Lecture 21:

- NP-Completeness
- Cook-Levin Theorem
- Reductions

Reading:
Sipser Ch 7.3-7.5

Mark Bun
April 15, 2020
Last time: Two equivalent definitions of NP

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

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

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

**Theorem:** A language \( L \in \text{NP} \) iff there is a polynomial-time verifier for \( L \)
Examples of NP languages: SAT

“Is there an assignment to the variables in a logical formula that make it evaluate to true?”

- **Boolean variable**: Variable that can take on the value true/false (encoded as 0/1)
- **Boolean operations**: \( \land \) (AND), \( \lor \) (OR), \( \neg \) (NOT)
- **Boolean formula**: Expression made of Boolean variables and operations. \( \text{Ex: } (x_1 \lor \overline{x}_2) \land x_3 \)
- **An assignment of 0s and 1s to the variables** satisfies a formula \( \varphi \) if it makes the formula evaluate to 1
- **A formula \( \varphi \) is satisfiable** if there exists an assignment that satisfies it
Examples of **NP** languages: SAT

Ex: \((x_1 \lor \overline{x_2}) \land x_3\)

Yes: \(x_1 = 1, x_2 = 1, x_3 = 1\) Satisfiable?

Ex: \((x_1 \lor x_2) \land (x_1 \lor \overline{x_2}) \land \overline{x_2}\)

\(x_1 = 1, x_2 = 0\) Satisfiable?

\[ SAT = \{ \langle \varphi \rangle | \varphi \text{ is a satisfiable formula} \} \]

\( \chi_1 \land \overline{\chi_2} \notin SAT \)

**Claim:** \(SAT \in NP\)

**On input \(\langle \varphi \rangle\):**

1) Guess an assignment to \(x_1, \ldots, x_k \in \{0,1\}\)

2) Check that assignment is satisfying

Accept if yes, reject o.w.
Examples of \textbf{NP} languages: TSP

“Given a list of cities and distances between them, is there a ‘short’ tour of all of the cities?”

More precisely: Given

- A number of cities \( m \)
- A function \( D: \{1, \ldots, m\}^2 \rightarrow \mathbb{N} \) giving the distance between each pair of cities
- A distance bound \( B \)

\[
TSP = \{ \langle m, D, B \rangle | \exists \text{ a tour visiting every city with length } \leq B \} \]
**P vs. NP**

**Question:** Does $P = NP$?

Philosophically: Can every problem with an efficiently verifiable solution also be solved efficiently?

A central problem in mathematics and computer science

[Diagram showing $P$ vs. $NP$ relationships]

If $P = NP$, all $NP$-complete problems are in $P$.

If $P \neq NP$, $NP$-complete problems are not in $P$. 

---

Millennium Problems

- Yang–Mills and Mass Gap
  - Experiment and computer simulations suggest the existence of a “mass gap” in the solution to the quantum versions of the Yang–Mills equations. But a proof of this property is unknown.

- Riemann Hypothesis
  - This problem asks about the distribution of the zeros of the Riemann zeta function. The Riemann hypothesis states that the real part of each non-trivial zero is 1/2. Forms in Riemann’s 1859 paper assert that all the non-trivial zeros of the zeta function have real part equal to 1/2.

- P vs. NP Problem
  - The answer to this question determines whether efficient solutions exist for the vast majority of practical problems that are computationally tractable but not known to be solvable efficiently.

- Navier–Stokes Equation
  - This equation governs fluid flow and is not yet proven to have smooth solutions. However, there is no proof for the most basic questions of whether solutions exist, are unique, and are smooth or not. There is a $1M prize for a correct proof. Because a counterexample gives no counterexample, but a counterexample is understood.

- Hodge Conjecture
  - The conjecture concerns the algebraic cycles on algebraic varieties, and the Hodge conjecture states that certain algebraic cycles are algebraic cycles on abelian varieties and that the Hodge conjecture holds for certain special cases.

- Poincaré Conjecture
  - This question is about 3-dimensional manifolds. The Poincaré conjecture states that every three-manifold is homeomorphic to a sphere. Its proof was published in 2002 by Grigori Perelman.

- Birch and Swinnerton-Dyer Conjecture
  - This conjecture relates the number of rational points on an elliptic curve to the rank of the group of rational points. It is considered one of the most important unsolved problems in algebraic geometry.

---

4/15/2020

CS332 - Theory of Computation
A world where $P = NP$

- Many important decision problems can be solved in polynomial time ($HAMPATH$, SAT, TSP, etc.)

- Many search problems can be solved in polynomial time (e.g., given a natural number, find a prime factorization)

- Many optimization problems can be solved in polynomial time (e.g., find the lowest energy conformation of a protein)
A world where $P = \text{NP}$

- Secure cryptography becomes impossible
  An NP search problem: Given a ciphertext $C$, find a plaintext $m$ and encryption key $k$ that would encrypt to $C$

- AI / machine learning become easy: Identifying a consistent classification rule is an NP search problem

- Finding mathematical proofs becomes easy: NP search problem: Given a mathematical statement $S$ and length bound $k$, is there a proof of $S$ with length at most $k$?

General consensus: $P \neq \text{NP}$
NP-Completeness
What about a world where $P \neq NP$

Believe this to be true, but very far from proving it

$P \neq NP$ implies that there is a problem in $NP$ which cannot be solved in polynomial time, but it might not be a useful one

**Question:** What would $P \neq NP$ allow us to conclude about 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

**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. **Run machine computing reduction to produce $f(w)$**
2. **Run $M$ on $f(w)$**
3. **If $M$ accepts, accept. o.w. reject.**

Runs in poly time. $M$ is poly-time, run on poly-length input $f(w)$.

4/15/2020

CS332 - Theory of Computation
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:

$\iff$ if $P = NP$ and $B \in NP$, then $B \in P$

$\implies$ if $B \in P$, let $A$ be an arbitrary language $A \in NP$

$\implies A \leq_P B \implies A \in P$ (because of $B \in P$)

$\implies P = NP$
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
Cook-Levin Theorem

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

**Proof:** Already know \( SAT \in \text{NP} \). Need to show every problem in NP reduces to \( SAT \) (later?)

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)

$\text{SAT} \leq_p C$

**Theorem:** If $C \in \text{NP}$ and $B \leq_p C$ for some NP-complete language $B$, then $C$ is also NP-complete

**Proof.** Need to show $A \leq_p C$ for every $A \in \text{NP}$

If $A \in \text{NP}$, $A \leq_p B$ (since $B$ is NP-complete)

$B \leq_p C$ (hypothesis)

$\Rightarrow A \leq_p C$ (transitivity)
New NP-complete problems from old

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

by definition of NP-completeness

DIAGRAM:

- SAT
  - 3SAT
    - INDEPENDENT SET
      - VERTEX COVER
        - SET COVER
    - DIR-HAM-CYCLE
      - HAM-CYCLE
    - GRAPH 3-COLOR
    - SUBSET-SUM
      - SCHEDULING

4/15/2020 CS332 - Theory of Computation
3SAT (3-CNF Satisfiability)

Definition(s):

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

Example: $C_1 \land C_2 \land \ldots \land C_m = (x_5 \lor \overline{x_7} \lor x_2) \land (\overline{x_3} \lor x_4 \lor x_1) \land \ldots \land (x_1 \lor x_1 \lor x_1)$

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

Theorem: 3SAT is NP-complete

Proof idea: 1) 3SAT is in NP (why?)
   2) Show that SAT \( \leq_p \) 3SAT

Idea of reduction: Given a poly-time algorithm converting an arbitrary formula \( \varphi \) into a 3CNF \( \psi \) such that \( \varphi \) is satisfiable iff \( \psi \) is satisfiable.

\[
\varphi = \bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_3 \land x_1 \land x_2 \land x_3
\]

\[
\psi = (a_1 \equiv a_2 \lor a_3) \land
       (a_2 \equiv x_1 \land \bar{x}_2) \land
       (a_3 \equiv x_3 \lor a_5) \land
       (a_4 \equiv x_3 \land x_2) \land
       (a_6 \equiv x_1 \land \bar{x}_2)
\]

In general:
- Can convert any function \( f: \{0,1\}^3 \to \{0,1\} \)
- Rule a 3CNF