Lecture 17: Cook-Levin Theorem, NP-Complete Problems
Is SAT solvable in \( O(n) \) time on a multitape TM?

Logic circuits of \( 6n \) gates for SAT?

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

THIS IS AN OPEN QUESTION!
Polynomial Time Reducibility

$f : \Sigma^* \rightarrow \Sigma^*$ is a polynomial time computable function
if there is a poly-time Turing machine $M$ that on every input $w$, halts with just $f(w)$ on its tape

Language $A$ is poly-time reducible to language $B$, written as $A \leq_p B$, if there is a poly-time computable $f : \Sigma^* \rightarrow \Sigma^*$ so that:

$$w \in A \iff f(w) \in B$$

$f$ is a polynomial time reduction from $A$ to $B$

Note there is a $k$ such that for all $w$, $|f(w)| \leq |w|^k$
f converts any string w into a string f(w) such that
\[ w \in A \iff f(w) \in B \]
Theorem: If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$
Theorem: If $A \leq_p B$ and $B \in P$, then $A \in P$

Proof: Let $M_B$ be a poly-time TM that decides $B$. Let $f$ be a poly-time reduction from $A$ to $B$.

We build a machine $M_A$ that decides $A$ as follows:

$M_A = \text{On input } w,$

1. Compute $f(w)$
2. Run $M_B$ on $f(w)$, output its answer

$w \in A \iff f(w) \in B$
Theorem: If $A \leq_p B$ and $B \in \text{NP}$, then $A \in \text{NP}$

Proof: Analogous...
Theorem: If $A \leq_p B$ and $B \in P$, then $A \in P$

Theorem: If $A \leq_p B$ and $B \in NP$, then $A \in NP$

Corollary: If $A \leq_p B$ and $A \notin P$, then $B \notin P$
Definition: A language B is NP-complete if:

1. $B \in \text{NP}$
2. Every $A$ in $\text{NP}$ 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 $L \leq_m A_{TM}$

$A_{TM}$ is “complete for recognizable languages”:
$A_{TM}$ is recognizable, and for all recognizable $L$, $L \leq_m A_{TM}$
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 $k$
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!
The Cook-Levin Theorem:
SAT and 3SAT are NP-complete

1. **3SAT ∈ 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: 3SAT ∈ P if and only if P = NP
Theorem (Cook-Levin): 3SAT is NP-complete
Proof Idea:

(1) $3SAT \in NP$ (done)

(2) Every language $A$ in NP is polynomial time reducible to $3SAT$ (this is the challenge)

We give a poly-time reduction from $A$ to SAT

The reduction converts a string $w$ into a 3cnf formula $\phi$ such that $w \in A$ iff $\phi \in 3SAT$

For any $A \in NP$, let $N$ be a nondeterministic TM deciding $A$ in $n^k$ time

$\phi$ will simulate $N$ on $w$
Deterministic Computation

accept or reject

Nondeterministic Computation

\[ n^k \]

accept or reject

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

<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>

Each “cell” contains a $\sigma \in Q \cup \Gamma \cup \{\#\}$.
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 cell of a tableau contains a symbol from $C$

cell[i,j] = symbol in the cell at row $i$ and column $j$
    = the $j$th symbol in the $i$th configuration

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

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

The $x_{i,j,s}$ variables represent the cells of a tableau

We will enforce the condition: for all $i, j, s$,

$$x_{i,j,s} = 1 \iff \text{cell}[i,j] = s$$
Idea: Make $\phi$ so that every \textit{satisfying assignment} to the variables $x_{i,j,s}$ corresponds to an \textit{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 \textit{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
$\phi_{\text{start}}$ : the first row of the table equals the start configuration of N on w

$$\phi_{\text{start}} = \text{\#}_1, 1, \# \land \text{\#}_2, q_0 \land \text{\#}_3, w_1 \land \text{\#}_4, w_2 \land \cdots \land \text{\#}_{n+2}, w_n \land \text{\#}_1, n+3, \square \land \cdots \land \text{\#}_{n^k-1}, \square \land \text{\#}_{n^k}, \#$$

$\text{O}(n^k)$ clauses
\( \phi_{\text{accept}} : \) the last row of the table has an accept state

\[
\phi_{\text{accept}} = \bigvee_{1 \leq j \leq n^k} x_{n^k,j, q_{\text{accept}}}
\]

<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>

The arrow indicates the direction of the table.
\( \phi_{\text{accept}} : \) the last row of the table has an accept state

\[
\phi_{\text{accept}} = \bigvee_{1 \leq j \leq n^k} x_{n^k,j, q_{\text{accept}}}
\]

How can we convert \( \phi_{\text{accept}} \) into a 3-cnf formula?

The clause \( (a_1 \lor a_2 \lor \ldots \lor a_t) \) is equivalent to

\[
(a_1 \lor a_2 \lor z_1) \land (\neg z_1 \lor a_3 \lor z_2) \land \ldots \land (\neg z_{t-3} \lor a_{t-1} \lor a_t)
\]

where \( z_i \) are new variables.

This produces \( \mathcal{O}(t) \) new 3cnf clauses.

\( \mathcal{O}(n^k) \) clauses
\( \phi_{cell} : \text{for all } i, j, \text{ there is a unique } s \in C \text{ with } x_{i,j,s} = 1 \)

\[
\phi_{cell} = \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \land \left( \bigwedge_{s,t \in C \land s \neq t} (\neg x_{i,j,s} \lor \neg x_{i,j,t}) \right) \right]
\]

for all \( i, j \)

at least one \( x_{i,j,s} \) is set to 1

at most one \( x_{i,j,s} \) is set to 1

\( O(n^{2k}) \) clauses
ϕ_{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!

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>q₁</td>
</tr>
<tr>
<td>#</td>
<td>b</td>
<td>a</td>
<td>q₂</td>
<td>a</td>
</tr>
</tbody>
</table>
$\phi_{\text{move}}$: 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!
\( \phi_{\text{move}} \): 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>#</th>
<th>( q_0 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>\ldots</th>
<th>( w_n )</th>
<th>□</th>
<th>\ldots</th>
<th>□</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>
<td></td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( i \) \( j \)

the \((i, j)\) window
If \( \delta(q_1,a) = \{(q_1,b,R)\} \) and \( \delta(q_1,b) = \{(q_2,c,L), (q_2,a,R)\} \),
which of the following windows are legal?
Key Lemma:
IF Every window of the tableau is legal, and
   The 1\textsuperscript{st} row is the start configuration of N on w
THEN for all $i = 1,\ldots,n^k - 1$, the ith row of the tableau is
a configuration which yields the (i+1)th row.

Proof Sketch: (Strong) induction on $i$.
The 1\textsuperscript{st} row is a configuration. If it \textit{didn’t} yield the 2\textsuperscript{nd}
row, there’s a 2 x 3 “illegal” window on 1\textsuperscript{st} and 2\textsuperscript{nd} rows
Assume rows 1,\ldots,L are all configurations which yield the
next row, and assume every window is legal.
If row $L+1$ did \textit{not} yield row $L+2$, then there’s a 2 x 3
window along those two rows which is “illegal”
The \((i, j)\) window of a tableau is the tuple \((a_1, ..., a_6) \in C^6\) such that:

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

\[ \phi_{\text{move}} = \bigwedge \left( \text{(the (i, j) window is legal)} \right) \]

\( 1 \leq i \leq n^k - 1 \)
\( 1 \leq j \leq n^k - 2 \)

(\text{(the (i, j) 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

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

(a_1, \ldots, a_6)

is NOT a legal window
\[ \phi_{\text{move}} = \bigwedge ( \text{the (i, j) window is "legal"}) \]

\[ 1 \leq i, j \leq n^k \]

the (i, j) window is "legal" =

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

ISN'T "legal"

\[ O(n^{2k}) \text{ clauses} \]
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 a string $w$ 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:

$$\phi = \phi_{\text{cell}} \land \phi_{\text{start}} \land \phi_{\text{accept}} \land \phi_{\text{move}}$$
Theorem (Cook-Levin): SAT and 3SAT are NP-complete

Corollary: SAT ∈ P if and only if P = NP

Given a favorite problem Π ∈ NP, how can we prove it is NP-hard?

Generic Recipe:
1. Take a problem Σ that you know to be NP-hard (3-SAT)
2. Prove that Σ ≤_p Π

Then for all A ∈ NP, A ≤_p Σ and Σ ≤_p Π
We conclude that A ≤_p Π, and Π is NP-hard
\[ \Pi \text{ is NP-Complete} \]
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