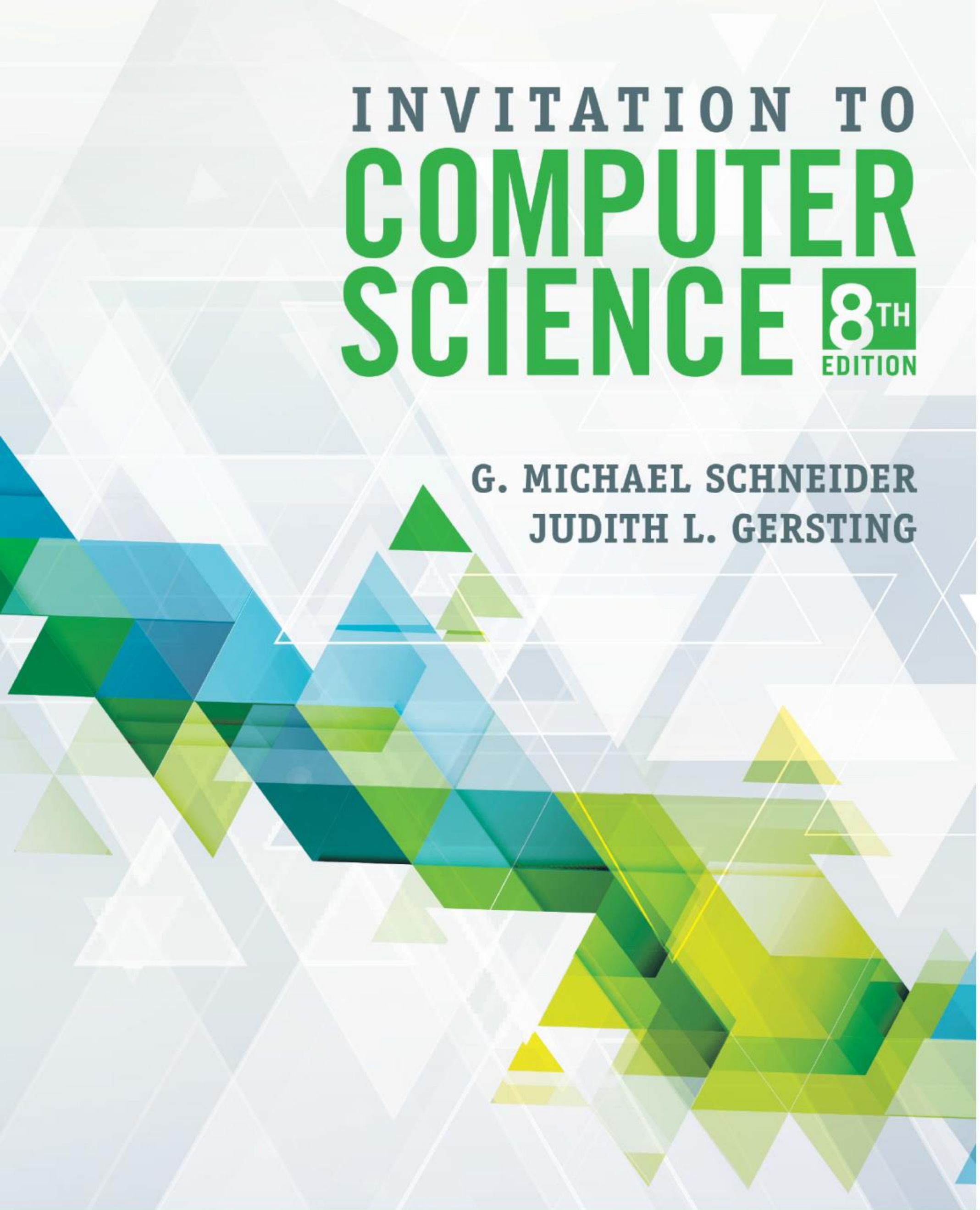


Chapter 1

An Introduction to Computer Science

CENGAGE

Learning Objectives

- Understand the definition of the term algorithm
- Understand the formal definition of computer science
- Write down everyday algorithms
- Determine if an algorithm is ambiguous or not effectively computable
- Understand the roots of modern computer science in mathematics and mechanical machines
- Summarize the key points in the historical development of modern electronic computers

Introduction

- Common misconceptions about computer science:
  - Computer science is the study of computers.
  - Computer science is the study of how to write computer programs.
  - Computer science is the study of the uses and applications of computers and software.

The Definition of Computer Science (1 of 4)

- 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 (2 of 4)

- The informal definition of an algorithm:
  - An ordered sequence of instructions that is guaranteed to solve a specific problem. For example:

    Step 1: Do something

    Step 2: Do something

    Step 3: Do something

    *   *

    *   *

    *   *

    Step N: Stop, you are finished

The Definition of Computer Science (3 of 4)

- Operations used to construct algorithms:
  - Sequential operations
    - Carries out a single well-defined task
  - Conditional operations
    - Ask a question and the next operation is then selected on the basis of the answer to that question
  - Iterative operations
    - Looping instructions that tell not to go on but go back and repeat the execution of a previous block of instructions

Figure 1.1 Programming your DVR: An example of an algorithm

Step 1	If the clock and the 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	Repeat Steps 3 through 6 for each program that you want to record
Step 3	Enter the channel number that you want to record and press the button labeled CHAN
Step 4	Enter the time that you want recording to start and press the button labeled TIME-START
Step 5	Enter the time that you want recording to stop and press the button labeled TIME-FINISH. This completes the programming of one show
Step 6	If you do not want to record anything else, press the button labeled END-PROG
Step 7	Turn off your DVR. Your DVR It is now in TIMER mode, ready to record

Figure 1.2 Algorithm for adding two m-digit numbers

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

The Definition of Computer Science (4 of 4)

- 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 (1 of 5)

- 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 (2 of 5)

- Well-ordered collection
  - Upon completion of an operation, we always know which operation to do next
- Unambiguous and effectively computable operations
  - It is not enough for an operation to be understandable, it must also be doable (effectively computable)
  - Ambiguous statements
    - Go back and do it again (Do *what* again?)
    - Start over (From *where*?)

Algorithms (3 of 5)

- Produces a result and halts in a finite amount of time
  - To know whether a solution is correct, an algorithm must produce a result that is observable to a user:
    - A numerical answer
    - A new object
    - A change in the environment

Algorithms (4 of 5)

- **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
- **Infinite loop**
  - Runs forever
  - Usually a mistake

Figure 1.3 A correct solution to the shampooing problem

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

Figure 1.4 Another correct solution to the shampooing problem

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

Algorithms (5 of 5)

- The Importance of Algorithmic Problem Solving
  - "Industrial revolution" of the nineteenth century
    - Mechanized and automated repetitive physical tasks
  - "Computer revolution" of the twentieth and twenty-first centuries
    - Mechanized and automated repetitive mental tasks
    - Used algorithms and computer hardware

A Brief History of Computing The Early Period: Up to 1940 (1 of 7)

- 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 (1642).
  - Gottfried Leibnitz constructed a mechanical calculator called Leibnitz’s Wheel (1673).

Figure 1.5 The pascaline, one of the earliest mechanical calculators

Source: INTERFOTO/Alamy

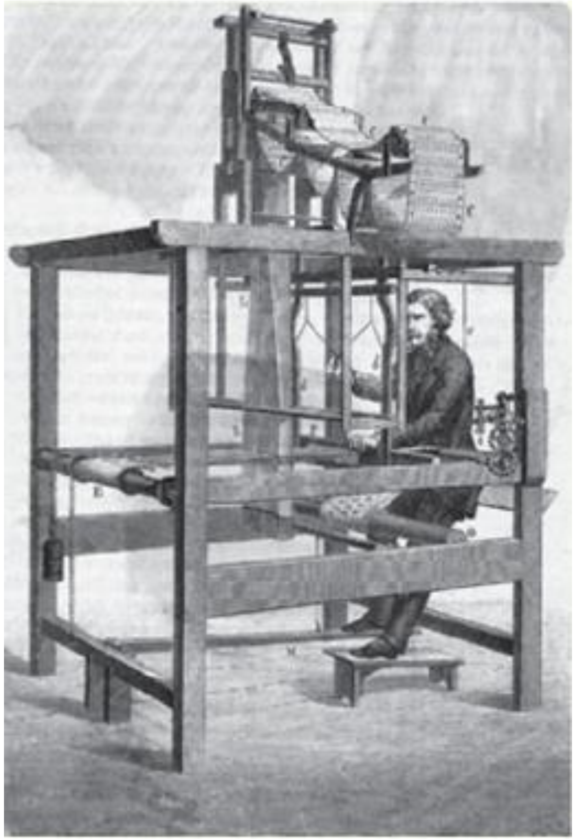
A Brief History of Computing The Early Period: Up to 1940 (2 of 7)

- 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 (3 of 7)

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

Figure 1.6 Drawing of the Jacquard loom



Source: © Bettmann/CORBIS

A Brief History of Computing The Early Period: Up to 1940 (4 of 7)

- Luddites
  - Originally opposed to the new manufacturing technology introduced by the Jacquard Loom
  - Now a term used to describe any group that is frightened or angered by the latest developments in any branch of science and technology, including computers

A Brief History of Computing The Early Period: Up to 1940 (5 of 7)

- 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

A Brief History of Computing The Early Period: Up to 1940 (6 of 7)

- Charles Babbage
  - Analytical Engine
    - Designed but never built
    - Mechanical, programmable machine with parts that mirror that of a modern-day computer:
      - Mill: Arithmetic/logic unit
      - Store: Memory
      - Operator: Processor
      - Output Unit: Input/Output

A Brief History of Computing The Early Period: Up to 1940 (7 of 7)

- 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 predesigned for complex operations

A Brief History of Computing The Birth of Computers: 1940–1950 (1 of 5)

- 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 the British Enigma project
- ENIAC (Electronic Numerical Integrator and Calculator) (1946)
  - First publicly known fully electronic computer

Figure 1.7 Photograph of the ENIAC computer

Source: From the Collections of the University of Pennsylvania Archives (U.S Army photo)

A Brief History of Computing The Birth of Computers: 1940–1950 (2 of 5)

- 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 1949
  - UNIVAC I, a version of EDVAC, the first commercially sold computer
  - Nearly all modern computers use the **Von Neumann architecture**

A Brief History of Computing The Modern Era: 1950 to the Present (3 of 5)

- 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: some of the first **high-level programming languages**

A Brief History of Computing The Modern Era: 1950 to the Present (4 of 5)

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

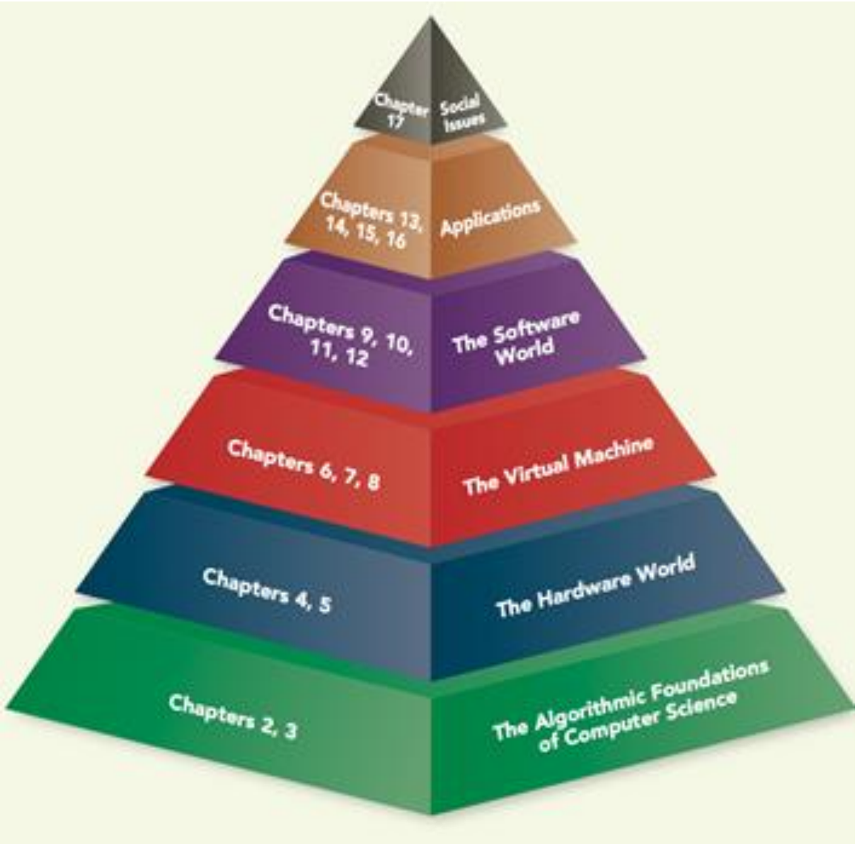
A Brief History of Computing The Modern Era: 1950 to the Present (5 of 5)

- Fifth generation (1985–?)
  - Massively parallel processors capable of quadrillions (10^{15}) of computations per second
  - Handheld digital devices
  - Powerful multimedia user interfaces incorporating sound, voice recognition, video, and television
  - Wireless communications
  - Massive cloud storage devices
  - Ubiquitous computing
  - Ultra-high-resolution graphics and virtual reality

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 Organization of the Text into a six-layer hierarchy



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.