Chapter 8

NP and Computational Intractability
8.3 Definition of NP
Decision Problems

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

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

PRIMES: $X = \{2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, \ldots\}$
### 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>MULTIPLE</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>RELPRIME</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>PRIMES</td>
<td>Is x prime?</td>
<td>AKS (2002)</td>
<td>53</td>
<td>51</td>
</tr>
<tr>
<td>EDIT-DISTANCE</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>LSOLVE</td>
<td>Is there a vector x that satisfies Ax = b?</td>
<td>Gauss-Edmonds elimination</td>
<td></td>
<td></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 $C(s, t) = \text{yes}$. "certificate" or "witness"

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

$C(s, t)$ is a poly-time algorithm and $|t| \leq p(|s|)$ 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 $s$. Note that such a certificate exists iff $s$ is composite. Moreover $|t| \leq |s|$.

**Certifier.**

```java
boolean C(s, t) {
    if (t ≤ 1 or t ≥ s)
        return false
    else if (s is a multiple of t)
        return true
    else
        return false
}
```

**Instance.** $s = 437,669$.

**Certificate.** $t = 541$ or $809$. $437,669 = 541 \times 809$

**Conclusion.** COMPOSITES is in NP.
Certifiers and Certificates: 3-Satisfiability

**SAT.** Given a CNF formula \( \Phi \), is there a satisfying assignment?

**Certificate.** An assignment of truth values to the \( n \) boolean variables.

**Certifier.** Check that each clause in \( \Phi \) has at least one true literal.

Ex.

\[
\left( \overline{x_1} \lor x_2 \lor x_3 \right) \land \left( x_1 \lor \overline{x_2} \lor x_3 \right) \land \left( x_1 \lor x_2 \lor x_4 \right) \land \left( \overline{x_1} \lor \overline{x_3} \lor \overline{x_4} \right)
\]

instance \( s \)

\[
x_1 = 1, \ x_2 = 1, \ x_3 = 0, \ x_4 = 1
\]

certificate \( t \)

**Conclusion.** \( \text{SAT} \) is in \( \text{NP} \).
Certifiers and Certificates: Hamiltonian Cycle

**HAM-CYCLE.** Given an undirected graph $G = (V, E)$, does there exist a simple cycle $C$ 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 in the permutation.

**Conclusion.** **HAM-CYCLE** is in NP.
P. Decision problems for which there is a poly-time algorithm.
EXP. Decision problems for which there is an exponential-time algorithm.
NP. Decision problems for which there is a poly-time certifier.

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 \( \text{yes} \), if \( C(s, t) \) returns \( \text{yes} \) for any of these. □
The Main Question: P Versus NP

**Does P = NP?** [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
- Is the decision problem as easy as the certification problem?
- Clay $1 million prize.

If yes: **Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, ...**
If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, ...

Consensus opinion on P = NP? Probably no.

would break RSA cryptography (and potentially collapse economy)
The Simpson's: $P = NP$?

Copyright © 1990, Matt Groening
Futurama: $P = NP$?
Looking for a Job?

Some writers for the Simpsons and Futurama.

8.4 NP-Completeness
Def. Problem X polynomial reduces (Cook) 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 transforms (Karp) 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.

\[ \text{we require } |y| \text{ to be of size polynomial in } |x| \]

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? We abuse notation \( \leq_p \) and blur distinction.
NP-Complete

**NP-complete.** A problem $Y$ in NP with the property that for every problem $X$ in NP, $X \leq_p Y$.

**Theorem.** Suppose $Y$ is an NP-complete problem. Then $Y$ is solvable in poly-time iff $P = NP$.

**Pf.** $\Leftarrow$ If $P = NP$ then $Y$ can be solved in poly-time since $Y$ is in NP.

**Pf.** $\Rightarrow$ Suppose $Y$ can be solved in poly-time.

- Let $X$ be any problem in NP. Since $X \leq_p Y$, we can solve $X$ in poly-time. This implies $NP \subseteq P$.
- We already know $P \subseteq NP$. Thus $P = NP$. □

**Fundamental question.** Do there exist "natural" NP-complete problems?
Circuit Satisfiability

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

![Circuit Diagram]

**yes:** 1 0 1

**hard-coded inputs**

**inputs**
The "First" NP-Complete Problem

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

**Pf.** (sketch)
- Any algorithm that takes a fixed number of bits \( n \) as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.

  sketchy part of proof: fixing the number of bits is important, and reflects basic distinction between algorithms and circuits

- Consider some problem \( X \) in NP. It has a poly-time certifier \( C(s, t) \).
  To determine whether \( s \) is in \( X \), need to know if there exists a certificate \( t \) of length \( p(|s|) \) such that \( C(s, t) = \text{yes} \).
- View \( C(s, t) \) as an algorithm on \( |s| + p(|s|) \) bits (input \( s \), certificate \( t \)) and convert it into a poly-size circuit \( K \).
  - first \( |s| \) bits are hard-coded with \( s \)
  - remaining \( p(|s|) \) bits represent bits of \( t \)
- Circuit \( K \) is satisfiable iff \( C(s, t) = \text{yes} \).
Example

Ex. Construction below creates a circuit $K$ whose inputs can be set so that $K$ outputs true iff graph $G$ has an independent set of size 2.

$G = (V, E), n = 3$

$(\binom{n}{2})$ hard-coded inputs (graph description)  \hspace{1cm} n inputs (nodes in independent set)
Establishing NP-Completeness

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

Recipe to establish NP-completeness of problem $Y$.

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

Justification. If $X$ is an NP-complete problem, and $Y$ is a problem in NP with the property that $X \leq_p Y$ then $Y$ is NP-complete.

Pf. Let $W$ be any problem in NP. Then $W \leq_p X \leq_p Y$.

- By transitivity, $W \leq_p Y$.
- Hence $Y$ is NP-complete. □
Theorem. 3-SAT is NP-complete.

Pf. Suffices to show that CIRCUIT-SAT \( \leq_p \) 3-SAT since 3-SAT is in NP.

- Let \( K \) be any circuit.
- Create a 3-SAT variable \( x_i \) for each circuit element \( i \).
- Make circuit compute correct values at each node:
  - \( x_2 = \neg x_3 \) \( \Rightarrow \) add 2 clauses: \( x_2 \lor x_3, \bar{x}_2 \lor \bar{x}_3 \)
  - \( x_1 = x_4 \lor x_5 \) \( \Rightarrow \) add 3 clauses: \( x_1 \lor \bar{x}_4, x_1 \lor \bar{x}_5, \bar{x}_1 \lor x_4 \lor x_5 \)
  - \( x_0 = x_1 \land x_2 \) \( \Rightarrow \) add 3 clauses: \( \bar{x}_0 \lor x_1, \bar{x}_0 \lor x_2, x_0 \lor \bar{x}_1 \lor \bar{x}_2 \)

- Hard-coded input values and output value.
  - \( x_5 = 0 \) \( \Rightarrow \) add 1 clause: \( \bar{x}_5 \)
  - \( x_0 = 1 \) \( \Rightarrow \) add 1 clause: \( x_0 \)

- Final step: turn clauses of length < 3 into clauses of length exactly 3.
**Observation.** All problems below are NP-complete and polynomial reduce to one another!

NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce to one another!
Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

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

Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.
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 (title, abstract, keywords).
  - more than "compiler", "operating system", "database"
- Broad applicability and classification power.
- "Captures vast domains of computational, scientific, mathematical endeavors, and seems to roughly delimit what mathematicians and scientists had been aspiring to compute feasibly."

NP-completeness can guide scientific inquiry.

- 1926: Ising introduces simple model for phase transitions.
- 1944: Onsager solves 2D case in tour de force.
- 19xx: Feynman and other top minds seek 3D solution.
More Hard Computational Problems

**Aerospace engineering**: optimal mesh partitioning for finite elements.
**Biology**: protein folding.
**Chemical engineering**: heat exchanger network synthesis.
**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**: find minimum risk portfolio of given return.
**Game theory**: find Nash equilibrium that maximizes social welfare.
**Genomics**: phylogeny reconstruction.
**Mechanical engineering**: structure of turbulence in sheared flows.
**Medicine**: reconstructing 3-D shape from biplane angiocardiogram.
**Operations research**: optimal resource allocation.
**Physics**: partition function of 3-D Ising model in statistical mechanics.
**Politics**: Shapley-Shubik voting power.
**Pop culture**: Minesweeper consistency.
**Statistics**: optimal experimental design.
8.9 co-NP and the Asymmetry of NP
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 giving such 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 giving such a Hamiltonian cycle.
  - How could we prove that a graph is not Hamiltonian?

Remark. SAT is NP-complete and $SAT \equiv_p TAUTOLOGY$, but how do we classify $TAUTOLOGY$?

↑

not even known to be in NP
NP and co-NP

**NP.** Decision problems for which there is a poly-time certifier.
**Ex.** SAT, HAM-CYCLE, COMPOSITES.

**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-HAM-CYCLE, PRIMES.
Fundamental 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] $\text{NP} \cap \text{co-NP}$. 
- If problem $X$ is in both NP and 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 $|N(S)| < |S|$. 
Good Characterizations

**Observation.** $P \subseteq NP \cap co-NP$.

- 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 $P = NP \cap co-NP$?

- Mixed opinions.
- Many examples where problem found to have a non-trivial good characterization, but only years later discovered to be in $P$.
  - linear programming [Khachiyan, 1979]
  - primality testing [Agrawal-Kayal-Saxena, 2002]

**Fact.** Factoring is in $NP \cap co-NP$, but not known to be in $P$.

↑

if poly-time algorithm for factoring, can break RSA cryptosystem
Theorem. PRIMES is in NP \cap co-NP.

Pf. We already know that PRIMES is in co-NP, so it suffices to prove that PRIMES is in NP.

Pratt's Theorem. An odd integer s is prime iff there exists an integer 
1 < t < s \ s.t.

\[ t^{s-1} \equiv 1 \pmod{s} \]
\[ t^{(s-1)/p} \not\equiv 1 \pmod{s} \]

for all prime divisors p of s-1

Input. \( s = 437,677 \)

Certificate. \( t = 17, 2^2 \times 3 \times 36,473 \)

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

Certifier.
- Check \( s-1 = 2 \times 2 \times 3 \times 36,473 \).
- Check \( 17^{s-1} = 1 \pmod{s} \).
- Check \( 17^{(s-1)/2} \equiv 437,676 \pmod{s} \).
- Check \( 17^{(s-1)/3} \equiv 329,415 \pmod{s} \).
- Check \( 17^{(s-1)/36,473} \equiv 305,452 \pmod{s} \).

use repeated squaring
FACTOR is in NP \cap co-NP

FACTORIZE. Given an integer \( x \), find its prime factorization.

FACTOR. Given two integers \( x \) and \( y \), does \( x \) have a nontrivial factor less than \( y \)?

Theorem. \( \text{FACTOR} \equiv_p \text{FACTORIZE} \).

Theorem. \( \text{FACTOR} \) is in NP \( \cap \) 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 certificate that each factor is prime.
We established: \( \text{PRIMES} \leq_p \text{COMPOSITES} \leq_p \text{FACTOR} \).

Natural question: Does \( \text{FACTOR} \leq_p \text{PRIMES} \)?

Consensus opinion. No.

State-of-the-art.
- PRIMES is in P. \( \leftarrow \) proved in 2001
- FACTOR not believed to be in P.

RSA cryptosystem.
- Based on dichotomy between complexity of two problems.
- To use RSA, must generate large primes efficiently.
- To break RSA, suffixes to find efficient factoring algorithm.
Extra Slides
<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII</th>
<th>Bits</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>
Not How To Give a PowerPoint Talk

(commercial break)
A Note on Terminology

Knuth. [SIGACT News 6, January 1974, p. 12 - 18]

Find an adjective $x$ that sounds good in sentences like.

- EUCLIDEAN-TSP is $x$.
- It is $x$ to decide whether a given graph has a Hamiltonian cycle.
- It is unknown whether FACTOR is an $x$ problem.

**Note:** $x$ does not necessarily imply that a problem is in NP, just that every problem in NP polynomial reduces to $x$. 
A Note on Terminology

Knuth's original suggestions.
- Hard.
- Tough.
- Herculean. \[\text{but Hercules known for strength not time}\]
- Formidable.
- Arduous.

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.

but Sisyphus never finished his task

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

and finished!
A Note on Terminology: Made-Up Words

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

**Polychronious.** [Ed Reingold] Enduringly long; chronic.

*like today's lecture*

**PET.** [Shen Lin] Probably exponential time.

*depending on P=NP conjecture: provably exponential time,*

*or previously exponential time*

**GNP.** [Al Meyer] Greater than or equal to NP in difficulty.

*costing more than GNP to resolve*
A Note on Terminology: Consensus

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

**NP-hard.** [Bell Labs, Steve Cook, Ron Rivest, Sartaj Sahni]
A decision problem such that every problem in NP reduces to it.

not necessarily in NP

**NP-hard search problem.** A problem such that every problem in NP reduces to it.

not necessarily a yes/no problem

"creative research workers are as full of ideas for new terminology as they are empty of enthusiasm for adopting it." -Don Knuth