Today

- Recap of PRGs and deterministic simulation of BPP.
- Nisan-Wigderson PRG.

BMY style PRGs

- Motivation from Cryptography.
- $G : \{0,1\}^s \to \{0,1\}^n$ is pseudorandom if for all poly sized circuits C and all polynomials p,

$$|\Pr_{x \leftarrow U_s}[C(G(x)) = 1] - \Pr_{y \leftarrow U_n}[C(y) = 1]| \leq \frac{1}{p(n)}.$$

- G is constructible in $\mathrm{poly}(s)$.
- [BMY]: OWF exist $\Longrightarrow$ imply $G : \{0,1\}^{n^\epsilon} \to \{0,1\}^n$ exist for all $\epsilon > 0$.

Det. Simulation of BPP

- BPP Simulation: Run BPP algorithm on pseudorandom string $G(x)$ for all possible seeds x and take majority vote.
- [BMY] OWF exist $\Longrightarrow$

$$\mathrm{BPP} \subseteq \bigcap_{\epsilon > 0} \mathrm{DTIME}(2^{n^\epsilon})$$

.

- Observation: Existence of PRGs not only imply the efficient det. simulation of BPP above but also imply $NP \neq P$ (Oops! more than what we intended to prove.) Is derandomizing BPP really as hard as proving $NP \neq P$?

Do we need such strong PRGs for BPP simulation?

- Running time of PRG can be relaxed to $2^{O(s)}$ since we anyway pay this overhead while cycling over all seeds.
- Enough to fool circuits of fixed size (say n) instead of all poly sized circuits. In fact, generator can take more time than the circuit it wants to fool. This gets us out of the $NP \neq P$ rut.

Alternate defn. of PRGs [Nisan-Wigderson style]

- $G : \{0,1\}^s \to \{0,1\}^n$ is (S,ϵ)-PRG if for all circuits C of size at most S,

$$|\Pr_{x \leftarrow U_s}[C(G(x)) = 1] - \Pr_{y \leftarrow U_n}[C(y) = 1]| \leq \epsilon.$$

- G is constructible in $\mathrm{DTIME}(2^{O(s)})$.

- With this definition, existence of $(n^2, \frac{1}{10})$-PRG implies

$$\mathrm{BPP} \subseteq \bigcup_{c>0} \mathrm{DTIME}(2^{O(s(n^c))})$$

.

- So, $(m^2, 1/10)$-PRG of form $G : \{0,1\}^{O(\log n)} \to \{0,1\}^n$ would imply BPP=P.

Average Case Hardness

- $f : \{0,1\}^n \to \{0,1\}$ is (S,ϵ)-hard if for every circuit C of size at most S,

$$\Pr_x[C(x) = f(x)] \leq \epsilon$$

.

- Note: If f is (S,ϵ)-hard, then the PRG $G : x \mapsto x \circ f(x)$ is a (S,ϵ)-PRG.

- Nice Idea! However, stretches by only 1 bit. Try applying f to several disjoint subsets of the input seed. More pseudorandom bits, but ratio of pseudorandom bits to amount of random bits invested still bad. Try non disjoint subsets instead with limited overlap. Concept of *design*

Designs

- A collection of subsets $\mathcal{S} = (S_1, \ldots, S_n)$ of a universe $U = [s] = \{1, 2, \ldots, s\}$ is called a (l,a)-design over $[s]$ if
  - $|S_i| = l, \forall i$.
  - $|S_i \cap S_j| \leq a, \forall i \neq j$.

- For any constant $\gamma > 0$ and any l, m, a, there exists a (l,a)-design over $[s]$ with $s = O(l^2/a), a = \gamma \log n$. Moreover, this design is constructible in time $\mathrm{poly}(s,n)$. [A simple greedy approach works!]

Nisan Wigderson PRG

- Let $\mathcal{S} = (S_1, \ldots, S_n)$ be a (l, a)-design over $[s]$ and $f : \{0,1\}^l \rightarrow \{0,1\}$ a function (supposedly hard).
- For any string z, let $z|_S$ denote the substring indexed by the bit positions of S. (e.g., $z = 01000111, S = \{1,3,5,7\}$ then $z|_S = 0001$.)
- NW PRG: $NW_{f,\mathcal{S}} : \{0,1\}^l \rightarrow \{0,1\}^n$ defined as follows:

$$z \mapsto f(z|_{S_1}) \circ f(z|_{S_2}) \circ \cdots \circ f(z|_{S_n})$$

.

Main Theorem

- If there exists a function $f : \{0,1\} \rightarrow \{0,1\}$ computable in $\mathrm{DTIME}(2^{O(l)})$ and such that for all sufficiently large l, f restricted to l bits is $(2^{\epsilon l}, 1/10n)$-hard, then $NW_{f,\mathcal{S}}$ is a $(n^2, 1/10)$-PRG.

Proof Sketch

- If $NW_{f,\mathcal{S}}$ not PRG, then there exist a predictor P that predicts the ith bit based on the (i-1) earlier bits better than random guessing.

$$\Pr_z[P(f(z|_{S_1}) \ldots f(z|_{S_{1-1}})) = f(z|_{S_i})] \geq \frac{1}{2} + \frac{1}{10n}.$$

- By averaging can fix all the bits of z except those in S_i and above statement still holds.
- Variables in each of $f(z|_{S_j}), j \neq i$ are at most $a = \gamma \log n$ in number. Thus, each of these can be computed using a lookup table T_j of size at most $2^a = n$(say). Thus, the circuit $P(T_1(x), T_2(x), \ldots, T_{i-1}(x))$

computes $f(x)$ better than random guessing contradicting f is hard on average.

Average Case vs. Worst Case

- We have show average case hardness translates into pseudorandomness. What about worst-case hardness? For Permanent, we know these are equivalent notions. Hence, if there does not exist a circuit of size at most $2^{\epsilon n}$ computing the Permanent, then BPP=P.
- Anything weaker? Impagliazzo, Wigderson, Trevisan, Sudan and Vadhan show how to convert a worst case hard function f in EXP into another function f' also in EXP but which is average case hard. This reduction involves error-correcting codes. Thus, worst case hardness suffices for pseudorandomness.

How close are we to prove BPP=P?

- We have proven circuit lower bounds (in some form) imply derandomization. But are they necessary?
- A similar result to derandomizing BPP shows AM=NP if NEXP has some form of circuit lower bound. [IKW] prove a weak converse. They show

$$AM \neq NEXP \iff NEXP \not\subseteq NP/\mathrm{poly}$$

. i.e., even a weak derandomization of AM is possible if and only if we prove circuit lower bounds for NEXP.

- Thus, derandomization could be as hard as proving circuit lower bounds!!

Alt. proof of $\mathrm{BPP} \in \Sigma_2$

- NW paradigm indicates that BPP is no harder than finding a hard function. This gives the following Σ_2 algorithm for BPP.
  - GUESS a "hard function" $f : \{0,1\}^l \to \{0,1\}$.
  - FORALL circuits C of size at most $2^{\epsilon l}$, check that C does not approximate f on more than $\frac{1}{2} + \epsilon$ fraction of inputs.
  - Use f to construct $NW_{f,\mathcal{S}}$ and use this PRG to derandomize the BPP algorithm.