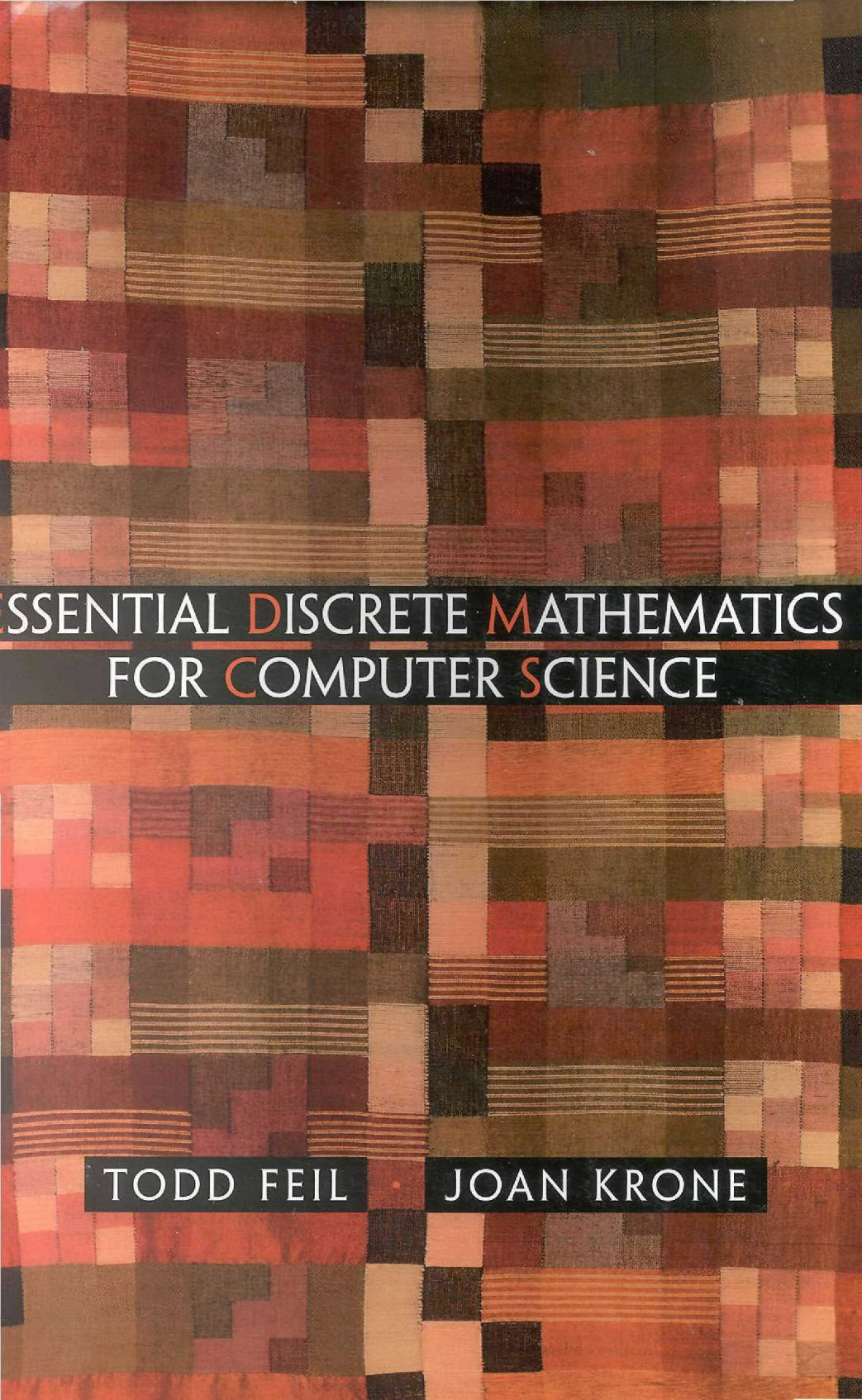
SSENTIAL DISCRETE MATHEMATICS
FOR COMPUTER SCIENCE
TODD FEIL
JOAN KRONE

Essential Discrete Mathematics for Computer Science

Todd Feil
Department of Mathematics and Computer Science
Denison University

Joan Krone
Department of Mathematics and Computer Science
Denison University

PEARSON EDUCATION, INC., Upper Saddle River, NJ 07458

Library of Congress Cataloging-in-Publication Data

Feil, Todd.
Essential discrete mathematics for computer science / Todd Feil and Joan Krone.
p. cm.
Includes bibliographical references and index.
ISBN: 0-13-018661-9
1. Computer science--Mathematics. I. Krone, Joan. II. Title

QA76.9.M35 F44 2003 2002030816
004'.01'51--dc21

Editor-in-Chief: *Sally Yagan*
Acquisition Editor: *George Lobell*
Vice-President/Director of Production and Manufacturing: *David W. Riccardi*
Executive Managing Editor: *Kathleen Schiaparelli*
Senior Managing Editor: *Linda Mihatov Behrens*
Assistant Managing Editor: *Bayani Mendoza de Leon*
Production Editor: *Jeanne Audino*
Manufacturing Buyer: *Michael Bell*
Manufacturing Manager: *Trudy Pisciotti*
Marketing Manager: *Halee Dinsey*
Marketing Assistant: *Rachel Beckman*
Art Director: *Jayne Conte*
Cover Designer: *Bruce Kenselaar*
Cover Photo: *Wallhanging made of wool and cotton by Benita Koch-Otte, 1923/24. Photo from Bauhaus-Archive, Berlin, from Bauhaus Weavings, Berlin 1998, Nr. 148.*

Pearson Education, Inc.
Upper Saddle River, New Jersey 07458

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-018661-9

Pearson Education LTD., *London*
Pearson Education Australia PTY, Limited, *Sydney*
Pearson Education Singapore, Pte. Ltd
Pearson Education North Asia Ltd, *Hong Kong*
Pearson Education Canada, Ltd., *Toronto*
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education -- Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd

Contents

Preface

If you are an instructor, why should you choose this textbook for your students? If you are a student, why should you read this text? The material included in this text provides an introduction to discrete mathematics and is intended for first year students so that their later courses in mathematics and/or computer science can be covered in more depth than they could be without this foundational background. The text is not intended to be a comprehensive collection of discrete mathematics topics, but rather it ties selected topics to concepts in computer science and it includes programming problems along with written exercises. Unlike the large, comprehensive texts, this one can be covered in a semester. For computer science students, there are programming exercises. For math students without an interest in programming, there are plenty of exercises of different levels to challenge them.

This text evolved over a 10-year period from notes for our second semester freshman course for computer science students. This course has included about two-thirds mathematics and about one-third programming. Our students have found immediate benefits in their next course, Data Structures and Algorithms Analysis, as well as all other upper level courses. You will find the style focused on the chosen topics; we make no attempt at a complete coverage of those concepts. We chose the topics with two goals in mind: to lay a strong mathematical foundation and to show that mathematics has immediate application in computer science.

There is little, if any, controversy over whether or not computer science students should study mathematics. The resounding consensus is that mathematics is critical to the study and practice of computer science. It is not so easy to gain agreement among academicians and practitioners as to exactly what areas of mathematics should be studied, how rigorous the

presentation should be, and at what points in the curriculum these ideas should be introduced.

Having been involved in the education of computer science students and having been responsible for teaching students who have taken a variety of mathematics courses, it is our belief that it is wise to include some fundamental mathematics in the first-year computer science curriculum. We believe that there are enough topics for which students can see immediate applications that it is worthwhile to make those topics a part of the CS1 or CS2 course. This is not to say that students would not need or benefit from other courses in mathematics in addition to what they learn at this point. Rather, we believe that students will enjoy and get more from later mathematics courses because they have some background in basic ideas.

This book is not intended to be "the" math course for computer science students. It is intended to help students understand the importance of mathematics and see its relevance in a variety of applications. Indeed, most students will take some sort of discrete mathematics course later in their careers. The most immediate application for students is in analyzing algorithms, something they will start doing in earnest in their next course or two. To understand not only standard arithmetic algorithms but also important algorithms in cryptology, students must understand modular arithmetic and basic number theory. Concerns arise later that require a foundation in mathematics.

Precision of expression is the key to carrying out the tasks of both program specification and program correctness, and mathematics provides the foundation for this precision. Mathematics teaches us to be exact in what we say and how we think. It gives us the capability to express our ideas in such a way as to avoid being misunderstood. The study of mathematics in general, regardless of specific content, promotes precision of expression and attention to detail in reasoning. However, we have chosen particular mathematical structures that have di-

rect applications in computer science, hence addressing two goals. First, we concern ourselves with the task of helping students develop reasoning skills and exactness of expression. Our second goal is to provide the fundamental mathematics necessary for computer scientists.

Many exercises are included at the end of each chapter. Some suggestions for programming problems have been included. Most are easily embellished or altered to meet the needs of the course. Some exercises and programming problems have been starred. These indicate more challenging problems.

> ✓ Throughout the text you will find questions displayed like this. These are usually straightforward questions to be done as the student reads the text to check if the material is understood.

Many people aided in the creation of this text. We'd like to thank first our students who, over many semesters, pointed out errors in the text (typographic and other) and offered suggestions about exercises. We'd like to mention particularly Rohit Bansal and Tony Fressola in this regard.

The editorial staff at Prentice-Hall has been particularly helpful: Patricia Daly, Jeanne Audino and George Lobell. The original manuscript has come a long way thanks to them.

And finally, we'd like to thank our spouses, Robin and Gil, for their encouragement and support.

Any errors and typos are, of course, our responsibility. We would like to hear from you if you find any. Please email us with any errors you find or comments you have about the text.

Todd Feil
feil@denison.edu

Joan Krone
krone@denison.edu

Chapter 0
Introduction to Proofs

Throughout this text, we have provided proofs for many of the theorems presented and have assigned some proofs in the exercises. Proofs are an important part of mathematics. But figuring out how to start a proof and how to proceed is especially difficult for someone new to this activity. The purpose of this chapter is to present a couple of proof techniques used in the text, give some simple yet illustrative examples, and supply some justification on why these techniques work. There are many other techniques used besides the ones we include here; this is only a starting point. Please note that one of the most important techniques, induction, is delayed until later; we have devoted Chapter 4 to this powerful method.

A proof is a convincing argument. Outside of mathematics, what constitutes a proof differs from the high standards set in mathematics. For example, if you wanted to prove that you climbed Mount Everest, you might supply photos of yourself taken on the summit or a letter from someone attesting to the fact or even the results of a lie detector test. A skeptic might protest that the photos and letters could be faked and the lie-detector fooled and therefore the evidence offers no proof at all. The courtroom is a source of many such "proofs." Bear in mind that mathematicians are the most skeptical of people, at least when it pertains to mathematics.

One of the things that separates mathematics from other intellectual endeavors is the preciseness of the claims made. This requires a certain formality in the language, and the proofs of these claims usually require the same formality. But it's true that not all proofs are formal. A goal of a proof is to communicate to someone the argument being made. (That someone may be yourself.) So the level of detail offered de-

pends on the audience. If you are a professional mathematician and you are communicating a proof to another mathematician who is an expert in the field, a proof might be a few sentences or paragraphs. The details that to an expert are well-known or easily worked out are skipped. This "handwaving" is common. An expert needs the main ideas of the proof. A less sophisticated audience needs more details to see the connections between steps in the proof. Indeed a very naive audience would be so ignorant of the subject that a great deal of effort would be necessary to make the terms intelligible. The real test here would be to keep the audience's interest maintained over the days and weeks (or longer) required.

What differs in the level of sophistication in various versions of a proof is the size of the steps used. The audience must believe that each step is justified. A mathematician believes that every theorem can be reduced to a series of formal statements, starting from a system of axioms, whose steps would be so simple that they would be easy, even trivial, to justify. However, the argument would be so long that the essence of the proof would be lost.

At the other extreme, a proof given in conversation between experts would involve huge steps, leaving a great deal of "detail" for the listener to justify. These steps, you could argue, give only the essence of the proof.

Fortunately, over the years, mathematicians have developed a style that is a blend of natural language and formalism that has evolved into a balance of preciseness and readability so you believe it would be possible to reduce the proof to a series of strictly formal statements (even though this would be a Herculean task). Your ability to read and produce proofs partly involves learning this style. We start with a short introduction to logic.

Propositional Logic

A proposition is a statement that has an associated truth