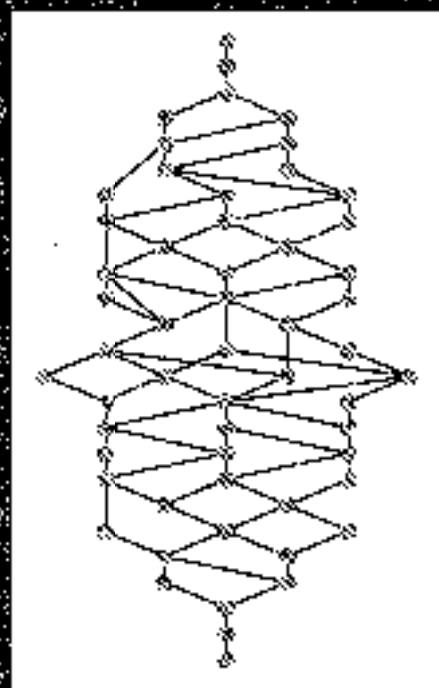


館際合作

# Computational Discrete Mathematics

Combinatorics and Graph Theory with *Mathematica*<sup>®</sup>

SRIRAM PEMMARAJU  
STEVEN SIKIENA



© 2011 WILEY



## Computational Discrete Mathematics

*Combinatorica*, an extension to the popular computer algebra system *Mathematica*<sup>®</sup>, is the most comprehensive software available for educational and research applications of discrete mathematics, particularly combinatorics and graph theory. This book is the definitive reference/user's guide to *Combinatorica*, with examples of all 450 *Combinatorica* functions in action, along with the associated mathematical and algorithmic theory. The authors cover classical and advanced topics on the most important combinatorial objects: permutations, subsets, partitions, and Young tableaux, as well as all important areas of graph theory: graph construction operations, invariants, embeddings, and algorithmic graph theory.

In addition to being a research tool, *Combinatorica* makes discrete mathematics accessible in new and exciting ways, by encouraging computational experimentation and visualization. The book is suitable for self-study and as a primary or supplementary textbook for discrete mathematics courses.

Saran Pemmaraju is Professor of Computer Science at The University of Iowa. His research interests are in discrete mathematics, graph algorithms, and distributed computing.

Steven Seiena is Professor of Computer Science at SUNY Stony Brook. He is also the author of *The Algorithm Design Manual*; *Calculus: Bits, Computers, Graphics, and Mathematical Modeling to Win*; and *Programming Challenge: The Programming Contest Training Manual*.



# Computational Discrete Mathematics

## Combinatorics and Graph Theory with *Mathematica*

SRIRAM PEMMARAJU

The University of Texas

STEVEN SKIENA

SUNY at Stony Brook

中國文化大學圖書館



B00573525



 CAMBRIDGE  
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE  
The Edinburgh Building, Shaftesbury Road, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS  
The Edinburgh Building, Cambridge CB2 2RU, UK  
40 West 20th Street, New York, NY 10011-4211, USA  
477 Williamstown Road, Port Melbourne, VIC 3207, Australia  
Ruiz de Alarcón 13, 28014 Madrid, Spain  
Dock House, The Waterfront, Cape Town 8001, South Africa

<http://www.cambridge.org>

© Cambridge University Press 2010

This book is in copyright. Subject to statutory exception  
and to the provisions of relevant collective licensing agreements,  
no reproduction of any part may take place without  
the prior written permission of Cambridge University Press.

First published 2010

Printed in the United States of America

Typeset Palatino System 10/12pt (10.5)

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

Library of Congress Cataloging in Publication Data

Fernández, Steven B., 1966-

Computational discrete combinatorics, combinatorics and graph theory with Mathematica / Steven Fernández, Steven B. Skiena.  
p. cm.

Includes bibliographical references and index.

ISBN 978-0-521-8666-1

1. Combinatorics (Computer science) – I. Fernández, Steven B. II. Skiena, Steven B. 2. Graph theory – Data processing.

I. Skiena, Steven B., 1966-

QA159.F45 2010

© 2010 by Cambridge University Press

ISBN 978-0-521-86666-1 (Hardback)

# Table of Contents

Preface.....	ix
• About Combinatorics • Where's the Book? • Why "Mathmark"? • Acknowledgments • Cover! • Dedication	

## Chapter 1. *Combinatorics: An Explorer's Guide*

1.1 Combinatorial Objects: Permutations, Subsets, Partitions.....	3
• Permutations and Subsets • Partitions, Compositions, and Young Tableaux	
1.2 Graph Theory and Algorithms.....	10
• Representing Graphs • Drawing Graphs • Generating Graphs • Properties of Graphs • Algorithmic Graph Theory	
1.3 Combinatorics Conversion Guide.....	12
• The Main Differences • Questions You may Wonder How to Change	
1.4 An Overview of <i>Mathematica</i> .....	41
• The Structure of <i>Mathematica</i> • Mathematical Operations • List Manipulation • Iteration • Ten Little <i>n</i> -Sums • Combinatorics • Compiling <i>Mathematica</i> Code	

## Chapter 2. *Permutations and Combinations*

2.1 Generating Permutations.....	55
• Lexicographically Ordered Permutation • Ranking and Unranking Permutations • Random Permutations • Minimum Change Permutations	
2.2 Inversions and Inversion Vectors.....	69
• Inversion Vectors • Counting Inversions • The Index of a Permutation • Runs and Eulerian Numbers	
2.3 Combinations.....	76
• Subsets via Binary Representation • Gray Codes • Lexicographically Ordered Subsets • Generating <i>k</i> -Subsets • Singing	
2.4 Exercises.....	89
• Thought Exercises • Programming Exercises • Experimental Exercises	

## Chapter 3. *Algebraic Combinatorics*

3.1 The Cycle Structure of Permutations.....	93
• Odd and Even Permutations • Types of Permutations • Finding Cycles • Counting Cycles	
3.2 Special Classes of Permutations.....	104
• Involutions • Derangements	
3.3 Poly Theory.....	109
• Permutation Groups • Group Action • Equivalence Classes and Orbits • Cycle Index of Permutation Groups • Applying Pólya's Theorem	
3.4 Exercises.....	131
• Thought Exercises • Programming Exercises • Experimental Exercises	

## Chapter 4. Partitions, Compositions, and Young Tableaux

4.1	Integer Partitions.....	135
	▪ Generating Functions ▪ Generating Functions and Partitions ▪ Ferrers Diagrams ▪ Random Partitions	
4.2	Compositions.....	146
	▪ Random Compositions ▪ Generating Compositions	
4.3	Set Partitions.....	149
	▪ Generating Set Partitions ▪ Stirling and Bell Numbers ▪ Accessing, Unranking, and Random Set Partitions ▪ Set Partitions and Restricted Growth Functions	
4.4	Young Tableaux.....	162
	▪ Insertion and Deletion ▪ Insertions and Pairs of Tableaux ▪ Generating Young Tableaux ▪ Counting Tableaux by Shape ▪ Random Tableaux ▪ Longest Increasing Subsequences	
4.5	Exercises.....	173
	▪ Thought Exercises ▪ Programming Exercises ▪ Technical Exercises	

## Chapter 5. Graph Representation

5.1	Data Structures for Graphs.....	179
	▪ The Internal Representation ▪ Edge Lists ▪ Adjacency Lists ▪ Adjacency Matrix ▪ Incidence Matrix	
5.2	Modifying Graphs.....	192
	▪ Additions, Deletions, and Changes ▪ Setting Graph Options	
5.3	Classifying Graphs.....	198
5.4	Displaying Graphs.....	200
	▪ The Vertex and Edge Options ▪ Inherited Options ▪ A Hierarchy of Options ▪ Highlighting and Animation	
5.5	Basic Graph Embeddings.....	213
	▪ Circular Embeddings ▪ Planar Embeddings ▪ Spherical Embeddings ▪ Routed Embeddings	
5.6	Improving Embeddings.....	219
	▪ Translating, Dilating, and Rotating Graphs ▪ Staking Graphs ▪ Spring Embeddings	
5.7	Storing and Editing Graphs.....	224
5.8	Exercises.....	226
	▪ Thought Exercises ▪ Programming Exercises ▪ Experimental Exercises	

## Chapter 6. Generating Graphs

6.1	Building Graphs from Other Graphs.....	231
	▪ Connecting Vertices ▪ Eulerian and Hamiltonian Graphs ▪ Unions and Intersections ▪ Stars and Disjointes ▪ Joins of Graphs ▪ Products of Graphs ▪ Line Graphs	
6.2	Regular Structures.....	244
	▪ Complete Graphs ▪ Circulant Graphs ▪ Complete Bipartite Graphs ▪ Cycles, Stars, and Wheels ▪ Grid Graphs ▪ Random Regular Graphs	



6.3	Trees.....	254
	• Labeled Trees • Complete Trees	
6.4	Random Graphs.....	263
	• Constructing Random Graphs • Realizing Degree Sequences	
6.5	Relations and Functional Graphs.....	269
	• Graphs from Relations • Functional Graphs	
6.6	Exercises.....	273
	• Thought Exercises • Programming Exercises • Experimental Exercises	
<b>Chapter 7. Properties of Graphs</b>		
7.1	Graph Traversals.....	277
	• Breadth-First Search • Depth-First Search	
7.2	Connectivity.....	283
	• Connected Components • Strong and Weak Connectivity • Cutting Edges • Biconnected Components • $k$ -Connectivity • Harary Graphs	
7.3	Cycles in Graphs.....	294
	• Acyclic Graphs • Cycles • Hamilton Cycles • Hamiltonian Cycles and Paths • Traveling Salesman Tour	
7.4	Graph Coloring.....	306
	• Bipartite Graphs • Chromatic Polynomials • Finding a Vertex Coloring • Edge Colorings	
7.5	Cliques, Vertex Covers, and Independent Sets.....	316
	• Maximum Clique • Minimum Vertex Cover • Maximum Independent Set	
7.6	Exercises.....	319
	• Thought Exercises • Programming Exercises • Experimental Exercises	
<b>Chapter 8. Algorithmic Graph Theory</b>		
8.1	Shortest Paths.....	323
	• Single-Source Shortest Paths • All-Pairs Shortest Paths • Applications of All-Pairs Shortest Paths • Number of Paths	
8.2	Minimum Spanning Trees.....	335
	• Union-Find • Kruskal's Algorithm • Growing Spanning Trees	
8.3	Network Flow.....	340
8.4	Matching.....	343
	• Maximal Matching • Bipartite Matching • Weighted Bipartite Matching and Vertex Cover • Stable Matchings	
8.5	Partial Orders.....	352
	• Topological Sorting • Transitive Closure and Reduction • Hasse Diagrams • Dilworth's Theorem	
8.6	Graph Isomorphism.....	357
	• Finding Isomorphisms • Tree Isomorphism • Self-Complementary Graphs	

<b>8.7 Planar Graphs</b> .....	<b>370</b>
■ Testing Planarity.....	
<b>8.8 Exercises</b> .....	<b>372</b>
■ Thought Exercises ■ Programming Exercises ■ Experimental Exercises.....	
<b>Appendix</b> .....	<b>375</b>
<b>Reference Guide</b> .....	<b>376</b>
<b>Bibliography</b> .....	<b>447</b>
<b>Index</b> .....	<b>459</b>

## Preface

The excitement of discrete mathematics comes from discovery, the act of uncovering beautiful and important properties of graphs, partitions, permutations, and other combinatorial objects. The appeal of discrete mathematics is its concreteness, that you can draw these objects on a blackboard and get a feel for them in a way quite different from more abstract areas of mathematics.

Unfortunately, only very small structures can be built on a blackboard; computers are needed to experiment with larger ones. The goal of *Combinatorica* is to advance the study of combinatorics and graph theory by creating a wide variety of functions available for active experimentation. Together, this book and *Combinatorica* provide a unique resource for discovering discrete mathematics.

### ■ About *Combinatorica*

*Combinatorica* has been perhaps the most widely used software for teaching and research in discrete mathematics since its initial release in 1990. *Combinatorica* is an extension to *Mathematica*, which has been used by researchers in mathematics, physics, computer science, economics, and anthropology. *Combinatorica* received a 1991 EDUCCOM Higher Education Software Award for Distinguished Mathematics Software and has been employed in teaching from grade school to graduate school.

But times change, in this case for the good. Desktop computers (and notebooks) are now more than 100 times faster than when *Combinatorica* was originally developed. Computational problems unimaginable on research machines then can now be done at home by high school students. *Mathematica* itself has gone through several versions, resulting in a significantly improved user interface, more functionality, better performance, and improved typesetting facilities.

This book presents the second-generation version of *Combinatorica*, which is a dramatic improvement over the original. Enhancements since the previous version include:

- *Improved Performance* – We have made incredible strides in efficiency since the original *Combinatorica*. All examples in the original edition of *Implementing Discrete Mathematics* involved graphs of fewer than 100 vertices, because larger examples were either impossible or hopelessly slow. Now we can work on interesting graphs with tens of thousands of vertices and hundreds of thousands of edges. Indeed, all the examples in this book run comfortably on a two-year-old laptop computer that has seen better days!
- *Improved Graph Representation and Visualization* – The *Combinatorica* graph structure has been completely revamped, enabling us to efficiently represent very large graphs, with edge/vertex weights, labels, and graphics. These improvements make *Combinatorica* a viable platform for developing significant graph applications in *Mathematica*. Color graphics displays and printers are now ubiquitous, although they were prohibitively expensive back in 1990. In response, we have brought a new degree of color and style to *Combinatorica* graphics. Graphs can now be drawn with a variety of edge and vertex presentations and give

complete freedom to highlight and position graph elements. We provide the power to animate interesting structures and algorithms on graphs, and to transform these views to animated GIF files for Web pages.

- *New Functionality* – *Combinatorica* now adds more than 430 functions for combinatorics and graph theory to *Mathematica* – twice as many as the previous version. Existing functions have been made more sophisticated, with new options and better algorithms. The new *Combinatorica* provides extensive support for Pólya's theory of counting and algebraic combinatorics, as well as new combinatorial objects such as set partitions and restricted growth functions. These changes reflect demands from our users, recent developments in combinatorics and graph theory, and the additional functionality of new releases of *Mathematica*.

*Combinatorica* is included with every copy of *Mathematica* as *DiscreteMath`Combinatorica`*. The *Combinatorica* described in this book appears in *Mathematica* Version 3.2 and beyond, but our package is backwards compatible with earlier versions of *Mathematica*. The latest release of *Combinatorica* is available from our Web site <http://www.combinatorica.com>, where we maintain software, data, and other materials of interest to the *Combinatorica* community. Register at [www.combinatorica.com](http://www.combinatorica.com) to hear about updates to the package.

### What's Between the Covers

This book is a successor to *Implementing Discrete Mathematics* [Sk99], the original description of *Combinatorica*. This book, like the program itself, is a complete rewrite. Here, we present a general computational approach to discrete mathematics that will serve as the definitive user's guide/manual to *Combinatorica*. What's new?

- *Introduction User's Guide* – *Combinatorica* has been widely used by groups of people who know relatively little graph theory, in interesting ways that we had never envisioned. We now begin with a substantial user's guide containing numerous examples of how to use the package, and surveying the range of what *Combinatorica* can do.
- *Conversion Guide* – We have tried our best to provide backwards compatibility, but in a small number of cases providing backwards compatibility conflicted with other goals such as increasing efficiency or providing better functionality. Thus a small number of *Combinatorica* functions will not work they way they used to in the old package. We include a section that enumerates functions that have changed and give new *Combinatorica* code that replicates the old behavior.
- *Selected Function Implementations* – The previous version of this book contained literally every line of *Mathematica* code that made up *Combinatorica*. With the new package three times larger than the old one, maintaining this policy is neither feasible nor desirable. Instead, we present the implementations of only the most enlightening and important algorithms. These have been kept away from the preliminary chapter to aid and comfort new *Combinatorica* users.
- *Expanded Tutorial Component* – We have significantly improved and expanded the tutorial component of the book. Now, in addition to being a reference, supplement, or guide to self-study,

it can serve as a textbook for *Mathematica*-based undergraduate discrete mathematics courses. In particular, the first author has taught such courses at both MIT Bombay and the University of Iowa. This is the only combinatorics/graph theory textbook we are aware of that is built around significant software.

There is more than enough material here to teach a full semester, experimentally enhanced course in combinatorics and graph theory. *Mathematica* is now quite well established at many colleges and universities around the world. It is highly likely that your school already has a site license for *Mathematica*. Contact your campus computing center if you need more information on this. Students in such schools/departments will particularly benefit from our approach.

- *Near Exercise* – Finally, we have included three interesting classes of exercises at the end of each chapter. Some are thought problems, typically requesting proofs of well-known or interesting theorems in combinatorics or graph theory that we illustrate as examples of *Combinatorica* in action. Some are programming exercises, where the reader is encouraged to extend or improve existing *Combinatorica* functions. The remainder suggest interesting discrete mathematics experiments to conduct using *Combinatorica*.

This book concentrates on two distinct areas in discrete mathematics. The first section deals with combinatorics, loosely defined as the study of counting. We provide functions for generating combinatorial objects such as permutations, partitions, and Young tableaux, as well as for studying various aspects of these structures.

The second section considers graph theory, which can be defined equally loosely as the study of binary relations. We consider a wide variety of graphs and provide functions to generate them. Although graphs are combinatorial structures, understanding them requires pictures or embeddings. Thus we provide functions to create a variety of graph embeddings, enabling a given structure to be viewed in several different ways. Algorithmic graph theory is an important interface between mathematics and computer science, and in this text we present a variety of polynomial and exponential time algorithms to solve computational problems in graph theory.

These two sections are relatively independent of each other, so feel free to jump in the middle if you are primarily interested in graph theory.

This book is designed as a guide to manipulating discrete structures. You will find no formal proofs in this book, but enough discussion to understand and appreciate the literally hundreds of algorithms and theorems contained within. Further, we provide extensive references as pointers to the appropriate results. Since the body of the text contains the most interesting of more than 450 *Combinatorica* functions, it is also an excellent guide for writing your own *Mathematica* programs. We include a brief guide to *Mathematica* for the uninitiated to help in this regard.

This book is also a complete reference manual for using *Combinatorica* to explore discrete mathematics. As you read the book we urge you to play with the package. Documentation for all *Combinatorica* functions appears in the *Glossary of Functions* at the end of the book, and cross-references to examples using a particular function in an interesting way appear in the index.

## ■ Why Mathematica?

At its initial release in 1990, *Combinatorica* was the largest package ever written for the then-recently released *Mathematica*. Today *Mathematica* has established itself as the mathematical tool of choice, with more than one million users.

Building a discrete mathematics package in *Mathematica* has several advantages. Arbitrary precision arithmetic means we are free of the burdens of computer word length. Where appropriate, we have access to portable PostScript graphics, bringing life to graphs and their embeddings. Working with symbolic formulas makes convenient such techniques as generating functions and chromatic polynomials. The freedom of a high-level language with so much mathematics already under the hood liberates us to explore a much larger fraction of what is known about discrete mathematics.

The chief drawback to using such a high-level language as *Mathematica* is that we lose sight (not to mention the time complexity) of our algorithms. The model of computation that it presents (the Wolf-RAM?) is dramatically and mysteriously different from the traditional random access machine. In this new version, we have tamed the Wolf-RAM – achieving efficiencies that allow interaction with large and interesting structures in real time.

## ■ Acknowledgments

We would like to thank all the people who helped make this book and the new *Combinatorica* a reality.

First, we thank the people at Wolfram Research Inc. (WRI). John Nozick provided an amazing amount of support throughout the development of the package, guiding us through some of the more obscure aspects of *Mathematica* and providing the interface between us and various *Mathematica* gurus at WRI. Eric Weisstein played with several versions of *Combinatorica*, reported lots of bugs, and contributed code for some of the graph constructors. Arnoud Buzing, Damien Gloagmeyer, Shraw Devma, Anna Pukin, and Andy Shoekh of the “bug testing department” bugged us regularly with bug reports. Danie Elendlauf and Robby Villegas made suggestions that improved the package. Bill White patiently answered questions about WRI’s version of LaTeX. Stephen Wolfram’s interest in *Combinatorica* since its beginning has had much to do with its becoming a reality.

The insights of four Trias into graph drawings and representation significantly shaped the graph data structure we employ in the new *Combinatorica*. Lenne Heath commented extensively on initial drafts of the first two chapters. Kaushal Kurapati helped with perl scripts for integrating *Combinatorica* code and *Combinatorica* examples into the text. Eugene Curtin’s package on Pólya theory helped crystallize some of our initial ideas on functionality needed for Pólya-theoretic computations. Even Hing-kuai built a nice Java-based editor for *Combinatorica* graphs and King Mak developed interesting algorithm animations – check these out. We especially thank Marty Golumbic for his efforts to bring us together to finish this book and Alberto Apostolico for providing a home where part of it was written.

Laura Cawley, Aziz Harvey, and Elise Oranges of Cambridge University Press helped us throughout the publication process.

Two batches of students at IIT Bombay and one at Iowa survived the first author's teaching experiments with *Combinatorica*. We thank all the people who have downloaded preliminary versions of the new *Combinatorica* and have sent us encouragement and bug reports.

We cannot neglect those who helped with the previous version of *Combinatorica* or the book. Anil Dhansali made a significant contribution to the original *Combinatorica* by writing many of the functions, managing the testing of the code, and performing sanity tests. The original work on *Combinatorica* was done at Apple Computer in the summer of 1988. Alan Wykle, then of Addison-Wesley, provided encouragement and a means to communicate this to the world. Fred Buckley, Nora Hartsfield, Matthew Markert, Marko Petkovšek, Ilan Vardi, Jürgen Koslowski, and Rick Wilson all provided helpful pre-publication feedback on *Combinatorica* and/or implementing *Discrete Mathematics*. A variety of other Stony Brook (Philip Hsu, Paul Lewis, Yaw-Ling Lin, Gene Sterk, Brian Lin, Alan Tucker, Shipei Zhang), Addison-Wesley (Jan Bebes, Laura Likely, Karl Masumoto), and WRI (Dave Ballman, John Benedies, Martin Buchholz, Joe Groenens, Igor Rivin, Monte Seyer, Lisa Shipley, Catherine Smith) people all helped out in one way or another.

On a personal level, we thank Anna, Nanna, Sachit, Geeta, and Rama Rao, who provided encouragement and support for the first author. Mom, Dad, Len, and Rob provided moral and emotional support for the second author.

## ■ Caveat

It is traditional for the author to magnanimously accept the blame for whatever deficiencies remain. We don't. Any errors, deficiencies, or problems in this book are somebody else's fault, but report them to [bugs@combinatorica.com](mailto:bugs@combinatorica.com) so we can determine who to blame.

Srinan Pemmaraju  
Dept. of Computer Science  
University of Iowa  
Iowa City, IA 52240-1419  
[srinan@combinatorica.com](mailto:srinan@combinatorica.com)

Steven S. Skiena  
Dept. of Computer Science  
State University of New York  
Stony Brook, NY 11794-4400  
[sskiena@combinatorica.com](mailto:sskiena@combinatorica.com)

## ■ Dedication

Combinatorialists know *age* to count their blessings. As a proof of correctness, both of us got the same answer of  $2 + e$  when we did the counting.

We dedicate this book to our respective wives, Santhi and Renee. The writing of this book was a long and complicated process involving cross-country travel, hours at the keyboard, long by phone calls, and even fears of terrorist actions and bioterrorist nuclear war! We thank them for their love and patience during this period, before, and beyond.

We also dedicate this book to our daughters, Pia and Bernie. We look forward to the day, not so long from now, when the pretty pictures in this book will help us explain to you the joy of doing and understanding discrete mathematics.

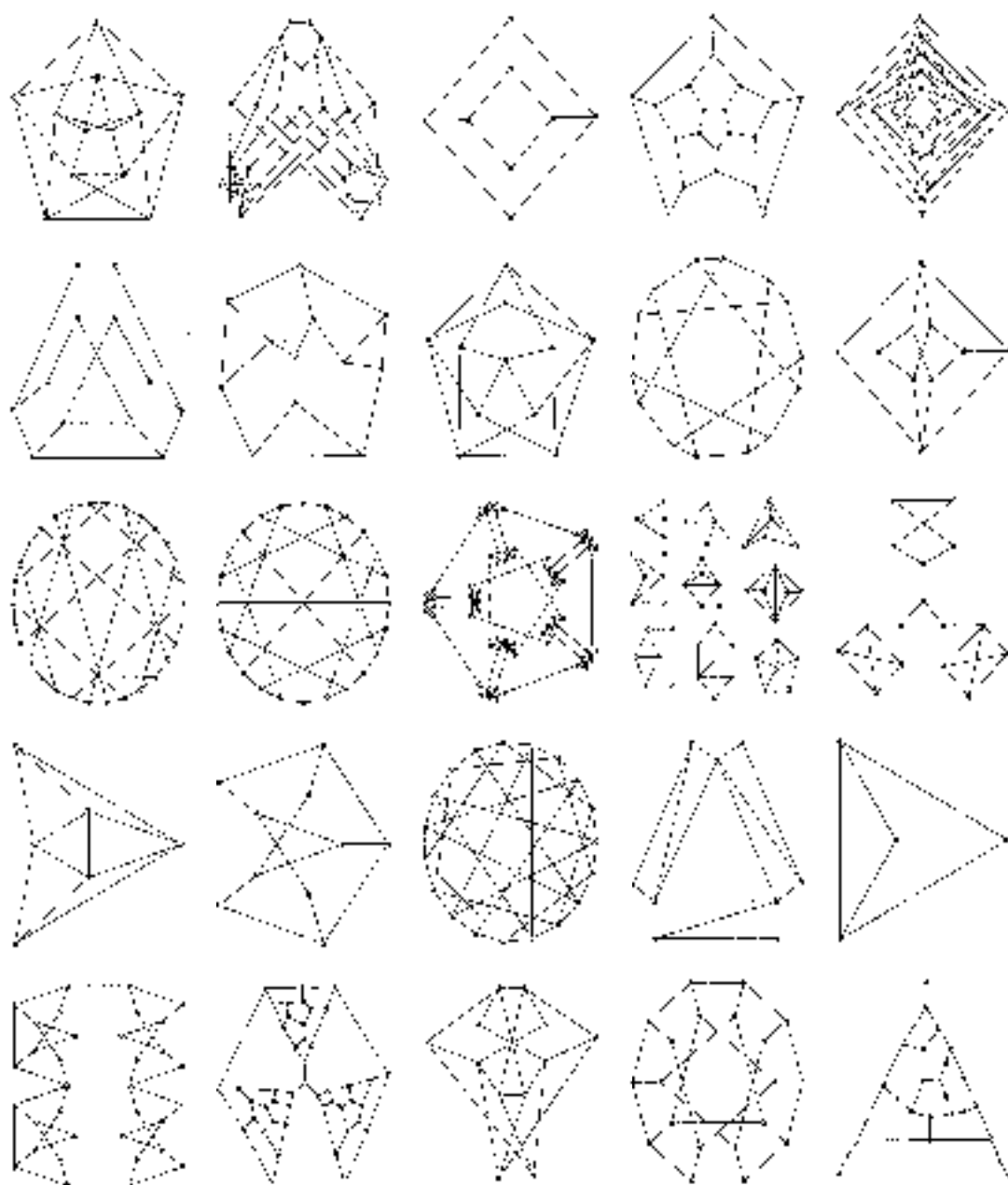
Finally, we dedicate this book to our respective future children, both well on the way to joining us. We will be meeting you both about the time this book is published. We don't know you yet, but certainly know that we love you.



---

# 1. *Combinatorica*: An Explorer's Guide

---



Discrete mathematics is a subject for explorers. *Combinatorica* has been designed to make it easy to generate and visualize graphs and experiment with other combinatorial structures such as permutations, partitions, and Young tableaux. Manipulating these objects with *Combinatorica* helps the curious mind to recognize patterns, make conjectures, and then conduct systematic searches to confirm or refute them. The sole prerequisite for this book is a curiosity about discrete structures.

This brief introductory chapter demonstrates the power of *Combinatorica* for exploring discrete mathematics. With over 450 functions, *Combinatorica* can be somewhat overwhelming for beginners. Here we give brief examples of some of its most popular functions in action. This overview serves as a user's guide to *Combinatorica*, showing the proper mindset to get started with the package.

This chapter contains tables with the names of all *Combinatorica* functions broken down by category. Skimming through these tables now will give you an idea of what the package can do. Referring to these tables later can help you to recall or identify the name of the function you need. Don't forget the glossary and index, as well as the online help available in *Mathematica*. These tools will help you to navigate the *Combinatorica* jungle and get on with your explorations.

The rest of this book provides a deeper look at how *Combinatorica* works, illustrating the theory and algorithms behind each function, as well as its implementation in the *Mathematica* programming language. We hope the sampling of *Combinatorica* examples in this chapter inspires you to dive into the rest of the book.

Despite our attempts to maintain backwards compatibility, a small number of *Combinatorica* functions no longer behave as they used to in the old package. In Section 1.3 we provide a guide for porting code from old *Combinatorica* into the new package. The examples in this chapter require only minimal knowledge of *Mathematica* to understand. We provide a brief introduction to *Mathematica* programming in Section 1.4 to help the uninitiated.

#### About the illustration overview:

*Combinatorica* provides many functions for constructing graphs. A few of the built-in graph construction functions do not have parameters and construct a single interesting graph. The function `FiniteGraphs` collects them together in one list for convenient reference. `ShowGraphArray` permits the display of multiple graphs in one window.

---

`ShowGraphArray[FiniteGraphs]` and `FiniteGraphs`, 51

---

# 1.1 Combinatorial Objects: Permutations, Subsets, Partitions

Basic combinatorial objects such as permutations, subsets, and partitions are the raw materials from which discrete mathematics is built. The ability to construct, count, and manipulate them has a surprising variety of applications, as well as its own inherent beauty.

This needs the `Combinatorica` package included with Mathematica, a necessary step before using any `Combinatorica` functions.

## 1.1.1 Permutations and Subsets

Permutations and subsets are the most basic combinatorial objects, dealing with the arrangement and selection of elements, respectively. `Combinatorica` provides functions for constructing these objects both randomly and deterministically, to rank and unrank them, and to compute invariants on them. Here we provide examples of some of these functions in action.

Permutations are arrangements or orderings of the numbers 1 to  $n$ .

```
in[2]:= Permutations[3]
out[2]= {{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1},
         {3, 1, 2}, {3, 2, 1}}
```

The `Permute` operator enables us to apply a permutation to rearrange objects or items.

```
in[3]:= Permute[{A,B,C,D}, Permutations[3]]
out[3]= {{A, B, C}, {A, C, B}, {B, A, C}, {B, C, A},
         {C, A, B}, {C, B, A}}
```

Each permutation has an *inverse*, a distinct permutation that rearranges it to the *identity* permutation.

```
in[4]:= Permute[{5,2,4,3,1}, InversePermutation[{5,2,4,3,1}]]
out[4]= {1, 2, 3, 4, 5}
```

These permutations are generated in increasing order, where successive permutations differ by exactly one transposition. The function `Permutations` constructs permutations in lexicographic order.

```
in[5]:= MinimalChangePermutations[{a,b,c}]
out[5]= {{a, b, c}, {b, a, c}, {c, a, b}, {a, c, b},
         {c, b, a}, {b, c, a}}
```

The ranking function takes a permutation of 1 through  $n$  and assigns it a unique integer in the range 0 through  $n! - 1$  to it.

```
in[6]:= RankPermutation[{6, 9, 7, 1, 5, 4, 3, 2}]
out[6]= 37 955
```

The unranking function is the inverse of the ranking function: here we take a rank and get the original permutation back.

With 21 = 6 distinct permutations of three objects, we are likely to see all of them in a sample of 20 random permutations. But it is not likely that the first 6 permutations in the sample will all be distinct.

Any permutation can be decomposed into disjoint cycles, providing an alternate and useful way of viewing permutations. A cycle of length  $k > 1$  is a *fixed point*, and it corresponds to an element being mapped onto itself by the permutation. Cycles of length 2 correspond to *swaps*.

A classic combinatorial problem is counting how many different ways necklaces can be made out of  $k$  beads using a given set of colors. The beauty of Pólya's approach to this problem is that it yields a polynomial, the coefficients of whose terms provide answers to a more general problem. From the polynomial computed in this example, we see that there are 10 6-bead necklaces with 3 beads of each color, 2 beads colored  $b$ , and 1 bead colored  $c$ . [Combinatorics provides extensive support for Pólya theory.](#)

Substituting 1 for each of the variables  $a$ ,  $b$ , and  $c$  in the above polynomial, gives us the total number of 6-bead necklaces that can be constructed using an infinite supply of beads of each of 3 colors.

The number of reversals (pairs of adjacent elements) in a permutation is equal to that of its inverse.

```
In[7]:= InversePermutation[2, 9]
Out[7]= {9, 6, 7, 1, 8, 4, 5, 3, 2}
```

```
In[8]:= TableOfRandomPermutation[3, {20}]
Out[8]= {{2, 1, 3}, {1, 3, 2}, {1, 2, 3}, {3, 1, 2},
  {1, 2, 3}, {3, 1, 2}, {1, 3, 2}, {2, 1, 3}, {1, 2, 3},
  {3, 1, 2}, {2, 3, 1}, {2, 1, 3}, {3, 1, 2}, {1, 3, 2},
  {3, 1, 2}}
```

```
In[9]:= p = RandomPermutation[10]; {p, ToCycles[p]}
Out[9]= {{1, 6, 2, 10, 9, 8, 5, 7, 3},
  {{1}, {6, 4, 10, 3, 2}, {9, 8, 7, 3}}}
```

```
In[10]:= CycleIndexPolynomial[6, {a, b, c}, Cycles]
Out[10]= a6 - a5 b + 3 a4 b2 - 4 a3 b3 + 3 a2 b4 + a b5 +
  b6 + a4 c + 5 a3 b c + 10 a2 b2 c + 10 a2 b3 c +
  5 a b4 c + b5 c + 7 a3 c2 - 10 a2 b c2 + 16 a2 b2 c2 +
  10 a b3 c2 + 3 a2 b2 c3 + 3 a2 b c3 + 10 a2 b c4 +
  10 a b2 c3 + 5 a2 c4 - 5 a2 b c4 + 5 a b2 c4 +
  a5 c + b5 c - c6
```

```
In[11]:= % /. {a -> 1, b -> 1, c -> 1}
```

```
Out[11]= 36
```

```
In[12]:= {PermutationInversion[60], {InversionCount[p],
  InversionCount[InversePermutation[p]]}}
Out[12]= {60, {60, 60}}
```

This consists of all 4-element subsets. They are listed in Gray code or minimum change order, where each subset differs in exactly one element from its neighbors.

```
rank31 := subsets(1,2,3,4)
rank15 := 4! - {4}, {3}, {2}, {3}, {2, 3}, {1}, {3, 4}, {2, 4},
           {3}, {1, 2}, {1}, {3, 4}, {1, 2, 3, 4}, {1, 2, 3},
           {1, 3}, {1, 3, 4}, {1, 4}, {1}
```

A  $k$ -subset is a subset with exactly  $k$  elements in  $\{1, \dots, n\}$ . Since the lead element is placed in first, the  $k$  subsets of a given set are generated in lexicographic order.

```
rank41 := subsets(1,2,3,4,6),3
rank41 := {1, 2, 3}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4},
           {1, 3, 5}, {1, 3, 6}, {2, 3, 4}, {2, 3, 5}, {2, 4, 5},
           {3, 4, 5}
```

BinarySearch	LexicographicPermutations
BinarySearch2	LongestIncreasingSubsequence
Combinations	MinimumChangePermutations
DistinctPermutations	NextPermutation
InterlockingListGen	PermutationG
FromTypes	PermutationType
FromInversionVector	PermutationWithCycle
HeapSort	Permute
Heapify	RandomHeap
HasCycles	RandomPermutation
IdentityPermutation	RankPermutation
Index	RevealCycles
InversePermutation	Runs
InversionPoset	SelectionSort
Inversions	SignalizePermutation
Involutions	Cycles
Involutions	ToInversionVector
Josephus	UnrankPermutation

Combinatorial functions for permutations.

BinarySubsets	RandomKSubset
DescruijnSequence	RandomSubset
GrayCodeKSubsets	RankBinarySubset
GrayCodeSubsets	RankGrayCodeSubset
KSubsets	RankSubset
LexicographicSubsets	RankSubset
NextBinarySubset	RankSubset
NextGrayCodeSubset	Strings
NextKSubset	Subsets
NextLexicographicSubset	UnrankBinarySubset
NextSubset	UnrankGrayCodeSubset
NthSubset	UnrankSubset
	UnrankSubset

Combinatorics functions for arrays.

AlternatingGroup	ListGraphs
AlternatingGroupIndex	ListYachlatac
CycleIndex	MultiplicationTable
CycleStructure	NSCyclicPolynomial
Cycles	OrbitInventory
Cyclic	OrbitRepresentatives
CyclicGroup	Orbits
CyclicGroupIndex	Ordered
Dihedral	PairGroup
DihedralGroup	PairGroupIndex
DihedralGroupIndex	PermutationGroupQ
EquivalenceClasses	SelfRelation
KSubsetGroup	SymmetricGroup
KSubsetGroupIndex	SymmetricGroupIndex

Combinatorics functions for Polys through connections.

## 1.1.2 Partitions, Compositions, and Young Tableaux

A partition of a positive integer  $n$  is a set of  $k$  strictly positive integers whose sum is  $n$ . A composition of a positive integer  $n$  is a particular arrangement of nonnegative integers that adds up to  $n$ . A composition differs from a partition in that 0's can be part of a composition and different arrangements of the same set of numbers correspond to different compositions. A set partition of  $n$  elements is



This list of tableaux of shape  $(2, 2, 1)$  illustrates the amount of freedom available for arranging numbers in tableau structures. The smallest element always sits in the upper left-hand corner, but the largest element has considerable freedom. For the shape  $(2, 2, 1)$ , the largest element can be the last element in the second row or in the first row.

By iterating through the different integer partitions of shapes, all tableaux of a particular size can be constructed.

The built-in formula can be used to count the number of tableaux for any shape. Using the built-in formula over all partitions of  $n$  computes the number of tableaux on  $n$  elements.

The function `RandomTableau` selects each one of the  $7,123,456,789$  tableaux of this shape with equal likelihood.

Any sequence of  $n \geq 1$  distinct integers must contain either an increasing or a decreasing subsequence of length  $\lceil n/2 \rceil$ . This result has an elegant proof using tableaux. For  $n=6$ 's example, at least one of the two numbers is at least 3.

```
In[20]:= Tableaux[{{2,2,1}}
Out[20]= {{1, 2, 3}, {2, 3}, {3}}, {{1, 3}, {2, 4}, {4}},
         {{1, 2}, {3, 4}, {4}}, {{1, 3}, {2, 4}, {5}},
         {{1, 2}, {3, 4}, {5}};
```

```
In[21]:= Tableaux[3]
Out[21]= {{1, 2, 3}, {1, 3}, {2}, {1, 2}, {3}},
         {{1}, {2}, {3}};
```

```
In[22]:= NumberOfTableaux[10]
Out[22]= 3439
```

```
In[23]:= TableForm[ RandomTableau[{{6,5,4,3,2}} ]
Out[23]= //TableForm =
      2   4   10  11  21
      3   5   12  15  19
      6   7   13  16  25
      8   14  20  22
      9   18  24
      17  23
```

```
In[24]:= p = RandomPermutation[7^2 + 1];
          Map[Length, LongestIncreasingSubsequences[*]] &,
          {p, Reverse[p]}
Out[25]= {13, 13}
```

Compositions	NextPartition
CommutingIntegerPartitions	PartitionQ
Combinatorica	Partitions
DurfeeSquare	RandomComposition
FerrersDiagram	RandomPartition
YoungComposition	TransposePartition

Combinatorica is an aid for integer partitions and compositions.



- [read online Space-Time Reference Systems](#)
- [read online The Berenstain Bears and Too Much Teasing](#)
- [click The Crediton Killings \(Knights Templar Mystery, Book 4\)](#)
- [click Leonard: My Fifty-Year Friendship with a Remarkable Man](#)
- [read Landscapes and Landforms of Ethiopia \(World Geomorphological Landscapes\)](#)
  
- <http://flog.co.id/library/The-Generosity-of-Women.pdf>
- <http://berttrotman.com/library/The-Berenstain-Bears-and-Too-Much-Teasing.pdf>
- <http://anvilpr.com/library/Being-Reduced--New-Essays-on-Reduction--Explanation--and-Causation.pdf>
- <http://nexson.arzamaszev.com/library/The-Ecstatic.pdf>
- <http://econtact.webschaefer.com/?books/Sex-Marks-the-Spot--Your-Sexual-Organs.pdf>