Review: Polynomial-Time Reduction

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

  ```
  solveY(yInput)
  Construct xInput // poly-time
  foo = solveX(xInput) // poly # of calls
  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 $Y \not\in P$ then $X \not\in P$

Independent Set $\leq_p$ Vertex Cover

**Claim:** Independent Set $\leq_p$ Vertex Cover. **Reduction:**

- On Independent Set instance $(G,k)$
- Construct Vertex Cover instance $(G,n-k)$
- Return Yes iff solveVC($(G,n-k)$) = Yes

**Correctness** for Yes instance:

- Suppose $G$ has independent set $S$ with $\geq k$ nodes
- Then $T = V - S$ is a vertex cover with $\leq n-k$ nodes
- The algorithm correctly outputs Yes

**Correctness** for No instance:

- Suppose $G$ has no independent set $S$ with $\geq k$ nodes
- Then there is no vertex cover with $T$ with $\leq n-k$ nodes, otherwise $S = V - T$ is an independent set with $\geq k$ nodes.
- The algorithm correctly outputs No

Aside: Decision versus Optimization

- For intractability and reductions we will focus on decision problems (Yes/No answers)

- Algorithms have typically been for optimization (find biggest/smallest)

- Can reduce optimization to decision and vice versa. **Discuss.**

Vertex Cover $\leq_p$ Independent Set

**Claim:** Vertex Cover $\leq_p$ Independent Set

**Reduction:**

- On Vertex Cover input $(G,k)$
- Construct Independent Set input $(G,n-k)$
- Return Yes if solveIS($(G,n-k)$) = Yes

**Proof:** similar

Reduction Strategies

- Reduction by equivalence

- Reduction to a more general case

- Reduction by “gadgets”
**Reduction to General Case: Set Cover**

**Problem.** Given a set $U$ of $n$ elements, subsets $S_1, \ldots, S_m \subset U$, and a number $k$, does there exist a collection of at most $k$ subsets $S_i$ whose union is $U$?

- Example: $U = \{A, B, C, D, E\}$ is the set of all skills, there are five people with skill sets:
  - $S_a = \{A, C\}$
  - $S_b = \{B, E\}$
  - $S_c = \{A, C, E\}$
  - $S_d = \{D\}$
  - $S_e = \{B, C, E\}$

  Find a small team that has all skills. $S_a, S_b, S_d$

**Theorem.** $\text{VERTEXCOVER} \leq_P \text{SETCOVER}$

**Clicker Question**

Vertex Cover is a special case of Set Cover with:

- **A.** $U = V$ and $S_v =$ the two endpoints of $e$ for each $e \in E$.
- **B.** $U = E$ and $S_v =$ the set of edges incident to $v$ for each $v \in V$.
- **C.** $U = V \cup E$ and $S_v =$ the set of neighbors of $v$ together with edges incident to $v$ for each $v \in V$.

**Reduction Strategies**

- Reduction by equivalence
- Reduction to a more general case
- Reduction by "gadgets"

**Reduction by Gadgets: Satisfiability**

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

  \[ (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor \overline{x_3}) \land (x_2 \lor \overline{x_3}) \]

- **Terminology**

  | Variables | $x_1, \ldots, x_n$ | variable or its negation |
  | Term      | $x_i$ or $\overline{x_i}$ | variable or its negation |
  | Clause    | $C = x_1 \lor x_2 \lor \overline{x_3}$ | "or" of terms |
  | Formula   | $C_1 \land C_2 \land \ldots \land C_k$ | "and" of clauses |
  | Assignment| $(x_1, x_2, x_3) = (1,0,1)$ | assign 0/1 to each variable |
  | Satisfying assignment | $(x_1, x_2, x_3) = (1,1,0)$ | all clauses are "true" |

**Reduction to General Case: Set Cover**

**Intractability: quiz 4**

Given the universe $U = \{1, 2, 3, 4, 5, 6, 7\}$ and the following sets, which is the minimum size of a set cover?

- **A.** 1
  - $U = \{1,2,3,4,5,6,7\}$
  - $S_a = \{1,4,6\}$ $S_b = \{1,6,7\}$
  - $S_c = \{1,2,3,6\}$ $S_d = \{1,3,5,7\}$
  - $S_e = \{2,6,7\}$ $S_f = \{3,4,5\}$

- **B.** 2
  - $U = \{1,4,6\}$ $S_a = \{1,6,7\}$

- **C.** 3
  - $U = \{1,2,3,6\}$ $S_a = \{1,3,5,7\}$

- **D.** None of the above.

**Proof**

- Given $\text{VERTEXCOVER}$ instance $\langle G, k \rangle$
- Given $\text{SETCOVER}$ instance $\langle U, S_1, \ldots, S_m, k \rangle$ with $U = E$, and $S_v =$ the set of edges incident to $v$
- Return $\text{YES}$ if $\text{solveSC}(\langle U, S_1, \ldots, S_m, k \rangle) = \text{YES}$

- This implies the algorithm correctly outputs:
  - $\text{YES}$ if $G$ has a vertex cover of size $\leq k$ and $\text{NO}$ otherwise
  - Polynomial # of steps outside of $\text{solveSC}$
  - Only one call to $\text{solveSC}$

**Reduction of Vertex Cover to Set Cover**

- Given $\text{VERTEXCOVER}$ instance $\langle G, k \rangle$
- Construct $\text{SETCOVER}$ instance $\langle U, S_1, \ldots, S_m, k \rangle$ with $U = E$, and $S_v =$ the set of edges incident to $v$
- Return $\text{YES}$ if $\text{solveSC}(\langle U, S_1, \ldots, S_m, k \rangle) = \text{YES}$
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 from 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

Claim: 3-SAT $\leq_P$ IndependentSet.
Reduction:
▶ Given 3-SAT instance $\Phi = \langle C_1, \ldots, C_m \rangle$, we will construct an independent set instance $\langle G, m \rangle$ such that $G$ has an independent set of size $m$ iff $\Phi$ is satisfiable
▶ Return Yes if solveIS($\langle G, m \rangle$) = Yes

Correctness

Claim: if $\langle C_1, \ldots, C_m \rangle$ is satisfiable, then $G$ has an independent set of size $m$
▶ Consider any satisfying assignment of $\langle C_1, \ldots, C_m \rangle$
▶ 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 $(x_i, \bar{x}_i)$ between clauses, at most one endpoint is selected, because $x_i = 1$ or $\bar{x}_i = 1$, but not both
▶ Therefore $S$ is an independent set
Toward a Definition of NP

Remember our problem hierarchy:

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

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 \textbf{Yes}, there is an easy proof

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

**Problem classes**

- \textbf{P}: Decision problems for which there is a polynomial time algorithm.
- \textbf{NP}: Decision problems for which there is a polynomial time certifier.
  
  - A solution can be “certified” in polynomial time.
  - \textbf{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 \textsc{Independent Set})

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

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

- \(t\) is the “certificate” or hint; size must also be polynomial in \(|s|\)

Certifier Example: \textsc{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 \(k\)

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

Polynomial time? Yes, linear in \(|E|\).

Example: \textsc{Independent Set}

- \textsc{Independent Set} \textbf{\in P}? \\
  - Unknown. No known polynomial time algorithm.

- \textsc{Independent Set} \textbf{\in NP}? \\
  - Yes. Easy to certify solution in polynomial time.

Example: \textsc{3-SAT}

Input: formula \(\Phi\) on \(n\) variables.

Problem: Is \(\Phi\) satisfiable?

Idea: Certificate \(t\) = the satisfying assignment

\[
\text{Certify3SAT}(\Phi, t) \\
\text{\quad Check if } t \text{ makes } \Phi \text{ true}
\]
- 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

- 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?

Satisfiable? Yes. Set inputs: 1, 1, 0.

Circuit-SAT reduction

- Vertex Cover – Does $G$ have VC of size at most $k$?

A Circuit-SAT reduction

- Cook-Levin Theorem: In 1971, Cook and Levin independently showed that particular problems were NP-Complete.
- We’ll look at Circuit-SAT as canonical NP-Complete problem.
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 $X \in \text{NP}$.
▶ Choose known NP-complete problem $Y$.
▶ Prove $Y \leq_P X$.

Theorem: 3-SAT is NP-Complete.
▶ In NP? Yes, check satisfying assignment in poly-time.
▶ Can show that Circuit-SAT $\leq_P$ 3-SAT (next time)

It’s easy to give a reduction from 3-SAT to Circuit-SAT, i.e., 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.