# Lecture 8 - Randomized Algorithms, Probabilistic TMs

**Rafael Oliveira**
rafael.oliveira.teaching@gmail.com
University of Waterloo

CS 860 - Graduate Complexity Theory
Fall 2022

# Overview

- Randomized Algorithms & Probabilistic TMs
- Relation to other classes

# Modelling Randomness

Two equivalent perspectives. Let $p : \mathbb{N} \to \mathbb{N}$ be a function.

### Definition 1 ("online" Probabilistic Turing Machines)

A probabilistic $p$-time Turing Machine (PTM) $M$ is a TM with two transition functions $\delta_0, \delta_1$ such that:

- at each step, it chooses with probability $1/2$ to apply $\delta_0$ otherwise $\delta_1$
- it always halts in $p(|x|)$ steps

# Modelling Randomness

Two equivalent perspectives. Let $p : \mathbb{N} \to \mathbb{N}$ be a function.

### Definition 1 (“online” Probabilistic Turing Machines)

A probabilistic $p$-time Turing Machine (PTM) $M$ is a TM with two transition functions $\delta_0, \delta_1$ such that:

- at each step, it chooses with probability $1/2$ to apply $\delta_0$ otherwise $\delta_1$
- it always halts in $p(|x|)$ steps

### Definition 2 (“offline” Probabilistic Turing Machines)

A probabilistic $p$-time Turing Machine (PTM) $M$ is a TM with two transition functions $\delta_0, \delta_1$ and two tapes: an input tape and a random-input tape s.t.

- size of random-input tape is $p(|x|)$
- $M(x, r)$ halts in $p(|x|)$ steps

# No time bound

## Definition 3 (Randomized Turing Machines)

A randomized Turing Machine $M$ is a TM with two transition functions $\delta_0, \delta_1$ such that:

- at each step, it chooses with probability $1/2$ to apply $\delta_0$ otherwise $\delta_1$

# No time bound

## Definition 3 (Randomized Turing Machines)

A randomized Turing Machine $M$ is a TM with two transition functions $\delta_0, \delta_1$ such that:

- at each step, it chooses with probability $1/2$ to apply $\delta_0$ otherwise $\delta_1$

Modelling running time:

- Given a randomized TM $M$ and input $x$, let $T(M, x)$ be random variable accounting for running time of $M$ on input $x$.
- We say that $M$ has expected running time $t(n)$ if

$$\mathbb{E}[t(M, x)] \leq t(|x|)$$

for all $x \in \{0, 1\}^*$.

# Randomized Languages

## Definition 4 (BPTIME)

Given function $t : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$ we say that a PTM $M$ decides $L$ in time $t(n)$ if for every $x \in \{0,1\}^n$:

- $M$ halts in $t(n)$ steps (regardless of random choices)
- $\Pr_r[M(x,r) = L(x)] \geq 2/3$

# Randomized Languages

## Definition 4 (BPTIME)

Given function $t : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$ we say that a PTM $M$ decides $L$ in time $t(n)$ if for every $x \in \{0,1\}^n$:

- $M$ halts in $t(n)$ steps (regardless of random choices)
- $\Pr_r[M(x,r) = L(x)] \geq 2/3$

Important that success probability is constant away from $1/2$.

# Randomized Languages

## Definition 4 (BPTIME)

Given function $t : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$ we say that a PTM $M$ decides $L$ in time $t(n)$ if for every $x \in \{0,1\}^n$:

- $M$ halts in $t(n)$ steps (regardless of random choices)
- $\Pr_r[M(x,r) = L(x)] \geq 2/3$

Important that success probability is constant away from $1/2$.

## Definition 5 (BPP)

The class BPP (bounded-error probabilistic polynomial-time) is defined as

$$\mathsf{BPP} := \bigcup_{c \in \mathbb{N}} \mathsf{BPTIME}(O(n^c))$$

# One-sided error

## Definition 6 (RP)

$L \subseteq \{0,1\}^*$ is in RP if there is a poly-time PTM $M$ such that

$$x \in L \Rightarrow \Pr_r[M(x,r) = 1] \geq 1/2$$

$$x \notin L \Rightarrow \Pr_r[M(x,r) = 1] = 0.$$

# One-sided error

## Definition 6 (RP)

$L \subseteq \{0,1\}^*$ is in RP if there is a poly-time PTM $M$ such that

$$x \in L \Rightarrow \Pr_r[M(x,r) = 1] \geq 1/2$$

$$x \notin L \Rightarrow \Pr_r[M(x,r) = 1] = 0.$$

## Definition 7 (coRP)

$L \subseteq \{0,1\}^*$ is in coRP if there is a poly-time PTM $M$ such that

$$x \in L \Rightarrow \Pr_r[M(x,r) = 1] = 1$$

$$x \notin L \Rightarrow \Pr_r[M(x,r) = 1] \leq 1/2.$$

# Zero error

## Definition 8 (ZPP)

$L \subseteq \{0,1\}^*$ is in ZPP if there is a poly-time PTM $M$ whose output can be $0, 1, ?$ such that

$$\forall x \in \{0,1\}^* \;\Rightarrow\; \Pr_r[M(x,r) = ?] \leq 1/2$$

$$\forall x, r \text{ s.t. } M(x,r) \neq ? \;\Rightarrow\; M(x,r) = L(x).$$

# Zero error

## Definition 8 (ZPP)

$L \subseteq \{0,1\}^*$ is in ZPP if there is a poly-time PTM $M$ whose output can be $0, 1, ?$ such that

$$\forall x \in \{0,1\}^* \;\Rightarrow\; \Pr_r[M(x,r) = ?] \leq 1/2$$

$$\forall x, r \text{ s.t. } M(x,r) \neq ? \;\Rightarrow\; M(x,r) = L(x).$$

## Proposition 9

*ZPP is the class of languages which have an expected poly-time randomized algorithm which always gives the right answer.*

# Equivalent definition of ZPP

- $L \in \mathrm{ZPP} \Rightarrow$ exists randomized expected poly-time algorithm which always gives the right answer

# Equivalent definition of ZPP

- $L \in$ ZPP $\Rightarrow$ exists randomized expected poly-time algorithm which always gives the right answer
- Given ZPP algorithm $M$ which runs in time $t(n)$, let $A$ be the following algorithm:
  1. On input $x$ and random input $r \in \{0,1\}^{t(|x|)}$, run $M(x,r)$
  2. If the output is ? go back to step 1

# Equivalent definition of ZPP

- $L \in \text{ZPP} \Rightarrow$ exists randomized expected poly-time algorithm which always gives the right answer
- Given ZPP algorithm $M$ which runs in time $t(n)$, let $A$ be the following algorithm:
  1. On input $x$ and random input $r \in \{0,1\}^{t(|x|)}$, run $M(x,r)$
  2. If the output is ? go back to step 1
- Running time:
  - know that $\Pr_r[M(x,r) = ?] \leq 1/2$
  - Hence
$$\mathbb{E}[t_A(x)] \leq \sum_{k \geq 1} \frac{1}{2^k} \cdot k \cdot t(|x|) = O(t(|x|))$$

# Equivalent definition of ZPP

- $L$ decided by randomized expected poly-time $(t : \mathbb{N} \to \mathbb{N})$ algorithm $A$ which always gives the right answer

# Equivalent definition of ZPP

- $L$ decided by randomized expected poly-time ($t : \mathbb{N} \to \mathbb{N}$) algorithm $A$ which always gives the right answer
- Let $M$ be the following algorithm:
  1. On input $x$, run $A(x)$ for $2 \cdot t(|x|)$ steps
  2. If $A$ has not halted, output ?

# Equivalent definition of ZPP

- $L$ decided by randomized expected poly-time ($t : \mathbb{N} \to \mathbb{N}$) algorithm $A$ which always gives the right answer
- Let $M$ be the following algorithm:
  1. On input $x$, run $A(x)$ for $2 \cdot t(|x|)$ steps
  2. If $A$ has not halted, output ?
- Running time clearly $2 \cdot t(|x|)$

# Equivalent definition of ZPP

- $L$ decided by randomized expected poly-time ($t : \mathbb{N} \to \mathbb{N}$) algorithm $A$ which always gives the right answer
- Let $M$ be the following algorithm:
  1. On input $x$, run $A(x)$ for $2 \cdot t(|x|)$ steps
  2. If $A$ has not halted, output ?
- Running time clearly $2 \cdot t(|x|)$
- Since expected time is $t(|x|)$

$$\Pr[M(x) = ?] = \Pr[A \text{ doesn't halt in } 2 \cdot t(n) \text{ steps}] \leq 1/2$$

# Randomized log-space

## Definition 10 (BPL)

BPL is the class of languages $L \subseteq \{0,1\}$ for which there is a $O(\log n)$ space PTM $M$ such that

$$\Pr[M(x) = L(x)] \geq 2/3, \quad \forall x \in \{0,1\}^*$$

# Randomized log-space

## Definition 10 (BPL)

BPL is the class of languages $L \subseteq \{0,1\}$ for which there is a $O(\log n)$ space PTM $M$ such that

$$\Pr[M(x) = L(x)] \geq 2/3, \quad \forall x \in \{0,1\}^*$$

## Definition 11 (RL)

RL is the class of languages $L \subseteq \{0,1\}$ for which there is a $O(\log n)$ space PTM $M$ such that

$$x \in L \Rightarrow \Pr[M(x) = 1] \geq 1/2$$

$$x \notin L \Rightarrow \Pr[M(x) = 1] = 0$$

# Examples of randomized algorithms

- Polynomial Identity Testing
  1. **Input:** straight-line program (algebraic circuit) and $1^d$ where $d$ is upper bound on degree
  2. **Output:** is the polynomial computed the zero polynomial?

# Examples of randomized algorithms

- Polynomial Identity Testing
  1. **Input:** straight-line program (algebraic circuit) and $1^d$ where $d$ is upper bound on degree
  2. **Output:** is the polynomial computed the zero polynomial?
- (Bipartite) Perfect Matching
  1. **Input:** graph $G(V, E)$
  2. **Output:** does $G$ have a perfect matching?

# Examples of randomized algorithms

- Polynomial Identity Testing
  1. **Input:** straight-line program (algebraic circuit) and $1^d$ where $d$ is upper bound on degree
  2. **Output:** is the polynomial computed the zero polynomial?
- (Bipartite) Perfect Matching
  1. **Input:** graph $G(V, E)$
  2. **Output:** does $G$ have a perfect matching?
- Primality Testing
  1. **Input:** $N \in \mathbb{N}$ in binary
  2. **Output:** is $N$ prime?

# Why was usefulness expected?

- Often times randomized algorithms follow template:
  1. Fix a particular (deterministic) action $A$
  2. Flip coins and perform $A$ based on the outcome of the flips
  3. Repeat the above enough times

# Why was usefulness expected?

- Often times randomized algorithms follow template:
  1. Fix a particular (deterministic) action $A$
  2. Flip coins and perform $A$ based on the outcome of the flips
  3. Repeat the above enough times
- From probability theory, we have many events which when done independently exhibit concentration of measure

# Why was usefulness expected?

- Often times randomized algorithms follow template:
  1. Fix a particular (deterministic) action $A$
  2. Flip coins and perform $A$ based on the outcome of the flips
  3. Repeat the above enough times
- From probability theory, we have many events which when done independently exhibit concentration of measure
- In cases above, saw that the problems which had:
  1. a property which distinguished the YES and NO instances
  2. quantitatively different number of witnesses

# Why was usefulness expected?

- Often times randomized algorithms follow template:
  1. Fix a particular (deterministic) action $A$
  2. Flip coins and perform $A$ based on the outcome of the flips
  3. Repeat the above enough times
- From probability theory, we have many events which when done independently exhibit concentration of measure
- In cases above, saw that the problems which had:
  1. a property which distinguished the YES and NO instances
  2. quantitatively different number of witnesses
- Then, $A$ just test for this property given a random witness

**Theorem 12 (Chernoff Bound)**

*If $X_1, \ldots, X_k \in \{0, 1\}$ are independent random variables and $\Pr[X_i = 1] = p$ for all $i \in [k]$ then*

$$\Pr\left[\left|\frac{1}{k} \cdot \sum_{i=1}^{k} X_i - p > \varepsilon\right|\right] \leq 2 \cdot \exp\left(-\frac{k\varepsilon^2}{2p(1-p)}\right)$$

- Randomized Algorithms & Probabilistic TMs

- Relation to other classes

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

- Hence we have P $\subseteq$ BPP

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

- Hence we have P $\subseteq$ BPP
- Also have P $\subseteq$ RP $\cap$ coRP

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

- Hence we have P $\subseteq$ BPP
- Also have P $\subseteq$ RP $\cap$ coRP
- RP $\subseteq$ NP

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

- Hence we have P $\subseteq$ BPP
- Also have P $\subseteq$ RP $\cap$ coRP
- RP $\subseteq$ NP
- ZPP $\subseteq$ RP (also in coRP)
  1. Whenever ZPP algorithm outputs ?, output $0$ instead.

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

- Hence we have $\mathsf{P} \subseteq \mathsf{BPP}$
- Also have $\mathsf{P} \subseteq \mathsf{RP} \cap \mathsf{coRP}$
- $\mathsf{RP} \subseteq \mathsf{NP}$
- $\mathsf{ZPP} \subseteq \mathsf{RP}$ (also in coRP)
  1. Whenever ZPP algorithm outputs ?, output $0$ instead.
- $\mathsf{RP} \subseteq \mathsf{BPP}$
  1. Just need to amplify success probability.

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

- Hence we have P $\subseteq$ BPP
- Also have P $\subseteq$ RP $\cap$ coRP
- RP $\subseteq$ NP
- ZPP $\subseteq$ RP (also in coRP)
  1. Whenever ZPP algorithm outputs ?, output $0$ instead.
- RP $\subseteq$ BPP
  1. Just need to amplify success probability.
- BPP $\subseteq$ PSPACE (more on BPP next lecture)

# Relation among complexity classes

- Note that can define P as class of languages $L$ decided by poly-time PTMs such that

$$\Pr_r[M(x,r) = L(x)] = 1$$

- Hence we have P $\subseteq$ BPP
- Also have P $\subseteq$ RP $\cap$ coRP
- RP $\subseteq$ NP
- ZPP $\subseteq$ RP (also in coRP)
    1. Whenever ZPP algorithm outputs ?, output $0$ instead.
- RP $\subseteq$ BPP
    1. Just need to amplify success probability.
- BPP $\subseteq$ PSPACE (more on BPP next lecture)
- BPL $\subseteq$ SPACE$(\log^{3/2} n)$

# Randomized Reductions

## Definition 13

Language $A$ reduces to language $B$ under randomized poly-time reductions, denoted $A \leq_r B$, if there is a PTM $M$ such that

$$\forall x \in \{0,1\}^*, \ \Pr[A(x) = B(M(x))] \geq 2/3.$$

# Randomized Reductions

## Definition 13

Language $A$ reduces to language $B$ under randomized poly-time reductions, denoted $A \leq_r B$, if there is a PTM $M$ such that

$$\forall x \in \{0,1\}^*, \ \Pr[A(x) = B(M(x))] \geq 2/3.$$

- Not transitive, but still useful since $A \leq_r B$ and $B \in$ BPP then $A \in$ BPP.

# Randomized Reductions

## Definition 13

Language $A$ reduces to language $B$ under randomized poly-time reductions, denoted $A \leq_r B$, if there is a PTM $M$ such that

$$\forall x \in \{0,1\}^*, \ \Pr[A(x) = B(M(x))] \geq 2/3.$$

- Not transitive, but still useful since $A \leq_r B$ and $B \in$ BPP then $A \in$ BPP.
- Randomized reductions are useful in several settings, and in this course we will see an application when we study counting

# References I


Arora, Sanjeev and Barak, Boaz (2009)
Computational Complexity, A Modern Approach — Chapter 7
Cambridge University Press

Trevisan, Luca (2002)
Lecture notes — Chapter 5
See webpage

Goldreich, Oded (2006)
Computational complexity: a conceptual perspective. — Chapter 6
https://www.wisdom.weizmann.ac.il/~oded/cc-drafts.html