
THE CLASSIC WORK
NEWLY UPDATED AND REVISED

The Art of Computer Programming

VOLUME 3

Sorting and Searching
Second Edition

DONALD E. KNUTH

Petitioner has added romanettes to pages (i)-(iii).
Otherwise, it leaves the original page numbering.

MICROSOFT
EXHIBIT 1041

Volume 3 / **Sorting and Searching**

THE ART OF COMPUTER PROGRAMMING

SECOND EDITION

Reading, Massachusetts · Harlow, England · Menlo Park, California
Berkeley, California · Don Mills, Ontario · Sydney
Bonn · Amsterdam · Tokyo · Mexico City

T_EX is a trademark of the American Mathematical Society
METAFONT is a trademark of Addison-Wesley

Library of Congress Cataloging-in-Publication Data

Knuth, Donald Ervin, 1938-

The art of computer programming / Donald Ervin Knuth. -- 2nd ed.
xiv, 780 p. 24 cm.

Includes bibliographical references and index.

Contents: v. 1. Fundamental algorithms. -- v. 2. Seminumerical
algorithms. -- v. 3. Sorting and Searching.

ISBN 0-201-03809-9 (v. 1)

ISBN 0-201-03822-6 (v. 2)

ISBN 0-201-89685-0 (v. 3)

1. Electronic digital computers--Programming. I. Title.
QA76.6.K64 1997 001.6'42 73-1830
CIP

Internet page <http://www-cs-faculty.stanford.edu/~knuth/taocp.html> contains
current information about this book and related books.

Copyright © 1998 by Addison Wesley Longman

All rights reserved. No part of this publication may be reproduced, stored in a retrieval
system, or transmitted, in any form, or by any means, electronic, mechanical, photo-
copying, recording, or otherwise, without the prior consent of the publisher. Printed
in the United States of America. Published simultaneously in Canada.

ISBN 0-201-89685-0

Text printed on acid-free paper

2 3 4 5 6 7 8 9 MA 0201009998

Second printing, May 1998

CONTENTS

Chapter 5 — Sorting	1
*5.1. Combinatorial Properties of Permutations	11
*5.1.1. Inversions	11
*5.1.2. Permutations of a Multiset	22
*5.1.3. Runs	35
*5.1.4. Tableaux and Involutions	47
5.2. Internal sorting	73
5.2.1. Sorting by Insertion	80
5.2.2. Sorting by Exchanging	105
5.2.3. Sorting by Selection	138
5.2.4. Sorting by Merging	158
5.2.5. Sorting by Distribution	168
5.3. Optimum Sorting	180
5.3.1. Minimum-Comparison Sorting	180
*5.3.2. Minimum-Comparison Merging	197
*5.3.3. Minimum-Comparison Selection	207
*5.3.4. Networks for Sorting	219
5.4. External Sorting	248
5.4.1. Multiway Merging and Replacement Selection	252
*5.4.2. The Polyphase Merge	267
*5.4.3. The Cascade Merge	288
*5.4.4. Reading Tape Backwards	299
*5.4.5. The Oscillating Sort	311
*5.4.6. Practical Considerations for Tape Merging	317
*5.4.7. External Radix Sorting	343
*5.4.8. Two-Tape Sorting	348
*5.4.9. Disks and Drums	356
5.5. Summary, History, and Bibliography	380
Chapter 6 — Searching	392
6.1. Sequential Searching	396
6.2. Searching by Comparison of Keys	409
6.2.1. Searching an Ordered Table	409
6.2.2. Binary Tree Searching	426
6.2.3. Balanced Trees	458
6.2.4. Multiway Trees	481

6.3. Digital Searching	492
6.4. Hashing	513
6.5. Retrieval on Secondary Keys	559
Answers to Exercises	584
Appendix A — Tables of Numerical Quantities	748
1. Fundamental Constants (decimal)	748
2. Fundamental Constants (octal)	749
3. Harmonic Numbers, Bernoulli Numbers, Fibonacci Numbers	750
Appendix B — Index to Notations	752
Index and Glossary	757

the bit string 01001011101... might mean that the first record in the file refers to a male, the second female, the next two male, etc.

Such methods suffice to handle simple queries about specific attribute values. A slight extension makes it possible to treat range queries, except that a comparison-based search scheme (Section 6.2) must be used instead of hashing.

For Boolean queries like "(MAJOR = MATHEMATICS) AND (RESIDENCE.STATE = OHIO)", we need to intersect two inverted lists. This can be done in several ways; for example, if both lists are ordered, one pass through each will pick out all common entries. Alternatively, we could select the *shortest* list and look up each of its records, checking the other attributes; but this method works only for AND's, not for OR's, and it is unattractive on external files because it requires many accesses to records that will not satisfy the query.

The same considerations show that a multilist organization as described above is inefficient for Boolean queries on an external file, since it implies many unnecessary accesses. For example, imagine what would happen if the index to this book were organized in a multilist manner: Each entry of the index would refer only to the last page on which its particular subject was mentioned; then on every page there would be a further reference, for each subject on that page, to the previous occurrence of that subject. In order to find all pages relevant to "[Analysis of algorithms] and [(External sorting) or (External searching)]", we would need to turn many pages. On the other hand, the same query can be resolved by looking at only two pages of the real index as it actually appears, doing simple operations on the inverted lists in order to find the small subset of pages that satisfy the query.

When an inverted list is represented as a bit string, Boolean combinations of simple queries are, of course, easily performed, because computers can manipulate bit strings at relatively high speed. For mixed queries in which some attributes are represented as sequential lists of record numbers while other attributes are represented as bit strings, it is not difficult to convert the sequential lists into bit strings, then to perform the Boolean operations on these bit strings.

A quantitative example of a hypothetical application may be helpful at this point. Assume that we have 1,000,000 records of 40 characters each, and that our file is stored on MIXTEC disks, as described in Section 5.4.9. The file itself therefore fills two disk units, and the inverted lists will probably fill several more. Each track contains 5000 characters = 30,000 bits, so an inverted list for a particular attribute will take up at most 34 tracks. (This maximum number of tracks occurs when the bitstring representation is the shortest possible one.) Suppose that we have a rather involved query that refers to a Boolean combination of 10 inverted lists; in the worst case we will have to read 340 tracks of information from the inverted file, for a total read time of $340 \times 25 \text{ ms} = 8.5 \text{ sec}$. The average latency delay will be about one half of the read time, but by careful programming we may be able to eliminate the latency. By storing the first track of each bitstring list in one cylinder, and the second track of each list in the next, etc., most of the seek time will be eliminated, so we can estimate the maximum seek time as about $34 \times 26 \text{ ms} \approx 0.9 \text{ sec}$ (or twice this if two independent disk