8. INTRACTABILITY II

- P vs. NP
- NP-complete
- co-NP
- NP-hard
Recap

3-Sat poly-time reduces to INDEPENDENT-SET

INDEPENDENT-SET

VERTEX-COVER

SET-COVER

DIR-HAM-CYCLE

HAM-CYCLE

TSP

3-Sat poly-time reduces to all of these problems (and many, many more)

GRAPH-3-COLOR

PLANAR-3-COLOR

SUBSET-SUM

SCHEDULING
Section 8.3

8. Intractability II

- P vs. NP
- NP-complete
- co-NP
- NP-hard
Decision problems

**Decision problem.**

- Problem $X$ is a set of strings.
- Instance $s$ is one string.
- Algorithm $A$ solves problem $X$: $A(s) = \text{yes}$ iff $s \in X$.

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

**Ex.**

- Problem $\text{PRIMES} = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, \ldots \}$.  
- Instance $s = 592335744548702854681$.  
- AKS algorithm $\text{PRIMES}$ in $O(|s|^8)$ steps.
### Definition of P

**P.** Decision problems for which there is a poly-time algorithm.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Algorithm</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MULTIPLE</strong></td>
<td>Is $x$ a multiple of $y$ ?</td>
<td>grade-school division</td>
<td>51, 17</td>
<td>51, 16</td>
</tr>
<tr>
<td><strong>REL-PRIME</strong></td>
<td>Are $x$ and $y$ relatively prime ?</td>
<td>Euclid (300 BCE)</td>
<td>34, 39</td>
<td>34, 51</td>
</tr>
<tr>
<td><strong>PRIMES</strong></td>
<td>Is $x$ prime ?</td>
<td>AKS (2002)</td>
<td>53</td>
<td>51</td>
</tr>
<tr>
<td><strong>EDIT-DISTANCE</strong></td>
<td>Is the edit distance between $x$ and $y$ less than 5 ?</td>
<td>dynamic programming</td>
<td>neither</td>
<td>acgggt ttttta</td>
</tr>
<tr>
<td><strong>L-SOLVE</strong></td>
<td>Is there a vector $x$ that satisfies $Ax = b$ ?</td>
<td>Gauss-Edmonds elimination</td>
<td>[0 1 1] , [4]</td>
<td>[1 0 0] , [1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[2 4 -2] , [2]</td>
<td>[1 1 1] , [1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[0 3 15] , [36]</td>
<td>[0 1 1] , [1]</td>
</tr>
<tr>
<td><strong>ST-CONN</strong></td>
<td>Is there a path between $s$ and $t$ in a graph $G$ ?</td>
<td>depth-first search (Theseus)</td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
**Certification algorithm intuition.**
- Certifier views things from "managerial" viewpoint.
- Certifier doesn't determine whether $s \in X$ on its own; rather, it checks a proposed proof $t$ that $s \in X$.

**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 $C(s, t) = \text{yes}$.

**Def.** **NP** is the set of 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 $p(\cdot)$

**Remark.** **NP** stands for **nondeterministic** polynomial time.
Certifiers and certificates: composite

**COMPOSITES.** Given an integer $s$, is $s$ composite?

**Certificate.** A nontrivial factor $t$ of $s$. Such a certificate exists iff $s$ is composite. Moreover $|t| \leq |s|$.

**Certifier.** Check that $1 < t < s$ and that $s$ is a multiple of $t$.

<table>
<thead>
<tr>
<th>instance $s$</th>
<th>437669</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificate $t$</td>
<td>541 or 809</td>
</tr>
</tbody>
</table>

\[ 437,669 = 541 \times 809 \]

**Conclusion.** **COMPOSITES $\in$ NP.**
Certifiers and certificates: 3-satisfiability

\textbf{3-Sat.} Given a CNF formula $\Phi$, is there a satisfying assignment?

\textbf{Certificate.} An assignment of truth values to the $n$ boolean variables.

\textbf{Certifier.} Check that each clause in $\Phi$ has at least one true literal.

\[
\begin{align*}
\text{instance } s & \quad \Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4) \\
\text{certificate } t & \quad x_1 = \text{true}, \; x_2 = \text{true}, \; x_3 = \text{false}, \; x_4 = \text{false}
\end{align*}
\]

\textbf{Conclusion.} $\text{3-Sat} \in \text{NP}$. 
Certifiers and certificates: Hamilton path

**HAM-Path.** Given an undirected graph $G = (V, E)$, does there exist a simple path $P$ that visits every node?

**Certificate.** A permutation of the $n$ nodes.

**Certifier.** Check that the permutation contains each node in $V$ exactly once, and that there is an edge between each pair of adjacent nodes.

**Conclusion.** $\text{HAM-Path} \in \text{NP}$. 
Definition of NP

**NP.** Decision problems for which there is a poly-time certifier.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Algorithm</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L-SOLVE</strong></td>
<td>Is there a vector $x$ that satisfies $Ax = b$ ?</td>
<td>Gauss-Edmonds elimination</td>
<td>$\begin{bmatrix} 0 &amp; 1 &amp; 1 \ 2 &amp; 4 &amp; -2 \ 0 &amp; 3 &amp; 15 \end{bmatrix}$, $\begin{bmatrix} 4 \ 2 \ 36 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 \ 1 &amp; 1 &amp; 1 \end{bmatrix}$, $\begin{bmatrix} 1 \ 1 \ 1 \end{bmatrix}$</td>
</tr>
<tr>
<td><strong>COMPOSITES</strong></td>
<td>Is $x$ composite ?</td>
<td>AKS (2002)</td>
<td>51</td>
<td>53</td>
</tr>
<tr>
<td><strong>FACTOR</strong></td>
<td>Does $x$ have a nontrivial factor less than $y$ ?</td>
<td>?</td>
<td>(56159, 50)</td>
<td>(55687, 50)</td>
</tr>
<tr>
<td><strong>SAT</strong></td>
<td>Is there a truth assignment that satisfies the formula ?</td>
<td>?</td>
<td>$\neg x_1 \lor x_2$</td>
<td>$\neg x_2$</td>
</tr>
<tr>
<td><strong>3-COLOR</strong></td>
<td>Can the nodes of a graph $G$ be colored with 3 colors?</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HAM-PATH</strong></td>
<td>Is there a simple path between $s$ and $t$ that visits every node?</td>
<td>?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Definition of NP

**NP.** Decision problems for which there is a poly-time certifier.

“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, and EXP

P. Decision problems for which there is a poly-time algorithm.

NP. Decision problems for which there is a poly-time certifier.

EXP. Decision problems for which there is an exponential-time algorithm.

Claim. $P \subseteq NP$.

Pf. Consider any problem $X \in P$.
   • By definition, there exists a poly-time algorithm $A(s)$ that solves $X$.
   • Certificate $t = \varepsilon$, certifier $C(s, t) = A(s)$.

Claim. $NP \subseteq EXP$.

Pf. Consider any problem $X \in NP$.
   • By definition, there exists a poly-time certifier $C(s, t)$ for $X$.
   • To solve input $s$, run $C(s, t)$ on all strings $t$ with $|t| \leq p(|s|)$.
   • Return yes if $C(s, t)$ returns yes for any of these potential certificates.

Remark. Time-hierarchy theorem implies $P \nsubseteq EXP$. 

The main question:  P vs. NP

Q. How to 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. No poly-time algorithm for 3-SAT.

"intractable"
The main question:  P vs. NP

Does $P = NP$?  [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
Is the decision problem as easy as the certification problem?

If $P = NP$
Efficient algorithms for 3-SAT, TSP, 3-COLOR, FACTOR, ...

If $P \neq NP$
No efficient algorithms possible for 3-SAT, TSP, 3-COLOR, ...

Consensus opinion.  Probably no.
Possible outcomes

$P \neq NP.$

“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
Possible outcomes

\( P \neq NP. \)

“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

“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
Possible outcomes

P = NP.

“P = NP. 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
Other possible outcomes

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

P $\neq$ NP, but with $O(n^{\log^* n})$ 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 will exists a polynomial time solution to SAT but we will never know its complexity! ”  — Donald Knuth
Millennium prize.

$1 \text{ million for resolution of } P = \text{NP} \text{ problem.}$
Looking for a job?

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).*
Princeton CS Building, West Wall, Circa 2001

<table>
<thead>
<tr>
<th>char</th>
<th>ASCII</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>80</td>
<td>1010000</td>
</tr>
<tr>
<td>=</td>
<td>61</td>
<td>0111101</td>
</tr>
<tr>
<td>N</td>
<td>78</td>
<td>1001110</td>
</tr>
<tr>
<td>P</td>
<td>80</td>
<td>1010000</td>
</tr>
<tr>
<td>?</td>
<td>63</td>
<td>0111111</td>
</tr>
</tbody>
</table>
8. Intractability II

- P vs. NP
- NP-complete
- co-NP
- NP-hard

Section 8.4
Polynomial transformation

Def. Problem $X$ polynomial (Cook) reduces to problem $Y$ if arbitrary instances of problem $X$ can be solved using:
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem $Y$.

Def. Problem $X$ polynomial (Karp) transforms to problem $Y$ if given any input $x$ to $X$, we can construct an input $y$ such that $x$ is a yes instance of $X$ iff $y$ is a yes instance of $Y$.

Note. Polynomial transformation is polynomial reduction with just one call to oracle for $Y$, exactly at the end of the algorithm for $X$. Almost all previous reductions were of this form.

Open question. Are these two concepts the same with respect to $\text{NP}$?
NP-complete

A problem $Y \in \mathbf{NP}$ with the property that for every problem $X \in \mathbf{NP}$, $X \leq_p Y$.

**Theorem.** Suppose $Y \in \mathbf{NP}$-complete. Then $Y \in \mathbf{P}$ iff $\mathbf{P} = \mathbf{NP}$.

**Pf.** $\Leftarrow$ If $\mathbf{P} = \mathbf{NP}$, then $Y \in \mathbf{P}$ because $Y \in \mathbf{NP}$.

**Pf.** $\Rightarrow$ Suppose $Y \in \mathbf{P}$.

- Consider any problem $X \in \mathbf{NP}$. Since $X \leq_p Y$, we have $X \in \mathbf{P}$.
- This implies $\mathbf{NP} \subseteq \mathbf{P}$.
- We already know $\mathbf{P} \subseteq \mathbf{NP}$. Thus $\mathbf{P} = \mathbf{NP}$.

**Fundamental question.** Do there exist "natural" $\mathbf{NP}$-complete problems?
Circuit satisfiability

**CIRCUIT-SAT.** Given a combinational circuit built from AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?

![Circuit Diagram]

- **Output:**
  - \( \land \)
  - \( \neg \)
  - \( \land \)
  - \( \lor \)
  - \( \lor \)

- **Hard-coded inputs:**
  - 1
  - 0

- **Variable inputs:**
  - ?

**Yes:** 1 0 1
The "first" NP-complete problem

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

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, propositional degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same propositional 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 \( \Sigma \). This alphabet is large enough to include symbols for all sets described here. All Turing machines are determinisitic recognition devices, unless the contrary is explicitly stated.

1. Tautologies and Polynomial Reducibility

Let us fix a formalism for the propositional calculus in which formulas are written as strings on \( \Sigma \). Since we will require infinitely many proposition symbols (atoms), each such symbol will consist of a member of \( \Sigma \). Let \( \Sigma \) be a set of strings in binary notation to distinguish that symbol. Thus a formula of length \( n \) can only have about \( 2^n \) distinct functions and predicate symbols. The logical connectives are \( \land \) (and), \( \lor \) (or), and \( \lnot \) (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 introduce the notion of reducibility, 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. In particular, if \( M \) is a query machine and \( T \) is a set of strings on some alphabet \( \Sigma \), then a computation of \( M \) is a computation of \( M \) in which initially \( M \) is in the initial state and there is an input string \( w \) on its input tape, and each line \( M \) assumes the query state there is a string \( u \) on the query tape, and the next state \( M \) assumes is the yes state if \( w\{u \} \) and the no state if \( w\{u \} \). 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 \((P \text{ for polynomial})\) to a set \( T \) of strings if there is a TM \( M \) that for each input string \( w \), the \( T \)-computation of \( M \) with input \( w \) halts within \( O(|w|^k) \) steps \((|w| \text{ is the length of } w) \) and ends in an accepting state iff \( w \in S \).

It is not hard to see that P-reducibility is a transitive relation. Thus the relation \( \in \) on

PRIME

1. PRIME

2. CIRCUIT-SAT

3. Tautologies

4. P

5. \( \Sigma \)

6. \( \Lambda \)

7. \( \Omega \)

8. \( \Theta \)

9. \( \Delta \)

10. \( \Gamma \)

11. \( \Phi \)

12. \( \Psi \)

13. \( \Xi \)

14. \( \Upsilon \)

15. \( \Omicron \)

16. \( \Pi \)

17. \( \Sigma \)

18. \( \Lambda \)

19. \( \Theta \)

20. \( \Delta \)

21. \( \Gamma \)

22. \( \Phi \)

23. \( \Psi \)

24. \( \Xi \)

25. \( \Upsilon \)

26. \( \Omicron \)

27. \( \Pi \)

28. \( \Sigma \)

29. \( \Lambda \)

30. \( \Theta \)

31. \( \Delta \)

32. \( \Gamma \)

33. \( \Phi \)

34. \( \Psi \)

35. \( \Xi \)

36. \( \Upsilon \)

37. \( \Omicron \)

38. \( \Pi \)

39. \( \Sigma \)

40. \( \Lambda \)

41. \( \Theta \)

42. \( \Delta \)

43. \( \Gamma \)

44. \( \Phi \)

45. \( \Psi \)

46. \( \Xi \)

47. \( \Upsilon \)

48. \( \Omicron \)

49. \( \Pi \)

50. \( \Sigma \)

51. \( \Lambda \)

52. \( \Theta \)

53. \( \Delta \)

54. \( \Gamma \)

55. \( \Phi \)

56. \( \Psi \)

57. \( \Xi \)

58. \( \Upsilon \)

59. \( \Omicron \)

60. \( \Pi \)
The "first" NP-complete problem

Theorem. \textsc{Circuit-Sat} $\in \textbf{NP}$-complete.

Pf sketch.

\begin{itemize}
  \item Clearly, \textsc{Circuit-Sat} $\in \textbf{NP}$.
  \item Any algorithm that takes a fixed number of bits $n$ as input and produces a \textit{yes} or \textit{no} answer can be represented by such a circuit.
  \item Moreover, if algorithm takes poly-time, then circuit is of poly-size.
\end{itemize}

\textit{sketchy part of proof; fixing the number of bits is important, and reflects basic distinction between algorithms and circuits}

\begin{itemize}
  \item Consider any problem $X \in \textbf{NP}$. It has a poly-time certifier $C(s, t)$:
    $s \in X$ iff there exists a certificate $t$ of length $p(|s|)$ such that $C(s, t) = \text{yes}$.
  \item View $C(s, t)$ as an algorithm with $|s| + p(|s|)$ input bits and convert it into a poly-size circuit $K$.
    \begin{itemize}
      \item first $|s|$ bits are hard-coded with $s$
      \item remaining $p(|s|)$ bits represent (unknown) bits of $t$
    \end{itemize}
  \item Circuit $K$ is satisfiable iff $C(s, t) = \text{yes}$.
\end{itemize}
Ex. Construction below creates a circuit $K$ whose inputs can be set so that it outputs 1 iff graph $G$ has an independent set of size 2.

$G = (V, E), n = 3$
Establishing NP-completeness

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

Recipe. To prove that $Y \in \text{NP}$-complete:
- Step 1. Show that $Y \in \text{NP}$.
- Step 2. Choose an NP-complete problem $X$.
- Step 3. Prove that $X \leq_p Y$.

Theorem. If $X \in \text{NP}$-complete, $Y \in \text{NP}$, and $X \leq_p Y$, then $Y \in \text{NP}$-complete.

Pf. Consider any problem $W \in \text{NP}$. Then, both $W \leq_p X$ and $X \leq_p Y$.
- By transitivity, $W \leq_p Y$.
- Hence $Y \in \text{NP}$-complete. □
3-satisfiability is NP-complete

**Theorem.** 3-SAT \(\in\) NP-complete.

**Pf.**

- Suffices to show that CIRCUIT-SAT \(\leq_p\) 3-SAT since 3-SAT \(\in\) NP.
- Given a combinational circuit \(K\), we construct an instance \(\Phi\) of 3-SAT that is satisfiable iff the inputs of \(K\) can be set so that it outputs 1.
3-satisfiability is NP-complete

**Construction.** Let $K$ be any circuit.

**Step 1.** Create a 3-SAT variable $x_i$ for each circuit element $i$.

**Step 2.** Make circuit compute correct values at each node:
- $x_2 = \neg x_3 \Rightarrow$ add 2 clauses: $x_2 \lor x_3$, $\neg x_2 \lor \neg x_3$
- $x_1 = x_4 \lor x_5 \Rightarrow$ add 3 clauses: $x_1 \lor \neg x_4$, $x_1 \lor \neg x_5$, $\neg x_1 \lor x_4 \lor x_5$
- $x_0 = x_1 \land x_2 \Rightarrow$ add 3 clauses: $\neg x_0 \lor x_1$, $\neg x_0 \lor x_2$, $x_0 \lor \neg x_1 \lor \neg x_2$

**Step 3.** Hard-coded input values and output value.
- $x_5 = 0 \Rightarrow$ add 1 clause: $\neg x_5$
- $x_0 = 1 \Rightarrow$ add 1 clause: $x_0$
3-satisfiability is NP-complete

Construction. [continued]

Step 4. Turn clauses of length 1 or 2 into clauses of length 3.
  • Create four new variables $z_1$, $z_2$, $z_3$, and $z_4$.
  • Add 8 clauses to force $z_1 = z_2 = false$:
    $$(\overline{z_1} \lor z_3 \lor z_4), \ (\overline{z_1} \lor z_3 \lor \overline{z_4}), \ (\overline{z_1} \lor \overline{z_3} \lor z_4), \ (\overline{z_1} \lor \overline{z_3} \lor \overline{z_4})$$
    $$(z_2 \lor z_3 \lor z_4), \ (z_2 \lor z_3 \lor \overline{z_4}), \ (z_2 \lor \overline{z_3} \lor z_4), \ (z_2 \lor \overline{z_3} \lor \overline{z_4})$$
  • Replace any clause with a single term $(t_i)$ with $(t_i \lor z_1 \lor z_2)$.
  • Replace any clause with two terms $(t_i \lor t_j)$ with $(t_i \lor t_j \lor z_1)$.
3-satisfiability is NP-complete

Lemma. Φ is satisfiable iff the inputs of K can be set so that it outputs 1.

Pf. ⇐ Suppose there are inputs of K that make it output 1.
   • Can propagate input values to create values at all nodes of K.
   • This set of values satisfies Φ.

Pf. ⇒ Suppose Φ is satisfiable.
   • We claim that the set of values corresponding to the circuit inputs constitutes a way to make circuit K output 1.
   • The 3-SAT clauses were designed to ensure that the values assigned to all node in K exactly match what the circuit would compute for these nodes. □
Implications of Karp

CIRCUIT-SAT

3-Sat

INDEPENDENT-SET

DIR-HAM-CYCLE

GRAPH-3-COLOR

SUBSET-SUM

VERTEX-COVER

HAM-CYCLE

PLANAR-3-COLOR

SCHEDULING

CIRCUIT-SAT poly-time reduces to all of these problems (and many, many more)
Implications of Cook-Levin

3-SAT poly-time reduces to INDEPENDENT-SET

3-SAT

CIRCUIT-SAT

All of these problems (and many, many more) poly-time reduce to CIRCUIT-SAT.
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 genres of NP-complete problems and paradigmatic examples.

- Packing + covering problems: SET-COVER, VERTEX-COVER, INDEPENDENT-SET.
- Constraint satisfaction problems: CIRCUIT-SAT, SAT, 3-SAT.
- Sequencing problems: HAM-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING, 3-COLOR.
- Numerical problems: SUBSET-SUM, PARTITION.

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

Notable exceptions. FACTOR, GRAPH-ISOMORPHISM, NASH-EQUILIBRIUM.

Theory. [Ladner 1975] Unless P = NP, there exist problems in NP that are neither in P nor NP-complete.
More hard computational problems

**Garey and 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 CiteSeerX 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 CiteSeerX database, since the database is continuously updated.


1. M R Garey, D S Johnson
   8595

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
   6082

5. T Cover, J Thomas
   *Elements of Information Theory* 1991
   6075

6. D E Goldberg
   5996

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
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.
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.
Statistics. Optimal experimental design.
Extent and impact of NP-completeness

Extent of NP-completeness. [Papadimitriou 1995]
- Prime intellectual export of CS to other disciplines.
- 6,000 citations per year (more than "compiler", "OS", "database").
- Broad applicability and classification power.

NP-completeness can guide scientific inquiry.
- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager finds closed-form solution to $2\text{D-ISING}$ in tour de force.
- 19xx: Feynman and other top minds seek solution to $3\text{D-ISING}$.
- 2000: Istrail proves $3\text{D-ISING} \in \text{NP-complete}$.

search for closed formula appears doomed

a holy grail of statistical mechanics
P vs. NP revisited

Overwhelming consensus (still). $P \neq NP$.

Why we believe $P \neq NP$.

“We admire Wiles' proof of Fermat's last theorem, the scientific theories of Newton, Einstein, Darwin, Watson and Crick, the design of the Golden Gate bridge and the Pyramids, precisely because they seem to require a leap which cannot be made by everyone, let alone a by simple mechanical device.” — Avi Wigderson
You NP-complete me
8. **Intractability II**

- $P$ vs. $NP$
- $NP$-complete
- $co-NP$
- $NP$-hard
Asymmetry of NP

Asymmetry of NP. We only need to have short proofs of yes instances.

Ex 1. SAT vs. TAUTOLOGY.
- Can prove a CNF formula is satisfiable by specifying an assignment.
- How could we prove that a formula is not satisfiable?

Ex 2. HAM-CYCLE vs. NO-HAM-CYCLE.
- Can prove a graph is Hamiltonian by specifying a permutation.
- How could we prove that a graph is not Hamiltonian?

Q. How to classify TAUTOLOGY and NO-HAMILTON-CYCLE?
- SAT ∈ NP-complete and SAT ≡ _p_ TAUTOLOGY.
- HAM-CYCLE ∈ NP-complete and HAM-CYCLE ≡ _p_ NO-HAM-CYCLE.
- But neither TAUTOLOGY nor NO-HAM-CYCLE are known to be in NP.
**NP and co-NP**

**NP.** Decision problems for which there is a poly-time certifier.

**Ex.** SAT, HAMILTON-CYCLE, and COMPOSITE.

**Def.** Given a decision problem $X$, its complement $\overline{X}$ is the same problem with the yes and no answers reverse.

**Ex.**  

$X = \{ 0, 1, 4, 6, 8, 9, 10, 12, 14, 15, \ldots \}$  

$\overline{X} = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, \ldots \}$

**co-NP.** Complements of decision problems in NP.

**Ex.** TAUTOLOGY, NO-HAMILTON-CYCLE, and PRIMES.
**NP = co-NP?**

---

**Fundamental open question.** Does $\text{NP} = \text{co-NP}$?

- Do *yes* instances have succinct certificates iff *no* instances do?
- Consensus opinion: no.

**Theorem.** If $\text{NP} \neq \text{co-NP}$, then $\text{P} \neq \text{NP}$.

**Pf idea.**

- $\text{P}$ is closed under complementation.
- If $\text{P} = \text{NP}$, then $\text{NP}$ is closed under complementation.
- In other words, $\text{NP} = \text{co-NP}$.
- This is the contrapositive of the theorem.
Good characterizations

**Good characterization.** [Edmonds 1965] $\textbf{NP} \cap \textbf{co-NP}$.

- If problem $X$ is in both $\textbf{NP}$ and $\textbf{co-NP}$, then:
  - for *yes* instance, there is a succinct certificate
  - for *no* instance, there is a succinct disqualifier

- Provides conceptual leverage for reasoning about a problem.

**Ex.** Given a bipartite graph, is there a perfect matching.

- If yes, can exhibit a perfect matching.
- If no, can exhibit a set of nodes $S$ such that $|N(S)| < |S|$. 

---

*A matroid $M$ is a finite set $M$ of elements with a family of subsets, called independent, such that (1) every subset of an independent set is independent, and (2) for every subset $A$ of $M$, all maximal independent subsets of $A$ have the same cardinality, called the rank $r(A)$ of $A$. It is proved that a matroid can be partitioned into as few as $k$ sets, each independent, if and only if every subset $A$ has cardinality at most $k \cdot r(A)$.**
Good characterizations

We seek a good characterization of the minimum number of independent sets into which the columns of a matrix of $M_F$ can be partitioned. As the criterion of “good” for the characterization we apply the “principle of the absolute supervisor.” The good characterization will describe certain information about the matrix which the supervisor can require his assistant to search out along with a minimum partition and which the supervisor can then use with ease to verify with mathematical certainty that the partition is indeed minimum. Having a good characterization does not mean necessarily that there is a good algorithm. The assistant might have to kill himself with work to find the information and the partition.
Good characterizations

**Observation.** \( P \subseteq \text{NP} \cap \text{co-NP} \).
- Proof of max-flow min-cut theorem led to stronger result that max-flow and min-cut are in \( P \).
- Sometimes finding a good characterization seems easier than finding an efficient algorithm.

**Fundamental open question.** Does \( P = \text{NP} \cap \text{co-NP} \)?
- Mixed opinions.
- Many examples where problem found to have a nontrivial good characterization, but only years later discovered to be in \( P \).
Linear programming is in $\text{NP} \cap \text{co-NP}$

**Linear programming.** Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$, does there exist $x \in \mathbb{R}^n$ such that $Ax \leq b$, $x \geq 0$ and $c^T x \geq \alpha$?

**Theorem.** [Gale-Kuhn-Tucker 1948] \textsc{Linear-Programming} $\in \text{NP} \cap \text{co-NP}$.

**Pf sketch.** If (P) and (D) are nonempty, then max = min.

\[
\begin{align*}
(P) \quad \text{max } c^T x & \quad \text{(D) } \text{min } y^T b \\
\text{s. t. } Ax & \leq b & \text{s. t. } A^T y & \geq c \\
& \quad x \geq 0 & \quad y \geq 0
\end{align*}
\]
Linear programming is in $\text{NP} \cap \text{co-NP}$

**Linear programming.** Given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^{m}$, $c \in \mathbb{R}^{n}$, and $\alpha \in \mathbb{R}$, does there exist $x \in \mathbb{R}^{n}$ such that $Ax \leq b$, $x \geq 0$ and $c^T x \geq \alpha$?

**Theorem.** [Khachiyan 1979] **Linear-Programming** $\in \text{P}$. 

---

**ЖУРНАЛ**
**ВЫЧИСЛИТЕЛЬНОЕ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ**

Том 20  /  Январь 1980  Февраль  /  № 1

УДК 519.852

**ПОЛИНОМИАЛЬНЫЕ АЛГОРИТМЫ В ЛИНЕЙНОМ**
**ПРОГРАММИРОВАНИИ**

А. Г. ХАЧЯН

(Москва)

Построены точные алгоритмы линейного программирования, трудоемкость которых ограничена полиномом от длины двоичной записи задач.
Primality testing is in \( NP \cap \text{co-NP} \)

**Theorem.** [Pratt 1975] \( \text{PRIMES} \in NP \cap \text{co-NP}. \)

---

*EVERY PRIME HAS A SUCCINCT CERTIFICATE*  

VAUGHAN R. PRATT†

**Abstract.** To prove that a number \( n \) is composite, it suffices to exhibit the working for the multiplication of a pair of factors. This working, represented as a string, is of length bounded by a polynomial in \( \log_2 n \). We show that the same property holds for the primes. It is noteworthy that almost no other set is known to have the property that short proofs for membership or nonmembership exist for all candidates without being known to have the property that such proofs are easy to come by. It remains an open problem whether a prime \( n \) can be recognized in only \( \log^2 n \) operations of a Turing machine for any fixed \( x \).

The proof system used for certifying primes is as follows.

**Axiom.** \((x, y, 1)\).  

**Inference Rules.**

\[ R_1 : (p, x, a), q \vdash (p, x, qa) \quad \text{provided } x^{(n-1)/q} \not\equiv 1 \pmod{p} \text{ and } q|(p - 1). \]

\[ R_2 : (p, x, p - 1) \vdash p \quad \text{provided } x^{p-1} \equiv 1 \pmod{p}. \]

**Theorem 1.** \( p \) is a theorem \( \equiv \) \( p \) is a prime.  
**Theorem 2.** \( p \) is a theorem \( \Rightarrow \) \( p \) has a proof of \( [4 \log_2 p] \) lines.
Primality testing is in $\text{NP} \cap \text{co-NP}$

**Theorem.** [Pratt 1975] \( \text{PRIMES} \in \text{NP} \cap \text{co-NP}. \)

**Pf sketch.** An odd integer \( s \) is prime iff there exists an integer \( 1 < t < s \) s.t.

\[
\begin{align*}
 t^{s-1} & \equiv 1 \pmod{s} \\
 t^{(s-1)/p} & \not\equiv 1 \pmod{s}
\end{align*}
\]

for all prime divisors \( p \) of \( s-1 \)

\[
\begin{align*}
\text{instance } s & \quad 437677 \\
\text{certificate } t & \quad 17, \ 2^2 \times 3 \times 36473
\end{align*}
\]

```
\text{CERTIFIER} (s)
```

```
\text{CHECK} \quad s - 1 = 2 \times 2 \times 3 \times 36473.
```

```
\text{CHECK} \quad 17^{s-1} \equiv 1 \pmod{s}.
```

```
\text{CHECK} \quad 17^{(s-1)/2} \equiv 437676 \pmod{s}.
```

```
\text{CHECK} \quad 17^{(s-1)/3} \equiv 329415 \pmod{s}.
```

```
\text{CHECK} \quad 17^{(s-1)/36,473} \equiv 305452 \pmod{s}.
```

prime factorization of \( s-1 \)
also need a recursive certificate to assert that 3 and 36,473 are prime

```
\text{use repeated squaring}
```


Theorem. [Agrawal-Kayal-Saxena 2004] PRIMES ∈ P.


PRIMES is in \( P \)

By Manindra Agrawal, Neeraj Kayal, and Nitin Saxena*

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.
Factoring is in $\text{NP} \cap \text{co-NP}$

**Factorize.** Given an integer $x$, find its prime factorization.

**Factor.** Given two integers $x$ and $y$, does $x$ have a nontrivial factor $< y$?

**Theorem.** $\text{Factor} \equiv_P \text{Factorize}.$

**Pf.**
- $\leq_P$ trivial.
- $\geq_P$ binary search to find a factor; divide out the factor and repeat.

**Theorem.** $\text{Factor} \in \text{NP} \cap \text{co-NP}.$

**Pf.**
- Certificate: a factor $p$ of $x$ that is less than $y$.
- Disqualifier: the prime factorization of $x$ (where each prime factor is less than $y$), along with a Pratt certificate that each factor is prime.
Is factoring in P?

**Fundamental question.** Is \textsc{Factor} \( \in \mathbf{P} \).

**Challenge.** Factor this number.

\begin{align*}
74037563479561712828046796097429573142593188889231289 \\
08493623263897276503402826627689199641962511784399589 \\
43305021275853701189680982867331732731089309005525051 \\
16877063299072396380786710086096962537934650563796359
\end{align*}

\textsc{RSA-704}  
($30,000$ prize if you can factor)
Exploiting intractability

Modern cryptography.
- Ex. Send your credit card to Amazon.
- Ex. Digitally sign an e-document.
- Enables freedom of privacy, speech, press, political association.

RSA. Based on dichotomy between complexity of two problems.
- To use: generate two random $n$-bit primes and multiply.
- To break: suffices to factor a $2n$-bit integer.

\[
\begin{align*}
P & \not\equiv Q \text{ PRIME} \\
N &= PQ \\
\text{ED} &= 1 \mod (P-1)(Q-1) \\
C &= M^E \mod N \\
M &= C^D \mod N
\end{align*}
\]

The RSA algorithm is the most widely used method of implementing public key cryptography and has been deployed in more than one billion applications worldwide.

RSA algorithm

RSA sold for $2.1$ billion

or design a t-shirt
Factoring on a quantum computer

Theorem. [Shor 1994] Can factor an $n$-bit integer in $O(n^3)$ steps on a "quantum computer."

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer

Abstract. A digital computer is generally believed to be an efficient universal computing device; that is, it is believed to be able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems that are generally thought to be hard on classical computers and that have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, for example, the number of digits of the integer to be factored.

2001. Factored $15 = 3 \times 5$ (with high probability) on a quantum computer.
2012. Factored $21 = 3 \times 7$.

Fundamental question. Does $P = BQP$?
8. **Intractability II**

- $P$ vs. $NP$
- $NP$-complete
- $co-NP$
- $NP$-hard
A note on terminology

A TERMINOLOGICAL PROPOSAL

D. F. Knuth

While preparing a book on combinatorial algorithms, I felt a strong need for a new technical term, a word which is essentially a one-sided version of polynomial complete. A great many problems of practical interest have the property that they are at least as difficult to solve in polynomial time as those of the Cook-Karp class NP. I needed an adjective to convey such a degree of difficulty, both formally and informally; and since the range of practical applications is so broad, I felt it would be best to establish such a term as soon as possible.

The goal is to find an adjective $x$ that sounds good in sentences like this:

The covering problem is $x$.
It is $x$ to decide whether a given graph has a Hamiltonian circuit.
It is unknown whether or not primality testing is an $x$ problem.

Note. The term $x$ does not necessarily imply that a problem is in NP, just that every problem in NP poly-time reduces to $x$. 
A note on terminology

Knuth's original suggestions.

- Hard.
- Tough.
- Herculean.
- Formidable.
- Arduous.

so common that it is unclear whether it is being used in a technical sense

assign a real number between 0 and 1 to each choice
A note on terminology

Some English word write-ins.

• Impractical.
• Bad.
• Heavy.
• Tricky.
• Intricate.
• Prodigious.
• Difficult.
• Intractable.
• Costly.
• Obdurate.
• Obstinate.
• Exorbitant.
• Interminable.
A note on terminology

**Hard-boiled.** [Ken Steiglitz] In honor of Cook.


**Sisyphean.** [Bob Floyd] Problem of Sisyphus was time-consuming.

**Ulyssean.** [Don Knuth] Ulysses was known for his persistence.

“creative research workers are as full of ideas for new terminology as they are empty of enthusiasm for adopting it.”

— Donald Knuth
A note on terminology: acronyms

PET. [Shen Lin] Probably exponential time.
  • If $P \neq NP$, provably exponential time.
  • If $P = NP$, previously exponential time.

GNP. [Al Meyer] Greater than or equal to NP in difficulty.
  • And costing more than the GNP to solve.
A note on terminology: made-up words

**Exparent.** [Mike Paterson] Exponential + apparent.

**Perarduous.** [Mike Paterson] Through (in space or time) + completely.

**Supersat.** [Al Meyer] Greater than or equal to satisfiability.

**Polychronious.** [Ed Reingold] Enduringly long; chronic.
A note on terminology: consensus

**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** polynomial-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 $P = NP$. 