CS 154
Cook-Levin Continued,
More NP-Complete Problems
Theorem (Cook-Levin): 3SAT is NP-complete

Proof Idea:

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

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

Poly-time reduction which 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 $w$
f turns any string $w$ into a 3-cnf formula $\phi$ such that $w \in A \iff \phi$ is satisfiable
$\phi$ will simulate an NP machine $N$ on $w$, where $A = L(N)$
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 $w$.
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 $O(|w|^k)$ clauses, describing logical constraints that every 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

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

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

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

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

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

$\phi_{\text{start}} = x_{1,1,#} \land x_{1,2,q_0} \land x_{1,3,w_1} \land x_{1,4,w_2} \land \cdots \land x_{1,n+2,w_n} \land x_{1,n+3,\square} \land \cdots \land x_{1,n^k-1,\square} \land x_{1,n^k,#}$
\[ \phi_{\text{accept}} \colon \text{the last row of the table has an accept state} \]

\[ \phi_{\text{accept}} = \bigvee_{1 \leq j \leq n^k} x_{n^k, j, q_{\text{accept}}} \]
Key Question: If one row yields the next row, how many cells can be different between the two rows?

Answer: AT MOST THREE CELLS!

\[
\begin{array}{cccccc}
# & b & a & a & q_1 & b & c & b & # \\
# & b & a & q_2 & a & c & c & b & # \\
\end{array}
\]
\( \phi_{\text{move}} \) : 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!

\[
\begin{array}{cccccc}
# & b & a & a & q_1 & b & c & b & # \\
# & b & a & q_2 & a & c & c & b & # \\
\end{array}
\]
\[ \phi_{\text{move}} : \text{ every row is a configuration that yields the configuration on the next row} \]

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>#</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>
<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>
<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>

\[ \begin{array}{cccccc}
    \# & q_0 & w_1 & w_2 & \ldots & w_n & \square & \ldots & \square & \# \\
    \# & & & & & & & & & \\
    \# & & & & & & & & & \\
\end{array} \]

\[ \begin{array}{ccc}
    \# & & \\
    \# & & \\
    \# & & \\
\end{array} \]
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 1st row is the start configuration of N on w
THEN   for all i = 1,...,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 1st row is a configuration. If it didn’t yield the 2nd row, there’s a 2 x 3 “illegal” window on 1st and 2nd rows
Assume rows 1,...,L are all configurations which yield the next row, and assume every window is legal.
If row L+1 did 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 \((a_1, ..., a_6) \in \mathbb{C}^6\) such that

\[
\begin{array}{ccc}
\text{col. } j & \text{col. } j+1 & \text{col. } j+2 \\
\text{row } i & a_1 & a_2 & a_3 \\
\text{row } i+1 & a_4 & a_5 & a_6 \\
\end{array}
\]
\( \phi_{move} : \) every row is a configuration that legally follows from the previous configuration

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

\[
1 \leq i \leq n^k - 1
\]

\[
1 \leq j \leq n^k - 2
\]

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

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

\((a_1, \ldots, a_6)\)

is a legal window

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

\((a_1, \ldots, a_6)\)

is NOT a legal window
How do we get 3SAT?

We got a CNF formula, but not a 3CNF... how do we convert the CNF into a 3CNF formula?

A nice trick to “shorten” clauses:

\[(a_1 \lor a_2 \lor \ldots \lor a_t)\] is equivalent to

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

where \(z_i\) are new variables
What’s the total length of $\phi$?

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

\[ O(n^{2k}) \text{ clauses} \]
$\phi_{\text{start}} = x_{1,1,\#} \land x_{1,2,q_0} \land x_{1,3,w_1} \land x_{1,4,w_2} \land \ldots \land x_{1,n+2,w_n} \land x_{1,n+3,\blacksquare} \land \ldots \land x_{1,n^{k-1},\blacksquare} \land x_{1,n^k,\#}$

$O(n^k)$ clauses
\( \phi_{\text{accept}} = \bigvee \ x_{n^k,j, q_{\text{accept}}} \)

\( 1 \leq j \leq n^k \)

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

\((a_1 \lor a_2 \lor z_1) \land (\neg z_1 \lor a_3 \lor z_2) \land (\neg z_2 \lor a_4 \lor z_3) \ldots\)

yields \(O(t)\) new 3cnf clauses

\(O(n^k)\) clauses
\[ \phi_{\text{move}} = \bigwedge ( \text{the (i, j) window is legal} ) \]
\[ 1 \leq i, j \leq n^k \]

the (i, j) window is legal =

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

ISN'T a legal window

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

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

We gave a generic way 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:

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

THESE ARE OPEN QUESTIONS!
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!
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?

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
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: CLIQUE is NP-Complete
$3\text{SAT} \leq_p \text{CLIQUE}$

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 $m$ be the number of clauses of $\phi$. Set $k=m$.

Make a graph $G$ with $m$ groups of 3 nodes each.

- **Group $i$** corresponds to a clause $C_i$ of $\phi$
- Each node in group $i$ is labeled with a literal of $C_i$
- We put edges between all pairs of nodes in different groups, **except those pairs of nodes with labels** $x$ and $\neg x$
- We 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| = 3(number of clauses) \quad k = number of clauses
\((x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor x_2) \land (x_2 \lor x_2 \lor x_2) \land (\neg x_2 \lor \neg x_2 \lor x_1)\)
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 \textit{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 $\phi$
Independent Set

IS: Given a graph $G = (V, E)$ and integer $k$, is there $S \subseteq V$ such that $|S| \geq k$ and no two vertices in $S$ have an edge?

CLIQUE: Given $G = (V, E)$ and integer $k$, is there $S \subseteq V$ such that $|S| \geq k$ and every pair of vertices in $S$ have an edge?

$\text{CLIQUE} \leq_p \text{IS}$:

Given $G = (V, E)$, output $G' = (V, E')$ where

$$E' = \{(u,v) \mid (u,v) \notin E\}.$$  

$(G, k) \in \text{CLIQUE}$ iff $(G', k) \in \text{IS}$
Vertex Cover

vertex cover = set of nodes C that cover all edges
For all edges, at least one endpoint is in C
VERTEX-COVER = \{ (G,k) \mid G \text{ is a graph with a vertex cover of size at most } k \} 

Theorem: VERTEX-COVER is NP-Complete

(1) VERTEX-COVER \in NP

(2) IS \leq_p VERTEX-COVER
Want to transform a graph $G$ and integer $k$ into $G'$ and $k'$ such that

$$(G,k) \in IS \iff (G',k') \in \text{VERTEX-COVER}$$
**IS \leq_p VERTEX-COVER**

Claim: For every graph \( G = (V,E) \), and subset \( S \subseteq V \),

\( S \) is an independent set if and only if \( (V - S) \) is a vertex cover

Proof: \( S \) is an independent set
\[ \iff (\forall u, v \in V)[ (u \in S \text{ and } v \in S) \implies (u,v) \notin E ] \]
\[ \iff (\forall u, v \in V)[ (u,v) \in E \implies (u \notin S \text{ or } v \notin S) ] \]
\[ \iff (V - S) \text{ is a vertex cover} \]

Therefore \( (G,k) \in IS \iff (G,|V| - k) \in VERTEX-COVER \)

Our polynomial time reduction: \( f(G,k) := (G, |V| - k) \)