Analysis and Design of Algorithms

Chapter 1: Introduction

School of Software Engineering © Ye Luo

Introduction of Teacher

- **Contact Information**

  Office: Jishi Building, Room 314

  Email: yeluo@tongji.edu.cn

- **Course Information:**

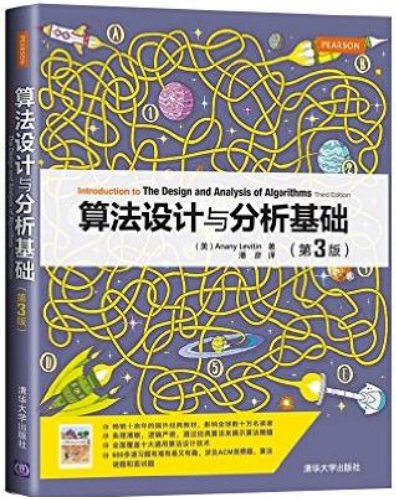
  Slides can be downloaded from:

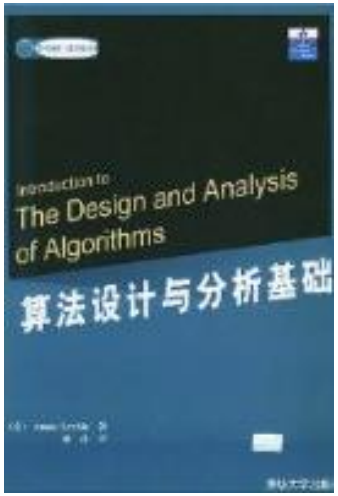
- *QQ 群 共享文件*

- **TA Information:**

Our References

算法设计与分析基础(第3版) .
（美）Anany Levitin 著，潘彦译 .
清华大学出版社 .2015年2月.

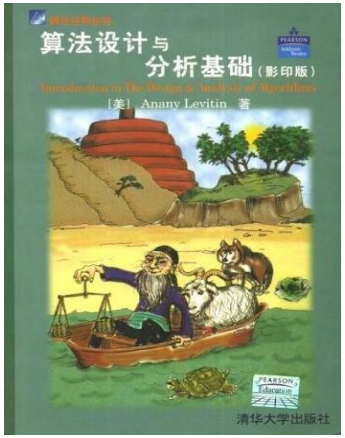




Introduction to the Design and Analysis of Algorithms.

Anany Levitin.

清华大学出版社. 2003年.



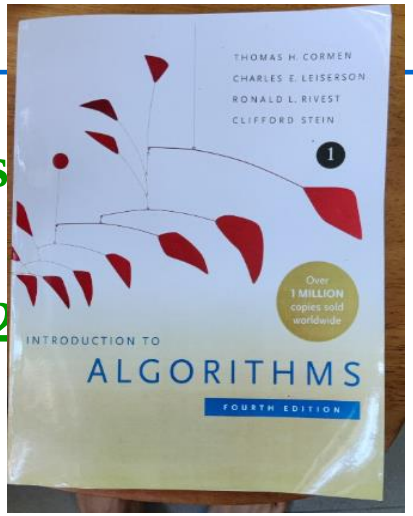
Survey of Reference Books Worldwide

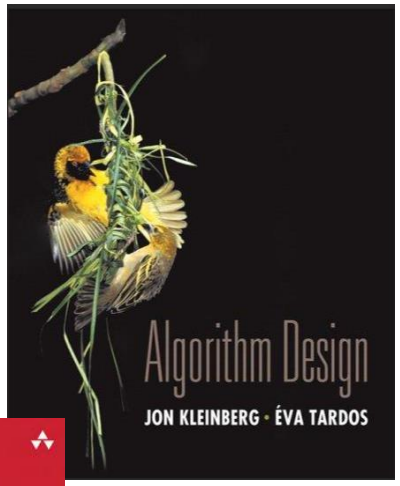
编号	教材	使用次数
[1]	Introduction to Algorithms, by Cormen, Leiserson, Rivest, and Stein.	32
[2]	Algorithm Design, by Kleinberg and Tardos.	20
[3]	Algorithms, by Dasgupta, Papadimitriou and Vazirani.	8
[4]	Algorithms, by Sedgewick and Wayne.	1
[5]	An Introduction to Algorithms: their Methods and Madness, by A.R.Siegel.	2
[6]	Data Structures and Network Algorithms, by R. E. Tarjan.	1
[7]	Randomized Algorithms, by Motwani and Raghavan.	2
[8]	Programming Pearls, by Bentley.	1
[9]	Introduction to Algorithms: a Creative Approach, by Manber.	1
[10]	Probability and Computing, by Mitzenmacher and Upfal.	1
[11]	The Probabilistic Method, by Alon and Spencer.	2
[12]	The Design of Approximation Algorithms, by Williamson and Shmoys.	2
[13]	The Art of Computer Programming, by Knuth.	2
[14]	Algorithmics, by Harel.	2
[15]	Data Structures and Algorithm Analysis, by Shaffer.	1
[16]	Algorithms in C, by Sedgewick.	1
[17]	Computer Algorithms, by Baase and Gelder.	1
[18]	The Design and Analysis of Computer Algorithms, by Aho, Hopcroft and Ullma	1
[19]	Intorduction to The Design and Analysis of Algorithms, by Levitin.	1
[20]	Algorithms, by Johnsonbaugh.	1
[21]	Approximation Algorithms, by Vazirani.	1
[22]	Algorithms Unlocked, by Cormen.	1
[23]	Data Structures and Algorithm Analysis in Java, by Weiss.	1
[24]	Fundamentals of Algorithms, by Brassard and Brately.	1
[25]	Introduction to the Theory of Computation, by Sipser.	1

Classical References

Introduction To Algorithms (MIT Press th Edition). Thomas H.Cormen.

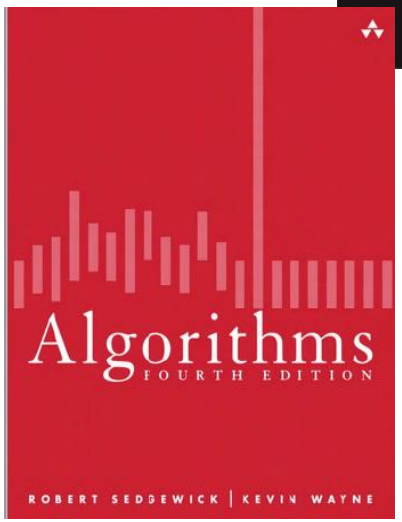
高等教育出版社& The MIT Press. 2022年.《算法导论》





Algorithm Design (Pearson Press 1st Edition). Jon Kleinberg, Eva Tardos, Cornell University

Algorithms (Pearson Press, 4th Edition). Robert Sedgewick, Kevin Wayne, 2011, Princeton University



Classical Reference 1

Textbook – CLRS Book

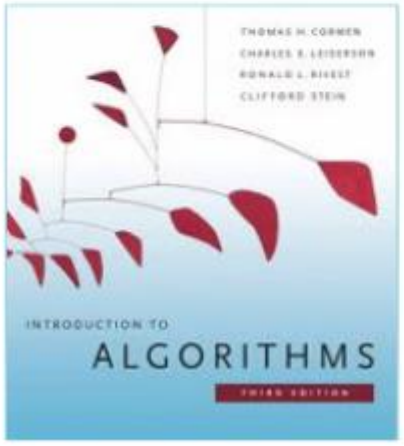
Title: Introduction to Algorithms

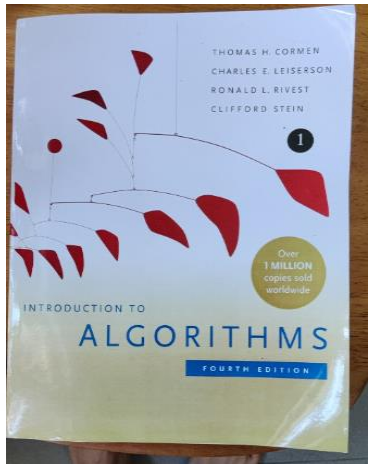
Author: T. Cormen, C. Leiserson, R. Rivest, C. Stein

Publisher: The MIT Press, 2022 (First Edition in 1990)

ISBN-10/13: 0262033844 / 978-0262033848

Edition: Second/Third Edition/Forth/Fifth Edition





Classical Reference 1

Universities using CLRS Book

- 24 / Top 40 Univ. use CLRS book (33 courses)

Classical Reference 2

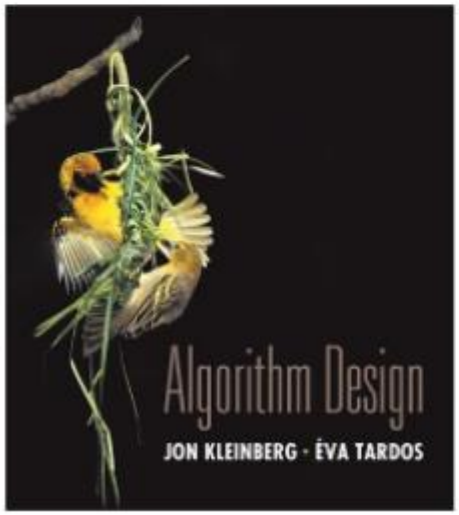
Textbook – Cornell Book

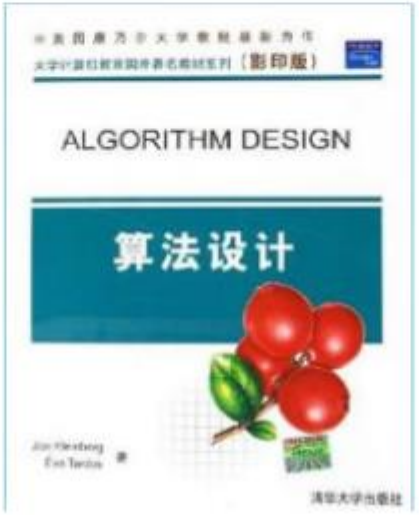
Title: Algorithm Design

Author: J. Kleinberg, E. Tardos

Publisher: Addison Wesley, 2006.

ISBN-10/13: 0321295358 / 978-0321295354





Classical Reference 2

The Authors – Cornell Book

Jon Kleinberg
Cornell
(Ph.D. from MIT)

Éva Tardos
Cornell University
(Hungarian mathematician, Gödel Prize)

- 1st Edition: 2006 – Coauthored with Éva Tardos
- 2nd Edition: 2013 – Sole Author

Jon Kleinberg's younger brother → Robert Kleinberg@Cornell

Chinese Version 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.

计算机算法导引: 设计与分析.
卢开澄. 清华大学出版社. 2006年.

Survey of Topics Introduced Worldwide

Totally 10 Topics

Algorithm Topics	Percentage
Algorithm Design and Analysis	95.2%
Graph Algorithms	92.9%
Approximation	88.1%
Dynamic Programming	83.3%
Greedy Approach	79.5%
Linear Programming	78.6%
Divide and Conquer	73.8%
Randomized Algorithm	73.8%
NP-Completeness	71.4%
Amortized Analysis	66.7%

Topics Introduced in Our Class

序号	上课时间	上课内容
1	第1周 星期4 第3节-第4节	基本概念和算法导引
2	第2周 星期4 第3节-第4节	算法效率分析基础
3	第3周 星期4 第3节-第4节	算法效率分析基础，蛮力法
4	第4周 星期4 第3节-第4节	分治法思想、框架及案例
5	第5周 星期4 第3节-第4节	减治、变治法思想及框架
6	第6周 星期4 第3节-第4节	变治法案例、搜索算法简介
7	第7周 星期4 第3节-第4节	回溯搜索算法及案例
8	第8周 星期4 第3节-第4节	分支限界搜索算法及案例
9	第9周 星期4 第3节-第4节	动态规划简介及案例1
10	第10周 星期4 第3节-第4节	五一放假
11	第11周 星期4 第3节-第4节	动态规划案例2
12	第12周 星期4 第3节-第4节	动态规划案例3
13	第13周 星期4 第3节-第4节	动态规划案例4及总结、拓展
14	第14周 星期4 第3节-第4节	贪心算法简介及案例1
15	第15周 星期4 第3节-第4节	贪心算法案例2
16	第16周 星期4 第3节-第4节	算法复杂度分析P\NP问题
17	第17周 星期4 第3节-第4节	习题课及总复习

Examination

- *Homework: 40%, 4 times(10%+10%+10%+10%)*
- *Final examination: 50%*
- *Attendance: 5% (being absent >=5 times, you will fail this course)*
- *Class activity: 5% being active in class and answering questions correctly*

Course Prerequisite

Data Structure

C, Java or other programming languages

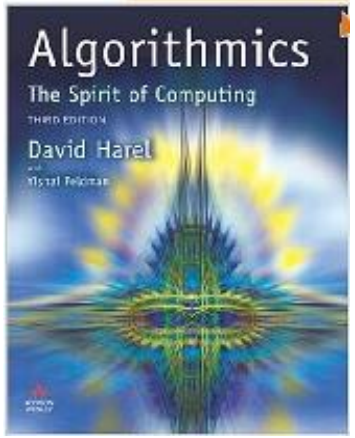
Discrete Mathematics

Advanced Mathematics

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*

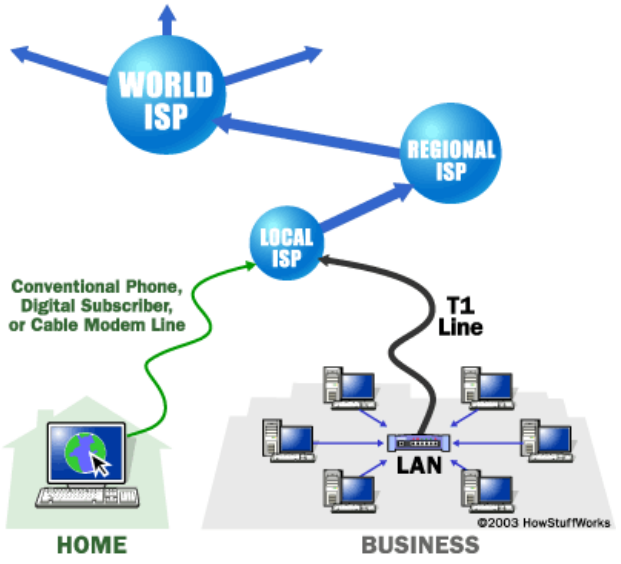
Applications of algorithms

- **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

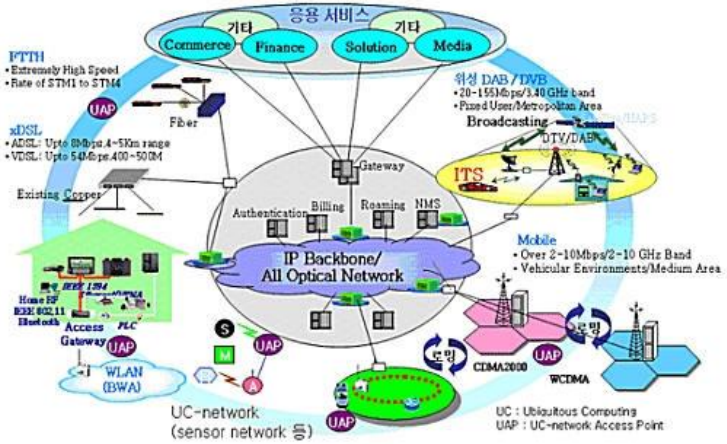
- **The Internet**
  - *quickly access and retrieve large amounts of information*
  - *algorithms are employed to manage and manipulate this large volume of data*
  - *e.g. finding good routes on which the data will travel*
  - *e.g. information search engine*



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.***

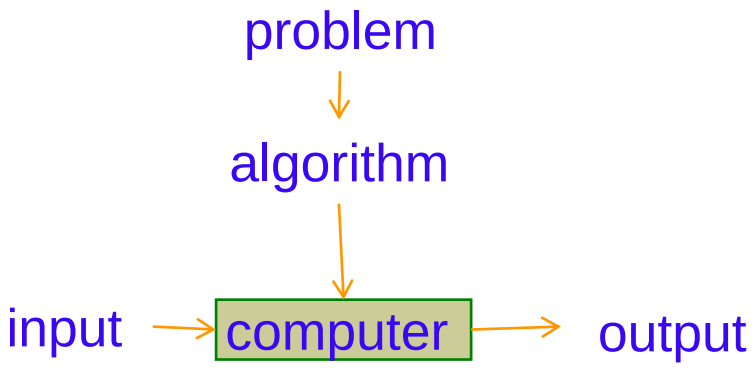
Chapter 1: Introduction

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

What's Algorithm

Notation

- *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
- ***Tangibility(确定性):***
  - 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

```
Algorithm ConsecutiveInteger(m, n)
//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 flowc*

  Program — *an implemented coding of a sc based on the algorithm*

  Algorithm + data structure = progra

Jeff Dean (@🏠)
@JeffDean

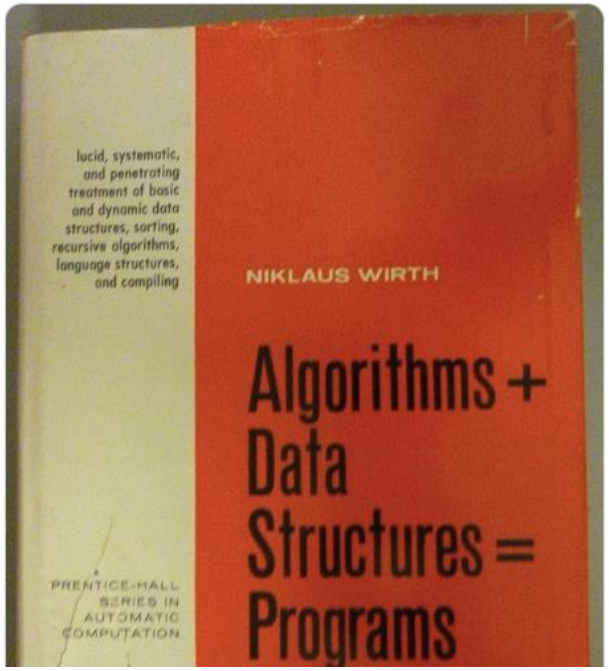
Very sad to hear. 😢 Niklaus Wirth made many enormous contributions to the field of computing. Pascal was the first language I used seriously (initially on the UCSD p-System and later via Turbo Pascal), and I got my hands on this great book that he wrote when I was in middle school. Later, I loved using Modula-2 for some things (and his indirect influence on Modula-3 was strong).

I also love the anecdote when asked about how to pronounce his name:

"Whereas Europeans generally pronounce my name the right way ('Ni-klows Wirt'), Americans invariably mangle it into 'Nick-les Worth'. This is to say that Europeans call me by name, but Americans call me by value.

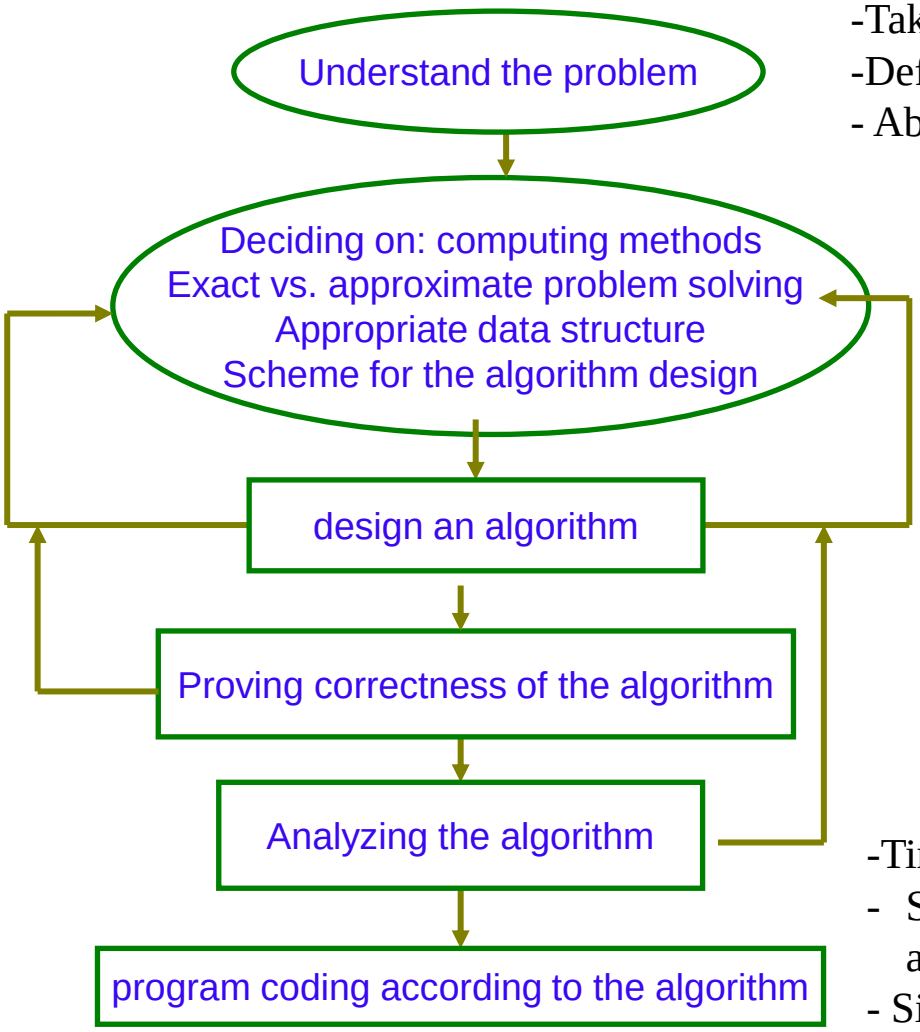
Niklaus Wirth"

翻译帖子



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

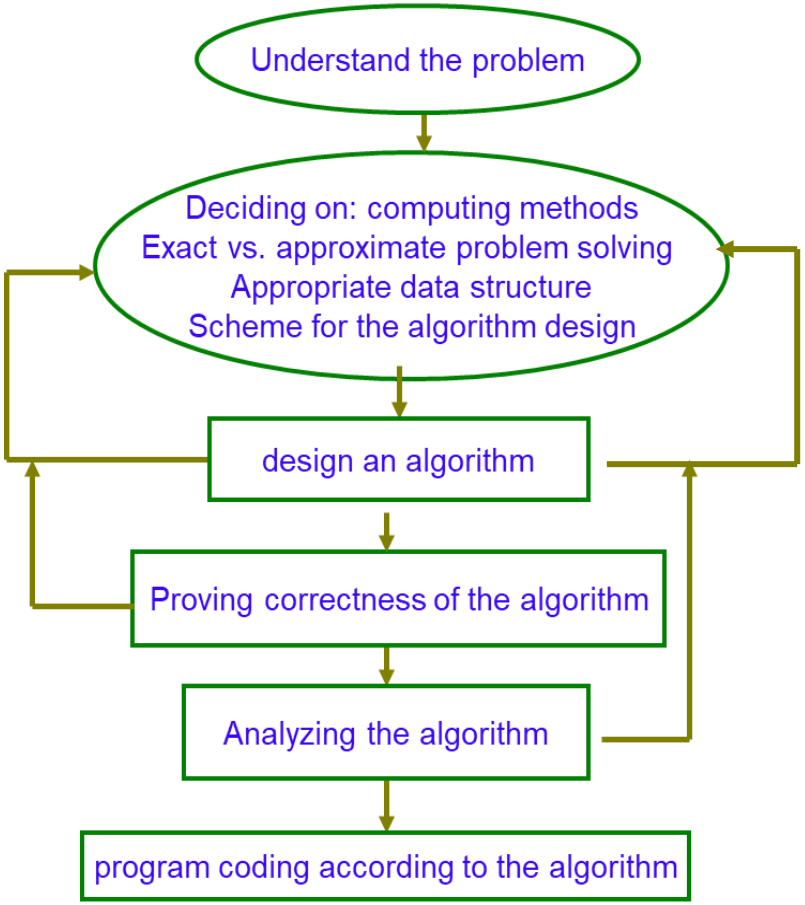
- Nature language
- pseudo code
- flow charts

- 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

Algorithmic problem solving

Algorithm Design and Analysis Process



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

Summary

- 算法的定义：在有限时间内，对问题求解的一个清晰的指令序列。算法的每个输入确定了该算法求解问题的一个实例。
- 算法的特点：输入，输出，确定性，有穷性，和可行性。
- 算法可以用自然语言或者伪代码表示，或计算机程序实现
- 一个好的算法常常是不懈努力和反复修改的结果。
- 算法操作的是数据，所以数据结构很重要。

思考题

证明：令d = gcd(m, n)，则d|m 且d|n。设m = kn + r(0≤ r < n)，则 d|(kn+r)。又有d|n，因此
d|kn，所以有d|r。即我们由d|m 且d|n 这个前提可以得出d|r。换就话说，我们也可以说成由d|m
个前提推出了d|n 且 d|r。使用类似的推理过程同样可以得到：由d|n 且 d|r 可以推出d|m 且 d|n
这里的r 即 m mod n。因此我们可以说gcd(m, n) = gcd(n, m mod n)是双向成立的，命题得证。

分享到

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

   http://blog.csdn.net/deserthero2013/article/details/51161696

1. 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

代码要求

1-1. GCD (1)

```
int gcd_Euclid (int m, int n)
{  // computes the greatest common divisor of two integers m and n
      using Euclid algorithm;
   // Input: two integers;
   // Output: their GCD


}
```

假定m, n 都是自然数，使用欧几里德算法求m, n的最大公约数，作为返回值;

注意：特殊情况 $m < n$

代码要求

1-1. GCD (2)

```
int gcd_ConsecutInteger (int m, int n)
{  // computes the greatest common divisor of two integers m and n
using Consecutive Integer Algorithm ;
    // Input: two integers;
    // Output: their GCD


}
```

假定m, n 都是自然数，使用连续整数递减法求m, n的最大公约数，作为返回值；

注意：特殊情况 $m < n$

代码要求

1-2. binary representation of a positive decimal integer

```
int convert_decimal_to_binary(int dec_number)
{  // find the binary representation of a positive decimal integer;
   //Input: a positive decimal integer;
   //Output: binary integer;


}
```

结果输出到屏幕上（cout，printf），结果输出到一行
注意：要求不可以有前置的0，例如，结果不能是 00001111000
而是 1111000

代码要求

1-3. Element uniqueness problem (1)

```
bool  UniqueElement (int A[0,…, n-1], int size)
 {  // Determines whether all the elements in a given array A are
       distinct using the definition based algorithm;
    // Input: an array A[0,…, n-1]) ;
    // Output: returns true if all the elements are distinct, and false
       otherwise;

}
```

代码要求

1-3. Element uniqueness problem (2)

```
bool  UniqueElement_sort (int A[0,…, n-1], int size)
 {  // Determines whether all the elements in a given array A are
       distinct using sorting firstly;
   // Input: an array A[0,…, n-1]) ;
   // Output: returns true if all the elements are distinct, and false
       otherwise;

}
```

代码要求

提交代码注意：

所有算法写入到一个c/cpp文件中，

文件命名 10xxxx_姓名_algo_assignment1.c

文档要求

对三个练习分别给出

1. 算法本身的思路，可以用伪代码或者自然语言描述
2. 程序中用于计算算法复杂度的方法，用文字或者公式描述清楚