Connections Between Total Search and Lower bounds

FNP:

$R \subseteq \{0,1\}^* \times \{0,1\}^*$

$(x, y) \in R \iff$ y is a "solution" to the "instance" x

membership in R testable in P, $|y| \leq poly(|x|)$

given x, find y

TFNP:

$\phi_R :=$ " $\underline{\forall x \, \exists y \text{ s.t. } (x,y) \in R}$ " is a true theorem

$$\boxed{\begin{array}{c}\text{complexity of search problem } R \\ \text{SS} \\ \text{"constructivity" of theorem } \phi_R\end{array}}$$

non-constructivity typically seen as purely negative

but can be useful!

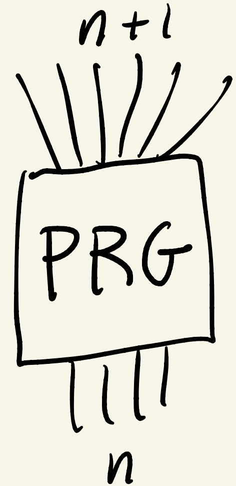
insight of cryptography:

$$\boxed{\text{nonconstructive theorems can have "effective counterexamples" which let us bypass information-theoretic barriers}}$$

Shannon: "for any function f, r.v. X,

$$H(f(X)) \le H(X)$$

Yao: "No,



"

This work:

Nonconstructivity $\Rightarrow$ Computational Advantage

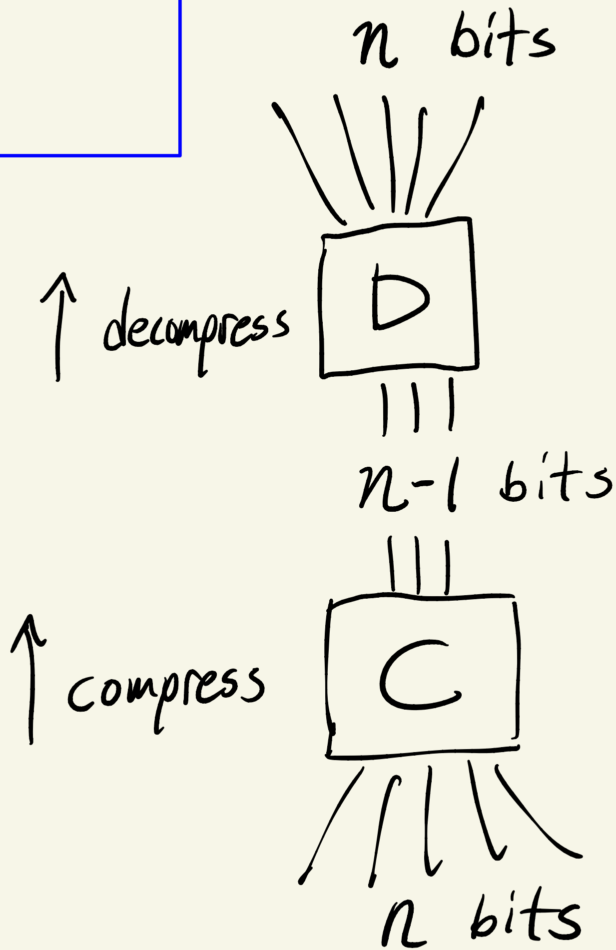
(of a certain PHP) (low-space simulation of RAM)

Contrapositive:

Uniform Lower Bound $\Rightarrow$ Constructivity

(time-space tradeoff) (polytime witnessing for PHP)

Bijective Weak PHP:



$$D \circ C \neq Id_n$$

$$\Downarrow$$

$$\exists x \text{ s.t. } D(C(x)) \neq x$$

"Uniform instances" of PHP:

C, D defined for all n by a pair of poly-time TMs

Main Theorem:

If there is an "effective counterexample" to PHP,

uniform instance C, D s.t. no poly-time algorithm can find x s.t. $D(C(x)) \neq x$

then there is a universal simulation of RAM computation in small space and near-linear time one is true:

(1) For every pair C, D of poly-time TMs s.t. $|C(x)| = |x| - 1$, $|D(x)| = |x| + 1$ there is a poly(n) time algorithm to construct $x \in \{0,1\}^n$ s.t. $D(C(x)) \neq x$

(2) For large enough* $T(n)$, every T-time RAM computation can be simulated in $T^{1+\varepsilon}$ time, T^{ε} space on 1-tape TM

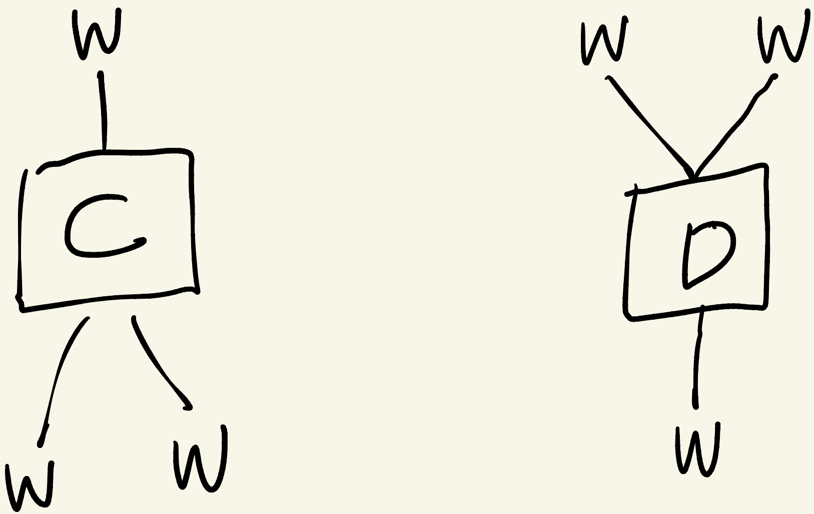
$\boxed{{}^{*}\ T = 2^{\Omega(n)}}$

Setup:

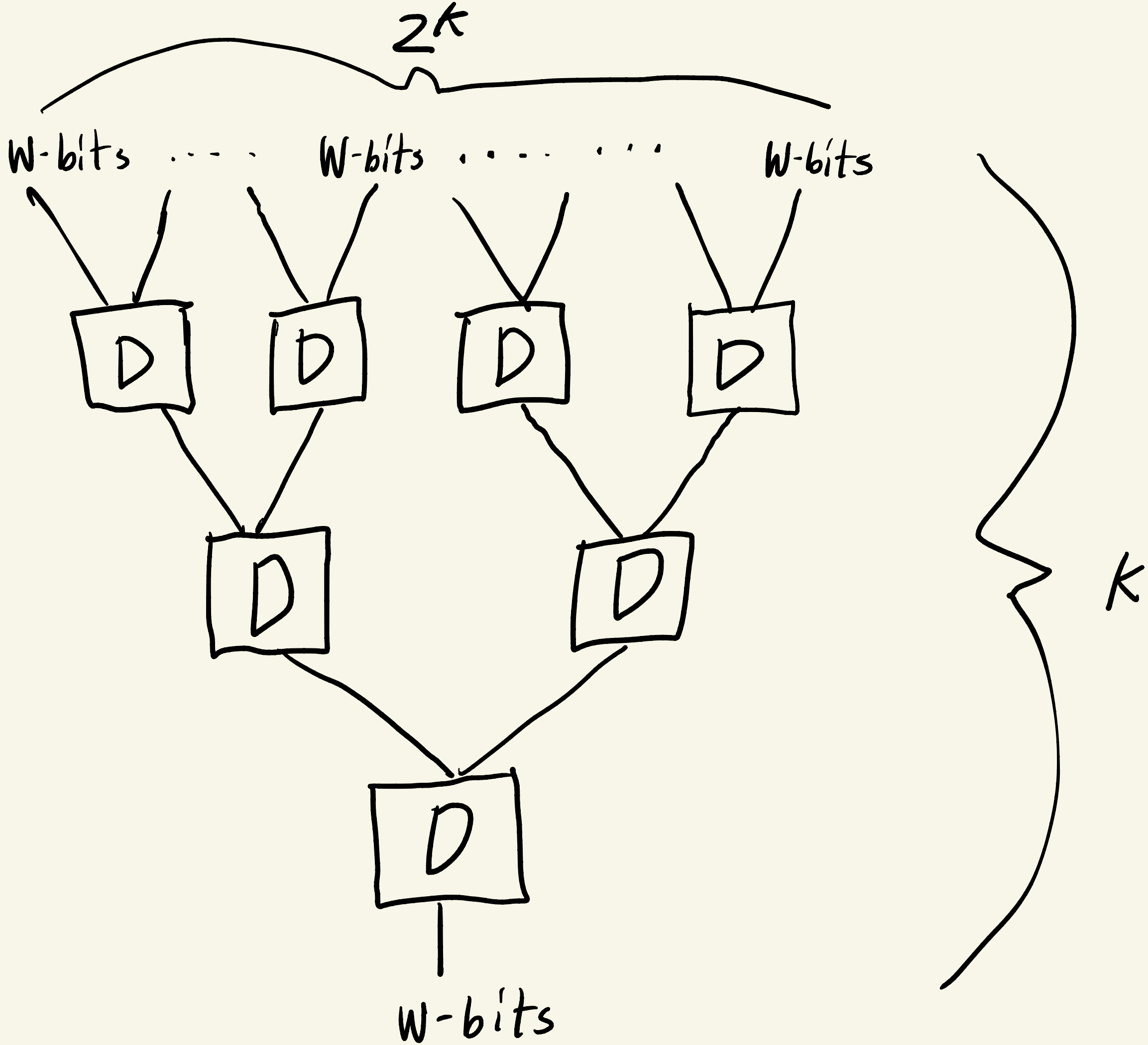
- Fix $\varepsilon > 0$

- Let M be RAM machine running in time T

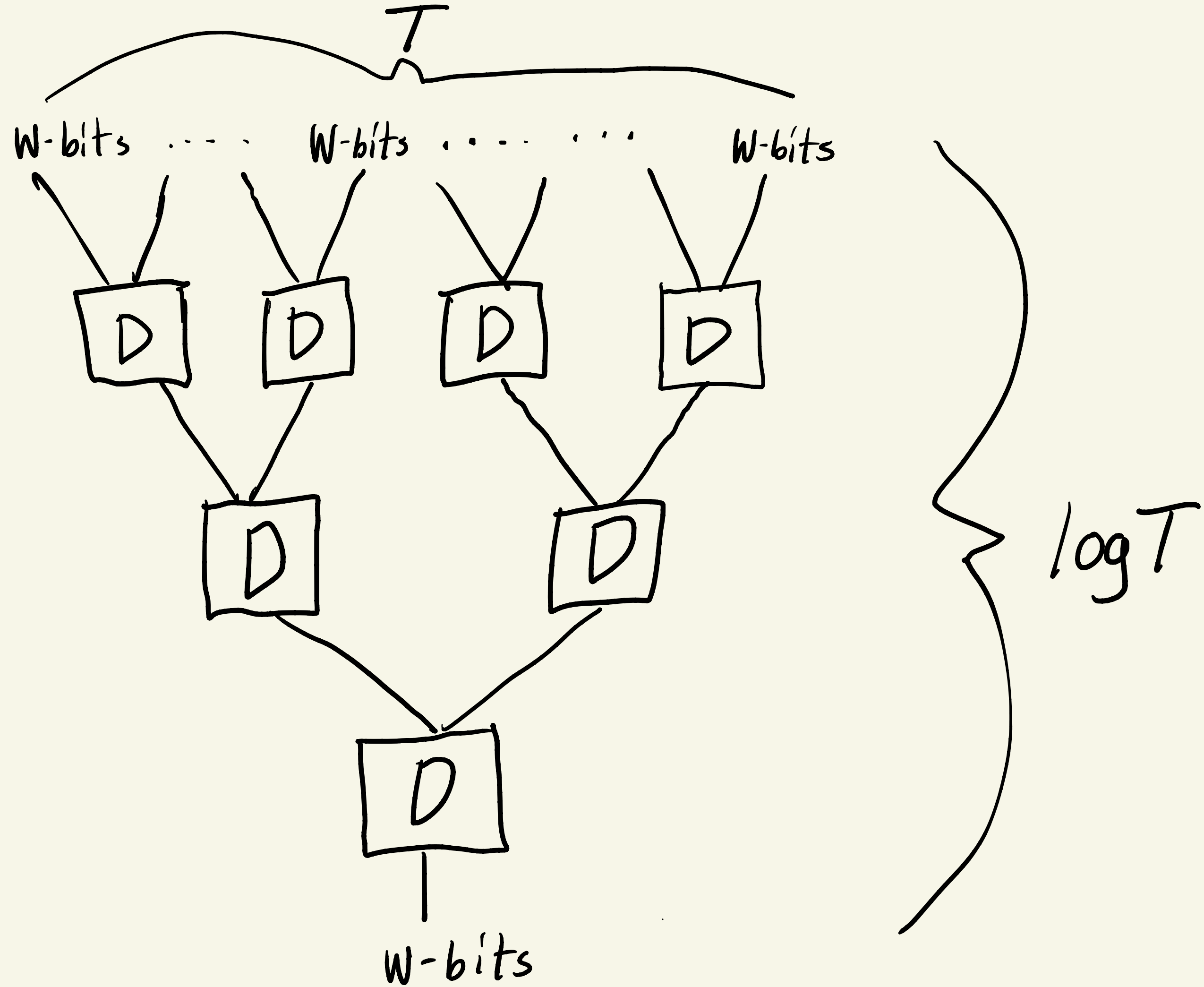
Let $W = T^{\varepsilon}$, focus on C_W, D_W:

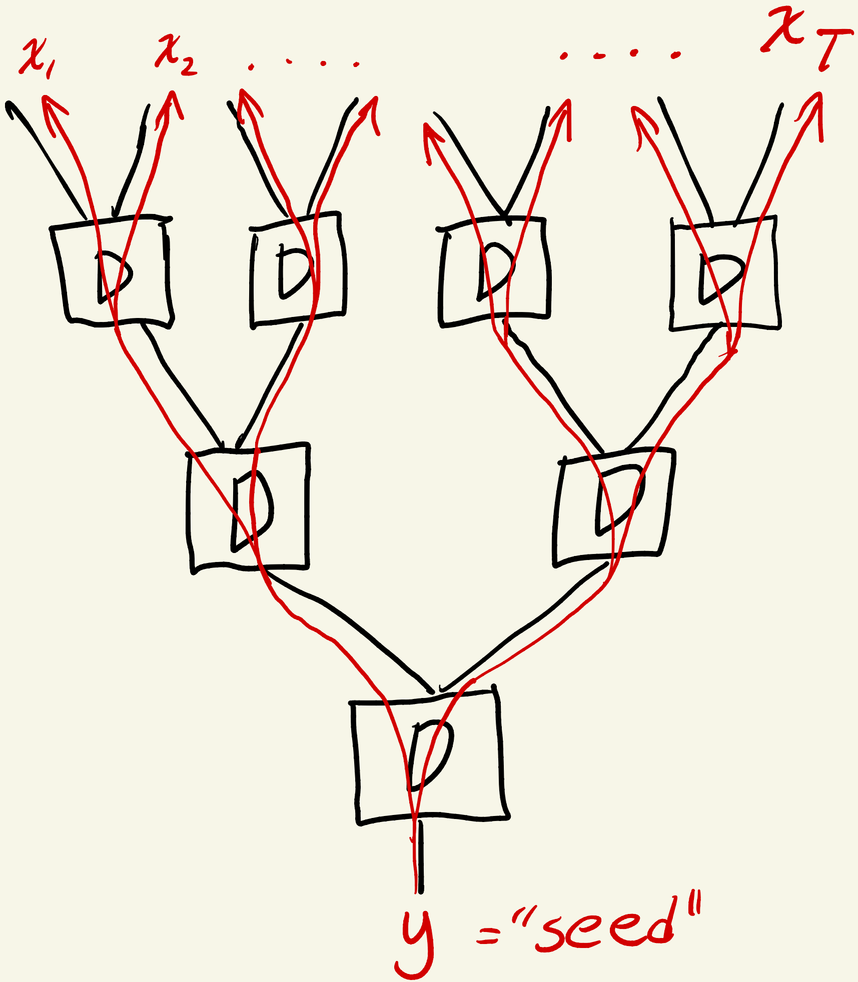


these are simply restrictions of our "uniform instance" to inputs of length W



2^k
W-bits
W-bits
W-bits
D
D
D
D
D
D
D
k
W-bits


T
W-bits
W-bits
W-bits
D
D
D
D
D
D
D
log T
W-bits



y is $W = T^{\varepsilon}$ bits long, but represents $\geq T$ bits of "memory"

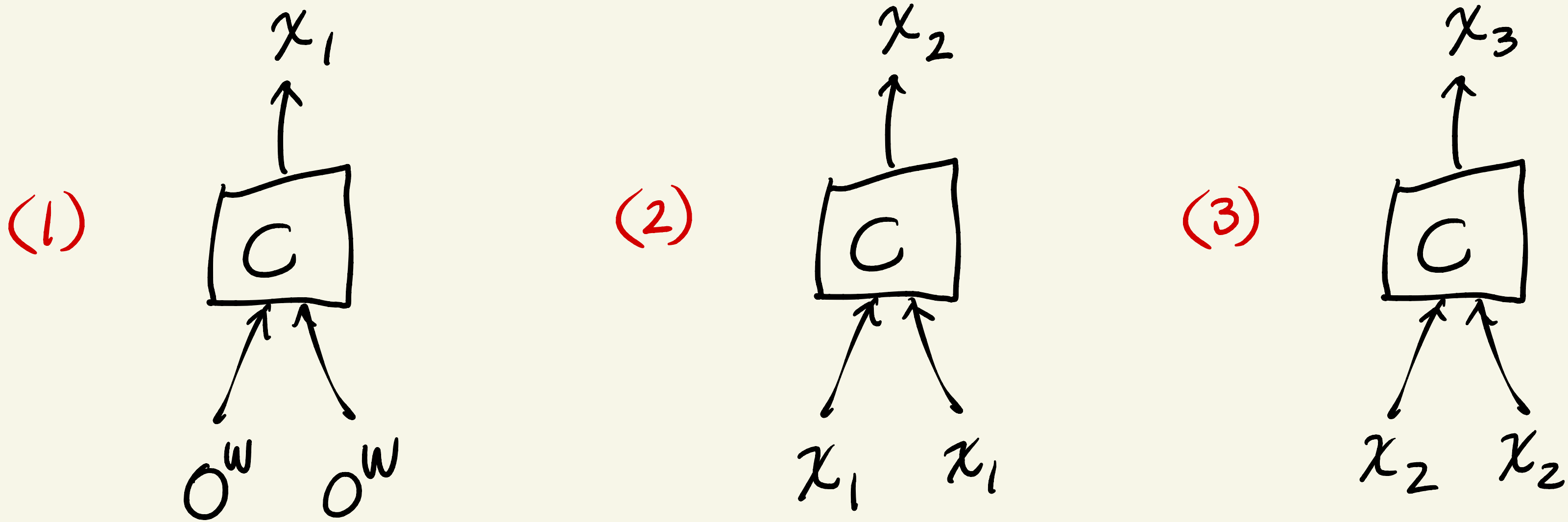
To make this "virtual memory" work we need 3 operations:

(1) **Innitialize:** Set all memory cells to 0

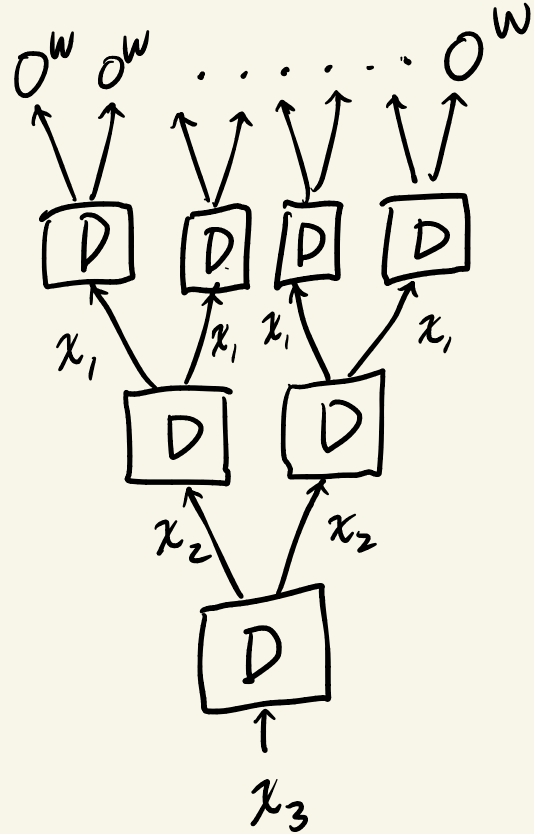
(2) **Access:** Read i^{th} memory cell

(3) **Update:** Set i^{th} cell to $b \in \{0,1\}$

Innitialize:



result:



$$\boxed{\text{Time spent:}}$$

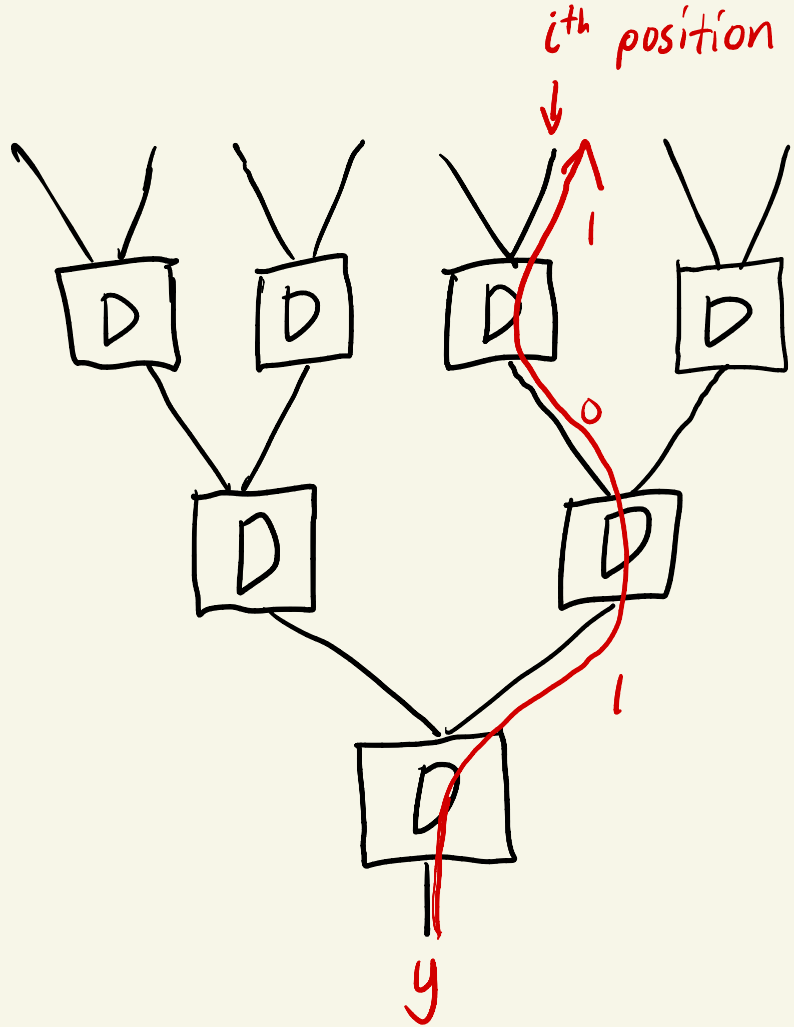
$$\log T \cdot \left(\begin{array}{c}\text{time to evaluate} \\ C/D\end{array} \right)$$

$$= \log T \cdot \mathrm{poly}(T^{\varepsilon})$$

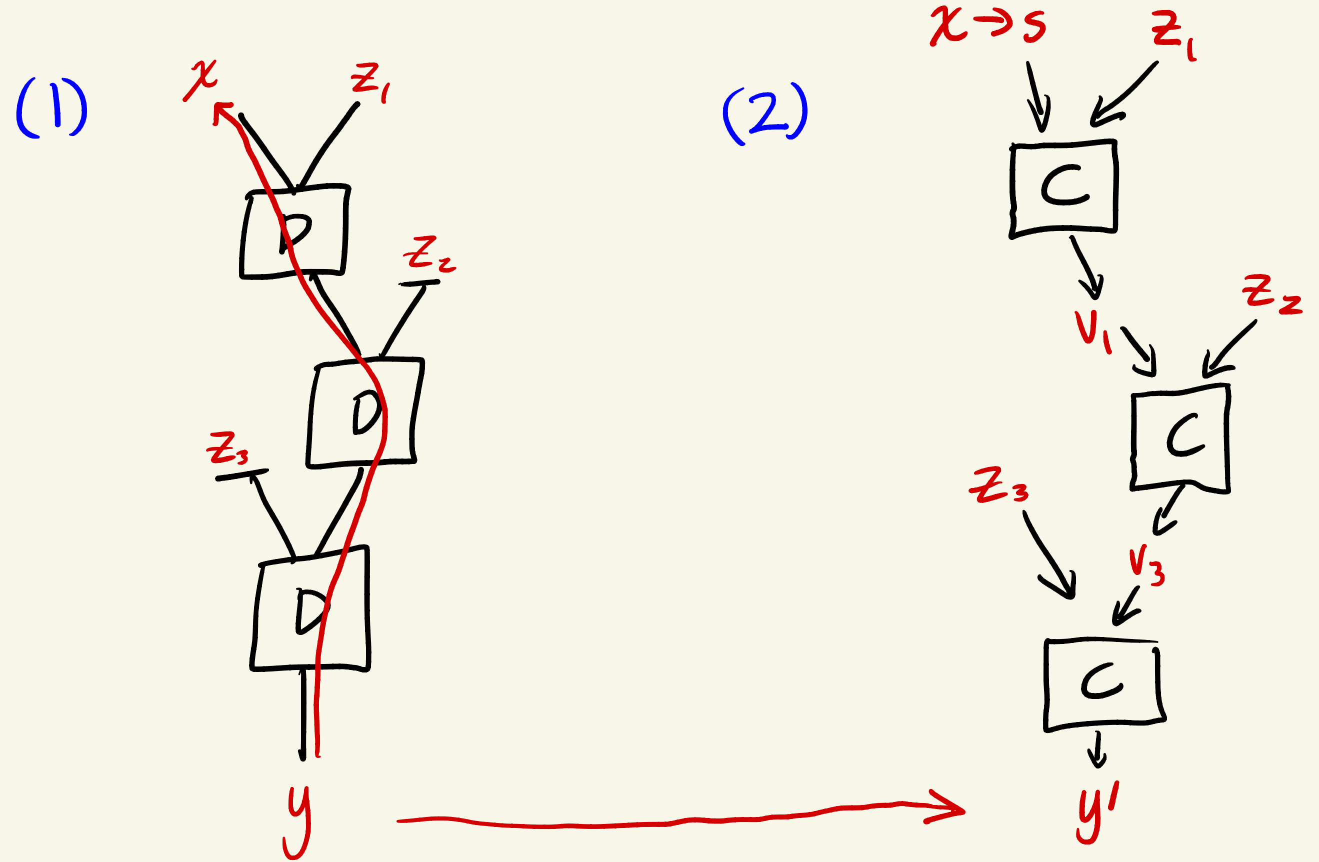
$$= T^{O(\varepsilon)}$$

Access:

given $i \in [T]$, access ith cell by following the path:



Update: Set position 100 to value $s \in \{0,1\}^w$



Recap:

- Simulate a T-time machine using "virtual RAM" data structure

- Data Struc. uses a "counterexample to PHP" on inputs of length $W = T^{\varepsilon}$

- All operations run in time $T^{O(\varepsilon)}$ and space $T^{O(\varepsilon)}$, and faithfully maintain the memory assuming PHP "fails" for C, D

Taking $\varepsilon \to 0$ we get simulation in time $T^{1+\varepsilon}$, space T^{ε} for arbitrarily small $\varepsilon > 0$

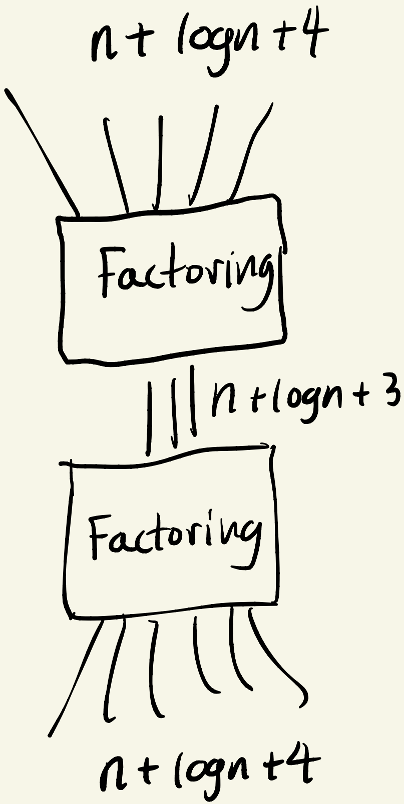
Contrapositive:

If such a simulation fails, we witness the PHP for C, D.

i.e. locate an x s.t. $D(C(x)) \neq x$

Interesting Uniform Instances of PHP:

Large Primes: [PWW '88]



solutions yield 32n-bit primes of magnitude $> 2^n$

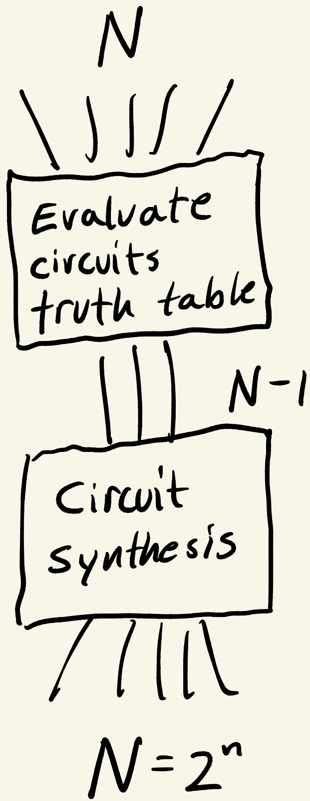
one is true:

(1) n-bit primes can be constructed in poly(n) time "pseudodeterministically" on a quantum computer

(2) T-time RAM machines can be simulated in time $T^{1+\varepsilon}$, space T^{ε} on a quantum computer

($\leq T^{\varepsilon}$ qubits)

Hard Truth Tables



description of circuit for our truth table

Solutions are hard truth tables

Nondeterministic Tradeoffs:

Thm:

If T-time nondeterministic machines can not be simulated by 1-tape nondeterministic machines in:

- $T^{1+\varepsilon}$ time
- T^{ε} space
- T^{ε} nondeterministic guesses

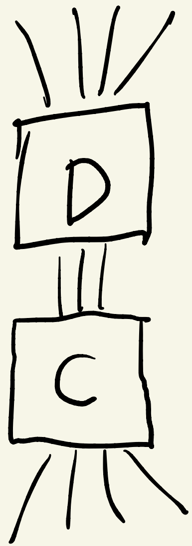
then $E^{NP} \not\subseteq \text{size}(2^n/2n)$

Premise is known for $T = O(n)$, we need for $T = 2^{\Omega(n)}$

__Question:__ Can we get more unlikely simulations from "effective counterexamples" to other principles?

perhaps this phenomenon is unique to derandomization problems...

"Lossy Code" as a search problem:



- lies in TFNP

  ($\in$ PWPP $\subseteq$ PPP, $\in$ APEPP)

- lies in FZPP

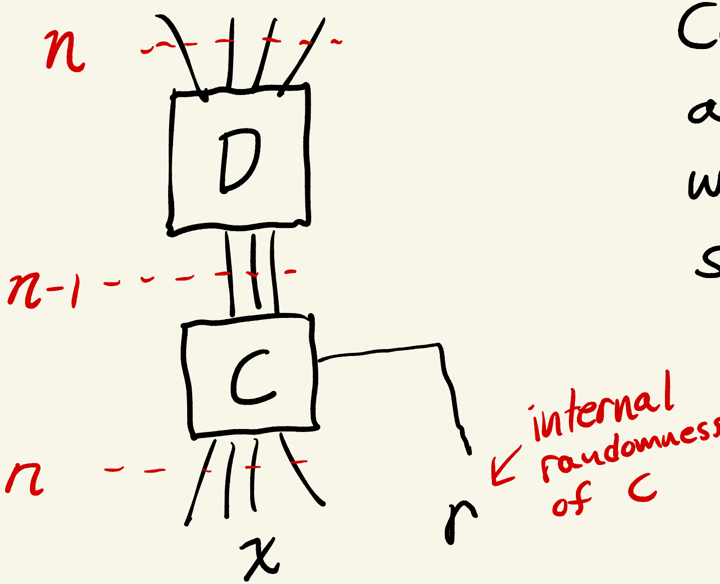
given C, D as circuits,

find x s.t. $D(C(x)) \neq x$

Can't show Lossy Code captures "full derandomization" (prBPP) without proving $BPP \subseteq NP$...

But a slight modifaction of the problem is indeed equivalent to prBPP!

R-Lossy Code:



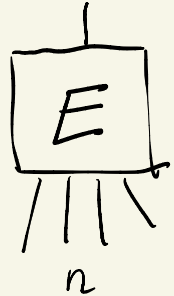
Compression algorithm randomized, and we seek a string with low probability of successful compression

Find x s.t.

$$\Pr_r[D(C(x)) = x] < 1/2$$

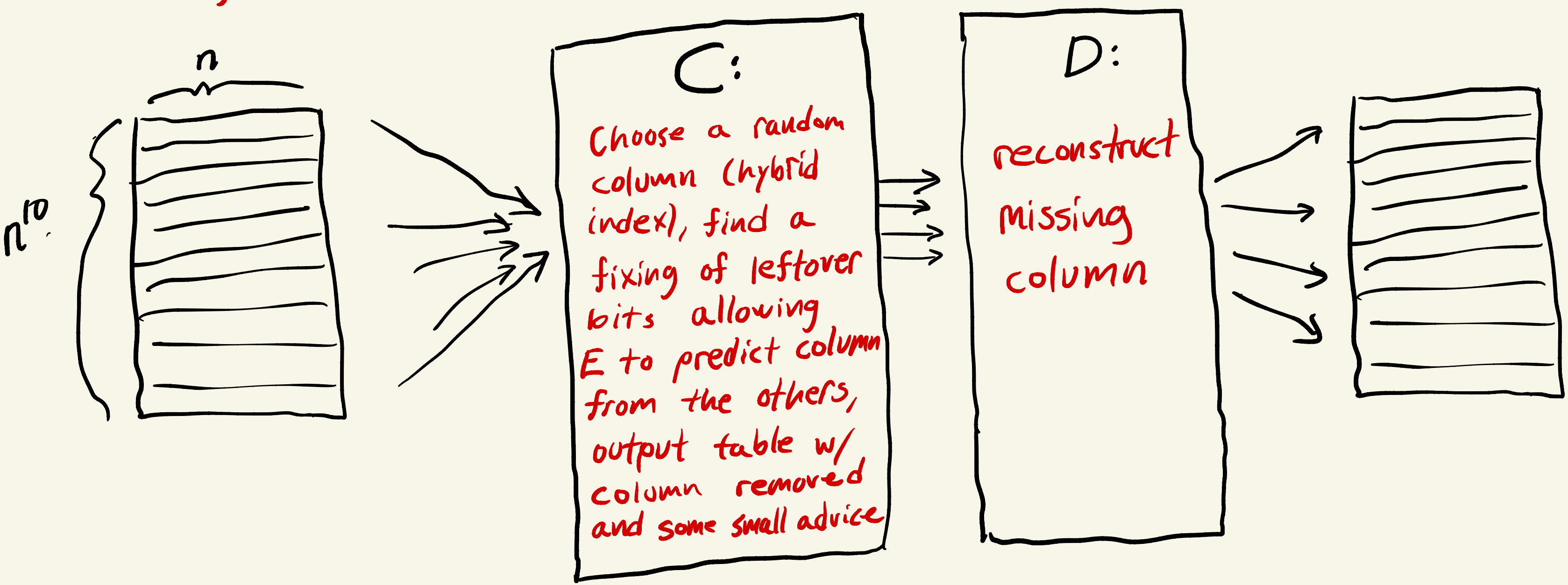
Thm: R-Lossy Code complete for prBPP

↳ straight-forward application of Yao's "next-bit unpredictable ≈ pseudorandom" lemma prBPP: given

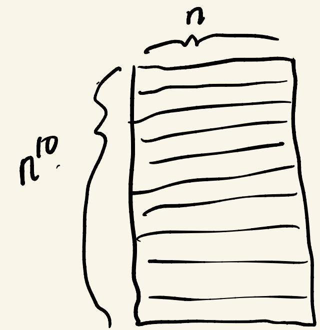


, approximate $\Pr_x[E(x)=1]$

input: sample of n-bit strings of size n^{10}



Yao's Lemma: if

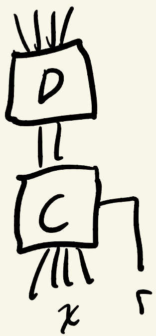


fails to approximate $\Pr[E(x)=1]$, then C compresses it w.h.p.

Application:

(another) easy proof that hitting sets $\Rightarrow$ prBPP $=$ P

pf:



(1) for x s.t. $\Pr_r[D \circ C(x,r) = x] \geq 1/2$,

we can find r s.t. $D \circ C(x,r) = x$ in any hitting set

↳ construct circuit $E(x)$ which outputs 1 if it fails to find such r, 0 else.

x non-solution $\Rightarrow E(x) = 0$

(2) w.p. $\geq 1/2$ over x, $x \notin \text{RANGE}(D)$, so no such r exists, so $E(x) = 1$

↳ hitting set for E finds such x