组合优化

浙江大学数学系　谈之奕

浙江大学
ZheJiang University

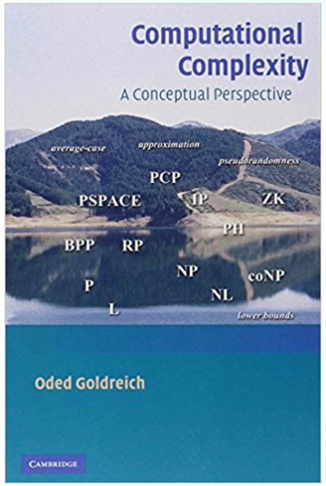
计算复杂性初步

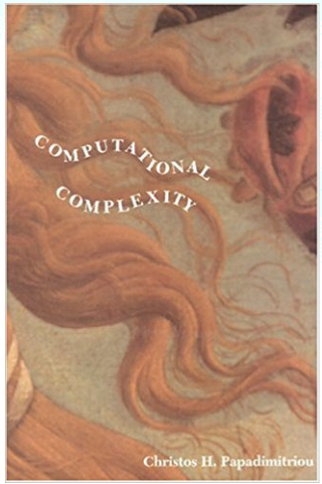
计算复杂性

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

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）

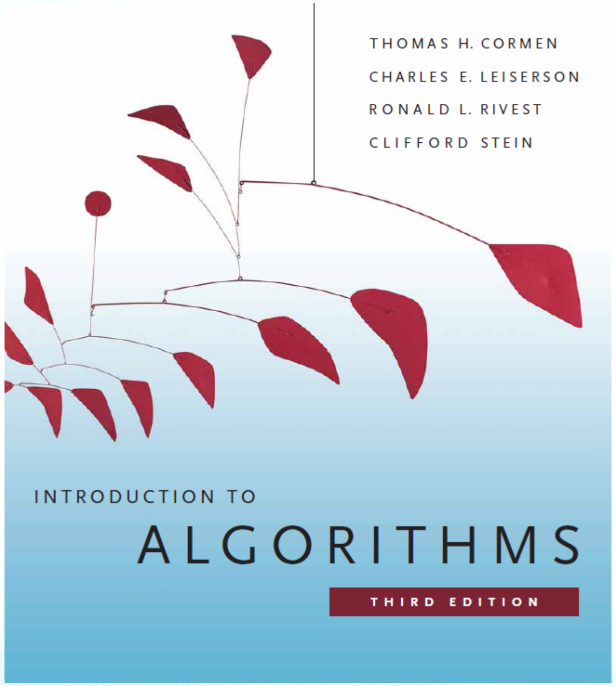
Abu Ja'far Muhammad ibn Musa al-Khwarizmi

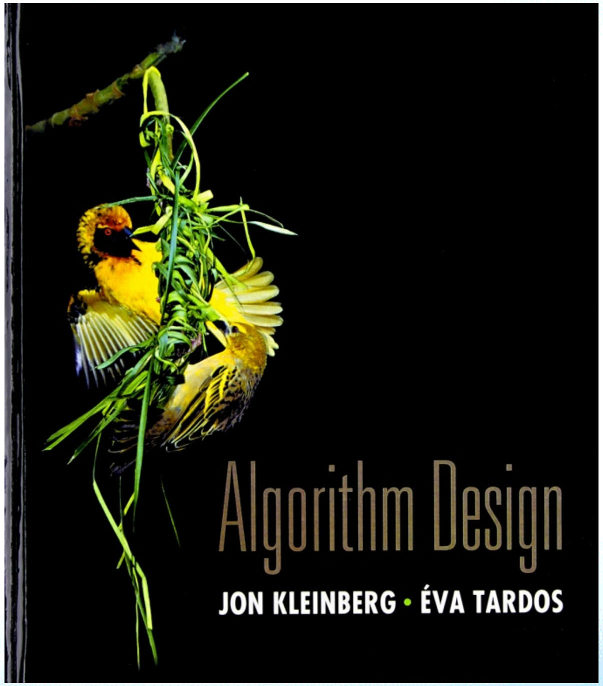
（约**780**-约**850**）

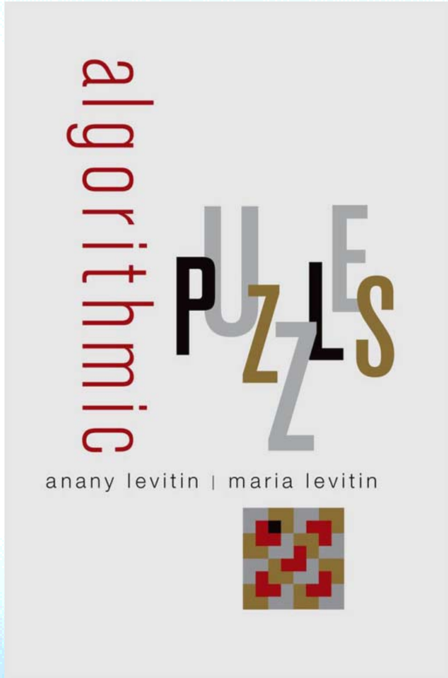
波斯数学家

算法

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

浙江大学 ZheJiang University

组合优化

- $f(n)=\mathrm{O}(g(n))$：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 $n>n_0$，$|f(n)|\ge C|g(n)|$
- $f(n)=\Theta(g(n))$：$f(n)=\mathrm{O}(g(n))$ 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

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

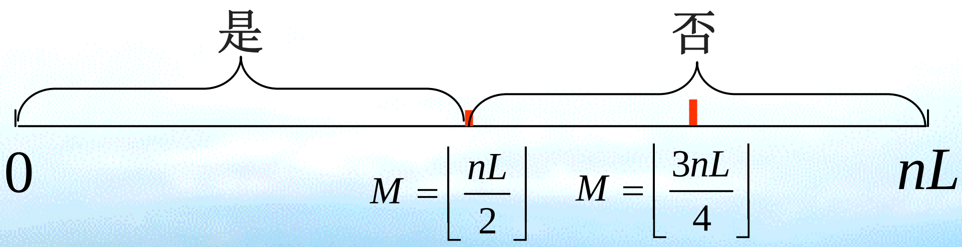
对极大化问题为“至少为”、“$\ge$”

优化问题与判定形式

组合优化

- 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



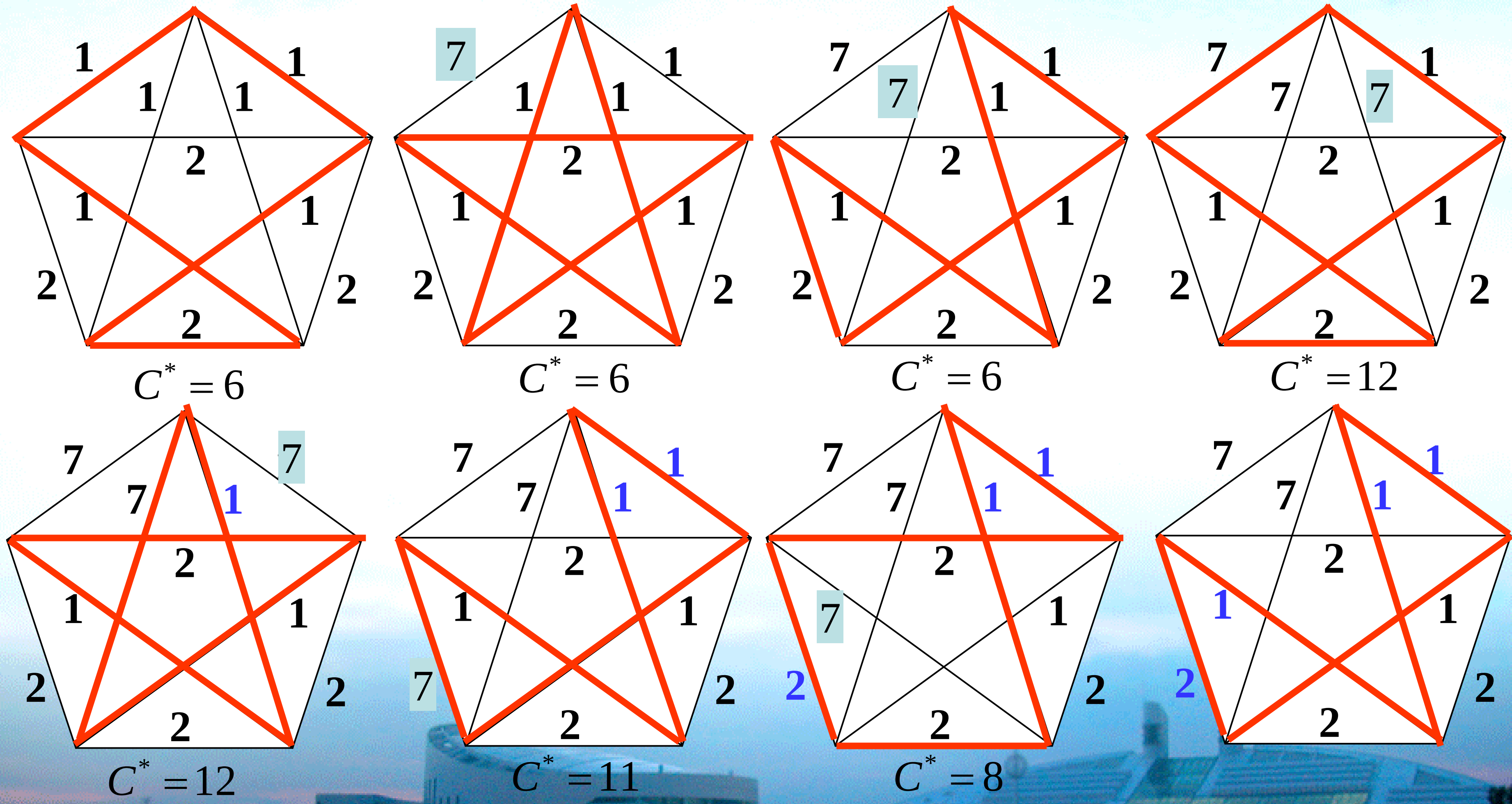
$$C^* \in [0, nL], L = \max_{i,j} c_{ij}$$

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

$\lceil \log(nL) \rceil$次调用后可确定C^*

优化问题与判定形式

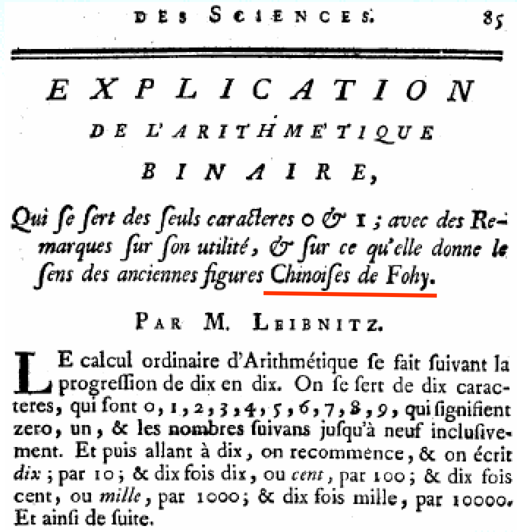
组合优化



规模和编码方案

组合优化

- 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
  - $f(I) \le \text{poly}(g(I))$ ，$g(I) \le \text{poly}(f(I))$



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

组合优化

- **TSP问题实例** $n, c_{ij}, i, j = 1, \cdots, n$ $\quad L = \max_{i,j} c_{ij}$
  - 规模可以表示成 $\sum_{i,j}\left(\left\lfloor \log_2 c_{ij} \right\rfloor + 2\right) = 2n^2 + \sum_{i,j}\left\lfloor \log_2 c_{ij} \right\rfloor$，也可以简单的表示成 $n + \left\lceil \log_2 L \right\rceil$
    - $2n^2 + \sum_{i,j}\left\lfloor \log_2 c_{ij} \right\rfloor \le 2n^2 + n^2\left\lceil \log_2 L \right\rceil \le \left(n + \left\lceil \log_2 L \right\rceil\right)^3$
    - $n + \left\lceil \log_2 L \right\rceil \le 2n^2 + \sum_{i,j}\left\lfloor \log_2 c_{ij} \right\rfloor$
  - 不能表示成 n^2 或 $n + L$
    - $2n^2 + \sum_{i,j}\left\lfloor \log_2 c_{ij} \right\rfloor \le p(n^2)$ $\quad n + L \le p\left(2n^2 + \sum_{i,j}\left\lfloor \log_2 c_{ij} \right\rfloor\right)$

浙江大学 ZheJiang University

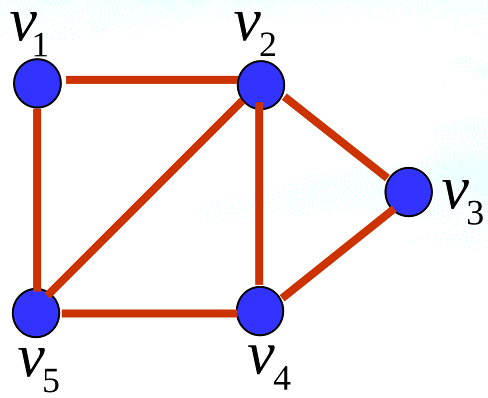
组合优化

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 $B = \max\left\{ n, \max_{i,j} c_{ij} \right\}$，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



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

浙江大学
ZheJiang University
组合优化

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 $8n+1$

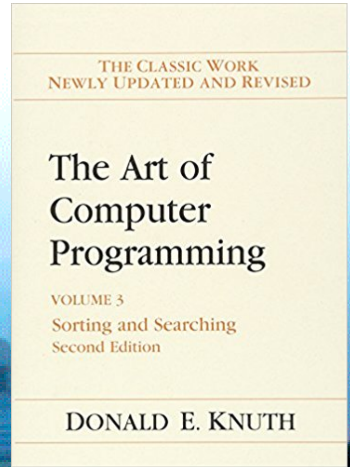
考虑基本运算次数，而非算法实现后的实际运行时间

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.



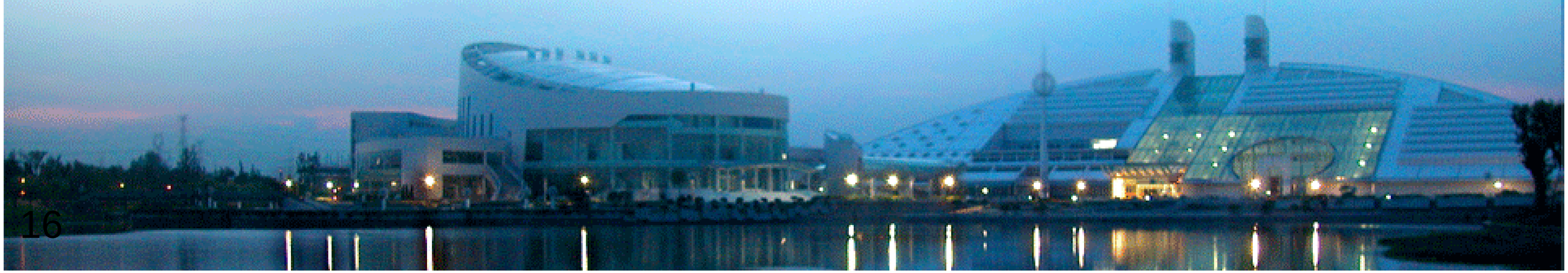
时间复杂度

- 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 $f(n)=O(\mathrm{poly}(n))$，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 $f(n)=O(\mathrm{poly}(n,B))$，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

组合优化

- 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

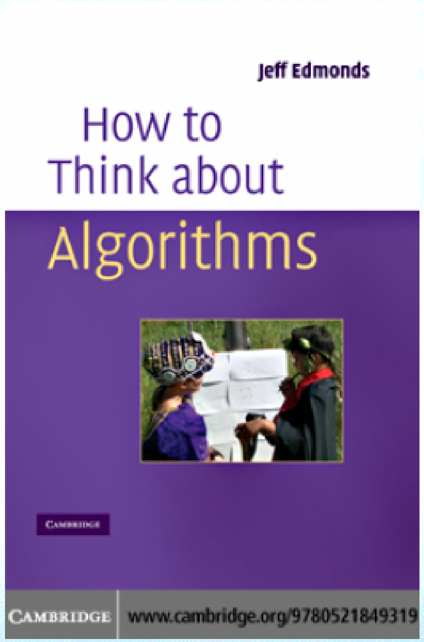
浙江大学
ZheJiang University

组合优化

Kathie Cameron
加拿大劳瑞尔大学
数学系教授

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

Jeff Edmonds
加拿大约克大学电气工程
和计算机科学系教授



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
  - $f(I) = O(\mathrm{poly}(\mathrm{size}(I), \mathrm{Max}(I)))$
    - 若 $\mathrm{Max}(I) = O(p_1(\mathrm{size}(I)))$，则 $f(I) = O(p_2(\mathrm{size}(I)))$
    - 若 $\mathrm{Max}(I) = \Omega(2^{\mathrm{size}(I)})$，则 $f(I) = \Omega(2^{\mathrm{size}(I)})$
  - 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

浙江大学 ZheJiang University

组合优化

- 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 $\mathrm{O}(n\log n)$ 的问题对前者是多项式时间的，对后者是指数时间的

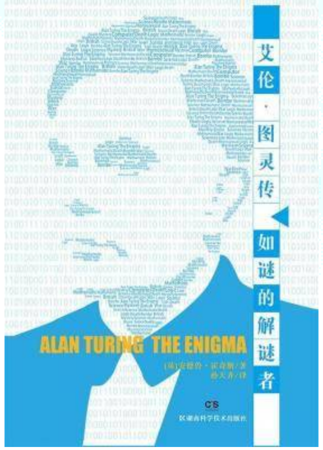
对实例规模即为存储实例所需空间这一定义不能机械地理解。既不要不敢于在多项式相关范围内作适当简化，更不能通过“浪费”空间以“降低”算法的时间复杂性

图灵机

组合优化

- 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)



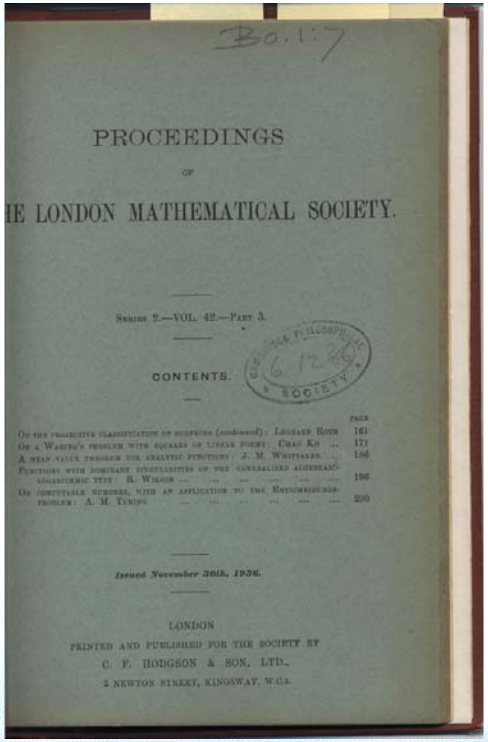
Hodges, A, *Alan Turing: The Enigma*, Princeton University Press, 2014

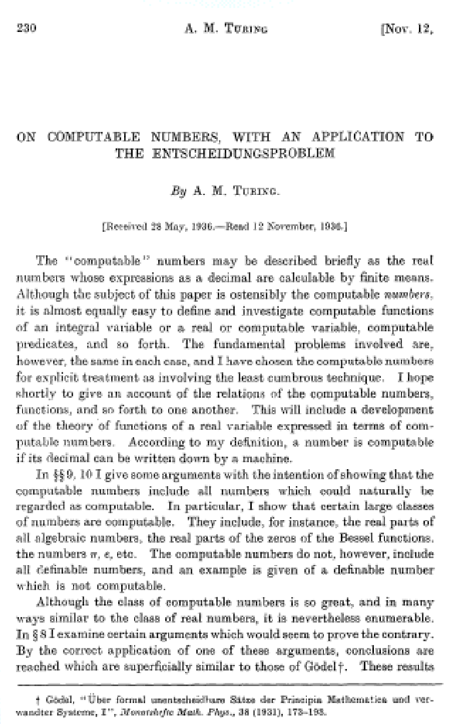
The Imitation Game (模仿游戏**)**
（**2014**年上映，第**87**届奥斯卡金像奖最佳改编剧本奖）

图灵机

ZheJiang University

组合优化





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

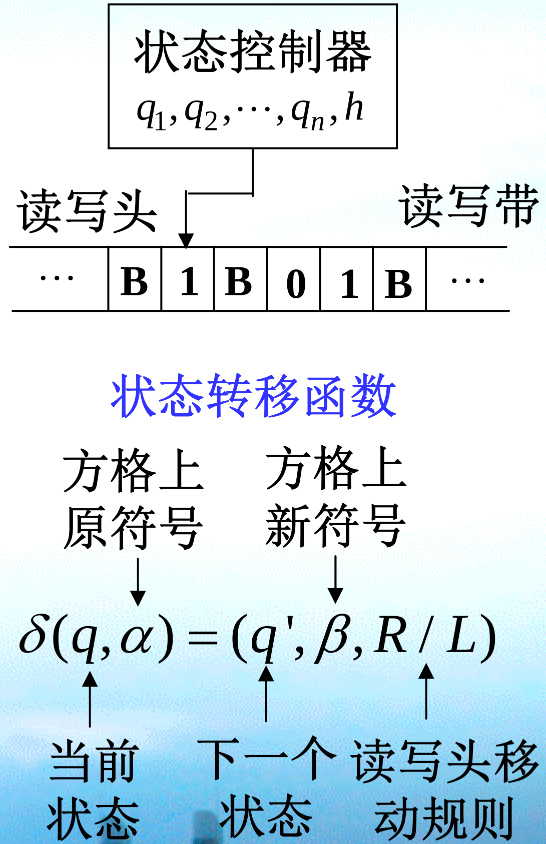
Turing, A., On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society*, S2-42, 230–265

(German)decision problem

图灵机

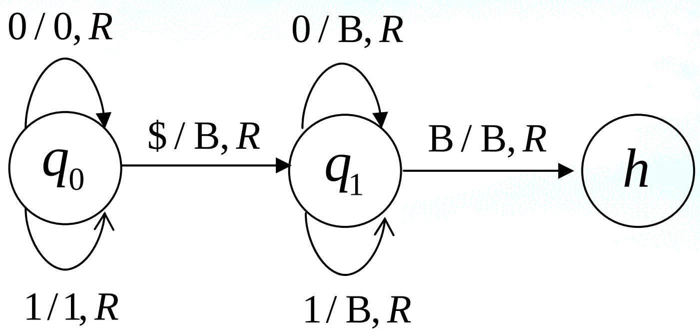
组合优化

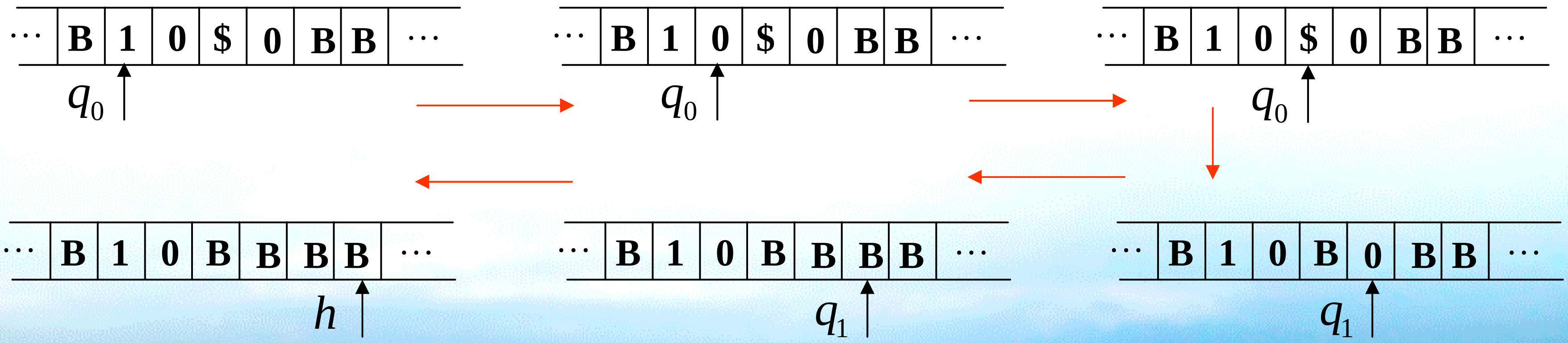
- 图灵机由一条（或多条）读写带、一个有限状态控制器和一个读写头构成。读写带长度无限，分为多个小方格，每个小方格上可写入字母表（**alphabet**）中的一个符号，读写头总是指向其中一个小方格
- 某时刻图灵机的当前状态、读写头所在位置、读写带上的非空白字符串构成瞬间像（**configuration**）。图灵机每一步运行时，根据该图灵机的状态转移函数，从一个瞬间像变化到另一个瞬间像
  - 读取当前状态和读写头所扫描的方格上的符号，获得以此为自变量的状态转移函数的函数值
  - 读写头在所扫描的方格上消去原符号，写上新符号
  - 读写头向左或向右移动一个方格
  - 图灵机由当前状态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



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

组合优化

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

组合优化

- 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



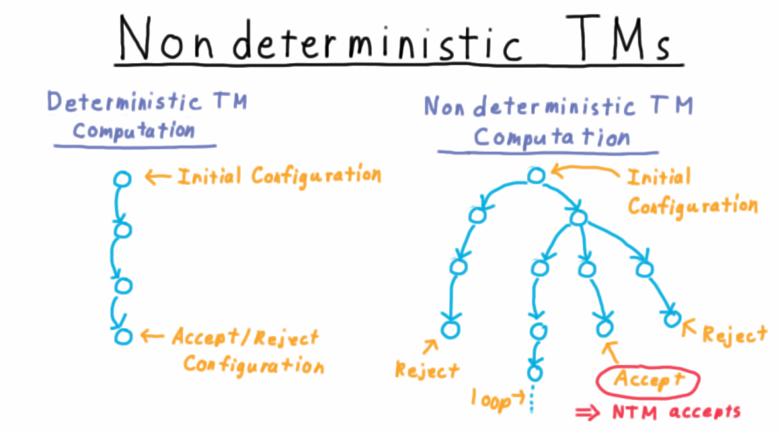
Manindra Agrawal：印度理工学院坎普尔学院（**Indian Institute of Technology Kanpur**）计算机科学与工程系教授

Neeraj Kayal与**Nitin Saxena**是**1997**年国际数学奥林匹克印度队成员，**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

$\mathcal{NP}$ 类

组合优化

- 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

组合优化

千年难题

- 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**

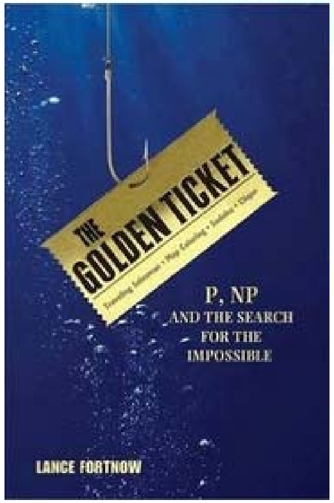
Clay Mathematics Institute（CWI）

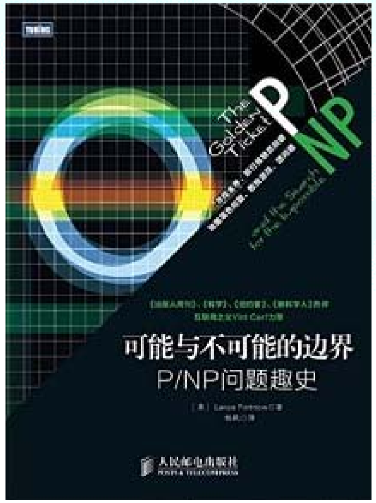
Devlin, K. J., *The Millennium Problems: The Seven Greatest Unsolved Mathematical Puzzles of Our Time*, Basic Books , 2003. （中译本：沈崇圣译，上海科技教育出版社，**2012**）

$\mathcal{P}=\mathcal{NP}$ 猜想

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.

$\mathcal{NP}$ 类

- 基于非确定性算法的$\mathcal{NP}$ 类定义
  - 对一判定问题，若存在一非确定性算法，使得对任何一个答案为“是”的实例，该算法能
    - 猜想出该实例的一个可行解
    - 该可行解规模不超过实例规模的多项式
    - 能在实例规模的多项式时间内验证猜想是否正确

  则称该问题属于$\mathcal{NP}$ 类

非确定性算法只是为研究而定义的一种理论算法模型，在现实生活中并不存在

$\mathcal{NP}$ 类

组合优化

- **TSP判定形式** $\in \mathcal{NP}$
  - 猜想出实例的一个可行解 π
  - π 可按经过城市的标号顺序用 n 个数 $i_1, i_2, \cdots, i_n$ 表示，可行解规模为 $n\log_2 n$ ，不超过实例规模 $n+\log_2 L$ 的某个多项式函数
  - 用 n 次加法即可验证 $\sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)} + c_{\pi(n)\pi(1)} \le M$ 成立

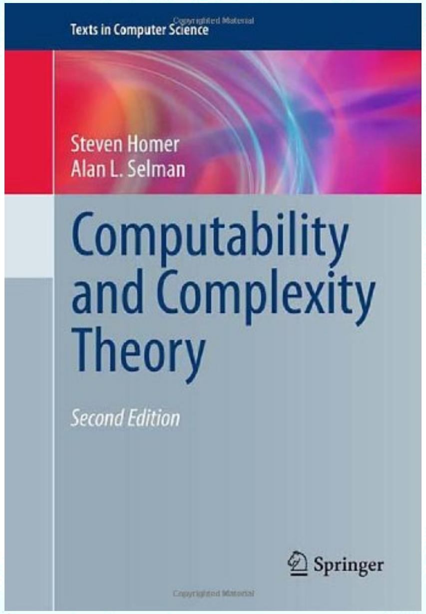
给定城市间距离 c_{ij} 和整数阈值 M，问是否存在排列 π ，使得环游长不超过 M

$\mathcal{NP}$ 类

组合优化

- 不在$\mathcal{NP}$ 类中的问题
  - 不可判定问题（ **Undecidable Problem**）
  - 非判定问题
  - 由时间分层定理（**time hierarchy theorem**）所给出的某些$\mathcal{NEXP}$ 类中的问题
- $\mathcal{NP}$ 类的结构
  - $\mathcal{P}$ 类中问题答案为“是”的实例对应的解可直接求出，故 $\mathcal{P} \subseteq \mathcal{NP}$

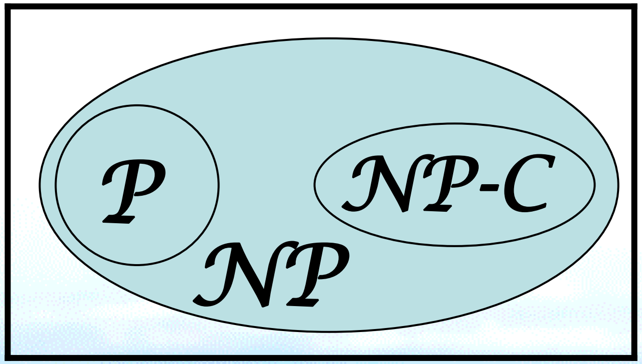
$\mathcal{NP}$ 类中最难的问题?



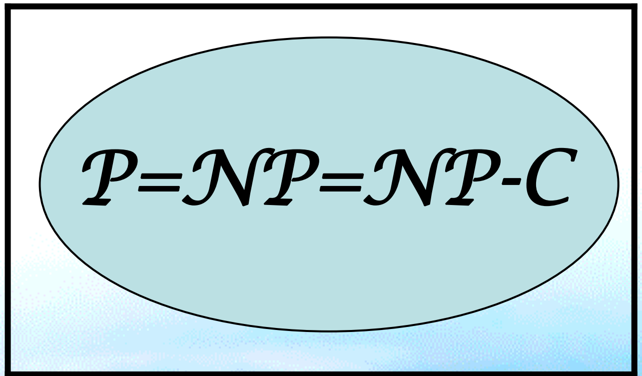
Homer S, Selman AL. ***Computability and Complexity Theory*****, Springer, 2011.**

NP-完全问题

- *NP* 类中最“难”的问题子集称为*NP* 一完全类，记为*NP-C*。*NP-C* 类中的问题称为*NP* 一完全问题（*NP* -**complete problem**）
  - 若*NP-C* 类中有一个问题有多项式时间算法，则*NP* 类中所有问题都有多项式时间算法



$P \neq NP\ (P \cup NP\text{-}C \neq NP)$



$P = NP$

P，NP，与NP—完全

组合优化

- 关于P，NP，与NP—完全问题的错误理解
  - ~~NP 问题指没有多项式时间算法的问题~~
  - ~~该问题是NP的，因此很难求解~~
  - ~~NP－完全问题是所有问题中最难求解的~~

2．P问题和NP问题

在计算机科学领域，问题一般可以分为可解问题和不可解问题。不可解问题也可以分为两类：一类如停机问题，的确无解；另一类虽然有解，但时间复杂度很高。例如，一个算法需要数月乃至数年，那肯定不能被认为是有效的算法。可解问题也分为多项式问题（Polynomial Problem，P问题）和非确定性多项式问题（Nondeterministic Polynomial Problem，NP问题）。

(2) NP问题

NP问题是指算法的时间复杂度不能使用确定的多项式来表示，通常它们的时间复杂度都是指数变量，如$O(10^n)$、$O(n!)$等。最短路径问题也类似，这类问题的时间复杂度就是上述$O(2^n)$，n是旅行商旅行要途经城市的数量。

通常，NP问题与最短路径问题类似，是一个很明显的大O指数问题。P问题已经被公认

184

P，NP，与NP—完全

浙江大学 ZheJiang University

组合优化

TEACHER'S BOOK　普通高级中学实验教科书 数学3（必修）教师教学用书

4. 计算的复杂性

计算的复杂性测度函数有三类：一类是指数型的，常写为 c^n 的形式（c 是常数）；
的，常写为 n^k 的形式（k 为非负整数）；另一类是对数型的，常写为 $\log n$ 的形式．
问题分别称为指数复杂性，多项式复杂性和对数复杂性．

人们习惯于把理论上可计算的问题类称为能行可计算的，而把具有多项式复杂性
效可计算的，通常称为 P 问题．NP 问题是指还未找到多项式复杂性算法的问题．

研究和实验表明，单纯靠提高计算机速度并不能解决 NP 问题．例如，根据某些
录，对于复杂性为 2^n 的问题，即使计算机速度提高 1 000 倍，也只能是多算约 10 道
为，解决 NP 问题的关键是要从数学上找出好的算法．事实上，数学家们也找到了各种各样的好办法，
大大简化了计算．例如，用计算机对卫星照片进行处理，如果在一张 10 cm² 的照片上以一微米为间隙
打上格子，则处理一张照片需要进行 10^{16} 次运算，即使用每秒百亿次的计算机也要连续算上十多个昼夜．
后来，有人发明了一种好的算法，大大降低了计算的复杂性，使得用同样的计算机计算只需要 $\frac{1}{3}$ 秒．

归约

- 设有判定问题 Π_1, Π_2，若对 Π_1 的任一实例 I_1，可在多项式时间内构造出 Π_2 的一个实例 I_2，使得 I_1 的答案为"是"当且仅当 I_2 的答案为"是"，则称 Π_1 可多项式时间归约到 Π_2，记为 $\Pi_1 \le_m^p \Pi_2$
- 若 $\Pi_1 \le_m^p \Pi_2$，则 Π_2 不会比 Π_1 更容易
  - 可以用求解 Π_2 的多项式时间算法设计出求解 Π_1 的多项式时间算法，但反之未必成立
- 归约作为两问题间的一种关系具有传递性

归约

Π_1 的实例 I_1

构造

如何对任意的 I_1 构造出相应的 I_2

Π_2 的实例 I_2

调用 Π_2 的算法 A_2

I_2 的答案

反馈

如何使 I_1 和 I_2 的答案恰相同

I_1 的答案

Π_1 的算法 A_1

Hamiltion圈

浙江大学 ZheJiang University

组合优化

- 经过图的所有顶点恰好一次的圈称为 **Hamilton圈**（**Hamiltion cycle**）。存在**Hamilton**圈的图称为**Hamilton图**
- **Hamilton图问题（HC）**：判断图 G 是否为一**Hamilton**图

Hamilton图问题是图论中最重要的问题之一。图论中有很多判别**Hamilton**图的充分/必要条件和对不同类型特殊图是否为**Hamilton**图的讨论。在计算复杂性理论中重点关注**Hamilton**图判别算法的复杂性

William Rowan Hamilton

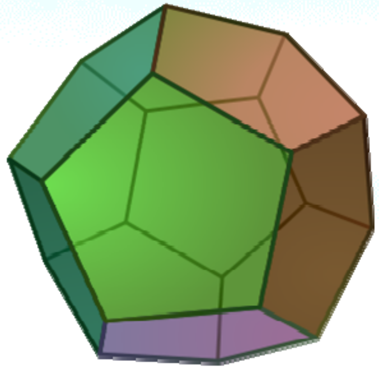
爱尔兰数学家

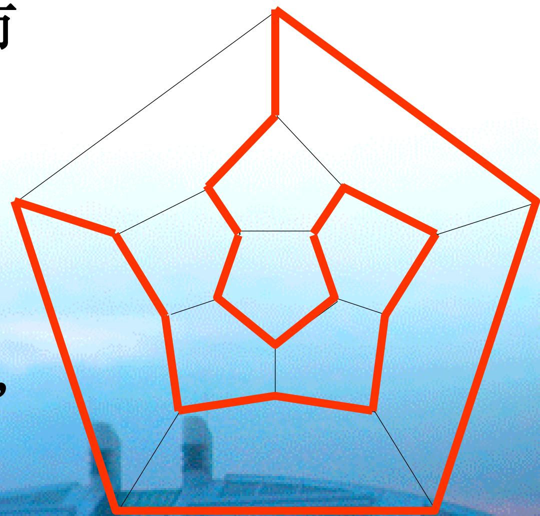
(1805-1865)

组合优化

周游世界

- **1859年Hamilton发明了周游世界的游戏icosian game**
  - 一个正十二面体的二十个顶点各代表一个城市，是否有一条从某个城市出发，沿正十二面体的棱行走，经过每个城市恰好一次，最后回到出发城市的路线



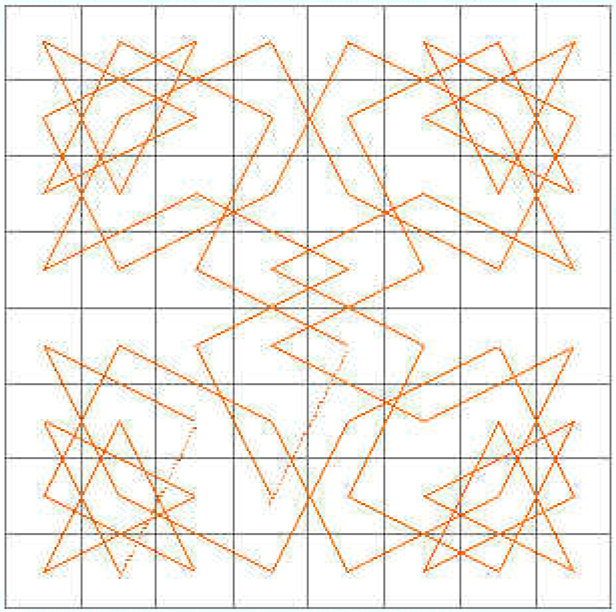


Amsterdam, Ann Arbor, Berlin, Budapest, Dublin, Edinburgh, Jerusalem, London, Melbourne, Moscow, Novosibirsk, New York, Paris, Peking, Prague, Rio di Janeiro, Rome, San Francisco, Tokyo, Warsaw

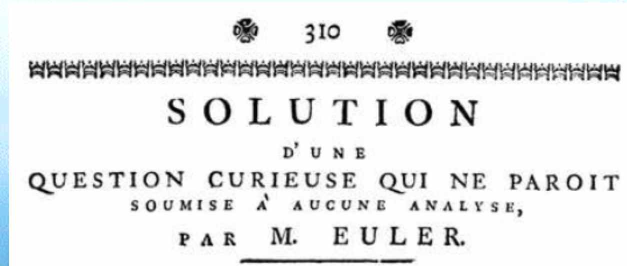
Knight's tour

组合优化

- 在 8×8 国际象棋棋盘上，马能否按其走子规则，从一个格子出发，经过其它格子恰好一次，最后回到起点
  - 构造“跳马图”，每一格子为图的一个顶点，两个格子之间有边相连当且仅当马可按走子规则从一个格子跳到另一个格子
- $m\times n\ (m\le n)$ 方格棋盘对应的“跳马图”为**Hamiltonian**图，除非
  - m, n 均为奇数
  - 或 $m = 1, 2, 4$
  - 或 $m = 3, n = 4, 6, 8$



Euler, L., Solution of a curious question which does not seem to have been subjected to any analysis, *Mémoires de l'Academie Royale des Sciences et Belles Lettres*, 15, 310–337, 1759


310

SOLUTION
D'UNE
QUESTION CURIEUSE QUI NE PAROIT
SOUMISE À AUCUNE ANALYSE,
PAR M. EULER.

HC $\leq_m^p$ TSP

浙江大学 ZheJiang University

组合优化

- 任给**HC**问题的实例 I_{HC}：图 $G=(V,E)$
- 构造**TSP**判定形式的实例 I_{TSP}
  - 城市数 $n=|V|$
  - 城市间距离 $c_{ij}=\begin{cases}1, \text{若 } (v_i,v_j)\in E, \\ 2, \text{若 } (v_i,v_j)\notin E,\end{cases} \quad i\neq j$
  - 整数阈值 $M=|V|$

图的顶点与城市一一对应。若两个顶点之间有边相连，对应城市间距离为**1**；若两个顶点之间无边相连，对应城市间距离为2

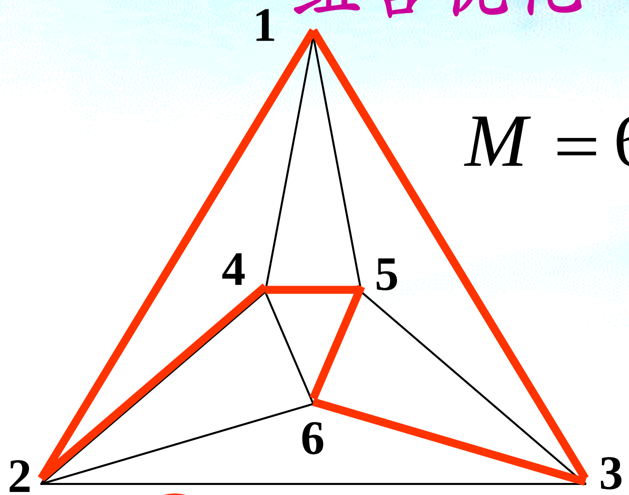
- I_{HC} 的答案为“是”当且仅当 I_{TSP} 的答案也为“是”
  - G 中存在一**Hamilton**圈当且仅当存在总长度不超过 $|V|$ 的环游

HC $\leq_m^p$ TSP

组合优化

- 若 G 中存在一**Hamilton** 圈，则按该圈经过各顶点顺序依次到达每个城市， 圈中每条边的两个端点所对应的两个城市间距离均为**1**，环游总长度恰为 $|V|$
- 若环游总长度不超过 $|V|$，由于环游需经过 $|V|$ 个城市，环游中相邻两个城市间距离均为**1**，它们所对应的 G 中两个顶点之间均有边相连，所有这些边恰组成 G 的一个**Hamilton**圈

$M = 6$



$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 2 \\ 1 & 0 & 1 & 1 & 2 & 1 \\ 1 & 1 & 0 & 2 & 1 & 1 \\ 1 & 1 & 2 & 0 & 1 & 1 \\ 1 & 2 & 1 & 1 & 0 & 1 \\ 2 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$\mathcal{NP}$-完全问题

组合优化

- 若 $\Pi \in \mathcal{NP}$，且对任意的 $\Pi' \in \mathcal{NP}, \Pi' \le_m^p \Pi$，则称 Π 是$\mathcal{NP}$ －完全问题（$\mathcal{NP}$- **complete problem**）
- 所有$\mathcal{NP}$ －完全问题的集合称为$\mathcal{NP}$ －完全类，记为$\mathcal{NP}$-$\mathcal{C}$
  - $\mathcal{NP}$-$\mathcal{C}$ 类是$\mathcal{NP}$类的一个子类，包含了$\mathcal{NP}$ 类中最“难”的问题
  - 若$\mathcal{NP}$-$\mathcal{C}$ 类中有一个问题有多项式时间算法，则$\mathcal{NP}$ 类中所有问题都有多项式时间算法

从定义出发证明一问题的$\mathcal{NP}$ －完全性是困难的

$\mathcal{NP}$ －完全性判定定理

- $\mathcal{NP}$ －完全性判定定理
  - 若 $\Pi \in \mathcal{NP}$，且存在某个 $\Pi^C \in \mathcal{NP}\text{-}C$，$\Pi^C \leq_m^p \Pi$，则 $\Pi \in \mathcal{NP}\text{-}C$
    - 任取 $\Pi' \in \mathcal{NP}$，由于 $\Pi^C \in \mathcal{NP}\text{-}C$，故由$\mathcal{NP}$ －完全问题的定义，$\Pi' \leq_m^p \Pi^C$
    - 由定理条件和归约的传递性 $\Pi' \leq_m^p \Pi$
    - 由Π'的任意性和$\mathcal{NP}$ －完全问题的定义，$\Pi \in \mathcal{NP}\text{-}C$

第一个$\mathcal{NP}$—完全问题

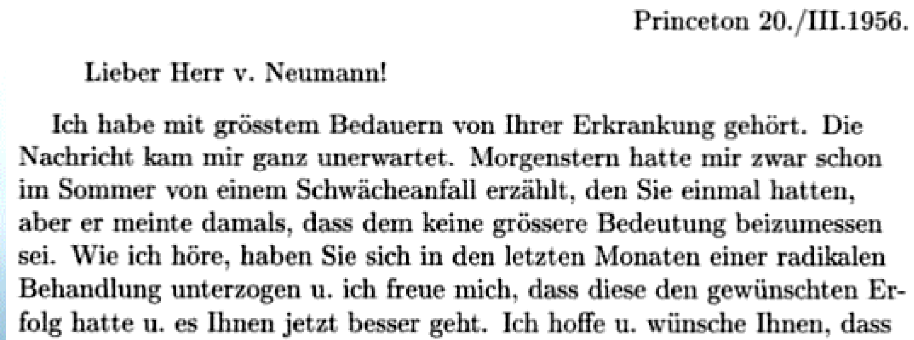
浙江大学
ZheJiang University

组合优化

数理逻辑

- 数理逻辑（**mathematical logic**）：用数学的方法研究逻辑推理和数学计算，将推理论证、数学计算的过程符号化、形式化、公理化的学科

1956年**Gödel**致**von Neumann**信，信中对若干数理逻辑问题算法和复杂性的讨论被认为是计算复杂性研究的开端



Princeton 20./III.1956.

Lieber Herr v. Neumann!

Ich habe mit grösstem Bedauern von Ihrer Erkrankung gehört. Die Nachricht kam mir ganz unerwartet. Morgenstern hatte mir zwar schon im Sommer von einem Schwächeanfall erzählt, den Sie einmal hatten, aber er meinte damals, dass dem keine grössere Bedeutung beizumessen sei. Wie ich höre, haben Sie sich in den letzten Monaten einer radikalen Behandlung unterzogen u. ich freue mich, dass diese den gewünschten Erfolg hatte u. es Ihnen jetzt besser geht. Ich hoffe u. wünsche Ihnen, dass

Kurt Friedrich Gödel
（1906－1978 ）
奥地利哲学家、数学家

John von Neumann
（1903－1957）
匈牙利裔美国科学家

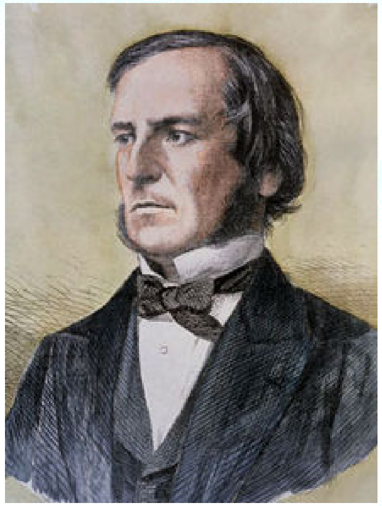
Boolean变量

组合优化

- 仅可取“真” 和“假”（True(T)和False(F)，1和0）两种值的变量称为**Boolean变量**
- **Boolean变量的运算**
  - 非（ **negation**）$\neg$ ： $\neg x = 1 \Leftrightarrow x = 0$
  - 析取（ **disjunction** ）$\vee$

    $$x_1 \vee x_2 = 1 \Leftrightarrow x_1 = 1 \text{或} x_2 = 1$$
  - 合取（ **conjunction** ）$\wedge$

    $$x_1 \wedge x_2 = 1 \Leftrightarrow x_1 = 1 \text{且} x_2 = 1$$
- **Boolean变量的运算遵从**双重否定律、 **De Morgan**律、析取对合取的分配率、合取对析取的分配率等运算定律



George Boole
（1815-1864）
英国数学家、哲学家、逻辑学家

Boolean表达式

- 若干**Boolean**变量用运算符和括号按一定的逻辑关系联结起来的表达式称为**Boolean表达式**（**Boolean expression**）
- 对出现在**Boolean**表达式中的所有**Boolean**变量各指定一个值，可按**Boolean**变量的运算法则确定表达式的值**1**或**0**
- 任一**Boolean**表达式都存在与之等价的合取范式（**conjunctive normal form, CNF**）
  - 文字（ **literal** ）：变量或变量的非
  - 子句（ **clause** ）：若干个文字的析取
  - **CNF**：若干个子句的合取

$$\neg(\neg x_1 \vee x_2) \vee x_3 \Leftrightarrow (\neg\neg x_1 \wedge \neg x_2) \vee x_3 \Leftrightarrow (x_1 \wedge \neg x_2) \vee x_3$$

$$\Leftrightarrow (x_1 \vee x_3) \wedge (\neg x_2 \vee x_3)$$

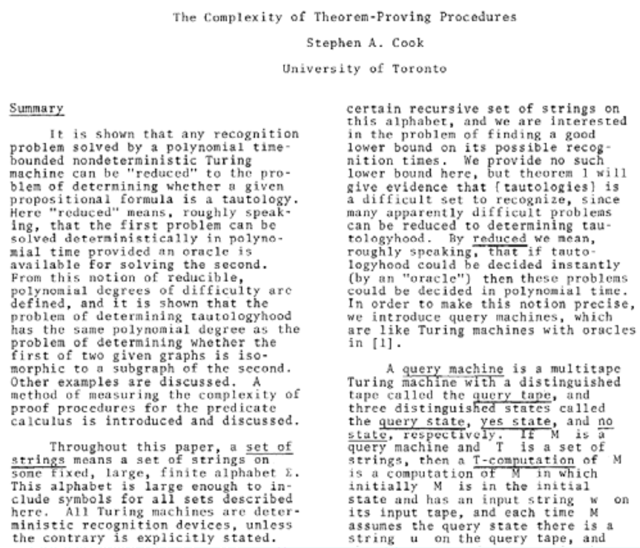
SAT

组合优化

- 可满足性问题（ **Satisfiability**，**SAT**）
  - 给定一合取范式，问是否存在其变量的一种赋值，使得该表达式值为真
    - $(x_1 \vee \neg x_2) \wedge \neg x_2$： x_1 取**1**，x_2 取**0** 可使表达式值为**1**
    - $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge \neg x_1$：不论 x_1, x_2 取值为何值，表达式值均为 **0**
  - 猜想所有变量的一种赋值，在多项式时间内可验证表达式值确为**1**，因此 $\mathrm{SAT} \in \mathcal{NP}$

SAT

- **1971年，Cook运用图灵机语言，通过$\mathcal{NP}$问题的一种等价定义，用$\mathcal{NP}$－完全问题的定义证明了SAT问题的$\mathcal{NP}$－完全性**
- **SAT问题被认为是第一个$\mathcal{NP}$－完全问题**



The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

Summary

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

Throughout this paper, a set of strings means a set of strings on some fixed, large, finite alphabet Σ. This alphabet is large enough to include symbols for all sets described here. All Turing machines are deterministic recognition devices, unless the contrary is explicitly stated.

certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By reduced we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].

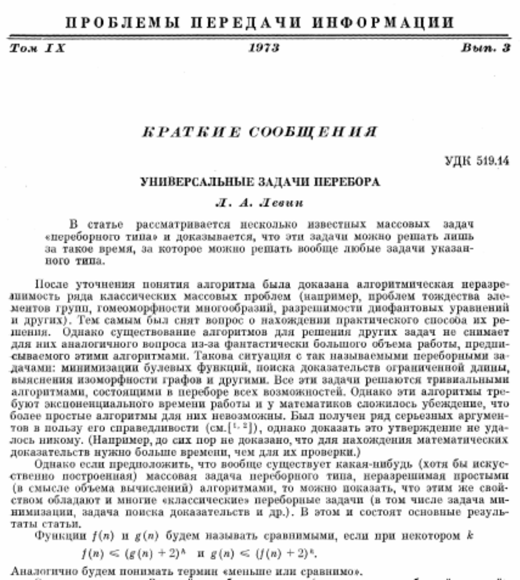
A query machine is a multitape Turing machine with a distinguished tape called the query tape, and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a set of strings, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and

Cook SA. The complexity of theorem-proving procedures, *Proceedings of the 3rd annual ACM Symposium on Theory of Computing*, 151-158, 1971.

Cook-Levin定理

组合优化

- 与**Cook**同时，**Levin**独立地给出了若干$\mathcal{NP}$－完全问题，SAT $\in \mathcal{NP}\text{-}C$ 因此被称作**Cook-Levin**定理



ПРОБЛЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

Том IX *1973* *Вып. 3*

КРАТКИЕ СООБЩЕНИЯ

УДК 519.14

УНИВЕРСАЛЬНЫЕ ЗАДАЧИ ПЕРЕБОРА

Л. А. Левин

В статье рассматривается несколько известных массовых задач «переборного типа» и доказывается, что эти задачи можно решать лишь за такое время, за которое можно решать вообще любые задачи указанного типа.

После уточнения понятия алгоритма была доказана алгоритмическая неразрешимость ряда классических массовых проблем (например, проблем тождества элементов групп, гомеоморфности многообразий, разрешимости диофантовых уравнений и других). Тем самым был снят вопрос о нахождении практического способа их решения. Однако существование алгоритмов для решения других задач не снимает для них аналогичного вопроса из-за фантастически большого объема работы, предписываемого этими алгоритмами. Такова ситуация с так называемыми переборными задачами: минимизации булевых функций, поиска доказательств ограниченной длины, выяснения изоморфности графов и другими. Все эти задачи решаются тривиальными алгоритмами, состоящими в переборе всех возможностей. Однако эти алгоритмы требуют экспоненциального времени работы и у математиков сложилось убеждение, что более простые алгоритмы для них невозможны. Был получен ряд серьезных аргументов в пользу его справедливости (см.[1, 2]), однако доказать это утверждение не удалось никому. (Например, до сих пор не доказано, что для нахождения математических доказательств нужно больше времени, чем для их проверки.)

Однако если предположить, что вообще существует какая-нибудь (хотя бы искусственно построенная) массовая задача переборного типа, неразрешимая простыми (в смысле объема вычислений) алгоритмами, то можно показать, что этим же свойством обладают и многие «классические» переборные задачи (в том числе задача минимизации, задача поиска доказательств и др.). В этом и состоят основные результаты статьи.

Функции $f(n)$ и $g(n)$ будем называть сравнимыми, если при некотором k

$$f(n) \leqslant (g(n)+2)^k \quad \text{и} \quad g(n) \leqslant (f(n)+2)^k.$$

Аналогично будем понимать термин «меньше или сравнимо».

Levin LA, Universal Sequential Search Problems, *Problems of Information Transmission*, 9, 115–116, 1973

Trakhtenbrot BA, A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6, 384-400, 1984

Leonid Anatolievich Levin

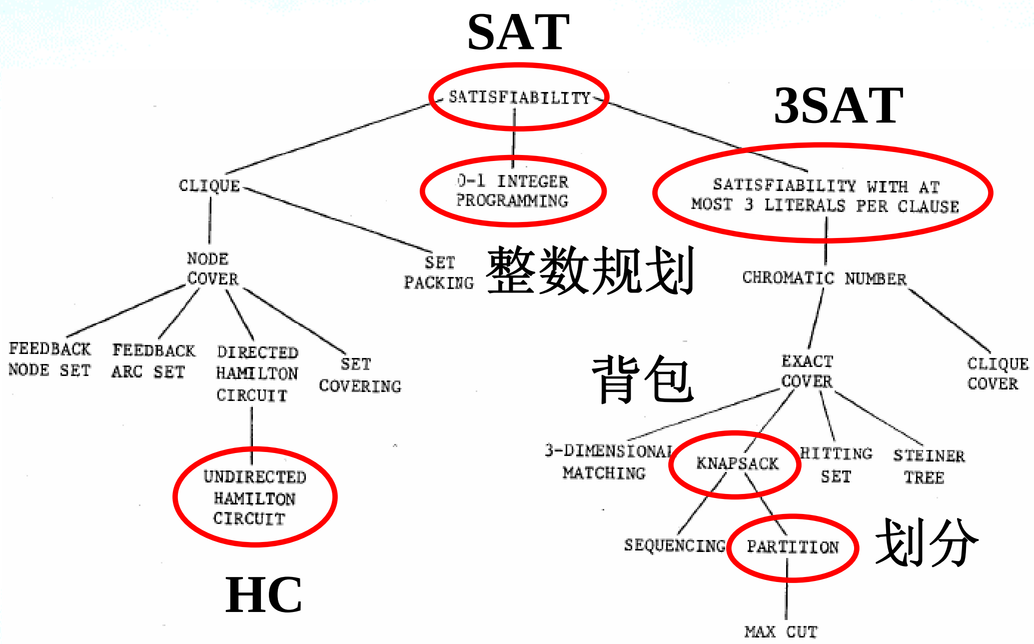
（1948－ ）

苏联计算机学家

浙江大学 ZheJiang University

组合优化

Karp归约



REDUCIBILITY AMONG COMBINATORIAL PROBLEMS†

Richard M. Karp

University of California at Berkeley

Abstract: A large class of computational problems involve the determination of properties of graphs, digraphs, integers, arrays of integers, finite families of finite sets, boolean formulas and elements of other countable domains. Through simple encodings from such domains into the set of words over a finite alphabet these problems can be converted into language recognition problems, and we can inquire into their computational complexity. It is reasonable to consider such a problem satisfactorily solved when an algorithm for its solution is found which terminates within a number of steps bounded by a polynomial in the length of the input. We show that a large number of classic unsolved problems of covering, matching, packing, routing, assignment and sequencing are equivalent, in the sense that either each of them possesses a polynomial-bounded algorithm or none of them does.

Karp RM. Reducibility Among Combinatorial Problems, *Proceedings of a Symposium on the Complexity of Computer Computations*, 85-103, 1972

$\mathcal{NP}$ －完全性证明

浙江大学 ZheJiang University

组合优化

- 问题 Π 的$\mathcal{NP}$ －完全性证明
  - 证明 $\Pi \in \mathcal{NP}$
  - 寻找与 Π 联系紧密，且实例结构较为**规范**、**简明**的已知$\mathcal{NP}$ －完全问题 Π^C
  - 证明 $\Pi^C \le_m^p \Pi$
    - 任取 Π^C的实例 I_1，构造 Π 的实例 I_2
    - 若 I_1 的答案为“是”，I_2 的答案也为“是”
    - 若 I_2 的答案为“是”， I_1 的答案也为“是”（或：若 I_1 的答案为“否”，I_2 的答案也为“否”）

3SAT

浙江大学
ZheJiang University
组合优化

- **3SAT：在SAT问题的描述中，限定任一子句包含文字数不超过3**

任取SAT问题的一实例

$$
\begin{aligned}
&(l_{i_{11}} \vee l_{i_{12}} \vee \cdots \vee l_{i_{1 l_1}}) \\
&\wedge (l_{i_{21}} \vee l_{i_{22}} \vee \cdots \vee l_{i_{2 l_2}}) \\
&\wedge \cdots \wedge (l_{i_{k1}} \vee l_{i_{k2}} \vee \cdots \vee l_{i_{k l_k}})
\end{aligned}
$$

任取3SAT问题的一实例

$$
\begin{aligned}
&(l_{i_{11}} \vee l_{i_{12}} \vee l_{i_{13}}) \\
&\wedge (l_{i_{21}} \vee l_{i_{22}} \vee l_{i_{23}}) \\
&\wedge \cdots \wedge (l_{i_{k1}} \vee l_{i_{k2}} \vee l_{i_{k3}})
\end{aligned}
$$

- **2SAT** $\in \mathcal{P}$

Krom MR. The Decision Problem for a Class of First-Order Formulas in which all Disjunctions are Binary. *Mathematical Logic Quarterly*, 13, 15-20, 1967.

$\mathrm{SAT} \le_m^p \mathrm{3SAT}$

- **任取SAT问题一实例** $F_{SAT} = c_1 \wedge c_2 \wedge \cdots \wedge c_m$，**构造3SAT问题的实例** F_{3SAT}
  - 若 c_i 中所含文字数不超过**3**，则 F_3 也 包含子句 c_i
  - 若 $c_i = l_{i_1} \vee l_{i_2} \vee \cdots \vee l_{i_k}$ ，则令

$$C_i = (l_{i_1} \vee l_{i_2} \vee y_{i1}) \wedge (\neg y_{i1} \vee l_{i_3} \vee y_{i2}) \wedge (\neg y_{i2} \vee l_{i_4} \vee y_{i3})$$

$$\wedge \cdots \wedge (\neg y_{i,k-4} \vee l_{i_{l-2}} \vee y_{i,k-3}) \wedge (\neg y_{i,k-3} \vee l_{i_{k-1}} \vee l_{i_k})$$

其中 $y_{i1}, y_{i2} \cdots y_{i,k-3}$ 是新变量且不出现在别处。F_3 中包含 C_i 中所有子句

$\text{SAT} \leq_m^p \text{3SAT}$

浙江大学 ZheJiang University

组合优化

- 若 F_{SAT} 答案为“是”，则存在一种赋值，使得任一子句 c_i 为真，因此 c_i 中文字至少有一个为真，不妨设为 l_{i_j}
- 存在一种赋值，使得 F_{3SAT} 中每个子句均为真，F_{3SAT} 答案也为“是”

$$c_i = l_{i_1} \vee l_{i_2} \vee \cdots \vee \overset{1}{l_{i_j}} \vee \cdots \vee l_{i_k}$$

F_{SAT} 中原有变量赋值保持不变

$$\begin{aligned} C_i = {} & (l_{i_1} \vee l_{i_2} \vee \overset{1}{y_{i1}}) \wedge (\neg \overset{0}{y_{i1}} \vee l_{i_3} \vee \overset{1}{y_{i2}}) \wedge \cdots \wedge (\neg \overset{0}{y_{i,j-4}} \vee l_{i_{j-2}} \vee \overset{1}{y_{i,j-3}}) \\ & \wedge (\neg \overset{0}{y_{i,j-3}} \vee l_{i_{j-1}} \vee \overset{1}{y_{i,j-2}}) \wedge (\neg \overset{0}{y_{i,j-2}} \vee \overset{1}{l_{i_j}} \vee \overset{0}{y_{i,j-1}}) \wedge (\neg \overset{1}{y_{i,j-1}} \vee l_{i_{j+1}} \vee \overset{0}{y_{i,j}}) \\ & \wedge (\neg \overset{1}{y_{i,j}} \vee l_{i_{j+2}} \vee \overset{0}{y_{i,j+1}}) \wedge \cdots \wedge (\neg \overset{1}{y_{i,k-4}} \vee l_{i_{k-2}} \vee \overset{0}{y_{i,k-3}}) \wedge (\neg \overset{1}{y_{i,k-3}} \vee l_{i_{k-1}} \vee l_{i_k}) \end{aligned}$$

$\text{SAT} \le_m^p \text{3SAT}$

组合优化

- 若F_{3SAT}答案为"是"，则存在一种赋值，使得 F_{3SAT} 中每个子句均为真
- 将该赋值限制到 F_{SAT} 中的变量，若存在一子句 c_i 为假，则 c_i 中所有文字均为假
- C_i中有子句值为假，矛盾，故c_i中所有子句均为真，F_{SAT}答案也为"是"

$$c_i = \overset{0}{l_{i_1}} \vee \overset{0}{l_{i_2}} \vee \cdots \vee \overset{0}{l_{i_j}} \vee \cdots \vee \overset{0}{l_{i_k}}$$

$$C_i = (\overset{0}{l_{i_1}} \vee \overset{0}{l_{i_2}} \vee \overset{1}{y_{i1}}) \wedge (\overset{0}{\neg y_{i1}} \vee \overset{0}{l_{i_3}} \vee \overset{1}{y_{i2}}) \wedge (\overset{0}{\neg y_{i2}} \vee \overset{0}{l_{i_4}} \vee \overset{1}{y_{i3}}) \wedge \cdots$$

$$\wedge \cdots \wedge (\overset{0}{\neg y_{i,k-5}} \vee \overset{0}{l_{i_{k-3}}} \vee \overset{1}{y_{i,k-4}}) \wedge (\overset{0}{\neg y_{i,k-4}} \vee \overset{0}{l_{i_{k-2}}} \vee \overset{1}{y_{i,k-3}}) \wedge (\overset{0}{\neg y_{i,k-3}} \vee \overset{0}{l_{i_{k-1}}} \vee \overset{0}{l_{i_k}})$$

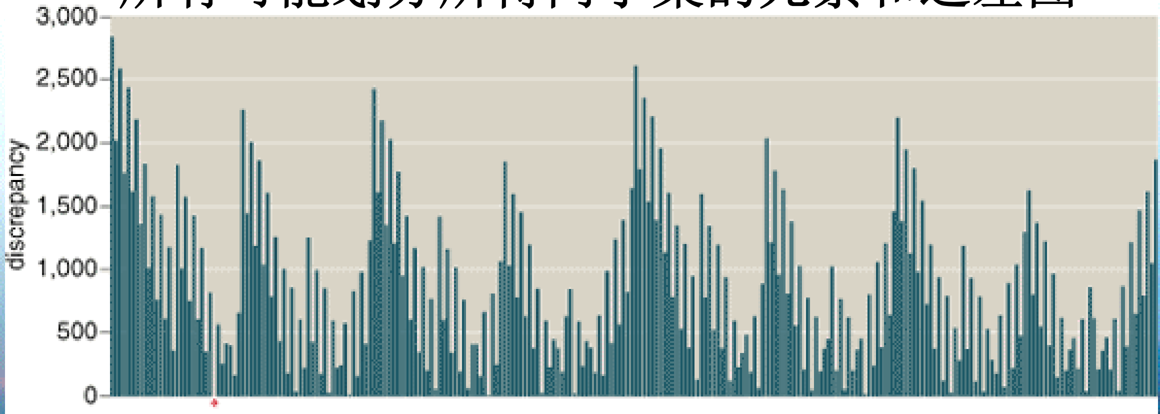
划分问题

- 划分问题（**Partition**）
  - 给定一正整数集 $A=\{a_1,a_2,\cdots,a_n\}$，问是否存在子集 A_1,A_2，使得 $A=A_1\cup A_2$，$A_1\cap A_2=\phi$，且满足 $\sum_{a_i\in A_1} a_i=\sum_{a_i\in A_2} a_i=\frac{1}{2}\sum_{j=1}^{n} a_j$

$$A=\{484,114,205,288,506,503,201,127,410\}$$

$$A_1=\{410,506,503\},A_2=\{484,114,205,288,201,127\}$$

所有可能划分所得两子集的元素和之差图



COMPUTING SCIENCE

THE EASIEST HARD PROBLEM

Brian Hayes

One of the cherished customs of childhood is choosing up sides for a ball game. Where I grew up, we did it this way: The two chief bullies of the neighborhood would appoint themselves captains of the opposing teams, and then they would take turns picking other players. On each round, a captain would choose the most capable (or, toward the end, the least inept) player from the pool of remaining candidates, until everyone present had been assigned to one side or the other. The aim of this ritual was to produce two evenly matched teams and, along the way, to remind each of us of our precise ranking in the neighborhood pecking order. It usually worked.

None of us in those days—not the hopefuls waiting for our name to be called, and certainly not the two thick-necked team leaders—recognized that our scheme for choosing sides implements a greedy heuristic for the balanced number partitioning problem. And we had no idea that this problem is NP-complete—that finding the optimum team rosters is certifiably hard. We just wanted to get on with the game.

And therein lies a paradox: If computer scientists find the partitioning problem so intractable, how come children the world over solve it every day? Are the kids *that* much smarter? Quite possibly they are. On the other hand, the success of playground algorithms for partitioning might be a clue that the task is not always as hard as that forbidding term "NP-complete" tends to suggest. As a matter of fact, finding a hard instance of this famously hard problem can be a hard problem—unless you know where to look. Some recent re-

jobs into two sets with equal running time will balance the load on the processors. Another example is apportioning the miscellaneous assets of an estate between two heirs.

So What's the Problem?

Here is a slightly more formal statement of the partitioning problem. You are given a set of *n* positive integers, and you are asked to separate them into two subsets; you may put as many or as few numbers as you please in each of the subsets, but you must make the sums of the subsets as nearly equal as possible. Ideally, the two sums would be exactly the same, but this is feasible only if the sum of the entire set is even; in the event of an odd total, the best you can possibly do is to choose two subsets that differ by 1. Accordingly, a perfect partition is defined as any arrangement for which the "discrepancy"—the absolute value of the subset difference—is no greater than 1.

Try a small example. Here are 10 numbers—enough for two basketball teams—selected at random from the range between 1 and 10:

2 10 3 8 5 7 9 5 3 2

Can you find a perfect partition? In this instance it so happens there are 23 ways to divvy up the numbers into two groups with exactly equal sums (or 46 ways if you count mirror images as distinct partitions). Almost any reasonable method will converge on one of these perfect solutions. This is the answer I stumbled onto first:

(2 5 3 10 7) (2 5 3 9 8)

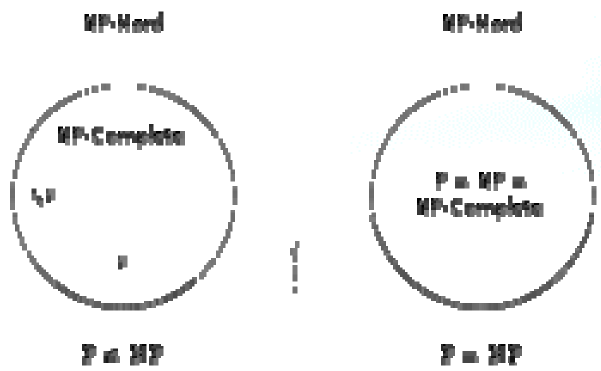
Both subsets sum to 27.

Hayes B. The easiest hard problem. ***American Scientist*, 90(2), 113-117, 2002.**

$\mathcal{NP}$ -难

组合优化

- 至少和$\mathcal{NP}$ -完全问题一样难的问题称为$\mathcal{NP}$ -难（$\mathcal{NP}$ -hard）问题
  - 在 $\mathcal{P} \neq \mathcal{NP}$ 的假设下，$\mathcal{NP}$ -难问题也没有多项式时间算法
  - 若一优化问题的判定形式是$\mathcal{NP}$ -完全的，则该优化问题是$\mathcal{NP}$ -难的



子问题

组合优化

- 在某问题 Π 的实例构成上增加一些限制就得到的一个子问题（**subproblem**）Π'
  - Π'的实例集包含在Π的实例集中，Π'的答案为“是”（“否”）的实例集恰为Π的答案为“是”（“否”）的实例集与 Π'的实例集之交
- 子问题的计算复杂性
  - 若 $\Pi \in \mathcal{P}$，则 $\Pi' \in \mathcal{P}$，但反之不然
  - 若 Π'是$\mathcal{NP}$-完全的，则 Π 也是$\mathcal{NP}$-完全的，但反之不然
  - 希望寻找“最特殊”的$\mathcal{NP}$–完全子问题和“最一般”的$\mathcal{P}$子问题

子集和问题

- 子集和问题（**Subset Sum**）
  - 给定正整数集 $A=\{a_1,a_2,\cdots,a_n\}$ 和数 B，问是否存在子集 $A_1\subseteq A$，使得 $\sum_{a_i\in A_1} a_i = B$
  - 取 $B=\frac{1}{2}\sum_{j=1}^{n} a_j$，划分问题成为子集和问题的子问题
- 子集和问题的优化形式

子集和是 $\mathcal{NP}$ －完全问题

  - 求子集 $A_1\subseteq A$，使得 $\sum_{a_i\in A_1} a_i \le B$ 且 $\sum_{a_i\in A_1} a_i$ 尽可能大
  - 若背包问题物品 j 的价值与大小均为 a_j，容量为 B，则子集和问题优化形式成为背包问题优化形式的子问题

背包问题判定形式是 $\mathcal{NP}$ －完全问题

子集和问题

组合优化

- 子集和问题的优化形式（极小化） 极小化背包的定义与应用
  - 求子集 $A_1 \subseteq A$，使得 $\sum_{a_i \in A_1} a_i \geq B$ 且 $\sum_{a_i \in A_1} a_i$ 尽可能小
- 第 K 个最大子集和问题
  - 给定正整数集 $A=\{a_1,a_2,\cdots,a_n\}$ 和数 B，问是否存在 A 的 K 个子集 $A_1,A_2,\cdots,A_K$，使得 $\sum_{a_i \in A_i} a_i \leq B, i=1,\cdots,K$
  - 第 K 个最大子集和问题可能不属于 $\mathcal{NP}$
    - 实例规模：$n+\max_{i=1,\cdots,n} \log a_i + \log B + \log K$
    - 可行解的规模：Kn

强$\mathcal{NP}$-完全

- 设Π为一$\mathcal{NP}$-完全问题，且存在多项式函数p，使得Π的某个所有实例满足$\text{Max}(I) \le p(\text{size}(I))$的子问题$\Pi$'是$\mathcal{NP}$-完全的，则称$\Pi$为强$\mathcal{NP}$-完全（**strongly $\mathcal{NP}$-complete**）的
- 在$\mathcal{P} \neq \mathcal{NP}$的假设下，任何强$\mathcal{NP}$-完全问题不存在伪多项式时间算法
  - 若Π存在伪多项式时间算法A，其时间复杂性为
    $$f(I) = O(\text{poly}(\text{size}(I), \text{Max}(I)))$$
  - 用A求解Π的子问题Π'，其时间复杂性为
    $$f(I) = O(\text{poly}(\text{size}(I), \text{Max}(I))) = O(\text{poly}(\text{size}(I), p(\text{size}(I)))) = O(\text{poly}(\text{size}(I)))$$
  - A为Π'的多项式时间算法，与Π'的$\mathcal{NP}$-完全性矛盾

强$\mathcal{NP}$-完全性的证明

- 问题是$\mathcal{NP}$-完全的，且其所有实例均满足 $\mathrm{Max}(I) \le p(\mathrm{size}(I))$
  - **Hamilton图问题、SAT问题**
- 归约证明某问题的$\mathcal{NP}$-完全性时所构造的该问题的实例均满足 $\mathrm{Max}(I) \le p(\mathrm{size}(I))$
  - **TSP问题的判定形式**
- 按强$\mathcal{NP}$-完全问题的定义证明
  - 寻找多项式函数 p 和所有实例满足 $\mathrm{Max}(I) \le p(\mathrm{size}(I))$ 的子问题，通过归约证明该子问题是$\mathcal{NP}$-完全的
- 用伪多项式时间归约从一个已知强$\mathcal{NP}$-完全问题出发证明某问题的强$\mathcal{NP}$-完全性

伪多项式时间归约

组合优化

- 设有判定问题 Π_1, Π_2，若对 Π_1 的任一实例 I_1，可在 $O(\mathrm{poly}(\mathrm{size}(I_1), \mathrm{Max}(I_1)))$ 时间内构造出 Π_2 的一个实例 $I_2 = f(I_1)$，使得

  实例构造的时间不太长

  - I_1 的答案为"是"当且仅当 I_2 的答案也为"是" 可以反馈
  - $\mathrm{Max}(I_2) \le p_1(\mathrm{size}(I_1), \mathrm{Max}(I_1))$，这里 p_1 为某个二元多项式

    最大数没有显著增大 实例规模没有显著缩小
  - $\mathrm{size}(I_1) \le p_2(\mathrm{size}(I_2))$，这里 p_2 为某个多项式

则称 Π_1 可伪多项式时间归约到 Π_2，记为 $\Pi_1 \le_m^{pp} \Pi_2$

伪多项式时间归约

- 若Π_1为强$\mathcal{NP}$−完全问题，$\Pi_2 \in \mathcal{NP}$且$\Pi_1 \le_m^{pp} \Pi_2$，则Π_2也是强$\mathcal{NP}$−完全问题
  - 由于Π_1为强$\mathcal{NP}$−完全问题，存在Π_1的子问题Π_1^S，其实例集$\mathcal{I}_1$中所有实例I_1满足$\mathrm{Max}(I_1) \le p(\mathrm{size}(I_1))$，且$\Pi_1^S$是$\mathcal{NP}$−完全的
  - 将以$\mathcal{I}_2 = f(\mathcal{I}_1)$为实例集的$\Pi_2$的子问题记为$\Pi_2^S$
  - 由于构造$f(\mathcal{I}_1)$可在$O(\mathrm{poly}(\mathrm{size}(I_1), \mathrm{Max}(I_1))) = O(\mathrm{poly}(\mathrm{size}(I_1)))$时间内完成，$f$也是$\Pi_1^S$到$\Pi_2^S$的多项式时间归约，即$\Pi_1^S \le_m^p \Pi_2^S$，因此$\Pi_2^S$是$\mathcal{NP}$−完全的
  - 若能证明$\mathcal{I}_2$中所有实例I_2满足$\mathrm{Max}(I_2) \le p'(\mathrm{size}(I_2))$，则由强$\mathcal{NP}$−完全问题的定义可知$\Pi_2$是强$\mathcal{NP}$−完全的

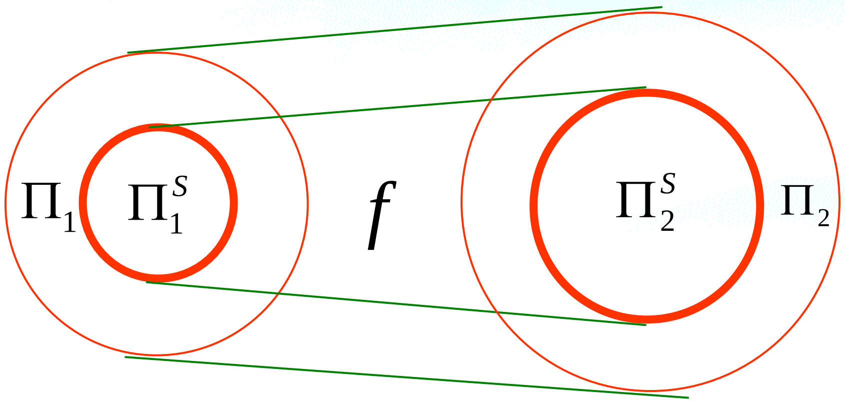
组合优化

伪多项式时间归约

$$\begin{aligned}\mathrm{Max}(I_2) &\le p_1(\mathrm{size}(I_1), \mathrm{Max}(I_1))\\ &\le p_1(\mathrm{size}(I_1), p(\mathrm{size}(I_1)))\\ &\le p_1'(\mathrm{size}(I_1))\\ &\le p_1'(p_2(\mathrm{size}(I_2)))\\ &\le p'(\mathrm{size}(I_2))\end{aligned}$$

若 $\mathrm{size}(I_1)=\Omega(2^{\mathrm{size}(I_2)})$，
即 $\mathrm{size}(I_2)=O(\log(\mathrm{size}(I_1)))$，
构造的实例规模显著缩小，
从而 $\mathrm{Max}(I_2)\le p'(\mathrm{size}(I_2))$ 未必成立，未能找到满足要求的 $\mathcal{NP}-$完全的子问题

第一个强 $\mathcal{NP}-$ 完全问题



$\mathrm{Max}(I_1)\le p(\mathrm{size}(I_1))$ $\quad$ $\mathrm{Max}(I_2)\le p'(\mathrm{size}(I_2))$?

$$\boxed{\begin{array}{c}\mathrm{Max}(I_2)\le p_1(\mathrm{size}(I_1),\mathrm{Max}(I_1))\\ \mathrm{size}(I_1)\le p_2(\mathrm{size}(I_2))\end{array}}$$

上述两个条件容易验证，实践中多数构造都能满足

3-划分问题

- **3-划分（3-partioning）**
  - 给定由 $3m$ 个正整数组成的集合 $A=\{a_1,a_2,\cdots,a_{3m}\}$ 和正整数 B，其中 $\frac{B}{4}<a_j<\frac{B}{2}, j=1,2,\cdots,3m, \sum_{j=1}^{3m}a_j=mB$，问 A 是否可划分成 m 个互不相交的集合 $A_1,A_2,\cdots,A_m$，使得对任意的 $i, 1\le i\le m, \sum_{a_j\in A_i}a_j=B$

6 8 10 17 19

3 5 7 9 11 12 13

划分

10 13 17

3 7 11 19

5 6 8 9 12

划分

103 108 119

105 112 113

106 107 117

109 110 111

3-划分

普通$\mathcal{NP}$-完全

浙江大学 ZheJiang University

组合优化

- 若 Π 为$\mathcal{NP}$-完全问题，且存在伪多项式时间算法，则 Π 不可能是强$\mathcal{NP}$-完全的，此时称 Π 为普通意义下的$\mathcal{NP}$-完全问题（$\mathcal{NP}$-**complete in the ordinary sense**）
  - 背包、划分均为普通意义下的$\mathcal{NP}$-完全问题
- 至少和强$\mathcal{NP}$-完全问题一样难的问题称为强$\mathcal{NP}$-难（**strongly** $\mathcal{NP}$-**hard**）问题
  - **TSP**为强$\mathcal{NP}$-难问题

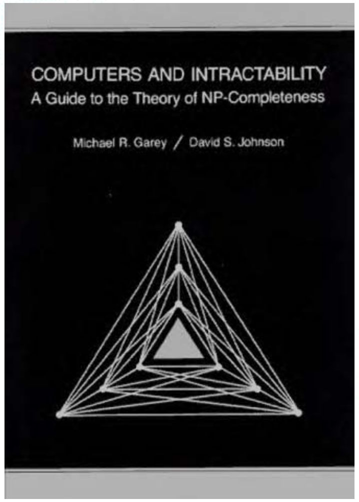
空间复杂性

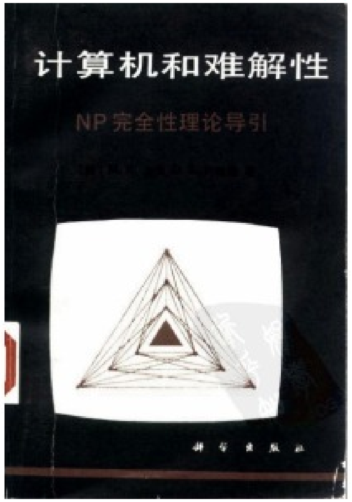
- 算法的空间复杂度（**space complexity**）是关于实例规模 n 的一个函数 $f(n)$，它表示用该算法求解所有规模为 n 的实例中算法使用空间量最多的那个实例算法使用的空间量
- 空间复杂度类
  - $\mathcal{PSPACE}$，$\mathcal{NPSPACE}$，$\mathcal{EXPSPACE}$，$\mathcal{PSPACE}$ -完全

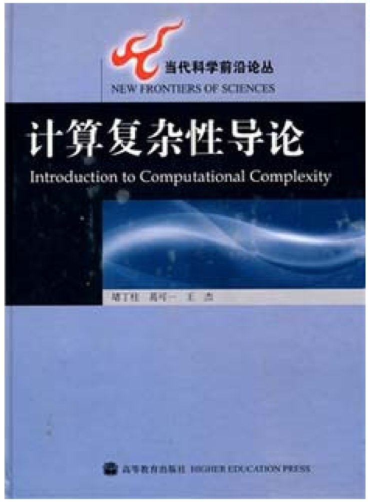
$$\mathcal{P} \underset{?}{\subseteq} \mathcal{NP} \underset{?}{\subseteq} \mathcal{PSPACE} \underset{?}{\subseteq} \mathcal{EXP} \underset{?}{\subseteq} \mathcal{NEXP} \underset{?}{\subseteq} \mathcal{EXPSPACE}$$

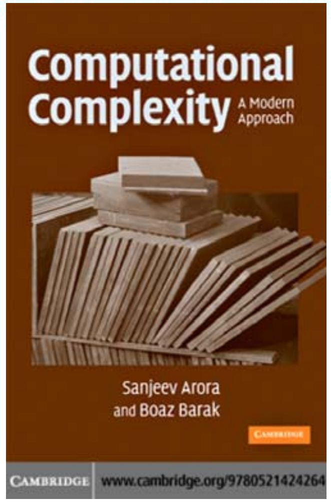
$$\mathcal{PSPACE} = \mathcal{NPSPACE}$$

参考资料









Garey MR, Johnson DS. ***Computers and intractability: a guide to the theory of NP-completeness*. Freeman, 1979.** （中译本：计算机和难解性： NP 完全性理论导引. 张立昂，沈泓译，科学出版社，1990.）

堵丁柱，葛可一，王杰，计算复杂性导论，高等教育出版社，**2002.**

Arora S, Barak B. ***Computational complexity: a modern approach*. Cambridge University Press, 2009.**

$\mathcal{NP}$ 优化问题汇编

浙江大学
ZheJiang University

组合优化

A compendium of NP optimization problems

Editors:
Pierluigi Crescenzi, and **Viggo Kann**
Subeditors:
Magnús Halldórsson (retired)
Graph Theory: Covering and Partitioning, Subgraphs and Supergraphs, *Sets and Partitions*.
Marek Karpinski
Graph Theory: Vertex Ordering, *Network Design*: Cuts and Connectivity.
Gerhard Woeginger
Sequencing and Scheduling.

This is a continuously updated catalog of approximability results for NP optimization problems. The compendium is also a part of the book Complexity and Approximation. The compendium has not been updated for a while, so there might exist recent results that are not mentioned in the compendium. If you happen to notice such a missing result, please report it to us using the web forms.

You can use web forms to report new problems, new results on existing problems, updates of references or errors.

http://www.nada.kth.se/~viggo/problemlist/compendium.html

$\mathcal{NP}$ 优化问题汇编

浙江大学
ZheJiang University
组合优化

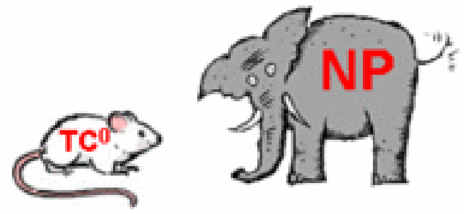
MAXIMUM KNAPSACK

- INSTANCE: Finite set U, for each $u \in U$ a size $s(u) \in Z^+$ and a value $v(u) \in Z^+$, a positive integer $B \in Z^+$.
- SOLUTION: A subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \le B$.
- MEASURE: Total weight of the chosen elements, i.e., $\sum_{u \in U'} v(u)$.
- *Good News*: Admits an FPTAS [266].
- *Comment*: The special case when *s(u)=v(u)* for all $u \in U$ is called MAXIMUM SUBSET SUM. The corresponding minimization problem where $\sum_{u \in U'} s(u) \ge B$ also admits an FPTAS, as well as several other variations of the knapsack problem [191].
- *Garey and Johnson*: MP9

Complexity Zoo

组合优化

- **There are now 496 classes and counting**



All Classes

Complexity classes by letter: Symbols - A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z

Lists of related classes: Communication Complexity - Hierarchies - Nonuniform

Symbols

0-1-NP_C - $1NAuxPDA^p$ - 2-EXP - 3SUM-hard - $\#AC^0$ - #L - #L/poly - #GA - #P - #W[t] - ⊕EXP - ⊕L - ⊕L/poly - ⊕P - $\oplus SAC^0$ - $\oplus SAC^1$

A

A_0PP - AC - AC^0 - $AC^0[m]$ - AC^1 - ACC^0 - AH - AL - ALL - ALOGTIME - AlgP/poly - Almost-NP - Almost-P - Almost-PSPACE - AM - AM_{EXP} - AM ∩ coAM - AM[polylog] - AmpMP - AmpP-BQP - AP - APP - APX - ATIME - AUC-SPACE(f(n)) - AuxPDA - AVBPP - AvgE - AvgP - AW[P] - AWPP - AW[SAT] - AW[*] - AW[t] - AxP - AxPP

B

βP - BH - $BP_d(P)$ - BPE - BPEE - $BP_HSPACE(f(n))$ - BPL - BP•NP - BPP - BPP^{cc} - BPP_k^{cc} - BPP^{KT} - BPP/log - BPP/mlog - BPP//log - BPP/rlog - BPP-OBDD - BPP_{path} - BPQP - BPSPACE(f(n)) - BPTIME(f(n)) - BQNC - BQNP - BQP - BQP/log - BQP/poly - BQP/mlog - BQP/mpoly - BQP/qlog - BQP/qpoly - BQP-OBDD - BQPSPACE - BQP_{CTC} - BQP_{tt}/poly - BQTIME(f(n)) - k-BWBP

NL: Nondeterministic Logarithmic-Space

Has the same relation to L as NP does to P.

In a breakthrough result, was shown to equal coNL [Imm88] [Sze87]. (Though contrast to mNL.)

Is contained in LOGCFL [Sud78], as well as NC^2.

Is contained in UL/poly [RA00].

Deciding whether a bipartite graph has a perfect matching is hard for NL [KUW86].

NL can be defined in a logical formalism as SO(krom) and also as FO(tc), reachability in directed graph is NL-Complete under FO-reduction.

https://complexityzoo.uwaterloo.ca/

浙江大学 ZheJiang University

组合优化

线性规划和
整数规划简介

数学规划

- 若干个变量在满足一些等式或不等式限制条件下，使一个或多个目标函数取得最大值或最小值
  - 满足所有约束条件的点称为可行点（解）（**feasible point**），可行点的集合称为可行域（**feasible region**），记为 S
  - 可行解 $\mathbf{x}^*$ 称为一（极小化）数学规划问题的最优解（**optimal solution**），若对任意 $\mathbf{x} \in S$，$f(\mathbf{x}^*) \le f(\mathbf{x})$；相应地 $f(\mathbf{x}^*)$ 称为最优值
- 单目标数学规划

$$
\begin{aligned}
\min \quad & f(\mathbf{x}) \\
s.t. \quad & g_j(\mathbf{x}) \ge 0 \quad j = 1, \cdots, s \\
& h_l(\mathbf{x}) = 0 \quad l = 1, \cdots, t \\
& \mathbf{x} \in \mathbb{R}^n
\end{aligned}
$$

目标函数

min 极小
max 极大

subject to:
以下为约束条件

变量取值
范围约束

等式约束

不等式约束

数学规划分类

- 线性规划与非线性规划
  - 线性规划：f, g_i, h_j 均为线性函数
  - 非线性规划：f, g_i, h_j 至少有一个是非线性函数
- 整数规划：至少有一个决策变量限定取整数值
  - 混合整数规划（**Mixed Integer Programming, MIP**）：部分决策变量取整数值
  - **0-1**规划：所有决策变量都取 **0** 或 **1**

$$
\begin{aligned}
\min \quad & f(\mathbf{x}) \\
s.t. \quad & g_j(\mathbf{x}) \geq 0 \quad j = 1, \cdots, s \\
& h_l(\mathbf{x}) = 0 \quad l = 1, \cdots, t
\end{aligned}
$$

$$x_i \in \mathbb{Z}$$

$$x_i \in \{0,1\}$$

线性规划

- 任何线性规划总可通过适当变形变为标准形标准型

$$
\begin{aligned}
\min\ & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
s.t.\ & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\
& a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\
& \cdots \\
& a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \\
& x_1, \cdots, x_n \geq 0
\end{aligned}
$$

$$
\begin{aligned}
\min\ & \mathbf{cx} \\
s.t.\ & \mathbf{Ax} = \mathbf{b} \\
& \mathbf{x} \geq \mathbf{0}
\end{aligned}
$$

价格系数向量

系数矩阵

右端向量

单纯形法

组合优化

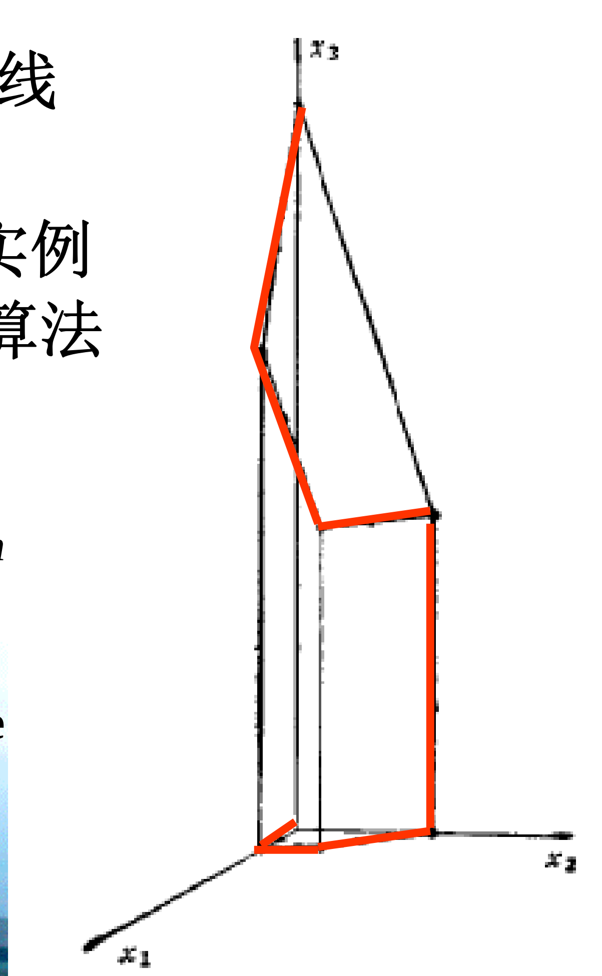
- **1947年Dantzig提出了求解线性规划的单纯形法**
- **1967年得到的Klee-Minty实例说明单纯形法是指数时间算法**

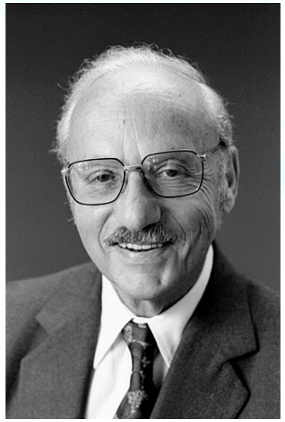
$$\max \sum_{i=1}^{m} 10^{m-i} x_i$$

$$s.t.\ \ 2\sum_{i=1}^{j-1} 10^{j-i} x_i + x_j \le 100^{j-1},\ j=1,\ \cdots,m$$

$$x_i \ge 0,\ i=1,\cdots,m$$

Klee V, Minty GJ, How good is the simplex algorithm? In *Inequalities – III* (Shisha O, Eds.), Academic Press, 159–175, 1972





George Bernard Dantzig (1914-2005)
美国运筹学家

多项式时间算法

组合优化

- **1979年，Khachiyan** 给出了求解线性规划的第一个多项式时间算法——椭球法（**Ellipsoid algorithm**），说明线性规划是多项式时间可解的
- **1984年，Karmarkar** 给出了实际效果更好的线性规划多项式时间算法——内点法（**Interior Point Method**），在数学规划领域产生了深远的影响

Narendra Karmarkar
(1957-)
印度数学家

Leonid Genrikhovich Khachiyan
(1952-2005)
苏联数学家

Khachiyan L, A polynomial algorithm in linear programming. ***Doklady Akademiia Nauk SSSR*****, 244, 1093-1096, 1979**

Karmarkar NK, A new polynomial-time algorithm for linear programming. ***Combinatorica*****, 4, 373–395, 1984**

浙江大学
ZheJiang University

The Mathematical Sputnik

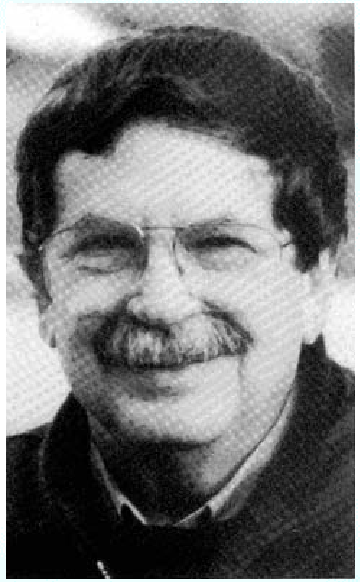
组合优化

- The New York Times of November 7, 1979 announced an event which its readers could easily believe had the importance of the launching of Sputnik. “A surprise discovery by an obscure Soviet mathematician has rocked the world of mathematics and computer analysis ... Apart from its profound theoretical interest… the theory of codes could eventually be affected by the Russian discovery, and this fact has obvious importance to intelligence agencies everywhere”.
- In England, the Guardian broke the story three days earlier, under the headline, “ Soviet Answer to ‘Traveling Salesmen’. ”

The New York Times

theguardian

Lawler EL. The great mathematical Sputnik of 1979. *The Mathematical Intelligencer*, 2, 191-198, 1980.

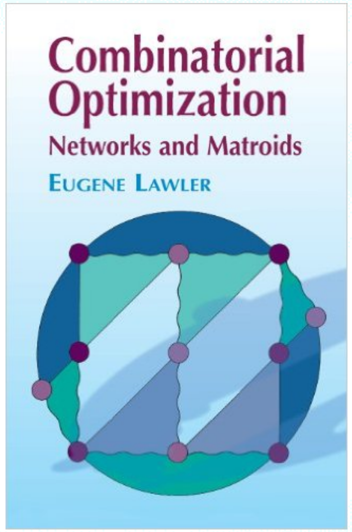


Eugene Leighton (Gene) Lawler (1933-1994)
美国运筹学家

The Mathematical Sputnik

浙江大学
ZheJiang University
组合优化

- **Khachiyan emphatically did not discover a polynomial bounded algorithm for the TSP…What Khachiyan did do was to answer a much smaller question in complexity theory. Linear programming had been known to be a problem in $\mathcal{NP}$. It had not been shown to be $\mathcal{NP}$-complete…Khachiyan squeezed linear programming into $\mathcal{P}$ and thereby resolved the issue.**
- **Only much later, on March 21, did the Times print a retraction,…But the headline read " A Russian's Solution in Math Questioned" and the subhead read " Americans Who Studied Khachiyan Linear Programming Method Express Doubt on Scope. " That made it sound as though poor Khachiyan had exaggerated the importance of his work, and Western mathematicians had cut him down to size.**



Lawler E. *Combinatorial Optimization: Networks and Matroids*, Dover Publications, 1976

整数线性规划的$\mathcal{NP}$ –完全性

- 划分问题 $\le_m^p$ 整数线性规划
  - 任取划分问题的实例 $A=\{a_1,a_2,\cdots,a_n\}$，考虑整数规划 $\sum_{j=1}^{n} a_j x_j = \frac{1}{2}\sum_{j=1}^{n} a_j, x_j \in \{0,1\}$
  - 整数规划有可行解当且仅当划分问题实例答案为“是”
- 整数线性规划 $\in \mathcal{NP}$
  - 若线性不等式组 $\mathbf{Ax} \ge \mathbf{b}$ 有整数解，必存在一整数解，其规模不超过实例规模的多项式

Papadimitriou CH. On the complexity of integer programming. *Journal of the ACM*, 28, 765-768, 1981

整数线性规划算法

浙江大学
ZheJiang University
组合优化

- **1958年，Gomory给出了求解整数线性规划的割平面法（cutting plane method）**
- **1960年，Land和Doig给出了求解整数规划的分支定界法（Branch and Bound ）**

Gomory RE, Outline of an Algorithm for Integer Solutions to Linear Programs, *Bulletin of the American Mathematical Society*, 64, 275–278, 1958

Land A, Doig A, An automatic method of solving discrete programming problems, *Econometrica* 28, 497–520, 1960

Ralph Edward Gomory (1929-)
美国运筹学家

左上:**Ailsa Land**
左下:**Alison Doig**

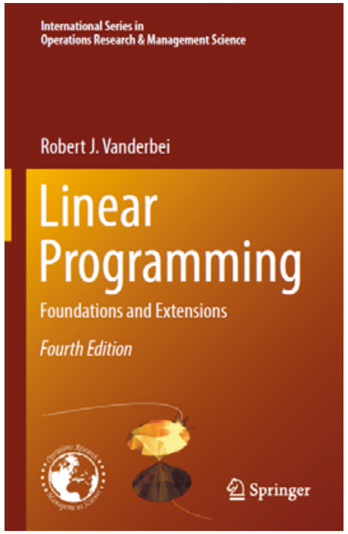
松弛

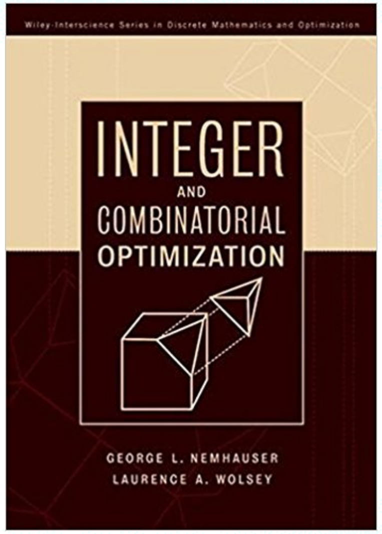
- 设有整数线性规划（**IP**），去除决策变量取整数约束后所得线性规划记为（**LP**），称（**LP**）为（**IP**）的松弛（**relaxation**）
  - （**IP**）的可行域包含于（**LP**）的可行域中
  - （**IP**）的可行解也是（**LP**）的可行解，但反之不然
  - （**IP**）的最优值不优于（**LP**）的最优值
  - 若（**LP**）的最优解为整数解，则它也是（**IP**）的最优解

$$
\begin{aligned}
&\min \quad \mathbf{cx} \\
(\mathrm{IP}) \quad & s.t. \quad \mathbf{Ax} = \mathbf{b} \\
&\qquad\quad \mathbf{x} \in \mathbb{Z}_+^n
\end{aligned}
$$

$$
\begin{aligned}
&\min \quad \mathbf{cx} \\
(\mathrm{LP}) \quad & s.t. \quad \mathbf{Ax} = \mathbf{b} \\
&\qquad\quad \mathbf{x} \in \mathbb{R}_+^n
\end{aligned}
$$

参考资料





黄红选，韩继业，数学规划，清华大学出版社，2006

Vanderbei RJ, *Linear Programming: Foundations and Extensions*, Springer, 2014

Schrijver A. *Theory of Linear and Integer Programming*. Wiley, 1998.

Wolsey LA, Nemhauser GL. *Integer and Combinatorial Optimization*, Wiley, 1999


浙江大学
ZheJiang University
谢　谢