The Book Review Column[1]

by William Gasarch
Department of Computer Science
University of Maryland at College Park
College Park, MD, 20742
email: gasarch@cs.umd.edu

In this column we review the following books.

1. Joint Review of

   (a) **The Honor Class: Hilbert's Problems and Their Solver** by Ben Yandell.

   (b) **Mathematical Developments arising from Hilbert's Problems** edited by Felix Browder. (Note this book is from 1974).

   Reviewed by William Gasarch. In 1900 Hilbert posed 23 problems for Mathematicians to work on over the next century. How much progress was made on them? Who made the progress and what were their lives like? Given the years they lived, both World Wars and the Cold War had a great effect. The first book is about the people, the second is about the math.

2. **Variable-length Codes for Data Compression** by David Salomon. Reviewed by Farhan Nasim. Say you wanted to code each letter of the English alphabet into a sequence of bits so that you could code any word. To make the code efficient you might code frequent letters like 'e' with very few bits and infrequent letters like 'z' with many bits. There are several ways to use Variable-length codes to do data compression. This book discusses four of them.

3. **History of Mathematics - Highways and Byways** by Amy Dahan-Dalmedico and Jeanne Peiffer (Translated by Sanford Segal). This is a book on the History of Math that organizes the subject by topic rather than chronologically. For example, the chapter on limits from Zeno's paradox to Dedekind. Reviewed by S.V.Nagaraj.

4. **Identity-Based Encryption** by Sanjit Chattarjee and Palash Sarkar. Reviewed by Jon Katz. Identity-based encryption (IBE) aims to simplify key management. At a high level, in an IBE scheme a user's identity is their public key; more to the point, a receiver's identity (along with some global information) is all that is needed in order to encrypt a message for that receiver. More formally... read the review and then the book to find out.

5. **Resources for Teaching Discrete Mathematics** edited by Brian Hopkins. Reviewed by Myriam Abramson. The book consists of three parts. The first and largest is a collection of classroom projects (19). The second part is about historical milestones in discrete math (12). The third and smallest part contains notable articles in discrete math (5). All projects and articles reflect typical classroom experiences. The intend is to be, as the title says, resources for teaching discrete mathematics.

6. **Proofs and Algorithms** by Gilles Dowek (translation by Maribel Fernandez). Reviewed by Michaël Cadilhac. This book contains the lecture notes of a course given by the author,

[1]

and focuses on logic (with a strong emphasis on proofs and a smaller one on model theory), computability (with a presentation of recursive functions, rewriting systems, lambda calculus, and Turing machines), and the links between the two.

7. **Introduction to Computational Proteomics** by Golan Yona. Reviewed by Dimitris Papamichail. Computational proteomics is a term that generally describes the use of computational methods to analyze proteins, primarily to determine their structure, dynamics and function. While called an *Introduction*, this book goes fairly deep into the field.

8. **Computability and Complexity Theory** by Steven Homer and Alan L. Selman. Reviewed by Jeffrey Shallit. This is a text on computability and complexity which is a revised version of a book from 2001.

9. **Quantum Computing Since Democritus** by Scott Aaronson. Review by Frederic Green. This is a book about Quantum Computing; however, it covers much more: Complexity theory, Computability theory, and even some philosophy.

10. **Algorithmic Puzzles** by Anany Levitin and Maria Levitin. This is a book of Algorithmic Puzzles. There is a great variety of them and answers are included!

BOOKS I NEED REVIEWED FOR SIGACT NEWS COLUMN

Algorithms

1. *Greedy Approximation* by Temlyakov
2. *Algorithmics of matching under preferences* By Manlove.
3. *Approximation algorithms and semidefinite programming* by Gartner and Matousek.
4. *Data clustering: Algorithms and Applications* Edited by Aggarawal and Reddy.
5. *Modern Computer Algebra* by Gathen and Gerhard (Third edition-2013).
6. *Algorithms Unplugged.* Edited by Vocking et. al.
7. *The LLL Algorithm.* Edited by Nguyen and Vallee.
8. *Basic Phylogenetic Combinatorics* by Dress, Huber, Koolen, Mouton, Spillner
9. *A guide to experimental algorithmics* by Mcgeoch.

Logic

1. *Proof Analysis: A Contribution to Hilbert's Last Problem* by Negri and Von Plato.
2. *Software Abstractions: Logic, Language, and Analysis* by Daniel Jackson.
3. *Introduction to reversible computing* by Perumalla.

Cryptography and Coding Theory

1. *Understanding Cryptography: A Textbook for Students and Practitioners* by Paar and Pelzl.
2. *The Block Cipher Companion* by Knudsen and Robshaw.
3. *Algebraic Shift Register Sequences* by Goresky and Klapper.

Recreational Mathematics and Computer Science

1. *Towers of Hanoi— Myths and Maths* by Hinz, Klavzar, Milutinovic, Petr.
2. *Visions of Infinity* By Ian Stewart.
3. *Games and Mathematics* by David Wells.

Misc

1. *Selected Papers on Computer Languages* by Donald Knuth.
2. *Handbook of finite fields* by Mullen and Panario.

Joint Review[2] of

The Honor Class: Hilbert's Problems and Their Solver
Author: Ben Yandell
Publisher: A.K Peters, 2002, $25.00, 486 pages

and

Mathematical Developments arising from Hilbert's Problems
Edited by Felix Browder
Publisher: American Math Society, 1974, $40.00
628 pages in Two Volumes
Vol 28 of Proceedings of Symposia in Pure Mathematics

Reviewer: William Gasarch gasarch@cs.umd.edu

1 Overview

In the year 1900 Hilbert posed 23 problems for mathematicians to work on over the next 100 years. These problems have shaped mathematics and (to some extent) guided what mathematicians work on. To solve one is, to paraphrase Hermann Weyl, to enter *The Honor Class*.

In this review I first say a little about each of the two books and then do a joint summary by saying what each one said about some of the problems. I chose to talk about the problems that I think will interest my readers.

1.1 The Honor Class

The book *The Honor Class* is about the people who solved the problems and the times they lived in. There is some mathematics in it as well—enough so that readers of this column can understand the problems and why they were important. A layperson can also get something out of the mathematics, though that varies from problem to problem. This book was written in 2002; however, for the most part, it is not dated.

Why do we study history? Some read history to get a better idea of how we got to where we are now. When we see the problems people worked on, and why, we are informed as to (1) why we are working on similar or different problems, (2) why we are or are not using certain approaches, and (3) what our expectations are and perhaps should be.

I heard a historian say that reading history is like going to a different country— when you go back to your own country you see things in a different light since you've seen an alternative way of doing things. This applies to history of math in two ways: (1) the math itself— seeing how they thought of math and how we do is enlightening. (2) how the culture of research has changed— Jews and Women were often banned from universities; people communicated by postal mail (you can ask your grandparents what that is); people would argue furiously about the degree of rigor needed in a proofs; and nonconstructive proofs were considered controversial. We do indeed live in different times. I will go out on a limb and say MUCH better times.

The Honor Class gives context for the math presented and presents how math was done in those days. There is also a good deal of history in this book. Not history of the sort *The Hitler-Stalin*

pact was signed in April 23, 1939 and had drastic consequences, but history of the sort that tells what ordinary citizens (largely mathematicians) were doing in those times. World War I, World War II, and the Cold war permeate the entire book. This book *is not* about the moral dilemmas of what a mathematician (or anybody) should do when working in (say) Nazi Germany (for that see *Mathematics under the Nazis* by Sanford Segal). This book *is* about the particular people involved with Hilbert's Problems- their math and their lives. The moral questions arise naturally.

The Honor Class is organized into eight sections, each of which have chapters. The book is *not* in order of the problems. Instead it groups the problems into natural categories such as *The Foundation Problems.*

1.2 Mathematical Developments arising from Hilbert's Problems

The book *Mathematical Developments arising from Hilbert's Problems* is the proceedings of a 1974 conference on Hilbert's Problems. There is one article about each problem except (1) there is no article on Problem 3 (showing that the volume formula for triangular pyramids requires continuous methods), and (2) there are two articles on Problem 8 (Riemann Hypothesis). The articles are on the problems and what happened *after* they were solved (if they were) or what mathematics has been created to *try to solve it* if it hasn't. The former goal is very interesting because it reminds us that once a problem is solved it is not the end of the story. This book was written in 1974; however, while I am sure some of it dated, much of it is not. There are two reasons for this: (1) Math moves rather slowly, and (2) the material about the problems, how they were solved, what happens next is still interesting.

2 Summary of Both Books

I abbreviate *Hilbert's nth Problem* by Hn, *The Honor Class* by *Honor*, and *Mathematical Developments arising from Hilbert's Problems* by *Math Dev.*

H1: The Continuum Hypothesis:

The Continuum Hypothesis (CH) is the statement that every infinite subset of the reals is of cardinality either that of N or that of R. Hilbert wanted a proof of CH though one suspects he would have been happy (though surprised) with a proof of $\neg CH$. He certainly thought it was either true or false. However, CH was shown independent of Set Theory by Godel and Cohen. This would have shocked Hilbert.

In *Honor* the chapter on Cantor reveals just what an advance Cantor made. In the 1800's mathematicians were grappling with the basic definitions of sets and functions. At one time only functions that were differentiable were considered functions. It is hard for us, 21st century people, to grasp that *function* meant *function that comes up in nature.* Dirichlet is often credited with the modern definition of function (though this is not clear). Cantor's set theory allowed virtually ANY collection of objects to be a set. This was revolutionary. Many mathematicians objected since they only wanted *well behaved sets* to be considered. Poincare famously said *Set theory is a disease from which mathematics will one day recover.* Cantor had a hard time forming a counterargument since he had various mental problems. Even so, Cantor's viewpoint won out and is now accepted today without controversy.

What did the Catholic Church think of Cantor's work? If they opposed it I am sure we would all know that. However, Cantor contacted them and they decided that it was good philosophy and

compatible with the Church's position. Not as interesting a story as Galileo (allegedly) saying under his breath "and yet it moves" but worth knowing.

The chapter of *Honor* on H1 discusses Godel's unhappy life. He was born in Germany and thought of himself as German; however, he had Jewish friends and looked Jewish; hence he was forced to leave the country. This was difficult as there was a blockade at the time, though he managed to get to Princeton. He also had mental problems which may have lead to his death (at 72 years old) of starvation.

In *Math Dev* Donald Martin has an article that talks about how CH may still be resolved some day via large cardinal axioms or some version of the Axiom of Determinacy (Donald Martin had recently proven AD for Borel Sets). AD does imply the version of CH stated here though this is sometimes called the weak version (the strong version is that the first uncountable ordinal is the same size as R). The article seems dated in a charming way since (I think) most set theorists nowadays think of CH as not having an answer.

For a modern take on why set theorists believe the axioms they do, see Penelope Maddy's articles *Believing the Axioms I, II* in *Journal of Symbolic Logic* Vol 53, No 2, 481–511, 1988, Vol 53, No 3, 736–764, 1988. Or go to the her webpage to find a free copy that is better since there are corrections.

H2: Prove that Arithmetic is Consistent. This was shown to be impossible by Godel.

The chapter in *Honor* on Godel also talks about Hilbert and the Second problem before getting to Godel. We see that an objection to Hilbert's axiomatic approach was that Hilbert wanted (or perhaps this was a caricature) all axiom systems to be considered equally worth working in. We DO have some of this attitude now with people asking *can I prove this without the Axiom of Choice?* rather than saying *is this TRUE.*

In *Math Dev* Georg Kreisel discusses how close one can come to realizing Hilbert's dream. In particular he discusses which mathematical theories are decidable, which ones are not, and how to tell the difference. He looks at real closed fields, certain geometries that are decidable, and Presburger arithmetic. The article also points out that what Hilbert asked for needs to be better refined to be understood.

H3: Show that the following Theorem requires continuous methods: *Two triangular pyramids of equal height and base have the same volume.* Proven by Max Dehn.

If you go by Google Hits this is the least well known of Hilbert's problems. This may be because it was solved very shortly before it was posed. (Recall that communication was slower in those days.) There is no chapter on it in *Math Dev.*

Hilbert (and most people) liked to have theorems in field X proved using tools just from field X. The theorem stated in my description of H3 was an example of this issue: a theorem in solid geometry that uses calculus to prove it. In Hilbert's day it was thought that such a proof was needed. And they were right.

Honor has a chapter about Max Dehn's fascinating life. By all accounts he was brilliant. He was a Jew in Germany during the Nazi era and had to flee. He managed to get out and teach at *Black Mountain College* which was a very odd place in that it ran by consensus of the faculty and the students. It no longer exists.

When one reads accounts of people who fled from Nazi Germany one tends to only read of those that escaped. Many did not. As a reminder of this *Honor* notes that one of Dehn's colleagues, Paul Epstein, also Jewish, killed himself rather than be taken by the Gestapo.

H4: Explore Different Geometries that use Different Notions of Distance.

This problem is more of a research plan than a problem. The Wikipedia entry on H4 said that the problem is too vague to be ever be considered solved. Pogorelov wrote a book in 1979 on the problem which formulates it in rigorous terms and solves it, using the work of Busemann. *Honor* credits the solution to *Busemann, Pogorelov, ...* (I do not know what the ... means.)

In *Math Dev* Busemann writes about how the problem was harder and less well defined than Hilbert intended. To quote: *The Fourth Problem concerns the Geometries in which ordinary lines, i.e., lines of an n-dimensional (real) projective space P^n or pieces of them are the shortest curves of geodesics. Specifically, Hilbert asks for the construction of all of these metrics and the study of the individual geometries. It is clear from Hilbert's comments that he was not aware of the immense number of these metrics, so that the second part of the problem is not a well posed question and has inevitably been replaced by the investigation of special, or special classes, of interesting geometries.* The article then goes on to rephrase what Hilbert should have asked and connect it to the calculus of variations.

In my first draft of this review I had a comment like *H4 began as a question on foundations but then lead to some real math.* However, this is a very 21st century view. The distinction between *foundations* and *real math* is not appropriate for that era and perhaps is sharper than it should be in ours.

H6: Axiomitize Physics Good luck.

This problem is more of a research plan than a problem. The chapter in *Honor* about this problem is not about people. Its about the on-again off-again marriage between Mathematics and Physics. Mathematics and Physics were joined at the hip before the 20th century in ways that are hard for us to imagine now. The article also claims that they are now reuniting. This may be; however, it will never be what it once was. As a major example or how closely they were connected, as noted in this review regarding H1, at one time *function* meant *function that occurs in nature.* As a minor example of how closely they were connected note that the Putnam exam used to have problems on Physics but now they are gone, replaced by problems on combinatorics [3]

Hilbert himself mentioned probability and the kinetic theory of gases as already having a rigorous treatment. Even though probability is not a branch of physics his point was that a previously non-rigorous study had been made rigorous. In *Math Dev* Wightman writes an article that has two parts. First he briefly surveys Hilbert's own work on gases, radiation, relativity, and quantum mechanics. Then he describes, at length, recent (circa 1974) attempts to axiomitize quantum mechanics and quantum field theory. This is the longest chapter by far in the book– around 90 pages. It is not a light read.

H7: Show that if a is algebraic and b is algebraic and irrational then a^b is transcendental. (For example show that $2^{\sqrt{2}}$ is transcendental.) The general theorem was shown by Gelfond, Schnider, and Siegel.

Siegel lead an interesting life. He was German. He avoided serving in WW I by pretending to be mentally ill. He fled Nazi Germany shortly before WW II but after the war went back. He truly wanted to revive Mathematics in Germany. He worked in number theory but also in mechanics. This type of well roundedness is rare today. He was also a member of the Frankfurt Historical

[3] See `http://blog.computationalcomplexity.org/2009/03/putnam-exam-some-thoughts.html` for a full study of this.

Math Society which read the original papers in the original languages and had an attitude of not publishing too much. (Dehn was also a member.)

Schneider was a student of Siegel's. He joined the Germany Army in WW II even though he disagreed with the Nazis. He hoped this would get him better treatment from them but it did not—his Habilitation[4] was rejected without being read. After the war his fortunes improved considerably.

Gelfond was Russian and had to deal with the oddities of the Soviet system. Decisions were capricious and arbitrary (e.g., political) as to who got promoted and who got jobs.

In *Math Dev* Tijdeman gives a survey of more recent work that extends Gelfond's technique. We give an example of different types of theorems.

1. Let $\alpha_1, \ldots, \alpha_n$ be nonzero algebraic numbers. If $\log(\alpha_1), \ldots, \log(\alpha_n)$ are linearly ind. over Q then they are linearly ind. over A (the algebraic numbers).

2. Let α, β be algebraic with $\alpha \neq 0, 1$. Assume that β has degree at least $2^n - 1$ with $n \geq 2$. At least n of the following numbers are algebraically independent: $\{\alpha^\beta, \alpha^{\beta^2}, \ldots, \alpha^{\beta^N}\}$ where $N = 2^{n+1} - 4$.

3. Let $k \in \mathsf{Z}$. Every solution to $y^2 = x^3 + k$ satisfies $|x|, |y| \leq \exp((10^{10}|k|)^{10^4})$. This can be seen as solving a subcase of H10 in the positive direction.

Both books tell the following story: Hilbert in 1920 predicted that (1) he would live to see the Riemann hypothesis solved, (2) some younger people in the audience would live to see Fermat's last theorem proved, but (3) H7 would not be seen by anybody in the audience. In reality (1) H7 was solved in 1934 (in Hilbert's lifetime), (2) Fermat's last theorem was solved in 1994 (if there was a 20 year old in the audience then they would have to live to 94 to see that there was a proof), and (3) the Riemann hypothesis is still open.

H10: Give an algorithm that will, given a polynomial in $Z[x_1, \ldots, x_n]$, determines if it has an integer solution. This was meant to be a problem in Number Theory but it is undecidable so it became a problem in logic. While this may have surprised Hilbert note that, in Hilbert's address, he noted the following which I quote in full: *Occasionally it happens that we seek the solution under insufficient hypotheses or in an incorrect sense, and for this reason do not succeed. The problem then arises: to show the impossibility of the solution under the given hypotheses, or in the sense contemplated.*

Davis-Putnam-Robinson(1964) made great strides towards showing the problems was undecidable. Matiyasevich(1970) completed the proof. The results is often attributed to all four. After the result of Davis-Putnam-Robinson, Davis predicted that the problem would be solved by a young Russian mathematician. And indeed he was right.

One (likely unintentional) feature of this chapter in *Honor* is a focus on the role of randomness in research. In 1944 Post told Davis that he thought Hilbert's 10th problem cried out for an unsolvability proof and this began Davis's lifelong obsession with the problem. A more drastic example is in the next paragraph.

In their attempt to code Turing machines into polynomials, Robinson, Davis, and Putnam had been reading Number Theory books full of obscure facts. Matiyasevich had just read the third

[4] In Gerrmany, then and now, an academic often writes a Thesis 4 to 10 years after their PhD. This is needed for later employment.

edition of Nikolia Vorob'ev's *Fibonacci Numbers*. In that book, published in 1969, there was a new result about the divisibility of Fibonacci numbers. Matiyasevich used it to prove that if F_n^2 divides F_m then F_n divides m. This allowed him to solve Hilbert's 10th problem (building on work of Davis-Putnam-Robinson). Robinson had read that book, but *not the third edition!*

This chapter also discusses Julia Robinson's difficulties being a female mathematician and Yuri Matiyasevich experience in Communist Russia. The lives of Davis and Putnam are of course also discussed. I've heard the following said though its not in the book: people mostly thought H10 was solvable and would take Number Theory— hence it took outsiders— Julia Robinson (a female), Hillary Putnam (a Philosopher), Martin Davis(a Logician)— to look at the problem in a different way.

The solution (or perhaps anti-solution) to H10 actually showed that for any computably enumerable set [5] A there is a polynomial $p(x_1, \ldots, x_n, x)$ such that

$$A = \{a \mid (\exists a_1, \ldots, a_n)[p(a_1, \ldots, a_n, a) = 0].$$

Hence many familiar sets such as the primes, at least theoretically, have such a representation. The proof is constructive so one can actually find such polynomials; however, it is not clear if one wants to.

Honor notes that we can now code many problems, including the Riemann Hypothesis, into polynomials. The book goes on to claim that (to paraphrase) *if H10 was solvable then we could solve all of these problems.* If this remark was made in 1965 then it would reflect what people thought about complexity at the time (they didn't). But this book was written in 2002 when P, NP, and issues of complexity are well known. Even if H10 was solvable it may be that the algorithm to solve it takes so long to run that it would not help us to solve anything. Many known decidable theories (e.g., S1S and S2S) are also know to be quite complex. A decision procedure for them is not practical.

In *Math Dev* Davis-Matiyasevich-Robinson have an article that explores the implications of the solution to H10. They give an explicit polynomial for the primes (due to Jones) and note that it is NOW possible to prove a number is prime with a bounded number of operations (this had been previously unknown). They also look at questions of what is the max number of variables needed for any computably enumerable set (13 when *Math Dev* was written, 9 now) and what is the max degree needed (4, though with more variables). These bounds are not known to be tight. Further implications of the solution to H10 can be found in the book *Hilbert's 10th problem* by Matiyasevich.

3 Other Parts of the Books

Both books include Hilbert's paper where he proposed the problems. *Honor* has some historical context about Hilbert. *Math Dev* has a long article on current open problems suggested by the contributors of the volume. It was written in 1974 so P vs NP is not on the list.

Honor has a very short summary of which problems have been solved. Since some of the problems are research programs this can be somewhat subjective. Nevertheless, it is a good guide. In his estimation the following are the only ones that have not been solved: (1) H6, axiomitize physics (2) H8, the Riemann Hypothesis, and (3) H16, which informally asks for descriptions of algebraic curves.

[5] Computable enumerable sets were once called Recursively Enumerable sets. Ask your grandparents why.

In the year 2000 The Clay institute posed seven problems (called *The Millennium problems*) and offered $1,000,000 each for them. H8 is one of them. The Poincare Conjecture was one of them but has been solved. Since there is nobody of the stature of Hilbert nowadays this seems to be the only way to get problems out there— money.

Neither book mentions these problems. Since *Math Dev* was written in 1974 that is quite reasonable. While *Honor* was published in 2002 it was likely written before 2000.

4 Opinion

Honor is an excellent book with many tidbits of information that brings these people alive. Mathematicians may think in an abstract world but they live in the real one. This book brings that home by examining the lives they lead. The chapters also do a pretty good job of describing the mathematics involved on a level that a layperson can understand.

Math Dev is a hard read but a rewarding one. For many of Hilbert's problems you can get a sense of the work done on them after they were solved.

Review of[6]
Variable-length Codes for Data Compression
by David Salomon
Springer-Verlag, 2007
208 pages, Paperback, $60.00

Reviewer: Farhan Nasim f.nasim01@gmail.com, Sylhet, Bangladesh

1 Introduction

Any digital data—be it text, image, audio, video, or others—consists of a set of symbols called an *alphabet*. Each symbol of the alphabet is represented in the computer as a distinct pattern of *bits* (0 and 1). In the simplest representations—ASCII codes for example—each symbol is represented by the same number of bits. In practice, however, not all symbols occur with the same frequency. It is easy to observe that assigning shorter length codes to more frequent symbols and the longer codes to less frequent symbols leads to a save on space; this technique is called *data compression*. Any data compression scheme (also called *coding scheme* or simply *code*) consists of an *encoder* and a *decoder*. An encoder takes a symbol represented in bits (called *dataword*) as input and outputs a possibly shorter bit representation (called *codeword*) for that symbol. A decoder, on the other hand, does just the opposite; it takes codewords as input and outputs datawords. Any compressed data are, in fact, long sequence of concatenated codewords. Any such sequence must be *uniquely decodable* or *uniquely decipherable* by the decoder; that means the decoder must produce only one sequence of datawords for a compressed data. During a transmission, one or some bits of the data may get corrupted, hence a practical coding scheme must also have *error detection* and *error correction* capabilities.

All modern digital data storage or communication systems use data compression in some form or other. As this is a well studied area, the ideas different coding schemes are based on varies widely from number theory to statistics or from Fourier transform to finite automata. They can be, however, divided into four broad categories based on the number of bits they take as input and the number of bits they output: (1) block-to-block , (2) block-to-variable, (3)variable-to-block, and (4) variable-to-variable codes.

The author of the book *Variable-length Codes for Data Compression*, David Salomon, is an authority in data compression. He says in the colophon of the book that the idea of the book came to him while he was writing his authoritative volume *Data Compression: The Complete Reference*; he found that book doesn't say much about variable-length codes. So he decided to write the book to survey the various variable-length coding schemes. Although the book describes all four types of coding stated earlier, but it focuses principally on the second type, which, the author says, is the most important among them.

2 Summary

The small book consists of only three chapters.

[6]

2.1 Chapter 1

This chapter aims to provide the necessary background to understand variable-length codes and to discuss several aspects of variable-length coding. It begins by stating the terminologies to be used throughout the text. Mathematical concepts such as information theory, Kraft-McMillan inequality, etc., are also reviewed in this chapter. Moreover it describes different approaches to variable-length coding such as prefix codes, universal codes, phased in codes, self-delimiting codes, etc. This chapter concludes with a long discussion on various aspects of Huffman coding, a very popular prefix coding method.

2.2 Chapter 2

In this chapter, the longest in the book, the author discusses more advanced codes. He starts with two simple coding methods for compressing integers: start-step-stop and start/stop codes.

Then the author treats codes that contain their own lengths. First of them is the Elias code with its variants. Then comes several other codes of this category: Lavenstein codes, Even-Rodeh code, punctured Elias code, ternary comma code, Stout code, Boldi-Vigna code, and Yamamato's recursive code.

Two closely related coding techniques, called suffix codes and flag codes, and their examples are then discussed. In suffix codes, end of the code is indicated by a special reserved pattern, and the pattern must not appear in the codewords. The two suffix codes described are: Taboo code and Wang's flag code. Flag codes also end with a fixed pattern, but—unlike suffix codes—the pattern may appear in the codewords. The only flag code the author discusses is Yamamato's flag code.

After a quick review of number bases, the author moves to the class of codes that arise from number theoretic observations. First of them is Fibonacci codes, which arise from the famous Fibonacci sequence. Another interesting code which arises from the famous Goldbach conjecture of twin primes—and called Goldbach code—is discussed then. The widely used Golomb code is discussed after that. Rice code which addresses some limitations of Golomb code is discussed then.

The chapter ends with a short discussion on codes ending with "1", which are important in some applications.

2.3 Chapter 3

The coding schemes presented in the first two chapters focused exclusively on efficient compression. But in order to make them robust for communication those coding must be coupled with error detection and correction mechanisms. This chapter describes some of the methods used to accomplish this purpose.

At first the author reviews the mathematical background of error correction such as Hamming distance, free distance, etc. Then he discusses synchronous prefix codewords and its application in Huffman coding. Synchronous prefix coding contains some synchronizing codewords that indicates an error and the decoder synchronizes its operation after it encounters one such code. The next method he discusses is bidirectional codes; data encoded in this coding can be decoded both in forward and reverse direction. Then comes symmetric codes, in which each codewords reads the same both in forward and backward direction. The last technique covered is variable-length error-correcting (VLEC) codes and three methods of constructing them: alpha-prompt codes, VLEC tree, and VLEC trellis.

3 Opinion

The author followed quite an informal style to write the book; this makes the book more readable than a usual technical book. Mathematical backgrounds required to understand any code or any class of codes are reviewed before describing that. Some of the codes are accompanied by *Mathematica* codes that implement them; readers who want to do experiments with the code will find them helpful. The book's fairly rich bibliography will help those who want to study the topic further. Besides all of these, the biographical sketches of people who made important contributions to variable-length codes, historical remarks, quotations, etc., make the book more enjoyable.

The book can serve well as a gentle introduction to the topic as well as a supplement to advanced books on the topic. Any advanced computer science undergraduate or students of other mathematical disciplines with introductory knowledge of computer and data communication can self-study the book.

Review of[7]
History of Mathematics - Highways and Byways
by Amy Dahan-Dalmedico and Jeanne Peiffer
Translated by Sanford Segal
MAA, 2010
341 pages, Hardcover,$67.50

Reviewer: S.V.Nagaraj svnagaraj@acm.org, RMK Engg. College, India

1 Introduction

Mathematics is a fascinating branch of science. It is often considered as the queen of the sciences. It has wide application to engineering, medicine as well as the social sciences. Mathematics is ever growing since mathematical discoveries are being made continuously even to the present day. Thus, the history of mathematics is indeed very engrossing. This book on the history of mathematics is an English translation of a book originally published in French in the year 1986. The authors have been associated with the French scientific institution CNRS. The translation from French to English has been done by Sanford Segal - a mathematician.

2 Summary

The book is made up of eight chapters. The first chapter looks at mathematical contributions during various time periods starting from the early ages. It begins with work done during the ancient civilizations. This includes the Babylonian, Egyptian, and Greek civilizations. The contributions of the Arab civilization and the power of the Church are highlighted. The work of Leonardo of Pisa, Copernicus, and Kepler are focused upon. The mathematical achievements of the seventeenth, eighteenth, and nineteenth centuries are spotlit.

The second chapter looks at the contributions of Greek mathematicians. The Ionian school of mathematics, the Pythagorean school, the Eleatic school, Plato's academy, the Euclidean school, the school of Apollonius, and the Alexandrian school are some of the famous Greek schools of thought which are discussed in this chapter.

The third chapter looks at the origins of algebra. Linear and quadratic equations, Euclidean algebra, the work of Diophantus, Arab mathematics, the solution of cubic equations, the contributions of Italians, and the work of Fermat and Abel are described. There is an appendix on constructions with straight edge and compass.

The fourth chapter looks at the origins of geometry. Greek geometry, Arab contributions, projective geometry, analytic geometry, descriptive geometry, and non-Euclidean geometries are talked about.

The fifth chapter focuses on the concept of limits. Zeno's paradoxes, Arab contributions, the work of Stevin, Valerio, Kepler, Cavalieri, Roberval, Fermat, Pascal, Newton, Leibniz, Euler, D'Alembert, Lagrange, Weierstrass, Cantor, and Dedekind towards the development of the concept of limits is described in sufficient detail.

The sixth chapter focuses on functions and analysis. Contributions from ancient times, the mathematical schools of Oxford and Paris, the work of Descartes, Newton, Leibniz, Euler, Dirichlet, Riemann, Cauchy, Weierstrass, Lebesgue and several others are studied in this chapter.

The seventh chapter focuses on complex numbers which are considered as being at the crossroads of algebra and geometry. The work of Gauss, Hamilton, Cauchy and a few others are described.

The eighth chapter which is the last chapter of the book looks at the emergence of algebraic structures. The work of Gauss, Galois, Cauchy, Boole, Cayley, Grassman, Camille Jordan, Kummer, Dedekind, Kronecker, and others are discussed.

3 Opinion

The book provides an interesting insight into the history of mathematics. The book will be of interest to all those keen to know the history of mathematics. It will be absorbing to mathematicians as well as those who are novices to mathematics. The book may be used for teaching courses related to the history of mathematics. It appears that the translator has done a good job in translating from the original French version. Mathematicians who are fluent in both French and English will be able to judge this perfectly. I don't belong to this category since I am not conversant with French. There are many books which describe the history of mathematics. However, many give chronological accounts without much focus on the growth of a particular branch of mathematics. This book does not follow that approach. It traces the growth of various branches of mathematics.

In the previous paragraph, some of the merits of the book were highlighted. On the other hand, the book has some demerits to which I draw attention. Mathematics as a subject is very vast and therefore a book of this size running to just over three hundred pages can only give a snapshot of the developments that have taken place in a few of its branches. It is unfortunate that the book does not have any focus on the mathematical developments in the twentieth century. The book fails to discuss the contributions made by pioneering Indian, Chinese, American, Hungarian, and Russian mathematicians. The fonts used in the book are small enough to make reading difficult. Many details are given in small fonts. The translator has listed as references only those works that are in English, thereby depriving readers the chance to look at other reference works mentioned in the original French edition. The bibliography lists just ten references. This is very inadequate for those eager to know more. The topic index does not seem satisfactory either. Despite these shortcomings, I recommend this book for an insight into the history of mathematics.

Review[8] of
Identity-Based Encryption
by Sanjit Chattarjee and Palash Sarkar
Springer, 2011
192 pages, Hardcover, $80.00

Reviewer: Jon Katz *jkatz@cs.umd.edu*

1 Identity-Based Encryption

Public-key encryption allows two parties, a *sender* and a *receiver*, to communicate privately without having previously shared any information. When using a "standard" public-key encryption scheme, the receiver begins by generating a pair of keys: the public key pk along with the corresponding secret key sk. The public key is transmitted (over a public channel!) to the sender, while the private key is stored secretly by the receiver. The sender can encrypt a message m using the public key obtained from the receiver, and then send the resulting ciphertext $\mathsf{Enc}_{pk}(m)$ to the receiver over the same public channel. Finally, the receiver recovers the message m by decrypting the ciphertext using its secret key. If the encryption scheme is secure, an eavesdropper who watches all the communication between the sender and receiver (and so sees both the public key and the ciphertext) cannot deduce anything whatsoever about m. (The fact that public-key encryption is possible is not at all obvious, and its invention in the 1970s was quite surprising.)

To use public-key encryption in practice, say, for sending an encrypted email to a colleague, we see that the sender must first obtain a (legitimate) copy of the intended receiver's public key. Since the receiver may not be on-line when the sender wants to encrypt a message, there must be some additional mechanism that enables the sender to do this. While there are many possibilities (e.g., the sender can look for the public key on the receiver's webpage, or search for the receiver's public key in some public directory), they all add an extra step to the process of sending an email. Moreover, the sender also needs to verify the validity of the public key it obtains before using it to encrypt, adding yet another step.

Identity-based encryption (IBE) aims to do away with the above and thus simplify key management. At a high level, in an IBE scheme a user's identity is their public key; more to the point, a receiver's identity (along with some global information) is all that is needed in order to encrypt a message for that receiver. A bit more formally: a trusted authority first generates some master parameters MPK that will be made public, along with an associated master secret key MSK. A user with identity id can, after authenticating its identity to the authority, obtain a personal secret key sk_{id} from the authority. Any sender who knows MPK and the intended receiver's identity id can now encrypt a message to that recipient. The receiver, who holds sk_{id}, will be able to decrypt the resulting ciphertext; moreover, an eavesdropper — even one who knows secret keys $sk_{id_1}, \ldots, sk_{id_n}$ for several other identities — learns no information about the message.

The idea of identity-based cryptography was suggested by Shamir in 1984, but for many years there were no satisfying realizations of identity-based encryption. (In contrast, identity-based signatures are much easier to construct.) This changed dramatically in 2001 when two different IBE schemes were proposed: one by Cocks (published, interestingly enough, in a second-tier conference

without much fanfare) and a second by Boneh and Franklin. Since then there has been a flurry of interest in this primitive, with scores of other papers on the topic. It is worth noting also that this work inspired a significant generalization of IBE called *predicate encryption* which is interesting in its own right.

2 Summary of the Book

The present book provides a fairly thorough survey of IBE. Publication of the book has come at an opportune time: the notion of IBE has been studied long enough that the field has stabilized (to some extent), yet IBE is new enough that there is still significant excitement about the topic as well as room for further research.

Following an introductory chapter, the book begins in Chapter 2 with definitions for standard public-key encryption as well several variant definitions that have been considered for IBE (see more below). They also include a brief overview of cryptographic security proofs that might be useful for readers who have not seen such proofs before.

IBE schemes in the literature have predominantly been constructed using relatively recent techniques and assumptions related to bilinear pairings over elliptic curves. In Chapter 3 the authors provide a concise review of elliptic curves, pairings, and some popular cryptographic hardness assumptions defined in that setting.

Chapter 4 begins an in-depth study of IBE constructions, starting with the initial Boneh-Franklin IBE scheme and its variants. Proofs of security against chosen-plaintext and chosen-ciphertext attacks are given, and some other variants are discussed. In addition, the Gentry-Silverberg *hierarchical* IBE scheme is discussed.

The Boneh-Franklin scheme (and its variants) were proven secure with respect to a strong security definition (where, roughly, the adversary can choose the receiver he wants to attack at any point) but in the random-oracle model (where, very informally, the existence of a truly random function is assumed). Chapter 4 presents subsequent work that shows how to remove the random oracle, but under a weaker definition of security where the adversary is forced to decide, in advance, which user he will target. The main portion of this chapter is a description of the Boneh-Boyen scheme, along with a security proof.

The next chapter take a slight detour to discuss methods for securing IBE schemes against chosen-ciphertext attacks, without relying on random oracles. Here, the focus is primarily on a technique suggested by Canetti, Halevi, and myself which converts any hierarchical IBE scheme secure against chosen-plaintext attacks into an IBE scheme secure against chosen-ciphertext attacks.

Chapters 7 and 8 return to the main thread, and present IBE schemes that achieve the strong security notion initially proposed by Boneh and Franklin, but without assuming random oracles. This includes descriptions and, in some cases, proofs for the IBE schemes of Boneh-Boyen, Waters (2005), Gentry, and Waters (2009). A few other schemes are mentioned in passing.

Chapter 9 discussing IBE schemes that do not rely on pairings. This includes the IBE scheme of Cocks and Boneh-Gentry-Hamburg, which are based on assumptions related to factoring, and the recent schemes of Gentry-Peikert-Vaikuntanathan and Agrawal-Boneh-Boyen that rely on lattice-based assumptions.

The concluding three chapters discuss some "miscellaneous" topics, including some work related to IBE in either technique or spirit (Chapter 10), techniques for achieving privacy against the

authority who generates the public parameters (Chapter 11), and a brief mention of some companies implementing IBE and related standardization efforts (Chapter 12).

3 Recommendation

I was impressed with the breath and scope of this book, which hits on all the key references in this area and is fairly up-to-date as well. (One of the schemes described in the book was published as recently as August, 2010.) I did not notice any omissions in the bibliography, either. The authors are to be commended for doing such a nice survey of the literature.

Very often, however, I did wish that more details were included. The book is relatively thin (under 180 pages), and the authors could have easily included proofs for *every* scheme included in the book. Similarly, though somewhat further removed from the core focus of the book, the authors could have also provided more information on pairing-based cryptography (including parameter choices) and lattice-based cryptography.

Nevertheless, I would recommend this book to any beginning graduate student or researcher interested in pursuing work on IBE. The book serves as a valuable guide to the literature, as well as a solid introductory treatment of the topic.

Review[9] of

Resources for Teaching Discrete Mathematics –
Classroom Projects, History Modules, and Articles
Edited by Brian Hopkins
MAA, 2009
322 pages, $54.95, Softcover

Reviewer: Myriam Abramson mabramson@faculty.umuc.edu

1 Introduction

Discrete mathematics can be challenging to teach because (1) not everybody in the class has the same math "maturity" and (2) not everybody in the class thinks like a computer scientist at this early stage of the curriculum. Discrete math texts often suffer from a circularity problem: they provide definitions and proofs that require knowledge of concepts not yet covered! There have been several attempts at introducing discrete math concepts through a problem-oriented approach [2, 5]. This supplemental book takes this approach a step further by introducing what can best be described as an activity-oriented approach.

2 Summary

The book consists of three parts. The first and largest is a collection of classroom projects (19). The second part is about historical milestones in discrete math (12). The third and smallest part contains notable articles in discrete math (5). All projects and articles reflect typical classroom experiences.

2.1 Classroom Projects

Each classroom project centers around one specific problem and has instructor notes, a worksheet for the students, and the solutions to questions in the worksheet. Variants or further explorations of the main questions are sometimes included. The instructor notes contain information on when to introduce the activities in the curriculum, the prerequisites, background references, and teaching suggestions. I particularly appreciated the suggestion to rename a problem to prevent the students from finding solutions on the internet (e.g., "Bulgarian Solitaire" becomes "The Coins Go 'Round 'n 'Round"). The worksheets contain a full description of the problem and the list of activities and/or questions. There are many concrete activities, often games and puzzles, that the students can actually do with coins, beads, tokens, peg boards, graphing calculators, rope, etc. . Those concrete activities are excellent for exercising intuition. Most of those classroom projects are meant to be done in groups in class for one or two class periods, but can also be adapted to an online classroom for participation. Since I am particularly interested in the latter, I'll describe below those that could best be adapted to an online classroom for beginning undergraduates.

Recursion is one of the discrete math topics that students struggle most with, perhaps because the math treatment of recursion with recurrence relation is somewhat "unnatural" compared with programming with recursion. Bigger and faster machines have killed the appreciation for what

a little thought can do to make programs run faster. There are two classroom projects on this subject: "Exploring Recursion with the Josephus Problem" and "Using Trains to Model Recurrence Relations." The Josephus problem is a well-known problem to introduce recursion and recurrence relations, and more material can be found in a textbook by the same author [1]. The project has interesting variations on this problem. Shai Simonson has a full online lecture on the Josephus problem[3] that can complement those projects nicely; I encourage my own students to watch it. The other sample classroom project address the problem of building trains of different length given cars of a certain length. The important questions to ask students are clearly stated.

Writing proofs is one of the distinguishing topics of a course in discrete math. "The Game of Take Away" is a classroom project whereby the student needs to communicate clearly and succinctly a winning strategy for the game. I gave my students the suggested Thai 21 variant for discussion and, from their comments, found it helpful to encourage them to bridge the gap from game playing (most students are avid game players) to proof writing. Several projects encourage the student to formulate conjectures such as "Bulgarian Solitaire." Proofs by induction, another difficult concept, is addressed head-on in two class projects, the "Pile Splitting Problem" and the "Two Color Theorem."

Graph theory is another topic distinguishing a course in discrete math. Several classroom projects such as "Further Explorations with the Tower of Hanoi" and "Bulgarian solitaire" help make the connection between problem solving, state-space search, and graph search algorithms. Since graph theory is often taught last in the curriculum, the student should already be familiar with the towers of Hanoi in recursion and recurrence relations and should know how to compute the optimal number of moves. This project will nicely make the connection to graph theory and provides an opportunity to examine one problem from several angles throughout the class. "Graph Complexity" is maybe one of my favorite projects because it forces the student to think out of the box and actively look for patterns using their knowledge of graph theory.

2.2 History Modules

It helps sometimes to know the historical environment leading to a concept in order to uncover the motivations and rationale. However, many discoveries in mathematics often give the feeling that they were pursued for their own sake and we marvel instead at their new-found use. Many history modules give this feeling. The history modules are in the form of open-ended classroom projects. Unfortunately, no solutions or suggested solutions are provided. What is provided in all those projects are direct quotes from the original sources. This is invaluable since the original sources are often hard to get and comprehend without context. Those projects give the student a chance to rewrite concepts in their own words. By providing the original sources, those projects also force a comparison between the prose style of the original thinkers and the modern mathematical notation. All the projects in this section have questions involving a rewrite of the concepts using the modern notation and terminology with the original text providing guidance. A few programming projects are suggested in this section.

The first two projects on the development of binary arithmetic ("From Leibniz to Von Neumann" and "From Shannon to the Chinese Abacus") are suitable for an introductory course in computer science or discrete math. "Pascal's Treatise on the Arithmetic Triangle" complements nicely the classroom project on "Generalizing Pascal: the Euler Triangles" to help understand the connection between binomial coefficients and combination numbers. There are three projects on "Early Writings in Graph Theory." One of them, "Euler Circuits and the Konisberg Bridge Problem", provides an

example of a modern proof with the missing steps highlighted to practice modern proof writing. "Counting Triangulations of a Convex Polygon" introduces the concept of triangulation from original sources and one can only marvel at the seminal work done by Euler and taken up by mathematicians in the 19th century that found its way into modern image processing and mesh generation. This project has numerous possibilities for pattern discoveries and conjectures. The last three projects in this section, "A study of Logic and Programming via Turing Machines," "Church's Thesis," and "Two-way Deterministic Finite Automata" address the issue of computability which is better suited for an advanced undergraduate or graduate class on the theory of computation. However, it is important to stress in a discrete math course that recursive definitions are Turing-computable functions to motivate their study.

2.3 Articles

The articles in this section provide either original or pedagogical content in discrete math.

Shai Simonson's "A Rabbi, Three Sums, and Three Problems" is an exposition of Rabbi Levi Ben Gershon's work on proving three formulas for the sum of consecutive integers (usually attributed to Carl Gauss), sum of consecutive squares of integers, and sum of consecutive cubes of integers. The original proofs themselves are not provided. Those problems lead to three related problems. One of them, counting the numbers of triangles in a triangular grid, is an original contribution by the author stressing the data-driven, pattern discovery approach to proofs.

"Storing Graphs in Computer Memory" removes some of the mystery of implementing graphs in a computer program that every computer scientist should know and is suitable to a beginning undergraduate class in computer science or discrete math. The original contributions by the author are the description of a linked list using the treasure hunt game and an implementation of a vector-based rather than matrix adjacency list.

"Guided Group Discovery" are notes on adapting Kenneth Bogart's "Combinatorics through Guided Discovery" notes to larger class size, weaker mathematical background, and lower motivation. The conclusions are not surprising: weekly expectations and feedback work, synchronization of the material will give every student a chance to participate in the discussions, and a lot of variability is found in classroom discussions. The example "labeled trees and Prufer codes" is given as a classroom project which I think will be very effective in promoting interaction in the classroom.

Susanna Epp's "Use of Logic in Teaching Proof" is excellent in identifying the mental pivot points necessary for doing proofs and provides several teaching suggestions. One suggestion, student critiques of proofs, can be applied in the context of a history-based project for example as we have seen in the preceding section. This article synthesizes nicely several works in the literature on proofs such as Polya's "How to Solve it" but ignore other important contributions [4, 6].

3 Opinion

The level required for each classroom project vary greatly. Some projects are elementary (e.g., the "Game of Take Away") while some require more rigor and maturity. Some projects have concrete activities while others have questions of an applied nature to the problem. Several classroom projects require knowledge or interest in other subjects such as chemistry ("Perfect Matchings and Benzenoids"), topology ("Exploring Polyhedra" and "Can you make the geodesic dome?"), typography ("A Problem in Typography"), and bioinformatics ("Codon Classes"). The level ranges from

beginning undergraduate courses to advanced undergraduate courses and include topics suitable for an undergraduate thesis or a capstone project.

The level required for the history-based projects also vary but most of them have elementary questions involving reading comprehension and rewriting using modern notation and terminology. The content addressed in the articles fits a beginning undergraduate course.

This book is a treasure trove of problems and activities in discrete mathematics that is sure to stimulate students and teachers alike. This book is accessible through Google Books, so I would recommend taking one suggestion found in the book: change the name of a project before offering it to your class.

References

[1] D.E. Ensley and J.W. Crawley. *Discrete mathematics: mathematical reasoning and proof with puzzles, patterns, and games.* Wiley, 2006.

[2] Daniel A. Marcus. *Combinatorics – A Problem-Oriented Approach.* The Mathematical Association of America, 1998.

[3] Shai Simonson. Lecture 10, the josephus problem. Retrieved from `http://video.google.com`.

[4] D. Solow. *How to read and do proofs: An introduction to mathematical thought processes.* Wiley, 1990.

[5] Clifford Stein, Robert L. Drysdale, and Kenneth Bogart. *Discrete Mathematics for Computer Scientists.* Pearson, 2011.

[6] D.J. Velleman. *How to prove it: a structured approach.* Cambridge Univ Pr, 2006.

Review of[10]
Proofs and Algorithms
Gilles Dowek (translation by Maribel Fernandez)
Springer, UTCS, 2011
xii+155 pages, Paperback, $32.00

Reviewer: Michaël Cadilhac michael@cadilhac.name

1 Overview

This book contains the lecture notes of a course given by the author, and focuses on logic (with a strong emphasis on proofs and a smaller one on model theory), computability (with a presentation of recursive functions, rewriting systems, lambda calculus, and Turing machines), and the links between the two.

The proclaimed goal of the book is to provide a partial answer to the question: "To what extent the construction of a proof can be replaced by the application of an algorithm?" which is answered by standard material on automated theorem proving. The book is said to be designed for undergraduate students, but is claimed to contain "all that philosophers, mathematicians, and computer scientists should know about logic," a claim undoubtedly strong.

Quantitatively, the book is split into three parts: proofs, algorithms, and the links between the two. It is divided into 8 chapters, plus introduction and conclusion. The first 6 chapters average 22 pages per chapter, and the last two 5. A grand-total of 52 exercises of varying difficulties helps the reader in understanding the concepts and deepens them. Finally, the book contains close to zero historical view of the fields involved, no reference (except for a few classical books), and a sparse index.

2 Summary

2.1 Part I: Proofs

Chapter 1 is a lengthy presentation of predicate logic (with the definitions of *language* and *term*, the former being a set of *sorts* (types) and a set of typed *relation* and *function symbols*, and the latter, type-respecting compositions of symbols), the notions of sequent, proof (with natural deduction), and theory (with a few examples including Zermelo-Fraenkel set theory). A discussion is given on other ways to express the axiom of excluded middle, with, as an important step, the introduction of the so-called "System D'" (the name is not motivated), i.e., natural deduction with multiple conclusions.

Chapter 2 presents the concept of model. Therein the soundness theorem (if a theory has a model, it is consistent) and the completeness theorem (if a theory is consistent, it has a model) are proved. Although Gödel's second incompleteness theorem is deemed "out of scope" for the book, a quick presentation of relative consistency proofs is given. The notions of extension and preservation under extension are then presented, together with Skolem normal form. The chapter is closed with a weak form of Löwenheim-Skolem theorem.

2.2 Part II: Algorithms

Chapter 3 focuses on computable (recursive) functions. Decidable and semi-decidable sets are introduced. Arithmetization is presented, leading to computability over lists and trees. It is then proved, using Gödel's β function, that the induction scheme can be replaced when arithmetic is given in addition as a primitive. *Programs* are then defined as expressions describing computable functions, and it is shown that the halting problem is undecidable — that is, given a program and its arguments, it is undecidable whether the function described by the program will have a value on these inputs. It is then shown that there exists an "interpreter," i.e., a universal function that takes a program and some arguments and returns the value of the program on the arguments. This highlights that programs can be executed in a step-by-step fashion.

Chapter 4 provides examples of computational models equivalent to computable functions, and shows for each of them that their execution can be seen as a sequence of (small) steps. The three models presented therein are: rewriting rules, lambda calculus, and Turing machines. The pattern is the same for the three models: it is first described how they represent functions, then shown that, within this interpretation, the computational power of the model is equivalent to computable functions.

2.3 Part III: Proofs and Algorithms

Chapter 5 starts off with the notion of reduction, which is defined verbatim as the following: "To show that a problem is undecidable, it is sufficient to build an algorithm that *reduces* the problem in question to a problem that has been shown to be undecidable." A method to translate a program to a proposition in arithmetic-capable languages and theory is given, so that the function described by the program has value q on $p_1, \ldots, p_n$ iff the proposition is provable when its arguments are replaced by $q, p_1, \ldots, p_n$. Together with the undecidability of the halting problem, this implies Church's theorem: within a language with at least one binary predicate symbol, provability is not decidable in the empty theory, although it is semi-decidable. Gödel's first incompleteness theorem is given as a corollary.

Chapter 6 focuses on automated theorem proving. Proof searching in natural deduction is hinted to be hard, and sequent calculus (with, then without, cuts) is introduced as an equivalent deduction system in which this task seems easier. In particular, the chapter includes John A. Robinson's unification algorithm, which deals with the choice of terms when backtracking the proof of a proposition.

Chapter 7 presents Presburger's theorem: provability in first-order arithmetic without multiplication is decidable.

Chapter 8 gives a quick presentation of constructivity, i.e., the property that if $(\exists x)[A]$ is provable in some sense, then there is a term t such that the proposition A with t replacing x is provable. It is shown that *constructive proofs*, i.e., proofs without excluded middle, respect this property. The effective correspondence between different *constructive* deduction systems is outlined, and it is hinted that converting a proof of a proposition to a cut-free sequent calculus proof corresponds, in some sense, to an *execution* of the program described by the proposition — i.e., cut elimination in a constructive proof of $(\exists x)[A]$ gives a way to find an x that verifies A.

Chapter 9 concludes and is followed by a sparse index.

3 Opinion

This book contains in a nutshell some essential results in logic and theoretical computer science. It offers a short run through the concepts of proofs, computability, computing models and their limits, Church-Turing thesis, and proof searching, with a notable effort to connect and unify those.

My global and very personal opinion is the following: I feel that this book is written in a heavy, hard to read style, lacks motivations and perspective, and offers none of the much needed historical backdrop; furthermore, as its content is fairly standard, one may consider relying on well-established texts to learn the topics. However, this is a challenging reading, if not by its contents, at least by its style, and it could be seen as a test for a student *who knows already the results therein* — it should be noted that the exercises are usually of great quality. The book can be used as the directing line of a one- to two-semester course, *if the prospective teacher agrees to the choice of topics* (i.e., with automated theorem proving in mind).

Overall, the book lacks a driving force at some key points and, to some extent, structure. Moreover, in the same manner that some notations are unmotivated, whole sections take a lot of time to make sense in the global text, and this distracts and demotivates the reader. Its style owes a lot to the fact that the text comes from lecture notes, and it should not be used for (and is probably not designed for) self-learning.

As I pointed the style of writing as a great flaw of this book, I should give a few arbitrary examples, especially as these criticisms are highly subjective; if you disagree with those examples, then you may assert that the book is well written and that I am a pedantic bloke, no offense taken. (1.) The author uses repetitions as a mean to outline similarities between concepts; this results in entire paragraphs being the same, with the exception of a few words. I feel this redundancy can be avoided, thus leading to a better flow of reading and to reduced boredom. (2.) The use made of ellipsis is somewhat hard to decipher: sentences such as "the first one of sort s_1, ..., the nth one of sort s_n" are common and sometimes entangled with other uses of ellipsis. These would probably be more readable as "the i-th of sort s_i, $1 \leq i \leq n$." (3.) I had a hard time reading function definitions such as "the function f which associates $\heartsuit$ to $\spadesuit$" (meaning that $f(\spadesuit) = \heartsuit$). The more common term "maps" could have been less ambiguous. (4.) The book starts with considerations on inductive definitions (orderings, fixed point theorems, structural induction, ...); the treatment is very formal. However, its explicit use in the rest of the book seems rather scarce, thus it comes to me as a curious choice of introduction. (5.) The concept of *language*, which is the unifying tool used to define logics (e.g., first-order logic with $+, \times$) is overly generic — its apparent genericity is even dropped later in the text. Its presentation is hard to follow, going back and forth between formal and informal — this can be considered as a comment on the whole book. (6.) A few terms are undefined, although the reader versed in model theory or logic should have no problem understanding them.

Finally, the book is claimed to be directed to "philosophers, mathematicians, and computer scientists" who should learn in it "all that [they] should know about logic." However, some topics which are essential (w.r.t. one of these parties) are only hinted (e.g., philosophers would need the historical perspective of the concepts; mathematicians, the main parts of set and model theory; computer scientists, database theory, complexity considerations on proof searching, or an in-depth study of Turing machines). Similarly, some of the topics treated can hardly be considered as required knowledge for the *three* parties. All in all, my thought on the aforementioned assertion is that philosophers, mathematicians, and computer scientists have different needs of logic, and "all that [they] should know about logic" (or, more likely, the union of their needs) is not expected to

fit in less than 200 pages.

Review of[11]
Introduction to Computational Proteomics
by Golan Yona
CRC Press, 2011
746 pages, Hardcover,$80.00

Reviewer: Dimitris Papamichail dimitris@cs.miami.edu, Dept. of CS, U. of Miami, USA

1 Introduction

Computational proteomics is a term that generally describes the use of computational methods to analyze proteins, primarily to determine their structure, dynamics and function. Proteins are probably the most important class of biochemical molecules and consist major structural and functional components of each cell in every organism. They are complex molecules and are assembled in chains from 20 different amino acids. Proteins can be described by a hierarchy of structural levels: The *primary structure* which is the raw sequence of amino acids, the *secondary structure* which labels the amino acids by their participation in local structural features such as helices and strands, the *tertiary structure* which describes the coordinates in space of each amino acid, and the *quaternary structure* which refers to arrangements of proteins to form complexes.

2 Summary

This book consists of two parts, 'The Basics', spanning chapters 1 through 9, and 'Putting All the Pieces Together', which includes the last 5 chapters. Despite the categorization, there does not seem to be a clear distinction between the parts, as chapters progressively move through the analysis of increasingly complex entities and, as expected, the concepts build progressively as well, with applications addressed throughout the book.

The contents of each chapter are outlined below.

Chapter 1 The six-page first chapter introduces computational proteomics and explains some of the challenges that motivate the problems and solutions presented in this book.

Chapter 2 A primer of basic concepts in molecular biology is provided in the second chapter, including DNA and protein molecules, the central dogma of molecular biology, and the different structure levels of proteins.

Chapter 3 Genomic sequence comparison is probably the most common operation in computational biology. It is used to identify, compare and retrieve not only proteins, but DNA and RNA molecules as well, since all can be described as chains of basic building blocks, which are assigned letters from the English alphabet. After a short description of the global and local alignment algorithms to compute string edit distance, the author delves into the statistics of alignments, in order to calculate the significance of a similarity score between two sequences. The chapter continues with the theory behind scoring matrix computation for protein alignment, such as PAM and BLOSUM, and ends with considerations of other distance functions.

Extended appendices expose some of the probabilistic and statistical concepts presented in this chapter.

Chapter 4 Alignment of two sequences can be extended naturally to multiple sequences. Multiple alignment is powerful in detecting highly conserved regions, which quite often indicate functionally important motifs. Since the problem of multiple alignment is NP-hard for most reasonable distance cost functions, most of the chapter is devoted to practical heuristics, such as probabilistic methods based on profiles and position specific scoring matrices. Iterative and progressive alignment techniques, and other extensions of multiple alignment such as partial order alignment are also presented.

Chapter 5 This chapter explores another computationally hard but very important problem, motif finding, the endeavor of identifying patterns that appear in a large subset of given sequences, where these patterns are not necessarily identical. The author describes statistical modeling techniques, including the Gibbs sampling method and the Multiple Expectation Maximization for Motif Elicitation (MEME) algorithm, as well as a couple of combinatorial heuristics.

Chapter 6 Here we are introduced to Hidden Markov Models (HMMs), initially through the example of gene prediction, progressing to more elaborate classification problems. Forward, Backward, Viterbi and Forward-Backward algorithms are presented in the context of using HMMs for protein family classification. These are followed by higher order HMMs, variable order HMMs and the use of Probabilistic Suffix Trees (PSTs) to model prediction problems.

Chapter 7 This chapter serves as an introduction to machine learning and its applications in computational biology. Several topics are explored, including linear classifiers, non-linear discriminant functions, mappings to higher dimensional spaces and kernels, use of Support Vector Machines (SVMs) for protein classification, decision trees and feature selection.

Chapters 8 & 9 The last two chapters of the first part of the book introduce concepts related to the tertiary structure of proteins. Chapter 8 in particular deals with structure prediction and folding of a protein in 3D space. It also examines secondary structure prediction, a computationally easier but still quite informative task. The author describes structure comparison methods and distance measures used, coupled with the analysis of the statistical significance of structural matches, in a matter analogous to the alignment score significance. Chapter 9 continues with protein domains, distinct parts of proteins that can be considered modular building blocks. Several methods to predict these ab initio or with the help of similar sequences are examined, including statistical machine learning methods.

Chapter 10 The first chapter of the second part, 'Putting Things Together', revisits clustering and classification, but this time provides a more comprehensive introduction and examines the most popular clustering methods. These include k-means, hierarchical and graph based algorithms, as well as methods to assess cluster quality. The chapter ends with applications of the clustering methods in grouping related proteins.

Chapter 11 Embedding algorithms are used to map dataset spaces to other (often Euclidean) spaces and lower dimensional spaces, to study or visualize their properties. In this chapter an array of embedding techniques, often encountered in the machine learning field, are introduced, including linear techniques such as Principal Component Analysis (PCA) and Singular Value

Decomposition (SVD), and non-linear such as Multidimensional scaling (MDS), Random Projections algorithms and Manifold Learning. Vectorial representations, meaning mappings of objects to vectors of features, are discussed at the end of the chapter.

Chapter 12 The analysis of gene expression data, as produced primarily by microarray experiments, is the topic of the 12th chapter. Based on different measures of similarity, the author presents methods to evaluate gene co-expression, discover classes of genes based on their expression patterns, and statistically evaluate the quality of these analyses.

Chapter 13 The 13th chapter deals with the interactions between proteins, either physically (quaternary structures) or functionally (participate in a common process). After an exposition of experimental techniques to determine protein interactions, an array of computational techniques is described, divided into structure-based and sequence-based inference methods. The latter includes a sequence signature based method, where the frequencies of over-represented motifs are used to predict interactions between protein domains. The chapter continues with an introduction to interaction networks of proteins and their topological properties.

Chapter 14 Cellular pathways are procedures that carry out specific functions in the cell. It is of great importance to construct these pathways and predict the participating proteins. A few methods to deterministically and probabilistically assign genes to pathways are presented in this chapter.

Chapter 15 The last chapter describes the usage of Bayesian networks to model gene networks. It serves mostly as an introduction to Bayesian networks and how to learn their parameters and structure.

3 Opinion

The last couple of years have seen an explosion of new books in the fields of computational biology and bioinformatics enter the market. The fields themselves have expanded significantly, with computational, statistical and engineering methodologies underlying or supplementing a wide variety of studies in the life sciences.

The *Introduction to Computational Proteomics* text is an interesting combination of general computational biology topics, statistical methods, machine learning techniques and proteomics theory and applications. It would make a lot of sense to add '*A statistical approach*' at the end of the title, as the book delivers a thorough exposition to a possibly larger number of statistical techniques than most biostatistics books available. If not for the large number of examples drawn primarily from the proteomic field, it could also very appropriately serve as a general computational biology text. Indeed, this text is used in a couple of computer science course offerings in US as a general computational biology textbook, albeit adopted by professors with machine learning and/or statistical research orientations.

If I had to describe the content of this book with one word, it would be 'valuable'. The third chapter, devoted to sequence comparison, offers a detailed and insightful exposure to the Karlin-Altchul statistics that underlie the BLAST heuristic and alignment statistical significance calculations, which is in par, if not more analytic, with the coverage of the concepts in the specialized BLAST book. The machine learning techniques throughout the text easily cover the majority of

topics found in an undergraduate machine learning course. And for topics not covered in sufficient depth (one can do only so much in 700 pages), 958 references provide ample pointers to external sources of specialized knowledge, covering most seminal and important papers in each subject.

This book requires a good background in statistics and probability, calculus and possibly some introductory biology, the latter if one wants to comprehend every example and application presented. Some 'basic' concepts are explained in conveniently placed appendices at the end of the chapters where they are initially used, but these short introductions most probably will not be sufficient for the unprepared reader who wishes to comprehend the material to its entirety.

Overall this is an excellent book, content rich and concise, covers a popular array of topics in computational proteomics and biology in general, and will become a valuable addition to the bookshelves of aspiring students as much as researchers in the computational biology field.

Review of[12]

Computability and Complexity Theory
by Steven Homer and Alan L. Selman
2nd Edition, Springer, 2011.
$52.00, Hardcover, 225 page

Beautiful and filled with deep results, the theories of computability and computational complexity form two of the most important parts of theoretical computer science. Although the core concepts are about 75 years old and 50 years old, respectively, both computability and computational complexity continue to remain active areas of research.

Despite the age and importance of these theories, there are surprisingly few textbooks available that are suitable for undergraduates and up-to-date. This book, written by two experts in the field, is a revised version (2nd edition) of a text originally published in 2001.

Chapter 1 covers the basics of words, languages, base-k representation, logic, and some algebra and number theory. About 8 pages are devoted to the background material needed later on to show that PRIMES are in NP (Chapter 7). Even before more recent results showed the much stronger result that PRIMES are in fact in P, this choice seems unwise.

Chapter 2 is an introduction to computability, covering the basics of Turing machines and variations on them.

Chapter 3 covers unsolvable problems (here called "undecidable"), the halting problem, the recursion theorem, Rice's theorem, and oracle Turing machines.

Chapter 4 is an introduction to computational complexity.

Chapter 5 contains the basic results of complexity theory, including linear compression and linear speedup, separation results, padding techniques, and the Immerman-Szelpcsényi theorem.

Chapter 6 introduces nondeterminism and the class NP; the Cook-Levin theorem is proved and some NP-complete problems are presented.

Chapter 7 talks about NP-hardness, the polynomial hierarchy, and some other complexity classes, such as PSPACE, are discussed briefly.

The new material, introduced in this 2nd edition, begins in Chapter 8. Here nonuniform complexity and circuits, and the low and high hierarchies, are covered.

Chapter 9 covers alternating Turing machines and uniform circuit families.

Chapter 10 introduces the probabilistic complexity classes PP, RP, ZPP, and BPP.

Chapter 11 discussing counting classes and proves Toda's theorem.

Chapter 12 covers interactive proof systems and proves that IP = PSPACE.

This book could be improved in various ways:

- The writing conveys little feeling of interest or excitement. It seems unlikely to me that a reader will be motivated to to pursue complexity theory from reading it. Little intuition behind proofs is presented.

- Very few open problems are listed — and even fewer that a young researcher could plausibly solve. Open problems in a book demonstrate that it is covering a dynamic field with plenty of opportunities for new contributions.

[12]

- Some terms, such as "fan-out", are not adequately defined for beginners.
- Although the preface claims "there is nothing in such a course [automata and formal languages] that a student needs to know before studying this text", that's not completely true. (For example, regular languages and context-free languages are mentioned in Corollary 3.7, but never defined!)
- There is nothing about parameterized complexity, Kolmogorov complexity, cryptography, quantum computation, or the hardness of approximation.
- Finally, the presentation has some deficiencies.

Let me expand on this last point, listing just a few of the problems I noticed. The book has not been adequately proofread, as several spelling and capitalization errors remain (e.g., "length of it's input strings", p. 189; the names of researchers "Presburger" and "Wigderson" are misspelled throughout; "Entscheidungsproblem" is not capitalized, as it must be, since it is a German noun).

Sometimes multi-letter functions are written (correctly) in the roman font (e.g., "card", p. 8); sometimes they are written (incorrectly) in the italic font (e.g., "*range*", p. 8), and in at least one place they are written in both fonts simultaneously ("c*ard*", p. 10)! Sometimes new terms are written in the italic font when they are introduced (e.g., "lexicographic", "partial function", "diverges") and sometimes not ("subword").

Sometimes the authors introduce new notation when existing notation is fine; e.g., "$\mathrm{rm}(a, m)$" for the function that is usually written as "$a \bmod m$". They use (p. 111) the symbol $\subset$ for strict set inclusion, when there is a suitable symbol $\subsetneq$ in the `amssymb` package that already means this, and is unlikely to be as confusing. The notation || is used on p. 128 without being defined; it seems to mean parallel execution of two Turing machines.

The references and the index are not really adequate. For example, Cobham's thesis is mentioned on pages 36 and 75, but no paper by Cobham is cited in the references. One paper (by Žák) is listed alphabetically under the letter V instead of Z.

One surprising innovation is the inclusion, in the *index* (!), of the abbreviated reference tag for certain selected papers. For example, a 2001 paper of Impagliazzo, Kabanets, and Wigderson appears in the index as "KW01", and a 1980 paper of Karp and Lipton appears in the index as "KL80". Although novel, I am not sure this practice is likely to be adopted by other authors. At least one item in the index, HITTING SET, has no page number attached to it at all.

Where was Springer in the editorial process, I wonder?

The bottom line is that this book would not be my first choice as a textbook on computability and complexity. If you are teaching a one-semester course, then Sipser [5] currently seems the best choice to me, despite its high price; it covers essentially Chapters 1–7 of the book under review, and it also treats regular and context-free languages. Hopcroft and Ullman's 1979 book [3] is another possible choice for a 1-semester course, but it is now out of print. (The new edition of their book, written with Motwani, removes much of the advanced material.) If you are teaching a two-semester course, then you will probably want to consider Arora and Barak [1] instead. Another possible choice is Papadimitriou [4] or Balcázar, Díaz, and Gabarró [2], although both books would have to be supplemented with additional material.

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I* and *Structural Complexity II*, Springer-Verlag, 1988, 1990.

[3] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[4] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. Reprinted with corrections, 1995.

[5] M. Sipser. *Introduction to the Theory of Computation*. Third Edition. Cengage Learning, 2013.

Review[13]

Quantum Computing Since Democritus
Author: Scott Aaronson
Cambridge University Press
$39.99, Soft cover, 370 pages, 2013

Reviewer: Frederic Green (fgreen@clarku.edu), Dept of Math and CS, Clark University

1 Introduction

Some books hold your attention just by virtue of their intellectual content. They are exceedingly well-written, and deal with such deep and fascinating subjects it's impossible to ignore them. Other books reach out and grab you by the lapels (figuratively, of course, since lapels aren't all that grabbable these days), not letting you go until you've let the author have his or her say. The goal of yet other books is to entertain.

"Quantum Computing Since Democritus" (henceforth, QCSD) does all of the above.

Now in case I'm not making myself clear, I hope the reader will forgive me in indulging in a bit of lapel-grabbing myself (in the spirit, indeed almost in the words, of QCSD's author, Scott Aaronson):

> Duuuuude, ... **read this book**!!!!

Indeed, how can I adequately convey the scope, erudition, virtuosity, panache, hilarity, the unabashed nerdiness, pugnacity, the overwhelming exuberance, the relentless good humor, the biting sarcasm, the *coolness* and, yes, the intellectual depth of this book? Well, perhaps its most obvious virtue is its style. QCSD is based on lecture notes from a course that Aaronson gave, under the same title, at the University of Waterloo in 2006. It very deliberately retains the flavor of a spoken lecture, and this is one of its most appealing features. But these aren't just any old spoken lectures. Those of you who are familiar with Aaronson's rhetoric (echoed in the exhortation above, and, to an extent, elsewhere in this review, with all du(d?)e respect[14]) probably know what I mean. Especially for those who aren't, I can do no better than quote the following passage from his own "critical review" of QCSD, included in the Preface.

> Having now read the book, I confess that I've had my mind blown, my worldview reshaped, by the author's truly brilliant, original perspectives on everything from quantum computing (as promised in the title) to Gödel's and Turing's theorems to the **P** versus **NP** question to the interpretation of quantum mechanics to artificial intelligence to Newcomb's Paradox to the black-hole information loss problem. So if anyone were perusing this book at a bookstore, or with Amazon's "Look inside" feature, I would *certainly* tell that person to buy a copy immediately. I'd also add that the author is extremely handsome.

[13]

[14] Be it known that QCSD has a "dude" count of 6, the highest of any book I have ever read. And mind you, I did not count exceptions to the standard orthography, such as "duuuuude."

Perhaps modesty forbids him from also pointing out that he is one of the world's experts in quantum computing ("QC"). He is one of QC's most ardent advocates[15], and probably its most outspoken one, via his scholarly work, the blog "Shtetl-Optimized" (reviewed on these very pages a few years back), and now this book, especially the chapter "Skepticism of quantum computing." He has been quoted in such august venues as the New York Times and by Australian fashion models in ads for printers. I have followed the blog virtually since its inception, although I don't believe I have ever left a comment on it. Maybe this review can make up for some missed opportunities.

So what's the book about already? Even Aaronson wasn't quite sure at one point. But happily, and ironically, the Australian model/printer ad incident[16] helped him articulate the central theme. The chapter headings also give us something of a clue. We start out with "Sets," "Gödel, Turing and friends," move on to "Paleocomplexity," "P, NP and friends," and, after lingering at such landmarks as "Quantum computing," "Penrose," "Decoherence and hidden variables," and "How big are quantum states," we eventually find ourselves in the realm of "Free will," "Time travel," and "Cosmology and complexity." Get the picture? No? Okay, let's take a closer look.

Here is Aaronson's perspective on what QCSD is about. He observes (more or less correctly, as I can attest from personal experience) that in the typical quantum mechanics course, physics students roughly follow the historical development, learning about the wave/particle duality, Schrödinger's equation and how to solve it, perturbation theory, etc.; that is, how to coax results out of the mathematics to make contact with experiment. Then, "only after years of study" (or in some cases only after years of *research*; okay, perhaps a bit of an exaggeration) do they learn that nature is described by complex probability amplitudes, rather than classical probabilities. But, he contends, quantum mechanics is not really "about" those topics that arise in the historical treatment; it's *really* about a generalization of classical probability theory, which leads to new ideas about information, probabilities and observables and how they relate to each other. This is the central theme of the book, stated much more eloquently and in more detail in the Preface.

However, it seems to me that QCSD has a broader agenda. While it is categorically "since Democritus," it's not *just* "quantum computing." QCSD seems to be aiming at a synthesis of key foundational issues in mathematics, computer science, physics and philosophy, having to do with the nature of computation, the natural world and human thought. Quantum computing plays such a pivotal role in that it allows us to "tunnel through" from mathematics (largely as manifested in computational complexity) into the deep waters of quantum mechanics and physical reality itself (more generally, Aaronson regards "computer science" as the field of study that "mediates between the physical world and the Platonic world"). You can't very well try to *understand* quantum computing (to the extent that that is possible) without grappling with quantum theory and its various interpretations. Indeed, one of David Deutsch's motivations in formalizing the quantum Turing machine was to provide an argument in favor of the many-worlds interpretation of quantum mechanics. QCSD (and the history of QC itself) demonstrates that in coming to terms with foundational issues, we not only gain insight into the nature of reality, but by following through with the resulting ideas (as witness Deutsch and the QTM) we actually find ourselves solving very concrete and interesting

[15]Caveats: This is not to say that he is certain that quantum computers will ever be built (and is extremely skeptical that anyone, e.g., the company D-Wave, already has). He even makes the point that if QC proves to be impossible in principle, then that would be even more interesting than if they are eventually built. And even then, QC methods are useful as a mathematical technique in much the same way as the probabilistic technique.

[16]Briefly: Back in 2007, an Australian TV commercial for printers featured actresses who quoted a passage plagiarized from the 2006 QCSD lecture notes. That passage contained the very "overarching theme" he was reaching for. See `http://www.scottaaronson.com/blog/?p=277`.

problems (think Shor's factoring algorithm!). It is notable that in one of the very last sentences of QCSD, which I read after this paragraph was essentially completed, Aaronson maintains that, with time, "the boundaries between CS, math, physics and so on are going to look less and less relevant, more and more like a formality." QCSD does a great deal to blur these boundaries, much to our advantage.

2 Summary

Okay, that's what the book is *about*. What exactly is in it? Seriously, although Aaronson does an admirable job in the preface, it may be unfair to try to summarize the contents chapter by chapter here. There are just too many threads to follow, too rich and incompressible in their implications and connections, to reduce to a few words in a review. So I opted instead to sketch out the structure of the book and mention some highlights.

Roughly the first half of the book (chapters 1 – 10) lays the foundation for the second half. We start with a brief but insightful introduction to logic and set theory, which leads directly into discussions of Gödel's completeness and incompleteness theorems, Turing machines and the unsolvability of the halting problem, and so forth. As Gödel and Turing not only laid the foundations of computability theory but both also had seminal views about the nature of mind and consciousness, the stage is set nicely for the chapter "Minds and Machines." This chapter ultimately takes off from Turing's paper "Computing Machinery and Human Intelligence" and, in only a few pages, touches on a surprising array of issues about this age-old debate. It is one, among several, examples where no time is wasted in engaging in a topic as soon as possible. (It is also revisited later in the book as soon as further background allows.)

Chapters on computational complexity (where we meet the usual suspects, **P**, **NP**, and also the polynomial hierarchy), randomness (where **PP** and **BPP** make their first appearance, along with a discussion of derandomization) and cryptography (prominently, public-key crypto systems) then enable a cogent introduction to quantum mechanics, presented as an alternative to the classical laws of probability. This is by now a somewhat familiar approach, beginning with classical probabilities and replacing them with complex probability amplitudes. But here the QCSD weltanshauung comes into full swing, by arguing that this purely mathematical generalization of classical probability is so natural that it could well have been discovered well before the phenomena of quantum mechanics were known. Quantum computing, notably the class **BQP** (bounded error quantum polynomial time), is introduced in Chapter 10. Here, as throughout the book, the emphasis is much more on complexity classes (as models of efficient computation) rather than concrete models of computation (e.g., quantum circuits). Quantum algorithms (e.g., Shor's and Grover's) are not explained in any detail, and some readers may regard this as one of the (IMHO, few) shortcomings of the book. But these *are* covered very well in other books and throughout the web. Aaronson has other fish to fry!

Those first 10 chapters are highly entertaining, and full of extremely interesting insights and perspectives. But after Chapter 10, things get more novel, and really, *really* interesting! In the chapter simply entitled "Penrose," Aaronson engages in a carefully reasoned, sharply critical (and, in my view, quite fair) discussion of Roger Penrose's ideas on human intelligence and consciousness, as detailed in the books "The Emperor's New Mind" and "Shadows of the Mind." Penrose's conjectures on the nature of human thought are based on quantum mechanics, which explains the position of this chapter. More provocatively, Penrose claimed that it is quantum *gravity* that plays a crucial role. Perhaps we can forgive Aaronson for not providing that prerequisite! In fact, part of the point is that

it's really not necessary. Because, most provocatively, Penrose's thesis was that the human brain is capable of solving (what we normally call) uncomputable problems, and his reasoning was based on Gödel's Incompleteness Theorem. Aaronson analyzes (and easily demolishes) the argument that leads to that latter conclusion, drawing for background on the earlier chapters on computability. He also goes to some lengths to give Penrose the benefit of the doubt, and imagine what he (Penrose) *might* have meant. In answer to this question computational complexity has some interesting things to say.

Most of the later chapters intersect with Aaronson's research, as well as work of others that was done since the original course in 2006. Another distinguishing feature, at least in my reading, is that it is often hard (and always fun) to guess what twists and turns the chapter would take, given its title. Consider, for example, the chapter entitled "How big are quantum states?" It is remarkable that it can even be asked! But I would not have immediately guessed that answers would follow by studying properties of the complexity class **QMA** (in retrospect, since the question is about the exponentiality of these states, it makes a lot of sense). In "Decoherence and hidden variable theories," Aaronson confronts what he regards as " the central conceptual problem in quantum mechanics: not that the future is indeterminate (who cares?), but that the past is *also* indeterminate." There we also find a most ingenious application of the max-flow/min-cut theorem (of all things!). I would not have guessed that a discussion of "Skepticism of quantum computing" would lead so quickly into a digression on the holographic bound in cosmology. Until reading QCSD, I was unaware of any relationship between cosmology and deterministic space-bounded classes (where the space bound is the inverse of the cosmological constant). And did you know we can have "fun with the anthropic principle" by studying the complexity class $\mathbf{BPP_{path}}$ and related quantum classes[17]? And that if "time travel" were possible, then $\mathbf{P} = \mathbf{PSPACE}$ (well... maybe)? The list goes on.

I'm leaving out a lot in this description. For example, there are two chapters on proofs (probabilistic and interactive, classical and quantum; circuit lower bounds and the barriers that hate them are found here), and one on computational learning theory (including the learning of quantum states).

The book becomes ever more conversational, ever more Socratic, and ever more Aaronsonian, beginning in the "anthropic" chapter, where Aaronson begins to insert snippets of dialogues between him and his students. This culminates in the final chapter, "Ask me anything" (in the spirit of Richard Feynman) which roams even more freely, consisting entirely of student questions and Aaronson's answers. This ventures into, among other things, a very intriguing discussion of the black hole information paradox, including some of the very recent work on "firewalls."

3 Opinion

This leads in turn to the inevitable question: Just who is this book *for*? Certainly, the book will be of interest to computer scientists (emphatically the SIGACT community), mathematicians, physicists, and philosophers with an interest in science. But I'm tempted to say that *any* thoughtful, curious reader would learn a lot from it. It is, for sure, far more for everybody than something like Penrose's "Road to Reality" (which begins by expounding on the joy of fractions for the math-phobic reader, and later proceeds to discuss sheaf cohomology). There are sizable tracts of QCSD that are relatively light on equations and complexity classes. For example, in the chapter "Free Will," we

[17] Now don't tell me you've never heard of $\mathbf{BPP_{path}}$!

encounter Newcomb's Paradox, a two-player game, in which one of the players (the "Predictor") can tell the future. It is used to establish a connection between free will and the ability to predict the future. The problem is intuitive and well-explained, and Aaronson proposes a resolution that I believe is quite accessible to any thoughtful reader, who may also follow parts of the sequel on the Conway/Kochen Free Will Theorem. An added bonus "for everybody" is the omnipresent humor. Leaf through it and you'll find a zinger or two (or three) on just about every page. For a fairly low-key example, on the cosmological limits of computing:

> You can't actually build a working computer whose radius is more than 20 billion light years or whatever. It's depressing, but true.

The high spirits are not the least bit gratuitous; it is one of Aaronson's many vehicles (along with exercises, puzzles, and detours) for conveying ideas to the reader.

But, like Penrose's "RTR," it is not a book to be taken lightly, and math-phobes will have a hard time with it. It *does* get pretty technical, although most details are left out (more on this below). Many exercises ("for the non-lazy reader") are interspersed throughout the book, although fortunately for the lay *or* lazy reader, it is not essential to work these exercises to follow the flow of ideas. There are also a number of puzzles, posed like cliff-hangers at the end of one chapter, and answered at the beginning of the next. Some of the exercises will prove challenging to some graduate students (it was originally a grad course, after all), to say nothing of the lay public, who will not know what to make of most of them.

So, since it does get technical, can you use it as a textbook? It doesn't pretend to be self-contained, and I would guess most instructors would be reluctant to use it as a primary textbook in a conventional course. It is, in Aaronson's auto-critical words, "too wide-ranging, breezy, and idiosyncratic" for this purpose. But QCSD is a great *companion* for study and research. Thus, it would be eminently suitable for a seminar-style course with a fairly open structure. Most importantly, it contains numerous explanations of *how the proofs work* (e.g., a convincing, example-driven two-page proof of the Cook-Levin Theorem), stripping away the messy technical details and giving the intuitive "big picture." What a pity such explanations are largely missing from the research literature! Of course technical details are essential to be truly convinced of a proof, but the expert reader can work them out for herself, or (in many cases) already knows them, and they are readily available on-line and in numerous conventional texts. Furthermore, the insights and pointers to open problems make this a valuable resource for anyone from graduate students to advanced researchers. I for one plan on keeping this book close at hand as I explore the relevant literature.

Like any book, QCSD is not without its glitches. For example, there are some "*déja vu*" moments in which topics are introduced as if they had not appeared earlier, whereas they did. This appears to be an artifact of the original lecture note format, which no doubt will be rectified in future versions/editions. There are plenty of figures, but perhaps there could have been a few more. And there may be readers who will be either overwhelmed or put off by the free-ranging scope of the book. But I take these defects as either trivial or matters of taste.

QCSD was a delightful summer read. Well, the summer may be past, but it's great fall, winter and spring reading too.

Review [18] of
Algorithmic Puzzles
by Anany Levitin and Maria Levitin
Oxford Press
250 pages, SOFTCOVER-$15.00, 2011
Review by William Gasarch gasarch@cs.umd.edu

1 Introduction

In Math there is a some distinction between between *recreational* and *serious* mathematics. The main distinction seems to be how much prior knowledge you need. If a problem requires you to know Algebraic Geometry then it is probably not recreational. Beyond that it hard to say— there are some math problems that only require one clever trick, but that's only known once that trick has been found. And some fields of math stem from so-called recreational problems (e.g., probability).

Since computer science has been around for a much shorter period of time than mathematics the distinction between *recreational* and *serious* (is there such a thing?) computer science is far less clear. One criteria: the answer must be understandable by someone without too much prior knowledge. If the solution begins *first take a z-transform* then the problem is not recreational.

The book *Algorithmic Puzzles* is mainly a collection or 150 algorithmic puzzles, hints on how to solve them, and then full solutions. Are these recreational or serious? Very hard to say. Some of them I've seen before in puzzle books. For example the Wolf-Goat-Cabbage problem. Others seem like a good way to introduce a serious topic. For example:

Jack bets Jill that she can do the following trick. Jack will recite 99 different numbers from 1 to 100 in a random order and she Will be able to name the only number in that range that he will have missed. What is the best way for Jill to do the trick? Of course, she will have to perform the task in her head, without taking notes.

This problem is an excellent introduction to the important field of streaming algorithms.

2 Summary

The book begins with a short tutorial on algorithmic paradigms. Some are quite familiar (e.g., dynamic programming) while others you may have seen but not realized they can be considered as paradigms (e.g., transform and conquer).

Then the book has 150 problems divided into three categories of Easy/Medium/Hard. It is hard to really tell if a problem is easy, medium, or hard. Some of the easy ones I found hard, and some of the hard ones I had already seen so I found easy (that's cheating!). I suspect that many of them will be unfamiliar to the readers of this column. The puzzles are numbered simply 1,2,3,...,150 which makes them easy to find. They are not divided into types of problems (e.g., 'graph problems'). This is a plus since sometimes you don't know a problem is a graph problem until after you solve it.

The next section is hints. If you are stuck but don't quite want to look at the solution you can look at a hint. The last section is solutions.

Some of the problems are really math problems in disguise, in that the algorithm part is easy to carry out, but the proof that it works requires some reasoning. Here is one:

[18]

Let (x_1, y_1), ..., (x_n, y_n) *indicate the locations of* n *houses. Devise and algorithm to find a location* (x, y) *that minimizes the sum of the Manhattan distances from each house.*

Some of the problems are really math problems *NOT* in disguise:

Find all values of n *such that the* $n \times n$ *chessboard with two missing squares of opposite colors can be tiled with* 2×1 *dominoes.*

Once I saw the solution to the following problem I was reminded of priority arguments in computability theory (I doubt this was the authors intention):

Consider the following one-person card game. It is played with 13 cards numbered 1,2,3,...,13. The cards are shuffled. Then the top card is revealed. If it is a 1 then the game is over. If it is an $n \geq 2$ *then the top* n *cards are removed from the deck and put back in reverse order. Does the game always halt after a finite number of iterations?*

The variety of problems is impressive: easy/median/hard, algorithmic/mathematical, classic/variant on classic/seems entirely new, graphs/non-graphs.

3 Opinion

This book is a delight. The puzzles really require very little background knowledge. The tutorial at the beginning will help but often is not needed. Anyone who goes through this entire book will come out of it knowing A LOT of algorithmic— Tricks? Techniques? It is hard to tell the difference. But recall the saying:

1. If it works ONCE it's a trick.
2. If it works TWICE it's a technique.
3. If it works THRICE it's a method.

This book has problems of all three types. This is yet one more way that the book has a variety of problems- some need tricks, some need techniques, and some need methods.

Who should buy this book? Anyone in computer science or in a field of math that relates to computer science should buy it. Laypeople who like puzzles may also like it especially since the answers are in the back. I also note that the low price makes me NOT say *have your library buy it.* You can buy this one yourself.