Review of *"Discrete Structures with Contemporary Applications"* by Alexander Stanoyevitch, CRC Press, 2011

ISBN: 978-1-4398-1768-1

M. Frederic Ezerman, CCRG, Nanyang TU, Singapore

September 2, 2014

1 Summary of the Review

This is a review of Stanoyevitch's modern introductory textbook on discrete mathematics and their contemporary applications.

The author's stated pupose is to provide a formal introduction to abstract objects essential in computing and exposure to an assortment of exciting applications. The tome is big, nearly reaching 1000 pages. In this review we briefly go through the main content of the chapters while noting down interesting exercises. Impressions on the book and possible utilization for different groups of potential audience conclude the review.

2 Summary of the Book

The book has ten chapters and three appendices. The exercises are divided into roughly three types. Some are meant to be done while reading the text, hence, their solutions are provided in the appendix. Then there are end-of-section exercises of varying degrees of difficulty. Sometimes these exercises are used to introduce new topics not covered in the *lecture* part of the text. The third type consists of computer exercises to be done in view of sharpening student's ability to perform scientific programming.

2.1 Chapter 1: Logic and Sets

Chapter 1 is foundational. Many students would come to this material thinking they know the topics well yet it is worth reviewing and working on the exercises for the readers (EFRs). Treatment on *logical operators*, *quantifiers* and *sets* constitute the sections in this chapter.

2.2 Chapter 2: Relations & Functions, Boolean Algebra, & Circuit Design

The text delves into *relations* and *functions* as well as (partial) orders. On the topic of *Boolean algebra*, it is shown how we build on Shannon's seminal work to design general electronic *circuits* with elements corresponding to basic Boolean operations. This is a topic that one does not often come across in a textbook on discrete structures so the treatment here is a plus.

Notable exercise: Exercise 2.1.32 on page 75 on cycles and orbits of a permutation.

2.3 Chapter 3: The Integers, Induction, and Recursion

Most of the coverage is a staple in many textbooks. Several snippets caught my interest though. The proof of Hall's marriage theorem by *induction* highlights the first section. Students with interest in the performance and comparison of algorithms can find plenty of problems and case studies in finance. The discussion on *iterative* versus *recursive* programs is clear and helpful.

The last section discusses the Prime Number Theorem (PNT), fast modular exponentiation, Fermat's Little Theorem and Euler's Theorem. They will come in handy in the discussion on the RSA cryptosystem in Section 4.5. One also finds a discussion on the differences between *floating point* and *symbolic computation* and their implications in translating algorithms in cryptology to working programs. The Miller-Rabin *primality test* wraps up the chapter and is presented as Appendix 3.3.

Notable exercises: Exercise 3.1.36 on page 124 and many of the computer exercises in Section 3.3.

2.4 Chapter 4: Number Systems

Chapter 4 splits the treatment into five sections. The first discusses the representations of integers in different bases. The second is on *modular arithmetic* and comes with plenty of nice exercises. Section 3 is about *matrices.* For me the highlight is on the treatment of the Strassen's method for fast matrix multiplication in the exercises. The accumulation of round-off errors in floating point arithmetic is explained in Section 4. Section 5 covers *public key cryptosystems* (PKCs) and shows how the math background developed is now sufficient to completely describe some of the most widespread PKCs: RSA, ElGamal, and Knapsack. The Diffie-Hellman key-exchange and discrete logarithm problems are expounded.

Notable exercises: Exercises 4.3.35–36, starting from Page 260 on how to multiply matrices efficiently. For those teaching an introductory course in cryptography, the computer exercises in Section 4.5 can be handy.

2.5 Chapter 5: Counting Techniques, Combinatorics, & Generating Functions

Quite a standard treatment. Section 1 is on *principles of counting*, which are then used in Section 2 to discuss *permutation*, *combination*, and the *binomial theorem. Generating functions*, treated as formal power series, form the third section.

Notable exercises: Exercises 5.3.27–32 on pages 365–366 and appendix to Section 5.3 on the application of computational game theory in weighted democracies.

2.6 Chapter 6: Discrete Probability and Simulation

Introduction to *discrete probability* opens the chapter, giving Kolmogorov's formulation that links probability and set theory. Bayes' formula is explained in considerable details. The second section discusses *random numbers*, *random variables*, and basic *simulations.* To do simulation, it's important to generate random permutations and subsets. Students are led to learn that probabilities can be interpreted as long term relative frequencies while expectations are seen as long term relative averages.

2.7 Chapter 7: Complexity of Algorithms

The chapter begins by mentioning two main aspects of technology: advances in hardware which is more or less still following Moore's law, and advances in algorithms which are translated into better softwares. The second is less predictable yet much more impactfull.

Section 1 is about *searching and sorting algorithms* with four sorting algorithms discussed in the main text plus one in the exercises. Two of them (linear search and binary search) are not efficient. The other two (selection sort and bubble sort) perform better and have equal efficiency. The last, merge sort, is shown to be optimal. The examples are illustrative enough to show differences in efficiency and the accompanying consequences.

Section 2 is on *growth rates of functions.* The analytic tools needed come from Sections 3.1, 3.2 and Chapter 5. Three types of complexity analysis (worst-case, average, and best-case) are performed on the algorithms of Section 7.1. Here the definitions of the class P and NP as well as the notion of NP complete and NP hard problems are discussed.

Notable exercises: Many of the computer exercises in Section 7.2 contain programming assignments to compare and contrast the performance of the search algorithms. Exercises 7.2.23 to 25, establish the complexity of the Euclidean Algorithm.

2.8 Chapter 8: Graphs, Trees and Associated Algorithms

This chapter is quite dense and important since graphs are ideal tools for modeling discrete problems and objects. The first section covers definitions and concepts and mentions *graph isomorphism* as a major open problem. Among the concepts discussed are *degrees* and *regular graphs* (leading to the handshaking theorem), *bipartite graph*, and *degree sequence*. Important families of simple graphs such as complete graph, cycle, and hypercube are explained. Some yardsticks to rule out isomporphism of simple graphs treated here include degree sequences agreement and the subgraph test, either of the graphs themselves or of their complements. Various choices of presenting graphs on computers depending on actual scenarios are explained. The section also contains three graph models for optimization problems, namely the shortest path in network, maximum matching, and the Traveling Salesman Problem (TSP).

Section 2 is almost as packed as the first one. Here the topics treated are *paths*, *connectedness*, and *distance* in (di)graphs which measures traversability. Quite a number of highly applicable scenarios are treated with considerable details here. Some examples include finding optimal locations for central hubs in a large network and how to use the powers of a graph's adjacency matrix to determine various distance measures plus how to use edge and vertex cuts in a connected (di)graph to measure the stability and reliability of a network.

Trees, which are simple connected graphs without cycles, are the focus of the third section. The coverage includes basic concepts, rooted trees and binary trees, models with rooted trees, applications in tournaments and the spread of information. This section's appendix discusses rooted trees for data compression and coding.

Notable exercises: Several exercises in Section 8.2 contain interesting details on how paths or cycles can be useful in determining isomorphism. Exercises for Section 8.3 on how to pinpoint counterfeit coin(s) in least number of weighing possible for various scenarios are quite delightful to try.

2.9 Chapter 9: Graph Transversal and Optimization Problems

This chapter is all about modeling practical problems using simple graphs. It's important to have efficient solution(s) and to know what is possible in case the problems are in NP. The three main topics treated are 1) two graph traversal problems, 2) some general optimization problems, and 3) network flow problems.

The graph traversal problems discussed in the first section are Euler and Hamilton *paths* and *tours*. A complete characterization of graphs admitting Euler path and tour is given. An algorithm to find such objects explicitly is presented. A complete characterization of Euler paths and tours in digraphs is also given. A Hamilton path is a simple path in a graph G that passes through every vertex exactly once. Some sufficient and necessary conditions for a graph to be Hamiltonian are established.

Section 9.2 is about *tree growing* and graph optimization algorithm, centering on the TSP, which is considered to be the epitome of discrete math problems and is one of the most famous problem. TSP is computationally very difficult yet it is very applicable *e.g.* in robotics, DNA mapping, routing, manufacturing of circuit boards and chips. As an edge-weighted graph problem it can be expressed as a problem of finding a Hamilton tour of minimum possible weight. Approaches to TSP discussed here are the so-called *insertion heuristics.*

Section three deals with *network flows* with the objective of optimizing the flow of objects through edge-weighted directed network. There are numerous applications: traffic management, scheduling and project completion estimation. The Ford-Fulkerson Algorithm (FFA), which is based on the max flow min cut theorem, is used to prove the Hall's marriage theorem. A very detailed proof of FFA is given.

Notable exercises: In Section 9.1 there are at least three sets of interesting exercises: 1) Fleury's Algorithm for the construction of Euler's paths and tours, 2) a proof of Ore's Theorem, and 3) another sufficiency condition for a graph G to be Hamiltonian. Section 9.2 also has some important sets of exercises: 1) a proof of Prim's Algorithm is outlined in Exercise 9.2.27, 2) upgrades vs maintenance costs comparisons in several practical contexts, and 3) a proof of Kruskal's Algorithm.

2.10 Chapter 10: Randomized Search and Optimization Algorithms

Randomized search and optimization algorithms are general multi-purpose algorithms that 1) use *randomness* as a key ingredient, and 2) are specifically designed for searches and optimization. They usually find good quality, albeit not necessarily optimal, solutions of problems believed to be intractable. They search for some discrete structure with distinguishing min or max properties. There are two sections in this last chapter. The first one explains general concepts, providing descriptions and examples of well-known families. The second focuses on *genetic algorithm* which is an optimization algorithm modeled on natural evolution. The randomness utilized in the search for optimal solution is associated to genetics. The use of computer is a must in this chapter since it is more practical and less theoretical.

It should be noted that aside from the main algorithms discussed at length in this chapter, three other prominent algorithms are also alluded to: 1) Tabu search that maintains a *tabu list* of elements in the discrete structure $\mathcal{D}$ to which the search is *not* permitted to be moved, 2) Ant Colony Optimization (ACO) that experiments through the search space by replicating the food foraging behavior of ants, and 3) Simulated Annealing which mimics the cooling of liquids or solids in physics. While they are not treated at length, the readers can find some useful historical notes and references for further reading.

Notable exercises: Exercise 10.1.9 on random search for maximum independent set and the eight queens problem as well as its generalization to n queens in Exercises 10.2 14–15 and 17–18.

2.11 The Appendices

Appendix A summarizes briefly how to read the *pseudocodes*. This is quite basic and those with a modicum of programming experience won't need it. Beginning students with no prior exposure to programming, however, may find this handy.

In Appendix B, solutions to all EFRs are provided . As per the intention of the author, EFRs are to help the readers gauge their mastery of the respective chapter's content in close reading. My take is that students and lecturers alike will find this appendix to be the most consulted of the three.

Answers (often in the form of outlines) to odd-numbered exercises form Appendix C.

3 What is the book like?

The book is a heavy tome. The size can be intimidating at first glance yet this textbook is excellent. The author clearly had put a lot of effort in presenting the topics clearly and as engaging as possible. His many years of teaching and mentoring clearly show. The more I read it through the more I like the book, especially how the exercises are so carefully selected and presented. For this reason alone, this book is worth keeping and using. There are ample notes, historical as well as biographical. My favorite is the one in Section 1.3.

There is no formal prerequisite although a pre-calculus course is assumed to have been completed by the readers. Discrete mathematics, being such a wide subject area, covers more topics than even this huge text does. This textbook is geared toward discrete structures in computing. It provides enough theoretical background to satisfy those with appetite for the more theoretical aspects yet contains a large amount of practical materials to cater to those with more applied and applicable mathematics bent.

Chapters 9 and 10 often feel like coming from a programming textbook not from a textbook in Discrete Math. Too much programming details. This may appeal more to students of algorithmics than to mathematics students preferring more concepts and ideas rather than programming details.

The topics covered should provide enough materials for 2 or even 3 semesters courses.

4 Would you recommend this book?

I heartily recommend this textbook and have been using it both in preparing teaching materials and in educating myself.

Instructors who regularly teach courses in discrete structures can find plenty of quality exercises readily usable for home works, tutorial discussions and exam material. Quite a number of them can be used as projects for further exploration by the students, either in groups or individually.

The level of difficulty makes this book suitable for undergraduate and beginning graduate students of mathematics and of computer science. Mathematics students will find the book sufficiently challenging to keep them interested while exposing them to how applicable discrete structures can be. Plenty of exercises requiring rigorous proofs and pointers at big open problems should whet their appetite to go deeper in the later stages of their training. Computer Science students or those primarily interested in the algorithmics will find enough materials to digest while getting exposed to the rigorous mathematics underlying the topics. Students seeking well formulated scientific programming exercises will find this book highly satisfactory.

5 Miscellaneous Notes

On page 40, the set $\mathbb{N}$ is defined to be $\{0, 1, 2, \ldots\}$. This is rather unusual since usually $0 \notin \mathbb{N}$. The author forgot to include [AlGrPo92] stated on page 181 as well as [Fo-Fu-56] and [Fo-Fu-62] mentioned in footnote 22 on page 697 in the list of references. Donald Knuth's bio is missing in Chapter 7. The title of the subsection on page 702 is missing the word *Cut* after *Minimum.* Theorem 10.1 on page 764 should read $R(3, 3) = 6$, **not** 9. There are several minor typographical errors which can be easily corrected.

The reviewer is a Research Fellow at Cryptology and Coding Theory Research Group, Division of Mathematical Sciences, Nanyang Technological University, Singapore, working mostly in the area of coding theory and cryptology. He is currently tutoring about 70 students in discrete mathematics taking the MH8300 (It's a discreetly discrete world) course at said university.