Review

- P – class of problems with polytime algorithm.
- NP – class of problems with polytime certifier.

Example

<table>
<thead>
<tr>
<th>Problem (X)</th>
<th>Independent-Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance (s)</td>
<td>Graph G and number k</td>
</tr>
<tr>
<td>Algorithm (A)</td>
<td>Try all subsets and check (but not poly-time)</td>
</tr>
<tr>
<td>Hint (t)</td>
<td>Which nodes are in the answer?</td>
</tr>
<tr>
<td>Certifier (C)</td>
<td>Are those nodes independent and size k?</td>
</tr>
</tbody>
</table>

NP-Complete Problems So Far

Theorem: Independent-Set, VertexCover, SetCover, SAT, 3-SAT are all NP-Complete.

Clicker Question 1

Which of the following statements is NOT true?

A: SAT ≤P 3-SAT
B: 3-SAT ≤P SAT
C: k-SAT ≤P SAT for all k ≥ 2
D: k-SAT is NP-complete for all k ≥ 2

Finding NP-Complete Problems

Want to prove problem X is NP-complete
- Check X ∈ NP.
- Choose known NP-complete problem Y.
- Prove Y ≤P X.
- Usually suffices to do single transformation s_Y → s_X s.t.
  - s_X is YES instance iff s_Y is YES instance

Traveling Salesman Problem

TSP. Given n cities and distance function d(i, j), is there a tour that visits all cities with total distance less than D?
- Tour: ordering of cities i_1, i_2, ..., i_n with i_1 = 1
- Distance is \( \sum_{j=1}^{n-1} d(i_j, i_{j+1}) + d(i_n, 1) \)
- Applications: traveling salesperson, moving robotic arms
- Let’s prove a simpler problem is NP-complete, and then use it to show TSP is NP-complete.
Hamiltonian Cycle Problem

- **HamCycle** – Hamiltonian Cycle. Given directed graph \( G = (V, E) \), is there a cycle that visits each vertex exactly once?

- \( v_1, v_3, v_2, v_5, v_6 \) is a Hamiltonian Cycle

Hamiltonian Cycle

- **Theorem.** HamCycle is NP-Complete.
  - It is in NP.
  - Need to reduce from some NP-Complete problem. Which one?

- **Claim.** 3-SAT \( \leq_p \) HamCycle.

  Reduction has two main parts.
  - Make a graph with 2\(^n\) Hamiltonian cycles, one per assignment.
  - Augment graph with clauses to invalidate assignments.

Reduction: Graph skeleton

- \( n \) rows (bidirected paths) \( P_1, \ldots, P_n \) (one per variable)
- Row has \( 3m + 3 \) vertices, connected to neighbors in forward/backward direction
- First and last vertex of row \( i \) connected to first and last of \( i + 1 \).
- Source \( s \) connected to first and last of row 1.
- First and last of row \( n \) connected to \( t \).
- Edge \((i, s)\)
- Skeleton has 2\(^n\) possible Hamiltonian Cycles, corresponding to truth assignments to \( x_1, \ldots, x_n \)
  - Traverse \( P_i \) L to R \( \iff x_i = 1 \)
  - Traverse \( P_i \) R to L \( \iff x_i = 0 \)

Reduction: Skeleton Construction

- For each clause \( C_i \) construct gadget to restrict possible truth assignments
  - New node \( e_i \)
  - If \( x_i \in C_i \)
    - Add edges \( (v_{2i}, e_i) \) and \( (e_i, v_{2i+1}) \)
    - \( e_i \) can be visited during L to R traversal of \( P_i \)
  - If \( \neg x_i \in C_i \)
    - Add edges \( (v_{2i+1}, e_i) \) and \( (e_i, v_{2i+3}) \)
    - \( e_i \) can be visited during R to L traversal of \( P_i \)

Proof of Correctness

Given a satisfying assignment, construct Hamiltonian Cycle

- If \( x_i = 1 \) traverse \( P_i \) from \( L \to R \), else \( R \to L \).
- Each \( C_i \) is satisfied, so one path \( P_i \) is traversed in the correct direction to “splice” \( e_i \) into our cycle
- The result is a Hamiltonian Cycle

Given Hamiltonian cycle, construct satisfying assignment:

- If cycle visits \( e_i \) from row \( i \), it will also leave to row \( i \) because of “buffer” nodes
- Therefore, ignoring clause nodes, cycle traverses each row completely from \( L \to R \) or \( R \to L \)
- Set \( x_i = 1 \) if \( P_i \) traversed \( L \to R \), else \( x_i = 0 \)
- Every node \( c_j \) visited \( \Rightarrow \) every clause \( C_j \) is satisfied
Traveling Salesman

TSP. Given \( n \) cities and distance function \( d(i,j) \), is there a tour that visits all cities with total distance less than \( D \)?

**Theorem.** TSP is NP-Complete

- Clearly in NP.
- Reduction? From Ham-Cycle

Reduction from Ham-Cycle to TSP

Given HamCycle instance \( G = (V,E) \) make TSP instance

- One city per vertex
- \( d(v_i,v_j) = 1 \) if \( (v_i,v_j) \in E \), else 2

**Claim:** there is a tour of distance \( \leq n \) if and only if \( G \) has a Hamiltonian cycle

- A Hamiltonian cycle clearly gives a tour of length \( n \)
- A tour of length \( n \) must travel \( n \) hops of length 1, which corresponds to a Hamiltonian cycle

HamPath

Similar to Hamiltonian Cycle, visit every vertex exactly once.

**Theorem.** HamPath is NP-Complete.

Two proofs.

- Modify 3-SAT to HamCycle reduction.
- Reduce from HamCycle directly.

Clicker Question 2

Suppose now I want to reduce HamPath to HamCycle. Which of the following statements is true?

A: Trivial: any cycle with all nodes is also a path with all nodes

B: HamPath \( \not\leq_P \) HamCycle since not all paths are cycles

C: HamPath \( \not\leq_P \) HamCycle since there are some graphs that have a Hamiltonian path but no Hamiltonian cycle

D: None of the above

NP-Complete Problems

3-SAT

Indep-Set

Vertex-Cover

Set-Cover

Circuit-SAT

Ham-Cycle

Traveling-Salesman

Numerical problems

**Subset Sum** decision problem: given \( n \) items with weights \( w_1,\ldots,w_n \), is there a subset of items whose weight is exactly \( W \)?

Dynamic programming: \( O(nW) \) pseudo-polynomial time algorithm (not polynomial in input length \( n \log W \))
**Subset Sum**

**Theorem.** Subset sum is NP-complete.

Reduction from 3-SAT. (n variables, m clauses, base 10).
- All weights have n + m digits
- Digits 1 to n: For variable $x_i$, create two items $t_i, f_i$
  - Both have $i$th digit equal to 1
  - All other items have zero in this digit
  - $i$th digit of $W = 1$ ⇒ select exactly one of $t_i, f_i$
- The $n + j$th digit corresponds to clause $C_j$
  - If $x_i \in C_j$, set $n + j$th digit of $t_i = 1$
  - If $\neg x_i \in C_j$, set $n + j$th digit of $f_i = 1$
  - Everything else 0.

**Subset Sum Proof**

- Set $n + j$th digit of $W = 3$
- Consider a subset of items corresponding to a truth assignment (exactly one of $t_i, f_i$)
- If $C_j$ is not satisfied, then total in position $n + j$ is 0, otherwise it is 1, 2, or 3
- Create two “dummy” items $y_j, z_j$ with 1 in position $n + j$
- Can select dummies to yield total of 3 in position $n + j$ iff $C_j$ is satisfied

**Subset Sum Example**

**Example.**

$$(x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3)$$

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$f_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$f_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$W$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$z_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$y_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$z_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$y_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$z_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Warning**

**Theorem.** $\text{SubsetSum}$ is NP-Complete.

But Subset Sum can be tricky!
- If reducing $\text{SubsetSum} \leq_P X$, reduction needs to be polynomial in $\log(W)$ (number of digits).

**Graph Coloring**

**Def.** A $k$-coloring of a graph $G = (V, E)$ is a function $f : V \rightarrow \{1, \ldots, k\}$ such that for all $(u, v) \in E$, $f(u) \neq f(v)$.

**Problem.** Given $G = (V, E)$ and number $k$, does $G$ have a $k$-coloring?

**Many applications**
- Actually coloring maps!
- Scheduling jobs on machine with competing resources.
- Allocating variables to registers in a compiler.
Graph Coloring

**Claim.** 2-COLORING ∈ P.

**Proof.** 2-coloring equivalent to bipartite testing.

**Theorem.** 3-COLORING is NP-Complete.

Reduction

▶ Reduce from 3-SAT.

Skeleton: 1 color for True, 1 for False
3 extra nodes in a clique T, F, B.
For each variable $x_i$, two nodes $v_{i0}, v_{i1}$.
Edges $(v_{i0}, B), (v_{i1}, B), (v_{i0}, v_{i1})$.
Either $v_{i0}$ or $v_{i1}$ gets the $T$ color.

Proof

▶ Graph is polynomial in $n + m$.
▶ If satisfying assignment
   ▶ Color B, T, F then $vi1$ as $T$ if $ϕ(x_i) = 1$.
   ▶ Since clauses satisfied, can color each gadget.
▶ If graph 3-colorable
   ▶ One of $v_{i0}, v_{i1}$ must get $T$ color.
   ▶ Clause gadget colorable iff clause satisfied.

**Question.** What about $k$-coloring? What is 2-coloring?

NP-Completeness Recap

Types of hard problems:

- Circuit-SAT
- 3-SAT
- Constraint satisfaction
- Graph-Coloring
- Ham-Cycle
- Hamiltonian Path
- Independent Set
- Induced Hamiltonian Cycle
- Knapsack
- Traveling Salesman
- TSP
- Numerical Sequencing
- Partitioning
- Packing
- Covering

... any many others. See book or other sources for more examples.
You can use any known NP-complete problem to prove a new problem is NP-complete.