6.841 Advanced Complexity Theory	May 04, 2005
Lecture 24	
Lecturer: Madhu Sudan	*Scribe: Arnab Bhattacharyya*

Lecture Topics

- Lectures 22-23 comments
- Extracting randomness
- Average-case complexity

1 Response to comments on lectures 22-23

- For much more on pseudorandomness, take Prof. Vadhan's class called "Pseudorandomness" at Harvard.
- With regards to how well derandomization works in practice, we know that some UNIX "random number generators" can be and have been fooled. Such issues are especially important in cryptography. Also sometimes, we think that we have a PRG but we haven't been able to prove it. For example, the DES encryption process is believed to contain a pseudorandom generator but this has not been proved.
- The Shaltiel-Umans PRG is an alternative to Nisan-Wigderson's. It is based on the ideas of the original Blum, Micali, Yao PRG.
- Kolmogorov's idea of randomness can be formulated more precisely as follows. A sequence of bits $b = b_1 b_2 \dots b_i \dots$ is c-random if for all TM's M of size c, M doesn't compute[1] the sequence b. A sequence is Kolmogorov-random if it is not c-random for all c. For example, the digits of π do not form a Kolmogorov random sequence because a TM can easily compute this particular sequence. Investigations have also been made into computationally-bounded Kolmogorov randomness. It is not clear how one can extract uniform randomness out of Kolmogorov-random or computationally-bound Kolmogorov random sequences.
- It is actually not very surprising that the Nisan-Wigderson PRG is so simple to describe. Most of the constructions that we have described in this class have ultimately reduced to a combination of a few simple ideas. Also, many of the combinatorial objects that we consider in derandomization, say, also turn out to be useful in various other areas, such as error-correcting codes. At a high-level, often times, we are searching for combinatorial objects with the same properties for different applications.

2 Extractors

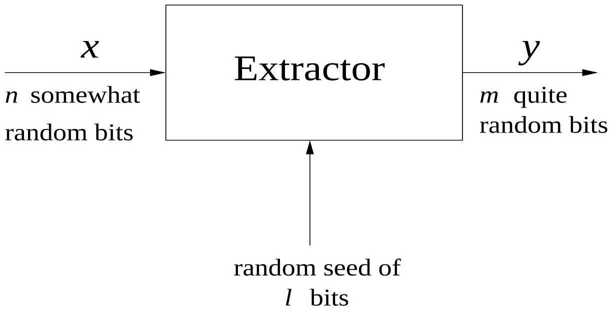
A physical process which we would call 'random' produces sequences of bits that are not predictable. For example, a radioactive source emits particles in an unpredictable manner; however, most of the time, it might emit nothing and so, the emitted sequence is mostly zeroes with a few ones thrown in unpredictably. In general, a random sequence of bits from a physical process is unpredictable but not uniform or independent. We will make these notions more precise shortly. On the other hand, for a random sequence to work for our BPP algorithm, we need it to be uniform and independent. An

[1] For this to make sense, assume that M writes the sequence on a write-only tape

extractor is an algorithm that accomplishes this conversion from the weak kind of randomness provided by physics to the strong kind of randomness needed for randomized algorithms.

Let's first consider how one might model a weak source of randomness. Von Neumann considered a sequence of random bits $b_1 b_2 \dots b_n \in \{0,1\}^n$ such that $Pr[b_i = 0] = p$ where p is unknown but constant; hence, the weak source produces a sequence of independent but biased random bits. Then, Von Neumann gave the following extractor to convert this source into a source appropriate for our BPP algorithms: take pairs of the weakly random bits, and output 0 if the pair is 01, output 1 if the pair is 10, and do nothing for the other cases. It is clear that this simple trick gets us an unbiased and independent source of random bits.

For a more general extraction procedure, we change our setting by allowing the extractor to take as input a small number of truly random bits. We can think of this seed as a random choice of an extractor from a small family of extractors. Then, our new picture of the extraction algorithm looks like this:



So, formally, an extractor is a map from $\{0,1\}^n \times \{0,1\}^l$ to $\{0,1\}^m$. Next, we define what we mean by "quite random" and "somewhat random":

- quite random: Output distribution, D', is statistically close to the uniform distribution, U_m:

$$2|D' - U_m| = \sum_{y \in \{0,1\}^m} |D'(y) - \frac{1}{2^m}| \leq 2\epsilon$$

- somewhat random: The input distribution D has min-entropy k, that is, k is the largest integer such that $D(x) \leq 2^{-k}$ for all $x \in \{0,1\}^n$.

We note that the above definition of "somewhat random" was not arrived at immediately but required several years of experimentation with definitions.

The following non-constructive result can be shown using the probabilistic method:

Theorem 1 *There exists an extractor* $Ext : \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^m$ *with* $l = O(\log n)$ *such that if the input distribution* D *has* $k = m(1 + o(1))$ *bits of min-entropy, then the output is* ϵ*-close to* U_m

Trevisan proved the following constructive result:

Theorem 2 *Suppose we have a source which emits* n*-bit strings with* $n^{.01}$ *bits of entropy. Then, we can extract* $m = n^{.001}$ *"quite random" bits while investing* $O(\log n)$ *bits in the seed* σ *for* $\epsilon = \frac{1}{100}$ *(say).*

Proof Idea: The proof is very novel in character. Equivalent to saying that $x \in \{0,1\}^n$ is giving a function $f_x : \{0,1\}^{\log n} \to \{0,1\}$. Now, most functions f_x's are hard to compute in the non-uniform sense. Using this family of hard functions, Trevisan uses the Impagliazzo-Wigderson theorem to construct the PRG $G_{IW}^{f_x} : \{0,1\}^l \to \{0,1\}^m$ and defines the extractor Ext to be $Ext(x, \sigma) = G_{IW}^{f_x}(\sigma)$. The analysis for the Impagliazzo-Wigderson result shows that this works. We do not go into more details.

3 Average case complexity

The notion of average-case complexity is motivated by cryptography. In the formalism we have considered so far in complexity theory, we have only looked at worst-case complexity. By saying that a problem is hard, we have meant that there is an instance of the problem for which we have no polynomial-time algorithm. But this is not enough to make cryptography feasible. Consider, for example, the following cryptographic protocol: I pick two integers p, q, compute $N = pq$, and I announce N to the whole world, hoping/assuming that it is too hard to factor N. But this factoring problem is hard only for specific choices of p and q; for example, if p is 2, then problem is trivial.

To incorporate the idea that the average-case complexity of a problem depends upon the distribution of its inputs, Levin defined the following:

- A computational problem is a pair (π, D) where π is the problem, "Given x, find y such that $\pi(x, y)$ holds," and $D = \{D_n\}$ where D_n is a distribution on all strings of length n.

We say that a computational problem (π, D) is easy if there exists an algorithm A and a $\delta > 0$ such that for all input x, A solves π correctly and

$$\sum_{y \in \Sigma^{|x|}} [D_{|x|}(y) \cdot T(y)^{\delta}] = poly(|x|)$$

where $T(y)$ denotes the number of steps carried out by A on input y. Then, we say that $(\pi, D) \in Avg$-P.

Initially, the above definition for expected polynomial-time looks rather strange. To motivate this, consider the following situation. Suppose D_n is the uniform distribution and let $t(x) = 2^n$ if $x = 0^n$ and $t(x) = 1$ otherwise. Then, $\boldsymbol{E}[t(x)]$ is $O(1)$. Now consider what happens when, because of the application of a reduction, n is replaced by $2n$; then $\boldsymbol{E}[t(x)]$ is $O(2^n)$. The definition given above remains invariant under such compositions.

In the next lecture, we will discuss more about average-case hardness, distributions, reductions, and completeness results.