Contents

- Poly time reduction
- CLIQUE $\leq_P$ Node cover
- Properties of $\leq_P$
- SAT $\leq_P$ 3SAT
- 3SAT $\leq_P$ CLIQUE
- NP-Completeness
- Cook’s Theorem

Recommended Reading:

Dewdney *The (New) Turing Omnibus* — all chaps on NP, Computability, etc.

Lewis and Papdimitriou *Elements of the Theory of Computation*, 2nd ed
Polynomial Time Reduction

**Definition 1** A problem $P(x)$ is **polynomial time reducible** to $Q(x)$ — $P \preceq_P Q$ — iff there is a polynomial time function $f$ s.t. $P(x) \equiv Q(f(x))$.

**Theorem 1** If $P \preceq_P Q$ and $Q \in \mathbf{P}$, then $P \in \mathbf{P}$.

The proof is immediate from Theorem 3 of Lecture 20.
**CLIQUE** $\leq_P$ **Node Cover**

CLIQUE: Is there a clique of size $\geq k$ in $G$?

Reduction: compute the complement $f(G) = \bar{G}$ and then ask, is there a node cover of size $\leq k$ in $\bar{G}$?

Equivalence of the properties is Theorem 7 of Lecture 20.

To compute the complement, flip every bit in $G$’s adjacency matrix $A$. This is $O(n^2)$.
Properties of \( \preceq_P \)

**Property 1** Reflexive: \( Q \preceq_P Q \)

*Proof.* Use the identity function.

**Property 2** Transitive: \( P \preceq_P Q \preceq_P R \Rightarrow P \preceq_P R \)

*Proof.* Let \( P(x) \equiv Q(f(x)) \) and \( Q(x) \equiv R(g(x)) \). So, \( Q(f(x)) \equiv R(g(f(x))) \) and \( P(x) \equiv R(g(f(x))) \). □

**Property 3** Non-symmetric: \( \exists P, Q \) s.t. \( P \preceq_P Q \) and \( \neg(Q \preceq_P P) \).

*Proof.* Otherwise \( P = \text{EXP} \), which we’ve proven false.
Polynomialization

So, ptime reduction gives us a potential recipe for spreading polynomiality far and wide through the algorithmic universe (not everywhere, of course, because of the last point above). Given a ptime algorithm for $Q$ and widespread reducibility to $Q$:

P-reduction: an efficient solution to $Q$ yields an efficient solution to $P$
**SAT \leq_P 3SAT**

3SAT: SAT over CNF formulas s.t. each clause is limited to 3 literals.

Reduction: for any clause in CNF formula $\phi$ with $k$ literals $> 3$, say

$$C = (X_1 \lor X_2 \lor \cdots \lor X_k)$$

replace by (where $Y_1$ is a new Boolean):

$$C' = (X_1 \lor X_2 \lor Y_1) \land (\overline{Y_1} \lor \cdots \lor X_k)$$

$C$ and $C'$ are equivalently satisfiable:

$\Rightarrow$ Suppose an instantiation of variables satisfies $C$. Then either

1. $X_1 \lor X_2$. In this case, set $Y_1 = \bot$.

2. $\bigvee_{j>2} X_j$. In this case, set $Y_1 = \top$.

$\Leftarrow$ Suppose an instantiation satisfies $C'$. Then either

1. $Y_1 = \top$. In this case, $\bigvee_{j>2} X_j = \top$.

2. $\overline{Y_1} = \top$. In this case, $X_1 \lor X_2 = \top$.

Recurse until $k \leq 3$ for each remaining clause. This procedure is ptime in the number of literals.
3SAT $\leq_P$ CLIQUE

Given $\phi = (A_1 \lor B_1 \lor C_1) \land \ldots (A_k \lor B_k \lor C_k)$, build a graph $G$ such that

1. Each literal is represented by one node
2. No nodes representing a clause are (directly) connected
3. No contradictory literals are connected
4. Otherwise, $G$ is fully connected

E.g., $(X_1 \lor X_2 \lor X_2) \land (\overline{X}_1 \lor X_2 \lor X_3)$
3SAT \leq_P CLIQUE

ϕ is satisfiable iff G has a k-clique.

⇒ ϕ is satisfiable. So, some truth assignment works. Take one true literal from each clause; the corresponding nodes make a k-clique.

⇐ G has at least one k-clique. Take one. Assign \top to the corresponding literals. This makes ϕ true, so ϕ is satisfiable.
**NP-Completeness**

What if we could find an NP problem $W$ so general that $\forall P \in \text{NP} \ P \preceq P W$?

**Definition 2** Any such problem $W$ is said to be NP-complete.

If we can find such a problem, and prove that it is actually polynomial time decidable, we could spread $P$ throughout the domain of NP:

**Theorem 2** If $W$ is NP-complete, $P = \text{NP}$ iff $W \in P$.

**Proof.**

$\Rightarrow P = \text{NP}$. Since $W \in \text{NP}, W \in P$.

$\Leftarrow W \in P$. Take any $P \in \text{NP}$. $P \preceq P W$ since $W$ is NP-complete. Hence, $P \in P$. Since $P$ is arbitrary, NP $\subseteq P$. Hence, $P = \text{NP}$.

□
NP-Completeness

**Theorem 3** If \( P \) is NP-complete and \( Q \) is NP and \( P \preceq_P Q \), then \( Q \) is NP-complete.

*Proof.* By transitivity of polynomial reduction.

**Definition 3** \( W \) is NP-hard iff \( \forall Y \in \text{NP} \ Y \preceq_P W \)

**Theorem 4** All NP-complete problems are NP-hard.

This is trivial. The converse is false, however: there are some NP-hard problems which are not NP --- *they are too hard to be NP!*
NP-Completeness

All \textbf{NP}-complete problems are ptime inter-reducible.

There are thousands of known \textbf{NP}-complete and \textbf{NP}-hard problems. Most of the really interesting research problems. E.g.,

- Traveling salesman problem
- Many other optimization problems
- Many graph problems (Hamilton cycle, clique, etc.)
- Satisfiability and its relatives
- Probabilistic reasoning

Practical value of the theory — people have stopped wasting time looking for poly time solutions to these problems (unless their goal is to prove $\textbf{P} = \textbf{NP}$!). So, the search for ptime answers is no longer considered application, but theory.
Cook’s Theorem

**Theorem 5 (Cook)**

SAT is NP-complete.

*Proof.* We already have that SAT ∈ NP.

Given an arbitrary $W \in \text{NP}$ we must produce a Boolean $\phi$ that represents the NTM $M$ which poly time decides $W$. I.e., $\phi$ is satisfiable iff $M$ asserts $W$ is true of its input.

I present the main points of the proof (from Sipser).

Basic idea: Boolean functions are the building blocks of logic circuits. So, build the given NTM with them.

(NB: To simplify this proof, I implicitly alter the definition of a TM in minor ways. E.g., it here has accepting states. These alterations do not change the time complexity of anything material to the proof.)
Cook’s Theorem

Suppose $M$ decides in $n^k - 3$ steps, with input $x$ s.t. $|x| = n$.

We will build a tableau:

Each row $i$ represents the NTM configuration on the $i$-step of a possible computation. Asserting $W(x)$ is equivalent to finding an accepting configuration in some tableau.
Cook’s Theorem

Let’s construct $\phi$.

Let $C = Q \cup S \cup \{\#\}$.

- $Q$ is the set of states of $M$
- $S$ is the alphabet of $M$
- $\#$ is a new delimiter

Then for each tableau cell position $(i, j) \in \{1, \ldots, n^k\}^2$ and each $c \in C$ we define one Boolean variable $x_{i,j,c}$ for $\phi$ such that

$$x_{i,j,c} = \top \text{ iff cell}[i,j] = c$$

So, e.g., there are $|C|$ many variables “observing” each cell; only one of them can be allowed to be true. We use $\phi$ to enforce this and other conditions:

$$\phi = \phi_{cell} \land \phi_{start} \land \phi_{accept} \land \phi_{move}$$
Cook’s Theorem

\[ \phi_{cell} = \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{c \in C} x_{i,j,c} \right) \wedge \left( \bigwedge_{c, d \in C; c \neq d} \overline{x_{i,j,c}} \vee \overline{x_{i,j,d}} \right) \right] \]

Left side: each cell contains at least one symbol.

Right side: each cell contains at most one symbol.

Outermost conjunction operator ranges over all cells in the tableau.
Cook’s Theorem

\[ \phi_{start} = x_{1,1,\#} \land x_{1,2,q_0} \land x_{1,3,x_1} \land \cdots \land x_{1,n+2,x_n} \land \cdots \land x_{1,n^k,\#} \]

The first row is forced to take the initial configuration.

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

Require an accepting state to occur.
Cook’s Theorem

$\phi_{move}$: intuitively, require each subsequent row to be computable in one step by $M$ from the prior row.

To do this, ensure that each 2x3 window on the tableau is legal.

Example: suppose $\langle q_1, q_1, s, s', R \rangle \in M$. Then these two are legal windows:

If this is the only quintuple, then the following is illegal:

Formal Methods II
Cook’s Theorem

Nondeterministic rules just imply that multiple legal windows will share their top rows.

Another illegal window:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>

Formal definition of *legal(window)* as a set of Boolean conditions: exercise for reader.

Fairly complex; also omitted from Sipser.
Cook’s Theorem

Let’s identify windows by their top-middle cell in the tableau. I.e.,

\[
\begin{array}{c|c|c|c|c}
\hline
i & (i,j) & \text{window } (i,j) \\
\hline
\end{array}
\]

Then the final condition is:

\[
\phi_{move} = \bigwedge_{1 \leq i \leq n^k - 1 \atop 2 \leq j \leq n^k - 1} \text{legal(window}(i,j))
\]
Cook’s Theorem

Property 4 If every window is legal, then the tableau corresponds to a possible $M$ computation.

Proof. Informally, because computations act locally on the tape and because other constraints (state transitions) are adhered to in the formal definition of legality.

Hence, $\forall W \in \text{NP}, W \preceq SAT$. Furthermore, $W \preceq_P SAT$.

The tableau is $n^k \times n^k = n^{2k}$. The number of variables is $O(n^{2k})$. So, given $M$ we can construct and check $\phi$ in polynomial time. \qed