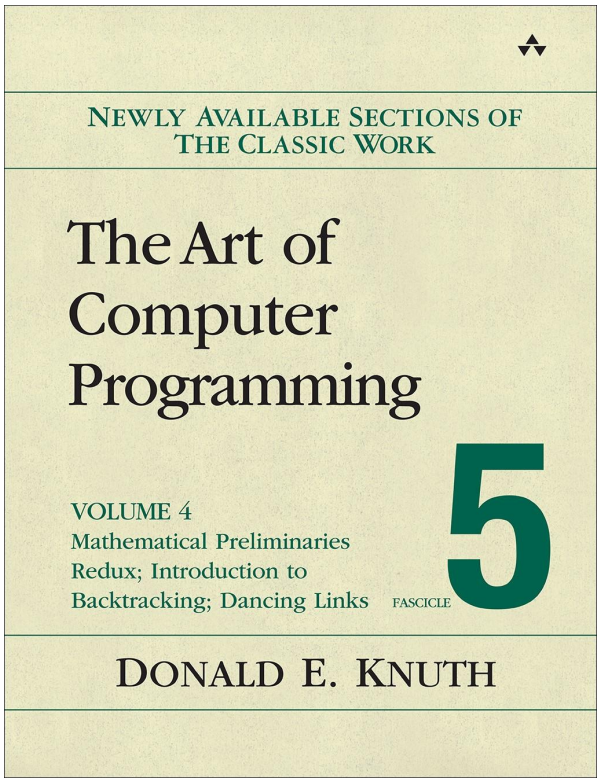
About *The Art of Computer Programming, Volume 4, Fascicle 5*

David Walden

Donald E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 5*.
Addison-Wesley, 2019, 382 pp., softcover, US$34.99,
ISBN 978-0-13-467179-6. tug.org/l/f5-aw



Fascicle 5 for Volume 4B of *The Art of Computer Programming* (*TAOCP*) was published shortly before Christmas 2019. It received some news coverage.[1,2,3]

I cannot presume to review Knuth's new fascicle in the sense of judging the mathematical or algorithm value of its contents. Also, there is no point in judging the presentation — Knuth always works to his own standard of what's "useful and beautiful". (Speaking about writing *TAOCP* with TEX, Knuth says that what he does first has to appeal to him and, if he didn't like something, he would change it.[4]) Instead, I will report a bit *about* the fascicle.

1 Topics in Fascicle 5

As shown on the cover image of Fascicle 5, its major sections are Mathematical Preliminaries Redux, (Introduction to) Backtracking, and Dancing Links.

Mathematical Preliminaries Redux. The fascicle begins with an unnumbered section that adds more probability theory techniques to what was described in section 1.2, Mathematical Preliminaries, of Volume 1 of *TAOCP*. Knuth says that he has "run across" various such techniques in his years of preparing Volume 4 and would have included them in section 1.2 if he "had been clairvoyant enough to anticipate them in the 1960s." These additional mathematical preliminaries are presented with 11.5 pages of explanation and 15.5 pages of exercises.

```
7.1 Zeros and ones
    7.1.1 Boolean basics
    7.1.2 Boolean evaluations
    7.1.3 Bitwise tricks and techniques
    7.1.4 Binary decision diagrams
7.2 Generating all possibilities
    7.2.1 Generating basic combinatorial patterns
      7.2.1.1 Generating all n-tuples
      7.2.1.2 Generating all permutations
      7.2.1.3 Generating all combinations
      7.2.1.4 Generating all partitions
      7.2.1.5 Generating all set partitions
      7.2.1.6 Generating all trees
      7.2.1.7 History and further references
 [Fascicle 5 below; volume 4A above]
    7.2.2 Backtrack programming
        [14 unnumbered but named subsections]
      7.2.2.1 Dancing links
        [20 unnumbered but named subsections]
 [Fascicle 5 above; Fascicle 6 below]
      7.2.2.2 Satisfiability
        [17 unnumbered but named subsections]
```

Figure 1: Contents of Volume 4A, Fascicle 5, and Fascicle 6.[6,7]

Backtracking. This second topic in Fascicle 5 is the first 16 subsubsections of section 7.2.2 of Volume 4 as shown in Figure 1.[5]

The section has 26 pages of explanation about backtrack programming followed by 79 exercises. The explanation introduces and compares several (some historical) algorithms for backtracking, ways to improve on the algorithms, and ways to better program the algorithms (e.g., data structure tricks).

Dancing Links. This third topic is subsection 7.2.2.1 of Volume 4. Knuth's discussion of dancing links takes about 50 pages of explanation followed by three sets of exercises (450 exercises total). The explanation starts by noting that in backtrack programming there is a lot of doing and undoing, and doubly linked lists are helpful for this. But, when elements are dropped out of a list, there is often a better approach than leaving a deleted list item for a garbage collector and creating a new list item when one is added to the list. A better approach at various points in backtracking is to leave a deleted list item where it is in memory and to later reconnect it to the list as if it had never been deleted. This is "dancing links".

Descriptive style. In both the backtracking and dancing links portions of the fascicle, the algorithms and some implementation examples are presented as sort of a verbal flow chart, perhaps including some instructions from Knuth's MMIX computer and assembly language, for example:

B1. [Initialize.] Set $l \leftarrow 1$, and initialize the data structures needed later.
B2. [Enter level l] (Now $P_{l-1}(x_1, \ldots, x_{l-1})$ holds.) If $l > n$, visit $x_1 x_2 \ldots x_n$ and goto B5. Otherwise set $l \leftarrow \min D_l$, the smallest element of D_l.

This method of sketching algorithms has been used at least since the third edition of Volume 1; however, there seem to be fewer instances of sequences of assembly language instructions than I remember being in Volumes 1, 2, and 3.[8,9] As a one-time computer programmer, I wish that Volume 4A and the Volume 4B fascicles used something a little closer to a programming language for showing algorithms.

There is still plenty of discussion in the fascicle of low-level ways to implement algorithms efficiently. Here is some example text from the bottom of page 65: "Interesting details arise when we flesh out the algorithm and look at appropriate low-level mechanisms. There's a doubly linked 'horizontal' list of all the active options that involve it." The discussion continues on the next two pages including two 22x16 diagrams of the list in memory.

Throughout *TAOCP* Knuth refers to prior volumes by section number without a volume number.[5] The presentation style also presumes the reader has read the prior volumes. For example, the definition of "visit" in step B2 above of Algorithm B is given in Volume 1 (page 320); it means "do whatever activity is intended as the tree is being traversed".

Exact cover. Early on in the discussion of dancing links Knuth also introduces exact covering. He gives a simple example of exact covering on page 64. Suppose there are the subsets {c e}, {a d g}, {b c f}, {a d f}, {b g}, and {d e g} of the set S of letters {a b c d e f g}. The first, fourth, and fifth subsets provide an exact cover for S, in that taken together the three subsets contain all the items in S once and only once. This is perhaps clearer if set up as finding an exact cover within a 7x6 matrix of zeros and ones (see Figure 2).

With the exact cover concept introduced and possibilities for efficient implementation discussed, Knuth then gives Algorithm X (for exact cover via dancing links), the suggestion that the reader do an exercise, and then 30 more pages of variations, applications, and optimizations.[10]

	a	b	c	d	e	f	g	
row 1	0	0	1	0	1	0	0	{c e}
row 2	1	0	0	1	0	0	1	{a d g}
row 3	0	1	1	0	0	1	0	{b c f}
row 4	1	0	0	1	0	1	0	{a d f}
row 5	0	1	0	0	0	0	1	{b g}
row 6	0	0	0	1	1	0	1	{d e g}

Figure 2: A combination of Knuth's formulas 5 and 6 on page 64 of Fascicle 5; rows 1, 4, and 5 form an exact cover of the set {a, . . . , g}.

Puzzles. Many puzzles are about exact covers, such as the eight queens puzzle where the goal is to place eight queens on a chess board such that no two queens are in the same column, row, or diagonal. Backtrack programming, perhaps with the help of dancing links, can often be used for finding an exact cover.

(While describing backtracking, Knuth had already noted that one of the best ways to understand backtracking is to execute the basic backtracking algorithm, Algorithm B, by hand for the four queens puzzle — placing four queens on a 4 by 4 chessboard so no queen attacks any other queen. I spent a bunch of time doing this, and it helped me understand both backtracking and the potential subtleties of implementing it in code.)

As the dancing links discussion continues, Knuth develops various algorithms and presents some theorems, often using puzzles as illustrations, e.g., sudoku, polyominoes, and kenken. Knuth chats about this in the fascicle's preface. He sees puzzles as often being the best way to illustrate an algorithm. The odds are good, he says, that a page selected at random in the fascicle will mention a puzzle. He makes the point that the methods that he is describing are useful for creating puzzles as well as solving them. He also discusses the history of the puzzles and sees the fascicle as a contribution to the world of recreational mathematics as well as teaching computer methods.

Knuth has said that Volume 4 covers the kind of algorithms he enjoys most.[11] A quote from the Fascicle 5 preface: "I have had loads of fun writing the other fascicles, but without a doubt this one has been the funnest."

I do wish, in addition to all the puzzle examples, that the fascicle spent more time on real world applications. On the Internet,[12] I found the following statement, which helped me somewhat:

> By far the most relevant, large size, important application of set covering is in personnel shift planning (mainly in large airline companies). There, elements to be covered are the single shifts (or single flights), and sets are legal combinations of

> work/no work schedules. These easily go to millions or even billions of variables, as the number of combinations is huge.

I guess I comprehend that the general topic of Volume 4, combinatorics, is relevant to a wide variety of real life problems.

Knuth lectures. If you haven't yet bought the book and want to know more about dancing links, Knuth's 2018 Christmas lecture is on the topic,[13] and there is a previous (2000) Knuth lecture also on dancing links.[14] Notice that these two lectures on the same topic are years apart; Knuth states in Volume 4A that he has been saving up various methods and examples for years to eventually select among them for Volume 4 of *TAOCP*. (He also emphasizes that there is much he doesn't cover; he has to "cut, cut cut", keeping only what he believes will remain of fundamental importance for decades.)

Knuth's 2019 Christmas lecture[15] is nominally about π; among other things, he gives a bunch of examples of π being used in his books as a source of random data. In the lecture Knuth also talks about Fascicle 5, gives examples (especially puzzle examples) from the fascicle, and promotes it ("it will be a good Christmas present"). Talk about Volume 4B starts at about minute 27 of the lecture's video, first with a bit about Fascicle 6 and then about Fascicle 5. Giving example after example of sudoku, Knuth says that he has studied sudoku so deeply, it is no wonder it took him a long time to write the book. He has said that this book is "tons of fun and teaches a few algorithms on the side".[16]

There is also a Knuth video on the subject of Fascicle 6, satisfiability and SAT solvers.[17] Figure 1 shows where Fascicle 6 fits within the topics of Volume 4 of *TAOCP*. Volume 4A discusses manipulation of 0s and 1s and methods of generating basic combinatorial patterns; Fascicle 5 discusses backtracking and how to do it more efficiently with dancing links; and then Fascicle 6 on satisfiability shows the use of those techniques to develop SAT solvers which can be applied to many, typically massive, real world problems. Knuth touches on some of the latter in the video. In the Preface to Fascicle 6, Knuth says, "The story of satisfiability is a tale of the triumph of software engineering blended with rich doses of beautiful mathematics." For any readers who have been wondering if Knuth's emphasis on efficient algorithms is still relevant with today's computers which are so much more powerful than in 1962 when Knuth started *TAOCP*, Fascicle 6 justifies the continuing thrust for maximum efficiency. Knuth reports that "modern SAT solvers are able to deal routinely with practical problems" involving "many thousands of variables" that were "regarded as hopeless just a few years ago". In Fascicle 6 he is describing a rapidly developing field — especially over the past few decades. Knuth has been actively learning and contributing to the field in various ways, but he says that he knows he must move on. Fascicle 6 is a 2016 snapshot of the field, leaning toward the implementation rather than theoretical side of things; and Knuth hopes that it contains a "significant fraction of concepts that will prove to be the most important as time passes".

2 Main text, exercises, and answers

Fascicle 5 is definitely an unusual book (although not so much for Knuth) in terms of the ratio of main text to exercises and answers. Fascicle 5 has 100 pages of main text, 88.5 pages of exercises (663 exercises total), and 176 pages of answers. Knuth says that he wrote 600 programs while writing this fascicle because he needs to program things to really understand them. A small set of the most important programs are available, written in CWEB, to help readers solve problems.

Digressing for a moment to Fascicle 6 (section 7.2.2.2, on satisfiability — see Figure 1), it has 132.5 pages of main text, 50.5 pages of exercises (526 exercises total), 106 pages of answers, and 310 pages altogether in the book. This gives us, so far in Volume 4B, 232.5 pages of main text, 139 pages of exercises (1,189 exercises), and 282 pages of answers.

In his Notes on the Exercises near the beginning of *TAOCP* Volume 4A, Knuth explains about the benefits of exercises, noting that the exercises allow (are designed for) "self-study as well as for classroom use." He continues, saying

> It is difficult, if not impossible, for anyone to learn a subject purely by reading about it, without applying the information to specific problems and thereby being encouraged to think about what has been read. Furthermore, we all learn best the things that we have discovered for ourselves. Therefore the exercises form a major part of this work; a definite attempt has been made to keep them as informative as possible and to select problems that are enjoyable as well as instructive.

Knuth has had this view a long time. I remember that in Knuth's interview in the book *Mathematical People*,[18] he said that when first in college he had doubts about his abilities. Therefore, he worked all the exercises in the textbook, not just the assigned exercises, and then found he really understood things. Also there was his problem solving course at Stanford: people we know who took this course said it was wonderful — the teacher and students jointly solved new problems with the teacher

using his experience to guide the students in useful directions.[19]

3 What *TAOCP* is and isn't

I previously spoke to what *TAOCP* is and isn't: see my 2011 "appreciation" of Volume 4A in *TUGboat*.[20] I will repeat a couple of points here because there has been a lot of overstatement about *TAOCP* in the popular press which is all many lay people know about the *TAOCP*: (1) *TAOCP* is not a book for teaching computer programming to the typical person learning to program. It does teach (explicitly) analysis of algorithms and (less explicitly) problem solving in the sense of finding algorithms to solve problems. (2) *TAOCP* is not a definitive treatment of computer science (although it may have been closer to comprehensive at the project's start in 1962); it doesn't cover lots of computer science, for example, artificial intelligence, computer networks, and parallel processing. What it does cover, though, it covers unusually deeply; it also contains a significant amount of history of mathematical and computing algorithms. Don't misunderstand me — *TAOCP* was and remains a monumental achievement and superb contribution to computer science and mathematics, regardless of its present state with only two-thirds of Volume 4B published.

TUGboat's reviews editor has asked me about the ideal reader for *TAOCP*. I think this has changed from volume to volume and over time. When Volumes 1, 2, and 3 came out in relatively rapid succession from 1968 to 1973, the volumes and their contents (all organized in distinct volumes) were more or less unprecedented. A practicing computer programmer could turn to the volumes to find the best approach or implementation for a not-unusual problem, e.g., random number generation, hash coding, or sorting. Later, undoubtedly partially stimulated by Knuth's work, many other books and papers on these topics were published, and a programmer needing a method might turn to one of these instead of *TAOCP*.

By the time Knuth got to his planned (not too long) chapter on combinatorial algorithms, the field was expanding rapidly (he "was confronted with ... a prodigious explosion of new ideas!"). Now, extrapolating from the length of fascicles 5 and 6, we can expect a total count in Volumes 4A and 4B of over 1,000 pages, and Knuth's outline continues on to Volumes 4C and 4D. The reader of the primary topics of Volume 4 is probably now a math or algorithms specialist (or someone studying to be one), or someone studying a particular type of puzzle who can find his or her way through the math, or someone developing a solution to a *big* real world problem. General computing practitioners may be more likely to skip to the bits of history Knuth includes; these remain fascinating.

Also, these days practitioners needing techniques covered in Volume 4 may well be able to find needed algorithms on the Web, perhaps even coded in the programming language the programmer is using for his or her larger project, perhaps even provided by an explicit library on the topic (in some cases the code found will be an implementation of an algorithm from Volume 4). On the other hand, having spent as many hours with Fascicle 5 as I have writing this description of the book, I am tempted to spend more time with the book in order to really understand the backtrack and exact cover algorithms, even lacking a problem to solve that needs the methods.

4 In conclusion ...

Fascicle 5 (and Fascicle 6) is another spectacularly impressive production by Knuth. It is incredible that one man can collect the topics and prior publications, understand both the problems and the solutions, sometimes extend them, write hundreds of programs and develop hundreds of exercises and answers, typeset the book himself, and do all this carefully enough that he can offer rewards for mistakes that are found. And, while doing all this, he also finds time for giving the occasional lecture, writing a major work for organ[21], and who knows what other projects he has underway. I eagerly await the next publication from Donald Knuth.

Notes

[1] `slashdot.org/story/364386`

[2] `tug.org/l/f5-xmas-pi`

[3] Readers of this journal likely will know, at least roughly, the story of Don Knuth's decades long work on his magnum opus, *The Art of Computer Programming*: he started it in 1962, published three volumes in 1968, 1969, and 1973, suspended work in 1977 to develop TEX, and returned to work on Volume 4 in 2001. For anyone who wants a more detailed history, there are descriptions on the Web about the original intention for the book(s), how the project was originally received, and how the project has evolved, for example: `tug.org/l/taocp-amsreview`, `tug.org/l/taocp-wiki`, `tug.org/l/taocp-softpano`.

Knuth has noted that one of the reasons things have taken so long is that he keeps discovering new content that needs to be included.

[4] `youtube.com/watch?v=2BdBfsXbST8`, minutes 1:36:00 to 1:38:00.

[5] Section 2.2.2 begins, "Now that we know how to generate simple combinatorial patterns ... we're ready to tackle more exotic patterns ..." Presumably we got this know-how from reading the 223 pages of narrative and

exercises and 149 pages of answers in section 7.2.1 in Volume 4A.

[6] In this and the other books of *TAOCP*, some of the numbered and unnumbered section titles are prefixed with an asterisk. These are sections that Knuth says can be skipped upon first reading and come back to later. Udo Wermuth pointed me to this explanation in Volume 1. Udo, who is well known to readers of *TUGboat*, is one of the few people Knuth acknowledges by name in Volume 4A and Fascicles 5 and 6.

[7] Donald E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 6*. Addison-Wesley, 2016, 310 pp., softcover, US$29.99, ISBN 978-0-13-439760.
tug.org/l/f5-aw

[8] In the days of MIX, before Knuth developed his MMIX RISC computer architecture:
www-cs-faculty.stanford.edu/~knuth/mmixware.html

[9] Martin Ruckert has written a book which reimplements the MIX code examples in Volumes 1, 2, and 3 as MMIX code examples: Martin Ruckert, *The MMIX Supplement: Supplement to The Art of Computer Programming Volumes 1, 2, 3 by Donald E. Knuth*, Addison-Wesley, 2015.

Ruckert has also published several papers in *TUGboat*, including some that stem from his work developing the MMIX supplement, e.g., Computer Modern Roman fonts for ebooks, *TUGboat* 37:3 (2016), pp. 277–280, tug.org/TUGboat/tb37-3/tb117ruckert.pdf.

[10] A worked example of Algorithm X is in Wikipedia:
en.wikipedia.org/wiki/Knuth%27s_Algorithm_X

[11] At youtube.com/watch?v=2BdBfsXbST8, about minute 32:10; keep watching for a few more minutes to hear Knuth describe the original purpose of *TAOCP*. See lexfridman.com/donald-knuth/ for a table of contents for the video. There is a lot of interesting stuff in this interview.

[12] tug.org/l/f5-cover-use

[13] tug.org/l/knuth-xmas18

[14] tug.org/l/knuth-xmas00

[15] tug.org/l/knuth-xmas19

[16] A list of Knuth lectures, including Christmas lectures, is at: tug.org/l/f5-xmas-list.

[17] youtube.com/watch?v=g4lhrVPDUG0

[18] Donald J. Albers and Gerald L. Alexanderson, *Mathematical People: Profiles and Interviews*, Mathematical Association of America, Birkhäuser, 1985.

[19] For example, see:
www-cs-faculty.stanford.edu/~knuth/papers/cs1055.pdf
i.stanford.edu/pub/cstr/reports/cs/tr/89/1269/
i.stanford.edu/pub/cstr/reports/cs/tr/87/1154/

[20] David Walden, An appreciation: *The Art of Computer Programming, Volume 4A*, *TUGboat* 32:2 (2011), pp. 230–232.
tug.org/TUGboat/tb32-2/tb101reviews-knuth.pdf

[21] youtube.com/watch?v=e_1a6bHGQGo

◇ David Walden
walden-family.com/texland