Nine Algorithms That Changed the Future: The Ingenious Ideas That Drive Today's Computers
John MacCormick

A valuable book for people who work around computers, designed to be accessible to those who don't

MacCormick targets this book at intelligent laypeople, folks who use computers but don't have a formal background in either computer science or mathematics. The book's greatest strength is in the examples he structures to illustrate some fairly deep computer concepts using concrete metaphors such as paint mixing and padlocks.

The algorithms he describes include the key insights that have gone into building search engines such as Google and its predecessor Alta Vista, public key cryptography and digital signatures, data compression, error correction, pattern recognition techniques, and relational databases.

The nature of the algorithms varies. Public-key cryptography and digital signatures are based on very elegant mathematics. Many of the other algorithms are simpler, insights into how people work and clever ways of programming. Many of the things he discusses involve whole families of different algorithms. There are lots of different schemes to compress data, each with advantages and disadvantages, most of which work better with some kinds of data than others. The same seems true of error detection and correction techniques. There is a lot of common sense, but nothing he describes in those realms seems like true genius.

I made my living with relational databases. MacCormick does a good job of describing a couple of the tricks that ensure data integrity, which as he explains is absolutely vital to the functioning of a database. Those tricks include a two-phase commit, rollbacks, and transaction logging. I think he did not devote enough explanation to the power of joins, selects, and the other operators that enable a programmer to easily assemble data in a useful format. Working in a relational database involves a major paradigm shift from working one record or transaction at a time to working in parallel with every element in a database which matches certain criteria. This was central to Codd’s insight; the guarantee of integrity is simply an essential feature of the implementation of that insight.

I'd recommend the MacCormick brush up on his HL Mencken or PT Barnum. You can go broke overestimating the intelligence of the American people. My guess is that the majority of people with patience enough to go through his examples already know more about computers than he expects. However, even a guy like me who has been working with computers pretty constantly since 1958 and had a passing familiarity with every algorithm he discusses certainly benefits from his illustrations.

How is this important? There should at least be footnotes for the mathematically or computer literate. For instance, he describes modulo arithmetic as "clock arithmetic." Every time you past 12 (or the arbitrarily chosen the biggest number, usually prime, on his metaphorical clock, you start over. Just like five hours after eight o'clock is one o'clock.

He uses multiplication to frame out the logic of the concepts of public key cryptography and digital signatures, which are operationally fairly similar. He then switches to exponentiation, which is the method which is really used, because it is not reversible. The book would have been stronger if he had given examples. Just as multiplication has an inverse function, division, exponentiation has a reverse function, logarithms. The difference is that given a number and one of its factors, it is trivial to divide to find the other factor. Conversely, given a number and a modulo exponential of that number, it is difficult to derive the logarithm in a modulo world. I think.

I would have enjoyed an explanation of why the modulo arithmetic works. In his multiplication example he takes advantage of the commutative property of multiplication: the order of the factors doesn't matter. The same is true of exponentiation. (5^3)^4 = (5^4)^3. Most college graduates have been exposed to this fact. I would have enjoyed reading an explanation of why it is also true for modulo arithmetic. In other words, if I raise five to the third power,modulo 11, and raise that to the fourth power,modulo 11, please provide a proof of the proposition that all get the same answer as if I did the operations in reverse. In other words, why does the principle of commutativity remained true in a modulo arithmetic world.

These quibbles aside, I will have to say that his paint mixing metaphor for public-key cryptography provides far and away the clearest explanation I have ever read. It is exactly what he intended: something an intelligent layperson could understand. He has a similarly elegant padlock and key metaphor for digital signatures. The strength of his argument falters a bit when he gets into numeric examples. He chooses one digit numbers for simplicity. In doing so he sacrifices communicating intuitively the power of very large numbers.

I have spent a lifetime with programmers, and I don't think I have known one who would have attempted even to explain these algorithms. I wish MacCormick luck with his intelligent laypeople, but I think it will be of most value to people within the profession to understand the tools they work with every day. A valuable book – glad to have it on my shelf.