Algebra and Discrete Mathematics Vol. 3

LECTURES ON DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

Bakhadyr Khoussainov
Nodira Khoussainova

World Scientific

Algebra and Discrete Mathematics

LECTURES ON DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

Bakhadyr Khoussainov
University of Auckland, New Zealand

Nodira Khoussainova
University of Washington, USA

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI

Published by

World Scientific Publishing Co. Pte. Ltd.
5 Toh Tuck Link, Singapore 596224
USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601
UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

Algebra and Discrete Mathematics — Vol. 3
LECTURES ON DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

ISBN-13 978-981-4340-50-2
ISBN-10 981-4340-50-2

Printed in Singapore by World Scientific Printers.

Algebra and Discrete Mathematics Vol.

LECTURES ON DISCRETE MATHEMATICS FOR COMPUTER SCIENCE

We dedicate this book to our parents and grandparents

Preface

About the book: This textbook is a result of many discussions between us, the authors. The first author (Bakh) is a well-established mathematician, and the second author (Nodira) is a young computer science PhD candidate. One is an expert in the field, and the other is a student. The first author favors rigorous definitions and proofs, and prefers formal explanations over informal ones. The second author prefers informal explanations with examples. She questions every definition and proof, and asks for the motivation behind each concept. These two views of the authors are complementary, and the synergy between the two perspectives is seen throughout the book. Along with every definition there is an example, along with every proof there is a discussion, and every chapter comes with real-world programming exercises. The textbook brings together two different viewpoints to create a unified textbook on discrete mathematics designed for students in computer science, software engineering, and mathematics.

Before we started writing this textbook, our conversations about computer science and mathematics often returned to a few fundamental questions. What is a definition? What is a theorem? What is a proof? How do we explain and motivate these concepts to students? Since algorithms lie at the heart of computer science and software engineering, our discussions evolved to become more focused on algorithms and their correctness. The questions arising were now the following. What is an algorithm? What does it mean for an algorithm to be correct? How does one prove the correctness of an algorithm? What should be emphasized? Why do we need to learn correctness proofs?

We soon realized that by putting our discussions into writing, we had started the preparation of a textbook. We designed this textbook for early stage undergraduate computer science, software engineering and mathematics students. For computer science and software engineering students, the book provides the mathematical background needed to reason about algorithms, programs and their properties. For instance, the book contains many examples of correctness proofs for algorithms. For mathematics students, the book presents mathematical ideas by connecting them to fundamental concepts in computer science. For instance, the book exposes close connections between inductive proofs and the concept of iteration used in many programming languages.

As our discussions continued, we often found ourselves discussing the similarities between the process of constructing a proof and the process of designing and writing a program. The first step in both processes is to obtain a clear understanding of the problem at hand. Furthermore, for both tasks one needs to familiarize oneself with the existing knowledge and utilize it. For programs, this means knowing previously defined classes, methods and existing application programming interfaces. For proofs, this includes knowing previously proven theorems, definitions, and techniques. Along the same lines, both proofs and algorithms are often designed piece-by-piece. In proving a theorem, we often break down the problem into subproblems and try to solve them first, thus constructing lemmas. Similarly, in constructing a program, we break it down into various methods and classes. Yet another similarity is the use of examples to acquire an intuition for the algorithm or proof that one is trying to construct. All these similarities showcase the prevalence of transferable skills between discrete mathematics and computer science, thus highlighting the importance of a textbook like this one.

Book highlights: We cover a wide variety of topics in discrete mathematics. However, we would like to emphasize some particular aspects of the textbook. The first is induction. The book showcases many uses of induction, including inductive definitions of objects (such as trees, formulas of propositional logic, and regular expressions), as well as various types of proofs by induction. We think that induction is an essential method for reasoning about algorithms, programs and the objects defined by them. This is because induction is a mathematical tool that reflects the concept of iteration, which is pervasive throughout computing.

The second feature is the numerous algorithms and their proofs of correctness. Almost every lecture presents and analyzes at least one algorithm. Most theorems are proven through the analysis of algorithms. Furthermore, because most algorithms use iteration, many of the proofs of correctness utilize induction via the loop invariant theorem, thus giving the reader more practice with induction.

The third highlight is the comprehensive coverage of finite automata. Finite automata constitute a simple yet powerful mathematical model of programs. Via finite automata one can discuss state transitions of programs and their representations, consider various types of design problems, talk about simulating one program by another, and study connections between various ways of defining problems. The lectures on automata appear near the end of this book. This organization helps the reader to apply the knowledge of graphs, algorithms, and induction acquired earlier on in order to learn about finite automata and their properties.

The fourth aspect of the textbook is that every lecture is followed by a set of exercises, including programming exercises. These allow the student to better understand the concepts presented in the lecture, and to apply the newly-acquired knowledge to real problems.

Book organization: This book is designed as a textbook for a single semester course in discrete mathematics. It consists of thirty three chapters, each chapter covering a 50-minute lecture on average. We aimed to keep the book compact, and at the same time clear and easy to understand. The reader can easily flip through the book to gain some sense of the material covered. However, we now briefly outline the key topics covered in this book.

Lectures 1 through 5 are introductory. They introduce the concepts of definitions, theorems, and proofs through integer and modulo arithmetic. Key topics include the fundamental theorem of arithmetic and the Euclidian algorithm. Lecture 5 covers the RSA encryption method, and can be included or omitted at the teacher's discretion. Lectures 6 through 9 study graphs and trees, as well as algorithms on these structures. These are introduced early because they are easy to explain, intuitively understandable, and relate to real-world problems. These lectures also prepare the reader for set-theoretic notations. Key topics in these chapters are the path problems, inductive definition of trees and proofs of some of their properties using induction. Lectures 10 through 13 introduce sets and relations. Unlike traditional textbooks in discrete mathematics, we present sets and relations as a useful language to reason about databases. We hope this keeps the readers engaged and interested, since they immediately see the application of sets to something concrete and useful. Lectures 14 through 19 present induction, inductive proofs, recursion, and correctness of algorithms. The key feature is the loop invariant theorem. These lectures prove the correctness of many algorithms, including Prim's minimum spanning tree algorithm and

Djikstra's shortest path algorithm. Lectures 20 and 21 study functions. In addition to the traditional topics such as surjective, injective and bijective functions, there is a discussion of transition functions. This connects functions with the forthcoming lectures on finite automata. Lectures 22 through 24 study propositional logic. In these lectures, we employ induction to reason about the propositional calculus. The key topics include the unique readability theorem, normal forms, logical equivalence, and models. Lectures 25 through 30 introduce finite automata. The lectures put an emphasis on designing automata, determinism and non-determinism, converting non-deterministic automata to deterministic automata, as well as regular expressions. The key topics presented include the determinization theorem, Kleene's theorem, and algorithms for checking various properties of automata. Finally, Lectures 31 through 33 introduce the basics of counting and probability. The topics include counting rules and principles, permutation and combinations, definition of probability, and probability distributions. Each lecture contains many examples.

Acknowledgments: Parts of this textbook have been used in several courses at the University of Auckland, National University of Singapore, and Cornell University. We would like to thank the students of these classes for testing this book, finding mistakes, and helping us better design the structure of the book. We also thank the following people for their support, comments and suggestions: Cristian Calude, Elena Calude, Michael Dinneen, Gillian Dobbie, Aniruddh Gandhi, Alexander Melnikov, Mia Minnes, André Nies, Eamonn O'Brien, Pavel Semukhin, and Mark Wilson. Without their careful reading of the initial drafts of this textbook, the book would not be possible in its current form. We also thank our friend Igor Polivanyi for his beautiful photograph that we used for the book cover. Finally, we thank you, our reader, for taking the time to read this book. We hope that it is educational, enjoyable and helpful for your computing and mathematical endeavors in the years to come.

Bakhadyr Khoussainov and Nodira Khoussainova

Contents

Lecture 1

Definitions, theorems, and proofs

I'm not telling you it is going to be easy.
I'm telling you it's going to be worth it.
Art Williams.

1.1 Definitions

Before writing a program, we must fully understand what the program is supposed to do. A description of the things that we would like our program to do is called the specification of the program. Correctly writing the specifications of a program requires a lot of time and intellectual effort. However, it is a necessary and helpful task. If we write our specification badly, then the corresponding program is usually hard to understand, incorrect or exhibits undesired behavior. We can use many different techniques and languages for preparing program specifications.

One way to describe the specifications is to explain them informally through conversations and documents, omitting many details. Informal specifications are a helpful first step in writing programs because they do not require the programmer to flesh out all the details of the program a priori and thus can be written quickly. That said, using only this method can have negative consequences. It often leads to an inaccurate understanding of the specifications and their different interpretations. Therefore, such an informal approach can result in incorrect programs.

A second method is to write a semi-formal description of program specifications. Usually, these specifications are written in document format with some structure and a little rigor. These types of specifications are more precise than the informal approach. However, semi-formal specifications