CSCI 355: Algorithm Design and Analysis
10. Intractability

- *poly-time reductions*
- *P and NP*
- *NP-completeness*
- *P vs. NP?*

Algorithm design patterns and antipatterns

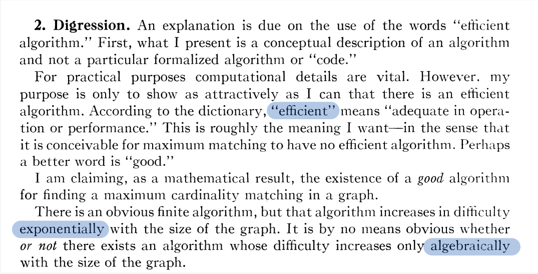
Algorithm design patterns.

- Greedy.
- Divide and conquer.
- Dynamic programming.
- Duality.
- Reductions.
- Local search.
- Randomization.

Algorithm design antipatterns.

• **NP**-completeness.	$O(n^k)$ algorithm unlikely.
• **PSPACE**-completeness.	$O(n^k)$ certification algorithm unlikely.
• Undecidability.	No algorithm possible.

Historical significance: Edmonds' *Paths, Trees, and Flowers*, 1965

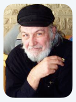


2. Digression. An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."

I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

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

Classifying problems according to computational requirements

Q. Which problems will we be able to solve in practice?
A working definition. Those with poly-time algorithms.

Turing machine, word RAM, uniform circuits, …

Theory. Definition is broad and robust.

constants tend to be small, e.g., $3n^2$

Practice. Poly-time algorithms scale to huge problems.

Classifying problems according to computational requirements

Q. Which problems will we be able to solve in practice?
A working definition. Those with poly-time algorithms.

yes	(probably) no
shortest path	longest path
min cut	max cut
2-satisfiability	3-satisfiability
planar 4-colourability	planar 3-colourability
bipartite vertex cover	vertex cover
matching	3d-matching
primality testing	factoring
linear programming	integer linear programming

Classifying problems

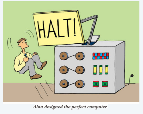
Desiderata. Classify problems according to those that can be solved in polynomial time and those that cannot.

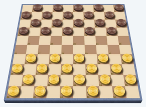
Problems that provably require exponential time.

input size = $c + \log k$

- Given a constant-size program, does it halt in at most k steps?
- Given a board position in an n-by-n generalization of checkers, can black guarantee a win?

using forced capture rule





Frustrating news. Huge number of fundamental problems have defied classification for decades.

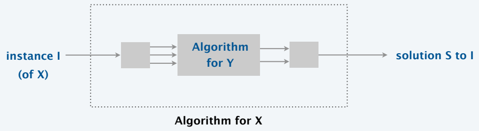
Poly-time reductions

Precise desiderata. Suppose we could solve a problem Y in polynomial time. What other problems could we solve in polynomial time?

Reduction. Problem X is polynomial-time reducible to problem Y if arbitrary instances of problem X can be solved using:

- a polynomial number of standard computational steps, plus
- a polynomial number of calls to an oracle that solves problem Y.

computational model supplemented by special piece of hardware that solves instances of Y in a single step



Poly-time reductions

Precise desiderata. Suppose we could solve a problem Y in polynomial time. What other problems could we solve in polynomial time?

Reduction. Problem X is polynomial-time reducible to problem Y if arbitrary instances of problem X can be solved using:

- a polynomial number of standard computational steps, plus
- a polynomial number of calls to an oracle that solves problem Y.

Notation. $X \leq_P Y$.

Note. We pay for the time to write down instances of Y sent to oracle ⇒ instances of Y must be of polynomial size.

Common mistake. Confusing $X \leq_P Y$ with $Y \leq_P X$.

Poly-time reductions

Designing algorithms. If $X \leq_P Y$ and Y can be solved in polynomial time, then X can be solved in polynomial time.

Establishing intractability. If $X \leq_P Y$ and X cannot be solved in polynomial time, then Y cannot be solved in polynomial time.

Proving equivalence. If both $X \leq_P Y$ and $Y \leq_P X$, then X can be solved in polynomial time iff Y can be solved in polynomial time; we write $X \equiv_P Y$.

Bottom line. Reductions classify problems according to relative difficulty.

Examples of problems

Satisfiability.

- **SAT.** Given a CNF formula Φ, does it have a satisfying truth assignment?
- **3-SAT.** An instance of SAT where each clause contains exactly 3 literals (and each literal corresponds to a different variable).

Packing and covering.

- **INDEPENDENT-SET.** Given a graph $G = (V, E)$ and an integer k, is there a subset of k (or more) vertices such that no two are adjacent?
- **VERTEX-COVER.** Given a graph $G = (V, E)$ and an integer k, is there a subset of k (or fewer) vertices such that each edge is incident to at least one vertex in the subset?
- **SET-COVER.** Given a set U of elements, a collection S of subsets of U, and an integer k, are there $\leq k$ of these subsets whose union is equal to U?

Examples of problems

Sequencing.

- **HAMILTON-CYCLE.** Given an undirected graph $G = (V, E)$, does there exist a cycle Γ that visits every vertex exactly once?
- **DIRECTED-HAMILTON-CYCLE.** Given a directed graph $G = (V, E)$, does there exist a directed cycle Γ that visits every vertex exactly once?

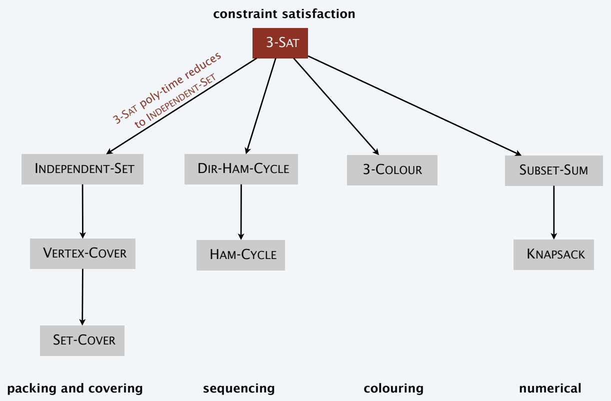
Colouring.

- **3-COLOUR.** Given an undirected graph G, can the vertices be coloured black, white, and blue so that no adjacent vertices have the same colour?

Numerical.

- **SUBSET-SUM.** Given n natural numbers $w_1, \ldots, w_n$ and an integer W, is there a subset that adds up to exactly W?
- **KNAPSACK.** Given $2n$ natural numbers $w_1, \ldots, w_n, v_1, \ldots, v_n$ and an integer W, is there a subset that maximizes v_i while adding up all values w_i to exactly W?

Tree of poly-time reductions between problems



constraint satisfaction

3-SAT

3-SAT poly-time reduces to INDEPENDENT-SET

INDEPENDENT-SET

DIR-HAM-CYCLE

3-COLOUR

SUBSET-SUM

VERTEX-COVER

HAM-CYCLE

KNAPSACK

SET-COVER

packing and covering

sequencing

colouring

numerical

Karp's *Reducibility Among Combinatorial Problems*, 1972

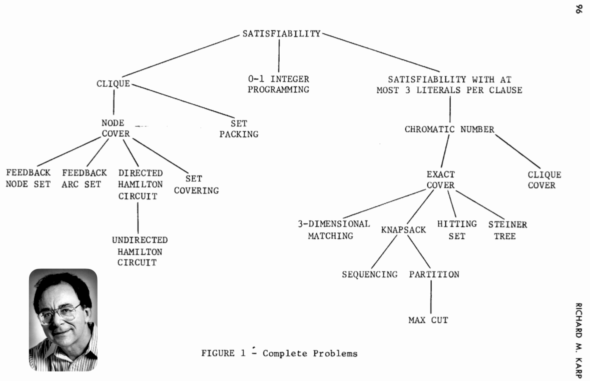


FIGURE 1 - Complete Problems

Karp
1985 Turing Award

CSCI 355: Algorithm Design and Analysis
10. Intractability

- poly-time reductions
- *P and NP*
- NP-completeness
- P vs. NP?

The class P

Decision problems.

- A problem X is a set of strings.
- An instance s of a problem is one string.
- An 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 $\leq p(|s|)$ "steps," where $p(\cdot)$ is some polynomial function.

length of s

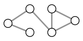
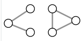
Def. **P** = set of decision problems for which there exists a poly-time algorithm.

on a deterministic Turing machine

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

Some problems in P

P. Set of decision problems for which there exists a poly-time algorithm.

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		

The class NP

Def. An 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 $C(s, t) = yes$.

"certificate" or "witness"

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: $|t| \le p(|s|)$ for some polynomial $p(\cdot)$.

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

Certifiers and certificates: satisfiability

SAT. Given a CNF formula Φ, does it have a satisfying truth assignment?
3-SAT. SAT where each clause contains exactly 3 literals.

Certificate. An assignment of truth values to the Boolean variables.

Certifier. Checks that each clause in Φ has at least one true literal.

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	$x_1 = true, \; x_2 = true, \; x_3 = false, \; x_4 = false$

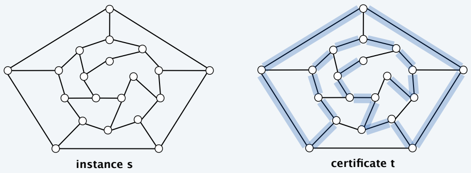
Conclusions. SAT ∈ **NP**, 3-SAT ∈ **NP**.

Certifiers and certificates: Hamiltonian path

HAMILTON-PATH. Given an undirected graph $G = (V, E)$, does there exist a simple path P that visits every vertex?

Certificate. A permutation π of the n vertices.

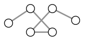
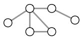
Certifier. Checks that π contains each vertex in V exactly once, and that G contains an edge between each pair of adjacent vertices.



Conclusion. HAMILTON-PATH ∈ **NP**.

Some problems in NP

NP. Set of decision problems for which there exists a poly-time certifier.

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 vertex?	???		

Significance of NP

NP. Set of decision problems for which there exists a poly-time certifier.

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

The classes P, NP, and EXP

P. Set of decision problems for which there exists a poly-time algorithm.
NP. Set of decision problems for which there exists a poly-time certifier.
EXP. Set of decision problems for which there exists an exp-time algorithm.

Proposition. **P** ⊆ **NP**.

Pf. Consider any problem $X \in$ **P**.

- By definition, there exists a poly-time algorithm $A(s)$ that solves X.
- Certificate is $t = \varepsilon$, certifier is $C(s, t) = A(s)$. ▪

Proposition. **NP** ⊆ **EXP**.

Pf. Consider any problem $X \in$ **NP**.

- By definition, there exists a poly-time certifier $C(s, t)$ for X where a certificate t satisfies $|t| \le p(|s|)$ for some polynomial $p(\cdot)$.
- To solve the instance s, run $C(s, t)$ on all strings t with $|t| \le p(|s|)$.
- Return *yes* iff $C(s, t)$ returns *yes* for any of these potential certificates. ▪

Fact. **P** ≠ **EXP** ⇒ either **P** ≠ **NP**, or **NP** ≠ **EXP**, or both.

CSCI 355: Algorithm Design and Analysis
10. Intractability

- *poly-time reductions*
- *P and NP*
- *NP-completeness*
- *P vs. NP?*

NP-completeness

NP-completeness. A problem $Y \in$ **NP** is NP-complete if it has the property that for every problem $X \in$ **NP**, $X \le_P Y$.

Proposition. Suppose $Y \in$ **NP**-complete. Then $Y \in$ **P** iff **P** = **NP**.

Pf.

[⇐] If **P** = **NP**, then $Y \in$ **P** because $Y \in$ **NP**.

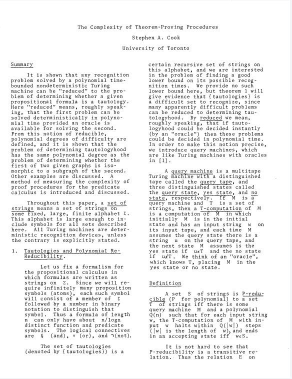
[⇒] Suppose $Y \in$ **P**.

- Consider any problem $X \in$ **NP**. Since $X \le_P Y$, we have $X \in$ **P**.
- This implies **NP** ⊆ **P**.
- We already know **P** ⊆ **NP**. Thus **P** = **NP**. ▪

Fundamental question. Are there any "natural" **NP**-complete problems?

The first NP-complete problem

Theorem. [Cook 1971, Levin 1973] SAT ∈ **NP**-complete.



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).

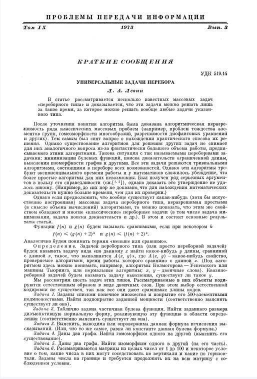
The set of tautologies (denoted by {tautologies}) is a certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a 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

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

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.

Establishing NP-completeness

Remark. Once we establish the first "natural" **NP**-complete problem, the others fall like dominoes.

Recipe. To prove that $Y \in$ **NP**-complete:

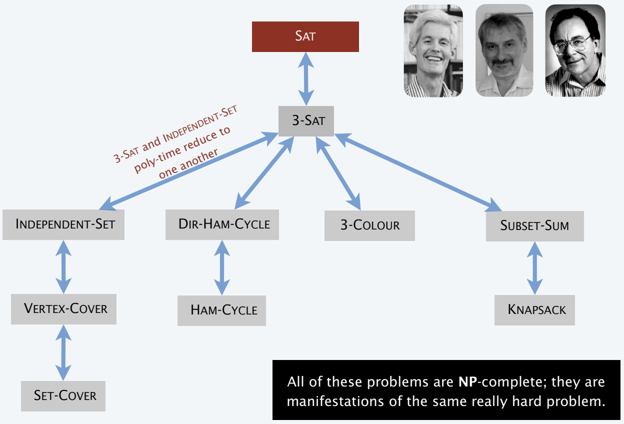
- Step 1. Show that $Y \in$ **NP**.
- Step 2. Choose an **NP**-complete problem X.
- Step 3. Prove that $X \le_P Y$.

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

Pf. Consider any problem $W \in$ **NP**. Then, both $\underset{\substack{\uparrow \\ \text{by definition of} \\ \textbf{NP}\text{-completeness}}}{W \le_P X}$ and $\underset{\substack{\uparrow \\ \text{by assumption}}}{X \le_P Y}$.

- By transitivity, $W \le_P Y$.
- Hence $Y \in$ **NP**-complete. ▪

Implications of Karp + Cook–Levin



All of these problems are **NP**-complete; they are manifestations of the same really hard problem.

Some NP-complete problems

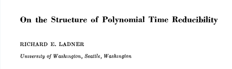
Basic classes of NP-complete problems and examples.

- Packing/covering problems: Set-Cover, Vertex-Cover, Independent-Set.
- Constraint satisfaction problems: Sat, 3-sat, Circuit-Sat.
- Sequencing problems: Hamilton-Cycle, TSP.
- Partitioning problems: 3-Colour, 3d-Matching.
- Numerical problems: Subset-Sum, Knapsack.

Practice. Most **NP** problems are known to be either in **P** or **NP**-complete.

"NP-intermediate" problems? Factor, Discrete-Log, Graph-Isomorphism, …

Theorem. [Ladner 1975] Unless **P** = **NP**, there exist problems in **NP** that are neither in **P** nor **NP**-complete.



On the Structure of Polynomial Time Reducibility

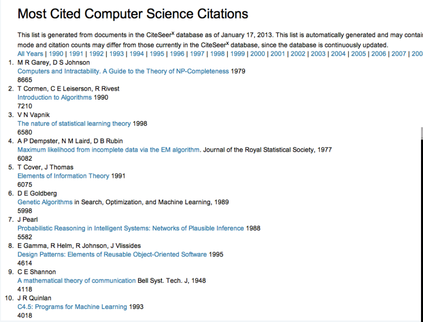
RICHARD E. LADNER

University of Washington, Seattle, Washington

More hard computational problems

M. R. Garey and D. S. Johnson. *Computers and Intractability.*

- Appendix includes over 300 **NP**-complete problems.
- Most cited reference in computer science literature.

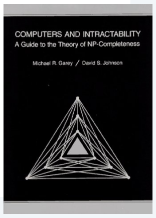


Most Cited Computer Science Citations

This list is generated from documents in the CiteSeer^x database as of January 17, 2013. This list is automatically generated and may contain errors. The list is generated in batch mode and citation counts may differ from those currently in the CiteSeer^x database, since the database is continuously updated.

All Years | 1990 | 1991 | 1992 | 1993 | 1994 | 1995 | 1996 | 1997 | 1998 | 1999 | 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013

1. M R Garey, D S Johnson
   Computers and Intractability. A Guide to the Theory of NP-Completeness 1979
   8665
2. T Cormen, C E Leiserson, R Rivest
   Introduction to Algorithms 1990
   7210
3. V N Vapnik
   The nature of statistical learning theory 1998
   6580
4. A P Dempster, N M Laird, D B Rubin
   Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, 1977
   6082
5. T Cover, J Thomas
   Elements of Information Theory 1991
   6075
6. D E Goldberg
   Genetic Algorithms in Search, Optimization, and Machine Learning, 1989
   5998
7. J Pearl
   Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference 1988
   5582
8. E Gamma, R Helm, R Johnson, J Vlissides
   Design Patterns: Elements of Reusable Object-Oriented Software 1995
   4614
9. C E Shannon
   A mathematical theory of communication Bell Syst. Tech. J, 1948
   4118
10. J R Quinlan
    C4.5: Programs for Machine Learning 1993
    4018


COMPUTERS AND INTRACTABILITY
A Guide to the Theory of NP-Completeness
Michael R. Garey / David S. Johnson

More hard computational problems

Aerospace engineering. Optimal mesh partitioning for finite elements.

Biology. Phylogeny reconstruction.

Chemical engineering. Heat exchanger network synthesis.

Chemistry. Protein folding.

Civil engineering. Equilibrium of urban traffic flow.

Economics. Computation of arbitrage in financial markets with friction.

Electrical engineering. VLSI layout.

Environmental engineering. Optimal placement of contaminant sensors.

Financial engineering. Minimum risk portfolio of given return.

Game theory. Nash equilibrium that maximizes social welfare.

Mathematics. Given integer $a_1, \ldots, a_n$, compute $\int_0^{2\pi} \cos(a_1\theta) \times \cos(a_2\theta) \times \cdots \times \cos(a_n\theta)\, d\theta$

Mechanical engineering. Structure of turbulence in sheared flows.

Medicine. Reconstructing 3d shape from biplane angiocardiogram.

Operations research. Traveling salesperson problem.

Physics. Partition function of 3d Ising model.

Politics. Shapley–Shubik voting power.

Recreation. Versions of Sudoku, Checkers, Minesweeper, Tetris, Rubik's Cube.

Statistics. Optimal experimental design.

CSCI 355: ALGORITHM DESIGN AND ANALYSIS
10. INTRACTABILITY

- poly-time reductions
- P and NP
- NP-completeness
- P vs. NP?

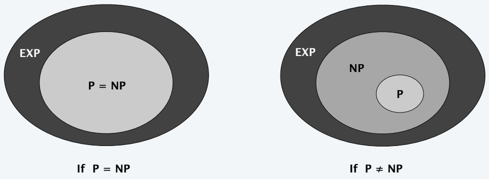
The big question: P vs. NP

Q. How do we solve an instance of 3-SAT with n variables?
A. Exhaustive search: try all 2^n truth assignments.

Q. Can we do anything substantially more clever?
Conjecture. There exists $\underbrace{\text{no poly-time algorithm}}_{\text{“intractable”}}$ for 3-SAT.

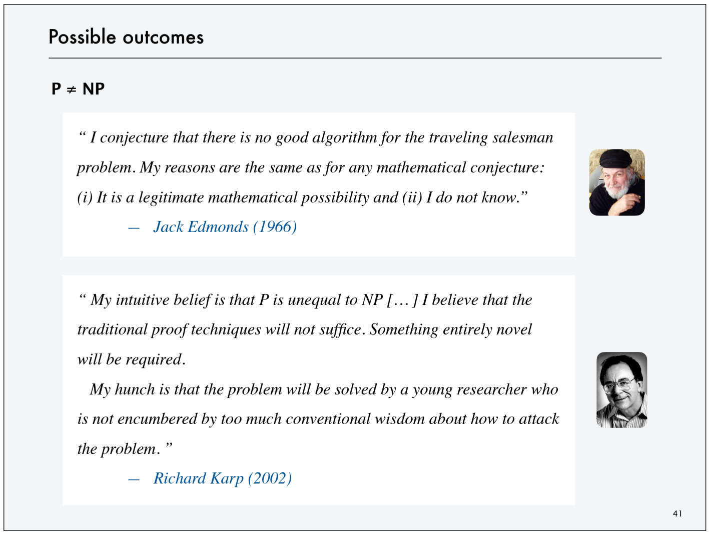
The big question: P vs. NP

Does P = NP? [Cook, Levin, ...]
Is the decision problem as easy as the certification problem?



If yes... Efficient algorithms exist for 3-SAT, TSP, VERTEX-COVER, FACTOR, ...
If no... No efficient algorithms are possible for 3-SAT, TSP, VERTEX-COVER, ...

Consensus opinion. Probably no.

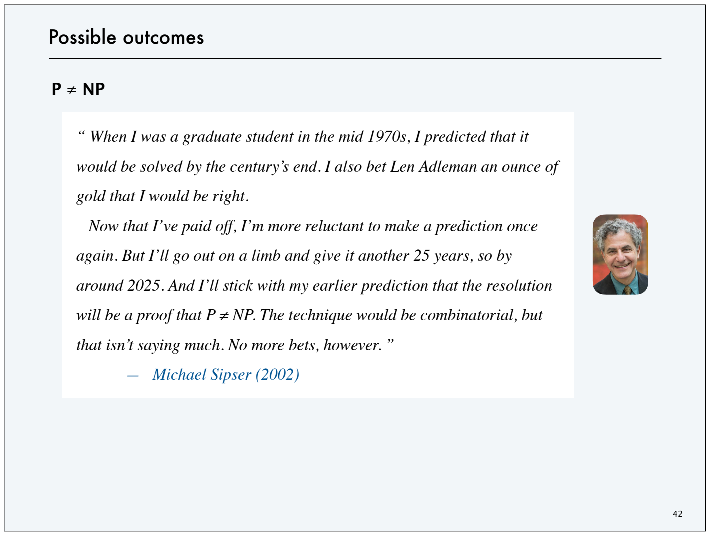


Possible outcomes

P ≠ 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)*

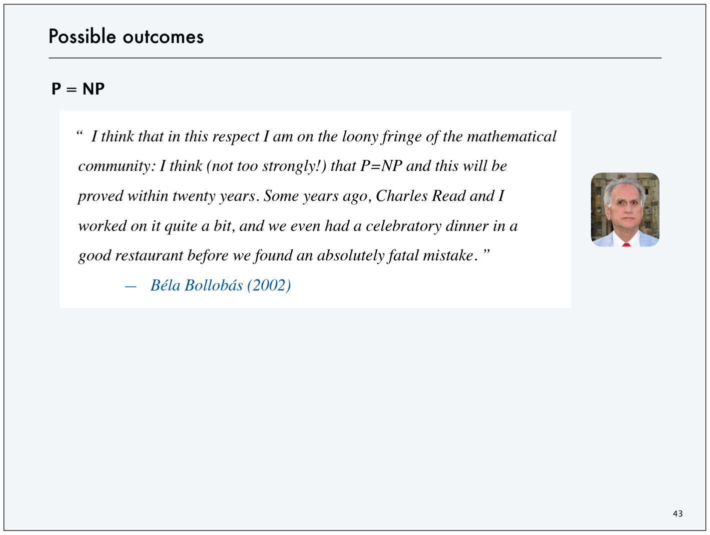
> *" My intuitive belief is that P is unequal to NP […] I believe that the traditional proof techniques will not suffice. Something entirely novel will be required.*
>
> *My hunch is that the problem will be solved by a young researcher who is not encumbered by too much conventional wisdom about how to attack the problem. "*
>
> *— Richard Karp (2002)*



Possible outcomes

P ≠ NP

> *" When I was a graduate student in the mid 1970s, I predicted that it would be solved by the century's end. I also bet Len Adleman an ounce of gold that I would be right.*
>
> *Now that I've paid off, I'm more reluctant to make a prediction once again. But I'll go out on a limb and give it another 25 years, so by around 2025. And I'll stick with my earlier prediction that the resolution will be a proof that P ≠ NP. The technique would be combinatorial, but that isn't saying much. No more bets, however. "*
>
> *— Michael Sipser (2002)*



Possible outcomes

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 a bit, and we even had a celebratory dinner in a good restaurant before we found an absolutely fatal mistake. "*
>
> *— Béla Bollobás (2002)*

Other possible outcomes

P = **NP**, but with a $\Omega(n^{100})$ algorithm for 3-SAT.

P ≠ **NP**, but with a $O(n^{\log^* n})$ algorithm for 3-SAT.

P = **NP** is independent of ZFC axiomatic set theory.

> *" It will be solved by either 2048 or 4096. I am currently somewhat pessimistic. The outcome will be the truly worst case scenario: namely that someone will prove P = NP because there are only finitely many obstructions to the opposite hypothesis; hence there exists a polynomial time solution to SAT but we will never know its complexity! "*
>
> — *Donald Knuth (2002)*

Other possible outcomes

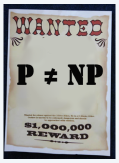
> *" I feel that theoretical computer scientists should devote a constant fraction of their lives to trying to resolve the P vs. NP question.*
>
> *I personally spend a few days each year thinking about it. I've proven (at least twice) that NP does not equal co-NP (and hence P does not equal NP). I've also proven (also at least twice) that NP equals co-NP.*
>
> *My most recent proof that NP does not equal co-NP occurred about a week ago as I write this, and the proof survived for about half an hour (not quite long enough for me to run it by someone else). My longest-surviving proof that NP does not equal co-NP survived for about 3 days and fooled some very smart people into believing it. "*
>
> — *Ronald Fagin (2002)*

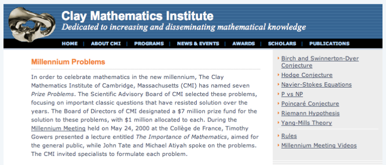
Millennium prize

Millennium Problems. $1 million for a resolution to the **P** vs. **NP** problem.

The only Millennium Problem relating to CS!

- Birch and Swinnerton-Dyer conjecture
- Hodge conjecture
- Navier-Stokes existence and smoothness
- **P vs. NP problem**
- ~~Poincaré conjecture~~ (solved)
- Riemann hypothesis
- Yang-Mills existence and mass gap




Clay Mathematics Institute
Dedicated to increasing and disseminating mathematical knowledge
HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

Millennium Problems

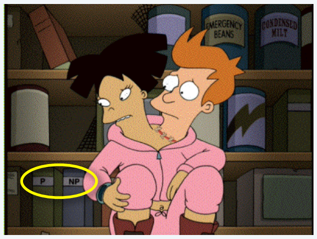
In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) has named seven *Prize Problems*. The Scientific Advisory Board of CMI selected these problems, focusing on important classic questions that have resisted solution over the years. 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.

P vs. NP and pop culture

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*).

Copyright © 1990, Matt Groening



Copyright © 2000, Twentieth Century Fox