# Module Five

## Learning Objectives

By the end of this module, you will meet these learning objectives:

- ✓ Analyze the application of algorithms in relation to the field of computer science and to daily life
- ✓ Apply the appropriate strategy to break a given problem into components
- ✓ Determine how to represent an algorithm with pseudocode
- ✓ Describe the functional differences between data structures
- ✓ Describe how the use of an algorithm affects specific data structures
- ✓ Analyze the running time of search algorithms

## Module Overview

Welcome to Module Five! In this module, we will look at trees. Trees are used to solve specific problems, usually within the scope of a larger problem. They behave differently than other data structures in how they are organized. Trees are useful data structures that allow inserting as many elements as we want, similar to hash tables, linked lists, and vectors. Trees dynamically allocate a node for every insertion, but they insert in constant time (as soon as they find where to insert).

A tree in computer science is implemented upside down from how we think of a tree in nature. The top of the tree is the root, and there is one node there. Branches connect the root to other nodes, and the nodes that do not have any other nodes below them are called leaves. The leaves are at the bottom of the tree. This is the opposite of natural trees, but the structure is the same.

```
         root
         /  \
left_child  right_child
```

Trees that have at most two children for any node are called binary trees. Binary trees can have 0, 1, or 2 children for each node. Trees that can have up to three children per node are called ternary trees. Trees that can have up to k children per node are called k-ary trees.

Binary trees, and all the derivations of binary trees, are probably the most widely used. One variation of a binary tree is called a binary search tree (BST). Binary search trees maintain ordering of the data, where any data that is smaller than a node will be placed in the left sub-tree, and any data that is larger than a node will be placed in the right sub-tree. If the data is inserted in a strategic manner (with the middle elements of the remaining lists inserted sequentially), you will maintain a balanced binary search tree, which is the optimal tree for searching. You would only need to check log2n nodes to find a value in a balanced binary search tree, as opposed to potentially checking n nodes in an unbalanced binary search tree.

## Module at a Glance

This is the recommended plan for completing the reading assignments and activities within the module. Additional information can be found in the module Resources section and on the module table of contents page.

1. Review the Module Five resources.
2. Complete the Module Five activities in zyBooks.
3. Complete the Module Five assignment.
4. Complete the milestone.