Review: Polynomial-Time Reduction

- \( Y \leq_P X \):
  - Problem \( Y \) is polynomial-time reducible to Problem \( X \),
  - \( \text{solveY}(y) \)
  - Construct \( x \) // poly-time
  - \( \text{foo} = \text{solveX}(x) \) // 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 \( X \in \mathcal{P} \), then \( Y \in \mathcal{P} \)
2. If \( Y \leq_P X \) and \( Y \not\in \mathcal{P} \) then \( X \not\in \mathcal{P} \)

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:

- \( S_1 = \{A, C\} \)
- \( S_2 = \{B, E\} \)
- \( S_3 = \{A, C, E\} \)
- \( S_4 = \{D\} \)
- \( S_5 = \{B, C, E\} \)

Find a small team that has all skills. \( S_1, S_4, S_5 \)

Theorem. \( \text{VertexCover} \leq_P \text{SetCover} \)

Clicker Question

Vertex Cover is a special case of Set Cover with:

A. \( U = V \) and \( S_e = \) 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 of Vertex Cover to Set Cover

Reduction.
- Given Vertex Cover instance \( (G,k) \)
- Construct Set Cover instance \( (U,S_1,\ldots,S_m,k) \) with
  - \( U = E \), and \( S_v = \) the set of edges incident to \( v \)
- Return \text{Yes} iff \( \text{solveSC}((U,S_1,\ldots,S_m,k)) = \text{Yes} \)

Proof
- Straightforward to see that \( S_{v_1},\ldots,S_{v_\ell} \) is a set cover of size \( \ell \)
  if and only if \( v_1,\ldots,v_\ell \) is a vertex cover of size \( \ell \)
- 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 Strategies

- Reduction by equivalence (Vertex Cover and Independent Set)
- Reduction to a more general case (Vertex Cover to Set Cover)
- Reduction by "gadgets": Satisfiability
Clicker Question: Variations of SAT

SAT – Given boolean formula \( C_1 \land C_2 \land \ldots \land C_m \) 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

What is the strongest statement below that follows trivially 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 \( \leq_P \) SAT

Reduction

<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} ) variable or its negation</td>
</tr>
<tr>
<td>Clause</td>
<td>( C = \overline{x_1} \land x_2 \land \overline{x_3} ) “or” of terms</td>
</tr>
<tr>
<td>Formula</td>
<td>( C_1 \land C_2 \land \ldots \land C_m ) “and” of clauses</td>
</tr>
<tr>
<td>Assignment</td>
<td>( (x_1, x_2, x_3) = (1, 0, 1) ) variable</td>
</tr>
<tr>
<td>Satisfying assignment</td>
<td>( (x_1, x_2, x_3) = (1, 1, 0) ) all clauses are “true”</td>
</tr>
</tbody>
</table>

Solving Satisfiability

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

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

Reduction

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

\( x_1 \) is unit clause: \( x_1 \land (\ldots \land (\ldots, x_i \land \ldots)) \)
\( \overline{x_1} \) is unit clause: \( x_i \land \ldots \) must be true

Propagation: if \( x_i \) is true, all clauses with \( x_i \) are true and \( \overline{x_i} \) can be removed from all clauses

But, if no more simplifications, must still try both cases for \( x_i \)

Seth: Strong Exponential Time Hypothesis (more than \( P \neq NP \)): SAT cannot be solved in subexponential time in the worst case.

Reduction by Gadgets: Satisfiability

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

\( x_1 \) is unit clause: \( x_1 \land (\ldots \land (\ldots, x_i \land \ldots)) \)
\( \overline{x_1} \) is unit clause: \( x_i \land \ldots \) must be true

Propagation: if \( x_i \) is true, all clauses with \( x_i \) are true and \( \overline{x_i} \) can be removed from all clauses

Seth: Strong Exponential Time Hypothesis (more than \( P \neq NP \)): SAT cannot be solved in subexponential time in the worst case.
Correctness

(\bar{x}_1 \lor x_2 \lor x_3) \land (x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_2 \lor \bar{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 \((x_i, \bar{x}_j)\) between clauses, at most one endpoint is selected, because \(x_i = 1\) or \(\bar{x}_j = 1\), but not both
- Therefore S is an independent set

Reducions So Far

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

3-SAT
Indep-Set
SAT
Vertex-Cover
Set-Cover

▶ Y \rightarrow X means Y \leq_P X.

Toward a Definition of NP

Remember our problem hierarchy:

EXP
NP
P

Let’s formally define NP.

Remember: exponential time means \(O(2^n)\) for some constant \(d\).

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 \(A(s) = YES\) iff correct answer is 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 \(C(s, t) = YES\) iff correct answer is YES, and running time is polynomial in |s|

- \(t\) is the “certificate” or hint. Must also be polynomial-size 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 \(k\)

- \textbf{INDEPENDENT SET} \(\in P\)
  - Unknown. No known polynomial time algorithm.
- \textbf{INDEPENDENT SET} \(\in NP\)
  - Yes. Easy to certify solution in polynomial time.

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

$\text{Certify3SAT}(\langle \Phi \rangle, t)$

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

P, NP, EXP

$\triangleright$ 3SAT and INDEPENDENT SET are in NP, as are many other problems that are hard to solve, but easy to certify!

$\triangleright$ Claim: $P \subseteq NP$

$\triangleright$ 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.

Cook-Levin Theorem

CIRCUIT-SAT is NP-Complete.

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

$\triangleright$ If $X \in NP$, then $X$ has a poly-time certifier $C(s, t)$

$\triangleright$ Construct a circuit where $s$ is hard-coded, and circuit is satisfiable iff $\exists t$ that causes $C(s, t)$ to output YES

$\triangleright$ $s$ is Yes instance $\iff \exists t$ such that $C(s, t)$ outputs YES

$\triangleright$ $s$ is Yes instance $\iff$ circuit is satisfiable

$\triangleright$ Algorithm for CIRCUIT-SAT implies an algorithm for $X$
**A Circuit-SAT reduction**
- Vertex Cover – Does $G$ have VC of size at most $k$?

**Clicker Question**

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 Circuit-SAT is NP-complete, then $3$-SAT is also NP-complete.

**Back to 3-SAT**

**Claim:** If $Y$ is NP-complete, $X$ is in NP, and $Y \leq_p X$, then $X$ is NP-complete.

**Theorem:** 3-SAT is NP-Complete.
- In NP? Yes, check satisfying assignment in poly-time.
- Prove by reduction from Circuit-SAT.

**Example.**

**Reduction:** Circuit-Sat $\leq_p$ 3-Sat
- One variable $x_v$ per circuit node $v$ plus clauses to enforce circuit computations
- Equality = equivalence (conjunction of two implications)
- Write implication $A \Rightarrow B$ as clause $\neg A \lor B$
- Negation node: $x_v = \neg x_u$
  - $x_u \Rightarrow \neg x_v$
  - $\neg x_u \Rightarrow x_v$

- AND node: $x_v = x_u \land x_w$
  - $x_u \Rightarrow x_v$
  - $x_w \Rightarrow x_v$
  - $x_v \Rightarrow x_u \lor x_w$

- OR node: $x_v = x_u \lor x_w$
  - $x_u \Rightarrow x_v$
  - $x_w \Rightarrow x_v$
  - $x_v \Rightarrow x_u \land x_w$

**Proving New Problems NP-Complete**

**Fact:** If $Y$ is NP-complete, $X$ is in NP, 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 $Y \leq_p X$. 

**Reduction:** Circuit-Sat $\leq_p$ 3-Sat
- Clause $C = x_v$ for input bits $v$ fixed to one
- Clause $C = \neg x_v$ for input bits $v$ fixed to zero
- Clause $C = x_v$ for output bit
- This formula is satisfiable iff circuit is satisfiable.
- Deal with clauses of size 1 and 2 by introducing two new variables and clauses that force them to be equal to zero.