**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>No poly-time algorithm known</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**

- NP-complete = a problem $Y \in \text{NP}$ with the property that $X \leq_P Y$ for every problem $X \in \text{NP}$!

**NP-Complete**

- **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.
**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 Diagram](image)

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 $\Leftrightarrow \exists t$ 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
  - $s$ is YES instance $\Leftrightarrow$ circuit is satisfiable
  - Algorithm for Circuit-SAT implies an algorithm for $X$

---

**A Circuit-SAT reduction**

See Independent Set example in other slides

---

**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)

![Vertex Cover Diagram](image)
Suppose \( X \) is in NP.

**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

---

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 there is no polynomial time algorithm for 3-SAT, then there is no polynomial time algorithm for Circuit-SAT.

---

To show that Circuit-SAT \( \leq_P \) 3-SAT, we’ll show how to construct a 3-SAT formula to model an arbitrary Circuit-SAT instance.

**Example.**
Reduction: Circuit-Sat \( \leq_p \) 3-Sat

- One variable \( x_v \), per circuit node \( v \) plus clauses to enforce circuit computations
- Express Negation, OR, and AND nodes using several implications of the form \( A \Rightarrow B \) (which is equivalent to the clause \( \neg A \lor B \))

- Negation node: \( x_v = \neg x_u \)
  - \( x_u \Rightarrow \neg x_v \)
  - \( \neg x_u \Rightarrow x_v \)

- OR node: \( x_v = x_u \lor x_w \)
  - \( x_u \Rightarrow x_v \)
  - \( x_w \Rightarrow x_v \)
  - \( x_v \Rightarrow x_u \lor x_w \)

- AND node: \( x_v = x_u \land x_w \)
  - \( x_v \Rightarrow x_u \)
  - \( x_v \Rightarrow x_w \)
  - \( \neg x_v \Rightarrow \neg x_u \lor \neg x_w \)

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 \( C = x_o \) 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.

Clicker

Which of the following statements is NOT true?

A. SAT \( \leq_p \) 3-SAT
B. 3-SAT \( \leq_p \) SAT
C. k-SAT \( \leq_p \) SAT for all \( k \geq 2 \)
D. k-SAT is NP-complete for all \( k \geq 2 \)

NP-Complete Problems So Far

Theorem: IndependentSet, VertexCover, SetCover, SAT, 3-SAT are all NP-Complete.
NP-Complete Problems: Preview

- 3-SAT
- Indep-Set
- Vertex-Cover
- Set-Cover
- Circuit-SAT
- Ham-Cycle
- Ham-Path
- Traveling-Salesman
- Subset-Sum
- 0-1 Knapsack
- Graph-Coloring
- Constraint satisfaction
- Partitioning
- Numerical
- Sequencing
- Packing
- Covering

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 salesman, 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?
- \( v_1, v_3, v_2, v_5, v_4, v_6 \) is a Hamiltonian Cycle

HAM-CYCLE

Theorem. HAM-CYCLE is NP-Complete.

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

Claim. 3-SAT \( \leq_p \) HAM-CYCLE.

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

- Correspondence between Hamiltonian cycles and truth assignments
  - $x_i = 1$: traverse $P_i$ from $L \to R$
  - $x_i = 0$: traverse $P_i$ from $R \to L$

- Node $c_j$ for clause $C_j$ must be visited in middle of some $P_i$
  - $x_i \in C_j \Rightarrow$ can visit $c_j$ during $L \to R$ traversal of $P_i$, $x_i = 1$ satisfies $C_j$
  - $\bar{x}_i \in C_j \Rightarrow$ can visit $c_j$ during $R \to L$ traversal of $P_i$, $x_i = 0$ satisfies $C_j$

- There is a Hamiltonian cycle
  $\iff$ can visit all clause nodes
  $\iff$ there is a truth assignment that satisfies all clauses

Reduction: Clause Gadgets

- $C_1 = x_1 \lor \bar{x}_2 \lor x_3$

Reduction: High-Level

- $n$ 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 $(t, s)$
- 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: Details

- Hamiltonian cycle corresponds to the $2^n$ possible truth assignments
- Correspondence between Hamiltonian cycles and truth assignments
- $x_i = 1$: traverse path $P_i$ from $L \to R$
- $x_i = 0$: traverse path $P_i$ from $R \to L$
- Node $c_j$ for clause $C_j$ must be visited in middle of some $P_i$
  - $x_i \in C_j \Rightarrow$ can visit $c_j$ during $L \to R$ traversal of $P_i$, $x_i = 1$ satisfies $C_j$
  - $\bar{x}_i \in C_j \Rightarrow$ can visit $c_j$ during $R \to L$ traversal of $P_i$, $x_i = 0$ satisfies $C_j$
- There is a Hamiltonian cycle
  $\iff$ can visit all clause nodes
  $\iff$ there is a truth assignment that satisfies all clauses
Reduction: Clause Gadgets

For each clause $C_\ell$ construct gadget to restrict possible truth assignments

- New node $c_\ell$
- If $x_i \in C_\ell$
  - Add edges $(v_i, 3\ell, c_\ell)$ and $(c_\ell, v_i, 3\ell + 1)$
  - $c_\ell$ can be visited during L to R traversal of $P_i$
- If $\neg x_i \in C_\ell$
  - Add edges $(v_i, 3\ell + 1, c_\ell)$ and $(c_\ell, v_i, 3\ell)$
  - $c_\ell$ 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 $C_\ell$ is satisfied, so one path $P_i$ is traversed in the correct direction to “splice” $c_\ell$ into our cycle
- The result is a Hamiltonian Cycle

Given Hamiltonian cycle, construct satisfying assignment:

- If cycle visits $c_\ell$ 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

Clicker

We want to show that $\text{HAM-CYCLE} \leq_P \text{TSP}$. How can we do so?

Given a HAM-CYCLE instance $G = (V, E)$ make TSP instance with one city per vertex and...

A. $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else 2. Tour distance: $\leq n$?
B. $d(v_i, v_j) = 2$ if $(v_i, v_j) \in E$, else 1. Tour distance: $\leq n$?
C. $d(v_i, v_j) = 1$ if $(v_i, v_j) \in E$, else 2. Tour distance: $\leq m$?
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

Ham-Path

Similar to Hamiltonian Cycle: is there a path that visits every vertex exactly once?

**Theorem.** Ham-Path is NP-Complete.

Two proofs:
- Modify 3-SAT to Ham-Cycle reduction.
- Show that Ham-Cycle reduces to Ham-Path

NP-Complete Problems

```
  Circuit-SAT
    /\         /
  3-SAT      Indep-Set
           |  /
         |  Indep-Set
         | /       
         V         
  Ham-Cycle   Ham-Cycle
           |  /
         |  Vertex-Cover
         | /       
         V         
  Traveling-Salesman
           |
         |       
         V       
  Set-Cover
```