The Book Review Column[1]

by Nicholas Tran
Department Mathematics & Computer Science
Santa Clara University
Santa Clara, CA 95053
ntran@scu.edu

Three new releases in data science, cryptography, and automata await expert opinions, as well as a recent ACM Book series title on approximation of Nash equilibrium. Dear readers, these books are *not* going to review themselves; please consider sharing your perspectives with the community.

1 This column

Sarvagya Upadhyay pores over *Property Testing: Problems and Techniques* by Arnab Bhattacharyya and Yuichi Yoshida, a new comprehensive survey of this fast advancing field that has a lot to offer to its targeted audience of graduate students and researchers.

Algorithms Illuminated, Omnibus Edition by Tim Roughgarden is another great textbook on the subject that I have reviewed in this column. Honed over a decade on a large-scale and diverse audience, it delivers on its promise to teach the basics of algorithms in the most accessible way possible.

2 How to contribute

Please contact me to write a review! Either choose from the books listed on the next page, or propose your own. The latter is preferred and quicker, as I can ask the publisher to send it directly to you.

[1] ©2023 Nicholas Tran

BOOKS THAT NEED REVIEWERS FOR THE SIGACT NEWS COLUMN

Algorithms & Complexity

1. Rubinstein, A. (2019). *Hardness of Approximation between P and NP* (ACM Books). Morgan & Claypool.

2. Knebl, H. (2020). *Algorithms and Data Structures: Foundations and Probabilistic Methods for Design and Analysis*. Springer.

3. Hidary, J. D. (2021). *Quantum Computing: An Applied Approach* (2nd ed.). Springer.

4. Murlak, F., Niwiński D., & Rytter, W. (2023). *200 Problems on Languages, Automata, and Computation*. Cambridge University Press.

Data Science

1. Amaral Turkman, M., Paulino, C., & Müller, P. (2019). *Computational Bayesian Statistics: An Introduction* (Institute of Mathematical Statistics Textbooks). Cambridge University Press.

2. Nakajima, S., Watanabe, K., & Sugiyama, M. (2019). *Variational Bayesian Learning Theory*. Cambridge University Press.

3. Hrycej, T., Bermeitinger, B., Cetto, M., & Handschuh, S. (2023). *Mathematical Foundations of Data Science*. Springer.

Cryptography and Security

1. Oorschot, P. V. C. (2020). *Computer Security and the Internet: Tools and Jewels* (Information Security and Cryptography). Springer.

2. Tyagi, H., & Watanabe, S. (2023). *Information-Theoretic Security*. Cambridge University Press.

Combinatorics and Graph Theory

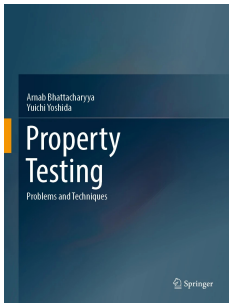
1. Golumbic, M. C., & Sainte-Laguë, A. (2021). *The Zeroth Book of Graph Theory: An Annotated Translation of Les Réseaux (ou Graphes)—André Sainte-Laguë (1926)* (Lecture Notes in Mathematics). Springer.

2. Beineke, L., Golumbic, M., & Wilson, R. (Eds.). (2021). *Topics in Algorithmic Graph Theory* (Encyclopedia of Mathematics and its Applications). Cambridge University Press.

Programming etc.

1. Sanders, P., Mehlhorn, K., Dietzfelbinger, M., & Dementiev, R. (2019). *Sequential and Parallel Algorithms and Data Structures: The Basic Toolbox*. Springer.

Review of[1]



Property Testing: Problems and Techniques

Arnab Bhattacharyya and Yuichi Yoshida
Published by Springer
Hardcover, 427 pages, $99.99

Review by
Sarvagya Upadhyay (supadhyay@fujitsu.com)
Fujitsu Research of America
350 Cobalt, Sunnyvale CA 94085, USA

1 Overview

Property testing concerns with testing whether an input object satisfies a certain property or not. Since it is a decision problem, the collection of potential input instances are partitioned into two sets: one set that satisfies the property, and the other set that is far from satisfying the property under a suitable distance measure. The set of instances which are not in these two cases is non-empty; however, they are promised not to be an input to the testing algorithm.

The primary goal of property testing is to design highly efficient testers for a given property. Such testers will hopefully read a part of the input and decide whether the property is satisfied or not. Since the algorithm does not read the whole input, it will err on certain cases. A tester reading the entire input will easily determine whether the property is satisfied or not. So the non-trivial aspect of property testing is to design either a sub-linear query algorithm for testing or demonstrate a non-trivial lower bound on the query complexity.

Since its inception and connections with several fields in theoretical computer science, the field has grown tremendously. This book is an attempt to survey the current state-of-the-art in this field. While there are topics that have not been covered in this book, I believe that the book has lot to offer for readers interested in property testing.

2 Summary of Contents

The book is divided into three parts. The first part contains two chapters that form the motivation and basic technical tools required to go through the book. The second part contains six chapters focusing on basic results in property testing. The final part contains five chapters that are geared towards advanced topics and results in property testing. A summary of each chapter in this book is given below. The few mathematical notations employed below are directly from the book.

Chapter 1: Introduction This chapter gives an introduction to property testing. It starts with a problem of monotonicity testing and discusses how we can test this property. While giving the testing algorithm, the authors highlight the naïve approaches and give counterexamples to why those approaches are not the best possible way to test the property. This is followed by a brief section motivating property testing. The rest of the chapter focuses on introducing property testing formally, the different aspects of property testing in view of its formal definition, variations of property testing, and formal proof of the monotonicity tester introduced in the first section. Finally, the chapter finishes with connection to different sub-fields of theoretical computer science and an organization of the book.

Chapter 2: Basic Techniques The second chapter is devoted to tools and techniques that are employed frequently in the book. It starts with illustrating analysis techniques by introducing few simple query problems. The second section of this chapter introduces gap-preserving local reductions between two properties, an important construct which is useful in proving lower bounds via reductions. The final section introduces Yao's minimax principle in the context of property testing. The authors show the power of minimax principle by demonstrating lower bounds for several property testing problems.

Chapter 3: Strings This chapter studies properties on strings. It covers four topics: palindromes, a slightly more involved variant of palindromes termed as two palindromes, the Dyck language, and monotonicity and permutation-freeness. Unsurprisingly, the tester for palindromes is very simple and the analysis not too hard. The authors' illustration of key points of the proof will be very illuminating to beginners in the field. The case of two palindromes is understandably highly non-trivial, and the authors provide a testing algorithm with almost optimal query complexity. They do this by showing an almost matching lower bound as well. The section on Dyck language is somewhat difficult to follow and requires a lot more attentive reading. The final section shows a non-adaptive tester for monotonicity of strings.

Chapter 4: Graphs in the Adjacency Matrix Model As the name of the chapter suggests, the focus is on representing graph by its adjacency matrix and analyze different properties of the underlying graph. The chapter starts with a very brief review on the basics of graph theory in the adjacency matrix model. There are several types of graph partitioning and cut problems whose property testing analogues have been studied in the literature. This forms one of the two main sections of the chapter. The other main section discusses the property of subgraph freeness. The two types of subgraphs considered are a square and a triangle. Apart from this, the chapter gives a very brief overview of additional topics involving digraphs and hypergraphs, graph isomorphism testers, and connection between testing algorithms and optimization problems in graph theory.

Chapter 5: Graphs in the Bounded-Degree Model This chapter focuses on graphs where each vertex has constant number of edges (hence, the name bounded-degree). In other words, the graph is sparse and the adjacency matrix representation is an overkill of resources. The first property testing problem considered in this model is the subgraph freeness. There is another section devoted to subgraph freeness when the subgraph is a cycle. In between these two sections, the chapter focuses on connectivity properties of graphs. All these properties can be tested by constant number of queries. The next two sections focus on testing graph colorability, and demonstrate that

the number of queries require is at least $\Omega(\sqrt{n})$ (for 2-colorability) and $\Omega(n)$ (for 3-colorability), where n is the number of vertices in the graph. The final two sections devote to developing constant-time randomized approximation algorithms for certain types of graph optimization problems.

Chapter 6: Functions over Hypercubes This chapter focuses on functions over hypercubes (i.e., functions of the form $f : \{0,1\}^n \to R \subseteq \mathbb{R}$). Of primary interest is the case when the range is Boolean. Studying such functional properties were the starting point of the field of property testing and have been widely used in inapproximability results, coding theory, and program checking. The chapter starts with testing monotonicity of Boolean functions and a brief review of Fourier analysis. Then the chapter focuses on linearity testing and testing juntas. The lower bounds are considered next by showing that testers for several function properties can be used to design two-party communication protocols, and then leverage known lower bounds in communication complexity to show query lower bounds on testers. The chapter's next section focuses on additional topics in function testing.

Chapter 7: Massively Parameterized Model This chapter focuses on a model where a tester requires a large number of parameters (usually a function of the input size) to test a specified property. The first problem discussed in this framework is testing bipartiteness (or 2-colorability) and how it admits a constant-query algorithm. Then the chapter focuses on monotonicity testing of a labeling function $f : V \to 0, 1$ on a directed acyclic graph. The authors show a sub-linear query upper bound. The focus on the next section is on a type of property testing problem where no sub-linear query algorithm exists to test the property. The final section discusses constraint satisfaction problems (CSPs). The main focus is on sub-linear testing algorithm for 2-SAT and linear lower bound on 3-SAT .

Chapter 8: Vectors and Matrices over the Reals This chapter is somewhat different from the previous chapters in the sense that it deals with vectors and matrices over real numbers. The first topic discussed in this chapter is to test whether a given matrix is low-rank or not. The next section considers the problem of testing whether a given vector belongs to a low-dimensional subspace. The section gives both an upper bound and a lower bound on the query complexity of one-sided testers. The final section devotes to additional topics related to matrices and vectors.

Chapter 9: Graphs in the Adjacency Matrix Model: Characterizations via Regularity Lemma This chapter takes a general view on graph property testing in the adjacency matrix representation. It defines few notions of graph property and characterizes the query complexity based on the notion. For instance, the first theorem of this chapter states that every monotone property of a graph (if the property holds for the graph, then it holds for every subgraph too) is testable with constant queries. The next section defines the notion of hereditary property and shows that hereditary graph properties are also testable with constant number of queries. The next section introduces the notion of oblivious tester and provides a characterization of a property being constant-query testable by an oblivious tester. The final section provides another characterization of existence of constant-query oblivious tester for a graph property.

Chapter 10: Graphs in the Bounded-Degree Model: General Testability Results via Matroid Theory and Graph Minor Theory In this chapter, the emphasis is on graph prop-

erties in the bounded-degree models. The main goal of this chapter is to derive conditions under which a property is constant-query testable. The chapter studies two classes of properties: (i) properties that are closed under edge addition, and (ii) properties that are closed under edge-removal. In the former case, the authors introduce a notion of (k, ℓ)-fullness. This property encompasses several graph properties, and the authors show a constant-query property testing algorithm for (k, ℓ)-fullness. In the latter case, the authors consider minor-closed properties (properties closed under subgraph and edge-contraction) and again show a constant-query property testing algorithm. In both scenarios, the tester has two-sided errors.

Chapter 11: Affine-Invariant Properties of Functions As the title of the chapter suggests, the main focus of this chapter is on properties that are invariant under affine transformation on vector spaces over finite fields ($\mathbb{F}^n$). The chapter defines the notion of affine-invariant subspace-hereditary property which roughly says that the affine-invariant property restricted to any affine subspace remains affine-invariant. The rest of the chapter is devoted to showing that subspace-hereditary property are constant-query testable with one-sided error.

Chapter 12: Linear Properties of Functions This chapter focuses on linear properties $\mathcal{P}$ of functions of the form $f : D \to \mathbb{F}$. The property set $\mathcal{P}$ is linear if for any $f, g \in \mathcal{P}$ and any $\alpha, \beta \in \mathbb{F}$, $\alpha f + \beta g \in \mathcal{P}$. Such properties have a strong connection with locally testable codes (LTCs). LTCs are one of the most important topics in coding and complexity theory simply because they satisfy good error-correcting properties and have been central to the development of most constructions of probabilistically checkable proofs (PCPs). The first section gives an overview of coding theory followed by a section on a generic methodology of testability of linear codes. The next two sections are devoted to low-degree testing over finite fields (whether a polynomial is of low-degree or far from it). The next section deals with testing membership in a tensor product codes, and the following section focuses on testability of lifted codes. The emphasis of final section is on discussing constructions of locally testable codes with good error-correcting properties.

Chapter 13: Massively Parameterized Model: Classification of Boolean CSPs The focus of this chapter is on Boolean CSPs from the perspective of massively parameterized model. The focus is on characterization of Boolean CSPs with respect to the query complexity of their testers. The main tool used is universal algebra and gap-preserving local reductions between CSPs.

3 Evaluation and Opinion

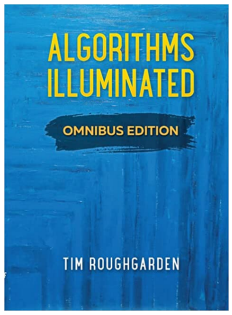
I read the book as a non-expert in this field. My knowledge of the field is rudimentary at best. I didn't read this book in one go and have kept coming back to it after every few weeks. This has allowed me to read it at a leisurely pace and internalize the topics covered.

My main issue with the book is with its presentation style. At various places, it seemed that I am reading a research paper rather than a book. For example, section headings such as "Proof of Lemma ..." are not illuminating. It can be difficult to follow through for someone with a tangential interest in the topic. It would help if the authors gave an overview of dependency between the chapters and a rationale for the organization. Finally, the third chapter makes use of techniques from graph theory, which makes me wonder if it could have been discussed after discussing graph properties.

Property testing as a field has evolved over the years. The analytical techniques used to show the correctness of testers have become quite sophisticated and varied. The authors have done very well to give a brief overview of the techniques (wherever required). Personally, the section focusing on higher-order Fourier analysis and Gowers norm was great to read. I would like to mention that mathematical rigor and maturity are very important for reading this book. The definitions and proofs are presented in full rigor, and they can overwhelm a casual reader. There are instances in the book where a series of definitions are presented one by one; in these scenarios, it is important to understand each definition carefully before moving on. There are also topics in the book where as a reader I wanted to know more. Reading Chapter 8 (property testing on real vectors and matrices) left me desiring for more on property testing in linear algebra.

Overall, I will recommend this book as a reference to advanced graduate students. For beginners, I will recommend starting with surveys and other books in property testing before diving into this book. I personally learnt a lot from this book specially in Chapters 7 and 12, and I firmly believe that this book will be a great reference to a researcher who wants to know more on this topic or related topics.

Review of[1]



Algorithms Illuminated, Omnibus Edition
by Tim Roughgarden
SoundLikeYourself Publishing, 2022
(distributed by Cambridge University Press)
690 pages, Hardcover, $60

Review by
Nicholas Tran (ntran@scu.edu)
Department of Mathematics & Computer Science
Santa Clara University

1 Overview

Algorithms Illuminated is the text version of a sequence of four Coursera algorithms courses taught by the author in the last ten years to a large-scale and diverse audience. This book covers topics in data structures (hash tables, Bloom filters, binary heaps, binary search trees, union-find, graphs), analysis techniques (asymptotic notations, the master method, expected values), design paradigms (divide-conquer, greedy, dynamic programming), and coping with hardness (NP-completeness, approximate algorithms, local search, integer programming and satisfiability solvers). Two appendices on proof techniques and probability, a short field guide to algorithm design, and an in-depth case study of the 2016-17 Federal Communications Commission's reverse auction of wireless spectrum are also included. The materials have been used to teach an undergraduate-level as well as a master's level course in algorithms at Stanford and other universities. Extensive learning resources (YouTube videos, slides, math supplements, test data sets, and discussion forums) are available at the eponymous web site.

Keeping with its single goal "*to teach the basics of algorithms in the most accessible way possible,*" the book maintains a conversational style and interacts effectively with the reader through the use of quizzes and footnotes sprinkled throughout the text where elaboration may be required or desired. The quizzes, whose answers with explanation appear a few pages later, force the reader to master a point needed in the subsequent discussion, while the footnotes provide optional additional insights and historical notes. Algorithms are described in very high-level pseudocode that omit implementational details to focus on the big computational picture. The standard treatment of covered topics at many places has been updated with modern problems, tweaked solutions, admonitions against common pitfalls, and references to latest developments. Each chapter ends with a summary, followed by a small number of problems classified into three types: routine, challenge, and programming. Solutions or hints are provided for all non-programming problems.

2 Summary of Contents

Part I: The Basics

Chapter 1 motivates the study of design and analysis of algorithms with a gentle development of Karatsuba multiplication and merge sort. The conventions of the field are presented: how to describe a problem, how to express algorithms in pseudocode, how to count the number of primitive operations, what theorems, lemmas, Q.E.D., and "fast" mean, and why focusing on analyzing the worst-case and long-term behavior of algorithms yields the right balance between mathematical tractability and accurate prediction of running times.

Chapter 2 explains asymptotic notations in English and pictorially with familiar examples and then formally defines them mathematically with additional examples. Common pitfalls are mentioned, such as using O to mean Θ and not realizing that 2^n and 4^n have different growth rates.

Chapter 3 illustrates the divide-and-conquer design paradigm with three classic problems: counting inversions in an array, matrix multiplication, and finding the closest pair of points in the plane. Exhaustive solutions to these problems are presented first, followed by improvements based on the divide-conquer-combine paradigm. Asymptotic analysis of these recursive algorithms is given informally, to be made precise using the master theorem in the next chapter. The closest-pair algorithm given in this chapter is a nice variation of the standard treatment that presorts the array of points twice (by x-coordinates and independently by y-coordinates) to achieve a clean conquer step.

Chapter 4 states the master theorem and applies it to recursive algorithms discussed in the previous chapters among others. A carefully explained proof of the theorem is also provided.

Chapter 5 is an in-depth study of quicksort implemented using Lomuto's version of the partition algorithm. Randomized algorithms are introduced, and analysis of the expected running time of randomized quicksort using indicator random variables is given. The chapter ends with a proof of the lower bound of $\Omega(n \log n)$ for comparison-based sorting algorithms and a discussion of faster sorting algorithms such as bucket sort, counting sort, and radix sort that are not comparison-based.

Chapter 6 describes the straightforward application of the partition algorithm to solve the selection problem (commonly known as quickselect) and shows that the expected running time of randomized quickselect is $O(n)$. The chapter ends with a detailed description and analysis of the $O(n)$ median-of-medians algorithm for selection.

Part II: Graph Algorithms and Data Structures

Chapter 7 defines graphs and graph terminology, provides examples of graph applications, and compares the adjacency matrix and adjacency list representations of graphs.

Chapter 8 proves the correctness of the generic graph search algorithm and then implements and analyzes breadth-first search and depth-first search as special cases. Finding least-edge paths and connected components are discussed as applications of breadth-first search; similarly, finding topological ordering and strongly connected components with Kosaraju's algorithm are discussed as applications of depth-first search. The chapter ends with a discussion of structural properties of strongly connected components of the Web graph, whose vertices are web pages and whose edges are hyperlinks.

Chapter 9 defines the single-source shortest path problem and gives some examples of its applications. Dijkstra's algorithm is then presented in data structure-agnostic pseudocode, shown

to run in polynomial time, and proven correct. The need for a correctness proof is motivated by two observations that i) Dijkstra's algorithm fails on some graphs with negative weights; and ii) adding a constant positive offset to all edge weights changes the shortest paths.

Chapter 10 introduces the heap data structure, its supported operations and their running times, and its applications. A heap-based version of Dijkstra's algorithm is then presented and analyzed. The chapter ends with an array-based implementation of heaps.

Chapter 11 defines the search tree data structure, its supported operations and their running times for the balanced variant. Implementation of basic binary search trees is discussed, with a hint on how to correct imbalance using rotations.

Chapter 12 defines the hash table data structure, its supported operations and their typical running times, and its applications. Implementations of hash tables using chaining and open addressing with linear probing and double hashing are explained, and their performances are analyzed in terms of the load factor. There is a nice discussion of pathological data sets causing poor performance that exist for any hash function, and a brief mention of universal hashing functions as basic good choices for everyday hashing. The chapter also defines Bloom filters, their supported operations and applications. It ends with discussions of their implementation as well as heuristic analysis of their performance.

Part III: Greedy Algorithms and Dynamic Programming

Chapter 13 introduces the greedy algorithm design paradigm and illustrates it with a scheduling problem that seeks to minimize the sum of weighted completion times. Two appealing greedy strategies that yield conflicting outcomes are compared to motivate the need for a correctness proof, which is based on an exchange argument.

Chapter 14 motivates the problem of finding optimal prefix-free code and rephrases it as finding a Σ-tree with the minimum average leaf depth. Huffman's algorithm is then presented in pseudocode and analyzed, followed by a discussion of speeding it up using either a heap or two queues. The chapter ends with a careful induction proof of correctness for Huffman's algorithm.

Chapter 15 defines the minimum spanning tree problem and presents and analyzes Prim's algorithm, first in high-level pseudocode, and then enhanced with the use of heaps. A proof of correctness based on the minimum bottleneck property is provided; the more traditional proof based on the cut property is covered in the exercises. Next, it presents and analyzes Kruskal's algorithm, first in high-level pseudocode, and then enhanced with the use of the union-find data structure. A tree-based implementation of union-find that uses union by rank but not path compression is sketched. A proof of correctness of Kruskal's algorithm based on the minimum bottleneck property is provided. The chapter ends with a discussion of the clustering problem and how to solve it using Kruskal's algorithm.

Chapter 16 gives a fresh introduction to the dynamic programming design paradigm and illustrates it with two problems: weighted independent set and knapsack. After showing that the problems are not susceptible to greedy strategies, it enumerates a three-step recipe for designing dynamic programming algorithms and points out the differences between this and the divide-and-conquer recipe. It also includes an amusing anecdote on the origin of the somewhat confusing paradigm name.

Chapter 17 reinforces the three-step recipe for designing dynamic programming algorithms with two classic problems: sequence alignment and construction of binary search trees with minimum average search time. The characteristic activities of finding a recurrence for the dependence

on optimal solutions to subproblems, caching, and reconstructing an actual solution that achieves the optimal objective function value are illustrated in detail.

Chapter 18 presents Bellman-Ford and Floyd-Warshall algorithms, which are dynamic programming solutions to single-source and all-pairs shortest path problems respectively. Again, the dependence on optimal solutions to subproblems is elaborated with examples. Notable discussions in this chapter include two ways to define shortest paths in the presence of negative-weight cycles and application of Bellman-Ford algorithm to internet routing.

Part IV: Algorithms for NP-hard Problems

Chapter 19 brings forth the existence of NP-hard problems such as traveling salesman with the following informally defined property: they have no known fast solutions, and if indeed no such solutions exist, then verifying a solution to a problem is fundamentally easier than finding one from scratch. Three strategies are suggested for coping with NP-hard problems: compromising on generality, correctness, or speed of their solutions. Several useful asides briefly address randomized and quantum solutions to NP-hard problems, the exponential time hypothesis, and common rookie mistakes.

Chapter 20 illustrates examples of approximately correct solutions to three NP-hard problems: schedule makespan minimization, maximum coverage, and influence maximization. These are fast greedy algorithms that produce solutions guaranteed to be within a constant factor of the optimal solutions. For traveling salesman, even existence of such approximately correct solutions would imply P = NP; high-quality heuristics based on local search are discussed instead.

Chapter 21 presents dynamic-programming solutions that perform better than the brute-force solutions to two NP-hard problems: traveling salesman and minimum-cost k-path. More generally it is shown how to use state-of-the-art solvers for mixed integer programming and satisfiability to obtain optimal solutions for medium-size instances of a variety of NP-hard problems.

Chapter 22 presents examples of NP-hard proofs. Starting with the assumption that 3-SAT is NP-hard, it is shown that independent set, hamiltonian path, traveling salesman, subset sum are NP-hard as well through a series of reductions.

Chapter 23 gives formal definitions of NP, NP-hard, and NP-complete problems. It also discusses the (strong) exponential time hypothesis and the unexpected relationship between it and sequence alignment.

Chapter 24 is an in-depth case study of the reallocation of spectrum previously licensed to television broadcasters to wireless broadband providers. Between 2016 and 2017 the FCC implemented a reverse auction to determine which television stations would release their spectrum licenses and at what price, an NP-hard problem even in its simplified form. Drawing together concepts and results discussed previously, this chapter walks through the details of a greedy heuristic solution which itself depends on multiple state-of-the-art satisfiability solvers working together plus various optimizing techniques. It ends with a discussion of the auction's technical and economic outcomes.

3 My opinion

This book sets high and specific goals for itself in the preface and to a high degree delivers on its promises. To make the materials accessible, it keeps a laser focus on algorithmic ideas in the text and renders analysis and implementation in broad strokes. The discussions and the footnotes in the book do feel like chats you would have with an expert colleague on their favorite topics. The book

does a great job in presenting greatest hits of algorithms, from the time-tested such as linear-time selection, randomized algorithms and their analysis, fast solutions for minimum spanning trees and shortest paths, to new ones such as influence maximization and fine-grained complexity. Finally, the chapter on the FCC spectrum reverse auction is a unique and convincing demonstration of bringing the algorithmic tools discussed in this book to bear on a real-world difficult problem.

I particularly enjoy the book's positive attitude about coping with NP-hardness and discussion on the different strategies including using integer programming and satisfiability solvers. Overall, I think this book is a modern, accessible but not lightweight textbook on algorithms that will be greatly enjoyed by a wide range of undergraduate and beginning graduate students and practitioners who want to learn how to design algorithmic solutions to problems in their areas.