Lecture 23:
• NP-completeness

Reading:
Sipser Ch 7.4-7.5

Mark Bun
April 21, 2021

Final due
Th 5pm (5/6)
Last time: Two equivalent definitions of \textbf{NP}

1) NP is the class of languages decidable in polynomial time on a nondeterministic TM

\[
\text{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k)
\]

2) A \textit{polynomial-time verifier} for a language \( L \) is a \textit{deterministic} \( \text{poly}(|w|) \)-time algorithm \( V \) such that \( w \in L \) iff there \textit{exists} a certificate \( c \) such that \( V((w, c)) \) accepts

\textbf{Theorem:} A language \( L \in \text{NP} \) iff there is a polynomial-time verifier for \( L \)
NP-Completeness
Understanding the $P$ vs. $NP$ question

Believe $P \neq NP$, but very far from proving it

**Question 1:** How can studying specific computational problems help us get a handle on resolving $P$ vs. $NP$?

**Question 2:** What would $P \neq NP$ allow us to conclude about specific problems we care about?

**Idea:** Identify the “hardest” problems in $NP$. Find $L \in NP$ such that $L \in P$ \iff $P = NP$
Recall: Mapping reducibility

Definition:
A function \( f : \Sigma^* \rightarrow \Sigma^* \) is **computable** if there is a TM \( M \) which, given as input any \( w \in \Sigma^* \), halts with only \( f(w) \) on its tape.

Definition:
Language \( A \) is **mapping reducible** to language \( B \), written \( A \leq_m B \), if there is a computable function \( f : \Sigma^* \rightarrow \Sigma^* \) such that for all strings \( w \in \Sigma^* \), we have \( w \in A \iff f(w) \in B \).
Polynomial-time reducibility

Definition:
A function $f : \Sigma^* \to \Sigma^*$ is polynomial-time computable if there is a polynomial-time TM $M$ which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape.

Definition:
Language $A$ is polynomial-time reducible to language $B$, written

$$A \leq_p B$$

if there is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \iff f(w) \in B$
Implications of poly-time reducibility

cf.  \( A \leq_m B \) and \( B \) decidable then \( A \) decidable

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

Proof: Let \( M \) decide \( B \) in poly time, and let \( f \) be a poly-time reduction from \( A \) to \( B \). The following TM decides \( A \) in poly time:

1. Compute \( f(w) \)
2. Run \( M \) on \( f(w) \). If accepts, accept. If rejects, reject.

On input \( w \):

- Run time:
  1. \( f \) poly-time computable
  2. \( |f(w)| = \text{poly}(|w|) \)
  \( \Rightarrow M \) is run on input of poly length

- Correctness:
  \( w \in A \iff f(w) \in B \)
  \( \iff M \) accepts \( f(w) \)
  \( \iff \) new TM accepts
Is **NP** closed under poly-time reductions?

Analog: \( P \) is decidable, \( NP \) is recognizable

If \( A \leq_p B \) and \( B \) is in NP, does that mean \( A \) is also in NP?

- **a)** Yes, the same proof works using NTMs instead of TMs
- **b)** No, because the new machine is an NTM instead of a deterministic TM
- **c)** No, because the new NTM may not run in polynomial time
- **d)** No, because the new NTM may accept some inputs it should reject
- **e)** No, because the new NTM may reject some inputs it should accept
NP-completeness

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

1) $B \in \text{NP}$, and

2) Every language $A \in \text{NP}$ is poly-time reducible to $B$, i.e., $A \leq_p B$ ("$B$ is NP-hard")
Implications of NP-completeness

Theorem: Suppose $B$ is NP-complete.

Then $B \in P$ iff $P = NP$

Proof:

$\leftarrow$ Suppose $P = NP$. Then since $B$ is NP-complete, $B \in NP \subseteq P$.

$\Rightarrow$ Suppose $B \in P$. Let $A \in NP$ be any language.

Now that since $B$ is NP-hard, $A \leq_P B$.

$\Rightarrow A \in P$.

$\Rightarrow NP \subseteq P$. $\Rightarrow NP = P.$
Implications of NP-completeness

**Theorem:** Suppose $B$ is NP-complete.

Then $B \in P$ iff $P = NP$

**Consequences of $B$ being NP-complete:**

1) If you want to show $P = NP$, you just have to show $B \in P$
2) If you want to show $P \neq NP$, you just have to show $B \notin P$
3) If you already believe $P \neq NP$, then you believe $B \notin P$
Cook-Levin Theorem and NP-Complete Problems
Do NP-complete problems exist?

Theorem: \( TMSAT = \{ \langle N, w, 1^t \rangle \mid \text{NTM } N \text{ accepts input } w \text{ within } t \text{ steps} \} \) is NP-complete

Proof sketch:  
1) \( TMSAT \in \text{NP} \): Certificate = \( t \) nondeterministic guesses made by \( N \), verifier checks that \( N \) accepts \( w \) within \( t \) steps under those guesses.
2) \( TMSAT \) is NP-hard: Let \( L \in \text{NP} \) be decided by NTM \( N \) running in time \( T(n) \). The following poly-time TM shows \( L \leq_p TMSAT \): “On input \( w \) (an instance of \( L \)):
   Output \( \langle N, w, 1^T(|w|) \rangle \).”
Cook-Levin Theorem

**Theorem:** \( SAT \) (Boolean satisfiability) is NP-complete

**“Proof”:** Already know \( SAT \in NP \). (Much) harder direction: Need to show every problem in NP reduces to \( SAT \)

Stephen A. Cook (1971)  
Leonid Levin (1973)
New NP-complete problems from old

**Lemma:** If \( A \leq_p B \) and \( B \leq_p C \), then \( A \leq_p C \)

(poly-time reducibility is transitive)

**Theorem:** If \( C \in \text{NP} \) and \( B \leq_p C \) for some NP-complete language \( B \), then \( C \) is also NP-complete
**New NP-complete problems from old**

All problems below are NP-complete and hence poly-time reduce to one another!

![Diagram showing reductions among NP-complete problems]

*by definition of NP-completeness*
3SAT (3-CNF Satisfiability)

Definitions:
• A literal either a variable of its negation \( x_5, \overline{x_7} \)
• A clause is a disjunction (OR) of literals \( \text{Ex. } x_5 \lor \overline{x_7} \lor x_2 \)
• A 3-CNF is a conjunction (AND) of clauses where each clause contains exactly 3 literals

\[ C_1 \land C_2 \land \ldots \land C_m = \]
\[ (x_5 \lor \overline{x_7} \lor x_2) \land (\overline{x_3} \lor x_4 \lor x_1) \land \ldots \land (x_1 \lor x_1 \lor x_1) \]

3SAT = \{ \langle \varphi \rangle | \varphi \text{ is a satisfiable } 3 - \text{CNF} \}

\[ \text{s.t. } \varphi(x_1, \ldots, x_n) = 1^3 \]
**3SAT** is NP-complete

**Theorem:** 3SAT is NP-complete

**Proof idea:** 1) 3SAT is in NP (why?)

2) Show that \( SAT \leq_p 3SAT \)

[Aside: Want 2SAT NP-hard \( \Rightarrow V A \in NP, A \leq_p 3SAT \) suffices by transitivity of \( \leq_p \)]

Your classmate suggests the following reduction from SAT to 3SAT: “On input \( \varphi \), a 3-CNF formula (an instance of 3SAT), output \( \varphi \), which is already an instance of SAT.” Is this reduction correct?

a) Yes, this is a poly-time reduction from SAT to 3SAT

b) No, because \( \varphi \) is not an instance of the SAT problem

c) No, the reduction does not run in poly time

d) No, this is a reduction from 3SAT to SAT; it goes in the wrong direction
**3SAT is NP-complete**

**Theorem:** 3SAT is NP-complete

**Proof idea:** 1) 3SAT is in NP (why?)

2) Show that SAT ≤ₚ 3SAT

Idea of reduction: Give a poly-time algorithm converting an arbitrary formula φ into a 3CNF ψ such that φ is satisfiable iff ψ is satisfiable

On input φ (formula, instance of SAT):

1) Convert φ to 3CNF ψ

2) Output ψ
Converting $\varphi$ to $\psi$

$\varphi$: $\neg x_1 \lor x_2 \land x_3$

$\psi$: $\neg a \lor b \land c$

$\hat{\varphi}(x_1, x_2, x_3, x_4, a, b, c) =$ $((a \Leftrightarrow x_4 \lor x_1) \land (b \Leftrightarrow x_2 \land x_3) \land (c \Leftrightarrow a \land b) \land (c \lor c \lor c))$

**Theorem:** A function $f: \{0,1\}^3 \rightarrow \{0,1\}$ is a 3CNF formula computing $f$ if

Use theorem to expand each into a 3CNF.
Independent Set

An independent set in an undirected graph $G$ is a set of vertices that includes at most one endpoint of every edge.

\[
\text{INDEPENDENT} \subseteq \text{SET} = \{ \langle G, k \rangle | G \text{ is an undirected graph containing an independent set with } \geq k \text{ vertices} \}\]

• Is there an independent set of size $\geq 6$?
  • Yes. \( \langle G, 6 \rangle \in \text{INDEPENDENT} \)

• Is there an independent set of size $\geq 7$?
  • No. \( \langle 6, 7 \rangle \notin \text{INDEPENDENT} \)
Independent Set is NP-complete

1) \text{INDEPENDENT \text{–} SET} \in \text{NP}
2) Reduce \text{3SAT} \leq_p \text{INDEPENDENT \text{–} SET}

**Proof of 1)** The following gives a poly-time verifier for \text{INDEPENDENT \text{–} SET}

**Certificate:** Vertices \(v_1, \ldots, v_k\)

**Verifier:**

"On input \(\langle G, k; v_1, \ldots, v_k \rangle\), where \(G\) is a graph, \(k\) is a natural number,

1. Check that \(v_1, \ldots, v_k\) are distinct vertices in \(G\)
2. Check that there are no edges between the \(v_i\)'s."

If all checks pass

\[4/21/2021\]

CS332 - Theory of Computation 22
Independent Set is NP-complete

1) \textit{INDEPENDENT} \textit{--- SET} \in \textit{NP}

2) \text{Reduce } 3SAT \leq_p \textit{INDEPENDENT} \textit{--- SET}

\textbf{Proof of 2)} The following TM computes a poly-time reduction.

“On input \(\varphi\), where \(\varphi\) is a 3CNF formula,

1. Construct graph \(G\) from \(\varphi\)
   - \(G\) contains 3 vertices for each clause, one for each literal.
   - Connect 3 literals in a clause in a triangle.
   - Connect every literal to each of its negations.

2. Output \(\langle G, k \rangle\), where \(k\) is the number of clauses in \(\varphi\).”
Example of the reduction

\[ \varphi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_3) \]
Proof of correctness for reduction

Let $k = \# \text{ clauses and } l = \# \text{ literals in } \varphi$

Correctness: $\varphi$ is satisfiable iff $G$ has an independent set of size $k$

$\implies$ Given a satisfying assignment, select one true literal from each triangle. This is an independent set of size $k$

$\iff$ Let $S$ be an independent set in $G$ of size $k$

- $S$ must contain exactly one vertex in each triangle
- Set these literals to true, and set all other variables in an arbitrary way
- Truth assignment is consistent and all clauses are satisfied

Runtime: $O(k + l^2)$ which is polynomial in input size