Chapter 1
An Introduction to Computer Science

INVITATION TO
Computer Science

6TH EDITION

Objectives

After studying this chapter, students will be able to:

- Understand the definition of computer science
- Write down everyday algorithms and evaluate them to determine if they are ambiguous or not effectively computable

Objectives (continued)

After studying this chapter, students will be able to:

- Understand the roots of modern computer science in mathematics thousands of years old and in mechanical machines hundreds of years old
- Summarize the key points in the historical development of modern electronic computers
- Map the organization of the text onto the definition of computer science

Introduction

- Misconceptions
  - Computer science is:
    - The study of computers
    - The study of how to write computer programs
    - The study of the uses and applications of computers and software

The Definition of Computer Science

- **Computer science** is the study of algorithms, including:
  - Their formal and mathematical properties
  - Their hardware realizations
  - Their linguistic realizations
  - Their applications

The Definition of Computer Science (continued)

- Algorithm
  - Informally, “an ordered sequence of instructions that is guaranteed to solve a specific problem.”
- Operations used to construct algorithms
  - Sequential operations
  - Conditional operations
  - Iterative operations

FIGURE 1.1

Step 1	If the clock and calendar are not correctly set, then go to page 9 of the instruction manual and follow the instructions there before proceeding to Step 2
Step 2	Place a blank disc into the DVR disc slot
Step 3	Repeat Steps 4 through 7 for each program that you want to record
Step 4	Enter the channel number that you want to record and press the button labeled CHAN
Step 5	Enter the time that you want recording to start and press the button labeled TIME-START
Step 6	Enter the time that you want recording to stop and press the button labeled TIME-FINISH. This completes the programming of one show
Step 7	If you do not want to record anything else, press the button labeled END-PROG
Step 8	Turn off your DVR. Your DVR is now in TIMER mode, ready to record

Programming your DVR: An example of an algorithm

FIGURE 1.2

Given: $m \geq 1$ and two positive numbers each containing m digits, $a_{m-1}\ a_{m-2} \ldots a_0$ and $b_{m-1}\ b_{m-2} \ldots b_0$

Wanted: $c_m\ c_{m-1}\ c_{m-2} \ldots c_0$, where $c_m c_{m-1}\ c_{m-2} \ldots c_0 = (a_{m-1}\ a_{m-2} \ldots a_0) + (b_{m-1}\ b_{m-2} \ldots b_0)$

Algorithm:

Step 1 Set the value of *carry* to 0

Step 2 Set the value of i to 0

Step 3 While the value of i is less than or equal to $m - 1$, repeat the instructions in Steps 4 through 6

Step 4 Add the two digits a_i and b_i to the current value of *carry* to get c_i

Step 5 If $c_i \geq 10$, then reset c_i to $(c_i - 10)$ and reset the value of *carry* to 1; otherwise, set the new value of *carry* to 0

Step 6 Add 1 to i, effectively moving one column to the left

Step 7 Set c_m to the value of *carry*

Step 8 Print out the final answer, $c_m\ c_{m-1}\ c_{m-2} \ldots c_0$

Step 9 Stop

Algorithm for adding two *m*-digit numbers

The Definition of Computer Science (continued)

- Why are formal algorithms so important in computer science?
  - If we can specify an algorithm to solve a problem, then we can automate its solution
- Computing agent
  - Machine, robot, person, or thing carrying out the steps of the algorithm
- Unsolved problems
  - Some problems are unsolvable, some solutions are too slow, and some solutions are not yet known

Algorithms

- The Formal Definition of an Algorithm
  - A well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time

Algorithms (continued)

- Well-ordered collection
  - Upon completion of an operation we always know which operation to do next
- Ambiguous statements
  - Go back and do it again (Do *what* again?)
  - Start over (From *where*?)

Algorithms (continued)

- Unambiguous operation, or **primitive**
  - Can be understood by the computing agent without having to be further defined or simplified
- It is not enough for an operation to be understandable
  - It must also be *doable* (**effectively computable**) by the computing agent

Algorithms (continued)

- Algorithm
  - Result must be produced after the execution of a finite number of operations
  - Result may be a number, text, a light, picture, sound, or a change in the computing agent's environment
- Infinite loop
  - Runs forever
  - Usually a mistake

FIGURE 1.3

Step	Operation
1	Wet your hair
2	Set the value of *WashCount* to 0
3	Repeat Steps 4 through 6 until the value of *WashCount* equals 2
4	Lather your hair
5	Rinse your hair
6	Add 1 to the value of *WashCount*
7	Stop, you have finished shampooing your hair

A correct solution to the shampooing problem

FIGURE 1.4

Step	Operation
1	Wet your hair
2	Lather your hair
3	Rinse your hair
4	Lather your hair
5	Rinse your hair
6	Stop, you have finished shampooing your hair

Another correct solution to the shampooing problem

Algorithms (continued)

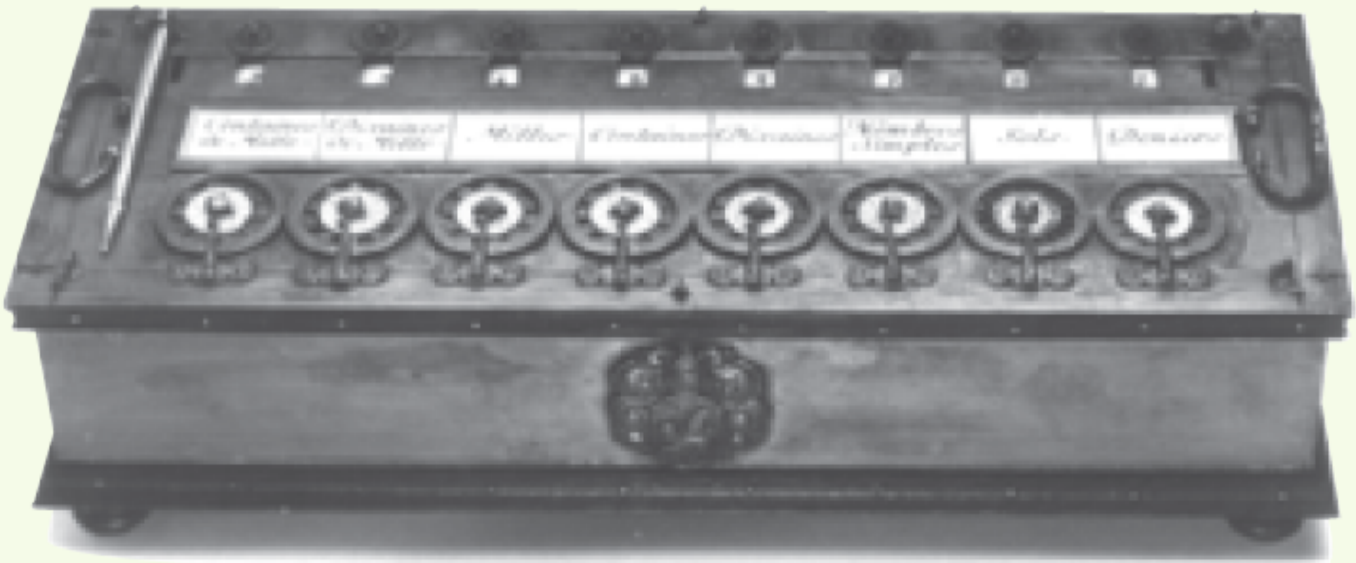
- The Importance of Algorithmic Problem Solving
  - "Industrial revolution" of 19^{th} century
    - Mechanized and automated repetitive physical tasks
  - "Computer revolution" of the 20^{th} and 21st centuries
    - Mechanized and automated repetitive mental tasks
    - Through algorithms and computer hardware

A Brief History of Computing

The Early Period: Up to 1940

- Seventeenth century: automation/simplification of arithmetic for scientific research
  - John Napier invented logarithms as a way to simplify difficult mathematical computations (1614)
  - The first slide rule appeared around 1622
  - Blaise Pascal designed and built a mechanical calculator named the Pascaline (1672)
  - Gottfried Leibnitz constructed a mechanical calculator called Leibnitz’s Wheel (1674)

FIGURE 1.5



The Pascaline, one of the earliest mechanical calculators

Source: Computer History Museum

A Brief History of Computing

The Early Period: Up to 1940 (continued)

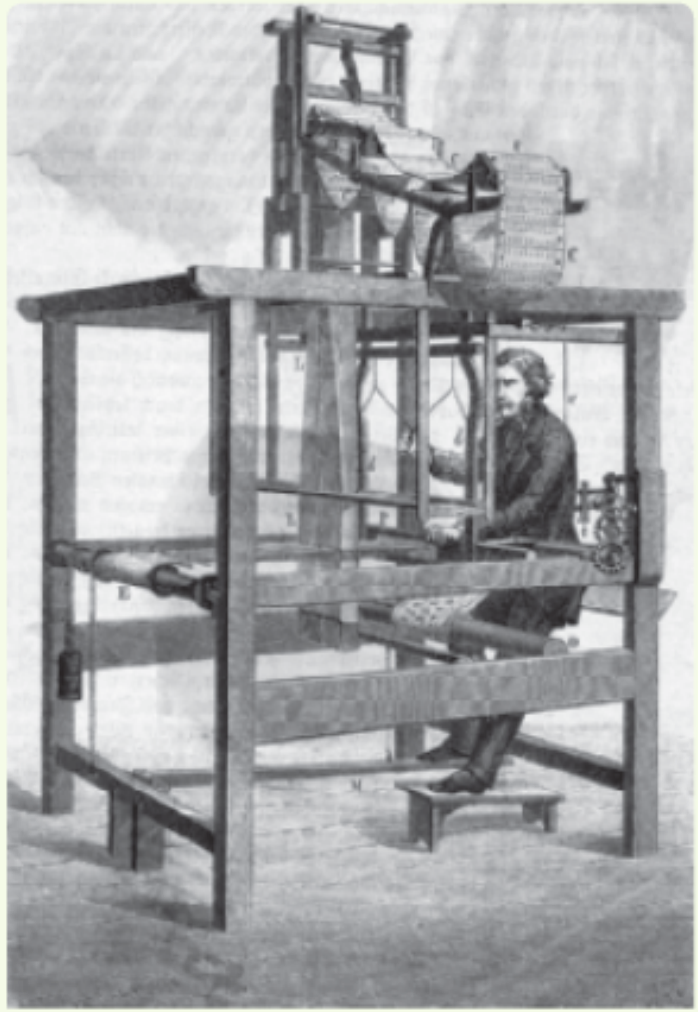
- Seventeenth century devices
  - Could represent numbers
  - Could perform arithmetic operations on numbers
  - Did not have a memory to store information
  - Were not **programmable** (a user could not provide a sequence of actions to be executed by the device)

A Brief History of Computing

The Early Period: Up to 1940 (continued)

- Nineteenth century devices
  - Joseph Jacquard designed an automated loom that used punched cards to create patterns (1801)
  - Herman Hollerith (1880s on)
    - Designed programmable card-processing machines to read, tally, and sort data on punched cards for the U.S. Census Bureau
    - Founded company that became IBM in 1924

FIGURE 1.6



Drawing of the Jacquard loom

A Brief History of Computing

The Early Period: Up to 1940 (continued)

- Charles Babbage
  - Difference Engine designed and built in 1823
    - Could do addition, subtraction, multiplication, and division to six significant digits
    - Could solve polynomial equations and other complex mathematical problems
  - Analytical Engine, designed but never built
    - Mechanical, programmable machine similar to a modern computer

A Brief History of Computing

The Early Period: Up to 1940 (continued)

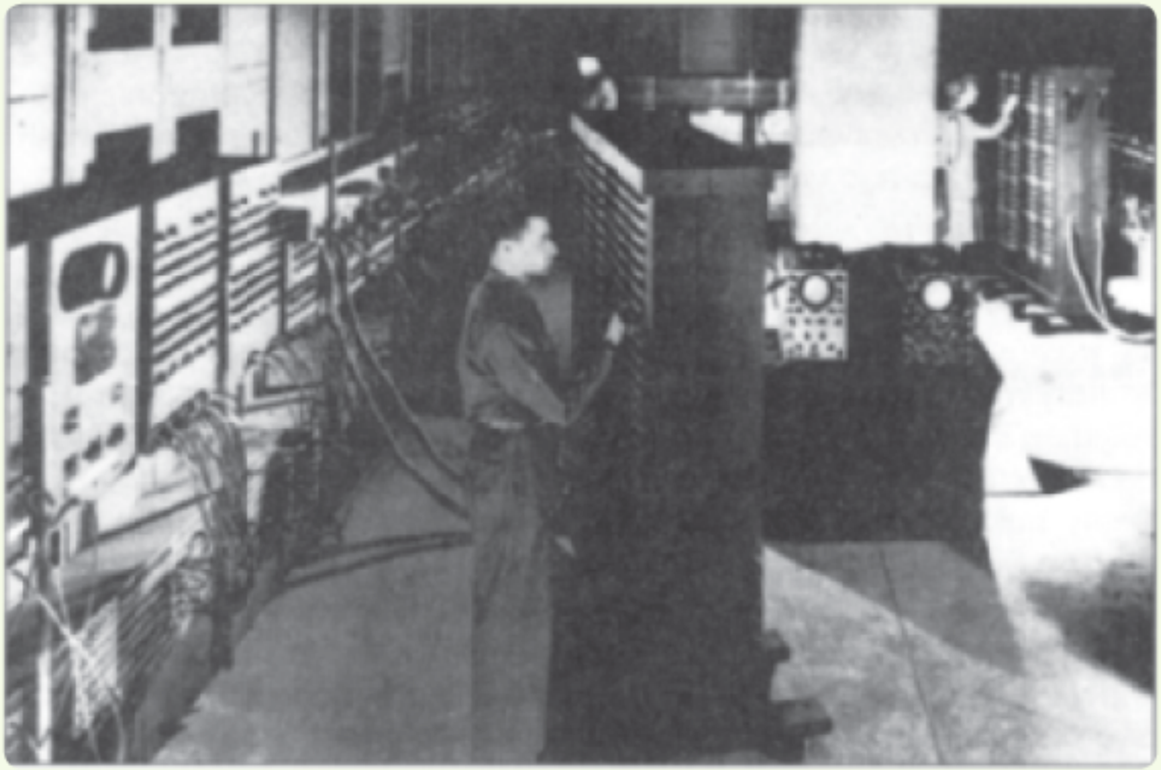
- Nineteenth century devices
  - Were mechanical, not electrical
  - Had many features of modern computers:
    - Representation of numbers or other data
    - Operations to manipulate the data
    - Memory to store values in a machine-readable form
    - Programmable: sequences of instructions could be pre-designed for complex operations

A Brief History of Computing

The Birth of Computers: 1940–1950

- Mark I (1944)
  – Electromechanical computer used a mix of relays, magnets, and gears to process and store data
- Colossus (1943)
  – General-purpose computer built by Alan Turing for British Enigma project
- ENIAC (Electronic Numerical Integrator and Calculator) (1946)
  – First publicly known fully electronic computer

FIGURE 1.7



Photograph of the ENIAC computer

A Brief History of Computing

The Birth of Computers: 1940–1950 (continued)

- John Von Neumann
  - Proposed a radically different computer design based on a model called the **stored program computer**
  - Research group at the University of Pennsylvania built one of the first stored program computers, called EDVAC, in 1951
  - UNIVAC-1, a version of EDVAC, first commercially-sold computer
  - Virtually all modern computers use the **Von Neumann architecture**

A Brief History of Computing

The Modern Era: 1950 to the Present

- First generation of computing (1950-1957)
  - Similar to EDVAC
  - Vacuum tubes for processing and storage
  - Large, expensive, and delicate
  - Required trained users and special environments
- Second generation (1957–1965)
  - Transistors and magnetic cores instead of vacuum tubes
  - Era of FORTRAN and COBOL: **high-level programming languages**

A Brief History of Computing
The Modern Era: 1950 to the Present (continued)

- Third generation (1965 to 1975)
  - Era of the integrated circuit
  - Birth of the first **minicomputer:** desk-sized, not room-sized computers
  - Birth of the software industry
- Fourth generation (1975 to 1985)
  - The first **microcomputers**: desktop machines
  - Development of widespread computer networks
  - Electronic mail, graphical user interfaces, and embedded systems

A Brief History of Computing

The Modern Era: 1950 to the Present (continued)

- Fifth generation (1985–?)
  - Massively parallel processors capable of quadrillions (10^{15}) of computations per second
  - Handheld digital devices
  - Powerful multimedia user interfaces incorporatng sound, voice recognition, images, video, television
  - Wireless communications
  - Massive storage devices
  - Ubiquitous computing

Organization of the Text

Computer science is the study of algorithms including:	**Levels of the text:**
1. Their formal and mathematical properties,	Level 1: The Algorithmic Foundations of Computer Science
2. Their hardware realizations,	Level 2: The Hardware World Level 3: The Virtual Machine
3. Their linguistic realizations,	Level 4: The Software World
4. Their applications.	Level 5: Applications Level 6: Social Issues

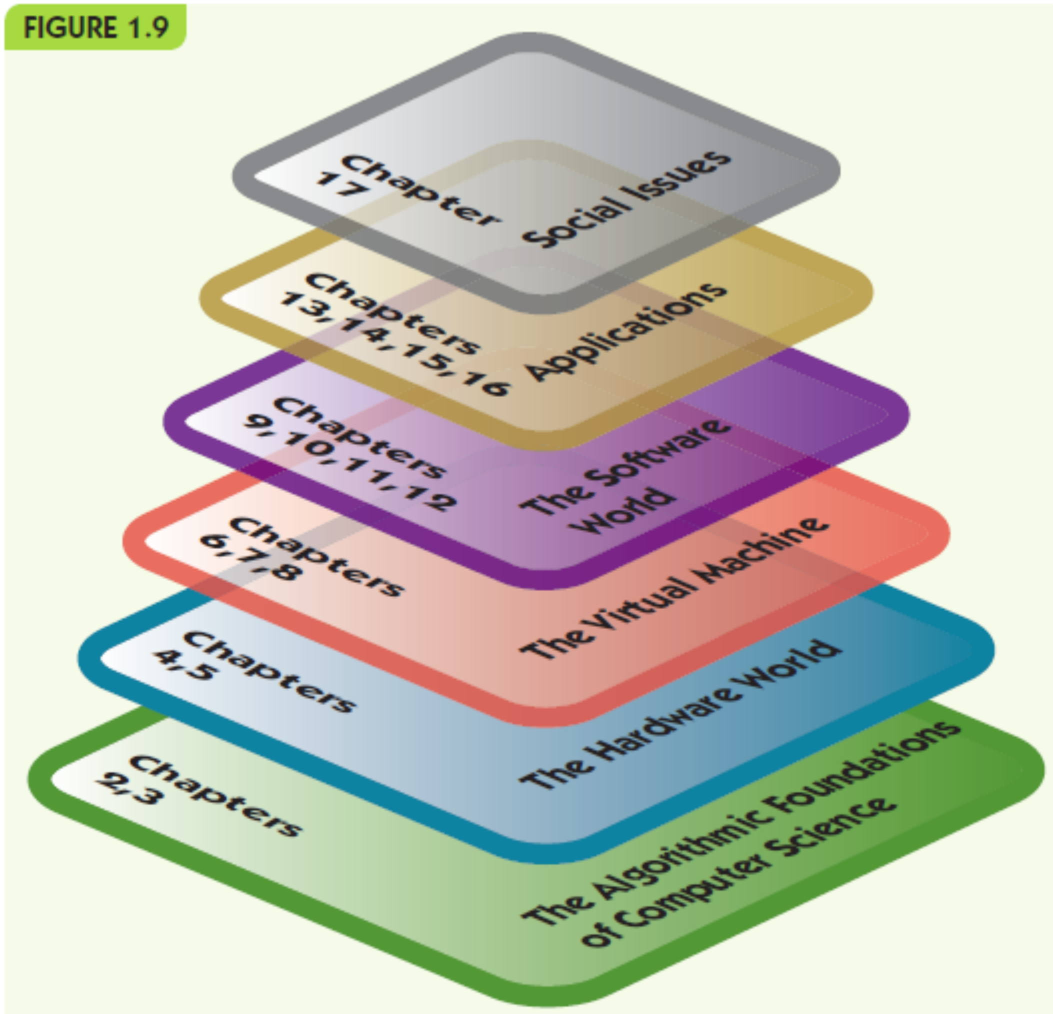


FIGURE 1.9
Chapter 17
Social Issues
Chapters 13,14,15,16
Applications
Chapters 9,10,11,12
The Software World
Chapters 6,7,8
The Virtual Machine
Chapters 4,5
The Hardware World
Chapters 2,3
The Algorithmic Foundations of Computer Science

Summary

- Computer science is the study of algorithms
- An algorithm is a well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time
- If we can specify an algorithm to solve a problem, then we can automate its solution
- Computers developed from mechanical calculating devices to modern electronic marvels of miniaturization

About the Presentation

- All chapter objectives are listed in the beginning of each presentation.
- You may customize the presentations to fit your class needs.
- A complete set of images from the book can be found on the Instructor Resources disc or the Instructor Companion site at *login.cengage.com*