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: $X = \{ 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, \ldots \}$

Certification algorithm intuition.
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 \( C(s, t) = \text{yes} \).

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

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 \leq 1 or t \geq 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.

$$
\overline{x_1} \lor x_2 \lor x_3 \land (x_1 \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_1 \lor \overline{x_3} \lor \overline{x_4})
$$

instance $s$

$x_1 = 1, \ x_2 = 1, \ x_3 = 0, \ x_4 = 1$

certificate $t$

Conclusion. SAT is in 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: $t = \epsilon$, 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.
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 $P \neq NP$:
- No efficient algorithms possible for 3-COLOR, TSP, FACTOR, SAT, ...

If $P = NP$:
- Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, ...
- $P = NP$ would break RSA cryptography (and potentially collapse economy)

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.
8.4 NP-Completeness
Polynomial Transformation

**Def.** Problem Y *polynomial reduces* (Cook) to problem X if arbitrary instances of problem Y can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem X.

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

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

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

**Open question.** Are these two concepts the same?

\[ \text{we abuse notation } \leq^p \text{ and blur distinction} \]
NP-Complete

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

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

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

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

- Let $Y$ be any problem in NP. Since $Y \leq_p X$, we can solve $Y$ 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?

yes: 1 0 1
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.

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

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

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

\[ \binom{n}{2} \text{ hard-coded inputs (graph description)} \quad n \text{ inputs (nodes in independent set)} \]
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 ≤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: $x_2 \lor x_3, \overline{x_2} \lor \overline{x_3}$
  - $x_1 = x_4 \lor x_5 \Rightarrow$ add 3 clauses: $x_1 \lor \overline{x_4}, x_1 \lor \overline{x_5}, x_1 \lor \overline{x_4} \lor \overline{x_5}$
  - $x_0 = x_1 \land x_2 \Rightarrow$ add 3 clauses: $\overline{x_0} \lor x_1, \overline{x_0} \lor x_2, x_0 \lor \overline{x_1} \lor \overline{x_2}$

- Hard-coded input values and output value.
  - $x_5 = 0 \Rightarrow$ add 1 clause: $\overline{x_5}$
  - $x_0 = 1 \Rightarrow$ add 1 clause: $x_0$

- Final step: turn clauses of length < 3 into clauses of length exactly 3.  

\[ \overline{x_5} \lor \overline{x_4} \lor x_3, \overline{x_5} \lor \overline{?} \lor ? , \overline{?} \lor \overline{?} \lor x_0 \]
Observation. All problems below are NP-complete and polynomial reduce to one another!

by definition of NP-completeness
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.
8.9 co-NP and the Asymmetry of NP
Asymmetry of NP. We only need to have short proofs of \texttt{yes} instances.

\textbf{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 \texttt{not} satisfiable?

\textbf{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 \texttt{not} Hamiltonian?
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. $\overline{X} = \{ 0, 1, 4, 6, 8, 9, 10, 12, 14, 15, \ldots \}$

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

co-NP. Complements of decision problems in NP.
Ex. TAUTOLOGY, NO-HAM-CYCLE, PRIMES.
Fundamental question. Does NP = co-NP?

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

Theorem. If NP ≠ co-NP, then P ≠ NP.

Pf idea.

- P is closed under complementation.
- If P = NP, then NP is closed under complementation.
- In other words, NP = 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]
A Note on Terminology: Consensus

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

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

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