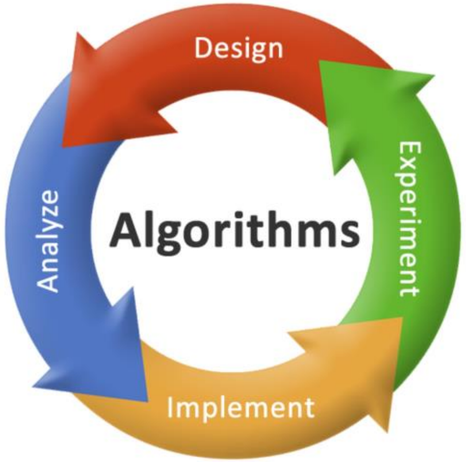


华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Algorithm Design and Analysis

算法设计与分析

Chapter 11: NP完全性-2

张乾坤

不可计算性-2

- P vs. NP
- NP完全
- NP难

不可计算性-2

- P vs. NP
- NP完全
- NP难

P问题

Decision problem.

- Problem X is a set of strings.
- Instance s is one string.
- Algorithm A solves problem X : $A(s) = \begin{cases} yes & \text{if } s \in X \\ no & \text{if } s \notin X \end{cases}$

Def. Algorithm A runs in polynomial time if for every string s, $A(s)$ terminates in $\le p(|s|)$ "steps," where $p(\cdot)$ is some polynomial function.

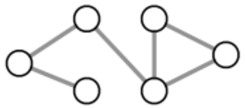
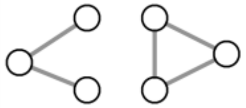
length of s

Def. **P** = set of decision problems for which there exists a poly-time algorithm.

on a deterministic Turing machine

problem PRIMES:	{ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, … }
instance s:	592335744548702854681
algorithm:	Agrawal–Kayal–Saxena (2002)

P问题：存在多项式算法的决策问题

problem	description	poly-time algorithm	yes	no
MULTIPLE	Is x a multiple of y ?	grade-school division	51, 17	51, 16
REL-PRIME	Are x and y relatively prime ?	Euclid's algorithm	34, 39	34, 51
PRIMES	Is x prime ?	Agrawal–Kayal–Saxena	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5 ?	Needleman–Wunsch	niether neither	acgggt ttttta
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss–Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
U-CONN	Is an undirected graph G connected?	depth-first search		

NP问题

Def. Algorithm $C(s, t)$ is a certifier for problem X if for every string s :
$s \in X$ iff there exists a string t such that $C(s, t) = yes$.

Def. **NP** = set of decision problems for which there exists a poly-time certifier.

- $C(s, t)$ is a poly-time algorithm.
- Certificate t is of polynomial size: $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.

"certificate" or "witness"

problem COMPOSITES:	$\{ 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, \ldots \}$
instance s:	437669
certificate t:	541 ← 437,669 = 541 × 809
certifier C(s, t):	grade-school division

可满足性问题验证

- SAT：给定一个CNF，问是否存在一个合法的真值赋值？
- 3-SAT：每个字句正好包含三个字

- Certificate：一组真值赋值
- Certifier：判断每个字句是否至少包含一个TRUE
- 例：**instance s** $\Phi = \left(\overline{x_1} \lor x_2 \lor x_3\right) \land \left(x_1 \lor \overline{x_2} \lor x_3\right) \land \left(\overline{x_1} \lor x_2 \lor x_4\right)$

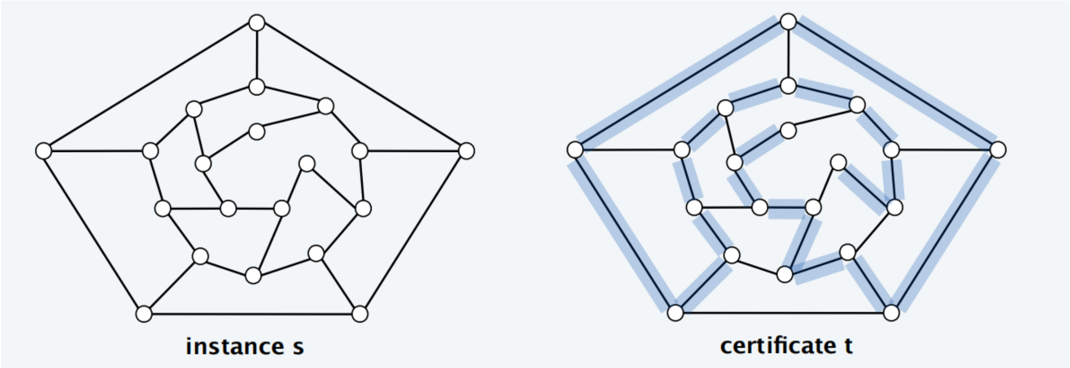
  certificate t $x_1 = \mathit{true},\ x_2 = \mathit{true},\ x_3 = \mathit{false},\ x_4 = \mathit{false}$

- 结论：SAT $\in$ **NP**, 3-SAT $\in$ **NP**.

哈密尔顿回路问题验证

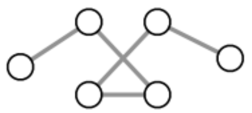
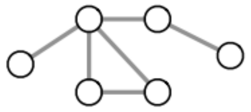
- SAT：给定一个无向图G=(V, E)，问是否存在一个经过所有点的简单路径?

- Certificate：n个点的排列
- Certifier：判断这个排列是否仅包含所有点一次，且包含全部V
- 例：



- 结论：哈密尔顿回路问题∈ **NP**, 3-SAT ∈ **NP**.

NP问题：存在多项式时间验证的决策问题

problem	description	poly-time algorithm	yes	no
L-SOLVE	Is there a vector x that satisfies $Ax = b$?	Gauss–Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$
COMPOSITES	Is x composite ?	Agrawal–Kayal–Saxena	51	53
FACTOR	Does x have a nontrivial factor less than y ?	???	(56159, 50)	(55687, 50)
SAT	Given a CNF formula, does it have a satisfying truth assignment?	???	$\neg x_1 \vee x_2 \vee \neg x_3$ $x_1 \vee \neg x_2 \vee x_3$ $\neg x_1 \vee \neg x_2 \vee x_3$	$\neg x_2$ $x_1 \vee x_2$ $\neg x_1 \vee x_2$
HAMILTON-PATH	Is there a simple path between u and v that visits every node?	???		

测试1

Which of the following graph problems are known to be in NP?

A. Is the length of the longest simple path $\leq k$?

B. Is the length of the longest simple path $\geq k$?

C. Is the length of the longest simple path $= k$?

D. Find the length of the longest simple path.

E. All of the above.

测试2

In complexity theory, the abbreviation NP stands for...

- **A.** Nope.
- **B.** No problem.
- **C.** Not polynomial time.
- **D.** Not polynomial space.
- **E.** Nondeterministic polynomial time.

NP的重要性

- NP问题：存在多项式时间验证的决策问题

“ *NP captures vast domains of computational, scientific, and mathematical endeavors, and seems to roughly delimit what mathematicians and scientists have been aspiring to compute feasibly.* ” — *Christos Papadimitriou*

“ *In an ideal world it would be renamed P vs VP.* ” — *Clyde Kruskal*

P, NP和EXP

- P：存在多项式时间算法的决策问题
- NP：可被多项式时间验证的决策问题
- EXP：存在指数时间算法的决策问题

- 事实：**P ⊆ NP ⊆ EXP；P $\neq$ EXP**

主要矛盾： P vs. NP

- Q： 如何求解3-SAT?
- A： 暴力搜索

- Q： 我们能稍微聪明点吗?
- 猜想： 3-SAT无多项式时间算法

主要矛盾： P vs. NP

- P=NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- 主流意见： 大概率不等于

P vs. NP讨论

- **P $\neq$ NP**

“ I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (i) It is a legitimate mathematical possibility and (ii) I do not know.”

— Jack Edmonds 1966

“ In my view, there is no way to even make intelligent guesses about the answer to any of these questions. If I had to bet now, I would bet that P is not equal to NP. I estimate the half-life of this problem at 25–50 more years, but I wouldn’t bet on it being solved before 2100. ”

— Bob Tarjan (2002)

P vs. NP讨论

- **P $\neq$ NP**

“ We seem to be missing even the most basic understanding of the nature of its difficulty…. All approaches tried so far probably (in some cases, provably) have failed. In this sense P =NP is different from many other major mathematical problems on which a gradual progress was being constantly done (sometimes for centuries) whereupon they yielded, either completely or partially. ”

— Alexander Razborov (2002)

P vs. NP讨论

- **P = NP**

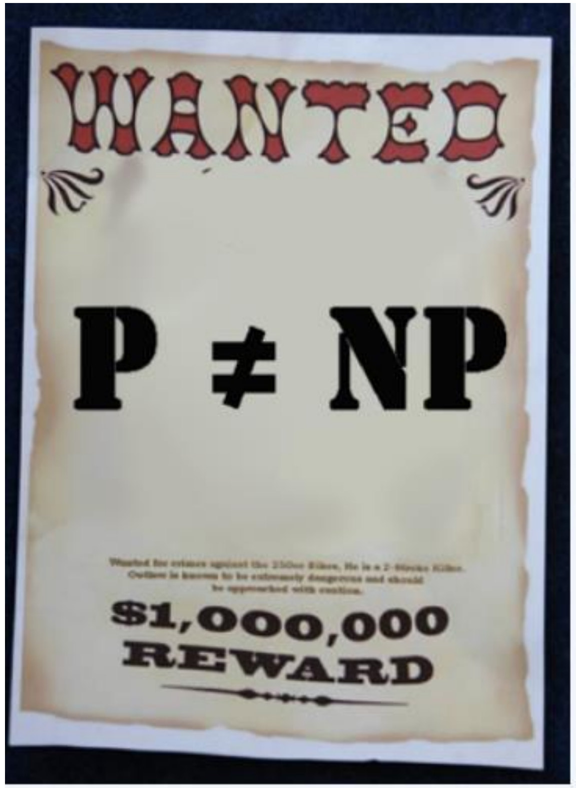
" *I think that in this respect I am on the loony fringe of the mathematical community: I think (not too strongly!) that P=NP and this will be proved within twenty years. Some years ago, Charles Read and I worked on it quite bit, and we even had a celebratory dinner in a good restaurant before we found an absolutely fatal mistake.* "

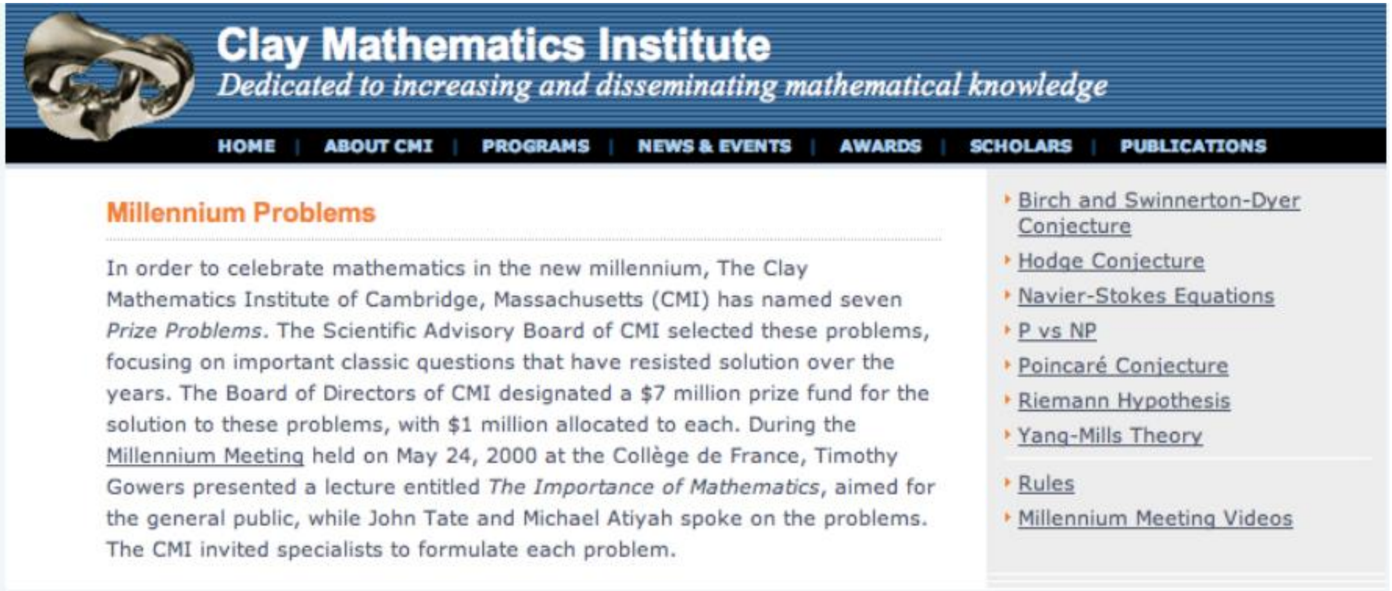
— *Béla Bollobás (2002)*

" *In my opinion this shouldn't really be a hard problem; it's just that we came late to this theory, and haven't yet developed any techniques for proving computations to be hard. Eventually, it will just be a footnote in the books.* " — *John Conway*

P vs. NP讨论

- **千禧年难题之首——1百万美金**





Millennium Problems

In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) has named seven *Prize Problems*. The Scientific Advisory Board of CMI selected these problems, focusing on important classic questions that have resisted solution over the years. The Board of Directors of CMI designated a $7 million prize fund for the solution to these problems, with $1 million allocated to each. During the Millennium Meeting held on May 24, 2000 at the Collège de France, Timothy Gowers presented a lecture entitled *The Importance of Mathematics*, aimed for the general public, while John Tate and Michael Atiyah spoke on the problems. The CMI invited specialists to formulate each problem.

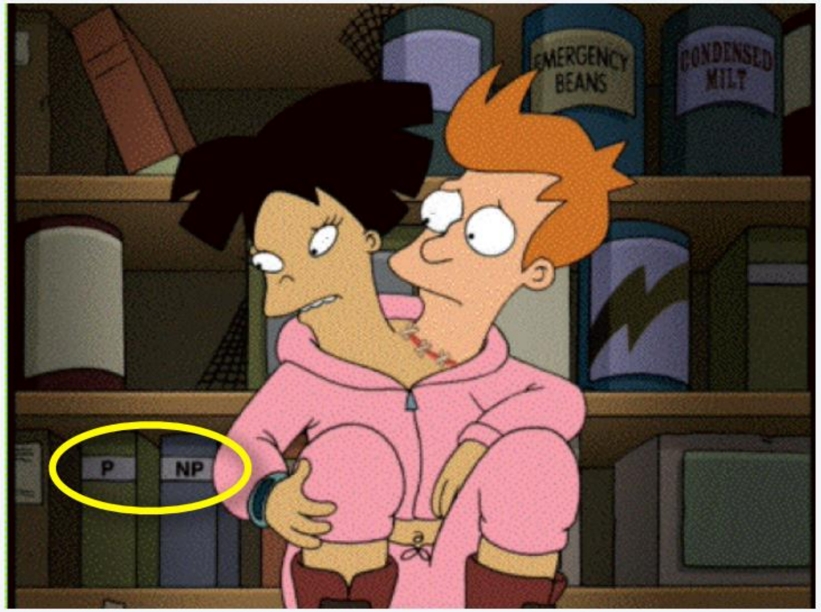
- Birch and Swinnerton-Dyer Conjecture
- Hodge Conjecture
- Navier-Stokes Equations
- P vs NP
- Poincaré Conjecture
- Riemann Hypothesis
- Yang-Mills Theory

- Rules
- Millennium Meeting Videos

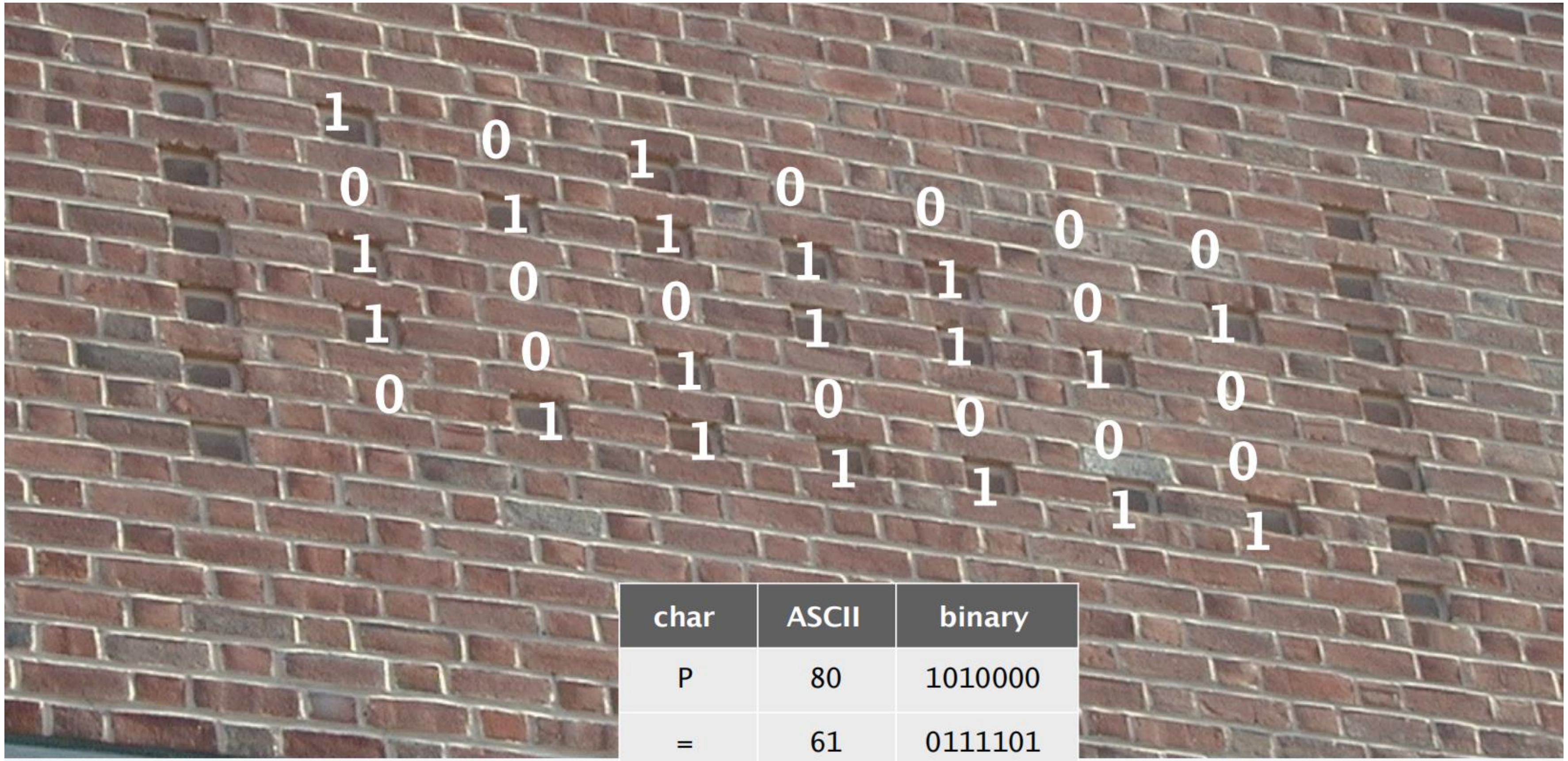
P vs. NP讨论

Some writers for the Simpsons and Futurama.

- J. Steward Burns. *M.S. in mathematics (Berkeley '93).*
- David X. Cohen. *M.S. in computer science (Berkeley '92).*
- Al Jean. *B.S. in mathematics. (Harvard '81).*
- Ken Keeler. *Ph.D. in applied mathematics (Harvard '90).*
- Jeff Westbrook. *Ph.D. in computer science (Princeton '89).*



普林斯顿大学CS大楼



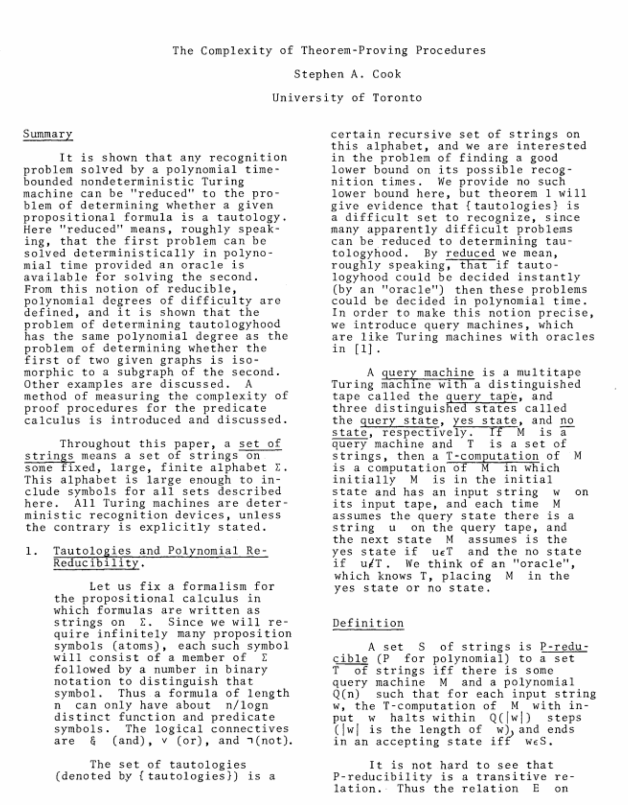
char	ASCII	binary
P	80	1010000
=	61	0111101
N	78	1001110
P	80	1010000
?	63	0111111

不可计算性-2

- P vs. NP
- NP完全
- NP难

NP-complete （NP完全）

- 定义：如果Y是NP问题，且对任意NP问题X，$X \leq_P Y$，则Y问题被称为NP-complete。
- 第一个NP完全问题：[Cook 1971, Levin 1973] SAT-NP完全



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.

1. Tautologies and Polynomial Re-Reducibility.

Let us fix a formalism for the propositional calculus in which formulas are written as strings on Σ. Since we will require infinitely many proposition symbols (atoms), each such symbol will consist of a member of Σ followed by a number in binary notation to distinguish that symbol. Thus a formula of length n can only have about n/logn distinct function and predicate symbols. The logical connectives are & (and), ∨ (or), and ¬(not).

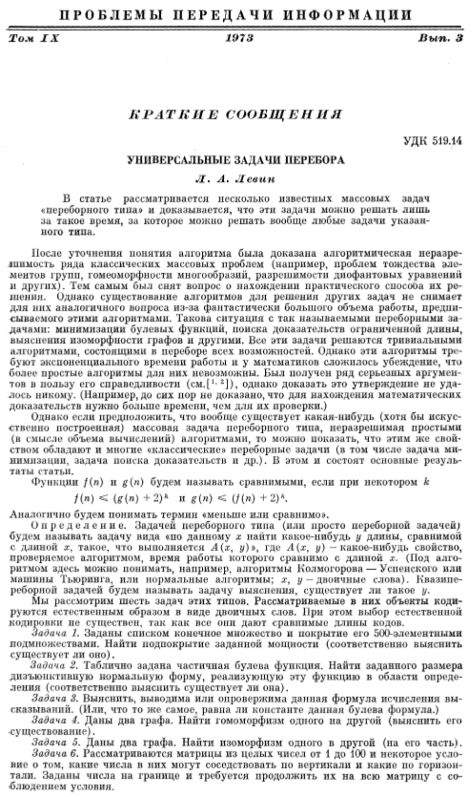
The set of tautologies (denoted by {tautologies}) is a certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a 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 the next state M assumes is the yes state if u∈T and the no state if u∉T. We think of an "oracle", which knows T, placing M in the yes state or no state.

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial Q(n) such that for each input string w, the T-computation of M with input w halts within Q(|w|) steps (|w| is the length of w), and ends in an accepting state iff w∈S.

It is not hard to see that P-reducibility is a transitive relation. Thus the relation E on



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

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

Аналогично будем понимать термин «меньше или сравнимо».

Определение. Задачей переборного типа (или просто переборной задачей) будем называть задачу вида «по данному x найти какое-нибудь y длины, сравнимой с длиной x, такое, что выполняется $A(x, y)$», где $A(x, y)$ – какое-нибудь свойство, проверяемое алгоритмом, время работы которого сравнимо с длиной x. (Под алгоритмом здесь можно понимать, например, алгоритмы Колмогорова — Успенского или машины Тьюринга, или нормальные алгоритмы; x, y – двоичные слова). Квазипереборной задачей будем называть задачу выяснения, существует ли такое y.

Мы рассмотрим шесть задач этих типов. Рассматриваемые в них объекты кодируются естественным образом в виде двоичных слов. При этом выбор естественной кодировки не существен, так как все они дают сравнимые длины кодов.

Задача 1. Заданы списком конечное множество и покрытие его 500-элементными подмножествами. Найти подпокрытие заданной мощности (соответственно выяснить существует ли оно).

Задача 2. Таблично задана частичная булева функция. Найти заданного размера дизъюнктивную нормальную форму, реализующую эту функцию в области определения (соответственно выяснить существует ли она).

Задача 3. Выяснить, выводима или опровержима данная формула исчисления высказываний. (Или, что то же самое, равна ли константе данная булева формула.)

Задача 4. Даны два графа. Найти гомоморфизм одного на другой (выяснить его существование).

Задача 5. Даны два графа. Найти изоморфизм одного в другой (на его часть).

Задача 6. Рассматриваются матрицы из целых чисел от 1 до 100 и некоторое условие о том, какие числа в них могут соседствовать по вертикали и какие по горизонтали. Заданы числа на границе и требуется продолжить их на всю матрицу с соблюдением условия.

证明NP完全性

- 如果我们知道了第一个NP完全问题，其他的就是多米诺骨牌。
- 如何证明Y问题是NP完全的：
  - 1. 证明Y是NP
  - 2. 找一个NP完全问题X
  - 3. 证明$X \leq_P Y$

Proposition. If $X \in$ **NP**-complete, $Y \in$ **NP**, and $X \leq_P Y$, then $Y \in$ **NP**-complete.

Quiz 3

Suppose that $X \in$ NP-COMPLETE, $Y \in$ NP, and $X \leq_P Y$. Which can you infer?

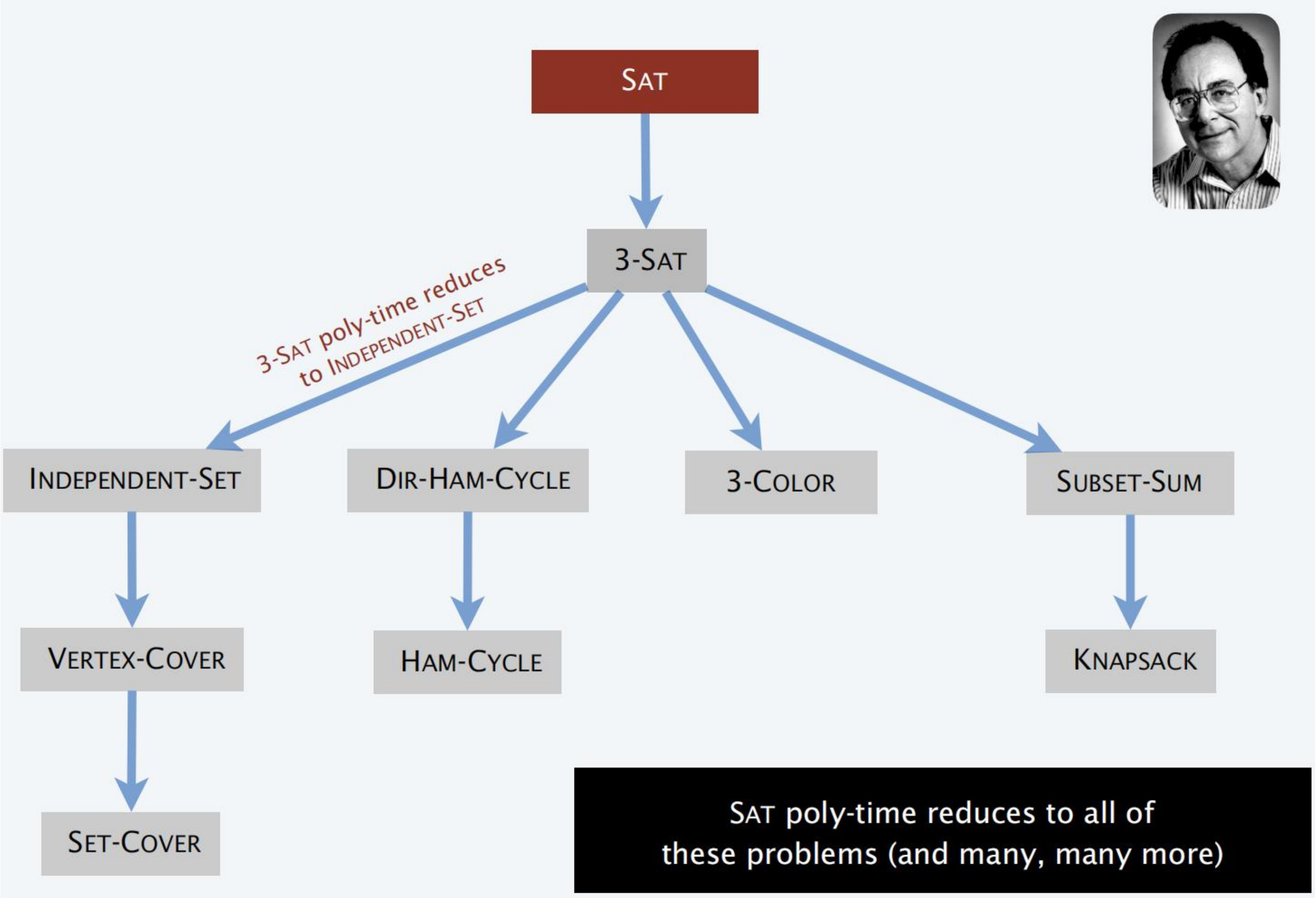
A. Y is **NP**-complete.

B. If $Y \notin$ **P**, then **P** ≠ **NP**.

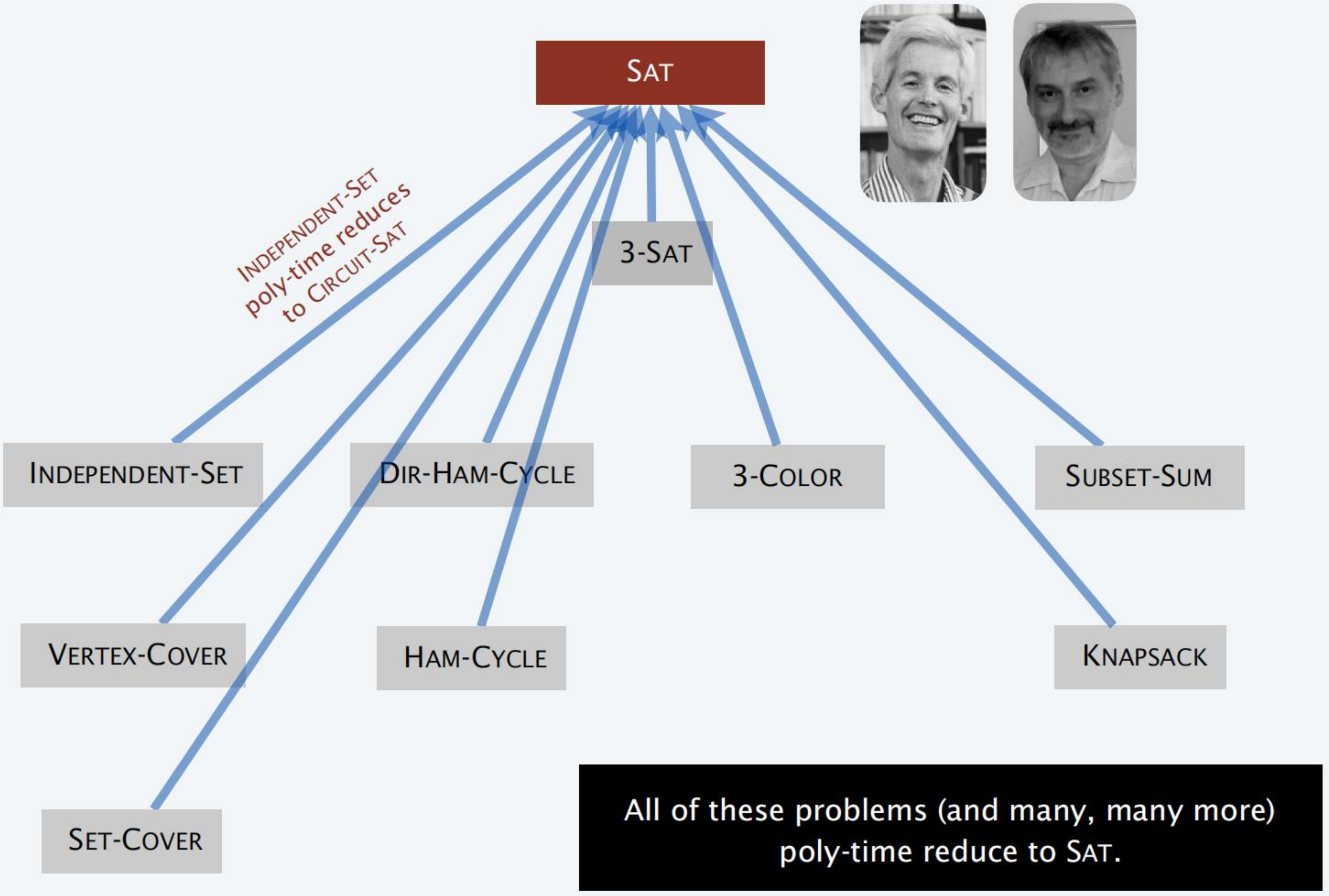
C. If **P** ≠ **NP**, then neither X nor Y is in **P**.

D. All of the above.

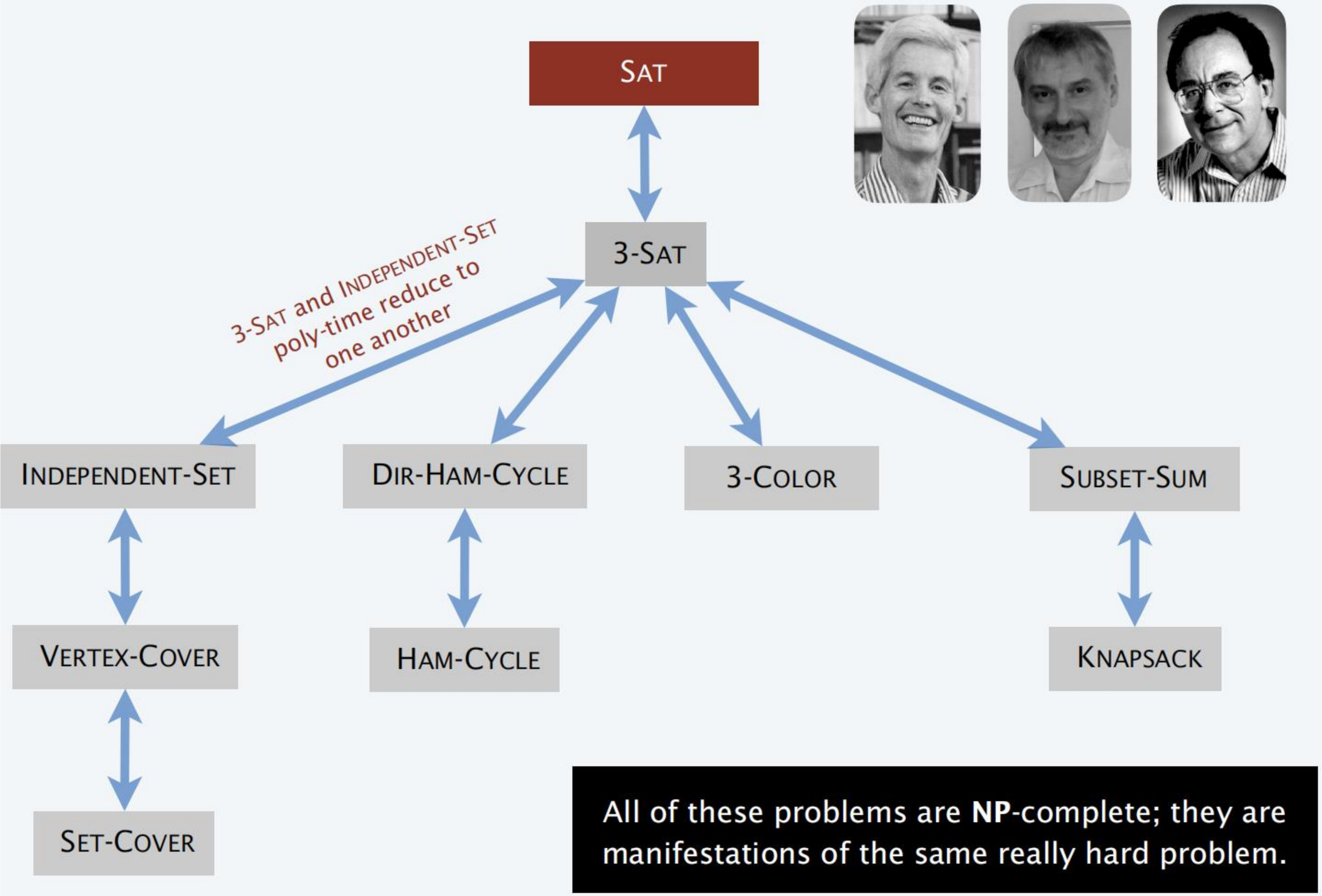
Karp的贡献


SAT
3-SAT
3-SAT poly-time reduces to INDEPENDENT-SET
INDEPENDENT-SET
DIR-HAM-CYCLE
3-COLOR
SUBSET-SUM
VERTEX-COVER
HAM-CYCLE
KNAPSACK
SET-COVER
SAT poly-time reduces to all of these problems (and many, many more)

Cook-Levin的贡献



Cook-Levin + Karp



不可计算性-2

- P vs. NP
- NP完全
- NP难

NP难的定义（共识）

NP-complete. A problem in **NP** such that every problem in **NP** poly-time reduces to it.

NP-hard. [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]

A problem such that every problem in **NP** poly-time reduces to it.

One final criticism (which applies to all the terms suggested) was stated nicely by Vaughan Pratt: "If the Martians know that P = NP for Turing Machines and they kidnap me, I would lose face calling these problems 'formidable'." Yes; if P = NP, there's no need for any term at all. But I'm willing to risk such an embarrassment, and in fact I'm willing to give a prize of one live turkey to the first person who proves that P = NP.