6.5420 Randomness and Computation	February 26, 2026
Lecture 6	
Lecturer: Ted Pyne	*Scribe: Adelmo Morrison Orozco*

For this lecture, the goal is to achieve the de-randomization of (two-sided error) randomized log-space only using $O(\log^2 n)$ space. To this end, we focus on the problem of approximating, using a small amount of space, the probability of a directed random walk starting from a fixed position arriving at another (arbitrary) node in L steps.

1 Approximating Random Walks

Let $H = (V, E)$ be a directed graph with n nodes. We will assume that H is 2-out-regular, so the out-degree of all nodes is exactly two. It'll become clear later why this assumption is fine for our de-randomization goals.

Let $L \leq n$ be a parameter for the number of steps in our random walk. Since the out-degree is exactly two, we can (in a fixed way) label each of the outgoing edges from any node by either a 0 or a 1. An L step walk starting from s can be uniquely represented as L bits.

For notational convenience, let $H[s, x]$ denote the location of a random walk starting from s taking path $x \in \{0,1\}^L$. Given a source vertex s and an sink vertex t, we wish to estimate

$$Pr_{x \sim \mathcal{U}_L}[H[s, x] = t]$$

where $x \sim \mathcal{U}_L$ indicates that x is uniformly sampled at random from $\{0,1\}^L$.

We will briefly introduce some additional notation. Let $H_{st}(x)$ be 1 if $H[s, x] = t$ and 0 otherwise. Then, for any set of paths $S \subseteq \{0,1\}^L$ we can define $H_{st}[S]$ to be the random variable $H_{st}(x)$ for x drawn uniformly at random from S.

This allows us to rephrase the above in terms of expectations, which we do to define the problem WalkEst.

Definition 1. *Let $H = (V, E)$ be a 2-out-regular graph with $|V| = n$. Let $1 \leq L \leq n$ be the length of a walk. The problem* WalkEst *is to find a value p estimating $\mathbf{E}[H_{st}(\mathcal{U}_L)]$ such that*

$$|\mathbf{E}[H_{st}(\mathcal{U}_L)] - p| \leq \varepsilon$$

2 BPL and WalkEst

Let BPL be the class of log-space two-sided error randomized algorithms with a polynomial runtime.

Definition 2. BPL *is the set of languages L such that there exists a log-space machine M such that*

$$x \in L \text{ implies } M \text{ accepts with probability} \geq 2/3$$
$$x \notin L \text{ implies } M \text{ rejects with probability} \geq 2/3$$

We can clearly extend this definition to multi-bit outputs, as opposed to a decision problem. Denote this class FBPL.

We can show that WalkEst is in FBPL for $\varepsilon = O(1)$ (say $\varepsilon = 1/100$). Given any graph $H = (V, E)$ and any s, t, we can simulate a random walk using the randomness. For $|V| = n$, we can store a pointer to the location of our current vertex in $O(\log n)$ space. Then, we can use our randomness to perform a random walk of L steps starting at s and see if it lands on t. Do this k times, for $k = \Theta(1/\varepsilon^2)$ (using Hoeffding's), and take the average of the indicator variables of whether we reached t from s. With at least constant probability, this will yield a ε additive approximation error.

Additionally, WalkEst is BPL-hard in the following way. If we can estimate WalkEst in $O(\log^b n)$ space, then we can immediately de-randomize any BPL machine using $O(\log^b n)$ space. Fix an input x and a BPL machine M. We can represent M running on input x with randomness as a read-once branching program[1] with $L \leq n$ layers. Here M and x are fixed and the branching program is over the (read-once) randomness. Then, the probability of M accepting x is the probability that a random walk reaches the unique (WLOG) accepting state at the end layer. Finally, we can distinguish between the case where this is $\geq 2/3$, when $x \in L$, and $\leq 1/3$ otherwise, when $\varepsilon < 1/6$. Importantly, as a digraph the read-once branching program has polynomial size in the input x and is 2-out-regular.

We briefly note that when the error is one sided the corresponding class is RL. As $RL \subseteq NL$ [2], Savitch's theorem implies that $NL \subseteq DSPACE(\log^2 n)$ already. Therefore, we restrict our attention to two-sided error which is the harder case!

3 Pseudo-random Generators (PRGs)

On our goal to show $BPL \subseteq L^2$. We noticed in the previous section that it is sufficient to solve WalkEst, with small constant ε, in space $O(log^2 n)$. In order to do so, the idea is to create a 'pseudo-random generator' which can take in a small truly random seed and output L bits which are 'pseudo-random' in a way that still (approximately) gives us the desired behavior. It is sufficient that on *any* 2-out-regular graph and *any* start or end nodes we can approximate the probability that a random L-length walk starting from the source will reach the sink.

Formally,

Definition 3. *Let* $G\colon \{0,1\}^d \to \{0,1\}^L$ *be a function. We call* G *a* pseudo-random generator *ε-fooling random walks if for all 2-out-regular graphs H (with n nodes) and all s, t:*

$$|\mathbf{E}[H_{st}(\mathcal{U}_L)] - \mathbf{E}[H_{st}(G(\mathcal{U}_d))]| \leq \varepsilon$$

The aim is to construct an efficiently computable pseudo-random generator where the seed d is small. The construction will proceed iteratively, so for now let's assume we're almost there. Observe that instead of having a pseudo-random generator to fool random walks on some particular graph, we solve the harder problem of fooling for *any graph*[3].

Assume we already have a pseudo-random generator $G'\colon \{0,1\}^{d'} \to \{0,1\}^{L/2}$ such that $\mathbf{E}[H_{st}(\mathcal{U}_L)] \approx_\varepsilon \mathbf{E}[H_{st}(G(\mathcal{U}_{d'}))]$. Here, $\approx_\varepsilon$ means the values are within additive ε of each other. Then, how can we construct $G\colon \{0,1\}^d \to \{0,1\}^L$ which can also fool random walks (perhaps with a slightly worse error)?

[1] For a formal definition, see for instance "Computational Complexity: A Modern Approach" by Sanjeev Arora and Boaz Barak.

[2] If a string is in the language there will be a witness for it (in fact $\geq 1/2$ of the random strings will be witnesses, a seemingly stronger condition!). If a string is not in the language no random string makes the log-space machine accept, so there is no such witness. Therefore, $RL \subseteq NL$.

[3] We will still do the analysis for a *fixed, arbitrary* graph H, but the theorem and construction of the generator will not depend on H.

3.1 Doubling PRG

A natural first idea is to concatenate two PRGs together. In particular, let $G_1 \colon \{0,1\}^{2d'} \to \{0,1\}^L$ be the PRG taking $G_1(\sigma_1\sigma_2)$ to $G'(\sigma_1)G'(\sigma_2)$ where $\sigma_1, \sigma_2 \in \{0,1\}^{d'}$. We will now analyze this 'doubling PRG'.

Fix graph H, and nodes s, t.
We want to show that $\mathbf{E}[H_{st}(\mathcal{U}_L)] = \mathbf{E}_r[H_{st}(r)]$ is close to $\mathbf{E}[H_{st}(G_1(\mathcal{U}_d))]$ for $d = 2d'$.
Observe that $\mathbf{E}[H_{st}(\mathcal{U}_L)] = \mathbf{E}_{r_1,r_2}[H_{st}(r_1r_2)]$ can be split as the expectation over the first d' bits and the last d' bits $\mathbf{E}_{r_1}[\mathbf{E}_{r_2}[H_{st}(r_1r_2)]]$.

If we let v be the node which is obtained by traveling from s by walk r_1, the inner expectation will be $\mathbf{E}_{r_2}[H_{vt}(r_2)]$ because r_1 will be fixed. Therefore,

$$\begin{gathered}\mathbf{E}[H_{st}(\mathcal{U}_L)] = \mathbf{E}_{r_1}[\mathbf{E}_{r_2}[H_{vt}(r_2)]] \\ \approx_\varepsilon \mathbf{E}_{r_1}[\mathbf{E}_{\sigma_2}[H_{vt}(\sigma_2)]] = \mathbf{E}_{r_1}[\mathbf{E}_{\sigma_2}[H_{st}(r_1G'(\sigma_2))]] = \\ \mathbf{E}_{\sigma_2}[\mathbf{E}_{r_1}[H_{st}(r_1G'(\sigma_2))]]\end{gathered}$$

using G' is a PRG and interchanging the expectations.
Then, observe that the inner part of the last term is calculating the expectation over all initial r_1 of the indicator variable which is 1 if taking walk $G'(\sigma_2)$ (after doing r_1) reaches vertex t. The last term is $E_{\sigma_2}[\mathbf{E}_{r_1}[H_{st}(r_1G'(\sigma_2))]] \approx_\varepsilon \mathbf{E}_{\sigma_2}[\mathbf{E}_{\sigma_1}[H_{st}(G'(\sigma_1)G'(\sigma_2))]]$. Finally, $\mathbf{E}_{\sigma_2}[\mathbf{E}_{\sigma_1}[H_{st}(G'(\sigma_1)G'(\sigma_2))]]$ is $\mathbf{E}[H_{st}(G_1(\mathcal{U}_d))]$ by definition.

Therefore, this implies that the doubling PRG G_1 will be 2ε-close. However, the doubling PRG has the issue that it now uses *double* the amount of randomness. Therefore, if we start at $d' = 1$ and continue this process iteratively, we will have $d = L$ by the end. This doesn't save any randomness!

3.2 INW PRG

We will now change the doubling PRG construction to give us something better.

For the doubling PRG we needed to sample σ_1 and then σ_2 independently from σ_1. If we imagine the complete graph of random seeds $\{0,1\}^{d'}$ this is equivalent to choosing a random vertex (σ_1) and then choosing a random neighbor to visit. This takes $d' + \log_2(2^{d'}) = 2d'$ bits to sample both the random vertex and random neighbor in the graph. We will try to reduce this at the cost of some additional additive error.

An important insight the Impagliazzo-Nisan-Widgerson (INW) generator uses is the following. An expander graph behaves (in many ways) like a complete graph. We can attempt to get away with picking a random σ_1 vertex, and visiting a random neighbor in (some) good expander graph on the set of random seeds. If the degree $\Delta \ll 2^{d'}$, then we need only $\log_2(\Delta) \ll d'$ bits of randomness for this step.

Recall expander graphs.

Definition 4. *We say that an undirected graph H is an (N, d, λ)* expander *if $H = (V, E)$ is d-regular, has $|V| = N$, and $\lambda(H) = \max(|\lambda_2|, |\lambda_n|)$ where λ_i are the eigenvalues in decreasing order of the random walk matrix* [4].

[4] This is the adjacency matrix normalized by the degree d.

Expander graphs can be defined for digraphs also, by defining

$$\lambda(H) := \max_{x \perp u} \frac{\|xM\|}{\|x\|}$$

where M is the random walk matrix and u is the uniform distribution over N vertices.
(See Salil Vadhan's "Pseudorandomness" book's expander chapter for additional information.)

Let D be a $(2^{d'}, \Delta, \lambda)$ expander graph where the vertex set corresponds to the set of seeds $\{0,1\}^{d'}$. Let $D[\sigma_1, e]$ be the seed obtained by walking along edge e (with first endpoint at σ_1) in the expander graph D. The INW PRG will now take $G' \colon \{0,1\}^{d'} \to \{0,1\}^{L/2}$, which we assume ε-fools random walks, and convert it to $G_2 \colon \{0,1\}^{2d'} \to \{0,1\}^{L}$ via $G_2(\sigma_1 e) = G'(\sigma_1)G'(D[\sigma_1, e])$. Now generator G_2 will use only $d' + \log_2(\Delta)$ random bits.

How do we analyze G_2? It seems slightly cumbersome. The idea is to simply compare it to the doubling PRG, which we've shown 2ε-fools random walks! We want to show that

$$\rho = \mathbf{E}_{\sigma_1,\sigma_2}[H_{st}(G'(\sigma_1)G'(\sigma_2))] \approx_\delta \mathbf{E}_{\sigma_1,e}[G'(\sigma_1)G'(D[\sigma_1, e])] = \rho'$$

for some additive error δ.

Consider a vertex m which we think as of the midpoint of an L step walk.
Define A_m to be the set of σ_1 that can reach midpoint m from s by taking steps $G'(\sigma_1)$. Namely, $A_m = \{\sigma \mid H[s, G'(\sigma)] = m\}$. Define $B_m = \{\sigma \mid H[m, G'(\sigma)] = t\}$ to be the set of σ_2 where a walk from m by $G'(\sigma_2)$ will reach t.

Observe that $H_{st}(G'(\sigma_1)G'(\sigma_2))$ is 1 iff there exists a midpoint m such that $\sigma_1 \in A_m$ and $\sigma_2 \in B_m$. Thus, we can rephrase ρ as

$$\rho = \sum_m \frac{|A_m|}{2^{d'}} \cdot \frac{|B_m|}{2^{d'}}$$

as walks can be partitioned by their midpoint.
We can then express ρ' as

$$\begin{aligned} \rho' &= \mathbf{E}_{\sigma_1,e}[G'(\sigma_1)G'(D[\sigma_1, e])] \\ &= \sum_m Pr_{\sigma_1,e}[\sigma_1 \in A_m, D[\sigma_1, e] \in B_m] \end{aligned}$$

We will use the following pseudo-randomness property of expander graphs.

Theorem 5. *Let H be a (N, d, λ) digraph expander. Fix (possibly overlapping) sets S and T each with density α and β in G respectively. If $e(S,T)$ is the number of edges between S and T, then*

$$\left| \frac{e(S,T)}{N \cdot d} - \alpha\beta \right| \leq \lambda \sqrt{\alpha(1-\alpha) \cdot \beta(1-\beta)} \leq \lambda$$

We omit the proof, but it can be found in Salil Vadhan's "Pseudorandomness" book. The theorem says, up to an additive error of λ, the density of edges between S and T is close to the expectation if we were to choose a d-regular H randomly.

The expander mixing lemma tells us $Pr_{\sigma_1,e}[\sigma_1 \in A_m, D[\sigma_1, e] \in B_m] \approx_\lambda \frac{|A_m|}{2^{d'}} \cdot \frac{|B_m|}{2^{d'}}$, where $\approx_\lambda$ is an additive λ error. One can check this is indeed the case by noticing that $e(A_m, B_m)/(N \cdot d)$ is the

fraction of edges that go between A_m and B_m and is equivalent to the probability above[5]. Here the number of vertices is $N = 2^{d'}$.

This implies that $\mathbf{E}_{\sigma_1,e}[G'(\sigma_1)G'(D[\sigma_2, e])] \approx_{N\lambda} \rho$. Finally, as the doubling PRG is 2ε close to $\mathbf{E}[H_{st}(\mathcal{U}_L)]$, applying the triangle inequality shows that the PRG G' will $\delta = (2\varepsilon + N\lambda)$ fool random walks. Recall that we wanted the degree of the λ expander to be small. It turns out we can efficiently construct[6] (N, d, λ) expanders with $\lambda = O(1/NL)$ and degree $d \leq poly(1/\lambda) = poly(NL)$.

Now onto constructing the INW PRG. Clearly, we can have a trivial PRG with $d = 1$ and $L = 1$ with additive error 0. Then, constructing the next PRG iteratively will at each time step increase the seed length by $\log(m)$ bits, change the error from ε to $2\varepsilon + N\lambda = 2\varepsilon + O(1/L)$, and double the number of output bits. For the above settings for each iteration, doing this $\log_2(L)$ times will give seed length $d = O(\log L \cdot \log(nL))$ and (small) constant error ε.

Then, we claim we can find $\mathbf{E}[H_{st}(G(\mathcal{U}_d))]$ for $d = O(\log L \cdot \log(nL))$ in $O(d)$ space. Given the graph as input, we can take random walks corresponding to the INW PRG on some seed. Finally, we can enumerate over all seeds and find $\mathbf{E}[H_{st}(G(\mathcal{U}_d))]$ in $O(\log L \cdot \log(nL)) = O(\log^2 n)$ space by averaging the indicator variables. This shows the main initial claim by the 'BPL-hardness' of WalkEst: $BPL \subseteq DSPACE(\log^2 n)$ as desired.

4 Smaller Space De-randomization and Prologue

It turns out that $O(\log^2 n)$ space is *not* the best we can do in terms of space usage.

Michael Saks and Shiyu Zhou proved (1999) that $BPL \subseteq DSPACE(\log^{3/2} n)$.
This remained the best upper bound until a breakthrough by William Hoza (2021) showing that $BPL \subseteq DSPACE(\log^{3/2} n/\sqrt{\log\log n}) = DSPACE(\log^{3/2-o(1)} n)$.

It is important to note that these do not immediately imply polynomial time bounds, since space $O(S)$ computation could take $2^{O(S)}$ time which is superpolynomial when $S = \omega(\log n)$. We know that $BPL \subseteq P$ and $BPL \subseteq DSPACE(\log^{3/2-o(1)} n)$, but can these be achieved simultaneously? By a result of Nisan (1994) we can get a $O(\log^2 n)$ space and polynomial time algorithm. Formally, $BPL \subseteq DTISP(\log^2 n,\ poly(n))$.

Both the INW and Saks-Zhou random walk estimation results are also known to hold over general directed graphs (instead of only 2-out-regular graphs). This generalization is not necessarily a requirement to fool read-once branching programs. It is widely believed that we can fully de-randomize log-space computation: (conjecture) $BPL = L$. This remains wide open, however.

[5]The probability $Pr_{\sigma_1,e}[\sigma_1 \in A_m, D[\sigma_1, e] \in B_m]$ can be equivalently stated as the probability that we pick a random edge and it happens to be from A_m to B_m.

[6]A important point is that we won't have to construct the graph in its entirety. This would blow up our space usage. We can do so locally, so that we can still perform random walks. These are called "fully explicit" expanders.