EDITOR: George Strawn, NITRD, gostrawn@gmail.com

Don Knuth: Mastermind of Algorithms

George Strawn, *NITRD*

Some algorithms make for "better" programs than others—that is, programs that execute in less time or require less memory. How can we quantify differences to determine which algorithms are better? No one has done more to answer this question than Don Knuth, who has been called the "father of the analysis of algorithms."

In his precollege days in Milwaukee, Knuth already showed signs of having a special talent, and his performance as an undergraduate at Case Institute of Technology was so outstanding that the faculty awarded him a concurrent master's degree. He undertook graduate study in mathematics at Cal Tech and received his PhD in 1963. Knuth remained on the Cal Tech faculty until 1968, when he moved to Stanford University, where he remains to this day as an Emeritus Professor.

In 1962, while still a mathematics graduate student, Knuth decided to write a book on programming languages and compilers—subjects for which he was already an important contributor. The first eight chapters would cover computing "preliminaries," and the last four chapters would deal with his intended subjects. However, as he gathered material, he noticed that the book was becoming very long. He decided that a thorough treatment was better than a brief one, so the "preliminaries" became a series of books that defined the analysis of algorithms. The series was called *The Art of Computer Programming*,[1] because Knuth viewed computer programming as an art as well as a science. Three volumes containing the first six chapters were published between 1968 and 1974 (the latest editions of these volumes were published in 1997 and 1998). Remarkably, the first part of volume 4 wasn't published until 2011!

Here, I describe some of the content of the first three volumes, indicate one reason for the publishing hiatus, and report on Knuth's intentions for the future of the series.

Volume 1: Fundamental Algorithms

The basic concepts covered in Chapter 1 begin with assembling the mathematics and statistics used throughout the volumes to analyze algorithms. The analyses include both the memory requirements and the execution time of the algorithm. Regarding execution time, Knuth tackles both the worst-case time, which depends on worst-case input, and the average-case time, which depends on the statistical distribution of possible inputs.

After laying out the math, Knuth defines a hypothetical computer called *MIX* and then discusses assembly language programming in MIX. Even in the late 1960s, assembly language programming was starting to go out of fashion, but he defended the choice as required for a careful analysis of run time. He also gave a high-level description of each algorithm to be analyzed. Finally, he defined an MIX simulator, so that MIX programs could be run as well as written.

The information structures described in Chapter 2 proved to be the basis for many of the following texts on algorithms and data structures. The basic information structures presented are *linear lists* and *trees*. The implementation techniques considered are *sequential* and *linked* allocation. Among other names, a sequentially allocated linear list is also called an *array*. Trees are often implemented by linking the subtrees to the parent nodes by pointers. Various combinations of these structures and implementation techniques provide the foundation for constructing complex data structures. However, the emphasis needs to be on the operations to be applied to a given data structure. That is, the best way to implement a data structure depends on the mix of operations expected to be used in the application.

Chapter 2 ends with a discussion of storage allocation methods: techniques for carving out a bit of memory in which to create a data structure. Unfortunately, even today, one of the notorious mistakes programmers make is failing to check for "buffer overflow" when allocating space for new data (that is, allocating space beyond that reserved for the purpose). Nefarious hackers often use this flaw to insert malicious code into a computer.

Volume 2: Seminumerical Algorithms

Random number generation is the subject of Chapter 3. Knuth made clear he was discussing *pseudorandom* sequences: deterministic but with characteristics of randomness. Determinism was important, because programs could then be debugged and otherwise rerun with the same (pseudo)random sequence as input. Because modeling and simulating physical systems were—and are—such important computing applications, and because such simulations often involve random selections from a distribution of values, doing the randomization correctly can be the difference between a good simulation and garbage.

Knuth analyzes the *linear congruential method*, $Xi + 1 = (a \times Xi + b) \bmod c$. That is, the next random number, $Xi + 1$, is computed from the previous random number, Xi, and is the remainder of dividing the expression $(a \times Xi + b)$ by c. If a, b, and c are carefully chosen, this simple technique returns a pseudorandom sequence of numbers uniformly distributed between zero and one. He then shows how such a sequence can be turned into a random sequence from any number of other probability distributions.

Computer arithmetic is the subject of Chapter 4. One clever reviewer of Volume 2 said, "Oh, no! I don't want to know that much about computer arithmetic." However, some computer programmers need to understand how to connect abstract numerical operations

One of the notorious mistakes programmers make is failing to check for "buffer overflow" when allocating space for new data.

with real computers (hence the designation as seminumerical). For example, how should exponentiation be implemented? If one naively assumes that computing x^{64} requires 63 multiplications (that's the way it's defined), then reading this chapter would be instructive. If the variable x is replaced by the value x^2 six times, the result is x^{64}, so it actually requires only six multiplications. A slight generalization of this squaring process works for raising a number to any integer power. If there are n bits in the exponent, the algorithm requires at most $2n$ multiplications.

Volume 3: Sorting and Searching

Sorting, the subject of Chapter 5, is divided into internal and external sorting. Internal sorting is when the file to be sorted fits into the computer's main memory. External sorting is required when disks or tapes are necessary to hold the file. Knuth says that sorting algorithms were among the first to be developed for the earliest computers. He suggests that the study of sorting algorithms can teach us much more than sorting. For example, it can teach us how to analyze algorithms and how to approach a computing problem from many directions. One of the slowest internal sorting algorithms described is called *bubble sort*, and one of the fastest is called *quick sort*. Knuth's analysis shows that quick sort would, on average, require approximately six minutes on a one gigaop computer to sort a list of a billion numbers. Bubble sort, on the same computer, would require approximately 192 years!

Searching is the subject of Chapter 6, and it's divided into three techniques: searching by *comparison of keys*, *digital searching*, and *hashing*. Searching by comparison of keys in an ordered file of n keys never takes more than log2 n searches. In digital searching, the search key is considered to be a sequence of small numbers, which act as indices into arrays that contain pointers to the part of the file where the key might be found. In hashing, the key is considered as a single number. In one hashing method, the key is divided by the length of the array that contains the keys. The array length should be chosen as a prime number, and the remainder of the division then produces a random index into the array where the key will be located, with no searching required.

Two keys occasionally hash to the same index, called a *collision*, and Knuth discusses collision resolution techniques. Each of these search methods can be used for both internal and external searching applications. As with sorting, a good algorithm will underlie a speedy search, which Google and other search engines can achieve.

TeX and Metafont

One reason for the long publishing pause between Volume 3 and the first part of Volume 4 was that Knuth got interested in improving the technology to typeset his books. Type setting had started to go digital in the 70s, but was immature. Finding no digital system up to his standards, he decided to create one. Perhaps the fact that his father had owned a small printing business gave him a leg up for this project. He called it TeX, taken from the Greek letters tau epsilon chi and pronounced "tech."[2] One enthusiastic observer called it the greatest advance in printing since Gutenberg, and it was honored by the printing, publishing, and scholarly communities alike.

In a related development, he created *Metafont*,[3] a system for developing digital fonts, and he used it to define a font he called *Computer Modern* for use in his books (and elsewhere). TeX was under development for several decades and has now been stabilized in a final version. A new packaging of TeX, called TeXML, provides an XML front end to the TeX system.

Knuth retired early (in 1992 at the age of 54), because he thought that he would need about 20 years of full-time work to complete *The Art of Computer Programming*. He now projects that Volume 4 (with chapters on combinatorial searching and recursion) and Volume 5 (with chapters on lexical scanning and syntactic techniques) will be completed by 2020 (when he will be 82). He states the following on his webpage (http://www-cs-faculty.stanford.edu/~uno):

> After Volume 5 has been completed, I will revise Volumes 1–3 again to bring them up to date.... Then I will publish a "reader's digest" edition of Volumes 1–5, condensing the most important material into a single book. And after Volumes 1–5 are done, God willing, I plan to publish Volume 6 (on the theory of context-free languages) and Volume 7 (on compiler techniques), but only if the things I want to say about those topics are still relevant and still haven't been said. Volumes 1–5 represent the central core of computer programming for sequential machines; the subjects of Volumes 6 and 7 are important but more specialized.

Regardless of its incomplete status, many people have judged *The Art of Computer Programming* to be the most monumental series of books on computer programming. For example, at the end of 1999, it was named one of the best 12 physical-science monographs of the century by *American Scientist*.[4] Knuth has also received numerous honors, including the Turing Award (1974) and the National Medal of Science (1979).

I'll leave the final word to Bill Gates, who is quoted on the jacket of the third edition of Volume 1 (Addison-Wesley Professional, 1997) as saying, "If you think you're a really good programmer... read [*The*] *Art of Computer Programming*... You should definitely send me a résumé if you can read the whole thing."

References

1. D.E. Knuth, *The Art of Computer Programming, Volumes 1–4A*, 1st ed., Addison-Wesley Professional, 2011.
2. D.E. Knuth, *The TeXbook*, 1st ed., Addison-Wesley Professional, 1984.
3. D.E. Knuth, *The Metafont Book*, 1st ed., Addison-Wesley, 1986.
4. P. Morrison, "100 or So Books that Shaped a Century of Science," *Am. Scientist*, Nov/Dec 1999; www.americanscientist.org/bookshelf/pub/100-or-so-books-that-shaped-a-century-of-science.

George Strawn *is director of the National Coordination Office for the Networking and Information Technology Research and Development Program (NITRD). Contact him at gostrawn@gmail.com.*

IT Professional (ISSN 1520-9202) is published bimonthly by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; voice +714 821 8380; fax +714 821 4010; IEEE Computer Society Headquarters, 1828 L St. NW, Suite 1202, Washington, DC 20036. Annual subscription: $49 in addition to any IEEE Computer Society dues. Nonmember rates are available on request. Back issues: $20 for members, $143 for nonmembers.
Postmaster: Send undelivered copies and address changes to *IT Professional*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.
Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *IT Professional* does not necessarily constitute endorsement by the IEEE or the Computer Society. All submissions are subject to editing for style, clarity, and space.