4.1 Quick & total recall

4.1.1 Recall...

4.1.1.1 NP Problems

Definition 4.1.1. Nondeterministic Polynomial Time (denoted by $\text{NP}$) is the class of all problems that have efficient certifiers (for YES instances).

4.1.2 Recall...

4.1.2.1 NP-Complete Problems

Definition 4.1.2. A problem $X$ is said to be $\text{NP-Complete}$ if

(A) $X \in \text{NP}$, and
(B) (Hardness) For any $Y \in \text{NP}$, $Y \leq_P X$. 
4.1.2.2 Recall...

NP Decision problems with a polynomial certifier. Examples: SAT, Hamiltonian Cycle, 3-Colorability.

Definition 4.1.3. co-NP: class of all decision problems $X$ s.t. $\overline{X} \in NP$.
Examples: UnSAT, No-Hamiltonian-Cycle, No-3-Colorable.

4.1.2.3 Recall...

(A) NP: languages that have polynomial time certifiers/verifiers. 
(B) A language $L$ is NP-Complete $\iff$
- $L$ is in NP
- for every $L'$ in NP, $L' \leq_P L$
(C) $L$ is NP-Hard if for every $L'$ in NP, $L' \leq_P L$.
(D) Cook-Level theorem...

Theorem 4.1.4 (Cook-Levin). Circuit-SAT is NP-Complete.

4.2 NP Completeness continued

4.2.1 Preliminaries

4.2.1.1 Circuits

Definition 4.2.1. A circuit is a directed acyclic graph with...
4.2.2 Cook-Levin Theorem

4.2.2.1 Cook-Levin Theorem

Definition 4.2.2 (Circuit Satisfaction (CSAT)). Given a circuit as input, is there an assignment to the input variables that causes the output to get value 1?

Theorem 4.2.3 (Cook-Levin). CSAT is NP-Complete.

Need to show
(A) CSAT is in NP.
(B) every NP problem X reduces to CSAT.

4.2.2.2 CSAT: Circuit Satisfaction

Claim 4.2.4. CSAT is in NP.

(A) Certificate: Assignment to input variables.
(B) Certifier: Evaluate the value of each gate in a topological sort of DAG and check the output gate value.

4.2.2.3 CSAT is NP-hard: Idea

(A) Need to show that every NP problem X reduces to CSAT.
(B) What does it mean that X ∈ NP?
(C) X ∈ NP implies that there are polynomials p() and q() and certifier/verifier program C such that for every string s the following is true:
   (A) If s is a YES instance (s ∈ X) then there is a proof t of length p(|s|) such that C(s, t) says YES.
   (B) If s is a NO instance (s ∉ X) then for every string t of length at p(|s|), C(s, t) says NO.
   (C) C(s, t) runs in time q(|s| + |t|) time (hence polynomial time).

4.2.2.4 Reducing X to CSAT

(A) X is in NP means we have access to p(), q(), C(·, ·).
(B) What is C(·, ·)? It is a program or equivalently a Turing Machine!
(C) How are p() and q() given?
   As numbers.
(D) Example: if 3 is given then p(n) = n³.
(E) NP problem ≡ ⟨p, q, C⟩. where C is a program or a TM.

4.2.2.5 Reducing X to CSAT

(A) NP problem: a three tuple ⟨p, q, C⟩.
   C: program or TM, p(·), q(·): polynomials.
(B) Problem X: Given string s, is s ∈ X?
(C) Equivalent:
   ∃ proof t of length p(|s|) & C(s, t) returns YES.
   ...C(s, t) runs in q(|s|) time.
(D) Reduce from X to CSAT...
   Need an algorithm alg that
(A) takes \( s \) (and \( \langle p, q, C \rangle \)).

Creates circuit \( G \) in poly time in \( |s| \).

\( \langle p, q, C \rangle \) is fixed so \( |\langle p, q, C \rangle| = O(1) \).

(B) \( G \) is satisfiable

\[ \iff \exists \text{ proof } t \text{ s.t. } C(s, t) \text{ returns YES.} \]

4.2.2.6 Reducing \( X \) to CSAT

(A) \textbf{Q:} How do we reduce \( X \) to \textbf{CSAT}?

(B) Need algorithm \( \text{alg} \) that:

(A) Input: \( s \) (and \( \langle p, q, C \rangle \)).

(B) creates circuit \( G \) in poly-time in \( |s| \) (\( \langle p, q, C \rangle \) fixed).

(C) \( G \) satisfiable \( \iff \exists \text{ proof } t : C(s, t) \text{ returns YES.} \)

(C) \textbf{Simple but Big Idea:} Programs are the same as Circuits!

(A) Convert \( C(s, t) \) into a circuit \( G \) with \( t \) as unknown inputs (rest is known including \( s \))

(B) Known: \( |t| \leq p(|s|) \) so express boolean string \( t \) as \( p(|s|) \) variables \( t_1, t_2, \ldots, t_k \) where \( k = p(|s|) \).

(C) Asking if there is a proof \( t \) that makes \( C(s, t) \) say YES is same as whether there is an assignment of values to "unknown" variables \( t_1, t_2, \ldots, t_k \) that will make \( G \) evaluate to true/YES.

4.2.2.7 Example: \textbf{Independent Set}

(A) Formal definition:

\textbf{Independent Set}

\begin{center}
\begin{tabular}{l}
\textit{Instance:} \( G = (V, E) \), \( k \) \\
\textit{Question:} Does \( G = (V, E) \) have an \textbf{Independent Set} of size \( \geq k \)
\end{tabular}
\end{center}

(B) \textbf{Certificate:} Set \( S \subseteq V \).

(C) \textbf{Certifier:} Check \( |S| \geq k \) and no pair of vertices in \( S \) is connected by an edge.

(D) \textbf{Q:} Formally, why is \textbf{Independent Set} in \textbf{NP}?

4.2.3 Example: \textbf{Independent Set}

4.2.3.1 Formally why is \textbf{Independent Set} in \textbf{NP}?

(A) Input is a “binary” vector:

\[ \langle n, y_{1,1}, y_{1,2}, \ldots, y_{1,n}, y_{2,1}, \ldots, y_{2,n}, \ldots, y_{n,1}, \ldots, y_{n,n}, k \rangle \]

encodes \( \langle G, k \rangle \).

(A) \( n \) is number of vertices in \( G \)

(B) \( y_{i,j} \) is a bit which is 1 if edge \( (i, j) \) is in \( G \) and 0 otherwise (adjacency matrix representation)

(C) \( k \): size of independent set.

(B) \textbf{Certificate:} \( t = t_1 t_2 \cdots t_n \).

Interpretation: \( t_i = 1 \) if vertex \( i \) is in independent set.

\( \ldots 0 \) otherwise.
4.2.3.2 Certifier for Independent Set

Certifier $C(s, t)$ for Independent Set:

\[
\text{if } (t_1 + t_2 + \ldots + t_n < k) \text{ then return NO else for each } (i, j) \text{ do if } (t_i \land t_j \land y_{ij}) \text{ then return NO return YES}
\]

4.2.4 Example: Independent Set

4.2.4.1 Certifier circuit for Independent Set of size at least 2 for graph with 3 vertices

\[
\begin{array}{c}
\text{Both ends of an edge?} \\
\text{At least two vertices?}
\end{array}
\]

Figure 4.1: Graph $G$ with $k = 2$

4.2.4.2 Programs, Turing Machines and Circuits

(A) $\text{alg:}$ “program” that takes $f(|s|)$ steps on input string $s$.

(B) Questions: What computer is used? What does step mean?

(C) “Real” computers difficult to reason with mathematically:

(A) instruction set is too rich

(B) pointers and control flow jumps in one step

(C) assumption that pointer to code fits in one word

(D) Turing Machines:

(A) simpler model of computation to reason with

(B) can simulate real computers with polynomial slow down

(C) all moves are local (head moves only one cell)

4.2.4.3 Certifiers that at TMs

(A) Assume $C(\cdot, \cdot)$ is a (deterministic) Turing Machine $M$

(B) Problem: Given $M$, input $s$, $p$, $q$ decide if:

(A) $\exists$ proof $t$ of length $\leq p(|s|)$
(B) $M$ executed on the input $s, t$ halts in $q(|s|)$ time and returns YES.

(C) $\text{ConvCSAT}$ reduces above problem to $\text{CSAT}$:

1. computes $p(|s|)$ and $q(|s|)$.
2. As such, $M$:
   (A) Uses at most $q(|s|)$ memory/tape cells.
   (B) $M$ can run for at most $q(|s|)$ time.

4.2.4.4 Simulation of Computation via Circuit

(A) $M$ state at time $\ell$: A string $x^\ell = x_1x_2 \ldots x_k$ where each $x_i \in \{0, 1, B\} \times Q \cup \{q_{-1}\}$.
(B) Time 0: State of $M$ = input string $s$, a guess $t$ of $p(|s|)$ “unknowns”, and rest $q(|s|)$ blank symbols.
(C) Time $q(|s|)$? Does $M$ stops in $q_{\text{accept}}$ with blank tape.
(D) Build circuit $C_\ell$: Evaluates to YES
   $\iff$ transition of $M$ from time $\ell$ to time $\ell + 1$ valid.
   (Circuit of size $O(q(|s|))$).
(E) $C$: $C_0 \land C_1 \land \cdots \land C_{q(|s|)}$.
   Polynomial size!
(F) Output of $C$ true $\iff$ sequence of states of $M$ is legal and leads to an accept state.

4.2.4.5 NP-Hardness of Circuit Satisfaction

Key Ideas in reduction:
(A) Use TMs as the code for certifier for simplicity
(B) Since $p()$ and $q()$ are known to $A$, it can set up all required memory and time steps in advance
(C) Simulate computation of the TM from one time to the next as a circuit that only looks at three adjacent cells at a time

Note: Above reduction can be done to $\text{SAT}$ as well. Reduction to $\text{SAT}$ was the original proof of Steve Cook.

4.3 Showing that $\text{SAT}$ is NP-Complete

4.3.1 Other NP Complete Problems

4.3.1.1 $\text{SAT}$ is NP-Complete

(A) We have seen that $\text{SAT} \in \text{NP}$
(B) To show $\text{NP-Hardness}$, we will reduce Circuit Satisfiability ($\text{CSAT}$) to $\text{SAT}$

Instance of $\text{CSAT}$ (we label each node):

![Circuit Diagram]

Output: $\neg k$

Input: $\neg d$

$\neg j$

$\neg i$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$

$\neg d$

$\neg j$

$\neg h$

$\neg g$

$\neg f$

$\neg e$
4.3.2 Converting a circuit into a CNF formula

4.3.2.1 Label the nodes

(A) Input circuit

(B) Label the nodes.

4.3.3 Converting a circuit into a CNF formula

4.3.3.1 Introduce a variable for each node

(B) Label the nodes.  

(C) Introduce var for each node.

4.3.4 Converting a circuit into a CNF formula

4.3.4.1 Write a sub-formula for each variable that is true if the var is computed correctly.

\[
\begin{align*}
x_k & \quad \text{(Demand a sat' assignment!)} \\
x_k &= x_i \land x_k \\
x_j &= x_g \land x_h \\
x_i &= \neg x_f \\
x_h &= x_d \lor x_e \\
x_g &= x_b \lor x_c \\
x_f &= x_a \land x_b \\
x_d &= 0 \\
x_a &= 1
\end{align*}
\]

(C) Introduce var for each node.  

(D) Write a sub-formula for each variable that is true if the var is computed correctly.
4.3.5 Converting a circuit into a CNF formula

4.3.5.1 Convert each sub-formula to an equivalent CNF formula

<table>
<thead>
<tr>
<th>$x_k$</th>
<th>$x_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k = x_i \land x_j$</td>
<td>$\neg x_k \land (\neg x_k \lor x_i) \land (\neg x_k \lor x_j)$</td>
</tr>
<tr>
<td>$x_j = x_g \land x_h$</td>
<td>$\neg x_j \land (\neg x_j \lor x_g) \land (\neg x_j \lor x_h)$</td>
</tr>
<tr>
<td>$x_i = \neg f$</td>
<td>$(x_i \lor x_f) \land (\neg x_i \lor \neg x_f)$</td>
</tr>
<tr>
<td>$x_h = x_d \lor x_e$</td>
<td>$(\neg x_d \lor \neg x_e) \land (\neg x_h \lor x_d \lor x_e)$</td>
</tr>
<tr>
<td>$x_g = x_b \lor x_c$</td>
<td>$(\neg x_b \lor \neg x_c) \land (\neg x_g \lor x_b \lor x_c)$</td>
</tr>
<tr>
<td>$x_f = x_a \land x_b$</td>
<td>$(\neg x_f \lor x_a) \land (\neg x_f \lor x_b) \land (x_f \lor \neg x_a \land \neg x_b)$</td>
</tr>
<tr>
<td>$x_d = 0$</td>
<td>$\neg x_d$</td>
</tr>
<tr>
<td>$x_a = 1$</td>
<td>$x_a$</td>
</tr>
</tbody>
</table>

4.3.6 Converting a circuit into a CNF formula

4.3.6.1 Take the conjunction of all the CNF sub-formulas

$\forall x_k (\neg x_k \land (\neg x_k \land (\neg x_k \land \cdots))$ both true.

4.3.6.2 Reduction: $\text{CSAT} \leq_P \text{SAT}$

(A) For each gate (vertex) $v$ in the circuit, create a variable $x_v$

(B) Case $\neg v$ is labeled $\neg$ and has one incoming edge from $u$ (so $x_v = \neg x_u$). In SAT formula generate, add clauses $(x_u \lor x_v)$, $(\neg x_u \lor \neg x_v)$. Observe that

$$x_v = \neg x_u \text{ is true} \iff (x_u \lor x_v) \land (\neg x_u \lor \neg x_v) \text{ both true.}$$

4.3.7 Reduction: $\text{CSAT} \leq_P \text{SAT}$

4.3.7.1 Continued...

(A) Case $\lor$: So $x_v = x_u \lor x_w$. In SAT formula generated, add clauses $(x_v \lor \neg x_u)$, $(x_v \lor \neg x_w)$, and $(\neg x_v \lor x_u \lor x_w)$. Again, observe that

$$(x_v = x_u \lor x_w) \text{ is true} \iff (x_v \lor \neg x_u), (x_v \lor \neg x_w), \neg x_v \lor x_u \lor x_w \text{ all true.}$$
4.3.8 Reduction: CSAT \( \leq_P \) SAT

4.3.8.1 Continued...

(A) Case \( \land \): So \( x_v = x_u \land x_w \). In SAT formula generated, add clauses \((\neg x_v \lor x_u), (\neg x_v \lor x_w), (x_v \lor \neg x_u \lor \neg x_w)\). Again observe that 

\[
x_v = x_u \land x_w \text{ is true } \iff \begin{cases} \neg x_v \lor x_u, \\ \neg x_v \lor x_w, \\ x_v \lor \neg x_u \lor \neg x_w \end{cases} \text{ all true.}
\]

4.3.9 Reduction: CSAT \( \leq_P \) SAT

4.3.9.1 Continued...

(A) If \( v \) is an input gate with a fixed value then we do the following. If \( x_v = 1 \) add clause \( x_v \). If \( x_v = 0 \) add clause \( \neg x_v \).

(B) Add the clause \( x_v \) where \( v \) is the variable for the output gate

4.3.9.2 Correctness of Reduction

Need to show circuit \( C \) is satisfiable iff \( \varphi_C \) is satisfiable

\( \Rightarrow \) Consider a satisfying assignment \( a \) for \( C \)

(A) Find values of all gates in \( C \) under \( a \)

(B) Give value of gate \( v \) to variable \( x_v \); call this assignment \( a' \)

(C) \( a' \) satisfies \( \varphi_C \) (exercise)

\( \Leftarrow \) Consider a satisfying assignment \( a \) for \( \varphi_C \)

(A) Let \( a' \) be the restriction of \( a \) to only the input variables

(B) Value of gate \( v \) under \( a' \) is the same as value of \( x_v \) in \( a \)

(C) Thus, \( a' \) satisfies \( C \)

Theorem 4.3.1. SAT is NP-Complete.

4.3.9.3 Proving that a problem \( X \) is NP-Complete

(A) To prove \( X \) is NP-Complete, show

(A) Show \( X \) is in NP.

(A) certificate-proof of polynomial size in input

(B) polynomial time certifier \( C(s, t) \)

(B) Reduction from a known NP-Complete problem such as CSAT or SAT to \( X \)

(B) SAT \( \leq_P X \) implies that every NP problem \( Y \leq_P X \). Why?

Transitivity of reductions:

(C) \( Y \leq_P SAT \) and SAT \( \leq_P X \) and hence \( Y \leq_P X \).

4.3.9.4 NP-Completeness via Reductions

(A) What we currently know:

(A) CSAT is NP-Complete.

(B) CSAT \( \leq_P \) SAT and SAT is in NP and hence SAT is NP-Complete.

(C) SAT \( \leq_P 3SAT \) and hence 3SAT is NP-Complete.

(D) 3SAT \( \leq_P \) Independent Set (which is in NP) and hence Independent Set is NP-Complete.
(E) **Vertex Cover** is NP-Complete.
(F) **Clique** is NP-Complete.
(B) Hundreds and thousands of different problems from many areas of science and engineering have been shown to be NP-Complete.
(C) A surprisingly frequent phenomenon!

4.4 More reductions...

4.4.0.1 Next...

Prove

- **Hamiltonian Cycle** Problem is NP-Complete.
- 3-Coloring is NP-Complete.
- **Subset Sum**.

4.5 NP-Completeness of Hamiltonian Cycle

4.6 Reduction from 3SAT to Hamiltonian Cycle

4.6.0.1 Directed Hamiltonian Cycle

**Input** Given a directed graph $G = (V, E)$ with $n$ vertices

**Goal** Does $G$ have a Hamiltonian cycle?

- A Hamiltonian cycle is a cycle in the graph that visits every vertex in $G$ exactly once

4.6.0.2 Directed Hamiltonian Cycle is NP-Complete

- Directed Hamiltonian Cycle is in $NP$
  - **Certificate**: Sequence of vertices
  - **Certifier**: Check if every vertex (except the first) appears exactly once, and that consecutive vertices are connected by a directed edge

- **Hardness**: Will prove...

  $3SAT \leq_P \text{Directed Hamiltonian Cycle.}$
4.6.0.3 Reduction

(A) 3SAT formula $\varphi$ create a graph $G_\varphi$ such that

- $G_\varphi$ has a Hamiltonian cycle $\iff$ $\varphi$ is satisfiable
- $G_\varphi$ should be constructible from $\varphi$ by a polynomial time algorithm $A$

(B) Notation: $\varphi$ has $n$ variables $x_1, x_2, \ldots, x_n$ and $m$ clauses $C_1, C_2, \ldots, C_m$.

4.6.0.4 Reduction: First Ideas

- Viewing SAT: Assign values to $n$ variables, and each clauses has 3 ways in which it can be satisfied.
- Construct graph with $2^n$ Hamiltonian cycles, where each cycle corresponds to some boolean assignment.
- Then add more graph structure to encode constraints on assignments imposed by the clauses.

4.6.0.5 The Reduction: Phase I

- Traverse path $i$ from left to right $\iff$ $x_i$ is set to true.
- Each path has $3(m+1)$ nodes where $m$ is number of clauses in $\varphi$; nodes numbered from left to right (1 to $3m+3$)

4.6.0.6 The Reduction: Phase II

- Add vertex $c_j$ for clause $C_j$. $c_j$ has edge from vertex $3j$ and to vertex $3j+1$ on path $i$ if $x_i$ appears in clause $C_j$, and has edge from vertex $3j+1$ and to vertex $3j$ if $\neg x_i$ appears in $C_j$. 

4.6.0.7 In the next lecture...

Correctness proof of the above reduction, and more NPC problems.