BPP

- Next: further explore the relationship between randomness and nonuniformity

- Main tool: pseudo-random generators

Derandomization

- Goal: try to simulate BPP in subexponential time (or better)
- Use Pseudo-Random Generator (PRG):

  ![Diagram showing seed to output string transformation]

- Often: PRG "good" if it passes (ad-hoc) statistical tests

Simulating BPP using PRGs

- Recall: $L \in \text{BPP}$ implies exists p.p.t. TM $M$
  - $x \in L \Rightarrow \Pr_p[M(x, y) \text{ accepts}] \geq 2/3$
  - $x \notin L \Rightarrow \Pr_p[M(x, y) \text{ rejects}] \geq 2/3$
- Given an input $x$:
  - Convert $M$ into circuit $C(x, y)$
  - Simplification: pad $y$ so that $|C| = |y| = m$
  - Hardwire input $x$ to get circuit $C_x$
    - $\Pr_p[C_x(y) = 1] \geq 2/3$ ("yes")
    - $\Pr_p[C_x(y) = 1] \leq 1/3$ ("no")

Simulating BPP using PRGs

- Use a PRG $G$ with
  - Output length $m$
  - Seed length $t \ll m$
  - Error $\epsilon < 1/6$
  - Fooling size $s = m$

- Compute $\Pr_p[C_x(G(z)) = 1]$ exactly
  - Evaluate $C_x(G(z))$ on every seed $z \in \{0, 1\}^t$
- Running time $(O(m) + \text{time for } G)2^t$
Simulating BPP using PRGs

- knowing \( \Pr[C_x(G(z)) = 1] \), can distinguish between two cases:

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1/3</th>
<th>1/2</th>
<th>2/3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;yes&quot;:</td>
<td></td>
<td></td>
<td>( \varepsilon )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;no&quot;:</td>
<td></td>
<td></td>
<td></td>
<td>( \varepsilon )</td>
<td></td>
</tr>
</tbody>
</table>
```

Blum-Micali-Yao PRG

- Initial goal: for all \( 1 > \delta > 0 \), we will build a family of PRGs \( \{G_m\} \) with:
  - output length \( m \)
  - fooling size \( s = m \)
  - seed length \( t = m^\delta \)
  - running time \( m^c \)
  - error \( \varepsilon < 1/6 \)

- implies: \( \text{BPP} \subseteq \bigcap_{\delta > 0} \text{TIME}(2^{m^\delta}) \not\subseteq \text{EXP} \)

- Why? simulation runs in time \( O(m + m^c)(2^{m^\delta}) = O(2^{m^{2\delta}}) = O(2^{n^{2k\delta}}) \)

Blum-Micali-Yao PRG

- believe one-way functions exist
  - e.g. integer multiplication, discrete log, RSA (w/ minor modifications)

**Definition:** One Way Function (OWF):
- function family \( f = \{f_n\} : \{0,1\}^n \rightarrow \{0,1\}^n \)
- \( f_n \) computable in poly\( (n) \) time
- for every family of poly-size circuits \( \{C_n\} \):
  \[ \Pr[C_n(f_n(x)) \in f_n^{-1}(f_n(x))] \leq \varepsilon(n) \]
- \( \varepsilon(n) = o(n^{-c}) \) for all \( c \)

First attempt

- attempt at PRG from OWP \( f \):
  - \( t = m^\delta \)
  - \( y_0 \in \{0,1\}^t \)
  - \( y_i = f_i(y_{i-1}) \)
  - \( G(y_0) = y_0y_1y_2y_3 \cdots y_0 \)
  - \( k = m/t \)
  - computable in time at most \( kt^c < m^{t-1} = m^c \)

**First attempt**

- output is "unpredictable":
  - no poly-size circuit \( C \) can output \( y_{i+1} \) given \( y_{i+1}y_{i+2}y_{i+3} \cdots y_i \) with non-negl. success prob.
  - if \( C \) could, then given \( y_i \) can compute \( y_{i+1}, y_{i+2}, y_{i+3}, \cdots, y_{i+k} \) and feed to \( C \)
  - result is poly-size circuit to compute \( y_{i+1} = f_i^{-1}(y_i) \) from \( y_i \)
  - note: we’re using that \( f_i \) is 1-1
First attempt

- one problem:
  - hard to compute $y_{i-1}$ from $y_i$
  - but might be easy to compute single bit (or several bits) of $y_{i-1}$ from $y_i$
  - could use to build small circuit $C$ that distinguishes $G$’s output from uniform distribution on $\{0,1\}^m$

Hard bits

- If $\{f_n\}$ is one-way permutation we know:
  - no poly-size circuit can compute $f_n^{-1}(y)$ from $y$ with non-negligible success probability
    \[
    \Pr[C_n(y) = f_n^{-1}(y)] \leq \epsilon(n)
    \]
  - We want to identify a single bit position $j$ for which:
    - no poly-size circuit can compute $(f_n^{-1}(x))_j$ from $x$ with non-negligible advantage over a coin flip
      \[
      \Pr[C_n(x) = (f_n^{-1}(y))_j] \leq \frac{1}{2} + \epsilon(n)
      \]

Hard bits

**Definition:** hard bit for $g = \{g_n\}$ is family $h = \{h_n\}$, $h_n : \{0,1\}^n \rightarrow \{0,1\}$ (more general: function $h_n : \{0,1\}^n \rightarrow \{0,1\}$)

- For some specific functions $f$ we know of such a bit position $j$
- More general:
  - function $h_n : \{0,1\}^n \rightarrow \{0,1\}$ rather than just a bit position $j$. 

Hard bits

- If $\{f_n\}$ is one-way permutation we know:
  - no poly-size circuit can compute $f_n^{-1}(y)$ from $y$ with non-negligible success probability
    \[
    \Pr[C_n(y) = f_n^{-1}(y)] \leq \epsilon(n)
    \]
  - We want to identify a single bit position $j$ for which:
    - no poly-size circuit can compute $(f_n^{-1}(x))_j$ from $x$ with non-negligible advantage over a coin flip
      \[
      \Pr[C_n(x) = (f_n^{-1}(y))_j] \leq \frac{1}{2} + \epsilon(n)
      \]

Hard bits

**Definition:** hard bit for $g = \{g_n\}$ is family $h = \{h_n\}$, $h_n : \{0,1\}^n \rightarrow \{0,1\}$ such that if circuit family $\{C_n\}$ of size $s(n)$ achieves:

\[
\Pr[C_n(x) = h_n(g_n(x))] \geq \frac{1}{2} + \epsilon(n)
\]

then there is a circuit family $\{C'_n\}$ of size $s'(n)$ that achieves:

\[
\Pr[C'_n(x) = g_n(x)] \geq \epsilon'(n)
\]

with:
- $\epsilon'(n) = \frac{\epsilon(n)n^{O(1)}}{n}$
- $s'(n) = (s(n)n^{O(1)})^{O(1)}$
Goldreich-Levin

• To get a generic hard bit, first need to modify our one-way permutation

• Define \( f'_{n}:\{0,1\}^{n} \times \{0,1\}^{n} \to \{0,1\}^{2n} \) as:

\[
f'_{n}(x,y) = (f_{n}(x), y)
\]

Goldreich-Levin

• Two observations:

\( f'_{n}(x,y) = (f_{n}(x), y) \)

– \( f' = \) \( f_{n} \) if \( f = \) \( f_{n} \)

– if circuit \( C_{n} \) achieves

\[
Pr_{x,y}[C_{n}(x,y) = f'_{n}^{-1}(x,y)] \geq \varepsilon(n)
\]

then for some \( y' \)

\[
Pr_{x}[C_{n}(x,y')=f'_{n}^{-1}(x,y')=(f_{n}^{-1}(x), y')] \geq \varepsilon(n)
\]

and so \( f' = \) one-way permutation if \( f = \).

Goldreich-Levin

• The Goldreich-Levin function:

\( GL_{2n}: \{0,1\}^{n} \times \{0,1\}^{n} \to \{0,1\} \)

is defined by:

\[
GL_{2n}(x,y) = \oplus_{i:y_{i}=1} x_{i}
\]

– parity of subset of bits of \( x \) selected by 1’s of \( y \)

– inner-product of \( n \)-vectors \( x \) and \( y \) in \( GF(2) \)

**Theorem** (G-L): for every function \( f \), \( GL \) is a hard bit for \( f' \). *(proof: problem set)*

Distinguishers and predictors

• Distribution \( D \) on \( \{0,1\}^{n} \)

• \( D \) \( \varepsilon \)-passes statistical tests of size \( s \) if for all circuits of size \( s \):

\[
|Pr_{y \leftarrow U_{n}}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| \leq \varepsilon
\]

– circuit violating this is sometimes called an efficient “distinguisher”

Distinguishers and predictors

• Distribution \( D \) on \( \{0,1\}^{n} \)

• \( D \) \( \varepsilon \)-passes prediction tests of size \( s \) if for all circuits of size \( s' \):

\[
Pr_{y \leftarrow U_{n}}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1] \leq \varepsilon
\]

– circuit violating this is sometimes called an efficient “predictor”

• predictor seems stronger

• Yao showed essentially the same!

\( \text{Theorem} \) (Yao): if a distribution \( D \) on \( \{0,1\}^{n} \)

\( (\varepsilon/n) \)-passes all prediction tests of size \( s' \),

then it \( \varepsilon \)-passes all statistical tests of size \( s' = s - O(n) \).
Distinguishers and predictors

- Proof:
  - idea: proof by contradiction
  - given a size $s$ distinguisher $C$:
    \[
    |Pr_{y \leftarrow U_n}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| > \epsilon
    \]
  - produce size $s$ predictor $P$:
    \[
    Pr_{y \leftarrow D}[P(y_1, 2, \ldots, i-1) = y_i] > \frac{1}{2} + \frac{\epsilon}{n}
    \]
  - work with distributions that are "hybrids" of the uniform distribution $U_n$ and $D$

Distinguishers and predictors

- given a size $s$ distinguisher $C$:
  \[
  |Pr_{y \leftarrow U_n}[C(y) = 1] - Pr_{y \leftarrow D}[C(y) = 1]| > \epsilon
  \]
- define $n+1$ hybrid distributions
  - hybrid distribution $D_i$:
    - sample $b_i = b_1 b_2 \ldots b_i$ from $D$
    - sample $r_i = r_1 r_2 \ldots r_i$ from $U_n$
    - output:
      \[
      b_1 b_2 \ldots b_i r_{i+1} r_{i+2} \ldots r_n
      \]

Distinguishers and predictors

- Hybrid distributions:
  \[
  D_0 = U_n
  
  D_i = D_{i-1} \oplus b_i
  
  D_n = D
  \]

Distinguishers and predictors

- Define: $p_i = Pr_{y \leftarrow D}[C(y) = 1]$
- Note: $p_0 = Pr_{y \leftarrow U_n}[C(y) = 1]$; $p_n = Pr_{y \leftarrow D}[C(y) = 1]$
- by assumption: $\epsilon < |p_n - p_0|$
- triangle inequality: $|p_n - p_0| \leq \sum_{1 \leq i \leq n} |p_i - p_{i-1}|$
- there must be some $i$ for which $|p_i - p_{i-1}| > \epsilon/n$
- WLOG assume $p_i - p_{i-1} > \epsilon/n$
  - can invert output of $C$ if necessary

Distinguishers and predictors

- define distribution $D'_i$ to be $D_i$ with $i$-th bit flipped
  - $p'_i = Pr_{y \leftarrow D}[C(y) = 1]$
- notice:
  \[
  D_{i+1} = (D_i + D'_i)/2 \quad p_{i+1} = (p_i + p'_i)/2
  \]

Distinguishers and predictors

- randomized predictor $P'$ for $i$th bit:
  - input: $u = y_1 y_2 \ldots y_{i-1}$ (which comes from $D$)
  - flip a coin: $d \in \{0,1\}$
  - $w = w_1 w_2 \ldots w_{i+1} \leftarrow U_n$
  - evaluate $C(u d w)$
  - if 1, output $d$; if 0, output $\neg d$

Claim:
\[
Pr_{y \leftarrow D, d, w \leftarrow U_n}[P'(y_1, \ldots, y_i) = y_i] > \frac{1}{2} + \frac{\epsilon}{n}
\]
Distinguishers and predictors

- **P’** is randomized procedure
- there must be some fixing of its random bits \( d, w \) that preserves the success prob.
- final predictor \( P \) has \( d’ \) and \( w’ \) hardwired:
  - Size is \( s’ + O(n) = s \) as promised

\[
\begin{align*}
\text{Distinguishers and predictors} & \\
\text{P’} & \text{is randomized procedure} \\
\text{there must be some fixing of its random} \quad & \text{bits } d, w \text{ that preserves the success prob.} \\
\text{final predictor } P & \text{has } d’ \text{ and } w’ \text{ hardwired:} \\
\text{Size is} & \quad s’ + O(n) = s \\
\end{align*}
\]

\[
\begin{align*}
\text{Proof of claim:} & \\
\Pr[y|w=d] & = 1/2 + \delta/(2p_{n-1}) \\
\Pr[y=0|w=d] & = (1 - p_{n-1})/2(1 - p_{n-1}) \\
\end{align*}
\]

\[
\begin{align*}
\text{Observe:} & \\
\Pr[y_i = d | C(u, d, w) = 1] & = p/(2p_{n-1}) \\
\Pr[y_i = \neg d | C(u, d, w) = 0] & = (1 - p_{n-1})/2(1 - p_{n-1}) \\
\end{align*}
\]

\[
\begin{align*}
\text{Pr}[y_i & = \neg d | C(u, d, w) = 0] = \Pr[y_i = d | C(u, d, w) = 1] \\
& = \Pr[C(u, d, w) = 1 | y_i = d] \Pr[y_i = d] / \Pr[C(u, d, w) = 1] \\
& = \Pr[C(u, d, w) = 1 | y_i = d] \Pr[y_i = d] / p/(2p_{n-1}) \\
\end{align*}
\]

\[
\begin{align*}
\text{The BMY Generator} & \\
\text{Recall goal:} & \text{for all } 1 > \delta > 0, \text{ family of} \\
\text{PRGs } \{G_m\} & \text{ with} \\
\text{output length } m & \quad \text{fooling size } s = m \\
\text{seed length } t & \quad \text{running time } m^c \\
\text{error } \epsilon & \quad < 1/6 \\
\text{If one way permutations exist then WLOG} & \text{there is OWP } f = \{f_n\} \text{ with hard bit } h = \{h_n\} \\
\end{align*}
\]
The BMY Generator

**Theorem** (BMY): for every $\delta > 0$, there is a constant $c$ s.t. for all $d, e$, $G^\delta$ is a PRG with

- error $\varepsilon < 1/m^d$
- fooling size $s = m^e$
- running time $m^c$

- Note: stronger than we needed
  - sufficient to have $\varepsilon < 1/6$; $s = m$

\[ \text{Proof:} \]
- a procedure to compute $h(f^{-1}(y))$
  - set $y_{\text{out}} = y$
  - $b_{i+1} = h_i(y_{\text{out}})$
  - compute $y_i, b_i$ for $j = m+i+1, m+i+2, \ldots, m-1$ as above
  - evaluate $P(b_{m}, b_{m+1}, \ldots, b_{m+1})$
  - $f$ a permutation implies $b_{m+2}, b_{m+3}, \ldots, b_{m_4}$ distributed as (prefix of) output of generator:
  \[ \Pr[P(b_{m+1}, b_{m+2}, \ldots, b_{m_4}) = b_{m+1}] > \frac{1}{2} + 1/m^{d+1} \]

\[ \text{The BMY Generator} \]

Generator $G^\delta = \{G^\delta_m\}$:
- $t = m^\delta$, $y_0 \in \{0,1\}^d$, $y_i = f_i(y_{i-1})$, $b_i = h_i(y_i)$
- $G^\delta_m(y_0) = b_{m+1}b_{m+2}b_{m+3} \ldots b_0$

- transform this **distinguisher** into a predictor $P$ of size $m^d + O(m)$:
  \[ \Pr[P(b_{m+1}, \ldots, b_{m_4}) = b_{m+1}] > \frac{1}{2} + 1/m^{d+1} \]

\[ \text{The BMY Generator} \]

Generator $G^\delta = \{G^\delta_m\}$:
- $t = m^\delta$, $y_0 \in \{0,1\}^d$, $y_i = f_i(y_{i-1})$, $b_i = h_i(y_i)$
- $G^\delta_m(y_0) = b_{m+1}b_{m+2}b_{m+3} \ldots b_0$

- What is $b_{m+1}$?
  \[ b_{m+1} = h_i(y_{m+1}) = h_i(f_i^{-1}(y_{m+1})) \]
  - We have described a family of polynomial-size circuits that computes $h_i(f_i^{-1}(y))$ from $y$ with success greater than $\frac{1}{2} + 1/\text{poly}(m)$
  - Contradiction.