Combinatorics for Computer Science

S. Gill Williamson

Combinatorics for Computer Science

S. Gill Williamson
University of California
at San Diego

COMPUTER SCIENCE PRESS

Computer Science Press, Inc.
1803 Research Boulevard
Rockville, Maryland 20850

1 2 3 4 5 6 89 88 87 86 85

Library of Congress Cataloging in Publication Data

Williamson, S. Gill (Stanley Gill)
Combinatorics for computer science.

Includes index.
1. Combinatorial analysis. 2. Electronic data processing—Mathematics. I. Title.
QA164.W55 1985 511'.6 84-17018
ISBN 0-88175-020-4

"Which shall be the Laureate's notebook?"

"'Sbodikins! I am wholly fuddled! Eight species of common notebook?"
"Sixteen, sir; sixteen, if I may," Bragg said proudly. "Ye may have

A thin plain cardboard folio,
A thin plain cardboard quarto,
A thin plain leather folio,
A thin ruled cardboard folio,
A fat plain cardboard folio,
A thin plain leather quarto,
A thin ruled cardboard quarto,
A fat plain cardboard quarto,
A thin ruled leather folio,
A fat ruled cardboard folio,
A fat plain leather folio,
A thin ruled leather quarto,
A fat ruled cardboard quarto,
A fat plain leather quarto,
A fat ruled leather folio, or
A fat ruled leather quarto."

"Stop!" cried Ebenezer, shaking his head as though in pain. "'Tis the Pit!"

—John Barth, *The Sotweed Factor*

PREFACE

This book is the result of a beginning graduate-level course taught by the author at the University of California, San Diego, for the last eight years. The students have had diverse backgrounds with the majority of them being applied mathematics or computer science majors. Many topics were covered during this period that do not appear in this book. The topics presented here are those that, in our opinion (author and students), contain the most basic and useful ideas for the computer scientist and applied mathematician. More recently, selections from this book have been the basis for an upper-division undergraduate course (see SUGGESTIONS ON HOW TO USE THIS BOOK).

This book's organization reflects my preference for presenting this material in a "seminar style," with many trips to the blackboard by students. Each PART is divided into a BASIC CONCEPTS chapter followed by four TOPICS chapters. The two PARTS reflect the general division of combinatorics into "enumeration" and "graph theory," although we have taken considerable liberties with the classical approach to both of these subjects. These PARTS are independent of each other and can be done in either order.

For most students, even very clever ones, the process of learning to express their mathematical ideas both orally and in writing is a somewhat painful experience. For pure mathematicians, this learning process traditionally takes place in the first rigorous course in analysis or algebra. The desire to provide similar training for applied mathematicians and computer scientists through the presentation of material more directly related to their needs was a principal concern in the organization of this book. Combinatorial mathematics provides an ideal subject area for students to learn basic techniques of proof. At the same time, we have tried to convey, through many figures and examples, the important role of intuition in the process of developing an explanation of a mathematical concept.

Finally, some comments regarding data structures and complexity of algorithms are in order. Basic data structures are introduced early in this book, and the student is periodically encouraged to think about the complexity of various algorithms. However, it is a mistake for the student at this level to be overly concerned with "optimality" of algorithms in terms of theoretical running time estimates. Optimality and complexity results are generally asymptotic in nature. If algorithms A for a family of problems $\{P_n\}$ has worst-case complexity $O(n)$, then algorithm B = "Use whatever algorithm seems to work best on your

computer for solving P_n. If no answer has been found by the end of 10,000,000,000 years, switch over to algorithm A'' also has worst-case complexity $O(n)$. In other words, to be really useful, complexity results must take into consideration programming and system-related factors that involve the actual program being constructed. On the positive side, the attempt to create theoretically optimal algorithms has led to the invention of many interesting and imaginative data structures. It is the general form of these data structures and not the optimality results per se that should be the first concern of the student. It is better for the student to experiment with these data structures, even if the algorithms produced are "suboptimal," than it is to memorize a collection of optimal algorithms. Combinatorial mathematics provides a powerful intuitive or "geometric" framework for the discussion of algorithmic concepts. The systematic exploitation of this geometry of algorithms is emphasized over the complexity of algorithms in this book.

S. Gill Williamson

ACKNOWLEDGMENTS

I am greatly indebted to my many fine graduate students who, over the years, have contributed to the contents of this book. David Perlman contributed to the material on constructive Polyá theory. Dennis White, in his dissertation and subsequent research on enumeration and symmetry, has contributed substantially to that subject. Chris Parrish's work on multivariate umbral calculus added to my understanding of that class of problems. My interest in planarity algorithms and related questions began with a series of lectures by Tony Trojanowski. Rod Canfield's lectures on sorting largely influenced the contents of that chapter. The chapter on triconnectivity, and other aspects of the material on graph theory, were inspired by the work of Phong Vo and Wayne Dick. Some important examples were contributed by Christine Alfaro. Most recently, a number of helpful ideas were contributed by my students Sue O. Hart, Paul Kaschube, and Mark Mummy. Sue Hart, in particular, read the entire manuscript and found and corrected numerous errors.

I also wish to express my thanks to my colleague and friend, Dominique Foata, for providing me with the opportunity to visit France during the academic year 1979–80 when I presented much of the material on graph theory to a helpful, critical, and highly capable audience at the Université Louis Pasteur, Strasbourg.

During the years that this material and many other topics were presented to and by graduate students, the activities of the class continued into the summer as a seminar. This seminar was inspired and organized by my colleague, Adriano Garsia, who has generously shared his many ideas and insights with me and all of my students. In addition, I have enjoyed workng with and have learned much from my colleagues Ed Bender, Jay Fillmore, and Gérard Viennot. For me personally, no acknowledgments would be complete without stating that my original interest in the field of combinatorics was inspired by the impressive combinatorial works of N. G. de Bruijn, G.-C. Rota, and M. P. Schutzenberger.

In regard to the preparation of this manuscript, I wish to thank Neola Crimmins who is not only a great technical typist, but an alert and most helpful critic as well. Also, I wish to thank my friend and long-time secretary, Elaine Morici, for an important assist at a difficult time in the preparation of the manuscript. The helpfulness and patience of the librarians at UCSD's Science and Engineering Library helped make the task of writing this book more pleasant. In particular, I wish to thank Beverlee French for her help on many occasions and, in particular, for explaining the workings of MATHFILE and INSPEC to me.

SUGGESTIONS ON HOW TO USE THIS BOOK

Here are some ways this book has been used in various classroom and programmatic situations:

1. In a small (fewer than 15 students), beginning-level graduate class, the material can be presented by the students in the style of a seminar. One may start with either PART I or PART II, each of which contains ample material for a semester course. Each PART begins with a BASIC CONCEPTS chapter which should be completed first. The STUDY GUIDES which follow these SUGGESTIONS are designed to facilitate the making of assignments for class presentation. It seems best to keep the presentations fairly short so that four or five students may make their presentations per class session. There should be a number of written assignments, at least during the first part of the course. Careful proofs should be demanded. After completing the BASIC CONCEPTS chapter, one can choose selections from the various TOPICS chapters according to the interests of the students and the instructor. Except for Chapters 7 and 8, these TOPICS chapters are essentially independent of each other. The seminar format sometimes results in slower coverage of the material than if the instructor were to give most of the presentations. If time is a factor, the instructor can present the material. The benefit to the students of presenting the material themselves generally seems well worth the price of a slightly slower pace.

2. In a small, upper-division undergraduate course, the seminar format described above can be followed. One must go slower and emphasize the BASIC CONCEPTS chapters. Generally, upper-division courses where this material has been used have been larger (40 to 50 students). In this case one can again start with PART I or PART II. The material is presented by standard classroom lectures. Written assignments are made each week. It was found helpful in such classes to periodically have the students work exercises and examples in class. The instructor can help the students get started and can give them hints. Students who finish first can then help the others, and so on. Generally, one or two programming projects have been assigned each quarter. One way to assign projects is to divide the class into small groups based on programming ability, including one good programmer in each group as "group leader." Each group is given a different assignment and demonstrates their software to the instructor

at the end of the course. A question for each group may be included on the final exam to see that all in that group have learned at least the basic ideas of the project. A typical one-quarter course from PART I might be Chapter 1, Chapter 2, and selections from Chapter 3, 4, or 5. A one-quarter course from PART II might include Chapter 6 and Chapter 7 together with a related programming project, or Chapter 6 together with selections from Chapters 7 and 9.

3. In a program which includes both a graduate and undergraduate course covering this material, the two BASIC CONCEPTS chapters (plus other selections as indicated in 2 above) may be covered in the undergraduate course. A two-quarter undergraduate course may be followed by a two-quarter graduate course. Some of the better undergraduate students may then take the graduate course. If the undergraduate course is not a prerequisite to the graduate course, one will be faced with a class where some of the students will not have studied the BASIC CONCEPTS chapters. In this case it is essential to have a rapid review of the BASIC CONCEPTS chapters. To speed things up, the instructor, perhaps together with some of the students who have taken the undergraduate course, can present the material with all students doing written exercises. One can then proceed to a selection of material from the TOPICS as in 1 above.

CONTENTS

PART I

Linear Order

Chapter 1

Basic Concepts of Linear Order

We shall be concerned with studying basic finite sets from a point of view that facilitates computations with these sets. Permutations, subsets, graphs, tree structures, partially and totally ordered sets, etc., are interesting in their own right. They are also the fundamental building blocks for describing and constructing a variety of algorithms. Our point of view will be motivated largely by these algorithmic considerations. The construction and manipulation of linear lists is one of the most fundamental techniques in the design and analysis of algorithms. We begin by looking at the idea of a *linear order* from a somewhat "formal" point of view. In the process we develop some basic ideas to be used later on.

1.1 DEFINITION.

Let S be a set. A relation on S is a function from the Cartesian product, $S \times S$ to any set T with two elements.

For example, $T = \{0,1\}$, $T = \{-1,+1\}$, and $T = \{\text{false, true}\}$ could all be used as the range T of a relation ρ on S. If the set T is fixed and S is finite, then it is easily seen that there are 2^p, $p = |S|^2$, functions from $S \times S$ to T (if S is a finite set, $|S|$ denotes the *cardinality* or *number of elements* of S). The notion of a relation is, of course, a triviality in full generality. Two special classes of relations, however, play an important descriptive role in the study of algorithms. For the next definition, let $\rho : S \times S \to T$ be a relation. Fix $T = \{\text{false, true}\}$. We shall use the equivalent but more suggestive notation "$x \rho y$" for "$\rho(x,y) = \text{true}$" and "$x \not\rho y$" for "$\rho(x,y) = \text{false}$."

1.2 DEFINITION.

A relation ρ on S is

(1) *Reflexive* if for all $x \in S$, $x \rho x$.
(2) *Symmetric* if for all $x, y \in S$, $x \rho y$ implies $y \rho x$.
(2') *Antisymmetric* if for all $x, y \in S$, $x \rho y$, and $y \rho x$ implies $x = y$.
(3) *Transitive* if for all $x, y, z \in S$, $x \rho y$ and $y \rho z$ implies $x \rho z$.

A relation ρ that satisfies 1, 2, and 3 is called an *equivalence relation*. A relation ρ that satisfies 1, 2', and 3 is called an *order relation*.

The general structure of equivalence relations is easy to understand because of the well-known correspondence between equivalence relations and partitions of a set.

1.3 DEFINITION.

Let S be a set. A *partition* of S is a collection $\mathcal{C}$ of subsets of S such that $\bigcup_{A \in \mathcal{C}} A = S$, and if A and B are elements of $\mathcal{C}$, then either A = B or A and B are disjoint. The elements of $\mathcal{C}$ (which are subsets of S by definition) are called the *blocks* of $\mathcal{C}$. $\mathcal{C}$ is *discrete* if each block has one element. The empty set $\phi \notin \mathcal{C}$.

Thus, if N^+ is the set of positive integers, then $\mathcal{C} = \{E,O\}$, where E is the set of even numbers and O is the set of odd numbers in N^+, is a partition of N^+. The collection $\{\{1,3,7\}, \{2,4,5,6\}, \{8\}\}$ is a partition of $S = \{1,\ldots,8\}$.

1.4 DEFINITION.

Let ρ be an equivalence relation on S. For each $s \in S$, let E_s be the set $\{x: x \in S, x \rho s\}$. The set E_s is called the *equivalence class of s with respect to ρ or the equivalence class of s.*

1.5 THEOREM.

If ρ is an equivalence relation on a set S, then the collection $\mathcal{C} = \{E_s: s \in S\}$ of all ρ equivalence classes is a partition of S. Conversely, if $\mathcal{C}$ is any partition of S, and x and y are elements of S, then define $x \rho y$ if x and y belong to the same block of $\mathcal{C}$. Then ρ is an equivalence relation on S, and $\mathcal{C}$ is the collection of equivalence classes.

THEOREM 1.5 finds its way into many undergraduate courses in mathematics (discrete math, real analysis, logic, algebra, group theory, linear algebra). The reader should attempt to reconstruct the proof and consult a reference if necessary. Although the general structure of equivalence relations is quite clear from THEOREM 1.5, particular relations might not obviously be equivalence relations at first glance. Transitivity, in particular, is sometimes a bit tricky to verify. Once the axioms are verified for ρ, then the partition into equivalence classes follows from THEOREM 1.5.

1.6 NOTATION.

The set of integers $1,\ldots,n$ will be denoted by $\underline{n}$. If A and B are sets, we write $f: A \to B$ for a function f with domain A and range B. The set of all such functions will be denoted by B^A (note that if A and B are finite with cardinality $|A| = a$ and $|B| = b$, then $|B^A| = b^a$, hence the notation). The *Image*(f) is the set $\{f(a): a \in A\}$. For each $b \in B$, $f^{-1}(b)$ is called the "inverse image of b" and is the set $\{a: a \in A, f(a) = b\}$. The collection $\mathcal{C} = \{f^{-1}(b): b \in \text{Image}(f)\}$

is a partition of A and is called the Coimage(f). If Image(f) = B, then f is a *surjection*. If Coimage(f) is discrete, then f is an *injection*. If f is both an injection and a surjection, it is a *bijection*. It is easily seen that if $|A| = |B|$ (both finite), then f is an injection if and only if it is a surjection. The injections of $A^{\underline{A}}$ are called the *permutations* of A, A finite.

As an example of the above ideas, consider $f \in \underline{6}^{\underline{6}}$ where $f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 1 & 3 & 2 & 2 \end{pmatrix}$. The Image(f) = {1,2,3}. Coimage(f) = {{1,3}, {5,6}, {2,4}}.

This function can be written in *one-line notation* as (1 3 1 3 2 2). This notation specifies the function if the domain is known and specified in some order. Some permutations of $\underline{6}$ in one-line notation are (5 6 3 2 1 4) or (4 2 3 5 6 1).

We shall see many examples in the text of equivalence relations. We mention a few examples here (it is conventional to use a symbol such as $\sim$ rather than ρ when working with equivalence relations).

1.7 EXAMPLES OF EQUIVALENCE RELATIONS.

(1) $S = \underline{2}^{\underline{3}}$ with $f \sim g$ if Image(f) = Image(g). The equivalence classes are {(1 1 1)}, {(1 1 2), (1 2 1), (2 1 1), (1 2 2), (2 1 2), (2 2 1)}, and {(2 2 2)}. We use one line notation for all functions.

(2) $S = \underline{2}^{\underline{3}}$ with $f \sim g$ if Coimage(f) = Coimage(g). The equivalence classes are {(1 1 1), (2 2 2)}, {(1 1 2), (2 2 1)}, {(1 2 1), (2 1 2)}, and {(1 2 2), (2 1 1)}.

(3) $S = \underline{2}^{\underline{3}}$ with $f \sim g$ if Max(f) = Max(g). There are two equivalence classes.

(4) $S = \underline{2}^{\underline{3}}$ with $f \sim g$ if f is a cyclic shift of g: (1 1 2),(2 1 1), and (1 2 1) are cyclic shifts of each other.

(5) $S = \underline{2}^{\underline{3}}$ with $f \sim g$ if f(1) + f(2) + f(3) = g(1) + g(2) + g(3).

For the reader who knows a little graph theory:

(6) Let G = (V,E) be a graph. Define an equivalence relation on V by $x \sim y$ if there is a path in G from x to y. The equivalence classes are used to define the connected components of G.

(7) Let G = (V,E) be a graph. Define an equivalence relation on E by $e \sim f$ if e and f lie on the same simple (not self-intersecting) cycle of G. The equivalence classes are used to define the biconnected components of G. Check transitivity here. Assume $e \sim e$.

(8) Let $S = \{(a,b): a \text{ and } b \text{ integers}, b \neq 0\}$. Define $(a,b) \sim (a',b')$ if $ab' = ba'$. This is the equivalence relation used in the formal definition of the rational numbers.

(9) Let $\{a_n\}$ and $\{b_n\}$ be two infinite sequences of rational numbers. Define $\{a_n\} \sim \{b_n\}$ if $\lim_{n \to \infty} (a_n - b_n) = 0$. Such an equivalence relation is used in the formal development of the real number system.

The general structure of order relations is more complex than that of equivalence relations. The notation $x \leq y$ or $x \preceq y$ is often used for order relations rather than $x \rho y$. A set S together with an order relation $\preceq$ is often called a "partially ordered set" or "poset." We write $(S, \preceq)$ to designate such a poset. We use $x \prec y$ to mean $x \preceq y$, but $x \neq y$.

1.8 DEFINITION.

Let $(S, \preceq)$ be an ordered set. We say y *covers* x if $x \prec y$ and if, for all $z \in S$, $x \preceq z \preceq y$ implies $x = z$ or $y = z$. If y covers x, we write $x \overset{*}{c} y$ and we say that y is a successor of x and x a predecessor of y.

1.9 EXAMPLES OF ORDERED SETS.

(1) $(R, \leq)$ Real numbers with usual ordering.
(2) $(N, \leq)$ Positive integers with usual ordering.
(3) Given a set A, let $S = \mathcal{P}(A)$, the set of all subsets of A. Then $(S, \subseteq)$ is a poset where "$\subseteq$" denotes the usual set inclusion.
(4) $(\pi(A), \preceq)$, where $\pi(A)$ = partitions of a set A, and $\pi_2 \preceq \pi_1$ if π_2 is a refinement of π_1, e.g., if $A = \underline{8}$, $\pi_1 = \{\{1,3,5\}, \{2,4,6\}, \{7,8\}\}$ and $\pi_2 = \{\{1,3\}, \{5\}, \{2,6\}, \{4\}, \{7,8\}\}$, then $\pi_2 \preceq \pi_1$. (Blocks of π_1 are split further to get blocks of π_2).
(5) $(N, \preceq)$, where $x \preceq y$ if and only if $x|y$ ("x divides y").
(6) $\underline{r}^{\underline{d}}$ (set of functions from $\underline{d}$ to $\underline{r}$), with $f \preceq g$ if and only if $f(i) \leq g(i)$ for all i.

1.10 DEFINITION.

Given a poset $P = (S, \leq)$, let $P_{cov} = (S, \overset{*}{c})$. We create a diagram of P_{cov} by connecting a to b with a line *if and only if* a covers b. This diagram of P_{cov} is called the *Hasse diagram* of P.

1.11 HASSE DIAGRAM FOR $S = \underline{12}$, WITH $x|y$ AS THE ORDER RELATION.

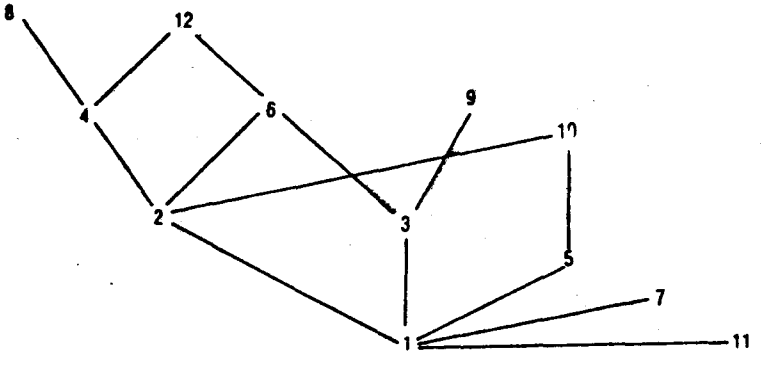

Figure 1.11

1.12 HASSE DIAGRAMS OF $(\mathcal{P}(\underline{2}),\subseteq)$, ALL SUBSETS OF $\underline{2}$ WITH INCLUSION AND $(\underline{5}, \leq)$.

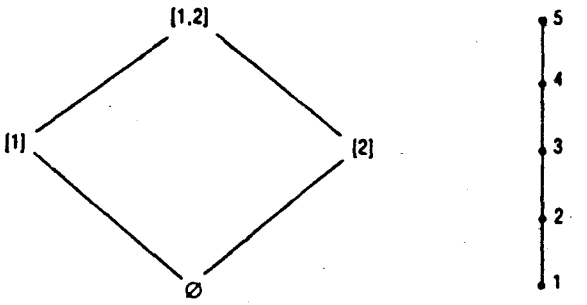

Figure 1.12

1.13 EXERCISE.

(1) Let $\mathcal{M}_n$ be the set of all $n \times n$ matrices with real entries. Define a relation $\sim$ on $\mathcal{M}_n \times \mathcal{M}_n$ by $(A,B) \sim (C,D)$ if $A + D = B + C$. Show that $\sim$ is an equivalence relation on $\mathcal{M}_n \times \mathcal{M}_n$.

(2) Let $\mathcal{M}_n$ be as in (1). Define $A \sim B$ if there is a nonsingular matrix P such that $P\,A\,P^{-1} = B$. Show that $\sim$ is an equivalence relation on $\mathcal{M}_n$. A student who has studied *linear algebra* should be able to give a good description of the equivalence classes for this equivalence relation.

(3) Let ρ_1 be a relation on S and ρ_2 a relation on T. Define a relation ρ_3 on $S \times T$ by $(s,t)\rho_3(s',t')$ if $s\rho_1 s'$ *and* $t\rho_2 t'$. Show that ρ_3 is an equivalence relation if both ρ_1 and ρ_2 are equivalence relations. Show ρ_3 is an order relation if both ρ_1 and ρ_2 are order relations.

(4) Construct the Hasse diagram of $(\mathcal{P}(\underline{4}), \subseteq)$ and $(\underline{18}, |)$. See EXAMPLES 1.9(3) and 1.9(5). See Figure 1.11 for $(\underline{12}, |)$.

(5) Let ρ be a reflexive, symmetric relation on S. Define a relation ρ' on S by $s\rho' t$ if there exists some sequence $u_1,\ldots,u_p$ in S such that $s\rho u_1, u_1\rho u_2,\ldots,u_p\rho t$. Show that ρ' is an equivalence relation on S.

(6) Referring to EXAMPLES 1.9(3) and 1.9(6), consider $(\mathcal{P}(\underline{d}), \subseteq)$ and $(\{0,1\}^{\underline{d}}, \leq)$. For $A \in \mathcal{P}(\underline{d})$, let $f_A \in \{0,1\}^{\underline{d}}$ be defined by $f_A(x) = 0$ if $x \notin A$, 1 if $x \in A$. Show that the map $\varphi(A) = f_A$ is a bijection from $\mathcal{P}(\underline{d})$ to $\{0,1\}^{\underline{d}}$. Show that $A \subseteq B$ if and only if $f_A \leq f_B$. Such a map φ is called an *order preserving bijection* between the two posets. The function f_A is called the *characteristic function* of A.

There is an extensive and quite fascinating mathematical theory of partially ordered sets. From an algorithmic point of view, however, the most basic techniques involve working with various types of *linear orders*. A poset $(S,\leq)$ is