Discrete Computational Structures

This is a volume in
COMPUTER SCIENCE AND APPLIED MATHEMATICS
A Series of Monographs and Textbooks

Editor: WERNER RHEINBOLDT

A complete list of titles in this series appears at the end of the volume.

DISCRETE COMPUTATIONAL STRUCTURES

Robert R. Korfhage
SOUTHERN METHODIST UNIVERSITY

ACADEMIC PRESS New York and London
A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1

Library of Congress Cataloging in Publication Data

Korfhage, Robert R
Discrete computational structures.

(Computer science and applied mathematics)
1. Electronic digital computers–Programming.
2. Electronic data processing–Mathematics. I. Title.
QA76.6.K68 001.6'4'044 73-9432
ISBN 0–12–420850–9

AMS (MOS) 1970 Subject Classifications: 02B01, 05C01, 06A01, 68A01
ACM Subject Classifications: 1.1, 5.30

PRINTED IN THE UNITED STATES OF AMERICA

Contents

Preface

The digital computer is a machine which is inherently finite in all of its aspects which are of importance to a programmer or user. Whether the problem posed to the computer relates to numerical computation, symbol manipulation, information retrieval, or picture generation, the procedures and results are restricted by the finiteness of word length and memory size, and by the discrete time steps in which a computer operates. Hence for a thorough understanding of the capabilities and limitations of computers, it is important to be aware of the impact of these restrictions. Thus the purpose of this text is to bring together many of the concepts from discrete mathematics which are important to computing. Throughout I have tried to keep two questions in mind: (1) How does this topic influence the theory and practice of computing? (2) How is the computer used to solve problems in this topic? These questions cannot always be given equally good answers, and there are topics, particularly early in the book, where any relation between the topic and computing is largely suppressed. Hopefully, these topics are few and brief, and in the nature of a mathematical background necessary to the remainder of the book.

The text is designed as a sophomore- or junior-level book, corresponding to the course B3, Introduction to Discrete Structures, in the ACM Curriculum 68. Thus I assume that the student knows and can use at least one high level programming language, and I use this assumption in both the text and the exercises. I view B3 as the more mathematical half of a one-year course. Thus the student who has covered the material presented here should be ready to use these mathematical concepts in courses which deal more specifically and directly with computers. He should have the basic mathematics to enable him to feel comfortable in undergraduate courses in data

structures (the natural follow-up to B3), switching circuits, and automata theory. With this in mind, topics such as lists, circuit design, and finite state machines have been largely omitted, to be covered in succeeding courses.

This book is not intended as a formal, rigorous introduction to the theory of discrete structures, although I believe that the mathematical content is accurate and not misleading. It has been my experience that most computer science students at this level do not appreciate the subtleties of the theory. Hence I have tried to motivate the theoretical constructs as thoroughly as possible from computing, and to present any arguments, however formal, in an informal setting. Nevertheless, I firmly believe that the instructor should be willing and able to put the challenge of formal work to those students who can absorb more theory.

The text is quite loosely structured, with ample material so that the instructor can pick and choose. Very little mathematical background is required of the student, although students who have had a course in the foundations of modern mathematics will find much of Chapters 1 and 9 to be review material. Basically there are five sections to the text, and almost any permutation of these sections makes sense. However, my experience with the material has led me to the present order, both among and within sections.

Chapter 1 constitutes the first section. This covers the basic mathematical background needed, and should be studied first. It also includes some discussion of the computer representation of sets.

The section on graph theory consists of Chapters 2–4. The organization progresses from the simplest concepts toward more complex ones. The sections on storage minimization and bandwidth present an example of the close interaction which is possible between mathematics and computing, to the benefit of both. Chapter 3 develops a conceptual framework which has been little publicized ouside of research papers, as does the last section of Chapter 4.

The algebra section, Chapters 5–7, concentrates on semigroups, groups, and lattices. This does not imply that rings, fields, and universal algebras are unimportant, but rather that at this level of text, and for the purposes of this course, these latter structures can well be omitted.

The concepts of Boolean algebra and propositional calculus are covered, in rather traditional fashion, in Chapters 8 and 9. A new tabular method of Boolean function minimization is described. Discussion of the Polish notation in Chapter 9 provides a nice tie back to the concepts of Chapter 3. The predicate calculus is omitted, for much the same reason that certain algebraic structures are omitted: little is lost by postponing discussion of it until a later course.

The final section of the book, centered around combinatorics and probability, is perhaps the most novel. Chapter 10 introduces the student to the

general process of enumerating objects, through a number of examples showing different techniques. Polya's theory, while extremely important, is sufficiently complex that it cannot properly be covered in a course at this level. Hence I have reluctantly omitted any reference to it. Chapter 11 is totally in the nature of a case study. The problem chosen is a complex one which has numerous ties into applications in operations research, and which illustrates both the use of a computer, and the importance of knowing something about the combinatorial nature of a problem before doing the programming. Finally, Chapter 12 introduces the student to some of the probabilistic concepts which he will need repeatedly in his career.

A few exercises, mostly programs, have been marked (*) or (**). The single asterisk identifies a difficult problem, while the double asterisk marks a problem which the students will probably find extremely difficult or impossible to do. These problems are included both to challenge the brighter students, and, more importantly, to provide material for class discussion. That is, I suggest assigning a (**) problem for the students to spend a week on (without really expecting them to solve it), and then spending time in class discussing the difficulties which they have encountered, and how to work around them.

Acknowledgments

The writing of any book is an iterative process, involving, in the case of textbooks, discussions with colleagues, trying the material out on several classes, and so forth. Much of this background is inherently vague and unrecognizable. However, I would like to thank the several classes of students at Purdue University and Southern Methodist University who have suffered through the preliminary versions of this text, and also to especially thank A. T. Berztiss, S. T. Hedetnemi, R. E. Nance, and M. B. Wells, among my colleagues, for their comments and criticisms. Any errors which remain after their careful scrutiny are, of course, my responsibility.

The production of this book has been greatly facilitated by my able typists, Talitha Williams and Mary Kateley, and by the staff of Academic Press.

CHAPTER 1

Basic Forms and Operations

1. INTRODUCTION

This is a book about structures. Between this page and the final one we shall examine a wide variety of mathematical frameworks. We shall compare them—finding at times that we have two or more quite distinct ways of describing a framework. We shall explore and map them, so that we can more easily find our position in a given framework, and can reference and discuss various locations within the framework. We shall clothe these frameworks with data and with programs, and examine what can be said about the resulting structures. Always, since our interest is in digital computation, our structures will be discrete. That is, they will have no more parts than there are integers.

A computer is basically a data transformer. Given certain input data, a computer will, by means of its programs and its internal circuitry, transform the data into other data which constitute the output of the machine. As users, we may interpret the data to represent numbers, words, music, pictures, and so on. The data become information. We propose to examine the concept of data as it relates to computers, to determine the basic forms

that data can take regardless of the interpretation, and to study these forms for properties which affect computation. We shall begin with the simplest form, the set, and progress into data forms with more complex structures.

2. ELEMENTS AND SETS

The simplest structure that we encounter is the set. Without defining *set* precisely, a set is a mere collection of "things": books, computers, concepts, colors, words, numbers, cars, and so forth. These "things" in a set are called the *elements* or *members* of the set. A set, as a set, is almost devoid of properties: there is no order to the set, no relationship among the elements other than that they all belong to the set. In fact, the only basic property which we shall assume is that, given a "thing" x and a set A, we can determine whether x is an element of A. We shall denote set membership by the symbol $\in$, writing $x \in A$ if x is an element of A, and $x \notin A$ if x is not an element of A.

A set may be described either by listing all of its elements within braces, or by stating a property that determines whether a "thing" is an element of the set.

Example 2.1

Describe the sets consisting of the elements (a) 1, 2, 3; (b) a, e, i, o, and u; (c) red, yellow, and blue.

ANSWER. (a) $\{1, 2, 3\}$, (b) $\{a, e, i, o, u\}$, (c) {red, yellow, blue}.

Whenever the meaning is clear, we may use ellipses (...) to abbreviate the list. Thus the set of all integers between 1 and 1,000,000 may be written $\{1, 2, 3, \ldots, 1{,}000{,}000\}$, and the set of all negative integers may be written $\{-1, -2, -3, \ldots\}$.

It should be noted that a change in the order in which the elements of a set are listed does not change the set. Nor does multiple listing of elements change the set, any more than multiple listing of houses for sale changes the set of houses available. Thus the listings $\{1, 2, 3\}$, $\{2, 3, 1\}$, and $\{1, 1, 3, 2\}$ all describe the same set.

The form used to describe a set by a property is $\{x \mid P(x)\}$, read "the set of all x such that $P(x)$ holds," where P is the defining property.