Review: Polynomial-Time Reduction

- \( Y \leq_p X \): Problem \( Y \) is polynomial-time reducible to Problem \( X \),
  \[
  \text{solveY(yInput)}\\
  \text{Construct xInput // poly-time}\\
  \text{foo = solveX(xInput) // poly # of calls}\\
  \text{return yes/no based on foo // poly-time}
  \]
- \( ...\) 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 \( Y \notin P \) then \( X \notin P \)

Reduction by Gadgets: Satisfiability

- Can we determine if a Boolean formula has a satisfying assignment?
  \[
  \left( x_1 \lor \overline{x_2} \right) \land \left( \overline{x_1} \lor x_2 \lor \overline{x_3} \right) \land \left( x_2 \lor \overline{x_3} \right)
  \]
- Terminology

<table>
<thead>
<tr>
<th>Variables</th>
<th>( x_1, \ldots, x_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term / literal</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 assignment</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

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

▶ 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 \( \langle C_1, \ldots, C_m \rangle \) 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**

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

**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
- Therefore \(S\) is an independent set

**Reductions So Far**

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

\[
\begin{align*}
3\text{-SAT} & \rightarrow \text{Indep-Set} \\
\text{Indep-Set} & \rightarrow \text{SAT} \\
\text{Vertex-Cover} & \rightarrow \text{Set-Cover}
\end{align*}
\]

➢ \(Y \rightarrow X\) means \(Y \leq_P X\).

**Toward a Definition of NP**

Remember our problem hierarchy:

\[
\begin{align*}
\text{EXP} & \rightarrow \text{NP} \\
\text{NP} & \rightarrow \text{P}
\end{align*}
\]

What is special about the mystery problems (NP)?

**P and NP**

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

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

**Problem classes**

- \(P\): Decision problems for which there is a polynomial time algorithm.
- \(NP\): Decision problems for which there is a polynomial time certifier.
  - A solution can be “certified” in polynomial time.
  - \(NP = \) “non-deterministic polynomial time”
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 $t = \text{an independent set of size } k$

```
CertifyIS(⟨G, k⟩, t)
```

- Check 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 $|E|$.

**Important:** If correct answer is $\text{YES}$, some $t$ makes $C$ output $\text{YES}$, else no way to make $C$ output $\text{YES}$. $C$ makes correct decision about $s$ if you can guess $t$.

Example: 3-SAT

**Input:** formula $\Phi$ on $n$ variables.

**Problem:** Is $\Phi$ satisfiable?

**Idea:** Certificate $t = \text{the satisfying assignment}$

```
Certify3SAT(⟨\Phi⟩, t)
```

- Check if $t$ makes $\Phi$ true

Example: Independent Set

- **Independent Set ∈ P?**
  - Unknown. No known polynomial time algorithm.

- **Independent Set ∈ NP?**
  - Yes. Easy to certify solution in polynomial time.
3SAT and Independent Set are in NP, as are many other problems that are hard to solve, but easy to certify!

- **Claim**: $P \subseteq NP$
- **Claim**: $NP \subseteq EXP$
- Both straightforward to prove, but not critical right now.

**NP-Complete**

- **Claim**: $NP$-complete = a problem $Y \in NP$ with the property that $X \leq_P Y$ for every problem $X \in NP$!

**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?

- **Circuit-SAT**: In 1971, Cook and Levin independently showed that particular problems were NP-Complete.
- We’ll look at Circuit-SAT as canonical NP-Complete problem.

Satisfiable? Yes. Set inputs: 1, 1, 0.
Circuit-SAT

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 satsifiable iff $\exists t$ that causes $C(s, t)$ to output Yes

A Circuit-SAT reduction

Vertex Cover – Does $G$ have VC of size at most $k$? (Counting gadget is an example for $v_3$, $v_4$ only)

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 $Y \leq_P X$. 

See Independent Set example in other slides
It’s easy to show that \(3\text{-SAT} \leq_p \text{Circuit-SAT}\). What can we conclude from this?

A. \(3\text{-SAT}\) is NP-complete.
B. \(3\text{-SAT}\) is in NP.
C. If \(3\text{-SAT}\) is NP-complete, then \(\text{Circuit-SAT}\) is also NP-complete.

**Theorem**: \(3\text{-SAT}\) is NP-Complete.
- In NP? Yes, check satisfying assignment in poly-time.
- Can show that \(\text{Circuit-SAT} \leq_p 3\text{-SAT}\) (next time)