Preface to the Instructor

This book is intended for an introductory course on data structures and asymptotic algorithm analysis. It covers the vector, set, heap, stack, queue, deque, binary tree, and binary search tree abstract data types and their common applications, along with the classical searching and internal sorting algorithms: the linear and binary searches, and the bubble, selection, heap, Shell, insertion, merge, and quick sorts. The containers are related to one another within a class hierarchy, the design of which proceeds from concrete classes to abstract classes in a bottom-up fashion that mirrors human concept formation. Sound object-oriented design and programming practices are emphasized throughout. The design language is UML; the programming language is Java.

This is a slim volume relative to its competitors, and that deserves an explanation. The typical introductory data structures text contains too much information. An overabundance of details makes it all too easy for the learner to lose sight of the book's primary thread of development, and it leaves too little discovery to the learner. This book develops its material in a natural and fluid way. Thus it is recommended that the chapters be covered in sequence. The abstract data types and algorithms are covered fairly comprehensively, but the Java programming language is not. The nuances of the chosen programming language are, after all, subsidiary to the computer science material. The relative dearth of information about Java encourages the learner to consult Java reference material; the importance to the information technology professional of using reference material effectively can hardly be overestimated.

One of this book's primary themes is that a collection has implications for its implementation. (The stack, for example, should be implemented using a singly linked list rather than an array because the stack, unlike the heap, cannot exploit the array's random access.) Thus only one implementation of each collection is presented, as the ideal implementation, obviating the common Java design pattern for data structures of using both interfaces and abstract classes.