算法设计与分析

倪庆剑
东南大学
nqj@seu.edu.cn

课程性质

- 在计算机相关学科中处于核心地位
- 必修的专业基础课

课程目标

- 掌握常用的、经典的算法
- 针对具体问题，学会设计解决问题的方法，进而可以分析所设计算法的时空复杂性
- ……

先修课程

- 数据结构
- 离散数学
- 程序设计语言
- ……

为什么要学习这门课

- 重要，必需
- 如果不学
  - 你可能会用一个很花时间和空间的算法去解决问题
  - 你可能会对一个NP-C/NP-Hard的问题寻找有效解法

课程内容

- 算法的基本概念和相关数学知识
- 常用的经典算法及其分析
- 课外的建议
  - 编码实现经典算法

考核方式

- 平时
  - 出勤以及课堂讨论
  - 课程练习
  - 讨论课主题报告
    - 新颖的排序算法
    - 经典算法的典型案例
    - NP-C问题/NP-Hard问题
    - 机器学习算法
    - 智能优化算法
    - 求解特定问题的算法
- 期末考试
  - 闭卷考试

课程参考书

- Introduction to Algorithms, 3rd Edition
  - *Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein*, MIT Press.
  - 算法导论，第三版，机械工业出版社，2012
- 算法设计与分析，王晓东，清华大学出版社
- The Design and Analysis of Computer Algorithms
  - Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman
- ……

课程参考阅读材料

- 迷茫的旅行商：一个无处不在的计算机算法问题
  - William J.Cook 著， 人民邮电出版社，2013
- 可能与不可能的边界：P/NP问题趣史
  - Lance Fortnow 著， 人民邮电出版社，2014
- 算法帝国（Automate this: how algorithms came to rule our world?）
  - Christopher Steiner著，人民邮电出版社，2014
- 算法谜题
  - Anany Levitin，Maria Levitin 著，人民邮电出版社，2014
- 算法的乐趣
  - 王晓华，人民邮电出版社，2015

学习方法

- 思考
  - 体会各种经典算法的思想
- 实践
  - 自己动手编码实现算法
- 思考
  - 求解实际问题的分析与设计

主要知识点

- 算法分析的基本概念
- 排序算法
- 递归与分治
- 动态规划
- 贪心算法
- 随机算法
- 回溯法与分枝定界法
- NP完全理论
- 近似算法
- 智能优化算法
- ……

第一讲

相关背景知识和基础知识

主要内容

- 算法的重要性
- 算法的基本概念
- 算法性能的衡量
- 算法分析

算法的重要性

- 算法在计算机及软件领域中占有重要地位的一个体现
  - 有近1/3的Turing奖获奖者，其成果与算法有关
  - 注：Turing奖（图灵奖）于 1966年开始设立，是ACM（美国计算机协会）在计算机科学技术领域中所授予的最高奖项

算法的重要性（续）

年份	中文译名	姓名	贡献领域
1966年	艾伦·佩利	Alan J. Perlis	高级程序设计技巧，编译器构造
1967年	莫里斯·威尔克斯	Maurice V. Wilkes	存储程序式计算机EDSAC，程序库
1968年	理查德·衛斯里·漢明	Richard Hamming	数值方法，自动编码系统，错误检测和纠错码
1969年	马文·闵斯基	Marvin Minsky	人工智能
1970年	詹姆斯·维尔金森	James H. Wilkinson	数值分析，线性代数，倒退错误分析
1971年	约翰·麦卡锡	John McCarthy	人工智能
1972年	艾兹格·迪科斯彻	Edsger Dijkstra	程序设计语言的科学与艺术
1973年	查理士·巴赫曼	Charles W. Bachman	数据库技术
1974年	高德纳	Donald E. Knuth	算法分析、程序设计语言的设计、程序设计
1975年	艾伦·纽厄尔 赫伯特·西蒙	Allen Newell Herbert A. Simon	人工智能，人类认知心理学和列表处理（list processing）
1976年	迈克尔·拉宾 达纳·斯科特	Michael O. Rabin Dana S. Scott	非确定性自动机
1977年	约翰·巴克斯	John Backus	高级编程系统，程序设计语言规范的形式化定义
1978年	罗伯特·弗洛伊德	Robert W. Floyd	设计高效可靠软件的方法学
1979年	肯尼斯·艾佛森	Kenneth E. Iverson	程序设计语言和数学符号，互动系统的设计，运用 APL进行教学，程序设计语言的理论与实践
1980年	東尼·霍爾	C. Antony R. Hoare	程序设计语言的定义与设计
1981年	埃德加·科德	Edgar F. Codd	数据库系统，尤其是关系型数据库
1982年	史提芬·古克	Stephen A. Cook	计算复杂度
1983年	肯·汤普逊 丹尼斯·里奇	Ken Thompson Dennis M. Ritchie	UNIX操作系统和C语言
1984年	尼克劳斯·维尔特	Niklaus Wirth	程序设计语言设计、程序设计
1985年	理查德·卡普	Richard M. Karp	算法理论，尤其是NP-完全性理论
1986年	约翰·霍普克罗夫特 罗伯特·塔扬	John Hopcroft Robert Tarjan	算法和数据结构的设计与分析

算法的重要性（续）

1987年	约翰·科克	John Cocke	编译理论，大型系统的体系结构，及精简指令集（RISC）计算机的开发
1988年	伊凡·苏泽兰	Ivan Sutherland	计算机图形学
1989年	威廉·卡亨	William Morton Kahan	数值分析
1990年	费尔南多·考巴托	Fernando J. Corbató	CTSS 和 Multics
1991年	罗宾·米尔纳	Robin Milner	LCF，ML語言，CCS
1992年	巴特勒·兰普森	Butler W. Lampson	分布式，个人计算环境
1993年	尤里斯·哈特马尼斯 理查德·斯特恩斯	Juris Hartmanis Richard E. Stearns	计算复杂度理论
1994年	爱德华·费根鲍姆 拉吉·瑞迪	Edward Feigenbaum Raj Reddy	大规模人工智能系统
1995年	曼纽尔·布卢姆	Manuel Blum	计算复杂度理论，及其在密码学和程序校验上的应用
1996年	阿米尔·伯努利	Amir Pnueli	时序逻辑，程序与系统验证
1997年	道格拉斯·恩格尔巴特	Douglas Engelbart	互动计算
1998年	詹姆斯·尼古拉·格雷	James Gray	数据库与事务处理
1999年	弗雷德里克·布鲁克斯	Frederick P. Brooks, Jr.	计算机体系结构，操作系统，软件工程
2000年	姚期智	Andrew Chi-Chih Yao	计算理论，包括伪随机数生成，密码学与通信复杂度
2001年	奥利-约翰·达尔 克利斯登·奈加特	Ole-Johan Dahl Kristen Nygaard	面向对象编程
2002年	罗纳德·李维斯特 阿迪·萨莫尔 伦纳德·阿德曼	Ronald L. Rivest Adi Shamir Leonard M. Adleman	公钥密码学（RSA加密演算法）
2003年	艾伦·凯	Alan Kay	面向对象编程
2004年	文特·瑟夫 罗伯特·卡恩	Vinton G. Cerf Robert E. Kahn	TCP/IP协议
2005年	彼得·诺尔	Peter Naur	Algol 60语言

算法的重要性（续）

2006年	法兰西斯·艾伦	优化编译器
2007年[5]	爱德蒙·克拉克 艾伦·爱默生 约瑟夫·斯发基斯	开发自动化方法检测计算机硬件和软件中的设计错误
2008年	芭芭拉·利斯科夫	编程语言和系统设计的实践与理论
2009年	查尔斯·萨克尔	帮助设计、制造第一款现代PC
2010年	莱斯利·瓦伦特	对众多计算理论所做的变革性的贡献
2011年	朱迪亚·珀尔	通过概率论和因果推理对人工智能领域作出的根本性贡献
2012年	莎菲·戈德瓦塞尔 希尔维奥·米卡利	在密码科学领域里，于复杂理论的基础之上，做出变革性工作；并领先发展出新的具有数学可证明性的有效验证机制[6]
2013年	莱斯利·兰波特	对于分布式及并形系统的理论与实践具有基础性贡献，尤其是诸如因果逻辑时序（causality and logical clocks）、安全性与存活度（safety and liveness）、复制状态机（replicated state machines）及循序一致性（sequential consistency）等理论概念的发明[7]
2014年	迈克尔·斯通布雷克	对现代数据库的概念和实践作出的根本性贡献
2015年	惠特菲尔德·迪菲 马丁·赫尔曼	发明迪菲-赫尔曼密钥交换，对公开密钥加密技术有重大贡献[8]
2016年	蒂姆·伯纳斯-李	发明了万维网、第一个浏览器和使得万维网得以扩展的基础协议及算法[9]
2017年	约翰·轩尼诗 大卫·帕特森	开创了一种系统的、定量的方法来设计和评价计算机体系结构，并对微处理器行业产生了持久的影响。[10]
2018年	约书亚·本希奥 杰弗里·辛顿 杨立昆	深度学习[11]
2019年	艾德文·卡特姆 帕特里克·汉拉恩	对于3D计算机图形学的基本贡献，以及这些技术对电影制作和其他应用中的计算机生成图像（CGI）的革命性影响。[12]
2020年	阿尔佛雷德·艾侯 杰弗瑞·乌尔曼	对程序语言实现的基础性算法和理论的贡献。[13]
2021年	杰克·唐加拉	对数值算法和库的开创性贡献，使高性能计算软件在40多年来跟上了指数级硬件改进的步伐。[14]
2022年	罗伯特·梅特卡夫	发明，标准化以及商业化以太网。[15]

算法的重要性（续）— Turing奖获得者

- 1972，Edsger W.Dijkstra
  - Dijkstra算法
  - PV操作
  - 结构化程序设计
  - “goto有害”
  - 他的一些言论
    - 编程的艺术就是处理复杂性的艺术
    - 优秀的程序员很清楚自己的能力是有限的，所以他对待编程任务的态度是完全谦卑的，特别是，他们会象逃避瘟疫那样逃避 “聪明的技巧”
    - 简单是可靠的先决条件

算法的重要性（续）— Turing奖获得者

- 1974，Donald E.Knuth（高德纳）
  - 多卷算法巨著（算法最早的奠基人之一）
  - 现代“算法”与“数据结构”名词及内涵的提出
  - KMP算法，LR(k)文法，Tex编辑器等
- 1976，Michael O.Rabin & Dana S.Scott
  - 非确定有穷自动机的提出、判定问题
- 1978，Robert W.Floyd
  - 求最短路径的Floyd动态规划算法，Heap-sort算法等
  - 编译及优化（优先文法等），程序正确性证明等

算法的重要性（续）— Turing奖获得者

- 1980，C. Anthony R.Hoare
  - 1983年ACM评出的1/4世纪中最有影响的25篇论文：Hoare与Dijkstra有两篇入选（其余人只有一篇）
  - 算法方面的成就：Quick-sort算法，程序设计（CASE、While语句等）
- 1982，Steven A.Cook
  - “NP-完全”概念的提出与理论的奠定，算法复杂性
- 1984，Niklaus Wirth
  - 结构化程序设计创始人
  - 程序=算法+数据结构

算法的重要性（续）— Turing奖获得者

- 1985，Richard M.Karp
  - 计算机系、数学系、工业工程与运筹学系“三栖教授”，哈佛文学士、理学硕士、数学博士
  - 分枝限界法的创始人（与Held）
  - Rabin-Karp子串匹配算法
  - 求网络最大流的Edmonds-Karp算法
  - NP-完全理论（Karp规约等），随机算法，并行算法等
- 1986，John E.Hopcroft & Robert E.Tarjan
  - 图论算法（深度优先、广度优先算法，连通性、平面图判定，etc.），各种数据结构

算法的重要性（续）— Turing奖获得者

- 1993，Juris Hartmanis & Richard E. Stearns
  - 计算复杂性理论…….
- 1995，Manuel Blum
  - 计算复杂性理论及其应用
- 2000，Andrew Yao
  - 唯一华裔图灵奖获得者
  - 主要贡献在于计算复杂性，量子计算，密码学（e.g. 单向函数）、通信理论等

算法的重要性（续）— Turing奖获得者

- 2002，Ronald L. Rivest，Adi Shamir，Leonard M. Adelman
  - 公共密钥算法
- 2007，Edmund Clarke（美），Allen Emerson（美），Joseph Sifakis（法）
  - “在将模型检查发展为被硬件和软件业中所广泛采纳的高效验证技术上的贡献”
  - “在发现计算机硬件和软件中设计错误的自动化方法方面的工作”
- 2010，Leslie G. Valiant（英）
  - Valiant的代数计算机论是对计算复杂性理论的关键贡献
  - 其1979年提出的上下文无关分析算法，至今仍然是最快算法之一
  - ……
- 2011，Judea Pearl
  - 提出贝叶斯网络(Bayesian network)，概率和因果性推理演算法
- 2012，Shafi Goldwasser和Silvio Micali
  - 密码学方面的贡献
- 2013，Leslie Lamport
  - 面包店算法，拜占庭将军问题，Paxos算法
- 2016，Tim Berners-Lee
  - 使得万维网得以扩展的基础协议及算法
- 2018，Yoshua Bengio, Geoffrey Hinton, Yann LeCun
  - 人工神经网络， 深度学习
- 2019，Patrick M. Hanrahan和Edwin E. Catmull
  - 计算机图形学，计算机生成图像（CGI）

算法的定义

- 目前尚无标准的定义
- An algorithm is a finite, definite, effective procedure, with some input and some output
  - Donald Knuth: The Art of Computer Programming
- Computer Science is the Study of Algorithms
  - Donald Knuth

算法的定义（续）

- 一般认为，算法是由若干条指令组成的有穷序列，具有下列五个特性：
  - 确定性：每条指令都是明确的、无二义的
  - 能行性：每条指令都必须是能够执行的
  - 输入：允许有0个或多个输入量，取自特定的集合
  - 输出：产生一个或多个输出，它（们）与输入量之间存在着某种特定的关系
  - 有穷性：每一条指令执行的次数都是有穷的

算法定义的解释

- 有穷指令序列若满足上述5条，即确定性、能行性、输入、输出和有穷性，则通常称之为算法
- 只满足前4条而不满足第5条（有穷性）的有穷指令序列通常称之为计算过程
- 只要不停电、机器不坏，计算过程就可以永远执行下去（死循环）
- 永远执行的计算过程并非毫无用处——OS就是计算过程

算法及其它概念的比较

- Algorithm: A set of steps that defines how a task is performed
- Program: A representation of an algorithm
- Programming: The process of developing a program
- Software: Programs and algorithms
- Hardware: Equipment

算法好坏的衡量尺度

- 最初，用所需要的计算时间来衡量一个算法的好坏
- 但不同的机器相互之间无法比较
- 需要用独立于具体计算机的客观衡量标准
  - 问题的规模
  - 基本运算
  - 算法的计算量函数

问题的规模

- 一个或多个整数，作为输入数据量的测度
  - 问题：在一个数据表中寻找X
    - 数表的长度（数据项的个数）
  - 问题：求两个实矩阵相乘的结果
    - 矩阵的最大维数（阶数）
  - 图论中的问题
    - 图中的顶点数和边数

基本运算

- 解决给定问题时占支配地位的运算
  - 一般一种，偶尔≥2种
- 在一个表中寻找数据元素x
  - x与表中的一个项进行比较
- 两个实矩阵的乘法
  - 实数的乘法（及加法）
- 数表的排序
  - 表中的两个数据项进行比较

算法的计算量函数

- 用问题规模的某个函数来表示算法的基本运算量,这个表示基本运算量的函数称为算法的时间复杂性（度）
- 时间复杂度用T(n)(或T(n,m)等)来表示
  - $T(n)=5n$
  - $T(n)=3n\log n$
  - $T(n)=4n^3$
  - $T(n)=2^n$
  - $T(n,m)=2(n+m)$
  - ……

渐近时间复杂度

- 当问题的规模趋于极限情形时（相当大）的时间复杂度
- 表示渐近时间复杂性的三个记号
  - $T(n)=O(f(n))$
  - $T(n)=\Omega(f(n))$
  - $T(n)=\Theta(f(n))$
- 请注意：这个概念很重要

T(n)= O(f(n))

- T(n)= O(f(n))
  - 若存在c > 0，和正整数$n_0 \geq 1$，使得当$n \geq n_0$时，总有$T(n) \leq c*f(n)$
  - 给出了算法时间复杂度的上界，复杂度不可能比c*f(n)更大

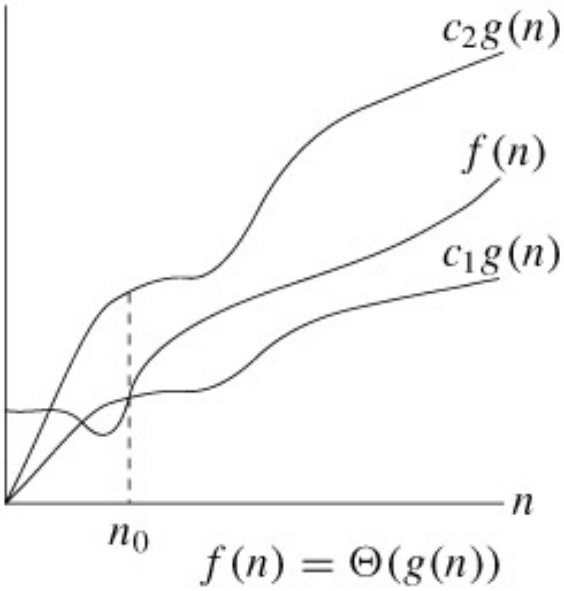
$T(n)=\Omega(f(n))$

- $T(n)=\Omega(f(n))$
  - 若存在$c > 0$，和正整数$n_0 \geq 1$，使得当$n \geq n_0$时，存在无穷多个n，使得$T(n) \geq c*f(n)$成立
  - 给出了算法时间复杂度的下界，复杂度不可能比$c*f(n)$更小

$T(n)=\Theta(f(n))$

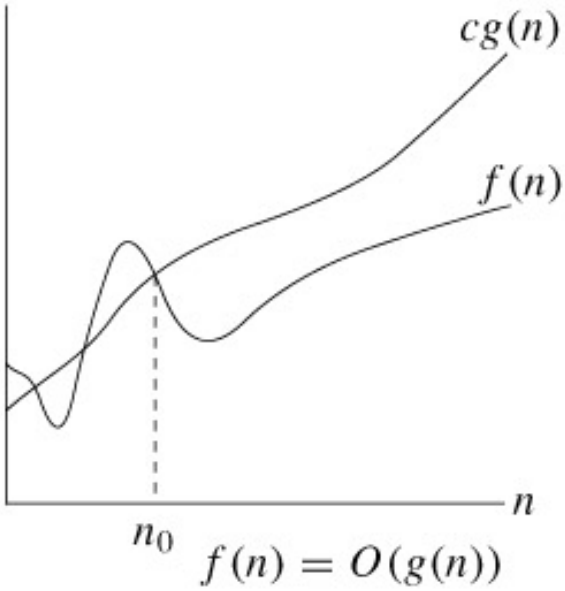
- $T(n)=\Theta(f(n))$
  - 若存在$c_1, c_2>0$，和正整数$n_0\geq 1$，使得当$n\geq n_0$时，总有 $T(n)\leq c_1*f(n)$，且有无穷多个n，使得$T(n)\geq c_2*f(n)$成立,即：$T(n)= O(f(n))$与$T(n)=\Omega(f(n))$都成立
  - 既给出了算法时间复杂度的上界，也给出了下界

记号O，Ω，Θ的图例



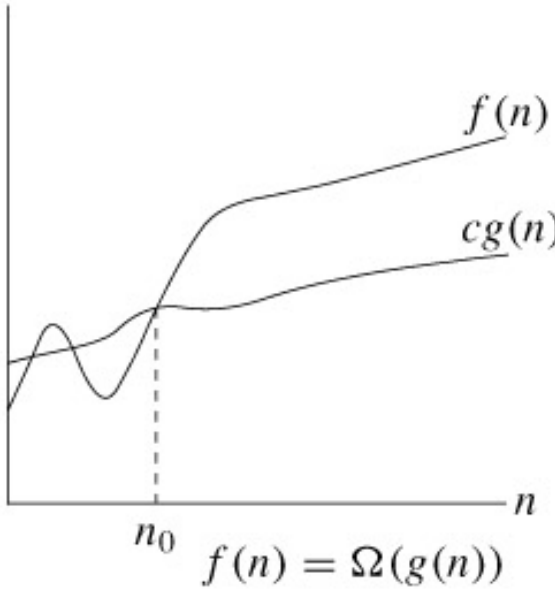
$f(n) = \Theta(g(n))$

(a)



$f(n) = O(g(n))$

(b)



$f(n) = \Omega(g(n))$

(c)

多项式时间和指数时间的比较

- 假设计算机每秒可做基本运算10^8，n=60

	算法1	**算法2**	**算法3**	**算法4**	**算法5**	**算法6**
复杂性	n	n^2	n^3	n^5	2^n	3^n
运行时间	$6*10^{-7}$s	$3.6*10^{-5}$s	0.00216s	0.13min.	3.66世纪	$1.3*10^{14}$世纪

- 多项式时间的算法互相之间虽有差距，一般可以接受
- 指数量级时间的算法对于较大的n无实用价值

最坏情况时间复杂度

- 规模为n的所有输入中，基本运算执行次数为最多的时间复杂度
- 问题：在一个顺序表中寻找数据元素x
  - 顺序查找：最坏情况为O(n)
  - 二分查找：最坏情况为O(logn)

平均情况时间复杂性

- 规模为n的所有输入的算法时间复杂度的平均值
- 一般均假设每种输入情况以等概率出现
- 问题：在一个顺序表中寻找数据元素x
  - 顺序查找：平均情况仍为O(n)
  - 二分查找：平均情况仍为O(logn)

算法研究的几个主要步骤

- 设计
  - 要根据不同的处理对象设计出高质量的算法
- 表示
  - 简明扼要，写出的算法要保证能在计算机上实现
- 确认
  - 对所有合法的输入，算法都要能得到正确的结果
  - 对所有不合法的输入，算法都要能够正确应对

算法研究的几个主要步骤（续）

- 分析
  - 预测算法能在什么样的环境中有效地运行
  - 在最坏、最好和平均情况下能够有什么样的时间复杂度
  - 还需要比较解决同一问题的不同算法各自的优缺点
- 实现和测试
  - 将所给的算法编程实现
  - 通过各种测试来检查所给算法是否正确

算法研究的几个主要步骤

- 设计
- 表示
- 确认
- 分析
- 实现和测试

评价算法的主要方面（1）

- 正确性：评价算法的首要因素
  - 程序正确性证明
  - 程序测试
  - 即使很小的错误也可能会引发巨大的连锁反应，甚至导致严重的后果

评价算法的主要方面（2）

- 健壮性
  - 算法/程序不仅对正确的输入要能计算出正确的结果
  - 对不正确的输入也要能够应对处理
- 简单性
  - 算法/程序的可读性好，易调试、改进
- 高效性
  - 时间、空间复杂度较小，特别是时间复杂度
- 最优性
  - 证明所给算法是解决同一类问题中最好的

算法可以解决哪些问题（应用领域）

- 算法的应用非常广泛，几乎无处不在
  - Caching
  - Compilers
  - Databases
  - Scheduling
  - Networking
  - Data analysis
  - Signal processing
  - Computer graphics
  - Scientific computing

算法可以解决哪些问题（应用领域续）

- 算法的应用非常广泛，几乎无处不在
  - Operations research
  - Artificial intelligence
  - Computational biology
  - ……
- 算法实现的方式：
  - Software
  - Hardware
  - Firmware

算法的应用——实例

- Google PageRank
  - 评估网页重要性
- Google Map, Baidu Map
  - 路径规划
- 金融领域
  - 算法交易，风险控制
- ……
- 太多了，不胜枚举