Last time:
- $EQ_{\text{REX}^1}$ is EXPSPACE-complete
- Thus $EQ_{\text{REX}^1} \not\in \text{PSPACE}$
- Oracles and P versus NP

Today:
- Probabilistic computation
- The class BPP
- Branching programs
Probabilistic TMs

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

![Computation tree for $M$ on $w$](attachment:image.png)

- **Deterministic step**
- **Coin flip step** - each choice has 50% probability

\[
\Pr[\text{branch } b] = 2^{-k} \quad \text{where } b \text{ has } k \text{ coin flips}
\]

\[
\Pr[ M \text{ accepts } w ] = \sum_{b \text{ accepts}} \Pr[\text{branch } b]
\]

\[
\Pr[ M \text{ rejects } w ] = 1 - \Pr[ M \text{ accepts } w]
\]

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

*i.e.,* $w \in A \rightarrow \Pr[ M \text{ rejects } w ] \leq \epsilon$

$w \notin A \rightarrow \Pr[ M \text{ accepts } w ] \leq \epsilon$. 
The Class BPP

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

Amplification lemma: If \( M_1 \) is a poly-time PTM with error \( \epsilon_1 < \frac{1}{2} \) then, for any \( 0 < \epsilon_2 < \frac{1}{2} \), there is an equivalent poly-time PTM \( M_2 \) with error \( \epsilon_2 \). Can strengthen to make \( \epsilon_2 < 2^{-\text{poly}(n)} \).

Proof idea: \( M_2 = \) “On input \( w \)
1. Run \( M_1 \) on \( w \) for \( k \) times and output the majority response.”

Details: Calculation to obtain \( k \) and the improved error probability.

Significance: Can make the error probability so small it is negligible.
NP and BPP

Computation trees for $M$ on $w$

$w \in A$
- $\geq 1$ accepting

$w \notin A$
- all rejecting

$BPP$
- Many accepting
- Few rejecting

$\geq 1$ accepting

Check-in 23.1
Which of these are known to be true? Check all that apply.
(a) BPP is closed under union.
(b) BPP is closed under complement.
(c) $P \subseteq BPP$
(d) $BPP \subseteq PSPACE$
**Example: 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*.

BP $B$ with query nodes $x_1, \ldots, x_m$ describes a Boolean function $f: \{0,1\}^m \rightarrow \{0,1\}$:
Follow the path designated by the query nodes’ outgoing edges from the start node until reach an output node.

**Example:** For $x_1 = 1$, $x_2 = 0$, $x_3 = 1$ BPs are *equivalent* if they describe the same Boolean function.

**Defn:** $EQ_{BP} = \{ (B_1, B_2) | B_1$ and $B_2$ are equivalent BPs (written $B_1 \equiv B_2$) $\}$

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

$EQ_{BP} \in BPP$?

Instead, consider a restricted problem.
**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:** \( EQ_{\text{ROBP}} = \{ (B_1, B_2) | B_1 \) and \( B_2 \) are equivalent read-once BPs\}

**Theorem:** \( EQ_{\text{ROBP}} \in \text{BPP} \)

---

**Check-in 23.2**

Assuming (as we will show) that \( EQ_{\text{ROBP}} \in \text{BPP} \), can we use that to show \( EQ_{\text{BP}} \in \text{BPP} \) by converting branching programs to read-once branching programs?

(a) Yes, there is no need to re-read inputs.

(b) No, we cannot do that conversion in general.

(c) No, the conversion is possible but not in polynomial-time.
Theorem: $\mathcal{EQ}_{\text{ROBP}} \in \text{BPP}$

Proof attempt: Let $M = \text{"On input } \langle B_1, B_2 \rangle$

1. Pick $k$ random input assignments and evaluate $B_1$ and $B_2$ on each one.
2. If $B_1$ and $B_2$ ever disagree on those assignments then reject.
   If they always agree on those assignments then accept.”

What $k$ to chose?

If $B_1 \equiv B_2$ then they always agree so $\Pr[M \text{ accepts } \langle B_1, B_2 \rangle] = 1$
If $B_1 \not\equiv B_2$ then want $\Pr[M \text{ accepts } \langle B_1, B_2 \rangle] \leq \frac{1}{3}$
   so want $\Pr[M \text{ rejects } \langle B_1, B_2 \rangle] \geq \frac{2}{3}$.

But $B_1$ and $B_2$ may disagree rarely, say in 1 of the $2^m$ possible assignments.
That would require exponentially many samples to have a good chance of finding a disagreeing assignment and thus would require $k > \left(\frac{2}{3}\right)2^m$.
But then this algorithm would use exponential time.

Try a different idea: Run $B_1$ and $B_2$ 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.

Obtain the labeling inductively by using these rules:

Label edges from nodes
Label nodes from incoming edges
Arithmetization Method

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 $1$

Works because the BP is acyclic.
The execution path can enter a node at most one time.
Non-Boolean Inputs

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:

1. Pick a random non-Boolean input assignment.
2. Evaluate \( x_1 \) and \( x_2 \) on that assignment.
3. If \( x_1 \) and \( x_2 \) disagree then reject. If they agree then accept.

Check-in 23.3

What is the output for this branching program using the arithmetized interpretation if \( x_1 = 1, \ x_2 = y \) ?

(a) \((1 - y)\)
(b) \((y + 1)\)
(c) \(y\)
Quick review of today

1. Defined probabilistic Turing machines
2. Defined the class BPP
3. Sketch the amplification lemma
4. Introduced branching programs and read-once branching programs
5. Started the proof that $E Q_{\text{ROBP}} \in \text{BPP}$
6. Introduced the arithmetization method