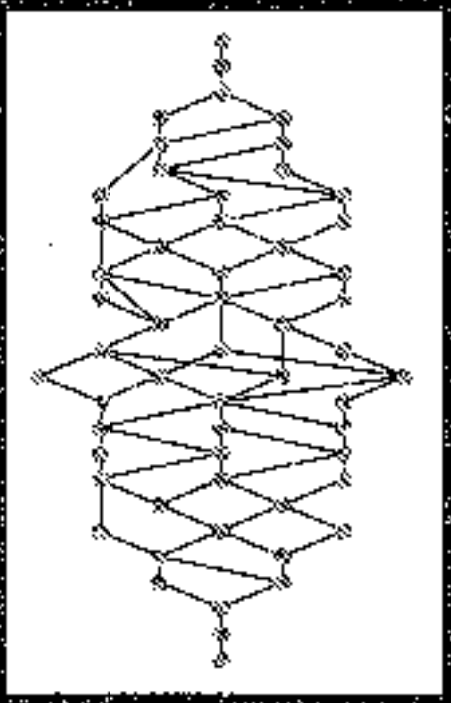
館際合作

Computational Discrete Mathematics

Combinatorics and Graph Theory with Mathematica®

SRIRAM PEMMARAJU
STEVEN SKIENA



CAMBRIDGE

Computational Discrete Mathematics

Combinatorica, an extension to the popular computer algebra system *Mathematica*®, 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.

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

Steven Skiena is Professor of Computer Science at SUNY Stony Brook. He is also the author of *The Algorithm Design Manual*; *Calculated Bets: Computers, Gambling, and Mathematical Modeling to Win*; and *Programming Challenges: The Programming Contest Training Manual*.

Computational Discrete Mathematics

Combinatorics and Graph Theory with *Mathematica*

SRIRAM PEMMARAJU
The University of Iowa

STEVEN SKIENA
SUNY at Stony Brook

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, 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

First published 2010

Printed in the United States of America

Typeface Palatino System LaTeX 2ε [AU]

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

Library of Congress Cataloging in Publication Data

Pemmaraju, Sriram V., 1969-
Computational discrete mathematics : combinatorics and graph theory with Mathematica / Sriram Pemmaraju, Steven Skiena.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-80686-1
1. Combinatorics–Computer-assisted instruction. 2. Combinatorial analysis–Data processing. 3. Graph theory–Data processing.
I. Skiena, Steven S. II. Title.
QA164.P75 2003
511'.6'0285 2002041585

ISBN 0 521 80686 1 hardback

Table of Contents

Chapter 4. Partitions, Compositions, and Young Tableaux

Chapter 5. Graph Representation

Chapter 6. Generating Graphs

Chapter 7. Properties of Graphs

Chapter 8. Algorithmic Graph Theory

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 making 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 EDUCOM 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 workstations) 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 visions to animated GIF files for Web pages.

- *New Functionality* – *Combinatorica* now adds more than 450 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` to hear about updates to the package.

3 What's Between the Covers

This book is a successor to *Implementing Discrete Mathematics* [Ski90], 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?

- *Interactive 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 twice 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 IIT 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 courses 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 computer computing center if you need more information on this. Students in such schools/departments will particularly benefit from our approach.

- *New Exercises* – 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 1991, *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 tight control over 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 Novak 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, Darren Glosemeyer, Sheral Devone, Anna Pikin, and Andy Shiekh of the "bug testing department" bugged us regularly with bug reports. Danie Lichtblau 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 Ivan Trias into graph drawings and representation significantly shaped the graph data structure we employ in the new *Combinatorica*. Lenny 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. Trevor Lloyd built a nice Java-version editor for *Combinatorica* graphs and Kirg 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.

Lauren Cowles, Alan 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 Dhamsali made a significant contribution to the original *Combinatorica* by writing many of the functions, managing the testing of the code, and performing sundry tasks. The original work on *Combinatorica* was done at Apple Computer in the summer of 1988. Alan Wylde, 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 Stark, Brian Tria, Alan Tucker, Shipei Zhang), Addison-Wesley (Jan Benes, Laura Likely, Karl Mamangakis), and WRI (Dave Ballman, John Bonadies, Martin Buchholz, Joe Gwinn, Igor Rivin, Monte Sayer, Lisa Shipley, Cameron Smith) people all helped out in one way or another.

On a personal level, we thank Amma, Nanna, Sachi, Geeta, and Rama Rao, who provided encouragement and support for the first author. Mom, Dad, Len, and Reb 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 so we can determine who to blame.

Sriram Pemmaraju
Dept. of Computer Science
University of Iowa
Iowa City, IA 52240-1419
sriram@combinatorica.com

Steven S. Skiena
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794-4400
skiena@combinatorica.com

Dedication

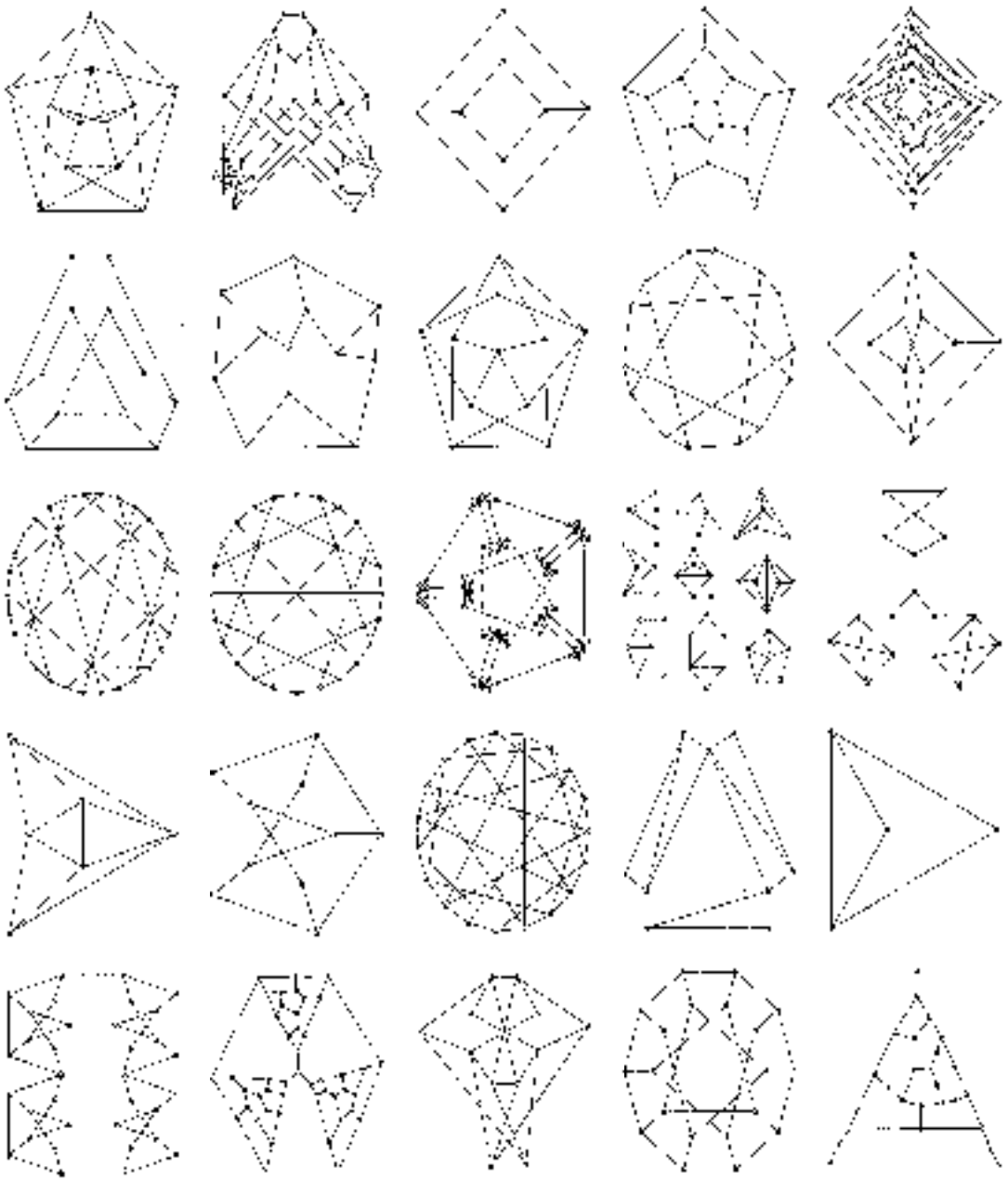
Combinatorialists know how to count their blessings. As a proof of correctness, both of us got the same answer of $2 + e$ when we did the counting.

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

We also dedicate this book to our daughters, Flo and Bonnie. 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 230 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* to 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 overleaf

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[]]; (1)

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 loads the *Combinatorica* package included with *Mathematica*, a necessary step before using any *Combinatorica* functions.

```
In[1]:= <<DiscreteMath`Combinatorica`
```

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 arbitrary items.

```
In[3]:= Permute[{A,B,C}, 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[{3,2,4,5,1}, InversePermutation[{5,2,4,3,1}]]
Out[4]= {1, 2, 3, 4, 5}
```

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

```
In[5]:= MinimumChangePermutations[{a,b,c}]
Out[5]= {{a, b, c}, {b, a, c}, {c, a, b}, {a, c, b},
   {b, c, a}, {c, b, a}}
```

The ranking function takes a permutation of 1 through *n* and associates a unique integer in the range 0 through *n*! − 1 to it.

```
In[6]:= RankPermutation[{6, 8, 7, 1, 5, 4, 3, 2}]
Out[6]= 22 955
```

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

```
In[7]:= UnrankPermutation[2, 9]
Out[7]= {3, 8, 7, 1, 6, 4, 5, 2, 0}
```

With 20! = 6 distinct permutations of these elements, we are likely to stumble on 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.

```
In[8]:= Table[RandomPermutation[3], {20}]
Out[8]= {{2, 1, 0}, {1, 2, 0}, {2, 1, 0}, {2, 0, 1},
  {1, 2, 0}, {2, 0, 1}, {2, 1, 0}, {1, 0, 2},
  {1, 2, 0}, {0, 1, 2}, {1, 0, 2}, {2, 1, 0},
  {1, 0, 2}, {2, 0, 1}, {2, 1, 0}, {2, 1, 0},
  {2, 0, 1}, {0, 2, 1}, {2, 0, 1}, {1, 2, 0}}
```

Any permutation can be decomposed into disjoint cycles, providing an alternate and useful way of viewing permutations. A cycle of length 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.

```
In[9]:= p = RandomPermutation[10]; {p, ToCycles[p]}
Out[9]= {{1, 6, 2, 10, 4, 8, 0, 7, 5, 3},
  {{1}, {6, 4, 10, 3, 2}, {9, 8, 7, 5}, {0}}}
```

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 colored a, 2 beads colored b, and 1 bead colored c. *Combinatorica* provides extensive support for Pólya theory.

```
In[10]:= BuchholzPolynomial[5, {a, b, c}, Cycle]
                3  3      4  2      3  2      2  4      5
Out[10]= a^5 + a  b + 3 a  b  - 4 a  b  + 3 a  b  + a  b  +
     6    2    4      3  2      2  3
    b  + a  c + 5 a  b  c + 10 a  b  c + 10 a  b  c +
     4  2      3  3      2  2  2
    3 a  b  c + 7 a  b  c  + 10 a  b  c  + 16 a  b  c  +
     2  4  2      3  3      2  4      4  2      5  5
    10 a  b  c + 3 a  b  c  + 6 a  b  c  + 10 a  b  c  +
     6  6
    a  + b  c  + c
```

Substituting 1 for each of the variables a, b, and c in the above polynomial gives us the total number of 5-bead necklaces that can be constructed using an infinite supply of beads of each of 3 colors.

```
In[11]:= % /. {a -> 1, b -> 1, c -> 1}
Out[11]= 30
```

The number of inversions (pairs of out-of-order elements) in a permutation is equal to that of its inverse.

```
In[12]:= {p=RandomPermutation[60]; {Inversions[p],
        Inversions[InversePermutation[p]]}
Out[12]= {1035, 1035}
```

This constructs 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.

```
In[13]:= Subsets[{1,2,3,4}]
Out[13]= {{}, {4}, {3, 4}, {3}, {2, 3}, {2, 3, 4}, {2, 4},
  {2}, {1, 2}, {1, 2, 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 it. Since the lead element is placed in first, the k-subsets of a given set are generated in lexicographic order.

```
In[14]:= KSubsets[{1,2,3,4,5},3]
Out[14]= {{1, 2, 3}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4},
  {1, 3, 5}, {1, 4, 5}, {2, 3, 4}, {2, 3, 5}, {2, 4, 5},
  {3, 4, 5}}
```

BinarySearch	LexicographicPermutations
DerangementQ	LongestIncreasingSubsequence
Derangements	MinimumChangePermutations
DistinctPermutations	NextPermutation
EncroachingListSet	PermutationQ
FromCycles	PermutationType
FromInversionVector	PermutationWithCycle
HeapSort	Permute
Heapify	RandomHeap
HideCycles	RandomPermutation
IdentityPermutation	RankPermutation
Index	RevealCycles
InversePermutation	Runs
InversionPoset	SelectionSort
Inversions	SignaturePermutation
InvolutionQ	ToCycles
Involutions	ToInversionVector
Josephus	UnrankPermutation

Combinatorica functions for permutations.

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

Combinatorics functions for subsets.

AlternatingGroup	ListGraphs
AlternatingGroupIndex	ListPolyhedra
CycleIndex	MultiplicationTable
CycleStructure	NecklacePolynomial
Cycles	OrbitInventory
Cyclic	OrbitRepresentatives
CyclicGroup	Orbits
CyclicGroupIndex	Ordered
Dihedral	PairGroup
DihedralGroup	PairGroupIndex
DihedralGroupIndex	PermutationGroup
EquivalenceClasses	SetwiseRelation
KSubsetGroup	SymmetricGroup
KSubsetGroupIndex	SymmetricGroupIndex

Combinatorics functions for Polyhedronics combinatorics.

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 its 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

a grouping of all the elements into nonempty, nonintersecting subsets. Finally, a Young tableau is a two-dimensional structure of integers $1, \ldots, n$, where the number of elements in each row is defined by an integer partition of n, the elements of each row and column are in increasing order, and the rows are left-justified. These four related combinatorial objects have a host of interesting applications and properties.

Here are the 11 integer partitions of 6. Observe that they are given in reverse lexicographic order.

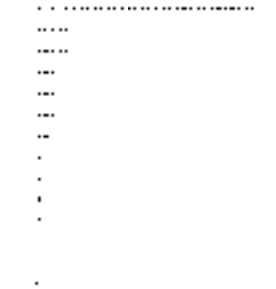
```
In[15]:= Partitions[6]
Out[15]= {{6}, {5, 1}, {4, 2}, {4, 1, 1}, {3, 3},
   {3, 2, 1}, {3, 1, 1, 1}, {2, 2, 2}, {2, 2, 1, 1},
   {2, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1}}
```

Related to the number of permutations is the number of subsets of n elements, the number of partitions of n grows slowly. Therefore, complete tables can be generated for larger values of n.

```
In[16]:= Table[ Length[Partitions[i]], {i,1,20}]
Out[16]= {1, 2, 3, 5, 7, 11, 15, 22, 30, 42, 56, 77, 101,
   135, 176, 231, 297, 385, 490, 627}
```

Ferrers diagrams represent partitions as patterns of dots. They provide a useful tool for visualizing partitions, because moving the dots around provides a mechanism for proving bijections between classes of partitions. Here we construct the Ferrers diagram of a random partition of 100.

```
In[17]:= FerrersDiagram[RandomPartition[100]]
```



A composition of n is a particular arrangement of nonnegative integers whose sum is n. Here, every composition of 5 into 3 parts is generated exactly once.

```
In[18]:= Compositions[5,3]
Out[18]= {{0, 0, 5}, {0, 1, 4}, {0, 2, 3}, {0, 3, 2},
   {0, 4, 1}, {0, 5, 0}, {1, 0, 4}, {1, 1, 3}, {1, 2, 2},
   {1, 3, 1}, {1, 4, 0}, {2, 0, 3}, {2, 1, 2}, {2, 2, 1},
   {2, 3, 0}, {3, 0, 2}, {3, 1, 1}, {3, 2, 0}, {4, 0, 1},
   {4, 1, 0}, {5, 0, 0}}
```

Set partitions represent the ways we can partition distinct elements into subsets. Among other things, they are useful for representing orderings and clusterings.

```
In[19]:= SetPartitions[3]
Out[19]= {{{1, 2, 3}}, {{1}, {2, 3}}, {{1, 3}, {2}},
   {{1, 2}, {3}}, {{1}, {2}, {3}}}
```

The list of tableaux of shape [2,2,1] illustrates the amount of freedom available for arranging numbers in tableau structures. The smallest element always is 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 third row.

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

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

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

The hook length formula can be used to count the number of tableaux for one shape. Using the hook length formula over all partitions of 8 computes the number of tableaux on 8 elements.

```
In[22]:= NumberOfTableaux[10]
Out[22]= 9496
```

The function RandomTableau selects each one of the 7,195,760 tableaux of this shape with equal likelihood.

```
In[23]:= TableForm[ RandomTableau[{6,5,5,4,3,2}] ]
Out[23]//TableForm=
1   2   4   7   10  11  21
3   5   12  15  19
6   7   13  16  25
9   14  20  22
8   18  24
17  22
```

Any sequence of n^2+1 distinct integers must contain either an increasing or a decreasing subsequence of length $n+1$. This result has an elegant proof using tableaux. Even for this example, at least one of the two numbers is at least 8.

```
In[24]:= p = RandomPermutation[7^2 + 1];
         Map[Length, LongestIncreasingSubsequence[#]&,
         {p, Reverse[p]}]
Out[25]= {10, 10}
```

Compositions	NextPartition
DominatingIntegerPartitions	Partitions
DominationLattice	PartitionsQ
FerrersDiagram	RandomComposition
NextComposition	RandomPartition
	TransposePartition

Combinatorics functions for integer partitions and compositions.