected are called **forests** and have the property that each of their connected components is a tree.

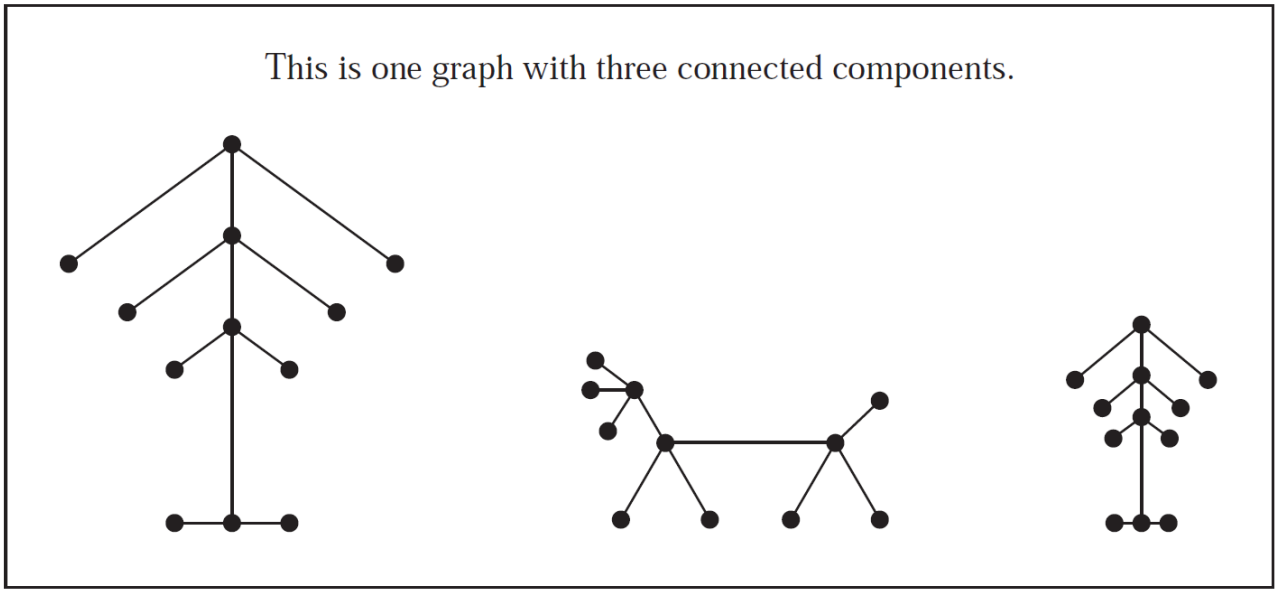


FIGURE 3 **Example of a Forest.**

1 Introduction to Trees – Rooted Trees

DEFINITION

A *rooted tree* is a tree in which one vertex has been designated as root and every edge is directed away from the root.

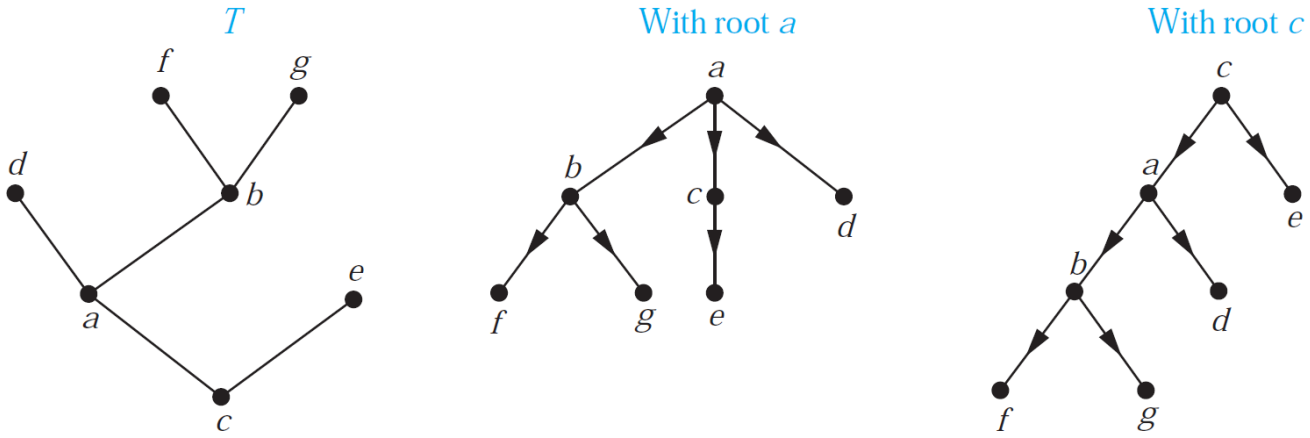


FIGURE 4 **A Tree and Rooted Trees Formed by Designating Two Different Roots.**

1 Introduction to Trees – Rooted Trees

Terminology

Suppose that T is a rooted tree:

- If v is a vertex in T other than the root, the **parent** of v is the unique vertex u such that there is a directed edge from u to v.
- When u is the parent of v, v is called a **child** of u.
- Vertices with the same parent are called **siblings**.
- The **ancestors** of a vertex other than the root are the vertices in the path from the root to this vertex.
- The **descendants** of a vertex v are those vertices that have v as an ancestor.
- A vertex of a rooted tree is called a **leaf** if it has no children.
- Vertices that have children are called **internal vertices**.

1 Introduction to Trees – Rooted Trees

Terminology

Suppose that T is a rooted tree:

- If a is a vertex in a tree, the **subtree** with a as its root is the subgraph of the tree consisting of a and its descendants and all edges incident to these descendants.

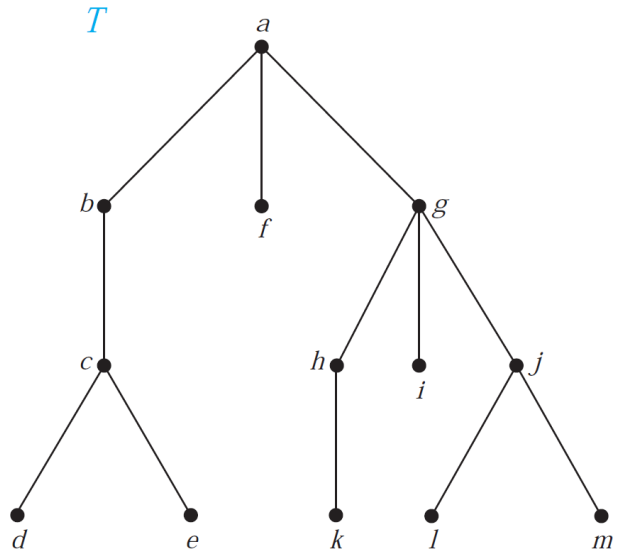


FIGURE 5 A Rooted Tree T.

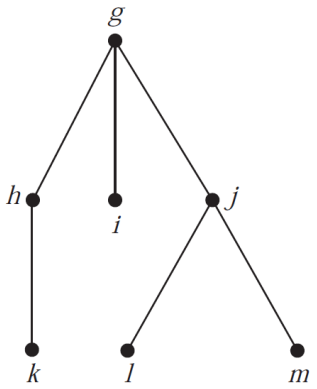


FIGURE 6 The Subtree Rooted at g.

1 Introduction to Trees – Rooted Trees

DEFINITION

A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children.

The tree is called a *full m-ary tree* if every internal vertex has exactly m children.

An *m-ary* tree with $m = 2$ is called a *binary tree*.

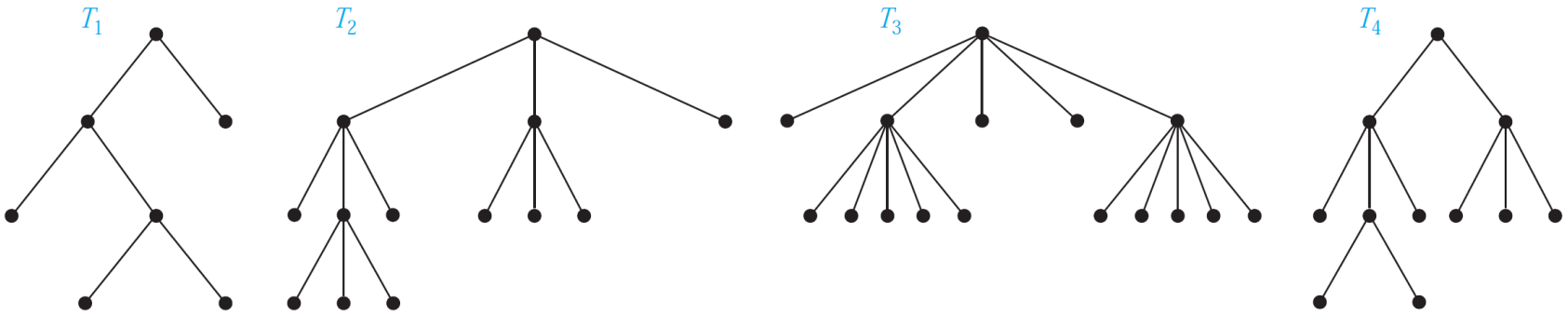


FIGURE 7 **Four Rooted Trees.**

1 Introduction to Trees – Rooted Trees

Ordered Rooted Trees

The children of each internal vertex are ordered and shown in order from left to right.

In an ordered binary tree, if an internal vertex has two children, the first child is called the **left child** and the second child is called the **right child**.

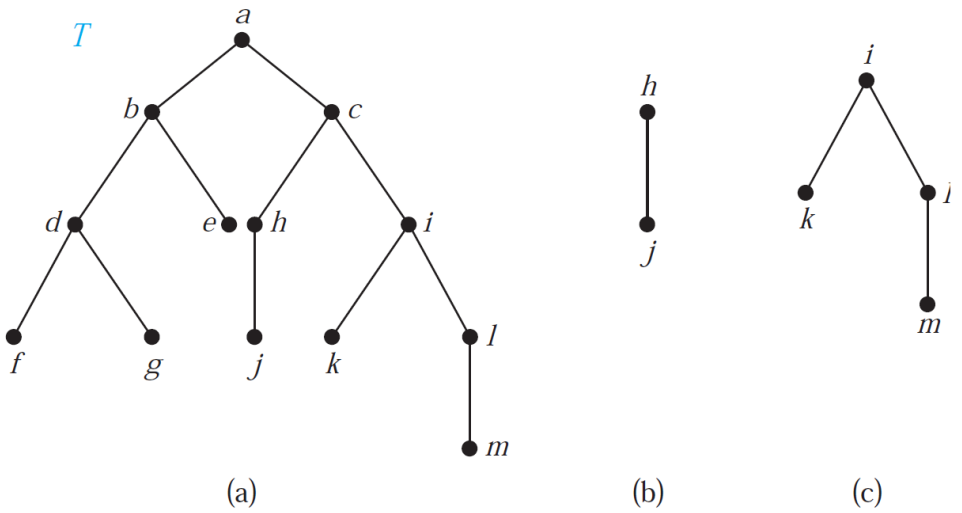


FIGURE 8 A Binary Tree T and Left and Right Subtrees of the Vertex c.

1 Introduction to Trees – Trees as Models

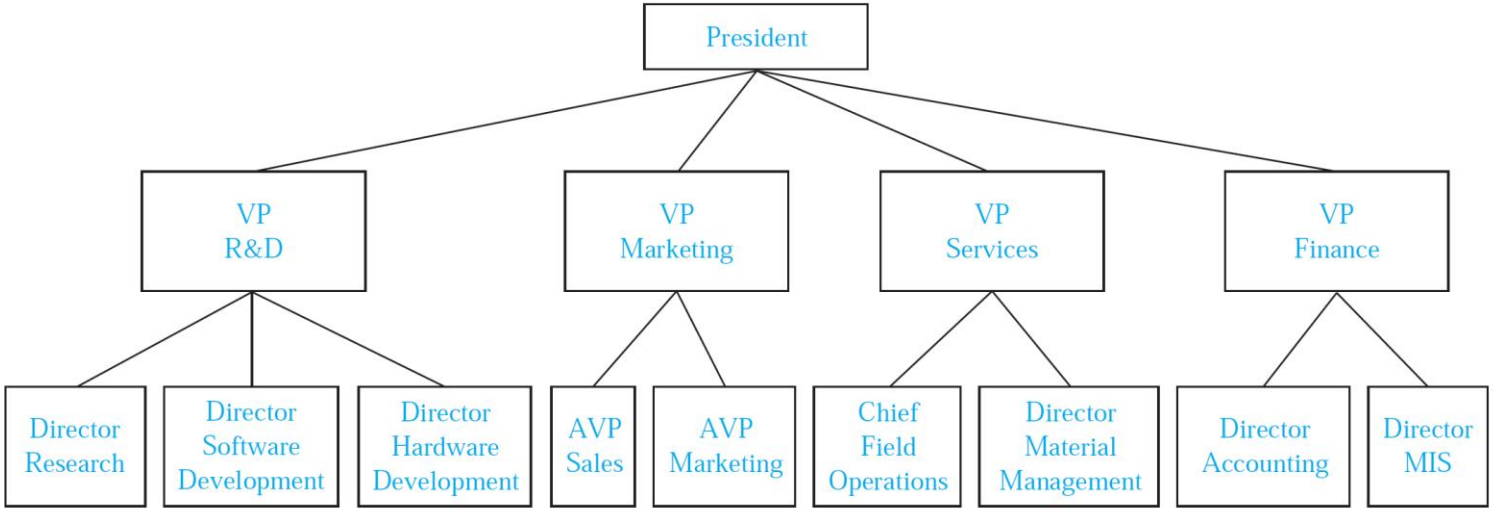


FIGURE 10 An Organizational Tree for a Computer Company.

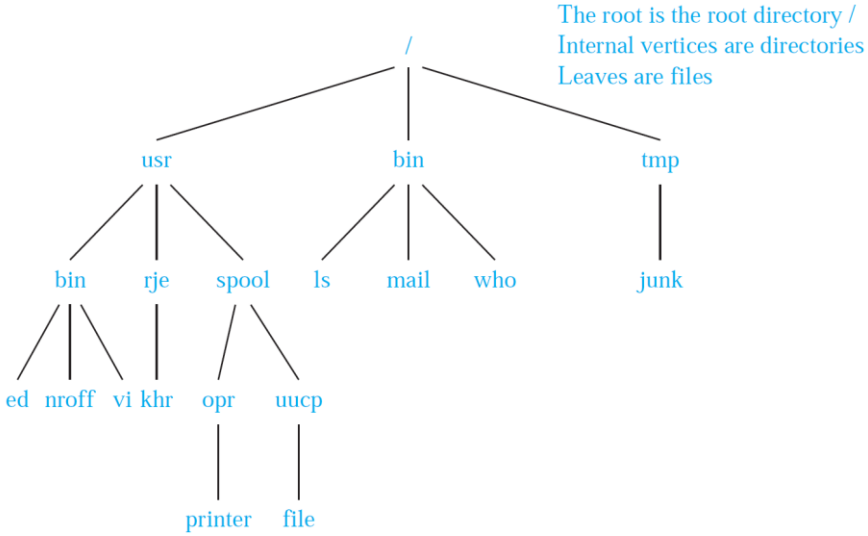


FIGURE 11 A Computer File System.

1 Introduction to Trees – Properties of Trees

THEOREM 1

An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

THEOREM 2

A tree with n vertices has $n-1$ edges.

THEOREM 3

A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

THEOREM 4

A full m-ary tree with

- n vertices has $i = (n-1)/m$ internal vertices and $l = [(m-1)n + 1]/m$ leaves.
- i internal vertices has $n = mi + 1$ vertices and $l = (m-1)i + 1$ leaves.
- l leaves has $n = (ml-1)/(m-1)$ vertices and $i = (l-1)/(m-1)$ internal vertices.

Balanced *m*-ary Trees

The subtrees at each vertex contains paths of approximately the same length.

The **level** of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.

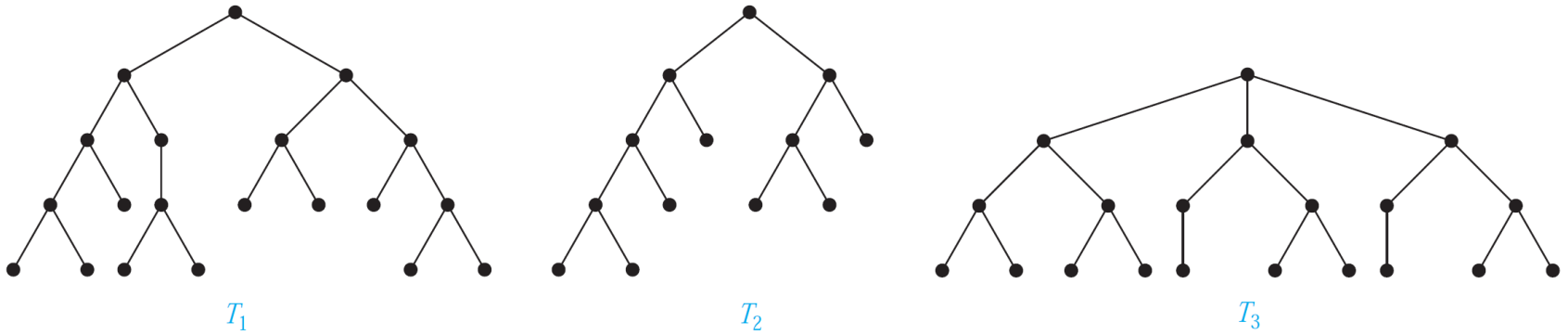


FIGURE 14 **Some Rooted Trees.**

2 Tree Traversal

Introduction

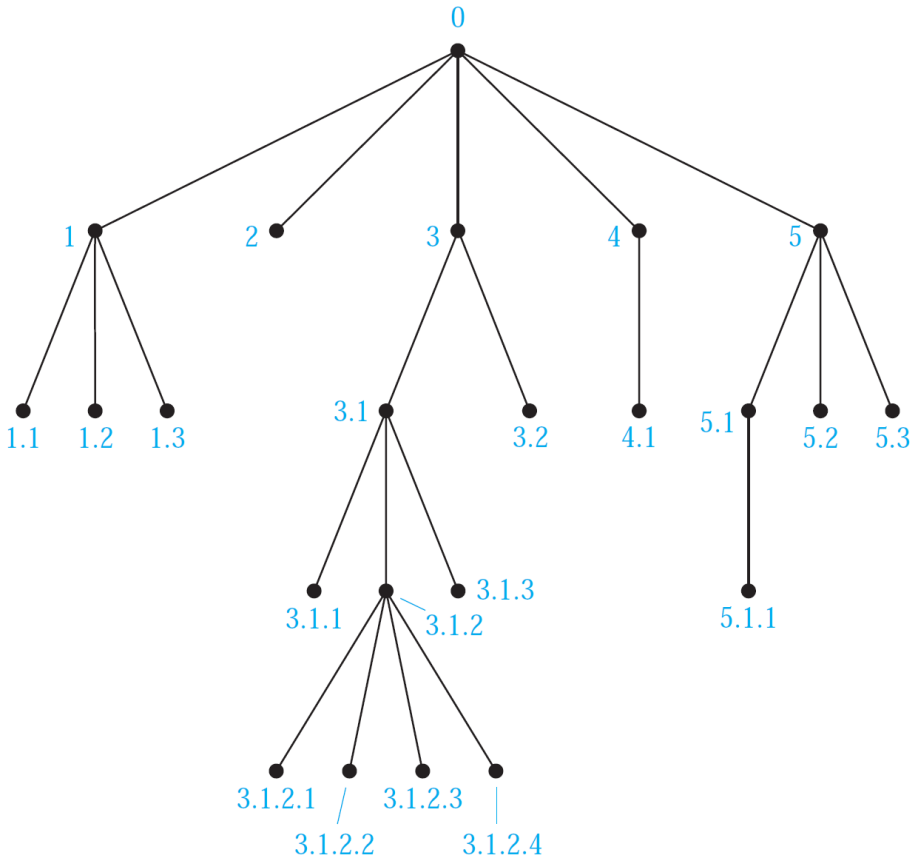


FIGURE 1 **The Universal Address System of an Ordered Rooted Tree.**

Ordered rooted trees are often used to store information.

We need procedures for visiting each vertex of an ordered rooted tree to access data.

2 Tree Traversal – Traversal Algorithms

DEFINITION

Let T be an ordered rooted tree with root r. If T consists only of r, then r is the *preorder traversal* of T. Otherwise, supose that $T1$, $T2$, …, Tn are the subtrees at r from left to right in T.

- The *preorder traversal* begins by visiting r. It continues by traversing $T1$ in preorder, then $T2$ in preorder, and so on, until Tn is traversed in preorder.
- The *inorder traversal* begins by traversing $T1$ in inorder, then visiting r. It continues by traversing $T2$ in order, then $T3$ inorder, …, and finally Tn in inorder.
- The *postorder traversal* begins traversing $T1$ in postorder, then $T2$ in postorder, …, then Tn in postorder, and ends by visiting r.

2 Tree Traversal – Traversal Algorithms

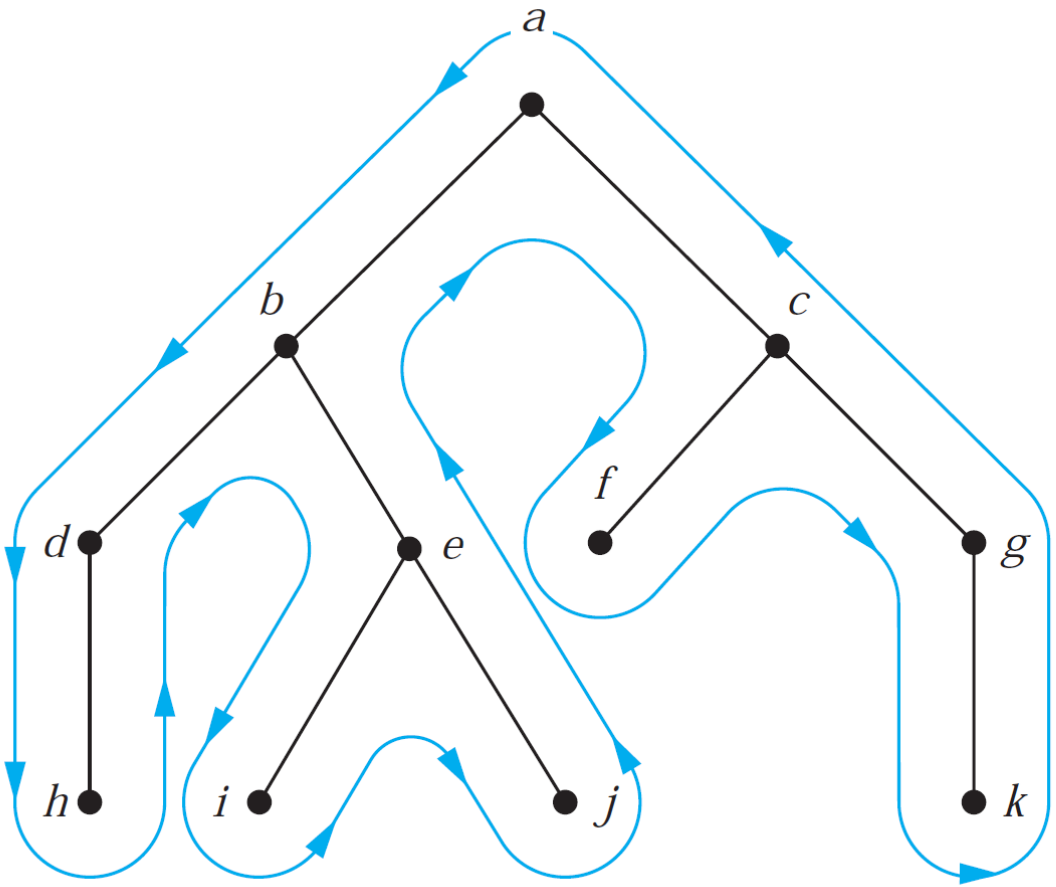


FIGURE 9 **A Shortcut for Traversing an Ordered Rooted Tree in Preorder, Inorder, and Postorder.**

2 Tree Traversal – Infix, Prefix, and Postfix Notation

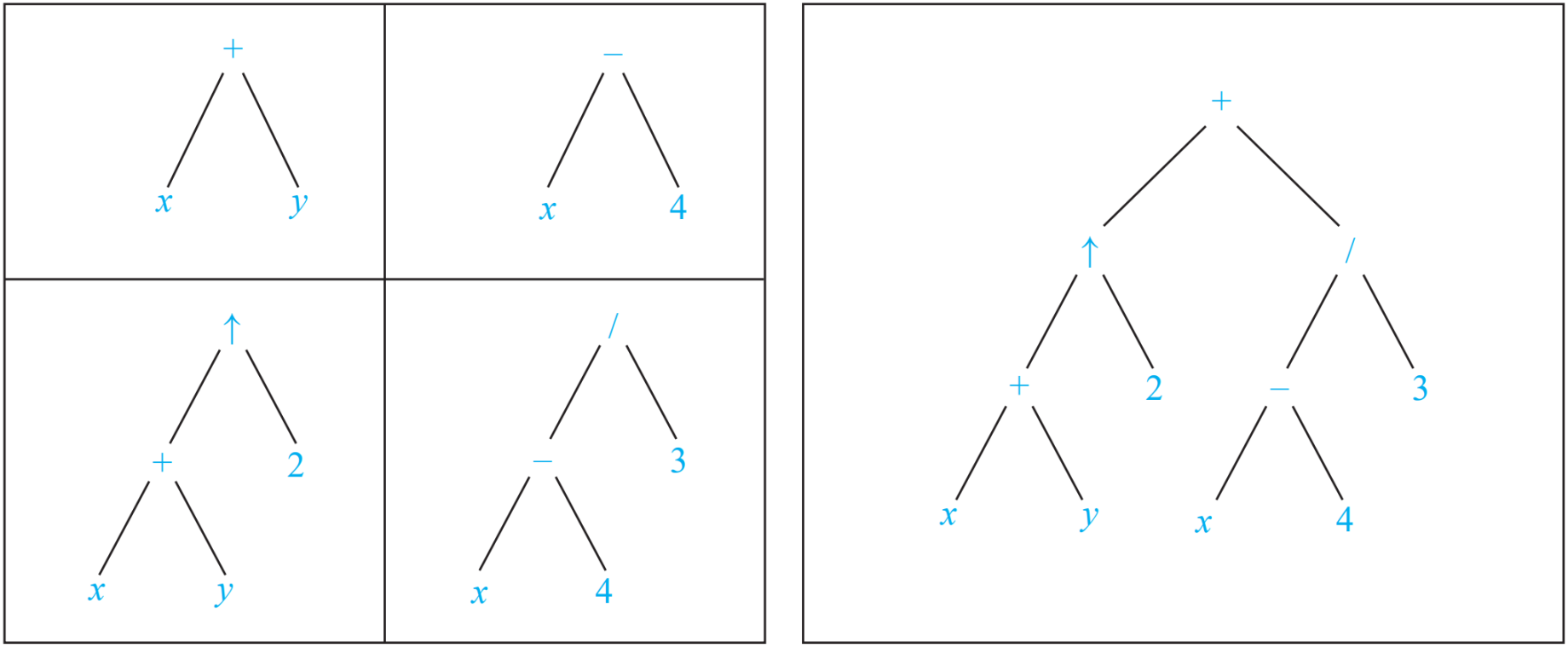


FIGURE 10 **A Binary Tree Representing $((x + y) \uparrow 2) + ((x - 4)/3)$.**

3 Spanning Trees – Introduction

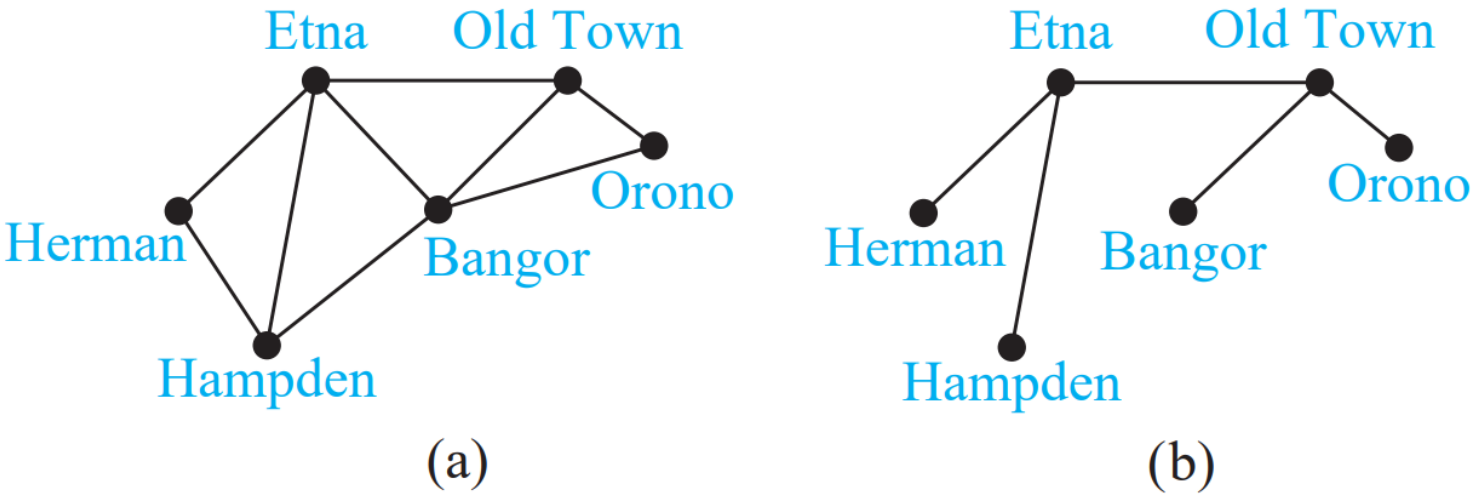


FIGURE 1 **(a) A Road System and (b) a Set of Roads to Plow.**

3 Spanning Trees – Introduction

DEFINITION

Let G be a simple graph. A *spanning tree* of G is a subgraph of G that is a tree containing every vertex of G.

EXAMPLE

Find a spanning tree of the simple graph G shown in Figure 2.

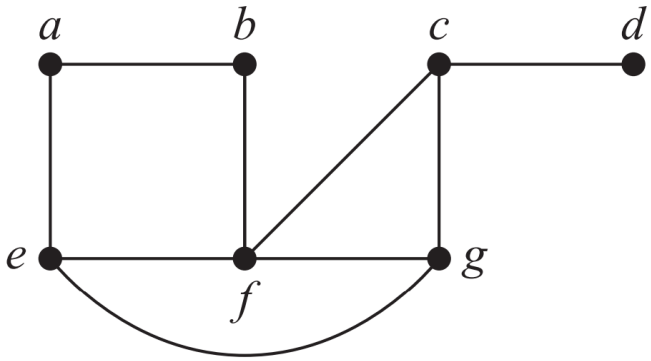


FIGURE 2 **The Simple Graph G.**

3 Spanning Trees – IP Multicasting

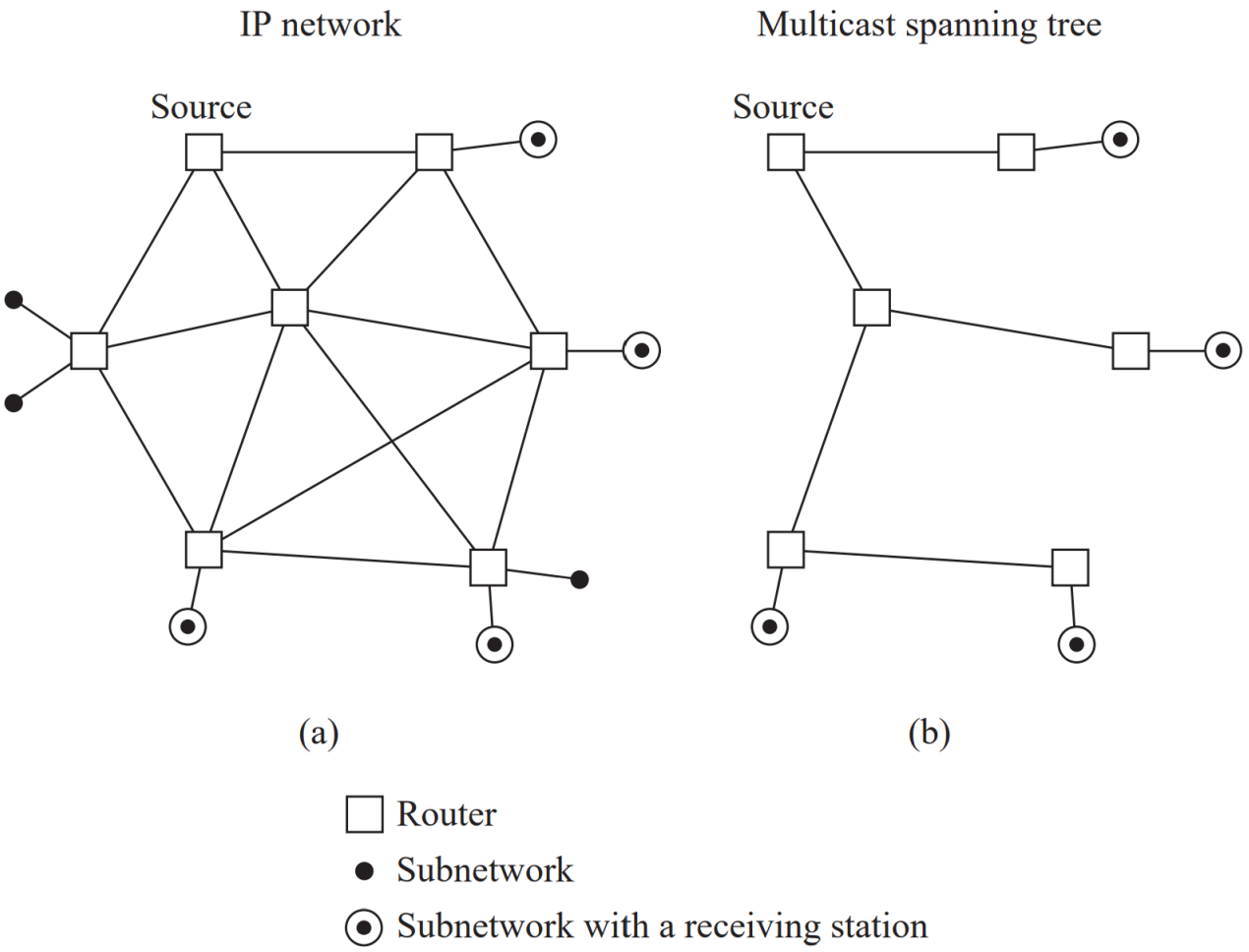


FIGURE 5 **A Multicast Spanning Tree.**

3 Spanning Trees – Depth-First Search

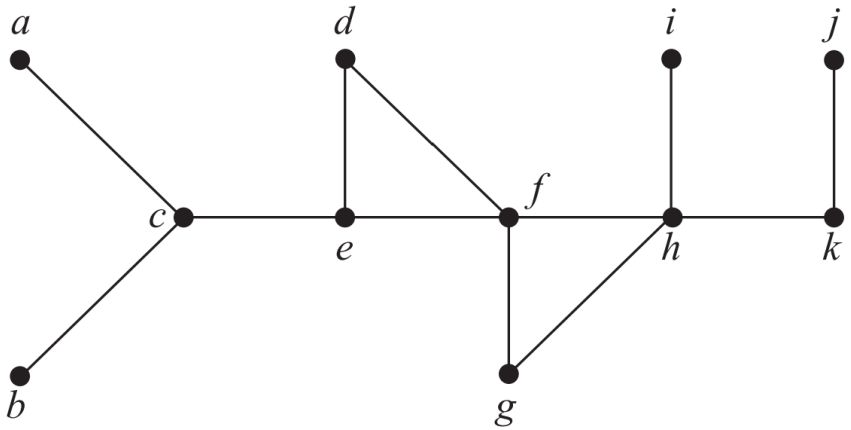


FIGURE 6 The Graph G.

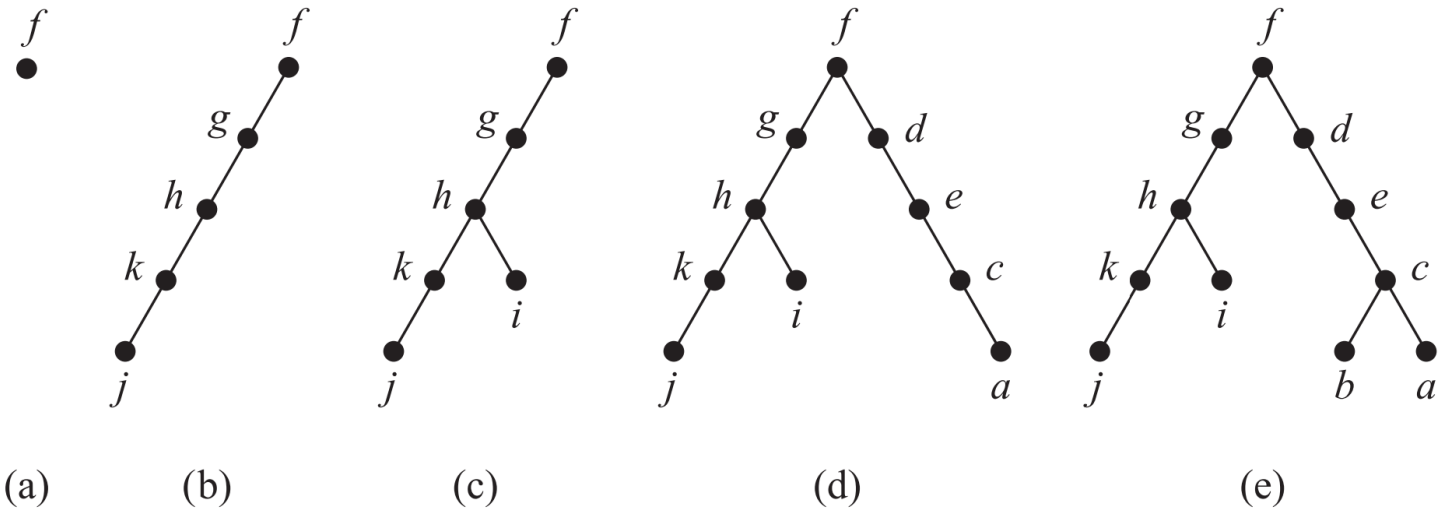


FIGURE 7 Depth-First Search of G.

3 Spanning Trees – Breath-First Search

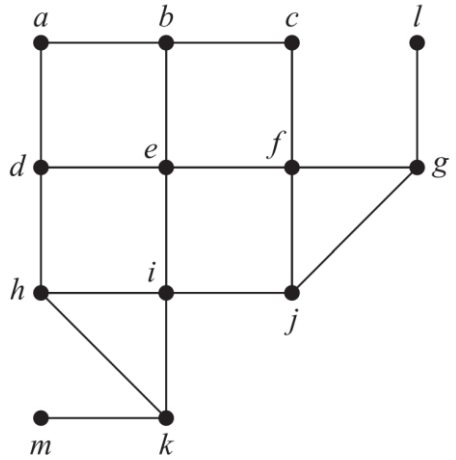


FIGURE 9 **A Graph *G*.**

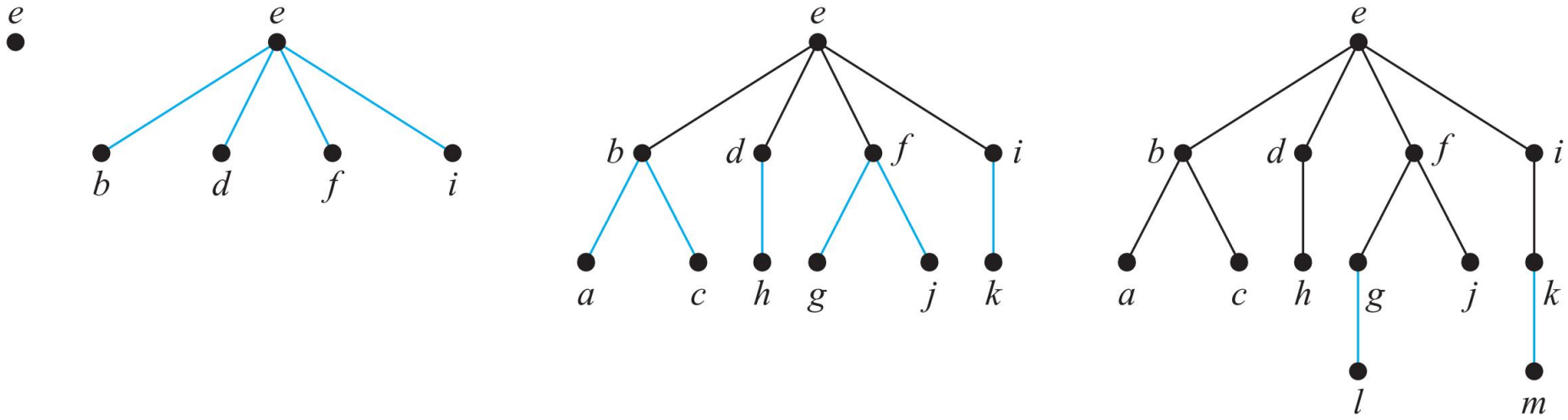
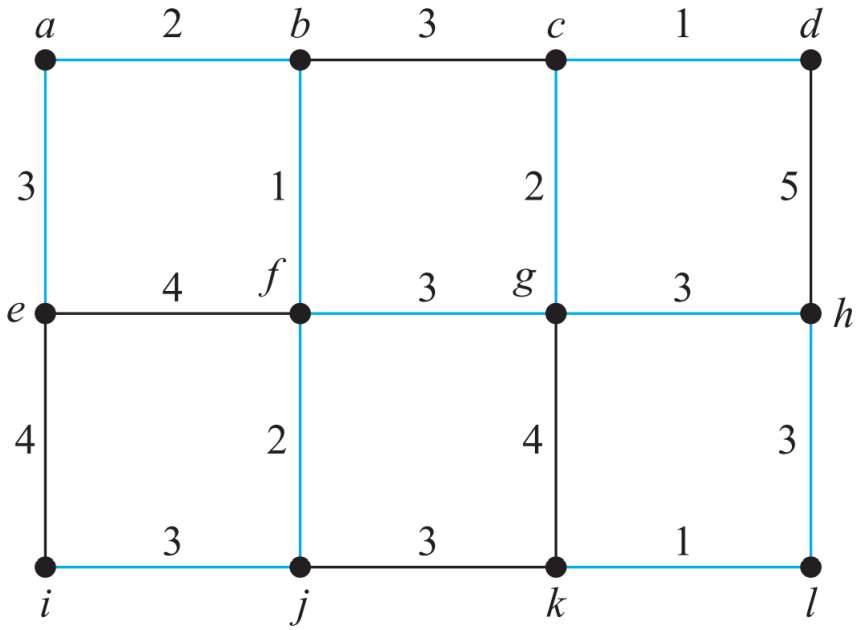


FIGURE 10 **Breadth-First Search of *G*.**

4 Minimum Spanning Trees – Prim's Algorithm



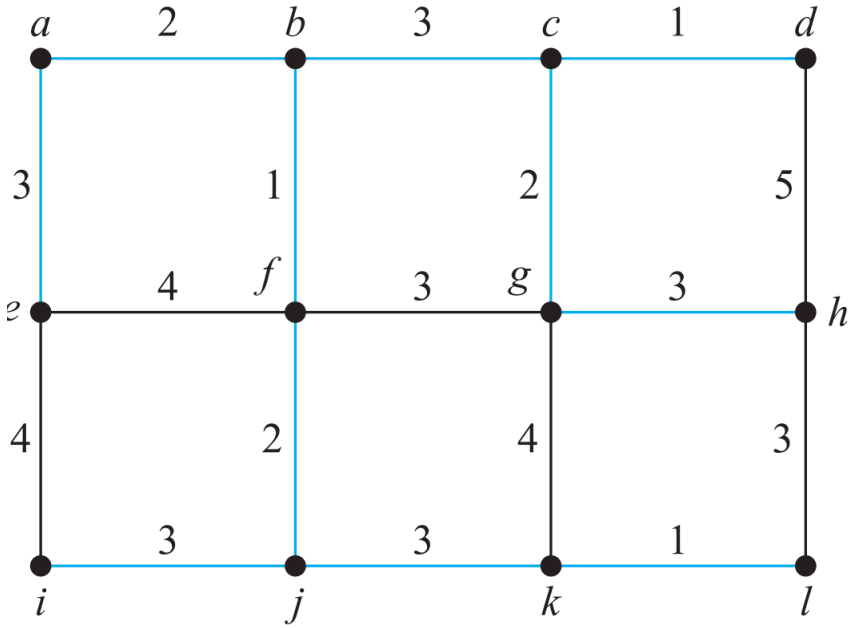
(a)

Choice	Edge	Weight
1	$\{b, f\}$	1
2	$\{a, b\}$	2
3	$\{f, j\}$	2
4	$\{a, e\}$	3
5	$\{i, j\}$	3
6	$\{f, g\}$	3
7	$\{c, g\}$	2
8	$\{c, d\}$	1
9	$\{g, h\}$	3
10	$\{h, l\}$	3
11	$\{k, l\}$	1
	Total:	24

(b)

FIGURE 4 **A Minimum Spanning Tree Produced Using Prim's Algorithm.**

4 Minimum Spanning Trees – Kruskal's Algorithm



(a)

Choice	Edge	Weight
1	$\{c, d\}$	1
2	$\{k, l\}$	1
3	$\{b, f\}$	1
4	$\{c, g\}$	2
5	$\{a, b\}$	2
6	$\{f, j\}$	2
7	$\{b, c\}$	3
8	$\{j, k\}$	3
9	$\{g, h\}$	3
10	$\{i, j\}$	3
11	$\{a, e\}$	3
	Total:	24

(b)

FIGURE 5 **A Minimum Spanning Tree Produced by Kruskal's Algorithm.**