Chapter 1:
The Role of Algorithms in Computing

《算法分析与设计》课程组
重庆大学计算机学院

Algorithm Design & Analysis

Introduction to Algorithm

The Role of Algorithms in Computing

OUTLINE

- 1.1 What are algorithms?
- 1.2 Why is the study of algorithms worthwhile?
- 1.3 Algorithms as a technology: How?
- 1.4 Textbook selected: What & Why?
- 1.5 Exercise & Mark

1.1 What are algorithm

- **What are algorithm?**
  - Definition{computational procedure, computational problem, Input, Output}
  - Examples with instance — sorting problem
  - Requirement: correct, precise description

Definition of algorithm

- Definition:
  - An ***algorithm*** is any well-defined computational procedure that
    - takes some value, or set of values, as ***input*** and
    - produces some value, or set of values, as ***output***.
- An algorithm is thus a sequence of computational steps that transform the input into the output.

Relationship: algorithm and problem

- Problem:
  - The statement of the problem specifies the desired input/output relationship.
- Algorithm:
  - The algorithm describes a specific computational procedure for achieving that input/output relationship.
- Relationship:
  - We can also view an algorithm as a tool for solving a well-specified ***computational problem***

Problem description: sorting problem

- Sorting problem:
  - **Input:** A sequence of n numbers $a_1, a_2, \ldots, a_n$
  - **Output:** A permutation (reordering) $a'_1, a'_2, \ldots, a'_n$ of the input sequence such that

$$a'_1 \leq a'_2 \leq \ldots \leq a'_n$$

- An instance of the sorting problem
  - Input: 31, 41, 59, 26, 41, 58 or <8 2 4 9 3 6>
  - Output: 26, 31, 41, 41, 58, 59
- Notation:
  - Sorting is a fundamental operation in CS
  - A large number of good sorting algorithms have been D&R

Algorithm description

- Specification:
  - Natural language, computer program, hardware design
    - An algorithm can be specified in English | Chinese, as a computer program, or even as a hardware design.
- Requirement
  - Precise description
    - The only requirement is that the specification must provide a precise description of the computational procedure to be followed.

Correctness of algorithm

- Correctness:
  - An algorithm is said to be ***correct*** if, for every input instance, it halts with the correct output.
- Incorrect algorithm
  - An incorrect algorithm might not halt at all on some input instances, or it might halt with an answer other than the desired one.
  - Incorrect algorithms can sometimes be useful, if their error rate can be controlled.
    - Example ?

1.2 Why is the study of algorithms w…?

- **Why is the study of algorithms worthwhile?**
  - **What is the role of algorithms?**
  - **What kinds of problems are solved by algorithms?**
    - The Human Genome Project:100,000 genes in human DNA, sequences of the 3 billion chemical base pairs
    - Internet: finding good routes.
    - Electronic commerce: Public-key cryptography
    - Road map: shortest path
    - Product of a sequence of n matrices $A_1\ A_2\ A_n$
    - Equation $ax \equiv b \pmod{n}$: integers
    - n points in the plane: find the convex hull

    - Know the strengths and limitations of ***data structures***
    - Hard problems: NP-complete, efficient algorithm, good | best

1.3 Algorithms as a technology

- **Algorithms as a technology**
  - infinitely fast: Terminates, with the correct answer
  - not infinitely fast: Computing time is therefore a bounded resource, algorithms that are efficient in terms of time or space
- **Efficiency:** algorithms-T, hardware-v, Software-c
  - Algorithms for sorting: Insertion sort, Merge sort

    $c_1=2$, $c_2=50$; $T_A=c_1n^2$, $T_B=c_2 n\log n$
  - Computer: A-v=1G, B-v=1M
  - n=1M: $t_A=?$ $t_B=?$

    n=100M: $t_A=?$ $t_B=?$

1.3 Algorithms as a technology

- **Algorithms as a technology**
  - infinitely fast: Terminates, with the correct answer
  - not infinitely fast: Computing time is therefore a bounded resource, algorithms that are efficient in terms of time or space
- **Efficiency:** algorithms-T, hardware-v, Software-c
  - Algorithms for sorting: Insertion sort, Merge sort

    $c_1=2$, $c_2=50$; $T_A=c_1n^2$, $T_B=c_2n\log n$
  - Computer: A-v=1G, B-v=1M
  - n=1M: $t_A=?2000s$ $t_B=?1000s$

    n=100M: $t_A=?23d$ $t_B=?1.5d$

Problem: Comparison of running times

- **Determine the largest size *n*** : For each function $f(n)$ and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							

1.4 Textbook selected: What & Why?

- **Textbook**
  - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.
  - **Introduction to Algorithms**.
  - 3rd Edition, the MIT Press, 2009.
- **Famous people with Classical Book in CS**
  - Donald Ervin Knuth, 1938.1.10-
  - Donald E. Knuth. The Art of Computer Programming, Volumes 1-4A,1st edition. Addison-Wesley Professional, March 13, 2011.

1.5 Exercise & Mark

- See 《算法分析与设计》课程介绍与要求.ppt

Exercises for Chapter 1

- **Exercises: 1.2-3: Find the smallest value of *n*…**
- **Problems 1-1: Comparison of running times…**

Exercises for Chapter 1

- 1. 1-3. Select a data structure that you have seen previously, and discuss its **strengths and limitations**.
- 1.1-4 How are the shortest-path and traveling-salesman problems given above **similar**? How are
- they **different**?
- 1.1-5 Come up with a real-world problem in which only the best solution will do. Then come up with one in which a solution that is "approximately" the best is good enough.

Exercises

- Exercises 1.2-3 What is **the smallest value of *n*** such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is 2^n on the same machine?
- Problems 1-1: **Comparison of running times**
  - For each function f(n) and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes f(n) microseconds.

Exercises

- Exercises 1.2-3 What is **the smallest value of *n*** such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is 2^n on the same machine?
- Problems 1-1: **Comparison of running times**
  - For each function f(n) and time determine the largest size n of solved in time t, assuming solve the problem takes f(

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$							
$\sqrt{n}$							
n							
$n \lg n$							
n^2							
n^3							
2^n							
$n!$							

《算法分析与设计》课程组
重庆大学计算机学院

End of Chapter