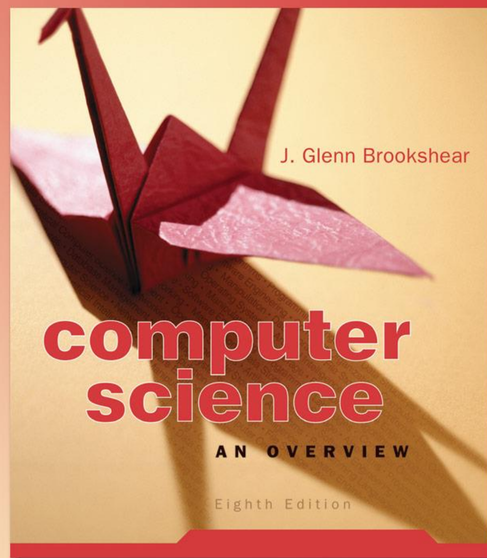
Chapter 0

Introduction



By:
J. Brookshear
Modified by:
Yacoub Sabatin

In this chapter

- Computer science as the discipline that seeks a scientific foundation for:
  - computer design,
  - computer programming,
  - algorithmic processes, etc.
- Informal introduction to the concept of an algorithm.
- Brief history of computing machinery.
- The concept of abstraction.
- Future discussions social/professional/ethical considerations.

Goals of this chapter

- Establish the concept of computer science.
- Underpinning for the development of the computer applications.
- Distinction between studying computer science and learning how to use today's computer application software.
- Explaining that computer science deals with the development of tomorrow's application software, rather than learning how to use the applications of today.

Chapter 0: Introduction

- 0.1 The Role of Algorithms
- 0.2 The Origins of Computing Machines
- 0.3 The Science of Algorithms
- 0.4 Abstraction
- 0.5 An Outline of Our Study
- 0.6 Social Repercussions

Algorithms: Definitions

- **Algorithm** = a set of steps that defines how a task is performed.
- **Program** = a representation of an algorithm.
- **Programming** = the process of developing a program.
- **Software** = programs + algorithms.
- **Hardware** = machinery: whatever isn't software.

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."

Figure 0.1 An algorithm for a magic trick (cont'd)

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.

History of Algorithms

- The study of algorithms was originally a subject in mathematics.
  - Algorithms were studied before computers existed.
- Early examples of algorithms
  - Long division algorithm
  - Euclidean Algorithm
- **Gödel's Incompleteness Theorem**: some problems cannot be solved by algorithms

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.

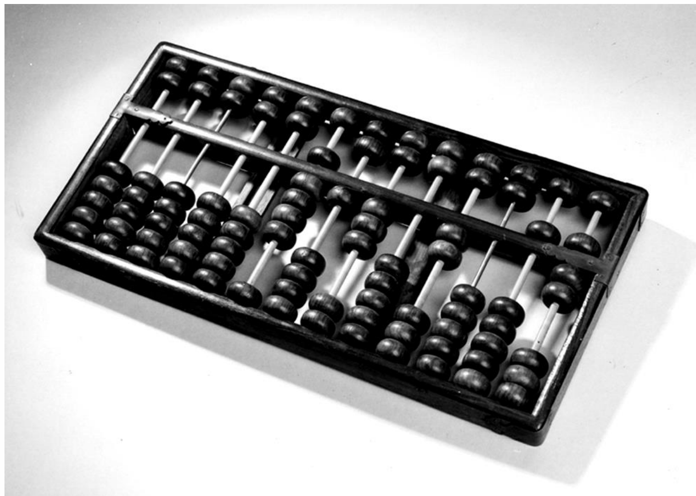
Step 2. Divide M by N, and call the remainder R.

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.

Origins of Computing Machines

- Early computing devices
  - Abacus: positions of beads represent numbers
  - Gear-based machines (1600s-1800s)
    - Positions of gears represent numbers
    - Blaise Pascal, Wilhelm Leibniz, Charles Babbage

Figure 0.3 An Abacus



Origins of Computing Machines

- Early data storage: punched cards
  - First used in Jacquard Loom (1801) to store patterns for weaving cloth
  - Stored programs in Babbage's Analytical Engine
  - Popular through the 1970's

Early computers

- Based on mechanical relays
  - 1940: Stibitz at Bell Laboratories
  - 1944: Mark I: Howard Aiken and IBM at Harvard
- Based on vacuum tubes
  - 1937-1941: Atanasoff-Berry at Iowa State
  - 1940s: Colossus: secret German code-breaker
  - 1940s: ENIAC: Mauchly & Eckert at U. of Penn.

Figure 0.4 The Mark I computer

Personal computers

- First used by hobbyists
- 1981: IBM introduces the PC
  - Accepted by business
  - Became the standard hardware design for most desktop computers
  - Most PCs use software from Microsoft

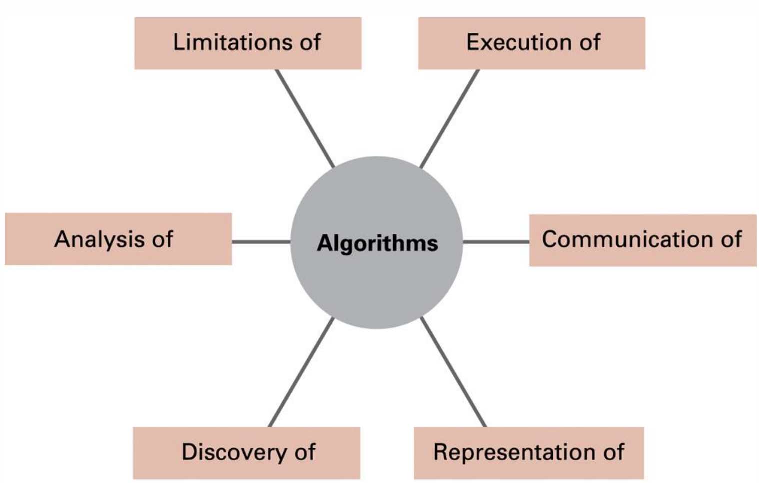
Computer Science

- The science of algorithms
- Draws from other subjects, including
  - Mathematics
  - Engineering
  - Psychology
  - Business Administration
  - Psychology

Central Questions of Computer Science

- Which problems can be solved by algorithmic processes?
- How can discovery of algorithms be made easier?
- How can techniques of representing and communicating algorithms be improved?
- How can our knowledge of algorithms and technology be applied to provide better machines?
- How can characteristics of different algorithms be analyzed and compared?

Figure 0.5 The central role of algorithms in computer science



Abstraction: Definitions

- **Abstraction** = the distinction between the external properties of an entity and the details of the entity's internal composition.
- **Abstract tool** = a component of a larger system whose internal composition we ignore.

Uses of abstraction

- Abstraction allows us to use things we don't fully understand.
- We all can use electrical devices, food, etc. that we either do not understand or cannot produce.
- Computer scientists can use algorithms implemented by others without understanding their details.

Outline of our study

- Design and construction of computing machines
  - Chapter 1: Data storage
  - Chapter 2: Data manipulation
- Chapter 3: Operating systems
- Chapter 4: Networks and the internet
- Chapter 5: Algorithms

Outline of our study (continued)

- Chapter 6: Programming languages
- Chapter 7: Software engineering
- Data organization
  - Chapter 8: Data abstractions
  - Chapter 9: Database systems
- Chapter 10: Artificial intelligence
- Chapter 11: Theory of computation

Social Repercussions

- Computers enable new activities, raising new questions:
  - Law: software producer's rights and liabilities
  - Ethics: what newly enabled activities are immoral?
  - Government: which newly enabled activities should be regulated?
  - Society: how do computers affect our lives?
- This book should prepare the reader to participate intelligently in debates of these issues.