In *What Should I Read Next?: 70 University of Virginia Professors Recommend Readings in History, Politics, Literature, Math, Science, Technology, the Arts, and More* edited by Jessica Feldman and Robert Stilling, University of Virginia Press, 2008.

How Computing Changes Thinking

David Evans

In their capacity as a tool,
computers will be but a ripple on the surface of our culture.
In their capacity as intellectual challenge,
they are without precedent in the cultural history of mankind.
Edsger Dijkstra, 1972 Turing Award Lecture

Computing has changed the human condition more than any other technology invented in the past hundred years. Without computing, the Allies may not have won World War II, humans would not have walked on the Moon, and Wal-Mart would be a small store in Arkansas. The commoditization of computing, along with the global communications it enabled, has empowered ordinary people to do things the richest, most powerful, dictators could not even dream of doing twenty years ago.

But the impact of computing goes beyond the geopolitical shifts and everyday conveniences it enabled, to profound revolutions in the way humans understand the world, as well as our own minds and thoughts. Whereas traditional mathematics explores declarative knowledge ("what is"), computer science concerns imperative knowledge ("how to"). Computer scientists study how to describe, understand, and implement information processes. The basic tool is the *procedure*, a precisely defined sequence of steps.

The three most fundamental questions in computing are: (1) *what can be computed?* (2) *what can be computed in a reasonable amount of time?* and (3) *can computers think?*.

The first question emerged from Alonzo Church and Stephen Kleene's work on Lambda Calculus in the 1930s, and was answered formally by Alan Turing in 1936. Turing proved that any computer with some minimal functionality can simulate any other computer, and hence, except for physical limits like the amount of memory available, all computers are equally powerful. Not all problems, however, can be solved by computing. For some problems, there is no program that a machine based on conventional physics can execute that always finishes and produces the correct answer. In particular, Turing showed that it is impossible to create a program that will always correctly answer questions about interesting properties of the information processes described by arbitrary programs such as *will this program run ever finish?* and *is this program a virus?*.

The second question is known as *P versus NP*. It was posed precisely by Stephen Cook in 1971, and remains an open problem today. It asks what problems can be solved by computers in a reasonable amount of time, where reasonable means that the time required to solve the problem does not grow exponentially with the size of the input (that is, when the input size increases by one, the amount of work to solve the problem does not multiply). Many interesting problems such as finding the shape of a protein, determining the best schedule for using a resource, finding the minimum number of colors needed to color a map, and winning the pegboard puzzle game, are deeply equivalent – a reasonable-time solution to any one of these problems can be used to also solve all of the other problems. What is not known, however, is whether all of the problems can be solved quickly, or none of them can. The question boils doing to determining if an omnipotent machine that can always guess the correct move at every step is able to solve problems significantly faster than a machine without this magical power. Although it seems obvious that omnipotence should help, we are surprisingly far away from being able to answer this question satisfactorily.

The third question concerns whether or not computing machines can think. It is the fuzziest and most difficult to pose meaningfully. The first notable attempt was Alan Turing's simulation test which simplified the question into whether or not a computer could convince a human it was thinking. Many others have since attempted to pose or answer the question differently, but no satisfying definition of "thinking" has yet emerged. Many of the specific tests, like playing champion-level chess and understanding language, have been achieved by computers, but by throwing tremendous computing power and resources at the problem, not by anything most humans would consider real thought.

The selected books explore these questions and provide insights into how computing changes thinking, how computing has impacted history, how the state of the world today reflects the early stages of the computing revolution, and how the continuation of that revolution will change human existence in the coming decades. The first two books provide insights into the fundamental questions of computing and how it changes the way we think, and the way we think about thinking; the final three books illustrate how computing impacts the past, present, and future of the world.

Douglas R. Hofstadter, *Gödel, Escher, Bach: an Eternal Golden Braid* (Basic Books, 1979; Twentieth-anniversary Edition, Basic Books, 1999)

> This is the most challenging, and longest, of the listed books, but I place it first because of the elegance, wit, and insightfulness it brings to the fundamental questions of computing. Hofstadter masterfully connects the big ideas in computing to music, art, and philosophy, illustrating complex ideas with dialogues in the spirit of *Alice in Wonderland*, Escher's artwork, Bach's music, and Hofstadter's own clever and thought-provoking inventions. Much of the book focuses on the mind-bending issues and remarkable expressiveness that can be achieved with recursive definitions (for example, your *descendants* are your children and the descendants of your descendants) and self-reference (statements like

"This sentence is meaningless."). This book won the 1980 Pulitzer Prize for non-fiction, and was instrumental in many computer scientists' decision to enter the field (including my own).

Paul Graham, *Hackers & Painters: Big Ideas from the Computer Age* (O'Reilly Media, 2004)

This is a delightful collection of essays by Paul Graham, co-founder of the first web application company which became Yahoo! shopping, as well as an accomplished painter. The essays explore fundamental ideas in computing, but from a very different perspective than Hofstadter's. Like Hofstadter, Graham highlights connections between computing and art, but from the viewpoint of a creative pragmatist rather than an academic philosopher. The essays delve into topics ranging from why nerds are not popular in high school, to what make design good, to how to grow a successful company. Each essay is meticulously crafted, lucid and compelling, and the book is full of radically unconventional and provocative ideas.

Simon Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* (Anchor Books, 2000)

The desire to keep and steal secrets has been with humans since before language existed. Singh's book tells an engaging and inspiring tale of the history of cryptography spanning several thousand years. Since the 1940s, that history has been closely entwined with computing. Colossus, arguably the first programmable electronic computer, was built by the British to break Nazi command codes, and Alan Turing was instrumental in breaking the Nazi Enigma code. Increasing computing power, combined with number theory, enabled the development of public-key cryptography. Whereas all previous ciphers required the sender and receiver to establish and maintain a shared secret key used for both encryption and decryption, public-key cryptography allows different keys to be used for encryption and decryption, eliminating the need to keep one of the keys secret. Singh explains the mathematics and technology behind cryptography in a clear and direct way, while conveying the excitement, adventure, and obsessive commitment of human efforts to keep and break secrets.

Thomas L. Friedman, *The World Is Flat: A Brief History of the Twenty-First Century* (Farrar, Straus and Giroux, 2005)

New York Times columnist Thomas Friedman's widely heralded book is an entertaining and insightful guide to the flat world that has been created by the rapid advance of computing technology, and the global telecommunications infrastructure it enabled. Over just the past few years, the digitization of everything and essentially free, high bandwidth worldwide communication, has enabled work and ideas to flow freely across borders and oceans so that the Harvard Crimson's archives can be processed by Cambodian war refugees, McDonald's drive-thru orders in Missouri are taken in Colorado Springs, and

hundreds of millions of Chinese and Indians have entered the middle class. Friedman's writing is sharp and carries an urgency in conveying both the opportunity and dangers of the new flat world.

Peter J. Denning, editor, *The Invisible Future: the Seamless Integration of Technology into Everyday Life* (McGraw-Hill, 2002)

This book grew out of a visionary conference organized by computing's main professional society in 2001. It contains essays from leading scientists speculating on how future computing technology will impact science, society, and everyday life over the coming decades. My favorite essay is *Science's Endless Golden Age*, by astrophysicist Neil DeGrasse Tyson, director of the Hayden Planetarium. It is an optimistic romp on the inexorable advance of human knowledge. Other notable essays include *How Biology Became an Information Science*, by David Baltimore, cancer researcher, president of Caltech and winner of the Nobel prize in medicine, and Ray Kurzweil's speculations on future technologies that will mimic and extend human brains.

Personal Profile

David Evans (http://www.cs.virginia.edu/evans) is an Associate Professor of Computer Science in the University of Virginia's School of Engineering and Applied Science and Chair of the Computer Science BA Degree Committee. His research interests include program analysis, using diversity and properties of the physical world for security, and applications of cryptography. He has SB, SM and PhD degrees in Computer Science from MIT.