8. INTRACTABILITY II

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

Recap

3-SAT poly-time reduces to independent-set

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

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 function.

Ex.
- Problem PRIMES = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. \}.
- Instance $s = 59233744548702854681$.
- AKS algorithm: solves PRIMES in $O(|s|^6)$ 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>aacggt tttta</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 0]</td>
<td>[1 0 0 1]</td>
</tr>
<tr>
<td><strong>U-CONN</strong></td>
<td>Is an undirected graph $G$ connected?</td>
<td>depth-first search (Theseus)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $|t| \leq |s|/2$.

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

\[
\text{instance } s = 437669 \\
\text{certificate } t = 541 \text{ or } 809
\]

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

**Conclusion.** \text{COMPOSITES} $\in$ \text{NP}.

NP

**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.** \text{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.** \text{NP} stands for non-deterministic polynomial time.

Certifiers and certificates: satisfiability

**SAT.** Given a CNF formula $\Phi$, 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.** Check that each clause in $\Phi$ has at least one true literal.

\[\text{instance } \Phi = \left(\overline{x_1} \lor x_2 \lor x_3\right) \land \left(x_1 \lor \overline{x_2} \lor x_3\right) \land \left(x_1 \lor x_2 \lor \overline{x_3}\right)\]

\[\text{certificate } t = x_1 = \text{true}, \ x_2 = \text{true}, \ x_3 = \text{false}, \ x_4 = \text{false}\]

Conclusions. \text{SAT} $\in$ \text{NP}, 3-SAT $\in$ \text{NP}.
Certifiers and certificates: Hamilton path

**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>L-SOLVE</td>
<td>Is there a vector ( x ) that satisfies ( Ax = b )?</td>
<td>Gauss–Edmonds elimination</td>
<td>[0, 1, 1, 1]</td>
<td>[1, 0, 0]</td>
</tr>
<tr>
<td>COMPOSITES</td>
<td>Is ( x ) composite?</td>
<td>AKS (2002)</td>
<td>51</td>
<td>53</td>
</tr>
<tr>
<td>FACTOR</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>SAT</td>
<td>Given a CNF formula, does it have a satisfying truth assignment?</td>
<td>?</td>
<td>( \neg x_1 \lor \neg x_2 \lor \neg x_3 \lor x_1 \lor \neg x_2 \lor x_3 \lor \neg x_1 \lor x_2 \lor x_3 )</td>
<td></td>
</tr>
<tr>
<td>3-COLOR</td>
<td>Can the nodes of a graph ( G ) be colored with 3 colors?</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HAM-PATH</td>
<td>Is there a simple path between ( u ) and ( v ) that visits every node?</td>
<td>?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 \( X \),
  where certificate \( t \) satisfies \( |t| \leq p(|s|) \) for some polynomial \( p(.) \).
- 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 \( P \subseteq EXP \).
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.

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 (2002)

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

\[ P = \text{NP}. \]

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

— Béla Bollobás (2002)

Millennium prize

Millennium prize. \$1 million for resolution of \( P = \text{NP} \) problem.

Other possible outcomes

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

\[ P \neq \text{NP}, \] but with \( O(n^{\log n}) \) algorithm for 3-SAT.

\[ P = \text{NP} \] 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 \( P = \text{NP} \) 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

Looking for a job?

Some writers for the Simpsons and Futurama.

- J. Stewart 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).
- Jeff Westbrook. Ph.D. in computer science (Princeton ’89).
8. INTRACTABILITY II

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

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

**NP-complete.** A problem \( Y \in \text{NP} \) with the property that for every problem \( X \in \text{NP} \), \( X \leq_p Y \).

**Theorem.** Suppose \( Y \in \text{NP-complete} \). Then \( Y \in \text{P} \) iff \( \text{P} = \text{NP} \).

**Pf.** ⇐ If \( \text{P} = \text{NP} \), then \( Y \in \text{P} \) because \( Y \in \text{NP} \).

**Pf.** ⇒ Suppose \( Y \in \text{P} \).

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

**Fundamental question.** Do there exist “natural” 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?

**The "first" NP-complete problem**

**Theorem.** CIRCUIT-SAT \( \in \text{NP-complete} \). [Cook 1971, Levin 1973]

**Pf sketch.**

- Clearly, CIRCUIT-SAT \( \in \text{NP} \).
- Any algorithm that takes a fixed number of bits \( n \) as input and produces a yes or no answer can be represented by such a circuit.
- Moreover, if algorithm takes poly-time, then circuit is of poly-size.

- Consider any problem \( X \in \text{NP} \). It has a poly-time certifier \( C(s, t) \), where certificate \( t \) satisfies \( |t| \leq p(|s|) \) for some polynomial \( p(\cdot) \).
- View \( C(s, t) \) as an algorithm with \(|s| + p(|s|)\) input bits and convert it into a poly-size circuit \( K \).
  - first \(|s|\) bits are hard-coded with \( s \)
  - remaining \( p(|s|)\) bits represent (unknown) bits of \( t \)
- Circuit \( K \) is satisfiable iff \( C(s, t) = \text{yes} \).
Example

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.

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. $3\text{-SAT} \in \text{NP}$-complete.

Pf.
- Suffices to show that CIRCUIT-SAT $\leq_p 3\text{-SAT}$ since 3-SAT $\in \text{NP}$.
- 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.

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:
- $x_2 = \neg x_3$ => add 2 clauses: $x_2 \lor \neg x_3$, $\neg x_2 \lor x_3$
- $x_1 = x_4 \lor x_5$ => add 3 clauses: $x_1 \lor \neg x_4$, $x_1 \lor \neg x_5$, $x_1 \lor \neg x_4 \lor \neg x_5$
- $x_0 = x_1 \lor x_2$ => add 3 clauses: $x_0 \lor x_1$, $x_0 \lor x_2$, $x_0 \lor \neg x_1 \lor \neg x_2$

Step 3. Hard-coded input values and output value.
- $x_5 = 0$ => add 1 clause: $\neg x_5$
- $x_0 = 1$ => 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 \( z_1 = z_2 = 0: \)
  \[
  (\neg z_3 \lor z_3 \lor z_4), (\neg z_3 \lor z_3 \lor \neg z_4), (\neg z_3 \lor z_3 \lor z_4), (\neg z_3 \lor z_3 \lor \neg z_4),
  (\neg z_2 \lor z_2 \lor z_4), (\neg z_2 \lor z_2 \lor \neg z_4), (\neg z_2 \lor z_2 \lor z_4), (\neg z_2 \lor z_2 \lor \neg z_4),
  \]
- 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 \( (t_i \lor t_j \lor z_1). \)

Lemma. \( \Phi \) is satisfiable iff the inputs of \( \mathcal{K} \) can be set so that it outputs 1.

**Pf.** \( \Leftarrow \) Suppose there are inputs of \( \mathcal{K} \) that make it output 1.
- Can propagate input values to create values at all nodes of \( \mathcal{K}. \)
- This set of values satisfies \( \Phi. \)

**Pf.** \( \Rightarrow \) Suppose \( \Phi \) is satisfiable.
- We claim that the set of values corresponding to the circuit inputs constitutes a way to make circuit \( \mathcal{K} \) output 1.
- The 3-SAT clauses were designed to ensure that the values assigned to all node in \( \mathcal{K} \) exactly match what the circuit would compute for these nodes. \( \blacksquare \)

Implications of Karp

Circuit-SAT

3-SAT

Independent-Set

Vertex-Cover

Set-Cover

TSP

HAM-Cycle

Planar-3-Color

Scheduling

Graph-3-Color

Circuit-SAT poly-time reduces to all of these problems (and many, many more)

Implications of Cook-Levin

Circuit-SAT

3-SAT

Independent-Set

Vertex-Cover

Set-Cover

TSP

HAM-Cycle

Planar-3-Color

Scheduling

Graph-3-Color

All of these problems (and many, many more) poly-time reduce to Circuit-SAT.
Implications of Karp + Cook-Levin

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.

Theorem. [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.

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 $\prod_{i=1}^{n} \cos(a_i \theta) \times \cos(a_i \theta) \times \cdots \times \cos(a_i \theta)$.
Mechanical engineering. Structure of turbulence in sheared flows.
Medicine. Reconstructing 3d shape from biplane angiography.
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 2D-ISING in tour de force.
- 19xx: Feynman and other top minds seek solution to 3D-ISING.

8. INTRACTABILITY II
   - P vs. NP
   - NP-complete
   - co-NP
   - NP-hard

P vs. NP revisited

Overwhelming consensus (still). P ≠ NP.

Why we believe P ≠ 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 by a simple mechanical device."
— Avi Wigderson

You NP-complete me
Asymmetry of NP

Asymmetry of NP. We need short certificates only for yes instances.

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

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

**UN-SAT.** Given a CNF formula \( \Phi \), is there no satisfying truth assignment?

Asymmetry of NP

Asymmetry of NP. We need short certificates only for yes instances.

**Q.** How to classify UN-SAT and NO-HAM-CYCLE?
- \( \text{SAT} \in \text{NP} \)-complete and \( \text{SAT} \equiv_p \text{UN-SAT} \).
- \( \text{HAM-CYCLE} \in \text{NP} \)-complete and \( \text{HAM-CYCLE} \equiv_p \text{NO-HAM-CYCLE} \).
- But neither UN-SAT 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, HAM-CYCLE, and COMPOSITES.

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

**Ex.** \( X = \{ 4, 6, 8, 9, 10, 12, 14, 15, \ldots \} \)
\[ \overline{X} = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, \ldots \} \]

ignore 0 and 1 (neither prime nor composite)

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

**Ex.** UN-SAT, NO-HAM-CYCLE, and PRIMES.
**Good characterizations**

- **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 \( \text{P} \neq \text{NP} \).
  - 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**

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

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

**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] \textbf{Linear Programming} \( \in \mathbf{NP} \cap \mathbf{co-NP} \).

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

\begin{align*}
\text{(P)} & \quad \max c^T x \\
& \text{s.t.} \ Ax \leq b \\
& \quad x \geq 0 \\
\text{(D)} & \quad \min y^T b \\
& \text{s.t.} \ A^T y \geq c \\
& \quad y \geq 0
\end{align*}

Primality testing is in \( \mathbf{NP} \cap \mathbf{co-NP} \)

**Theorem.** [Pratt 1975] \textbf{Primes} \( \in \mathbf{NP} \cap \mathbf{co-NP} \).

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

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

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

\[
\text{Certifier} (s) \\
\text{Check} \quad s - 1 = 2 \times 2 \times 3 \times 36473, \\
\text{Check} \quad 17^{17-1} \equiv 437676 \pmod{s}, \\
\text{Check} \quad 17^{(s-1)/2} \equiv 329415 (\text{mod } s), \\
\text{Check} \quad 17^{(s-1)/3} \equiv 305452 \pmod{s}.
\]
Primality testing is in $P$

**Theorem.** [Agrawal-Kayal-Saxena 2004] $\text{PRIMES} \in P$.

---

**Abstract**

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

---

Factoring is in $NP \cap \text{co-NP}$

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

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

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

**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.

---

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.

---

Is factoring in $P$?

**Fundamental question.** Is $\text{FACTOR} \in P$?

**Challenge.** Factor this number.

74037563479561712828046796097429573142593188889231289
0849362326389727650340282662768919964196251178439589
43305021275853701189689828673317327310893005525051
1687706329907396380786720086096962537934650563796359

**RSA-704** ($\$30,000$ prize if you can factor)

---

Factoring is in $P \cap \text{co-NP}$

**THEOREM.** $\text{FACTOR} = P \text{FACTORIZE}$.  

**Pf.**

- $\leq$ trivial.
- $\geq$ binary search to find a factor; divide out the factor and repeat.

---

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."

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

Fundamental question. Does $P = BQP$?

A note on terminology

Knuth's original suggestions.

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

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
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: acronyms

PET. [Shen Lin] Probably exponential time.
- If P ≠ 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

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


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

Ulyssian. [Donald 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

Exparent. [Mike Paterson] Exponential + apparent.

Perarduous. [Mike Paterson] Throughout (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 $P = NP$. 