Today

- Computational Pseudorandomness.
- Blum-Micali-Yao paradigm: Based on 1-way functions.
- Nisan-Wigderson paradigm: Based on hard functions.

What is random?

- Initially: Sequence of random bits, independent, uniform are random. Nothing else is.
- Shannon: Uniformity not necssary. Independence not necessary. Can attribute to any distribution an entropy which measures amount of randomness in it. Roughly - uniform distribution on $S \subseteq \{0,1\}^n$ has $\log_2 |S|$ bits of entropy. "Large sets are be random; small sets aren't".
- Kolmogorov-Chaitin-Solomonoff: "Random is what can't be described": If finite TM produces an infinite sequence, then

sequence is not random, else it is. What about finite sequences?

- Blum-Micali-Yao: Random is when can't predict the next bit in time polynomial in n. Equivalently, set is pseudorandom if no polynomial time algorithm behaves differently on string than on uniform distribution.

BMY: Strings vs. Sets of Strings

- Roughly: A fixed finite length string can't be random in any meaningful sense.
- However a long string generated from a short random seed can appear random to some.
- What is random?
  - If you can't distinguish given distribution from random, then distribution is pseudo-random to you.
  - if You = { Class of all polytime algorithms (circuits) }, then distribution is pseudo-random.

BMY: Indistinguishability as random

- Distributions D_1 and D_2 are ϵ-indistinguishable to Boolean A if

$$|\Pr_{x \leftarrow D_1}[A(x) = 1] - \Pr_{x \leftarrow D_2}[A(x) = 1]| \leq \epsilon.$$

- Distributions D_1 and D_2 are statistically indistinguishable if they are $1/p(n)$ indistinguishable to every A and every polynomial p.

- Distributions D_1 and D_2 are computationally indistinguishable if they are $1/p(n)$ indistinguishable to every polytime computable function (poly size circuit) A and every polynomial p.

- BMY Notion: D_1 is pseudo-random if it is computationally indistinguishable from uniform distribution.

BMY: Pseudorandom generators

- $G : \{0,1\}^s \to \{0,1\}^n$ is a pseudorandom-generator if $\{G(s)\}_s$ is computationally pseudo-random and G is polytime computable in input.

- Note G is easy, but G^{-1} hard. Thus prg needs $NP \neq P$. Even more! $DNP \neq Avg - P$.

- Focus on polynomial length stretching, not more.

BMY: Alternately, Unpredictable is random

- ith bit of G is δ-unpredictable to A if $\Pr_s[G(s)[i+1] = A(G(s)[1,\ldots,i])] \leq \frac{1}{2} + \delta$.

- G pseudorandom if for all i and for all prob. poly time A, and all $\delta = 1/p(n)$, ith bit of G is δ-unpredictable to A.

- Thm: Two defns are equivalent.

- Proof. One direction obvious. Other direction is hybridization + case analysis.

Consequence: 1-bit stretcher suffices

- Let G map s bits to $s+1$.
- Will construct prg mapping s bits to n from this.
- Let $S_0 = S$ be initial seed. Let $x_i = G(S_{i-1})$ and let $S_i =$ first s bits of x_i and let $y_i =$ last bit of x_i. Then the map from S to $y_1 \cdots y_n$ is pseudorandom.
- Proof: Consider a predictor predicting y_i given $y_{i+1} \cdots y_n$ (Aha! Reversing the output). Then the predictor can also predict y_i given S_i (since $y_{i+1} \cdots y_n$ can be computed from S_i). But this is predicting the last bit in the ith application of G!

Constructions & Applications

- First notice we didn't really need G to be pseudo-random, only that its last bit be unpredictable given the first s.
- Blum-Micali: Prove that the map $G : (p, g, x) \mapsto (p, g, g^x (\bmod\ p), \mathrm{msb}(x))$ satisfies this property if we believe discrete log. to be hard. Use this to construct prg.
- Applications: Mostly in cryptography. Often easy to show that "knowledge" is not leaked by some string, by showing it is computationally pseudorandom.
- Our quest: Complexity-theoretic use. Use pseudo-randomness to show BPP=P. First steps by Yao. Later Nisan-Wigderson.

Nisan-Wigderson paradigm

- $G : \{0,1\}^s \to \{0,1\}^n$ is a pseudorandom-generator if $\{G(s)\}_s$ is computationally pseudo-random to circuits of size n and G is polytime computable in output.
- Now don't need to show NP $\neq$ P! Still need to show some function in, say, time(n^2) does not have size n circuits.
- Main theorem: Suffices to have such functions.
  - Step 1: If function in E is hard on average for subexp. circuits then BPP=P.
  - Step 2: If function in E is hard on worst-case for subexp. circuits then there exists function in E is hard on average.