1-2016

Discrete Mathematics: Chapter 0, Table of Contents and Preface

Calvin Jongsma
Dordt College, calvin.jongsma@dordt.edu

Discrete Mathematics: Chapter 0, Table of Contents and Preface

Abstract

Discrete mathematics, you may be disappointed to discover, is not the sort of stuff you talk about discreetly, in secretive whispers behind closed doors. It can be discussed in public by anyone; it has lots of important everyday connections. The term discrete means separate or disconnected; its opposite is continuous. Discrete mathematics deals with non-continuous quantitative phenomena — activities as mundane as counting objects and as arcane as ana-lyzing the output of a Turing machine. It also includes familiar topics like functions, but it explores their structural algebraic features and examines them with respect to computational complexity; matters connected to continuity or differentiability are left for analytic geometry and calculus.

As a field of mathematics, discrete mathematics is both very old and very new. Counting and basic arithmetic, of course, go back to pre-historic times. The rise of discrete mathematics in recent decades, however, is a result of the increased power and ongoing development of computer technology. Concepts and results in discrete mathematics provide the mathematical foundation for computer science. In return, computer science supplies crucial stimuli and resources for important contemporary developments and applications in discrete mathematics.

Discrete mathematics is a field every computer scientist should know about, and one that today's mathematicians should be familiar with as well. This textbook is written for students in both majors. While the primary focus here is on mathematics, we will investigate topics that are crucial for computer science as well.

Keywords

axiomatic set theory, calculus, algebra, functions, computational complexity, probability, logic

Disciplines

Christianity | Computer Sciences | Mathematics

Comments

- From Discrete Mathematics: An Integrated Approach, a self-published textbook for use in Math 212

DISCRETE MATHEMATICS

An Integrated Approach

Calvin Jongsma

✠ TEX In-House Printing ✠

TABLE OF CONTENTS

✠ PREFACE

The purpose of computation is insight, not numbers.
Richard Hamming

What is Discrete Mathematics, and Who Needs It?

Discrete mathematics, you may be disappointed to discover, is not the sort of stuff you talk about discreetly, in secretive whispers behind closed doors. It can be discussed in public by anyone; it has lots of important everyday connections. The term *discrete* means *separate* or *disconnected*; its opposite is *continuous*. *Discrete mathematics* deals with non-continuous quantitative phenomena — activities as mundane as counting objects and as arcane as analyzing the output of a Turing machine. It also includes familiar topics like functions, but it explores their structural algebraic features and examines them with respect to computational complexity; matters connected to continuity or differentiability are left for analytic geometry and calculus.

As a field of mathematics, discrete mathematics is both very old and very new. Counting and basic arithmetic, of course, go back to pre-historic times. The rise of discrete mathematics in recent decades, however, is a result of the increased power and ongoing development of computer technology. Concepts and results in discrete mathematics provide the mathematical foundation for computer science. In return, computer science supplies crucial stimuli and resources for important contemporary developments and applications in discrete mathematics.

Discrete mathematics is a field every computer scientist should know about, and one that today's mathematicians should be familiar with as well. This textbook is written for students in both majors. While the primary focus here is on mathematics, we will investigate topics that are crucial for computer science as well.

I've assumed a certain level of mathematical maturity on the reader's part — being able to follow moderately-paced deductive arguments and to think somewhat abstractly — but no exceptional mathematical ability or specific college-level mathematical knowledge is presupposed, not calculus or linear algebra or any course building upon them. Gifted freshmen should be able to master the material, but this text has been used over many years for a transition course on the sophomore level to prepare students for doing more abstract upper-level mathematics, where proof becomes an essential ingredient of the learning process.

Topics Selected

The topics and subfields of discrete mathematics are a bit of a grab-bag. They include algebraic structures, algorithms, combinatorics, discrete probability, finite state machines, formal languages, graphs and networks, logic, set theory, and more. A one-semester course can't hope to cover them all, not even in an introductory way — at least not without becoming a superficial non-cohesive collage. My solution to this dilemma of riches is to weave together a select number of important elementary topics into a less-expansive but more-integrated whole.

We start by studying some *Deductive Logic* (Chapters 1 – 2). This provides a sturdy basis for understanding the structure and nature of mathematical proofs, and it gives the formal background for understanding various key developments in computer engineering. Connected to this, we also explore the proof techniques of *mathematical induction* and definition by *recursion* (Chapter 3), topics that are central to both mathematics and computer science. This unit ends by looking briefly at *Peano Arithmetic*, an elementary topic done in an advanced setting.

The second part of the text focuses on topics connected to *Set Theory* (Chapters 4 – 5). We begin by considering elementary set-theoretic operations and relations and then use that knowledge to study some significant counting principles from *Combinatorics*. We also discuss numerosity and venture a little ways into the realm of infinite sets. While this topic may seem somewhat removed from the practical concerns of computer scientists, its relevance to theoretical computational concerns becomes clear as we look at the *Halting Problem*.

Using the framework of *Set Theory*, we then proceed into an area that is more algebraic in nature (Chapters 6 – 7). After considering *functions* and *relations* more generally, we go on to investigate several algebraic structures involving them. The main focus here is *Boolean Algebra*, which nicely ties together the earlier topics of *Logic* and *Set Theory*. While this material is a bit more abstract and theoretically demanding, it also provides a satisfying conceptual basis for treating switching circuits and logic gates.

The final unit of the book, still in the writing stage, is projected to focus on parts of *Graph Theory* (Chapter 8). Like many of the earlier topics, this field deserves a full course of its own, but we will look at how the field began with the work of Euler on the Königsberg Bridge Problem and will touch on some basic types of graphs as well as several applications. As *Graph Theory* has a number of interesting well-known problems associated with its historical development, this seemed to be a good way to conclude, presenting discrete mathematics as a dynamic field of interest to practitioners and theoreticians alike.

Goals and Approach

This text was written to help you learn some significant mathematical content and theoretical applications, but just as importantly, to help you understand how to reason mathematically. Mathematics develops from certain elementary human intuitions about quantitative properties of and relations among things. These suggestive but sometimes fuzzy notions are refined and made precise through definition and deductive argumentation. Mathematics is more than proof, of course, but without proof and logical organization there is no genuine, full-fledged mathematics.

An overarching central goal of this text, therefore, is to help students decipher what proofs are and how they are constructed. Over the years, there have been divergent views about how to accomplish this. An older approach experienced by many of us in the field might be called the *sink-or-swim method*. Students were thrown into an upper-level course where the content was abstract and the methodology was proof-based and expected to pick up the appropriate mathematical habits and techniques almost by osmosis, watching a sage-on-the-stage perform clever deductive maneuvers with complex concepts and then trying it on their own in the homework, which attempts were then subjected to many red-mark comments by the instructor. Those who eventually managed to master the material were deemed worthy of continuing in the field; those who didn't survive the ordeal, well, they probably didn't have what it takes to succeed, anyway.

Today's students often encounter a more welcoming practice in a transition course. Learning how to evaluate and construct proofs can still be tough, but if you take time to study the process itself in connection with some elementary content, it becomes more comprehensible. A somewhat leisurely study of logic at the outset of this text meets this objective. Logic is often pressed into service by mathematicians engaged in technical foundational questions, resulting in its being studied along the lines advanced by Frege, Russell, and Hilbert. As our interest, however, is on its value for learning how to construct proofs, we'll adopt the natural deduction outlook first put forward by the Polish logician Jaśkowski. This approach should help you feel at home in any situation where proof is used or demanded. After studying this text, you should be able to read any well-written book on mathematics that you have the background for, because you will have the tools needed to grasp the logical organization behind the argumentation. Interestingly, while you're learning how to do proofs (methodology), you're also learning the material needed to understand more advanced mathematics and computer science (content).

Even though we value precision and deductive rigor, our results are not expressed pedantically in the Spartan definition-theorem-proof style of some mathematics texts. Results are formulated somewhat informally, but carefully, using correct terminology and valid argumentation. In addition to stating and proving mathematical results, we will talk *about* them:

explaining how they arose historically, motivating and developing an intuitive feel for them with examples, and telling why they are important as we apply them.

The intended readers of this text are students and not just their profs. If you have had one or two semesters of college-level mathematics, you should be able to follow the discussion here by paying careful attention. Of course, a mathematics text is not a novel, so you shouldn't read it the way you do a story. First try to get the overall drift of the discussion by reading the introductory and concluding material and by looking at the main headings. Then go back through the text more slowly, making sure the main points make sense to you. Minor details are occasionally left for you to fill in; use a pencil and paper to help you follow the discussion where necessary. The examples given are important for understanding the ideas and procedures, but don't expect them to prepare you for everything you will need to do. The more deeply you understand the mathematics, the better able you will be to apply it to a wide variety of situations.

Mathematics is never just a matter of filling in templates; its use and extension requires understanding that goes far beyond knowing how to change the numbers to fit new situations. As noted by the well-known computer scientist Richard Hamming in the quote above, even when the issue is simple calculation, the goal is never just getting the right answer; it is gaining insight into what you're doing.

Exercises

Like any activity worth mastering, mathematics requires genuine effort. It is not a spectator sport. You can only become competent in a mathematical topic by getting your hands dirty, by working a fair number of problems. Each section of the text has plenty of exercises for you to try. Some of them are rather routine, asking you to demonstrate familiarity with the basic ideas, results, and methods. Others are more challenging, requiring you to make connections or expand on what is present in the text. More extended explorations ask you to develop a subtopic on your own, doing some mini-research. As you work the exercises, don't get too discouraged if you don't immediately see how to solve a problem; set-backs can be as instructive as success if you persevere until you've made some progress. Hardly anyone grasps everything the first time through. As you work problems, you may need some assistance. Answers to starred exercises are provided as aids, but use these to check your work or get a hint on how to get started, not to do the work for you.

All the best with your work here. I hope you come to enjoy this material as much as I do and as my students have. If you have corrections or suggestions you'd like to share for making the textbook better, I'd be happy to hear from you. You can contact me at calvin.jongsma@dordt.edu with your response.

Acknowledgments

I'd like to thank the many students I've taught over the years who have given me feedback as this text was being developed and refined. In particular, I'd like to thank Dr. Kyle Fey, a student assistant (now a professor of mathematics) who worked with me one summer to develop hints for various exercises. I'd also like to thank Jim Bos, Dordt College's computer-savvy registrar, who in the early years helped me puzzle out some of the mysteries of Plain TeX regarding the construction of proof diagrams and tables, and I'm indebted to then-colleague Dr. Rick Faber (now an NSA mathematician) for transcribing my hand diagrams for function domains and codomains into a form that I could easily incorporate into the text.

January, 2016