CS 6815 Pseudorandomness and Combinatorial Constructions **Fall 2019**

Lecture 22: November 19

Lecturer: Eshan Chattopadhyay *Scribe: Benjamin Y Chan*

22.1 Space bounded derandomization: Nisan's generator

In this lecture, we explore how to derandomize space-bounded computation. In particular, recall the notation we use to describe algorithmic space-complexity:

Definition 22.1 (DSPACE, BPSPACE)

$$\begin{aligned} \mathsf{DSPACE}(s(n)) &= \{L \ : \ \exists \text{ a deterministic Turing Machine deciding } L \text{ using } O(s(n)) \text{ space}\} \\ \mathsf{BPSPACE}(s(n)) &= \{L \ : \ \exists \text{ a deterministic TM } M_L \text{ deciding } L \text{ using } O(s(n)) \text{ space,} \\ &\quad \text{with single-pass read-only input tape, and 2-sided bounded error} \\ &\quad \text{s.t. } \forall x \in \{0,1\}^*, \Pr_{r \xleftarrow{R} \{0,1\}^*} [M_L(x,r) = L(x)] \geq 2/3\} \end{aligned}$$

Here $L(x) = 1$ if $x \in L$, and $L(x) = 0$ if $x \notin L$. Note that we can alternatively characterize the deterministic Turing Machine with random string input, as a 'probabilistic' Turing Machine.)

The central open question with regards to space-bounded derandomization is the following:

Open Question 22.2 (Space Bounded Derandomization) *For $s(n) \geq c \cdot \log n$ (for constant c), does*

$$\mathsf{BPSPACE}(s(n)) \stackrel{?}{=} \mathsf{DSPACE}(s(n))$$

In this lecture we show the following weaker result:

Lemma 22.3 *For $s(n) \geq c \cdot \log n$,*

$$\mathsf{BPSPACE}(s(n)) = \mathsf{DSPACE}((s(n))^2)$$

Note that in the state of the art, it is known that $\mathsf{BPSPACE}(s(n)) = \mathsf{DSPACE}((s(n))^{1.5})$, but we will not cover that.

22.2 Read-once Branching Programs

To reason about space-bounded algorithms, we introduce a non-uniform model of computation called 'Read-Once Branching Programs' (ROBPs) that exactly capture the power of space-bounded TMs.

Definition 22.4 ((n, w)-ROBP) *A (n, w)-ROBP is a tuple $(G = (V, E), A_E, A_V)$ of a directed graph G, an assignment on the edges $A_E : |E| \to \{0, 1\}$, and an assignment on the vertices A_V, with the following structure:*

- ***(layered structure):*** *Each vertex $v \in V$ is contained in one of $n+1$ disjoint sets of vertices, $L_0, L_1, ..., L_n$, and where for each edge (u, v) it is drawn between two adjacent layers, s.t. $u \in L_i$ and $v \in L_{i+1}$ where $i \in \{0, ..., n-1\}$. Moreover, each L_i contains exactly w vertices.*
- ***(edges representing binary decisions):*** *For each vertex $v \in V$, it is the source of exactly two edges in E, denote e_1 and e_2. Moreover, exactly one of these edges is labeled with a 1, and the other is labeled with a 0 (by A_E).*
- ***(start, accept, reject states):*** *There exists $v \in L_0$ such that $A_V(v) = \mathsf{start}$. Finally, $\forall v \in L_n$, $A_V(v) \in \{\mathsf{accept}, \mathsf{reject}\}$.*

We can now interpret a ROBP as taking an input $x \in \{0,1\}^n$. Denote $x = x_0 x_1 ... x_n$. To 'run' this program, we start at the start vertex (in layer L_0), and then for each bit $x_i \in \{0,1\}$, follow the edge corresponding to the value of the bit, moving into layer L_i. If we end at a vertex labelled accept, output 'accept'. Else, we end at a vertex labelled reject, and so we output 'reject'.

Another way to interpret a ROBP is as a function $f : \{0,1\}^n \to \{\mathsf{accept}, \mathsf{reject}\}$. Note that these ROBP's are a non-uniform model of computation, in the sense that inputs with different lengths can be processed by ROBPs of different size. Note that for a (n, w)-ROBP, we also call n the 'length' of the ROBP, and w its width.

Now, we connect ROBPs and the power of space-bounded randomized algorithms.

Lemma 22.5 *Let $A(\cdot, R)$ be a randomized algorithm using space $s(n)$ and $|R|$ random bits. Fix an input z to A, and define $B(\cdot) := A(z, \cdot)$. Then B can be computed by a $(2^{s(n)}, 2^{s(n)})$-ROBP.*

Proof Sketch: First, observe that a tape of length $s(n)$ has $2^{s(n)}$ possible configurations. Construct a ROBP simulating B:

- Each layer L_i contains $2^{s(n)}$ vertices. Let each vertex correspond to a different configuration of the tape, such that in the course of running the ROBP, our location in the BP corresponds to an equivalent state of the Turing Machine.
- Denote the start state the vertex corresponding to the starting configuration of B.
- Moreover, for each state configuration $u \in L_i$, draw two edges (u, v_0), (u, v_1) (labelled 0 and 1 respectively) where v_0 corresponds to the state of B after starting in u and reading a random bit 0, and v_1 corresponds to reading a random bit 1.

Finally, note that the ROBP has a length of $2^{s(n)}$, because there are only $2^{s(n)}$ configurations of B's working tape, by a simple counting argument any execution of length $> 2^{s(n)}$ must revisit a configuration; and thus by some very unlucky random input a corresponding B would loop forever, which is a contradiction.

Note that it is important that the random tape cannot be used to store additional state; in particular, depending on the model, we should not be able to encode state in the head location on the random tape. ■

22.3 Nisan's PRG

In this section we present Nisan's PRG construction (or rather, a 'morally'-equivalent version).

In order to prove that $\mathsf{BPSPACE}(s(n)) = \mathsf{DSPACE}(s(n))$, it suffices to show a $s(n)$-space computable PRG $G : \{0,1\}^{O(\log((2^{s(n)})^2))} \to \{0,1\}^{2^{s(n)}}$, that fools the class of $(2^{s(n)}, 2^{s(n)})$-ROBPs for some small $\epsilon = 1/10$. (Note that the log term is $\log(\ell \cdot w)$, where ℓ is the length of the ROBPs we want to fool, and w is the width. We want to generate ℓ bits of randomness, in order to execute any program. Then, we can run our

ROBP for every possible preimage, of which there are $O(2^{s(n)})$, but each run takes only $s(n)$ space, so this is allowed; then take the majority output.) A proof is not given in class, and left as an exercise for the reader.

To prove that $\mathsf{BPSPACE}(s(n)) = \mathsf{DSPACE}((s(n))^2)$, which we do here, it suffices to show that:

Lemma 22.6 (Nisan's PRG) *For any ℓ, w, $\exists$ an $O(d)$-space efficient (and $O(poly(n))$-time efficient) PRG $G : \{0,1\}^d \to \{0,1\}^\ell$ with seed-length $d = O(\log \ell \cdot \log \frac{\ell w}{\epsilon})$, with $\epsilon = 1/10$ error, indistinguishable by (ℓ, w)-ROBPs.*

Let B be any (ℓ, w)-ROBP. Let D denote a pseudorandom distribution generated by the PRG (which we will show how to construct) on a uniform random seed. We want to show that $|\Pr[B(U_\ell) = 1] - \Pr[B(D) = 1]| \leq \epsilon$ for $\epsilon = 1/10$ and all such B (*Uniform Computational Indistinguishability*).

Before we get to the proof, first we look at a naive attempt.

Imagine the following PRG construction with seed length $\ell/2$. Of course this seed length is not close to the desired $O(\log(\ell w))$, but it helps illustrate a key idea in Nisan's construction. First, cut the distinguisher ROBP B in half, where the first half comprises the first $\ell/2$ layers, and the second half the last $\ell/2$ layers. Now, we use the seed $y = y_1 y_2 ... y_{\ell/2}$ to traverse the first $\ell/2$ layers of the ROBP; denote this $B_{\ell/2}(y)$. Let $V_{\ell/2}$ denote the random variable representing the possible ending locations in layer $L_{\ell/2}$; clearly $V_{\ell/2}$ is indistinguishable from $B_{\ell/2}(U_{\ell/2})$.

We hope to reuse the seed y for the second half. Unintuively (perhaps), for any vertex $v \in L_{\ell/2}$, the entropy of the seed conditioned on us reaching v can be lost, namely $H(y \mid B_{\ell/2}(y) = v) \neq \ell/2 - \log w$. To see why, notice that if some v were reachable by only a single path, then $\Pr[B_{\ell/2}(U_{\ell/2}) = v] = 2^{-n/2}$. So we cannot reuse the seed in a naive way!

Instead, we can sample an additional $z := O(\log \ell w)$ random bits, so the seed length $|y| + |z|$ is now $\ell/2 + O(\log \ell w)$ bits. Denote $z = \text{Ext}(y, z) \overset{\epsilon}{\approx} U_{\ell/2}$, where $\overset{\epsilon}{\approx}$ denotes ϵ-ROBP indistinguishability. Then we can use z to finish the walk on the second half of B. This idea that there is a set of vertices in $L_{\ell/2}$ such that the probability of reaching them is not too low - thus $V_{\ell/2}$ is a weak-source - ends up being crucial to the construction.

For Nisan's construction, we assume a nice extractor, that is reminiscent of the expander-walk extractor (we do not show its existence):

Lemma 22.7 *For any $\epsilon' > 0$, i, there exists a function:*

$$\text{Ext}_i \ : \ \{0,1\}^{i \cdot d_{\text{Ext}}} \times \{0,1\}^{d_{\text{Ext}}} \to \{0,1\}^{i \cdot d_{\text{Ext}}}$$

such that Ext_i *is an $(i \cdot d_{\text{Ext}} - \log w - \log(1/\epsilon'), \epsilon')$-seeded extractor with $d_{\text{Ext}} = O(\log{(w/\epsilon')})$.*

We now proceed to prove Lemma 22.6.

Proof: We present a recursive construction, on i. For every $0 \leq i \leq ?$:

By Lemma 22.7 we have Ext_i where Ext_i is an $(i \cdot d_{\text{Ext}} - \log w - \log(1/\epsilon'), \epsilon')$-seeded extractor. Now construct the function $G_i : \{0,1\}^{i \cdot d_{\text{Ext}}} \to \{0,1\}^{2^i}$ as follows:

$$G_i(y \| z) = \begin{cases} G_{i-1}(y) \| G_{i-1}(\text{Ext}_{i-1}(y, z)) & i > 1 \\ y \| z & i = 1 \end{cases}$$

Note that $|y| = d_{\text{Ext}} \cdot (i-1)$ and $|z| = d_{\text{Ext}}$. Thus, in the base case, we have $G_1(x) = x$ for $|x| = d_{\text{Ext}}$.

Let $G = G_{\log \ell} : \{0,1\}^{\log(\ell)\cdot O(\log(w/\epsilon'))} \to \{0,1\}^{\ell}$. This gives seed length $d_G = \log(\ell)\cdot O(\log(w/\epsilon'))$. We also choose ϵ' s.t. $\epsilon = 4^{\log \ell}\cdot \epsilon'$. We now show that for all distinguishing (ℓ, w)-ROBPs B, $|\Pr[B(U_\ell) = 1] - \Pr[B(G(U_{d_G})) = 1]| \leq \epsilon$.

We do so by induction on i. First, denote $B_{v,m}$ the program starting at vertex v (in some layer L_i), such that it reads an input of length m and outputs $v' \in [w]$, the vertex it stops at (in some layer L_{i+m}) (as opposed to $\{0,1\}$, or $\mathsf{accept}, \mathsf{reject}$). Then, denote $\epsilon_i = 4^i \cdot \epsilon'$. We prove that:

$$\forall v, i \quad B_{v,2^i}(U_{2^i}) \overset{\epsilon_i}{\approx} B_{v,2^i}(G_i(U_{i\cdot d_{\mathrm{Ext}}})) \tag{22.1}$$

Where $\overset{\epsilon_i}{\approx}$ denotes statistical distance $\leq \epsilon_i$. Proving this statement 22.1 finishes the proof.

Proof by induction on i. For the base case, where $i = 1$, and clearly since $G_1(U_{d_{\mathrm{Ext}}}) = U_{d_{\mathrm{Ext}}}$, then $B_{v,d_{\mathrm{Ext}}}(U_{d_{\mathrm{Ext}}})$ and $B_{v,d_{\mathrm{Ext}}}(G_i(U_{d_{\mathrm{Ext}}}))$ are identically distributed.

For the inductive step, we prove that for all $i >= 1$, assuming statement 22.1 is true for i, then the statement is also true for $i+1$, by hybrid argument. Consider the following 4 hybrids:

$$\begin{aligned}
&D_1 : U_{2^{i-1}} \parallel U'_{2^{i-1}}\\
&D_2 : U_{2^{i-1}} \parallel G_{i-1}(U'_{d_{\mathrm{Ext}}\cdot(i-1)})\\
&D_3 : G_{i-1}(U_{d_{\mathrm{Ext}}\cdot(i-1)}) \parallel G_{i-1}(U'_{d_{\mathrm{Ext}}\cdot(i-1)})\\
&D_4 : G_{i-1}(U_{d_{\mathrm{Ext}}\cdot(i-1)}) \parallel G_{i-1}(\mathrm{Ext}_{i-1}(U_{d_{\mathrm{Ext}}\cdot(i-1)}), U'_{d_{\mathrm{Ext}}})
\end{aligned}$$

(Note that U_c, U'_c denote independent uniform random variables of length c)

- $B_{v,2^i}(D_1) \overset{\epsilon_{i-1}}{\approx} B_{v,2^i}(D_2)$.
  Proof: by the correctness of G_{i-1}; we know that $\forall B_{v',2^{i-1}}$

  $$B_{v',2^{i-1}}(G_{i-1}(U'_{d_{\mathrm{Ext}}\cdot(i-1)})) \overset{\epsilon_{i-1}}{\approx} B_{v',2^{i-1}}(G_{i-1}(U'_{2^{i-1}}))$$

  Note that under both D_1 and D_2 the parent program $B_{v,2^i}$ reaches vertex v' in layer $L_{2^{i-1}}$ with equal probability. Thus, for any $x \in [w]$,

  $$\Pr[B_{v,2^i}(D_1) \in T] = \sum_{v'\in[w]} \Pr[B_{v,2^{i-1}}(U_{2^{i-1}}) = v'] \cdot \Pr[B_{v',2^{i-1}}(U'_{2^{i-1}}) \in T]$$
  $$\Pr[B_{v,2^i}(D_2) \in T] = \sum_{v'\in[w]} \Pr[B_{v,2^{i-1}}(U_{2^{i-1}}) = v'] \cdot \Pr[B_{v',2^{i-1}}(G_{i-1}(U'_{d_{\mathrm{Ext}}\cdot(i-1)})) \in T]$$

  So by this being a convex combination $\Pr[B_{v,2^i}(D_2) \in T]$ and $\Pr[B_{v,2^i}(D_1) \in T]$ differ by at most ϵ_{i-1}.

- $B_{v,2^i}(D_2) \overset{\epsilon_{i-1}}{\approx} B_{v,2^i}(D_3)$. By the same argument as in $D_1 \approx D_2$.

- $B_{v,2^i}(D_1) \overset{2\cdot\epsilon'}{\approx} B_{v,2^i}(D_2)$. This case is less straightforward:
  First, some definitions. Denote, for any fixed distinguisher $B_{v,2^i}$:

  $$p(u) := \Pr[B_{v,2^{i-1}}(G_{i-1}(U_{d_{\mathrm{Ext}}(i-1)})) = u]$$
  $$\mathsf{Bad} := \{u \in L_{2^{i-1}} : p(u) \leq \epsilon'/w\}$$

  Intuitively, this Bad set contains vertices in the middle layer which $B_{v,2^i}$ rarely reaches: as a result, an execution that traverses $u \in \mathsf{Bad}$ in the middle layer is rare and thus the associated input string has low entropy. (Again, by the intuition we built previously, conditioned on reaching u, we know too much about the input and cannot reuse the seed).

Now, we need to show that executions that reach a middle point that is not Bad can reuse its seed:

$$\forall q \notin \mathsf{Bad},\ (\mathrm{Ext}_{i-1}(U_{d_{\mathrm{Ext}}(i-1)}, U_{d_{\mathrm{Ext}}}) \mid (B_{v,2^{i-1}}(G_{i-1}(U_{d_{\mathrm{Ext}}(i-1)})) = q)) \overset{\epsilon'}{\approx} U_{d_{\mathrm{Ext}}(i-1)}$$

This equates to showing:

$$H_\infty(U_{d_{\mathrm{Ext}}(i-1)} \mid B_{v,2^{i-1}}(G_{i-1}(U_{d_{\mathrm{Ext}}(i-1)})) = q) \geq d_{\mathrm{Ext}}(i-1) - \log w - \log(1/\epsilon')$$

(which is left as an exercise). Then we choose an Ext_{i-1} that can handle a $k = d_{\mathrm{Ext}}(i-1) - \log w - \log(1/\epsilon')$ weak source, as given by Lemma 22.7. Finally, we compute the probability we reach any $u \in \mathsf{Bad}$, by union bound, which has maximum size w and thus the sum of $p(u)$ is $\leq \epsilon'$.

Finally, by triangle inequality then

$$\begin{aligned} |B_{v,2^i}(D_1) - B_{v,2^i}(D_4)| &\leq 2 \cdot \epsilon_{i-1} + 2 \cdot \epsilon' \\ &= 2 \cdot 4^{i-1} \cdot \epsilon' + 2 \cdot \epsilon' \\ &\leq 4^i \cdot \epsilon' \\ &= \epsilon_i \end{aligned}$$

■