算法设计
——NP完全理论概述

陈卫东

chenwd@scnu. edu. cn

华南师范大学计算机学院

2021-11

提纲

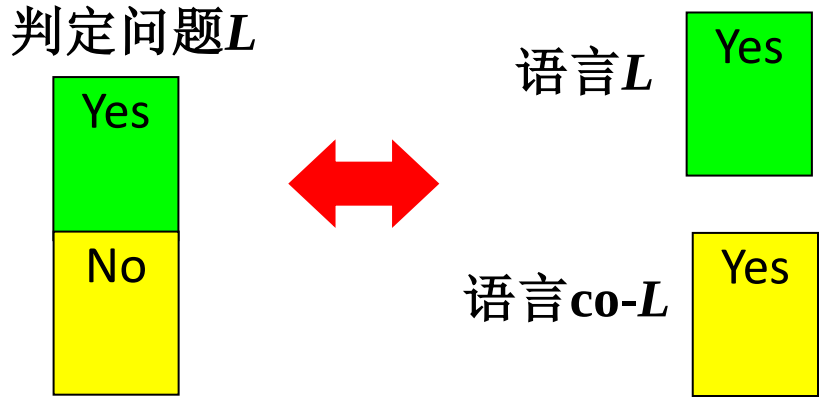
- P类问题、NP类问题
- 归约
- NP完全问题、NP难问题
- Cook定理
- NP难问题的证明
- P = NP ?

预备知识

- 一个问题A的三种常见形式
  - 判定形式A_{DEC}
  - 求值形式A_{EVAL}
  - 最优化形式 A_{OPT}

（三种形式通常容易相互转化，故一般仅考虑判定形式）

【例】SAT问题



- **判定问题*L*、语言*L*、补语言co-*L***

① 任一实例*x*：

对于判定问题***L***来说，*x*被接受或者被拒绝

对于语言***L***来说，*x*属于***L***或不属于***L***

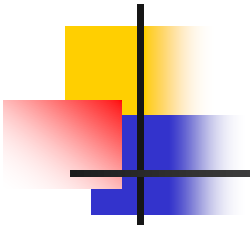
对于语言**co-*L***来说，*x*不属于**co-*L***或属于**co-*L***

② ***L*=co-(co-*L*)**

【例】判定问题**Prime**：任一给定的大于1的正整数是否为素数？

Composite：任一给定的大于1的正整数是否为合数？

语言 **Prime=co-Composite** 且 **Composite=co-Prime**



P类问题

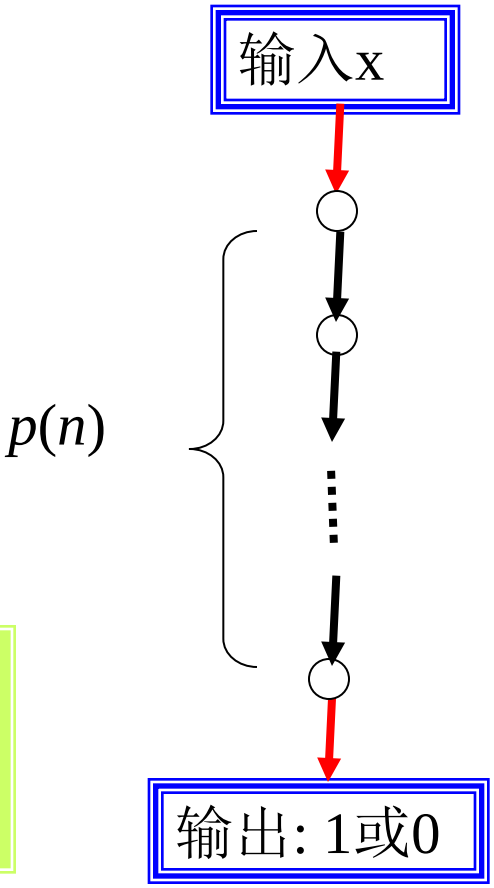
- 确定型图灵机
- P类问题
- P类问题的判别

确定型图灵机（确定型算法）

$q_1\ 0\ 1\ L\ q_2$
$q_1\ 1\ 0\ L\ q_3$
$q_1\ b\ b\ R\ q_4$
$q_2\ 0\ 0\ L\ q_2$
$q_2\ 1\ 1\ L\ q_2$
$q_2\ b\ b\ R\ q_4$
$q_3\ 0\ 1\ L\ q_2$
$q_3\ 1\ 0\ L\ q_3$
$q_3\ b\ b\ R\ q_4$

特点：每一步都是确定的
$x \in L$，输出1.
$x \notin L$，输出0.

P算法计算树

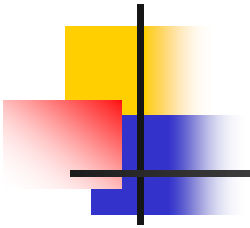


- **P类问题**

——多项式时间求解的问题类（有效求解的问题类）

- **P类问题($A \in \mathbf{P}$)的判别方法**

——给出问题*A*的一个多项式时间求解算法



NP类问题

- 非确定型图灵机
- NP类问题
- NP类问题的判别

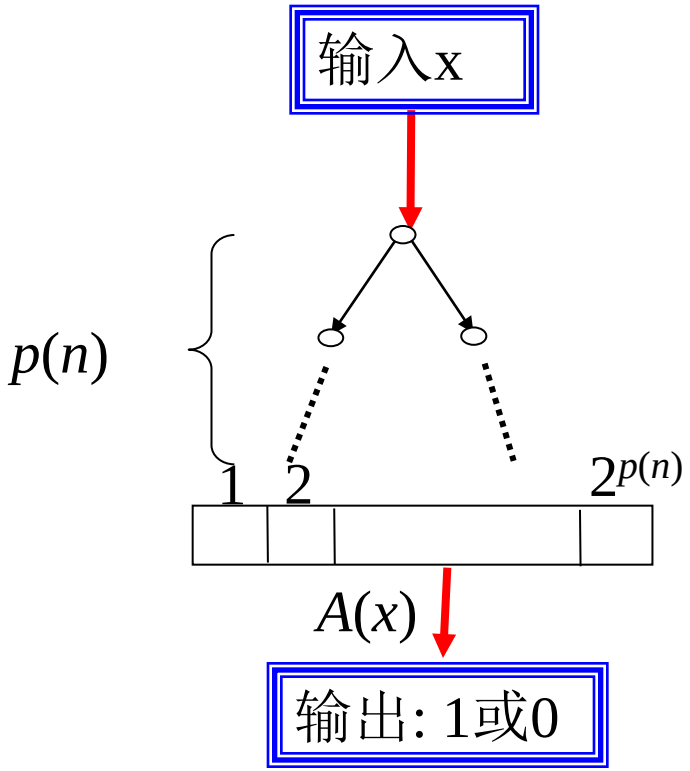
非确定型图灵机（非确定型算法）

- 非确定型算法是一个数学概念，算法的每一步可有多个选择。

- 当$x \in \boldsymbol{L}$时，必存在一计算路径接受x (可“猜”到这条路径)；
  当$x \notin \boldsymbol{L}$时，所有计算路径都拒绝x 。

$q_1\ 0\ 1\ L\ q_2$
$q_1\ 1\ 0\ L\ q_3$
$q_1\ b\ b\ R\ q_4$
$q_2\ 0\ 0\ L\ q_2$
$q_2\ 0\ 1\ R\ q_4$
$q_2\ 1\ 1\ L\ q_2$
$q_2\ b\ b\ R\ q_4$
$q_3\ 0\ 1\ L\ q_2$
$q_3\ 1\ 0\ L\ q_3$
$q_3\ b\ b\ R\ q_4$

NP算法计算树



特点：每一步可有多个选择

$x \in L$，则至少有一条计算路径是接受路径，即$A(x)$中包含至少一个1，从而输出1.

$x \notin L$，则所有计算路径都是拒绝,即$A(x)$全0, 从而输出0.

NP类问题

——非确定型图灵机多项式时间内可求解的问题类

- **事实： NP类足够大，包含了我们感兴趣的几乎所有的问题。**

NP类问题($A \in NP$)的判别方法

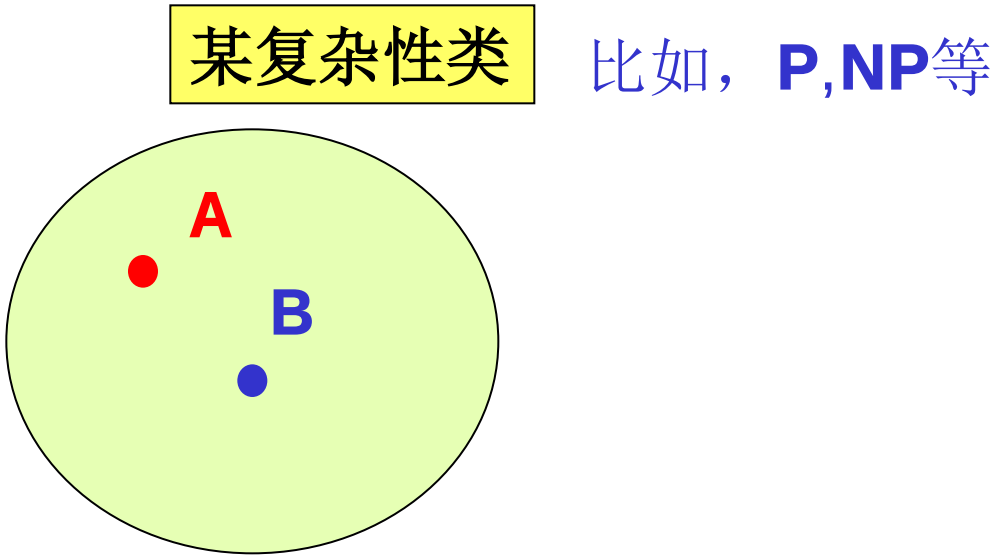
- **方法1（根据NP的定义）**——计算树具有多项式高度且其上存在一条接受路径就接受，否则就拒绝。

- **方法2——NP**是多项式时间可验证的问题类。

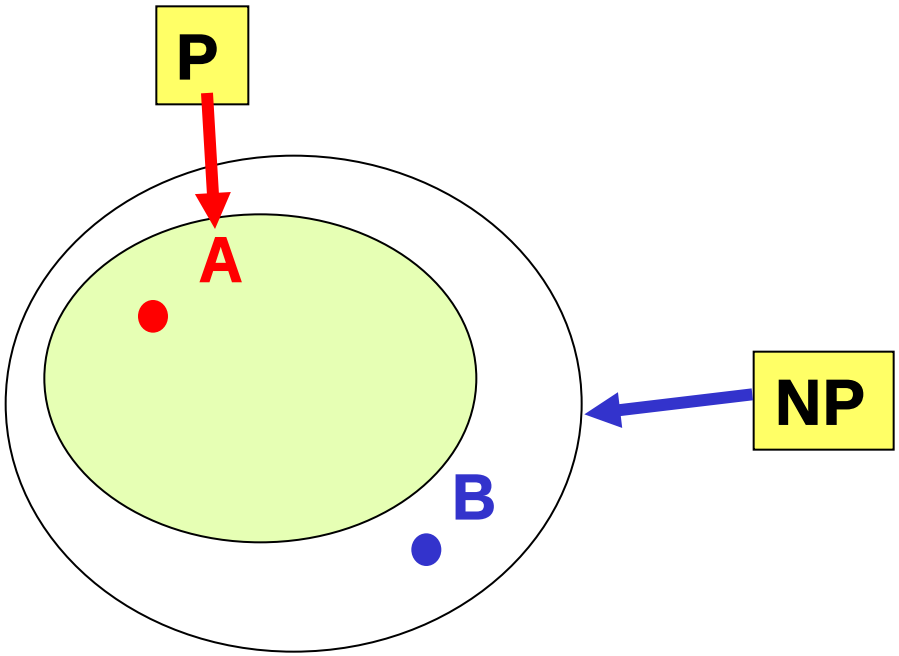
(多项式时间可验证任意解是否符合要求的问题就是NP中的问题)

归约（Reduction）

- 如何判断两个问题复杂性相似？



如何判断 $B \in NP$ 但 $B \notin P$ ？



研究思路

一问题计算上不难于另一问题：$A \leq B$

两问题难度相似：$A \cong B$（ $A \leq B$ 且$B \leq A$ ）

两问题复杂性相似：

$A \cong B \quad \Rightarrow$ A, B 同属于一个复杂性类

问题*A*可归约到问题*B*

Definition *Problem A is algorithmically no more difficult than problem B if there is an algorithm that solves problem A that may make use of an algorithm for B and has the following properties:*

- *The runtime of the algorithm for A, not counting the calls to the algorithm that solves B, is bounded by a polynomial $p(n)$.*
- *The number of calls to the algorithm that solves B is bounded by a polynomial $q(n)$.*
- *The length of each input to a call of the algorithm solving B is bounded by a polynomial $r(n)$.*

Algorithm for A

……

Call Algorithm for B

……

$$t_A(n) \le p(n) + q(n)\, t_B(r(n))$$

问题*A*的难度不超过*B*，记作$A \le B$，并称*A*可(多项式时间)归约到*B*.

归约（Reduction）

- $A \leq B$ ——问题A可(多项式时间)归约到问题B

  - “≤”是一种二元关系，即“不会比…更难”。

    （即如果B有多项式时间算法，则A也有）

  - 如果$A \leq B$，且 $B \leq A$ ，则称A与 B是等价的（难度相似，即复杂性相似），记作$A \equiv B$。

归约（Reduction）

- “≤”是一种序关系，满足自反性、传递性。
- “≡”是一种等价关系，满足自反性、对称性、传递性。
- 归约的力量在于能建立不同类型问题间的难度关系，可建立复杂度未知的问题的难度等价性。

归约（Reduction）

- 一般性结论：$A_{DEC} \le A_{EVAL} \le A_{OPT}$

- 很多问题成立：$A_{EVAL} \le A_{DEC}$
  - 通常使用大约$\log n$次对A_{DEC}算法的调用就得到一个A_{EVAL}算法。但对**TSP**和**Knapsack**问题一般不成立。

- 很多问题成立：$A_{OPT} \le A_{EVAL}$
  - 以几个问题为例介绍如何由A_{EVAL}算法逐步构造问题的一个最优解的方法（即如何得到一A_{OPT}算法）。

归约（Reduction）

- 结论

许多我们感兴趣的优化问题的优化、求值、判定形式都是难度等价的。从复杂性理论的角度来看，我们仅讨论其判定形式的问题是合适的。

2-DM ≤NETWORKFLOW

- 证明思路

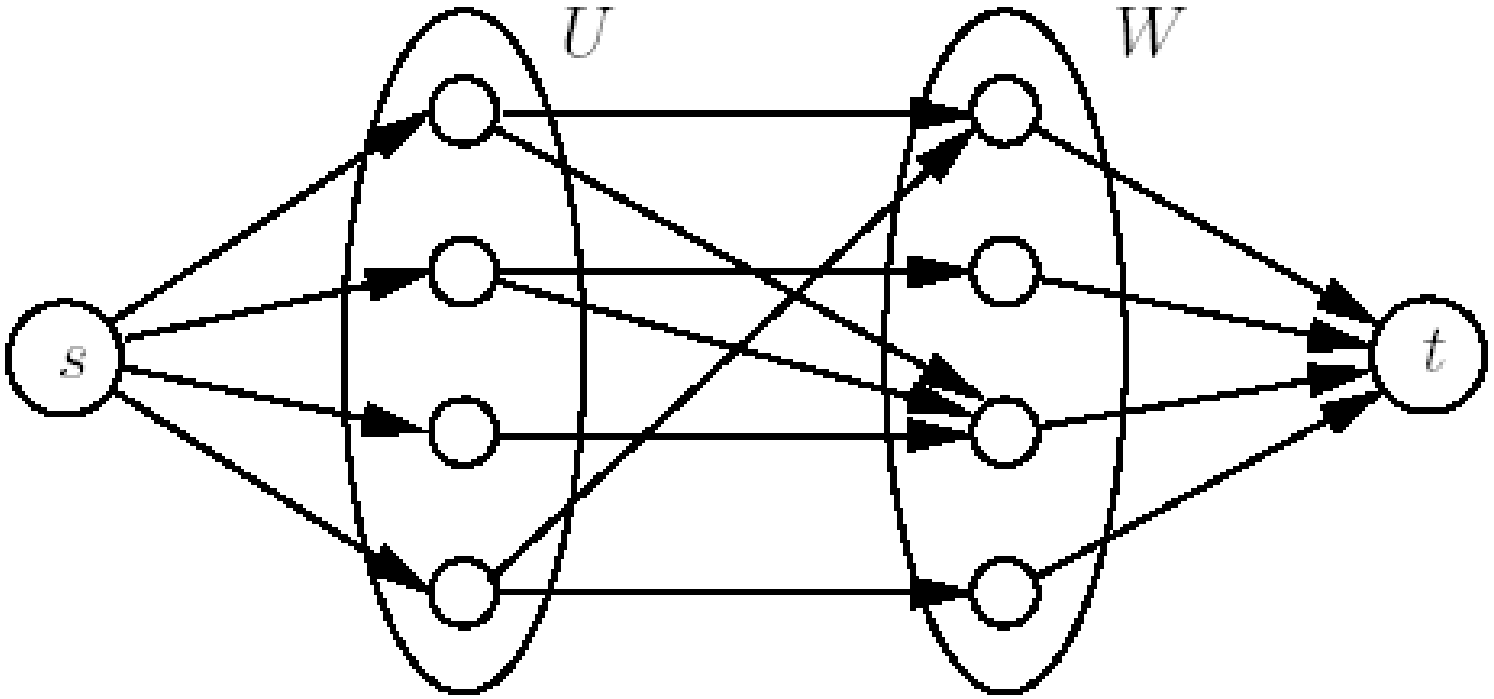


Illustration of the Turing reduction 2-DM $\leq$ NetworkFlow

NP完全问题、NP难度问题

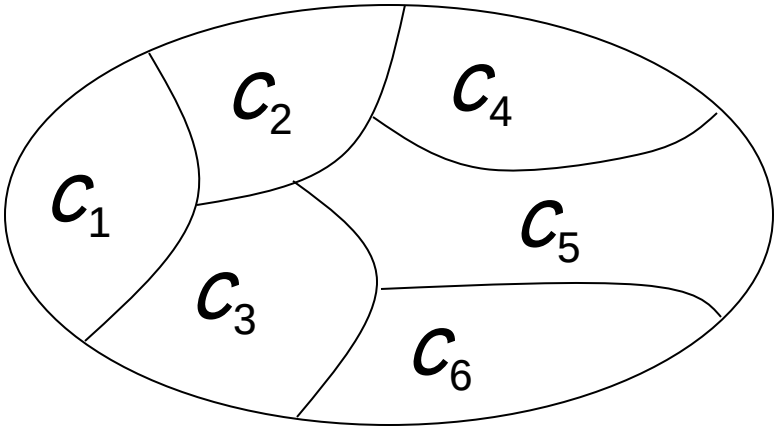
归约 “≤” 可用来比较两问题之间的难度

- ***A*≤*B*** ： 忽略多项式因子意义上，问题***A***的难度不超过***B***的难度。

等价 “≡” 可用来建立复杂度相似的问题类

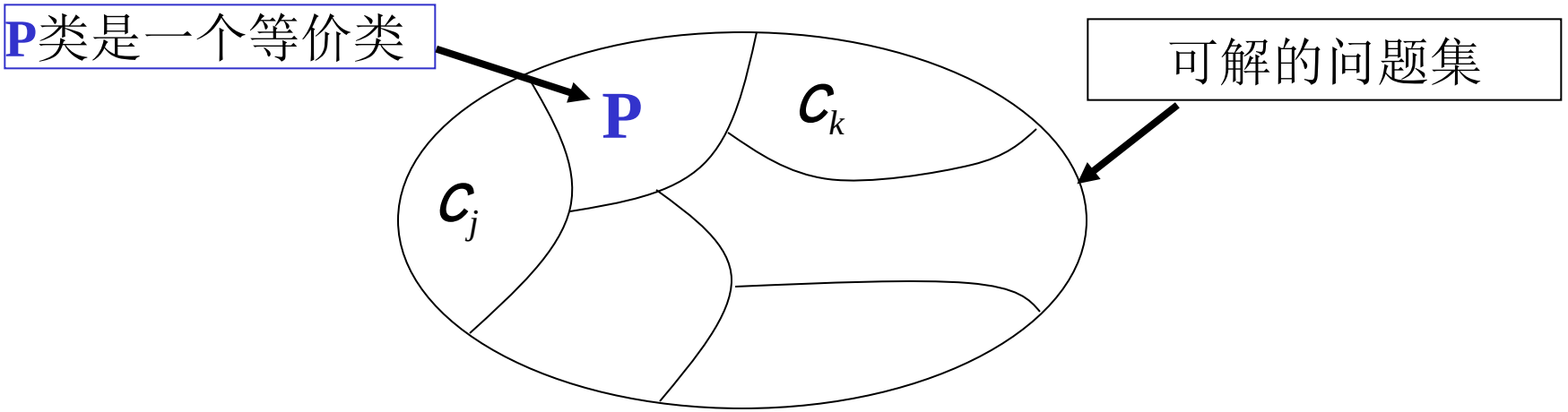
- ***A* ≡ *B*** ： 忽略多项式因子意义上，问题***A***的难度与***B***的难度相似。

回顾： 集合上的一个等价关系可用来划分集合



- 同一等价类c_i中所有元素都是等价的
- 不同等价类中的元素不等价

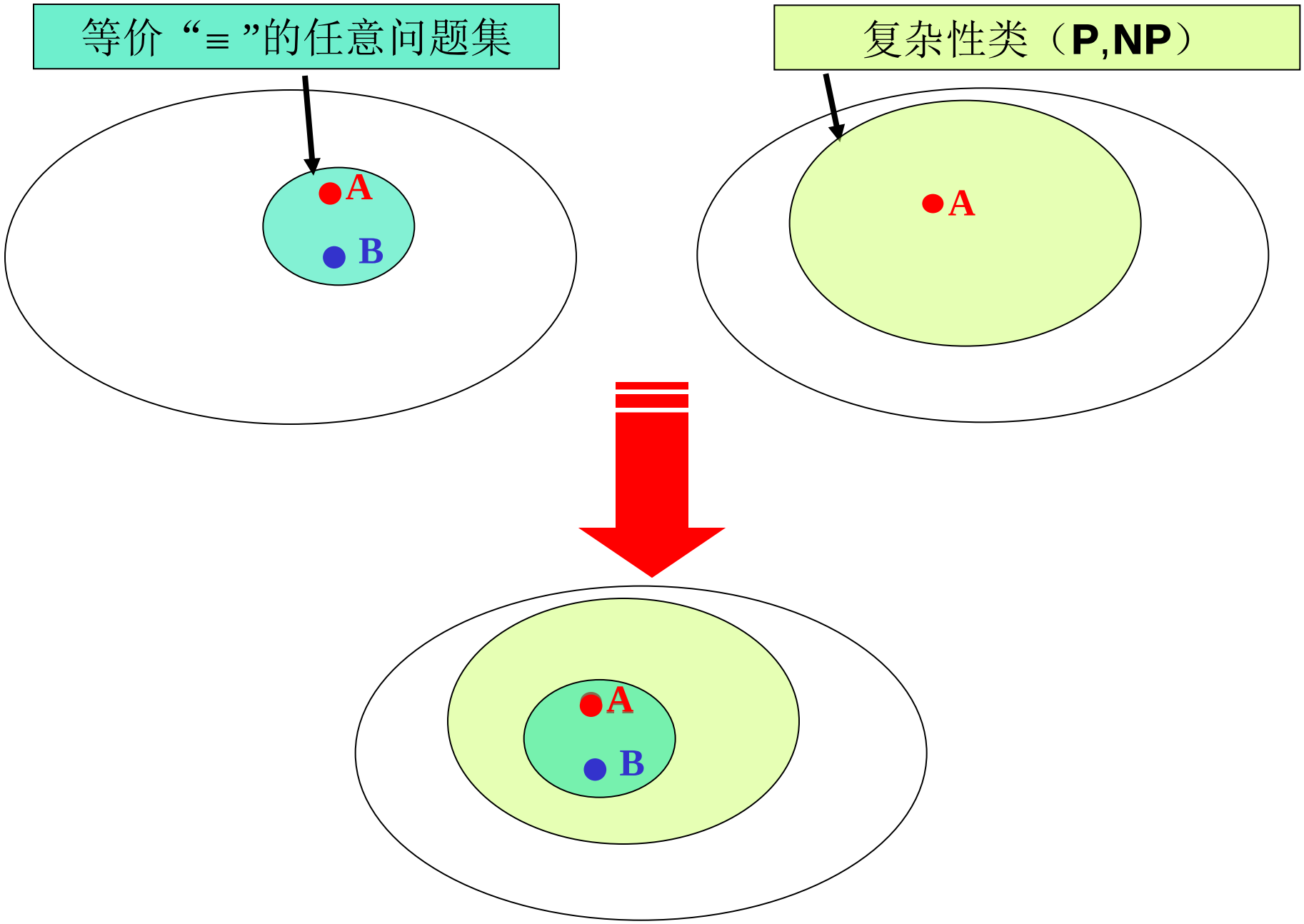
问题的等价 “≡ ”和归约 “≤”

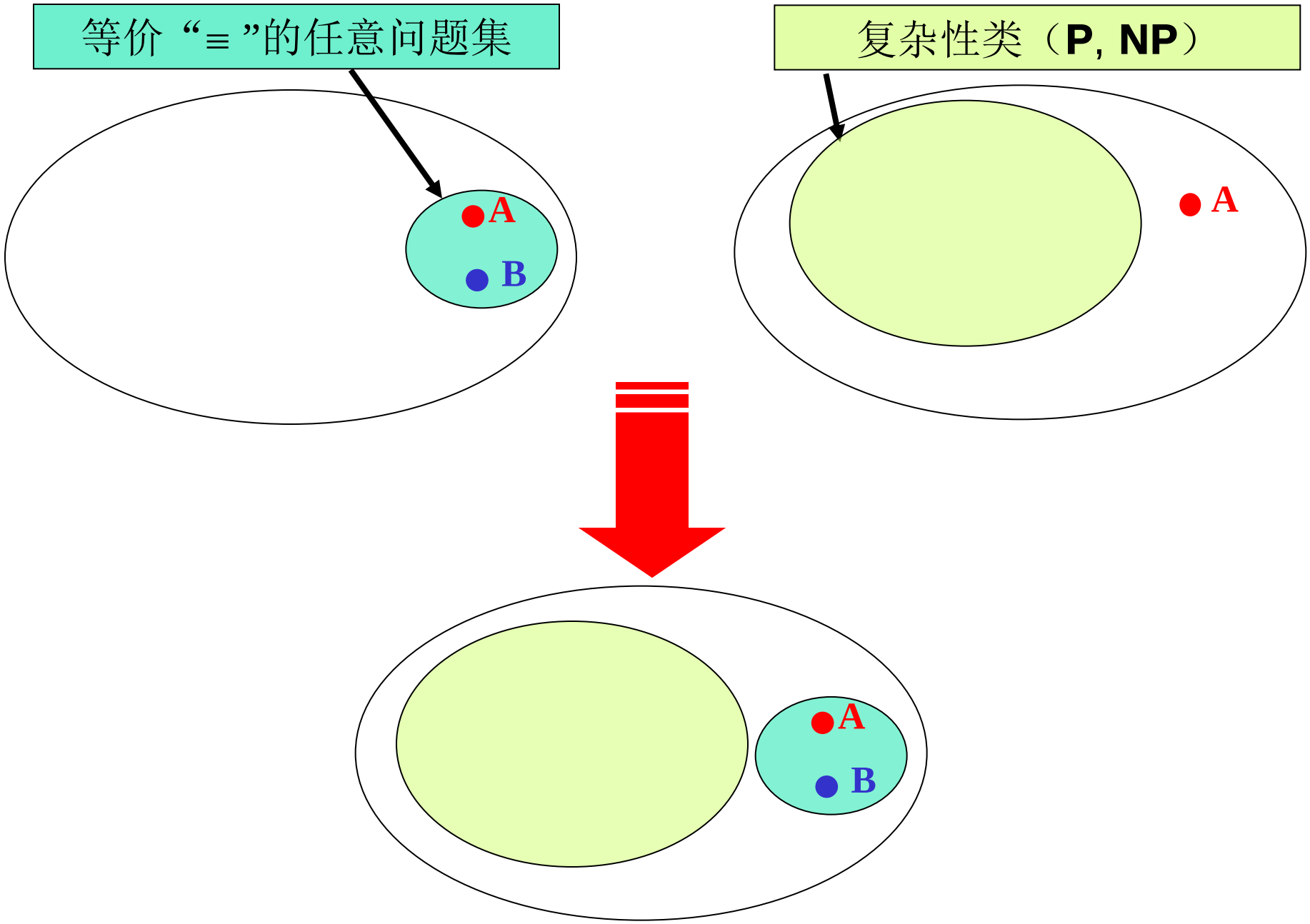


- 对于任意i，等价类C_i中所有问题难度相同
- 等价类C_j与C_k($j \neq k$)中的问题难度不相同，归约 “≤”是这些等价类集合上的一偏序关系：

——$C_j \leq C_k$：存在$A \in C_j$和$B \in C_k$使得$A \leq B$

然而，我们对此偏序关系还知之甚少！


等价“≡”的任意问题集
A
B
复杂性类（P,NP）
A
A
B


等价“≡”的任意问题集
A
B
复杂性类（P, NP）
A
A
B

NP完全问题、NP难度问题

- **关于NP问题的猜想：NP≠P。**

  意味着NP中的最难的问题不能在多项式时间内解决。

NP完全问题、NP难度问题

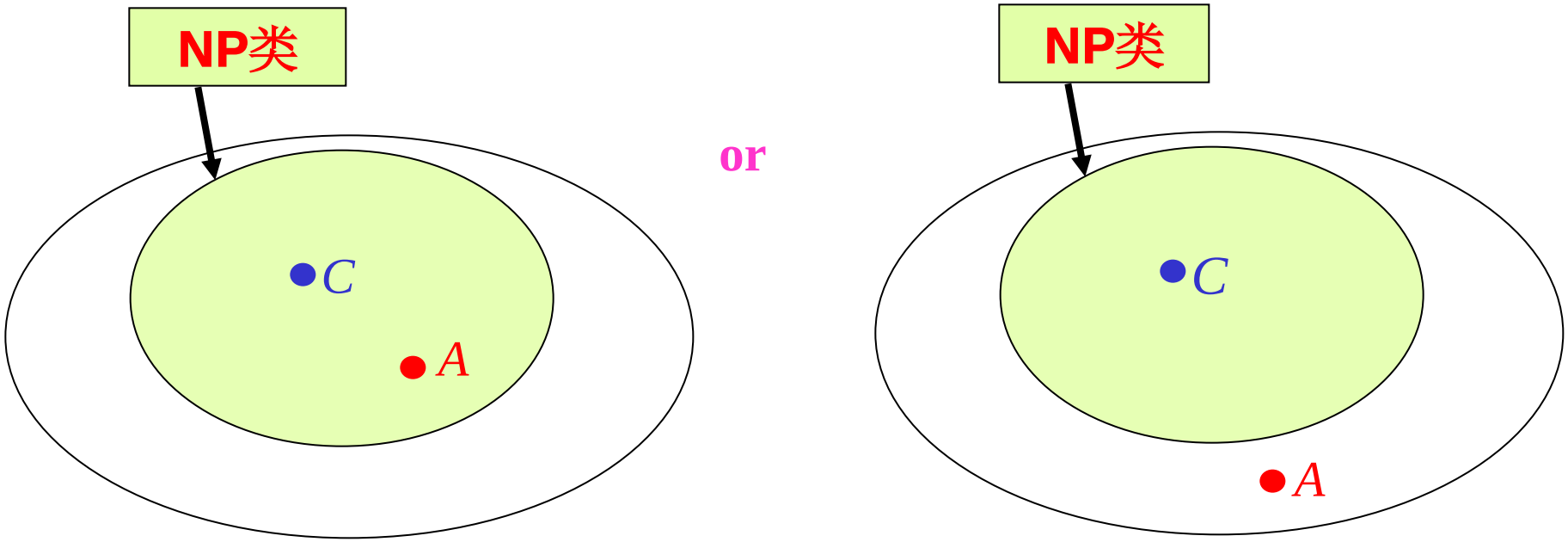
- **基本概念**

设 A 是一判定问题， C 是一判定问题类.

(i) A 是 C–hard： 对任意 $C \in C$ ，均有 $C \leq A$.

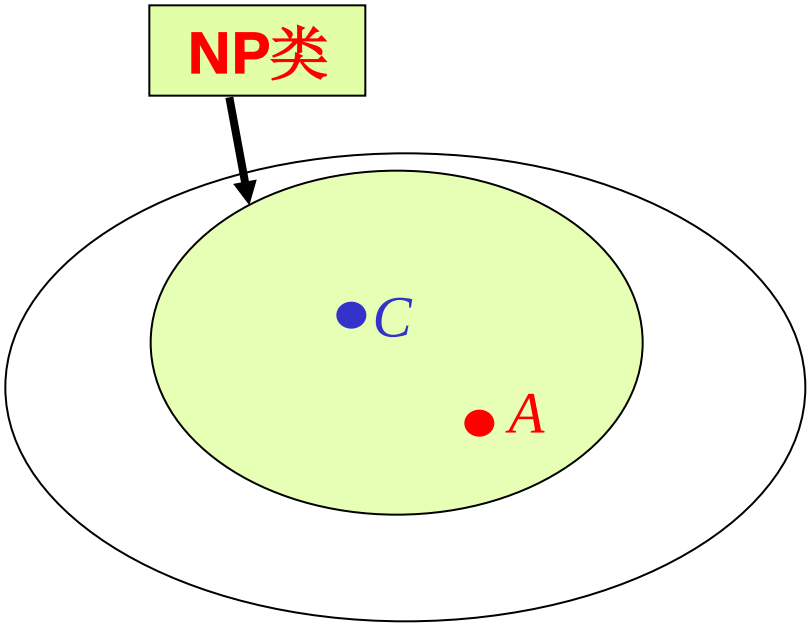
(ii) A 是 C-complete： A 是 C–hard的，且 $A \in C$.

(i) A 是NP难（NP–hard）问题：对任意$C \in$ **NP**均有 $C \leq A$

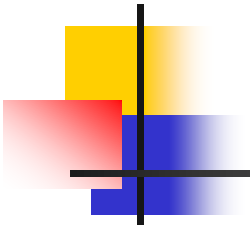


问题A的难度不低于**NP**类中所有问题的难度！

(ii) *A*是NP完全(**NP**-complete)问题： *A* 是**NP**–hard且$A \in C$.



问题*A*是**NP**类中最难的问题！



Cook定理

- **Cook定理**
- **Cook定理的证明**
- **Cook定理的意义**

Cook定理 【*Cook-Levin's Theorem*】

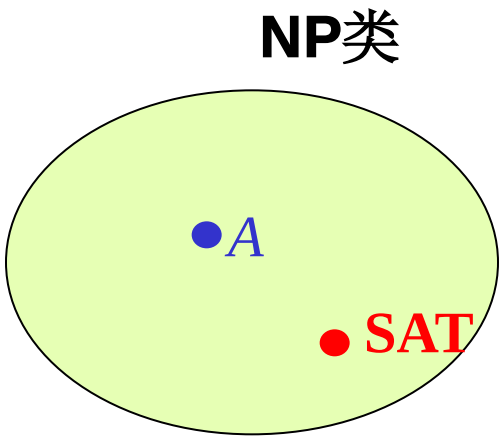
SAT *is* **NP-***complete. Thus* **NP** = **P** *if and only if* **SAT**∈**P.**

- 关于Cook-Levin定理，参见
http://en.wikipedia.org/wiki/Cook%E2%80%93Levin_theorem

Cook定理的证明

- Cook-Levin定理的证明思路， 参见 Cook定理.ppt

设**NP** 中的任何问题***A***的相应的图灵机是 **M**。对于**M**的任何输入 w, 构造一布尔公式 $\varphi_{M,w}$ 使得它是可满足的当且仅当 **M** 接受 w.



Algorithm for *A*

Step 1. 根据***A***的图灵机M构造布尔公式 $\varphi_{M,w}$

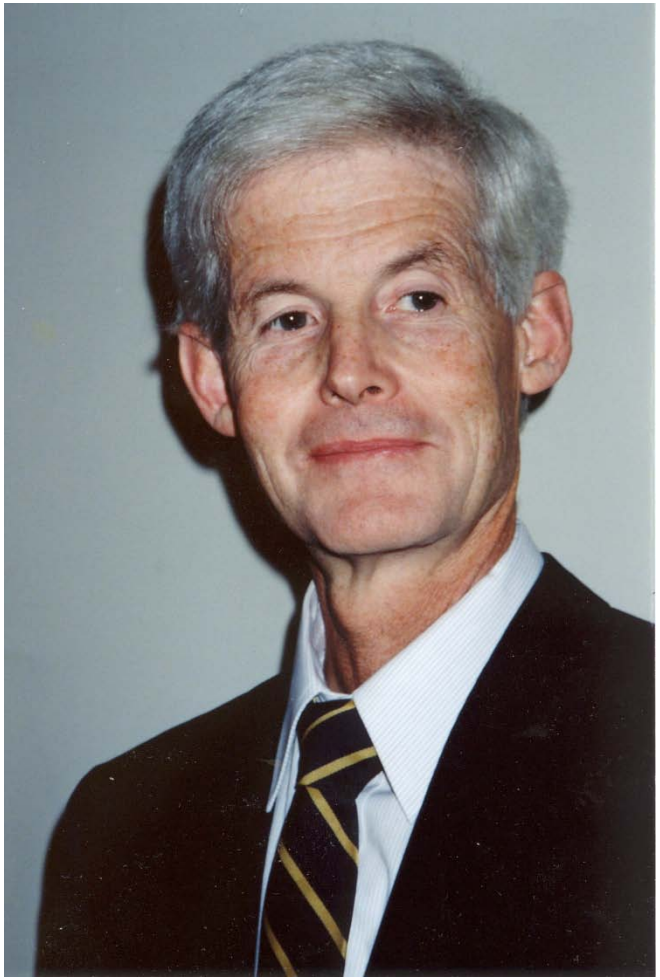
Step 2. Call Algorithm for SAT 来判别布尔公式$\varphi_{M,w}$是否可满足

Step 3. 根据Step2的结果输出“接受”或“拒绝”

由此证明了对任意$A \in$**NP**均有 $A \leq$ **SAT**

Cook定理的意义

- **SAT**问题是第一个被证明的**NP**-complete问题。Cook-Levin定理的贡献？
- 通过将**SAT**归约到**NP**中的其他某问题***A***可以证明问题***A***也是**NP**-complete问题。



Stephen Arthur Cook
Professor, University of Toronto

Stephen Arthur Cook's homepage:
http://www.cs.toronto.edu/~sacook/

Stephen在哈佛大学的博导是中国人Hao Wang.可参见如下：
http://genealogy.math.ndsu.nodak.edu/html/id.phtml?id=29869

Professor Cook at the Fall school of LOGIC & COMPLEXITY in Prague, September 24, 2008

http://en.wikipedia.org/wiki/Stephen_Cook

Stephen Arthur Cook (born December 14, 1939, Buffalo, New York) is a noted computer scientist.

Cook formalized the notion of NP-completeness in a famous 1971 paper "The Complexity of Theorem Proving Procedures", which also contained Cook's theorem, a proof that the boolean satisfiability problem is NP-complete. The paper left unsolved the greatest open question in theoretical computer science - whether complexity classes P and NP are equivalent.

Cook received the Turing Award in 1982 for his discovery. His citation reads:

For his advancement of our understanding of the complexity of computation in a significant and profound way. His seminal paper, The Complexity of Theorem Proving Procedures, presented at the 1971 ACM SIGACT Symposium on the Theory of Computing, laid the foundations for the theory of NP-Completeness. The ensuing exploration of the boundaries and nature of NP-complete class of problems has been one of the most active and important research activities in computer science for the last decade.

Leonid Levin

Professor, Boston U
PhD, Moscow U and MIT

Leonid Levin's homepage:
http://www.cs.bu.edu/fac/lnd/

http://en.wikipedia.org/wiki/Leonid_Levin

Leonid Anatolievich Levin,a Russian-American computer scientist,is currently a professor of computer science at Boston University,where he began teaching in 1980.

He is well known for his work in randomness in computing, algorithmic complexity and intractability, average-case complexity, foundations of mathematics and computer science, algorithmic probability, theory of computation, and information theory.

His life is described in a chapter of the book Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists.

Levin independently discovered a theorem that was also discovered and proven by Stephen Cook. This NP-completeness theorem, often called by inventors' names (see Cook-Levin Theorem) was a basis for one of the seven "Millennium Math. Problems" declared by Clay Mathematics Institute with a $1,000,000 prize offered. It was a breakthrough in computer science and is the foundation of computational complexity. Levin's journal article on this theorem was published in 1973; he had lectured on the ideas in it for some years before that time (see Trakhtenbrot's survey), though complete formal writing of the results took place after Cook's publication.

NP难问题的证明

- 区别下面概念：

  NP-complete， **NP**-hard

- 【Cook定理】 **SAT是NP完全的。**

  如何证明一个问题***B*是NP完全的？**

证明问题 *B*是NP完全的思路：

（1）$B \in NP$

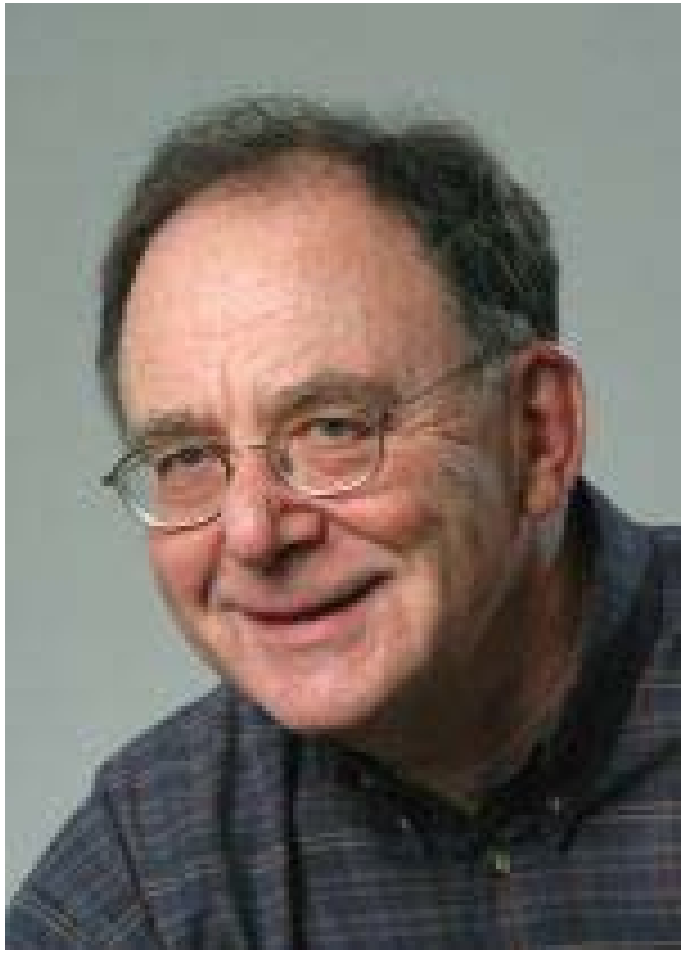
（2）$A \leq B$，其中*A*是任一给定的NP难问题

- 证明：**BMST**是NP完全问题。

  - 什么是**MST** (minimum spanning tree)?
  - 什么是**BMST** (bounded-degree minimum spanning tree)?
  - 证明思路

    HP ≤ BMST

注：哈密尔顿路径问题 **HP** (Hamiltonian Path) 是 **NP**完全问题。



Richard M. Karp
Professor, University of California at Berkeley
Ph.D., Harvard University

Richard M. Karp’homepage:
http://www.eecs.berkeley.edu/~karp/

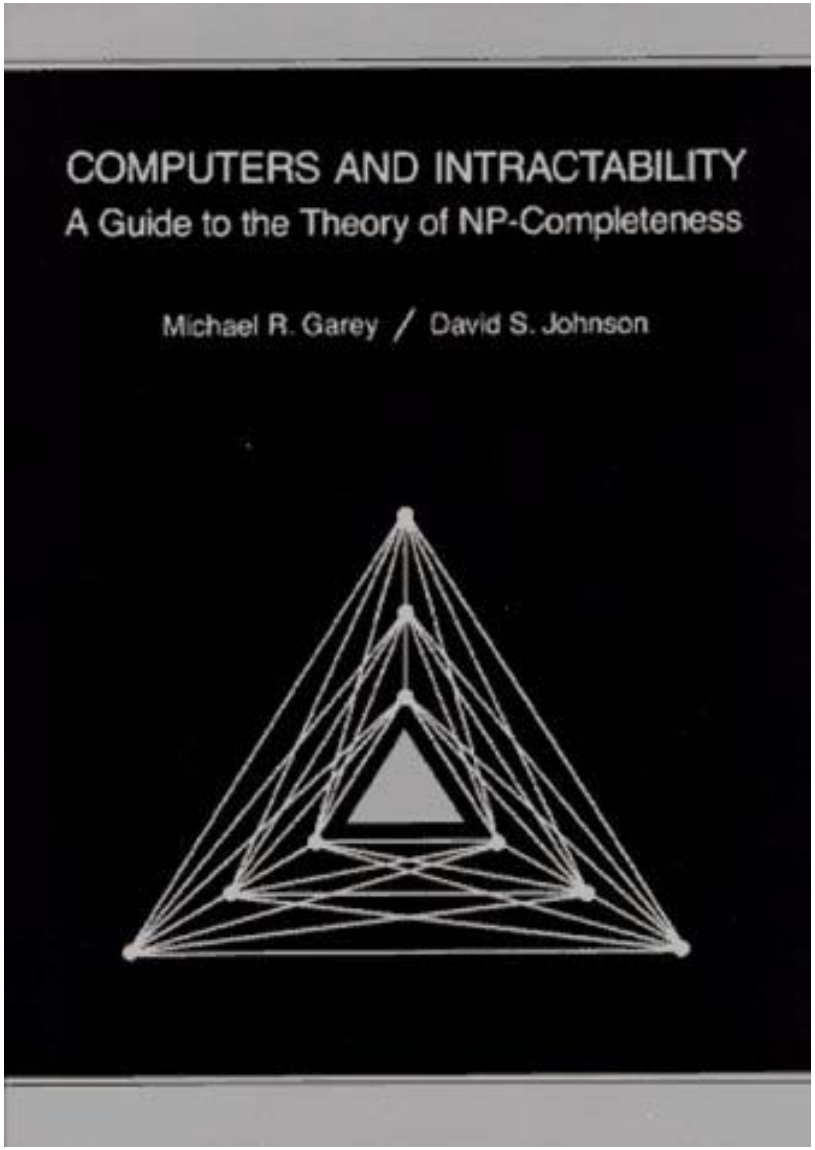
http://en.wikipedia.org/wiki/Richard_Karp

Richard Manning Karp (born 1935) is a computer scientist and computational theorist, notable for research in the theory of algorithms, for which he received a Turing Award in 1985 and the Kyoto Prize in 2008.

The unifying theme in Karp's work has been the study of combinatorial algorithms. His 1972 paper, **"Reducibility Among Combinatorial Problems,"** showed that many of the most commonly studied combinatorial problems are NP-complete, and hence likely to be intractable. Much of his work has concerned parallel algorithms, the probabilistic analysis of combinatorial optimization algorithms and the construction of randomized algorithms for combinatorial problems.

Karp received the Turing Award in 1985. His citation reads:

For his continuing contributions to the theory of algorithms including the development of efficient algorithms for network flow and other combinatorial optimization problems, the identification of polynomial-time computability with the intuitive notion of algorithmic efficiency, and, most notably, contributions to the theory of NP-completeness. Karp introduced the now standard methodology for proving problems to be NP-complete which has led to the identification of many theoretical and practical problems as being computationally difficult.



Garey, Michael R.; David S. Johnson (1979). ***Computers and Intractability: A Guide to the Theory of NP-Completeness*****. W. H. Freeman. ISBN 0716710455**

- NP完全问题汇集：

http://en.wikipedia.org/wiki/List_of_NP-complete_problems
http://www.answers.com/topic/list-of-np-complete-problems
http://en.wikipedia.org/wiki/Karp%27s_21_NP-complete_problems

- NP-hard问题汇集：

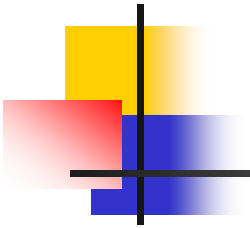
http://www.ensta.fr/~diam/ro/online/viggo_wwwcompendium/wwwcompendium.html

- 关于计算复杂性的一些游戏和智力题：

http://www.ics.uci.edu/~eppstein/cgt/hard.html

- 关于各种计算复杂性问题的关系图，可参阅：

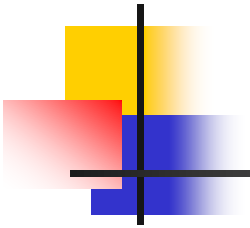
http://en.wikipedia.org/wiki/Complexity_class



P=NP?

- 基本事实

  *Thousands of important problems are **NP**-complete or **NP**-hard. Either all of these problems can be solved in polynomial time and **NP** =**P**, or none of these problems can be solved in polynomial time and **NP**≠**P**.*



P=NP?

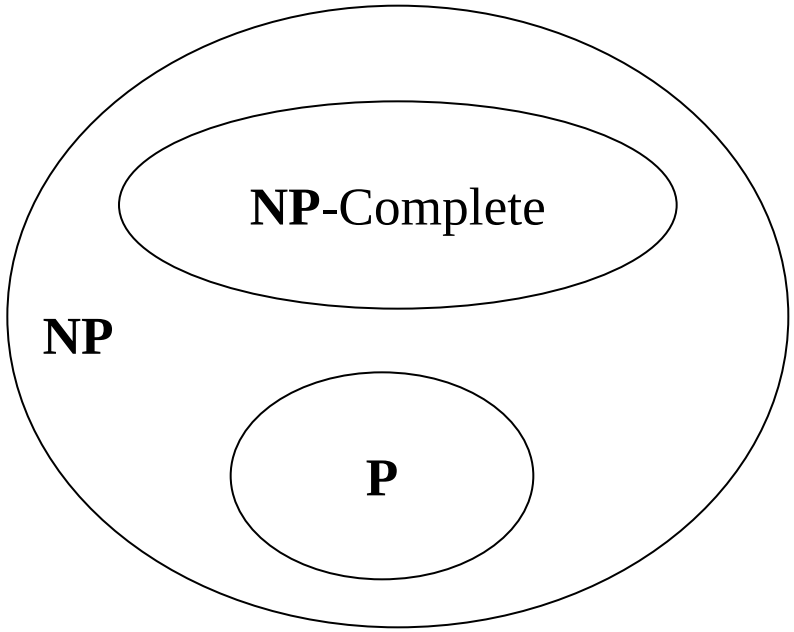
- 基本事实

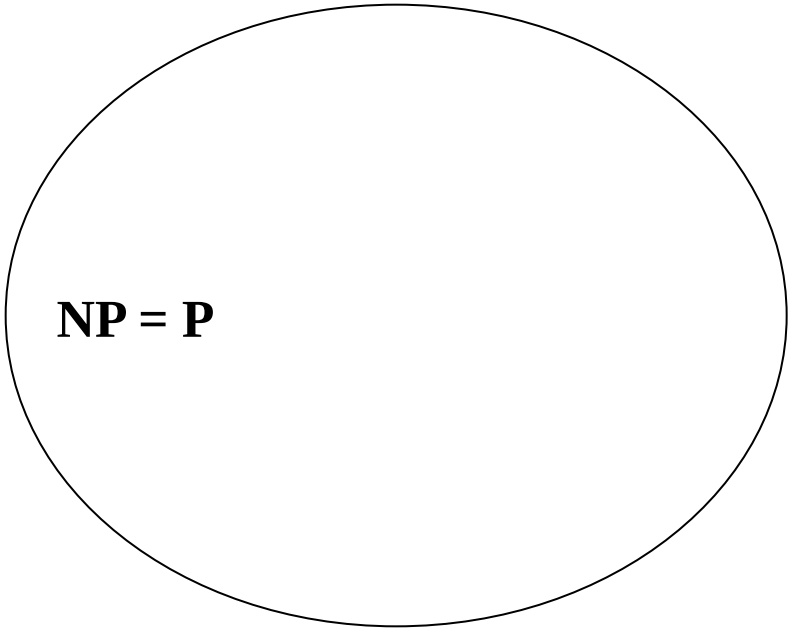
  We do not know if ***NP = P*** *or if* ***NP≠ P****. Since all experts believe with good reason that* ***NP ≠ P****, a proof that a problem is* ***NP****-complete, or* ***NP****-hard is a strong indication that the problem is not solvable in polynomial time.*

世纪难题**P=NP?**

P是否等价与**NP**的问题已经成为数学界、计算理论界的一个著名难题。不管是谁能精确证明**P**等价于**NP**，或**P**不等价于**NP**，都可获得百万美金悬赏。

http://www.claymath.org/millennium/P_vs_NP/





Clay Mathematics Institute—Millennium Problems

- **Clay Mathematics Institute** (**CMI**) is a private, non-profit foundation, based in Cambridge, Massachusetts. The Institute is dedicated to increasing and disseminating mathematical knowledge. It gives out various awards and sponsorships to promising mathematicians. The institute was founded in 1998 through the vision and generosity of Boston businessman. Harvard mathematician Arthur Jaffe was the first president of CMI.
- While the institute is best known for its Millennium Prize Problems, it carries out a wide range of activities, including a postdoctoral program and an annual summer school, the proceedings of which are published jointly with the American Mathematical Society.

http://en.wikipedia.org/wiki/Clay_Mathematics_Institute

- **P versus NP**
  - P = NP problem

The question is whether, for all problems for which a computer can *verify* a given solution quickly (that is, in polynomial time), it can also *find* that solution quickly. This is generally considered the most important open question in theoretical computer science as it has far-reaching consequences in mathematics, philosophy and cryptography (see P=NP proof consequences).

http://en.wikipedia.org/wiki/Complexity_classes_P_and_NP

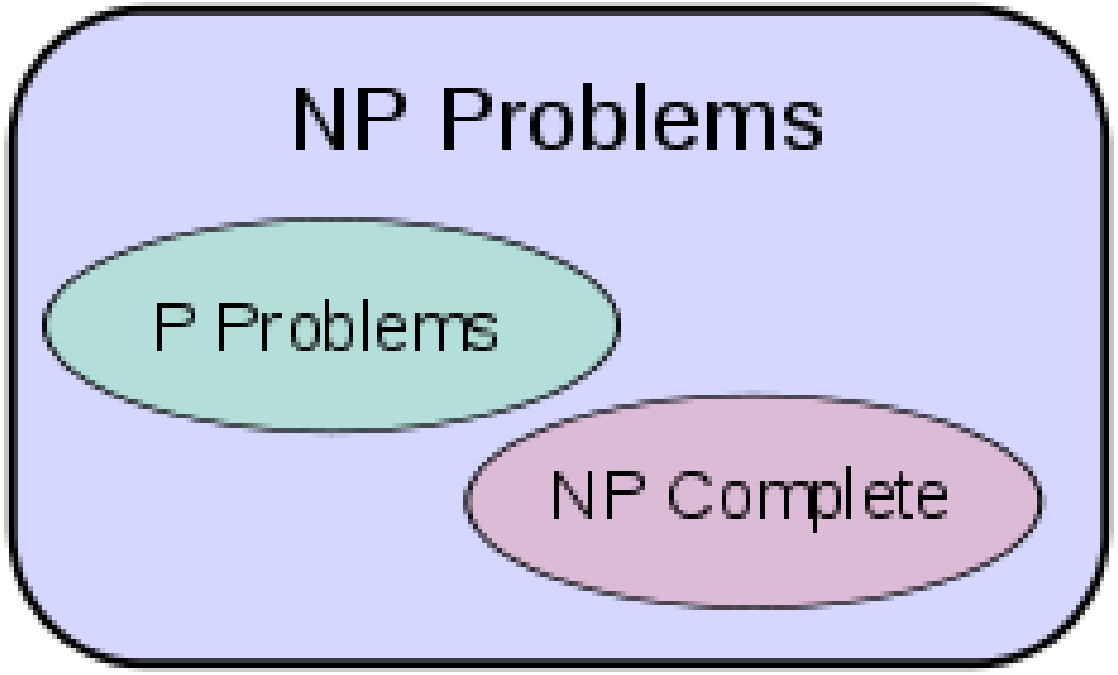


Diagram of complexity classes provided that **P** ≠ **NP**. The existence of problems outside both **P** and **NP**-complete in this case was established by Ladner
（R. E. Ladner “On the structure of polynomial time reducibility,” J.ACM, 22, pp. 151–171, 1975. Corollary 1.1. ACM site ）

各种计算复杂性类的关系图参阅：
http://en.wikipedia.org/wiki/Complexity_class