6.895 Randomness and Computation April 16, 2008

Lecture 19: PRG for Space-Bounded Computation

Lecturer: Ronitt Rubinfeld (lecture given by Krzysztof Onak) *Scribe: Ning Xie*

1 Definitions and Models of Computation

A randomized algorithm A can be thought of as a function $A : \{0,1\}^n \times \{0,1\}^{R(n)} \to \{\text{Accept}, \text{Reject}\}$, that is, function A is a *deterministic* algorithm that takes two input strings x and y, where x is the "real" input to the randomized algorithm and y is the random string used during the computation.

Definition 1 *A (deterministic) function $G : \{0,1\}^m \to \{0,1\}^{R(n)}$ is a* pseudorandom generator (PRG) *for algorithm A with parameter ϵ if for all x,*

$$\left| \Pr_y[A(x,y) \text{ accepts}] - \Pr_z[A(x, G(z)) \text{ accepts}] \right| \leq \epsilon.$$

We are going to study the following PRG construction.

Theorem 2 (Nisan 1990) *For any algorithm A that runs in $S(n) = \Omega(\log n)$ space and uses $R(n)$ random bits, there is a pseudorandom generator for A with parameter $\frac{1}{10}$ that uses $O(S(n) \log R(n))$ random bits and runs in $O(S(n) \log R(n))$ space.*

The following claim easily follows.

Corollary 3 (Nisan 1990) *If a randomized algorithm A runs in $S(n) = \Omega(\log n)$ space and uses $R(n)$ random bits, then A can be converted into a randomized algorithm A' that runs in $O(S(n) \log R(n))$ space and uses $O(S(n) \log R(n))$ random bits.*

Consider the model of Turing machine computation when space complexity is our main concern. The TM has two tapes, one is read-only input tape of size n and the other is a work tape of size $S(n)$. The space complexity of the TM is $S(n)$ (that is, the read-only input tape will not be counted). Such a TM has at most $n \cdot 2^{O(s(n))}$ states, and if $s(n) = \Omega(\log n)$, this can be bounded by $2^{O(s(n))}$ states.

2 Pairwise Independent Hash Functions and Hash Mixing Lemma

Definition 4 *Let $H = \{h : \{0,1\}^r \to \{0,1\}^r\}$ be a set of functions. H is called a family of* pairwise independent hash functions *(or a* universal family of hash functions*) if for all $x_1 \neq x_2$ and for all $y_1, y_2 \in \{0,1\}^r$,*

$$\Pr_{h \in H}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = 2^{-2r}.$$

For our PRG construction purposes, we only need the following well-known fact about universal hash functions.

Fact 5 *For every $r > 0$, there exists a small family H of pairwise independent hash functions that go from $\{0,1\}^r$ to $\{0,1\}^r$ such that each $h \in H$ can be represented by $O(r)$ bits and $h(x)$ can be computed in $O(r)$ space.*

For example, we can take H to be the set of all affine functions over the field $\mathbb{F}_{2^r}$.

The next lemma about families of pairwise independent hash functions will be the main technical tool in our proof. We first need to introduce the following definitions.

Definition 6 *For a subset A of $\{0,1\}^r$, $\mu(A) = \frac{|A|}{2^r}$.*

Definition 7 *Let $A, B \subseteq \{0,1\}^r$, $h : \{0,1\}^r \to \{0,1\}^r$ and $\epsilon > 0$. We say h is (ϵ, A, B)-*good *if*

$$\left| \Pr_{y \in \{0,1\}^r}[y \in A \text{ and } h(y) \in B] - \Pr_{y,z \in \{0,1\}^r}[y \in A \text{ and } z \in B] \right| \leq \epsilon,$$

or equivalently,

$$\left| \Pr_{y \in \{0,1\}^r}[y \in A \text{ and } h(y) \in B] - \mu(A)\mu(B) \right| \leq \epsilon$$

Lemma 8 (Hash Mixing Lemma) *Let H be a universal family of hash functions that map $\{0,1\}^r$ to $\{0,1\}^r$, then for any $A, B \subseteq \{0,1\}^r$,*

$$\Pr_{h \in H}[h \text{ is not } (\epsilon, A, B)\text{-good}] \leq \epsilon,$$

where $\epsilon = 2^{-r/3}$.

Proof We would like to bound the number of $h \in H$ such that $\left|\Pr_{y \in \{0,1\}^r}[y \in A \text{ and } h(y) \in B] - \mu(A)\mu(B)\right| > \epsilon$, or equivalently, the number of h's with

$$\left| \Pr_{y \in A}[h(y) \in B] - \mu(B) \right| > \frac{\epsilon}{\mu(A)}. \tag{1}$$

Now define an indicator random variable Z_y^h by

$$Z_y^h = \begin{cases} 1 & \text{if } h(y) \in B, \\ 0 & \text{otherwise.} \end{cases}$$

By multiplying both sides by A, we can rewrite (1) in terms of Z_y^h

$$\left| \sum_{y \in A} Z_y^h - |A|\mu(B) \right| > \frac{\epsilon |A|}{\mu(A)} = \epsilon \cdot 2^r.$$

First note that, since H is a pairwise independent family of hash functions, one can easily check that it is also 1-wise independent. Namely, $\Pr_{h \in H}[h(x) = y] = 2^{-r}$ for all x and y. It follows that $\mathbb{E}[Z_y^h] = \mu(B)$ and $\mathbb{E}\left[\sum_{y \in A} Z_y^h\right] = |A|\mu(B)$.

Let $Y = \sum_{y \in A} Z_y^h$, a random variable that depends on h. As we already know that $\mathbb{E}[Y] = |A|\mu(B)$, our plan is to compute the variance of Y and use Chebyshev's inequality to bound from above the probability that Y deviates from its mean.

$$\begin{aligned}
\mathbb{E}\big[Y^2\big] &= \mathbb{E}\left[(\sum_{y \in A} Z_y^h)^2\right] = \mathbb{E}\left[\sum_{y \in A}\sum_{z \in A} Z_y^h Z_z^h\right] \\
&= \mathbb{E}\left[\sum_{y \in A} Z_y^h Z_y^h\right] + \mathbb{E}\left[\sum_{y \in A}\sum_{z \in A, z \neq y} Z_y^h Z_z^h\right] \\
&= \mathbb{E}\left[\sum_{y \in A} Z_y^h\right] + \sum_{y \in A}\sum_{z \in A, z \neq y} \mathbb{E}[Z_y^h]\mathbb{E}[Z_z^h] \\
&= |A|\mu(B) + |A|(|A| - 1)\mu(B)^2,
\end{aligned}$$

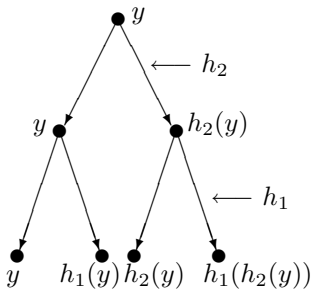

Figure 1: An example of how to construct the generator when $\ell = 2$. We assign a randomly chosen hash function for each layer (in this example h_1 and h_2 are the hash functions). The left child is simply the same string as the parent node and the right child is obtained by applying the hash function of that layer to the string at the parent node. The output of the generator is the concatenation of all the strings on the bottom layer (in this example, the output is $y \circ h_1(y) \circ h_2(y) \circ h_1(h_2(y))$).

where in the second-to-last step, we use the fact that H is a family of universal (pairwise independent) hash functions. Therefore,

$$\mathrm{Var}[Y] = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 = |A|\mu(B) + |A|(|A|-1)\mu(B)^2 - (|A|\mu(B))^2 \leq |A|\mu(B).$$

Now applying Chebyshev's inequality, which says $\Pr[|Y - \mathbb{E}[Y]| > \delta] < \frac{\mathrm{Var}[Y]}{\delta^2}$, with $\delta = \epsilon \cdot 2^r$, we get

$$\Pr\left[\left|\sum_{y \in A} Z_y^h - |A|\mu(B)\right| > \epsilon 2^r\right] < \frac{|A|\mu(B)}{\epsilon^2 2^{2r}} \leq \frac{2^r \cdot 1}{2^{-2r/3} \cdot 2^{2r}} = 2^{-r/3} = \epsilon.$$

This completes the proof of the lemma. ■

3 Nisan's Pseudorandom Generator

Now we describe how to construct the PRGs that "fool" space-bounded computation. Define a generator $G_\ell : \{0,1\}^r \times H^\ell \to (\{0,1\}^r)^{2^\ell}$, where H is a family of universal hash functions. We define G_ℓ recursively as:

$$G_0(y) = y;$$

and

$$G_\ell(y, h_1, \ldots, h_{\ell-1}, h_\ell) = G_{\ell-1}(y, h_1, \ldots, h_{\ell-1}) \circ G_{\ell-1}(h_\ell(y), h_1, \ldots, h_{\ell-1}),$$

where $\circ$ denotes concatenation. That is, we first randomly pick ℓ hash functions from H and then recursively apply these hash functions to the seed input y of length r to obtained a pseudorandom string $G_\ell(y, h_1, \ldots, h_\ell)$ of length $2^\ell \cdot r$. An example with $\ell = 2$ is illustrated in Figure 1.

We now consider the following model of randomized computation. Let us fix x, the input to our algorithm A. We now create a finite state automaton Q with states corresponding to all possible configurations of the Turing machine on x. Transitions between states in the automaton are driven by consecutive random bits delivered to the algorithm. Let us denote the number of all states of the automaton by T. Recall that $T = 2^{O(S(n))}$. One of the states is the start state, and some of the states are marked as accepting states. If after $R(n)$ transitions corresponding to $R(n)$ random bits, Q ends up at an accepting state, it accepts the input. Otherwise, Q rejects the input.

Let D be a distribution over $\{0,1\}^k$, sequences of k bits. We denote by $Q(D)$ the probability transition matrix of size $T \times T$. The (i,j)-th entry of $Q(D)$ equals the probability that the length-k sequence of bits chosen according to D results in transition from the i-th state to the j-th state. If we know $Q(D)$ for a distribution D on $R(n)$ bits, we can compute the corresponding probability of accepting the input.

Let $U_{\{0,1\}^n}$ denote the uniform distribution over $\{0,1\}^n$. Therefore, if the random bits are truly random, the transition matrix of Q will be $Q(U_{\{0,1\}^{r \cdot 2^k}})$. However, if the random bits come from a pseudorandom generator, the corresponding transition matrix may be different. We need to define a measure of distance between the effects of two distributions. We use the standard ℓ_1-norm for this purpose.

Definition 9 *For any $x \in \mathbb{R}^s$, $\|x\| = \sum_{i=1}^{s} |x_i|$. For any $s \times s$ real-valued matrix Q, the ℓ_1-norm of Q is*

$$\|Q\| = \sup_{\|x\|=1} \|xQ\|.$$

Definition 10 *A sequence of hash functions $(h_1, \ldots, h_k)$ is called ϵ-good if*

$$\|Q(G(U_{\{0,1\}^r}, h_1, \ldots, h_k)) - Q(U_{\{0,1\}^{r \cdot 2^k}})\| \leq \epsilon.$$

The correctness of Nisan's PRG follows from the following main lemma.

Lemma 11

$$\Pr[(h_1, \ldots, h_k) \textit{ is not } (2^k - 1)T^2\epsilon\textit{-good}] \leq kT^3\epsilon.$$