Review: Polynomial-Time Reduction

- $Y \leq_P X$: Problem $Y$ is polynomial-time reducible to Problem $X$.

$\text{solveY(yInput)}$

- Construct $xInput$ // poly-time
- $\text{foo} = \text{solveX(xInput)}$ // poly # of calls
- return yes/no based on $\text{foo}$ // poly-time

- ...if any instance of Problem $Y$ can be solved using
  1. A polynomial number of standard computational steps
  2. A polynomial number of calls to a black box that solves problem $X$

- Statement about relative hardness
  1. If $Y \leq_P X$ and $X \in P$, then $Y \in P$
  2. If $Y \leq_P X$ and $Y \not\in P$ then $X \not\in P$

Reduction Strategies

- Reduction by equivalence
  (Vertex-Cover $\leq_P$ Indept-Set and vice versa)

- Reduction to a more general case
  (Vertex-Cover $\leq_P$ Set-Cover)

- Reduction by "gadgets"

Reduction by Gadgets: Satisfiability

- Can we determine if a Boolean formula has a satisfying assignment?

\[ (x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor x_2 \lor \overline{x}_3) \land (x_2 \lor \overline{x}_3) \]

- "Clause"

Terminology

<table>
<thead>
<tr>
<th>Variables</th>
<th>$x_1, \ldots, x_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>$x_i$ or $\overline{x}_i$</td>
</tr>
<tr>
<td>Clause</td>
<td>$C = \overline{x}_1 \lor x_2 \lor \overline{x}_3$</td>
</tr>
<tr>
<td>Formula</td>
<td>$C_1 \land C_2 \land \ldots \land C_k$</td>
</tr>
<tr>
<td>Assignment</td>
<td>$(x_1, x_2, x_3) = (1, 0, 1)$</td>
</tr>
<tr>
<td>Satisfying assign</td>
<td>$(x_1, x_2, x_3) = (1, 1, 0)$</td>
</tr>
</tbody>
</table>
Reduction by Gadgets: Satisfiability

SAT – Given boolean formula $C_1 \land C_2 \ldots \land C_m$ over variables $x_1, \ldots, x_n$, does there exist a satisfying assignment?

3-SAT – Same, but each $C_i$ has exactly three terms

2-SAT — each $C_i$ has exactly two terms

Clicker. What is the strongest statement below that follows easily from the definitions above?

A. $2$-SAT $\leq_P 3$-SAT $\leq_P$ SAT
B. $2$-SAT $\leq_P$ SAT and $3$-SAT $\leq_P$ SAT
C. SAT $\leq_P$ 3-SAT $\leq_P$ 2-SAT

Reduction

Idea: construct graph $G$ where independent set will select one term per clause to be true

$(\overline{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3)$

One node per term

Edges between all terms in same clause (select at most one)

Edges between a literal and all of its negations (consistent truth assignment)

Correctness

Claim: if $G$ has an independent set of size $m$, then $(C_1, \ldots, C_m)$ is satisfiable

Suppose $S$ is an independent set of size $m$

Assign variables so selected literals are true. Edges from terms to negations ensure non-conflicting assignment.

Set any remaining variables arbitrarily

At most one term per clause is selected. Since $m$ are selected, every clause is satisfied.
Correctness

\[(\overline{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x}_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_3)\]

Claim: if \((C_1, \ldots, C_m)\) is satisfiable, then \(G\) has an independent set of size \(m\)

- Consider any satisfying assignment of \((C_1, \ldots, C_m)\)
- Let \(S\) consist of one node per triangle corresponding to true literal in that clause.
  - Then \(|S| = m\).
- For \((u, v)\) within clause, at most one endpoint is selected
- For edge \((x_i, \overline{x}_i)\) between clauses, at most one endpoint is selected, because \(x_i = 1\) or \(\overline{x}_i = 1\), but not both

Reductions So Far

Partial map of problems we can use to solve others in polynomial time, through transitivity of reductions:

\[
\begin{array}{c}
\text{3-SAT} \\
\overline{\text{Indep-Set}} \\
\text{SAT} \\
\overline{\text{Vertex-Cover}} \\
\overline{\text{Set-Cover}}
\end{array}
\]

▶ means \(Y \leq_p X\).

Toward a Definition of NP

Remember our problem hierarchy:

\[
\begin{array}{c}
\text{EXP} \\
\text{NP} \\
\text{P}
\end{array}
\]

Intuition. For many “hard” decision problems, at least one thing is “easy”: if the correct answer is \textbf{Yes}, there is an easy proof

- Independent set: show an independent set of size at least \(k\)
- SAT: show a satisfying assignment

Problem classes

- \textbf{P}: Decision problems for which there is a \textit{polynomial time algorithm}.
- \textbf{NP}: Decision problems for which there is a \textit{polynomial time certifier}.
  - A solution can be “certified” in polynomial time.
  - \(NP = \) “non-deterministic polynomial time”

What is special about the mystery problems (NP)?
Solver vs. Certifier

Let $X$ be a decision problem and $s$ be problem instance (e.g., $s = (G, k)$ for Independent Set).

Poly-time solver. Algorithm $A(s)$ such that $A(s) = \text{Yes}$ iff correct answer is \text{Yes}, and running time polynomial time in $|s|$.

Poly-time certifier. Algorithm $C(s, t)$ such that for every instance $s$, there is some $t$ such that $C(s, t) = \text{Yes}$ iff correct answer is \text{Yes}, and running time is polynomial in $|s|$.

- $t$ is the “certificate” or hint; size must also be polynomial in $|s|$.

Certifier Example: Independent Set

Input $s = (G, k)$.
Problem: Does $G$ have an independent set of size at least $k$?
Idea: Certificate $t$ = an independent set of size $k$.

CertifyIS($G, k, t$)
if $|t| < k$ return \text{No}
for each edge $e = (u, v) \in E$ do
  if $u \in t$ and $v \in t$ return \text{No}
Return \text{Yes}

Polynomial time? Yes, linear in $|E|$.

Example: 3-SAT

Input: formula $\Phi$ on $n$ variables.
Problem: Is $\Phi$ satisfiable?
Idea: Certificate $t$ = the satisfying assignment.

Certify3SAT$(\Phi, t)$
\begin{itemize}
    \item Check if $t$ makes $\Phi$ true
\end{itemize}

Example: Independent Set

- Independent Set $\in \text{P}$?
  - Unknown. No known polynomial time algorithm.
- Independent Set $\in \text{NP}$?
  - Yes. Easy to certify solution in polynomial time.

Example: 3-SAT

- Formula $\Phi$ on $n$ variables.
- Problem: Is $\Phi$ satisfiable?
- Idea: Certificate $t$ = the satisfying assignment.

Certify3SAT$(\Phi, t)$
\begin{itemize}
    \item Check if $t$ makes $\Phi$ true
\end{itemize}
3SAT and Independent Set are in NP, as are many other problems that are hard to solve, but easy to certify!

- **Claim**: \( P \subseteq \text{NP} \)
- **Claim**: \( \text{NP} \subseteq \text{EXP} \)

Both straightforward to prove, but not critical right now.

**NP-complete** = a problem \( Y \in \text{NP} \) with the property that \( X \leq_p Y \) for every problem \( X \in \text{NP} \! \)

**Circuit-SAT**

**Problem**: Given a circuit built of AND, OR, and NOT gates with some hard-coded inputs, is there a way to set remaining inputs so the output is 1?

Satisfiable? Yes. Set inputs: 1, 1, 0.
Cook-Levin Theorem: Circuit-SAT is NP-Complete.

Proof Idea: encode arbitrary certifier $C(s, t)$ as a circuit

- If $X \in NP$, then $X$ has a poly-time certifier $C(s, t)$:
  - $s$ is $Yes$ instance $\iff \exists t$ such that $C(s, t)$ outputs $Yes$
  - Construct a circuit where $s$ is hard-coded, and circuit is satisfiable iff $\exists t$ that causes $C(s, t)$ to output $Yes$
  - Algorithm for Circuit-SAT implies an algorithm for $X$

A Circuit-SAT reduction

See Independent Set example in other slides

Proving New Problems NP-Complete

Fact: If $Y$ is NP-complete and $Y \leq_p X$, then $X$ is NP-complete.

Want to prove problem $X$ is NP-complete

- Check $X \in NP$.
- Choose known NP-complete problem $Y$.
- Prove $Y \leq_p X$. 
It's easy to show that $3$-SAT $\leq_P$ Circuit-SAT. What can we conclude from this?

A. $3$-SAT is NP-complete.
B. $3$-SAT is in NP.
C. If $3$-SAT is NP-complete, then Circuit-SAT is also NP-complete.