Uses of randomness in computation

- algorithms (randomized P) ✓
- "complexity-theoretic" algorithms (randomized NP) ✓
- (circuit) lower bound proofs
- cryptography & pseudorandomness
- derandomization
- learning algorithms
- average-case complexity

Distribution over strings	Single string
$U_n : \forall x \in \{0,1\}^n,$	1000101100111
$U_n(x) = \Pr[x \sim U_n] = \frac{1}{2^n}$	0011011100101
Shannon's Entropy	Kolmogorov Complexity

Shannon's Entropy

$$H(\mathcal{D}_n) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}_n}\left[\log \frac{1}{\mathcal{D}_n(x)}\right]$$

$$= \sum_{x \in \{0,1\}^n} \mathcal{D}_n(x) \cdot \log \frac{1}{\mathcal{D}_n(x)}$$

$$H(U_n) = n$$

"most" random dist.

Kolmogorov Complexity

$K(x) = \min\{\, |(M, w)| :$

TM M on input w prints out x.

$$K(x) \geq |x|$$

"most" random string

Universal Derandomization

Given a BPP algorithm $M(x, r)$,
for every $x \in \{0,1\}^n$,
want to estimate

$$\Pr_{r \sim U_m}\left[M(x,r) \text{ accepts}\right] \pm \varepsilon$$

$(\varepsilon = 1/10)$

with little (no) randomness.

Idea: For x, find some $\mathcal{D}_m$ (of small support), such that

$$\left| \Pr_{r \sim U_m}\left[M(x,r) \text{ accepts} \right] - \underbrace{\Pr_{r \sim \mathcal{D}_m}\left[M(x,r) \text{ accepts} \right]}_{\substack{\text{compute "brute force"} \\ \text{in time } \mathrm{poly}(|x|, |\mathrm{supp}(\mathcal{D}_m)|)}} \right| \leq \varepsilon$$

$\mathcal{D}_m$ ε-fools $M(\underline{x}, \cdot)$ for a given x

More ambitiously: ε-fool $M(x, \cdot)$ for all x

Observe: $M(x,r)$ in time $t(|x|)$

$\Rightarrow \forall x$, $\exists$ circuit $C_x(r)$ of size $\tilde{O}(t(|x|))$

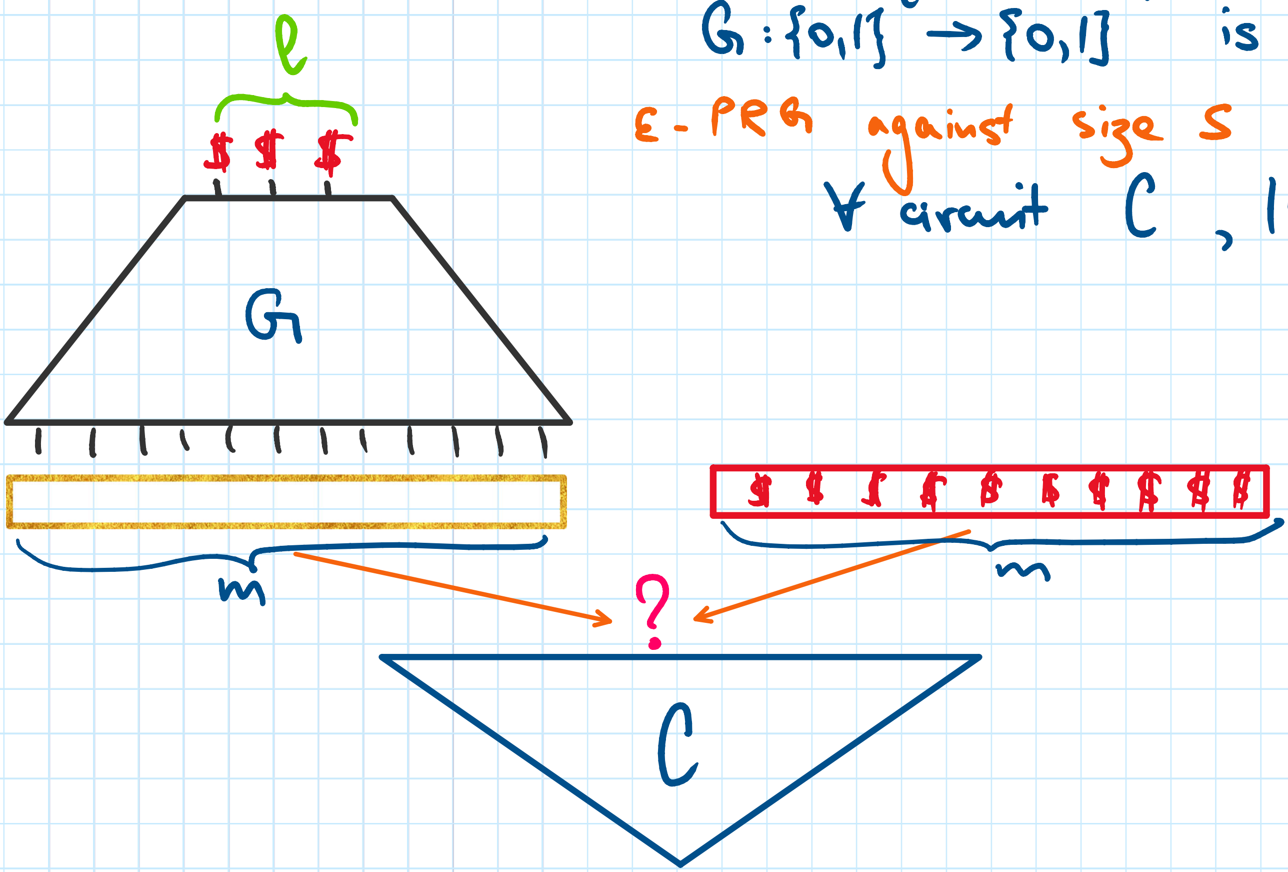
Hence, sufficient to ε-fool all circuits of size $S \leq \tilde{O}(t)$.

Pseudorandom Generator (PRG):

$G : \{0,1\}^\ell \to \{0,1\}^m$ is

ε-PRG against size s if

$\forall$ circuit C, $|C| \le s$



$$\left| \Pr_{r \sim U_\ell}\left[C(G(r)) = 1\right] - \Pr_{w \sim U_m}\left[C(w) = 1\right] \right| \le \varepsilon$$

Also, require G to be computable in time $2^{O(\ell)}$.

To estimate $\Pr_{w \sim U_m}[C(w) = 1]$

compute $\Pr_{r \sim U_\ell}[C(G(r)) = 1]$

in det. time $2^{O(\ell)} \cdot \mathrm{poly}(|C|)$.

Goal: ε-PRG $G : \{0,1\}^\ell \to \{0,1\}^m$
for $\ell \le O(\log(m/\varepsilon))$.

Non-constructively, such a G exists.

Open Question: Construct (any nontrivial) such G.

Why is it so hard to construct a PRG?
(because it implies circuit lower bounds!)

Observation: Suppose have PRG $G : \{0,1\}^\ell \to \{0,1\}^m$
secure against size m circuits.

Define $f : \{0,1\}^{\ell+1} \to \{0,1\}$:

$f \quad \{ \; 0 \quad \text{if} \quad \exists z \in \{0,1\}^\ell \quad G(z)| \qquad = x$

$$f(x) = \begin{cases} 0 & \text{if } \exists z \in \{0,1\}^{\ell} \ \ G(z)\big|_{[1..\ell+1]} = x \\ 1 & \text{otherwise} \end{cases}$$

Note: (1) $f(G(z)) = 0$, $\forall z \in \{0,1\}^{\ell}$

(2) $\Pr_{x \sim U_{\ell+1}} [f(x) = 1] \geq \frac{1}{2}$

Hence, $f \notin \text{Size}[m]$

For $\ell \leq O(\log m)$, $f \notin \text{Size}[2^{o(\ell)}]$

but $f \in \text{Time}[2^{O(\ell)}]$.

So, $\exists$ PRG $G : \{0,1\}^{O(\log m)} \longrightarrow \{0,1\}^{m}$

$\Longrightarrow$ $E \notin \text{Size}[2^{o(n)}]$

(Recall: PIT $\in$ P $\Rightarrow$ circuit lower bounds for NEXP or permanent.)

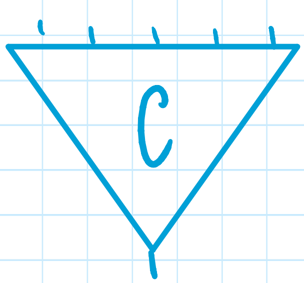
Actually, have an equivalence!

Thm [Impagliazzo, Wigderson '97]:
$\exists$ PRG $G : \{0,1\}^{O(\log m)} \to \{0,1\}^m$ $\iff$
$\exists$ $L \in E$ s.t. L requires circuit size $2^{\Omega(n)}$.

<u>Interpretation</u>: truth table of HARD boolean function

pseudorandom distribution of strings

<u>Intuition</u>: ~~PRG~~ HSG from a Kolmogorov-incompressible string



<u>Assume</u>:
$|C| = m$, on m inputs

$$\Pr_{w \sim U_m}[C(w) = 0] \le \frac{1}{2^{\sqrt{m}}}$$

$\forall\, w \in \{0,1\}^m$, $C(w) = 0 \Rightarrow$

$K(w \mid C) \le \log(2^{m-\sqrt{m}}) + O(1)$

$$K(w \mid C) \leq \log_2\left(2^{m-\sqrt{m}}\right) + O(1)$$
$$\leq m - \sqrt{m} + O(1)$$

So, C must accept a string of high (conditional) Kolmogorov complexity!

$K^{2^{O(m)}}$ suffices, but still ...

Want a much more efficient compression of w if a PRG based on w fails to fool some C.

Want a "small" circuit for w.

Yao's "distinguisher $\to$ predictor" idea:

Yao's "distinguisher → predictor" idea:

Suppose a circuit $C : \{0,1\}^m \to \{0,1\}$

ε-distinguishes a distribution $\mathcal{D}_m$ from $\mathcal{U}_m$:

$$\Pr[C(\mathcal{D}_m)=1] - \Pr[C(\mathcal{U}_m)=1] > \varepsilon$$

Hybrid Argument:

define distribution $H^i = \underbrace{\mathcal{D}_m}_{i} \; \underbrace{\mathcal{U}_m}_{m-i}$

[sample $w \sim \mathcal{D}_m$; output i-length prefix of w, followed by $m-i$ unif. rand. bits]

$$\Pr[C(\mathcal{D}_m)=1] - \Pr[C(\mathcal{U}_m)=1]$$

$$= \Pr[C(H^m)=1] - \Pr[C(H^0)=1]$$

$$= \Pr[C(H^m)] - \cancel{\Pr[C(H^{m-1})]}$$
$$+ \cancel{\Pr[C(H^{m-1})]} - \cancel{\Pr[C(H^{m-2})]}$$
$$+ \quad \cdots$$
$$+ \cancel{\Pr[C(H^1)]} - \Pr[C(H^0)] \quad > \varepsilon$$

By averaging, $\exists\ 1 \le i \le m$ s.t.

$$\Pr[C(H^i)] - \Pr[C(H^{i-1})] > \frac{\varepsilon}{m}$$

$$\underbrace{X_m}_{i}{}^{i}\ \$ \ldots \$ \quad \text{vs.} \quad \underbrace{X_m}_{i-1}\ \$ \ldots \$$$

Predictor for the i^{th} bit of X_m given the previous $(i-1)$ bits $z_1, \ldots z_{i-1}$:

given the previous $(i-1)$ bits $z_1 \dots z_{i-1}$:

$$\boxed{\begin{array}{l} \text{Sample } b_i, b_{i+1}, \dots, b_m \text{ uniformly at random.} \\ \text{If } C(z_1, \dots, z_{i-1}, b_i, b_{i+1}, \dots, b_m) = 1 \\ \text{then output } b_i \\ \text{else output } \neg b_i \end{array}}$$

... is correct with prob. $\geq \frac{1}{2} + \frac{\varepsilon}{m}$

(Exercise.)

NW PRG

For any $f : \{0,1\}^n \to \{0,1\}$

define $G : \{0,1\}^n \to \{0,1\}^{n+1}$:

$$G(z) = z \; f(z)$$

$$z_1 \dots z_n \; f(z)$$

If some $C : \{0,1\}^{n+1} \to \{0,1\}$

ε-distinguishes G from U_{n+1},

ε- distinguishes G from $\mathcal{U}_{n+1}$,
then a circuit of size $\approx |C|$
computes f on $\geq \frac{1}{2} + \frac{\varepsilon}{n+1}$ inputs.

That is, f is "easy" on average.

Why?

Upshot: f is "very hard" on average
$\Rightarrow$ PRG but with only 1-bit stretch.

How to get a PRG with a better stretch?

Idea 1: Direct Product Generator f^k:

$$G(x_1, x_2, \ldots, x_k) = x_1 \, x_2 \ldots x_k \, f(x_1) \, f(x_2) \ldots f(x_k)$$

$$G(x_1, x_2, \dots, x_k) = x_1\, x_2 \dots x_k\, f(x_1)\, f(x_2) \dots f(x_k)$$

where each $x_i \in \{0,1\}^n$.

Still secure if f is "very hard" on average.

Stretch: $n \cdot k \longmapsto n \cdot k + k$ still very poor!

Idea 2: "Derandomize" XP Generator above, using designs:

$$S = (S_1, S_2, \dots, S_m)$$ ← maximize

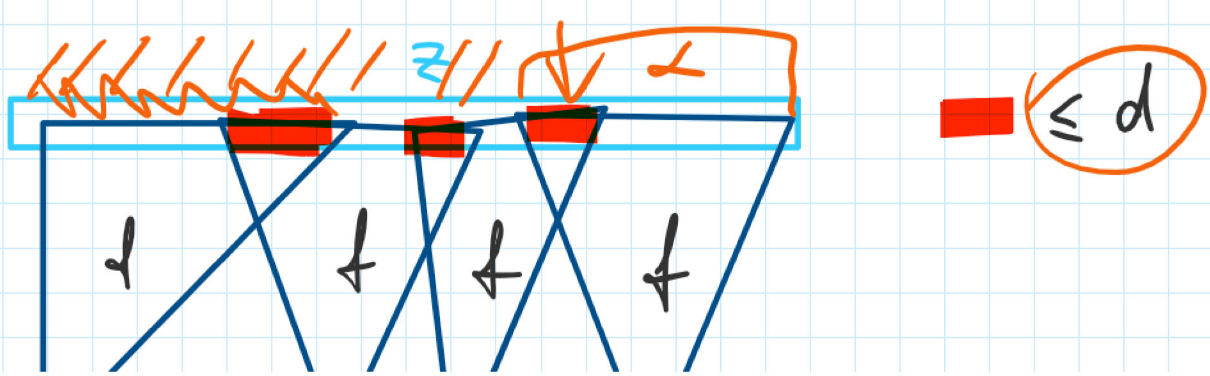
- each $|S_i| = n$
- $\forall\, i \neq j, \quad |S_i \cap S_j| \leq d$
- each $S_i \subseteq \{1, 2, \dots, \ell\}$

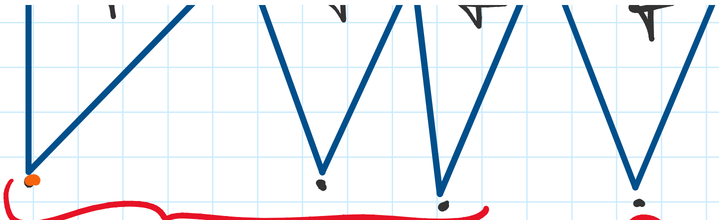
minimize

NW PRG $G^f : \{0,1\}^\ell \to \{0,1\}^m$:

$\forall$ seed $z \in \{0,1\}^\ell$

$$G^f(z) = f(z|_{S_1}) \dots f(z|_{S_m})$$





$\underbrace{z_1 = 0 \qquad z_2 = 1 \qquad z_{n-1}}$ $= g\left(\frac{T}{d}\right) \leftarrow 2^d$

Construction of designs: Given n, d,

can construct design $(S_1, \dots, S_m)$

- $|S_i| = n$, $\forall i$
- $|S_i \cap S_j| \leq d$, $\forall i \neq j$
- $\ell \leq O\left(\frac{n^2}{d}\right)$, $m = 2^{d/10}$

in time $\leq 2^{O(\ell)}$.

[Greedy algorithm works.]

Example:

$f : \{0,1\}^n \to \{0,1\}$, $d = \frac{n}{10} \Rightarrow G^f : \{0,1\}^{O(n)} \to \{0,1\}^{2^{n/100}}$

exponential stretch !!!

Security analysis: $m = 2^{n/100}$.

Circuit C of size m ε-"breaks" $G^f : O(n) \to m$

$\Rightarrow$ $\exists$ circuit R, $|R| \leq m + \underbrace{m}\cdot 2^d \leq 2^{n/5}$,

s.t.

$$\Pr_{x \sim U_n}\left[R(x) = f(x)\right] \geq \frac{1}{2} + \frac{\varepsilon}{2^{n/100}}$$

Why?

Need $f:\{0,1\}^n \to \{0,1\}$ exponentially hard on average!

Corollary [Nisan, Wigderson]:
E has a language of exp. average-case hardness
$\Rightarrow$ BPP = P.

Thm [IW, STV]:
E has a language of exp. average-case hardness
$\Leftrightarrow$ E has a language of exp. worst-case hardness.
[using list-decodable ECC]

[using list-decodable ECC]

SAT-oracle

Corollary: E requires circuit size $2^{\Omega(n)}$ $\Rightarrow$ BPP = P. also, MA = NP.

$\Rightarrow$ AM = NP

Other hardness-randomness trade-offs can be also proved, e.g.,

Thm [BFNW]: $\mathrm{EXP} \not\subseteq \mathrm{PolySize}$

$\Rightarrow \forall \varepsilon > 0,\ \mathrm{BPP} \subseteq \text{i.o.-}\mathrm{Time}[2^{n^{\varepsilon}}]$.

Some Applications

Easy Witness Lemma [IKW]:

Suppose $\mathrm{NEXP} \subseteq \mathrm{PolySize}$.

Then, $\forall\ L \in \mathrm{NTime}[2^{n^c}]$

with verifier $V : \{0,1\}^n \times \{0,1\}^{2^{n^c}} \to \{0,1\}$

$\exists\ d \geq 0$

$\forall x \in L$ of sufficiently large length n

$\forall x \in L$ of sufficiently large length n

$\exists$ circuit $T : \{0,1\}^{n^c} \to \{0,1\}$ of size $\le n^d$

s.t. $V(x, \text{truth table}(T)) = 1$.

Proof: Suppose not.

$\exists\ L \in \mathrm{NTime}[2^{n^c}]$

$\forall\ d \geq 0$

$\exists$ infinitely many $x \in \{0,1\}^n$

s.t. $x \in L$ but every witness $y \in \{0,1\}^{2^{n^c}}$ requires circuit size $> n^d$.

Oversimplifying:

(1) Can nondeterministically guess super-poly-hard t.t.

(2) Via [BFNW], can derandomize

$$MA \subseteq NSUBEXP.$$

(3) $NEXP \subseteq PolySize$

$\Rightarrow EXP \subseteq PolySize$

$\Rightarrow EXP = MA$ [Karp-Lipton]

(4) $EXP = MA \subseteq NSUBEXP$

(5) NEXP $\subseteq$ PolySize $\Longrightarrow$ NSUBEXP $\subseteq$ Size$[n^k]$
for a fixed constant $k > 0$.
[using universal TMs]

(4) + (5) $\Longrightarrow$
$\exists\, k > 0$, EXP $\subseteq$ Size$[n^k]$. Contradiction
(via diagonalization).

<u>Corollary</u>: NEXP $\subseteq$ PolySize $\Longleftrightarrow$ NEXP = MA.

Summary

Randomness is useful
- for algorithms
- for reasoning about complexity classes
- . . .

Pseudo-randomness $\equiv$ Circuit lower bounds

Pseudo-randomness vs. Learning

. . .