



算法设计 与分析

Design and
Analysis of
Algorithms



"On my summer vacation I wrote a computer algorithm. I sold it, for thirty million dollars, to a social networking website."

哈尔滨工业大学（威海）计算机科学与技术学院暨软件学院

2019/9/3

2019年9月 Autumn

1

算法设计与分析

Design and Analysis of Algorithms

- 对象：软件工程专业
- 学分：2 学分
- 学时：32课堂+ 8实验
- 考试：笔试（70分）、实验（20分）
• 平时（10分）
- 先修：数据结构、程序设计语言、
• 操作系统、数据库原理

主讲：朱东杰 博士、硕导
地点：M楼305
电话/微信：18953856806
Email：zhudongjie@hit.edu.cn



算法设计与分析

Design and Analysis of Algorithms

一、课程教学目的

软件的执行效率和稳定性取决于其中所采用的算法，本课程的目的是使学生系统地掌握算法设计与分析的理论与方法，为从事计算机软件开发技术方面的工作，奠定坚实的理论基础。

二、课程教学目的

- 掌握计算机算法分析的基本方法及常见算法设计方法
- 训练逻辑思维
- 利用常见的算法设计方法解决软件开发中的实际问题
- 通过学习算法设计与分析课程，开阔学生的编程思路，提高编程质量

算法设计与分析

Design and Analysis of Algorithms

		第1学年			第2学年			第3学年			第4学年	
		秋	春	夏	秋	春	夏	秋	春	夏	秋	春
通识教育课程 50学分	公共基础课程	体育		体育(游泳)	体育							
		大学英语			大学英语							
		思想道德修养与法律基础	中国近代史纲要		毛泽东思想和中国特色社会主义理论体系概论	马克思主义原理						
		军训及军事理论	形式与政策									
		文化素质教育课程(10学分)										
专业教育课程 110学分	文理通识课程	工科数学分析										
	文化素质教育课程	代数与几何	微积分与数理统计									
	数学与自然科学基础课程											
	专业基础课程	软件工程导论	离散数学I 数字逻辑		离散数学II 计算机组成原理 数据结构	操作系统 数据库系统 计算机网络		算法设计与分析 编译原理				
	专业核心课程	高级程序设计语言I	高级程序设计语言II		软件工程概论	系统分析与设计 面向对象建模技术		体系结构与模式 软件开发过程与项目管理	软件测试与质量保证			
	方向核心课				Java程序设计(限选)	汇编与接口技术(限选)		方向核心课: 服务科学与工程(4) 物联网工程(5) 云计算与大数据(4) 数字媒体技术(4)	方向核心课: 服务科学与工程(6) 物联网工程(5) 云计算与大数据(6) 数字媒体技术(6)			
	专业选修课											
	实习实训			程序设计实践	数据结构课程设计		软件开发实践 工业实训	综合课程设计			工业实习与毕业论文	
个性化发展课程 (10学分)				专业任选课、外专业基础与核心课程(8学分)								
				创新创业课程、创新创业实践(2学分)								

本课程

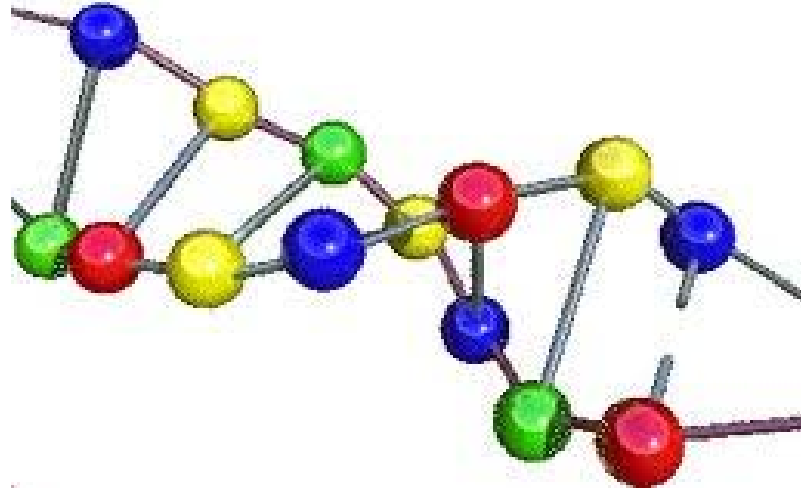
为什么要学习算法？

- 算法不仅是计算机科学的一个分支，它更是计算机科学的核心，而且，可以毫不夸张地说，它和绝大多数的**科学、商业和技术**都是相关的。——David Harel 《算法：计算的灵魂》
- **程序=数据结构+算法 (+编程)**
- 开发人们的分析能力
- 作为一种技术的算法

一个人只有把知识教给“计算机”，才能“真正”掌握它。

算法可以解决哪些问题

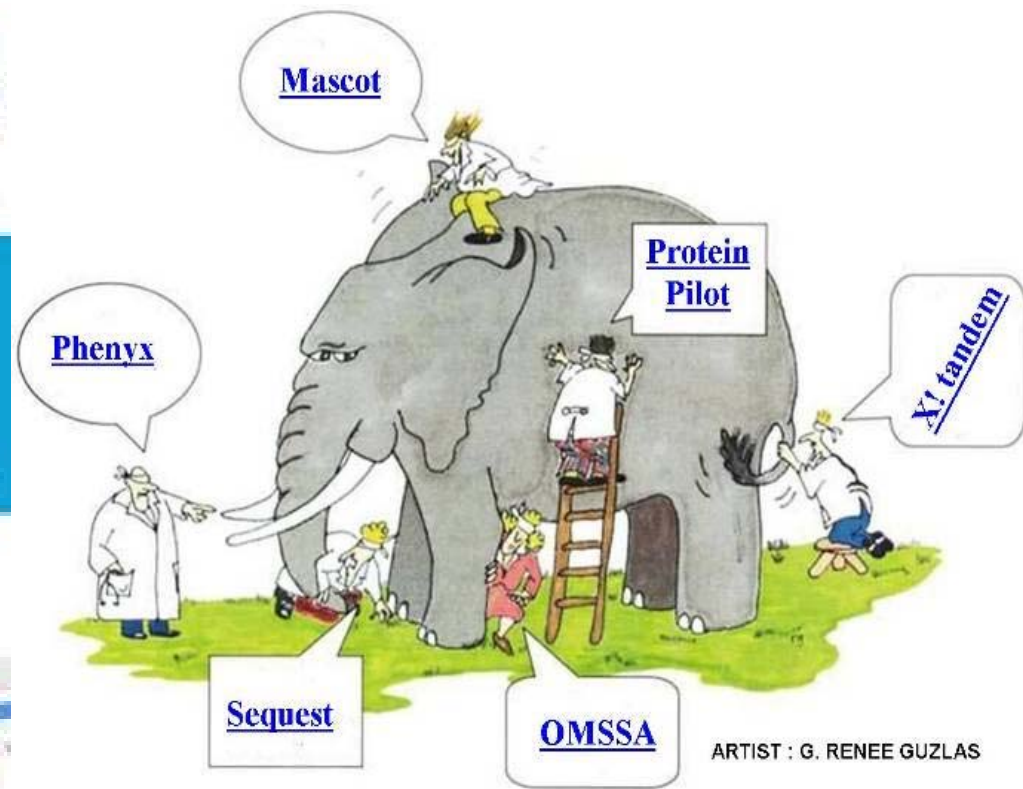
- 找出人类DNA中所有1000000中基因，确定构成人类DNA的30亿种化学基对的各种序列。



典型大数据问题

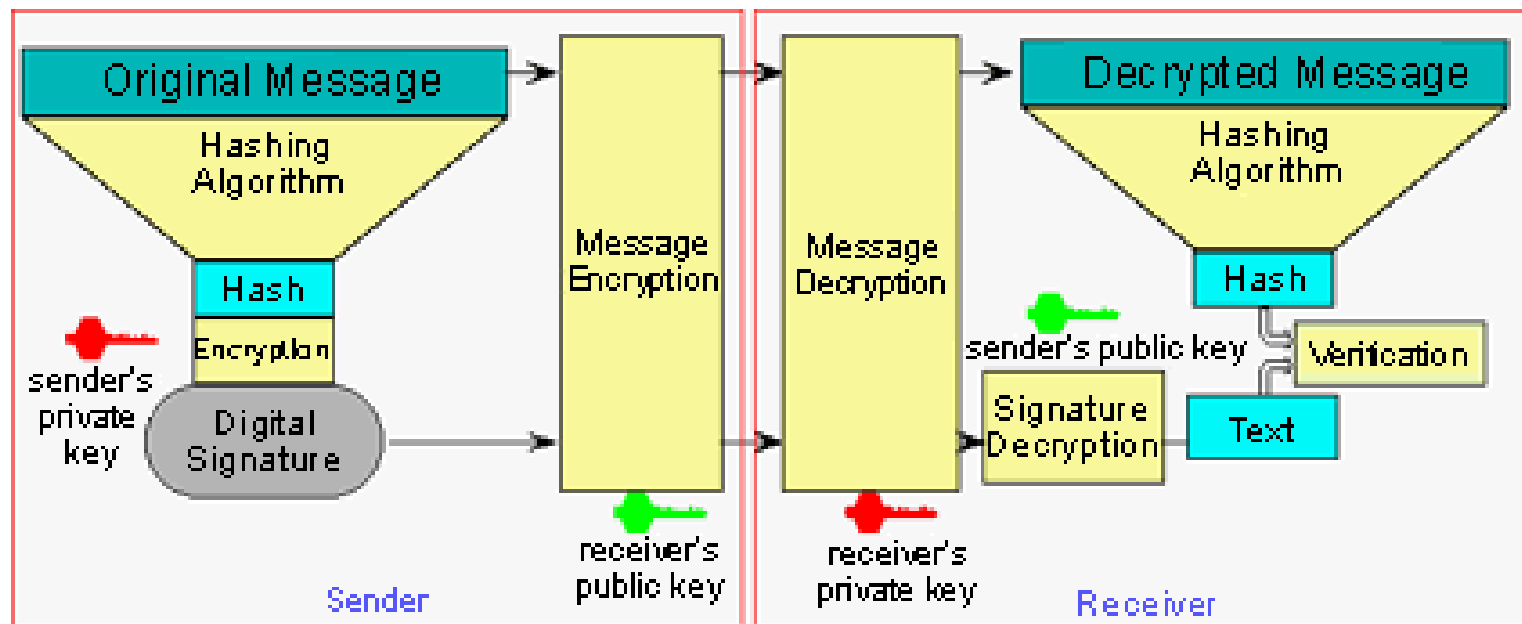
算法可以解决哪些问题

- 快速地访问和检索互联网数据



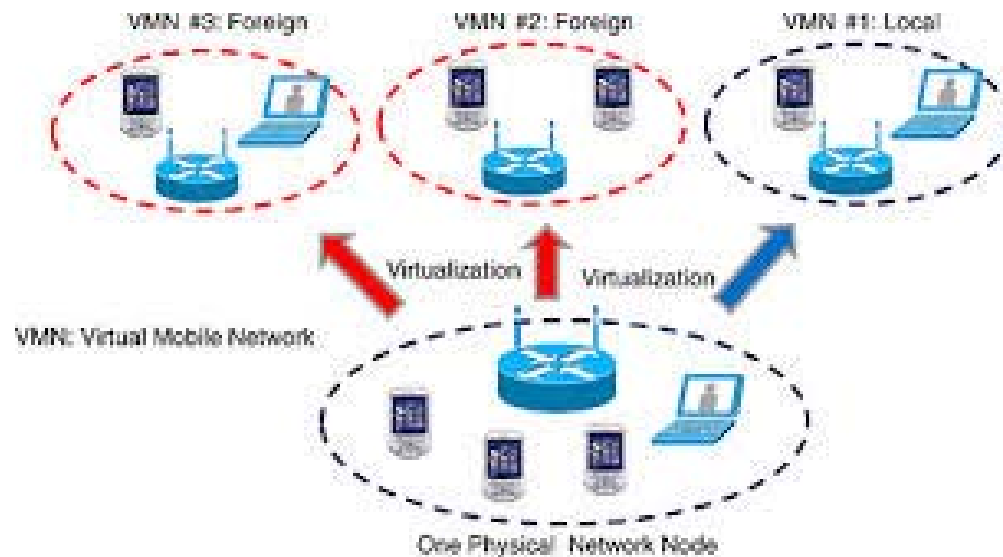
算法可以解决哪些问题

- 电子商务活动中各种信息的加密及签名



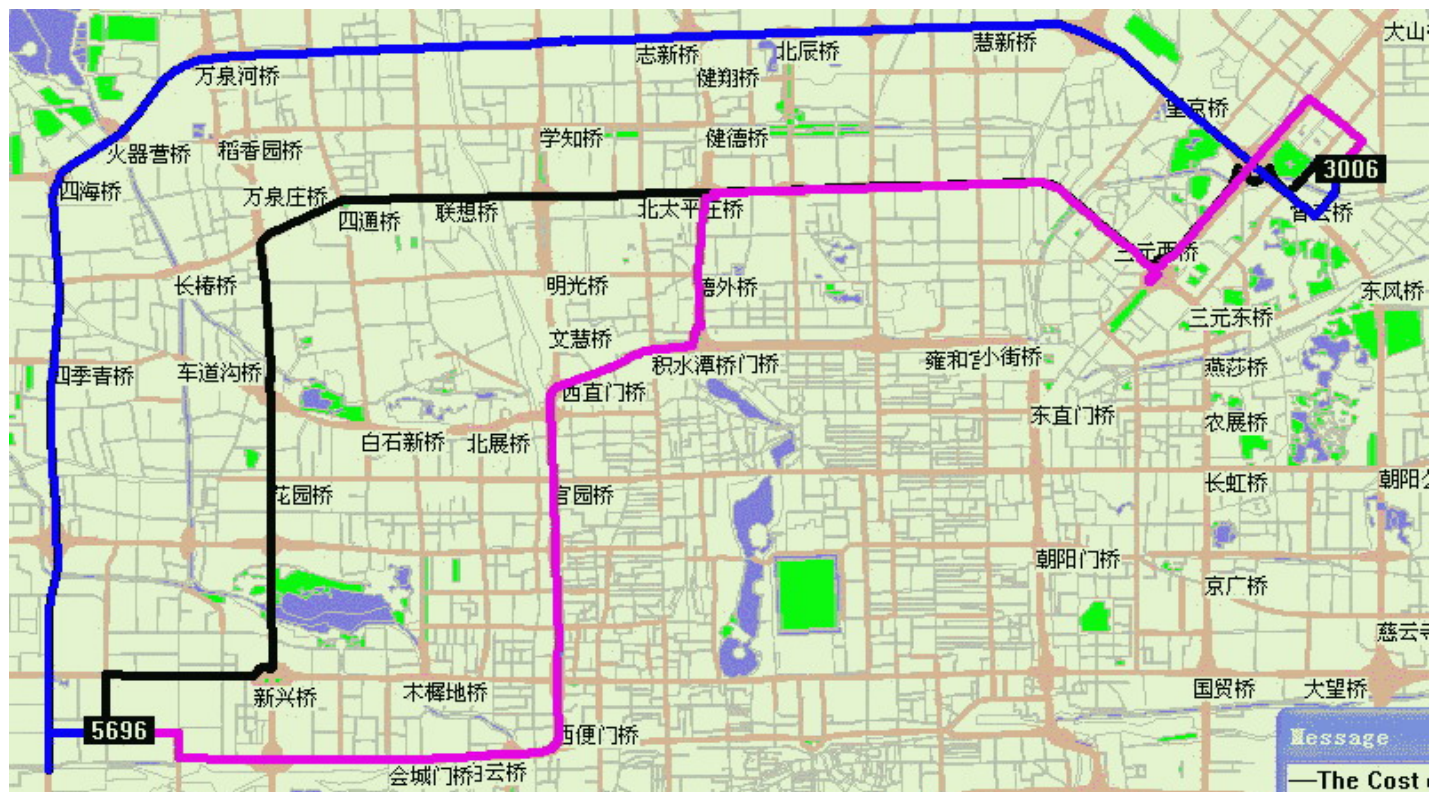
算法可以解决哪些问题

- 制造业中各种资源的有效分配



算法可以解决哪些问题

- 确定地图中两地之间的最短路径

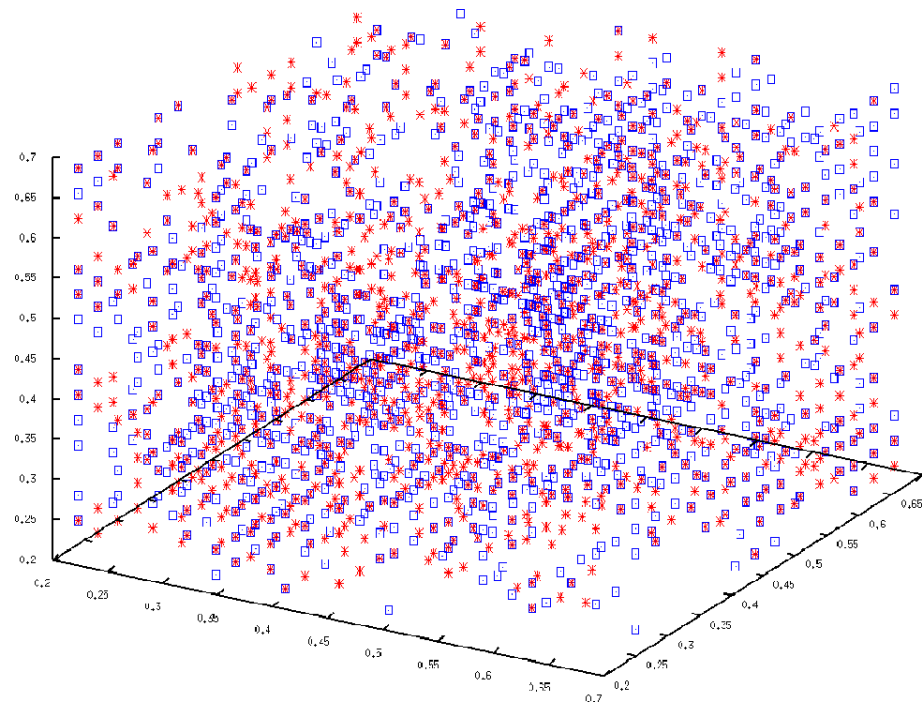


无人驾驶

算法可以解决哪些问题

- 各种数学、几何计算（矩阵、方程、集合）

2	9	5	7	34	134	8	6	134
47	3	1	8	6	5	49	2	47
8	47	6	1249	2349	12349	459	1349	13457
1349	148	7	249	5	249	249	13489	6
1469	146	29	3	8	7	2459	149	145
5	48	2389	249	1	6	7	3489	348
367	678	38	5	2347	234	1	48	9
179	2	89	6	479	149	3	5	48
139	5	4	19	39	8	6	7	2



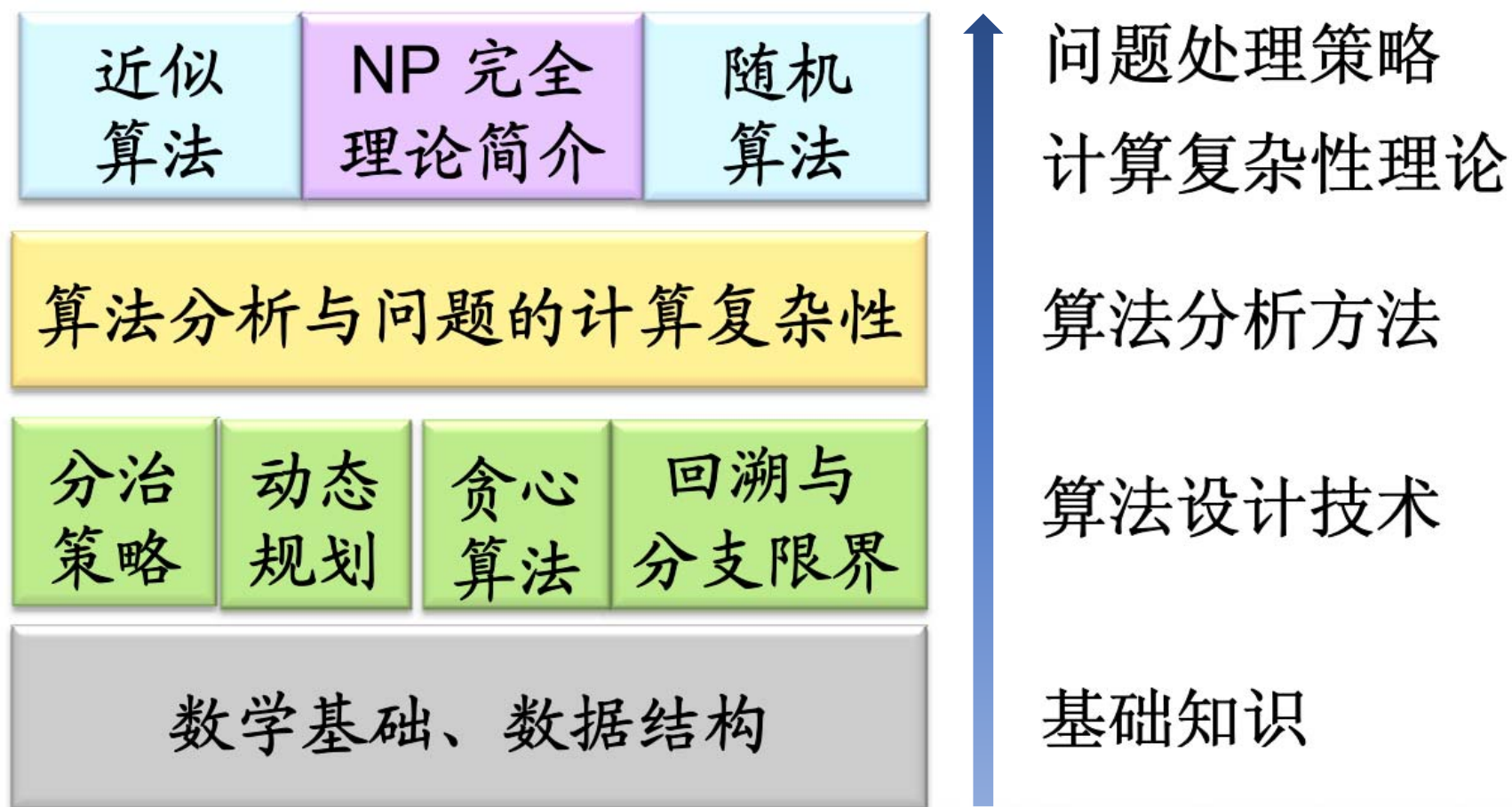
算法可以解决哪些问题



大数据、智能计算

算法设计与分析

Design and Analysis of Algorithms

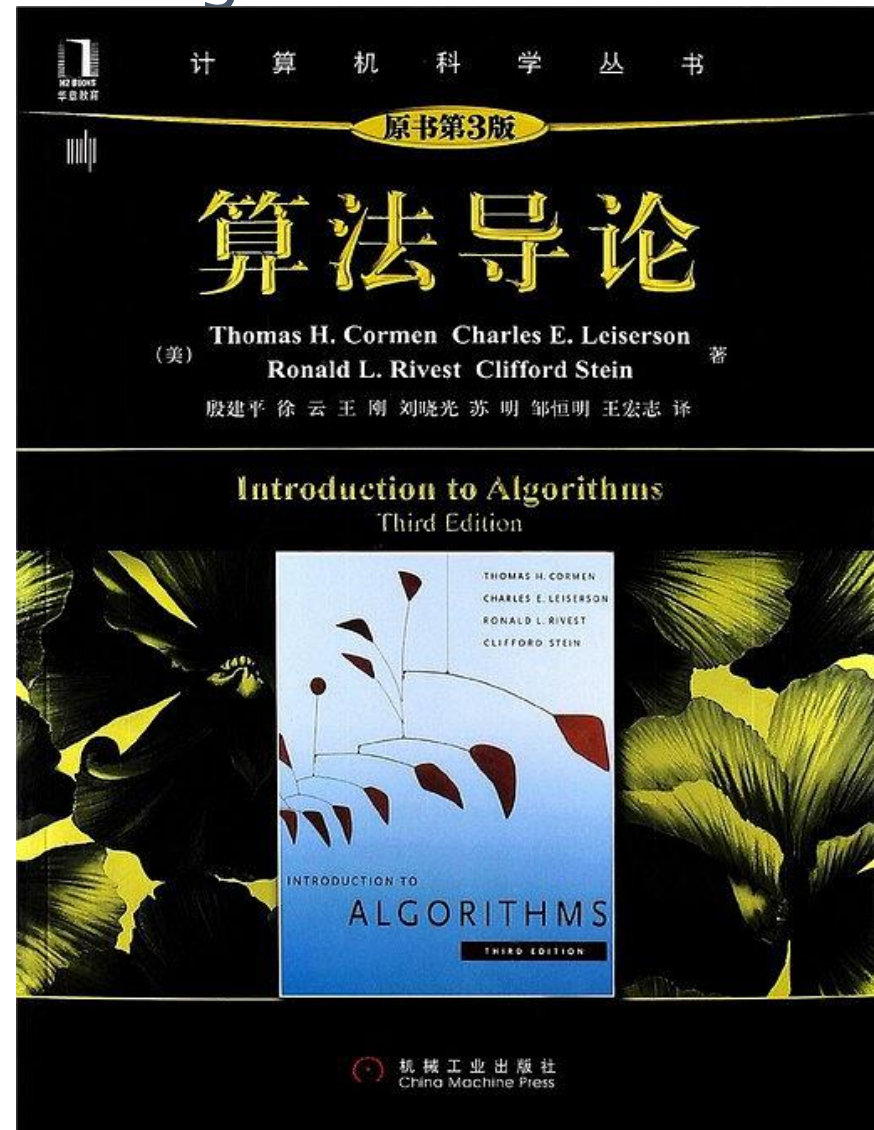
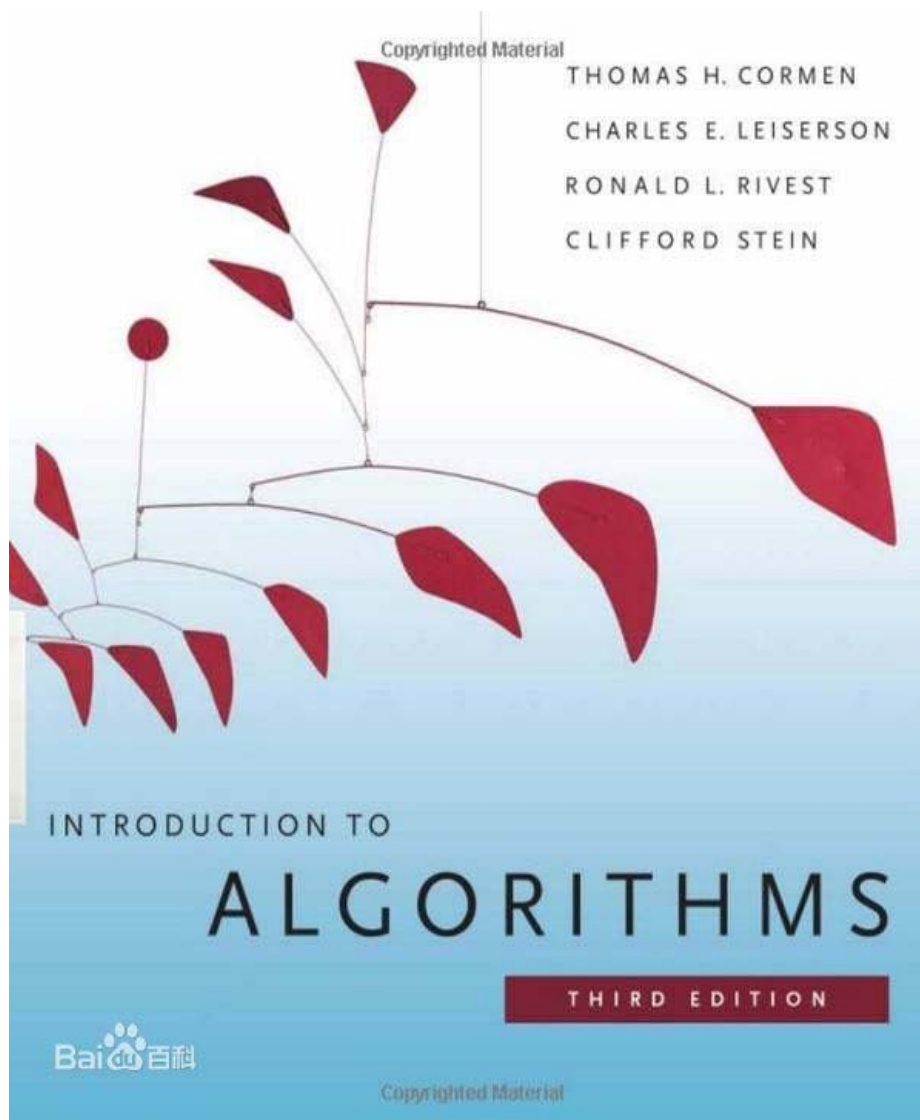


本课程主要内容介绍

算法设计与分析

Design and Analysis of Algorithms

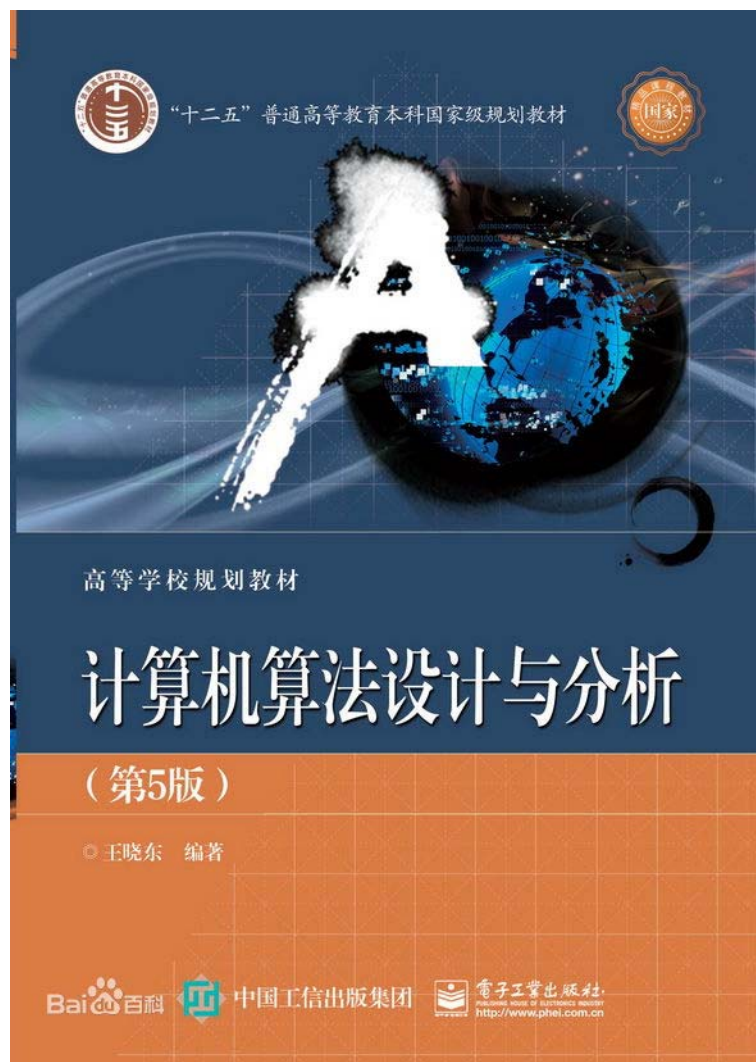
教材



算法设计与分析

Design and Analysis of Algorithms

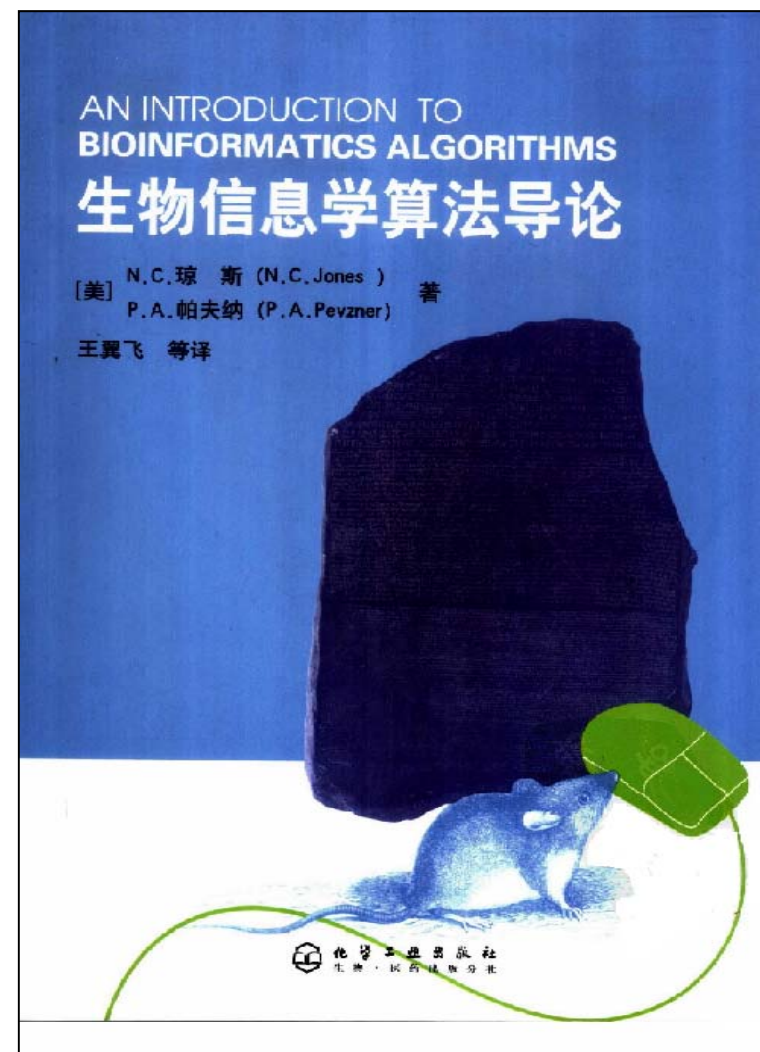
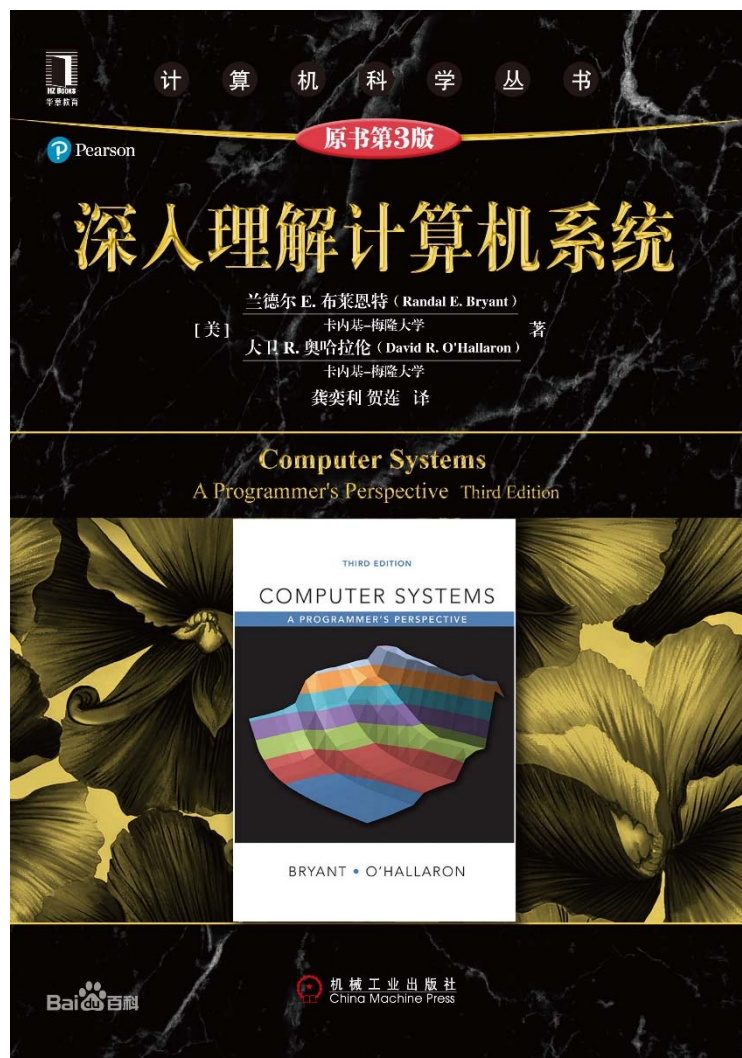
参考书



算法设计与分析

Design and Analysis of Algorithms

参考书



算法设计与分析

- 算法设计与分析理论简介
 - ◆ 70年代前，计算机科学基础的主题没有被清楚地认清。
 - ◆ 70年代，**Knuth**出版了《计算机程序设计技巧》（**The Art of Computer Programming**）（三卷），以各种算法研究为主线，确立了算法为计算机科学基础的重要主题，1974年获得图灵奖。
 - ◆ 70年代后，算法作为计算机科学的核心推动了计算机科学技术的飞速发展

算法设计与分析

计算机发展史简介

一、机械计算机时代的拓荒者

在西欧，由中世纪进入文艺复兴时期的社会大变革，大大促进了自然科学技术的发展，人们长期被神权压抑的创造力得到空前释放。其中制造一台能帮助人进行计算的机器，就是最耀眼的思想火花之一。

算法设计与分析

计算机发展史简介

一、机械计算机时代的拓荒者

- 1614: 苏格兰人**John** Napier (1550-1617)发表了一篇论文, 其中提到他发明了一种可以**计算四则运算和方根运算的精巧装置**。
- 1642: 法国数学家**Pascal** 在WILLIAM Oughtred计算尺的基础上将**计算尺加以改进, 能进行八位计算**。并且还卖出了许多, 成为一种时髦的商品。
- 1671: 德国数学家Gottfried **Leibniz**设计了一架可以进行乘法, 最终答案可以最大达到16位。
- 1848: 英国数学家George **Boole****创立二进制代数学**。提前差不多一个世纪为现代二进制计算机铺平了道路。

算法设计与分析

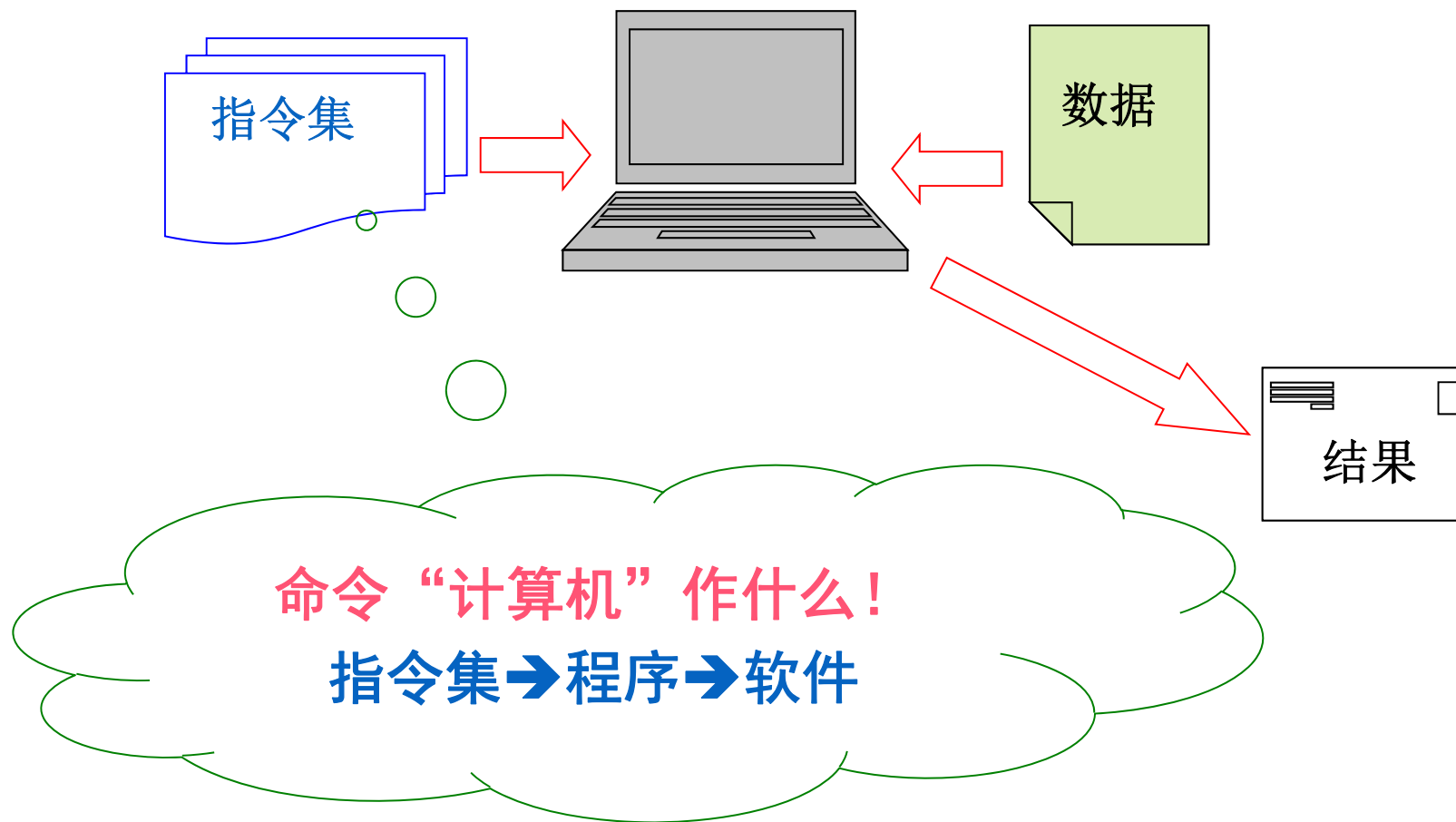
英国剑桥大学的图灵（Alan M. Turing）（1912-1954）；
在1937年，出版了他的论文，并提出了被后人称之为“图灵机”
的数学模型。

回答了什么叫计算的问题，即利用有限状态、状态改变、输入输出三种概念来定义了“计算”的概念，给出了“计算”的数学模型。

该计算模型的特点是利用符号序列描述了“计算”过程，这样人可以命令“计算机”干什么，机器能够理解人的命令成能可能。

图灵贡献在于，提出了独立于机器的指令集(程序)命令硬件机器完成任务的思想，并建立了相关的数学模型。

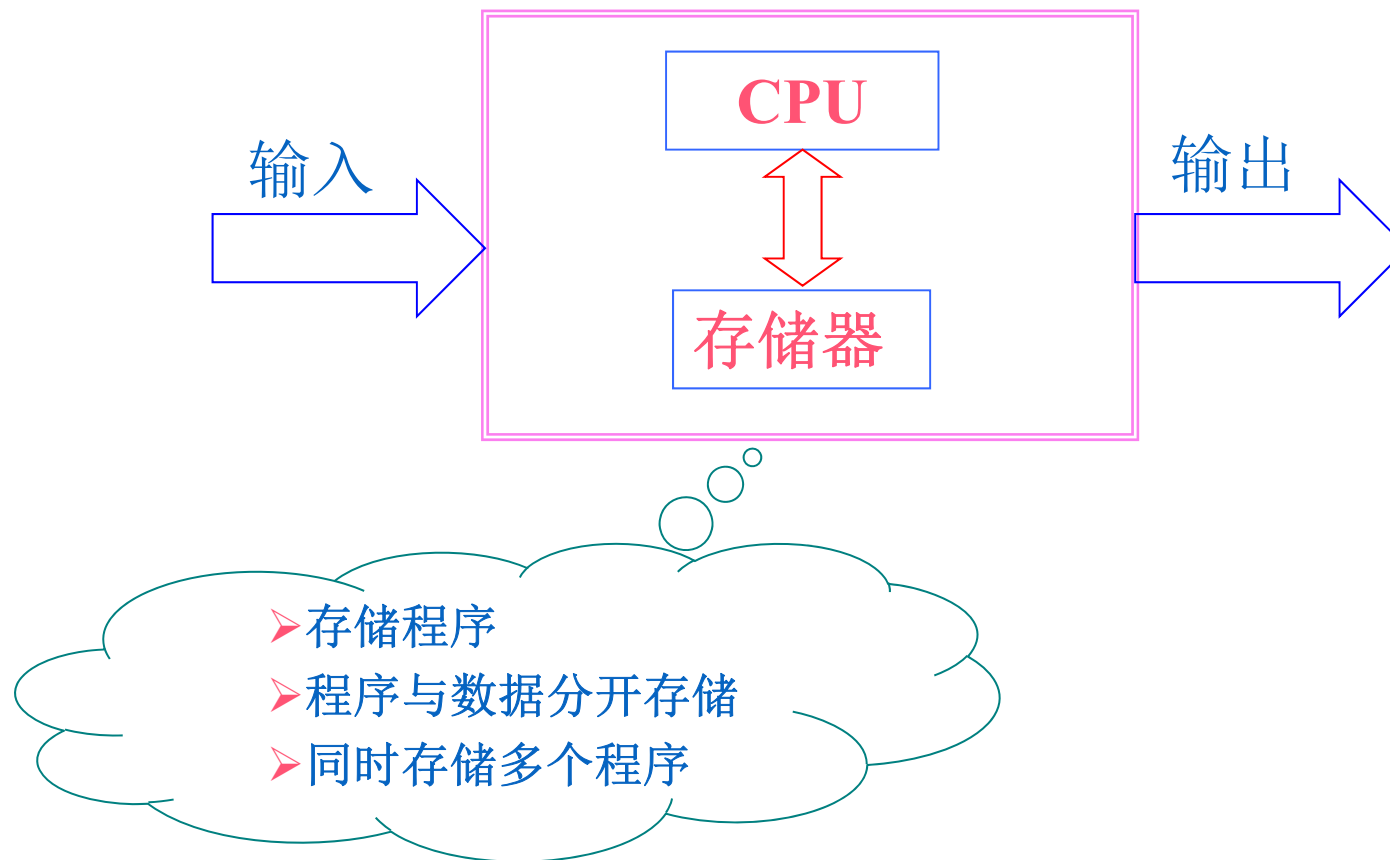
算法设计与分析



算法设计与分析

1949年，EDVAC (electronic discrete variable computer): 第一台使用磁带的计算机。这是一个突破，可以多次在其上存储程序。

算法设计与分析



算法设计与分析

- 决定一个软件性能的因素有很多，包括算法的设计与分析、算法的实现、软件系统结构、计算机系统等等，在其中算法设计与分析扮演了什么角色？
- 是否算法设计与分析是最重要的角色呢？
- 请发表自己的观点

计算机导论

——计算机系统技术

计算机硬件技术：

集成电路技术（微处理器，存储设配等等）

计算机组成技术（**PC**机，服务器，主板设计技术）

嵌入式系统设计技术（传感器、控制技术）

等等。。。。。

计算机导论

——计算机系统技术

计算机系统软件:

操作系统 (windows, Linux等等)

计算机网络 (传输协议, 加密解密技术, 网络与信息安全)

数据库系统 (Oracle, SQL Server, 等等)

等等。。。。。

计算机导论

——计算机应用技术

计算机软件开发工具:

应用软件开发语言 (汇编语言, C语言, Java语言, 等等)

数据库操作语言 (SQL语言, 等)

软件开发平台 (MS.NET, JavaEE 等等)

嵌入式软件开发平台 (手机软件开发平台等。。。)

云计算软件开发平台

网站设计平台

常用应用软件工具 (办公软件、压缩软件, 输入法) 等等。。。。。

计算机导论

——软件工程

什么叫软件工程：

- 软件工程是开发过程管理的方法，包括优化设计与开发软件，保证品质与开发进度，降低开发成本等等。
- 计算技术是“局部”的，软件工程是“全局”的解决方案。
- 大型软件的开发需要，复杂问题分解成相对独立的子问题，分而自治，并行开发，最后相互集成。
- 软件设计师应具有很强的分析问题能力，建模能力，问题分解能力，表述能力等等，我们国家急缺这类人才。
- 软件开发人员应具有熟练掌握相关的软件开发技术，应具有很强的工业规范意识，很强的团队合作精神。

计算机导论

——计算机科学

计算机科学研究内容：

可计算性理论（哪些问题可求解）

计算复杂性理论（问题的固有计算复杂性）

算法设计与计算复杂性（如何设计算法，算法的计算复杂性）

计算模型理论（自动机）

形式语言（编译原理，高级语言的理论基础）

人工智能（自适应能力）

机器推理与证明

等等。。。。。

算法设计与分析

- 计算复杂性理论研究内容
 - ◆问题的固有复杂性分析，即求解该问题的任何一个算法都不可逾越的计算复杂性
- 算法设计和分析研究内容
 - ◆可计算问题的形式化描述
 - ◆设计算法的理论、方法和技术
 - ◆分析算法的理论、方法和技术

算法设计与分析

• 计算复杂性理论

◆ 问题

不可求解问题
不可计算问题
可计算问题

无通用求解算法问题：

例如，求解丢翻图方程问题。

无最优求解算法问题：

选秘书问题： n 名应聘人中选择最优秀的1名，每天只面试一人，3天之内通知面试结果。

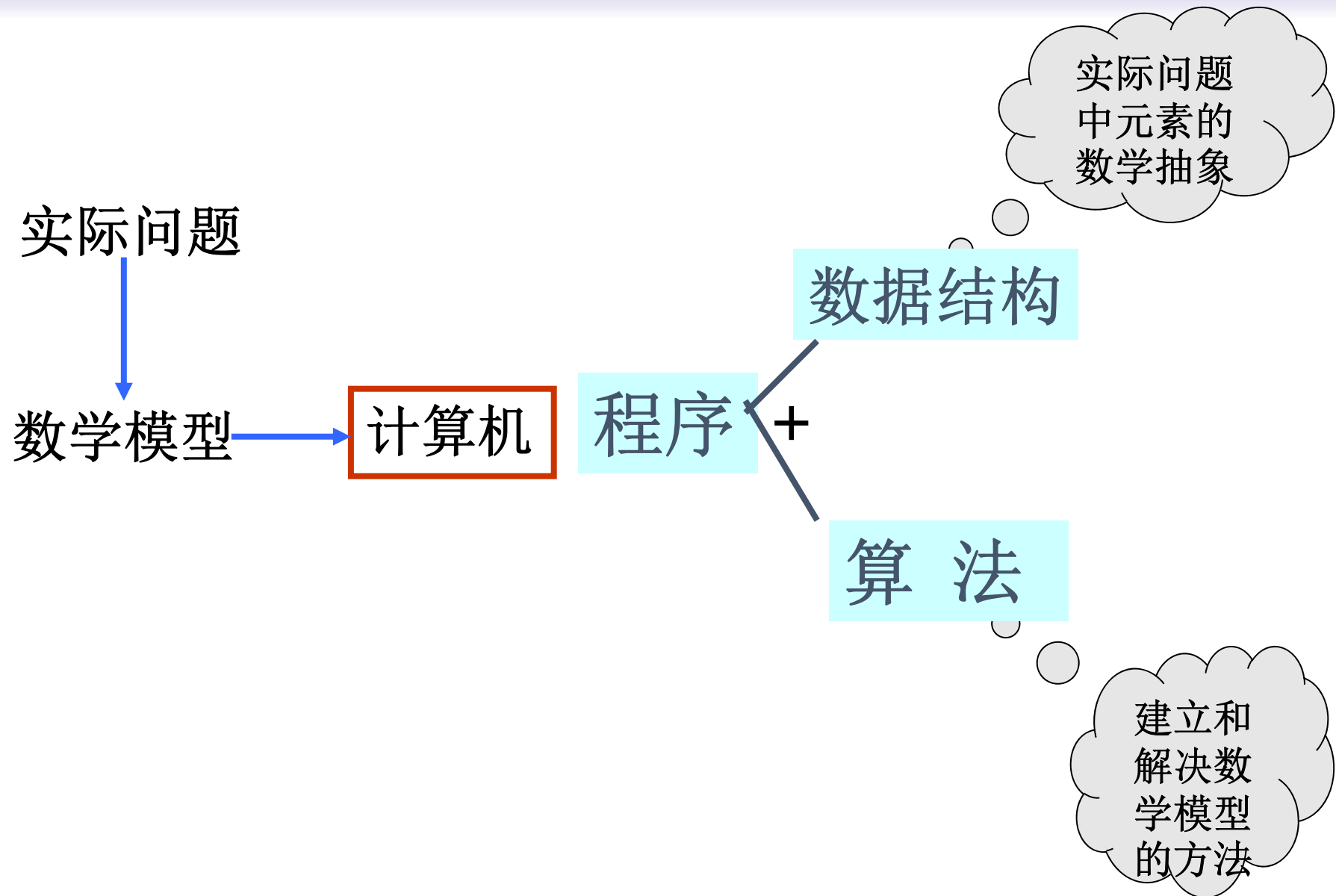
该问题不存在最优算法！！！！

多项式时间内可求解问题：
设计优化算法。

NP困难问题：

多项式计算复杂性问题，设计近似算法。

算法设计与分析



算法设计与分析

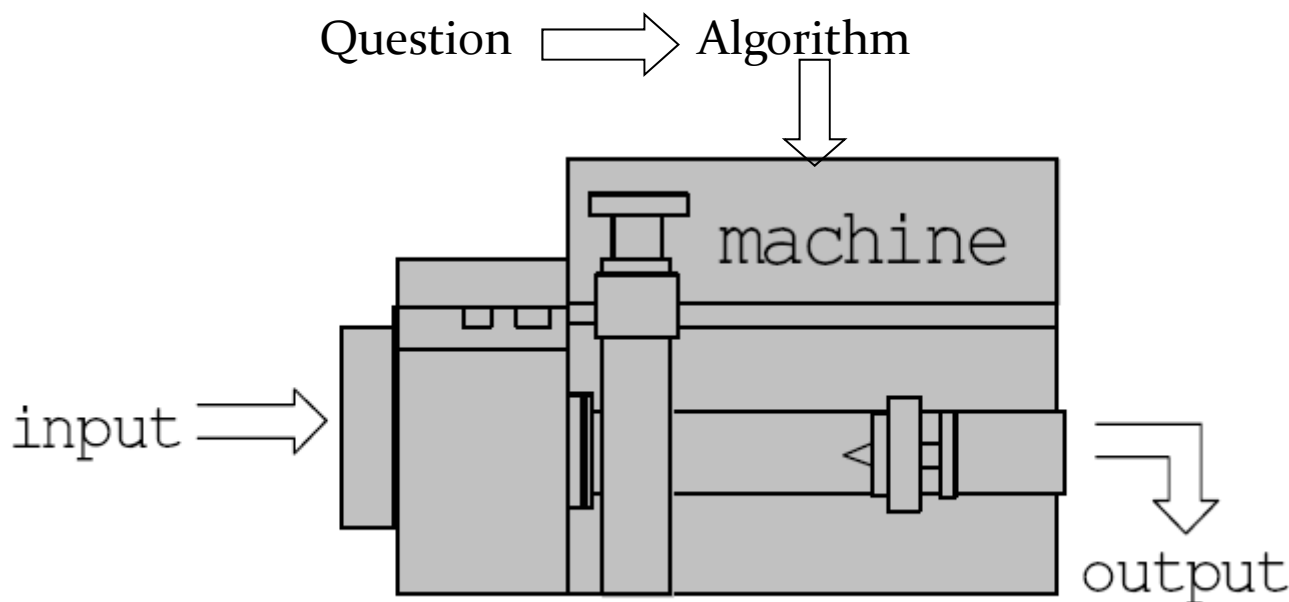
- 解决一个计算问题的过程
- 是否可求解-----→不可求解问题
- └─→是否可计算-----→ NP问题
- └─→算法设计与分析
- └─→用计算机语言实现算法
- └─→软件系统



注重实践和过程

算法设计与分析

- What's an Algorithm?
 - 算法是一系列解决问题的清晰指令，也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。



算法 Algorithm

算法是一个满足下列条件的计算过程：

- 确定性： 算法的每一步骤必须有确切的定义；
- 能行性： 算法原则上能够精确地运行，而且人们用笔和纸做有限次运算后即可完成。
- 输入： 一个算法有0个或多个输入，以刻画运算对象的初始情况，所谓0个输入是指算法本身定出了初始条件；
- 输出： 一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的；
- 有穷性： 一个算法必须保证执行有限步之后结束；

■ 计算过程

Example

- 求两个正整数 m, n 的最大公约数 $\gcd(m, n)$
- 欧几里得算法基于的方法是重复应用下列式子，直到 $m \bmod n = 0$
 - $\gcd(m, n) = \gcd(n, m \bmod n)$ 例子 $\gcd(36, 24)$

$\gcd(m, n)$ 的欧几里得算法

第一步：如果 $n=0$ ，返回 m 的值作为结果，结束；否则进入第二步

第二步：用 n 去除 m ，将余数赋给 r 。

第三步：将 n 的值赋给 m ，将 r 的值赋给 n ，回第一步。

算法 Euclid(m, n)

//使用欧几里得算法计算 $\gcd(m, n)$

//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

While $n \neq 0$ **do**

$r \leftarrow m \bmod n$

$m \leftarrow n$

$n \leftarrow r$

Return m

同一个算法有不同的表达方式

Example

$\gcd(m,n)$ 的连续整数检测算法

第一步：将 $\min\{m,n\}$ 的值赋给 t 。

第二步： m 除以 t ，如果余数为 o ，则进入第三步；否则，进入第四步。

第三步： n 除以 t ，如果余数为 o ，则返回 t 值作为结果；否则，进入第四步。

第四步：把 t 的值减 1 。返回第二步。

同一个问题有不同的解决方法

$\gcd(m,n)$ 的中学计算算法 $\gcd(36,24)$

第一步：找到 m 的所有质因数。 $36=2*2*3*3$

第二步：找到 n 的所有质因数 $24=2*2*2*3$

第三步：从第一步和第二步中求得的质因数分解式找出所有的公因数

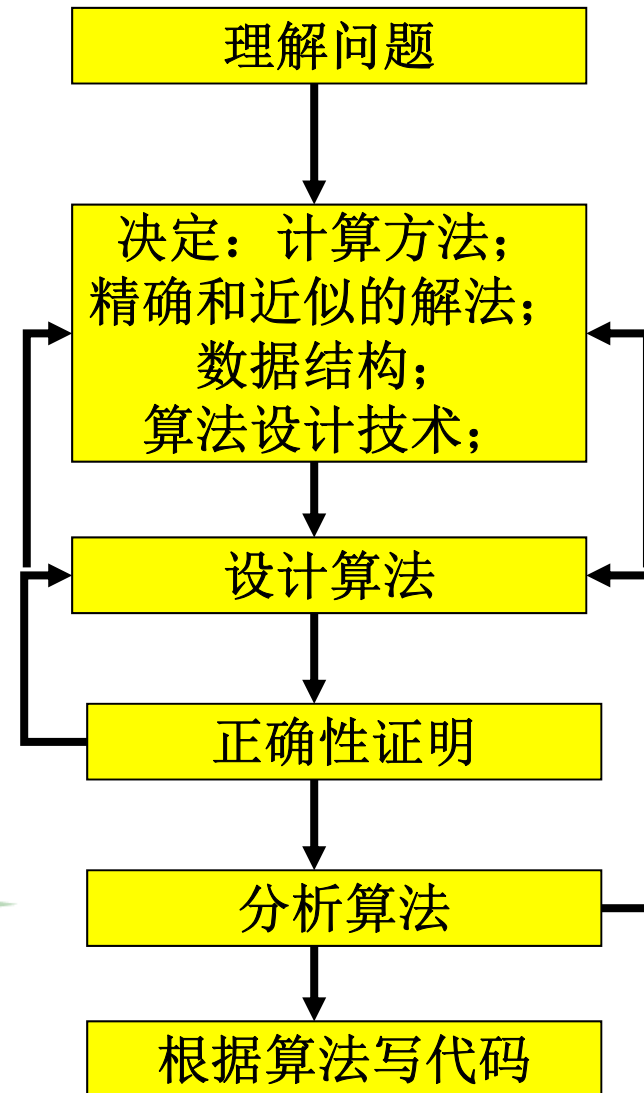
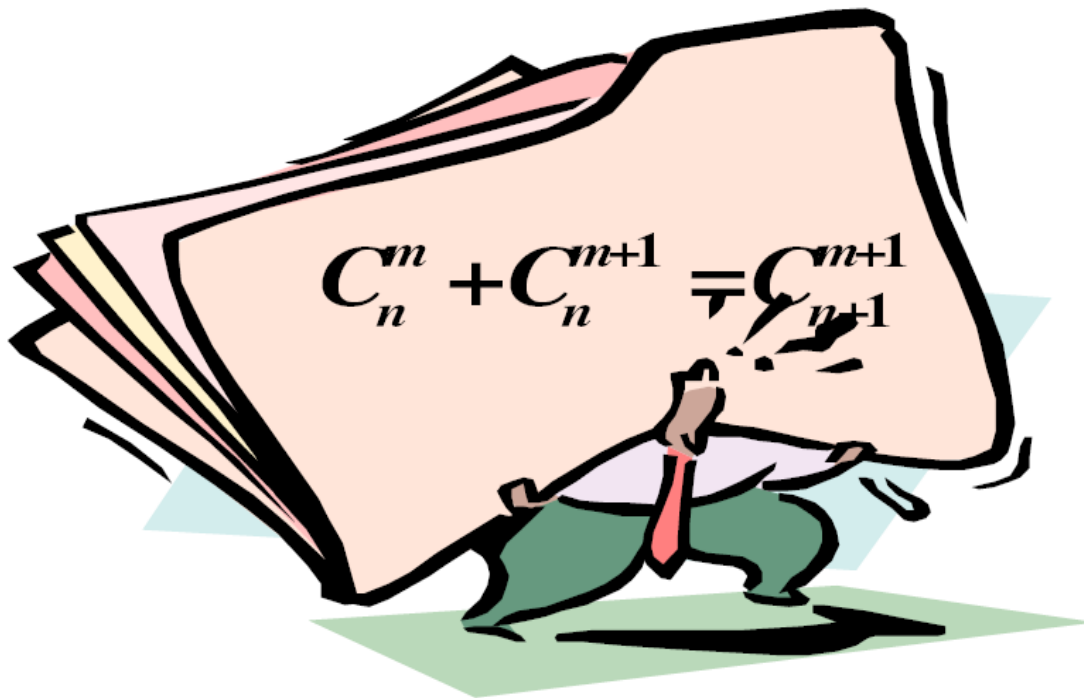
第四步：将第三步中找的质因数相乘，其结果作为给定数字的最大公因数。

算法的几个要点

- 算法的每一个步骤都必须清晰、明确。
- 算法所处理的输入的值域必须仔细定义。
- 同样的一个算法可以用几种不同的形式来描述。
- 可能存在几种解决相同问题的算法。
- 针对同一个问题的算法可能会基于完全不同的解题思路，而且解题的速度也会有明显区别。

算法问题求解基础

- 算法是问题的程序化解决方案。



算法问题求解基础

- 理解问题
 - 设计算法前做的第一件事情
 - 仔细阅读问题的描述
 - 提出疑问
 - 手工处理一些实例
 - 考虑特殊情况
 - 确定输入
 - **抽象出问题，用数学表达式描述**

算法问题求解基础

- 了解计算设备的性能
 - 确定计算方法
 - RAM (*Random-access memory*) 结构下的顺序算法
 - 并行算法
- 选择精确解和近似解
 - 某些重要的问题无法求得精确解
 - 某些问题利用精确解速度慢，无法接受

例：26个英文字母的全排列。

$$26! = 4 \times 10^{26}$$

$$365 \times 24 \times 3600 = 3.1536 \times 10^7 \text{秒/年}$$

$$4 \times 10^{26} / (3.1536 \times 10^7 \times 10^7) = 1.2 \times 10^{12} \text{年}$$

算法问题求解基础

- 确定适当的数据结构
 - 算法 + 数据结构 = 程序
- 算法设计技术
 - 使用算法解题的一般性方法，用于解决计算领域的多种问题。
- 详细表述算法的方法
 - 自然语言
 - 伪代码
 - 流程图

Example

- 求两个正整数 m, n 的最大公约数 $\gcd(m, n)$

$\gcd(m, n)$ 的欧几里得算法

第一步：如果 $n=0$ ，返回 m 的值作为结果，结束；否则进入第二步

第二步：用 n 去除 m ，将余数赋给 r 。

第三步：将 n 的值赋给 m ，将 r 的值赋给 n ，回第一步。

同一个算法有不同的表达方式

算法 Euclid(m, n)

//使用欧几里得算法计算 $\gcd(m, n)$
//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

While $n \neq 0$ **do**

$r \leftarrow m \bmod n$

$m \leftarrow n$

$n \leftarrow r$

Return m

算法问题求解基础

◆证明算法的正确性

◆证明对于每一个合法的输入，该算法都会在有限的时间内输出一个满足要求的结果。

◆一般方法：数学归纳法

◆证明算法的正确性与不正确哪一个更容易？

◆分析算法

◆算法有两种效率：时间效率和空间效率

◆算法的另外两个特性：简单性和一般性

Example查找

考虑问题1：已知不重复且已经按从小到大排好的 m 个整数的数组 $A[1..m]$ （为简单起见。还设 $m=2^k$ ， k 是一个确定的非负整数）。对于给定的整数 c ，要求寻找一个下标 i ，使得 $A[i]=c$ ；若找不到，则返回一个0。

算法1：从头到尾扫描数组A

照此，或者扫到A的第i个分量，经检测满足 $A[i]=c$ ；或者扫到A的最后一个分量，经检测仍不满足 $A[i]=c$ 。我们用一个函数Search来表达这个算法：

Function Search (c:integer):integer;

Var J:integer;

Begin

J:=1; {初始化} {在还没有到达A的最后一个分量且等于c的分量还没有找到时，查找下一个分量并且进行检测}

While ($A[j]<c$)and($j<m$) do

 j:=j+1;

If $A[j]=c$ then

 search:=j {在数组A中找到等于c的分量，且此分量的下标为j}

 else Search:=0; {在数组中找不到等于c的分量}

End;

容易看出，在最坏的情况下，这个算法要检测A的所有m个分量才能判断在A中找不到等于c的分量。

蛮力法

算法2 二分查找算法

利用到已知条件中**A已排序**的性质。它可以用函数B_Search来表达:

```
Function B_Search ( c: integer):integer;  
Var L,U,I : integer; {U和L分别是要查找的数组的下标的上界和下界}  
Found: boolean;  
Begin L:=1; U:=m; {初始化数组下标的上下界}  
Found:=false; {当前要查找的范围是A[L]..A[U]。} {当等于c的分量还没有找到  
且U>=L时, 继续查找}  
While (not Found) and (U>=L) do  
    Begin I:=(U+L) div 2; {找数组的中间分量}  
        If c=A[I] then Found:=True  
        else if c>A[I] then L:=I+1 else U:=I-1;  
    End;  
    If Found then B_Search:=1  
    else B_Search:=0;  
End;
```

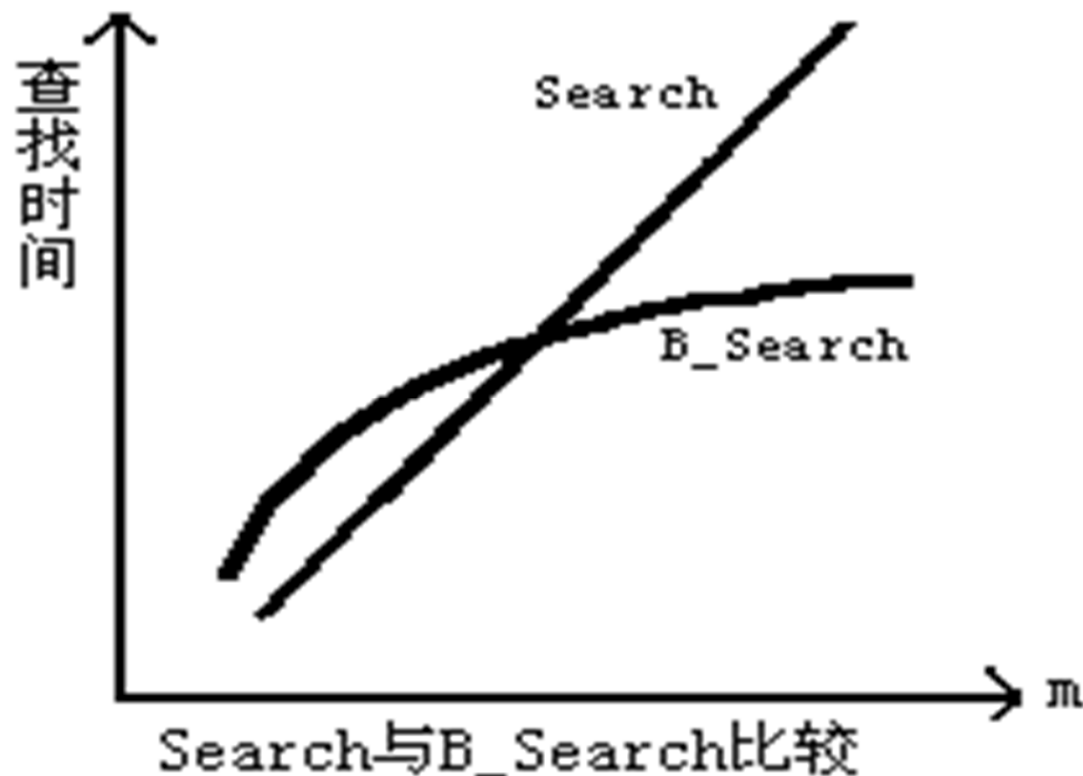
容易理解, 在最坏的情况下最多只要测A中的 $k+1$ ($k=\log m$, 这里的log以2为底, 下同) 个分量, 就判断c是否在A中。

举例: 1,2,3,4,5,6,7,8找 2

算法分析

算法Search和B_Search解决的是同一个问题，但在最坏的情况下（所给定的 c 不在 A 中），两个算法所需要检测的分量个数却大不相同，前者要 $m=2^k$ 个，后者只要 $k+1$ 个。

可见算法B_Search比算法Search高效得多。

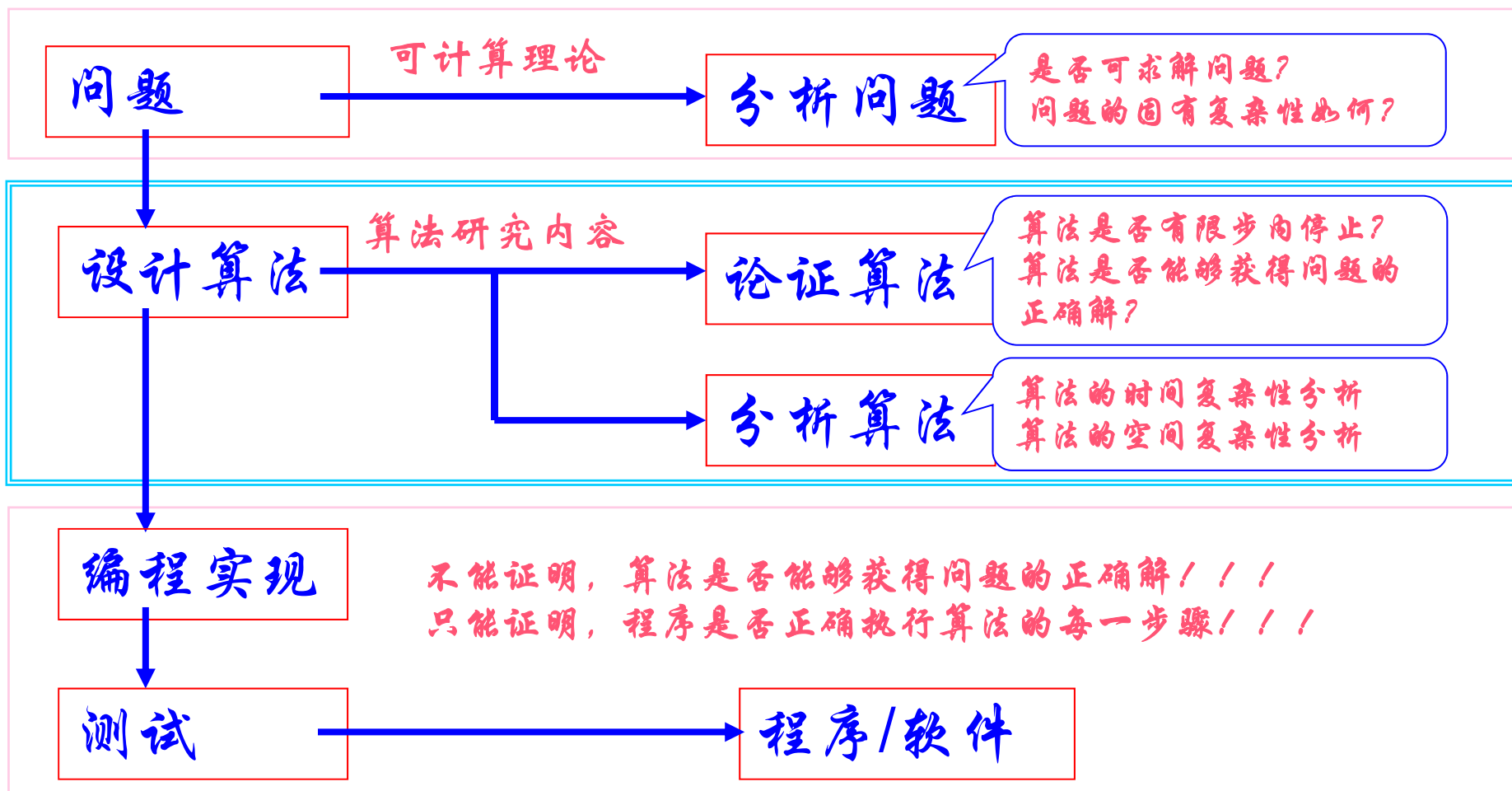


图是运行这两种算法的时间曲线。该图表明，当 m 适当大（ $m > m_0$ ）时，算法B_Search比算法Search省时，而且当 m 更大时，节省的时间急剧增加。

以上例子说明：解同一个问题，算法不同，则计算的工作量也不同，所需的计算时间随之不同，即复杂性不同。

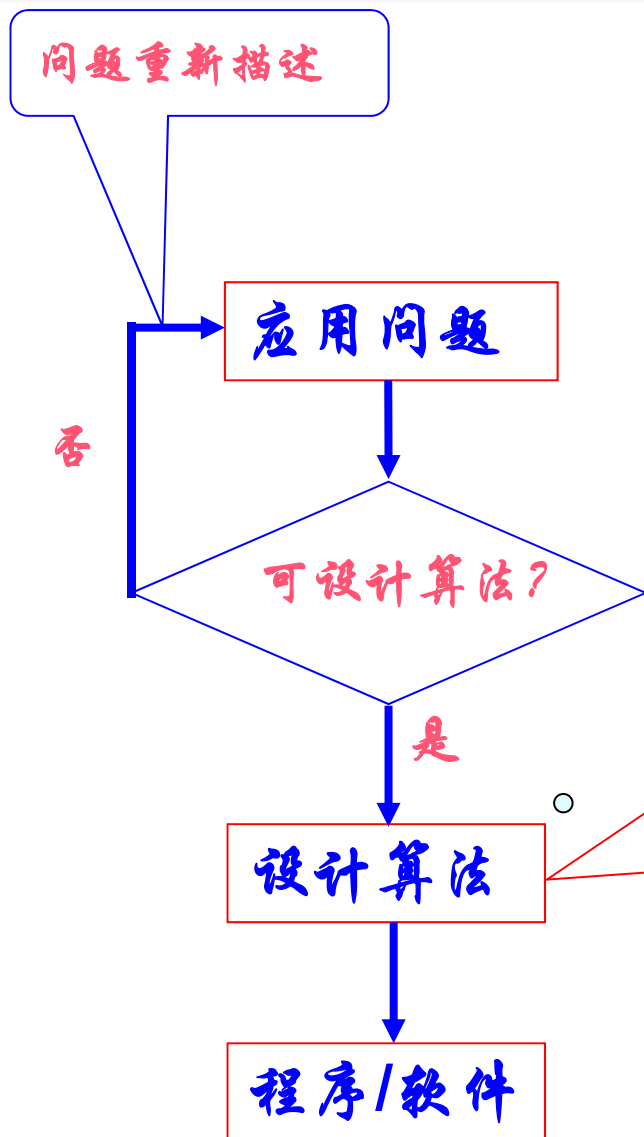
算法设计与分析

算法与程序——问题求解过程



算法设计与分析

算法与程序—算法评价



什么叫好算法?

- ✓ 算法在有限步内停机!
- ✓ 算法能够获得问题的正确解!
- ✓ 时间复杂性: 多项式复杂性? 指数复杂性?
- ✓ 空间复杂性: 线性增长? 多项式增长?
- ✓ 容易理解? 容易实现?

算法设计与分析

算法设计与分析——主要研究内容

- ✓ 算法在有限步内停机!
- ✓ 算法能够获得问题的正确解!
- ✓ 时间复杂性: 多项式复杂性? 指数复杂性?
- ✓ 空间复杂性: 线性增长? 多项式是增长?
- ✓ 容易理解? 容易实现?

算法设计与分析

算法设计与分析——评价好算法的标准

- ✓ 时间复杂性：多项式复杂性？指数复杂性？
- ✓ 空间复杂性：线性增长？多项式增长？

算法复杂性 complexity

算法的复杂性是算法效率的度量，是评价算法优劣的重要依据。

一个算法的复杂性的高低体现在运行该算法所需要的计算机资源的多少上面，所需的资源越多，我们就说该算法的复杂性越高；

反之，所需的资源越低，则该算法的复杂性越低。

计算机的资源，最重要的是时间和空间（即存储器）资源。因而，算法的复杂性有

时间复杂性

空间复杂性

算法复杂性 complexity

不言而喻，对于任意给定的问题，设计出复杂性尽可能低的算法是我们在设计算法是追求的一个重要目标；另一方面，当给定的问题已有多种算法时，选择其中复杂性最低者，是我们在运用算法适应遵循的一个重要准则。因此，算法的复杂性分析对算法的设计或运用有着重要的指导意义和实用价值。

关于算法的复杂性，有两个问题要弄清楚：

1. 用怎样的一个量来表达一个算法的复杂性；
2. 对于给定的一个算法，怎样具体计算它的复杂性。

算法复杂性分析

计算机类型 “通用”

- 运算分类

- 基本运算

- 由一些更基本运算的任意长序列所组成

- 确定能反映出算法在各种情况下工作的数据集

- 算法分析分两个阶段

- 事前分析

- 事后测试

乘法和加法运算是基本运算

一般说，基本运算的数量和运算的时间成正比

例——基本运算

◆ 求n次多项式的值

$$\begin{aligned} P_n(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \\ &= \sum_{i=0}^n a_i x^i \end{aligned}$$

$$P_n(x) = (\dots ((a_n x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0$$

如： $n = 3$

$$\begin{aligned} &a_3 x^3 + a_2 x^2 + a_1 x^1 + a_0 \\ &= x(a_3 x^2 + a_2 x^1 + a_1) + a_0 \\ &= x(x(a_3 x^1 + a_2) + a_1) + a_0 \end{aligned}$$

例——基本运算

Horner Algorithm:

Step1 $i = n; p = 0.$

Step2 $p = p * x + a_i; i = i - 1.$

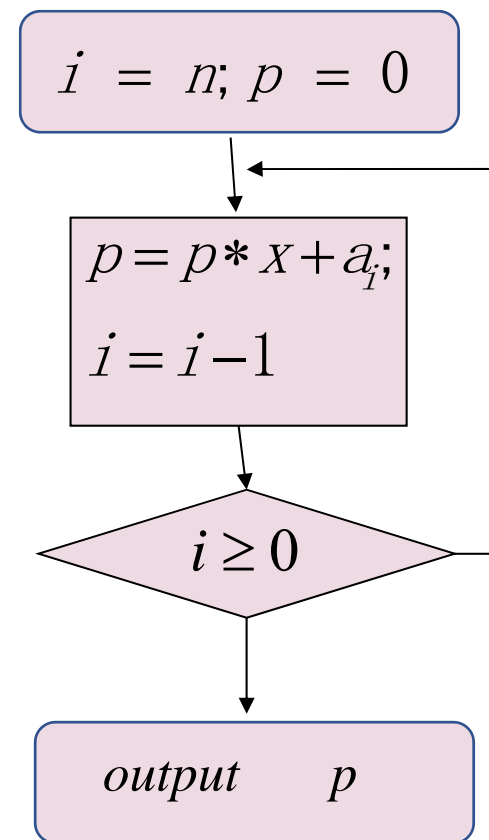
Step3 *if* $i \geq 0$; *then* *goto* *Step2*

Step4 *else* *output* p

分析

乘法和加法运算是基本运算

一般说，基本运算的数量和运算的时间成正比



算法的事前估计

- 空间复杂度:单位由1增加到 n , $f(n)$
- 时间复杂度:单位由1增加到 n , $g(n)$

Or 算法的后期测试

- 在算法中的某些部位插装时间函数 $\text{time}()$
- 测定算法完成某一功能所花费的时间

空间复杂度度量

◆ 存储空间的固定部分

程序指令代码的空间，常数、简单变量、定长成分(如数组元素、结构成分、对象的数据成员等)变量所占的空间

◆ 可变部分

尺寸与实例特性有关的成分变量所占空间、引用变量所占空间、递归栈所用的空间、通过new和delete命令动态使用的空间

时间复杂度度量

◆编译时间

◆运行时间

◆程序步

- ◆语法上或语义上有意义的一段指令序列


- ◆执行时间与实例特性无关

- ◆例如：

注释：程序步数为0

声明语句：程序步数为0

表达式：程序步数为1



算法问题求解基础

程序步确定方法

- 插入计数全局变量 *count*
 - ◆ 建表，列出个语句的程序步

例 以迭代方式求累加和的函数

$$S = \sum a_i$$

```
行  float sum ( float a[ ], const int n )  
1   {  
2       float s = 0.0;  
3       for ( int i=0; i<n; i++ )  
4           s += a[i];  
5       return s;  
6   }
```

在求累加和程序中加入 $count$ 语句

```
float sum ( float a[ ], const int n ) {  
    float s = 0.0;  
    count++; //count统计执行语句条数  
    for ( int i=0; i<n; i++ ) {  
        count++; //针对for语句  
        s += a[i];  
        count++;  
    } //针对赋值语句  
    count++; //针对for的最后一次  
    count++; //针对return语句  
    return s;  
} 执行结束得 程序步数 $count = 2 * n + 3$ 
```

程序的简化形式

```
void sum ( float a[ ], const int n )  
{  
    for ( int i=0; i<n; i++ )  
        count += 2;  
    count += 3;  
}
```

注意：

一个语句本身的程序步数可能不等于该语句一次执行所具有的程序步数。

例如：赋值语句

$x = \text{sum}(R, n);$

本身的程序步数为1;

一次执行对函数 $\text{sum}(R, n)$ 的调用需要的程序步数为 $2*n+3$;

一次执行的程序步数为

$$1+2*n+3 = 2*n+4$$

累加和程序

程序步数计算工作表格

程 序 语 句	一次执行所需程序步数	执行频度	程序步数
{	0	1	0
float $s = 0.0;$	1	1	1
for (int $i=0; i<n; i++$)	1	$n+1$	$n+1$
$s += a[i];$	1	n	n
return $s;$	1	1	1
}	0	1	0
	总程序步数		$2n+3$

渐进时间复杂度: $T(n)=O(f(n))$

算法问题求解基础

- 为算法写代码
 - 用计算机程序实现算法
 - 在把算法转变为程序的过程中，可能会发生错误或者效率非常低

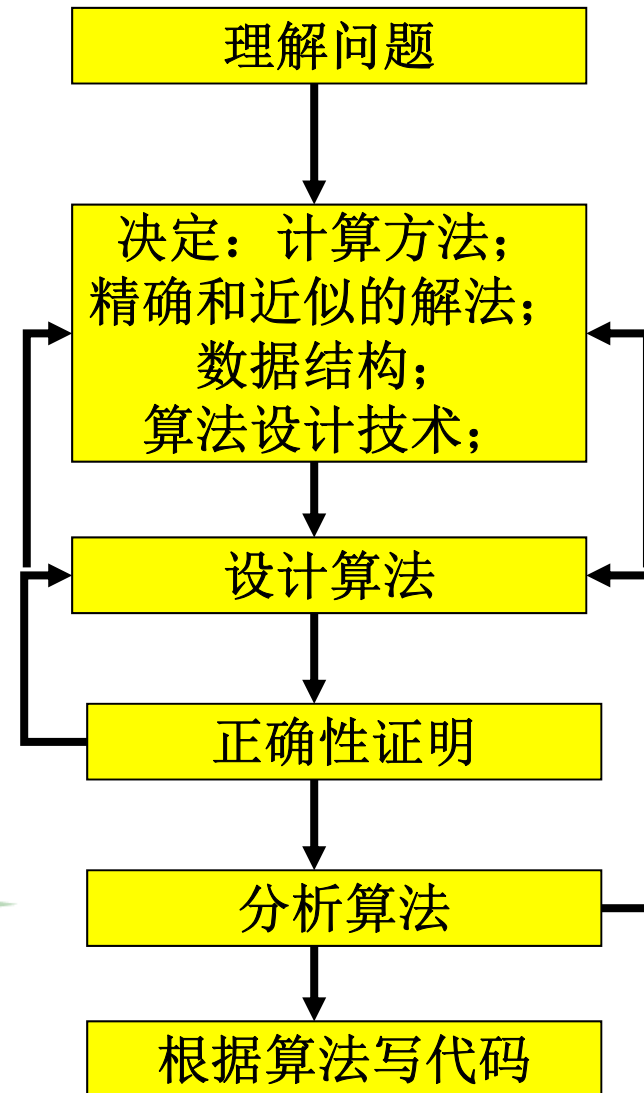
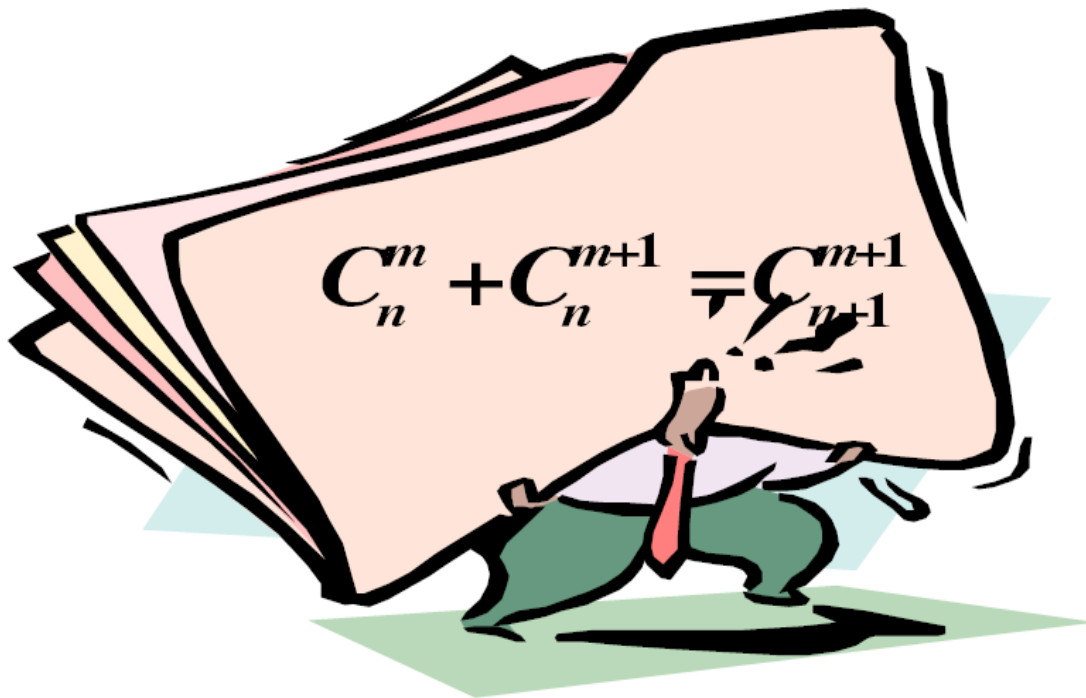
作为一种规律，一个好的算法是反复努力和重新修正的结果

- 算法是一个最优性问题：对于给定的问题需要花费多少力气（资源）？
- 是不是每个问题都能够用算法的方法来解决？

发明或者发现算法是一个非常有创造性和非常值得付出的过程！

算法问题求解基础

- 算法是问题的程序化解决方案。



重要的问题类型

◆ 排序(Sorting)

◆ 查找(Search)

◆ 串处理(String)

◆ 图问题(Graph)

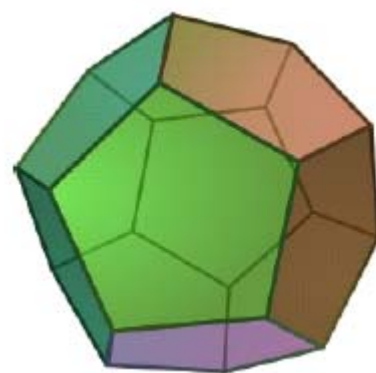
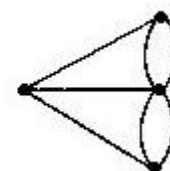
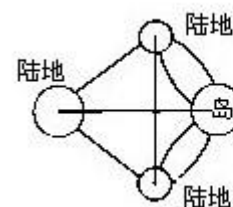
◆ 组合问题(Combination)

◆ 几何问题(Geometry)

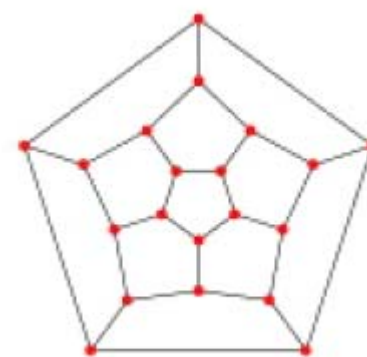
◆ 数值问题



柯尼斯堡七桥问题 欧拉回路



Icosian游戏



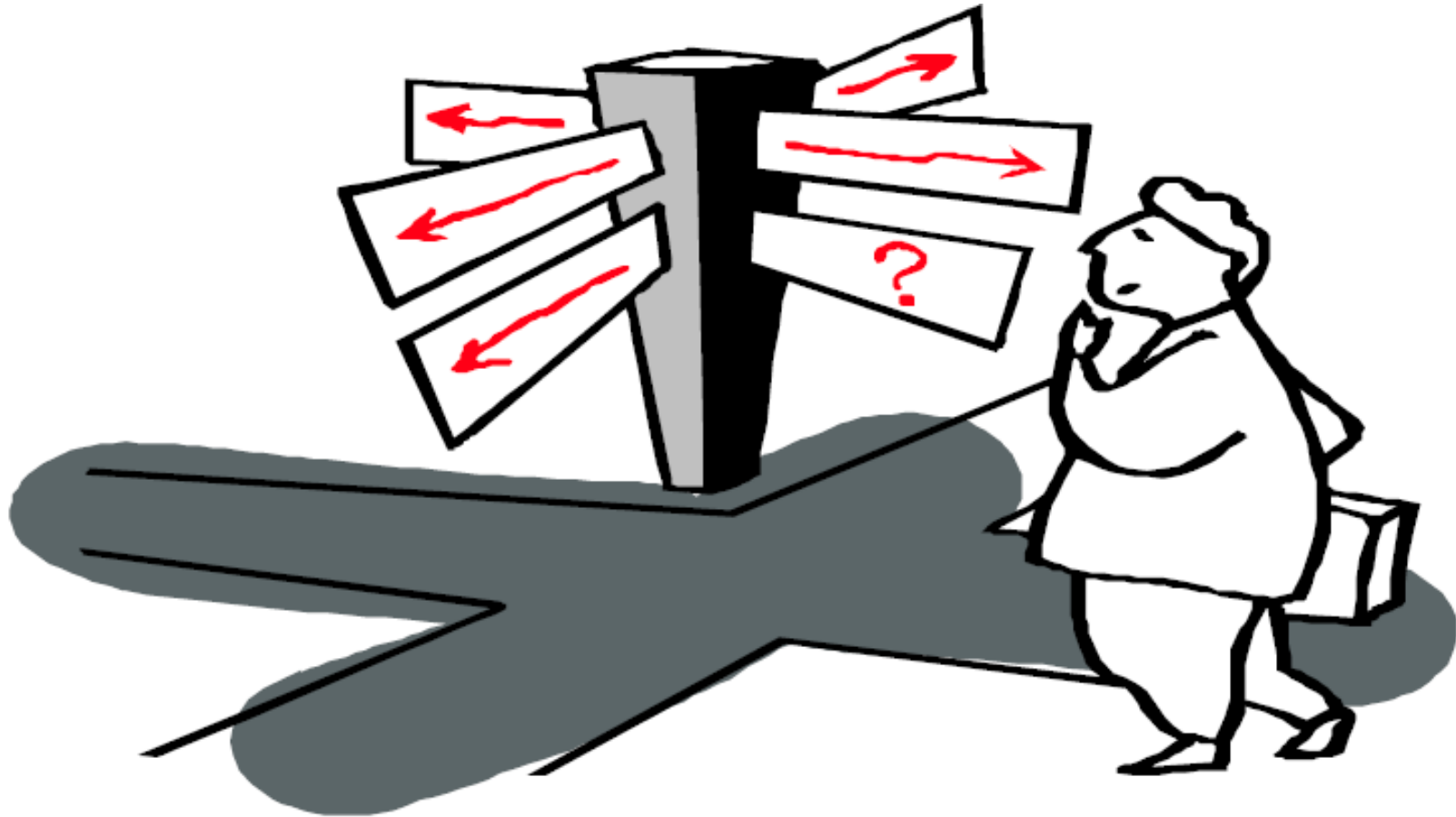
曼哈顿回路

基本数据结构

- 线性数据结构
 - 数组，串，单（双）链表，堆、栈，队列
- 图
 - 有向图，无向图，加权图
- 树
- 集合与字典

算法设计与分析

How to Study Algorithm?



"Sometimes we have experiences, and sometimes not.
Therefore, the better way is to learn more."

算法设计与分析

 中国大学MOOC

课程 名校 2020考研 学校云 名师专栏 新

 哈尔滨工业大学
HARBIN INSTITUTE OF TECHNOLOGY

算法设计与分析入门 申请认证证书
王宏志



公告

评分标准

课件

测验与作业

考试

讨论区

课件 > 第一讲 算法概述 > 算法概述

 哈尔滨工业大学 海量数据计算研究中心
Massive Data Computing Lab @ HIT

算法设计与分析—入门篇

第一讲 算法概述

哈尔滨工业大学
王宏志

算法设计与分析



屈婉玲教授

算法设计与分析

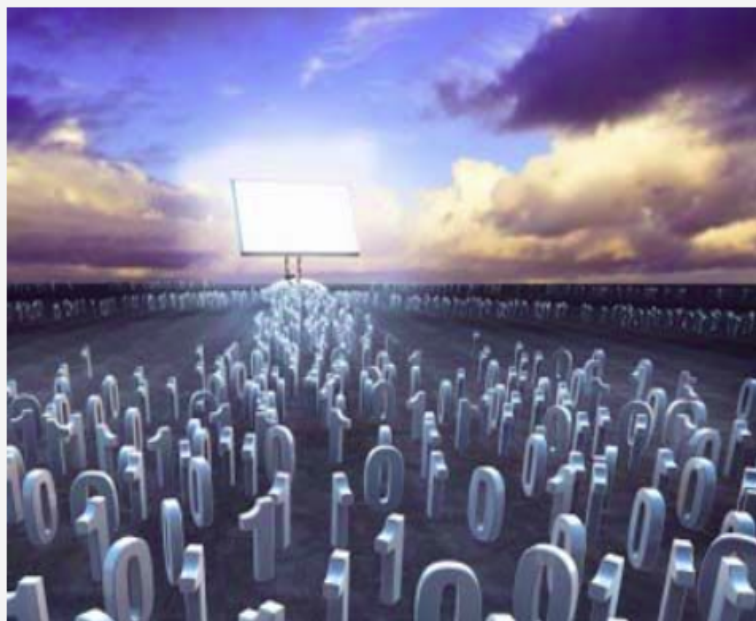
网易公开课

搜索课程、视频、策划



欢迎来到网易公开课! [登录/注册](#) [我的公开课](#) [+关注我们](#) [客户端下载](#)

国际名校公开课 > 麻省理工学院公开课: 算法导论



麻省理工学院公开课: 算法导论

本课程共23集 翻译完 欢迎学习

课程介绍

课程教授高效率算法的设计及分析技巧，并着重在有实用价值的方法上。课程主题包含了：排序、堆积及散列；各个击破法、动态规划、网络流、计算几何、数字理论性算法、高速缓存技术及并行运算等。

字幕由人人字幕组和人人oCourse字幕组提供，网易仅转载并保留全部翻译版权信息，由衷感谢他们的贡献。

分享

☆ 收藏



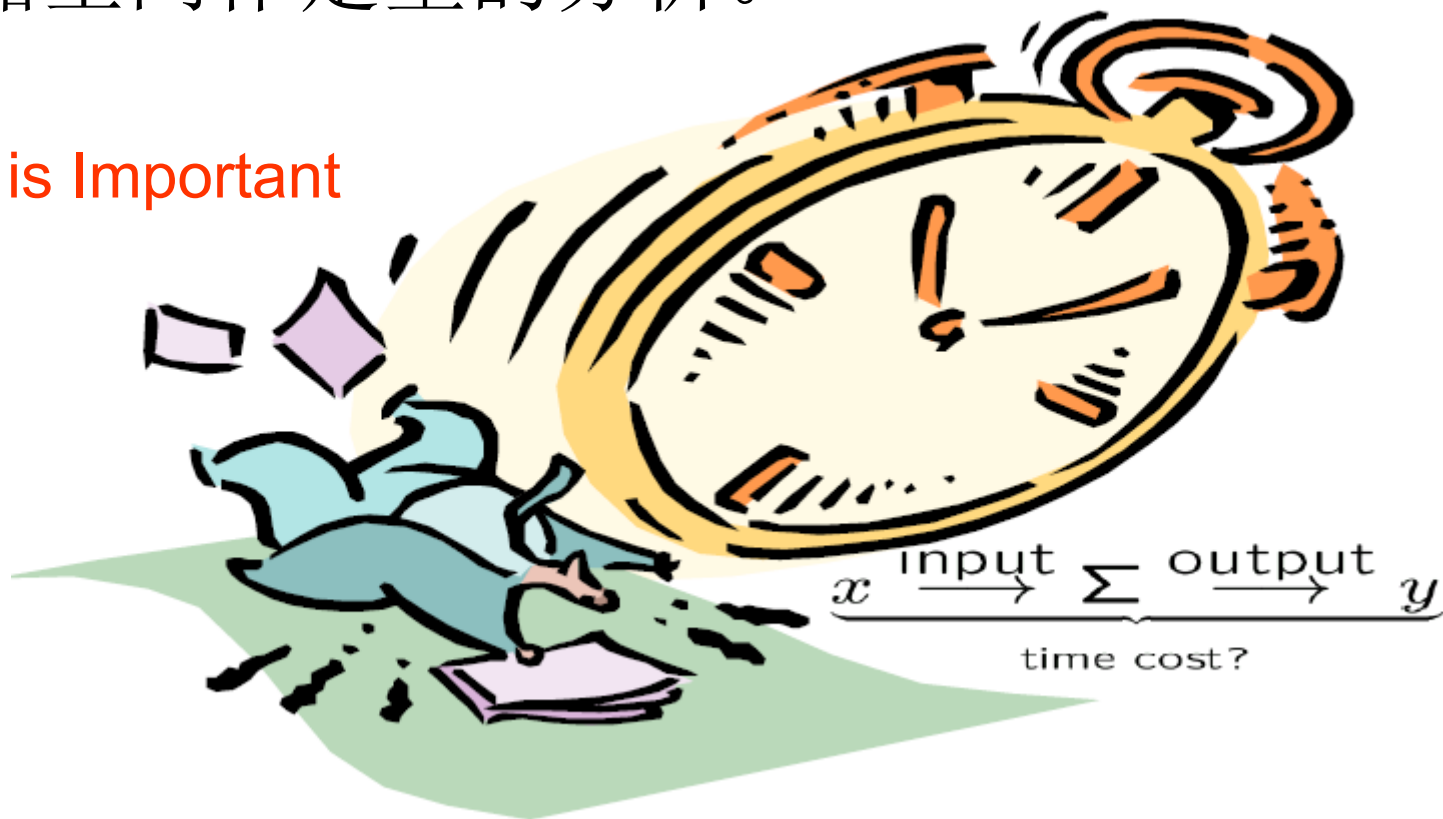
小结

- ◆ “算法”是在有限时间内，对问题求解的一个清晰的指令序列。
- ◆ 算法具有5个特点。
- ◆ 算法常用自然语言或伪代码描述。
- ◆ 对算法的分类主要有两种方法：
 - ◆ 按照求解问题的类型对算法分类
 - ◆ 按照其内在的设计技术对算法分类
- ◆ 重要的问题类型：排序、查找、串处理、图问题、组合问题、几何问题和数值问题。
- ◆ “算法设计技术”是用算法解题的一般性方法，适用于解决不同计算领域的多种问题。
- ◆ 一个好的算法常常是不懈努力和反复修正的结果。
- ◆ 解决同一个问题的算法常常有好几种。

下一节：算法效率分析基础

- 算法分析是对一个算法需要多少计算时间和存储空间作定量的分析。

Time is Important



不是所有能计算的都有价值，不是所有有价值的都能被计算 ——阿尔伯特·爱因斯坦