NP-Completeness and Cook’s Theorem

Theorem: \( L \in \text{NP} \iff \) there exists a poly-time Turing machine \( V \) and \( k \) s.t. we can define \( L \) as:
\[
L = \{ x \mid \exists y \text{ [ } |y| \leq |x|^k \text{ and } V(x,y) \text{ accepts } \} \}
\]

3SAT = \{ \phi \mid \exists y \text{ such that } y \text{ is a satisfying assignment to } \phi \text{ and } \phi \text{ is in 3cnf} \}

SAT = \{ \phi \mid \exists y \text{ such that } y \text{ is a satisfying assignment to } \phi \}

NP = \{ L \mid \text{ all the problems for which the solvable instances have short answers, and once you have the answer, you can efficiently verify it} \}

\[
L \in \text{NP} \iff \text{ } L = \{ x \mid \exists \text{ Answer A of } |x|^k \text{ bits) [Answer A to x is correct]} \}
\]
Assume a reasonable encoding of graphs (example: the adjacency matrix is reasonable)

HAMPATH = \{ (G,s,t) \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t \} 

Theorem: HAMPATH \in NP

The Hamiltonian path is a proof that (G,s,t) is a member of HAMPATH

The k-Clique Problem

A language is in NP if and only if there exist “polynomial-length proofs” for membership in the language

P = the problems that can be efficiently solved

NP = the problems where proposed solutions to the solvable instances can be efficiently verified

P = NP?

“Can Problem Solving Be Efficiently Automated?”
A Clay Institute Millennium Problem

Poly-Time Reducibility

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

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

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

f is a polynomial time reduction from A to B

Note: |f(w)| \leq |w|^k for some constant k

If P = NP...

Mathematicians may be out of a job
Cryptography as we know it would probably be impossible
In principle, every aspect of daily life could be efficiently and globally optimized

Conjecture: P \neq NP
Theorem: If \( A \leq_p B \) and \( B \in P \), then \( A \in P \)

\[ \text{Proof: Let } M_B \text{ be a poly-time TM that decides } B. \]
\[ \text{Let } f \text{ be a poly-time reduction from } A \text{ to } B. \]
\[ \text{We build a machine } M_A \text{ that decides } A \text{ as follows:} \]
\[ M_A = \text{On input } w, \]
\[ 1. \text{Compute } f(w) \]
\[ 2. \text{Run } M_B \text{ on } f(w), \text{ output its answer} \]
\[ w \in A \iff f(w) \in B \]

Definition: A language \( B \) is \( NP \)-complete if:

1. \( B \in NP \)
2. Every \( A \) in \( NP \) is poly-time reducible to \( B \)

That is, \( A \leq_p B \)

\( (B \text{ is } NP\text{-hard}) \)

Suppose \( B \) is \( NP \)-Complete...

Then assuming the conjecture \( P \neq NP \),

\( B \) is not solvable in \( O(n^k) \) time, for any \( k \)

Suppose \( B \) is \( NP \)-Complete...

If \( B \) is \( NP \)-Complete and \( B \in P \), then \( P = NP \)!
Theorem (Cook-Levin): SAT and 3-SAT 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 polytime 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 ∈ NP (already 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 turns a string w into a 3cnf formula φ such that w ∈ A iff φ ∈ 3SAT.

For any A ∈ NP, let N be a nondeterministic TM that decides A in n^k time

φ will simulate the NP machine N for A on w.
A tableau is accepting if any 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, our Boolean 3cnf formula \( \phi \) will describe all the 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.

\[ x_{i,j,s} = 1 \]

means

\[ \text{cell}[i,j] = s \]

\( \text{Variables of our formula } \phi \)

Let \( C = Q \cup \Gamma \cup \{ \# \} \)

Each of the \( (n^2)^2 \) entries of a tableau is a cell

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

= the jth symbol in the ith configuration

For every i and j (1 ≤ i, j ≤ n^k) and for every s ∈ C we have a variable \( x_{i,j,s} \)

total number of variables = \( |C|(n^2) \), 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:

\[ x_{i,j,s} = 1 \iff \text{cell}[i,j] = s \]

We now design \( \phi \) so that a satisfying assignment to the variables \( x_{i,j,s} \) corresponds to an accepting tableau for N on w (an assignment to the cell[i,j])

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, exactly one s ∈ C has \( x_{i,j,s} = 1 \)

\( \phi_{\text{start}} : \) the first row of the tableau is the start configuration of N on w

\( \phi_{\text{accept}} : \) an accepting configuration is the last row of the table

\( \phi_{\text{move}} : \) every row is a configuration that legally follows from the previous configuration

\( \phi_{\text{cell}} : \) for all i, j, exactly one s ∈ C has \( x_{i,j,s} = 1 \)

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

for all i, j

at least one variable is set to 1

at most one variable is set to 1

\( \phi_{\text{start}} : \) the first row of the tableau is the start configuration of N on w

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

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

\[ x_{1,n+3,#} \land \ldots \land x_{1,n^k-1,#} \land x_{1,n^k,#} \]

\[ \begin{array}{cccccc}
\# & q_0 & w_1 & w_2 & \ldots & w_n & \# \\
\# & \# & \# & \# & \# & \# & \# \\
\end{array} \]
\[ \phi_{accept} : \text{an accepting configuration is the last row of the table} \]

\[ \phi_{accept} = \bigvee_{1 \leq j \leq n^2} X_{n^2-1} \cdot q_{accept} \]

**Example N = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_f)**

Suppose \( a, b, c \in \Sigma \), \( q_1, q_2 \in Q \) and

\[ \delta(q_1, a) = (q_1, b, R) \]

\[ \delta(q_2, b) = (q_2, c, L) , (q_2, a, R) \]

Legal = Consistent with N's transition function

\[
\begin{array}{cccc}
(a) & b & c & a \\
(\delta q_1 a) & a & b & c \\
(\delta q_2 b) & c & a & b \\
\end{array}
\]

Illegal = Inconsistent with N's transition function

\[
\begin{array}{cccc}
(a) & a & b & c \\
(\delta q_1 a) & a & b & c \\
(b) & b & c & a \\
\end{array}
\]

The (i, j) window of a tableau is the tuple \((a_1, \ldots, a_6)\) such that

<table>
<thead>
<tr>
<th>col. j</th>
<th>col. j+1</th>
<th>col. j+2</th>
</tr>
</thead>
<tbody>
<tr>
<td>row i</td>
<td>(i, j)</td>
<td>(i, j+1)</td>
</tr>
<tr>
<td></td>
<td>(a_1)</td>
<td>(a_2)</td>
</tr>
<tr>
<td>row i+1</td>
<td>(i+1, j)</td>
<td>(i+1, j+1)</td>
</tr>
<tr>
<td></td>
<td>(a_4)</td>
<td>(a_5)</td>
</tr>
</tbody>
</table>

**Lemma:**

IF

- the top row of the tableau is the start configuration, and
- every window of the tableau is legal,

THEN

each row of the table (as a configuration) yields the next row on the table (as a configuration)

Proof: Very similar to the PCP’s undecidability. If there is no state in the window, then this is easy: the top of the window equals the bottom. If there is a state, then the window has to obey what the transition function can do in one step.
3-SAT?
We had some crazy 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

\[(a_1 \lor a_2 \lor ... \lor a_t) = (a_1 \lor a_2 \lor z_1) \land (\neg z_1 \lor a_3 \lor z_2) \land (\neg z_2 \lor a_4 \lor z_3) ...\]

What's the total length of \(\phi\)?

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

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

\[\text{O}(n^{2k}) \text{ clauses}\]

\[\phi_{\text{start}} = x_{1,1,\#} \land x_{1,2,0} \land x_{1,3,w_1} \land x_{1,4,w_2} \land ... \land x_{1,n+2,w_n} \land x_{1,n+3,\square} \land ... \land x_{1,nk-1,\square} \land x_{1,nk,\#}\]

\[\text{O}(nk) \text{ clauses}\]

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

\[(a_1 \lor a_2 \lor ... \lor a_j) = (a_1 \lor a_2 \lor z_1) \land (\neg z_1 \lor a_3 \lor z_2) \land (\neg z_2 \lor a_4 \lor z_3) ...\]
yields \(O(t)\) new 3cnf clauses

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

\[\text{the } (i, j) \text{ window is legal} = \left( \bigwedge_{(a_1, ..., a_d)} (x_{i,j,a_1} \lor x_{i,j+1,a_2} \lor x_{i+1,j,a_3} \lor x_{i+1,j+1,a_4} \lor x_{i+1,j+2,a_5} \lor x_{i+1,j+3,a_6} \lor x_{i+1,j+4,a_7}) \right)\]

\[\equiv \bigwedge_{(a_1, ..., a_d)} (\neg x_{i,j,a_1} \lor \neg x_{i,j+1,a_2} \lor \neg x_{i+1,j,a_3} \lor \neg x_{i+1,j+1,a_4} \lor \neg x_{i+1,j+2,a_5} \lor \neg x_{i+1,j+3,a_6} \lor \neg x_{i+1,j+4,a_7})\]

ISN'T a legal window

\[\text{O}(nk) \text{ clauses}\]
Summary: Made $\phi$ so that a satisfying assignment to the variables $x_{i,j,s}$ corresponds to an accepting tableau for $N$ on $w$ (an assignment to the cell $[i,j]$)

The formula $\phi$ is 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 $i, j$, exactly one $s \in C$ has $x_{i,j,s} = 1$
- $\phi_{\text{start}}$: the first row of the table is the start configuration of $N$ on $w$
- $\phi_{\text{accept}}$: an accepting configuration is the last row of the table
- $\phi_{\text{move}}$: every row is a configuration that legally follows from the previous configuration

Reading Assignment

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

Sketch: For every deterministic $n^k$ time $V(x,y)$,

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 $V$ can be computed with a logical circuit of $O(n^{2k})$ gates
3. Reduce CIRCUIT-SAT to 3-SAT in polytime
4. Conclude 3-SAT is also NP-hard

There are thousands of NP-complete problems

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

Even the other sciences are not safe: biology, chemistry, physics have NP-complete problems too!

Is 3SAT solvable in $O(n)$ time on a multitape TM?

If yes, then not only is $P=NP$ true, but we would have a “dream machine” that could crank out short proofs of theorems, extreme optimization in all aspects of life...

**THIS IS AN OPEN QUESTION!**

Next Episode:

More NP-Completeness