Today

- Recap of PRGs and deterministic simulation of BPP
- \((s, \epsilon)-PRG\) [Nisan-Widgerson]

Definition of PRGs [Blum-Micali-Yao]

From last lecture, we defined a function \(G : \{0,1\}^s \rightarrow \{0,1\}^n\) to be a PRG, if no polynomial size circuit can distinguish its output from a purely random output. That is, if \(\forall\) polynomial sized circuits \(C\) and \(\forall\) polynomials \(p\):

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

and \(G\) is efficient, meaning it is constructible in \(p(s)\).

Derandomizing BPP

- Given a BPP algorithm, we can run the algorithm on a pseudorandom string \(G(x)\), \(\forall x\), and take majority vote.
- Notice that OWF exists \(\Rightarrow BPP = \bigcap_{\epsilon > 0} DTIME(2^{\epsilon n})\)

Observation

Do we really need such a hard definition of PRGs for BPP simulation? Using the Blum-Micali-Yao definition of PRG, we can not only derandomize BPP, but we can prove a much stronger statement: \(PRG\) exist \(\Rightarrow P \neq NP\).

Let’s weaken the definition of PRGs. If our goal is just to derandomize BPP, then it suffices that:

- \(G\) runs in \(2^{O(s)}\)
- \(G\) works against circuits of size \(n\) or even some simple polynomial \(n^2\)

Alternate Definition of PRGs [Nisan-Widgerson]

We define a function \(G : \{0,1\}^s \rightarrow \{0,1\}^n\) to be a \((S,\epsilon)-PRG\), if \(\forall\) circuits \(C\) of size at most \(S\):

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

Differences from previous definition:

- \(G\) fools circuits of size \(S\).
- Probabalistically, the difference between a purely random string and the string generated by \(G\) is \(\epsilon \neq \frac{1}{p(n)}\).
- \(G\) constructible in \(2^{O(s)}\).
Observation

If \((n^2, 1/m)\)-PRG \(G : \{0,1\}^{O(\log n)} \rightarrow \{0,1\}^n\) exists \(\Rightarrow BPP \subseteq \bigcap_{c>0} DTIME(2^{cn})\). This really implies \(BPP = P\).

Definition of Average Case Hardness

A function \(f : \{0,1\}^n \rightarrow \{0,1\}\) is \((S, \epsilon)\)-hard if \(\forall \) circuit \(C\) of size at most \(S\),

\[|Pr_x[C(x) = f(x)]| \leq \frac{1}{2} + \epsilon\]

Notice that given an \((S, \epsilon)\)-hard function \(f\), we can construct a PRG \(G : x \mapsto x \circ f(x)\) which is \((S, \epsilon)\)-hard. (We will use a similar idea to define designs.)

Definition of Designs

A collection of subsets \(S = \{S_1, ..., S_n\}\) of a universe \(U = [s] = \{1, ..., s\}\) is a \((l, \alpha)\)-design over \([s]\) if:

- \(|S_i| = l, \forall i\)
- \(|S_i \cap S_j| \leq \alpha, \forall i \neq j\)

Notation

- Let \(S = \{S_1, ..., S_n\}\) be a \((l, \alpha)\)-design over \([s]\).
- Let \(f : \{0,1\}^n \rightarrow \{0,1\}\) be a hard function.
- For any string \(z\), let \(z|_S\) denote the substring indexed by the bit positions of \(S\). (E.g., if \(z = 01001111, S = \{1, 3, 5, 7\}\), then \(z|_S = 0001\).)
- Define the Nisan-Widgerson PRG \(NW_{f,s} : \{0,1\}^l \rightarrow \{0,1\}^n\) as follows:

\[z \mapsto f(z|S_1) \circ ... \circ f(z|S_n)\]
Lemma
For any constant $\gamma > 0$ and any $l, n, a$, there exists an $(l, s)$-design over $[s]$ that is constructible in polynomial time in $s$ and $n$, where:

- $s = O\left(\frac{l^2}{n}\right)$
- $a = \gamma \log n$

Theorem
There exists a function $f : \{0, 1\}^l \rightarrow \{0, 1\}$, computable in $\text{DTIME}(2^{O(l)})$, which \forall sufficiently large $n$, when $f$ is restricted to $n$ bits, is $(2^n, \frac{1}{10^n})$-hard

$\Rightarrow$ there exists a function $NW_{f,s} : \{0, 1\}^{O(\log n)} \rightarrow \{0, 1\}^n$ which is $(n^2, \frac{1}{10})$-PRG.

Proof
If $NW_{f,s}$ is not a PRG, then there exists a predictor $P$ that predicts bit, based on $\{\text{bit}_1, \ldots, \text{bit}_{i-1}\}$ better than a random guess. That is:

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

By averaging, we can determine all the bits of $z$ except for those in $S_i$ and the above statement would still hold. 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 $2^a = a$. So:

$$\Pr[P[T_1, \ldots, T_{i-1}] = f(z|S_i)] \geq \frac{1}{2} + \frac{1}{10n},$$  

where $P$ is of size $2^{O(n)} = n^2$, $T_i$ is of size $n$, and $P[T_1, \ldots, T_{i-1}]$ is of size $O(n^3)$. Thus, $f$ is approximatable by a circuit of size $n^3$. Which is a contradiction because $f$ is $(2^n, \frac{1}{10})$-hard.

Average Case versus Worse Case
So far, we have shown that average case hardness translates into pseudorandomness. But what can we say about worst case hardness? In the case of a Permanent (See Lecture 15 and Lecture 16), worst case hardness is equivalent to average case harness. Thus, if there exists no circuit of size at most $2^n$ that computes the Permanent, then $\text{BPP} = \text{P}$.

Impagliazzo, Widgerson, Trevisan, Sudan, and Vadhan show how to convert a worst case hardness function $f \in \text{EXP}$ into another function $f' \in \text{EXP}$ which is average case hard, using error-correcting codes. Thus, we conclude that worst-case hardness also translates into pseudorandomness.

How close are we to prove $\text{BPP} = \text{P}$?
We have shown that proving circuit lower bound imply derandomization. Similarly, if $\text{NEXP}$ has some form of circuit lower bounds, then $\text{AM} = \text{NP}$, IKW prove a weaker converse: $\text{AM} \neq \text{NEXP} \Leftrightarrow \text{NEXP}$ is not $\subseteq \text{NP/Poly}$
Alternate Proof of $\text{BPP} \in \Sigma_2$

The NW paradigm indicate that BPP is no harder than finding a hard function. This gives the following $\Sigma_2$ algorithm for BPP:

- Guess a hard function $f : \{0,1\}^l \longrightarrow \{0,1\}$.
- For all circuits $C$ of size at most $2^c$, check that $C$ does not approximate $f$ on more than $\frac{1}{2} + \epsilon$ fraction of inputs.
- Use $f$ to construct $NW_{f,s}$ and use this PRG to derandomize the BPP algorithm.