- Measuring Complexity (Section 7.1)
- The Class P (Section 7.2)
- The Class NP (Section 7.3)
- Polynomial Time Reducibility (Section 7.4)

Some material from slides by M. Sipser
Announcements

- Homework 3 due Wednesday March 17, 2021 at 11:59pm (NY Time)
Time Complexity

- How many steps are needed to decide a language A?
  - A function of the input
  - We give an upper bound for all inputs of lengths n.
    - Called **worst-case time complexity**.
Number of steps to decide \( A = \{ a^k b^k \mid k \geq 0 \} \)

- \( M = \) On input \( w \)
  1. Scan input to check if \( w \in a^*b^* \). If not, reject.
  2. Repeat until all crossed off.
     - Scan tape, crossing off one \( a \) and one \( b \).
     - Reject if only \( a \)'s or only \( b \)'s remain.
  3. Accept if all crossed off.

- **Theorem:** A single tape TM can decide \( A \) in at most \( cn^2 \) steps for some fixed constant \( c \).
Deciding $A = \{ a^k b^k \mid k \geq 0 \}$ faster

- **Theorem:** A single tape TM can decide $A$ using $O(n \log n)$ steps.
Deciding $A = \{ a^k b^k \mid k \geq 0 \}$ even faster

- **Theorem:** A multi-tape TM can decide $A$ using $O(n)$ steps.
Model Dependence

- Number of steps to decide $A = \{a^kB^k \mid k \geq 0\}$ depends on the model.
  - Single tape TM: $O(n \log n)$
  - Multi-tape TM: $O(n)$

- Computability theory: model independent
- Complexity theory: model dependent
  - For “reasonable” deterministic models, dependence is low (polynomial).
  - So, we will focus on single tape (deterministic) TM as basic model for complexity.
Time Complexity Classes

- A TM runs in time $t(n)$ if $M$ always halts within $t(n)$ steps on all inputs of length $n$.
- $\text{TIME}(t(n)) = \{ B \mid \text{some deterministic single-tape TM decides } B \text{ in time } O(t(n)) \}$
**Theorem**: Every $t(n)$ time multitape TM has an equivalent $O(t^2(n))$ time single-tape TM.
Relationship Among Models

- Informal Definition: Two models of computation are **polynomially equivalent** if each can simulate the other with a polynomial overhead:

  So $t(n)$ time $\rightarrow t^k(n)$ time on the other model, for some $k$.

- All reasonable deterministic models are polynomially equivalent.
  - Single tape TMs
  - Multi-tape TMs
  - Multi-dimensional TMs
  - Cellular automata
The Class $P$

- $P$ is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine

\[ P = \bigcup_k \text{TIME}(n^k) \]

- $P$ is invariant for all models of computation that are polynomially equivalent.

- “$P$ roughly corresponds to the class of problems that are realistically solvable on a computer.”
The PATH Problem

- \( PATH=\{(G,s,t) \mid G \text{ is a directed graph with a path from } s \text{ to } t \} \)
- Is there a path from \( s \) to \( t \) in \( G \)?
$PATH \in P$

- $PATH=\{\langle G, s, t \rangle \mid G \text{ is a directed graph with a path from } s \text{ to } t \}$
The HAMPATH Problem

- $HAMPATH = \{(G, s, t) \mid G$ is a directed graph with a path from $s$ to $t$
  and the path goes through every node of $G$ $\}$
A Nondeterministic Algorithm for HAMPATH
Nondeterministic Complexity

- In a nondeterministic TM that is a decider, all computation paths halt on all inputs.
- \( \text{NTIME}(t(n)) = \{B \mid \text{some single-tape nondeterministic TM decides } B \text{ and runs in time } O(t(n)) \} \)

\[
NP = \bigcup_k \text{NTIME} \left( n^k \right)
\]

- NP is invariant for all reasonable nondeterministic models.
The COMPOSITES Problem

- \( \text{COMPOSITES} = \{ x | x = pq \text{ for integers } p, q > 1 \} \) \( \text{COMPOSITES} \in NP \)
For the algorithms for HAMPATH and COMPOSITES, there were exponentially many computation paths.

- For each path, the TM guesses a solution and checks (verifies) whether the solution is valid.

A **verifier** for a language $A$ is an algorithm $V$ where $A = \{ w \mid V \text{ accepts } <w,c> \text{ for some string } c \}$

- A verifier uses the extra information in $c$ to check that $w$ is a member of $A$.
- $c$ is called the **certificate** or the **proof**.

A **polynomial time verifier** runs in time polynomial in the length of $w$.

A language $A$ is **polynomially verifiable** if it has a polynomial time verifier.
Definition: NP is the class of languages that have polynomial time verifiers.

Theorem: A language has a polynomial time verifier if and only if it is decided by some nondeterministic polynomial time Turing machine.
**Theorem:** A language has a polynomial time verifier if and only if it is decided by some nondeterministic polynomial time Turing machine.
A Boolean formula $\phi$ has Boolean variables (True/False values) and the Boolean operations And ($\land$), Or ($\lor$), and Not ($\neg$).

$\phi$ is **satisfiable** if $\phi$ evaluates to $\text{TRUE}$ for some assignment to its variables.

$SAT = \{ \langle \phi \rangle | \phi \text{ is a satisfiable Boolean formula} \}$

**Cook-Levin Theorem:** $SAT \in P$ if and only if $P = NP$
Polynomial Time Reducibility

- A function $f : \Sigma^* \to \Sigma^*$ is a polynomial time computable function if there exists a polynomial time Turing machine $M$ that, on every input $w$, halts with just $f(w)$.

- Language $A$ is polynomial time mapping reducible (also called polynomially time reducible) to language $B$ (written $A \leq_p B$) if there is a polynomial time computable function $f$ where $w \in A$ if and only if $f(w) \in B$.

  The function $f$ is called a polynomial time reduction of $A$ to $B$. 
Proving Problems are in P

- **Theorem:** If \( A \leq_P B \) and \( B \in P \) then \( A \in P \)