Lecture 7: Pesudorandom Generators (I)

Lecturer: He Sun

When we design randomized algorithms, we assume that all the randomized algorithms can get truly random bits, i. e. the bits are unbiased and completed independent. This assumption leads to the question of generating truly random bits. Can we generate truly random bits using the source with little randomness? Can we effectively generate "almost random bits" that any polynomial-time algorithm fails to distinguish from truly random bits?

Another motivation of studying pseudorandomness is to understand the roles and limitations of randomness. It was shown that, under reasonable assumptions, a number of randomized complexity classes, e. g. **BPP** and **RL**, can be derandomized.

1 Indistinguishability

Definition 7.1 (probability ensembles) *A probability ensemble $\mathcal{X}$ is a family $\mathcal{X} = \{X_n\}_{n\geq 1}$ such that X_n is a probability distribution on some finite domain.*

Definition 7.2 (Computational Indistinguishability) *Let $\mathcal{D}$ and $\mathcal{E}$ be probability ensembles. The success probability of algorithm A for distinguishing $\mathcal{D}$ and $\mathcal{E}$ is*

$$sp_n(A) = \big|\Pr[A(X) = 1] - \Pr[A(Y) = 1]\big|,$$

where X has distribution $\mathcal{D}$ and Y has distribution $\mathcal{E}$. Distributions $\mathcal{D}$ and $\mathcal{E}$ are called computationally indistinguishable if for any probabilistic polynomial-time algorithm A, for any positive polynomial $p(\cdot)$, and for all sufficiently large n's $sp_n(A) < 1/p(n)$.

We use $\mathcal{D} \sim^c \mathcal{E}$ to express that $\mathcal{D}$ and $\mathcal{E}$ are computationally indistinguishable.

Definition 7.3 (Statistical Distance) *Let $\mathcal{D}$ and $\mathcal{E}$ be two distributions on a set Ω. The statistical distance between $\mathcal{D}$ and $\mathcal{E}$ is defined by*

$$\Delta(\mathcal{D}, \mathcal{E}) = \max_{X\subseteq\Omega} \left| \Pr_{\mathcal{D}}(X) - \Pr_{\mathcal{E}}(X) \right|.$$

We say that $\mathcal{D}$ and $\mathcal{E}$ are ε-close if $\Delta(\mathcal{D}, \mathcal{E}) \leq \varepsilon$.

Lemma 7.4 $\Delta(\mathcal{D}, \mathcal{E}) = \frac{1}{2} \cdot \sum_{\alpha\in\Omega} |\Pr[\mathcal{D} = \alpha] - \Pr[\mathcal{E} = \alpha]|$.

Definition 7.5 (Statistical Indistinguishability) *Two probability ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are called statistically indistinguishable if for any positive polynomial $p(\cdot)$, and for all sufficiently large n's, it holds that*

$$\Delta(X_n, Y_n) < \frac{1}{p(n)}.$$

We use $\mathcal{D} \sim^s \mathcal{E}$ to express that $\mathcal{D}$ and $\mathcal{E}$ are statistically indistinguishable.

Theorem 7.6 *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two distributions. For any function f, we have*

$$\Delta(f(\mathcal{D}_1), f(\mathcal{D}_2)) \leq \Delta(\mathcal{D}_1, \mathcal{D}_2).$$

Additionally, if f is polynomial-time computable, and $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable, then $f(\mathcal{D}_1)$ and $f(\mathcal{D}_2)$ are also computationally indistinguishable.

Proof: Let B be any event on Ω' and $f : \Omega \to \Omega'$, then

$$\begin{aligned}
\Pr_{f(\mathcal{D}_1)}[B] - \Pr_{f(\mathcal{D}_2)}[B] &= \Pr_{\mathcal{D}_1}\left[f^{-1}(B)\right] - \Pr_{\mathcal{D}_2}\left[f^{-1}(B)\right] \\
&\leq \left|\Pr_{\mathcal{D}_1}\left[f^{-1}(B)\right] - \Pr_{\mathcal{D}_2}\left[f^{-1}(B)\right]\right| \\
&\leq \Delta(\mathcal{D}_1, \mathcal{D}_2)
\end{aligned}$$

If f is polynomial-time computable, and there is an algorithm A to distinguish $f(\mathcal{D}_1)$ and $f(\mathcal{D}_2)$ with probability δ, then there is a new algorithm A' to distinguish $\mathcal{D}_1$ and $\mathcal{D}_2$: For the input $x \in \Omega$, Algorithm A' outputs $A(f(x))$.

The complexity of A' is essentially equal to A, and

$$\Pr_{x \in \mathcal{D}_1}\left[A'(x) = 1\right] = \Pr_{f(x) \in f(\mathcal{D}_1)}[A(f(x)) = 1].$$

So A' can distinguish $\mathcal{D}_1$ and $\mathcal{D}_2$ with probability δ.

In conclusion, if $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable, so are $f(\mathcal{D}_1)$ and $f(\mathcal{D}_2)$ computationally indistinguishable. ■

Proposition 7.7 *1. Given two distributions $\mathcal{D}$ and $\mathcal{E}$, if $\mathcal{D} \sim^s \mathcal{E}$, then $\mathcal{D} \sim^c \mathcal{E}$.*

2. The both are closed under function applications.(Note that for computational indistinguishability, the function must be computable.)

3. The both are closed under direct products with independent distributions.

We note that there are computationally indistinguishable probability ensembles which are statistically distinguishable, i. e. the notion of computational indistinguishability is a relaxation of the notion of statistical indistinguishability.

2 Measures of Entropy

Definition 7.8 (Entropy) *Let $\mathcal{D} = \{p_1, p_2, \cdots, p_n\}$ be a probability distribution. Then the quantity*

$$\mathbf{H}_b(\mathcal{D}) = -\sum_{i=1}^{n} p_i \log_b p_i = \sum_{i=1}^{n} p_i \log_b \frac{1}{p_i}$$

is called the b-ary entropy of the distribution $\mathcal{D}$.

In general, we use the 2-ary entropy. In this situation, we ignore suffix of $\mathbf{H}_2$ and only write $\mathbf{H}$.

The entropy measures both the amount of uncertainty in a distribution before sampling, and the amount of information obtained by sampling.

Definition 7.9 *Given a distribution $\mathcal{D}$ on $\{0,1\}^n$, the min-Entropy $\mathbf{H}_\infty(\mathcal{D})$ is defined as*

$$\mathbf{H}_\infty(\mathcal{D}) = \min_{x\in\{0,1\}^n} \{-\log_2 (\Pr[\mathcal{D} = x])\}.$$

A k-source is a distribution with min-entropy at least k. The entropy rate of a k-source on $\{0,1\}^n$ is k/n; we sometimes call a k-source a rate-k/n-source.

Note that the Shannon entropy measures the amount of randomness a distribution contains on average and the min-entropy measures the amount of randomness on the worst case.

Definition 7.10 (Renyi Entropy) *Let $\mathcal{D}$ be a distribution on a set S. The Renyi entropy of $\mathcal{D}$ is*

$$\mathbf{H}_{\text{Ren}}(\mathcal{D}) = -\log(\Pr[X = Y])$$

where $X \in_{\mathcal{D}} S$ and $Y \in_{\mathcal{D}} S$ are independent.

Definition 7.11 (Collision Probability) *Let $\mathcal{D}$ be a distribution on a set S. The collision probability of $\mathcal{D}$ is*

$$\text{CP}(\mathcal{D}) = \Pr_{X,Y\in S}[X = Y]$$

where $X \in_{\mathcal{D}} S$ and $Y \in_{\mathcal{D}} S$ are independent.

From the above notation, we found that the collision probability can be expressed as $\sum_{x\in S} (\Pr[\mathcal{D} = x])^2$ and $\mathbf{H}_{\text{Ren}}(\mathcal{D}) = -\log \text{CP}(\mathcal{D})$. Especially, if $\mathcal{D}$ is uniform on $\{0,1\}^n$, then $\text{CP}(\mathcal{D}) = 2^{-n}$ and $\mathbf{H}_{\text{Ren}}(\mathcal{D}) = n$.

Lemma 7.12 *For any distribution $\mathcal{D}$ on a set S, it holds that*

$$\frac{1}{2} \cdot \mathbf{H}_{\text{Ren}}(\mathcal{D}) \le \mathbf{H}_\infty(\mathcal{D}) \le \mathbf{H}_{\text{Ren}}(\mathcal{D}) \le \mathbf{H}(\mathcal{D}).$$

Proof: Since

$$\text{CP}(\mathcal{D}) = \sum_{x\in S} \big(\Pr[\mathcal{D} = x]\big)^2 \le \max_{x\in S} \big\{\Pr[\mathcal{D} = x]\big\} \cdot \sum_{y\in S} \Pr[\mathcal{D} = y] = \max_{x\in S} \big\{\Pr[\mathcal{D} = x]\big\},$$

therefore $-\log \text{CP}(\mathcal{D}) \ge -\log \big(\max_{x\in S} \big\{\Pr[\mathcal{D} = x]\big\}\big)$, i. e. $\mathbf{H}_{\text{Ren}}(\mathcal{D}) \ge \mathbf{H}_\infty(\mathcal{D})$.

Secondly, because

$$\sum_{x\in S} \big(\Pr[\mathcal{D} = x]\big)^2 \ge \max_{x\in S} \Big\{(\Pr[\mathcal{D} = x])^2\Big\},$$

we get

$$-\log\left(\sum_x (\Pr[\mathcal{D} = x])^2\right) \le -\log\left(\max_{x\in S} \Big\{(\Pr[\mathcal{D} = x])^2\Big\}\right),$$

i. e. $\mathbf{H}_{\text{Ren}}(\mathcal{D}) \le 2\mathbf{H}_\infty(\mathcal{D})$.

Thirdly, by AG inequality[1] we have

$$\sum_{x\in S} (\Pr[\mathcal{D} = x])^2 \ge \prod_{x\in S} (\Pr[\mathcal{D} = x])^{\Pr[\mathcal{D}=x]}.$$

[1] Let $x_1, \cdots, x_n > 0$, and $\delta_1, \cdots, \delta_n > 0$, $\sum_{i=1}^n \delta_i = 1$. Then $\sum_{i=1}^n \delta_i x_i \ge \prod_{i=1}^n x_i^{\delta_i}$.

Thus

$$-\log\left(\sum_{x\in S}(\Pr[\mathcal{D}=x])^2\right) \le -\log\left(\prod_{x\in S}(\Pr[\mathcal{D}=x])^{\Pr[\mathcal{D}=x]}\right)$$
$$= -\sum_{x\in S}\Pr[\mathcal{D}=x]\cdot\log\Pr[\mathcal{D}=x],$$

which implies $\mathbf{H}_{\mathrm{Ren}}(\mathcal{D}) \le \mathbf{H}(\mathcal{D})$.

Combining with the inequalities above, we get the result. ■

Besides Lemma 7.12, all the three measures of entropy satisfies the following properties:

- $0 \le \tilde{\mathbf{H}}(\mathcal{D}) \le \log|\mathrm{support}(\mathcal{D})|$. Here $\tilde{\mathbf{H}} \in \{\mathbf{H}, \mathbf{H}_{\mathrm{Ren}}, \mathbf{H}_\infty\}$.
- For every deterministic function f, we have $\tilde{\mathbf{H}}(f(\mathcal{D})) \le \tilde{\mathbf{H}}(\mathcal{D})$.

3 Pseudorandom Generators

Motivated by the need of making nuclear weapon, Monte Carlo method is introduced by Ulam in 1940s. Closely related is the notion of pseudorandom generators. In 1982, Blum and Micali introduced the idea of a generator which procedures its output in polynomial time such that its output passes a polynomial time test. In the same year, Yao gave another definition of pseudorandom generators, and proved this definition is equal to Blum's definition.

Loosely speaking, pseudorandom generators are defined as efficient *deterministic* algorithms which stretch *short random seeds* into longer pseudorandom sequences. There are three fundamental aspects for pseudorandom generators.

- Efficiency: The generator must be efficient, which means that the pseudorandom generators must produce pseudorandom sequences with polynomial-time. In fact, pseudorandom generators are one kind of *deterministic polynomial-time algorithm*.
- Stretching: The generator is required to stretch its input seed to a longer output sequence. Specifically, the generator stretches an n-bit input into an $\ell(n)$-bit long output, where $\ell(n) > n$. The function ℓ is called the *stretching function* of the generator.
- Pseudorandomness: The generator's output has to look random to any efficient observer. That is, any procedure should fail to distinguish the output of a generator (on a random seed) from a truly random sequence of the same length in the polynomial time. For instance, a procedure could count the number of 0's and 1's and any pseudorandom generator need output almost the same number of 0'1 and 1's.

Definition 7.13 (Pseudorandom Generators) *A deterministic polynomial-time algorithm G is called a pseudorandom generator if there exists a stretching function $\ell : \mathbb{N} \mapsto \mathbb{N}$, such that the following two probability ensembles, denoted $\{\mathcal{G}_n\}_{n\in\mathbb{N}}$ and $\{\mathcal{U}_n\}_{n\in\mathbb{N}}$, are computationally indistinguishable.*

1. *Distribution $\mathcal{G}_n$ is defined as the output of G whose length is $\ell(n)$ on a uniformly selected seed in $\{0,1\}^n$.*
2. *Distribution $\mathcal{U}_n$ is defined as the uniform distribution on $\{0,1\}^{\ell(n)}$, $\ell(n) > n$.*

That is, letting $\mathcal{U}_m$ denote the uniform distribution over $\{0,1\}^m$, we require that for any probabilistic polynomial-time algorithm A, for any positive polynomial $p(\cdot)$, and for all sufficiently large n's, it holds that

$$\left| \Pr\left[A(G(\mathcal{U}_n)) = 1\right] - \Pr\left[A\left(\mathcal{U}_{\ell(n)}\right) = 1\right] \right| < \frac{1}{p(n)}.$$

From the above definition, we know that pseudorandomness is defined in terms of its observer. It is a distribution which cannot be told apart from a uniform distribution by any polynomial-time observer. However, pseudorandom sequences may be distinguished from truly random ones by infinitely powerful observers or more powerful observers. For instance, the pseudorandom sequence that cannot be distinguished from truly random ones by any polynomial-time observer could be distinguished from truly random ones by an exponential-time machine. So pseudorandomness is subjective to the abilities of the observer.

The Formulation of Pseudorandom Generators

The formulation of pseudorandom generators consists of three aspects: (1) The stretching measure of the generators; (2) The class of distinguishers that the generators are supposed to fool, i. e. the class of algorithms that are allowed to distinguish the output of generators and the truly uniform distributions; (3) The resources that generators are allowed to use.

As mentioned above, pseudorandom generators and computational difficulty are strongly related. To show the computational difficulty, we introduce the notion of one-way functions.

One-way functions is the foundation of modern cryptography, and closely related to public-key cryptosystem, pseudorandom generators, and digital signature. Intuitively, one-way functions are the class of functions that are easy computed and hard inverted. The formal definition of one-way functions is as follows.

Definition 7.14 (One-way Functions) *A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if f satisfies the following two conditions:*

1. *There exists a polynomial-time algorithm A to compute f, i. e. $\forall x : A(x) = f(x)$.*
2. *For all probabilistic polynomial-time A', polynomials $p(\cdot)$ and sufficiently large n's, it holds that*
$$\Pr\left[A'(f(\mathcal{U}_n)) = f^{-1} \circ f(\mathcal{U}_n)\right] < \frac{1}{p(n)}.$$

We call the algorithm that tries to invert a one-way function or tries to distinguish the output of a pseudorandom generator from a truly random string *an adversary*.

In 1993, Håstad, Impagliazzo, Levin and Luby proved the following definitive theorem: Starting with any one-way function, one can construct a pseudorandom generator.

Theorem 7.15 *Pseudorandom generators exist if and only if one-way functions exist.*

There are a few problems that seem to be one-way in practice and are conjectured to be one-way. A typical example is the discrete logarithm function.

Problem 7.16 (The Discrete Logarithm Problem) *Given a prime modulus p and a generator $g \in \mathbb{Z}_p^*$, for $y = g^x \mod p$ find the index x.*

Before giving the constructions of PRGs, we show that it suffices to construct PRGs with stretching function $\ell(n) = n + 1$.

Theorem 7.17 (amplification of stretch function) *Suppose we have a pseudorandom generator G with a stretch function $n + 1$, then for every polynomial $\ell(n) > n$ there exists a pseudorandom generator with stretch function $\ell(n)$.*

Proof: Let G be a pseudorandom generator with a stretching function $n+1$. We construct a pseudorandom generator G^i with stretching function $\ell(n) = n + 1$. Define

$$G^i(X_n) = \begin{cases} G\left(X_n\right) & i = 1 \\ G^{i-1}\left(G\left(X_n\right)_{1\cdots n}\right) \circ G\left(X_n\right)_{n+1} & i > 1 \end{cases}$$

where $G(X_n)_i$ is the i-bit of $G(X_n)$, $G(X_n)_{i\cdots j}$ is the substring of $G(X_n)$ from the i-th bit up to the j-th bit, and $\circ$ represents the concatenation operator between two strings.

Define a sequence of distributions as follows:

$$\mathcal{D}_0 : X_n \circ \mathcal{U}_m, \quad \mathcal{D}_1 : G(X_n) \circ \mathcal{U}_{m-1}, \ \cdots, \ \mathcal{D}_m : G^m(X_n).$$

Now we show that each G^i is a PRG. At first, an obvious fact is that $G^m(X_n) \sim^c \mathcal{U}_{n+m}$ if and only if $G^m(X_n) \sim^c \mathcal{D}_0$. So it suffices to show that there is no algorithm to distinguish $G^m(X_n)$ from $\mathcal{D}_0$. Assume that there is an algorithm A, and polynomial p, such that

$$\left|\Pr\left[A(\mathcal{D}_m) = 1\right] - \Pr\left[A(\mathcal{D}_0) = 1\right]\right| \geq \frac{1}{p(n)}.$$

Because

$$\begin{aligned} \Big|\Pr[A(\mathcal{D}_m) = 1] - \Pr[A(\mathcal{D}_0) = 1]\Big| &= \left|\sum_{i=1}^{m} \left(\Pr[A(\mathcal{D}_i) = 1] - \Pr[A(\mathcal{D}_{i-1}) = 1]\right)\right| \\ &\leq \sum_{i=1}^{m} \Big|\Pr[A(\mathcal{D}_i) = 1] - \Pr[A(\mathcal{D}_{i-1}) = 1]\Big| \end{aligned}$$

therefore there is $i \in \{1, \cdots, m\}$ such that

$$\Big|\Pr[A(\mathcal{D}_i) = 1] - \Pr[A(\mathcal{D}_{i-1}) = 1]\Big| > \frac{1}{m \cdot p(n)}$$

which contradicts the assumption that $\mathcal{D}_i \sim^c \mathcal{D}_{i+1}$. ■