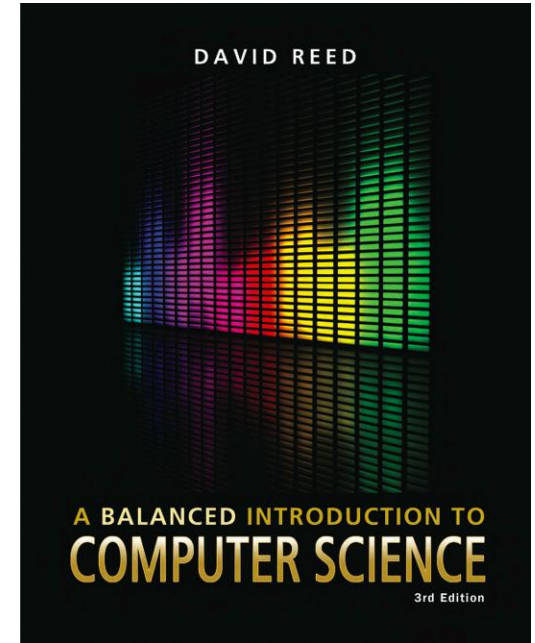# A Balanced Introduction to Computer Science, 3/E

David Reed, Creighton University

©2011 Pearson Prentice Hall
ISBN 978-0-13-216675-1

Chapter 10
Computer Science as a Discipline

Computer "Science"

some people argue that computer science is not a science in the same sense that biology and chemistry are

- the interdisciplinary nature of computer science has made it hard to classify

computer science is the study of *computation* (more than just machinery)

- it involves all aspects of problem solving, including
    - the design and analysis of algorithms
    - the formalization of algorithms as programs
    - the development of computational devices for executing programs
    - the theoretical study of the power and limitations of computing

whether this constitutes a "science" is a matter of interpretation

- certainly, computer science represents a rigorous approach to understanding complex phenomena and problem solving
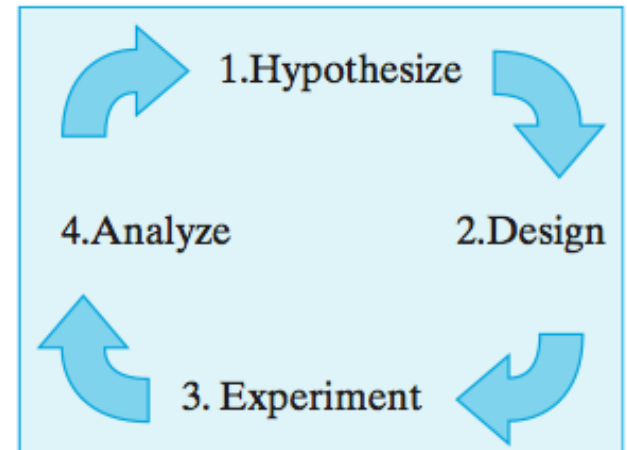
Scientific Method

the process developed by the scientific community for examining observations and events is known as the *scientific method*

The Scientific Method:

1. Formulate a hypothesis that explains an observed behavior.
2. Design an experiment to test your hypothesis.
3. Conduct the experiment.
4. Analyze the results of the experiment — if the results do not support the hypothesis, revise and repeat.

1. Hypothesize
2. Design
3. Experiment
4. Analyze

many activities carried out by computer scientists follow the scientific method

- e.g., designing and implementing a large database system requires hypothesizing about its behavior under various conditioning, experimenting to test those hypotheses, analyzing the results, and possibly redesigning
- e.g., debugging a complex program requires forming hypotheses about where an error might be occurring, experimenting to test those hypotheses, analyzing the results, and fixing the bugs

Artificial Science

the distinction between computer science and natural sciences like biology, chemistry, and physics is the type of systems being studied

- natural sciences study naturally occurring phenomena and attempt to extract underlying laws of nature
- computer science study human-made constructs: programs, computers, and computational modes

Herbert Simon coined the phrase "artificial science" to distinguish computer science from the natural sciences

in Europe, computer science is commonly called "Informatics"

- emphasizes the role of information processing as opposed to machinery

the term "Algorithmics" has also been proposed

- emphasizes the role of algorithms and problem solving

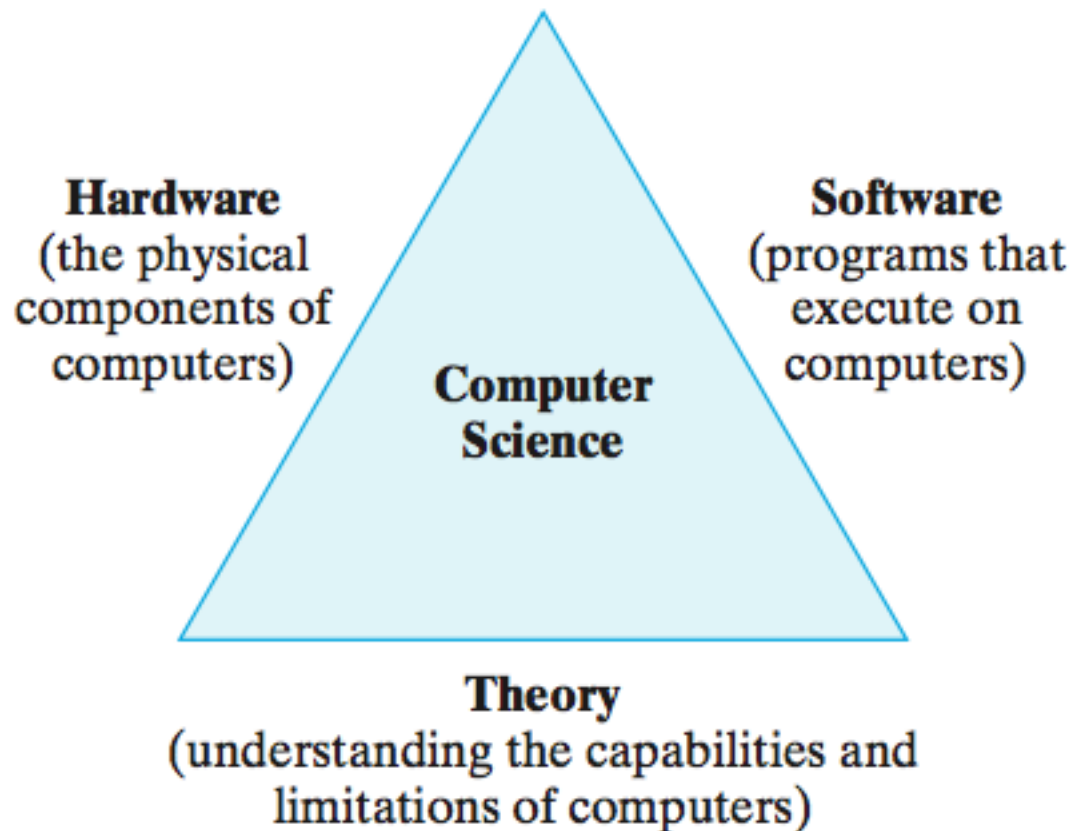other related fields study computation from different perspectives

- *computer engineering* focuses on the design and construction of computers
- *information systems management* focuses on business applications

Computer Science Themes

since computation encompasses many different types of activities, computer science research is often difficult to classify

- three recurring themes define the discipline

Hardware
(the physical components of computers)

Software
(programs that execute on computers)

Computer Science

Theory
(understanding the capabilities and limitations of computers)

Hardware

hardware refers to the physical components of a computer and its supporting devices

most modern computers implement the von Neumann architecture
- CPU + memory + input/output devices

ongoing research seeks to improve hardware design and organization
- *circuit designers* create smaller, faster, more energy-efficient chips
- *microchip manufacturers* seek to miniaturize and streamline production
- *systems architects* research methods to increase *throughput* (the amount of work done in a given time period)
  - e.g., parallel processing – splitting the computation across multiple CPUs
  - e.g., networking – connecting computers to share information and work

Software

software refers to the programs that execute on computers

3 basic software categories

1. *systems software:* programs that directly control the execution of hardware components (e.g., operating systems)
2. *development software:* programs that are used as tools in the development of other programs (e.g. Microsoft.NET, Java SDK)
3. *applications software:* all other programs, which perform a wide variety of tasks (e.g., web browsers, word processors, games)

many careers in computer science are related to the design, development, testing, and maintenance of software

- *language designers* develop and extend programming languages for easier and more efficient solutions
- *programmers* design and code algorithms for execution on a computer
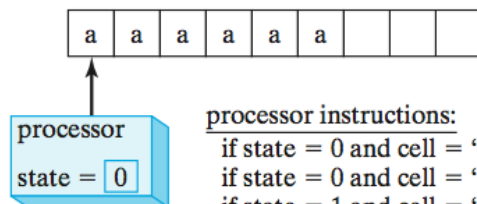- *systems analysts* analyze program designs and manage development

Theory

theoretical computer scientists strive to understand the capabilities of algorithms and computers (deeply rooted in math and formal logic)

example: the *Turing machine* is an abstract computational machine invented by computer pioneer Alan Turing
- consists of: a potentially infinite tape on which characters can be written
  a processing unit that can read and write on the tape, move in either direction, and distinguish between a finite number of states
- significance of the Turing machine
  - it is programmable (example below is programmed to distinguish between an even or odd number of a's on the tape)
  - provably as powerful as any modern computer, but simpler so provides a manageable tool for studying computation

1. Initially, the processor is positioned at the left end of the sequence in state 0.
2. Following the processor instructions, the processor moves right, alternating between state 0 (on odd-numbered cells) and state 1 (on even-numbered cells).
3. If the processor reaches the end of the sequence (marked by a space) in state 0, then there was an even number of a's — write "Y" in the cell and HALT.
4. If the processor reaches the end of the sequence (marked by a space) in state 1, then there was an odd number of a's — write "N" in the cell and HALT.

Turing used this simpler model to prove there are problems that cannot be solved by any computer!

| a | a | a | a | a | a | | | |

processor

state = 0

processor instructions:
if state = 0 and cell = "a", move right and set state = 1.
if state = 0 and cell = " ", write " Y" and HALT.
if state = 1 and cell = "a", move right and set state = 0.
if state = 1 and cell = " ", write " N" and HALT.

8

Subfields of Computer Science

Subfields of Computer Science[1]	
Algorithms and Data Structures	The study of methods for solving problems, designing and analyzing algorithms, and effectively using data structures in software systems.
Architecture	The design and implementation of computing technology, including the integration of effective hardware systems and the development of new manufacturing methods.
Operating Systems and Networks	The design and development of software and hardware systems for managing the components of a computer or network of communicating computers.
Software Engineering	The development and application of methodologies for designing, implementing, testing, and maintaining software systems.
Artificial Intelligence and Robotics	The study and development of software and hardware systems that solve complex problems through seemingly "intelligent" behavior.
Bioinformatics	The application of computing methodologies and information structures to biological research, such as the characterization and analysis of the human genome.
Programming Languages	The design and implementation of languages that allow programmers to express algorithms so that they are executable on computers.
Databases and Information Retrieval	The organization and efficient management of large collections of data, including the development of methods for searching and recognizing patterns in the data.
Graphics	The design of software and hardware systems for representing physical and conceptual objects visually, such as with images, video, or three-dimensional holograms.
Human–Computer Interaction	The design, implementation, and testing of interfaces that allow users to interact more effectively with computing technology.
Computational Science	Explorations in science and engineering that utilize high-performance computing, such as modeling complex systems or simulating experimental conditions.
Organizational Informatics	The development and study of management processes and information systems that support technology workers and organizations.

computer science can be divided into subfields

- each subfield takes a unique approach to computation
- however the common themes of computer science (hardware, software, and theory) influence every subfield

(Denning, Peter. "Computer Science: The Discipline." In Encyclopedia of Computer Science, 4th ed., 2000.)
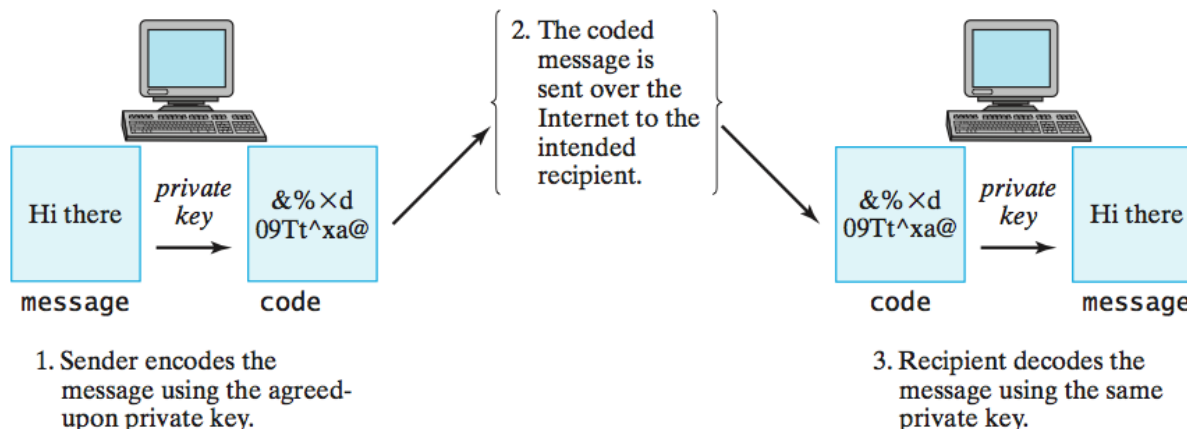
9

Algorithms and Data Structures

subfield that involves developing, analyzing, and implementing algorithms for solving problems

application: *encryption*

- encryption is the process of encoding a message so that it is decipherable only by its intended recipient
  - Caesar cipher: shift each letter three down in the alphabet
    - e.g., ET TU BRUTE → HW WX EUXWH
- Caesar cipher is an example of *private-key encryption*
  - relies on the sender and the recipient sharing a secret key
- some modern encryption algorithms rely on private keys
  - e.g., Advanced Encryption Standard (AES) utilizes 256-bit keys ($2^{256} \approx 10^{77}$ possibilities)

2. The coded message is sent over the Internet to the intended recipient.

Hi there | *private key* → | &%×d 09Tt^xa@

message | code

1. Sender encodes the message using the agreed-upon private key.

&%×d 09Tt^xa@ | *private key* → | Hi there

code | message

3. Recipient decodes the message using the same private key.
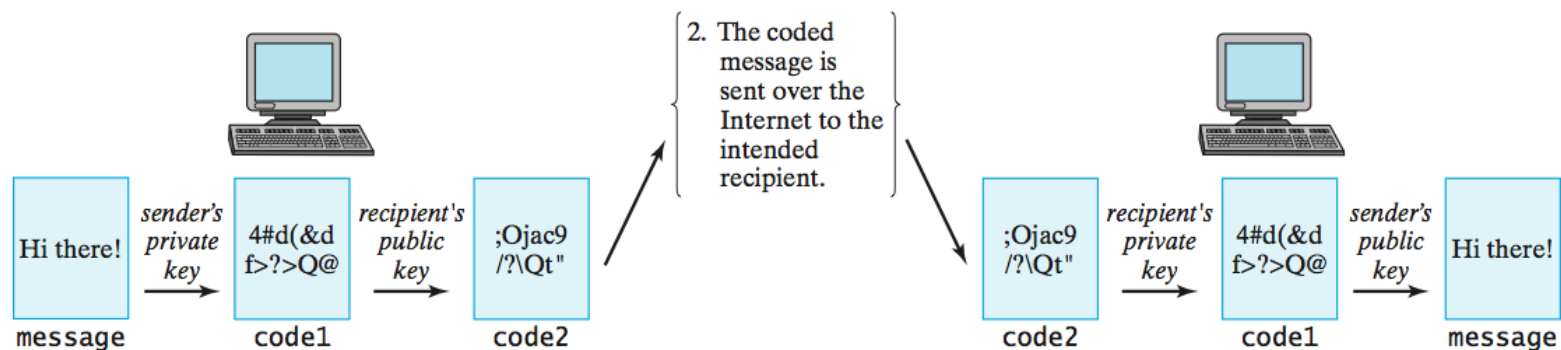
Public-Key Encryption

private-key encryption assumes that the sender and the recipient have agreed upon some key ahead of time (which introduces other security risks)

Whitfield Diffie and Martin Hellman proposed *public-key encryption*
- assign each party a pair of associated keys, one is public and the other is private
- a message encoded with a public key requires the corresponding private key for decoding, and vice versa
- almost all secure communications on the Internet use public key encryption

- allows for double encryption to also verify the identity of the sender
  - you can encode messages with your own private key and the recipient's public key, and decode the message in reverse

2. The coded message is sent over the Internet to the intended recipient.

| message | sender's private key → | code1 | recipient's public key → | code2 |

Hi there! — sender's private key → 4#d(&d f>?>Q@ — recipient's public key → ;Ojac9 /?\Qt"

;Ojac9 /?\Qt" — recipient's private key → 4#d(&d f>?>Q@ — sender's public key → Hi there!

| code2 | recipient's private key → | code1 | sender's public key → | message |

1. Sender encodes the message using his private key and the recipient's public key.

3. Recipient decodes the message using her private key and the sender's public key.

Architecture

subfield concerned with methods of organizing hardware components into efficient, reliable systems

application: *parallel processing*
- multiple processors can sometimes be utilized to share the computational load
- there are costs associated with coordinating the processors and dividing the work, so not well suited for all tasks
- understanding when parallel processing can be used effectively is a common task for computer architects

- e.g., Core 2 Duo & i3 processors integrate the circuitry for 2 processors
  - can execute two different instructions simultaneously, potentially double execution speed
- similarly, Core 2 Quad, i5 and i7 processors integrate the circuitry for 4 processors

- e.g., high-end Web Servers utilize multiple processors
  - can service multiple requests simultaneously by distributing the load among the processors

- Deep Blue, IBM's chess playing computer, contained 32 general purpose processors and 512 special-purpose chess processors
  - the processors worked in parallel to evaluate chess moves (could generate and evaluate 200 million chess moves per second)
  - in 1997, Deep Blue became the first computer to beat a world champion in a chess tournament

12

Operating Systems and Networks

subfield concerned with mechanisms to control the hardware and software components of computer systems

application: *operating systems* mediate between hardware and software
- *time-sharing* - allowed for multiple users to work on the same computer
  - each user is allocated a portion of the processor, and the processor rotates among tasks so rapidly that it appears to be executing tasks simultaneously
- multitasking – a single user can run multiple programs simultaneously
  - each application is allocated a portion of the memory

application: *networks* allow computers to communicate and share resources
- wide area network (WAN) – for long distances (e.g., Internet)
- local area network (LAN) – for short distances (e.g., same room or building)

- Ethernet is a family of technologies for building LANs
  - can broadcast data at 100Mbits/sec up to 1 Gbits/sec
- wireless (Wi-Fi) networks utilize a *router/access point* to transmit via radio signals
  - range of 50-100 yards, but slower than Ethernet (54 Mbits/sec)
- 3G networks utilize cellular towers to transmit data
  - widespread, but even slower (5.8 Mbits/sec upload, 14 Mbits/sec download)

Software Engineering

subfield concerned with creating effective software systems
- large projects can encompass millions of lines of code
- teams of programmers work together to make an integrated whole
  - coordination and testing are key to successful projects

Stages in the Software Life Cycle

1. *Requirement analysis and specification:* Initially, the needs of the customer must be assessed and the intended behavior of the software specified in detail.
2. *Design:* Next, the software system must be designed, including its breakdown into manageable pieces, and the underlying algorithms and data structures to be used.
3. *Implementation:* After completion of the design documents, the code must be written. For large software projects, teams may work independently on separate modules and then integrate those modules to complete the system.
4. *Testing:* The software must be systematically tested, both as independent modules and as a whole. As testing reveals errors or unforeseen conditions, reimplementation and even redesign may need to take place.
5. *Operation and maintenance:* Once the software system is complete, it must be installed and supported. It is estimated that maintenance, which involves fixing errors and updating code to meet changing customer needs, accounts for as much as half of a software project's total development budget.

- software demand continues to grow, placing pressure on programmers to produce at faster rates
  - clearly, there is a limit to personal productivity
  - simply adding more programmers does not solve the problem: increasing numbers means increased complexity, and coordination becomes an even bigger challenge
- in recent years, the adoption of the object-oriented programming methodology has made it easier to reuse code

Artificial Intelligence

subfield that attempts to make computers exhibit human-like characteristics (e.g., the ability to reason and think)

- in 1950, Turing predicted intelligent computers by 2000 (still not even close)
- but, progress has been made in many A.I. realms
    - robots in manufacturing
    - expert systems – programs that encapsulate expert knowledge in a specific domain (e.g., for medical diagnosis, credit card fraud detection)
    - neural computing – design of architectures that mimic the brain

The Turing Test

In his 1950 paper, *Computing Machinery and Intelligence,* Alan Turing proposed what is still considered the ultimate test for artificial intelligence. The Turing Test is based on the observation that we, as humans, make assumptions about intelligence and self-awareness in other humans by monitoring and interacting with them. Turing proposed that if the behavior of a machine were indistinguishable from a human's behavior, then we ought to give the machine the same credit for being intelligent that we give people.

The Turing Test involves a human judge and two contestants, one being the computer to be tested and the second being a human control subject. The job of the judge is to converse with the two contestants via computer terminals, without knowing which contestant is which. If, after a sufficiently long period of conversation, the judge is unable to identify the computer, then the computer is said to have passed the test and must be considered to possess human-like intelligence.

Bioinformatics

subfield that bridges the gap between biology and computer science

- focuses on using computers and computer science techniques to solve biological problems

- computers are integrated with various scientific tools
  - e.g., microscopes connected to computers and digital cameras

- computer are used to model biological systems
  - e.g., pharmaceutical companies model drug interactions to save time and money

- computers are used to store and process large amounts of biological data
  - e.g., Human Genome Project stores and provides tools for studying DNA
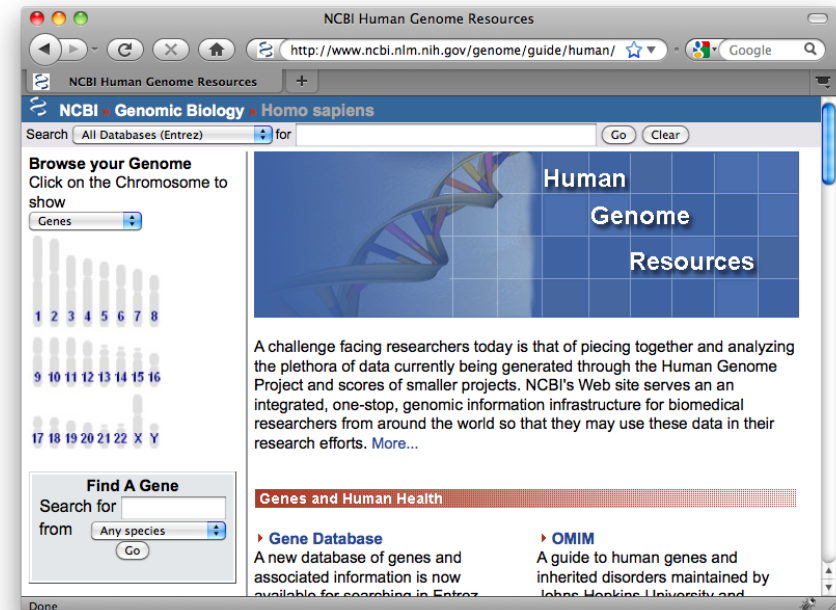
The Ethics of Computing

as technology becomes more prevalent in society, computing professionals must ensure that hardware and software are used safely, fairly, and effectively

ACM Code of Ethics and Professional Conduct[7]
Copyright © 1997, Association for Computing Machinery, Inc.

General Moral Imperatives. *As an ACM member I will . . .*

1.1 Contribute to society and human well-being.
1.2 Avoid harm to others.
1.3 Be honest and trustworthy.
1.4 Be fair and take action not to discriminate.
1.5 Honor property rights including copyrights and patent.
1.6 Give proper credit for intellectual property.
1.7 Respect the privacy of others.
1.8 Honor confidentiality.

More Specific Professional Responsibilities. *As an ACM computing professional I will . . .*

2.1 Strive to achieve the highest quality, effectiveness and dignity in both the process and products of professional work.
2.2 Acquire and maintain professional competence.
2.3 Know and respect existing laws pertaining to professional work.
2.4 Accept and provide appropriate professional review.
2.5 Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.
2.6 Honor contracts, agreements, and assigned responsibilities.
2.7 Improve public understanding of computing and its consequences.
2.8 Access computing and communication resources only when authorized to do so.

17