Lecture 9: Pseudo-random generators against space bounded computation, Primality Testing

Topics in Pseudorandomness and Complexity (Spring 2018)
Rutgers University
Swastik Kopparty

Scribes: Harsha Tirumala, Jiyu Zhang

1 Pseudo Random Generators against small space branching programs

In the previous lectures, we have seen constructions of efficient randomness extractors which help extract random bits from a weakly random source. In this lecture, we will use these randomness extractors to show that read-once branching programs operating on low space can be simulated (with very little error) using at most $O(\log n^2)$ random bits.

1.1 definitions

Definition 1. *A random variable X on $\{0,1\}^n$ has minimum entropy $H_\infty(X) \geq k$ if* $\forall x \in \{0,1\}^n \ Pr[X = x] \leq 2^{-k}$

Definition 2. *A function $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ) extractor if :* $\forall$ *random variable X with $H_\infty(X) \geq k$,*

$$\Delta(E(X, U_d), U_m) \leq \epsilon$$

The randomness extractor E helps extract the randomness hidden in a weak random source X by investing d bits of randomness (which are recovered in the process). We have already seen the existence of expander-based extractors with the following guarantees :
$\exists E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ which is a (k, ϵ) extractor for $d = O(n - k + \log(\frac{1}{\epsilon}))$

We will use the above expander-based extractor to construct a pseudo-random generator that fools $LOGSPACE$ read-once branching programs.

Fact 3. *If X_1 and X_2 are independent random variables with $H_\infty(X_1) \leq k_1$ and $H_\infty(X_2) \leq k_2$, then $f(X_1, X_2)$ has $H_\infty \leq k_1 + k_2$*

Theorem 4. *For any $\delta > 0$, there is a pseudorandom generator taking a uniformly random seed of length $O(log^2 n)$ and producing n bits that δ-fool any $s = O(\log n)$ space read-once branching program.*

Proof. The construction is as follows. Let t, d be constants to be fixed later. Let $G_0 : \{0,1\}^t \to \{0,1\}$ be a function that returns the first bit of the input; i.e.

$$G_0(x) = x_1$$

Let $G_i\{0,1\}^{t+id} \to \{0,1\}^{2^i}$ be a function such that

$$G_i(x,y) = G_{i-1}(x)G_{i-1}(E_{i-1}(x,y)),$$

where x is the first $t+(i-1)d$ bits of input, and $E_i : \{0,1\}^{t+(i-1)d} \times \{0,1\}^d \to \{0,1\}^{t+(i-1)d}$ is a $(t+(i-1)d-2s, \epsilon')$-extractor, with ϵ' to be chosen later.

We can take E_i to be the adjacency map of a good absolute eigenvalue expander. Let $k = t+(i-1)d-2s$, and let $n' = t+(i-1)d$. If $E_i : \{0,1\}^{n'} \times \{0,1\}^d \to \{0,1\}^{t+(i-1)d}$, then we get $d = O(n'-k+\log(\frac{1}{\epsilon'})) = O(2s - \log(\frac{1}{\epsilon'}))$.

To fool a read-once branching program $B : \{0,1\}^n \to \{0,1\}$, we will need to use $G_{\log n}$, which will use $t + d\log n$ random bits. It will turn out that we can take $\epsilon' = \frac{1}{poly(n)}$, so $d = \mathrm{O}(\log n)$. So, we will need an $O(\log^2 n)$ length random seed.

Claim 5. *G_i $(\epsilon' + 2^{-s})(2^{i+1}-1)$-fools read-once branching programs of space s.*

We will prove the claim by induction on i.
For v in layer 2^{i-1} of the branching program, let $w - G_{i-1}(x)$, and let $p_v = Pr[B(w) = v]$. Let $X_v = X|_{B(w)=v}$.

Claim 6. $H_\infty(X_v) \geq t + (i-1)d - \log(\frac{1}{p_v})$.

Call v unimportant if $p_v \leq 2^{-2s}$. Then the probability of ending up in any unimportant state at all is given by $\Sigma_v p_v \leq 2^s 2^{-2s} \leq 2^{-s}$. For important states v, we have $H_\infty(X_v) \geq t + (i-1)d - 2s$.

Pick v according to p_v. By induction, this is $(\epsilon' + 2^{-s})(2^{i+1}-1)$close to the v chosen from $B(z)$, for uniformly random z. If v is good, then $E(x,v)$ is ϵ'-close to $U_{t+(i-1)d}$, so $G_{i-1}(E(x_v,y))$ is ϵ'-close o $G_{i-1}(U_{t+(i-1)d})$.

Let B_v be the length 2^{i-1} branching program starting at v. We know $B_v(G_{i-1}(U_{t+(i-1)d}))$ is ϵ'-close to $B_v(G_{i-1}(U_{2^{i-1}}))$. By induction, $B_v(G_{i-1}(U_{t+(i-1)d}))$ is $(\epsilon' + 2^{-s})(2^i - 1)$-close to $B_v(U_{2^{i-1}})$.

So, for v chosen from the distribution of $G_{i-1}(U_{t+(i-1)d})$ and v' chosen from the distribution of $B(U_{2^{i-1}})$, we have that $(v, B_v(G_{i-1}(E(x,y))))$ is $(\epsilon' + 2^{-s})(2^i-1) + 2^{-s} + \epsilon' + (\epsilon' + 2^{-s})(2^i - 1)$-close to $(v', B_v(U_{2^{i-1}}))$.

Since $i \leq \log n$, provided that s is a sufficiently large multiple of $\log n$ and $\epsilon' 2^{\log n}$ is sufficiently small relative to ϵ, it follows that $G_{\log n}$ ϵ-fools any branching program of size s, using $O(\log^2 n)$ random bits. □

2 Primality Testing

Problem: Given an integer n, decide if n is prime.
Moreover, we want to decide this in time $poly(\log n)$. (Since the input size is $\log n$)

History of Algorithmic Approach
1976 Miller-Rabin (Randomized)
1999 Agrawal, Biswas (Randomized)
2002 Agrawal, Biswas (Deterministic)

Randomized Algorithm[AB99]

We will now discuss the **randomized algorithm** due to Agrawal and Biswas.
Consider the polynomial $(x+1)^n$, which expanded to be $x^n + \binom{n}{1}x^{n-1} + \cdots + \binom{n}{n-1}x + 1$. We have the following facts:

1. If n is prime, then for all $0 < i < n$, $n \mid \binom{n}{i}$. (By $''|''$ we mean n divides $\binom{n}{i}$) Then we have
2. If n is prime, $(x+1)^n \equiv x^n + 1 \ (mod\ n)$.

Note that the congruence relation above is a congruence of polynomials with integer coefficients: $A(x) \equiv B(x) \ (mod\ n)$ if $A(x) - B(x) = nC(x)$ for some $C(x) \in Z[x]$.

Lemma 7. *n is prime iff $(x+1)^n \equiv x^n + 1 \ (mod\ n)$*

Proof: Given the fact 2 above, we want to prove the other direction by contraposition. To be specific, we want to show that if n is composite, then $\exists i,\ 0 < i < n$ such that $n \nmid \binom{n}{i}$.
Here is a quick observation: if $p \mid n$, then $\binom{n}{p} = \frac{n(n-1)\cdots(n-p+1)}{p\cdots 1}$ where the n above is divided by p below, so n no longer divides $\binom{n}{p}$.

A Naive Algorithm

A naive algorithm is to think that we can utilize methods in polynomial identity testing. To be specific, the algorithm goes as below:

1. Pick random $x \in \{0, \ldots, n-1\}$
2. Check if $(x+1)^n - x^n - 1 \equiv 0 \ (mod\ n)$

But this approach doesn't work. Why? Because in polynomial identity testing we need n to be prime as prerequisite to ensure that we choose x from a field.(This may not seem obvious but we have another reason as follows). In addition, we require that the degree of the polynomial is low in the sense that it must be less than the field size(or the number of possible values) of x. While in the above naive algorithm, the size and degree are both n, and in fact n can be extremely large.

Instead, we have the following modified algorithm due to Agrawal and Biswas.

Agrawal and Biswas' Algorithm

1. Pick random polynomial $Q(x)$ of degree $\leq d$ where d is of size $O(\log n)$.
2. Check if $(x+1)^n - x^n - 1 \equiv 0 \ (mod\ n, Q(x))$.
3. If yes, then output *Prime*, else output *Composite*.

Proof of Correctness:
It is easy to see that there's no false negative given the lemma above. That is to say, when n is prime it always outputs the correct answer. We'd like to show that if n is indeed composite, then with high probability over the choice of $Q(x)$, $(x+1)^n - x^n - 1 \not\equiv 0 \ (mod\ n, Q(x))$ holds.
Now consider a prime p where $p \mid n$(so n is composite). Let p^i be the largest power of p that divides n, so $n = p^i s$ for some s. Consider $\binom{n}{p^i} = \frac{n(n-1)\cdots(n-p^i+1)}{p^i(p^i-1)\cdots 1}$, it's easy to see that $\binom{n}{p^i} \not\equiv 0 \ (mod\ p)$. This indicates that given composite integer n, then both $(x+1)^n - x^n - 1 \not\equiv 0 \ (mod\ n)$ and $(x+1)^n - x^n - 1 \not\equiv 0 \ (mod\ p)$ hold.
Now given n is composite, we want to show that if we pick $Q(x)$ at random, then with high probability $(x+1)^n - x^n - 1 \not\equiv 0 \ (mod\ Q(x),\ p)$. By above discussion we have $(x+1)^n - x^n - 1 \not\equiv 0 \ (mod\ Q(x),\ n)$ with high probability.
Notice that picking a $Q(x)$ of degree $d = O(\log n)$ and do operations modulo p is the same as picking a $Q(x)$ of degree d with coefficients in Z_p. (that is to say, pick $Q(x)$ from $F_p[x]$).
Assume that the algorithm outputs *Prime* with probability at least α, that is,

$$\Pr_{Q(x)\in Z[x], deg(Q)\leq d}[(x+1)^n - x^n - 1 \equiv 0 \ (mod\ Q(x),\ n)] \geq \alpha$$

which is equivalanet to

$$\Pr_{Q(x)\in F_p[x], deg(Q)\leq d}[Q(x) | (x+1)^n - x^n - 1] \geq \alpha$$

then

$$\Pr_Q[Q(x) \textit{ is not irreducible}] + \Pr_Q[Q(x) \textit{ is irreducible but } Q(x) \nmid (x+1)^n - x^n - 1] \leq 1 - \alpha$$

(Note that $(x+1)^n - x^n - 1$ has unique factorization of irreducible polynomials. Also in above we write range Q below Pr just for convenience, the range is the same as above.)

Fact 8. $\Pr_{Q(x)\in F_p[x], deg(Q)=d}[Q(x) \textit{ is an irreducible polynomial }] \geq \frac{1}{d}$

Therefore,

$$\Pr_Q[Q(x) \textit{ is irreducible and } Q(x) \mid (x+1)^n - x^n - 1] \geq \frac{1}{d} - (1-\alpha)$$

Now we prove by contradiction. If $\alpha \geq 1 - \frac{1}{2d}$,then the above probability is at least $\frac{1}{2d}$. So we have at least $\frac{p^{d-1}}{2d}$ such $Q(x)$ that divides $(x+1)^n - x^n - 1$. Given $(x+1)^n - x^n - 1$, there are at most n/d irreducible factors of degree at most d. Therefore, we should have $\frac{p^{d-1}}{2d} \leq \frac{n}{d}$. But since

we have $d = poly(\log n)$, we reach a contradiction. Therefore we have

$$\underset{Q(x)\in Z[x],deg(Q)\leq d}{Pr}[(x+1)^n - x^n - 1 \not\equiv 0 \ (mod\ Q(x),\ n)] \leq \frac{1}{2d}$$

The remaining key issue in analysis of this algorithm is that how to compute $(x+1)^n - x^n - 1\ mod\ Q(x)\ mod\ p$ in a reasonable time. But we shall see that this can be done by repeatedly squaring, and it is in time $O(\log n)$.