Edward M. Reingold
Wilfred J. Hansen

Data Structures

Little, Brown Computer Systems Series

Data Structures

Edward M. Reingold
University of Illinois at Urbana-Champaign

Wilfred J. Hansen
Carnegie-Mellon University

Little, Brown and Company
Boston Toronto

Library of Congress Cataloging in Publication Data

Reingold, Edward M., 1945-
Data structures.

Includes bibliographies.
1. Data structures (Computer science)
I. Hansen, Wilfred J. II. Title.
QA76.9.D35R44 1983 001.64'2 82-12697
ISBN 0-316-73951-0

Library of Congress Catalog Card No. 82-12697

ISBN 0-316-73951-0

9 8 7 6 5 4 3 2

MV

Published simultaneously in Canada by Little, Brown & Company (Canada) Limited

Printed in the United States of America

The cover is a reproduction of the first panel of the triptych *Three Trees* by Karel Appel, reproduced by permission of Karel Appel.

Acknowledgment

The authors gratefully acknowledge permission to use material from Reingold/Nievergelt/Deo, *Combinatorial Algorithms: Theory and Practice*, © 1977, pp. 280–315. Reprinted by permission of Prentice-Hall, Inc., Englewood Cliffs, N.J.

Little, Brown Computer Systems Series

Gerald M. Weinberg, *Editor*

Barnett, Michael P., and Graham K. Barnett
Personal Graphics for Profit and Pleasure on the Apple II® Plus Computer

Basso, David T., and Ronald D. Schwartz
Programming with FORTRAN/WATFOR/WATFIV

Chattergy, Rahul, and Udo W. Pooch
Top-down, Modular Programming in FORTRAN with WATFIV

Coats, R. B., and A. Parkin
Computer Models in the Social Sciences

Conway, Richard, and David Gries
An Introduction to Programming: A Structured Approach Using PL/I and PL/C, Third Edition

Conway, Richard, and David Gries
Primer on Structured Programming: Using PL/I, PL/C, and PL/CT

Conway, Richard, David Gries, and E. Carl Zimmerman
A Primer on Pascal, Second Edition

Cripps, Martin
An Introduction to Computer Hardware

Easley, Grady M.
Primer for Small Systems Management

Finkenaur, Robert G.
COBOL for Students: A Programming Primer

Freedman, Daniel P., and Gerald M. Weinberg
Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products, Third Edition

Graybeal, Wayne, and Udo W. Pooch
Simulation: Principles and Methods

Greenfield, S. E.
The Architecture of Microcomputers

Greenwood, Frank
Profitable Small Business Computing

Healey, Martin, and David Hebditch
The Microcomputer in On-Line Systems: Small Computers in Terminal-Based Systems and Distributed Processing Networks

Lemone, Karen A., and Martin E. Kaliski
Assembly Language Programming for the VAX-11

Lias, Edward J.
Future Mind: The Microcomputer—New Medium, New Mental Environment

Lines, M. Vardell, and Boeing Computer Services Company
Minicomputer Systems

Mashaw, B. J.
Programming Byte by Byte: Structured FORTRAN 77

Mills, Harlan D.
Software Productivity

Monro, Donald M.
Basic BASIC: An Introduction to Programming

Morrill, Harriet
Mini and Micro BASIC: Introducing Applesoft®, Microsoft®, and BASIC Plus

Mosteller, William S.
Systems Programmer's Problem Solver

Nahigian, J. Victor, and William S. Hodges
Computer Games for Businesses, Schools, and Homes

Nahigian, J. Victor, and William S. Hodges
Computer Games for Business, School, and Home for TRS-80® Level II BASIC

Orwig, Gary W., and William S. Hodges
The Computer Tutor: Learning Activities for Homes and Schools

Parikh, Girish
Techniques of Program and System Maintenance

Parkin, Andrew
Data Processing Management

Parkin, Andrew
Systems Analysis

Pizer, Stephen M., with Victor L. Wallace
To Compute Numerically: Concepts and Strategies

Pooch, Udo W., William H. Greene, and Gary G. Moss
Telecommunications and Networking

Reingold, Edward M., and Wilfred J. Hansen
Data Structures

Savitch, Walter J.
Abstract Machines and Grammars

Shneiderman, Ben
Software Psychology Human Factors in Computer and Information Systems

Simpson, Tom, and Shaffer & Shaffer Applied Research & Development, Inc.
VisiCalc® Programming: No Experience Necessary

Walker, Henry M.
Problems for Computer Solutions Using FORTRAN

Walker, Henry M.
Problems for Computer Solutions Using BASIC

Weinberg, Gerald M.
Rethinking Systems Analysis and Design

Weinberg, Gerald M.
Understanding the Professional Programmer

Windeknecht, Thomas G.
6502 Systems Programming

Preface

I saw, when at his word the formless mass,
This world's material mould, came to a heap:
Confusion heard his voice, and wild uproar
Stood ruled, stood vast infinitude confin'd;
Till at his second bidding darkness fled,
Light shone, and order from disorder sprung.

Paradise Lost, John Milton

All information processed by a computer is ultimately encoded as a sequence of bits; the specialized field of data structures considers how to impose order and structure on those bits so that the encoded information is readily available and easy to manipulate. This field thus includes the design, implementation, and analysis of structures and techniques for information processing at all levels of complexity—from individual bits, characters, and words to aggregates such as records and files, and from abstract structures such as stacks, trees, and graphs to algorithms for searching, sorting, and storage management.

Data structures are central to computer science in general and to the discipline of programming in particular. In the more analytic areas of computer science, appropriate data structures have often been the key to significant advances in the design of algorithms. Their role in programming is no less profound: in most cases, once the appropriate data structures are carefully defined, all that remains to be done is routine coding. A comprehensive understanding of data structure techniques is thus essential in the design of algorithms and programs for all but the simplest applications.

Where there is such practical importance, college courses and textbooks are sure to follow. Since the publication of *Curriculum 68* by the Association for Computing Machinery in 1968, a course in data structures has become a core requirement in virtually every undergraduate and graduate program in computer science. A number of texts have appeared, which by now seem outdated or

inadequate. Moreover, in our teaching we have adopted a number of approaches as preferable to those in most texts. In the present text, we have assembled a core of material that is unlikely to be supplanted or revised by further research.

Organization. The chapters are organized in increasing degree of complexity and abstraction, so each can be based on earlier ones. Throughout the book, for each abstract structure we emphasize its conceptual identity as a set of operations and its possible implementations in terms of the lower level structures already discussed.

Chapter 1 introduces the algorithmic and mathematical notations we employ throughout the book by discussing a sample table search problem. This discussion also serves to show the reader the scope of the techniques presented in later chapters. Chapter 2 discusses elementary data objects at the machine level—integers, characters, and so on—and how they are represented in bits (in some curricula this material is covered in a different course; if so, it can be freely omitted without interfering with later chapters). Chapter 3 then considers primitive data structures composed of aggregates of primitive objects. It shows how structures such as arrays, records, and pointers are represented in machines and in typical high-level programming languages.

Building on this basis, Chapter 4 presents material on lists, their various implementations, and the applications to, for example, stacks and queues, sparse matrix representation, and graph representation. Chapter 5 discusses trees in similar fashion: implementations and applications. These two chapters form the core of the course, presenting between them the most important tools in the design of data structures.

The next three chapters cover various more specialized problems that have wide applicability. Chapter 6 examines the techniques used to allocate and deallocate storage, Chapter 7 examines the organization of data for efficient search, and Chapter 8 examines techniques for sorting.

The material in this book is more than sufficient for a one-semester course in data structures; we have provided enough to fill a two quarter course. By choosing only the first five chapters and selected material from the rest, the instructor could cover most important topics in a single quarter, while a semester would allow the inclusion of important additional topics. A two-semester course might include discussions of the more important exercises as well as outside reading to supplement their exposition.

Presentation. Our presentation is unique in several ways. We present only a carefully chosen fraction of the material available but supplement it with a wide variety of exercises, many of which lead the student to discover interesting alternatives. The more complex exercises and those requiring advanced techniques are marked with a ★.

No single book or course can successfully discuss all known data structures and algorithms; far too many minor variants have been devised for special purposes. Rather than an encyclopedic catalog, we present the *art* of designing data structures to prepare the student to devise his own special-purpose structures for problems he will encounter.

Examples illustrating the techniques presented have been selected from many different application areas, in order to indicate the importance and ubiquity of data structures. In selecting examples and applications for presentation, we have taken care to keep the presentations self-contained and to avoid undue digressions.

Our presentation is machine and programming language independent. We use **if-then-else, while-do, repeat-until**, and **case** for flow of control, and we have chosen a functional notation for reference to the fields of a record. In this way our notation is readily understood and unambiguous, but unencumbered by the syntax of any specific language.

The presentation has been organized to be clear and interesting to both undergraduate and graduate students. The material covered is accessible to students who have completed an introductory programming course.

We recognize that the student must eventually be able to choose among implementations on the basis of the analyses of the behavior of the corresponding algorithms, but it is beyond the scope of this book to teach any but a few of the basic mathematical techniques of algorithm analysis. We have skirted this issue in part by giving brief sketches of the methods of analysis for certain key algorithms, but mainly by just summarizing the results of analysis for most algorithms. Where mathematical arguments have been unavoidable, we have emphasized the intuition behind the argument.

We have not cluttered the presentation with involved discussions of the origins of the various techniques, except where such discussions are necessary to put the material in proper perspective. The annotated bibliography that concludes each chapter provides sources for students interested in deeper treatments of the topics.

ACKNOWLEDGMENTS

> Nothing corrupts a man as deeply as writing a book, the myriad of temptations are overpowering.
>
> Nero Wolfe, in John McAleer, *Rex Stout: A Biography*

No good textbook can be written in a vacuum. Without the comments and suggestions of critical readers, the authors' impatience would introduce many errors and careless presentations.

We are fortunate to have had the benefit of comments from a number of critical readers, some voluntary (colleagues and reviewers) and some involuntary (students). All added immeasurably to this book, but it is with special gratitude that we thank Amitava Bagchi, Marcia Brown, and Nachum Dershowitz for the time they spent in looking at the manuscript and making comments about it.

A very special acknowledgment is due to John S. Tilford who, in writing the very complete solution manual available for this text, suggested innumerable improvements in the presentation.

E.M.R.
W.J.H.

TO THE READER

Si qua videbuntur chartis tibi, lector, in istis
  sive obscura nimis sive Latina parum,
non meus est error: nocuit librarius illis
  dum properat versus adnumerare tibi.
quod si non illum sed me peccasse putabis,
  tunc ego te credam cordis habere nihil.
"Ista tamen mala sunt." quasi nos manifesta negemus!
  haec mala sunt, sed tu non meliora facis.

Martial, *Epigrams, II, viii*

Contents

Chapter 1

Concepts and Examples

> So she went on, wondering more and more at every step, as everything turned into a tree the moment she came up to it.
>
> *Through the Looking Glass,* Lewis Carroll

Data structures represent *data* and relationships among data for manipulation by computer systems; *data bases* are an important class of computer system. To distinguish these three terms and to establish the context for our study of data structures, this section briefly explores the topics of data and data bases.

Data are considerably more than just a collection of numbers; each number must be *interpreted* as part of a *model.* For example, the values 153 and 169 are meaningless until described as the 1970 and 1980 counts of the number of United States cities with population over 100,000. To a company that sells to large cities, these values help define the market and its possible growth, but only when they are interpreted as part of a model of cities. In general, a model has *representations* of *objects.* In typical models the objects might be physical objects, persons, events, or abstract concepts. For the purpose of the model, certain *attributes* of each object are selected and the value for each is recorded. The collection of attribute values is then the representation of the object.

The usual purpose of a model is to analyze the behavior of the objects represented. For instance, models of wing sections of aircraft are evaluated in wind tunnels to determine whether the corresponding real wing will fly without turbulence; models of the economy are evaluated to determine the effect of possible changes in the money supply. Because models do not reflect all properties of the real object, care must be taken in evaluating the results of computations with the model. The models of the wings or the economy may predict good

behavior, but disaster can occur in reality if crucial parameters have been omitted or misrepresented.

A computer data base is one common form of model. The simplest data bases have one *record* for each object, and each record contains values corresponding to the object's various attributes. The data base itself is a model of the entire collection of objects, while each record is a model of a specific object. The term "record" derives from manual filing systems where, say, a personnel file will have one folder (record) for each employee (object). Each captioned space of a form in the folder is an attribute of the employee.

Whether manual or mechanized, the personnel data base solves both *periodic* and *one-time* problems. One periodic problem is that of paying all employees each week; this requires a sequential process that goes through the entire data base writing checks. A possible one-time problem is that of finding an employee with the skill for some special task—for example, guiding a visitor who speaks Mandarin Chinese. Such a request for information from a data base is called a *query*. It may be met by scanning the entire file, but if an appropriate index exists, only a fraction of the records need be examined.

Large corporations maintain data bases to model many aspects of their business. Human relations are covered by records on employees, departments, suppliers, customers, and stockholders. The work is represented by files on plans, designs, warehouse contents (inventory), and projects in progress. The financial picture is given by budget, accounts payable, accounts receivable, general ledger, assets, and liabilities. As with all data bases, there are many relationships among these files. An employee is in a department, is working on projects, is accounted for by a budget line, may be a customer, and may be a stockholder. A project is performed by employees, may have budget items, requires material from suppliers, and may be slated for delivery to some customer. Typical contents of corporate files are shown in Table 1.1, and possible relationships are illustrated in Figure 1.1.

Table 1.1 shows two visits on April 1. In the first, Willie Clout proposed purchase of a software package for sorting. In the second, Al Bennet, software salesman, presented the benefits and costs of enhanced software for the visitor's log. From the budget data it seems the log has higher priority than sorting.

With on-line data base systems a manager can pose *queries* and get answers quickly. For example, suppose the president is impressed by Clout's sort presentation. The decision-making process might involve several queries, perhaps starting with

> Find *budget_item* with name containing "sort."
> Show *amount_budgeted*.

In response to this query the system searches the data base, or part of it, to find the appropriate record(s).

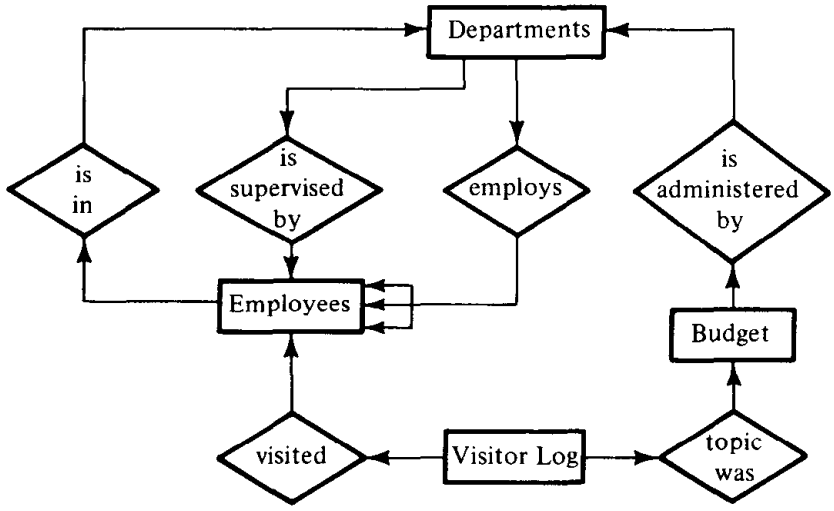

Figure 1.1
Relationships among the data bases of a corporation. Rectangles are data bases; diamonds are relationships between them. Items in the data base at the tail of an arrow refer to items in the data base at the head of the arrow. For example, the upper right diamond represents the fact that each budget item lists the department responsible for administering that item. The multiheaded arrow indicates that departments employ more than one person.

Such search problems are quintessential to computer science; they are the principal focus of the study of data structures. In the remainder of this chapter we will use a simple search problem together with two elementary solutions to illustrate our notation and important aspects of algorithmic analysis. More difficult search problems and intricate solutions are presented in Chapter 7. It is often necessary to sort data into order (for example, alphabetical or numerical) to accomodate searching it; sorting algorithms are discussed in Chapter 8. To facilitate searching and sorting, data are stored in lists and trees, presented in Chapters 4 and 5, respectively. In turn, the manipulation of lists and trees requires that memory be organized for rapid allocation. Such memory management techniques are treated in Chapter 6. Chapters 2 and 3 present the basic components from which lists, trees, and other structures are composed.

Exercises

Complex exercises and those requiring advanced techniques are keyed ★.

1. Typical automated weather forecast systems subdivide a region and represent weather conditions in each sector. What data attributes might be chosen for a sector? What is a reasonable size for a sector? With your above choices, how many data values must be stored for the 10^8 square miles of the earth?

Departments

No.	*Name*	*Supervisor*	*Personnel*
37	Information Systems	(R. S. Teague)	(P. Larson,...)
52	Executive Suite	(H. R. Ahner)	(R. Stocks,...)

Budget

No.	*Name*	*Dept.*	*Amount*	*Increase (%)*
37-291.5	Sort Package	(37)	4750	4
52-153	Visitor Log	(52)	7000	15

Employees

No.	*Name*	*Dept.*	*Salary*	*Hire Date*
1728	H. R. Ahner	(52)	65,200	5/28/79
1967	R. S. Teague	(37)	54,360	4/11/67
2053	P. Larson	(37)	13,200	10/30/81
2271	R. Stocks	(52)	25,600	1/1/82

Visitor Log

Date	*Time*	*Min.*	*Visited*	*Visitor*	*Budget Item*	*Topic*
4/1/84	11:30	13	(Ahner)	W. Clout	(37-291.5)	Better sort
4/1/84	11:55	86	(Ahner)	A. Bennet	(52-153)	Improved log

Table 1.1
Data bases for a corporation. The attributes shown are only a selection of the much larger number of attributes in the file. A value in parentheses is a reference to a record in another file; it may be just the value shown, or it may be an encoding of the location of the appropriate record.

2. Sometimes it is difficult to distinguish an object from its representations. For instance, a computer program may have many different physical representations. List at least five and for each, argue that it, rather than the others, is the "program" itself.

3. Each record in a payroll file represents a person as its object. What attributes might objects in this file have?

4. What steps might a system take to respond to a query from the president for all budget changes that were above average in value?

5. The visitor log data base contains a reference to the budget item discussed during the visit. This might be implemented as the number of the budget item or as the *address* that describes where the budget item is stored. Each solution has an advantage over the other; give these advantages. Consider speed of operation and the difficulty of changing the data base.

6. What data bases might a college or university implement? What are some queries that might be generated to work on the following problems?
   (a) How can we give raises when total income has not risen?

(b) Should we limit the number of faculty given tenure?
(c) Have academic standards changed? Is there grade inflation or deflation or has the quality of the students changed?
(d) Shall we build a new building for the Computer Science Department?

1.1 DATA STRUCTURES

A data structure can be defined informally as an organized collection of values and a set of operations on them. For the integer data structure, the operations are just addition, multiplication, and so on; for a data base, the operations may include complex queries. But most data structures fall between these extremes: an array has storage and retrieval via subscripts; a table of names has the operations of insertion and search.

Somewhat more formally, we define a data structure to have three components:

A set of *function definitions*: each function is an operation available to the rest of the program.
A *storage structure* specifies classes of values, collections of variables, and relations between variables as necessary to implement the functions.
A set of algorithms, one for each function: each algorithm examines and modifies the storage structure to achieve the result defined for the corresponding function.

The function definitions separate the implementation of the data structure from the construction of the rest of the program. They define the externally observable behavior of the data structure, while the storage structures and algorithms are the internal details. The latter can be changed without modifying routines that use these functions.

One simple example of a data structure is an array. As an introduction, we will define an array of twenty **real** *values* with **integer** *subscripts*. The set of functions is

store: Must be given a value and a subscript. Associates the value with that particular subscript.
retrieve: Must be given a subscript. Reports back the value most recently associated with that subscript.

As these descriptions show, functions must sometimes be given values. We can also say the values are required by the function or are its *arguments*. A function may report back or *return* a value, as does *retrieve*. A function may also communicate back a value by assigning it to a variable.