组合优化

浙江大学数学系　谈之奕

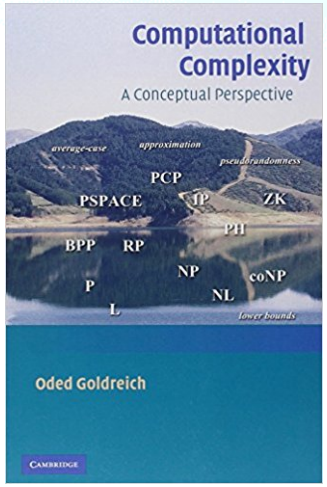
计算复杂性初步

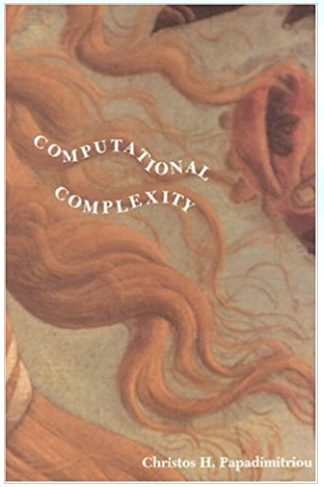
计算复杂性

浙江大学 ZheJiang University

组合优化

- 计算复杂性（**computational complexity**）是理论计算机科学的一个重要分支，主要研究如何界定各类计算任务所需的最低计算资源
  - **Complexity Theory is concerned with the study of the intrinsic complexity of computational tasks**
  - **A typical complexity theoretic study refers to the computational resources required to solve a computational task, rather than referring to a specific algorithm or an algorithmic schema**
  - **Any book on algorithms ends with a chapter on complexity**





Goldreich, O, *Computational Complexity: A Conceptual Perspective*, Cambridge University Press, 2008

Papadimitriou, CH, *Computational Complexity*, Pearson, 1993

算法

浙江大学 ZheJiang University

组合优化

- 算法：在有限步骤内求解某一问题的一组含义明确的可以完全机械执行的规则
- **Algorithm is a sequence of computational steps that transform the input into the output**

Algoritmi（**al-Khwarizmi**的拉丁译名）

↓

Algorism(us)（阿拉伯数字系统，十进制）

↓

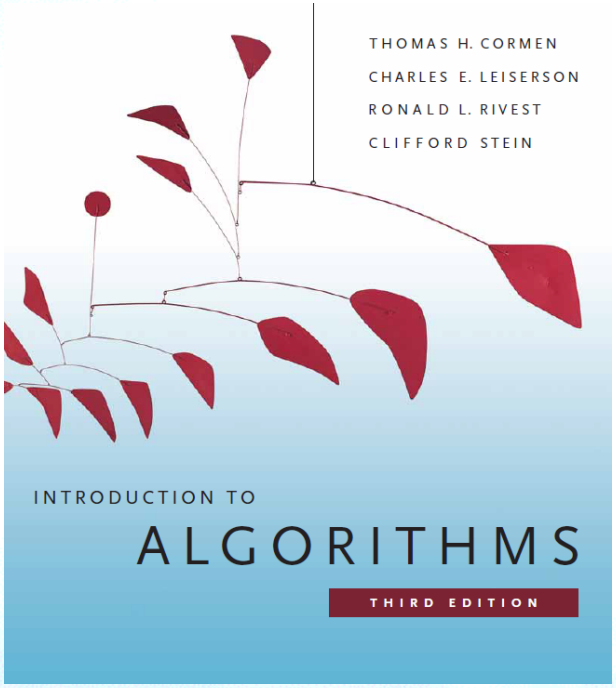
Algorithm （仿**logarithm**所造法语单词，后引入英语，**19**世纪转为现义）

Abu Ja'far Muhammad ibn Musa al-Khwarizmi
（约**780-**约**850**）
波斯数学家

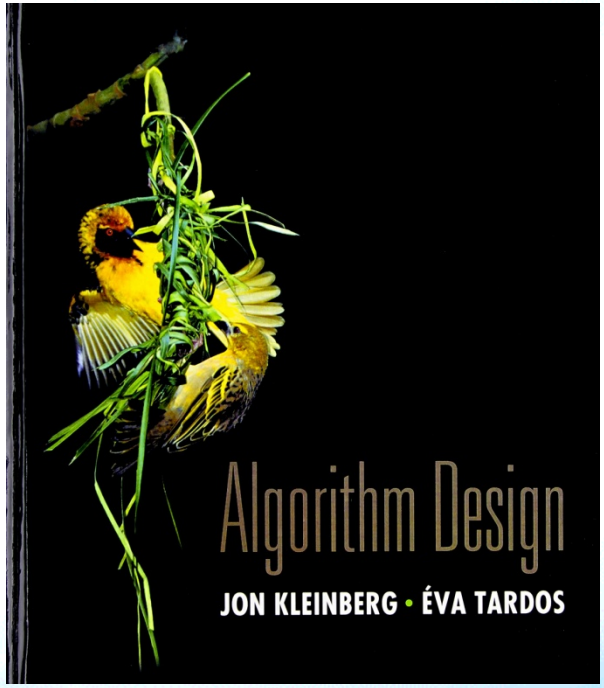
算法

浙江大学 ZheJiang University

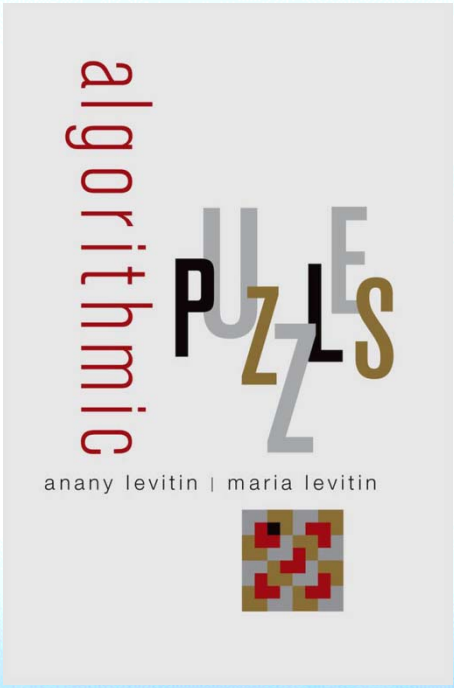
组合优化



Cormen TH, Leiserson CE, Rivest RL, Stein C. *Introduction to Algorithms*. MIT press, 2009



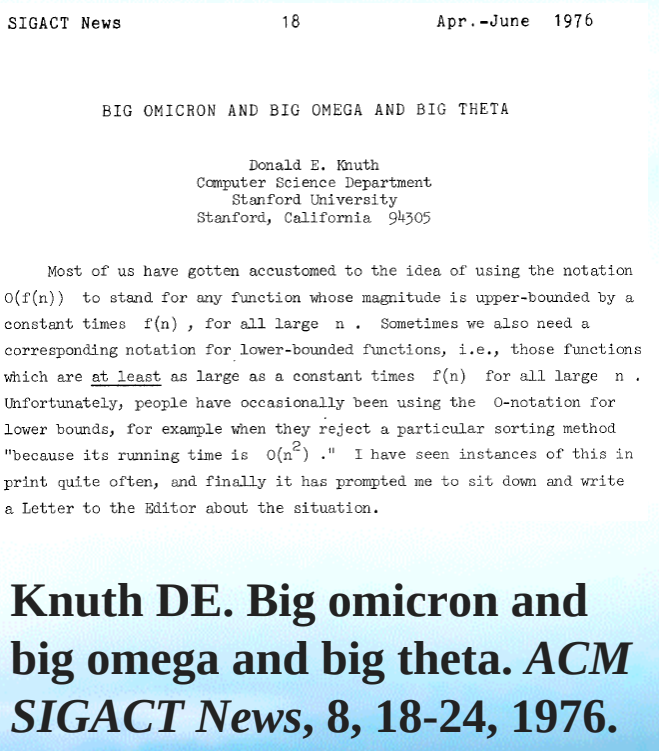
Kleinberg J, Tardos É. *Algorithm Design*. Pearson Education India, 2006



Levitin A, Levitin M. *Algorithmic puzzles*. Oxford University Press, 2011.

O，Ω和Θ

- $f(n)=O(g(n))$：存在常数$C>0$和$n_0>0$，使得对任意$n>n_0$，$|f(n)|\le C|g(n)|$
- $f(n)=\Omega(g(n))$：存在常数$C>0$和$n_0>0$，使得对任意$n>n_0$，$|f(n)|\ge C|g(n)|$
- $f(n)=\Theta(g(n))$：$f(n)=O(g(n))$且$f(n)=\Omega(g(n))$



SIGACT News 18 Apr.-June 1976

BIG OMICRON AND BIG OMEGA AND BIG THETA

Donald E. Knuth
Computer Science Department
Stanford University
Stanford, California 94305

Most of us have gotten accustomed to the idea of using the notation O(f(n)) to stand for any function whose magnitude is upper-bounded by a constant times f(n), for all large n. Sometimes we also need a corresponding notation for lower-bounded functions, i.e., those functions which are at least as large as a constant times f(n) for all large n. Unfortunately, people have occasionally been using the O-notation for lower bounds, for example when they reject a particular sorting method "because its running time is $O(n^2)$." I have seen instances of this in print quite often, and finally it has prompted me to sit down and write a Letter to the Editor about the situation.

Knuth DE. Big omicron and big omega and big theta. *ACM SIGACT News*, 8, 18-24, 1976.

问题与实例

- 问题（**problem**）指需要回答的一般性提问，通常带有若干参数；对一问题的所有参数指定具体值可得到该问题的一个实例（**instance**）
- 问题类型
  - 判定问题（**decision problem**）：任一实例只有“是”、“否”两个可能答案的问题
  - 优化问题（**optimization problem**）
    - 求值问题（**evaluation problem**）：求实例最优值的问题
    - 构造问题（**construction problem**）：求实例最优解的问题

优化问题与判定形式

- **TSP问题**
  - 现有 n 个城市，城市 i 与城市 j 之间的距离为整数 c_{ij}。求城市的一个排列 π，使得环游长 $\sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)}$ 最小
  - （判定形式）给定 c_{ij} 和整数阈值 M，问是否存在排列 π，使得环游长不超过 M，即

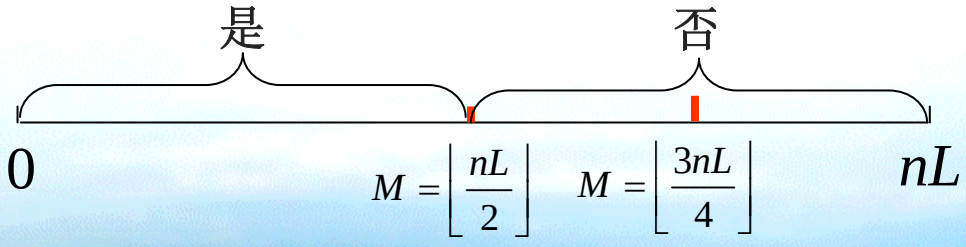
$$\sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)} \le M$$

对极大化问题为“至少为”、“$\ge$”

优化问题与判定形式

- 优化问题与其判定形式在求解上的等价性
  - 求解优化问题的算法可直接用来求解其判定形式

    比较最优值C^*与阈值M的大小
  - 利用二分法的思想，可通过多次调用求解判定形式的算法来求解优化问题



$$C^* \in [0, nL], L = \max_{i,j} c_{ij}$$

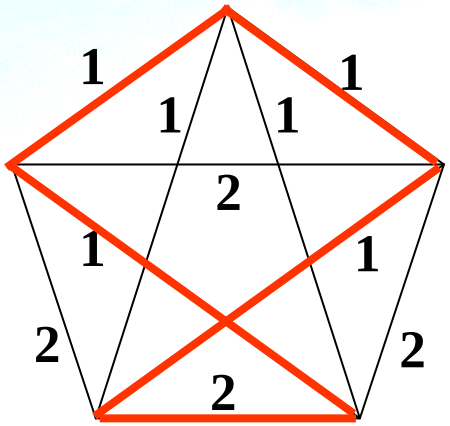
$$C^* \in \left[0, \frac{nL}{2}\right] \text{或} C^* \in \left[\frac{nL}{2}, nL\right]$$

$\lceil \log(nL) \rceil$次调用后可确定C^*

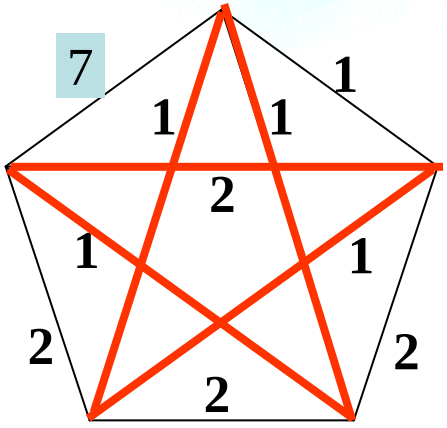
优化问题与判定形式

浙江大学 ZheJiang University

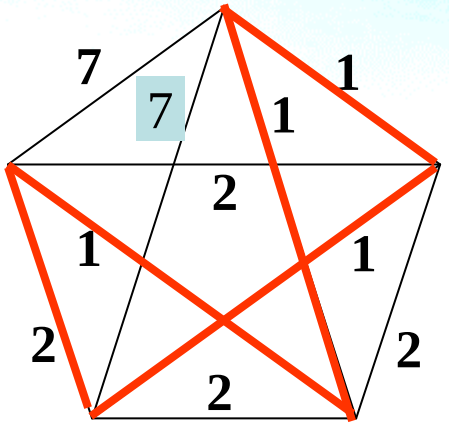
组合优化



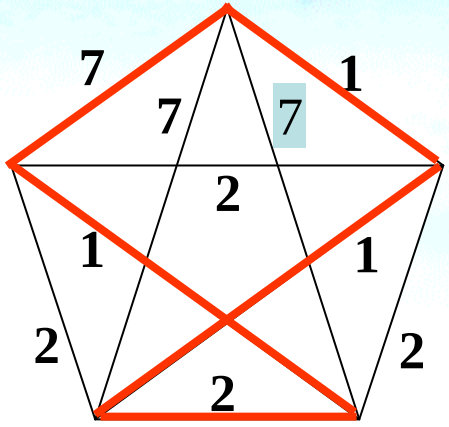
$C^* = 6$



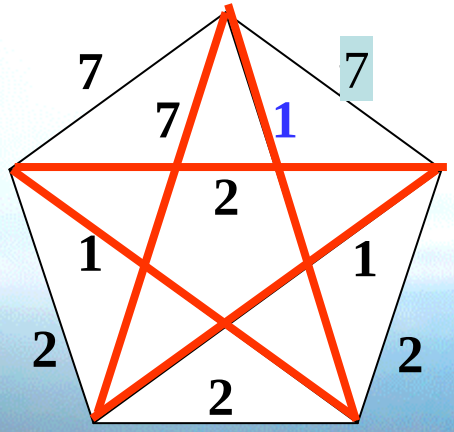
$C^* = 6$



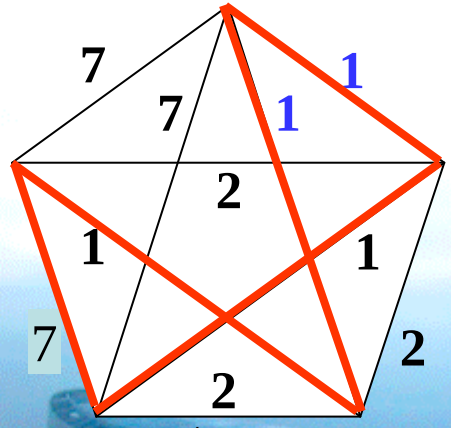
$C^* = 6$



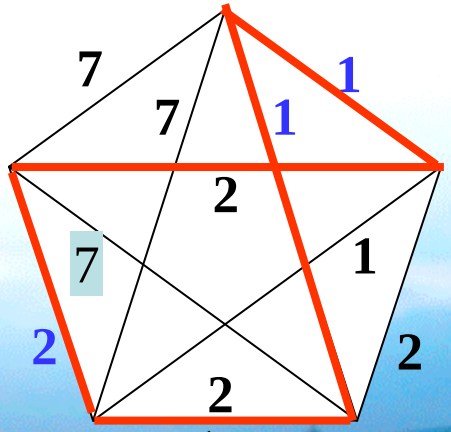
$C^* = 12$



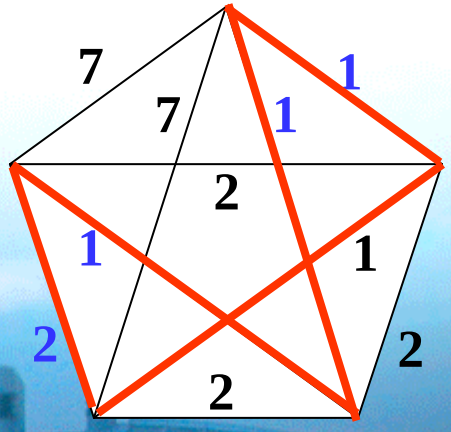
$C^* = 12$



$C^* = 11$

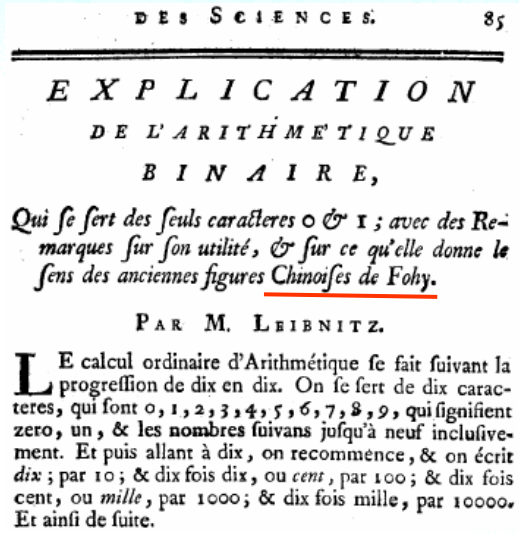


$C^* = 8$



规模和编码方案

- 描述一实例所需的计算机存储单元数称为该实例的规模（**size**）
- 在计算机中，常用二进制表示整数，因此存储大小为 k 的整数所需字节数为 $\lfloor \log_2 k \rfloor + 1$
- 为消除编码方式和存储方式不同所可能带来的影响，可以用多项式函数相互限制的一切规模表达式都是合理的
  - $f(I) = O(\text{poly}(g(I)))$，$g(I) = O(\text{poly}(f(I)))$



DES SCIENCES. 85

EXPLICATION
DE L'ARITHMETIQUE
BINAIRE,

Qui ſe ſert des ſeuls caracteres 0 & 1 ; avec des Remarques ſur ſon utilité, & ſur ce qu'elle donne le ſens des anciennes figures Chinoiſes de Fohy.

PAR M. LEIBNITZ.

LE calcul ordinaire d'Arithmétique ſe fait ſuivant la progreſſion de dix en dix. On ſe ſert de dix caracteres, qui ſont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, qui ſignifient zero, un, & les nombres ſuivans juſqu'à neuf inclusivement. Et puis allant à dix, on recommence, & on écrit *dix* ; par 10 ; & dix fois dix, ou *cent*, par 100 ; & dix fois cent, ou *mille*, par 1000 ; & dix fois mille, par 10000. Et ainſi de ſuite.

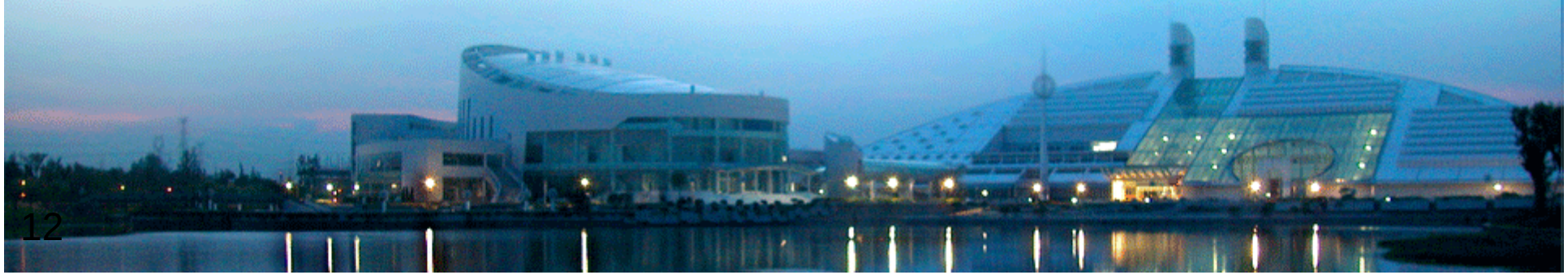
Leibniz G., Explication de l'Arithmétique Binaire, *Memoires de mathématique et de physique de l'Académie royale des sciences*, Académie royale des sciences, 1703

规模

- **TSP问题实例** $n, c_{ij}, i, j = 1, \cdots, n$ $L = \max_{i,j} c_{ij}$
  - 规模可以表示成 $\sum_{i,j}\left(\lfloor \log_2 c_{ij} \rfloor + 2\right) = 2n^2 + \sum_{i,j}\lfloor \log_2 c_{ij} \rfloor$，也可以简单的表示成 $n + \lceil \log_2 L \rceil$
    - $2n^2 + \sum_{i,j}\lfloor \log_2 c_{ij} \rfloor \le 2n^2 + n^2\lceil \log_2 L \rceil \le \left(n + \lceil \log_2 L \rceil\right)^3$
    - $n + \lceil \log_2 L \rceil \le 2n^2 + \sum_{i,j}\lfloor \log_2 c_{ij} \rfloor$
  - 不能表示成 n^2 或 $n + L$
    - $2n^2 + \sum_{i,j}\lfloor \log_2 c_{ij} \rfloor \not\le \text{poly}(n^2)$, $n + L \not\le \text{poly}\left(2n^2 + \sum_{i,j}\lfloor \log_2 c_{ij} \rfloor\right)$

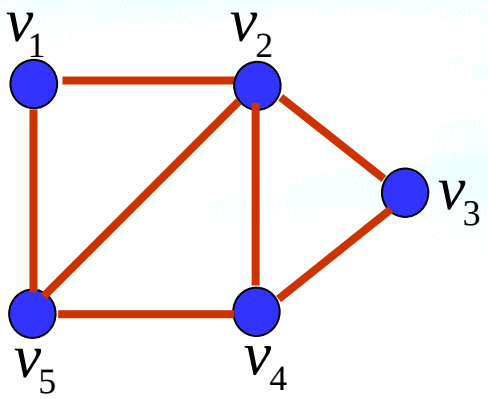
最大数

- 实例中出现的最大整数，称为实例的最大数
  - 最大数是一个数值，不是存储该数所用的字节数
  - 最大数可以是规模的指数函数，也可以是规模的多项式函数
- **TSP**问题实例的最大数 $B=\max\left\{n,\max\limits_{i,j}c_{ij}\right\}$，是实例规模 $n+\lceil\log_2 L\rceil$ 的指数函数



规模与最大数

- **Euler图问题：判断一简单图是否为Euler图**
- 图在计算机中的表示
  - 邻接矩阵（**adjacency matrix**）法
- 含 n 个顶点的图的实例的规模为 n^2，最大数为 n，最大数是规模的多项式函数



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

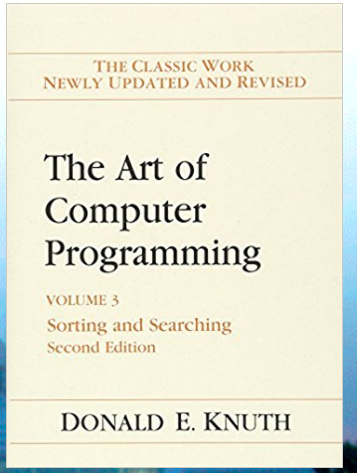
时间复杂度

- 用算法执行过程中所需的加、乘、比较、赋值等基本运算次数表示算法所用的时间
  - 冒泡排序（**Bubble Sort**）算法的时间复杂度为 $O(n^2)$
    - 对 n 个数进行排序，需要进行的读取、交换和比较次数至多为 $7.5n^2+0.5n+1$，至少为 $8n+1$

考虑基本运算次数，而非算法实现后的实际运行时间
与输入规模相关，而非绝对次数
考虑最坏情况，而非平均或最好情况

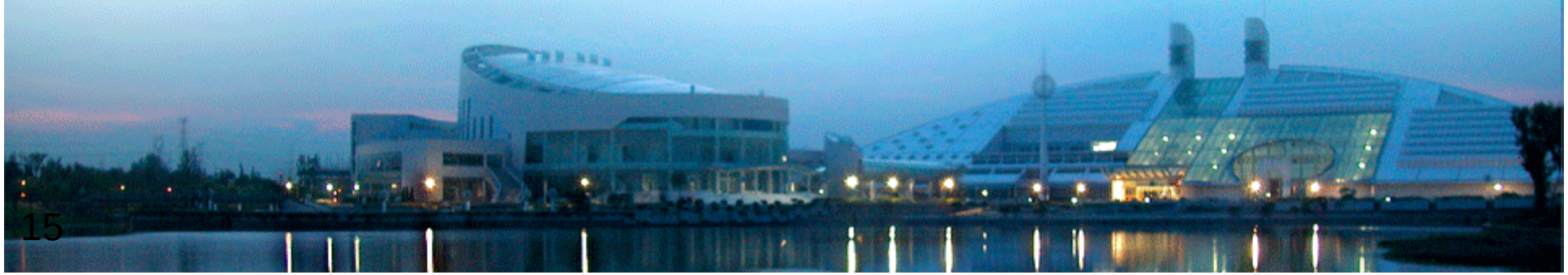
6 5 3 1 8 7 2 4

Knuth DE. *The Art of Computer Programming*.（Vol. 1-4）Addison-Wesley, 1968-2015.



时间复杂度

- 算法的时间复杂度（**time complexity**）是关于实例规模 n 的一个函数 $f(n)$，它表示用该算法求解所有规模为 n 的实例中所需基本运算次数最多的那个实例的基本运算次数
- 若一算法的时间复杂度 $f(n)=O(\text{poly}(n))$，则称它为多项式时间算法；不能这样限制时间复杂度函数的算法称为指数时间算法 $O((\ln n)^{\ln n})$
- 若某算法的时间复杂度是实例规模 n 和最大数 B 的二元多项式函数 $f(n)=O(\text{poly}(n,B))$，但不是实例规模 n 的（一元）多项式函数，则称它为伪多项式时间算法（**pseudo-polynomial time algorithm**）



背包问题

- 实例规模与最大数
  - n件物品，物品 j 的价值为 p_j，大小为 w_j，背包容量为 C
  - 规模 $\sum_{j=1}^{n}\left(\lfloor \log_2 p_j \rfloor+2\right)+\sum_{j=1}^{n}\left(\lfloor \log_2 w_j \rfloor+2\right)+\left(\lfloor \log_2 C \rfloor+2\right)$或 $n+\lceil \log_2 L \rceil$，其中 $L=\max\left\{\max_{j=1,\cdots,n} p_j, \max_{j=1,\cdots,n} w_j, C\right\}$
  - 最大数 $B=\max\{n,L\}$是规模的指数函数
- 算法时间复杂性举例
  - 时间复杂性为 $O(n^2)$，$O(n\log B)$等的算法都是多项式时间算法
  - 时间复杂性为 $O(n!)$，$O(2^n B)$，$O(n2^B)$等的算法都是指数时间算法
  - 时间复杂性为 $O(nB)$，$O(B\log n)$等的算法都是伪多项式时间算法

高效算法

组合优化

- 结合关于函数增长速度的比较，和算法的实际运行效果，通常将多项式时间算法称为高效算法（**efficient algorithm**）
  - 多项式次调用多项式时间算法仍是多项式时间算法

2. Digression. An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

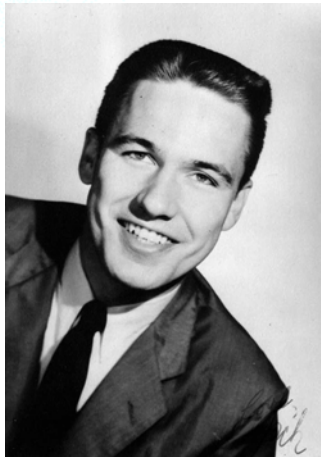
There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

—— Edmonds, J. Paths, trees, and flowers.
***Canadian Journal of Mathematics*, 17, 449–467, 1965**

高效算法

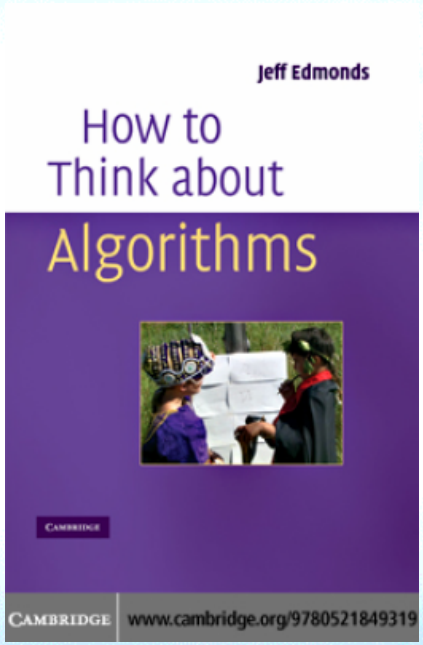
浙江大学 ZheJiang University

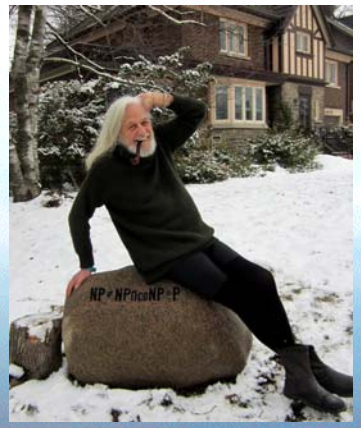
组合优化



Kathie Cameron
加拿大劳瑞尔大学
数学系教授

Jeff Edmonds
加拿大约克大学电气工程
和计算机科学系教授



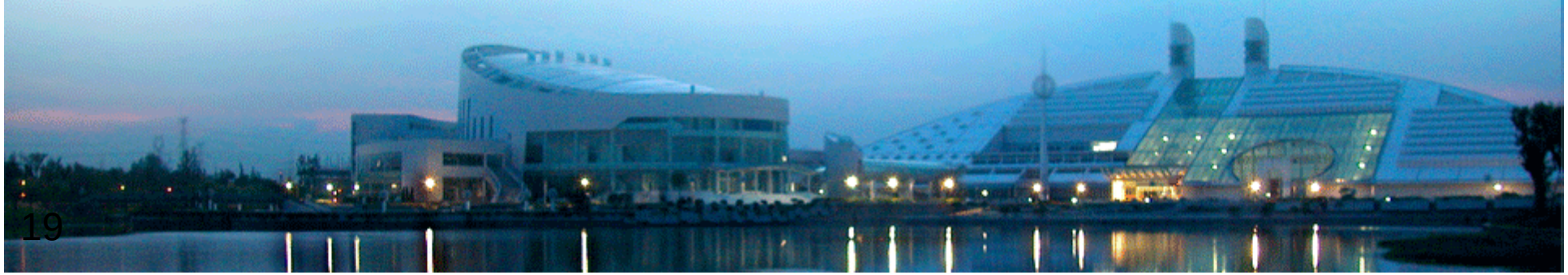


Jack Edmonds
（1934-）
原加拿大滑铁卢
大学数学系教授
(上图摄于1957年)

Edmonds J. *How to think about algorithms*. Cambridge University Press, 2008.

伪多项式时间算法

- 若实例中不出现“很大的”数，则伪多项式时间算法也是高效的
  - $f(I) = O(\text{poly}(\text{size}(I), \text{Max}(I)))$
    - 若 $\text{Max}(I) = O(p_1(\text{size}(I)))$，则 $f(I) = O(p_2(\text{size}(I)))$
    - 若 $\text{Max}(I) = \Omega(2^{\text{size}(I)})$，则 $f(I) = \Omega(2^{\text{size}(I)})$
  - 实例中不出现“很大的”数的可能情形
    - **TSP**问题：“所有城市之间的距离为**1**或**2**”等情形
    - 大部分无权简单图上的问题等非数字问题



问题规模

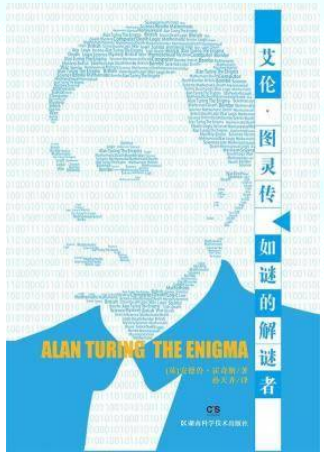
- 任务安排问题
  - 现有 n 项任务，任务 j 所需时间为 p_j
    - 实例规模为 $n+\lceil \log_2 L \rceil$ ，其中 $L=\max\limits_{j=1,\cdots,n} p_j$
  - 现有 n 项任务，每项任务所需时间均为 p
    - 实例规模为 $\log n+\log p$
  - 时间复杂性为 $O(n\log n)$ 的问题对前者是多项式时间的，对后者是指数时间的

对实例规模即为存储实例所需空间这一定义不能机械地理解。既不要不敢于在多项式相关范围内作适当简化，更不能通过“浪费”空间以“降低”算法的时间复杂性

图灵机

- 计算复杂性理论建立在一种名为图灵机（**Turing machine**）的抽象计算机模型之上，该模型由**Turing**于**1936**年提出
- **Church–Turing Thesis**：所有在某个合理的计算模型上可计算的函数在图灵机上也是可计算的

Alan Turing
英国数学家、计算机学家
(1912-1954)



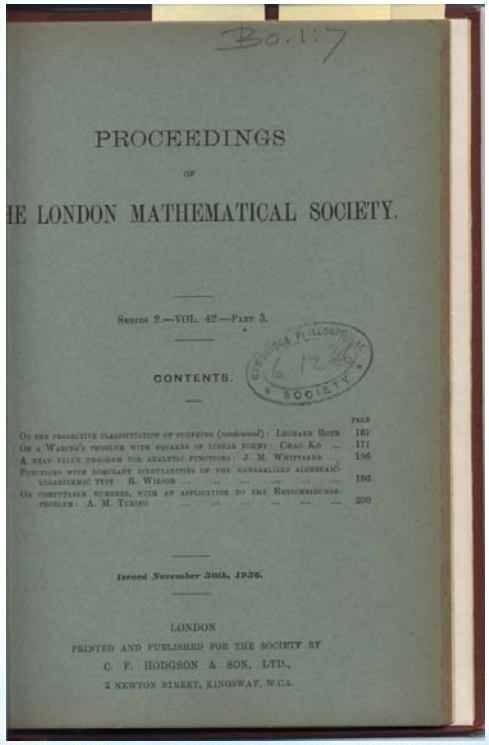
Hodges, A, *Alan Turing: The Enigma*, Princeton University Press, 2014

The Imitation Game (模仿游戏)
（2014年上映，第87届奥斯卡金像奖最佳改编剧本奖）

图灵机

浙江大学 ZheJiang University

组合优化



PROCEEDINGS

OF

HE LONDON MATHEMATICAL SOCIETY.

SERIES 2.—VOL. 42.—PART 3.

CONTENTS.

PAGE

On the Projective Classification of Surfaces (continued): LEONARD ROTH 161

On a Waring's problem with squares of linear forms: CHAO KO ... 171

A mean-value theorem for analytic functions: J. M. WHITTAKER ... 186

Functions with dominant singularities of the generalized algebraic-logarithmic type: B. WILSON 196

On computable numbers, with an application to the Entscheidungs-problem: A. M. TURING 230

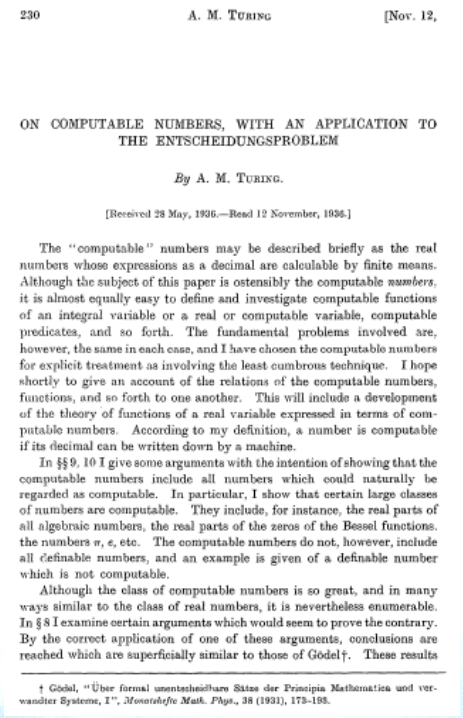
Issued November 30th, 1936.

LONDON

PRINTED AND PUBLISHED FOR THE SOCIETY BY

C. F. HODGSON & SON, LTD.,

2 NEWTON STREET, KINGSWAY, W.C.2.



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

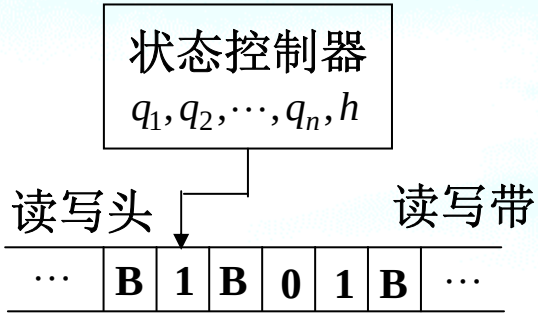
2012年6月23日Turing诞辰 100 周年当日Google发布的互动Doodle

Turing, A., On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, S2-42, 230–265

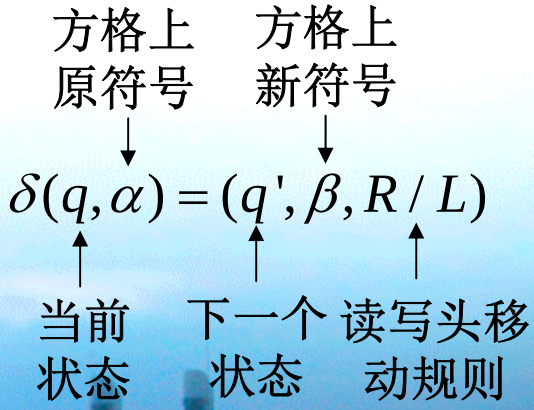
(German)decision problem

图灵机

- 图灵机由一条（或多条）读写带、一个有限状态控制器和一个读写头构成。读写带长度无限，分为多个小方格，每个小方格上可写入字母表（**alphabet**）中的一个符号，读写头总是指向其中一个小方格
- 某时刻图灵机的当前状态、读写头所在位置、读写带上的非空白字符串构成瞬间像（**configuration**）。图灵机每一步运行时，根据该图灵机的状态转移函数，从一个瞬间像变化到另一个瞬间像
  - 读取当前状态和读写头所扫描的方格上的符号，获得以此为自变量的状态转移函数的函数值
  - 读写头在所扫描的方格上消去原符号，写上新符号
  - 读写头向左或向右移动一个方格
  - 图灵机由当前状态转向下一个状态

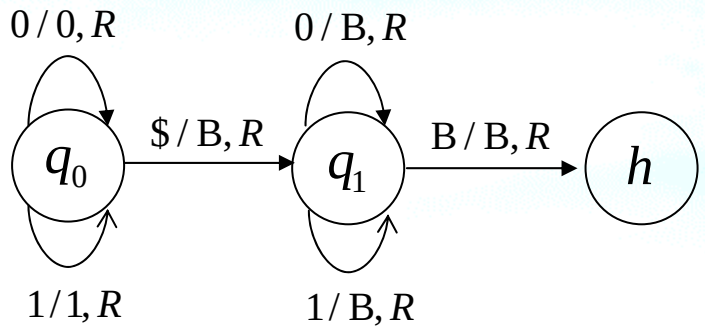


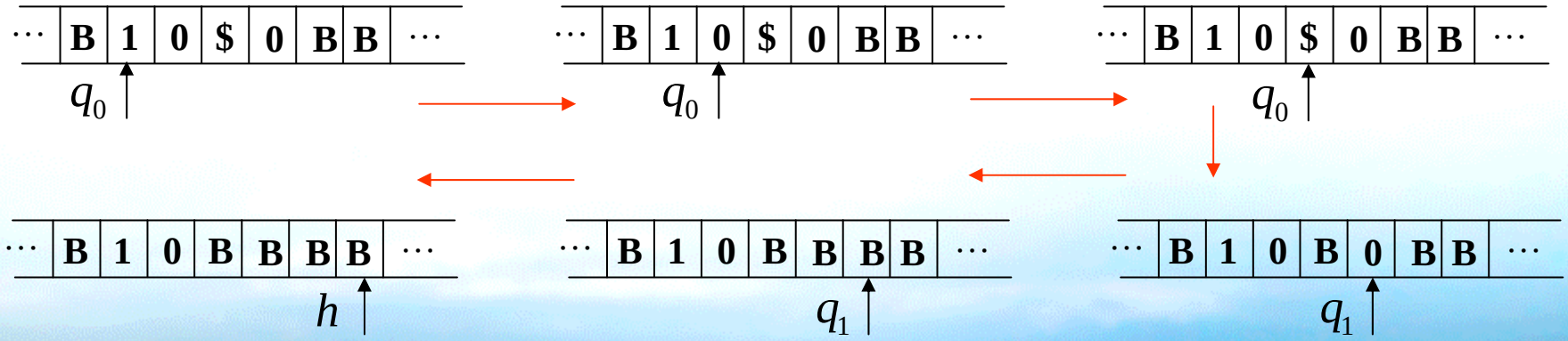
状态转移函数



投影

δ	0	1	\$	B
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, R)	
q_1	(q_1, B, R)	(q_1, B, R)		(h, B, R)

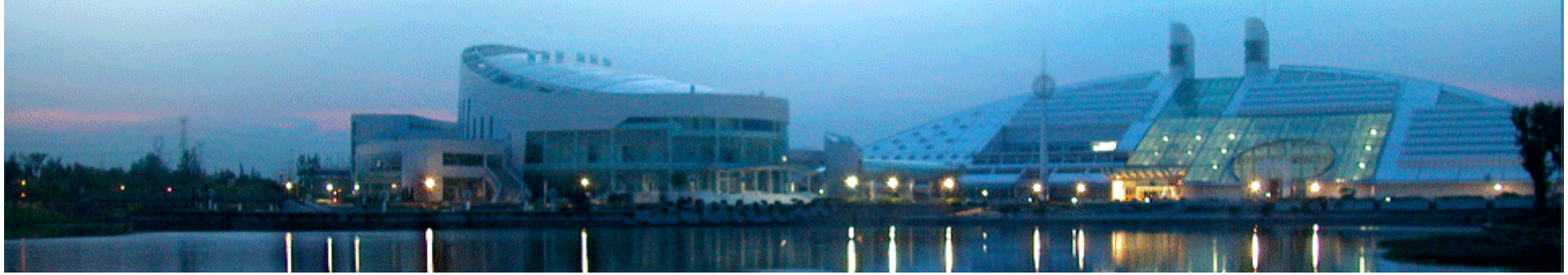




$\mathcal{P}$ 类

- 图灵机计算的输入、时间和空间
  - 计算开始时读写带上的非空白字符即为输入
  - 从计算开始到终止的移动次数为计算时间
  - 从计算开始到终止读写头扫描过的小方格数为计算空间
- 确定性图灵机多项式时间可解问题类（**polynomial solvable problem class**），称为$\mathcal{P}$类，简记为$\mathcal{P}$
- 基本运算均能通过图灵机经过多项式步移动实现，证明一问题属于$\mathcal{P}$类只需设计出求解该问题的多项式时间算法

$\mathcal{P}$ 类问题	多项式时间算法
最小生成树	**Krasual**算法
指派问题	匈牙利算法
中国邮递员问题	**Edmonds-Johnson** 算法
素性测试	**AKS**算法
线性规划	椭球法或内点法



素性测试

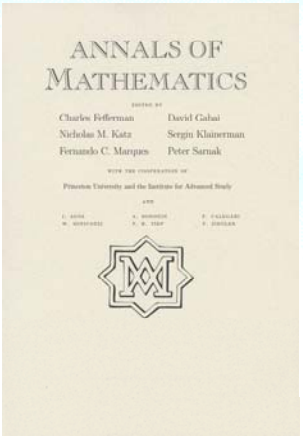
Eratosthenes筛法（Sieve of Eratosthenes）

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

Prime numbers

Eratosthenes of Cyrene
（约公元前276-
约公元前194）
古希腊科学家

素性测试



- 素性测试问题：给定整数 n，判断 n 是否为素数
  - 实例规模为 $\log_2 n$，**Eratosthenes**筛法是指数时间算法
  - **AKS**素性测试算法可在 $O(\log_2^{7.5} n \cdot \mathrm{poly}(\log\log n))$时间内完成

Agrawal M, Kayal N, Saxena N, PRIMES is in P.
***Annals of Mathematics*, 160, 2, 781-793, 2004**

EATCS与ACM学会Godel Prize（2006）
outstanding papers in the area of theoretical computer science

MOS与AMS学会Fulkerson Prize（2006）
outstanding papers in the area of discrete mathematics

AKS算法

浙江大学 ZheJiang University

组合优化

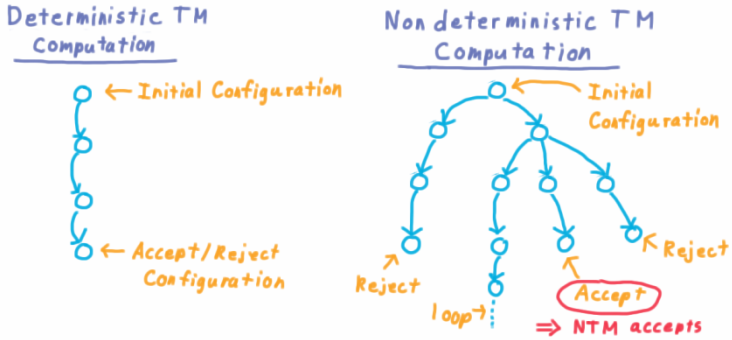
- **Manindra Agrawal：印度理工学院坎普尔学院（Indian Institute of Technology Kanpur）计算机科学与工程系教授**
- **Neeraj Kayal与Nitin Saxena在2002年时均为该系本科生，“Towards a Deterministic Polynomial-Time Primality Test”是他们的一项本科生科研项目**

Here enter the heros of our story, off stage until now, the students Neeraj Kayal and Nitin Saxena. Both were members of the Indian team in the 1997 International Mathematical Olympiad. Studying computer science instead of mathematics because of better employment prospects, they found in complexity theory a way to continue working with mathematics on a high level.

Bornemann F, PRIMES Is in P: A Breakthrough for “Everyman”，*Notices of the AMS*, 2003.5, 545-552

$\mathcal{NP}$ 类

- 非确定性图灵机多项式时间可解问题类（**nondeterministic polynomial solvable problem class**）称为$\mathcal{NP}$ 类，简记为$\mathcal{NP}$
  - 确定性图灵机的状态转移函数是单值的，非确定性图灵机的状态转移函数可能是多值的
  - 确定性图灵机是一种特殊的非确定性图灵机 $\mathcal{P} \subseteq \mathcal{NP}$
  - 任一非确定性图灵机可用一确定性图灵机进行模拟，完成同样计算，但需花费更多时间 $\mathcal{P} = \mathcal{NP}$?



Brubaker C, Fortnow L. Computability, Complexity & Algorithms, Online Course(CS 6505), Georgia Institute of Technology

千年难题

浙江大学 ZheJiang University

组合优化

- 是否有 $\mathcal{P} = \mathcal{NP}$ 成立是数学和理论计算机科学中一个重要课题

- **Millennium Problems**
  - **Yang–Mills and Mass Gap**
  - **Riemann Hypothesis**
  - **P vs NP Problem**
  - **Navier–Stokes Equation**
  - **Hodge Conjecture**
  - **Poincaré Conjecture**
  - **Birch and Swinnerton-Dyer Conjecture**

Clay Mathematics Institute（CWI）

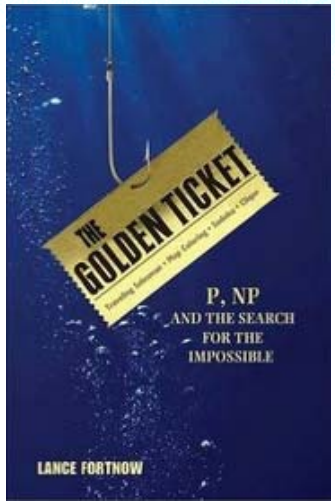
Devlin, K. J., *The Millennium Problems: The Seven Greatest Unsolved Mathematical Puzzles of Our Time*, Basic Books , 2003.（中译本：沈崇圣译，上海科技教育出版社，2012）

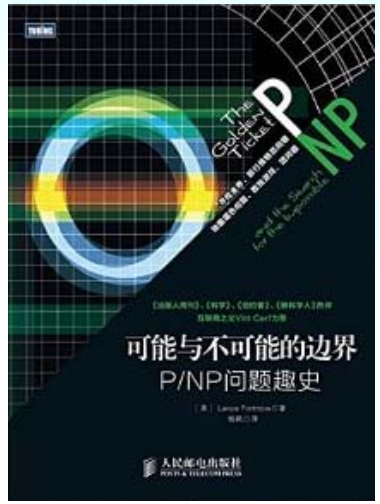
$\mathcal{P}=\mathcal{NP}$ 猜想

浙江大学 ZheJiang University

组合优化

- 尽管 $\mathcal{P}=\mathcal{NP}$猜想是未决问题，但是目前普遍相信 $\mathcal{P}\neq\mathcal{NP}$ 成立，并在此假设下进一步研究 $\mathcal{NP}$ 类内部的结构





Fortnow, L., *The Golden Ticket: P, NP, and the Search for the Impossible*, Princeton University Press , 2013.（中译本：可能与不可能的边界：**P/NP问题趣史**，杨帆译，人民邮电出版社，**2014.**

Fortnow L. The status of the P versus NP problem. *Communications of the ACM*, 2009, 52(9): 78-86.


浙江大学
ZheJiang University
谢　谢