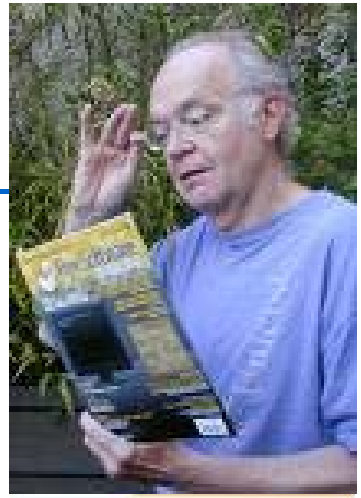
Analysis and Design of Algorithms

Chapter 1: Introduction

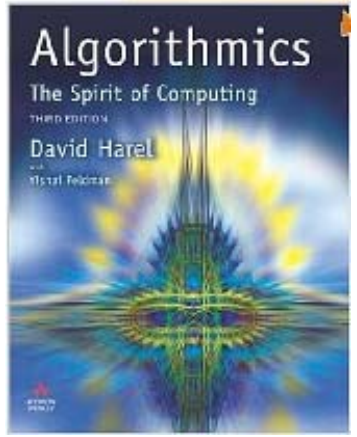
School of Software Engineering © YanlingXu
crystalxyl@163.com

Why we learn Algorithm ?



- ***Donald E. Knuth*** ***Stanford Univ. Turning Award 1974***
  - ***The Art of Computer Programming***

Computer Science is the study of algorithms.
Cornerstone of computer science.
Programs will not exist without algorithms.



- ***Algorithmics: the Spirit of Computing***

Prof. David Harel Dean of Faculty of Mathematics and Computer Science, the Weizmann Institute of Science

Algorithmics is more than a branch of computer science. It is the core of computer science, and, in all fairness, can be said to be relevant to most of science, business, and technology.

Only when you teach your computer technologies, you can get REAL control of it

Why we learn Algorithm ?

- ***Closely related to our lives***





- ***Help to guide how to analyze and solve problems***

- ***Help to develop the ability of analyzing and solving problems via computers***

Applications of algorithms

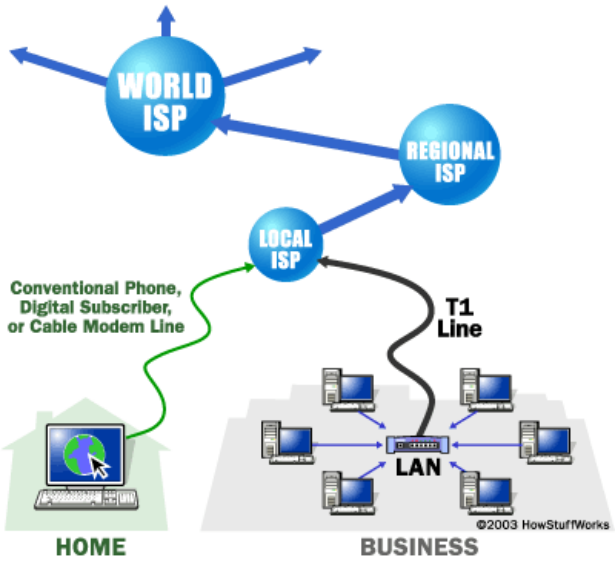
- ***Human Genome Project***
  - *identifying all the 100,000 genes in human DNA*
  - *determining the sequences of that make up human DNA, the 3 billion chemical base pairs*
  - *storing this information in databases*
  - *developing tools for data analysis*

  - ***ideas and techniques in this course are used in the solution of these biological problems***
  - ***accomplish tasks while using resources efficiently***
  - ***Savings in time, human, machine, and money***

Applications of algorithms

The Internet

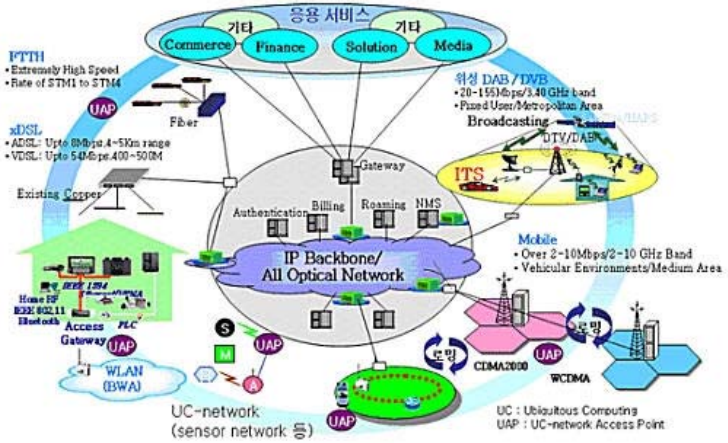
- *quickly access and retrieve large amounts of information*
- *algorithms are employed to manage and manipulate this large volume of data*
- *e.g. finding good routes on which the data will travel*
- *e.g. information search engine*



Applications of algorithms

Communications

- *How to transmit multimedia data*
- *How to organize different information streams on the network*
- *How to storage data on the network*
- *multimedia information retrieval*



Applications of algorithms

Cryptography in e- commerce

- ***to keep information such as credit card numbers, passwords, and bank statements private***
- ***Public-key cryptography and digital signatures***

Applications of algorithms

- ***In manufacturing and other commercial settings,***
  - ***An oil company may wish to know where to place its wells in order to maximize its expected profit.***
  - ***An airline may wish to assign crews to flights in the least expensive way possible, making sure that each flight is covered***
  - ***An Internet service provider may wish to determine where to place additional resources in order to serve its customers more effectively.***

— ***linear programming***

What we learn in this course ?

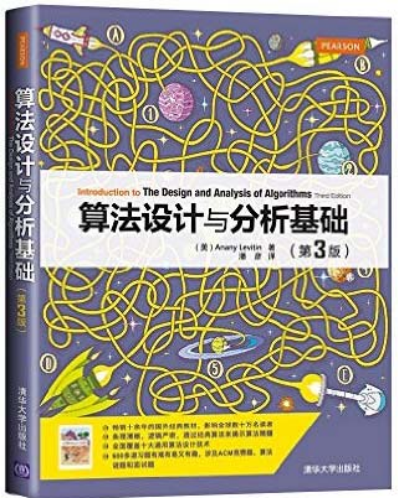
- ***systematical study of classical algorithms in the computer science area***
- ***master the typical techniques and methods of algorithms design***
- ***abilities of analyzing complexity of algorithms***
- ***be able to design algorithms for simple or complex practical problems***
- ***try to make the algorithms efficient and effective to enhance the quality of programming.***

References

算法设计与分析基础(第2版) .

（美）Anany Levitin 著，潘彦译 .

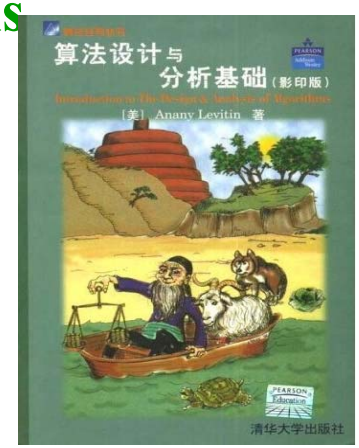
清华大学出版社 .2007年1月.



Introduction to the Design and Analysis of Algorithms.

Anany Levitin.

清华大学出版社. 2003年.

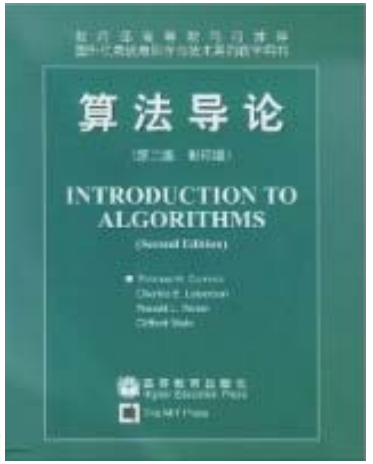


References

算法设计与分析(第三版). 王晓东.
电子工业出版社.2007年5月.

Introduction To Algorithms (Mit Press 2nd Edition). Thomas H.Cormen.
高等教育出版社& The MIT Press. 2002年5月.

计算机算法导引：设计与分析.
卢开澄. 清华大学出版社. 2006年.



Course Prerequisite

Data Structure

C, Java or other programming languages

Discrete Mathematics

Advanced Mathematics

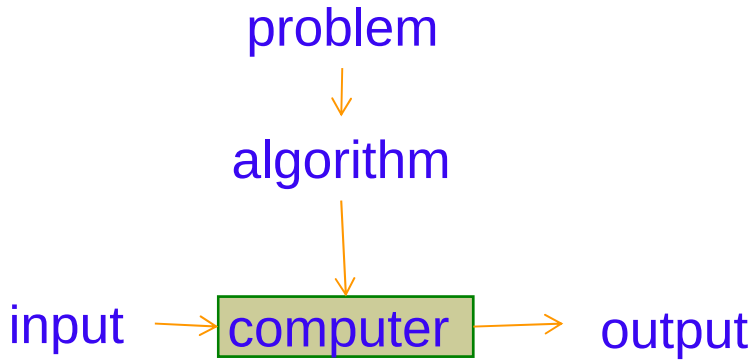
Chapter 1: Introduction

- **What's *Algorithm***
- **Example of Algorithm**
- **Algorithm vs. Program**
- **Algorithmic problem solving**
- **Contents of Algorithm**

What's Algorithm

Notion

- *The algorithm describes a specific computational procedure for solving a well-specified computational problem.*
- *The statement of the problem specifies in general terms the desired input/output relationship. An algorithm is for achieving that input/output relationship.*
- *Can achieve the desired output for any specific legitimate input in a finite amount of time.*



- *An algorithm is a finite sequence of unambiguous instructions*

What's Algorithm

Properties of algorithms

- ***Input:*** 0 or more valid input values, to provide the initialization conditions
- ***Output:***
  - produce the correct output given a valid input
  - at least one value is produced by the algorithm
  - desired input/output relationship specified by the problem
- ***Definition:***
  - each instruction / each step is clearly
  - precisely and unambiguously specified

Example: 不符合确定性的运算

- 5/0
- 将6或7与x相加
- 未赋值变量参与运算

What's Algorithm

Properties of algorithms

- ***Finiteness:***
  - finite instructions,
  - finite execution times for each instruction
  - finite running time for each instruction

- ***Feasibility:*** could be precisely executed and effectively computable;
  Steps must be sufficiently simple and basic.

What's Algorithm

Some points for algorithms

- Each step of an algorithm must be ***unambiguous.***
- ***Different algorithms*** for a certain problem
- ***Different representations*** to describe a certain algorithm
- ***Different ideas and different execution speed*** for different algorithms

Example of Algorithm

- ***Problem:*** Computing the Greatest Common Divisor of two integers
  - *gcd(m, n): the largest integer that divides both m and n*
- ***Algorithm I***
  - ***Euclid's algorithm:***

    gcd(m, n) = gcd(n, m mod n) iteratively while n≠0

    gcd(m, 0) = m
  - ***Natural language***

Step1: If n = 0, return the value of m as the answer and stop;

otherwise, proceed to Step 2.

Step2: Divide m by n and assign the value of the remainder to r.

Step 3: Assign the value of n to m and the value of r to n. Go to Step 1.

Example of Algorithm

- ***Pseudocode***
  - *A mixture of a natural language and programming language-like structures*
  - *Precise and succinct.*
  - *Pseudocode in this course*
    - omits declarations of variables
    - use indentation to show the scope of such statements as for, if, and while.
    - Use ←for assignment

Algorithm *Euclid(m, n)*
//Computes gcd(m, n) by Euclid's algorithm
//Input: Two nonnegative, not-both-zero integers m and n
//Output: Greatest common divisor of m and n
while n ≠ 0 do
    r ← m mod n
    m ← n
    n ← r
return m

Example of Algorithm

Algorithm II

Consecutive Integer Algorithm

Step1: Assign the value of min{m, n} to t.

Step2: Divide m by t. If the remainder of this division is 0, go to Step3; otherwise, go to Step 4.

Step3: Divide n by t. If the remainder of this division is 0, return the value of t as the answer and stop;
otherwise, proceed to Step4.

Step4: Decrease the value of t by 1. Go to Step2.

Example of Algorithm

Algorithm II

Consecutive Integer Algorithm

```
//使用连续整数检测法计算gcd(m，n)
//输入：两个不全为0的非负整数m,n
//输出：m,n的最大公约数
if n=0 return n
      t=min{m,n}
     while t>0 do
           if (m mod t)==0
               if (n mod t)==0
                  return t
               else t=t-1
          else t=t-1
      return t
```

Example of Algorithm

Algorithm III ?

Middle-school procedure

Step1: Find the prime factors of m. ?

Step2: Find the prime factors of n. ?

Step3: Identify all the common factors in the two prime expansions found in Step1 and Step2. (If p is a common factor occurring Pm and Pn times in m and n, respectively, it should be repeated in min{Pm, Pn} times.)

Step4: Compute the product of all the common factors and return it as the gcd of the numbers given.

Algorithm vs. Program

Similarity

- *Finite sequence of instructions*

Difference

- *Presentation:*

  Algorithm —— *Nature language, pseudo code, flow charts*

  Program —— *Coded using some specific programming language*
  Could be executed by some specific machine

- *Execution:*

  Algorithm —— *finite steps*

  Program —— *could be infinitely executed*

  e.g. Operating system

  - *not an algorithm, but a program running in infinite circles*
  - *each task could be viewed as subprogram according to specific algorithm*

Algorithm vs. Program

Difference

Definition:

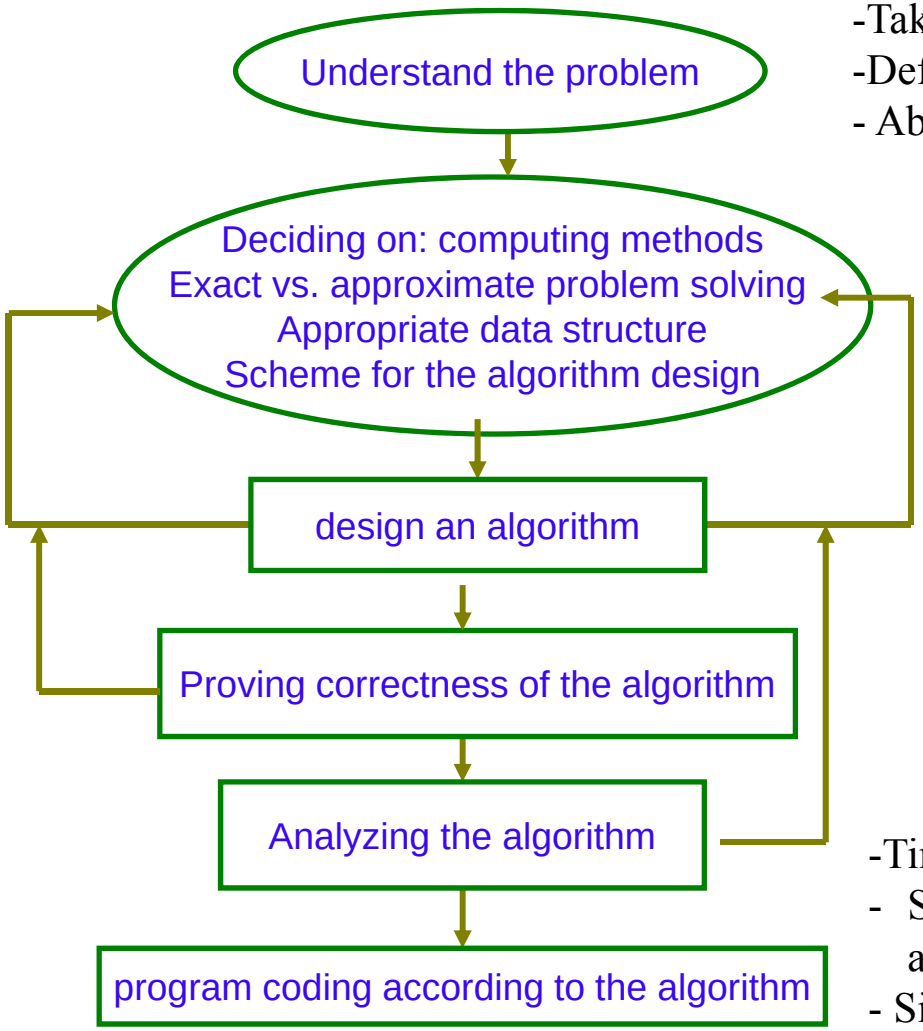
Algorithm *— a step by step outline or flowchart how to solve a problem*

Program *— an implemented coding of a solution to a problem based on the algorithm*

Algorithm + data structure = program

Algorithmic problem solving

Algorithm Design and Analysis Process



-Understand the problem description
-Try some examples manually
-Take into consideration special examples
-Define the input
- Abstract the problem and get its mathematical description

- Equipment performance
- Computing methods: sequential or parallel
- Exact solution is unavailable or speed is unacceptably low
- Algorithm + data structure = program

- Nature language
- pseudo code
- flow charts

- For every legal input, the algorithm will produce a desired output in finite time
- Mathematical Induction
- to prove its correctness or incorrectness ?

-Time efficiency : how fast the algorithm runs
- Space efficiency: how much extra memory the algorithm needs.
- Simpleness and commonness

Contents of Algorithm

Algorithm Design Techniques/Strategies

- *Brute force* 蛮力法
- *Divide and conquer* 分治法
- *Decrease and conquer* 减治法
- *Transform and conquer* 变治法
- *Greedy approach* 贪心算法
- *Dynamic programming* 动态规划
- *Back tracking* 回溯法
- *Branch and bound* 分支界限法

Contents of Algorithm

- ***How to analyze algorithm efficiency***
  - ***How good is the algorithm?***
    - ***time efficiency***
    - ***space efficiency***
  - ***Does there exist a better algorithm?***
    - ***lower bounds***
    - ***optimality***

Contents of Algorithm

Important problem types

- *sorting* 排序
- *searching* 查找
- *string processing* 串处理
- *graph problems* 图问题
- *combinatorial problems* 组合问题
- *geometric problems* 几何问题
- *numerical problems* 数值问题

Contents of Algorithm

Fundamental data structures

- *linear data structure*
  - *array* 数组
  - *linked list* 单（双）链表
  - *string* 串
  - *stack* 栈
  - *queue* 队列
- *graph* 图
- *tree* 树
- *set and dictionary* 集合

思考题

1. Prove the equality gcd(m, n) = gcd(n, m mod n) for every pair of positive integers m and n.

2. What does Euclid's algorithm do for a pair of numbers in which the first number is smaller than the second one? What is the largest number of times this can happen during the algorithm's execution on such an input?

上机练习

1-1. Computing *gcd(m, n)*

1) Compose a program using Euclid's algorithm
2) Compose a program using Consecutive Integer Algorithm
3) Find gcd(31415, 14142) by applying Euclid's algorithm
4) Estimate how many time faster it will be to find gcd(31415, 14142) by Euclid's algorithm compared with the algorithm based on checking consecutive integers from $\min\{m,n\}$ down to gcd(m,n)

1-2. find the binary representation of a positive decimal integer

Compose a program

上机练习

1-3. Element uniqueness problem

1) a) Compose a program using *UniqueElement* algorithm on P63

b) Check its efficiency in worst case, best case, and average case, in your program

2) a) Compose a program using the method in which the array is sorted firstly

b) Check its efficiency in worst case, best case, and average case, in your program