Joint Review of
Algorithmic Barriers Falling: P=NP?
by Donald E. Knuth and Edgar G. Daylight
Publisher: Lonely Scholar
$20.00 Paperback, 100 pages, 2014

and

The Essential Knuth
by Donald E. Knuth and Edgar G. Daylight
Publisher: Lonely Scholar
$15.00 Paperback, 90 pages, 2014
Reviewer: William Gasarch gasarch@cs.umd.edu

1 Introduction

Both of these books are Edgar Daylight interviewing Donald Knuth. They talk on many topics including computer science, mathematics, and the history of science. Given Donald Knuth's place in our history his perspective is worth listening to. *Algorithmic Barriers Falling: P=NP?* (henceforth ALG) is more about algorithms and theory, *The Essential Knuth* (henceforth KNU) is more about Knuth.

While Knuth is making his points he seems to *not* be criticizing others. I'm not saying *he is careful to not criticize others*, I think being nice just comes naturally to him. As an example, when discussing pointer machines and RAM's he says the following:

RAM's and pointer machines are polynomially equivalent. They differ only when we make finer distinctions, like between linear time and $n\alpha(n)$ (α is the inverse of Ackerman's function). The pointer model hasn't become more popular than the RAM model, because complexity theorists are happiest with a model that makes it easiest to prove theorems.

Those guys have a right to study polynomial fuzzy models, because those models identify fundamental aspects of computation. But such models aren't especially relevant to my own work as a programmer. I treat them with respect but I don't spend too much time with them when they're not going to help me with the practical problems.

2 Very Short Summary

Since the books are short, the review will also be short. I'll just give a list of points that struck me. If you read the books then you may generate a different list.

The following points from ALG struck me:

1. Math papers shouldn't hide how they got to their results.
2. Notation actually drives research. By not using $o(n)$ Knuth forced himself to obtain sharper results.
3. Much asymptotic work has no real application to computing.

4. In the 1960's compiler research represented about 1/3 of computer science. I doubt this is meant as a precise estimate; but suffice to say that it was a large part.

5. It's important to know the history of the field since observing how others created new results may help you to create new results.

6. The history of the Knuth part of the Knuth-Morris-Pratt algorithm: Cook had shown that any set of strings recognized by a 2-way PDA has a linear-time algorithm on a RAM. Knuth went through the construction for the case of pattern matching and came up with a usable algorithm. This surprised him since he didn't think automata theory would ever lead to a simple algorithm that he couldn't come up with using just his programmers intuition.

7. Right after Cook's paper on (what is now called) the Cook-Levin Theorem there was *optimism*— all we need to do is show that SAT is in P and we'll have many other problems in P! This lasted for a few months.

8. LaTeX: while working on it he became an expert on typefaces and fonts. Each letter is somewhat complicated and involves 65 parameters.

9. Knuth thinks that $P = NP$ (yes, that $P = NP$) but that there will still be some sort of distinction within P. Problems that are NP-complete will still be hard, because we won't know explicit polytime algorithms for them — we'll only know that such algorithms exist.

The following points from KNU struck me:

1. Knuth is really a programmer at heart. He may need some esoteric piece of math for an analysis, but his real goal is faster or better programs. It is common in theory (in science? in life?) to use a simple model as a starting point for what you really want to study, but then mistake the model for reality. Knuth never fell into this trap.

2. Knuth was doing statistical analysis of sports (Basketball) way before the moneyball revolution.

3. Knuth wrote a tic-tac-toe program in the early days of computing when it required using symmetries to save space.

4. Dijkstra coined the term *the pleasantness problem* to mean the gap between what we specify (perhaps formally) what we want a program to do and what it really does.

5. Structured programming was a very big breakthrough; however, gotos are sometimes useful.

6. Knuth is bothered by the trend in History of Science to dumb down the science. If these two books were an attempt to counter that trend then they have succeeded. (Actually, even if that was not the intent then they have still succeeded.)

3 Opinion

These are a wonderful books that gives great insights from THE founder of algorithmic analysis. What is most remarkable is that he is truly a computer scientist — he will learn and come up with hard mathematics, but he never loses sight of the original goal: faster real world algorithms for real world problems.