

# **Chapter 1**

## **An Introduction to Computer Science**

### **At a Glance**

#### **Instructor's Manual Table of Contents**

- Overview
- Objectives
- Teaching Tips and Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

## Lecture Notes

### Overview

Chapter 1 introduces the definition of computer science and surveys the history of the field. It addresses common misconceptions about the field of computer science, and carefully examines the Gibbs-Tucker definition of computer science. This definition focuses on the idea of an *algorithm*; the chapter defines the concept of an algorithm, gives examples of algorithms, and explains the importance of algorithmic problem-solving. The chapter surveys the history of computing, beginning with calculating devices that pre-date modern computers by centuries, and including key developments from the 1940s on. The chapter ends by explaining how the organization of the textbook connects directly to the Gibbs-Tucker definition of computer science. This organizational structure is an essential feature of the text, and understanding it is a central goal of the course.

### 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
- 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

### Teaching Tips

#### 1.1 Introduction

1. Begin by asking students for their intuitive description of the field of computer science. Discuss the fact that many people do not have a clear understanding of what computer science professionals do and study.
2. Discuss the following misconceptions about the field of computer science (if students have proposed descriptions, categorize them using these misconceptions):

- a. Computer science is the study of computers
- b. Computer science is the study of how to write computer programs
- c. Computer science is the study of the uses and applications of computers and software

## 1.2 The Definition of Computer Science

1. Introduce the term **algorithm**, and discuss everyday examples of algorithms: recipes, driving directions, instruction manuals, etc.
2. Present and pick apart the Gibbs and Tucker definition of computer science. Note that the Gibbs and Tucker definition says that it is the task of the computer scientist to design and develop algorithms to solve a range of important problems. Give real-world examples for each part of the definition (Figures 1.1 and 1.2 contain two possible examples).
3. Prepare everyday examples of each of the three types of operations, and use them to illustrate that all operations used to construct algorithms belong to one of the three types:
  - a. Sequential operations
  - b. Conditional operations
  - c. Iterative operations
4. Introduce the term **computing agent**, making clear that that a computing agent may be hardware, software, or even human.
5. Describe the different kinds of unsolved problems: those that have no algorithmic solution, those that take too long to solve (see example of chess), and those that we do not yet know how to solve. Emphasize the importance of continuing work to solve new problems.

### *Teaching Tip*

Refer students to the following site to see current challenges for computing:  
<http://www.cra.org/ccf/initiatives>

## 1.3 Algorithms

1. Introduce the formal definition of the term **algorithm**. Discuss each piece of the definition, asking students for their ideas about what each piece means.
2. Introduce the terms **unambiguous operation**, **primitive**, and **effectively computable**. Discuss the idea that what counts as unambiguous and effectively computable operations depends on the abilities of the computing agent. Take a

simple example, and rewrite it for an adult expert, an average adult, a 12-year old, and a 5-year old.

3. Note the importance of every algorithm producing an observable result, whether it is a number, text, sound, or some other change to its environment.
4. Discuss why it is important that an algorithm stop and produce its result after a finite amount of time. What would happen otherwise? Introduce the term **infinite loop**.
5. Use Figures 1.3 and 1.4 or some similar set of algorithms to discuss correct, yet different, solutions to a simple problem. Poll students to find out which algorithm they prefer, and why.
6. Compare the automation of physical tasks in the industrial revolution to the automation of repetitive mental tasks in the computer revolution. Let students discuss the similarities and differences, and the social pros and cons of such revolutions.

### Quick Quiz 1

1. Which kind of operation is “*Add water until the cup is full*”?  
Answer: iterative
2. (True or false) All algorithms are known, computer scientists simply select the correct algorithm for each new problem.  
Answer: false
3. Operations that a given computing agent can perform are called \_\_\_\_\_.  
Answer: effectively computable
4. List at least two flaws in the “algorithm” below.  
 Given a jar full of jelly beans,  
 Pick a jelly bean from the jar  
 Add one to the total count  
 Repeat until the jar is empty  
 Answer: Possible answers include ambiguity about which lines to repeat, no initial value given to “total count,” and lack of clarity about what the result is.

## 1.4 A Brief History of Computing

This section contains many facts and details. The instructor must choose which facts to present: avoid extensive lectures that cram in every possible fact. Where possible, encourage students to engage with the material by presenting pieces of it to the class, preparing web sites on particular topics, or searching for additional information online.

1. Explain that mathematical algorithms have been created for thousands of years, particularly for applications in building, navigation, and commerce.
2. Discuss a variety of early devices (slide rules, Pascaline, Jacquard looms, Difference and Analytic Engines, etc.) and what features of a modern computing device each device includes. These features include the ability to represent numbers or other kinds of information, operations (such as arithmetic) to manipulate the information, a memory to store information, and programmability, the ability to change the process or algorithm of the device. Discuss how each device implements the features it includes.

**Teaching Tip**

Refer students to the following site to see more about early computing devices (site includes many images):  
<http://www.computersciencelab.com/ComputerHistory/History.htm>

3. Discuss the importance of Babbage's Analytic Engine design and its similarity to modern computer designs. Discuss with the class: why was the Analytic Engine never completed and what was the importance of Ada Augusta's work with Babbage?
4. Compare and contrast the Mark 1, ENIAC, Colossus, and other early computers. Describe the various early electronic computers and the kind of tasks they were designed to solve during World War II.

**Teaching Tip**

The video *Top Secret Rosies: The Female Computers of World War II* is an excellent resource about early computing and World War II.

5. Introduce the terms **Von Neumann architecture** and **stored program computer**, and discuss how Von Neumann's design differed from earlier designs, including the Analytic Engine. The primary innovation is to place the program itself in the computer's memory, removing the need to physically modify the computer in order to solve new problems. Virtually all modern computers are Von Neumann machines.
6. Survey the first through fifth generations of computing. Introduce the terms **high-level programming languages**, **minicomputer**, and **microcomputer**. Emphasize the memory and capabilities of each generation, and the areas of application. Figure 1.8 summarizes this information.

## Quick Quiz 2

1. Which of the following was the primary innovation of the Von Neumann architecture?
  - a. Use of transistors instead of vacuum tubes
  - b. Ability to perform floating-point (real number) calculations
  - c. Storage of program instructions in the internal memory unit
  - d. Purely electronic design, no mechanical parts for computation

Answer: c

2. The early general-purpose computer used by the British to crack the German Enigma code was called \_\_\_\_\_.

Answer: Colossus

3. The first computer built for sale was called \_\_\_\_\_.

Answer: The UNIVAC

4. (True or false) Microcomputers, the first desktop computers, were developed in the mid-1970s.

Answer: True

5. Herman Hollerith, working for the U.S. Census Bureau in the late 1800s, developed what early computing machine?

Answers: Many possible phrasings, but the answer should center on the “programmable card-processing machines that could automatically read, tally, and sort data entered on punch cards.” (See page 22 of *Invitation*)

## 1.5 Organization of the Text

1. The organization of the book is based on the Gibbs and Tucker definition of computer science. Connect each “level” of the book to the relevant portion of the definition and explain the connection. Figure 1.9 gives a diagrammatic view of the book’s organization. Otherwise you could connect them as follows:

<b>Gibbs and Tucker definition:</b>	<b>Levels of <i>Invitation to Computer Science</i></b>
Computer science is the study of algorithms, including:	
1. Their formal and mathematical properties,	Level 1: The Algorithmic Foundations of Computer Science (Ch. 2, 3)
2. Their hardware realizations,	Level 2: The Hardware World (Ch. 4, 5) Level 3: The Virtual Machine (Ch. 6-8)
3. Their linguistic realizations,	Level 4: The Software World (Ch. 9-12)
4. Their applications.	Level 5: Applications (Ch. 13-16) Level 6: Social Issues (Ch. 17)

2. Introduce the term **virtual machine/environment**. A virtual machine is composed only of the resources that the user perceives rather than all the resources that exist. The virtual machine is an abstract view of the machine.

## **Class Discussion Topics**

1. What are devices in your home that appear to use computers or algorithms? Can you name at least one device for every room in your house? Describe one algorithm each device performs.
2. Think of a problem or task in your life that you wish had an algorithmic solution. How difficult is this problem to solve? What might an algorithmic solution require?
3. Discuss the social changes caused by the computer revolution of the past hundred years. What are the downsides to automating routine mental tasks? What are the upsides? Are there times when we might want to decide *not* to use a technical innovation? When and why?

## **Additional Projects**

1. Working with one or two teammates, collect algorithms that you use in your everyday life, both those that are written down and those that you “just know.” Organize your algorithms into categories. What features are shared by all algorithms in a given category?
2. (This project is a kinesthetic in-class activity, and thus will be described from the instructor’s point of view.) When introducing the notion of an algorithm, and the importance of it being unambiguous and effectively computable, ask the students to work in teams to develop an algorithm for an everyday problem. Examples others have used include making a peanut butter sandwich, counting a jar of M&Ms or buttons, or making a paper airplane. You, the instructor, are the computing agent. Place all necessary materials on a table in front of you. Perform *exactly* the algorithms given to you, without inferring any steps. For instance, if the algorithm says to “Put peanut butter on the bread”, then place the jar of peanut butter on top of the bread. Have students refine their algorithms until they get one that works.
3. Search the web for articles about the future of computing. Collect a list of the new innovations in computing in the next few years. How will our daily lives change as a result?

## Additional Resources

1. Computational Tales: <http://computationaltales.blogspot.com/p/posts-by-topic.html>  
This link reflects the work of Dr. Jeremy Kubica, software engineer and manager at Google. Dr. Kubica has gathered a set of Computer Science concepts written as fairy tales in order to provide students with an accessible overview of key concepts before diving into technical details. These fun and quirky stories cover a broad range of topics, from algorithms and data structures to general programming concepts.
2. Careers in Computing: <http://computingcareers.acm.org/>
3. Algorithm: <http://en.wikipedia.org/wiki/Algorithm>
4. Charles Babbage: <http://www.computerhistory.org/babbage/>
5. Computing history timeline: <http://www.computerhistory.org/timeline/>
6. Early computing machines:  
[http://en.wikipedia.org/wiki/History\\_of\\_computing\\_hardware](http://en.wikipedia.org/wiki/History_of_computing_hardware)

## Key Terms

- **Algorithm:** Informally, an ordered sequence of instructions that is guaranteed to solve a problem; formally, a well ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time.
- **Analytic Engine:** A machine designed by Charles Babbage in the 1830s. Many consider it to be the first computer, although he never completed it.
- **Computer science:** The study of algorithms, including their mathematical properties, hardware and linguistic realizations, and their applications.
- **Computing agent:** The entity (machine, robot, person, or thing) that executes the steps of an algorithm.
- **Conditional operations:** Algorithmic operations that ask a question and select the next step based on the answer to that question.
- **Difference Engine:** A mechanical calculator that could do addition, subtraction, multiplication, and division to six significant digits and could solve polynomial equations and other complex mathematical problems as well.
- **Effectively computable:** There exists a method for actually carrying out the intent of the operation.
- **ENIAC:** The first fully electronic general-purpose programmable computer, completed in 1946; it contained 18,000 vacuum tubes and nearly filled a building.
- **High-level programming language:** A programming language that uses both natural language constructs and mathematical notation.
- **Infinite loop:** The repetitive execution of a block of operations that will never end. This is a fatal error when it occurs in an algorithm.



- **Iterative operations:** Algorithmic operations that repeat a block of instructions.
- **Luddites:** People who fear and are opposed to the use of new technologies.
- **Microcomputer:** Desktop computer that uses integrated circuit technology, developed in the mid-1970s, smaller than a minicomputer.
- **Minicomputer:** Smaller than mainframe computer, less expensive, developed in the mid-1960s.
- **Primitive:** When an operation is unambiguous for the agent carrying out the algorithm.
- **Sequential operation:** An algorithmic operation that carries out a single task and then moves on to the next operation in sequence.
- **Stored program computer:** A model of computation in which the instructions to be executed are represented as binary strings and stored in the memory of the computer.
- **Unambiguous operation:** An operation is unambiguous if it can be understood by the computing agent without having to be further defined or simplified.
- **Virtual machine (virtual environment):** The computer system as perceived by the user as opposed to the hardware that actually exists; the set of services and resources created by the software and seen by the user.
- **Von Neumann architecture:** The computational model designed by John Von Neumann and first implemented in the EDSAC computer of 1947; the structure and organization of virtually all modern computers.
- **Well-ordered collection:** Upon completion of an operation we always know which operation to do next.