Applied Discrete Structures

Alan Doerr and Kenneth Levasseur

Department Of Mathematical Sciences
University of Massachusetts Lowell

Version 1.0
March 2012



Home Blog Errata

Home: http://faculty.uml.edu/klevasseur/ADS2/"
Blog: http://applieddiscretestructures.blogspot.com/
Errata: http://faculty.uml.edu/klevasseur/ADS2/errata.html

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States License.

(http://creativecommons.org/licenses/by-nc-nd/3.0/us/)

Previously published by Pearson Education, Inc. under the title *Applied Discrete Structures for Computer Science*

To our families
Donna, Christopher, Melissa, and Patrick Doerr
and
Karen, Joseph, Kathryn, and Matthew Levasseur

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

Table of Contents

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

Preface - what a difference 21 years make!

Twenty-one years after the publication of the 2nd edition of *Applied Discrete Structures for Computer Science*, in 1989 the publishing and computing landscape have both changed dramatically. We signed a contract for the second edition with Science Research Associates but by the time the book was ready to print, SRA had been sold to MacMillan. Soon after, the rights had been passed on to Pearson Education, Inc. In 2010, the long-term future of printed textbooks is uncertain. In the meantime, textbook prices (both printed and e-books) have increased and a growing open source textbook market movement has started. One of our objectives in revisiting this text is to make it available to our students in an affordable format. In its original form, the text was peer-reviewed and was adopted for use at several universities throughout the country. For this reason, we see *Applied Discrete Structures* as not only an inexpensive alternative, but a high quality alternative.

As indicated above the computing landscape is very different from the 1980's and accounts for the most significant changes in the text. One of the most common programming languages of the 1980's, Pascal; and we used it to illustrate many of the concepts in the text. Although it isn't totally dead, Pascal is far from the mainstream of computing in the 21st century. In 1989, *Mathematica* had been out for less than a year — now a major force in scientific computing. The open source software movement also started in the 1980's and in 2005, the first version of Sage, an open-source alternative to *Mathematica* was first released. In *Applied Discrete Structures* we have replaced "Pascal Notes" with "*Mathematica* Notes" and "Sage Notes." Finally, 1989 was the year that World Wide Web was invented by Tim Berners-Lee. There wasn't a single www in the 2nd edition. In this version, we intend to make use of extensive web resources, including video demonstrations.

We would like to thank Tony Penta, Sitansu Mittra, and Dan Klain for using the preliminary versions of *Applied Discrete Structures*. The corrections and input they provided was appreciated.

We repeat the preface to *Applied Discrete Structures for Computer Science* below. Plans for the instructor's guide, which is mentioned in the preface are uncertain at this time.

Preface to Applied Discrete Structures for Computer Science, 2nd Ed.

We feel proud and fortunate that most authorities, including MAA and ACM, have settled on a discrete mathematics syllabus that is virtually identical to the contents of the first edition of Applied Discrete Structures for Computer Science. For that reason, very few topical changes needed to be made in this new edition, and the order of topics is almost unchanged. The main change is the addition of a large number of exercises at all levels. We have "fine-tuned" the contents by expanding the preliminary coverage of sets and combinatorics, and we have added a discussion of binary integer representation. We have also added an introduction including several examples, to provide motivation for those students who may find it reassuring to know that mathematics has "real" applications. "Appendix B—Introduction to Algorithms," has also been added to make the text more self-contained.

How This Book Will Help Students

In writing this book, care was taken to use language and examples that gradually wean students from a simpleminded mechanical approach and move them toward mathematical maturity. We also recognize that many students who hesitate to ask for help from an instructor need a readable text, and we have tried to anticipate the questions that go unasked.

The wide range of examples in the text are meant to augment the "favorite examples" that most instructors have for teaching the topics in discrete mathematics.

To provide diagnostic help and encouragement, we have included solutions and/or hints to the odd-numbered exercises. These solutions include detailed answers whenever warranted and complete proofs, not just terse outlines of proofs.

Our use of standard terminology and notation makes Applied Discrete Structures for Computer Science a valuable reference book for future courses. Although many advanced books have a short review of elementary topics, they cannot be complete.

How This Book Will Help Instructors

The text is divided into lecture-length sections, facilitating the organization of an instructor's presentation.

Topics are presented in such a way that students' understanding can be monitored through thought-provoking exercises. The exercises require an understanding of the topics and how they are interrelated, not just a familiarity with the key words.

An Instructor's Guide is available to any instructor who uses the text. It includes:

(a) Chapter-by-chapter comments on subtopics that emphasize the pitfalls to avoid;

(b) Suggested coverage times;

(c) Detailed solutions to most even-numbered exercises;

(d) Sample quizzes, exams, and final exams.

How This Book Will Help the Chairperson/Coordinator

The text covers the standard topics that all instructors must be aware of; therefore it is safe to adopt Applied Discrete Structures for Computer Science before an instructor has been selected.

The breadth of topics covered allows for flexibility that may be needed due to last-minute curriculum changes.

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

Since discrete mathematics is such a new course, faculty are often forced to teach the course without being completely familiar with it. An Instructor's Guide is an important feature for the new instructor.

What a Difference Five Years Makes!

In the last five years, much has taken place in regards to discrete mathematics. A review of these events is in order to see how they have affected the Second Edition of Applied Discrete Structures for Computer Science.

(1) Scores of discrete mathematics texts have been published. Most texts in discrete mathematics can be classified as one-semester or two-semester texts. The two-semester texts, such as Applied Discrete Structures for Computer Science, differ in that the logical prerequisites for a more thorough study of discrete mathematics are developed.

(2) Discrete mathematics has become more than just a computer science support course. Mathematics majors are being required to take it, often before calculus. Rather than reducing the significance of calculus, this recognizes that the material a student sees in a discrete mathematics/structures course strengthens his or her understanding of the theoretical aspects of calculus. This is particularly important for today's students, since many high school courses in geometry stress mechanics as opposed to proofs. The typical college freshman is skill-oriented and does not have a high level of mathematical maturity. Discrete mathematics is also more typical of the higher-level courses that a mathematics major is likely to take.

(3) Authorities such as MAA, ACM, and A. Ralson have all refined their ideas of what a discrete mathematics course should be. Instead of the chaos that characterized the early '80s, we now have some agreement, namely that discrete mathematics should be a course that develops mathematical maturity.

(4) Computer science enrollments have leveled off and in some cases have declined. Some attribute this to the lay-offs that have taken place in the computer industry; but the amount of higher mathematics that is needed to advance in many areas of computer science has also discouraged many. A year of discrete mathematics is an important first step in overcoming a deficiency in mathematics.

(5) The Educational Testing Service introduced its Advanced Placement Exam in Computer Science. The suggested preparation for this exam includes many discrete mathematics topics, such as trees, graphs, and recursion. This continues the trend toward offering discrete mathematics earlier in the overall curriculum.

Acknowledgments

The authors wish to thank our colleagues and students for their comments and assistance in writing and revising this text. Among those who have left their mark on this edition are Susan Assmann, Shim Berkovitz, Tony Penta, Kevin Ryan, and Richard Winslow.

We would also like to thank Jean Hutchings, Kathy Sullivan, and Michele Walsh for work that they did in typing this edition, and our department secretaries, Mrs. Lyn Misserville and Mrs. Danielle White, whose cooperation in numerous ways has been greatly appreciated.

We are grateful for the response to the first edition from the faculty and students of over seventy-five colleges and universities. We know that our second edition will be a better learning and teaching tool as a result of their useful comments and suggestions. Our special thanks to the following reviewers: David Buchthal, University of Akron; Ronald L. Davis, Millersville University; John W Kennedy, Pace University; Betty Mayfield, Hood College; Nancy Olmsted, Worcester State College; and Pradip Shrimani, Southern Illinois University. Finally, it has been a pleasure to work with Nancy Osman, our acquisitions editor, David Morrow, our development editor, and the entire staff at SRA.

A.W. D.

K.M.L.

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

Introduction

What Is Discrete Mathematics/Structures?

What is Discrete Mathematics?

As a general description one could say that discrete mathematics is the mathematics that deals with "separated" or discrete sets of objects rather than with continuous sets such as the real line. For example, the graphs that we learn to draw in high school are of continuous functions. Even though we might have begun by plotting discrete points on the plane, we connected them with a smooth, continuous, unbroken curve to form a straight line, parabola, circle, etc. The underlying reason for this is that hand methods of calculation are too laborious to handle huge amounts of discrete data. The computer has changed all of this.

Today, the area of mathematics that is broadly called "discrete" is that which professionals feel is essential for people who use the computer as a fundamental tool. It can best be described by looking at our Table of Contents. It involves topics like sets, logic, and matrices that students may be already familiar with to some degree. In this Introduction, we give several examples of the types of problems a student will be able to solve as a result of taking this course. The intent of this Introduction is to provide an overview of the text. Students should read the examples through once and then move on to Chapter One. After completing their study of discrete mathematics, they should read them over again.

We hope discrete mathematics is as fascinating and enjoyable to the student as it has been to us.

Example I.a. Analog-to-digital Conversion. A common problem encountered in engineering is that of analog-to-digital (a-d) conversion, where the reading on a dial, for example, must be converted to a numerical value. In order for this conversion to be done reliably and quickly, one must solve an interesting problem in graph theory. Before this problem is posed, we will make the connection between a-d conversion and the graph problem using a simple example. Suppose a dial in a video game can be turned in any direction, and that the positions will be converted to one of the numbers zero through seven in the following way. As depicted in Figure I.a.1, the angles from 0 to 360 are divided into eight equal parts, and each part is assigned a number starting with 0 and increasing clockwise. If the dial points in any of these sectors the conversion is to the number of that sector. If the dial is on the boundary, then we will be satisfied with the conversion to either of the numbers in the bordering sectors. This conversion can be thought of as giving an approximate angle of the dial, for if the dial is in sector k, then the angle that the dial makes with east is approximately $45\,k°$.

FIGURE I.a.1

Now that the desired conversion has been described, we will describe a "solution" that has one major error in it, and then identify how this problem can be rectified. All digital computers represent numbers in binary form, as a sequence of Os and Is called bits, short for binary digits. The binary representations of numbers 0 through 7 are:

$$0 = 000 = 0\times4 + 0\times2 + 0\times1$$

$$1 = 001 = 0\times4 + 0\times2 + 1\times1$$

$$2 = 010 = 0\times4 + 1\times2 + 0\times1$$

$$3 = 011 = 0\times4 + 1\times2 + 1\times1$$

$$4 = 100 = 1\times4 + 0\times2 + 0\times1$$

$$5 = 101 = 1\times4 + 0\times2 + 1\times1$$

$$6 = 110 = 1\times4 + 1\times2 + 0\times1$$

$$7 = 111 = 1\times4 + 1\times2 + 1\times1$$

We will discuss the binary number system in Chapter 1. The way that we could send those bits to a computer is by coating parts of the back of the dial with a metallic substance, as in Figure I.a.2. For each of the three concentric circles on the dial there is a small magnet. If a magnet lies under a part of the dial that has been coated with metal, then it will turn a switch ON, whereas the switch stays OFF when no metal is detected above a magnet. Notice how every ON/OFF combination of the three switches is possible given the way the back of the dial is coated.

If the dial is placed so that the magnets are in the middle of a sector, we expect this method to work well. There is a problem on certain boundaries, however. If the dial is turned so that the magnets are between sectors three and four, for example, then it is unclear what the result will be. This is due to the fact that each magnet will have only a fraction of the required metal above it to turn its switch ON. Due to expected irregularities in the coating of the dial, we can be safe in saying that for each switch either ON or OFF could be the result, and so if the dial is between sectors three and four, any number could be indicated. This problem does not occur between every sector. For example, between sectors 0 and 1, there is only one switch that cannot be predicted. No matter what the outcome is for the units switch in this case, the indicated sector must be either 0 or 1, which is consistent with the original objective that a positioning of the dial on a boundary of two sectors should produce the number of either sector.

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

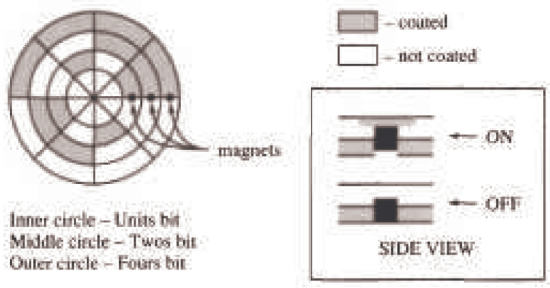

FIGURE I.a.2

Is there a way to coat the sectors on the back of the dial so that each of the eight patterns corresponding to the numbers 0 to 7 appears once, and so that between any two adjacent sectors there is only one switch that will have a questionable setting? One way of trying to answer this question is by using an undirected graph called the 3-cube (Figure I.a.3). In general, an undirected graph consists of vertices (the circled 0's and 1's in the 3-cube) and the edges, which are lines that connect certain pairs of vertices. Two vertices in the 3-cube are connected by an edge if the sequences of the three bits differ in exactly one position. If one could draw a path along the edges in the 3-cube that starts at any vertex, passes through every other vertex once, and returns to the start, then that sequence of bit patterns can be used to coat the back of the dial so that between every sector there is only one questionable switch. Such a path is not difficult to find; so we will leave it to you to find one, starting at 000 and drawing the sequence in which the dial would be coated.

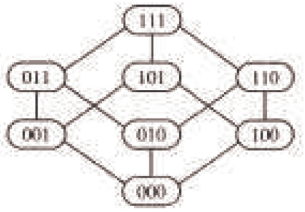

FIGURE I.a.3

Many A-D conversion problems require many more sectors and switches than this example, and the same kinds of problems can occur. The solution would be to find a path within a much larger yet similar graph. For example, there might be 1,024 sectors with 10 switches, resulting in a graph with 1,024 vertices. One of the objectives of this text will be to train you to understand the thought processes that are needed to attack such large problems. In Chapter 9 we will take a closer look at graph theory and discuss some of its applications.

One question might come to mind at this point. If the coating of the dial is no longer as it is in Figure I.a.2, how would you interpret the patterns that are on the back of the dial as numbers from 0 to 7? In Chapter 14 we will see that if a certain path is used, this "decoding" is quite easy.

The 3-cube and its generalization, the *n*-cube, play a role in the design of a multiprocessor called a hypercube. A multiprocessor is a computer that consists of several independent processors that can operate simultaneously and are connected to one another by a network of connections. In a hypercube with $M = 2^n$ processors, the processors are numbered 0 to $M - 1$. Two processors are connected if their binary representations differ in exactly one bit. The hypercube has proven to be the best possible network for certain problems requiring the use of a "supercomputer." Denning's article in the May-June 1987 issue of "American Scientist" provides an excellent survey of this topic.

Example I.b. Logic is the cornerstone of all communication, whether we wish to communicate in mathematics or in any other language. It is the study of sentences, or propositions, that take on the values true or false, 1 or 0 in the binary system. Its importance was recognized in the very early days of the development of logic (hardware) design, where Boolean algebra, the algebra of logic, was used to simplify electronic circuitry called gate diagrams. Consider the following gate diagram:

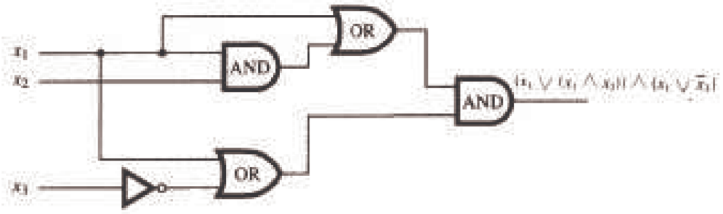

FIGURE I.b.1

Each symbol in this diagram is called a gate, a piece of hardware. In Chapter 13 we will discuss these circuits in detail. Assume that this circuitry can be placed on a chip which will have a cost dependent on the number of gates involved. A classic problem in logic design is to try to simplify this circuitry to one containing fewer gates. Indeed, the gate diagram can be reduced to

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works

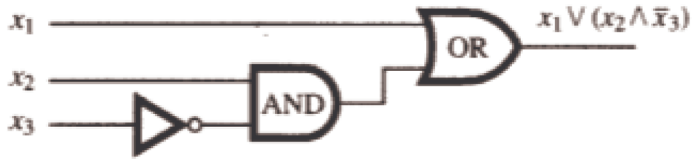

FIGURE I.b.2

The result is a less costly chip. Since a company making computers uses millions of chips, we have saved a substantial amount of money.

This use of logic is only the "tip of the iceberg." The importance of logic for computer scientists in particular, and for all people who use mathematics, cannot be overestimated. It is the means by which we can think and write clearly and precisely. Logic is used in writing algorithms, in testing the correctness of programs, and in other areas of computer science.

Example I.c. Suppose two students miss a class on a certain day and borrow the class notes in order to obtain copies. If one of them copies the notes by hand and the other walks to a "copy shop," we might ask which method is more efficient. To keep things simple, we will only consider the time spent in copying, not the cost. We add a few more assumptions: copying the first page by hand takes one minute and forty seconds (100 seconds); for each page copied by hand, the next page will take five more seconds to copy, so that it takes 1:45 to copy the second page, 1:50 to copy the third page, etc.; photocopiers take five seconds to copy one page; walking to the "copy shop" takes ten minutes, each way.

One aspect of the problem that we have not specified is the number of pages to be copied. Suppose the number of pages is n, which could be any positive integer. As with many questions of efficiency, one method is not clearly better than the other for all cases. Since the only variable in this problem is the number of pages, we can simply compare the copying times for different values of n. We will denote the time it takes (in seconds) to copy n pages manually by $t_h(n)$, and the time to copy n pages automatically by $t_a(n)$. Ideally, we would like to have formulas to represent the values of $t_h(n)$ and $t_a(n)$. The process of finding these formulas is an important one that we will examine in Chapter 8. The formula for $t_a(n)$ is not very difficult to derive from the given information. To copy pages automatically, one must walk for twenty minutes (1,200 seconds), and then for each page wait five seconds. Therefore, $t_a(n) = 1200 + 5n$

The formula for $t_h(n)$ isn't quite as simple. First, let $p(n)$ be the number of seconds that it takes to copy page n. From the assumptions, $p(1) = 100$, and if n is greater than one, $p(n) = p(n-1) + 5$. The last formula is called a *recurrence relation*. We will spend quite a bit of time discussing methods for deriving formulas from recurrence relations. In this case $p(n) = 95 + 5n$. Now we can see that if n is greater than one,

$$t_h(n) = p(1) + p(2) + \cdots + p(n) = t_h(n-1) + p(n) = t_h(n-1) + 5n + 95$$

This is yet another recurrence relation. The solution to this one is $t_h(n) = 97.5n + 2.5n$.

Now that we have these formulas, we can analyze them to determine the values of n for which hand copying is most efficient, the values for which photocopying is most efficient, and also the values for which the two methods require the same amount of time.

WHAT IS DISCRETE STRUCTURES?

So far we have given you several examples of that area of mathematics called discrete mathematics. Where does the "structures" part of the title come from? We will look not only at the topics of discrete mathematics but at the structure of these topics. If two people were to explain a single concept, one in German and one in French, we as observers might at first think they were expressing two different ideas, rather than the same idea in two different languages. In mathematics we would like to be able to make the same distinction. Also, when we come upon a new mathematical structure, say the algebra of sets, we would like to be able to determine how workable it will be. How do we do this? We compare it to something we know, namely elementary algebra, the algebra of numbers. When we encounter a new algebra we ask ourselves how similar it is to elementary algebra. What are the similarities and the dissimilarities? When we know the answers to these questions we can use our vast knowledge of basic algebra to build upon rather than learning each individual concept from the beginning.

Applied Discrete Structures by Alan Doerr & Kenneth Levasseur is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works