Notes for Lecture 11

In this lecture we will begin with the proof of the Nisan-Wigderson Theorem that we stated last time.

**Theorem 1 (Nisan-Wigderson)** Suppose there is a language $L$ decidable in time $2^{O(n)}$ and there is $\delta > 0$ such that $L$ is $(2^{\delta n}, \frac{1}{2^{\delta n}})$-hard on inputs of length $n$. Then ultimate pseudorandom generators exist.

**Proof:**

Fix input length $n$. Completely analogous to the Blum-Micali-Yao pseudorandom generator of stretch $n+1$, we will first show that $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ such that $x \mapsto x, f(x)$ is $(2^{\delta n}, \frac{1}{2^{\delta n}})$-pseudorandom.

Assume, towards contradiction, that there is a circuit $\Delta$ of size $S \leq 2^{\delta n}$ such that for $\epsilon \geq \frac{1}{2^{\delta n}}$ we have:

$$\left| Pr_{x \sim \{0,1\}^n, b \sim \{0,1\}} [\Delta(x, b) = 1] - Pr[\Delta(x, f(x) = 1] \right| \geq \epsilon$$

Then one of the following circuits: $\Delta(x, 0), \Delta(x, 1), \Delta(x, 0)^\perp, \Delta(x, 1)^\perp$ computes $f$ on $\geq 1/2 + \epsilon$ fraction of the inputs. Therefore, the following algorithm computes $f$ in $\geq 1/2 + \epsilon$ fraction of the inputs:

**Algorithm A (x)**

Choose uniformly at random $b$

if $\Delta(x, b) = 1$ output $b$ else output $\overline{b}$

It follows that $Pr_{x,b}[A(x) = f(x)] > 1/2 + \epsilon$ which is a contradiction. Therefore $G$ is a $(S, \epsilon)$ pseudorandom generator with stretch $n+1$.

In order to construct the ultimate generator, we need to have stretch $N = 2^{O(n)}$. However, we cannot use the same construction as in the B-M-Y pseudorandom generator $G^N$, because we need to compute $f(x)$ $N$ times. In the Nisan-Wigderson case, $f$ is computed in time $2^{O(n)}$ but we can only have distinguishers of size $2^{\delta n}$. What we do instead can be illustrated in the above figure.

The main idea of the constructions lies in the fact that function $f$ evaluated in a random input may be hard to compute, but evaluated in correlated inputs may be easier. Formally, we give the following construction:

**Construction of $G$ from $O(n) = t$ - bit random input $z$ and form $f : (S, \epsilon)$ - hard.**
We first construct $N$ subsets of $\{1, \ldots, t\}$, $S_1, \ldots, S_N$. Each one of them will have size $|S_i| = n$ and the intersection of any two of them will be $|S_i \cap S_j| \leq \log N$. The following figure indicates the construction for values $t = 50, n = 30, N = 2^{20}, |S_i| = 30, |S_i \cap S_j| \leq 20$

We choose $N = 2^{\frac{\delta n}{2}}$. We want to prove that if $f$ is $(S, \epsilon)$-hard then the output of the generator is $(S - N^2, \epsilon N)$ - pseudorandom.

Suppose, towards contradiction that there is a circuit $\Delta$ such that 

$$\Pr_z[\Delta(f(x_1)f(x_2)\ldots f(x_N)) = 1] - \Pr[\Delta(r_1, r_2, \ldots, r_N) = 1] \geq \epsilon \quad (1)$$

Consider the following distributions of inputs for $\Delta$:

$f(x_1)f(x_2)\ldots f(x_N)$

$r_1 f(x_2)\ldots f(x_N)$

$\vdots$

$r_1, r_2, \ldots, r_N$

By a hybrid argument, there must be two consecutive distributions such that 

$$\Pr_z[\Delta(r_1, \ldots, r_{i-1}, f(x_i) \ldots f(x_N)) = 1] - \Pr[\Delta(r_1, \ldots, r_{i-1}, f(x_{i+1}) \ldots f(x_N)) = 1] \geq \epsilon / N \quad (**)$$

Consider the following algorithm $A$ which takes input $x$ and $b$ and wants to distinguish whether $b = f(x)$ or $b$ is a random bit.

Algorithm $A(x, b)$

1. Define $z \in \{0, 1\}^t$ such that $z|_{S_i} = x$ and $z_{\{1, \ldots, t\} - S_i}$ is random.
2. Compute $x_1 = z|_{S_1}, x_2 = z|_{S_2}, \ldots, x_N = z|_{S_N}$
3. Pick at random $r_1, \ldots, r_{i-1}$
4. output $\Delta(r_1, \ldots, r_{i-1}, b, f(x_{i+1}) \ldots f(x_N))$

If we could show that 

$$\Pr_{x \sim \{0, 1\}^n, \text{randomness of } A}[A(x, f(x)) = 1] - \Pr_{x \sim \{0, 1\}^n, \text{randomness of } A, r \sim \{0, 1\}}[A(x, r) = 1] \geq \epsilon / N \quad (**)$$

Then $f$ is not $(\text{size of } A, \epsilon / N)$ - hard.

The problem with this idea is that we will need to compute $f(x_{i+1}), \ldots, f(x_N)$ so the size of $A$ will be bigger than the size of a circuit that computes $f$, therefore we could distinguish $f$ from $b$ just by computing $f(x)$ from scratch. The above difficulty can be overcome with the following idea: since $A$ probabilistic, there is a choice of randomness $z_{\{1, \ldots, t\} - S_i}$ (consider the best possible), such that the distinguishing probability is still $> \epsilon / N$. Therefore, we can fix this randomness and hardwire it to the circuit. More precisely, in the new algorithm we have:
\[ z|_{S_i} = x \]
\[ z|_{\{1,...,t\} - S_i} \] is a good choice of randomness.

For the rest of \( z|_{S_j} \) we have some fixed bits \((t - n)\) total and some bits \((n)\) total that belong to \( x \).

To summarize, in each \( z|_{S_j} \) we have \( \leq \log N \) bits of \( x \) and \( \geq n - \log N \) constants. We therefore define the following functions that depend only on \( k = \log N \) bits of \( x \):

\[
\begin{align*}
\hat{f}(x_{i+1}) &= g_{i+1}(x) \\
\vdots \\
\hat{f}(x_N) &= g_N(x)
\end{align*}
\]

Since \( g_j \) depends only on \( k \) bits, it can be computed by a circuit of size \( O(2^k) = O(N) \). Therefore, size of \( A = \text{size of } \Delta + O(N^2) \) and we conclude that if the generator is not \((S,\epsilon)\) - pseudorandom then \( f \) is not \((\text{size of } \Delta + O(N^2), \epsilon/N)\)-hard. By assumption, \( f \) is \((2^{bn}, \frac{1}{2^{2n}})\) - hard so taking \( N = c \cdot 2^{bn/2} \), \( S = 1/2 \cdot 2^{bn} \) and \( \epsilon = \frac{c'}{2^{bn/2}} \), we can see that Algorithm A is a distinguisher for \( f \), reaching the desired contradiction. In the following lecture, we will see how to construct the \( S_i \). \( \square \)