

Why Require the Study of Computer Science?

Matthew A. Brenner
mbrenner@csteachlearn.com

Abstract: This document presents an argument in favor of making the study of computer science a required element of compulsory education. The reasons for such a requirement are enumerated (Section 1), and then explained by defining and exploring the nature of algorithms and algorithmic thinking (Section 2), and the perfect fit between algorithms and computers. Afterwards, the need for algorithmic thinking is placed into its historical context to show that a combination of technological, economic, and social factors have recently combined to elevate this unique mode of thought to become a required part of a modern education (Sections 3 and 4). Finally, the details of a Computer Science requirement are specified, and examples provided of important societal issues that cannot be deeply explored without a basic knowledge of the capabilities and limitations of computing (Section 5).

1 - The Need for a Computer Science Requirement

Computer Science is fundamentally concerned with the invention, application, refinement and analysis of algorithms. There are three reasons for students to learn about algorithms, algorithmic thinking, and computer software technology:

- Algorithmic thinking, the fundamental mode of thinking that underlies all of computer science, is a unique mode of thought distinct from others encountered in the arts, social sciences, mathematics and other sciences. Algorithmic thinking skills are necessary to create computer software, and valuable to everyone who uses or interacts with computer software.
- The idea of computer, over its history, has evolved from its ancient role as a counting aid (e.g. the abacus developed 7,000 years ago), to accurate, arithmetic calculator in the 17th century, to high speed, programmable, general-purpose calculating machine in the middle of the 20th century. In the second half of the 20th century computers began, for the first time, to render judgments rather than calculate results. Now, at the dawn of the 21st century, the future of the computer lies in the direction of artificial judgment¹: making complex decisions, providing advice, and exercising judgment.
- The computer has become inexpensive and therefore ubiquitous. A typical American home contains many computers including desktop or portable ones, as well as those embedded in devices as varied as the coffee maker, microwave, television, radio, remote control, printer, scanner, fax machine, wrist watch, cell phone, kitchen timer, and camera. Reasons (1) and (2), though interesting, were not compelling reasons for all students to study algorithmic thinking, and the nature of artificial judgment, until this precise historical moment. The computer, that once cost millions of dollars, occupied a vast room, and was reserved for the

¹ I have coined the term artificial judgment to avoid using the term artificial intelligence, which incites debate over the nature of intelligence and which, if realized, would encompass all aspects of artificial judgment and more.

exclusive use of mathematicians, physicists and engineers, is now so small and inexpensive that it pervades—even invades—nearly every aspect of our lives. Today, in modern societies, people must be able to use computers inside and outside of their work.

For today's students to be productive and responsible citizens of tomorrow they must learn not just how to use computers, but also to understand the fundamental nature of computers, and the software that controls them, for three broad reasons:

- 1) Interacting with computerized systems makes more sense to, and produces less stress in, people who understand the nature of software.
- 2) In the course of learning to create computer software, students learn abstract ideas and how to express them with extraordinary precision. They invent things entirely in their imaginations, and then manufacture them with nothing more than a keyboard and determination. They learn ideas and techniques that enable them to use the breathtaking speed of inexpensive, modern computers to amplify their own abilities and creativity to solve practical problems that cannot be solved by humans acting without them. They also develop a sense of what kinds of problems do not yield to the brute force of computers performing billions of instructions each second.
- 3) Political debate over an ever-increasing number of public policy, privacy rights, national security, and military expenditures depend on an understanding of the nature of software and algorithms to make informed decisions.

The remainder of this document defines and explains algorithmic thinking, and then places computers, algorithms, and computer software into a historical context. The aim is to reveal the path of Computer Science to date, and identify the momentum that indicates its future direction.

2 - Just What is Algorithmic Thinking?

What is algorithmic thinking? It is thinking directed at creating, understanding, applying or analyzing algorithms. So, what is an algorithm? An algorithm:

- a) is a specification of a step-by-step process that produces a desired result
- b) must specify each step unambiguously
- c) must have a clear starting point
- d) must have a clear stopping point which the algorithm must ensure will eventually be reached

There are some algorithms that everyone learns. In elementary school you learned how to add a column of numbers algorithmically. You learned to start at the rightmost column of digits, adding each digit from top to bottom, recording the least significant digit at the bottom of the column and then carrying the remaining digits to the top of the next column. You learned to repeat that process, working your way from the rightmost to the leftmost column of digits before stopping. Similarly, you learned algorithms for performing subtraction, multiplication, and long division. None of those algorithms require you to exercise any judgment; taken together, requirements (b), (c) and (d) insist that an algorithm never rely upon the judgment of whoever (e.g. person), or whatever (e.g. computer), is being guided by the algorithm. While it is possible that you will make a mistake when you perform one of

these algorithms, that mistake will be entirely due to careless, human error, rather than any trace of ambiguity in the algorithm or poor judgment on your part.

Well, you might ask, "What's the difference if I make a mistake balancing my checkbook due to my own carelessness or my own poor judgment?" Frankly, the answer is, "None." Your checkbook is out of balance in either case, and it may not be any easier to eliminate your carelessness than to improve your judgment. As long as the entity executing the algorithm is error prone (human), then much of the value of algorithms cannot be realized. Now, suppose a special potion became available to eliminate all possibility of careless errors. After drinking that potion the only errors possible would be ones of judgment. Since algorithms are unambiguous specifications that never rely on judgment, any human who consumes the potion could execute any algorithm flawlessly. The single, remaining obstacle to performing tasks algorithmically would be time.

Algorithms have been well known to mathematicians for thousands of years (e.g. Euclid's Algorithm for finding the greatest common divisor of two whole numbers). Until the 20th century algorithms were performed by people. Even when performed correctly they could only be performed at human speed. With the development of high-speed digital computers, the world of algorithms changed. It became apparent that algorithms requiring hundreds, thousands, or even millions of steps—far beyond the capacity of humans to perform in a lifetime—could be quickly and flawlessly performed by computers. That is precisely why algorithms are perfectly suited to computers. Today, a typical desktop computer can perform hundreds of millions of steps per second, blindly, without any intention, knowledge or sense of satisfaction. This combination of speed and mindlessness works effectively because the algorithm a computer follows leaves no room for judgment or interpretation. The computer follows the algorithm perfectly. Every time a computer performs a particular algorithm (program), using the same information, it does exactly the same thing. No other possibility exists. The unambiguous instructions force the computer, step-by-step, to the only result that algorithm can yield.

3 - The Emerging Importance of Algorithmic Thinking

The invention of fully-electronic, digital computers in the mid-20th century led mathematicians, physicists, engineers and other scientists to invent and apply complex, computationally-intensive algorithms to their work. Few others had any need for, or opportunity to apply, algorithmic thinking skills, because only researchers had access to high-speed computers.

As the numbers of computers increased in both the scientific and commercial sectors, especially in the 1960's, efforts to create *artificial intelligence* began for the first time on a larger scale. Researchers began developing software to play games like chess, understand natural language (e.g. written English), and look at the world through electronic eyes (computerized vision). As the cost of computers declined through the ensuing decades these become commercial products, along with machines and software able to read aloud to the blind, respond to voice commands, provide driving directions, recommend spelling and grammar corrections, recognize human faces, write poetry (not so good), invent new electronic circuits (some patentable), and search staggering volumes of data for specific information (e.g. Internet search engines like Google).

Now, at the beginning of the 21st century, all but the most disadvantaged people living in modern societies have ready access to inexpensive computers of staggering power. In the same way that computers found their way into mathematics, and every branch of science and engineering in the

second half of the 20th century, so they are spreading rapidly into every facet of the lives of people living in modern societies. Teaching, learning, writing, buying, selling, advertising, distributing, corresponding, accounting, banking, researching, entertaining, editing (documents, images, video) are all being transformed by the computer. As the computer continues to fall in price, and increase in speed and capacity, more and more opportunities for automation arise, while at the same time new algorithms improve the quality of the work performed by computers. Our modern world, today, is awash in algorithms that affect us every day, and they will never go away.

4 – What About Education?

Should we allow compulsory education to ignore the nature of the most profound and far-reaching technological revolution since the steam engine of the Industrial Revolution? Well, you might say, "I don't know how a steam engine works, or a combustion engine, or an electric motor, and what of it?" The specific technologies of those machines were not the transformative concepts of the Industrial Revolution. The transformative concepts were the ideas of mechanization and automation. The specific technology of the computer is not transformative either. The fundamental concepts of The Computer Revolution or The Information Age are not computers or information, but rather the algorithm.

Nearly all of the technology of modern society depends on computer software (expressions of algorithms) that was, until now, beyond the reach of all but the technocrats. The state-of-the-art of electrical engineering, the current telecommunications infrastructure, and the economies of scale brought on by consumer awareness and then demand have dropped the algorithm into the laps of us all. If we don't know what they are or how to think about, invent, and apply them then we cannot use them to improve our lives and our society, nor can we understand how others use or wish to use them to the advantage or disadvantage of ourselves and our society. We can encourage our students to ignore algorithmic thinking by including it, marginally, in our curriculum as the topic of elective courses, or we can embrace it as a fundamental mode of thinking important enough in modern times to justify an exposure as a fundamental element in a 21st century education.

5 - What Form Should a Computer Science Requirement Take?

As part of every student's education she should learn algorithmic thinking by learning how to invent algorithms and express them as computer programs. This is a mode of thinking qualitatively different from those demanded in other disciplines. I state this with certainty though I have no empirical evidence to offer as proof. *(Anecdotally, I add that every time I teach such a course in a high school the class is rather evenly distributed across grade levels (9-12), and 12th graders don't do any better or worse than 9th graders. Some students certainly have a greater aptitude for the subject, but performance does not correlate well with age, level of mathematics, or gender—though many more boys than girls enroll. Naturally, students with prior programming experience tend to perform relatively well, no doubt because they come to class more experienced in algorithmic thinking.)*

Students might satisfy this requirement in several ways. They may successfully complete:

- 1) an already offered introductory computers science course (as offered in many high schools)
- 2) other existing courses that include a significant computer software component (e.g. a high

school robotics course as offered through some physics departments)

3) a new course, as for example, *Programming and Public Policy*

The portion of students already interesting in computers science or robotics will likely satisfy such a requirement as would they would have satisfied their own curiosity—by taking an already offered course. The majority of students, those who would not otherwise take the existing courses, should be offered an alternative. Not a dumbed-down alternative, but rather a course designed to help them learn about algorithms and also helps them make connections between their studies and their societies. Such a *Programming and Public Policy* course should devote approximately 75% of classroom and homework time to learning the concepts and practice of algorithmic thinking through programming, because it is not possible to understand the nature of computer software and the application of algorithms without writing programs. The balance of the course time will be used examining contemporary, sometimes contentious, public policy, social welfare, privacy, national defense, and military projects wherein computer software plays a central role. Here are some possible examples:

- Cookies are sometimes being used by companies to track the movements of Internet users as they click from web site to site. Sometimes, companies are able to determine the specific identity of the person moving from site to site. What are the privacy implications, and should companies be permitted to monitor people's activity without their knowledge or consent? Is it even technically possible to stop them? How can citizens sensibly consider these matters while knowing nothing of the way the Internet operates, the nature or structure of cookies, or how much and what kind of information users of the Internet present about themselves merely by using the Internet?
- From the Reagan administration to the present one, each has proposed spending hundreds of billions of dollars to protect the United States and possibly some of its allies from missile attacks by creating a space-based shield capable of intercepting and destroying ground launched missiles before they can reach and harm their destinations. This sounds like a great idea. But so does ice cream for everyone. A citizenry thoroughly ignorant of the limitations of computer software cannot know that the software necessary to control such a defense shield does not exist, and that there is no evidence that it can be developed.
- The Justice Department prevailed over the Microsoft Corporation in a federal antitrust case. Microsoft was found guilty of violating the Sherman Antitrust Act by abusing its monopoly position in microcomputer operating systems. The medium- and long-term consequences of this case will affect the future of everyone who uses desktop and portable computers. How can the citizenry make any sense of this case or the problems that brought it to the courts if they don't understand such basic software concepts as the distinction between an operating system and an application?
- The FBI has deployed a computer program named Carnivore that examines (eats-up) e-mail moving across the Internet. It searches those documents for trigger words (perhaps bomb or kill) and alerts authorities when the software thinks a danger may be present. How can citizens evaluate the extent to which Carnivore infringes on individuals' privacy rights if they are ignorant of all aspects of how information moves on the Internet, the structure of e-mail messages, where and how Carnivore connects to the Internet, and the extent to which programs like Carnivore are able to think or make judgments regarding risk or danger?

- The Medical Privacy Act was passed by the previous administration. Some elements of that act were crafted to facilitate the centralization and sharing of medical information about citizens, and some elements were intended to limit the dissemination of medical information about individuals without their consent. There was much discussion and debate about the act, and some of the deepest concerns about the act revolved around the likelihood, and even feasibility, of controlling access to individuals' private medical histories after information finds its way into medical databases. A citizenry with almost no knowledge of computer security issues was only marginally involved in the debate, yet all participants in the nation's health care system will be affected.

Some of these issues revolve around artificial judgment (Defense Shield and Carnivore), some raise concerns about privacy rights (Carnivore, cookies and the Medical Privacy Act), another addresses illegal monopoly behavior (Microsoft was found guilty) of an industry giant and appropriate remedies. All of them have far-reaching consequences. Some will require massive expenditures of public funds, and others will shape the future of industries. All of these issues are informed by a basic knowledge of the nature of software technology and algorithmic thinking. It is time for schools, all schools, to provide each of their graduates with knowledge and training in this modern subject that has become a basic ingredient in personal productivity, scientific advance, and public policy.

About the Author

Matt Brenner currently teaches computer science at Sidwell Friends School in Washington, D.C., to middle and high school students, while developing computer science curricula across the K – 12 range. Before teaching at Sidwell, he was a tenured instructor of computer science at Phillips Exeter Academy in Exeter, NH, where he developed the grade 9 – 12 computer science curriculum, and created and presented the argument contained herein that led the school to adopt a diploma requirement in computer science.

Before teaching in secondary schools, he taught software engineering, analysis and design, and modern computer languages to professional software developers. He consulted to companies, both large and small, domestically and abroad. Early in his career he worked at Bell Laboratories (when it was part of a monopoly), and the New York Blood Center (a non-profit). He operated a small software company creating turnkey hardware and software systems and sold them to several industries. He was a columnist for a computer magazine (Java Report). He did bilingual (English and Albanian) volunteer software development for the Mother Theresa Society (an NGO) in Kosovo after their war in 1999. He holds an Ed.M. in *Technology, Innovation, and Education* from the Harvard Graduate School of Education.