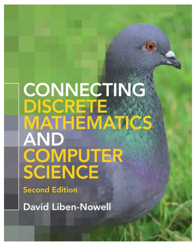
Review of [1]



Connecting Discrete Mathematics and Computer Science
by **David Liben-Nowell**

Cambridge University Press, 2022
Hardcover/eBook, 674 pages, 2nd ed., $74.99

Reviewed by **David Luginbuhl**
(`dluginbu@samford.edu`)
Dept. of Mathematics and Computer Science, Samford University

1 Overview

For those of us who teach discrete math in a computer science program, one of the challenges is making the topics relevant to computer science. I often find students struggle with this: "Why am I taking another math class, and why this one in particular?" David Liben-Nowell, in *Connecting Discrete Mathematics and Computer Science*, agrees: "Computer science students taking a class like this one sometimes don't see why this material has anything to do with computer science — particularly if you enjoy CS because you enjoy programming" (p. 2).

As the title of his text suggests, Liben-Nowell intends to tackle relevancy head-on. It covers most topics one would expect in a course in discrete math – logic, proofs, counting, and graphs and trees, to name a few – but it also weaves in content, examples, and additional features that can assist an instructor in establishing a "connection" to computer science.

2 Summary of Contents

Each chapter has several common features that are designed to help students see relevancy to computer science or reinforce the current topic:

- A *Why You Might Care* section introduces each chapter and provides hooks to computing topics that will help motivate computer science students.
- There are smaller print notes sometimes labeled *Problem-solving tip* or *Taking it further* embedded throughout the text that provide clarification or additional useful information. Some of these also allow interested students to dig deeper into the concept being covered.
- The *Computer Science Connections* at the end of each section provide immediate relevance and help keep the topic grounded in computer science applications. I could see assigning these as mini-research projects, where students explore the topic in further detail and either write a short paper or provide a brief presentation to the rest of the class (depending on class size, of course). I will highlight some of these in the *Chapter Highlights* below.

[1]©2023 David Luginbuhl

- The summaries in the last section of each chapter (*Chapter at a Glance*), organized by subsection, provide a valuable study tool and a way to query students in class to see if they understand what they've been reading.

Below I provide a summary and some thoughts on each of the twelve chapters (an intro and closing chapter, and ten devoted to various discrete math topics).

3 Chapter Highlights

Chapter 1 ***On the Point of this Book*** – I am impressed by the readability of this first chapter. In three pages, Liben-Nowell does a good job of describing the purpose of the book, previewing the features discussed above, showing the interconnection of the chapters, and ending with "three very reasonable ways to think about this book" ("mathematical foundations," "practice," and "application"). It is not long – only three pages. I would probably assign it with a brief reflection essay as homework, asking students what they anticipate most about this course having read this chapter.

Chapter 2 ***Basic Data Types*** – A popular starting point for discrete math is logic, which make sense – a solid logical foundation is necessary for thinking through all the other topics in this broad area. The challenge with logic is that it usually involves new terminology, new notation, and formality that students may be uncomfortable with.

Liben-Nowell takes a different approach: he starts with data types. The advantage is that this is familiar territory for students in a standard computer science curriculum, so it might be seen as a more gentle introduction to the world of discrete math for computer science students.

The chapter starts with scalar types (Boolean, integers, reals, and associated operations). It then moves to sets, then ordered collections (such as sequences and matrices), then to functions.

For this chapter, *Computer Science Connections* include representation of integers and reals in the machine, algorithms for computing square root, and MapReduce (for processing collections).

Chapter 3 ***Logic*** – This chapter includes sections on propositional logic, predicate logic, and nested quantifiers. There is also a section on "extensions" to propositional logic, where we are introduced to concepts like tautologies and satisfiability, as well as circuits and normal forms.

The *Computer Science Connections* in this chapter include fuzzy logic in natural language computing and short-circuit evaluation of logical connectives in programming languages.

Chapter 4 ***Proofs*** – The *Why You Might Care* section of this chapter provides a solid connection to computer science, explaining, for example, that proofs are useful for:

- demonstrating one algorithm is more efficient than another
- finding bugs in programs through program proving
- understanding the concept of computability (e.g., the halting problem)

The chapter begins with "an extended exploration of error-correcting codes" as an example for showing how to apply various types of proof techniques. This provides an appropriate application area (certainly relevant to computer science) to introduce the techniques that will be elaborated on in the remainder of the chapter. For instructors who would rather not cover the additional material on error-correcting codes, this section is fairly self-contained, and there are not many back-references to it in the rest of the chapter.

There are numerous examples in the following sections demonstrating the various proof techniques, many drawn from propositional logic. The final section on errors is very helpful and provides exercises where students have to determine whether or not a proof is valid.

Computer Science Connections for proofs include the Four-Color Theorem and some of the controversy surrounding using a computer to prove a mathematical result and "Some Famous Bugs in CS" including the Pentium bug and the Therac-25 tragedy.

Chapter 5 ***Mathematical Induction*** – Liben-Nowell begins this chapter on induction by drawing a connection to (not surprisingly) recursion. Technical sections include an introduction to both weak and strong induction.

I like his informal introduction to proof by induction: "to prove a statement $P(n)$ is true for all nonnegative integers n, we can prove that P 'starts being true' (the *base case*) and that P 'never stops being true' (the *inductive case*)" (p. 217). He uses the analogy of dominoes falling as another informal way to explain induction. This would be a terrific way to introduce the concept of induction, using a physical demonstration.

The final section is on *Recursively Defined Structures and Structural Induction.* Examples here include linked lists and binary trees, so the connection to computer science is direct and should be easily picked up by students.

Computer Science Connections for this chapter include using loop invariants for program proving and structural induction on parse trees.

Chapter 6 ***Analysis of Algorithms*** – This chapter is a comprehensive introduction to algorithmics, including a detailed discussion of big Oh, big Omega, and big Theta, analysis of searching and sorting algorithms, and recurrence relations for analyzing recursive algorithms.

Computer Science Connections for this chapter on algorithms include discussions of Moore's Law (to illustrate exponential growth) and why we count *operations* in algorithm analysis rather than wall clock time.

Chapter 7 ***Number Theory*** – With the rise in importance of cybersecurity, it is only natural to see more emphasis on number theory. Liben-Nowell addresses this at the beginning of the chapter: "more so than any other chapter of the book, the technical material in this chapter leads directly to a single absolutely crucial (and ubiquitous) modern application of computer science: *cryptography* ..." (p. 328).

The chapter provides a relatively deep dive into the theory behind the RSA encryption algorithm, but I think it is also possible to avoid some of the longer in-depth proofs in this chapter and still cover the critical concepts.

It starts with modular arithmetic, then moves to prime numbers (and what it means to be relatively prime). Following this is a discussion of multiplicative inverses in modular arithmetic. The chapter culminates in a section on cryptography, and in particular, the RSA encryption algorithm. The treatment of RSA is quite comprehensive and includes proofs of correctness.

Computer Science Connections include another example of error-correcting codes (this time with Reed-Solomon) and an exposition on the Diffie-Hellman key exchange protocol.

For this chapter, more than some of the others, it might be good to start at the end, with the *Chapter at a Glance* section. It provides a helpful TL;DR explanation that ties everything together. From there, the instructor (and students) could back up to the beginning of the chapter and work forward.

Chapter 8 ***Relations*** – This chapter is a straightforward introduction to relations, covering inverse relations, composition, properties such as reflexivity and transitivity, and equivalence relations and partial and total orders. There is a subsection on asymptotics, which will be relevant if an instructor chooses to cover the immediately preceding algorithms chapter.

Computer Science Connections for this chapter cover deterministic finite automata (where each state represents an equivalence class) and the Painter's Algorithm used for hidden-surface removal in computer graphics (as an example of building a partial order).

Chapter 9 ***Counting*** – The chapter on counting covers the usual ground: counting rules (Sum, Product, Inclusion-Exclusion, and Generalized Product), rules for transforming from a set that is easier to count (Mapping and Division), and combinations and permutations.

Computer Science Connections in this chapter include switching from IP addresses to IPv6 addresses (how many more addresses are made possible by moving from a 32-bit sequence to a 128-bit sequence?) and the Enigma machine (alphabetic permutations).

Chapter 10 ***Probability*** – This is a detailed treatment of probability and randomness. The *Why You Might Care* section describes some relatable connections to computer science: randomized algorithms for solving certain kinds of problems, hashing, and creating realism in graphics with randomness. In fact, Liben-Nowell uses hashing as a "Running Example" (he provides a brief introduction to hashing in the first section, and comes back to it throughout the rest of chapter).

The next section introduces basic concepts and definitions (outcomes, probability, events) and provides some standard examples (cards, coin flips, words on a page). The use of tree diagrams to explain probability is helpful here, with examples that involve hashing and the Monty Hall problem.

Subsequent sections cover independence, conditional probability, random variables, and expectation. These sections are detailed and quite thorough. If students in your program are not going to be otherwise exposed to probability, this chapter would provide a good introduction to the topic.

One *Computer Science Connection* in this chapter that I think is particularly motivating for computer science students involves using randomness to fuzz data for privacy protection.

Chapter 11 ***Graphs and Trees*** – The last technical chapter of the text introduces graph theory, covering directed and undirected graphs, connectivity, isomorphisms, trees traversals, and Dijsktra's and Kruskal's algorithms. There are plenty of relevant examples throughout the chapter.

I thought the discussion of tree traversals was especially good, and the examples that demonstrate the difference between pre-, post-, and inorder traversals should be very easy for students to follow.

The *Computer Science Connections* in this section are particularly apropos, including graph drawing in computer graphics, the way some programming languages use reachability of nodes for garbage collection algorithms, and the use of random walks in page rank algorithms.

Chapter 12 ***Looking Forward*** – This final chapter touches on the impact of computer science to society and the importance of understanding how our choices as computing professionals can affect those around us. One way to effectively make use of this small (2.5 pages) concluding chapter is for an instructor to look for links to societal impact throughout the book (some are provided in this chapter) and emphasize them throughout the course so they can be reinforced here at the end.

4 Summary

Liben-Nowell delivers on his promise to "Connect Discrete Math with Computer Science." There are plenty of computing-related examples peppered throughout each chapter, and the *Computer Science Connections* at the end of each section make this even more explicit. Starting out with data types helps establish computer science relevancy as well.

Beyond that, the summaries and key terms provided at the end of each chapter will be useful for students in assuring they know what they need to understand.

Each section ends with many exercises at various levels of difficulty.

As with most discrete math texts, it will be necessary to pick and choose, both in terms of subjects ("Do I want to cover Analysis of Algorithms in this course?") and depth ("Do students need to see a detailed proof of this particular theorem?"). One note about depth: as I have indicated above, the book's treatment of some subjects (number theory, for example) is quite extensive and detailed. Students who have already taken at least a couple of math courses will have an easier time engaging the content in those sections. Still, there is plenty to select from in a book this large and comprehensive. It should definitely be in the mix of candidates if you are looking for a new text in this area.