Lecture 22:

- NP-Completeness Example
- Space Complexity
- Savitch’s Theorem

Reading:
Sipser Ch 8.1-8.2
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")

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

\[ \text{e.g. SAT, 3-SAT} \]
3SAT (3-CNF Satisfiability)

Definition(s):

• A literal is 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
  \( \text{Ex. } C_1 \land C_2 \land \ldots \land C_m = \)
  \( (x_5 \lor \overline{x}_7 \lor x_2) \land (\overline{x}_3 \lor x_4 \lor x_1) \land \ldots \land (x_1 \lor x_1 \lor x_1) \)

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

Cook-Levin Theorem: 3SAT is NP-complete
3SAT $\leq_P A_{TM}$ \ ($A_{TM}$ is NP-hard)

Idea: Hardcode input formula $\Phi$ into a TM $M$

s.t., say, $M$ accepts $w$ if $\Phi$ is satisfiable

$M = "\text{On input } w:\"

1) Ignore $w$
2) Try $\Phi$ on all possible assignments and check if any accept"

Computing $\langle M, \Phi \rangle$ from $\langle \Phi \rangle$ can be done in polynomial

$A_{TM} \notin \text{NP}$ \ ($\therefore A_{TM} \not\text{ is NP-complete}$) \quad A \leq_P \Pi \in \text{NP} \Rightarrow A \in \text{NP} \quad 4/22/2020$
Some general reduction strategies

• Reduction by simple equivalence

  Ex. \textsc{Independent} – \textsc{Set} \leq_p \textsc{Vertex} – \textsc{Cover}
  and \textsc{Vertex} – \textsc{Cover} \leq_p \textsc{Independent} – \textsc{Set}

  \begin{align*}
  G \text{ has an ind. set of size } \frac{3}{2}n & \iff G \text{ has a vertex cover of size } 3n \\
  \text{ (complement of an ind. set is a vertex cover) }
  \end{align*}

• Reduction from special case to general case

  Ex. \textsc{Vertex} – \textsc{Cover} \leq_p \textsc{Set} – \textsc{Cover}

• Gadget reductions

  Ex. \textsc{3Sat} \leq_p \textsc{Independent} – \textsc{Set}
Independent Set

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

$\text{INDEPENDENT} - \text{SET} = \{ (G, k) | G \text{ is an undirected graph containing an independent set with } \geq k \text{ vertices} \}$

- Is there an independent set of size $\geq 6$?
  - Yes.

- Is there an independent set of size $\geq 7$?
  - No.
Independent Set is NP-complete

1) \textbf{INDEPENDENT} – \textbf{SET} \in \text{NP} \quad \text{[certificate = candidate ind. set of size } k]\]

2) Reduce \textbf{3SAT} \leq_p \text{INDEPENDENT} – \text{SET}

   Convert \( \varphi \) into a pair \( \langle G, k \rangle \) s.t. \( \varphi \) satisfiable \( \iff \) \( G \) has an ind. set of size \( k \).

Proof. “On input \( \langle \varphi \rangle \), 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 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

\[ \varphi = (\overline{x_1} \vee x_2 \vee x_3) \land (x_1 \vee \overline{x_2} \vee x_3) \land (\overline{x_1} \vee x_2 \vee x_4) \]

\[ u = 3 \]

\[ \text{# clauses} \]

\[ \chi_1 = 0 \]
\[ \chi_2 = \text{either 0 or 1} \]
\[ \chi_3 = 1 \]
\[ \chi_4 = 1 \]

This is a satisfying assignment to \( \varphi \)
Proof of correctness for reduction

Let \( k = \# \) clauses and \( l = \# \) literals in \( \varphi \)

Claim: \( \varphi \) is satisfiable iff \( G \) has an ind. set of size \( k \)

\[ (x_1 \lor x_2 \lor \overline{x_3}) \land (x_2 \lor \overline{x_4} \lor \overline{x_5}) \land (x_5 \lor \overline{x_1} \lor \overline{x_2}) \]

\[ \frac{\alpha(l \log l)}{b \cdot k} \text{ bin/clause} \Rightarrow \text{total input words} = \frac{\alpha(n \log l)}{l} \]  

\[ \begin{array}{c}
\text{Runtime: } O(k + l^2) \text{ which is polynomial in input size } \\
\Rightarrow O(k^2)
\end{array} \]

\[ 4/22/2020 \quad \text{CS332 - Theory of Computation} \]
Space Complexity
Complexity measures we’ve studied so far

• Deterministic time \( \text{TIME} \)
• Nondeterministic time \( \text{NTIME} \)
• Classes \( \text{P}, \text{NP} \)

Many other resources of interest:

Space (memory), randomness, parallel runtime / #processors, quantum entanglement, interaction, communication, ...
Space analysis

Space complexity of a TM (algorithm) = maximum number of tape cells it uses on a worst-case input

Formally: Let \( f : \mathbb{N} \rightarrow \mathbb{N} \). A TM \( M \) runs in space \( f(n) \) if on every input \( w \in \Sigma^* \), \( M \) halts on \( w \) using at most \( f(n) \) cells

For nondeterministic machines: Let \( f : \mathbb{N} \rightarrow \mathbb{N} \). An NTM \( N \) runs in space \( f(n) \) if on every input \( w \in \Sigma^* \), \( N \) halts on \( w \) using at most \( f(n) \) cells on every computational branch
Space complexity classes

Let $f : \mathbb{N} \rightarrow \mathbb{N}$

A language $A \in \text{SPACE}(f(n))$ if there exists a basic single-tape (deterministic) TM $M$ that
1) Decides $A$, and
2) Runs in space $O(f(n))$

A language $A \in \text{NSPACE}(f(n))$ if there exists a single-tape nondeterministic TM $N$ that
1) Decides $A$, and
2) Runs in space $O(f(n))$
Example: Space complexity of SAT

Theorem: $SAT \in SPACE(n)$

Proof: The following deterministic TM decides $SAT$ using linear space

On input $\langle \varphi \rangle$ where $\varphi$ is a Boolean formula:

1. For each truth assignment to the variables $x_1, \ldots, x_m$ of $\varphi$:
2. Evaluate $\varphi$ on $x_1, \ldots, x_m$
3. If any evaluation $= 1$, accept. Else, reject.
Example: NFA analysis

Theorem: Let $ALL_{NFA} = \{ A \mid A$ is an NFA with $L(A) = \Sigma^* \}$
Then $ALL_{NFA} \in \text{NSPACE}(n)$.

Proof: The following NTM decides $ALL_{NFA}$ in linear space

On input $\langle A \rangle$ where $A$ is an NTM:
1. Place a marker on the start state of $A$.
2. Repeat $2^q$ times where $q$ is the # of states of $A$:
   - Nondeterministically select $a \in \Sigma$.
3. Adjust the markers to simulate all ways for $A$ to read $a$.
4. Accept if at any point none of the markers are on an accept state. Else, reject.
Example

Step 0: Mark 1
Step 1: Simulate on 1  
Current state: 1
Step 2: Simulate on 0  
Current: 10
Step 3: Simulate on 1  
Current: 101

Conclude: 101 \in L(A)  
\Rightarrow A \in ALLNFA
Space vs. Time
Space vs. Time

\[ \text{TIME}(f(n)) \subseteq \text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n)) \]

How about the opposite direction? Can low-space algorithms be simulated by low-time algorithms?
Reminder: Configurations

A configuration is a string $uqv$ where $q \in Q$ and $u, v \in \Gamma^*$

- Tape contents $= uv$ (followed by blanks $\sqcup$)
- Current state $= q$
- Tape head on first symbol of $v$

Example: $101q_50111$

Start configuration: $q_0w$
Accepting configuration: $q = q_{\text{accept}}$
Rejecting configuration: $q = q_{\text{reject}}$
Reminder: Configurations

Consider a TM with

- \( k \) states
- tape alphabet \( \{0, 1\} \)
- space \( f(n) \)

How many configurations are possible when this TM is run on an input \( w \in \{0,1\}^n \)?

- \( k \cdot f(n) \cdot 3^{f(n)} \) possible strings repeating tape content
- \( k \) states
- \( f(n) \) locations for head

Observation: If a TM enters the same configuration twice when run on input \( w \), it loops forever.

Corollary: A TM running in space \( f(n) \) also runs in time \( 2^{O(f(n))} \)

\[ \text{SPACE}(f(n)) \subseteq \text{TIME}(2^{O(f(n))}) \]
Savitch’s Theorem
Savitch’s Theorem: Deterministic vs. Nondeterministic Space

Theorem: Let $f$ be a function with $f(n) \geq n$. Then

$$NSPACE(f(n)) \subseteq SPACE\left( (f(n))^2 \right).$$

\[
\begin{align*}
\text{PSPACE} & = \bigcup_{k=0}^{\infty} \text{SPACE}(n^k) \\
\text{NPSPACE} & = \bigcup_{k=0}^{\infty} \text{NSPACE}(n^k) \\
\text{PSPACE} & = \text{NPSPACE} & \text{Corollary: } \forall n \in \mathbb{N}, \text{ALL}_{n^k} \subseteq \text{PSPACE}
\end{align*}
\]