Lecture 23:
Finish Randomized Complexity,
Begin Review
The Plan For This Week

Today: We’ll finish randomized complexity, and start reviewing the major topics

Thursday: When review is done... Ask Me Anything!

Ask questions in person, or post questions anonymously on piazza. I will answer them in class on Thursday!
Randomized / Probabilistic Complexity
A probabilistic TM $M$ is a nondeterministic TM where:

Each nondeterministic step is called a *coin flip*.

Each nondeterministic step has only two legal next moves *(heads or tails)*.

The probability that $M$ runs on a branch $b$ is:

$$\Pr [ b ] = 2^{-k}$$

where $k$ is the number of coin flips that occur on branch $b$. 
Definition. A probabilistic TM $M$ decides a language $A$ with error $\varepsilon$ if for all strings $w$,

$$w \in A \Rightarrow \Pr[ M \text{ accepts } w ] \geq 1 - \varepsilon$$

$$w \notin A \Rightarrow \Pr[ M \text{ doesn’t accept } w ] \geq 1 - \varepsilon$$

Theorem: A language $A$ is in NP if there is a nondeterministic polynomial time TM $M$ such that for all strings $w$:

$$w \in A \Rightarrow \Pr[ M \text{ accepts } w ] > 0$$

$$w \notin A \Rightarrow \Pr[ M \text{ accepts } w ] = 0$$
BPP = Bounded Probabilistic P

\[ \text{BPP} = \{ L \mid L \text{ is recognized by a probabilistic polynomial-time TM with error at most } \frac{1}{3} \} \]

Why \( \frac{1}{3} \)?

It doesn’t matter what error value we pick, as long as the error is smaller than \( \frac{1}{2} \).

When the error is smaller than \( \frac{1}{2} \), we can make it very small by repeatedly running the TM.
Checking Matrix Multiplication

CHECK = \{ (M_1, M_2, N) \mid M_1, M_2 \text{ and } N \text{ are matrices and } M_1 \cdot M_2 = N \}

If \( M_1 \) and \( M_2 \) are \( n \times n \) matrices, computing \( M_1 \cdot M_2 \) takes \( O(n^3) \) time normally, and \( O(n^{2.373}) \) time using very sophisticated methods.

Here is an \( O(n^2) \)-time randomized algorithm for CHECK:

Pick a 0-1 bit vector \( r \) at random, test if \( M_1 \cdot M_2 r = Nr \)

Claim: If \( M_1 \cdot M_2 = N \), then \( \Pr [M_1 \cdot M_2 r = Nr] = 1 \)
If \( M_1 \cdot M_2 \neq N \), then \( \Pr [M_1 \cdot M_2 r = Nr] \leq 1/2 \)

If we pick 20 random vectors and test them all, what is the probability of incorrect output? \( 1/2^{20} < 0.000001 \)
An arithmetic formula is like a Boolean formula, except it has $+, -, \text{ and } \ast$ instead of OR, NOT, AND.

$\text{ZERO-POLY} = \{ p \mid p \text{ is an arithmetic formula that is identically zero} \}$

Identically zero means: all coefficients are 0

Theorem: $\text{ZERO-POLY} \in \text{BPP}$

Algorithm A: Given polynomial $p$,
For all $i = 1, \ldots, k$, choose $r_i$ randomly from $\{1, \ldots, 3n^2\}$
If $p(r_1, \ldots, r_k) = 0$ then output zero
else output nonzero

Schwartz-Zippel $\Rightarrow$ This works!
Equivalence of Arithmetic Formulas

\[ \text{EQUIV-POLY} = \{ (p, q) \mid p \text{ and } q \text{ are arithmetic formulas computing the same polynomial} \} \]

Corollary: \( \text{EQUIV-POLY} \in \text{BPP} \)

There is a big contrast with Boolean formulas!

\[ \text{EQUIV} = \{ (\phi, \psi) \mid \phi \text{ and } \psi \text{ are Boolean formulas computing the same function} \} \]

We showed \( \text{EQUIV} \) is in \( \text{coNP} \). It’s also \( \text{coNP-complete} \)!

\( \text{TAUTOLOGY} \leq_p \text{EQUIV}: \) map \( \phi \) to \((\phi, T)\)
BPP = \{ L \mid L \text{ is recognized by a probabilistic polynomial-time TM with error at most } 1/3 \} 

BPP \subseteq NP \text{ is open}

BPP \subseteq \text{PSPACE} \text{ is known}

BPP \subseteq \text{NP}^{\text{NP}} \text{ and } BPP \subseteq \text{coNP}^{\text{NP}}, \text{ but } BPP \subseteq \text{P}^{\text{NP}} \text{ is still open!}

\text{NP} \subseteq \text{BPP} \text{ is open}
Is BPP = EXPTIME?

THIS IS AN OPEN QUESTION!?

It’s widely conjectured that P = BPP!

Certain lower bounds \(\implies P = BPP\)
Definition: A language $A$ is in $\text{RP}$ (Randomized P) if there is a nondeterministic polynomial time TM $M$ such that for all strings $x$:

- $x \not\in A \Rightarrow \Pr[M(x) \text{ accepts}] = 0$
- $x \in A \Rightarrow \Pr[M(x) \text{ accepts}] > 2/3$

$\text{NONZERO-POLY} = \{ \text{p} \mid \text{p is an arithmetic formula that is not identically zero} \}$

Theorem: $\text{NONZERO-POLY} \in \text{RP}$
(Our proof of $\text{ZERO-POLY}$ in $\text{BPP}$ shows this)
Is RP \subseteq NP?

Yes!

Being RP means that not only are there “nifty proofs” but in fact most strings are nifty proofs!
Is $\text{RP} \subseteq \text{BPP}$?

Yes!

RP has “one-sided error”
BPP has “two-sided error”
Review
Deterministic Finite Automata

transition: for every state and alphabet symbol

states

start state (q_0)

accept states (F)

states
Deterministic Computation

Non-Deterministic Computation

Are these equally powerful???
YES for finite automata
Regular Languages are closed under all of the following operations:

**Union:** \( A \cup B = \{ w | w \in A \text{ or } w \in B \} \)

**Intersection:** \( A \cap B = \{ w | w \in A \text{ and } w \in B \} \)

**Complement:** \( \overline{A} = \{ w \in \Sigma^* | w \notin A \} \)

**Reverse:** \( A^R = \{ w_1 \ldots w_k | w_k \ldots w_1 \in A \} \)

**Concatenation:** \( A \cdot B = \{ vw | v \in A \text{ and } w \in B \} \)

**Star:** \( A^* = \{ w_1 \ldots w_k | k \geq 0 \text{ and each } w_i \in A \} \)
DFA Minimization:

There is an **efficient algorithm** which, given any DFA $M$, will output the unique minimum-state DFA $M^*$ equivalent to $M$.

If this were true for more general models of computation, that would be an engineering breakthrough!!
(Would imply $P=NP$ and more)

Table-Filling Algorithm
to find “distinguishable” pairs of states
Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$

$x \equiv_L y$ iff for all $z \in \Sigma^*$, $xz \in L \iff yz \in L$

The Myhill-Nerode Theorem:
A language $L$ is regular if and only if
the number of equivalence classes of $\equiv_L$ is finite.

Regular = “easy”
Not Regular = “hard”
The Myhill-Nerode Theorem gives us a (universal) way to prove that a given language is not regular:

L is not regular
if and only if
there are infinitely many equiv. classes of \( \equiv_L \)

L is not regular
if and only if
There are infinitely many strings \( w_1, w_2, \ldots \) so that for all \( w_i \neq w_j \), \( w_i \) and \( w_j \) are distinguishable to \( L \):
there is a \( z \in \Sigma^* \) such that
exactly one of \( w_i z \) and \( w_j z \) is in \( L \)
Streaming Algorithms

Have three components

Initialize:
<variables and their assignments>

When next symbol seen is $\sigma$:
<pseudocode using $\sigma$ and vars>

When stream stops (end of string):
<accept/reject condition on vars>
(or: <pseudocode for output>)

Algorithm A computes $L \subseteq \Sigma^*$ if
A accepts the strings in L, rejects strings not in L
L = \{x \mid x \text{ has odd number of } 1\text{’s}\}

Has streaming algorithms using \(O(1)\) space (that is, it has a DFA)

“very easy”

L = \{x \mid x \text{ has more } 1\text{’s than } 0\text{’s}\}

Has streaming algorithms using \(O(\log n)\) space, no streaming algorithm uses much less

“easy”

L = \{x \mid x \text{ is a palindrome}\}

Has streaming algorithms using \(O(n)\) space, no streaming algorithm uses much less

“hard”
Streaming Lower Bounds via DFAs

For any $L \subseteq \Sigma^*$ define $L_n = \{x \in L \mid |x| \leq n\}$

**Theorem:** Suppose $L'$ is such that for all $n$, every DFA $M$ for $L'_n$ requires at least $Q(n) := 2^{S(n)+1}$ states. Then every streaming algorithm for $L'$ requires at least $\log_2(Q(n))$ space.

State lower bounds for DFAs

$\rightarrow$ Space lower bounds for streaming algorithms
L is not regular if and only if there are infinitely many strings $w_1, w_2, \ldots$ so that for all $i \neq j$, there's a string $z$ such that exactly one of $w_iz$ and $w_jz$ is in $L$.

In fact, Myhill-Nerode shows that the size of a distinguishing set for $L$ is a lower bound on the number of states in a DFA for $L$.

In other words, if $S$ is a distinguishing set for $L$, then any DFA for $L$ must have at least $|S|$ states.

We can use this fact to prove lower bounds on streaming algorithms!
Communication Complexity

A theoretical model of distributed computing

• **Function** $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$
  - Two inputs, $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$
  - We assume $|x| = |y| = n$. Think of $n$ as HUGE

• **Two computers:** Alice and Bob
  - Alice *only* knows $x$, Bob *only* knows $y$

• **Goal:** Compute $f(x, y)$ by communicating as few bits as possible between Alice and Bob

*We do not count computation cost.* We *only* care about the number of bits communicated.
Connection to Streaming and DFAs

Let $L \subseteq \{0,1\}^*$

Def. $f_L: \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$

for $x, y$ with $|x| = |y|$ as:

$$f_L(x, y) = 1 \iff xy \in L$$

Theorem: If $L$ has a streaming algorithm using $\leq s$ space, then $cc(f_L)$ is at most $2s + 1$.

Lower bounds on $cc$ ➔ Lower bounds on streaming
Connection to Streaming and DFAs

Let $L \subseteq \{0,1\}^*$

Def. $f_L : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$

for $x, y$ with $|x| = |y|$ as:

$$f_L(x, y) = 1 \iff xy \in L$$

Examples:

$L = \{ x \mid x \text{ has an odd number of } 1s\}$

$\Rightarrow f_L(x, y) = \text{PARITY}(x,y) \text{ has } \Theta(1) \text{ comm. compl.}$

$L = \{ x \mid x \text{ has more } 1\text{s than } 0\text{s} \}$

$\Rightarrow f_L(x, y) = \text{MAJORITY}(x,y) \text{ has } \Theta(\log n) \text{ comm. compl.}$

$L = \{ xx \mid x \in \{0,1\}^* \}$

$\Rightarrow f_L(x, y) = \text{EQUALS}(x,y) \text{ has } \Theta(n) \text{ comm. compl.}$
Theorem: \( L \) is decidable iff both \( L \) and \( \neg L \) are recognizable

\( \begin{align*}
\text{L is decidable} & \quad \text{“easy”} \\
\text{L is recognizable} & \quad \text{“not so easy”}
\end{align*} \)
Theorem: L is recognizable \iff There is a TM V halting on all inputs such that
\[ L = \{ x \mid \exists y \in \Sigma^* [V(x, y) \text{ accepts}] \} \]
Are these equally powerful???
NO for Turing Machines
Decidable = Recognizable ∩ Co-recognizable

Church-Turing Thesis
\[ A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \]

**Thm.** \( A_{TM} \) is undecidable. (proof by contradiction)

Suppose \( H \) is a machine that decides \( A_{TM} \)

\[
H( (M,w) ) = \begin{cases} 
\text{Accept} & \text{if } M \text{ accepts } w \\
\text{Reject} & \text{if } M \text{ does not accept } w 
\end{cases}
\]

Define a new TM \( D \) with the following spec:

\( D(M) \): Run \( H \) on \( (M,M) \) and output the opposite of \( H \)

\[
D( D ) = \begin{cases} 
\text{Reject} & \text{if } D \text{ accepts } D \\
\text{Accept} & \text{if } D \text{ does not accept } D 
\end{cases}
\]

Contradiction!
Mapping Reductions

$f : \Sigma^* \rightarrow \Sigma^*$ is a computable function if there is a Turing machine $M$ that halts with just $f(w)$ written on its tape, for every input $w$.

A language $A$ is *mapping reducible* to language $B$, written as $A \leq_m B$, if there is a computable $f : \Sigma^* \rightarrow \Sigma^*$ such that for every $w \in \Sigma^*$,

$$w \in A \iff f(w) \in B$$

$f$ is called a mapping reduction (or many-one reduction) from $A$ to $B$. 
Rice’s Theorem: *Program Analysis is Hard*

Let \( P : \{\text{Turing Machines}\} \rightarrow \{0,1\} \).
(Think of 0=false, 1=true) Suppose \( P \) satisfies:

1. **(Nontrivial)** There are TMs \( M_1 \) and \( M_0 \) where \( P(M_1) = 1 \) and \( P(M_0) = 0 \)

2. **(Semantic)** For all TMs \( M \) and \( M' \),
   If \( L(M) = L(M') \) then \( P(M) = P(M') \)

Then, \( \{M \mid P(M) = 1\} \) is undecidable.
In other words, function \( P \) is undecidable.

A Huge Hammer for Undecidability!
Recursion Thm: For every computable t, there is a computable r such that $r(w) = t(R,w)$ where $R$ is a description of a TM computing r

Moral: Suppose we can design a TM T of the form “On input $(x,w)$, do bla bla with $x$, do bla bla bla bla with $w$, etc. etc.”

We can always find a TM R with the behavior: “On input $w$, do bla bla bla with code of $R$, do bla bla bla bla with $w$, etc. etc.”

We can use the operation: “Obtain your own description” in Turing machine pseudocode!
Limitations on Mathematics

For every consistent and interesting $F$,

**Theorem 1.** (Gödel 1931) $F$ is incomplete:
There are mathematical statements in $F$ that are true but cannot be proved in $F$.

**Theorem 2.** (Gödel 1931) The consistency of $F$ cannot be proved in $F$.

**Theorem 3.** (Church-Turing 1936) The problem of checking whether a given statement in $F$ has a proof is undecidable.
Kolmogorov Complexity

Def: The *shortest description of* $x$, denoted as $d(x)$, is the lexicographically shortest string $<M,w>$ such that $M$ on $w$ halts with only $x$ on its tape.

Def: The *Kolmogorov complexity of* $x$, $K(x)$, is $|d(x)|$.

COMPRESS = $\{(x,c) \mid K(x) \leq c\}$ is undecidable (but is recognizable)
-Time Complexity

**Definition:**

\[
\text{TIME}(t(n)) = \{ L' \mid \text{there is a Turing machine } M \text{ with time complexity } O(t(n)) \text{ so that } L' = L(M) \} \\
= \{ L' \mid L' \text{ is a language decided by a Turing machine with } \leq c t(n) + c \text{ running time} \}
\]

**The Time Hierarchy Theorem**

**Intuition:** The more computing time you have, the more problems you can solve.

**Theorem:** For all “reasonable” \( f, g : \mathbb{N} \rightarrow \mathbb{N} \) where for all \( n \), \( g(n) > n^2 f(n)^2 \), \( \text{TIME}(f(n)) \not\subseteq \text{TIME}(g(n)) \)
Deterministic Poly-Time Computation

Non-Deterministic Poly-Time Computation

"easy"
accept or reject

"probably not easy"
accept

Are these equally powerful???
P = NP ????
**Theorem:**  $L \in \text{NP} \iff$ There is a constant $k$ and polynomial-time TM $V$ such that

$$L = \{ x \mid \exists y \in \Sigma^* \ [ |y| \leq |x|^k \text{ and } V(x,y) \text{ accepts} \} \}$$

**Moral:** A language $L$ is in $\text{NP}$ if and only if there are polynomial-length ("nifty") proofs for membership in $L$

**Theorem:**  $L$ is recognizable $\iff$ There is a TM $V$ that halts on all inputs such that

$$L = \{ x \mid \exists y \in \Sigma^* \ [ V(x,y) \text{ accepts} ] \}$$
Definition: A language $B$ is NP-complete if:

1. $B \in \text{NP}$

2. Every $A$ in NP is poly-time reducible to $B$
   That is, $A \leq_p B$
   When this is true, we say “$B$ is NP-hard”

NP-complete problems: “probably hard”
3SAT, SAT, CLIQUE, IS, VC, SUBSET-SUM, KNAPSACK, PARTITION, BIN-PACKING, ...
Definition: \( \text{coNP} = \{ L \mid \neg L \in \text{NP} \} \)

*What does a coNP problem \( L \) look like?*

The instances *not* in \( L \) have *nifty proofs*. Recall we can write any NP problem \( L \) in the form:

\[
L = \{ x \mid \exists y \text{ of poly}(|x|) \text{ length so that } V(x,y) \text{ accepts} \}
\]

\[
\neg L = \{ x \mid \neg \exists y \text{ of poly}(|x|) \text{ length so that } V(x,y) \text{ accepts} \}
\]

\[
= \{ x \mid \forall y \text{ of poly}(|x|) \text{ length, } V(x,y) \text{ rejects} \}
\]

Instead of using an "existentially guessing" (nondeterministic) machine, we can define a "universally verifying" machine!
Complexity Classes With Oracles

Let $B$ be a language.

$P^B = \{ L \mid L \text{ can be decided by some } \text{polynomial-time TM } \text{ with an oracle for } B \}$

$P^{NP} = \text{the class of languages decidable by } \text{some polynomial-time oracle TM with an oracle for some } B \text{ in } NP$

$NP^{NP} = \text{the class of languages decidable by } \text{some nondeterministic polynomial-time oracle TM with an oracle for some } B \text{ in } NP$
NP-complete problems:

SAT, 3SAT, CLIQUE, VC, SUBSET-SUM, ...

coNP-complete problems:

UNSAT, TAUTOLOGY, NOHAMPATH, ...

PSPACE-complete problems:

TQBF, GG

There are also $NP^{NP}$-complete and $coNP^{NP}$ problems

(but you don’t need to know them for the final!)
Factors: coNP, SAT, FACTORING, MIN-FORMULA.

Classes:
- P
- NP
- coNP
- PNP
- NPNP
- EXPTIME
- coNPNP

Time Hierarchy
Poly-time Reductions
NP Completeness

Oracles:
- PNP
- NPNP
- coNPNP
What’s next?

A few possibilities...

6.046 – Design and Analysis of Algorithms
6.841/18.405 – Advanced Complexity Theory
18.408 – Topics in Theoretical Computer Science
18.416 – Randomized Algorithms
6.875 – Cryptography and Cryptanalysis

Many more! There’s a big group at MIT!
Ask Me Anything!