sided upper confidence limit for $\mu + \lambda\sigma^2$ is $y + \delta\beta s\{\nu^{-\frac{1}{2}}C(\beta s; \nu, \alpha^*) + \beta s/2\}$, where $\delta = (\nu + 1)/(2\lambda\gamma^2)$, $\beta = 2|\lambda|\gamma/(\nu + 1)^{\frac{1}{2}}$, and $\alpha^* = \alpha$ or $1 - \alpha$ according as λ is positive or negative. Similarly, the corresponding lower and upper two-sided limits are $y + \delta\beta s\{\nu^{-\frac{1}{2}}C_{\mp \operatorname{sgn}(\lambda)}(\beta s; \nu, \alpha) + \beta s/2\}$.

One-sided standard limits $C(s; \nu, \alpha)$ are given to 3D in Table 1 for $s = 0.1(0.1)1(0.25)2(0.5)10$, $\nu = 2(1)30(5)50(10)100(20)200(50)500(100)1000$, and α (and $1 - \alpha$) = 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5. Two-sided standard limits $C_{\mp}(s; \nu, \alpha)$ are presented (also to 3D) in pairs in Table 2 for the same range of s, and for $\nu = 2(2)20$, $\alpha = 0.5, 0.8, 0.9, 0.95, 0.98, 0.99, 0.995$. The author recommends quadratic or cubic interpolation on s and ν to obtain intermediate values from these tables.

The present tabular entries were calculated with the aid of extensive tables of critical values [1] by the same author. Abridged versions of the tables under review, together with a discussion of their evolution and construction, have been published by the author [2].

J. W. W.

1. CHARLES E. LAND, *Tables of Critical Values for Testing Hypotheses about Linear Functions of the Normal Mean and Variance* II, ms. deposited in the UMT file. (See RMT 42, *Math. Comp.*, v. 28, 1974, p. 887.)

2. CHARLES E. LAND, "Standard confidence limits for linear functions of the normal mean and variance," *J. Amer. Statist. Assoc.*, v. 68, 1973, pp. 960–963.

55 [12].—DONALD E. KNUTH, *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*, Addison-Wesley Publishing Co., Reading, Mass., 1973, xi + 722 pp. Price $19.50.

One of the more important events to occur in the field of computers and information sciences has been the appearance of the first three volumes of *The Art of Computer Programming* by Donald E. Knuth. In all, a total of seven volumes are to be published, comprising twelve chapters. Volume 1 presents basic material in discrete mathematics, basic programming concepts via a hypothetical computer (MIX), and fundamental algorithms for the manipulation of data structures (see *Math. Comp.*, v. 23, 1969, pp. 447–450, RMT 18). Volume 2 is concerned with random number generation and with approximate and exact computer arithmetic and topics in computer algebra (see *Math. Comp.*, v. 24, 1970, pp. 479–482, RMT 26). Volume 3, the subject of this review, treats the fundamental algorithms for sorting and searching. Volume 4 will deal with combinatorial algorithms, Volume 5 with syntactic algorithms, Volume 6 with mathematical linguistics, and finally Volume 7 with compilers.

Prior to the appearance of the first volume, authorship of a series of books of such scope and magnitude by a single person was the object of reasonable skepticism.

The amount of material to cover is so immense and the degree of detail required for thorough treatment so great that one would expect rather a Bourbaki-like group as the author. The appearance of Volume 1 did a great deal to remove such skepticism and, with the publishing of Volume 2 and now Volume 3, there can no longer be any skeptics. The latest book, Volume 3, together with the first two volumes, represents a degree of scholarship that is unsurpassed in the field of computer science and, indeed, ranks among the very finest of scientific textbooks.

In the preface of Volume 1, Knuth set forth the theme of this set of books: nonnumerical analysis or the analysis of algorithms, which is to be interpreted as the "theory of the properties of particular computer algorithms." One thus finds for the class of algorithms treated in each volume the following essential ingredients:

(1) algorithm descriptions in a clear semiformal step-by-step notation,
(2) programs implementing many of the algorithms in the assembly language of the hypothetical (though representative) computer MIX,
(3) space and computing time analyses of the algorithms and programs, occasionally supplemented by empirical simulation studies.

In addition, a standard feature of each book is a large selection of exercises, divided among the various sections which cover the above three general categories. The exercises range in degree of difficulty from the novice to the expert, some being unsolved research problems, and are ranked accordingly via the author's "logarithmic" scale. The answers, or sketches thereof, for most of the exercises are given in a separate section near the end of each volume and represents a major part of the attractiveness of the books. Also included are sections giving bibliographies and historical surveys of the subject areas, appendices listing important constants to high precision, definitions and references for notation.

Volume 3, entitled Sorting and Searching, fits this general description. It consists of two chapters: Chapter 5, which is devoted to sorting, and Chapter 6, which is concerned with searching. The methods of each of these topics fall into two classes depending on the size of the file involved. If the file (and the required auxiliary storage) fits in the computer's high-speed internal memory, internal sorting and internal searching methods apply; otherwise, relatively slow external memory devices (tapes, disks, drums) must be used and external sorting and external searching methods then apply. This is the basis for the main division of sorting methods in Chapter 5 into internal sorting and external sorting. However, Knuth first introduces the reader to the combinatorial properties of permutations, a topic essential to the analysis of sorting algorithms. This material requires a mathematically mature reader and may be skipped by one not intending to pursue the analysis of the algorithms. The important topics of inversions, permutations of multisets, runs, and tableaus and involutions are treated. One more major section is included in Chapter 5 on optimum sorting methods. This concerns the inherent complexity of sorting and is directed towards minimizing the comparison of

keys in sorting or merging files or selecting the kth largest item; the design of sorting networks is also included. Although most of this material may be omitted without detriment to an understanding of the algorithms, those who are mathematically inclined will find the mathematical analyses enjoyable.

The treatment of internal and external sorting are comprehensive and well organized. Knuth's organization of internal sorting methods, although admittedly "not always clearcut," is based on the "dominant characteristic" of a method and is as follows. The subject is introduced with enumeration methods: comparison counting and distribution counting. Next follow the insertion sort methods: straight and binary insertion, diminishing increments (Shell's method), list insertion, and address calculation (multiple list insertion). Following this are the exchange sort methods, exchange selection ("bubble sort"), merge exchange (Batcher's parallel sort), partition exchange (Hoare's quicksort), and radix exchange. The fourth group is characterized by repeated selection of items in increasing (or decreasing) order of key size: straight selection and its refinements, tree selection and its evolution to William's Heapsort (with improvements by Floyd). The next group is based on merging: straight two-way merge sort and list merge sort. The final category, sorting by distribution, is concerned with radix sorting, especially linked list techniques. The treatment of internal sorting is concluded with a useful summary and an analytical and empirical comparison of these methods.

The treatment of external sorting is no less well organized, presenting the subject as an "internal sorting" phase followed by an "external merging" phase. The method of replacement selection is presented as the most desirable in constructing the initial runs in the internal sorting phase. Several methods are then presented for doing the external merging phase for tapes, with the merge patterns being obtained for somewhat idealized situations. The methods treated are polyphase merge and cascade merge, variations of these when reading tape backward is possible, and a method which oscillates between distribution and merging. The physical properties of tapes and their influence on external sort algorithms (e.g., stop-start delay, rewinding, overlapping I/O, etc.) are discussed, along with the technique of forecasting. Also presented is a revealing illustration of the effectiveness of ten representative merge strategies applied to a file of 100,000 100-character records. Estimates of the run times, obtained from analytically derived formulae, are compared with the actual run times, a history of these runs being displayed on a remarkable foldout. After treating two specialized topics, external radix sorting and two-tape sorting, Knuth concludes the topic of external sorting with a discussion of disk and drum sorting.

Chapter 6 is concerned with searching, a subject related to sorting but not as well developed; this is reflected in its relatively shorter length, being only half as long as Chapter 5. Still most of the searching techniques that are known have been covered. Sequential search methods, being the most simple, are first described and the probability of access considered, including Zipf's law, the so-called "80-20" rule, etc. Quick and quicker sequential search algorithms are then given (the reviewers were disappointed that

these were not followed by a "quickest" search algorithm) and tape searching is considered. There then follow several groups of methods based on key comparisons. Searching ordered sequential tables is treated, including binary search and its improvements, and Fibonacci search. Binary tree search next follows with some treatment of optimum trees. Balanced binary trees (AVL trees) then follow with algorithms for insertion, deletion, concatenation and splitting being described. Also included in this topic is the representation of linear lists by balanced trees and alternatives to them. A topic in external searching is next presented, namely multiway trees (a generalization of binary trees) and a special case called *B*-trees. These are useful for organizing large files on disks and drums.

Digital searching methods, particularly useful for natural language dictionary lookups, are developed. The Trie search of Fredkin, a method used in a system called PATRICIA, and a digital tree search are compared. The reviewers would have expected to see more description of Sussenguth's work, which encompasses Trie search. The treatment of primary key searching is concluded with a long section on hashing. The best general techniques for computing hash functions, those based on simple multiplication and division, are treated in detail, although a number of standard techniques are ignored. Of course, methods of resolving collisions such as "open addressing" and "chaining" are described, along with numerous variations. However, the interesting method of random probing is relegated to an exercise. Methods based on hashing for external searching on direct-access devices are also described. The search methods based on hashing are then compared with each other and with other search methods.

The last section of the book treats retrieval on secondary keys, which arise in queries based on possibly several attributes or keys of the records. Several important techniques for secondary key retrieval are considered: inverted files, compound attributes, binary attributes, superimposed coding, combinatorial hashing, and balanced filing schemes. Owing to the recent development of many of these methods, not as much is known about secondary key retrieval. Consequently, the treatment of this challenging subject is rather brief and is not afforded the same thoroughness which characterizes the preceding sections. Still, the reviewers were delighted to see its inclusion.

Computer science has been looked upon by many as a collection of ad hoc tricks for its practitioners, where indeed the term "science" does not even belong. The work of Knuth and others is establishing the mathematical foundations associated with the many techniques, and is removing the "ad hockery" which has characterized the field. Based on this, one can quantitatively determine which techniques are superior in general or under certain conditions. Volume 3 is a good example of the success of Professor Knuth's viewpoint on the importance of algorithm analysis in computer science.

To be sure, this book is not completely free of flaws, although in this case the flaws appear only in applying the measure of excellence which the author is helping to establish for this field. Regarding errata, the author maintains a complete list of

errors and intended changes (a one or two dollar prize for each new error found by a reader is offered) and makes this available to his readers (see Donald E. Knuth, *Sorting and Searching–Errata and Addenda*, Stanford University Computer Science Department Technical Report #STAN-CS-73-392 (Oct. 1973)). Although the number of corrections is quite large, they are mostly minor, consisting of notational changes, amended references to previous or current work, new, corrected, or reranked exercises and answers, and expansions and corrections to the text. These changes will be incorporated into later editions. Outright misprints are few. Knuth views *The Art of Computer Programming* as an ongoing project and consequently many of the corrections are intended merely to make a good thing better. Regarding the book as a whole, the weaknesses, if there are any, lie in the latter part of Chapter 6, as noted previously. The reviewers would not be surprised to see this portion expanded and subdivided in future editions as the subject develops.

The book is capable of being read at several levels of sophistication, can serve several purposes, and is pertinent to a variety of interests. It may be used for self-study, as a reference source, and as a textbook for a second course in data structures, for the analysis of algorithms, and for concrete complexity studies. The style of writing is interesting and engaging, and the treatment is often so engrossing that it is difficult to stop reading. Those who are interested in the historical aspects of the techniques will find many historical insights into events dating back to the days before the development of computers. The book is mostly self-sufficient, except for some references to material from Volume 1 in discrete mathematics, basic data structures (a little), and, of course, MIX programming. This illustrates a nice coordination of the separate volumes. Volume 3 shares with the preceding volumes the role of a stimulus for research and further development of the area (especially searching) and as somewhat of a forum for the establishment of historical fact, for the standardization of terminology, for the collection of results and problems (new and used). Incidentally, Volume 3, which is 722 pages long, dedicates 102 pages to exercises (73 on sorting and 29 on searching) and contains a 130-page section of answers to the exercises.

The major strength of this book lies not with the amassing of a large number of algorithms, although the organization and coverage is superb, but with the analyses which accompany the algorithms. Many readers with some programming expertise could, if presented with a sketch of a method, devise good algorithms for its implementation. However, it is not the case that they would be able to specify quantitatively why one algorithm should be used in a particular situation in place of another. This is so even for those who are mathematicians. The reviewers know of no book on searching and/or sorting which approaches this book in scope and depth.

There is little doubt that here is a major work that every person working in or teaching programming or computer science must consult and, better yet, seriously read. It is a very worthwhile investment. Moreover, it is highly recommended that every unde

graduate computer science curriculum offer a course based on Volume 3. It is through scholarship such as evidenced by Professor Knuth's work that the science belongs in computer science.

MICHAEL T. MCCLELLAN
JACK MINKER

Department of Computer Science
University of Maryland
College Park, Maryland 20742

56 [12].–R. P. VAN DE RIET, *ABC Algol, A Portable Language for Formula Manipulation Systems*. Part I: *The Language*, Part II: *The Compiler*, Mathematisch Centrum, Amsterdam, 1973, Part I: iv + 173 pp. Price Dfl 18.–; Part II: 116 pp. Price Dfl 12.–.

The ABC Algol system is an extension of Algol 60 designed for algebraic manipulation, with some thought given to more general symbol manipulation. It is a small-scale system, not designed to compete with larger systems such as FORMAC or MATHLAB. It bears some resemblance to Perlis' old Formula Algol System [1], but is somewhat less general in the manipulations that it allows.

ABC Algol provides a datatype *formula*; values of this type can be symbolic expressions. Through an extension to the procedure-defining mechanism of Algol, operators can be defined on operands of type *formula*. The formula manipulation system consists of a set of such definitions, including facilities for manipulating rational numbers and performing a standardized kind of expression simplification.

The system is written in Algol and includes the necessary dynamic storage allocation features and a garbage collector as runtime procedure in Algol. The tract describing the system consists of two volumes, specifying the system and giving the full text of the algebraic manipulation system as well as the compiler.

This system does not seem to contribute anything new; it may be of some interest to specialists in algebraic manipulation because of its self-contained description and its portability. A summary of this publication can also be found in [2].

PAUL ABRAHAMS

Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, New York 10012

1. A. J. PERLIS & R. ITURRIAGA, "An extension to ALGOL for manipulating formulae," *Comm. ACM* (2), v. 7, 1964, pp. 127–130.

2. R. P. VAN RIET, "ABC Algol: A portable language for formula manipulation," *SIGPLAN Notices* (3), v. 9, 1974, p. 1.