CS 6810: Theory of Computing **Spring 2026**

Lecture 13: March 10, 2026

Lecturer: Eshan Chattopadhyay *Scribe: Erald Sinanaj*

1 Introduction

So far we've seen various extensions and conditions on computation with Turing machines, each of which defines a complexity class of its own. An often extremely useful resource for algorithms is *randomness*. For example, one of the earliest and most well known primality testing algorithms is that due to Miller and Rabin [6], an algorithm that is probabilistic in nature: it outputs whether the input number is prime with some reasonable probability that is correct (specifically, at least $1 - (1/4)^k$, where k is the number of iterations)[1]. In this section we will delve into computation with randomness and define the most fundamental randomized complexity classes.

Our first course of action is to formally define computation with randomness and what the analogue of a Turing machine should be.

Definition 1.1 (Probabilistic Turing Machine)**.** *A* probabilistic Turing machine *is a (multitape) Turing machine equipped with an extra* random input *tape. The machine takes as usual an input string $x \in \{0,1\}^*$ and another* random string *$r \in \{0,1\}^*$.*

Note: The input of "interest" is x and we will define the runtime of such machines with respect to the length of that input. So if the runtime of the machine is $T(|x|)$, then we can assume that $|r| \leq T(|x|)$. We'll only focus on poly-time machines and thus we can always say that $|r| = \text{poly}(|x|)$.

Remark 1.2. *There's a definition of random computation that is equivalent to that given above. Specifically, we can define a probabilistic Turing machine as a TM with two transition functions δ_0, δ_1. The Turing machine has access to independent and fair coins, which it uses to choose which transition function to use. While as mentioned, this definition turns out to characterize the same computation (each one can simulate the other with polynomial overhead), we will work with the one formally stated previous to it, as it is more convenient.*

2 Randomized Complexity Classes

Definition 2.1 (**BPP**: Bounded-error Probabilistic Polynomial time)**.** *BPP is defined as the class of languages L for which there exists a PTM M such that M decides L, in the following sense:*

$$\forall x \in \{0,1\}^* : \mathbb{P}_r\left(M(x,r) = L(x)\right) > 2/3$$

and M runs in polynomial time (in the length $|x|$ of the input x).

Remark 2.2. *It's reasonable to ask "why 2/3?". One can wonder why we did not define BPP with a lower bound of $1/2$ on the probability of the algorithm being correct. This defines the larger class PP. As we will see later, we can increase the error to decide a BPP language from $1/3$ to any constant arbitrarily close to 0, with a small overhead* [2]*. This is not true for PP.*

[1] Even though now we know that primality testing is in P, that is, there is a polynomial time deterministic algorithm: the AKS algorithm [2].

[2] And in fact, we could have defined BPP with a lower bound of $1/2 + c$ for any constant c, or even c inverse-polynomially related to the input length, these all define the same class.

The class BPP gives what we call a "two-sided error", but sometimes we can design algorithms that don't give false positives (or negatives). This motivates us to define and study the following two classes.

Definition 2.3 (**RP**: Randomized Polynomial time)**.** *RP is the class of languages L for which there exists a poly-time PTM M such that for all $x \in \{0,1\}^*$:*

$$\begin{aligned} x \in L &\implies \mathbb{P}_r\left(M(x,r)=1\right) \geq 1/2 \\ x \notin L &\implies \mathbb{P}_r\left(M(x,r)=1\right) = 0 \end{aligned}$$

Note: the guarantee of $1/2$ is non-trivial here, as we have a complete lack of false positives.
We then define co-RP, the no false-negative analogue of RP.

Definition 2.4 (**co-RP**)**.** *co-RP is the class of languages L for which there exists a poly-time PTM M such that for all $x \in \{0,1\}^*$:*

$$\begin{aligned} x \in L &\implies \mathbb{P}_r\left(M(x,r)=0\right) = 0 \\ x \notin L &\implies \mathbb{P}_r\left(M(x,r)=0\right) \geq 1/2 \end{aligned}$$

Note: The class co-RP is indeed the complement-class of RP. If a language $L \in$ RP then its complement $\bar{L} \in$ co-RP. To see this, one makes the trivial observation that $x \notin \bar{L} \Leftrightarrow x \in L$ and so one can use the RP machine that "decides" L and flip its output to get a co-RP machine that decides $\bar{L}$.

Randomness can come in other flavors in the design of algorithms, not just pertaining to the probability of the correctness. It could be a property of its *running time*, as is the case for Las Vegas algorithms.

Definition 2.5 (**ZPP**: Zero error Probabilistic Polynomial time)**.** *ZPP is the class of languages L for which there exists a PTM M such that for all $x \in \{0,1\}^*$:*

$$\mathbb{P}_r\left(M(x,r) = L(x)\right) = 1 \qquad \text{(correct a.s.)}$$

and

$$\mathbb{E}_r\left[T_M(x,r)\right] = \text{poly}(|x|) \qquad \text{(polynomial expected running time)}$$

where $T_M(x,r)$ is the random, running time of M with input (x,r).

We now state the following interesting result that relates this seemingly unrelated class to RP and co-RP.

Theorem 2.6. *ZPP = RP ∩ co-RP*

Proof. We first prove the inclusion ZPP $\subseteq$ RP $\cap$ co-RP. Let L $\in$ ZPP and the PTM M that decides it. Consider the PTM M' that on input (x,r), runs $M(x,r)$ for time $2\mathbb{E}\left[T_M(|x|)\right]$ (where $T_M(|x|)$

is the running time of M under input x). M' will output $M(x,r)$ if M has halted in that time, else it outputs 0. Clearly:

$$\begin{aligned}\mathbb{P}\left(M'(x,r)=L(x)\right) &\geq \mathbb{P}\left(T_m(|x|)\leq 2\mathbb{E}\left[T_m(|x|)\right]\right)\\ &= 1-\mathbb{P}\left(T_m(|x|)>2\mathbb{E}\left[T_m(|x|)\right]\right)\\ &\geq 1-\frac{\mathbb{E}\left[T_m(|x|)\right]}{2\mathbb{E}\left[T_m(|x|)\right]} && \text{(Markov's inequality)}\\ &= 1/2\end{aligned}$$

Thus, L $\in$ RP. Similarly, we can show that also L $\in$ co-RP.

We now move on to showing the inverse inclusion: RP $\cap$ co-RP $\subseteq$ ZPP. Suppose L $\in$ RP $\cap$ co-RP. Then, there exist polynomial-time PTMs M_1 and M_2, such that M_1 gives no false-positives and is otherwise correct with probability at least 1/2, while M_2 has the inverse properties. We construct the PTM M' that on input (x,r) simulates $M_1(x,r)$ and $M_2(x,r)$ in the following way. M' first runs M_1, if M_1 responds with 1, then M also outputs 1, else M' runs M_2. Then, if M_2 responds with 0, M' outputs 0 too. If not, then M' repeats this alternating simulation procedure.

First, observe that M' is correct a.s., as M_1 gives no false positives and M_2 gives no false negatives. It remains to argue that M' runs in expected polynomial time. To do that, first suppose $x \in$ L and let α_{M_1} be the random number of iterations until M_1 outputs the correct output, 1. Observe that α_1 is geometrically distributed with success probability at least a 1/2, i.e. $\mathbb{P}\left(\alpha_{M_1}=i\right)\leq\left(\frac{1}{2}\right)^i$ and $\mathbb{E}\left[\alpha_1\right]=2$. Similarly, if $x\notin$ L the expected number of rounds α_2 for M_2 to output 0 is 2. We have for the running time of M' (M_1 and M_2 run in deterministic time):

$$\begin{aligned}\mathbb{E}\left[T_{M'}(|x|)\right] &= \mathbf{1}(x\in L)T_{M_1}(|x|)\mathbb{E}\left[\alpha_1\right]+\mathbf{1}(x\notin L)T_{M_2}(|x|)\mathbb{E}\left[\alpha_2\right]\\ &\leq 2\left(T_{M_1}(|x|)+T_{M_2}(|x|)\right)\\ &= \text{poly}(|x|)\end{aligned}$$

□

3 Error reduction

In this section we'll see how the classes we defined are actually stronger than they seem and how in general we can reduce an algorithm's error to be arbitrarily close to 0.

Theorem 3.1. *Define RP^ε to be the class of languages L for which there exists a poly-time M such that if $x\in L$, then $\mathbb{P}\left(M(x,r)=1\right)\geq 1-\varepsilon$, else if $x\notin L$, then $\mathbb{P}\left(M(x,r)=1\right)=0$.*

Then, $RP = RP^\varepsilon$, for any $\varepsilon\in[2^{-\text{poly}(|x|)},1/2]$.

Proof. It's easy to see that $\text{RP}^\varepsilon\subseteq$ RP. We will show the inverse inclusion. Let $L\in$ RP and let M be a PTM that decides it. We construct M', which on input $(x,r_1||r_2||\ldots||r_k)$ simulates $M(x,r_1),M(x,r_2),\ldots,M(x,r_k)$ sequentially and outputs the disjunction (OR) of the outputs. Observe that if $r=r_1||r_2||\ldots||r_k$ is uniformly random, then $r_i, i\in[k]$ are independent and also uniform. Clearly, if $x\notin L$, then $M(x,r)=0$. On the other hand, if $x\in L$, then $\mathbb{P}\left(M(x,r)=0\right)=\mathbb{P}\left(M(x,r_1)=0\wedge\ldots\wedge M(x,r_k)=0\right)<\left(\frac{1}{2}\right)^k$. Choosing $k=\log(1/\varepsilon)$, then $\mathbb{P}\left(M(x,r)=0\right)\leq\varepsilon$. Since $\varepsilon\geq 2^{-\text{poly}(|x|)}$ and M runs in polytime, we also have that M' runs in poly-time. □

Obviously, one can state a similar result for co-RP. In the previous section we had hinted that BPP also allows for such error reduction. To prove so we'll have to work a bit differently than RP and to do so, we'll need Chernoff's bound, a result on the concentration of random variables:

Theorem 3.2 (Left-tail Chernoff bound)**.** *Suppose $X_1, \ldots, X_n$ are independent random variables with $X_i \in \{0,1\}$. Let $X = \sum_{i=1}^{n} X_i$ with $\mathbb{E}[X] = \mu$. The following holds for all $\delta \in [0,1]$:*

$$\mathbb{P}(X < (1-\delta)\mu) \leq e^{-\frac{1}{2}\mu\delta^2}$$

Now we are ready to state and prove the result.

Theorem 3.3. *Define BPP^{ε} to be the class of languages L for which there exists a poly-time M such that $\mathbb{P}(M(x,r) = L(x)) \geq 1 - \varepsilon$.*

Then, $BPP = BPP^{\varepsilon}$, for any $\varepsilon \in [2^{-\text{poly}(|x|)}, 1/3)$.

Proof. It is immediate to see that $\text{BPP}^{\varepsilon} \subseteq \text{BPP}$. To show the opposite inclusion, let $L \in \text{BPP}$ and M a machine that decides it. Once again, we construct a PTM M' that simulates a number of independent runs of M, say k. Let $b_1, \ldots, b_k$ be the output bit of each such simulation. M' will output the majority of these bits. We will bound the error of M' using Chernoff's bound. Let $X_i := \mathbf{1}(b_i = L(x))$ and $X = \sum_{i=1}^{n} X_i$. Observe that $\mathbb{E}[X_i] = \mathbb{P}(M(x,r) = L(x)) > 2/3$ and thus $\mathbb{E}[X] > 2k/3$.

Now, we have:

$$\begin{aligned}
\mathbb{P}\left(M'(x,r) \neq L(x)\right) &= \mathbb{P}(X < k/2) \\
&= \mathbb{P}(X < (1-1/4)\mu) \\
&\leq e^{-\frac{1}{2}\mathbb{E}[X]\left(\frac{1}{4}\right)^2} && \text{(Chernoff's bound)} \\
&\leq e^{-\frac{1}{48}k}
\end{aligned}$$

By choosing $k = 48\log(1/\varepsilon)$ we have what's claimed. □

4 A few questions

There's a wealth of interesting questions in randomized complexity.

Open problem 4.1. *P vs BPP*

It is not known if P = BPP. Obviously, P ⊆ BPP. Primality testing was a problem known to be in BPP but only relatively recently proven to be in P. A problem that is in BPP and not yet known if it's in P is polynomial identity testing. Nisan and Wigderson [5] (and follow up works) showed that if sufficiently hard problems exist, in the sense that there exists a language $L \in \text{E} = \text{DTIME}(2^{O(n)})$ that can't be computed by circuits of size $2^{\delta n}$, for a small δ, then it is the case that $P = BPP$.

A few other questions that have been answered and we'll see in the following lecture are a result by Adleman [1] and a result that comes from the effort of Sipser [7], Gács [3] and Lautemann [4].

Theorem 4.2 (Adleman, 1978)**.** $BPP = P_{/Poly}$

Theorem 4.3 (Sipser-Gács-Lautemann)**.** $BPP \subseteq \Sigma_2 \cap \Pi_2$

References

[1] Leonard Adleman. "Two theorems on random polynomial time". In: *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*. 1978, pp. 75–83. DOI: 10.1109/SFCS.1978.37.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Annals of Mathematics* 160.2 (2004), pp. 781–793. DOI: 10.4007/annals.2004.160.781.

[3] Peter Gács. "On the symmetry of algorithmic information". In: *Soviet Mathematics Doklady* 27.4 (1983). English translation of Doklady Akademii Nauk SSSR, 271(4):777–780, pp. 799–803.

[4] Clemens Lautemann. "BPP and the polynomial hierarchy". In: *Information Processing Letters* 17.4 (1983), pp. 215–217. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(83)90044-3. URL: https://www.sciencedirect.com/science/article/pii/0020019083900443.

[5] Noam Nisan and Avi Wigderson. "Hardness vs randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167. ISSN: 0022-0000. DOI: https://doi.org/10.1016/S0022-0000(05)80043-1. URL: https://www.sciencedirect.com/science/article/pii/S0022000005800431.

[6] Michael O Rabin. "Probabilistic algorithm for testing primality". In: *Journal of Number Theory* 12.1 (1980), pp. 128–138. ISSN: 0022-314X. DOI: https://doi.org/10.1016/0022-314X(80)90084-0. URL: https://www.sciencedirect.com/science/article/pii/0022314X80900840.

[7] Michael Sipser. "A complexity theoretic approach to randomness". In: *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*. STOC '83. New York, NY, USA: Association for Computing Machinery, 1983, pp. 330–335. ISBN: 0897910990. DOI: 10.1145/800061.808762. URL: https://doi.org/10.1145/800061.808762.