REVIEW OF *ALGORITHM DESIGN*

I bought this while taking an algorithms class earlier this year, because it was a recommended supplemental text. I didn't end up needing it for the class, but having unread books lying around the house weighs on my conscience, so I made time to read it afterward.

Students who were struggling in the aforementioned class often complained that they didn't know how to build the 'intuition' needed for coming up with algorithms. The gap between understanding how a solution works once it's been explained to you, versus understanding how you could come up with that solution if you'd never seen the problem before, can be substantial. (It makes sense that the latter would be difficult to teach—after all, if we had a clear algorithm for designing algorithms, we'd just have computers do the designing instead of bothering to teach humans in the first place.) This book promises to walk you through the process, not just show you the end result:

> Sophisticated algorithms are often best understood by reconstructing the sequence of ideas —including false starts and dead ends—that led from simpler initial approaches to the eventual solution.[1]

I'm not sure it really does this consistently—I suspect many of my classmates would still have felt they were being asked to 'draw the rest of the owl' even if they had read this book.

I'll probably remember this more for its discussions of how to *prove* things (like time/space complexity, or that an algorithm's output is optimal, or bounds on how far from optimal it is). Strategies include:

- Finding a "*progress measure* for the algorithm"[2]
- Proving "the greedy algorithm stays ahead"[3]—e.g., proving that the *i*th element of the greedy solution is at least as desirable as the *i*th element of any optimal solution
- Devising an "exchange argument" where "one considers any possible solution to the problem and gradually transforms it into the solution found by the greedy algorithm without hurting its quality"[4]
- The "pricing method" where you "consider a price one has to pay to enforce each constraint of the problem"[5]
- Creating a "potential" (as in "potential energy") function to use as a progress measure[6]
- Using the simpler "Union Bound" instead of trying to precisely calculate the probability of a union of non-independent events[7]
- Using the "probabilistic method" to prove stuff like "every instance of 3SAT has an assignment satisfying at least 7/8 of the clauses"[8]

I appreciated the sections on co-NP, PSPACE, approximation algorithms for NP-Complete problems, and randomized algorithms, among others, for going into topics where my previous education had stopped short.

1. Jon Kleinberg and Éva Tardos, *Algorithm Design* (Boston: Pearson/Addison-Wesley, 2006), xiv.

2. Ibid., 7.

3. Ibid., 115.

4. Ibid., 116.

5. Ibid., 599.

6. Ibid., 697.
7. Ibid., 712.
8. Ibid., 793.

Posted 2024-10-06 by Jacob Williams on brokensandals.net. If you have feedback, email me!