Teaching the Art of Computer Programming (TAOCP)

Frank Ruskey[*]
Dept. of Computer Science
University of Victoria
Victoria, B.C., V8W 3P6
(last-name)@cs.uvic.ca

ABSTRACT

Donald Knuth's magnum opus, *The Art of Computer Programming* (TAOCP), is often bought, frequently cited, sometimes browsed, occasionally read, but almost never used for teaching. The purpose of this paper is to describe the author's experience in teaching two courses, each based on different sections of TAOCP volume 4a, using the pre-fascicles and fascicles that were available at the time. The conclusion reached is that such an adventurous undertaking can be extremely rewarding, not only for the students, but also for the instructor.

Keywords

The Art of Computer Programming, Donald E. Knuth, Advanced undergraduate and graduate student classes.

1. INTRODUCTION

In the 1960's Don Knuth was approached by the publisher Addison-Wesley to produce a book that would summarize the major ideas and results of computer science at the time. Don agreed to the task and so the *Art of Computer Programming* came to life. It soon became apparent that it could not be done in a single book, and Knuth laid out a plan for a series of seven volumes. Volumes 1,2, and 3 appeared in 1968, 1969, and 1973, respectively [4], [5], [6] (the latest editions of these books appeared in 1997, 1998, 1998, respectively). The influence of these books on Computer Science has been incredible.

"At the end of 1999, these books were named among the best twelve physical-science monographs of the century by *American Scientist*, along with: Dirac on quantum mechanics, Einstein on relativity, Mandelbrot on fractals, Pauling on the chemical bond, Russell and Whitehead on foundations of mathematics, von Neumann and Morgenstern on game theory, Wiener on cybernetics, Woodward and Hoffmann on orbital symmetry, Feynman on quantum electrodynamics, Smith on the search for structure, and Einstein's collected papers."

The following statement of Bill Gates from his blog in 1995 is often quoted:

> "If you think you're a really good programmer, or if you want to challenge your knowledge, read the 'Art of Computer Programming' by Donald Knuth. Be sure to solve the problems. ... If some people are so brash that they think they know everything, Knuth will help them understand that the world is deep and complicated. ... It took incredible discipline, and several months, for me to read it. I studied 20 pages, put it away for a week and came back for another 20 pages. You should definitely send me a resume if you can read the whole thing."

Of course, that was many years ago, but consider this more recent exchange in an interview by Maria Klawe (formerly at UBC, then Dean of Science at Princeton, and now President of Harvey Mudd) of Bill Gates in 2005 (the emphases below are mine):

> MARIA KLAWE: One of the things that we all know is that as fields go, computer science probably has the most rapidly changing content, and technology evolves so quickly. And both as head of a computer science department and then now my second job as a dean, having computer science departments, I would always get into these discussions with people in the math department saying, it makes sense that your people teach more courses per semester than the computer scientists do because you're still teaching the same courses that you taught 50 years ago, whereas in the computer science you have to continually redevelop materials so that you really are covering the most up to date things.
>
> And I wondered if you had just ideas that would help us or whether Microsoft has ideas that might help computer science departments stay on top of the most recent technological developments.
>
> BILL GATES: Well, certainly it's the goal of our University Relations Group to make sure that

[*]Research supported in part by NSERC.

> we're talking about what we think the state of the art problems are, finding out from the universities and a lot of dialogue back and forth about that. In a certain sense, yeah, the curriculum has changed, but *say somebody came for an interview and they said, "Hey, I read the 'Art of Computer Programming', that's all I ever read, I did all the problems, I would hire them right then."*
>
> MARIA KLAWE: You'd hire them right then.
>
> BILL GATES: Yeah, that's right.
>
> MARIA KLAWE: So would I.
>
> BILL GATES: Even if they didn't do the double-star problems, I mean, just the fact that they'd read the whole book, you know, *those are the kinds of things you need to know to be a good programmer.* Actually, there's some of that you don't even need to know, but *the kind of algorithmic thinking that's promoted there.*
>
> *So in a sense, we want to teach the same thing, but we'd like to teach it in a forum that's most interesting.* So, for example, the idea of, OK, let's take some of those basic ideas and program a robot to go somewhere or figure something out, you want to inject that into the field. But what you're really teaching the person about design is pretty much the same as you wanted to teach them 30 years ago. I mean, we still haven't gotten past that. And so there may be rich runtimes that we can give to universities that make sure that the person learning those things feels like they're doing something very cool and very interesting.

Note that Gates is not saying that he would hire the person because they have to be extraordinarily intelligent to read TAOCP; rather it is because having read the book means that they have read things necessary to being a good programmer and solving problems algorthmically.

$\boxed{\text{Why then aren't we all trying to get our students to read and understand TAOCP?}}$

We will get back to this question later.

2. TEACHING FROM TAOCP

Ok, I[1] have to admit that I was indoctrinated from an early age. In 1973 at UCSD I took a course from Clark Crane, a student of Knuth's. Volume 1 was the textbook. It was a course on data structures and assembly language! E.g., we learned about linked lists, not by implementing them in some high-level language, but rather by implementing them in MIXAL, the assembly language for Knuth's idealized machine MIX. Such courses do not exist anymore, but I still remember some of the problems from that course, such as Knuth's crossword problem (1.3.2, problem 23); a delightful problem which I still occasionally put on homework assignments in data structures courses.

[1]Please excuse the non-standard use of the first person here and in the rest of the paper.

But I actually know of only one other course that used the Art of Computer Programming as the primary textbook; although many courses list it for supplementary reading. That one course was similar to the courses described here in the sense that it was focussed on Volume 4, particularly in answering the questions that Knuth asked for help on; see [2]. There have been some books devoted to other books of Knuth, for example courses about Literate Programming and about the MMIX architecture.

3. THE UVIC COURSES

After a long hiatus, mainly devoted to the development of TeX, Knuth resumed work on Volume 4 early in the 21st century. His modus operandi was to issue *pre-fascicles* which were approximately 100 page previews of his writing of various sections of Volume 4. Initially they were "hidden" on his website and called to the attention of selected researchers, which preceded announcement of their availability to the general public. I was fortunate to be one of the ones contacted and thought that it might be interesting to fashion a course around these pre-fascicles. Knuth agreed to let me bind together various pre-fascicles, print them, and sell them to the students at cost. I thought that this provided a unique opportunity for students to get to see a master at work, a chance to get a glimpse at, and perhaps even be involved with, a piece of computer science history.

The two courses that I taught are listed below. In the UVic numbering scheme a 400 level course is a 4th year undergraduate course and a 500 level course is a graduate course. These websites are still active and their content can be viewed.

CSC 483A/583A F01 (2004): Knuth Volume IV
`http://www.cs.uvic.ca/~ruskey/classes/KnuthIV/`.

and

CSC 483A/583A S01 (2009): Zeroes and Ones
`http://www.cs.uvic.ca/~ruskey/classes/ZeroesOnes/`.

3.1 The Knuth Volume IV Course

The purpose of this material was to teach the students the basics of algorithms for exhaustively listing the combinatorial structures that most frequently occur and which have well-understood recursive structures. Examples include multiradix numbers, permutations, combinations, set partitions, numerical partitions, necklaces, and various classes of trees. We also learned about Gray codes for many of these structures. In a Gray code listing, successive strings used in the listing are required to differ by some small amount. For example, in the classic binary reflected Gray code for listing all subsets, each successive binary string is required to differ from its predecessor by a bit flip in one position. Gray code listings are a necessary condition for the development of the so-called "loopless" generation algorithms which are favored by Knuth.

The minimum pre-requisites for the course were a previous course on data structures and a previous course on discrete mathematics; students were advised to have taken at least one further course on theoretical computer science or combinatorics. In this course there was 1 undergraduate student and 11 graduate students; two of the graduate students were from the mathematics department.

This course covered the following topics and sections from TAOCP Volume 4 [7]:

7.2 Generating All Possibilities
- 7.2.1 Combinatorial Generators
  - 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

The marks were distributed 40% for homework (4 assignments), 30% for quizzes (4 20 minute in-class quizzes), and 30% for a project. The students had trouble completing the quizzes in 20 minutes and on a couple of occasions I gave them 1/2 hour.

Here are some typical questions from the assignments in that course.

- *Is the solution to exercise 49 from Section 7.2.1.1 correct? If so prove it, if not provide a correct solution.* This was from the first assignment. The answer was incorrect, and I wanted to illustrate for them early on that it was not always difficult to find an error in these preliminary drafts.
- *Explain the solution to exercise 58 in 7.2.1.3. You may assume the result of exercise 49 without explanation.* Although Knuth provides solutions to all of his exercises, in many cases they are more like indications of the main idea needed to solve it, but many details are missing. So an easy source of good problems is to ask them to explain a solution.
- *Explain the last equality in equation (16) on page 29 of 7.2.1.5.* Here they were asked to explain something in the main text. Knuth's explanations are often rather terse, and so the instructor has to fill in the missing steps, or relegate them to assignments.
- *Develop a ranking algorithm for Algorithm H of Section 7.2.1.4. Your algorithm should be efficient (i.e., polynomial in m and n). What is the rank of 10+10+8+5+5+5+1+1?* And sometimes I just made up new problems.

I gave them much latitude in their project topics but encouraged them to do them on either (a) open problems mentioned in the book or (b) the specific topics that Knuth had mentioned on his website that he wanted readers to check. Here is what he had on his website (before Volume 4a was published):

> '*Thus I would like to enter here a plea for some readers to tell me explicitly, "Dear Don, I have read exercise N and its answer very carefully, and I believe that it is 100% correct," where N is one of the following:*'

Here is a small selection of the list that followed the above comment.

- 7.1.2–76 (Uhlig's cloning algorithm, computes twice as fast as expected)
- 7.1.2–85 and 86 (introduction to Razborov's monotone lower bounds)
- 7.1.3–124 (lower bounds on a basic RAM)
- 7.1.3–179 (online algorithm for components of a bitmap)
- 7.1.4–107 (testing whether a BDD defines a 2SAT instance)
- 7.2.1.3–42 (analysis of an algorithm for near-perfect combination generation)
- 7.2.1.6–33 (representing binary tree links with a single permutation)

3.2 The Zeroes and Ones Course

The purpose of this material was to teach, at a high level, basic facts about dealing with bits. First, we considered Boolean operations and the basics of Boolean algebra and Boolean functions. Next was a consideration of circuits for implementing Boolean functions. Then came study of how to take advantage of the fact that computers organize bits into words and provide operations on words. And lastly we studied binary decision diagrams, which are often the method of choice for representing and manipulating Boolean functions in a computer.

In this course there were 2 undergraduate students and 10 graduate students. I got a lot of grief from my colleagues about the title of this course; but I was just using Knuth's own title for Section 7.1. The second course covered the following topics and sections from TAOCP Volume 4 [7]:

7.1 Zeros and Ones
- 7.1.1 Boolean Basics
- 7.1.2 Boolean Evaluation
- 7.1.3 Bitwise Tricks and Techniques
- 7.1.4 Binary Decision Diagrams

Most of the time was spent in section 7.2.1, which is mainly about Boolean algebra and ramifications, section 7.2.2, mainly about Boolean circuits and their minimization, and section 7.2.3, which is about various ways of taking advantage of word operations in various algorithmic contexts. For example, in 7.2.3 you study things like $X\&(X-1)$ which turns the rightmost 1 in X into a 0. Knuth is often criticized for implementing many of his algorithms in assembly language and section 7.2.3 is rife with little pieces of assembly code for his new machine MMIX. In this section it was particularly helpful to have MMIX for implementing and comparison. For example, MMIX has a `SADD` (population count) instruction that is not particularly easy to implement if you are restricted to C or Java.

These were topics with which I was not already an expert, so much learning time was spent by me. To help ease the load and get the students more involved with the teaching process, each student was required to pick a relevant topic from the fascicles and present it to the class. In the student evaluations this was one part of the course that was criticized. Students did not feel that their fellow students did a good job of presenting the material.

Again there were 4 assignments and 4 quizzes and a final project.

Here is a list of the final projects that were undertaken by the graduate students:

- KY: Median graphs generalize distributive lattices. The prism of any distributive lattice is Hamiltonian. Are the prisms of median graphs Hamiltonian?
- JW: Exercises 7.1.4.215 and 7.1.4.216 on Tatami tilings. *This project led to a couple of papers.*
- JK: 7.1.1-123. Determine the exact number of 10-variable self-dual Boolean functions that are also threshold functions.
- AE: 7.1.4-260: generating all set partitions with ZDDs.
- JS: Investigation of "universal operations" like NAND and NOR.
- AS: Analysis of Knuth's XOR (2-forward, 1-back) game. Can something be said for general n?
- JL: Platologic computation applied to rasterization. *This student is also an engineer at Intel working on graphics hardware.*
- ED: 7.1.4 226. Generation of all simple cycles of a graph using a ZDDs.
- DG: Investigations of broadword computation of exhaustive lists of combinatorial objects such as combinations and well-formed parentheses strings, implemented in MMIX.

3.3 Knuth movies

We have become accustomed to having "supplementary materials" supplied by publishers in the form of presentations, model solutions, example exams, and so on. These supplementary materials do not exist for this course. Of course, the many exercises have solutions, which is often helpful. However, there are several movies on the web of Knuth introducing some of the topics covered in these courses. These are from talks that are given to a audience that consists mainly of undergraduate and graduate students at Stanford University. They are given at a level that a typical upper-level computer science or mathematics student should be able to follow. On several occasions we watched the movie, stopping occasionally when the students had a question or some point required further elaboration. They formed a great supplement for book.

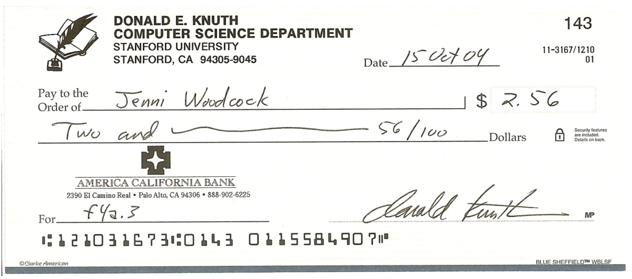
The following "Musings" [3] were relevant to one course or the other. Some were viewed in class and others were suggested for viewing outside of class. The students always seemed to enjoy watching them.

Dec 9, 2008	Fun With ZDDs
Jun 5, 2008	Fun With BDDs
Dec 3, 2007	Sideways Heaps
Oct 24, 2006	Platologic Computation
May 6, 2005	Integer Partitions and Set Partitions
Dec 13, 2004	Sand Piles and Spanning Trees
Dec 16, 2003	Finding All Spanning Trees
Dec 3, 2002	Chains of Subsets
Feb 9, 1999	MMIX: A RISC Computer ...

I did not want to rely on the university wireless system to have these streamed onto my laptop during class, so I got a knowledgeable student to capture the streams and give them to me as `*.avi` files.

4. OUTCOMES

In both of these classes there were several students who received the famous Knuth checks for spotting errors in or making useful suggestions for TAOCP Volume 4. Not only that, but the checks were accompanied by Knuth's handwritten comments about each of the errors or suggestions, even for the suggestions that he decided not to follow. Knuth pays \$2.56 for each error and \$0.32 for each useful suggestion. In October 2008 he stopped writing checks drawn on a real bank, because "it is no longer safe to write personal checks." He started a fictitious bank, and now writes checks in hexadecimal dollars. See Figure 1.


DONALD E. KNUTH
COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY
STANFORD, CA 94305-9045

143

11-3167/1210
01

Date 15 Oct 04

Pay to the Order of Jenni Woodcock $ 2.56

Two and ——— 56/100 Dollars

Security features are included. Details on back.

AMERICA CALIFORNIA BANK
2390 El Camino Real • Palo Alto, CA 94306 • 888-902-6225

For f42.3

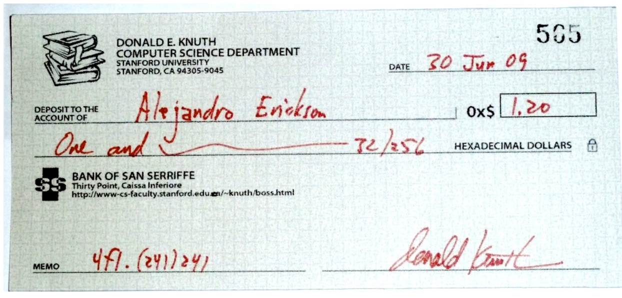
[illegible]

MP

:121031673:0143 0115584907"

©Clarke American

BLUE SHEFFIELD™ WBLSF


DONALD E. KNUTH
COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY
STANFORD, CA 94305-9045

565

DATE 30 Jun 09

DEPOSIT TO THE ACCOUNT OF Alejandro Erickson 0x$ 1.20

One and ——— 32/256 HEXADECIMAL DOLLARS

BANK OF SAN SERRIFFE
Thirty Point, Caissa Inferiore
http://www-cs-faculty.stanford.edu/~knuth/boss.html

MEMO 4f1.(241)241

[illegible]

Figure 1: Scans of checks received by two students. The top check is from an actual bank and is in U.S. dollars. The bottom check is from a mythical bank and is in hexadecimal dollars.

Aaron Williams' thesis topic arose from the first class [11]. Not only that, but Knuth liked one of his results so much that he put it into the Volume 4. The result is about a new way to generate combinations, as represented by binary strings with s 0s and t 1s. Amazingly, Aaron discovered that the following simple rule produces an exhaustive list of all such combinations: Move the leftmost bit and place it after the first 10; if there is no 10 place the bit at the right end. That result was published in [8]. One of Knuth's open questions in a pre-fascicle was answered in [9] and that result is now mentioned in [7].

Jenni Woodcock and I published a paper that arose from trying (and succeeding) to solve a question in TAOCP Volume 4 about Tatami Tilings [10]. That work has been extended, together with Alejandro Erickson, another student from the Zeroes-Ones class [1]. Enumeration and complexity of Tatami tiling problems are the topic of his PhD thesis, which is in progress.

Three students from the courses are named in the Index: Gilbert Lee, Jenni Woodcock, and Aaron Williams.

5. CONCLUSION

Returning to my question: Why then aren't we all trying to get our students to read and understand TAOCP? I think that the answer is multi-faceted, but that some of the reason are: (a) The material is bloody difficult, not only for students but also for instructors. One minute we are considering tricky assembly instructions for multi-linked data structures and the next we are in the midst of using complex analysis for determining the asymptotics of the analysis of an algorithm. One has to learn the art of determining what to try to understand and what to ignore. It is quite humbling to come face-to-face with the incredible amount of knowledge that is presented in these books. (b) Many of the topics found in Volumes 1 and 3 are taught to undergraduate students, but they really do not mesh well with the standard curriculum. You could fashion a super-course from Volume 1 that replaced the standard courses in discrete mathematics, assembly language programming (including subroutines, coroutines, and memory management) and the introductory course on data structures, but I don't really foresee that happening.

But there is no reason not to base advanced undergraduate and graduate courses on the material in the later volumes.

Would I do it again? Definitely. I consider the teaching of these courses as two of the highlights of my career at UVic. It wasn't easy, but it was rewarding. The department has slated me to teach a graduate course in the fall. If Knuth has some more pre-fascicles ready by then, then I will teach from those; otherwise I will teach a course from Volume 4a, combining some of the topics from the two courses discussed in this paper.

6. ACKNOWLEDGMENTS

Thanks to all the students who took these courses. Their energy and enthusiasm was instrumental in making them a success. I also that the referees for their constructive comments and suggestions; unfortunately, there was not time to follow them all.

7. REFERENCES

[1] M. Schurch A. Erickson, F. Ruskey and J. Woodcock. Auspicious tatami mat arrangements. In *The 16th Annual International Computing and Combinatorics Conference (COCOON 2010)*, pages 288–297. LNCS 6169, 2010.

[2] S. Richter J. Kneis, A. Langer and P. Rossmanith. Helping Donald Knuth. `http://tcs.rwth-aachen.de/lehre/Help-Knuth/WS2008/`.

[3] D. E. Knuth. Computer musings. `http://scpd.stanford.edu/knuth/index.jsp`.

[4] D.E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms.* Addison-Wesley, Reading, MA, third edition, 1997.

[5] D.E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Addison-Wesley, Reading, MA, third edition, 1998.

[6] D.E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison-Wesley, Reading, MA, second edition, 1998.

[7] D.E. Knuth. *The Art of Computer Programming, Volume 4: Combinatorial Algorithms, Part 1.* Addison-Wesley, Reading, MA, 2011.

[8] F. Ruskey and A. Williams. The coolest way to generate combinations. *Discrete Mathematics*, 309:5305–5320, 2009.

[9] F. Ruskey and A. Williams. An explicit universal cycle for the $(n-1)$-permutations of an n-set. *ACM Transactions on Algorithms*, 6:12 pages, 2010.

[10] F. Ruskey and J. Woodcock. Counting fixed-height tatami tilings. *Electronic Journal of Combinatorics*, 16:20 pages, 2009.

[11] A.M. Williams. *Shift Gray Codes.* PhD thesis, Dept. of Computer Science, University of Victoria, Victoria, B.C., Canada, 2009.