算法设计与分析

蒋婷婷

课程信息

- 上课时间和地点：
  - 大班课：每周一**3-4**节，每周三**5-6**节，一教**201**
  - 小班课：每周五**5-6**节，小班助教上课，分班名单和上课地点周四会在教学网公布

- 授课老师：蒋婷婷

  单位：计算机学院数字媒体所

  研究方向：计算机视觉、图像视频质量评价

  电子邮件：**ttjiang@pku.edu.cn**

  个人主页：**http://www.vie.group/ttj**

- 助教：

  史若画**: ruohuashi@pku.edu.cn**

计算思维与人才培养

- 2006年3月周以真(Jeannette M. Wing，卡内基·梅隆大学计算机系系主任)首次提出Computational Thinking的概念：运用计算机科学的基础概念去求解问题、设计系统和理解人类的行为，它包括了涵盖计算机科学之广度的一系列思维活动。

数学思维与工程思维的互补与融合：抽象与实现

编程	⇒	评价程序 优化程序	⇒	问题分析 方法确定
技能：会做		能力：做得好		思维：认知、方法论

实验思维、理论思维、计算思维

- 三种思维的共同特点：
  用语言文字表达、有语法与语义规则、推理逻辑

	实验思维	理论思维	计算思维
起源	物理学	数学	计算机科学
过程步骤	1. 实验观察归纳建立简单数学公式 2. 导出数量关系 3. 实验验证	1. 定义概念 2. 提出定理 3. 给出证明	1. 建模(约简、嵌入、转化、仿真、…) 2. 抽象与分解,控制系统复杂性 3. 自动化实现…
特点	解释以往现象 无矛盾 预见新的现象	公理集 可靠协调推演规则 正确性依赖于公理	结论表示有限性 语义确定性 实现机械性

算法与计算思维

- 算法课程是训练计算思维的重要课程；涉及到对问题的抽象，建模，设计好的求解方法，复杂性的控制，...
- 可计算性与计算复杂性： 形式化、确定性、有限性，抽象与逻辑证明
- 算法设计与分析：抽象建模、归约、正确性证明、效率分析、…

课程简介

- 课程名称

  算法设计与分析

- 基本内容
  - 组合算法设计的基本技术
  - 算法分析的基本方法
  - 计算复杂性理论的基本概念
  - 用算法理论处理实际问题
  - 学科的新进展

课程内容

近似算法	NP 完全理论简介	随机算法	在线算法	问题处理策略 计算复杂性理论
算法分析与问题的计算复杂性				算法分析方法
分治策略	动态规划	贪心算法	回溯与分支限界	算法设计技术
数学基础、数据结构				基础知识

教材

书名：

《算法设计与分析》

作者：

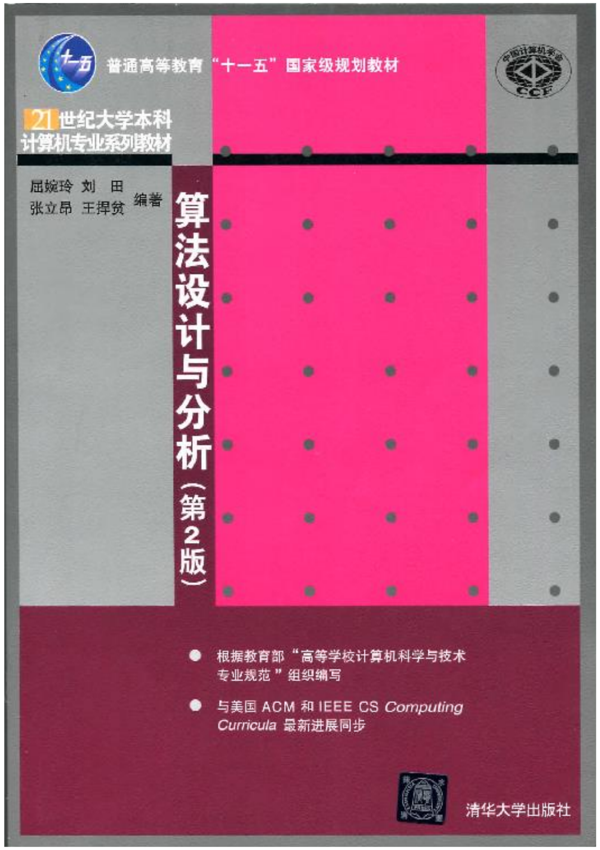
屈婉玲, 刘田, 张立昂, 王捍贫

出版社：

清华大学出版社

出版时间：

2016年第2版



参考书

1. **Jon Kleinberg, Eva Tardos, Algorithm Design, Addison-Wesley, 清华大学出版社影印版，2006.**
2. **Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest, Introduction to Algorithms(3rd edtion), The MIT Press 2009.**
3. **张立昂，可计算性与计算复杂性导引（第3版），北京大学出版社，2011.**
4. **堵丁柱，葛可一，王洁，计算复杂性导论，高教出版社2002.**
5. **Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach, Cambridge University Press，2009.**
6. **屈婉玲, 刘田, 张立昂, 王捍贫, 算法设计与分析习题解答与学习指导，清华大学出版社，2016年第2版**

学习安排

- 授课形式
  - 大班上课：布置书面作业
  - 小班讨论：回课、作业点评、习题、讨论
- 成绩评定：
  - 大班成绩：期中考试(20%)+期末考试（40%）
  - 小班成绩：作业（20%）+ 考勤(5%)+平时表现（15%）
  - 缓考重修的同学，如果不需要重修小班课，请联系助教说明以前上课的年份和小班老师，以便查找以前的小班课成绩

引言：理论上的可计算与现实上的可计算

- 算法研究的重要性
- 理论上的可计算
  ——可计算性理论
- 现实上的可计算
  ——计算复杂性理论

几个例子

例1：求解调度问题

任务集 $S=\{1, 2, \ldots, n\}$，

第 j 项加工时间：$t_j, \in Z^+, j=1, 2, \ldots, n$

一个可行调度方案：$1, 2, \ldots, n$ 的排列 $i_1, i_2, \ldots, i_n$

求：总等待时间最少的调度方案，即求

$$S\text{的排列}i_1, i_2, \ldots, i_n\text{使得}\quad \min\{\sum_{k=1}^{n}(n-k+1)t_{i_k}\}$$

求解方法

贪心策略：加工时间短的先做

如何描述这个方法？这个方法是否对所有的实例都能得到最优解？如何证明？这个方法的效率如何？

例2：投资问题

问题：

m元钱，投资给n个项目，效益函数 $f_i(x)$，表示第 i 个项目投入x元钱的效益，$i=1,2,\ldots,n$. 如何分配每个项目的钱数使得总效益最大？

令x_i 是第 i 个项目的钱数

$$\max \sum_{i=1}^{n} f_i(x_i),$$

$$\sum_{i=1}^{n} x_i = m, \quad x_i \in \mathrm{N}$$

采用什么算法？效率怎样？

蛮力算法的代价

Stirling公式：$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$

非负整数解 <$x_1, x_2, \ldots, x_n$> 的个数估计：

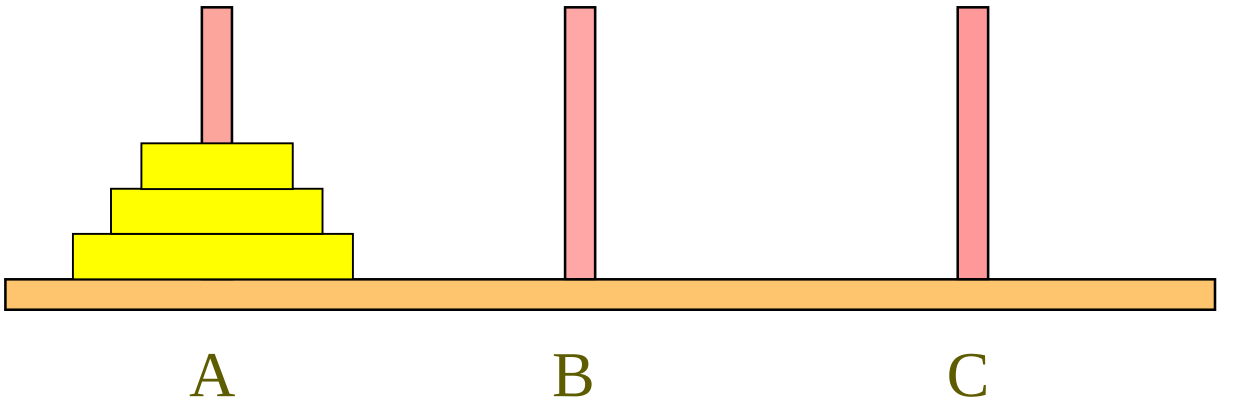
$$C_{m+n-1}^{m} = \frac{(m+n-1)!}{m!(n-1)!} = \Omega((1+\varepsilon)^{m+n-1})$$

蛮力算法——穷举法代价太大

能否利用解之间的依赖关系找到更好的算法？

结论：需要算法设计技术

例3 Hanoi塔问题



$T(n) = 2\,T(n-1) + 1$，$T(1) = 1$，解得 $T(n) = 2^n - 1$

1秒移1个，64个盘子要多少时间？（5000亿年），千万亿次/秒，4个多小时.

思考：是否存在更好的解法？

Reve难题：Hanoi塔变种，柱数增加，允许盘子相等.

其他变种：奇偶盘号分别放置

例4 排序算法的评价

已有的排序算法：考察元素比较次数

插入排序、冒泡排序：最坏和平均状况下都为$O(n^2)$

快速排序：最坏状况为$O(n^2)$，平均状况下为$O(n\log n)$

堆排序、二分归并排序：最坏和平均状况下都为$O(n\log n)$

…

问题

哪个排序算法效率最高？

是否可以找到更好的算法？排序问题的计算难度如何估计？

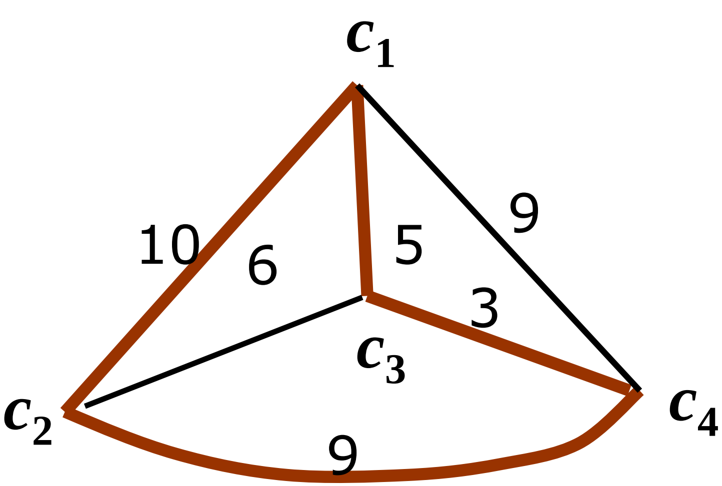
例5 货郎问题

问题：有穷个城市的集合$C = \{ c_1, c_2, \ldots, c_m \}$, 距离

$$d(c_i,c_j) = d(c_j,c_i) \in Z^+, \quad 1 \le i < j \le m$$

求1, 2 …, m的排列$k_1, k_2, \ldots, k_m$使得

$$\min\{\sum_{i=1}^{m-1} d(c_{k_i}, c_{k_{i+1}}) + d(c_{k_m}, c_{k_1})\}$$



现状：至今没有找到有效的算法，存在大量问题与它难度等价

问题：是否存在有效算法？

如何处理这类问题？

Algorithm + Data Structure = Programming

- 好的算法
  - 提高求解问题的效率
  - 节省存储空间
- 需要解决的问题
  - 问题→寻找求解算法　　算法设计技术
  - 算法→算法的评价　　算法分析技术
  - 算法类→问题复杂度的评价　　问题复杂性分析
  - 问题类→能够求解的边界　　计算复杂性理论

算法研究的重要性

- 算法设计与分析技术在计算机科学与技术领域有着重要的应用背景
- 算法设计分析与计算复杂性理论的研究是计算机科学技术的核心研究领域
  - **1966-2005期间，Turing奖获奖50人，其中10人以算法设计，7人以计算理论、自动机和复杂性研究领域的杰出贡献获奖**
  - 计算复杂性理论的核心课题 **“P=NP?”** 是本世纪7个最重要的数学问题之一
- 通过算法设计与分析课程的训练对提高学生的素质和分析问题解决问题的能力，对培养计算思维有着重要的作用

理论上的可计算：可计算性理论

- 研究目标

  确定什么问题是可计算的，即存在求解算法
- 合理的计算模型

  已有的：递归函数、**Turing**机、λ演算、**Post**系统等

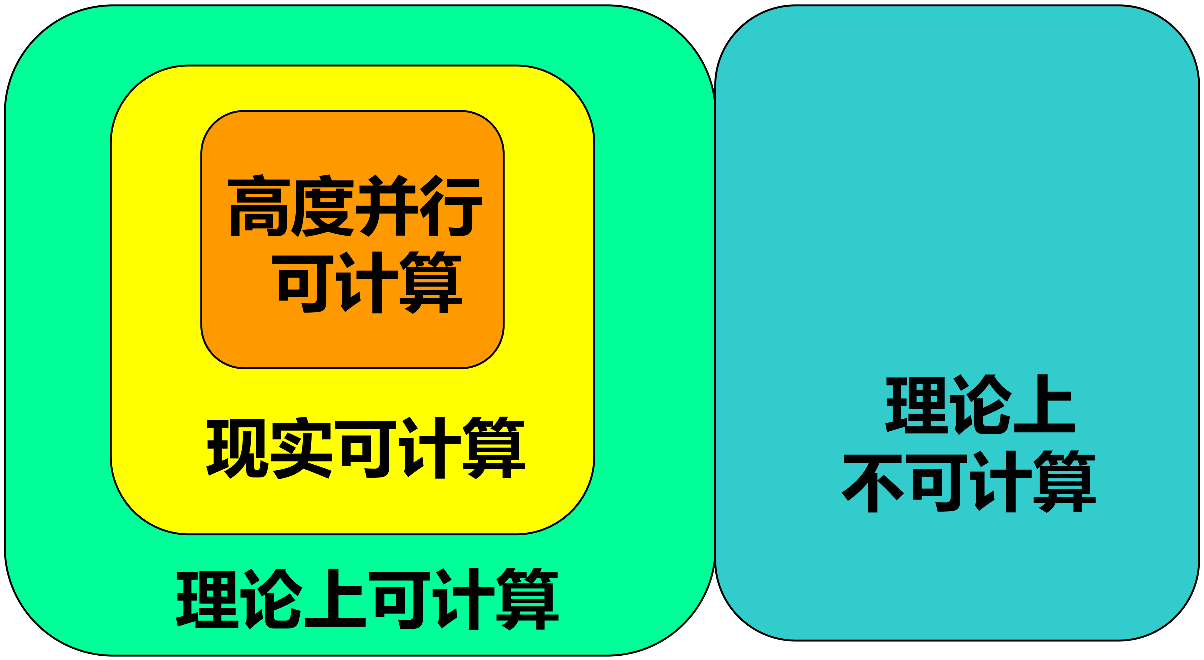
  条件：计算一个函数只要有限条指令

  每条指令可以由模型中的有限个计算步骤完成

  指令执行的过程是确定的
- 核心论题：**Church-Turing**论题

  如果一个函数在某个合理的计算模型上可计算，那么它在**Turing**机上也是可计算的
- 可计算性是不依赖于计算模型的客观性质

理论上与现实上的可计算性



算法至少具有指数时间：理论上可计算——难解的

多项式时间的算法：现实上可计算——多项式时间可解的

对数多项式时间的算法：高度并行可解的

计算复杂性理论

内容

- 算法复杂度——算法所使用的时间、空间的估计
- 问题复杂度——估计问题的难度

术语和概念

- 问题
- 算法
- 算法的时间复杂度
- 函数的阶
- 多项式时间的算法与指数时间的算法
- 问题的复杂度分析

问题

问题：

需要回答的一般性提问，通常含有若干参数

问题描述所包含的内容：

对问题参数的一般性描述

解满足的条件

问题的实例：

对问题的参数的一组赋值

一个问题是由它的全体实例构成的集合

算法

非形式定义

有限条指令的序列

确定了解决某个问题的运算或操作

输入个数大于等于0

输出个数大于0

形式定义

对所有的有效输入停机的Turing机

算法 *A* 解问题 *P*

把问题*P*的任何实例作为算法*A*的输入，*A*能够在有限步停机，并输出该实例的正确的解

算法的描述：伪码

- 保持程序的主要结构
  - 类 C, Pascal
  - 赋值语句：←
  - 分支语句：**if …then …[else…]**
  - 循环语句：**while, for，repeat ..until**
  - 转向语句：**goto**
  - 调用
  - 注释：//…
- 允许使用自然语言
- 常忽略数据结构、模块、异常处理等细节
- 常忽略变量说明

伪码的例子：插入排序

算法 INSERTION-SORT(A)

1. for $j \leftarrow 2$ to $length[A]$
2. do $key \leftarrow A[j]$ // 将$A[j]$插入排好序的序列 $A[1..j-1]$
3. $i \leftarrow j-1$
4. while $i > 0$ and $A[i] > key$
5. do $A[i+1] \leftarrow A[i]$
6. $i \leftarrow i-1$
7. $A[i+1] \leftarrow key$

实例：求最大公约数

算法1.1 Euclid(m,n)

输入：非负整数m,n，其中m与n不全为0

输出：m与n的最大公约数

1. while $m>0$ do
2. $\quad r \leftarrow n \bmod m$
3. $\quad n \leftarrow m$
4. $\quad m \leftarrow r$
5. return n

实例：改进的顺序检索

算法1.2 Search(L,x)

输入：数组 $L[1..n]$，其元素按照从小到大排列，数x.

输出：若x在L中，输出x的位置下标 j；否则输出0.

1. $j \leftarrow 1$
2. **while** $j \le n$ **and** $x > L[j]$ **do** $j \leftarrow j+1$
3. **if** $x < L[j]$ **or** $j > n$ **then** $j \leftarrow 0$
4. **return** j

算法的时间复杂度

- 最坏情况下的时间复杂度

  算法求解输入规模为n的实例所需要的最长时间$W(n)$

- 平均情况下的时间复杂度

  算法求解输入规模为n的实例所需要的平均时间$A(n)$

$$A(n)=\sum_{I\in S} t_I\, p_I$$

S:实例集，t_I: 实例I的基本运算次数，p_I: I的概率

- 复杂度表示

  针对问题选择基本运算

  将基本运算次数表示为输入规模的函数

实例

搜索问题

输入：非降顺序排列的数组L，元素数为n； 数x

输出：j. 若x在L中，j 是 x 首次出现的序标；否则 $j=0$

算法 顺序搜索

假设 x 在 L 中的概率为 p

x 在 L 中不同位置是等概分布的，则

$$W(n)=n$$

$$A(n)=\sum_{i=1}^{n} i\frac{p}{n}+(1-p)n=\frac{p(n+1)}{2}+(1-p)n$$