NP and Computational Intractability

P = NP?
Review

Basic reduction strategies.

- Simple equivalence: INDEPENDENT-SET \(\equiv_p\) VERTEX-COVER.
- Special case to general case: VERTEX-COVER \(\leq_p\) SET-COVER.
- Encoding with gadgets: 3-SAT \(\leq_p\) INDEPENDENT-SET.

Transitivity. If \(X \leq_p Y\) and \(Y \leq_p Z\), then \(X \leq_p Z\).

Pf idea. Compose the two algorithms.

Ex: 3-SAT \(\leq_p\) INDEPENDENT-SET \(\leq_p\) VERTEX-COVER \(\leq_p\) SET-COVER.
HAM-CYCLE: given an undirected graph $G = (V, E)$, does there exist a simple cycle $\Gamma$ that contains every node in $V$.

NO: bipartite graph with odd number of nodes.
**Directed Hamiltonian Cycle**

**DIR-HAM-CYCLE:** given a **digraph** $G = (V, E)$, does there exist a simple directed cycle $\Gamma$ that contains every node in $V$?

**Claim.** DIR-HAM-CYCLE $\leq_P$ HAM-CYCLE.

**Pf.** Given a directed graph $G = (V, E)$, construct an undirected graph $G'$ with $3n$ nodes.

![Diagram](image)
Directed Hamiltonian Cycle

Claim.  $G$ has a Hamiltonian cycle iff $G'$ does.

Pf.  $\Rightarrow$

- Suppose $G$ has a directed Hamiltonian cycle $\Gamma$.
- Then $G'$ has an undirected Hamiltonian cycle (same order).

Pf.  $\Leftarrow$

- Suppose $G'$ has an undirected Hamiltonian cycle $\Gamma'$.
- $\Gamma'$ must visit nodes in $G'$ using one of following two orders:
  - ..., B, G, R, B, G, R, B, G, R, B, ...
- Blue nodes in $\Gamma'$ make up directed Hamiltonian cycle $\Gamma$ in $G$, or reverse of one.  □
Claim. 3-SAT $\leq_P$ DIR-HAM-CYCLE.

Pf. Given an instance $\Phi$ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamiltonian cycle iff $\Phi$ is satisfiable.

Construction. First, create graph that has $2^n$ Hamiltonian cycles which correspond in a natural way to $2^n$ possible truth assignments.
3-SAT Reduces to Directed Hamiltonian Cycle

**Construction.** Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.

- Construct $G$ to have $2^n$ Hamiltonian cycles.
- Intuition: traverse path $i$ from left to right $\iff$ set variable $x_i = 1$. 

![Graph representation of the construction](image-url)
3-SAT Reduces to Directed Hamiltonian Cycle

Construction. Given 3-SAT instance \( \Phi \) with \( n \) variables \( x_i \) and \( k \) clauses.
- For each clause: add a node and 6 edges.

\[
C_1 = x_1 \lor \overline{x_2} \lor x_3 \\
C_2 = \overline{x_1} \lor x_2 \lor \overline{x_3}
\]
Polynomial-Time Reductions

INDEPENDENT SET

3-SAT reduces to INDEPENDENT SET

VERTEX COVER

SET COVER

packing and covering

DIR-HAM-CYCLE

HAM-CYCLE

sequencing

GRAPH 3-COLOR

PLANAR 3-COLOR

partitioning

SUBSET-SUM

SCHEDULING

numerical

Dick Karp (1972) 1985 Turing Award

constraint satisfaction

Dick Karp (1972) 1985 Turing Award

constraint satisfaction
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, neither</td>
<td>acggggt, 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.

$\uparrow$

$C(s, t)$ is a poly-time algorithm and $|t| \leq p(|s|)$ for some polynomial $p(\cdot)$.

Remark. NP stands for nondeterministic polynomial-time.
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| ≤ |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 × 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.** 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 $P = \text{NP}$? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
- Is the decision problem as easy as the certification problem?
- Clay $1$ million prize.

Consensus opinion on $P = \text{NP}$? Probably no.

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

If $P \neq \text{NP}$

would break RSA cryptography (and potentially collapse economy)
NP-Completeness
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. $\iff$ 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?
**Observation.** All problems below are NP-complete and polynomial reduce to one another!