Indian Association for the Cultivation of Science
(Deemed to be University under *de novo* Category)

Master's/Integrated Master's-PhD Program/ Integrated Bachelor's-Master's Program/PhD Course

Theory of Computation II: COM 5108

Lecture VII

Instructor: Goutam Biswas *Autumn Semester 2023*

1 Randomized Computation

A Turing machine model is augmented with the capability of making random choice of transition during computation.

1.1 Probabilistic Turing Machine

A *probabilistic Turing machine* is a mathematical model for *randomized algorithms*. This machine is structurally similar to a non-deterministic machine. But their computations are very different. The definition of a NTM is completely *structural* or *syntactic*. Whereas the definition of a probabilistic Turing machine has *semantic* component.
A class of randomized algorithms (and the corresponding machine model) may give erroneous results. But the probability of error can be made very low (as low as the probability of software or hardware failure or the probability of some other catastrophic event). These are known as *Monte Carlo algorithms*. Another class of algorithms give correct results on termination. But the termination is not guaranteed. But its expected running time is "good" (polynomial). These are known as *Las Vegas algorithm*.

Randomness is introduced in a Turing machine model essentially in two different ways. The machine may have a pair of transition functions that are selected uniformly at random from each configuration. The other ways is to introduce a tape contains a sequence of random bits. The machine is essentially deterministic and takes the input and the random bits into consideration for transition. A more formal definition of the first kind of machine is as follows.

Definition 1. A *probabilistic Turing machine (PTM) M* has two transition functions δ_0 and δ_1. At each step of computation M "flips a fair coin" and chooses one of them with equal probability $(1/2)$ to go to the next configuration. The choice at any state is independent of the previous choices.
Finally the machine halts either in *accept (output 1)* or in *reject (output 0)* state.

A random variable $M(x)$ is associated with with the output of the machine. We talk about $Pr[M(x) = 1]$ and $Pr[M(x) = 0]$.
If the machine halts within time $T(|x|)$ on all inputs $x \in \{0,1\}^*$, and irrespective of the outcome of coin toss, we call it $T(n)$-time bounded, where $n = |x|$.

If a PTM is running for t steps, the computation tree will have 2^t branches. Each branch will be taken with a probability $1/2^t$. So the probability of acceptance of an input x is the fraction of branches where M finally writes 1 as output, $Pr[M(x) = 1]$.

Let $L \subseteq \{0,1\}^*$. We use the notation $L(x) = 1$ if $x \in L$, and $L(x) = 0$ if $x \notin L$. Now we define the first set of probabilistic complexity classes.

Definition 2. A language $L \subseteq \{0,1\}^*$ is in the class $RTIME(T(n))$ if there is a $T(n)$ ($T : \mathbb{N}_0 \to \mathbb{N}_0$) time bounded probabilistic Turing machine (PTM), so that

$$\begin{aligned} x \in L &\Rightarrow Pr[M(x) = 1] \geq 2/3, \\ x \notin L &\Rightarrow P[M(x) = 1] = 0. \end{aligned}$$

The definition means that if $M(x) = 1$, the input x certainly belongs to L. But there may be an error if the output is 0. As there is a finite probability that even if $x \in L$, the machine may print 0.
The class $\mathbf{RP} = \bigcup_{i>0} RTIME(n^i)$. So PTMs for **RP** are polynomial time bounded. The value $2/3$ is rather arbitrary. It can be replaced by any constant $> 1/2$.

Example 1. As an example of COMPOSITE is in **RP**. Following algorithm uses *Fermat Test*. The little Fermat's theorem states that, if p is a prime and $p \not| a$, then $a^{p-1} \equiv 1 \pmod{p}$. The contrapositive statement is, if $\gcd(a, n) = 1$ and $a^{n-1} \not\equiv 1 \pmod{n}$, then n is composite.

isCompositeFT1(n) // n is odd ≥ 3
  $a \leftarrow$ rand$\{2, \cdots n-2\}$
  `if` $(a^{n-1} \bmod n) \neq 1$ `return` 1 // composite
  `else return` 0 // most likely prime

If the procedure returns 1, then n is certainly composite. But if it returns 0, there is no certainty[1].

If $L \in \mathbf{RP}$ and $x \in L$, according to the definition of **RP** there are more than $2/3$ branches of *accepting* computation of a polynomial time bounded PTM M. Treating the **RP** machine as an NTM, one accepting branch is sufficient to accept x. Therefore $\mathbf{RP} \subseteq \mathbf{NP}$.

Definition 3. A language $L \subseteq \{0,1\}^*$ is in the class **coRP** if there is a polynomial time bounded PTM M such that

$$\begin{aligned} x \in L &\Rightarrow Pr[M(x) = 0] = 0, \\ x \notin L &\Rightarrow P[M(x) = 0] \geq 2/3. \end{aligned}$$

In this case if the output is 0, then we are certain that $x \notin L$. But if the output is 1 we are not sure. In fact $\mathbf{coRP} = \{L : \overline{L} \in \mathbf{RP}\}$.

In **RP** we have error in half of the output (*one-sided*). Following class is more general and allow two sided error.

[1] $2^{340} \equiv 1 (\text{mod } 341)$ and $341 = 11 \times 31$.

Definition 4. The class **BPTIME**($T(n)$) is the collection of languages decided by some PTM in time $T(n)$ such that $Pr[M(x) = L(x)] \geq 2/3$ (**BPTIME** means, *bounded-error probabilistic* $T(n)$ *time*).
The class **BPP** (*bounded-error probabilistic polynomial-time*) is the class of languages decided by probabilistic polynomial-time turing machine with bounded error. So, $\mathbf{BPP} = \cup_{i \in \mathbb{N}} \mathbf{BPTIME}(n^i)$.

The definition says that the machine outputs the correct membership status of x and L with a probability larger than $2/3$. The choice of $2/3$ is arbitrary. From the symmetry of the definition it is clear that **BPP** is closed under complementation i.e. $\mathbf{BPP} = \mathbf{coBPP}$.

A deterministic Turing machine may be viewed as special case of a PTM. So $\mathbf{P} \subseteq \mathbf{BPP}$. The open question is whether $\mathbf{BPP} = \mathbf{P}$. Many people believe that they are i.e. every polynomial time, bounded error, probabilistic algorithm can be transformed to a polynomial time deterministic algorithm with only polynomial slowdown. This is an open question in complexity theory.
Following is an alternate definition of the class **BPP**. In case of **NP** we replaced the NTM by a DTM verifier with a second input as a *witness*. In case of **BPP** a PTM is replaced by a DTM. It is supplied with a sequence of *random bits* at every step of computation (polynomial length) as the second input.

Definition 5. A language $L \subseteq \{0,1\}^*$ is in **BPP**, if there is a polynomial-time Turing machine M and a polynomial $p(n)$ so that for every $x \in \{0,1\}^*$,

$$Pr_{r \in \{0,1\}^{p(|x|)}}[M(x,r) = L(x)] \geq 2/3.$$

Here the probability is over the sequence of random bits r. First we show that this definition is equivalent to the first definition.
Let $L \in \mathbf{BPP}$ according to the first definition. There is a polynomial ($p(n)$) time bounded PTM M and $Pr[M(x) = L(x)] \geq 2/3$.
So with $2/3$ probability we find a polynomial length choice sequence $r \in \{0,1\}^{p(|x|)}$ of M so that $M(x) = L(x)$. We can construct a DTM M' that will simulate M on x and using the choice sequence w reach $L(x) = M(x) = M'(x,w)$.
Let $L \in \mathbf{BPP}$ according to the second definition. There there is a polynomial ($p()$) length random string w and a DTM M' so that $M(x,w) = L(x)$ with probability $2/3$. We design a PTM M that randomly generates w of length $p(|x|)$ and simulates $M'(x,w)$ so that $Pr[M(x) = L(x)] = Pr[M'(x,w) = L(x)] \geq 2/3$.

The definition tells us that $\mathbf{BPP} \subseteq \mathbf{EXPTIME}$. As for every $p(n)$ the number of random bit-strings is $2^{p(n)}$. The exponential machine can enumerate them and simulate the deterministic machine. So we have $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{EXPTIME}$. Little else is known about inclusion relation of this class.

There are randomized algorithms that give correct results, but may fail to terminate.

Definition 6. Let M be a PTM. We define a random variable $T_{M,x}$, for the running time of M on input x. So $Pr[T_{M,x} = t] = p$, if M halts on x within t steps with a probability p, over the random choices of made by M. It is said that the **expected running time** of M is $T(n)$, if $\mathbb{E}[T_{M,x}] \leq T(|x|)$ for every $x \in \{0,1\}^*$.

Definition 7. The class $ZTIME(T(n))$ is the collection languages L for which there is PTM M such that on every input x its *expected run-time* is $O(T(n))$. But when it halts, $M(x) = L(x)$.

The class **ZPP** ("zero-sided" error) is the collection of languages L, such

that for each L there is a PTM M whose expected running time is bounded by a polynomial $p(|x|)$, for every $x \in \{0,1\}^*$. But when it halts, $M(x) = L(x)$.

There is an alternate way of looking at it. $L \in \mathbf{ZPP}$ if there is a polynomial time bounded PTM M such that

1. $M(x) \in \{0, 1, \text{`}unknown\text{'}(\perp)\}$,
2. if $x \in L$, then $M(x) = 1$ or $\perp$,
3. if $x \notin L$, then $M(x) = 0$ or $\perp$,
4. $Pr[M(x) = \perp] \leq 1/2$.

Example 2. An example of such an algorithm (not a decision problem) is finding the square-root of -1 modulo a prime p of the form $4k + 1$. As an example $p = 13 = 3 \cdot 4 + 1$. We want to solve the quadratic congruence $x^2 \equiv -1 (\text{mod } 13)^2$. One value of x is 5, as $13|(5^2 + 1)$.

We want to find an element $a \in \mathbb{Z}_p^*$ so that $a^2 \equiv -1 (\text{mod } p)$. If we can find an element $b \in \mathbb{Z}_p^* \setminus (\mathbb{Z}_p^*)^2$, we may take $a = b^{\frac{p-1}{4}}$, as $a^2 \equiv (b^{\frac{p-1}{4}})^2 = b^{\frac{p-1}{2}} \equiv -1 (\text{mod } p)$ (Euler's criterion).

We know that half of the elements of $\mathbb{Z}_p^*$ are quadratic non-residue[3]. So we can use the following randomized algorithm.

```
sqrt-1(p)
    do
        b ← rand{1, ··· , p − 1}
        a ← b^((p−1)/4)
    while (a^2 mod p ≠ p − 1)
    return a
```

The probability of picking a quadratic non-residue is $\frac{1}{2}$. So the expected number of times the loop is executed is 2. The probability that the algorithm has not found a quadratic non-residue after k iterations is $1/2^k$. The algorithm when terminates gives the correct a. But its running time is a *random variable* and its expected value is bounded. This type of algorithms are known as *Las Vegas algorithm*.

Proposition 1. $\mathbf{RP} \cup \mathbf{coRP} \subseteq \mathbf{BPP}$

Proof: Let the language $L \subseteq \{0,1\}^*$ be in **RP**. There is a polynomial time PTM M so that for all $x \in \{0,1\}^*$,

- if $x \in L$ i.e. $L(x) = 1$, then $Pr[M(x) = 1] \geq 2/3$,
- if $x \notin L$ i.e. $L(x) = 0$, then $Pr[M(x) = 0] = 1$

[2]Let p be an odd prime and a is an integer so that $\gcd(a, p) = 1$. If the quadratic congruence $x^2 \equiv a (\text{mod} p)$ has a solution, then a is called a *quadratic residue* of p. Otherwise it is a *quadratic non-residue* of p. When p is of the form $4k+1$, then a is a *quadratic residue* of p if $a^{(p-1)/2} \equiv 1 (\text{mod} p)$ and it is a *quadratic non-residue* of p if $a^{(p-1)/2} \equiv -1 (\text{mod} p)$ (Euler's theorem).

[3]Let $p = 13$, $\mathbb{Z}_{13}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. Consider a generator of $\mathbb{Z}_{13}^*$, say 2, $2^0 \equiv 1, 2^1 \equiv 2, 2^2 \equiv 4, 2^3 \equiv 8, 2^4 \equiv 3, 2^5 \equiv 6, 2^6 \equiv 12, 2^7 \equiv 11, 2^8 \equiv 9, 2^9 \equiv 5, 2^{10} \equiv 10, 2^{11} \equiv 7, 2^{12} \equiv 1$. All even powers of 2 are *quadratic residues* e.g. $2^4 = (2^2)^2 = 3$. So the solution of $x^2 \equiv 3 (\text{mod } 13)$ is $2^2 = 4$. Odd powers are *quadratic non-residue* e.g. $2^9 \equiv 5$. $5^{(13-1)/2} = 5^6 \equiv 12 \equiv -1$.

So we have $Pr[M(x) = L(x)] \geq 2/3$ i.e. $L \in \mathbf{BPP}$. Similarly we prove that $\mathbf{coRP} \subseteq \mathbf{BPP}$. QED.

Proposition 2. $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$.

Proof: We prove that $\mathbf{ZPP} \subseteq \mathbf{RP}$: Let $L \in \mathbf{ZPP}$. So there is a PTM M with expected running time bounded by a polynomial $p(n)$ such that $M(x) = L(x)$ on termination.

We design a polynomial time bounded PTM N such that N works as follows:

1. N simulates M for $5p(n)$ (five times expected running time of M).
2. Return $M(x)$ on termination.
3. Return 0 if the simulation does not terminate within that time.

Probability of the simulation running beyond $5p(n)$ is $\frac{1}{5}$ (Markov's inequality). So we have the following behavior of N:

1. If $x \in L$, $M(x) = 1$ or M does not terminate and $N(x) = 0$ is incorrect. This is with a probability $< 1/3$.
2. If $x \notin L$, $N(x) = 0$ irrespective of whether the simulation terminates or not.

So we have $L(x) = 0 \Rightarrow Pr[N(x) = 0] = 1$ and $L(x) = 1 \Rightarrow Pr[N(x) = 1] \geq \frac{2}{3}$. $N(x) = 1$ guarantee that $x \in L$ and $L \in \mathbf{RP}$.

Similarly we can design a polynomial time PTM N' by modifying the step (3) of N so that $L(N') = L \in \mathbf{coRP}$. So we conclude that $\mathbf{ZPP} \subseteq \mathbf{RP} \cap \mathbf{coRP}$.

In the other direction, we prove that $\mathbf{RP} \cap \mathbf{coRP} \subseteq \mathbf{ZPP}$.

Let $L \in \mathbf{RP} \cap \mathbf{coRP}$. There are two polynomial time bounded PTM M_1 and M_2 such that

1. if $M_1(x) = 1$ then $L(x) = 1$, but $M_1(x) = 0$ is not certain. But if $L(x) = 0$ the $M_1(x) = 0$
2. if $M_2(x) = 0$ then $L(x) = 0$, but $M_2(x) = 1$ is not certain.
3. In both the cases the error probability is $< \frac{1}{3}$.

We construct a PTM N as follows:

1. Simulate M_1 and M_2 alternately until $M_1(x) = 1$ or $M_2(x) = 0$.
2. If $M_1(x) = 1$, then $x \in L$. If $M_2(x) = 0$, then $x \notin L$.

The expected running time is bounded by the running time of M_1 and M_2. So $L \in \mathbf{ZPP}$. QED.

1.2 Error Reduction in RP

Let $L \in \mathbf{RP}$. We have a polynomial time ($p()$) PTM M so that

$$\begin{aligned} x \in L &\Rightarrow Pr[M(x) = 1] \geq 2/3, \\ x \notin L &\Rightarrow P[M(x) = 1] = 0. \end{aligned}$$

Consider the following machine where $q()$ is a polynomial.

M_e : input x

(i) Run M on x for $q(|x|)$ number of times.

(ii) The output is disjunction of $q(|x|)$ output.

If $M'(x) = 1$ then certainly $x \in L$ as at least once $M(x) = 1$.
If $M'(x) = 0$ (all outputs are zero), then the probability that $x \in L$ is $\frac{1}{3^{q(|x|)}}$, rather small. There cannot be any error if $x \notin L$. Therefore

$$\begin{aligned} x \in L &\Rightarrow Pr[M(x) = 1] \geq 1 - \left(\frac{1}{3}\right)^{q(|x|)}, \\ x \notin L &\Rightarrow P[M(x) = 1] = 0. \end{aligned}$$

1.3 Error Reduction in BPP

For the class **BPP** $Pr[M(x) = L(x)] \geq \frac{2}{3}$. Again this $\frac{2}{3}$ is arbitrary. Following lemma states that it is good enough to have $Pr[M(x) = L(x)] = 1/2 + 1/n^c$, where $|x| = n$ and $c > 0$ is a constant.

Lemma 1. Let $\mathbf{BPP}_{1/2+1/n^c}$ be the class of languages L so that there is a polynomial time bounded PTM M such that for each $x \in \{0,1\}^*$, $Pr[M(x) = L(x)] \geq 1/2 + 1/n^c$.
We claim that $\mathbf{BPP}_{1/2+1/n^c} = \mathbf{BPP}$.

It is clear that $\mathbf{BPP} \subseteq \mathbf{BPP}_{1/2+1/n^{-c}}$. We have to prove the other direction i.e. given a polynomial time PTM M with success probability $\frac{1}{2} + \frac{1}{n^c}$, we can construct a polynomial time PTM M' with success probability $2/3$.
Following theorem proves a stronger result. Given M we can construct a machine M' with success probability exponentially close to 1.

Theorem 2. Let there be a polynomial time bounded PTM M for the language $L \subseteq \{0,1\}^*$, so that for all $x \in \{0,1\}^*$, $Pr[M(x) = L(x)] \geq \frac{1}{2} + \frac{1}{n^c}$. Then for each $d > 0$, there is a polynomial time PTM M', such that for all $x \in \{0,1\}^*$, $Pr[M(x) = L(x)] \geq 1 - \frac{1}{2^{n^d}}$.

Let the machine M' runs M on every input $x \in \{0,1\}^*$ for $8n^{2c+d} = k$ (say) times, and let the k outputs be $o_1, \cdots, o_k \in \{0,1\}$. The value of $M'(x) = 1$ if the majority is 1, else it is 0.
We use *Chernoff bound* to show that $L \in \mathbf{BPP}_{1-1/(2^{n^d})}$.

For every $i = 1, \cdots, k$, we define a boolean random variable X_i so that $X_i = 1$ if $o_i = L(x)$, else it is 0. The random variables are independent. The expected value of X_i, $E[X_i] = Pr[X_i = 1] = \rho \geq p = \frac{1}{2} + \frac{1}{n^c}$.
Let $X = \sum_{i=1}^{k} X_i$, $\mathbb{E}[X] = \sum_{i=1}^{k} \mathbb{E}[X_i] = k\rho \geq kp$.

The PTM M' makes a mistake when the majority of answers of the runs of PTM M are wrong i.e. $X < \frac{k}{2}$. We need to find $Pr[X < \frac{k}{2}]$.

According to Chernoff bound, sufficiently small δ, $0 < \delta < 1$,

$$Pr\left[X < (1-\delta)\mathbb{E}[X]\right] \leq e^{-\frac{\delta^2 pk}{4}},$$

We have $p = 1/2 + |x|^{-c}$ and we take $\delta = |x|^{-c}/2$. If we output the majority answer, $X \geq (1-\delta)\mathbb{E}[X]$, the probability of wrong output is bounded by

$$e^{-\frac{\delta^2}{4}pk} = e^{-\frac{1}{4|x|^{2c}} \cdot (\frac{1}{2} + \frac{1}{|x|^c}) \cdot 8|x|^{2c+d}} \leq 2^{-|x|^d}.$$

1.4 BPP and Other Classes

Theorem 3. BPP $\subseteq$ P/poly
Proof: Suppose $L \in$ **BPP**. There is a polynomial time PTM M' so that for every $x \in \{0,1\}^*$, $Pr[M'(x) = L(x)] \geq 2/3$.
By error reduction we claim the following:
For every $d > 0$ there is a polynomial time PTM M'' so that for every $x \in \{0,1\}^*$, $Pr[M''(x) = L(x)] \geq 1 - 1/(2^{n^d})$, $n = |x|$.
Using the second definition of BPP, there is a polynomial time DTM M and a polynomial $p()$ such that for all $x \in \{0,1\}^*$ and all $w \in \{0,1\}^{p(n)}$ where $n = |x|$, $Pr_w[M(x,w) = L(x)] \geq 1 - 1/(2^{n^2})$. The TM is determinsitic and the is probability is over all w.
Therefore for all $x \in \{0,1\}^*$ and all $w \in \{0,1\}^{p(n)}$, $Pr_w[M(x,w) \neq L(x)] < 1/(2^{n^2})$.

We show by counting argument that there is a string $w_0 \in \{0,1\}^{p(n)}$ so that for all $x \in \{0,1\}^n$, $M(x,w_0) = L(x)$. The single string can be hardwired to get a circuit C_n such that $C_n(x) = M(x,w) = L(x)$ for all $x \in \{0,1\}^n$. The size of such C_n is quadratic in the running time of M.

We use a counting argument to show that there is such a string. A string $w \in \{0,1\}^{p(n)}$ is '*bad*' for an $x \in \{0,1\}^n$ if $M(x,w) \neq L(x)$; otherwise it is 'good'. Number of '*bad*' strings for an x cannot be more than $\frac{2^m}{2^{n^2}}$, where $|w| = m$, the total number of strings of length m are 2^m and fraction of strings that are 'bad' cannot exceed $1/(2^{n^2})$.
Considering all x, the number of '*bad*' strings are less than

$$2^n \times \frac{2^{p(n)}}{2^{n^2}} = 2^{n+p(n)-n^2} < 2^{p(n)}.$$

So there are 'good' strings and there is a polynomial size circuit family $\{C_n\}$ for L such that $x \in L$ if and only if $C_{|x|}(x) = 1$. QED.

We have the following facts (i) **BPP** $\subseteq$ **P/poly**, (ii) if **NP** $\subseteq$ **P/poly**, then **PH** $= \Sigma_2^p$. So if 3SAT can be solved in probabilistic polynomial time then **PH** collapses to Σ_2^p.

1.5 BPP Complete Problem?

The probabilistic complexity class **BPP** is defined using the class **BPTIME**(n^c), where the defining notion is *semantic* in contrast to *syntactic* notion of NDTM. Any input string $x \in \{0,1\}^*$ is either accepted with probability $\geq 2/3$ or is accepted with probability $< 1/3$.
Given a string $x \in \{0,1\}^*$, it is easy to check syntactically whether it is a valid NTM. But checking of valid encoding of a **BPP** machine is *undecidable*.

References

[MS] *Theory of Computation* by Michael Sipser, (3rd. ed.), Pub. Cengage Learning, 2007, ISBN 978-81-315-2529-6.

[SABB] *Computational Complexity, A Modern Approach* by Sanjeev Arora & Boaz Barak, Pub. Cambridge University Press, 2009, ISBN 978-0-521-42426-4.