Randomized Complexity Classes

Model: probabilistic TM
Deterministic TM w/ an additional read-only input tape containing coin flips.

$\text{BPP: Bounded-Error Probabilistic Poly-time.}$

Let $\text{BPP} \equiv \{ \text{p.p.t. TM M} \}$

\begin{align*}
    x \in L & \Rightarrow \text{Prob}_{y} [M(x,y) \text{ accepts}] \geq \frac{2}{3} \\
    x \notin L & \Rightarrow \text{Prob}_{y} [M(x,y) \text{ rejects}] \geq \frac{2}{3}
\end{align*}

$\text{RP:}$

\begin{align*}
    x \in L & \Rightarrow \text{Prob}_{y} [M(x,y) \text{ accepts}] \geq \frac{1}{2} \\
    x \notin L & \Rightarrow \text{Prob}_{y} [M(x,y) \text{ rejects}] = 1
\end{align*}

$\text{ZPP: (zero error poly-time)} \quad \text{ZPP} = \text{RP} \cap \text{co-RP}$

\begin{align*}
    \text{Prob}_{y} [M(x,y) \text{ outputs "fail"}] & \leq \frac{1}{2} \\
    \text{otherwise it outputs the correct answer.}
\end{align*}

→ Or runs in expected poly time and always produces the right answer.

These classes may better capture "efficiently computable" better than $P$.

→ The $\frac{1}{2}$ in the defn of ZPP, RP, co-RP can be replaced with any $\frac{1}{\text{poly}(n)}$.

→ The $\frac{2}{3}$ in the defn of BPP can be replaced with any $\frac{1}{2} + \frac{1}{\text{poly}(n)}$ (via Error reduction).

Suppose we have $L, \text{p.p.t. M}$

\begin{align*}
    x \in L & \Rightarrow \text{Prob}_{y} [M \text{ accepts}] \geq \frac{2}{3} \\
    x \notin L & \Rightarrow \text{Prob}_{y} [M \text{ rejects}] = \frac{1}{2}
\end{align*}
Simulate $M \frac{k}{\varepsilon^2}$ times, each time with independent coin flips.
- Accept if any simulation accepts
- O.w. reject.

If $x \in L$, prob a given simulation "bad" $\leq (1 - \varepsilon)^{k/6} \sim e^{-k}$
Prob $M$ accepts $\geq 1 - e^{-k}$

If $x \notin L$, prob $M'$ rejects $= 1$.

Error reduction for BPP:
$x \in L$ $\Pr[M$ accepts $] \geq \frac{1}{2} + \varepsilon$
$x \notin L$ $\Pr[M$ rejects $] \geq \frac{1}{2} + \varepsilon$

Simulate $M \frac{k}{\varepsilon^2}$ times with independent coin flips.
Take the majority answer.

$X_i$: random variable $= 1$ if $i^{th}$ answer is correct
$= 0$ otherwise.

$\Pr[X_i = 0] \leq \frac{1}{2} - \varepsilon$ $\Pr[X_i = 1] \geq \frac{1}{2} + \varepsilon$
$E[X_i] = \frac{1}{2} + \varepsilon$.

$X_i$'s are mutually independent
$X = \sum X_i$ $\mu = E[X] = \left(\frac{1}{2} + \varepsilon\right) \frac{k}{\varepsilon^2}$  $m = \frac{k}{\varepsilon^2}$

Chernoff's Inequality says $\Pr[X \leq m/2] \leq \frac{2^{-\frac{m^2}{2(\varepsilon^2m)}}}{2^{-\frac{m^2}{2\varepsilon^2}}}$

As long as $\varepsilon > \frac{1}{\poly(n)}$ and $k = O(\poly(n))$,
The running time is polynomial and error exp small.
RP, co-RP, BPP, ZPP are all contained in P
(you can always just ignore the random string).

They are also all contained in PSPACE:

\[ P_{\text {accept}} = \frac{\# \text{ y's } M(x,y) = \text{acc}}{\# \text{ all possible } y}. \]

Also RP \subseteq NP (and co-RP \subseteq co-NP)

An NTM can guess y, then compute M(x,y)

\[ \begin{align*}
&\text{if } y \in L \quad M(x,y) = \text{reject} \\
&\text{if } x \in L \quad \text{for at least half of the } y \text{'s } M(x,y) = \text{accept} \\
\text{Exp} &
\end{align*} \]

PSPACE \supset \text{NP, BPP, co-NP, EXP, RP, co-RP, P}

How powerful is BPP?

We have an example of a problem in BPP
that we only know how to solve in EXP.

Not known if BPP = EXP (or even NEXP)
Strong hints that BPP \neq EXP however.
Is there a deterministic simulation of BPP that does better than brute-force search? 

Yes, if we allow non-uniformity.

**Theorem**

\[ \text{BPP} \subseteq \text{P/poly} \quad \text{(Adleman)} \]

**Take** \( L \in \text{BPP} \)

Error reduction gives TM \( M \) s.t.

\[
\begin{align*}
\text{if } x \in L \quad &|x| = n \quad \Pr_y [M(x,y) \text{ accepts}] \geq 1 - (\frac{1}{2})^n \\
\text{if } x \notin L \quad &|x| = n \quad \Pr_y [M(x,y) \text{ rejects}] \geq 1 - (\frac{1}{2})^n
\end{align*}
\]

\( y \) is "bad" for \( x \) if \( M(x,y) \) gives the wrong answer.

**Fix** \( x \)

\[
\begin{align*}
\Pr_y [y \text{ is bad for } x] &\leq (\frac{1}{2})^n \\
\Pr_y [y \text{ is bad for some } x] &\leq 2^n (\frac{1}{2})^n < 1
\end{align*}
\]

\( y \) for which \( y \) is good for all inputs \( x \) of length \( n \).
This \( y \) is the hint for inputs of length \( n \).

( hard code \( y \) into \( L_n \).)

\[ \Rightarrow \text{ if } \text{BPP} = \text{EXP} \quad \text{then } \text{EXP} \subseteq \text{P/poly}. \]

**Does BPP have complete problems?**

Determining if a TM \( M \) is an NTM is easy.
Determining if a TM \( M \) is in \( \text{BPTIME} \) is undecidable since if requires that every string is accepted w/ probability \( \leq 1/3 \) or \( \geq 2/3 \).
A natural candidate for a BPP-complete language would be \((M, x, 1^t)\): \(M\) accepts \(x\) w.p. \(\geq 2/3\) in time \(t\). This problem is BPP-hard.

However: is it in BPP? A BPP machine can't just simulate \(M\) on input \(x\) because it could be that \(M\) accepts w/ prob \(1/2\) on input \(x\).

However if BPP = P (conjectured to be true) then it does have complete problems because P does.

Next: try to de-randomize BPP by pseudo-random generators.

Simulate BPP in subexponential time or better.

**Pseudo-random Generator (PRG)**

\[
\begin{align*}
\text{seed} \quad &\quad \Rightarrow \quad G \quad \Rightarrow \quad \text{output string} \\
+ t \text{ bits} \quad &\quad \quad \text{in bits.}
\end{align*}
\]

\(G\) must be efficiently computable.

Sketches \(t\) into \(m\) bits.

"fools" small circuits. For all \(C\) of size \(\leq s\):

\[
\left| \Pr_{y} \left[ C(y) = 1 \right] - \Pr_{z} \left[ C(1(z)) = 1 \right] \right| \leq \varepsilon.
\]

Simulating BPP w/ a PRG:

Recall: \(L \in \text{BPP} \implies \exists \ p.p.t \ \text{Th} \ M\)

\[
\begin{align*}
x \in L &\quad \implies \Pr_{y} \left[ h(x,y) \ \text{accepts} \right] \geq 2/3 \\
x \notin L &\quad \implies \Pr_{y} \left[ h(x,y) \ \text{rejects} \right] \geq 2/3
\end{align*}
\]
Convert $M$ into a circuit $C(x,y)$

Simplification: pad $y$ s.t. $|C| = |y| = m$.

Hardwire $x$ into circuit to get $C(x,y)$

\[
\Pr_y [C(y) = 1] \geq \frac{2}{3} \quad \text{"yes"}
\]

\[
\Pr_y [C(y) = 1] \leq \frac{2}{3} \quad \text{"no"}
\]

**Pro**: output length: $m$

Seed length: $t \ll m$

error $\varepsilon < \frac{1}{6}$

fooling size $s = m$.

Compute $\Pr_z [C_z (f(z)) = 1]$ exactly.

evaluate $C_x (f(z))$ for every $z \in \{0, 1\}^t$.

running time $(O(m) + \text{time for } f) 2^t$.

This can distinguish between the two cases.

\[
\forall x \in L \quad \Pr_y [C(y) = 1] \geq \frac{2}{3} - \varepsilon > \frac{1}{2}.
\]

\[
\exists x \in L \quad \Pr_y [C(y) = 1] \leq \frac{1}{3} + t < \frac{1}{2}.
\]
Blum-Micali-Yao PRGs:

Initial goal: for all $1 > \varepsilon > 0$ we will build a family of PRGs $G_m$ with:
- Output length $= m$
- Seed length $= t = m^s$
- Fooling size $s = m$
- Running time: $m^c$
- Error: $\varepsilon < \frac{1}{6}$

Implies $\text{BPP} \subseteq \text{NTIME}(2^{n^c}) \subseteq \text{EXP}$

Why? Simulation runs in time:

\[ O \left( (m + m^c) 2^{n^c} \right) = O \left( 2^{n^{2c}} \right) = O \left( 2^{n^{1.5c}} \right) \]

(Note: in order to get $\text{BPP} \subseteq \text{P}$, need $t = O(\log m)$)

Will require some kind of complexity assumption.
(PRGs of this type imply the existence of one-way functions.)

Definition: One Way Function (OWF)

A function family $f = \{ f_m \}_{m \in \mathbb{N}}$:
- $f_m : 0,1^n \rightarrow 0,1^m$
- Each function is computable in polynomial time
- For every polynomial size circuit $C_{n,\delta}$
  \[ \Pr_x \left[ C_{n,\delta}(f(x)) \neq f^{-1}(f(x)) \right] \leq \varepsilon(n) \]
- $\varepsilon(n) = o(1/n^c)$ for all $c$. Note this requires hardness on average which is stronger than worst-case hardness.

It is generally believed that one-way functions exist: (integer multiplication, discrete log, etc.)
Widely used in cryptography.
Definition: One Way Permutation: OWF \( f \)
which is one-to-one.

Can simplify \( \Pr_x [ C_n (f(x)) \in f^{-1}(f_n (x)) ] \leq \epsilon(n) \)

to \( \Pr_y [ C_n (y) = f^{-1} (y) ] \leq \epsilon(n) \).

Here's an attempt at a PRG from an ONF:

\[
\begin{align*}
t &= m^8, & \text{Computable in time} & \quad k t^c < m t^{c-1} = \quad m \quad m^{8(c-1)} = m^c. \\
y_0 & \in \{0,1\}^t \\
y_i &= f_k (y_{i-1}) \\
g (y_0) &= y_{k-1} y_{k-2} \cdots y_0 \\
k &= m/t.
\end{align*}
\]

The output is "unpredictable".

No poly size ckt \( C \) can output \( y_{i-1} \) given \( y_{k-1} \ldots y_i \) w/ non-negligible success prob.

If \( C \) could, then given \( y_i \), compute \( y_{k-1} \ldots y_{i+1} \)

Use \( y_{k-1} \ldots y_i \) to get \( y_{i-1} \).

This would be a ckt to invert \( f \):

\[
f_k^{-1} (y_i) = y_{i-1}
\]

\( \Rightarrow \) the 1-1 assumption makes \( f^{-1} \) unique.

2 Problems:

1. Although it's hard to compute \( y_{i-1} \) from \( y_i \), it may be possible to compute one or more bits of \( y_{i-1} \)

Which could be used to distinguish \( b \)'s output from the uniform distribution over \( \{0,1\}^n \).
This notion of "unpredictability" is not necessarily enough to meet the fooling requirement:
\[ \Pr_y[ C(y) = 1] - \Pr_z[ C(\hat{g}(\hat{z})) = 1] \leq \epsilon. \]

**Hard Bits**

If \( \hat{g}, \hat{z} \) is a one-way permutation, we know that no poly-size circuit can compute \( \hat{f}^{-1}(y) \) from \( y \) w/ non-negligible success prob:
\[ \Pr_y[ C_n(y) = \hat{f}^{-1}(y) ] \leq \epsilon(n) \]

We want to identify a single bit position \( j \) for which:
\[ \text{no poly-size ekt can compute} \ (\hat{f}^{-1}(y))_j \ \text{from} \ y \]
\[ \text{w/ non-negligible advantage over a coin flip.} \]
\[ \Pr_y[ C_n(y) = (\hat{f}^{-1}(y))_j ] \leq \frac{1}{2} + \epsilon(n) \]

For some specific functions we know a bit position \( j \), but would like a more general:
\[ h_n : \{0,1\}^n \rightarrow \{0,1\}^m \]
\[ \text{rather than just a bit position } j. \]

**Definition:** hard bit for \( g = \hat{g}, \hat{z} \) is a family \( h = \{ h_n \} \)
\[ h_n : \{0,1\}^n \rightarrow \{0,1\}^m \] such that if circuit family
\[ \{ C_n \} \] of size \( s(n) \) achieves:
\[ \Pr_y[ C_n(y) = h_n(\hat{g}(\hat{z})) ] \leq \frac{1}{2} + \epsilon(n) \]

Then there is a ekt family \( \{ C'_n \} \) of size \( s'(n) \) that achieves
\[ \Pr_y[ C'_n(y) = g_n(y) ] \geq \epsilon'(n) \]
\[ \epsilon'(n) = (\frac{\epsilon(n)}{\epsilon(n)} )^{o(1)} \]
\[ s'(n) = (s(n) n / \epsilon(n))^{o(1)} \]
In order to get a generic hard bit, we need to modify our one-way permutation.

Define \( f_n : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2 \times \mathbb{Z}_2^n \)

\[
f_n(x, y) = (f_n(x), y).
\]

1. \( f \) is a permutation iff \( f' \) is a permutation.
2. \( f \) is a one-way perm iff \( f' \) is a one-way perm.

Goldreich-Levin function:

\( GL_n : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2 \times \mathbb{Z}_2^n \)

\[
GL_n(x, y) = \bigoplus_{i=1}^{n} x_i \quad (\text{inner product over } GF_2)
\]

\( y \) selects a subset of \( x \)'s bits for parity.

Theorem: (b-L) for every function \( f \), \( GL \) is a hard bit for \( f' \).

We won't prove this here, but let's discuss how it will be used.

We can assume here that if we have a one-way function, then it has a hard bit for \( f' \). (Use the modified one-way perm and the b-L function for the hard bit).

This is what the PRG looks like.
$y_0$ is chosen uniformly from $\{0,1\}^t$.
This yields a distribution over $(b_{n-1} b_{n-2} \ldots b_0)$ $n$-bit strings.

Note that because $f_i$ is a permutation, the distribution is the same if we pick $y_i$ at random and compute

$y_{i-1} \leftarrow y_{i-1} \oplus f_i^{-1}(y_i) \oplus f_{i-1}(y_{i-2}) \oplus \ldots \oplus f_1(y_0)$

This may be difficult to compute, but it is well defined and produces the same distribution as if we start at $y_0$.

We know that there are no poly-sized circuits that can predict $b_{i-1}$ from $y_i$ w/ non-negligible bias away from a random bit:

$$\Pr_{y_i}[C(y_i) = b_{i-1}] = \frac{1}{2} + \epsilon.$$  

Given $y_i$, we can use $f$ and $g^{-1}$ to produce $b_{n-1} b_{n-2} \ldots b_i$.

If $y_i$ is chosen at random, this will be the same induced distribution as if $b_{n-1} \ldots b_0$ is produced starting at $y_0$ and then tossing out $b_i$.

There is no poly-size circuit that can take $b_{n-1} \ldots b_i$ and predict $b_{i-1}$ w/ probability better than a random bit when $b_{n-1} \ldots b_i$ is chosen according to this induced distribution.

Now we need to relate this notion of predictability to distinguishability.
**Distinguishers and Predictors:**

Distribution $D$ on $\{0,1\}^n$

$D$ $\epsilon$-passes statistical tests of size $s$ if for all circuits of size $s$:

$$Pr_{y \leftarrow U} [C(y) = 1] - Pr_{y \leftarrow D} [C(y) = 1] \leq \epsilon$$

A circuit violating this is called an efficient distinguisher.

$D$ $\epsilon$-passes prediction tests of size $s$ if for all circuits of size $s$:

$$Pr_{y \leftarrow D} [C(y_{1, \ldots, y_{s-1}}) = y_s] \leq \frac{1}{2} + \epsilon$$

A circuit violating this is called a predictor.

Having a predictor seems stronger than having a distinguisher.

We have that our distribution has no predictor but we need to be able to say that there is no distinguisher.

Yao showed that these are essentially the same.

**Theorem (Yao):** If a distribution $D$ over $\{0,1\}^n$ $\left( \frac{\epsilon}{n^2} \right)$-passes all prediction tests of size $s$, then it $\epsilon$-passes all statistical tests of size $s' = s - O(n)$.

Proof by contradiction:
given an $\varepsilon'$ distinguisher $C$:

$$\Pr_{y \sim D_0} [C(y) = 1] - \Pr_{y \sim D} [C(y) = 1] > \varepsilon'$$

We will show that there is a predictor $P_i$ (for some $i$)

$$\Pr_{y \sim D} [P(y_{i+1}, \ldots, y_n) = y_i] > \frac{1}{2} + \frac{\varepsilon}{n^2}$$

Consider hybrid distributions between $D$ and $Un$:

$$D_0 = Un \quad D_i \quad D_n = D$$

\text{generate } b_1 \ldots b_n \quad \text{toss and bit: } b_1 \ldots b_n \quad \text{induced by } D \quad \text{uniform}

Let $P_i = \Pr_{y \sim D_i} [C(y) = 1]$

$$p_0 = \Pr_{y \sim Un} [C(y) = 1] \quad p_n = \Pr_{y \sim D} [C(y) = 1]$$

by assumption $|p_n - p_0| > \varepsilon$.

$$\varepsilon < |p_n - p_0| \leq \frac{1}{n} \sum_{i=1}^{n} |P_i - P_{i-1}|$$

$$\Rightarrow \exists i \text{ s.t. } |P_i - P_{i-1}| > \varepsilon/n.$$  (assume w.l.o.g. $P_i > P_{i-1}$ otherwise can just reverse the output).

Let $D_i^* = D_i$ except flip the $i$th bit.

$$P_i' = \Pr_{y \sim D_i^*} [C(y) = 1]$$
\[ \begin{align*}
D_i & : b_1 \ldots b_{i-1} y_i y_{i+1} \ldots y_n \\
D_i^* & : b_1 \ldots b_{i-1} \bar{b}_i y_i \ldots y_n \\
p(x) &= p(x_i \ldots x_{i-1}) z^{-n+i} \\
p(x) &= p(x_i \ldots x_{i-1})p(y_i|x_i, \ldots, x_{i-1})z^{-n+i} \\
p(x) &= p(x_i \ldots x_{i-1})(1 - p(y_i|x_i, \ldots, x_{i-1}))z^{-n+i}
\end{align*} \]

\[ \Rightarrow D_{i-1} = \frac{D_i + D_i^*}{2}. \]

\[ P_{i-1} = \frac{P_i + P_i^*}{2}. \]

**Randomized predictor \( P' \) for \( i \) th bit:**

- **Input:** \( b = b_1 \ldots b_{i-1} \) (generated by \( D \))
- Flip a coin \( d \in \{0,1\} \).
- \( w = w_1 w_2 \ldots w_n \leftarrow U_{n-1} \)
- Evaluate \( C(b, d, w) \)
- If 1 \( \Rightarrow \) output \( d \) \( \; \; \) if 0 \( \Rightarrow \) output \( \bar{d} \).

**Claim:** \( P_{i-1}[P'(b_1 \ldots b_{i-1}) = b_i] > \frac{1}{2} + \epsilon/n. \)

\( b_1 \ldots b_i \leftarrow D \)

\( C \) may need an extra gate.

\( d^* \), \( w^* \)

\( \text{Size is } g' + O(n) = s. \)

\( P' \) is a randomized procedure, we will choose a way to fix the random bits to preserve the probability of success. Call these settings \( d^* \) and \( w^* \). \( P \) has these hardwired.
\[ \Pr_{b_1 \cdots b_{i-1} \in D; \theta, \omega \in \mathcal{U}} \left[ p' (b_1 \cdots b_{i-1}) = b_i \right] = \]

\[ \Pr \left[ b_i = d \mid C(b, d, \omega) = 1 \right] \Pr \left[ C(b, d, \omega) = 1 \right] + \]
\[ \Pr \left[ b_i = 7d \mid C(b, d, \omega) = 0 \right] \Pr \left[ C(b, d, \omega) = 0 \right] \]

\[ \cdot (1 - p_{i-1}) \]

\[ \Pr \left[ b_i = d \mid C(b, d, \omega) = 1 \right] = \frac{\Pr \left[ C(b, d, \omega) = 1 \mid b_i = d \right] \Pr \left[ b_i = d \right]}{\Pr \left[ C(b, d, \omega) = 1 \right]} = p_i \]

\[ \Pr \left[ b_i = 7d \mid C(b, d, \omega) = 0 \right] = \frac{\Pr \left[ C(b, d, \omega) = 0 \mid b_i = 7d \right] \Pr \left[ b_i = 7d \right]}{\Pr \left[ C(b, d, \omega) = 0 \right]} = (1 - p_{i-1}) \]

So:
\[ \Pr_{b_1 \cdots b_{i-1} \in D; \theta, \omega \in \mathcal{U}} \left[ p' (b_1 \cdots b_{i-1}) = b_i \right] = \]

\[ \frac{p_i \cdot (p_{i-1})}{2 (p_{i-1})} + \frac{(1 - p_i) \cdot (1 - p_{i-1})}{2 (1 - p_{i-1})} = \frac{1}{2} + \frac{1}{2} (p_i - p_{i-1}) \]

\[ = \frac{1}{2} + \frac{1}{2} (p_i - p_{i-1}) \]

\[ > \frac{1}{2} + \frac{\epsilon}{2n} \]  

Generator \( G^S = \frac{S}{2} G_m^S \)

\[ t = m^S \quad y_0 = \{0, 1\}^t \quad y_i = f_k (y_{i-1}) \quad b_i = h_k (y_i) \]

\( G_m^S (y_0) = b_{m-1} b_m \ldots b_0 \)
Theorem (BMY) For every $\delta > 0$ there is a constant $c$ s.t. for all $d, e$ $G^e$ is a PRG with

\[ \text{error } \epsilon < 1/m^d \quad ? \]
\[ \text{fooling size } S = m^e \]
\[ \text{running time } m^c \]

Proof: Time to compute $G^e(y_0)$ is $m^{t^c} < m^{c+1}$

Assume $f^S$ does not $(1/m^d)$ pass a statistical test \( C = 3C_{\text{en}} \) of size $m^e$

\[ \left| \Pr_{y \in U_m} [C(y)=1] - \Pr_{z \in D} [C(z)=1] \right| > 1/m^e \]

We can transform this into a predictor $P$ of size $m^e + O(m)$:

\[ \Pr_{z \in D} [P(b_{m-1}, \ldots, b_{m-i}) = b_{m-i+1}] > 1/2 + 1/m^{i+1} \]

We will use this to devise a procedure to compute $h_t(f^{-1}(y))$

Set $y < y_{m-1}$ \quad $b_{m-i} = h_t(y_{m-i})$

Compute $y_j$ for $j = m-i+1, \ldots, m-1$ as above. \quad $b_j = h_t(y_j)$.

Evaluate $P(b_{m-1}, \ldots, b_{m-i}) \xRightarrow{\text{distributed according to the prefix of the generator.}}$
Pr[y \geq \frac{1}{2} + \frac{1}{\text{poly}(m)}]

Initially chosen uniformly

bm-i: \text{ht}(y_{m-i}) = \text{ht}(f^{-1}(y_{m-i})) = \text{ht}(f^{-1}(y))

This is a family of circuits that computes \text{ht}(f^{-1}(y)) from y with success greater than \frac{1}{2} + \frac{1}{\text{poly}(m)}

\Rightarrow \text{Contradiction.}

To get BPP = P need t = O(\log m)

(need to run over all seeds of length t \rightarrow 2^t).

BMY building block one-way \textbf{if} \colon \{0,1\}^t \rightarrow \{0,1\}^t
required to fool circuits of size \text{poly} \textbf{for all e}.
But with these settings \textbf{f} can be inverted by brute force!

BMY generator:

one generator fooling all poly-sized circuits
one-way permutation is a hard function.
implies hard function in NP \cap \text{co-NP}.

Computing \textbf{f}^{-1}(x) is hard
but can show \textbf{f}(y)=x: y is witness

Nisan-Wigderson generator:

for each poly-size bound, a different generator.

hard function can be in \textbf{E} = \bigcup \Sigma^* \text{DTIME}(2^n)
This allows them to get $t = O(\log m)$.

Hardness assumption still average case.
Can be made worst case using error-correcting codes.