Preface

This book is for college-level students who are attending accompanying lectures on data structures and algorithms. Except for the introduction, exercises, and notes for each chapter, page breaks have been put in manually and the format corresponds to a lecture style with ideas presented in "digestible" page-size quantities.

It is assumed that the reader's background includes some basic mathematics and computer programming. Algorithms are presented with "pseudo-code", not programs from a specific language. It works well to complement a course based on this book with a programming "lab" that uses a manual for a specific programming language and assigns projects relating to the material being taught.

Chapters 1 through 4 go at a slower pace than the remainder of the book, and include sample exercises with solutions; some of the sections in these chapters are starred to indicate that it may be appropriate to skip them on a first reading. A first course on data structures and algorithms for undergraduates may be based on Chapters 1 through 4 (skipping most of the starred sections), along with portions of Chapter 5 (algorithm design), the first half of Chapter 6 (hashing), the first half of Chapter 12 (graphs), and possibly an overview of Chapter 13 (parallel models of computation). Although Chapters 1 through 4 cover more elementary material, and at a slower pace, the concise style of this book makes it important that the instructor provide motivation, discuss sample exercises, and assign homework to complement lectures. For upper class undergraduates or first-year graduate students who have already had a course on elementary data structures, Chapters 1 through 4 can be quickly reviewed and the course can essentially start with the algorithm design techniques of Chapter 5.

There is no chapter on sorting. Instead, sorting is used in many examples, which include bubble sort, merge sort, tree sort, heap sort, quick sort, and several parallel sorting algorithms. Lower bounds on sorting by comparisons are included in the chapter on heaps, in the context of lower bounds for comparison based structures. There is no chapter on NP-completeness (although a section of the chapter on graph algorithms does overview the notion). Formal treatment of complexity issues like this is left to a course on the theory of computation.

Although traditional serial algorithms are taught, the last chapter presents the PRAM model for parallel computation, discusses PRAM simulation, considers how the EREW PRAM, HC/CCC/Butterfly, and mesh models can be used for PRAM simulation, and concludes with a discussion of hardware area-time tradeoffs as a way of comparing the relative merits of simulation models. Although it is not clear what parallel computing models will prevail in practice in the future, the basic concepts presented in the chapter can provide a foundation for further study on specific practical architectures. Also, from the point of view of studying algorithms, it is instructive to see how basic principles taught in the previous chapters can be adapted to other models.