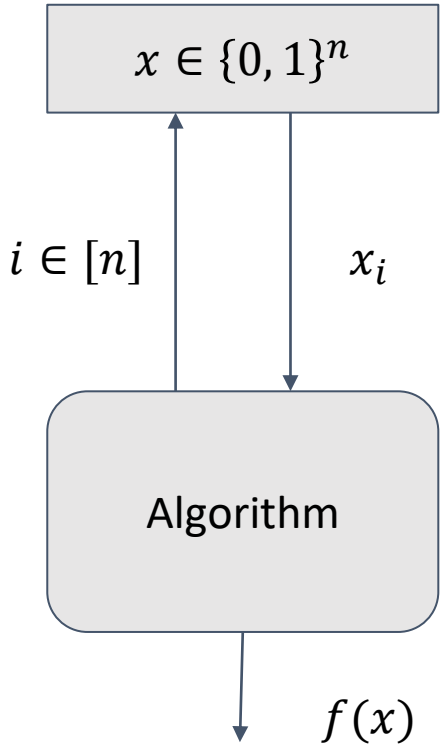
Approximate degree lower bounds for oracle identification problems

Mark Bun, Nadezhda Voronova
Boston University

Query model

Given: oracle (black box) access to $x \in \{0,1\}^n$
Goal: compute $f(x)$
Cost: # of queries



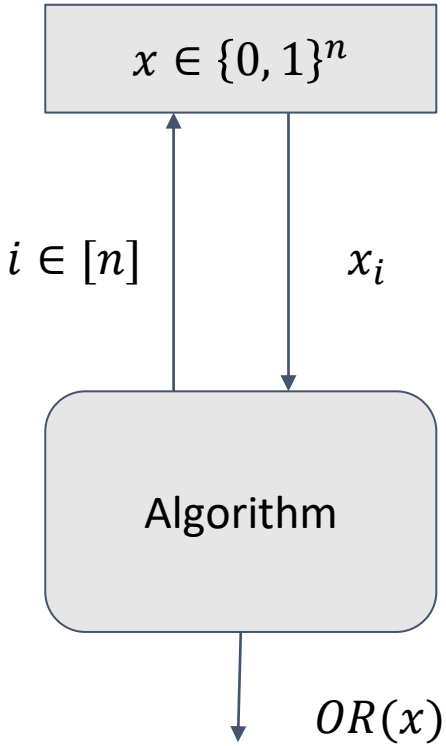
Why consider?

- Strips away implementation details.
- Can prove unconditional lower bounds!

Resources in query model: example

Types of queries:

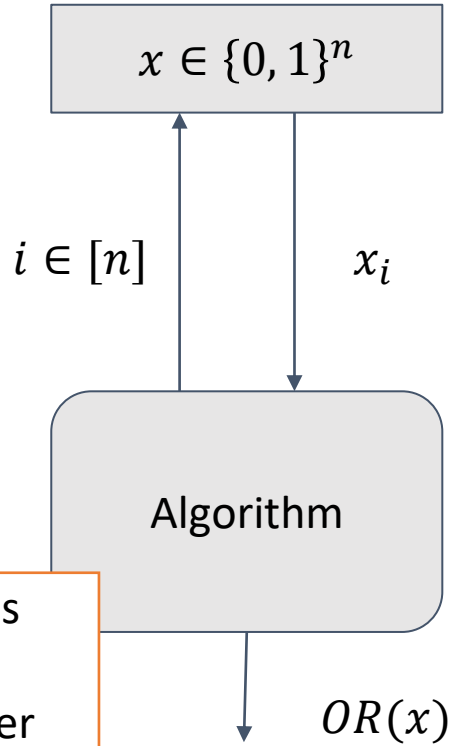
- Deterministic
  - No additional resources
  - Without error
  - Computing OR requires $\Theta(n)$ queries
- Randomized
  - Access to unbiased random bits
  - Correct with success Pr ⅔
  - Computing OR requires $\Theta(n)$ queries
- Quantum : $\Theta(\sqrt{n})$
  - Query in superposition
  - Correct with success Pr ⅔
  - Computing OR requires $\Theta(\sqrt{n})$ queries



Resources in query model: example

Types of queries:

- Deterministic
  - No additional resources
  - Without error
  - Computing OR requires $\Theta(n)$ queries
- Randomized
  - Access to unbiased random bits
  - Correct with success Pr ⅔
  - Computing OR requires $\Theta(n)$ queries
- Quantum : $\Theta(\sqrt{n})$
  - Query in superposition
  - Correct with success Pr ⅔
  - Computing OR requires $\Theta(\sqrt{n})$



Bounded error: success probability ⅔
Unbounded error: better than random guessing

Approximate degree

Let $f: D \to \{0,1\}$ where $D \subseteq \{0,1\}^n$.

A polynomial $p: \{0,1\}^n \to \mathbb{R}$ is ε-**approximation** to f if

- $-\varepsilon \leq p(x) \leq 1 + \varepsilon$ for all $x \in \{0,1\}^n$
- $|p(x) - f(x)| \leq \varepsilon$ for all $x \in D$

Approximate degree $\widetilde{deg}_\varepsilon(f)$ **of** $\boldsymbol{f}$ is the least degree of a polynomial that ε-approximates f.

Default error $\varepsilon = \frac{1}{3}$

Corresponds to bounded error query model

Applications of $\widetilde{deg}$

Upper bounds:

- Learning Algorithms

  [Klivans-Servedio03, Klivans-Servedio06, Kalai-Klivans-Mansour-Servedio06]

- Approximate Inclusion-Exclusion

  [Kahn-Linial-Samorodnitsky96, Sherstov08]

- Differentially Private Query Release

  [Thaler-Ullman-Vadhan12, Chandrasekaran-Thaler-Ullman-Wan14]

- Formula & Graph Complexity Lower Bounds [Tal14,16ab]

Lower bounds:

- Communication Complexity

  [Sherstov07, Shi-Zhu07, Chattopadhyay-Ada08, Lee-Shraibman08,...]

- Circuit Complexity

  [Minsky-Papert69, Beigel93, Sherstov08]

- Oracle Separations

  [Beigel94, Bouland-Chen-Holden-Thaler-Vasudevan16]

- Secret Sharing Schemes

  [Bogdanov-Ishai-Viola-Williamson16]

- Quantum query complexity

Polynomial method

Acceptance probability of a randomized T-query algorithm is a polynomial of degree T.

The ε-approximate degree of a function is always at most its randomized query complexity with error ε.

Polynomial method

Acceptance probability of a quantum T-query algorithm is a polynomial of degree 2T.

The ε-approximate degree of a function is always at most ½ its quantum query complexity with error ε.

Polynomial method

Acceptance probability of a quantum T-query algorithm is a polynomial of degree 2T.

The ε-approximate degree of a function is always at most ½ its quantum query complexity with error ε.

Advantages

- Robust.
  - Lifts to quantum communication [She11],
  - Can yield lower bounds against zero-, small-, and unbounded-error quantum algorithms [BBC+01, BCdWZ99],
  - Gives time-space tradeoffs [KSdW07]
- Transparent in identifying the hardness
  - Search vs decision problem
  - Can be more intuitive than some other methods

Other ways to prove lower bounds: "adversary" methods

- "Positive-weights" method [Ambainis02]
  - Easy to apply, but limited in power
- "Negative-weights" method [Høyer-Lee-Špalek07, ..., Reichardt11]
  - Tight characterization, but difficult to apply

Problems

Ordered search OS_{2^n}

Given: string $0^x 1^{N-x}, N = 2^n$
Output: x

Hidden string

Given: collection of bits $\tau_s(x)$, where
$\tau_s(x) = 1$ iff s is a substring of hidden x
Output: x

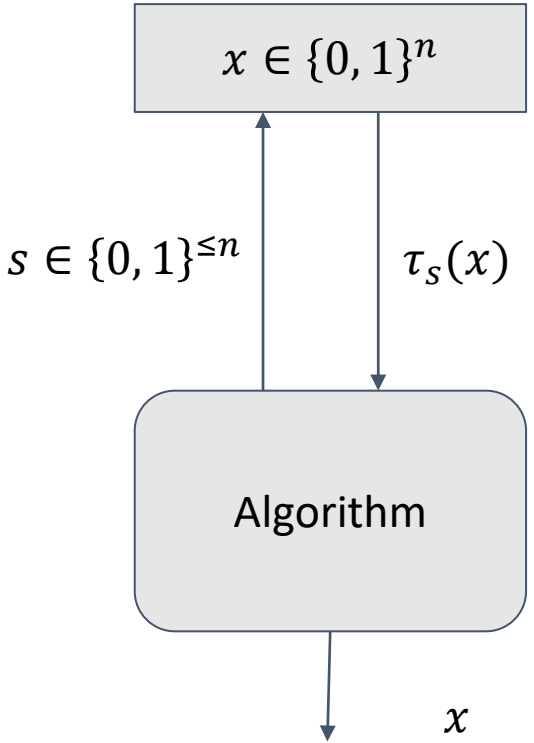
Problems

Ordered search OS_{2^n}

Given: string $0^x 1^{N-x}, N = 2^n$
Output: x

Hidden string

Given: collection of bits $\tau_s(x)$, where $\tau_s(x) = 1$ iff s is a substring of hidden x
Output: x



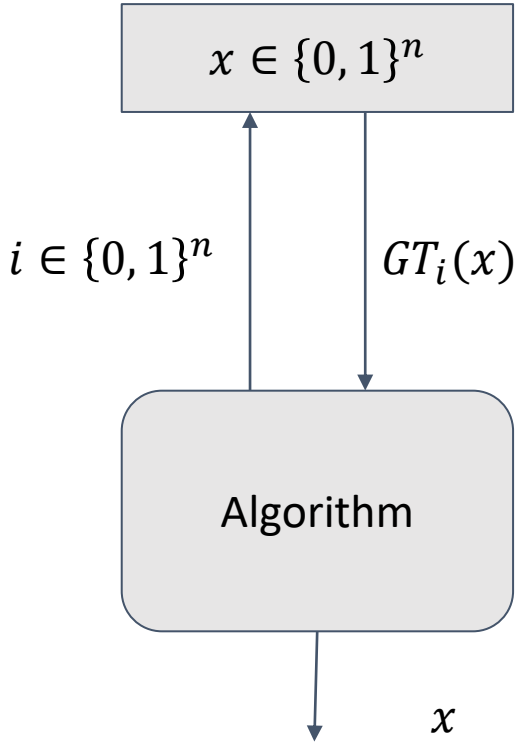
Problems

Ordered search OS_{2^n}

Given: string $0^x 1^{N-x}, N = 2^n$
Output: x

Hidden string

Given: collection of bits $\tau_s(x)$, where $\tau_s(x) = 1$ iff s is a substring of hidden x
Output: x



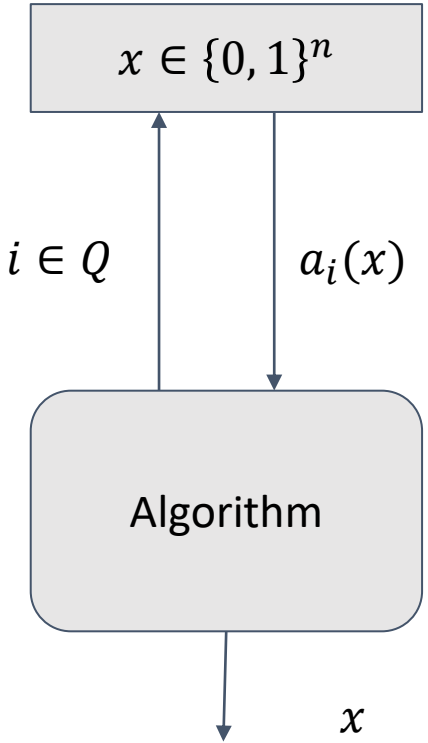
Problems

Ordered search OS_{2^n}

Given: string $0^x 1^{N-x}, N = 2^n$
Output: x

Hidden string

Given: collection of bits $\tau_s(x)$, where $\tau_s(x) = 1$ iff s is a substring of hidden x
Output: x



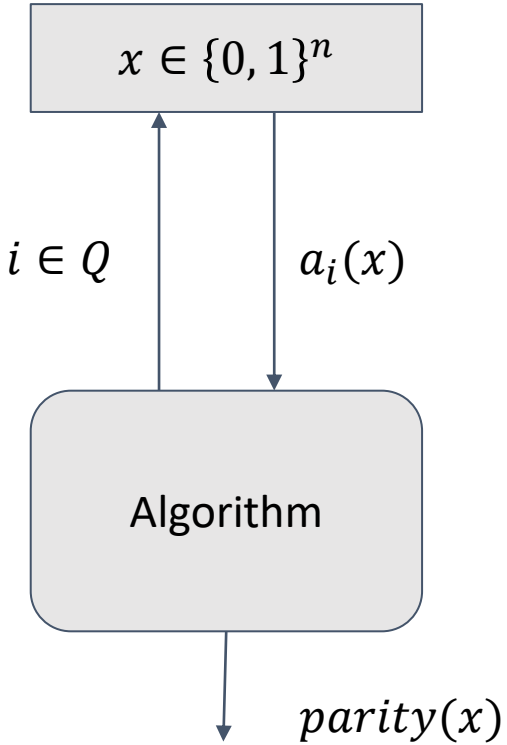
Problems

Ordered search OS_{2^n}

Given: string $0^x 1^{N-x}, N = 2^n$
Output: $parity(x)$

Hidden string

Given: collection of bits $\tau_s(x)$, where $\tau_s(x) = 1$ iff s is a substring of hidden x
Output: $parity(x)$



Results: state-of-the-art

	Ordered Search OS_{2^n}	Hidden String
Approximate degree, Quantum query complexity for decision problem	$O(n), \Omega(\sqrt{n})$ [BdW99]	$O(n)$ [SS95][CIG+12]
Quantum query complexity for reconstruction problem	$\Theta(n)$ [BdW99, FGGS98, Amb99, HNS02, CL08]	$O(n)$[SS95], $\Omega(n/\log^2 n)$ [CIG+12]

Results: bounded error

	Ordered Search OS_{2^n}	Hidden String
Approximate degree, Quantum query complexity for decision problem	$O(n), \Omega(\sqrt{n})$ [BdW99] This work: $\Omega(n/\log^2 n)$	$O(n)$ [SS95][CIG+12] This work: $\Omega(n/\log^2 n)$
Quantum query complexity for reconstruction problem	$\Theta(n)$ [BdW99, FGGS98, Amb99, HNS02, CL08]	$O(n)$[SS95], $\Omega(n/\log^2 n)$ [CIG+12] This work: $\Omega(n/\log^2 n)$

Results: unbounded error

	Ordered Search OS_{2^n}	Hidden String
Approximate degree, Quantum query complexity for decision problem Success pr $\frac{1}{2}+\gamma$	$O\left(n-\log\frac{1}{\gamma}\right), \Omega(\sqrt{n}-\log\frac{1}{\gamma})$ [BdW99] This work: $\Omega(\frac{n}{\log^2 n}-\log\frac{1}{\gamma})$	$O(n)$ [SS95] [CIG+12] This work:$\Omega(\frac{n}{\log^2 n}-\log\frac{1}{\gamma})$
Quantum query complexity for reconstruction problem Success pr γ	$\Theta(n-\log\frac{1}{\gamma})$ [Implicit in Amb99]	$O\left(n-\log\frac{1}{\gamma}\right)$ [SS95], $\Omega(\gamma^2\frac{n}{\log^2 n})$ [CIG+12] This work: $\Omega(\frac{n}{\log^2 n}-\log\frac{1}{\gamma})$

- All results in this work were achieved using the same framework.

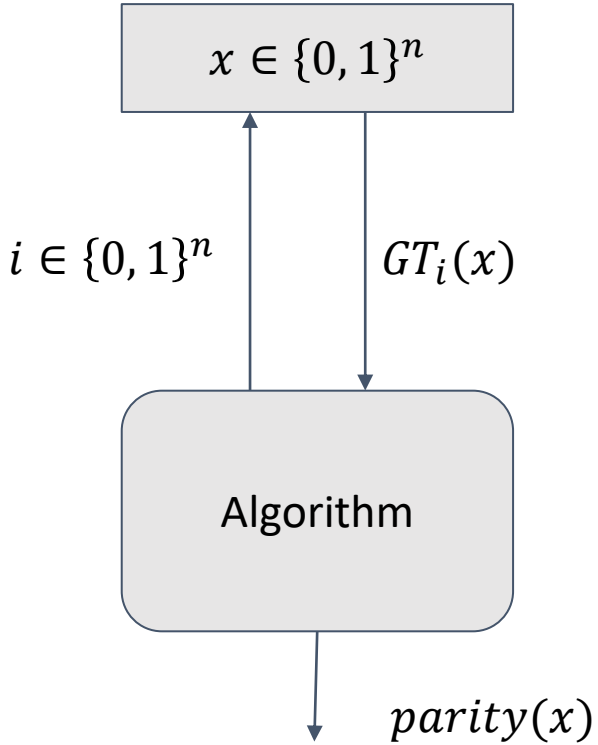
Problems

Ordered search OS_{2^n}

Given: string $0^x 1^{N-x}, N = 2^n$
Output: $parity(x)$

Hidden string

Given: collection of bits $\tau_s(x)$, where $\tau_s(x) = 1$ iff s is a substring of hidden x
Output: $parity(x)$

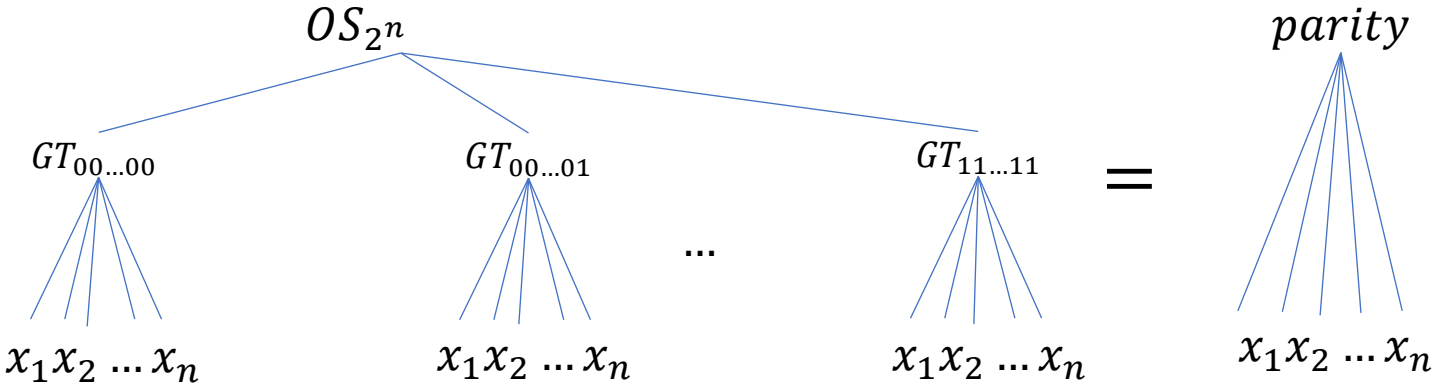


Buhrman and de Wolf's argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(\sqrt{n})$.

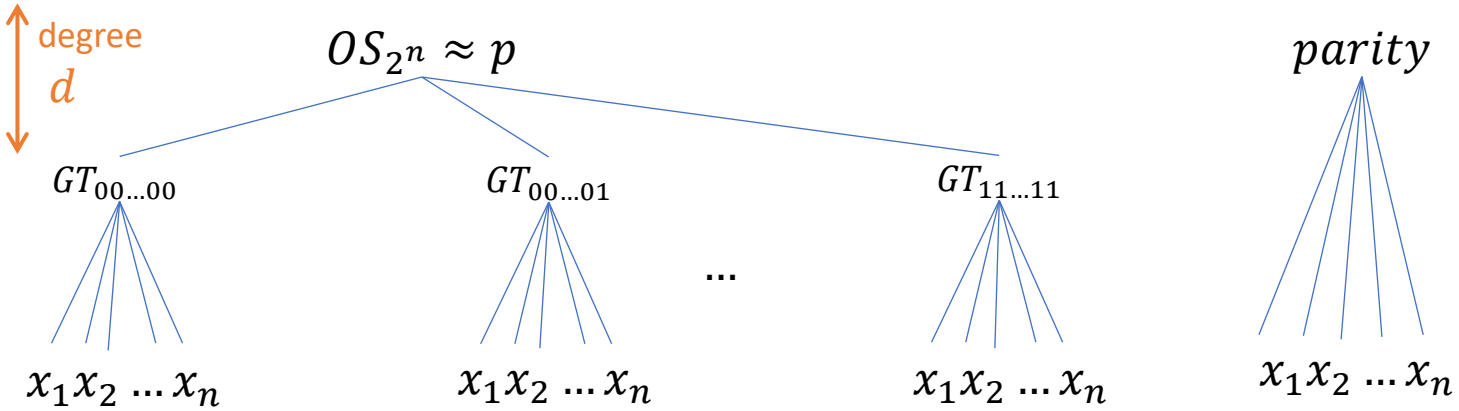
Buhrman and de Wolf's argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(\sqrt{n})$.



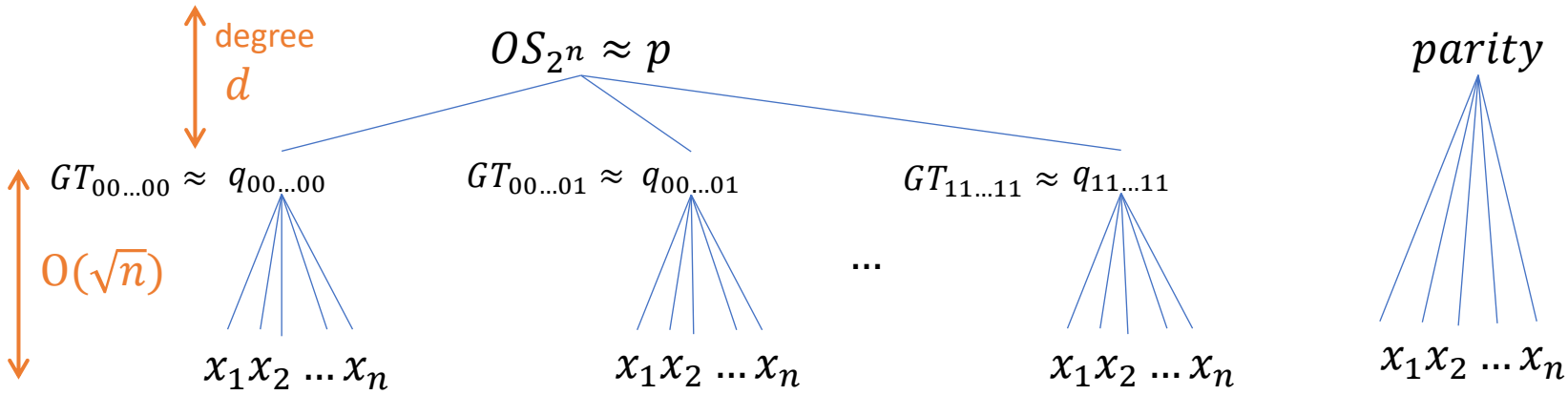
Buhrman and de Wolf's argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(\sqrt{n})$.



Buhrman and de Wolf's argument

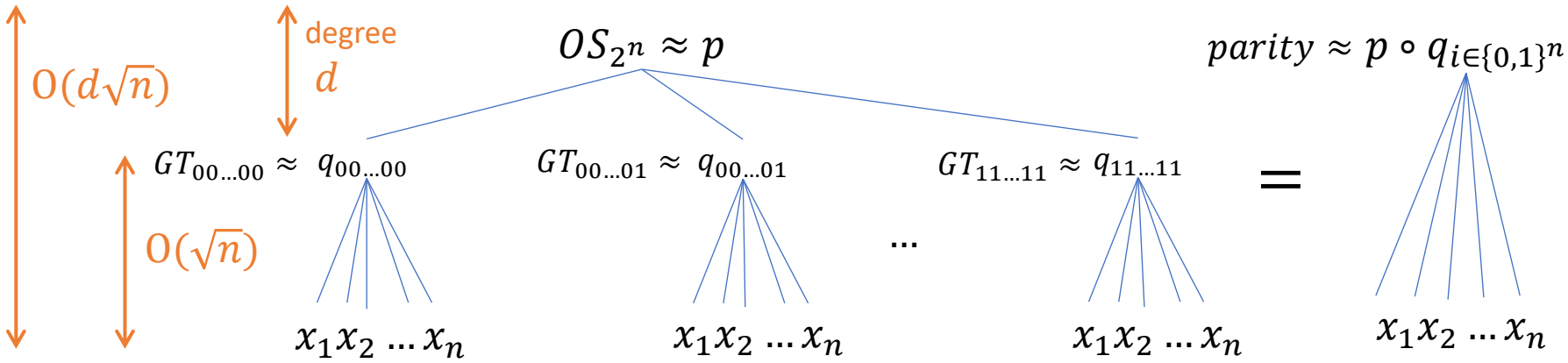
Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(\sqrt{n})$.



Claim 1. $\forall i \in \{0,1\}^n$ there exists a polynomial of x of degree $O(\sqrt{n})$ that approximates $GT_i(x)$.

Buhrman and de Wolf's argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(\sqrt{n})$.

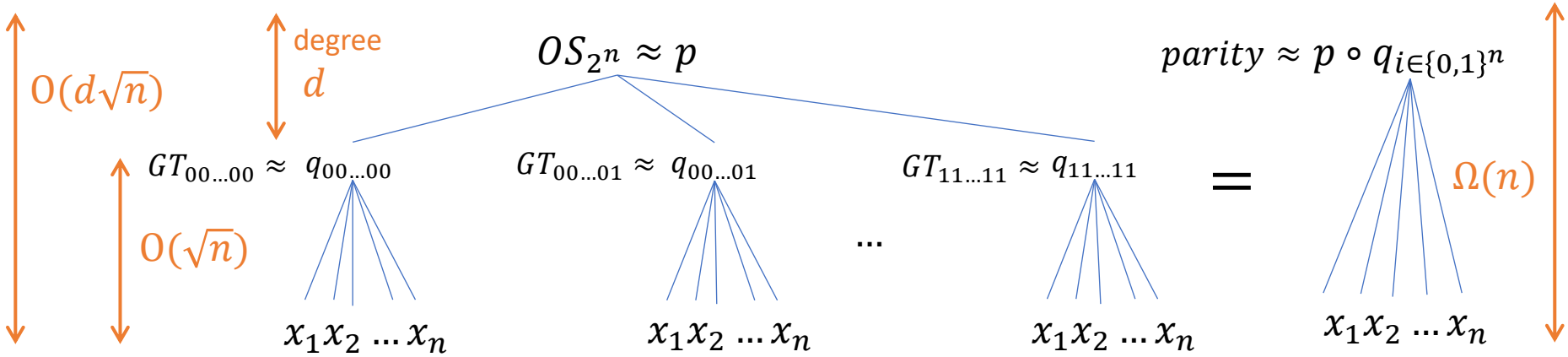


Claim 1. $\forall i \in \{0,1\}^n$ there exists a polynomial of x of degree $O(\sqrt{n})$ that approximates $GT_i(x)$.

[She12] Every polynomial can be made robust to constant noise with constant blowup in degree.

Buhrman and de Wolf's argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(\sqrt{n})$.



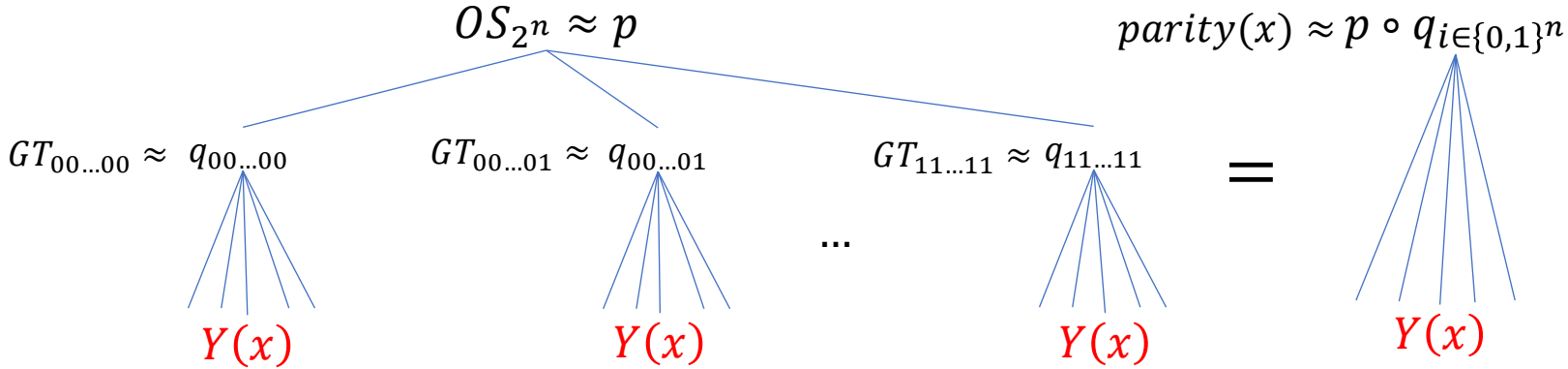
Claim 1. $\forall i \in \{0,1\}^n$ there exists a polynomial of x of degree $O(\sqrt{n})$ that approximates $GT_i(x)$.

[She12] Every polynomial can be made robust to constant noise with constant blowup in degree.

Claim 2. Every polynomial of x that approximates $parity(x)$ requires degree $\Omega(n)$.

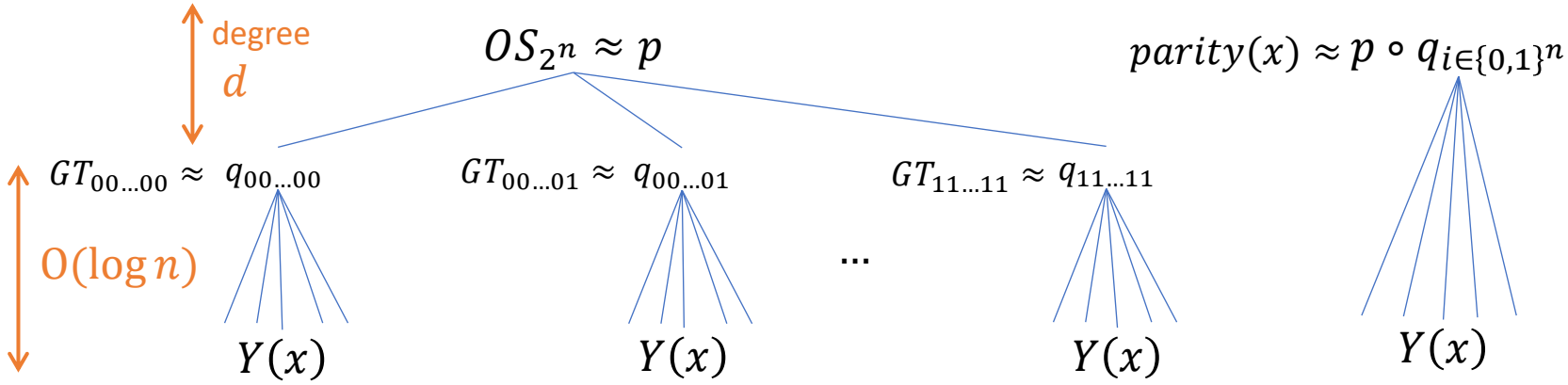
Our argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(n/\log^2 n)$.



Our argument

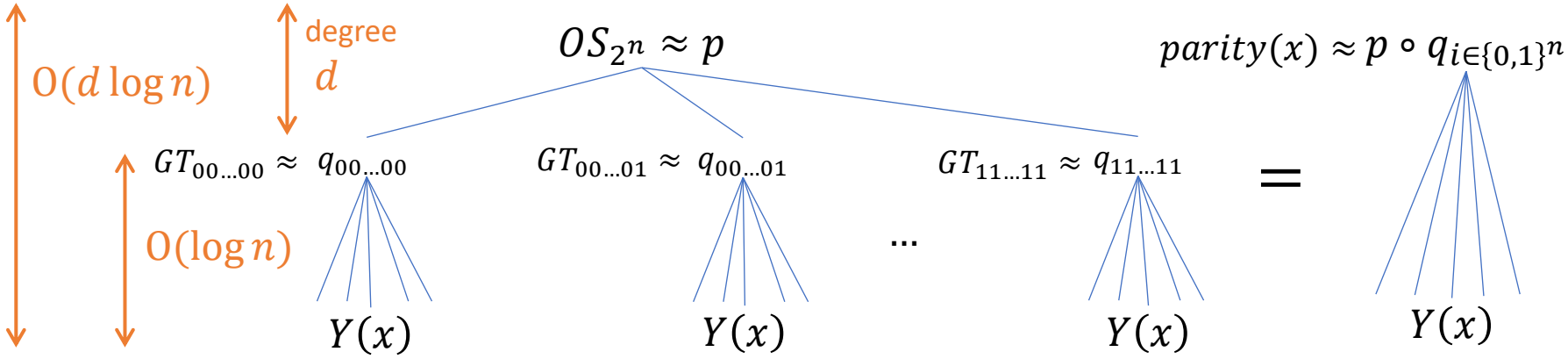
Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(n/\log^2 n)$.



Claim 1. $\forall i \in \{0,1\}^n$ there exists a polynomial of $Y(x)$ of degree $O(\log n)$ that approximates $GT_i(x)$.

Our argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(n/\log^2 n)$.

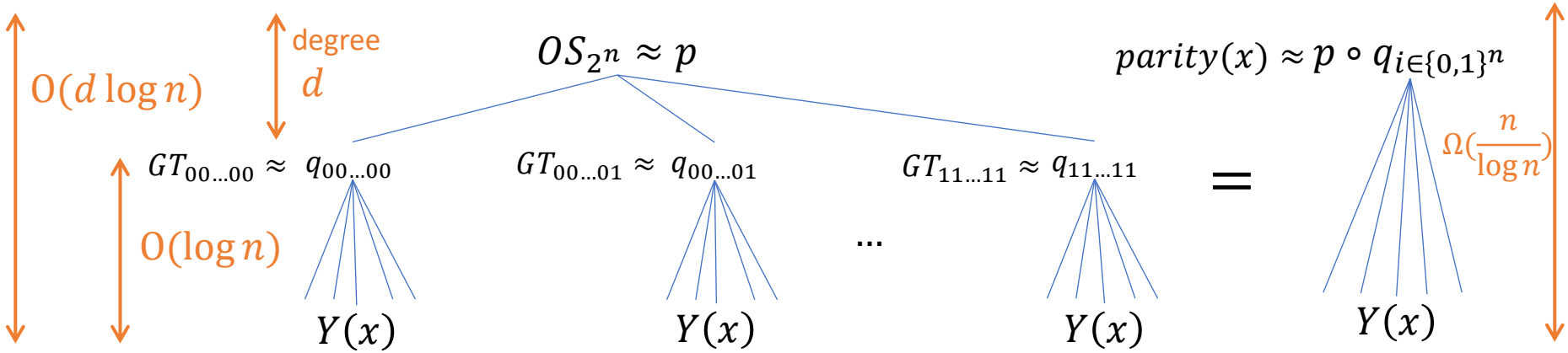


Claim 1. $\forall i \in \{0,1\}^n$ there exists a polynomial of $Y(x)$ of degree $O(\log n)$ that approximates $GT_i(x)$.

[She12] Every polynomial can be made robust to constant noise with constant blowup in degree.

Our argument

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(n/\log^2 n)$.



Claim 1. $\forall i \in \{0,1\}^n$ there exists a polynomial of $Y(x)$ of degree $O(\log n)$ that approximates $GT_i(x)$.

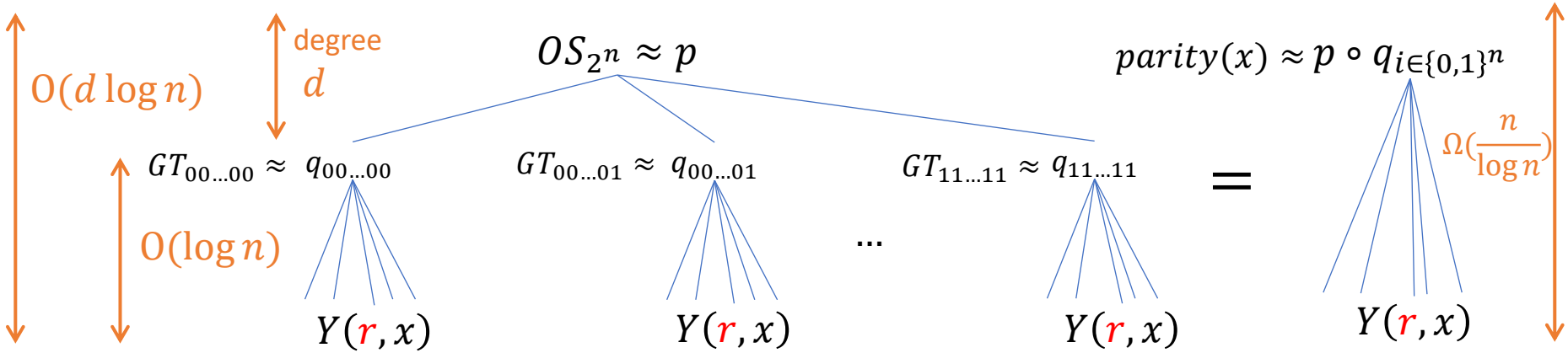
[She12] Every polynomial can be made robust to constant noise with constant blowup in degree.

Claim 2. Every polynomial of $Y(x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Our argument

Sample $r \leftarrow R$

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(n/\log^2 n)$.



Claim 1. W.h.p. $\forall i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n)$ that approximates $GT_i(x)$.

[She12] Every polynomial can be made robust to constant noise with constant blowup in degree.

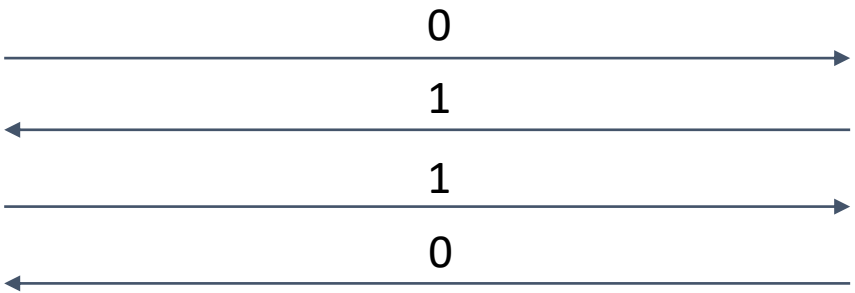
Claim 2. W.h.p. every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Communication complexity

$a \in A$

Random string

$b \in B$



$f(a, b)$

How many bits of communication are required?

GT communication problem

Random string

$a \in \{0,1\}^n$

$b \in \{0,1\}^n$

$GT(a,b) = 1$ iff $a > b$

Nisan'93: $O(\log n)$ protocol

Today: $O(\log n \log\log n)$ protocol

EQ communication problem

Random string

$a \in \{0,1\}^n$

$b \in \{0,1\}^n$

$EQ(a,b) = 1$ iff $a = b$

EQ communication protocol

1	1	0	0	1	1	0	1

$a \in \{0,1\}^n$

Random string

1	1	0	0	0	1	1	1

$b \in \{0,1\}^n$

$\langle a,$  $\rangle$

Compare to

Equal

$\langle b,$  $\rangle$

Only need constant communication for constant error!

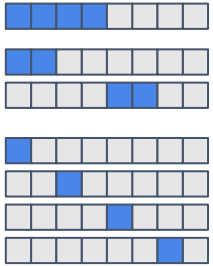
Need $O(\log\log n)$ for $\frac{1}{\log n} = 2^{-\log\log n}$ error.

GT communication protocol using EQ

1	1	0	0	1	1	0	1

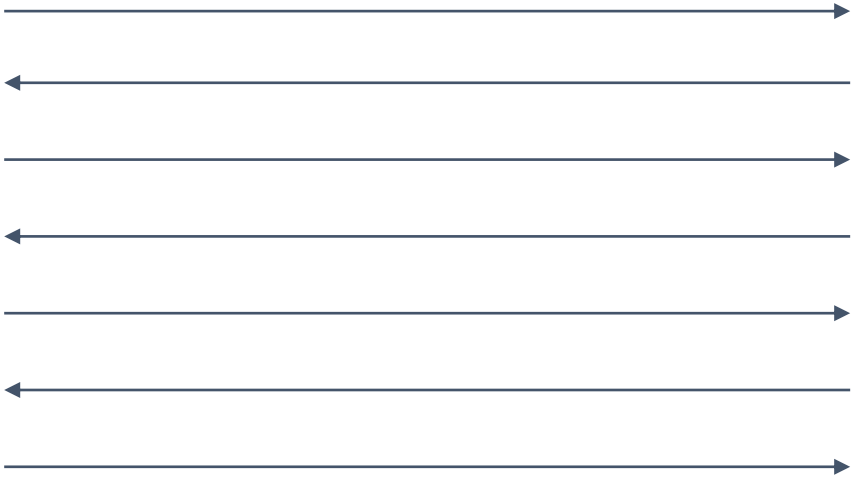
$a \in \{0,1\}^n$

Random strings



1	1	0	0	0	1	1	1

$b \in \{0,1\}^n$

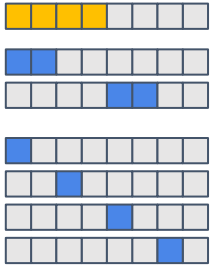


GT communication protocol using EQ

1 1 0 0 1 1 0 1

$a \in \{0,1\}^n$

Random strings



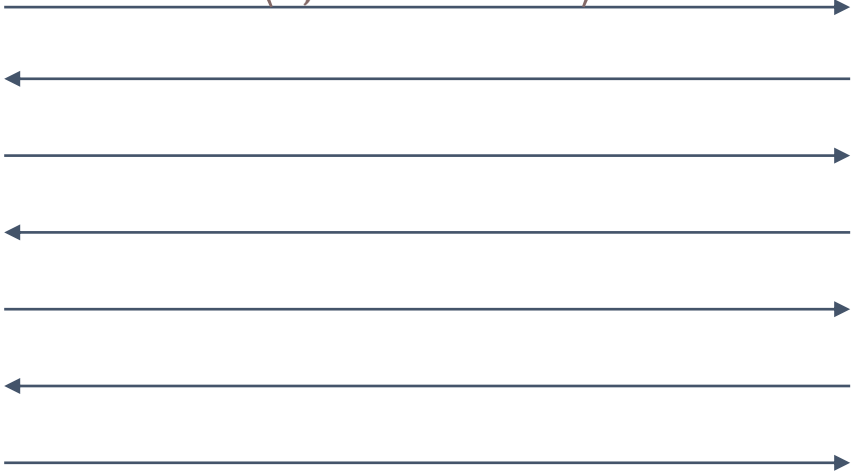
1 1 0 0 0 1 1 1

$b \in \{0,1\}^n$

Compare to

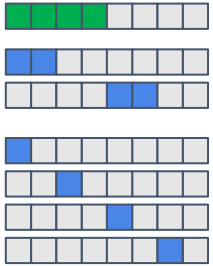
$\langle b,$  $\rangle$

$\langle a,$  $\rangle$



GT communication protocol using EQ

Random strings

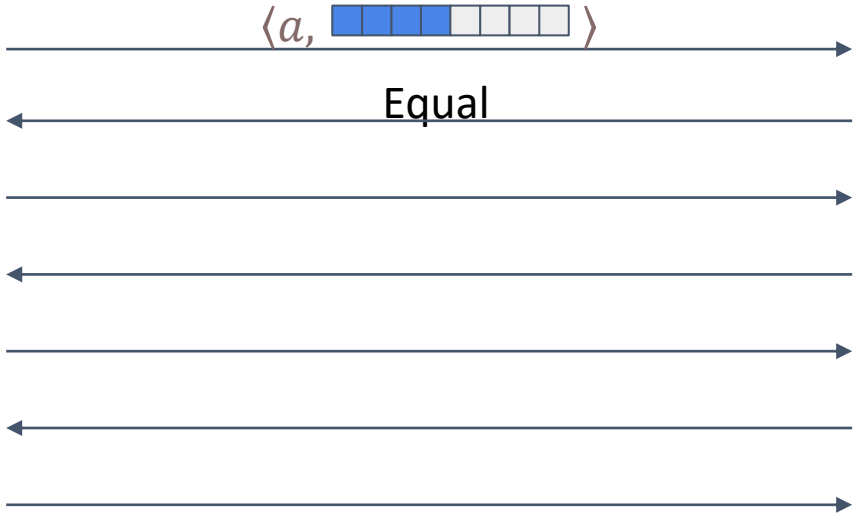


1 1 0 0 1 1 0 1

$a \in \{0,1\}^n$

1 1 0 0 0 1 1 1

$b \in \{0,1\}^n$

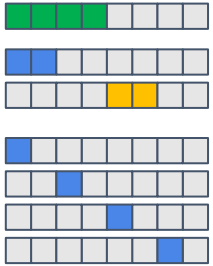


GT communication protocol using EQ

1 1 0 0 1 1 0 1

$a \in \{0,1\}^n$

Random strings



1 1 0 0 0 1 1 1

$b \in \{0,1\}^n$

$\langle a,$  $\rangle$

Equal

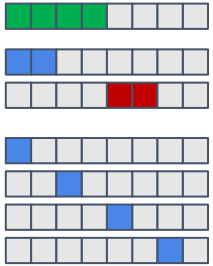
Compare to

$\langle a,$  $\rangle$

$\langle b,$  $\rangle$

GT communication protocol using EQ

Random strings

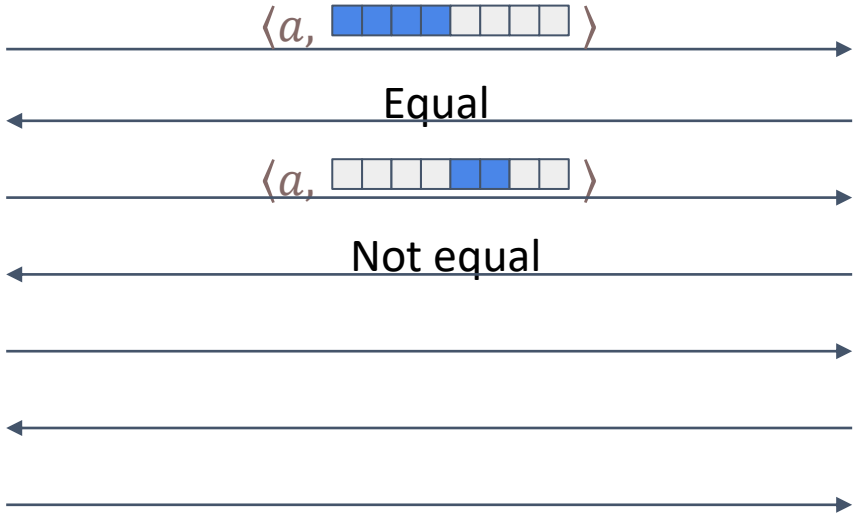


1	1	0	0	1	1	0	1

$a \in \{0,1\}^n$

1	1	0	0	0	1	1	1

$b \in \{0,1\}^n$

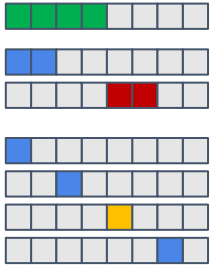


GT communication protocol using EQ

1 1 0 0 1 1 0 1

$a \in \{0,1\}^n$

Random strings



1 1 0 0 0 1 1 1

$b \in \{0,1\}^n$

$\langle a,$  $\rangle$

Equal

$\langle a,$  $\rangle$

Not equal

$\langle a,$  $\rangle$

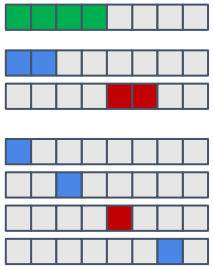
Compare to

$\langle b,$  $\rangle$

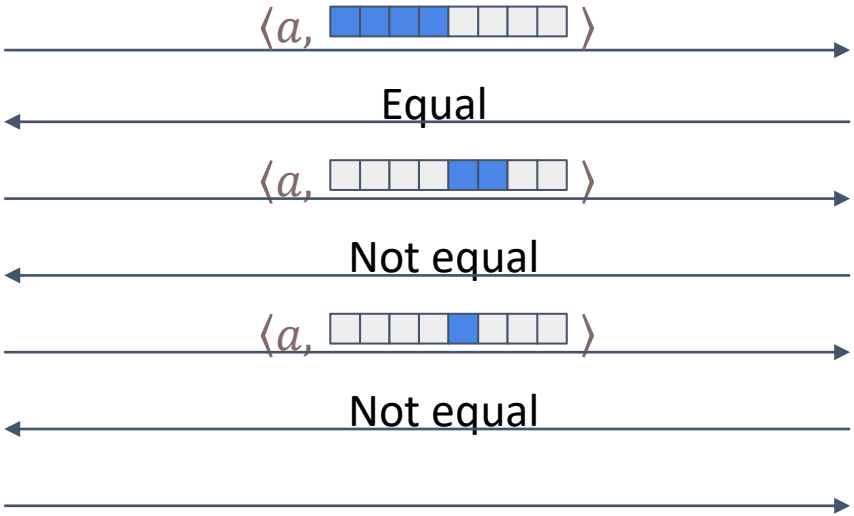
GT communication protocol using EQ

$a \in \{0,1\}^n$

Random strings



$b \in \{0,1\}^n$

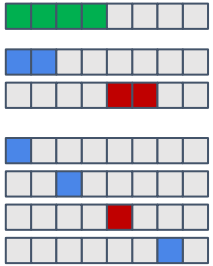


GT communication protocol using EQ

1	1	0	0	1	1	0	1

$a \in \{0,1\}^n$

Random strings



1	1	0	0	0	1	1	1

$b \in \{0,1\}^n$

$\langle a,$  $\rangle$

Equal

$\langle a,$  $\rangle$

Not equal

$\langle a,$  $\rangle$

Not equal

$\langle a,$  $\rangle$

Compare to

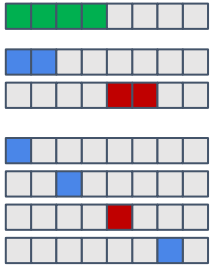
$\langle b,$  $\rangle$

GT communication protocol using EQ

1 1 0 0 1 1 0 1

$a \in \{0,1\}^n$

Random strings



1 1 0 0 0 1 1 1

$b \in \{0,1\}^n$

$\langle a,$  $\rangle$

Equal

$\langle a,$  $\rangle$

Not equal

$\langle a,$  $\rangle$

Not equal

$\langle a,$  $\rangle$

Compare to

$\langle b,$  $\rangle$

Communication complexity $O(\log n \log\log n)$

First attempt

$$Y(x) = (\langle r', x \rangle)_{r' \in \{0,1\}^n}$$

The oracle Y contains all the possible partial parities of x.

First attempt

$$Y(x) = (\langle r', x \rangle)_{r' \in \{0,1\}^n}$$

The oracle Y contains all the possible partial parities of x.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

First attempt

$$Y(x) = (\langle r', x \rangle)_{r' \in \{0,1\}^n}$$

The oracle Y contains all the possible partial parities of x.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

- There exists a randomized algorithm that given i (hardcoded) and access to $Y(x)$, outputs $GT_i(x)$ with Pr $2/3$ with query complexity $O(\log n \log\log n)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

First attempt

$$Y(x) = (\langle r', x \rangle)_{r' \in \{0,1\}^n}$$

The oracle Y contains all the possible partial parities of x.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

- There exists a randomized algorithm that given i (hardcoded) and access to $Y(x)$, outputs $GT_i(x)$ with Pr $2/3$ with query complexity $O(\log n \log\log n)$.
- Every randomized query algorithm converts to a polynomial of the same degree as query complexity.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

First attempt

$Y(x) = (\langle r', x \rangle)_{r' \in \{0,1\}^n}$

The oracle Y contains all the possible partial parities of x.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

- There exists a randomized algorithm that given i (hardcoded) and access to $Y(x)$, outputs $GT_i(x)$ with Pr $2/3$ with query complexity $O(\log n \log\log n)$.
- Every randomized query algorithm converts to a polynomial of the same degree as query complexity.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

First attempt

$Y(x) = (\langle r', x \rangle)_{r' \in \{0,1\}^n}$

The oracle Y contains all the possible partial parities of x.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

- There exists a randomized algorithm that given i (hardcoded) and access to $Y(x)$, outputs $GT_i(x)$ with Pr $2/3$ with query complexity $O(\log n \log\log n)$.
- Every randomized query algorithm converts to a polynomial of the same degree as query complexity.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Exists a polynomial of degree 1: $parity(x) = \langle 1^n, x \rangle$

First attempt

$Y(x) = (\langle r', x \rangle)_{r' \in \{0,1\}^n}$

The oracle Y contains all the possible partial parities of x.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

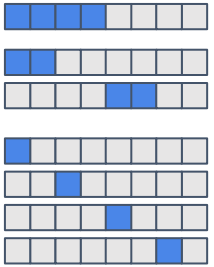
- There exists a randomized algorithm that given i (hardcoded) and access to $Y(x)$, outputs $GT_i(x)$ with Pr $2/3$ with query complexity $O(\log n \log\log n)$.
- Every randomized query algorithm converts to a polynomial of the same degree as query complexity.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Exists a polynomial of degree 1: $parity(x) = \langle 1^n, x \rangle$

Second attempt

$r \leftarrow R$



$$Y(r,x) = (\langle r', x \rangle)_{r' \in r}$$

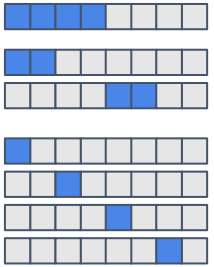
The oracle Y has enough info for one full run of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Second attempt

$r \leftarrow R$



$$Y(r,x) = (\langle r', x\rangle)_{r' \in r}$$

The oracle Y has enough info for one full run of GT.

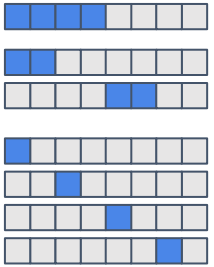
Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Not enough information to reconstruct $parity(x)$

Second attempt

$r \leftarrow R$



$$Y(r,x) = (\langle r', x\rangle)_{r' \in r}$$

The oracle Y has enough info for one full run of GT.

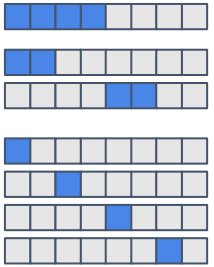
Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Not enough information to reconstruct $parity(x)$

Second attempt

$r \leftarrow R$



$Y(r, x) = (\langle r', x \rangle)_{r' \in r}$

The oracle Y has enough info for one full run of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r, x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

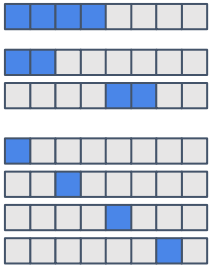
- There exists a deterministic algorithm that for all $i, x \in \{0,1\}^n$ with Pr $2/3$ outputs $GT_i(x)$ with query complexity $O(\log n \log\log n)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r, x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Not enough information to reconstruct $parity(x)$

Second attempt

$r \leftarrow R$



$$Y(r,x) = (\langle r', x \rangle)_{r' \in r}$$

The oracle Y has enough info for one full run of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

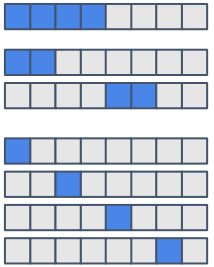
- There exists a deterministic algorithm that for all $i, x \in \{0,1\}^n$ with Pr $2/3$ outputs $GT_i(x)$ with query complexity $O(\log n \log\log n)$.
- Wrong order!

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Not enough information to reconstruct $parity(x)$

Second attempt

$r \leftarrow R$



$$Y(r,x) = (\langle r', x \rangle)_{r' \in r}$$

The oracle Y has enough info for one full run of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

- There exists a deterministic algorithm that for all $i, x \in \{0,1\}^n$ with Pr 2/3 outputs $GT_i(x)$ with query complexity $O(\log n \log\log n)$.
- Wrong order!

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Not enough information to reconstruct $parity(x)$

Final attempt

$r \leftarrow R$

$$Y(r,x) = (\langle r', x \rangle)_{r' \in r}$$

The oracle Y has enough info for $t = poly(n)$ full runs of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Final attempt

$r \leftarrow R$

$$Y(r,x) = (\langle r', x \rangle)_{r' \in r}$$

The oracle Y has enough info for $t = poly(n)$ full runs of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Still not enough information to reconstruct $parity(x)$

Final attempt

$r \leftarrow R$

$$Y(r,x) = (\langle r', x \rangle)_{r' \in r}$$

The oracle Y has enough info for $t = poly(n)$ full runs of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Still not enough information to reconstruct $parity(x)$

Final attempt

$r \leftarrow R$

$$Y(r,x) = (\langle r', x \rangle)_{r' \in r}$$

The oracle Y has enough info for $t = poly(n)$ full runs of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

- There exists a randomized algorithm that with Pr $2/3$ over $r \leftarrow R$ for all $i, x \in \{0,1\}^n$ outputs $GT_i(x)$ with Pr 2/3 over internal randomness with query complexity $O(\log n \log\log n)$.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Still not enough information to reconstruct $parity(x)$

Final attempt

$r \leftarrow R$

$$Y(r,x) = (\langle r', x\rangle)_{r' \in r}$$

The oracle Y has enough info for $t = poly(n)$ full runs of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

- There exists a randomized algorithm that with Pr $2/3$ over $r \leftarrow R$ for all $i, x \in \{0,1\}^n$ outputs $GT_i(x)$ with Pr 2/3 over internal randomness with query complexity $O(\log n \log\log n)$.
- Every randomized query algorithm converts to a polynomial of the same degree as query complexity.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Still not enough information to reconstruct $parity(x)$

Final attempt

$r \leftarrow R$

$$Y(r,x) = (\langle r', x\rangle)_{r' \in r}$$

The oracle Y has enough info for $t = poly(n)$ full runs of GT.

Claim 1. With probability 2/3 over the choice of $r \leftarrow R$ for each $i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n \log\log n)$ that approximates $GT_i(x)$.

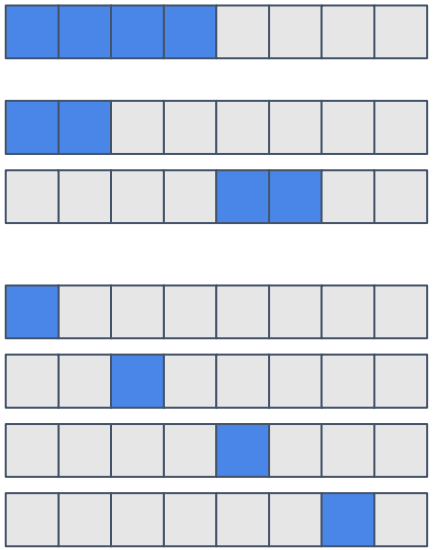
- There exists a randomized algorithm that with Pr $2/3$ over $r \leftarrow R$ for all $i, x \in \{0,1\}^n$ outputs $GT_i(x)$ with Pr 2/3 over internal randomness with query complexity $O(\log n \log\log n)$.
- Every randomized query algorithm converts to a polynomial of the same degree as query complexity.

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Still not enough information to reconstruct $parity(x)$

Oracle structure

$\alpha = O(\log \log n)$ copies



$$\big(Y(r,x)\big)_i = \langle r_i, x \rangle$$

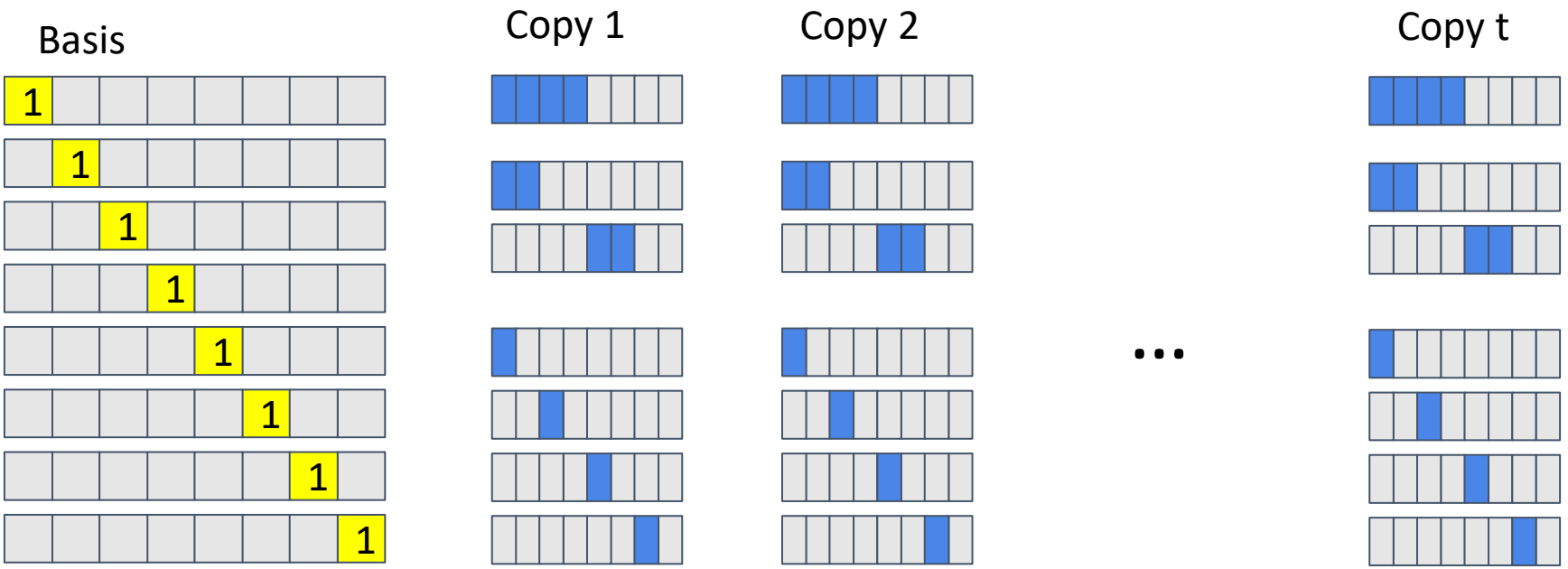
$$r = (r_1, r_2, \ldots, r_m), r_i \in \{0,1\}^n$$

$$r \leftarrow R$$

Random bit

Zero

Updated oracle structure R

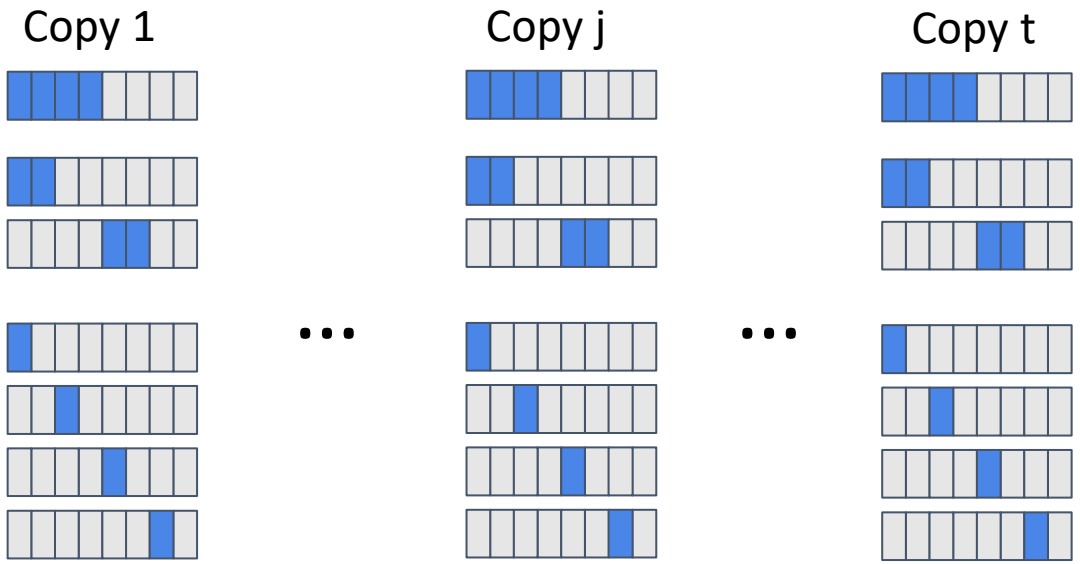


$t = poly(n)$

Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

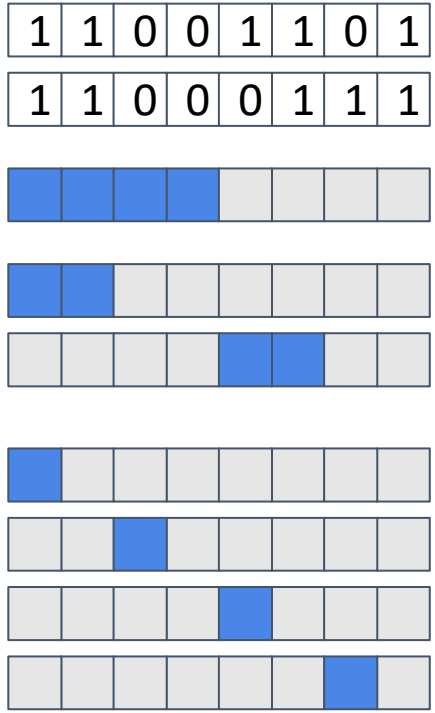
1. Sample $j \leftarrow [t]$ u.a.r



Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j

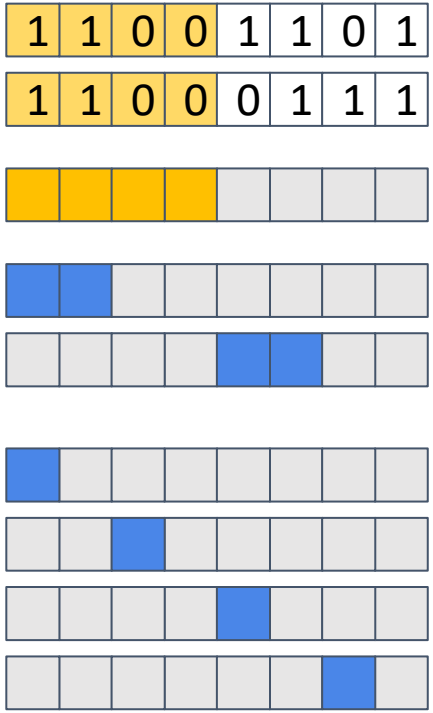


0 r 1

Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

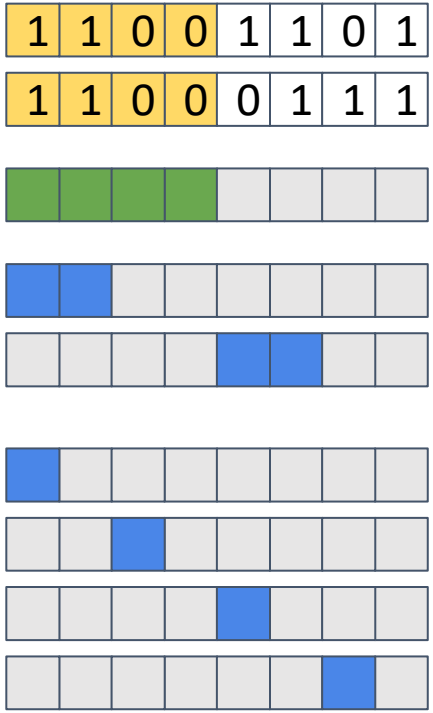
1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values



Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

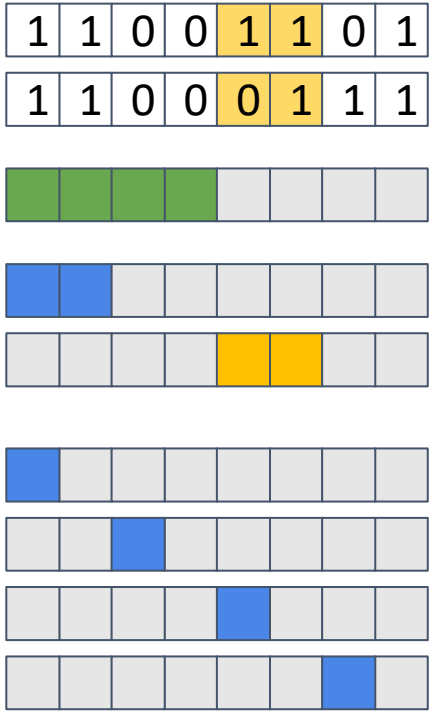
1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values



Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

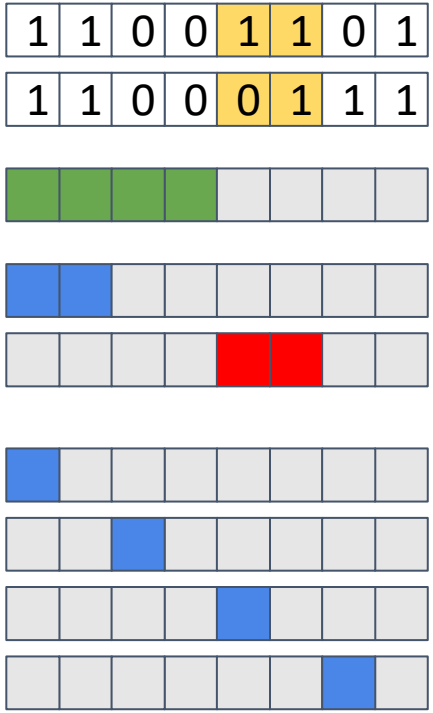
1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values



Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

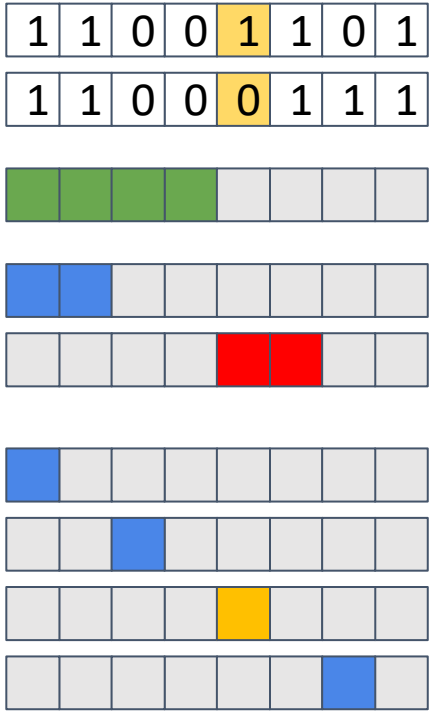
1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values



Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

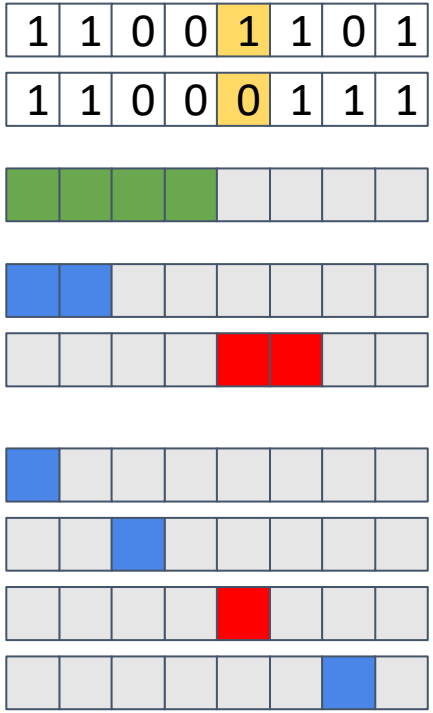
1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values



Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

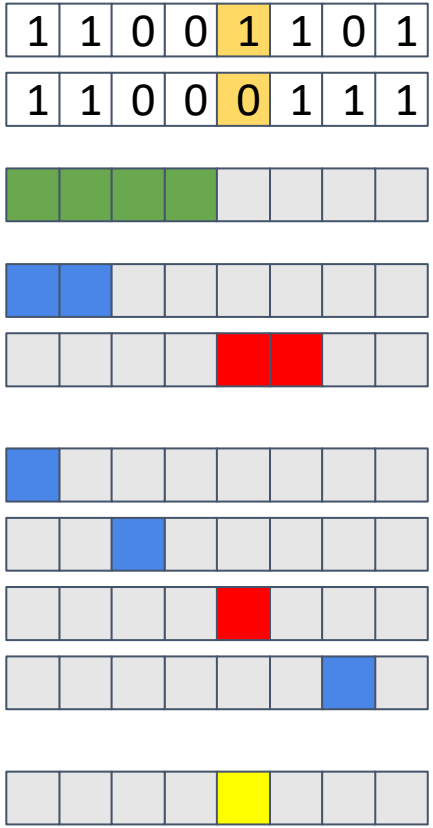
1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values



Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
3. Query most significant bit and compare it to the value in i
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values

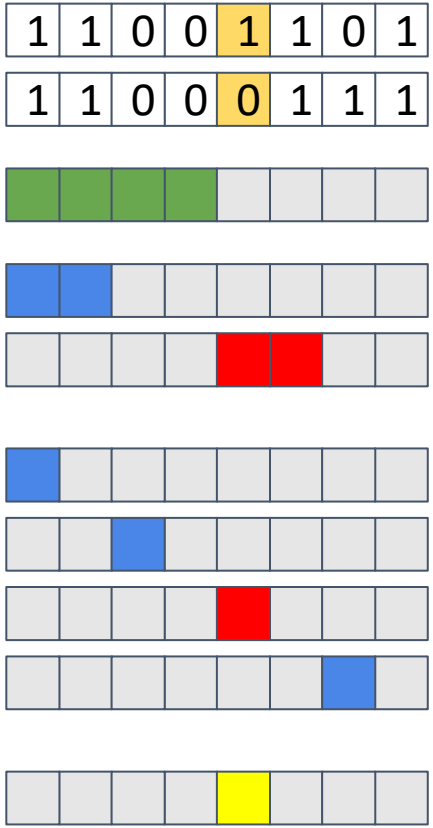


Query algorithm for GT_i

Input: oracle access to $Y(r, x)$

1. Sample $j \leftarrow [t]$ u.a.r
2. Simulate communication protocol using inner products from copy j
3. Query most significant bit and compare it to the value in i
   - Compute $\langle i,$  $\rangle$
   - Query $\langle x,$  $\rangle$
   - Compare values

Query complexity $O(\log n \log\log n)$



Upper bound for GT_i

Claim 1'. With probability 2/3 over $r \leftarrow R$ u.a.r. for all $i \in \{0,1\}^n$ there exists an algorithm A_i with oracle access to $Y(r,x)$ that compute the corresponding GT_i with probability 2/3 over internal randomness and has query complexity at most $O(\log n \log\log n)$.

Acceptance probability of a randomized T-query algorithm is a polynomial of degree T.

Claim 1. With probability 2/3 over $r \leftarrow R$ u.a.r. for all $i \in \{0,1\}^n$ there exists a polynomial q_i of $Y(r,x)$ that approximate the corresponding GT_i and has degree at most $O(\log n \log\log n)$.

Upper bound for GT_i

Claim 1'. With probability 2/3 over $r \leftarrow R$ u.a.r. for all $i \in \{0,1\}^n$ there exists an algorithm A_i with oracle access to $Y(r,x)$ that compute the corresponding GT_i with probability 2/3 over internal randomness and has query complexity at most $O(\log n)$.

Acceptance probability of a randomized T-query algorithm is a polynomial of degree T.

Claim 1. With probability 2/3 over $r \leftarrow R$ u.a.r. for all $i \in \{0,1\}^n$ there exists a polynomial q_i of $Y(r,x)$ that approximate the corresponding GT_i and has degree at most $O(\log n)$.

Lower bound for parity

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r, x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

- Generalizes for any R with a "good" structure! $\Omega(\frac{n}{\log(size\ of\ oracle\ Y)})$
- Works for approximation to any error

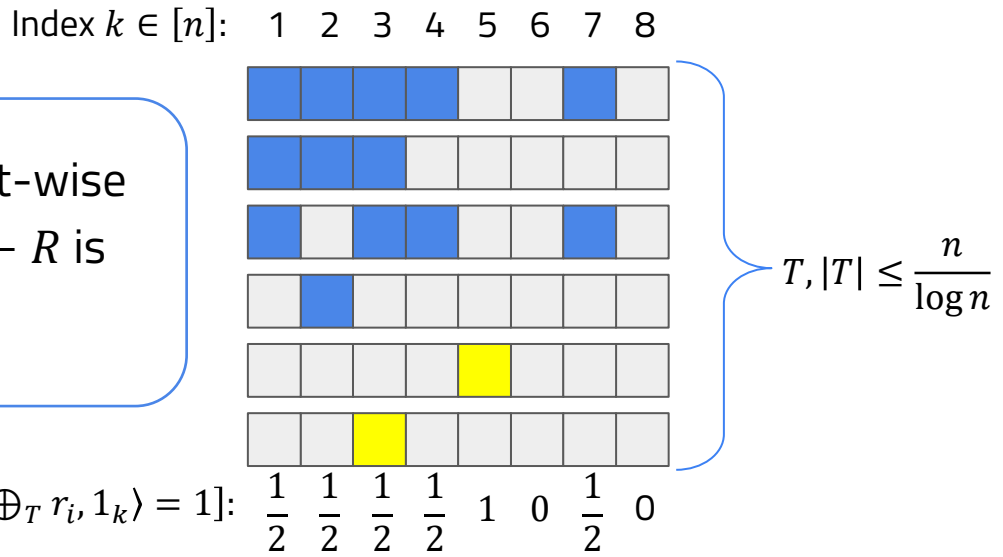
Lower bound for parity

Claim 2. With probability 2/3 over the choice of $r \leftarrow R$ every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

- **Proof idea 1:** We consider polynomials over $\{-1,1\}$ instead of $\{0,1\}$
- **Proof idea 2:** All monomials of degree $< n$ are orthogonal to parity in $\{-1,1\}$ basis.
- **Proof idea 3:** Getting a monomial of x of degree n by multiplying $\frac{n}{\log n}$ bits of $Y(r,x)$ is improbable.

Lower bound for parity: key lemma

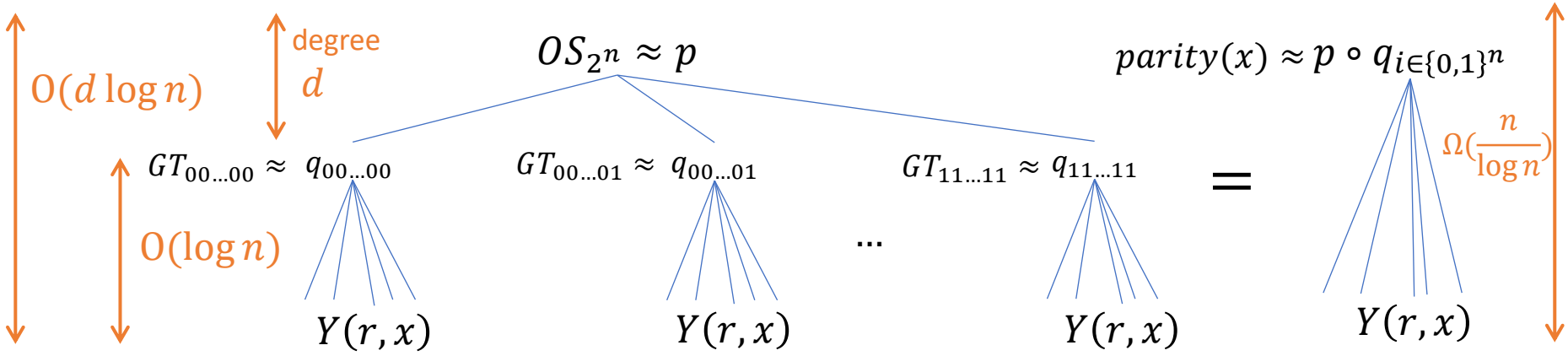
Getting the all-ones string by taking bit-wise XOR of $\frac{n}{\log n}$ strings from a sample $r \leftarrow R$ is improbable over the choice of r.



Final result for OS

Sample $r \leftarrow R$

Theorem. Every polynomial that approximates OS_{2^n} requires degree $\Omega(n/\log^2 n)$.



Claim 1. W.h.p. $\forall i \in \{0,1\}^n$ there exists a polynomial of $Y(r,x)$ of degree $O(\log n)$ that approximates $GT_i(x)$.

[She12] Every polynomial can be made robust to constant noise with constant blowup in degree.

Claim 2. W.h.p. Every polynomial of $Y(r,x)$ that approximates $parity(x)$ requires degree $\Omega(n/\log n)$.

Summary

- New (and almost tight) lower bound for approximate degree of Ordered search: $\Omega(n/\log^2 n)$
- Generalizable to unbounded error regime: $\Omega(\frac{n}{\log^2 n} - \log\frac{1}{\gamma})$
- Same lower bounds for Hidden string problem using the same approach
- As a corollary, lower bounds on quantum query complexity of decision versions
- New framework for lower bounds on oracle identification problems

Open problems:

- Using the easiness of one problem in the presence of additional information to prove the hardness of another
- Using this framework for other open problems
- Using this framework in other settings: circuit complexity, proof complexity, massive parallel computation model, …
- Closing the gap for ordered search and hidden string: $\Omega(n/\log^2 n)$ and $O(n)$

Summary

- New (and almost tight) lower bound for approximate degree of Ordered search: $\Omega(n/\log^2 n)$
- Generalizable to unbounded error regime: $\Omega(\frac{n}{\log^2 n} - \log\frac{1}{\gamma})$
- Same lower bounds for Hidden string problem using the same approach
- As a corollary, lower bounds on quantum query complexity of decision versions
- New framework for lower bounds on oracle identification problems

Thank you!

Open problems:

- Using the easiness of one problem in the presence of additional information to prove the hardness of another
- Using this framework for other open problems
- Using this framework in other settings: circuit complexity, proof complexity, massive parallel computation model, …
- Closing the gap for ordered search and hidden string: $\Omega(n/\log^2 n)$ and $O(n)$