Introduction

Algorithms to Live By

Imagine you're searching for an apartment in San Francisco—arguably the most harrowing American city in which to do so. The booming tech sector and tight zoning laws limiting new construction have conspired to make the city just as expensive as New York, and by many accounts more competitive. New listings go up and come down within minutes, open houses are mobbed, and often the keys end up in the hands of whoever can physically foist a deposit check on the landlord first.

Such a savage market leaves little room for the kind of fact-finding and deliberation that is theoretically supposed to characterize the doings of the rational consumer. Unlike, say, a mall patron or an online shopper, who can compare options before making a decision, the would-be San Franciscan has to decide instantly either way: you can take the apartment you are currently looking at, forsaking all others, or you can walk away, never to return.

Let's assume for a moment, for the sake of simplicity, that you care only about maximizing your chance of getting the very best apartment available. Your goal is reducing the twin, Scylla-and-Charybdis regrets of the "one that got away" and the "stone left unturned" to the absolute minimum. You run into a dilemma right off the bat: How are you to know that an apartment is indeed the best unless you have a baseline to judge it by? And how are

you to establish that baseline unless you look at (and *lose*) a number of apartments? The more information you gather, the better you'll know the right opportunity when you see it—but the more likely you are to have already passed it by.

So what do you do? How do you make an informed decision when the very act of informing it jeopardizes the outcome? It's a cruel situation, bordering on paradox.

When presented with this kind of problem, most people will intuitively say something to the effect that it requires some sort of balance between looking and leaping—that you must look at enough apartments to establish a standard, then take whatever satisfies the standard you've established. This notion of balance is, in fact, precisely correct. What most people *don't* say with any certainty is what that balance is. Fortunately, there's an answer.

Thirty-seven percent.

If you want the best odds of getting the best apartment, spend 37% of your apartment hunt (eleven days, if you've given yourself a month for the search) noncommittally exploring options. Leave the checkbook at home; you're just calibrating. But after that point, be prepared to immediately commit—deposit and all—to the very first place you see that beats whatever you've already seen. This is not merely an intuitively satisfying compromise between looking and leaping. It is the *provably optimal* solution.

We know this because finding an apartment belongs to a class of mathematical problems known as "optimal stopping" problems. The 37% rule defines a simple series of steps—what computer scientists call an "algorithm"—for solving these problems. And as it turns out, apartment hunting is just one of the ways that optimal stopping rears its head in daily life. Committing to or forgoing a succession of options is a structure that appears in life again and again, in slightly different incarnations. How many times to circle the block before pulling into a parking space? How far to push your luck with a risky business venture before cashing out? How long to hold out for a better offer on that house or car?

The same challenge also appears in an even more fraught setting: dating. Optimal stopping is the science of serial monogamy.

Simple algorithms offer solutions not only to an apartment hunt but to all such situations in life where we confront the question of optimal stopping. People grapple with these issues every day—although surely poets have spilled more ink on the tribulations of courtship than of parking—and they do so with, in some cases, considerable anguish. But the anguish is unnecessary. Mathematically, at least, these are solved problems.

Every harried renter, driver, and suitor you see around you as you go through a typical week is essentially reinventing the wheel. They don't need a therapist; they need an algorithm. The therapist tells them to find the right, comfortable balance between impulsivity and overthinking.

The algorithm tells them the balance is thirty-seven percent.

* * *

There is a particular set of problems that all people face, problems that are a direct result of the fact that our lives are carried out in finite space and time. What should we do, and leave undone, in a day or in a decade? What degree of mess should we embrace—and how much order is excessive? What balance between *new* experiences and *favored* ones makes for the most fulfilling life?

These might seem like problems unique to humans; they're not. For more than half a century, computer scientists have been grappling with, and in many cases solving, the equivalents of these everyday dilemmas. How should a processor allocate its "attention" to perform all that the user asks of it, with the minimum overhead and in the least amount of time? When should it switch between different tasks, and how many tasks should it take on in the first place? What is the best way for it to use its limited memory resources? Should it collect more data, or take an action based on the data it already has? Seizing the day might be a challenge for humans, but computers all around us are seizing milliseconds with ease. And there's much we can learn from how they do it.

Talking about algorithms for human lives might seem like an odd juxtaposition. For many people, the word “algorithm” evokes the arcane and inscrutable machinations of big data, big government, and big business: increasingly part of the infrastructure of the modern world, but hardly a source of practical wisdom or guidance for human affairs. But an algorithm is just a finite sequence of steps used to solve a problem, and algorithms are much broader—and older by far—than the computer. Long before algorithms were ever used by machines, they were used by people.

The word “algorithm” comes from the name of Persian mathematician al-Khwārizmī, author of a ninth-century book of techniques for doing mathematics by hand. (His book was called *al-Jabr wa’l-Muqābala*—and the “al-jabr” of the title in turn provides the source of our word “algebra.”) The earliest known mathematical algorithms, however, predate even al-Khwārizmī’s work: a four-thousand-year-old Sumerian clay tablet found near Baghdad describes a scheme for long division.

But algorithms are not confined to mathematics alone. When you cook bread from a recipe, you’re following an algorithm. When you knit a sweater from a pattern, you’re following an algorithm. When you put a sharp edge on a piece of flint by executing a precise sequence of strikes with the end of an antler—a key step in making fine stone tools—you’re following an algorithm. Algorithms have been a part of human technology ever since the Stone Age.

* * *

In this book, we explore the idea of *human algorithm design*—searching for better solutions to the challenges people encounter every day. Applying the lens of computer science to everyday life has consequences at many scales. Most immediately, it offers us practical, concrete suggestions for how to solve specific problems. Optimal stopping tells us when to look and when to leap. The explore/exploit tradeoff tells us how to find the balance between trying new things and enjoying our favorites. Sorting theory tells us how (and whether) to arrange our offices. Caching theory tells us how to fill our closets. Scheduling theory tells us how to fill our time.

At the next level, computer science gives us a vocabulary for understanding the deeper principles at play in each of these domains. As Carl Sagan put it, “Science is a way of thinking much more than it is a body of knowledge.” Even in cases where life is too messy for us to expect a strict numerical analysis or a ready answer, using intuitions and concepts honed on the simpler forms of these problems offers us a way to understand the key issues and make progress.

Most broadly, looking through the lens of computer science can teach us about the nature of the human mind, the meaning of rationality, and the oldest question of all: how to live. Examining cognition as a means of solving the fundamentally computational problems posed by our environment can utterly change the way we think about human rationality.

The notion that studying the inner workings of computers might reveal how to think and decide, what to believe and how to behave, might strike many people as not only wildly reductive, but in fact misguided. Even if computer science did have things to say about how to think and how to act, would we want to listen? We look at the AIs and robots of science fiction, and it seems like theirs is not a life any of us would want to live.

In part, that’s because when we think about computers, we think about coldly mechanical, deterministic systems: machines applying rigid deductive logic, making decisions by exhaustively enumerating the options, and grinding out the exact right answer no matter how long and hard they have to think. Indeed, the person who first imagined computers had something essentially like this in mind. Alan Turing defined the very notion of computation by an analogy to a human mathematician who carefully works through the steps of a lengthy calculation, yielding an unmistakably right answer.

So it might come as a surprise that this is not what modern computers are actually doing when they face a difficult problem. Straightforward arithmetic, of course, isn’t particularly challenging for a modern computer. Rather, it’s tasks like conversing with people, fixing a corrupted file, or winning a game of Go—problems where the rules aren’t clear, some of the required information is missing, or finding exactly the right answer would

require considering an astronomical number of possibilities—that now pose the biggest challenges in computer science. And the algorithms that researchers have developed to solve the hardest classes of problems have moved computers away from an extreme reliance on exhaustive calculation. Instead, tackling real-world tasks requires being comfortable with chance, trading off time with accuracy, and using approximations.

As computers become better tuned to real-world problems, they provide not only algorithms that people can borrow for their own lives, but a better standard against which to compare human cognition itself. Over the past decade or two, behavioral economics has told a very particular story about human beings: that we are irrational and error-prone, owing in large part to the buggy, idiosyncratic hardware of the brain. This self-deprecating story has become increasingly familiar, but certain questions remain vexing. Why are four-year-olds, for instance, still better than million-dollar supercomputers at a host of cognitive tasks, including vision, language, and causal reasoning?

The solutions to everyday problems that come from computer science tell a different story about the human mind. Life is full of problems that are, quite simply, *hard*. And the mistakes made by people often say more about the intrinsic difficulties of the problem than about the fallibility of human brains. Thinking algorithmically about the world, learning about the fundamental structures of the problems we face and about the properties of their solutions, can help us see how good we actually are, and better understand the errors that we make.

In fact, human beings turn out to consistently confront some of the hardest cases of the problems studied by computer scientists. Often, people need to make decisions while dealing with uncertainty, time constraints, partial information, and a rapidly changing world. In some of those cases, even cutting-edge computer science has not yet come up with efficient, always-right algorithms. For certain situations it appears that such algorithms might not exist at all.

Even where perfect algorithms haven’t been found, however, the battle between generations of computer scientists and the most intractable real-

world problems has yielded a series of insights. These hard-won precepts are at odds with our intuitions about rationality, and they don't sound anything like the narrow prescriptions of a mathematician trying to force the world into clean, formal lines. They say: Don't always consider all your options. Don't necessarily go for the outcome that seems best every time. Make a mess on occasion. Travel light. Let things wait. Trust your instincts and don't think too long. Relax. Toss a coin. Forgive, but don't forget. To thine own self be true.

Living by the wisdom of computer science doesn't sound so bad after all. And unlike most advice, it's backed up by proofs.

* * *

Just as designing algorithms for computers was originally a subject that fell into the cracks between disciplines—an odd hybrid of mathematics and engineering—so, too, designing algorithms for humans is a topic that doesn't have a natural disciplinary home. Today, algorithm design draws not only on computer science, math, and engineering but on kindred fields like statistics and operations research. And as we consider how algorithms designed for machines might relate to human minds, we also need to look to cognitive science, psychology, economics, and beyond.

We, your authors, are familiar with this interdisciplinary territory. Brian studied computer science and philosophy before going on to graduate work in English and a career at the intersection of the three. Tom studied psychology and statistics before becoming a professor at UC Berkeley, where he spends most of his time thinking about the relationship between human cognition and computation. But nobody can be an expert in all of the fields that are relevant to designing better algorithms for humans. So as part of our quest for algorithms to live by, we talked to the people who came up with some of the most famous algorithms of the last fifty years. And we asked them, some of the smartest people in the world, how their research influenced the way they approached their own lives—from finding their spouses to sorting their socks.

The next pages begin our journey through some of the biggest challenges faced by computers and human minds alike: how to manage finite space, finite time, limited attention, unknown unknowns, incomplete information, and an unforeseeable future; how to do so with grace and confidence; and how to do so in a community with others who are all simultaneously trying to do the same. We will learn about the fundamental mathematical structure of these challenges and about how computers are engineered—sometimes counter to what we imagine—to make the most of them. And we will learn about how the mind works, about its distinct but deeply related ways of tackling the same set of issues and coping with the same constraints. Ultimately, what we can gain is not only a set of concrete takeaways for the problems around us, not only a new way to see the elegant structures behind even the hairiest human dilemmas, not only a recognition of the travails of humans and computers as deeply conjoined, but something even more profound: a new vocabulary for the world around us, and a chance to learn something truly new about ourselves.