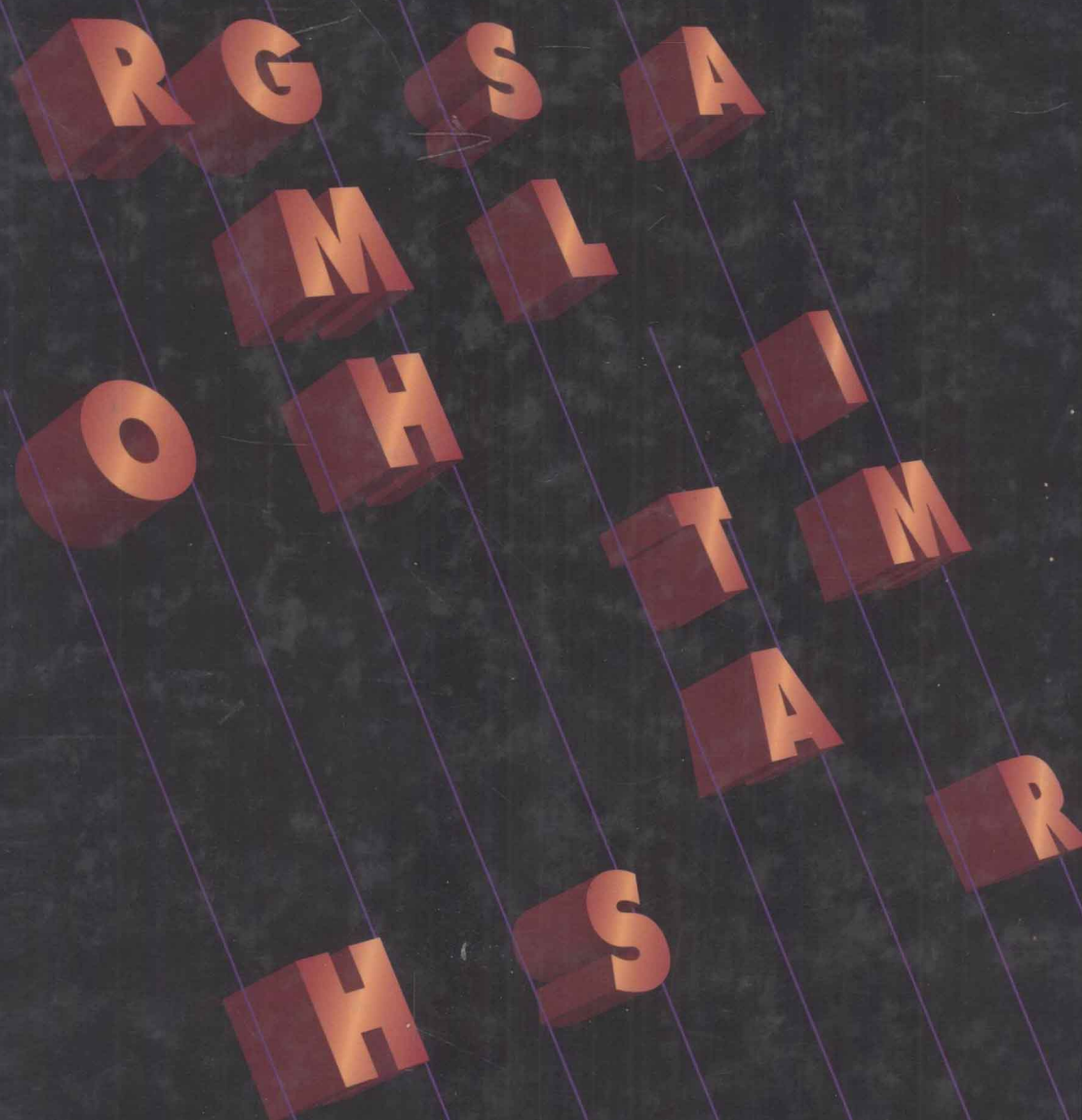
# Randomized Algorithms

Rajeev Motwani and Prabhakar Raghavan



# Randomized Algorithms

**Rajeev Motwani**
*Stanford University*

**Prabhakar Raghavan**
*IBM Thomas J. Watson Research Center*

CAMBRIDGE
UNIVERSITY PRESS

Published by the Press Syndicate of the University of Cambridge
The Pitt Building, Trumpington Street, Cambridge CB2 1RP
40 West 20th Street, New York, NY 10011-4211, USA
10 Stamford Road, Oakleigh, Melbourne 3166, Australia



First published 1995

Printed in United States of America

*Library of Congress Cataloguing-in-Publication Data*

Motwani, Rajeev.
Randomized algorithms / Rajeev Motwani, Prabhakar Raghavan.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-47465-5
1. Stochastic processes–Data processing. 2. Algorithms.
I. Raghavan, Prabhakar. II. Title.
QA274.M68 1995
004'.01'5192–dc20 94-44271

A catalog record for this book is available from the British Library.

ISBN 0-521-47465-5 hardback

TAG

The Stanford–Cambridge Program is an innovative publishing venture resulting from the collaboration between Cambridge University Press and Stanford University and its Press.

The Program provides a new international imprint for the teaching and communication of pure and applied sciences. Drawing on Stanford's eminent faculty and associated institutions, books within the Program reflect the high quality of teaching and research at Stanford University.

The Program includes textbooks at undergraduate level, and research monographs, across a broad range of the sciences.

Cambridge University Press publishes and distributes books in the Stanford–Cambridge Program throughout the world.

# Randomized Algorithms

# Preface

THE last decade has witnessed a tremendous growth in the area of randomized algorithms. During this period, randomized algorithms went from being a tool in computational number theory to finding widespread application in many types of algorithms. Two benefits of randomization have spearheaded this growth: simplicity and speed. For many applications, a randomized algorithm is the simplest algorithm available, or the fastest, or both.

This book presents the basic concepts in the design and analysis of randomized algorithms at a level accessible to advanced undergraduates and to graduate students. We expect it will also prove to be a reference to professionals wishing to implement such algorithms and to researchers seeking to establish new results in the area.

### Organization and Course Information

We assume that the reader has had undergraduate courses in Algorithms and Complexity, and in Probability Theory. The book is organized into two parts. The first part, consisting of seven chapters, presents basic tools from probability theory and probabilistic analysis that are recurrent in algorithmic applications. Applications are given along with each tool to illustrate the tool in concrete settings. The second part of the book also contains seven chapters, each focusing on one area of application of randomized algorithms. The seven areas of application we have selected are: data structures, graph algorithms, geometric algorithms, number theoretic algorithms, counting algorithms, parallel and distributed algorithms, and online algorithms. Naturally, some of the algorithms used for illustration in Part I do fall into one of these seven categories. The book is not meant to be a compendium of every randomized algorithm that has been devised, but rather a comprehensive and representative selection. The Appendices review basic material on probability theory and the analysis of algorithms.

We have taught several regular as well as short-term courses based on the material in this book, as have some of our colleagues. It is virtually impossible to cover all the material in the book in a single academic term or in a week's intensive course. We regard Chapters 1–4 as the core around which a course may be built. Following the treatment of this material, the instructor may continue with that portion of the remainder of Part I that supports the material of Part II (s)he wishes to cover. Chapters 5–13 depend only on material in Chapters 1–4, with the following exceptions:

1. Chapter 5 on Probabilistic Methods is a prerequisite for Chapters 6 (Random Walks) and 11 (Approximate Counting).
2. Chapter 6 on Random Walks is a prerequisite for Chapter 11 (Approximate Counting).
3. Chapter 7 on Algebraic Techniques is a prerequisite for Chapters 14 (Number Theory and Algebra) and 12 (Parallel and Distributed Algorithms).

We have included three types of problems in the book. **Exercises** occur throughout the text, and are designed to deepen the reader's understanding of the material being covered in the text. Usually, an exercise will be a variant, extension, or detail of an algorithm or proof being studied. **Problems** appear at the end of each chapter and are meant to be more difficult and involved than the **Exercises** in the text. In addition, **Research Problems** are listed in the Discussion section at the end of each chapter. These are problems that were open at the time we wrote the book; we offer them as suggestions for students (and of course professional researchers) to work on.

Based on our experience with teaching this material, we recommend that the instructor use one of the following course organizations:

- A comprehensive basic course: In addition to Chapters 1–4, this course would cover the material in Chapters 5, 6, and 7 (thus spanning all of Part 1).
- A course oriented toward algebra and number theory: Following Chapters 1–4, this course would cover Chapters 7, 14, and 12.
- A course oriented toward graphs, data structures, and geometry: Following Chapters 1–4, this course would cover Chapters 8, 9, and 10.
- A course oriented toward random walks and counting algorithms: Following Chapters 1–4, this course would cover Chapters 5, 6, and 11.

Each of these courses may be pruned and given in abridged form as an intensive course spanning 3–5 days.

## Paradigms for Randomized Algorithms

A handful of general principles lies at the heart of almost all randomized algorithms, despite the multitude of areas in which they find application. We briefly survey these here, with pointers to chapters in which examples of these

principles may be found. The following summary draws heavily from ideas in the survey paper by Karp [243].

**Foiling an adversary.** The classical adversary argument for a deterministic algorithm establishes a lower bound on the running time of the algorithm by constructing an input on which the algorithm fares poorly. The input thus constructed may be different for each deterministic algorithm. A randomized algorithm can be viewed as a probability distribution on a set of deterministic algorithms. While the adversary may be able to construct an input that foils one (or a small fraction) of the deterministic algorithms in the set, it is difficult to devise a single input that is likely to defeat a randomly chosen algorithm. While this paradigm underlies the success of *any* randomized algorithm, the most direct examples appear in Chapter 2 (in game tree evaluation), Chapter 7 (in efficient proof verification), and Chapter 13 (in online algorithms).

**Random sampling.** The idea that a random sample from a population is representative of the population as a whole is a pervasive theme in randomized algorithms. Examples of this paradigm arise in almost all the chapters, most notably in Chapters 3 (selection algorithms), 8 (data structures), 9 (geometric algorithms), 10 (graph algorithms), and 11 (approximate counting).

**Abundance of witnesses.** Often, an algorithm is required to determine whether an input (say, a number $x$) has a certain property (for example, "is $x$ prime?"). It does so by finding a *witness* that $x$ has the property. For many problems, the difficulty with doing this deterministically is that the witness lies in a search space that is too large to be searched exhaustively. However, by establishing that the space contains a large number of witnesses, it often suffices to choose an element at random from the space. The randomly chosen item is likely to be a witness; further, independent repetitions of the process reduce the probability that a witness is not found on any of the repetitions. The most striking examples of this phenomenon occur in number theory (Chapter 14).

**Fingerprinting and hashing.** A long string may be represented by a short *fingerprint* using a random mapping. In some pattern-matching applications, it can be shown that two strings are likely to be identical if their fingerprints are identical; comparing the short fingerprints is considerably faster than comparing the strings themselves (Chapter 7). This is also the idea behind *hashing*, whereby a small set $S$ of elements drawn from a large universe is mapped into a smaller universe with a guarantee that distinct elements in $S$ are likely to have distinct images. This leads to efficient schemes for deciding membership in $S$ (Chapters 7 and 8) and has a variety of further applications in generating pseudo-random numbers (for example, two-point sampling in Chapter 3 and pairwise independence in Chapter 12) and complexity theory (for instance, algebraic identities and efficient proof verification in Chapter 7).

**Random re-ordering.** A striking use of randomization in a number of problems in data structuring and computational geometry involves randomly re-ordering the input data, followed by the application of a relatively naive algorithm. After the re-ordering step, the input is unlikely to be in one of the orderings that is pathological for the naive algorithm. (Chapters 8 and 9).

**Load balancing.** For problems involving choice between a number of resources, such as communication links in a network of processors, randomization can be used to "spread" the load evenly among the resources, as demonstrated in Chapter 4. This is particularly useful in a parallel or distributed environment where resource utilization decisions have to be made locally at a large number of sites without reference to the global impact of these decisions.

**Rapidly mixing Markov chains.** For a variety of problems involving counting the number of combinatorial objects with a given property, we have approximation algorithms based on randomly sampling an appropriately defined population. Such sampling is often difficult because it may require computing the size of the sample space, which is precisely the problem we would like to solve via sampling. In some cases, the sampling can be achieved by defining a Markov chain on the elements of the population and showing that a short random walk using this Markov chain is likely to sample the population uniformly (Chapter 11).

**Isolation and symmetry breaking.** In parallel computation, when solving a problem with many feasible solutions it is important to ensure that the different processors are working toward finding the same solution. This requires isolating a specific solution out of the space of all feasible solutions without actually knowing any single element of the solution space. A clever randomized strategy achieves *isolation*, by implicitly choosing a random ordering on the feasible solutions and then requiring the processors to focus on finding the solution of lowest rank. In distributed computation, it is often necessary for a collection of processors to break a deadlock and arrive at a consensus. Randomization is a powerful tool in such deadlock-avoidance, as shown in Chapter 12.

**Probabilistic methods and existence proofs.** It is possible to establish that an object with certain properties exists by arguing that a randomly chosen object has the properties with positive probability. Such an argument gives no clue as to how to find such an object. Sometimes, the method is used to guarantee the existence of an algorithm for solving a problem; we thus know that the algorithm exists, but have no idea what it looks like or how to construct it. This raises the issue of *non-uniformity* in algorithms (Chapters 2 and 5).

## Conventions

Most of the conventions we use are described where they first arise. One worth mentioning here is the issue of *integer breakage:* as long as it does not materially affect the algorithm or analysis being considered (and the intent is unambiguous from the context), we omit ceilings and floors from numbers that strictly should be integers. Thus, we might say "choose $\sqrt{n}$ elements from the set of size $n$" even when $n$ is not a perfect square. Our intent is to present the crux of the algorithm/analysis without undue notational clutter from ceilings and floors. The expression $\log x$ denotes $\log_2 x$, and the expression $\ln x$ denotes the natural logarithm of $x$.

## Acknowledgements

This book would not have been possible without the guidance and tutelage of Dick Karp. It was he who taught us this field and gave us invaluable guidance at every stage of the book – from the initial planning to the feedback he gave us from using a preliminary version of the manuscript in a graduate course at Berkeley.

We thank the following colleagues, who carefully read portions of the manuscript and pointed out many errors in early versions: Pankaj Agarwal, Donald Aingworth, Susanne Albers, David Aldous, Noga Alon, Sanjeev Arora, Julien Basch, Allan Borodin, Joan Boyar, Andrei Broder, Bernard Chazelle, Ken Clarkson, Don Coppersmith, Cynthia Dwork, Michael Goldwasser, David Gries, Kazuyoshi Hayase, Mary Inaba, Sandy Irani, David Karger, Anna Karlin, Don Knuth, Tom Leighton, Mike Luby, Keju Ma, Karthik Mahadevan, Colin McDiarmid, Ketan Mulmuley, Seffi Naor, Daniel Panario, Bill Pulleyblank, Vijaya Ramachandran, Raimund Seidel, Tom Shiple, Alistair Sinclair, Joel Spencer, Madhu Sudan, Hisao Tamaki, Martin Tompa, Gert Vegter, Jeff Vitter, Peter Winkler, and David Zuckerman. We apologize in advance to any colleagues whose names we have inadvertently omitted.

Special thanks go to Allan Borodin and the students of his CSC 2421 class at the University of Toronto (Fall 1994), as well as to Gudmund Skovbjerg Frandsen, Prabhakar Ragde, and Eli Upfal for giving us detailed feedback from courses they taught using early versions of the manuscript. Their suggestions and advice have been invaluable in making this book more suitable for the classroom.

We thank Rao Kosaraju, Ron Rivest, Joel Spencer, Jeff Ullman, and Paul Vitanyi for providing us with much help and advice on the process of writing and improving the manuscript.

The first author is grateful to Stanford University for the environment and resources which made this effort possible. Several colleagues in the Computer Science Department provided invaluable advice and encouragement. Don Knuth played the role of mentor and his faith in this project was a tremendous source of encouragement. John Mitchell and Jeff Ullman were especially helpful with the mechanics of the publication process. This book owes a great deal to the students, teaching assistants, and other participants in the various offerings of the course CS 365 (Randomized Algorithms) at Stanford. The feedback from these people was invaluable in refining the lecture notes that formed a partial basis for this book. Steven Phillips made a significant contribution as a teaching assistant in CS 365 on two different occasions. Special thanks are due to Yossi Azar, Amotz Bar-Noy, Bob Floyd, Seffi Naor, and Boris Pittel for their guest lectures and help in preparing class notes. The following students transcribed some lecture notes, and their class participation was vital to the development of this material: Julien Basch, Trevor Bourget, Tom Chavez, Edith Cohen, Anil Gangolli, Michael Goldwasser, Bert Hackney, Alan Hu, Jim Hwang, Vasilis Kallistros, Anil Kamath, David Karger, Robert Kennedy, Sanjeev Khanna,

Daphne Koller, Andrew Kosoresow, Sherry Listgarten, Alan Morgan, Steve Newman, Jeffrey Oldham, Steven Phillips, Tomasz Radzik, Ram Ramkumar, Will Sawyer, Sunny Siu, Eric Torng, Theodora Varvarigou, Eric Veach, Alex Wang, and Paul Zhang.

The research and book-writing efforts of the first author have been supported by the following grants and awards: the Bergmann Award from the US-Israel Binational Science Foundation; an IBM Faculty Development Award; gifts from the Mitsubishi Corporation; NSF Grant CCR-9010517; the NSF Young Investigator Award CCR-9357849, with matching funds from IBM Corporation, Schlumberger Foundation, Shell Foundation, and Xerox Corporation; and various grants from the Office of Technology Licensing at Stanford University.

The second author is indebted to his colleagues at the Mathematical Sciences Department of the IBM Thomas J. Watson Research Center, and to the IBM Corporation for providing the facilities and environment that made it possible to write this book. He also thanks Sandeep Bhatt for his encouragement and support of a course on Randomized Algorithms taught by the author at Yale University; the class notes from that course formed a partial basis for this book.

We are indebted to Lauren Cowles of Cambridge University Press for her editorial help and advice in the preparation of the manuscript; this book has emerged much improved as a result of her untiring efforts.

Rajeev Motwani thanks his wife Asha for her love, encouragement, and cheerfulness; without her distractions this book would have been completed several months earlier. This task would not have been possible without the constant support and faith of his family over the years. Finally, the two mutts Tipu and Noori deserve special mention for giving company during the many late night editing sessions.

Prabhakar Raghavan thanks his wife Srilatha for her love and support, his parents for their inspiration, and his children Megha and Manish for ensuring that there was never a dull moment when writing this book.

**World-Wide Web**

Current information on this book may be found at the following address on the World-Wide Web:

http://www.cup.org/Reviews&blurbs/RanAlg/RanAlg.html

This address may be used for ordering information, reporting errors and viewing an edited list of errors found by other readers.

# Contents

PART ONE

# Tools and Techniques