

DATA STRUCTURES

FROM ARRAYS TO PRIORITY QUEUES

WAYNE AMSBURY





Data Structures

From Arrays to Priority Queues

Wayne Amsbury

Northwest Missouri State University

Wadsworth Publishing Company

Belmont, California

A Division of Wadsworth, Inc.

Computer Science Editor: *Frank Ruggirello*
Production: *Mary Forkner, Publication Alternatives*
Text Designer: *Michael Rogondino*
Copy Editor: *Joan Pendleton*
Technical Illustrator: *Carol Johnston*
Cover Design: *Stephen Osborn*
Signing Representative: *Myron Flemming*

© 1985 by Wadsworth, Inc. All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher, Wadsworth Publishing Company, Belmont, California 94002, a division of Wadsworth, Inc.

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10—89 88 87 86 85

ISBN 0-534-04590-1

Library of Congress Cataloging in Publication Data

Amsbury, Wayne, 1935—
Data structures.

Bibliography: p.
Includes index.

1. Data structures (Computer science) I. Title.
QA76.9.D35A47 1985 001.64'4 85-3159
ISBN 0-534-04590-1

Data Structures

From Arrays to Priority Queues

Wadsworth Series in Computer Information Systems

Senn, Information Systems in Management, Second Edition

Clarke and Prins, Contemporary Systems Analysis and Design

Vasta, Understanding Data Base Management Systems

Ageloff and Mojena, Applied Structured BASIC

Rob, Big Blue BASIC: Programming the IBM PC and Compatibles

Athappilly, Programming and Problem Solving in VAX-11 BASIC

Rob, Introduction to Microcomputer Programming

Amsbury, Data Structures: From Arrays to Priority Queues

Brown, From Pascal to C: An Introduction to the C Programming Language



Preface

Many application programs and discussions of topics in computer science are awkward without a relatively extensive acquaintance with data structures. Hence, the data structures course is moving rapidly downward in the curriculum. In the last few years it has changed from being a graduate-level course in analytical techniques to a foundation course for undergraduates and nonmajors. If it isn't already, data structures soon will be taught at four-year colleges as a prerequisite for most upper-level computer science courses and as a terminal computer science course in two-year programs and for many nonmajors. Consequently, the course is taught on a wide range of levels. The broadening of the audience requires some adjustment in the way a data structures course is taught.

Although its wide range of content and academic rigor make it suitable for use in higher-level courses, *Data Structures: From Arrays to Priority Queues* is intended to serve a broader audience. The book can function as an introduction to data structures quite early in the development of programming mastery. It is aimed at the student who has had one or more semesters of programming in a structured language and no specific college mathematics. At lower levels, a data structures course needs to develop algorithmic sophistication rather than exercise mathematical skills. Thus, this book keeps the comparison of data structures and algorithms in constant view, but it does not require a great deal of mathematical sophistication. *Data Structures* brings the reader to an awareness of the timing of algorithms, particularly at the level of the order of growth, but extensive analysis of algorithms applied to data structures is left to later courses. The intent is to foster an appreciation of the effectiveness of data structures and their associated algorithms, but in an introduction it is more important to construct a workable set of tools than to hone a smaller set. Data structures *themselves* are the primary focus, not their detailed analysis.

The topic of data structures plays much the same essential role in computer science that calculus plays in mathematics. The *apparent agenda* in a data struc-

tures course is the comparative study of alternate ways to solve and implement program solutions to problems. The *hidden agenda* is the development of an understanding of the use of abstraction in problem solving. This book focuses on the hidden agenda as a foundation for study of the apparent agenda. Skill in abstraction is developed gradually, as a natural part of problem solving with programs. Three features that explicitly address the hidden agenda are:

1. Use of Pascal-like pseudocode
2. Comparison of algorithmic and implementation variations
3. Generalized algorithms for data structure traverses

A form of pseudocode is used to present algorithms for several reasons. One reason is simply that no one language is ideal for all purposes, is used universally, or is dominant forever. A number of languages, including Algol, Pascal, ADA, and Modula-2, have common features extracted here. Another reason for presenting algorithms in pseudocode for the reader to translate into a working language is that the task requires some effort, but not too much. Translation requires a choice of parameter-passing classes for procedure variables and attention to some other details. Translating pseudocode does not, however, require the time and effort needed to *derive* an algorithm. An extensive amount of conceptual material can be presented succinctly in pseudocode. Use of pseudocode thus provides an appropriate balance between covering the maximum amount of ground and providing in-depth understanding of the details of the geography that is traversed. Use of pseudocode, plus examples in the text, also helps readers to develop skill in abstraction and understanding of shell-structuring and modularity.

The management of data structures involves pitfalls that lead to subtle bugs in programs, and such potential problems are best exposed by being viewed from several perspectives. Therefore, *Data Structures* emphasizes the variety of solutions to problems. For example, general traverse algorithms are developed for lists, binary trees, and priority queues. Such general algorithms provide a framework within which variations can be created and problems can be solved. The algorithms can be tailored to specific applications, and the exercise of doing so has merit in itself. The general algorithms are better guides to the variations needed for applications than is one specific variation that often fails to fit. General algorithms also promote the development of skill in abstraction in a relatively gentle way.

A separate collection of expansion and enrichment sections is included in Part II. The sections introduce or explore in depth topics that are related to the mainstream of the text. Some of these sections contain substantial working Pascal programs, initially developed in pseudocode, which provide examples of program development. The instructor thus has options for major examples, enrichment, and independent reading assignments, yet because the additional sections are separate, they do not distract from the main content.

To help the instructor tailor the course to students with varied backgrounds, most topics in the book can be covered at a variety of levels:

1. Pictorial discussion and general comprehension
2. Tracing of the crucial process(es) acting on a specific example
3. Pseudocode detailing of an algorithm with discussion of problems and variations
4. Program assignments based on class discussion
5. Program assignments based on reading
6. Experimentation with programs and analysis of the results

Potential assignments are separated into exercises, problems, programs, and projects, and the amount of effort required increases in that order. Possible assignments are mentioned in the text where they are appropriate, but they are grouped together at the end of the chapter so they can be found easily at any time. Summaries offer the reader an overview of material in each chapter.

The instructor's manual includes answers to the exercises and problems and a Pascal solution to a program assignment from each chapter.

The flexibility designed into this text has allowed it to be used at NWMSU for classes with students of mixed backgrounds, many of whom are second-semester freshmen. The central thrust of the course is coverage of the sections in Part One of this text that are not marked *optional*. In that course, Chapter 7 is considered remedial but is available for self-study, the timing-function portion of Chapter 1 is the only part of that chapter requiring class time, and Chapter 10 is seldom reached. Some optional sections in Part I and various sections of Part II are selected by instructors for supplementary readings, assignments, and sometimes class discussion.

Thorough coverage of the entire text requires a second semester or a very fast pace for even well-prepared students. The intent of the rich offering of Part II and the optional sections in Part I is to support flexibility through the selection of sections and levels of coverage. (Besides, students *will* explore accessible material on their own.)

The reviewers listed below made many suggestions that were incorporated into this book and improved it immeasurably. They are not, however, responsible for the remaining errors, whether inadvertant or stubborn: Cathy Dickerson, Black Hawk College; William H. Ford, University of the Pacific; Ken Friedenbach, University of Santa Clara; Judith L. Gersting, Indiana University-Purdue University; Henry Gordon, Kutztown State College; Nancy Griffeth, Georgia Institute of Technology; Greg Jones, Providence, Utah; Leonard Larsen, University of Wisconsin, Eau Claire; Louise Moyer, California State University, Hayward; Ron Peterson, Weber State College; Douglas Re, San Francisco Community College; Dean Sanders, Illinois State University.

Two groups of people aided and abetted this work in ways that must remain untold in detail but are appreciated in depth. One group is a team assembled by Wadsworth that includes: Myron Flemming, Frank Ruggirello, Serina Beauparlant, Mary Forkner, and Joan Pendleton. The other group consists of colleagues at

NWMSU: Merry McDonald, Gary McDonald, Robert Franks, Hong-Shi Yuan, Phil Heeler, and Margaret Adams.

My brother, David, put considerable effort into improving my technical writing in the early stages of this book, and I can only hope that the effect carried through the project.

Finally, this book could not have been produced if my wife, Carlene, had not taken over virtually all of the management of a busy household while working full time.

Wayne Amsbury



Contents

Preface	xiii
 PART I The Mainstream	 1
 CHAPTER 1 Algorithms and Timing Analysis	 3
1.1 Sophistication in Programming	4
1.2 The Pseudocode Language SUE	6
1.3 The Timing of Algorithms	13
■ 1.4 Comparison of Three Sorts (Optional)	20
■ 1.5 Random Processes (Optional)	29
SUMMARY	29
EXERCISES and PROBLEMS	30
 CHAPTER 2 The Array Structure	 32
■ 2.1 Static Cell Groups (Optional)	32
2.2 The Array as a Structure	36
2.3 Array Addresses	39
2.4 A Sparse and Static Table	43
■ 2.5 Hash Tables (Optional)	48
■ 2.6 Unfolding Indices (Optional)	49
SUMMARY	52
FURTHER READING	53
EXERCISES, PROBLEMS, PROGRAMS, and PROJECTS	53

CHAPTER 3	List Processing	57
3.1	The List as a Structure	58
3.2	Linked Lists	60
3.3	Common List Operations	62
■ 3.4	The Upper Crust at Ballyhoo U (Optional)	76
3.5	Array Support for Linked Lists	79
3.6	Ordered Lists	83
3.7	Circular Lists	89
■ 3.8	Polynomial Arithmetic (Optional)	90
■ 3.9	Arithmetic of Unbounded Precision (Optional)	92
	SUMMARY	93
	EXERCISES, PROBLEMS, PROGRAMS, and PROJECTS	94
CHAPTER 4	Stacks	97
4.1	The Stack Structure	98
4.2	Stacks in Language	106
■ 4.3	Stacks in the Language FORTH (Optional)	115
■ 4.4	The p-Machine (Optional)	116
■ 4.5	The Use of Stacks in Simulations (Optional)	116
■ 4.6	Backtracking with Stacks (Optional)	117
	SUMMARY	118
	EXERCISES, PROBLEMS, PROGRAMS, and PROJECTS	118
CHAPTER 5	Queues	120
5.1	Queueing Behavior	122
5.2	Queue Implementation and Management	123
5.3	Circular-Track Queues	129
5.4	Generalized Queues	131
■ 5.5	Queues in Hardware (Optional)	132
■ 5.6	On the Simulation of a Doctor's Waiting Room (Optional)	133
	SUMMARY	138
	EXERCISES, PROBLEMS, PROGRAMS, and PROJECTS	139

CHAPTER 6	General Lists: Multiple Access Paths	142
6.1	Doubly Linked Lists	143
6.2	n -braids	146
6.3	Matrices and Graph Adjacency	149
■ 6.4	Sparse Matrix Representation (Optional)	154
■ 6.5	The Orthogonal List (Optional)	155
■ 6.6	A Dynamic Storage Allocation Model (Optional)	159
	SUMMARY	166
	EXERCISES, PROBLEMS, PROGRAMS, and PROJECTS	167
CHAPTER 7	Recursion	169
7.1	A Recursion Sampler	169
7.2	The Fibonacci Connection	174
7.3	Search in an Ordered List	181
■ 7.4	The Towers of Hanoi (Optional)	185
■ 7.5	QuickSort (Optional)	185
■ 7.6	The Eight-Queens Problem (Optional)	186
■ 7.7	A General-List Visitation Algorithm (Optional)	190
■ 7.8	LISP: Lists and Recursion (Optional)	192
	SUMMARY	196
	EXERCISES, PROBLEMS, PROGRAMS, and PROJECTS	197
CHAPTER 8	Binary Trees	199
8.1	General Features of Binary Trees	202
8.2	Implementation of Binary Trees	207
8.3	Recursive Binary Tree Traverses	209
8.4	Iterative Tree Walks	215
8.5	Ordered Binary Trees	220
8.6	Heaps	226
8.7	Balanced Binary Trees	235
■ 8.8	Threaded Binary Trees (Optional)	241
	SUMMARY	244
	EXERCISES, PROBLEMS, and PROGRAMS	247

CHAPTER 9	Multiple Access Paths to Data	248
9.1	Access by Indirection	249
9.2	External Files	253
9.3	Doubly Linked Two-Way Trees	254
9.4	m-Way Search Trees	258
9.5	B-trees	262
■ 9.6	2–3–4 Trees (Optional)	276
■ 9.7	Sets and Their Trees (Optional)	281
■ 9.8	Trees as Decision Tools (Optional)	286
■ 9.9	Game Trees (Optional)	287
	SUMMARY	288
	EXERCISES, PROBLEMS, and PROGRAMS	291
CHAPTER 10	Graph Algorithms	292
10.1	Graph Representation	295
10.2	The Priority-Queue Traverse of a Graph	299
10.3	Applications of Priority-Queue Traverses	306
10.4	Traverses of Weighted Graphs	318
	SUMMARY	323
	FURTHER READING	323
	EXERCISES, PROBLEMS, and PROGRAMS	324
PART II	Expansions and Applications	327
SECTION A	Random Processes	329
A.1	Mean and Standard Deviation	331
A.2	Frequency Distributions	332
A.3	Normal Distributions	333
A.4	Exponential Distributions	335
	EXERCISES, PROBLEM, PROGRAM, and PROJECT	336
SECTION B	Hash Tables	337
B.1	A Choice of Hash Functions	339
B.2	Linear Probing	342

B.3	Secondary Clustering and Double Hashing	344
B.4	Deletion and Rehashing	346
	PROBLEMS and PROGRAMS	346
SECTION C	Circular List Operations	347
	EXERCISES and PROBLEMS	352
SECTION D	Integer Arithmetic of Unbounded Precision	353
D.1	The Entry Sequence	353
D.2	Utility Routines	354
D.3	Arithmetic Operations	358
SECTION E	The Stack Machine	377
SECTION F	Stacks in the Language FORTH	381
F.1	FORTH Control Structures	386
SECTION G	The p-Machine	390
SECTION H	Following a Maze	398
SECTION I	The Queue Machine	403
SECTION J	Queues in Hardware	408
	PROGRAMS	412
SECTION K	A Simulation of an Airport Customs Station	413
K.1	The Customs Station	414
K.2	The Event-Cycle Level of <i>Airport</i>	416
K.3	The Gathering of Data	418
	PROJECTS	419
K.4	The Program <i>Airport</i>	420
SECTION L	Chaining in Scatter Storage	431
	PROBLEM and PROGRAM	434

SECTION M	Walking a General List	435
	PROJECTS	442
SECTION N	The Main Diagonal Circulation Sparse Matrix	443
SECTION O	A QuickSort Program	449
	O.1 The QuickSort Demonstration Program	452
SECTION P	A Balanced Static BST	456
	PROBLEM and PROGRAM	459
SECTION Q	Huffman Code Trees	460
	Q.1 Building a Huffman Tree	463
	EXERCISES, PROBLEM, PROGRAM, and PROJECT	466
SECTION R	AVL Trees	467
	R.1 The Insertion Algorithm for AVL Trees	472
	EXERCISES, PROBLEM, and PROJECT	475
SECTION S	File Merging and MergeSort	476
	S.1 A Linked-List Merge	476
	S.2 Binary Merge	479
	S.3 Merge Sorting	480
	PROGRAMS	482
SECTION T	Priority Queue Extension of Merge Runs	483
	PROGRAM	485
SECTION U	Red-Black Trees	486
	PROBLEM and PROGRAM	498
SECTION V	The Eight Coins Problem	499
	PROGRAM	502

SECTION W	Node Evaluation and Pruning of Game Trees	503
	W-1 Alpha-Beta Pruning of a Game Tree	506
	EXERCISE, PROBLEMS, PROGRAM, and PROJECT	509
REFERENCES		511
INDEX		513



PART ONE

The Mainstream