Definition: A language $B$ is NP-complete if:

1. $B \in \text{NP}$

2. Every $A$ in NP is poly-time reducible to $B$
   That is, $A \leq_p B$

When this is true, we say “$B$ is NP-hard”

On homework, you showed
A language $L$ is recognizable iff $L \leq_m A_{\text{TM}}$

$A_{\text{TM}}$ is “complete for recognizable languages”:
$A_{\text{TM}}$ is recognizable, and for all recognizable $L$, $L \leq_m A_{\text{TM}}$
Suppose $L$ is NP-Complete...

If $L \in P$, then $P = NP$

If $L \notin P$, then $P \neq NP$
Suppose $L$ is NP-Complete...

Then assuming the conjecture $P \neq NP$,

$L$ is not decidable in $n^k$ time, for every $k$
The Cook-Levin Theorem: SAT and 3SAT are NP-complete

1. **3SAT \(\in\) NP
   
   A satisfying assignment is a “proof” that a 3cnf formula is satisfiable

2. **3SAT is NP-hard**

   Every language in NP can be polynomial-time reduced to 3SAT (complex logical formula)

**Corollary:** \(3\text{SAT} \in \text{P if and only if } \text{P} = \text{NP}\)
Theorem (Cook-Levin): 3SAT is NP-complete

Proof Idea:

(1) \( 3\text{SAT} \in \text{NP} \) (done)

(2) Every language \( A \) in \( \text{NP} \) is polynomial time reducible to \( 3\text{SAT} \) (this is the challenge)

We give a poly-time reduction from \( A \) to \( \text{SAT} \)

The reduction converts a string \( w \) into a 3cnf formula \( \phi \) such that \( w \in A \) iff \( \phi \in 3\text{SAT} \)

For any \( A \in \text{NP} \), let \( N \) be a nondeterministic TM deciding \( A \) in \( n^k \) time

\( \phi \) will simulate \( N \) on \( w \)
Deterministic Computation

Nondeterministic Computation

accept or reject

\( n^k \)

\( \exp(n^k) \)
Let $L(N) \in \text{NTIME}(n^k)$. A tableau for $N$ on $w$ is an $n^k \times n^k$ table whose rows are the configurations of some possible computation history of $N$ on $w$.

Each “cell” contains an element $\sigma \in Q \cup \Gamma \cup \{\#\}$.
A tableau is accepting if the last row of the tableau is an accepting configuration.

$N$ accepts $w$ if and only if there is an accepting tableau for $N$ on $w$.

Given $w$, we’ll construct a 3cnf formula $\phi$ with $O(|w|^{2k})$ clauses, describing logical constraints that any accepting tableau for $N$ on $w$ must satisfy.

The 3cnf formula $\phi$ will be satisfiable if and only if there is an accepting tableau for $N$ on $w$. 
Variables of formula $\phi$ will encode a tableau

Let $C = Q \cup \Gamma \cup \{\#\}$

Each of the $(n^k)^2$ entries of a tableau is a cell containing value in $C$

$cell[i,j] = \text{value of the cell at row } i \text{ and column } j$

$= \text{the } j\text{th symbol in the } i\text{th configuration}$

For every $i$ and $j$ ($1 \leq i, j \leq n^k$) and for every $s \in C$ we have a Boolean variable $x_{i,j,s}$ in $\phi$

Total number of variables $= |C|n^{2k}$, which is $O(n^{2k})$

These $x_{i,j,s}$ are the variables of $\phi$ and represent the contents of the cells

We will have: for all $i,j,s$, $x_{i,j,s} = 1 \iff cell[i,j] = s$
Idea: Make $\phi$ so that every satisfying assignment to the variables $x_{i,j,s}$ corresponds to an accepting tableau for $N$ on $w$ (an assignment to all cell[i,j]'s of the tableau).

The formula $\phi$ will be the AND of four CNF formulas:

$$
\phi = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{accept}} \land \phi_{\text{move}}
$$

$\phi_{\text{cell}}$ : for all $i, j$, there is a unique $s \in C$ with $x_{i,j,s} = 1$

$\phi_{\text{start}}$ : the first row of the table equals the start configuration of $N$ on $w$

$\phi_{\text{accept}}$ : the last row of the table has an accept state

$\phi_{\text{move}}$ : every row is a configuration that yields the configuration on the next row
$\phi_{cell}$: for all $i, j$, there is a unique $s \in C$ with $x_{i,j,s} = 1$

\[
\phi_{cell} = \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \land \left( \bigwedge_{s,t \in C} (x_{i,j,s} \lor \neg x_{i,j,t}) \right) \right] \\
\text{for all } i, j \quad \text{at least one } \quad x_{i,j,s} \text{ is set to } 1 \quad \text{at most one } \quad x_{i,j,s} \text{ is set to } 1
\]
\[ \phi_{\text{start}} : \text{the first row of the table equals the start configuration of } N \text{ on } w \]

\[ \phi_{\text{start}} = X_{1,1,\#} \land X_{1,2,q_0} \land \]

\[ X_{1,3,w_1} \land X_{1,4,w_2} \land \ldots \land X_{1,n+2,w_n} \land \]

\[ X_{1,n+3,\square} \land \ldots \land X_{1,n^k-1,\square} \land X_{1,n^k,\#} \]
$\phi_{\text{accept}}$ : the last row of the table has an accept state

$\phi_{\text{accept}} = \bigvee_{1 \leq j \leq n^k} x_{n^j, j, q_{\text{accept}}}$
\( \phi_{\text{move}} \): every row is a configuration that yields the configuration on the next row

**Key Question**: If one row yields the next row, how many cells can be different between the two rows?

**Answer**: at most three cells
\[ \phi_{\text{move}} : \text{every row is a configuration that yields the configuration on the next row} \]

**Idea:** check that every $2 \times 3$ “window” of cells is legal (consistent with the transition function of $N$)

<table>
<thead>
<tr>
<th></th>
<th>$q_0$</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$\ldots$</th>
<th>$w_n$</th>
<th>$\square$</th>
<th>$\ldots$</th>
<th>$\square$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Diagram of a $2 \times 3$ window with cells shaded yellow]
Example: Let $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$
Suppose $a, b, c \in \Gamma$, $q_1, q_2 \in Q$ and
\[
\delta(q_1, a) = \{ (q_1, b, R) \} \\
\delta(q_1, b) = \{ (q_2, c, L), (q_2, a, R) \}
\]

Legal = Consistent with $N$’s transition function

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>q₁</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₂</td>
<td>a</td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>q₁</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>q₂</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>#</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>b</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>q₁</td>
<td>q₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>b</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Illegal = Inconsistent with $N$’s transition function

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>q₁</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>a</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>q₁</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₂</td>
<td>b</td>
<td>q₂</td>
<td>q₂</td>
</tr>
</tbody>
</table>
Key Lemma:
IF  Every window of the tableau is legal, and
    The top row is the start configuration
THEN  Each row of the tableau is a configuration that yields the
next row on the tableau

Proof Sketch: (Strong) induction on the rows.
The top row is a configuration. If it does not yield the next row, then there is a $2 \times 3$ window that is “illegal”
Suppose the first $1,...,k$ rows are configurations which yield the next, and assume every window is legal.
If row $k+1$ did not yield row $k+2$, then there must be a $2 \times 3$ window along those two rows which is “illegal” – contradiction.
The \((i, j)\) window of a tableau is the tuple \((a_1, \ldots, a_6) \in \mathbb{C}^6\) such that:

<table>
<thead>
<tr>
<th>row i</th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>row (i+1)</td>
<td>(a_4)</td>
<td>(a_5)</td>
<td>(a_6)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>col. (j)</th>
<th>col. (j+1)</th>
<th>col. (j+2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_1)</td>
<td>(a_2)</td>
<td>(a_3)</td>
</tr>
<tr>
<td>(a_4)</td>
<td>(a_5)</td>
<td>(a_6)</td>
</tr>
</tbody>
</table>
\( \phi_{\text{move}} : \) every row is a configuration that legally follows from the previous configuration

\[
\phi_{\text{move}} = \bigwedge_{1 \leq i \leq n^k-1} \bigwedge_{1 \leq j \leq n^k-2} (\text{the } (i, j) \text{ window is legal})
\]

\[
\bigvee_{(a_1, \ldots, a_6)} \left( x_{i,j,a_1} \land x_{i,j+1,a_2} \land x_{i,j+2,a_3} \land x_{i+1,j,a_4} \land x_{i+1,j+1,a_5} \land x_{i+1,j+2,a_6} \right)
\]

\((a_1, \ldots, a_6)\) is a legal window

\[
\equiv \bigwedge_{(a_1, \ldots, a_6)} \left( x_{i,j,a_1} \lor x_{i,j+1,a_2} \lor x_{i,j+2,a_3} \lor x_{i+1,j,a_4} \lor x_{i+1,j+1,a_5} \lor x_{i+1,j+2,a_6} \right)
\]

\((a_1, \ldots, a_6)\) is NOT a legal window
How do we get 3SAT?

We had some long clauses in there... how do we convert the whole thing into a 3-cnf formula?

Everything was an AND of ORs (a CNF). We just need to make those ORs small

\[(a_1 \lor a_2 \lor \ldots \lor a_t) \text{ is equivalent to} \]
\[(a_1 \lor a_2 \lor z_1) \land (\neg z_1 \lor a_3 \lor z_2) \land (\neg z_2 \lor a_4 \lor z_3) \ldots \land (\neg z_{t-3} \lor a_{t-1} \lor a_t)\]

(SAT is polynomial time reducible to 3SAT)
What's the total length of $\phi$?

$$\phi = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{accept}} \land \phi_{\text{move}}$$

- $O(n^{2k})$ clauses
- $O(n^k)$ clauses
- $O(n^k)$ clauses
- $O(n^{2k})$ clauses
Summary. We wanted to prove:
Every $A$ in NP has a polynomial time reduction to 3SAT

For every $A$ in NP, we know $A$ is decided by some nondeterministic $n^k$-time Turing machine $N$

We gave a generic method to reduce $(N, w)$ to a 3CNF formula $\phi$ of $O(|w|^{2k})$ clauses such that satisfying assignments to the variables of $\phi$ directly correspond to accepting computation histories of $N$ on $w$

The formula $\phi$ is the AND of four 3CNF formulas:
$\phi = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{accept}} \land \phi_{\text{move}}$
Reading Assignment

Read Luca Trevisan’s notes for an alternative proof of the Cook-Levin Theorem

Sketch:
1. Define CIRCUIT-SAT: Given a logical circuit \( C(y) \), is there an input \( a \) such that \( C(a) = 1 \)?
2. Show that CIRCUIT-SAT is NP-hard: The \( n^k \times n^k \) tableau for \( N \) on \( w \) can be simulated using a logical circuit of \( O(n^{2k}) \) gates
3. Reduce CIRCUIT-SAT to 3SAT in polytime
4. Conclude 3SAT is also NP-hard
Theorem (Cook-Levin):
SAT and 3SAT are NP-complete

Corollary: SAT $\in$ P if and only if P = NP
Is 3SAT solvable in $O(n)$ time on a multitape TM?

Are there logic circuits of size $6n$ for 3SAT?

If yes, then not only is P=NP, but there would be a “dream machine” that could crank out short proofs of theorems, quickly optimize all aspects of life...

recognizing quality work is all you need to produce
There are thousands of NP-complete problems

Your favorite topic certainly has an NP-complete problem somewhere in it

Even the other sciences are not safe: biology, chemistry, physics have NP-complete problems too! 
Given a favorite problem $\Pi \in \text{NP}$, how can we prove it is NP-hard?

Generic Recipe:
1. Take a problem $\Sigma$ that you know to be NP-hard (3-SAT)
2. Prove that $\Sigma \leq_p \Pi$

Then for all $A \in \text{NP}$, $A \leq_p \Sigma$ and $\Sigma \leq_p \Pi$
We conclude that $A \leq_p \Pi$, and $\Pi$ is NP-hard
$\Pi$ is NP-Complete
The Clique Problem

Given a graph $G$ and positive $k$, does $G$ contain a complete subgraph on $k$ nodes?

CLIQUE = $\{ (G,k) \mid G$ is an undirected graph with a $k$-clique $\}$

Theorem (Karp): CLIQUE is NP-complete
Proof Idea: $3\text{SAT} \leq_p \text{CLIQUE}$

Transform a 3-cnf formula $\phi$ into $(G,k)$ such that

$\phi \in 3\text{SAT} \iff (G,k) \in \text{CLIQUE}$

Want transformation that can be done in time that is polynomial in the length of $\phi$

How can we encode a logic problem as a graph problem?
3SAT $\leq_p$ CLIQUE

We transform a 3-cnf formula $\phi$ into $(G, k)$ such that

$$\phi \in 3\text{SAT} \iff (G, k) \in \text{CLIQUE}$$

Let $C_1, C_2, ..., C_m$ be clauses of $\phi$. Assign $k := m$. Make a graph $G$ with $m$ groups of 3 nodes each.

Group $i$ corresponds to clause $C_i$ of $\phi$. Each node in group $i$ is labeled with a literal of $C_i$.

Put edges between all pairs of nodes in different groups, except pairs of nodes with labels $x_i$ and $-x_i$.

Put no edges between nodes in the same group.

When done putting in all the edges, erase the labels.
\((x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)\)

\(|V| = 9\) \hspace{1cm} k = 3
\[(x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor x_2) \land (x_2 \lor x_2 \lor x_2) \land (\neg x_2 \lor \neg x_2 \lor x_1)\]
Claim: \( \phi \in 3\text{SAT} \iff (G,m) \in \text{CLIQUE} \)

Claim: If \( \phi \in 3\text{SAT} \) then \( (G,m) \in \text{CLIQUE} \)

Proof: Given a SAT assignment \( A \) of \( \phi \), for every clause \( C \) there is at least one literal in \( C \) that’s set true by \( A \)

For each clause \( C \), let \( v_C \) be a vertex from group \( C \) whose label is a literal that is set true by \( A \)

Claim: \( S = \{v_C : C \in \phi\} \) is an \( m \)-clique

Proof: Let \( v_C,v_C' \) be in \( S \). Suppose \( (v_C,v_C') \notin E \).

Then \( v_C \) and \( v_C' \) must label inconsistent literals, call them \( x \) and \( \neg x \)

But assignment \( A \) cannot satisfy both \( x \) and \( \neg x \)

Therefore \( (v_C,v_C') \in E \), for all \( v_C,v_C' \in S \).

Hence \( S \) is an \( m \)-clique, and \( (G,m) \in \text{CLIQUE} \)
Claim: $\phi \in 3\text{SAT} \iff (G,m) \in \text{CLIQUE}$

Claim: If $(G,m) \in \text{CLIQUE}$ then $\phi \in 3\text{SAT}$
Proof: Let $S$ be an $m$-clique of $G$. We construct a satisfying assignment $A$ of $\phi$.

Claim: $S$ contains exactly one node from each group.

Now for each variable $x$ of $\phi$, make assignment $A$:
Assign $x$ to 1 $\iff$ There is a vertex $v \in S$ with label $x$

For all $i = 1,\ldots,m$, at least one vertex from group $i$ is in $S$. Therefore, for all $i = 1,\ldots,m$. $A$ satisfies at least one literal in the $i$th clause of $\phi$. Therefore $A$ is a satisfying assignment to $\phi$. 