ALGORITHMS : THEIR COMPLEXITY AND EFFICIENCY

Lydia I. Kronsjö

The Computer Centre
The University of Birmingham

A Wiley–Interscience Publication

JOHN WILEY & SONS

Chichester · New York · Brisbane · Toronto

Library of Congress Cataloging in Publication Data:

Kronsjö, Lydia I.
Algorithms: their complexity and efficiency.

(Wiley series in computing)
'A Wiley–Interscience publication.'
Includes index.
1. Electronic digital computers—Programming.
2. Algorithms. 3. Numerical analysis—Data processing.
4. Computational complexity. I. Title.
QA76.6.K76 519.4 78–22090

ISBN 0 471 99752 8

Photosetting by Thomson Press (India) Limited, New Delhi
and printed in Great Britain by The Pitman Press, Bath, Avon.

ALGORITHMS : THEIR COMPLEXITY AND EFFICIENCY

WILEY SERIES IN COMPUTING

Consulting Editor
Professor D. W. Barron, *Department of Mathematics, Southampton University*

Numerical Control—Mathematics and Applications
P. Bézier

Communication Networks for Computers
D. W. Davies and D. L. A. Barber

Macro Processors and Techniques for Portable Software
P. J. Brown

A Practical Guide to Algol 68
Frank G. Pagan

Programs and Machines
Richard Bird

The Codasyl Approach to Data Base Management
T. William Olle

Data Structures and Operating Systems
Teodor Rus

Computer Networks and their Protocols
D. W. Davies, D. L. A. Barber, W. L. Price and C. M. Solomonides

Algorithms: Their Complexity and Efficiency
Lydia I. Kronsjö

To
my parents Ivan and Varvara,
my father-in-law Erik,
my husband Tom,
my son Tim.

Preface

This book is concerned with the study of algorithms and evaluation of their performance. Analysis of algorithms is a new area of research, and emerged as a new scientific subject during the sixties and has been quickly established as one of the most active fields of study, becoming an important part of computer science. The reason for this sudden interest in the study of algorithms is not difficult to trace as the fast and successful development of digital computers and their uses in many different areas of human activity has led to the construction of a great variety of computer algorithms. At present it is often the case that several different algorithms exist for the solution of a single problem or a class of problems and these algorithms need to be carefully analysed in order to provide a basis for selecting the best one for the purpose. In many cases, analysis of algorithms also leads to the revelation of completely new algorithms that are even faster than all algorithms known before. On the other hand, the study of algorithms has brought about many no less startling discoveries of certain natural problems for which all algorithms are inefficient.

The book is intended as a text for an intermediate course in computer science or computational mathematics which focuses on the basic principles and concepts of algorithms. It requires general familiarity with computers, preferably some courses on programming and introductory computer science. Each chapter is devoted to one particular class of problems and their algorithms. Chapter 1 introduces the subject, while Chapters 2, 3, 4, and 5 discuss four different classes of problems that are termed as numerical, i.e. the mathematical problems, solution of which is of a numerical nature. For such problems numerical accuracy of the computed results is of particular importance. Chapter 6 discusses the problem of (asymptotically) fast multiplication of two numbers and is based on the fast Fourier transforms and Chapters 7, 8, and 9 discuss sorting and searching, the most common non-numerical problems encountered in computing.

The exercises at the end of each chapter are used to provide examples as well as to complete or generalize some proofs.

All algorithms in the text are given in a form of a sequence of steps, each

step describing the actions to be undertaken, in natural English. This seems to be a most neutral way of introducing the algorithms. Actual implementation details, e.g. a specific programming language, various programming tricks etc., are left to an interested reader. The reference list at the end of the book contains published sources for the algorithms and theoretical results discussed in the text.

I would like to thank all the people who have critically read various portions of the manuscript and offered many helpful improvements. In particular, I would like to thank Peter Jarratt, Stuart Hollingdale, Michael Atkinson, Nelson Stevens, and Tom Axford. Sincere thanks go to Ilsie Browne for her excellent typing of the manuscript.

L. I. K.

Birmingham, November 1978

Table of Notations

Written	Denotes
$a \in A$	a is contained in the set A
$x \leftarrow y + z$	x is assigned the value $y + z$
$\lceil a \rceil$	the least integer greater than or equal to a
$\lfloor b \rfloor$	the greatest integer less than or equal to b
$\ln a$	natural logarithm
$a \approx b$	a approximately equal to b
$\mathbf{A}^*$	conjugate transpose of the matrix $\mathbf{A}$
$\mathbf{a}^*$	conjugate transpose of the vector $\mathbf{a}$
$\mathbf{A}^{\mathrm{T}}$	transpose of the real matrix $\mathbf{A}$
$\mathbf{a}^{\mathrm{T}}$	transpose of the real vector $\mathbf{a}$
$\mu[y]$	mean value of the statistical variable y
$\sigma[y]$	standard deviation of the statistical variable y
$a * b$	a is multiplied by b
$a \equiv b$	a is equivalent to b
$x \gg y$	x is much larger than y

Contents

Chapter 1

Introduction

This book will be chiefly concerned with an investigation of algorithms in order to evaluate their performance. This comparatively new field of study is known as algorithmic analysis and forms part of the more general discipline of computer science. In practical terms, a goal of algorithmic analysis is 'to obtain sufficient understanding about the relative merits of complicated algorithms to be able to provide useful advice to someone undertaking an actual computation' (Gentleman, 1973). In broader interpretation, however, algorithmic analysis includes the study of all aspects of performance in computational problem solving, from the preliminary formulation, through the programming stages, to the final task of interpreting the results obtained. We would also like to prove lower bounds on the computation time of various classes of algorithms. In order to show that there is no algorithm to perform a given task in less than a certain amount of time, we need a precise definition of what constitutes an algorithm. First, then, what is an algorithm?

1.1 Definition of an Algorithm

> A procedure consisting of a finite set of unambiguous rules which specify a finite sequence of operations that provides the solution to a problem, or to a specific class of problems, is called an algorithm.

Several important features of this definition must now be emphasized.

First, each step of an algorithm must be unambiguous and precisely defined. The actions to be carried out must be rigorously specified for each case.

Secondly, an algorithm must always arrive at a problem solution after a finite number of steps. Indeed, the general restriction of finiteness is not sufficient in practice, as the number of steps needed to solve a specific problem, although finite, may be too large for practicable computation. A useful algorithm must require not only a finite number of steps, but a reasonable number.

Thirdly, every meaningful algorithm possesses zero or more inputs and provides one or more outputs. The inputs may be defined as quantities which are

given to the algorithm initially, before it is executed, and the outputs as quantities which have a specified relation to the inputs and which are delivered at the completion of its execution.

Fourthly, it is preferable that the algorithm should be applicable to any member of a class of problems rather than only to a single problem. This property of generality, though not a necessity, is certainly a desirable attribute of a useful algorithm.

Finally, we would like to mention that although the concept of an algorithm is a very broad one, in this book we restrict ourselves to algorithms designed to be executed on a computer. Such an algorithm must be embodied in a computer program (or set of programs), and so in the sequel the two terms will be used interchangeably.

1.2 Measures of Efficiency

It is relatively easy to invent algorithms. In practice, however, one wants not only algorithms, one wants good algorithms. Thus, the objective is to invent good algorithms and prove that they are good. The 'goodness' of an algorithm can be appraised by a variety of criteria. One of the most important is the time taken to execute it. There are several aspects of such a time criterion. One might be concerned with the execution time required by different algorithms for solution of a particular problem on a particular computer. However, such an empirical measure is strongly dependent upon both the program and the machine used to implement the algorithm. Thus, a change in a program may not represent a significant change in the underlying algorithm but may, nevertheless, affect the speed of execution. Furthermore, if two programs are compared first on one machine and then another, the comparisons may lead to different conclusions. Thus, while comparison of actual programs running on real computers is an important source of information, the results are inevitably affected by programming skill and machine characteristics.

A useful alternative to such empirical measurements is a mathematical analysis of the intrinsic difficulty of solving a problem computationally. Judiciously used, such an analysis provides an important means of evaluation the cost of algorithm execution.

The performance time of an algorithm is a function of the size of the computational problem to be solved. However, assuming we have a computer program which eventually terminates, solving a particular problem requires only sufficient time and sufficient storage. Of more general interest are algorithms which can be applied to a collection of problems of a certain type. For these algorithms, the time and storage space required by a program will vary with the particular problem being solved. Consider, for example, the following classes of problems, and note the role of the value of the parameter n.

1. Find the largest in a sequence of n integers.
2. Solve a set of linear algebraic equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ real matrix and $\mathbf{b}$ is a real vector of length n.