CHAPTER

0

Introduction

In this preliminary chapter we consider the scope of computer science, develop a historical perspective, and establish a foundation from which to launch our study.

Computer science is the discipline that seeks to build a scientific foundation for such topics as computer design, computer programming, information processing, algorithmic solutions of problems, and the algorithmic process itself. It provides the underpinnings for today's computer applications as well as the foundations for tomorrow's computing infrastructure.

This book provides a comprehensive introduction to this science. We will investigate a wide range of topics including most of those that constitute a typical university computer science curriculum. We want to appreciate the full scope and dynamics of the field. Thus, in addition to the topics themselves, we will be interested in their historical development, the current state of research, and prospects for the future. Our goal is to establish a functional understanding of computer science—one that will support those who wish to pursue more specialized studies in the science as well as one that will enable those in other fields to flourish in an increasingly technical society.

0.1 The Role of Algorithms

We begin with the most fundamental concept of computer science—that of an algorithm. Informally, an **algorithm** is a set of steps that defines how a task is performed. (We will be more precise later in Chapter 5.) For example, there are algorithms for cooking (called recipes), for finding your way through a strange city (more commonly called directions), for operating washing machines (usually displayed on the inside of the washer's lid or perhaps on the wall of a laundromat), for playing music (expressed in the form of sheet music), and for performing magic tricks (Figure 0.1).

Before a machine such as a computer can perform a task, an algorithm for performing that task must be discovered and represented in a form that is compatible with the machine. A representation of an algorithm is called a **program.** For the convenience of humans, computer programs are usually printed on paper or displayed on computer screens. For the convenience of machines, programs are encoded in a manner compatible with the technology of the machine. The process of developing a program, encoding it in machine-compatible form, and inserting it into a machine is called **programming.** Programs, and the algorithms they represent, are collectively referred to as **software,** in contrast to the machinery itself, which is known as **hardware.**

The study of algorithms began as a subject in mathematics. Indeed, the search for algorithms was a significant activity of mathematicians long before the development of today's computers. The goal was to find a single set of directions that described how all problems of a particular type could be solved. One of the best known examples of this early research is the long division algorithm for finding the quotient of two multiple-digit numbers. Another example is the Euclidean algorithm, discovered by the ancient Greek mathematician Euclid, for finding the greatest common divisor of two positive integers (Figure 0.2).

Once an algorithm for performing a task has been found, the performance of that task no longer requires an understanding of the principles on which the algorithm is based. Instead, the performance of the task is reduced to the process of merely following directions. (We can follow the long division algorithm to find a quotient or the Euclidean algorithm to find a greatest common divisor without understanding why the algorithm works.) In a sense, the intelligence required to solve the problem at hand is encoded in the algorithm.

Figure 0.1 An algorithm for a magic trick

Effect: The performer places some cards from a normal deck of playing cards face down on a table and mixes them thoroughly while spreading them out on the table. Then, as the audience requests either red or black cards, the performer turns over cards of the requested color.

Secret and Patter:

Step 1. From a normal deck of cards, select ten red cards and ten black cards. Deal these cards face up in two piles on the table according to color.

Step 2. Announce that you have selected some red cards and some black cards.

Step 3. Pick up the red cards. Under the pretense of aligning them into a small deck, hold them face down in your left hand and, with the thumb and first finger of your right hand, pull back on each end of the deck so that each card is given a slightly *backward* curve. Then place the deck of red cards face down on the table as you say, "Here are the red cards in this stack."

Step 4. Pick up the black cards. In a manner similar to that in step 3, give these cards a slight *forward* curve. Then return these cards to the table in a face-down deck as you say, "And here are the black cards in this stack."

Step 5. Immediately after returning the black cards to the table, use both hands to mix the red and black cards (still face down) as you spread them out on the tabletop. Explain that you are thoroughy mixing the cards.

Step 6. As long as there are face-down cards on the table, repeatedly execute the following steps:

6.1. Ask the audience to request either a red or a black card.

6.2. If the color requested is red and there is a face-down card with a concave appearance, turn over such a card while saying, "Here is a red card."

6.3. If the color requested is black and there is a face-down card with a convex appearance, turn over such a card while saying, "Here is a black card."

6.4. Otherwise, state that there are no more cards of the requested color and turn over the remaining cards to prove your claim.

Figure 0.2 The Euclidean algorithm for finding the greatest common divisor of two positive integers

Description: This algorithm assumes that its input consists of two positive integers and proceeds to compute the greatest common divisor of these two values.

Procedure:

Step 1. Assign M and N the value of the larger and smaller of the two input values, respectively.

Step 2. Divide M by N, and call the remainder R.

Step 3. If R is not 0, then assign M the value of N, assign N the value of R, and return to step 2; otherwise, the greatest common divisor is the value currently assigned to N.

It is through this ability to capture and convey intelligence (or at least intelligent behavior) by means of algorithms that we are able to build machines that perform useful tasks. Consequently, the level of intelligence displayed by machines is limited by the intelligence that can be conveyed through algorithms. We can construct a machine to perform a task only if an algorithm exists for performing that task. In turn, if no algorithm exists for solving a problem, then the solution of that problem lies beyond the capabilities of machines.

Identifying the limitations of algorithmic capabilities solidified as a subject in mathematics in the 1930s with the publication of Kurt Gödel's incompleteness theorem. This theorem essentially states that in any mathematical theory encompassing our traditional arithmetic system, there are statements whose truth or falseness cannot be established by algorithmic means. In short, any complete study of our arithmetic system lies beyond the capabilities of algorithmic activities.

This realization shook the foundations of mathematics, and the study of algorithmic capabilities that ensued was the beginning of the field known today as computer science. Indeed, it is the study of algorithms that forms the core of computer science.

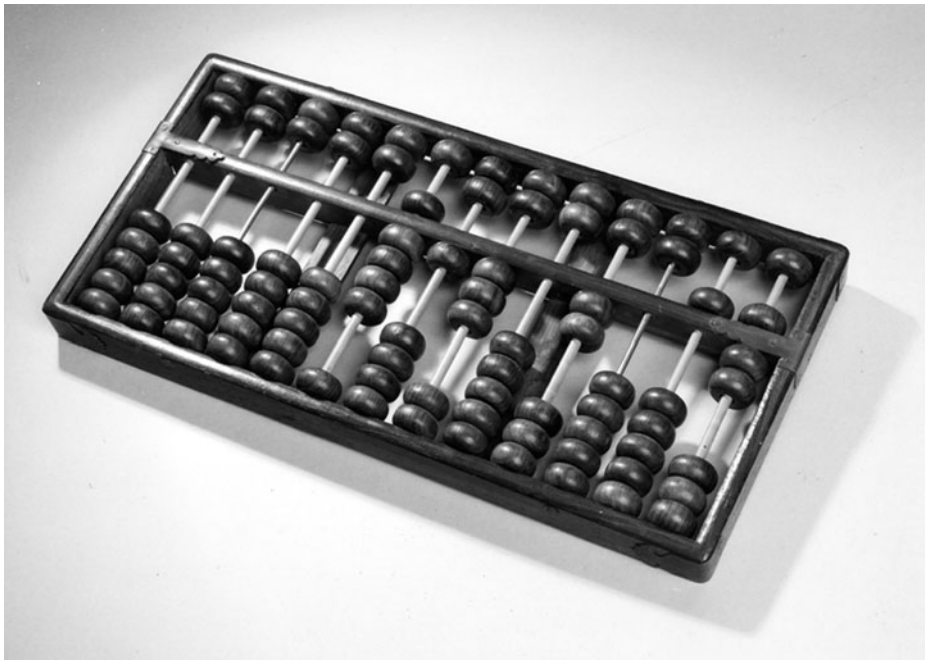
0.2 The History of Computing

Today's computers have an extensive genealogy. One of the earlier computing devices was the abacus. History tells us that it most likely had its roots in ancient China and was used in the early Greek and Roman civilizations. The machine is quite simple, consisting of beads strung on rods that are in turn mounted in a rectangular frame (Figure 0.3). As the beads are moved back and forth on the rods, their positions represent stored values. It is in the positions of the beads that this "computer" represents and stores data. For control of an algorithm's execution, the machine relies on the human operator. Thus the abacus alone is merely a data storage system; it must be combined with a human to create a complete computational machine.

In the time period after the Middle Ages and before the Modern Era the quest for more sophisticated computing machines was seeded. A few inventors began to experiment with the technology of gears. Among these were Blaise Pascal (1623–1662) of France, Gottfried Wilhelm Leibniz (1646–1716) of Germany, and Charles Babbage (1792–1871) of England. These machines represented data through gear positioning, with data being input mechanically by establishing initial gear positions. Output from Pascal's and Leibniz's machines was achieved by observing the final gear positions. Babbage, on the other hand, envisioned machines that would print results of computations on paper so that the possibility of transcription errors would be eliminated.

As for the ability to follow an algorithm, we can see a progression of flexibility in these machines. Pascal's machine was built to perform only addition. Consequently, the appropriate sequence of steps was embedded into the structure of the machine itself. In a similar manner, Leibniz's machine had its algorithms firmly embedded in its architecture, although it offered a variety of arithmetic operations from which the operator could select. Babbage's Difference Engine (of which only a demonstration model was constructed) could be modified to perform a variety of calculations, but his Analytical Engine (the construction for which he

Figure 0.3 An abacus (photography by Wayne Chandler)



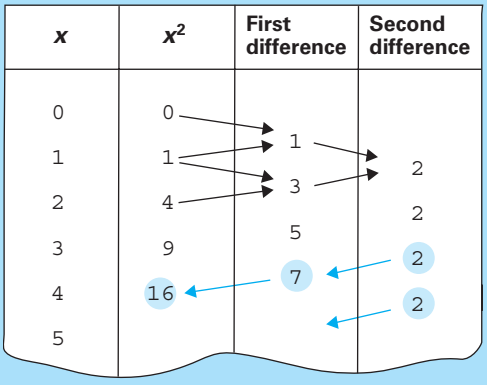
never received funding) was designed to read instructions in the form of holes in paper cards. Thus Babbage's Analytical Engine was programmable. In fact, Augusta Ada Byron (Ada Lovelace), who published a paper in which she demonstrated how Babbage's Analytical Engine could be programmed to perform various computations, is often identified today as the world's first programmer.

The idea of communicating an algorithm via holes in paper was not originated by Babbage. He got the idea from Joseph Jacquard (1752–1834), who, in 1801, had developed a weaving loom in which the steps to be performed during the weaving process were determined by patterns of holes in large thick cards made of wood (or cardboard). In this manner, the algorithm followed by the loom could be changed easily to produce different woven designs. Another beneficiary of Jacquard's idea was Herman Hollerith (1860–1929), who applied the concept of representing information as holes in paper cards to speed up the tabulation process in the 1890 U.S. census. (It was this work by Hollerith that led to the creation of IBM.) Such cards ultimately came to be known as punched cards and survived as a popular means of communicating with computers well into the 1970s. Indeed, the technique lives on today, as witnessed by the voting issues raised in the 2000 U.S. presidential election.

The technology of the time was unable to produce the complex gear-driven machines of Pascal, Leibniz, and Babbage in a financially feasible manner. But with the advances in electronics in the early 1900s, this barrier was overcome. Examples of this progress include the electromechanical machine of George Stibitz, completed in 1940 at Bell Laboratories, and the Mark I, completed in 1944

Babbage's Difference Engine

The machines designed by Charles Babbage were truly the forerunners of modern computer design. If technology had been able to produce his machines in an economically feasible manner and if the data processing demands of commerce and government had been on the scale of today's requirements, Babbage's ideas could have led to a computer revolution in the 1800s. As it was, only a demonstration model of his Difference Engine was constructed in his lifetime. This machine determined numerical values by computing "successive differences." We can gain an insight to this technique by considering the problem of computing the squares of the integers. We begin with the knowledge that the square of 0 is 0, the square of 1 is 1, the square of 2 is 4, and the square of 3 is 9. With this, we can determine the square of 4 in the following manner (see the following diagram). We first compute the differences of the squares we already know: $1^2 - 0^2 = 1$, $2^2 - 1^2 = 3$, and $3^2 - 2^2 = 5$. Then we compute the differences of these results: $3 - 1 = 2$, and $5 - 3 = 2$. Note that these differences are both 2. Assuming that this consistency continues (mathematics can show that it does) we conclude that the difference between the value $(4^2 - 3^2)$ and the value $(3^2 - 2^2)$ must also be 2. Hence $(4^2 - 3^2)$ must be 2 greater than $(3^2 - 2^2)$, so $4^2 - 3^2 = 7$ and thus $4^2 = 3^2 + 7 = 16$. Now that we know the square of 4, we could continue our procedure to compute the square of 5 based on the values of 1^2, 2^2, 3^2, and 4^2. (Although a more in-depth discussion of successive differences is beyond the scope of our current study, students of calculus may wish to observe that the preceding example is based on the fact that the derivative of $y = x^2$ is a straight line with a slope of 2.)



at Harvard University by Howard Aiken and a group of IBM engineers (Figure 0.4). These machines made heavy use of electronically controlled mechanical relays. In this sense they were obsolete almost as soon as they were built, because other researchers were applying the technology of vacuum tubes to construct totally electronic computers. The first of these machines was apparently the Atanasoff-Berry machine, constructed during the period from 1937 to 1941 at Iowa State College (now Iowa State University) by John Atanasoff and his assistant, Clifford Berry. Another was a machine called Colossus, built under the direction of Tommy

Figure 0.4 The Mark I computer (Courtesy of IBM archives. Unauthorized use is not permitted.)

Flowers in England to decode German messages during the latter part of World War II. (Actually, as many as ten of these machines were apparently built, but military secrecy and issues of national security kept their existence from becoming part of the "computer family tree.") Other, more flexible machines, such as the ENIAC (electronic numerical integrator and calculator) developed by John Mauchly and J. Presper Eckert at the Moore School of Electrical Engineering, University of Pennsylvania, soon followed.

From that point on, the history of computing machines has been closely linked to advancing technology, including the invention of transistors (for which physicists William Shockley, John Bardeen, and Walter Brattain were awarded a Nobel Prize) and the subsequent development of complete circuits constructed as single units, called integrated circuits (for which Jack Kilby also won a Nobel Prize in physics). With these developments, the room-sized machines of the 1940s were reduced over the decades to the size of single cabinets. At the same time, the processing power of computing machines began to double every two years (a trend that has continued to this day). As work on integrated circuitry progressed, many of the circuits within a computer became readily available on the open market as integrated circuits encased in toy-sized blocks of plastic called chips.

A major step toward popularizing computing was the development of desktop computers. The origins of these machines can be traced to the computer hobbyists who built homemade computers from combinations of chips. It was within this "underground" of hobby activity that Steve Jobs and Stephen Wozniak built a commercially viable home computer and, in 1976, established Apple Computer, Inc. (now Apple Inc.) to manufacture and market their products. Other companies that marketed similar products were Commodore, Heathkit, and Radio Shack. Although these products were popular among computer hobbyists, they

Augusta Ada Byron

Augusta Ada Byron, Countess of Lovelace, has been the subject of much commentary in the computing community. She lived a somewhat tragic life of less than 37 years (1815–1852) that was complicated by poor health and the fact that she was a nonconformist in a society that limited the professional role of women. Although she was interested in a wide range of science, she concentrated her studies in mathematics. Her interest in "compute science" began when she became fascinated by the machines of Charles Babbage at a demonstration of a prototype of his Difference Engine in 1833. Her contribution to computer science stems from her translation from French into English of a paper discussing Babbage's designs for the Analytical Engine. To this translation, Babbage encouraged her to attach an addendum describing applications of the engine and containing examples of how the engine could be programmed to perform various tasks. Babbage's enthusiasm for Ada Byron's work was apparently motivated by his hope that its publication would lead to financial backing for the construction of his Analytical Engine. (As the daughter of Lord Byron, Ada Byron held celebrity status with potentially significant financial connections.) This backing never materialized, but Ada Byron's addendum has survived and is considered to contain the first examples of computer programs. The degree to which Babbage influenced Ada Byron's work is debated by historians. Some argue that Babbage made major contributions whereas others contend that he was more of an obstacle than an aid. Nonetheless, Augusta Ada Byron is recognized today as the world's first programmer, a status that was certified by the U.S. Department of Defense when it named a prominent programming language (Ada) in her honor.

were not widely accepted by the business community, which continued to look to the well-established IBM for the majority of its computing needs.

In 1981, IBM introduced its first desktop computer, called the personal computer, or PC, whose underlying software was developed by a newly formed company known as Microsoft. The PC was an instant success and legitimized the desktop computer as an established commodity in the minds of the business community. Today, the term *PC* is widely used to refer to all those machines (from various manufacturers) whose design has evolved from IBM's initial desktop computer, most of which continue to be marketed with software from Microsoft. At times, however, the term *PC* is used interchangeably with the generic terms *desktop* or *laptop*.

As the twentieth century drew to a close, the ability to connect individual computers in a world-wide system called the **Internet** was revolutionizing communication. In this context, Tim Berners-Lee (a British scientist) proposed a system by which documents stored on computers throughout the Internet could be linked together producing a maze of linked information called the **World Wide Web** (often shortened to "Web"). To make the information on the Web accessible, software systems, called **search engines,** were developed to "sift through" the Web, "categorize" their findings, and then use the results to assist users researching particular topics. Major players in this field are Google, Yahoo, and Microsoft. These companies continue to expand their Web-related activities, often in directions that challenge our traditional way of thinking.

At the same time that desktop computers (and the newer mobile laptop computers) were being accepted and used in homes, the miniaturization of computing machines continued. Today, tiny computers are embedded within various devices. For example, automobiles now contain small computers running Global Positioning Systems (GPS), monitoring the function of the engine, and providing voice command services for controlling the car's audio and phone communication systems.

Perhaps the most potentially revolutionary application of computer miniaturization is found in the expanding capabilities of portable telephones. Indeed, what was recently merely a telephone has evolved into a small hand-held general-purpose computer known as a **smartphone** on which telephony is only one of many applications. These "phones" are equipped with a rich array of sensors and interfaces including cameras, microphones, compasses, touch screens, accelerometers (to detect the phone's orientation and motion), and a number of wireless technologies to communicate with other smartphones and computers. The potential is enormous. Indeed, many argue that the smartphone will have a greater effect on society than the PC.

The miniaturization of computers and their expanding capabilities have brought computer technology to the forefront of today's society. Computer technology is so prevalent now that familiarity with it is fundamental to being a member of modern society. Computing technology has altered the ability of governments to exert control; had enormous impact on global economics; led to startling advances in scientific research; revolutionized the role of data collection, storage, and applications; provided new means for people to communicate and interact; and has repeatedly challenged society's status quo. The result is a proliferation of subjects surrounding computer science, each of which is now a significant field of study in its own right. Moreover, as with mechanical engineering and physics, it is often difficult to draw a line between these fields and

Google

Founded in 1998, Google Inc. has become one of the world's most recoginzed techology companies. Its core service, the Google search engine, is used by millions of people to find documents on the World Wide Web. In addition, Google provides electronic mail service (called Gmail), an Internet based video sharing service (called YouTube), and a host of other Internet services (including Google Maps, Google Calendar, Google Earth, Google Books, and Google Translate).

However, in addition to being a prime example of the entrepreneurial spirit, Google also provides examples of how expanding technology is challenging society. For example, Google's search engine has led to questions regarding the extent to which an international company should comply with the wishes of individual governments; YouTube has raised questions regarding the extent to which a company should be liable for information that others distribute through its services as well as the degree to which the company can claim ownership of that information; Google Books has generated concerns regarding the scope and limitations of intelectual property rights; and Google Maps has been accused of violating privacy rights.

computer science itself. Thus, to gain a proper perspective, our study will not only cover topics central to the core of computer science but will also explore a variety of disciplines dealing with both applications and consequences of the science. Indeed, an introduction to computer science is an interdisciplinary undertaking.

0.3 The Science of Algorithms

Conditions such as limited data storage capabilities and intricate, time-consuming programming procedures restricted the complexity of the algorithms utilized in early computing machines. However, as these limitations began to disappear, machines were applied to increasingly larger and more complex tasks. As attempts to express the composition of these tasks in algorithmic form began to tax the abilities of the human mind, more and more research efforts were directed toward the study of algorithms and the programming process.

It was in this context that the theoretical work of mathematicians began to pay dividends. As a consequence of Gödel's incompleteness theorem, mathematicians had already been investigating those questions regarding algorithmic processes that advancing technology was now raising. With that, the stage was set for the emergence of a new discipline known as *computer science.*

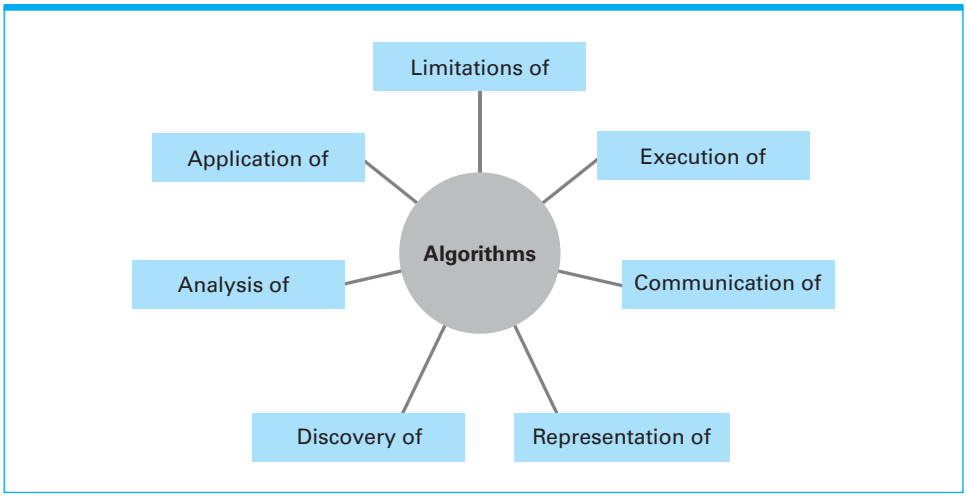
Today, computer science has established itself as the science of algorithms. The scope of this science is broad, drawing from such diverse subjects as mathematics, engineering, psychology, biology, business administration, and linguistics. Indeed, researchers in different branches of computer science may have very distinct definitions of the science. For example, a researcher in the field of computer architecture may focus on the task of miniaturizing circuitry and thus view computer science as the advancement and application of technology. But, a researcher in the field of database systems may see computer science as seeking ways to make information systems more useful. And, a researcher in the field of artificial intelligence may regard computer science as the study of intelligence and intelligent behavior.

Thus, an introduction to computer science must include a variety of topics, which is a task that we will pursue in the following chapters. In each case, our goal will be to introduce the central ideas in the subject, the current topics of research, and some of the techniques being applied to advance knowledge in the area. With such a variety of topics, it is easy to lose track of the overall picture. We therefore pause to collect our thoughts by identifying some questions that provide a focus for its study.

- Which problems can be solved by algorithmic processes?
- How can the discovery of algorithms be made easier?
- How can the techniques of representing and communicating algorithms be improved?
- How can the characteristics of different algorithms be analyzed and compared?
- How can algorithms be used to manipulate information?
- How can algorithms be applied to produce intelligent behavior?
- How does the application of algorithms affect society?

Note that the theme common to all these questions is the study of algorithms (Figure 0.5).

Figure 0.5 The central role of algorithms in computer science



0.4 Abstraction

The concept of abstraction so permeates the study of computer science and the design of computer systems that it behooves us to address it in this preliminary chapter. The term **abstraction,** as we are using it here, refers to the distinction between the external properties of an entity and the details of the entity's internal composition. It is abstraction that allows us to ignore the internal details of a complex device such as a computer, automobile, or microwave oven and use it as a single, comprehensible unit. Moreover, it is by means of abstraction that such complex systems are designed and manufactured in the first place. Computers, automobiles, and microwave ovens are constructed from components, each of which is constructed from smaller components. Each component represents a level of abstraction at which the use of the component is isolated from the details of the component's internal composition.

It is by applying abstraction, then, that we are able to construct, analyze, and manage large, complex computer systems, which would be overwhelming if viewed in their entirety at a detailed level. At each level of abstraction, we view the system in terms of components, called **abstract tools,** whose internal composition we ignore. This allows us to concentrate on how each component interacts with other components at the same level and how the collection as a whole forms a higher-level component. Thus we are able to comprehend the part of the system that is relevant to the task at hand rather than being lost in a sea of details.

We emphasize that abstraction is not limited to science and technology. It is an important simplification technique with which our society has created a lifestyle that would otherwise be impossible. Few of us understand how the various conveniences of daily life are actually implemented. We eat food and wear clothes that we cannot produce by ourselves. We use electrical devices and communication systems without understanding the underlying technology. We use the services of others without knowing the details of their professions. With each new advancement, a small part of society chooses to specialize in its implementation while the rest of us learn to use the results as abstract tools. In this manner, society's warehouse of abstract tools expands, and society's ability to progress increases.

Abstraction is a recurring theme in our study. We will learn that computing equipment is constructed in levels of abstract tools. We will also see that the development of large software systems is accomplished in a modular fashion in which each module is used as an abstract tool in larger modules. Moreover, abstraction plays an important role in the task of advancing computer science itself, allowing researchers to focus attention on particular areas within a complex field. In fact, the organization of this text reflects this characteristic of the science. Each chapter, which focuses on a particular area within the science, is often surprisingly independent of the others, yet together the chapters form a comprehensive overview of a vast field of study.

0.5 An Outline of Our Study

This text follows a bottom up approach to the study of computer science, beginning with such hands-on topics as computer hardware and leading to the more abstract topics such as algorithm complexity and computability. The result is that our study follows a pattern of building larger and larger abstract tools as our understanding of the subject expands.

We begin by considering topics dealing with the design and construction of machines for executing algorithms. In Chapter 1 (Data Storage) we look at how information is encoded and stored within modern computers, and in Chapter 2 (Data Manipulation) we investigate the basic internal operation of a simple computer. Although part of this study involves technology, the general theme is technology independent. That is, such topics as digital circuit design, data encoding and compression systems, and computer architecture are relevant over a wide range of technology and promise to remain relevant regardless of the direction of future technology.

In Chapter 3 (Operating Systems) we study the software that controls the overall operation of a computer. This software is called an operating system. It is a computer's operating system that controls the interface between the machine and its outside world, protecting the machine and the data stored within from unauthorized access, allowing a computer user to request the execution of various programs, and coordinating the internal activities required to fulfill the user's requests.

In Chapter 4 (Networking and the Internet) we study how computers are connected to each other to form computer networks and how networks are connected to form internets. This study leads to topics such as network protocols, the Internet's structure and internal operation, the World Wide Web, and numerous issues of security.

Chapter 5 (Algorithms) introduces the study of algorithms from a more formal perspective. We investigate how algorithms are discovered, identify several fundamental algorithmic structures, develop elementary techniques for representing algorithms, and introduce the subjects of algorithm efficiency and correctness.

In Chapter 6 (Programming Languages) we consider the subject of algorithm representation and the program development process. Here we find that the search for better programming techniques has led to a variety of programming methodologies or paradigms, each with its own set of programming languages. We investigate these paradigms and languages as well as consider issues of grammar and language translation.

Chapter 7 (Software Engineering) introduces the branch of computer science known as software engineering, which deals with the problems encountered when developing large software systems. The underlying theme is that the design of large software systems is a complex task that embraces problems beyond those of traditional engineering. Thus, the subject of software engineering has become an important field of research within computer science, drawing from such diverse fields as engineering, project management, personnel management, programming language design, and even architecture.

In next two chapters we look at ways data can be organized within a computer system. In Chapter 8 (Data Abstractions) we introduce techniques traditionally used for organizing data in a computer's main memory and then trace the evolution of data abstraction from the concept of primitives to today's object-oriented techniques. In Chapter 9 (Database Systems) we consider methods traditionally used for organizing data in a computer's mass storage and investigate how extremely large and complex database systems are implemented.

In Chapter 10 (Computer Graphics) we explore the subject of graphics and animation, a field that deals with creating and photographing virtual worlds. Based on advancements in the more traditional areas of computer science such as machine architecture, algorithm design, data structures, and software engineering, the discipline of graphics and animation has seen significant progress and has now blossomed into an exciting, dynamic subject. Moreover, the field exemplifies how various components of computer science combine with other disciplines such as physics, art, and photography to produce striking results.

In Chapter 11 (Artificial Intelligence) we learn that in order to develop more useful machines computer science has turned to the study of human intelligence for leadership. The hope is that by understanding how our own minds reason and perceive, researchers will be able to design algorithms that mimic these processes and thus transfer these capabilities to machines. The result is the area of computer science known as artificial intelligence, which leans heavily on research in such areas as psychology, biology, and linguistics.

We close our study with Chapter 12 (Theory of Computation) by investigating the theoretical foundations of computer science—a subject that allows us to understand the limitations of algorithms (and thus machines). Here we identify some problems that cannot be solved algorithmically (and therefore lie beyond the capabilities of machines) as well as learn that the solutions to many other problems require such enormous time or space that they are also unsolvable from a practical perspective. Thus, it is through this study that we are able to grasp the scope and limitations of algorithmic systems.

In each chapter our goal is to explore to a depth that leads to a true understanding of the subject. We want to develop a working knowledge of computer science—a knowledge that will allow you to understand the technical society in which you live and to provide a foundation from which you can learn on your own as science and technology advance.

0.6 Social Repercussions

Progress in computer science is blurring many distinctions on which our society has based decisions in the past and is challenging many of society's long-held principles. In law, it generates questions regarding the degree to which intellectual property can be owned and the rights and liabilities that accompany that

ownership. In ethics, it generates numerous options that challenge the traditional principles on which social behavior is based. In government, it generates debates regarding the extent to which computer technology and its applications should be regulated. In philosophy, it generates contention between the presence of intelligent behavior and the presence of intelligence itself. And, throughout society, it generates disputes concerning whether new applications represent new freedoms or new controls.

Although not a part of computer science itself, such topics are important for those contemplating careers in computing or computer-related fields. Revelations within science have sometimes found controversial applications, causing serious discontent for the researchers involved. Moreover, an otherwise successful career can quickly be derailed by an ethical misstep.

The ability to deal with the dilemmas posed by advancing computer technology is also important for those outside its immediate realm. Indeed, technology is infiltrating society so rapidly that few, if any, are independent of its effects.

This text provides the technical background needed to approach the dilemmas generated by computer science in a rational manner. However, technical knowledge of the science alone does not provide solutions to all the questions involved. With this in mind, this text includes several sections that are devoted to social, ethical, and legal issues. These include security concerns, issues of software ownership and liability, the social impact of database technology, and the consequences of advances in artificial intelligence.

Moreover, there is often no definitive correct answer to a problem, and many valid solutions are compromises between opposing (and perhaps equally valid) views. Finding solutions in these cases often requires the ability to listen, to recognize other points of view, to carry on a rational debate, and to alter one's own opinion as new insights are gained. Thus, each chapter of this text ends with a collection of questions under the heading "Social Issues" that investigate the relationship between computer science and society. These are not necessarily questions to be answered. Instead, they are questions to be considered. In many cases, an answer that may appear obvious at first will cease to satisfy you as you explore alternatives. In short, the purpose of these questions is not to lead you to a "correct" answer but rather to increase your awareness, including your awareness of the various stakeholders in an issue, your awareness of alternatives, and your awareness of both the short- and long-term consequences of those alternatives.

We close this section by introducing some of the approaches to ethics that have been proposed by philosophers in their search for fundamental theories that lead to principles for guiding decisions and behavior. Most of these theories can be classified under the headings of consequence-based ethics, duty-based ethics, contract-based ethics, and character-based ethics. You may wish to use these theories as a means of approaching the ethical issues presented in the text. In particular, you may find that different theories lead to contrasting conclusions and thus expose hidden alternatives.

Consequence-based ethics attempts to analyze issues based on the consequences of the various options. A leading example is utilitarianism that proposes that the "correct" decision or action is the one that leads to the greatest good for the largest portion of society. At first glance utilitarianism appears to be a fair way of resolving ethical dilemmas. But, in its unqualified form, utilitarianism

leads to numerous unacceptable conclusions. For example, it would allow the majority of a society to enslave a small minority. Moreover, many argue that consequence-based approaches to ethical theories, which inherently emphasize consequences, tend to view a human as merely a means to an end rather than as a worthwhile individual. This, they continue, constitutes a fundamental flaw in all consequence-based ethical theories.

In contrast to consequence-based ethics, duty-based ethics does not consider the consequences of decisions and actions but instead proposes that members of a society have certain intrinsic duties or obligations that in turn form the foundation on which ethical questions should be resolved. For example, if one accepts the obligation to respect the rights of others, then one must reject slavery regardless of its consequences. On the other hand, opponents of duty-based ethics argue that it fails to provide solutions to problems involving conflicting duties. Should you tell the truth even if doing so destroys a colleague's confidence? Should a nation defend itself in war even though the ensuing battles will lead to the death of many of its citizens?

Contract-based ethical theory begins by imagining society with no ethical foundation at all. In this "state of nature" setting, anything goes—a situation in which individuals must fend for themselves and constantly be on guard against aggression from others. Under these circumstances, contract-based ethical theory proposes that the members of the society would develop "contracts" among themselves. For example, I won't steal from you if you won't steal from me. In turn, these "contracts" would become the foundation for determining ethical behavior. Note that contract-based ethical theory provides a motivation for ethical behavior—we should obey the "contracts of ethics" because we would otherwise live an unpleasant life. However, opponents of contract-based ethical theory argue that it does not provide a broad enough basis for resolving ethical dilemmas since it provides guidance only in those cases in which contracts have been established. (I can behave anyway I want in situations not covered by an existing contract.) In particular, new technologies may present uncharted territory in which existing ethical contracts may not apply.

Character-based ethics (sometimes called virtue ethics), which was promoted by Plato and Aristotle, argues that "good behavior" is not the result of applying identifiable rules but instead is a natural consequence of "good character." Whereas consequence-based ethics, duty-based ethics, and contract-based ethics propose that a person resolve an ethical dilemma by asking, "What are the consequences?"; "What are my duties?"; or "What contracts do I have?" character-based ethics proposes that dilemmas be resolved by asking, "Who do I want to be?" Thus, good behavior is obtained by building good character, which is typically the result of sound upbringing and the development of virtuous habits.

It is character-based ethics that underlies the approach normally taken when "teaching" ethics to professionals in various fields. Rather than presenting specific ethical theories, the approach is to introduce case studies that expose a variety of ethical questions in the professionals' area of expertise. Then, by discussing the pros and cons in these cases, the professionals become more aware, insightful, and sensitive to the perils lurking in their professional lives and thus grow in character. This is the spirit in which the questions regarding social issues at the end of each chapter are presented.

Social Issues

The following questions are intended as a guide to the ethical/social/legal issues associated with the field of computing. The goal is not merely to answer these questions. You should also consider why you answered as you did and whether your justifications are consistent from one question to the next.

1. The premise that our society is *different* from what it would have been without the computer revolution is generally accepted. Is our society *better* than it would have been without the revolution? Is our society worse? Would your answer differ if your position within society were different?
2. Is it acceptable to participate in today's technical society without making an effort to understand the basics of that technology? For instance, do members of a democracy, whose votes often determine how technology will be supported and used, have an obligation to try to understand that technology? Does your answer depend on which technology is being considered? For example, is your answer the same when considering nuclear technology as when considering computer technology?
3. By using cash in financial transactions, individuals have traditionally had the option to manage their financial affairs without service charges. However, as more of our economy is becoming automated, financial institutions are implementing service charges for access to these automated systems. Is there a point at which these charges unfairly restrict an individual's access to the economy? For example, suppose an employer pays employees only by check, and all financial institutions were to place a service charge on check cashing and depositing. Would the employees be unfairly treated? What if an employer insists on paying only via direct deposit?
4. In the context of interactive television, to what extent should a company be allowed to retrieve information from children (perhaps via an interactive game format)? For example, should a company be allowed to obtain a child's report on his or her parents' buying patterns? What about information about the child?
5. To what extent should a government regulate computer technology and its applications? Consider, for example, the issues mentioned in Questions 3 and 4. What justifies governmental regulation?
6. To what extent will our decisions regarding technology in general, and computer technology in particular, affect future generations?
7. As technology advances, our educational system is constantly challenged to reconsider the level of abstraction at which topics are presented. Many questions take the form of whether a skill is still necessary or whether students should be allowed to rely on an abstract tool. Students of trigonometry are no longer taught how to find the values of trigonometric functions using tables. Instead, they use calculators as abstract tools to find these values. Some argue that long division should also give way to abstraction. What other subjects are involved with similar controversies? Do modern word processors eliminate the need to develop spelling skills? Will the use of video technology someday remove the need to read?

8. The concept of public libraries is largely based on the premise that all citizens in a democracy must have access to information. As more information is stored and disseminated via computer technology, does access to this technology become a right of every individual? If so, should public libraries be the channel by which this access is provided?
9. What ethical concerns arise in a society that relies on the use of abstract tools? Are there cases in which it is unethical to use a product or service without understanding how it works? Without knowing how it is produced? Or, without understanding the byproducts of its use?
10. As our society becomes more automated, it becomes easier for governments to monitor their citizens' activities. Is that good or bad?
11. Which technologies that were imagined by George Orwell (Eric Blair) in his novel *1984* have become reality? Are they being used in the manner in which Orwell predicted?
12. If you had a time machine, in which period of history would you like to live? Are there current technologies that you would like to take with you? Could your choice of technologies be taken with you without taking others? To what extent can one technology be separated from another? Is it consistent to protest against global warming yet accept modern medical treatment?
13. Suppose your job requires that you reside in another culture. Should you continue to practice the ethics of your native culture or adopt the ethics of your host culture? Does your answer depend on whether the issue involves dress code or human rights? Which ethical standards should prevail if you continue to reside in your native culture but conduct business with a foreign culture?
14. Has society become too dependent on computer applications for commerce, communications, or social interactions? For example, what would be the consequences of a long-term interruption in Internet and/or cellular telephone service?
15. Most smartphones are able to identify the phone's location by means of GPS. This allows applications to provide location-specific information (such as the local news, local weather, or the presence of businesses in the immediate area) based on the phone's current location. However, such GPS capabilities may also allow other applications to broadcast the phone's location to other parties. Is this good? How could knowledge of the phone's location (thus your location) be abused?
16. On the basis of your initial answers to the preceding questions, to which ethical theory presented in Section 0.6 do you tend to subscribe?

Additional Reading

Goldstine, J. J. *The Computer from Pascal to von Neumann*. Princeton: Princeton University Press, 1972.

Kizza, J. M. *Ethical and Social Issues in the Information Age*, 3rd ed. London: Springer-Verlag, 2007.

Mollenhoff, C. R. *Atanasoff: Forgotten Father of the Computer*. Ames: Iowa State University Press, 1988.

Neumann, P. G. *Computer Related Risks*. Boston, MA: Addison-Wesley, 1995.

Ni, L. *Smart Phone and Next Generation Mobile Computing*. San Francisco: Morgan Kaufmann, 2006.

Quinn, M. J. *Ethics for the Information Age*, 2nd ed. Boston, MA: Addison-Wesley, 2006.

Randell, B. *The Origins of Digital Computers*, 3rd ed. New York: Springer-Verlag, 1982.

Spinello, R. A. and H. T. Tavani. *Readings in CyberEthics*, 2nd ed. Sudbury, MA: Jones and Bartlett, 2004.

Swade, D. *The Difference Engine*. New York: Viking, 2000.

Tavani, H. T. *Ethics and Technology: Ethical Issues in an Age of Information and Communication Technology*, 3rd ed. New York: Wiley, 2011.

Woolley, B. *The Bride of Science, Romance, Reason, and Byron's Daughter*. New York: McGraw-Hill, 1999.