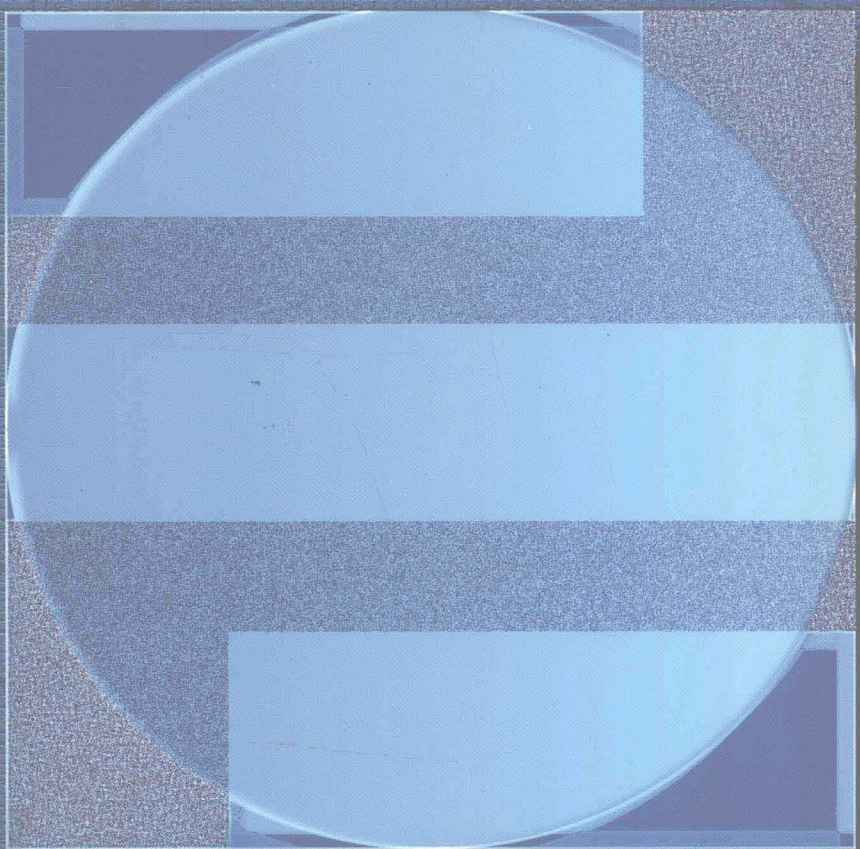
COMBINATORIAL ALGORITHMS

ENLARGED SECOND EDITION



T. C. HU AND M. T. SHING

COMBINATORIAL ALGORITHMS

ENLARGED SECOND EDITION

T. C. HU
Department of Computer Science
University of California, San Diego

M. T. SHING
Department of Computer Science
Naval Postgraduate School
Monterey, California

DOVER PUBLICATIONS, INC.
Mineola, New York

Published in Canada by General Publishing Company, Ltd., 895 Don Mills Road, 400-2 Park Centre, Toronto, Ontario M3C 1W3.
Published in the United Kingdom by David & Charles, Brunel House, Forde Close, Newton Abbot, Devon TQ12 4PU.

Bibliographical Note

This Dover edition, first published in 2002, is an unabridged reprint of the original edition published in 1982 by Addison-Wesley Publishing Co., Inc. Reading, Mass., with the correction of some minor errors and the addition of two new chapters, two appendices and an enlarged index.

Library of Congress Cataloging-in-Publication Data

Hu, T. C. (Te Chiang), 1930–
Combinatorial algorithms.— [Enl. ed.] / T.C. Hu and M.T. Shing.
p. cm.
Includes bibliographical references and index.
ISBN 0-486-41962-2 (pbk.)
1. Combinatorial analysis—Data processing. 2. Operations research—Data processing. 3. Computer algorithms. I. Shing, Man-tak, 1953– II. Title.

QA164 .H8 2001
511'.6—dc21

2001042237

Manufactured in the United States of America
Dover Publications, Inc., 31 East 2nd Street, Mineola, N.Y. 11501

To Dr. Ralph E. Gomory

who taught me integer programming

To my brother Deyi Hu

who taught me English

To the memory of my Aunt, Yu-Fen Hu

who taught me algebra and geometry

To the memory of my parents

Kwong and Mei-Yung

who gave me my wings

To my wife Dawn and my children

Andrew and Leslie who are

the wind beneath my wings

PREFACE

This book presents some combinatorial algorithms common in computer science and operations research. The presentation is to stress intuitive ideas in an algorithm and to illustrate it with a numerical example. The detailed implementation of the algorithms in PASCAL are in a separate manual. No background in linear programming and advanced data structure is needed. Most of the material can be taught to undergraduates while more difficult sections are only suitable to graduate students. Chapters can be read somewhat independently so that the instructor can select a subset of chapters for his course. This book should also be useful as a reference book since it contains much material not available in Journals or any other books.

Chapters One and Two can be used in a one quarter course in network theory or graph algorithms. Chapter One goes in-depth into several shortest-path problems and introduces a decomposition algorithm for large sparse networks. Chapter Two deals with network flows, and contains a large amount of new material, such as the algorithms of Dinic and Kazanov which have never appeared in English before, the optimum communication spanning tree and the description of PERT in terms of longest paths and cheapest cuts. Also in Chapter Two is a section on multi-terminal flows where a subset of nodes are terminal nodes.

Chapters Three and Four cover dynamic programming and backtrack (branch and bound) which are two general optimization techniques. Both topics are usually not covered in detail in computer science departments. Chapter Three introduces the concept of dynamic programming by using examples carefully selected to show the variety of problems solvable by dynamic programming. After the knapsack problem is solved, the periodic nature of the solutions is

discussed. (The solution to the two-dimensional knapsack problem is based on the papers of Gilmore and Gomory.) This chapter ends with a brief discussion of the work of Dr. F.F. Yao. Chapter Four includes standard material on backtracking as well as a detailed description of $\alpha-\beta$ pruning in a game tree. It also gives an example of the Monte Carlo technique of estimating the size of the decision tree.

Chapters Five and Six contain a large amount of new material which should be of interest to computer scientists and operations researchers. Chapter Five introduces the Huffman algorithm, the Hu-Tucker algorithm, including a new reconstruction phase, and the generalization of both algorithms to regular cost functions. This generalization is based on the paper by Hu, Kleitman and Tamaki. Chapter Five also describes and illustrates the Garsia-Wachs construction. Chapter Six deals with heuristic algorithms. It contains the one-point theorem of Magazine, Nemhauser and Trotter and the new bin-packing algorithm of Yao. The treatment of job-scheduling for the tree-constraint is a revision of the author's paper published in 1961.

The subject of Chapter Seven is matrix multiplications. This chapter contains two combinatorial results, the Strassen's result on the multiplication of two large matrices and the results on the optimum order of multiplying a chain of matrices of different dimensions. Although the problem of optimum order can be solved by an $O(n^3)$ algorithm based on dynamic programming, the problem can now be solved by an O(nlogn) algorithm based on combinatorial insights. Since the subject of finding the optimum order is a book by itself, we give the main theorems on the subject and a heuristic O(n) algorithm which has a 15% error bound.

The final chapter, Chapter Eight, introduces the concepts of NP-complete problems. The purpose here is to give the reader some intuitive notions but not a complete treatment since a book has been published dealing with this subject in detail.

It is a pleasure to thank all persons who helped to make this book possible. To the National Science Foundation and Dr. J. Chandra and Dr. P. Boggs of the U. S. Army Research Office for their financial help. To Drs. F. Chin, S. Dreyfus,

F. Ruskey, W. Savitch, A. Tucker, M. Wachs, F. Yao for reading various parts of the drafts. To Professor L. E. Trotter, Jr. and Professor Andrew Yao for reading the next-to-final version of the whole book and made many valuable suggestions. To Mrs. Mary Deo for her effort in editing the earlier versions. To Mrs. Annetta Whiteman for her excellent technical typing of so many versions of the book. To Ms. Sue Sullivan, for skillfully converting the material into the book formats using the UNIX system. To Mr. Y.S. Kuo for preparing the index and writing parts of the manual. And last but most to Dr. Man-Tak Shing, for writing the manual and his technical and general assistance throughout the writing and production.

La Jolla, California

October 19, 1981

T. C. Hu

PREFACE TO THE 2ND EDITION

The revised and expanded edition is really a new book because it has added two new chapters (9 and 10) with materials which have never been published in journals or any other forms. The new materials are the research results of the authors during the last seven years. Chapter 9 unifies many well-known algorithms and invites the reader to mix and invent new algorithms. Chapter 10 deals with the subject of getting minimum cuts in a network directly. Most papers in network flows deal with the flow first and obtain the minimum cut based on the Max Flow Min Cut theorem of Ford and Fulkerson. In Chapter 10, the goal is to get the $n-1$ fundamental minimum cuts of an undirected network, i.e., the Gomory-Hu tree. Our investigations on getting minimum cuts are far from completion. However, we have extended our deadline of delivering the manuscripts of Chapters 9 and 10 by more than one year. Hopefully, some of the insights in Chapter 10 would be of use to most readers.

This edition has also two new appendices. Appendix A updates the material in the first eight chapters of the first edition. Appendix B deals with the subject which we call network algebra. In vector spaces, we have vectors and scalars. In network algebra, we have circles and edges. In the special case of three circles and one edge, we can have the two-valued logic of the boolean algebra. Due to the time and space restrictions, we can only describe the intuitive ideas and illustrate them with numerical examples. Much more work needs to be done. Hopefully, we could write an entirely new book in the future.

We have accepted Dover as publisher due to its tradition of publishing important classical works at very low prices. Many readers have pointed out printing errors (which are corrected in this edition) and have given valuable comments. Special thanks are due to Dr. Paul A. Tucker who worked with us in 1996-1999 and produced the Technical report CS99-625 in June 1999. The production of the camera-ready copy of the new materials is the work of Mr. Robert Ellis, who also made valuable technical suggestions.

It is our hope that the reader will join us in investigating the subject of combinatorial algorithms, and in particular, the subject of network algebra and its applications.

La Jolla and Monterey, California

November 22, 2001

T. C. Hu and M. T. Shing

COMBINATORIAL ALGORITHMS

CONTENTS

Chapter 3. Dynamic Programming

Chapter 4. Backtracking

Chapter 5. Binary Tree

Chapter 6. Heuristic and Near Optimum

Chapter 7. Matrix Multiplication

Chapter 8. NP-complete

Chapter 9. Local Indexing Algorithms

Chapter 10. Gomory-Hu Tree

CHAPTER 1. SHORTEST PATHS

There is no shortest path to success.

§ 1.1 GRAPH TERMINOLOGY

When we try to solve a problem, we often draw a graph. A graph is often the simplest and easiest way to describe a system, a structure, or a situation. The Chinese proverb "A picture is worth one thousand words" is certainly true in mathematical modeling. This is why graph theory has a wide variety of applications in physical, biological, and social sciences. Due to the wide variety of applications, we also have diverse terminology. Papers on graph theory are full of definitions, and every author has his own definitions. Here, we introduce a minimum number of definitions which are intuitively obvious. The notation and terminology adopted here is similar to that of Knuth [18].

A graph consists of a finite set of vertices and a set of edges joining the vertices. We shall draw small circles to represent vertices and lines to represent edges. A system or a structure can often be represented by a graph where the lines indicate the relations among the vertices (the elements of the system). For example, we can use vertices to represent cities and edges to represent the highways connecting the cities. We can also use vertices to represent persons and draw an edge joining two vertices if the two persons know each other.

The reader should keep in mind that graph theory is a theory of relations, *not* a theory of definitions; however, a minimum number of definitions is needed here. Vertices are also called nodes, and edges are also called arcs, branches, or links. We usually assume that there are n vertices in the graph G and at most one edge joining any two vertices and there is no edge joining a node to itself. The vertices are denoted by V_i ($i = 1,2,...,n$) and the edge joining V_i and V_j is denoted by e_{ij}. Two vertices are *adjacent* if they are joined by an edge (the two vertices are also called *neighbors*); two edges are adjacent if they are both incident to the same vertex. A vertex is of *degree* k if there are k edges incident to it.

A sequence of vertices and edges

$$(V_1,e_{12},V_2,e_{23},V_3,...,V_n)$$

is said to form a *path* from V_1 to V_n. We can represent a path by only its vertices as

$$(V_1,V_2,...,V_n)$$

or by only the edges in the path as

$$(e_{12}, e_{23}, \ldots, e_{n-1,n}).$$

A graph is *connected* if there is a path between any two nodes of the graph. A path is of length k if there are k edges in the path. A path is a *simple path* if all the vertices $V_1, V_2, \ldots, V_{n-1}, V_n$ are distinct. If $V_1 = V_n$, then it is called a cycle. In other words, a cycle is a path of length three or more from a vertex to itself. If all vertices in a cycle are distinct, then the cycle is a simple *cycle*. Unless otherwise stated, we shall use the word "path" to mean a simple path, "cycle" to mean a simple cycle, and "graph" to mean a connected graph.

If an edge has a direction (just like a street may be a one-way street), then it is called a *directed edge*. If a directed edge is from V_i to V_j, then we cannot follow this edge from V_j to V_i. Thus in the definition of a path, we want an edge to be undirected or to be a directed edge from V_i to V_{i+1}. In all other definitions, the directions of edges are ignored. A graph is called a *directed graph* if all edges are directed and a *mixed graph* if some edges are directed and some are not. A cycle formed by directed edges is called a *directed cycle* (or *circuit*). A directed graph is called *acyclic* if there are no directed cycles. The words "graph" and "edge" are used for an undirected graph and an undirected edge throughout this section.

A *tree* is a connected graph with no cycles. If a graph has n vertices, then any *two* of the following conditions characterize a tree and automatically imply the third condition.

1. The graph G is connected.
2. The graph has n-1 edges.
3. The graph contains no cycles.

We shall denote a graph by $G = (V; E)$ where "V" is the set of nodes or vertices, and "E" is the set of edges in the graph. A graph $G' = (V'; E')$ is a *subgraph* of $G = (V; E)$ if $V' \subseteq V$ and $E' \subseteq E$.

A subgraph which is a tree and which contains all the vertices of a graph is called a *spanning tree* of the graph. We shall illustrate these intuitive definitions of graph theory in Figure 1.1.

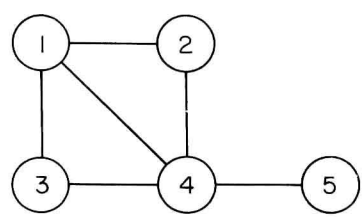


Figure 1.1

There are three paths between V_1 and V_5, namely (V_1, V_2, V_4, V_5), (V_1, V_4, V_5), (V_1, V_3, V_4, V_5). The edges e_{14}, e_{24}, e_{34}, and e_{45} form a spanning tree and so do the edges e_{12}, e_{24}, e_{34}, and e_{45}. Also, we may pick e_{12}, e_{13}, e_{34}, and e_{45} to be the spanning tree. Here the node V_1 is of degree 3 in the graph G but is of degree 2 in the last spanning tree. If the edge e_{45} was directed from V_4 to V_5, then there are still three paths from V_1 to V_5 but none from V_5 to V_1.

In most applications, we associate numbers with edges or vertices. Then the graph is called a *network*. All the definitions of graph theory apply to networks as well. In network theory, we usually use "nodes" and "arcs" instead of "vertices" and "edges".

§1.2 SHORTEST PATH

One of the fundamental problems in network theory is to find shortest paths in a network. Each arc of the network has a number which is the length of the arc.

In most cases, the arcs have positive lengths, but the arcs may have negative lengths in some applications. For example, the nodes may represent the various states of a physical system, where the length associated with the arc e_{ij} denotes the energy absorbed in transforming the state V_i to the state V_j. An arc with negative length then indicates that energy is released in transforming the state V_i into the state V_j. If the total length of a circuit or cycle is negative, we say that the network contains a negative circuit.

The length of a path is the sum of lengths of all the arcs in the path. There are usually many paths between a pair of nodes, say V_s and V_t, but a path with the minimum length is called a *shortest path* from V_s to V_t.

The problem of finding a shortest path is a fundamental problem and often occurs as a subproblem of other optimization problems. In some applications, the numbers associated with arcs may represent characteristics other than lengths and we may want optimum paths where optimum is defined by a different criterion. But the shortest path problem is the most common problem in the whole class of optimum path problems. The shortest path algorithm can usually be modified slightly to find other optimum paths. Thus we shall concentrate on the shortest paths.

If we denote a path from V_1 to V_k by $(V_1, V_2, ..., V_k)$, then $e_{i,i+1}$ must be either a directed arc from V_i to V_{i+1} or an undirected arc joining V_i and V_{i+1} $(i = 1,...,k-1)$. In most applications, we can think of an undirected arc between V_i and V_j as two directed arcs, one from V_i to V_j and the other from V_j to V_i. We usually are interested in three kinds of shortest-path problems: