Lecture 24

Last time:
- Probabilistic computation
- The class BPP
- Branching programs
- Arithmetization
- Started showing $EQ_{ROBP} \in \text{BPP}$

Today:
- Finish $EQ_{ROBP} \in \text{BPP}$

Posted: Sample problems for the final exam (to be held on Thursday, December 17).
Review: Probabilistic TMs and BPP

**Defn:** A probabilistic Turing machine (PTM) is a variant of a NTM where each computation step has 1 or 2 possible choices.

**Defn:** For $\epsilon \geq 0$ say PTM $M$ decides language $A$ with error probability $\epsilon$ if for every $w$, $\Pr[M$ gives the wrong answer about $w \in A] \leq \epsilon$.

**Defn:** $\text{BPP} = \{A |$ some poly-time PTM decides $A$ with error $\epsilon = \frac{1}{3}\}$

**Amplification lemma:** $2^{-\text{poly}(n)}$

**Check-in 24.1**
Actually using a probabilistic algorithm presupposes a source of randomness. Can we use a standard pseudo-random number generator (PRG) as the source?

(a) Yes, but the result isn’t guaranteed.
(b) Yes, but it will run in exponential time.
(c) No, a TM cannot implement a PRG.
(d) No, because that would show $P = \text{BPP}$. 

---

**Check-in 24.1**

---

**Check-in 24.1**

---
Review: Branching Programs

Defn: A branching program (BP) is a directed, acyclic (no cycles) graph that has
1. Query nodes labeled $x_i$ and having two outgoing edges labeled 0 and 1.
2. Two output nodes labeled 0 and 1 and having no outgoing edges.
3. A designated start node.

Theorem: $EQ_{BP}$ is coNP-complete (on pset 6)

Defn: A BP is read-once if it never queries a variable more than once on any path from the start node to an output.

Defn: $EQ_{ROBP} = \{(B_1, B_2) | B_1$ and $B_2$ are equivalent read-once BPs$\}$

Theorem: $EQ_{ROBP} \in BPP$
Proof idea: Run $B_1$ and $B_2$ on a randomly selected non-Boolean input and accept if get same output.
Method: Use arithmetization (simulating $\land$ and $\lor$ with $+$ and $\times$) to define BP operation on non-Boolean inputs.
Boolean Labeling

Alternative way to view BP computation

Show by example: Input is $x_1 = 0$, $x_2 = 1$, $x_3 = 1$
The BP follows its execution path.
Label all nodes and edges on the execution path with 1
and off the execution path with 0.
Output the label of the output node 1.

Obtain the labeling inductively by using these rules:

Label outgoing edges from nodes
Label nodes from incoming edges
**Method:** Simulate $\land$ and $\lor$ with $+$ and $\times$.

- $a \land b \rightarrow a \times b = ab$
- $\overline{a} \rightarrow (1 - a)$
- $a \lor b \rightarrow a + b - ab$

Replace Boolean labeling with arithmetical labeling

Inductive rules:
- Start node labeled 1

Simulate $\lor$ with $+$ because the BP is acyclic.

The execution path can enter a node at most one time.
Non-Boolean Labeling

Use the arithmetized interpretation of the BP’s computation to define its operation on non-Boolean inputs.

Example: $x_1 = 2$, $x_2 = 3$

Recall labeling rules:

Algorithm sketch for $EQ_{ROBP}$: “On input $\langle B_1, B_2 \rangle$
1. Pick a random non-Boolean input assignment.
2. Evaluate $B_1$ and $B_2$ on that assignment.
3. If $B_1$ and $B_2$ disagree then reject.
   If they agree then accept.”

More details and correctness proof to come.
First some algebra...
Let \( p(x) = a_0 x^d + a_1 x^{d-1} + a_2 x^{d-2} + \cdots + a_d \) be a polynomial. If \( z \) is some constant and \( p(z) = 0 \) call \( z \) a root of \( p \).

**Polynomial Lemma:** If \( p(x) \neq 0 \) is polynomial of degree \( \leq d \) then \( p \) has \( \leq d \) roots.
Proof by induction (see text).

**Corollary 1:** If \( p_1(x) \) and \( p_2(x) \) are both degree \( \leq d \) and \( p_1 \neq p_2 \) then \( p_1(z) = p_2(z) \) for \( \leq d \) values \( z \).
Proof: Let \( p = p_1 - p_2 \).

Above holds for any field \( \mathbb{F} \) (a field is a set with + and \( \times \) operations that have typical properties).
We will use a finite field \( \mathbb{F}_q \) with \( q \) elements where \( q \) is prime and +, \( \times \) operate mod \( q \).

**Corollary 2:** If \( p(x) \neq 0 \) has degree \( \leq d \) and we pick a random \( r \in \mathbb{F}_q \), then \( \Pr[ p(r) = 0 ] \leq d/q \).
Proof: There are at most \( d \) roots out of \( q \) possibilities.

**Theorem** (Schwartz-Zippel): If \( p(x_1, \ldots, x_m) \neq 0 \) has degree \( \leq d \) in each \( x_i \) and we pick random \( r_1, \ldots, r_m \in \mathbb{F}_q \) then \( \Pr[ p(r_1, \ldots, r_m) = 0 ] \leq md/q \)
Proof by induction (see text).
Symbolic Execution

Leave the $x_i$ as variables and obtain an expression in the $x_i$ for the output of the BP.

Recall labeling rules:

- $a (1 - x_i)$
- $a x_i$

Exponents $\leq 1$ due to "read-once"
Assume read exactly once so that for each $i$, $(x_i)$ or $(1 - x_i)$ appears in every row

Form of output:

$$
= (1 - x_1)(x_2)^\times (1 - x_3)(x_4) \cdots (1 - x_m) + (x_1)(x_2)(x_3)(1 - x_4) \cdots (x_m) + (x_1)(1 - x_2)(1 - x_3)(x_4) \cdots (x_m) + \cdots + (x_1)(x_2)(1 - x_3)(x_4) \cdots (x_m)
$$

Corresponds to the True rows in the truth table of the Boolean function.
Algorithm for $E_{\text{ROBP}} = \text{“On input } \langle B_1, B_2 \rangle \text{ [on variables } x_1, \ldots, x_m \rangle$

1. Find a prime $q \geq 3m$.
2. Pick a random non-Boolean input assignment $r = r_1, \ldots, r_m$ where each $r_i \in \mathbb{F}_q$.
3. Evaluate $B_1$ and $B_2$ on $r$ by using arithmetization.
4. If $B_1$ and $B_2$ agree on $r$ then accept.
   If they disagree then reject.”

Claim: (1) $B_1 \equiv B_2 \rightarrow \Pr [ p_1 (r) = p_2 (r) ] = 1$
(2) $B_1 \not\equiv B_2 \rightarrow \Pr [ p_1 (r) = p_2 (r) ] \leq \frac{1}{3}$

Proof (1): If $B_1 \equiv B_2$ then they agree on all Boolean inputs. Thus their functions have the same truth table. Thus their associated polynomials $p_1$ and $p_2$ are identical. Thus $p_1$ and $p_2$ always agree (even on non-Boolean inputs).

Proof (2): If $B_1 \not\equiv B_2$ then $p_1 \neq p_2$ so $p = p_1 - p_2 \neq 0$. From Schwartz-Zippel, $\Pr [ p_1 (r) = p_2 (r) ] \leq \frac{d m}{q} \leq \frac{m}{3m} = \frac{1}{3}$. (Note that $d = 1$.)

Check-in 24.2
If the BPs were not read-once, the polynomials might have exponents $\geq 1$. Where would the proof fail?
(a) $B_1 \equiv B_2$ implies they agree on all Boolean inputs
(b) Agreeing on all Boolean inputs implies $p_1 = p_2$
(c) Having $p_1 = p_2$ implies $p_1$ and $p_2$ always agree

$p_1$ and $p_2$ each have the form:
$$
(1 - x_1) (x_2) (1 - x_3) (x_4) \ldots (1 - x_m) + (x_1) (x_2) (x_3) (1 - x_4) \ldots (x_m) + (x_1) (1 - x_2) (1 - x_3) (x_4) \ldots (x_m) + \ldots + (x_1) (x_2) (1 - x_3) (x_4) \ldots (x_m)
$$
Algorithm for $EQ_{ROBP}$ = “On input $\langle B_1, B_2 \rangle$ [on variables $x_1, \ldots, x_m$]
1. Find a prime $q \geq 3m$.
2. Pick a random non-Boolean input assignment $r = r_1, \ldots, r_m$ where each $r_i \in \mathbb{F}_q$.
3. Evaluate $B_1$ and $B_2$ on $r$ by using arithmetization.
4. If $B_1$ and $B_2$ agree on $r$ then accept.
   If they disagree then reject.”

Claim: (1) $B_1 \equiv B_2 \rightarrow \Pr[p_1(r) = p_2(r)] = 1$
(2) $B_1 \not\equiv B_2 \rightarrow \Pr[p_1(r) = p_2(r)] \leq 1/3$

Proof (1): If $B_1 \equiv B_2$ then they agree on all Boolean inputs.
Thus their functions have the same truth table.
Thus their associated polynomials $p_1$ and $p_2$ are identical.
Thus $p_1$ and $p_2$ always agree (even on non-Boolean inputs).

Proof (2): If $B_1 \not\equiv B_2$ then $p_1 \neq p_2$ so $p = p_1 - p_2 \neq 0$.
From Schwartz-Zippel, $\Pr[p_1(r) = p_2(r)] \leq \frac{dm}{q} \leq \frac{m}{3m} = \frac{1}{3}$.
(Note that $d = 1$.)

Check-in 24.3
If $p_1$ and $p_2$ were exponentially large expressions, would that be a problem for the time complexity?
(a) Yes, but luckily they are polynomial in size.
(b) No, because we can evaluate them without writing them down.

$p_1$ and $p_2$ each have the form:

$\begin{align*}
(1 - x_1) & \quad (x_2) & \quad (1 - x_3) & \quad (x_4) & \quad \cdots & \quad (1 - x_m) \\
+ (x_1) & \quad (x_2) & \quad (x_3) & \quad (1 - x_4) & \quad \cdots & \quad (x_m) \\
+ (x_1) & \quad (1 - x_2) & \quad (1 - x_3) & \quad (x_4) & \quad \cdots & \quad (x_m) \\
\vdots & & & & & \\
+ (x_1) & \quad (x_2) & \quad (1 - x_3) & \quad (x_4) & \quad \cdots & \quad (x_m)
\end{align*}$

Check-in 24.3
Quick review of today

1. Simulated Read-once Branching Programs by polynomials
2. Gave probabilistic polynomial equality testing method
3. Showed $EQ_{ROBP} \in \text{BPP}$