Quantum Adversary (Upper) Bound

Shelby Kimmel

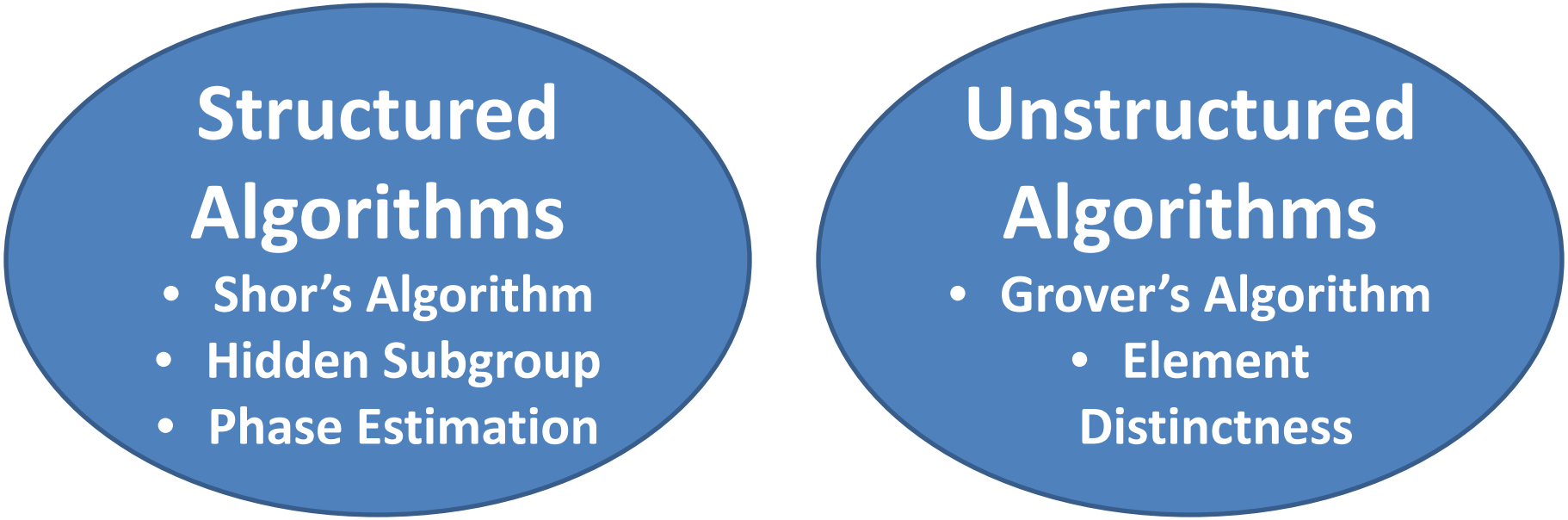
Massachusetts Institute of Technology

arXiv:1101.0797

Big Goal:

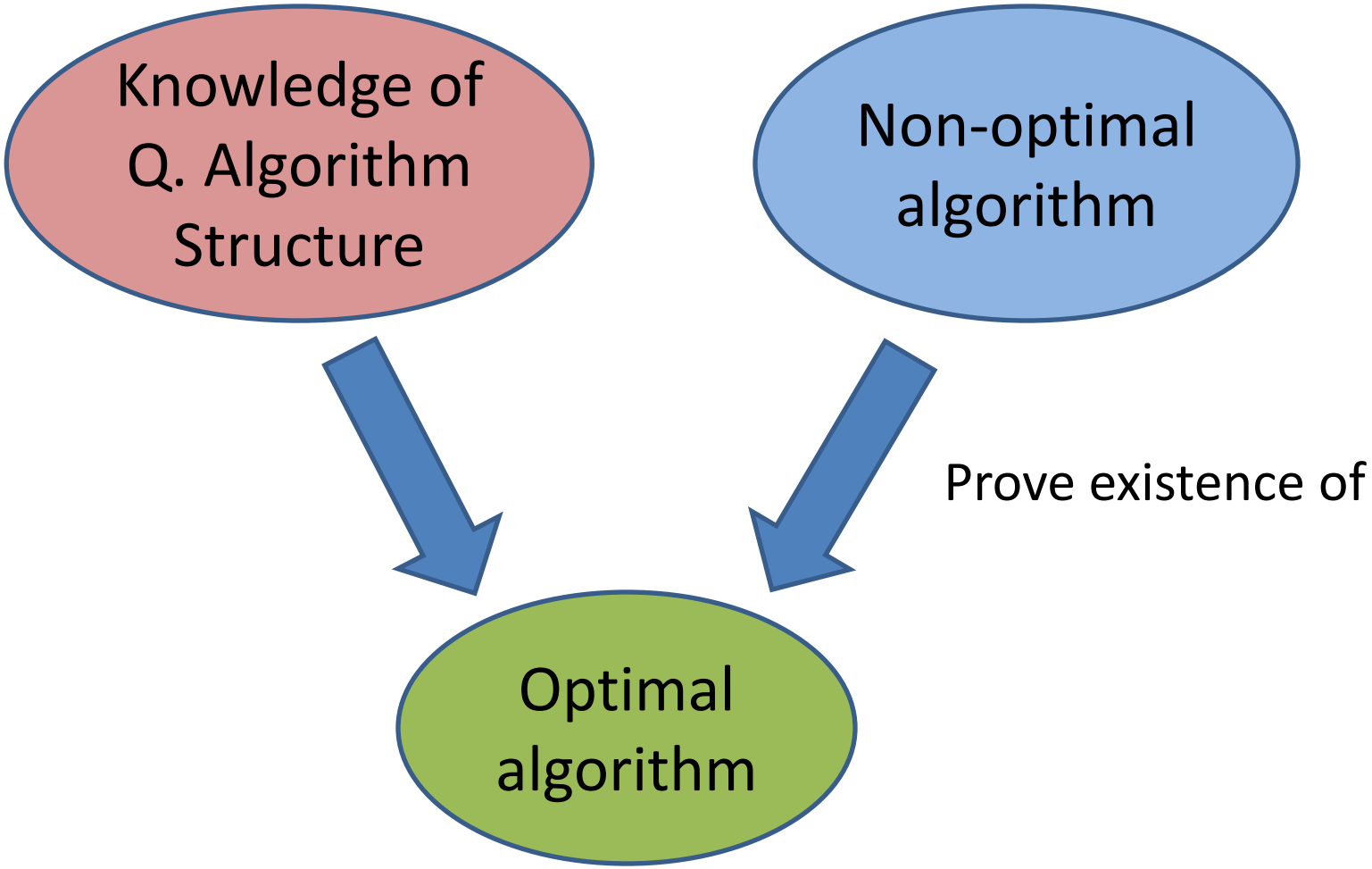
Design new quantum algorithms

Types of Quantum Algorithms



By understanding the structure underlying quantum algorithms, can we find and design new algorithms?

Result


Knowledge of Q. Algorithm Structure
Non-optimal algorithm
Prove existence of
Optimal algorithm

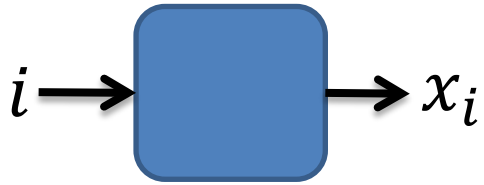
Outline

- Oracle Model and Query Complexity
- Quantum Adversary (Upper) Bound
- Application

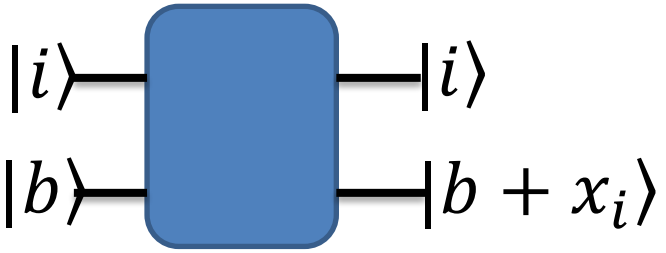
Oracle Model

Goal: Determine the value of $f(x_1, \ldots, x_n)$ for a known function *f*, with an oracle for *x*

Classical Oracle



Quantum Oracle



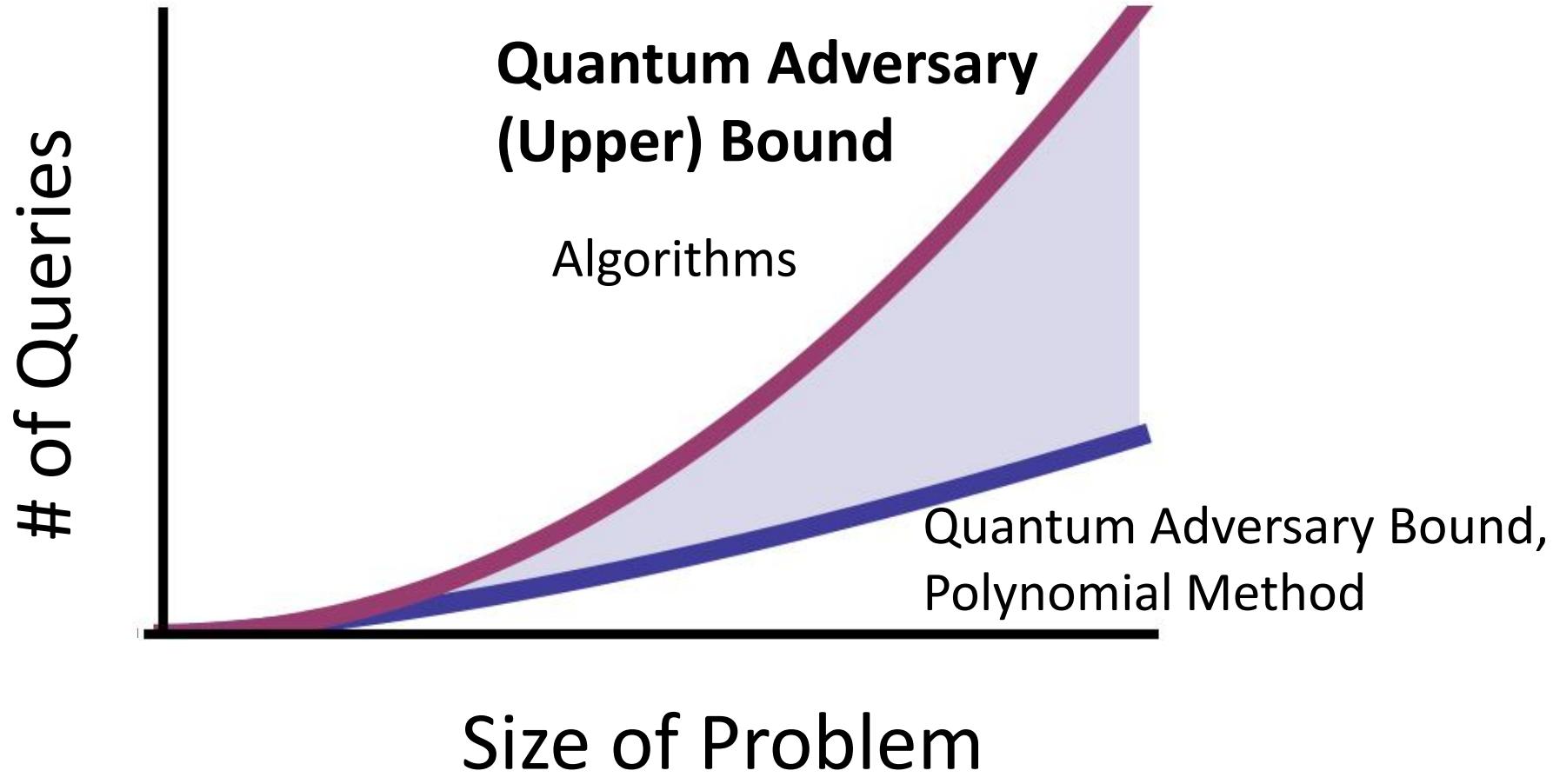
$\mathrm{Q}(f)$
(quantum bounded error query complexity)

Only care about # of oracle calls (queries)

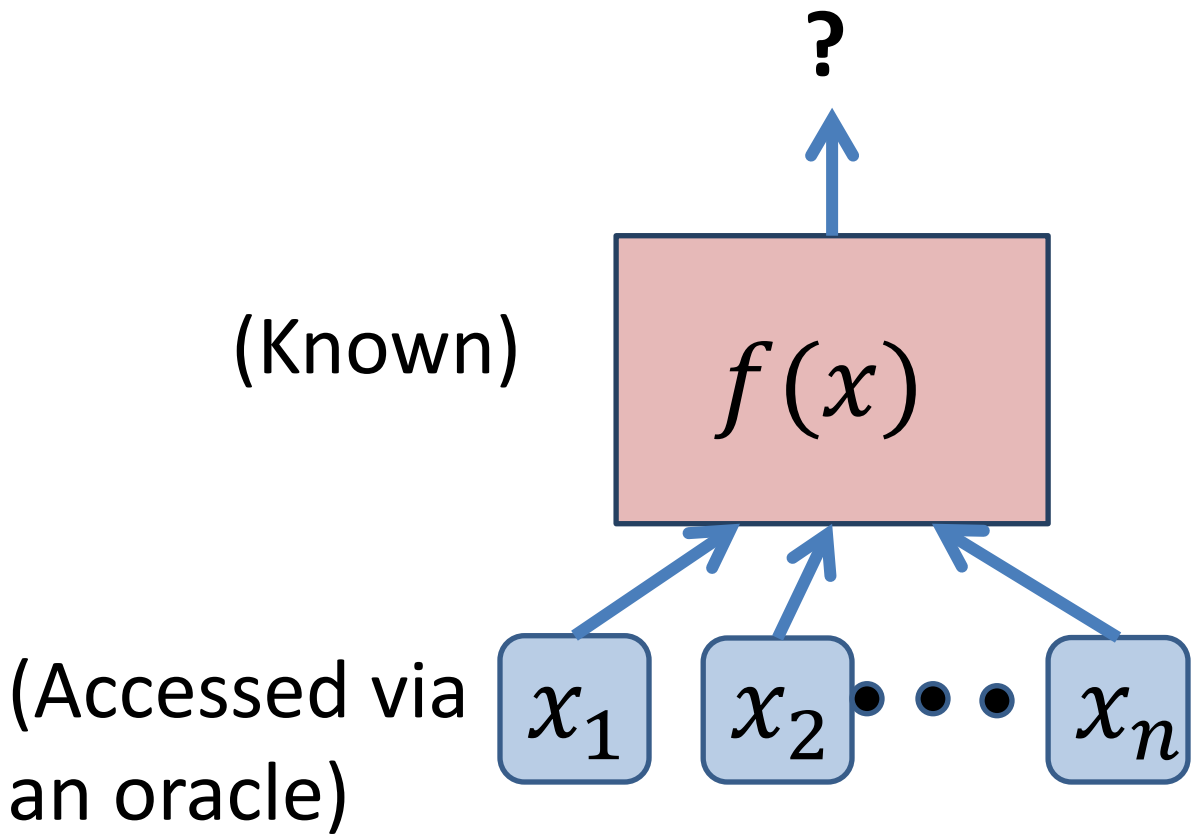
Example

x		f
$\{0,0,\cdots,0,0\}$	→	0
$\{1,1,\cdots,1,1\}$	→	0
$50\%\ 0,\quad 50\%\ 1$	→	1

Query Complexity

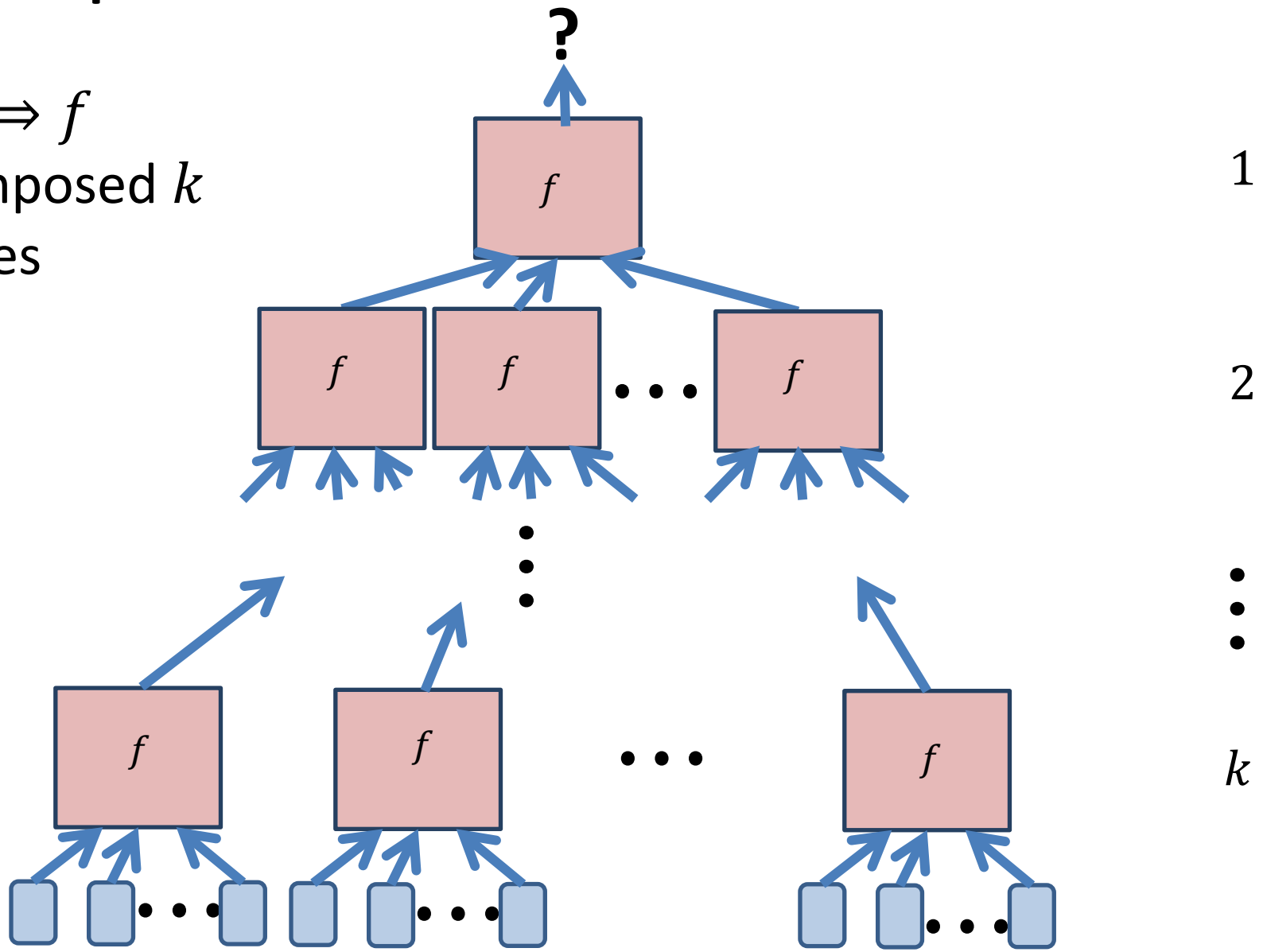


Composed Functions



Composed Functions

$f^k \Rightarrow f$
composed k
times



Quantum Adversary Upper Bound

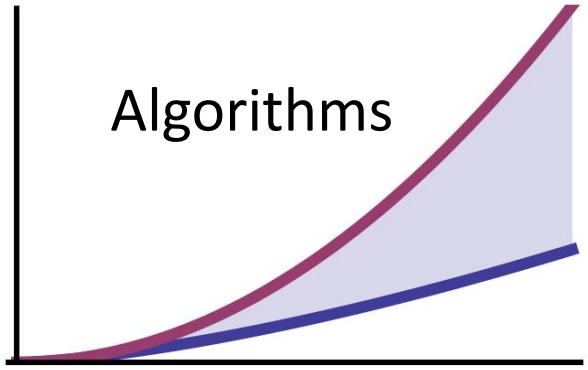
Let f be a Boolean function.

Create an algorithm for f^k, with T queries, so learn $Q(f^k)$ is upper bounded by T.

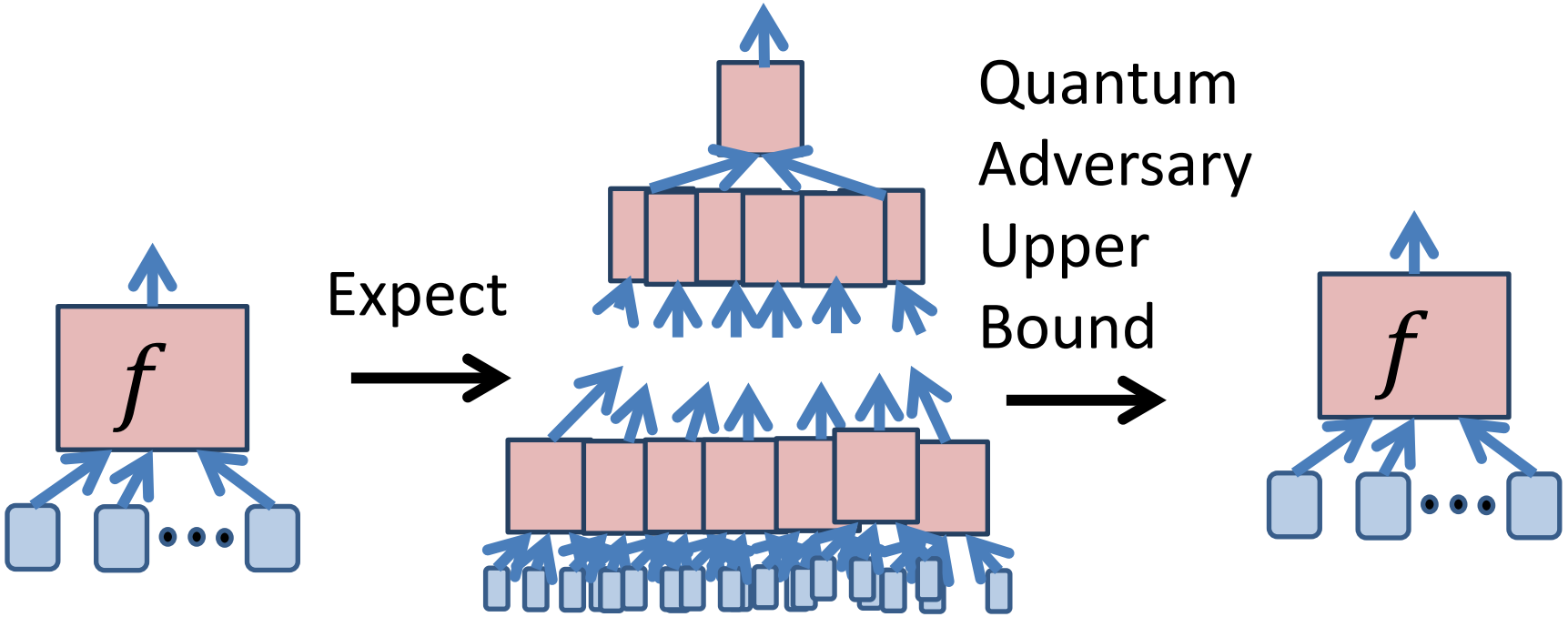
Then the quantum query complexity of f is upper bounded by $T^{1/k}$.

Surprising:

- Does not give algorithm for f
- This is a useful theorem!



Quantum Adversary Upper Bound



$Q(f) \preccurlyeq T$

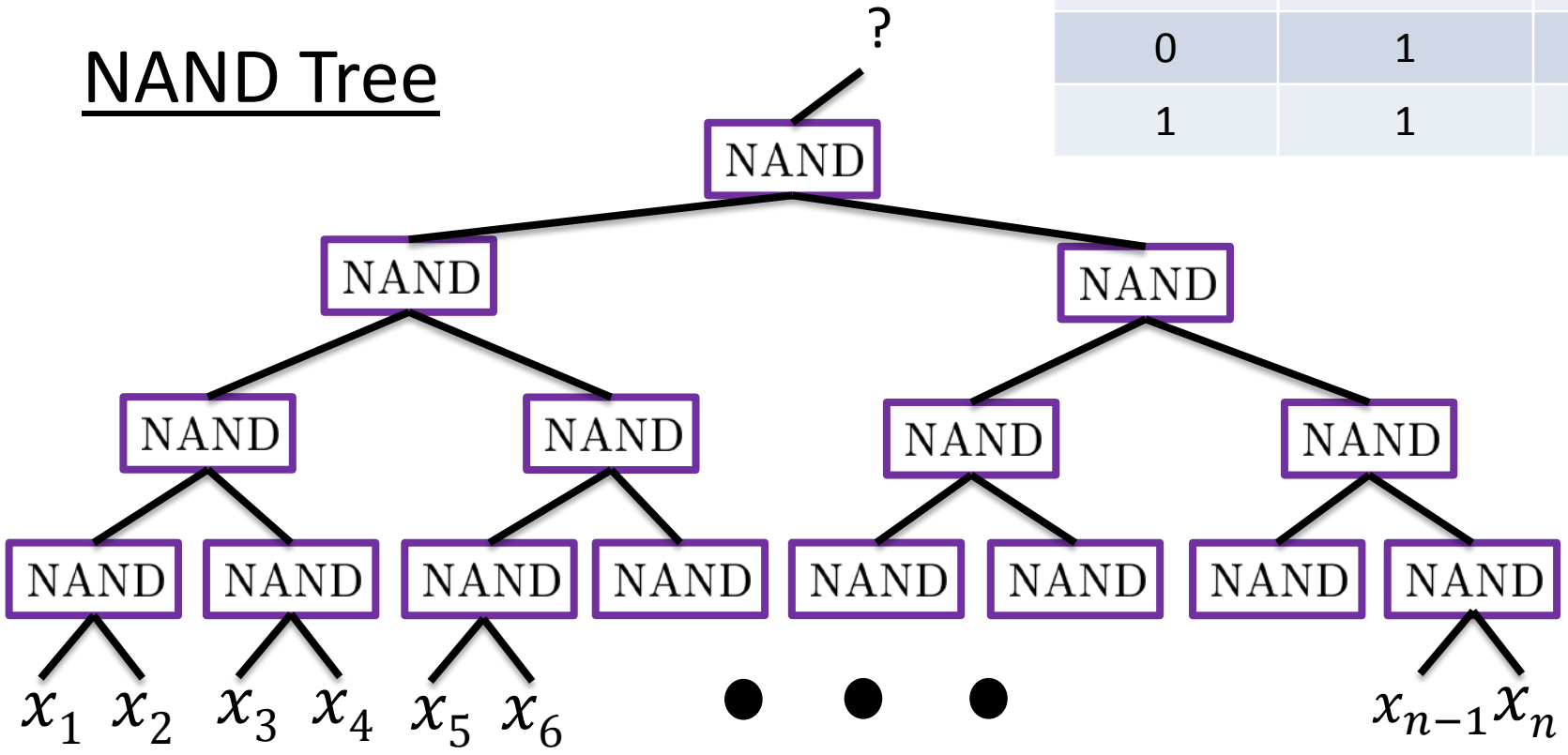
$Q(f^k) \preccurlyeq T^k$

$Q(f) \preccurlyeq T$

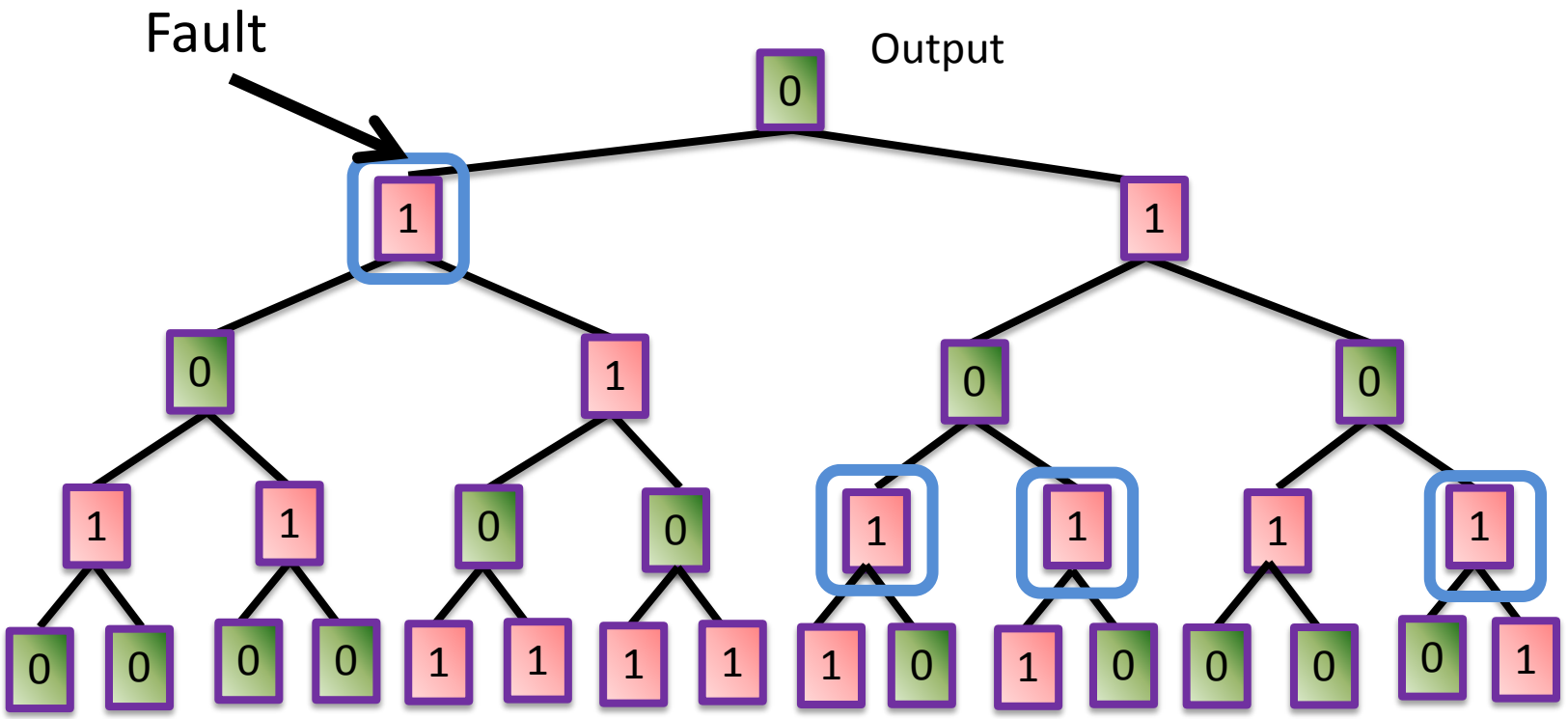
Example: 1-Fault NAND Tree

Input 1	Input 2	NAND
0	0	1
1	0	1
0	1	1
1	1	0

NAND Tree



Example: 1-Fault NAND Tree

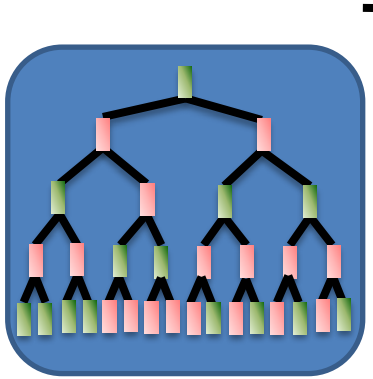


Another view point: 1-Fault NAND Tree is a game tree where the players are promised that they will only have to make one critical decision in the game.

Example: 1-Fault NAND Tree

[Zhan, Hassidim, SK `12]

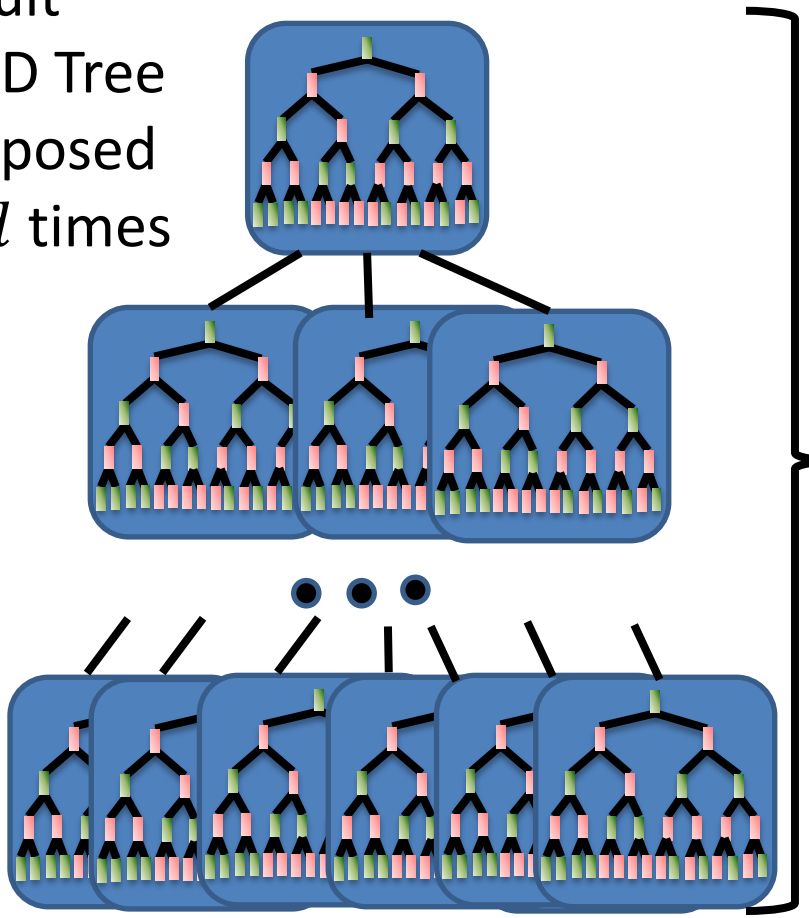
1-Fault NAND Tree



Depth d

$$Q(f) \preccurlyeq d^2$$

1-Fault NAND Tree Composed $\log d$ times



Depth $d \log d$

$$Q\left(f^{\log d}\right) \preccurlyeq d^3$$

Quantum Adversary Upper Bound

$1-\text{Fault NAND Tree}$ is a Boolean function

Quantum query complexity of $[1-\text{Fault NAND Tree}]^{\log d}$ is $\preccurlyeq d^3$

Then the quantum query complexity of $[1-\text{Fault NAND Tree}]$ is

$$d^{3/\log d} = 2^{3\log d/\log d} = constant$$

Proving Quantum Adversary Upper Bound

Lemma 1: $ADV^{\pm}(f) \approx Q(f)$ [Reichardt, '09, '11]

Lemma 2: $ADV^{\pm}(f^k) \geq ADV^{\pm}(f)^k$
[Hoyer, Lee Spalek, '07, SK '11 (for partial functions)]

Proof:

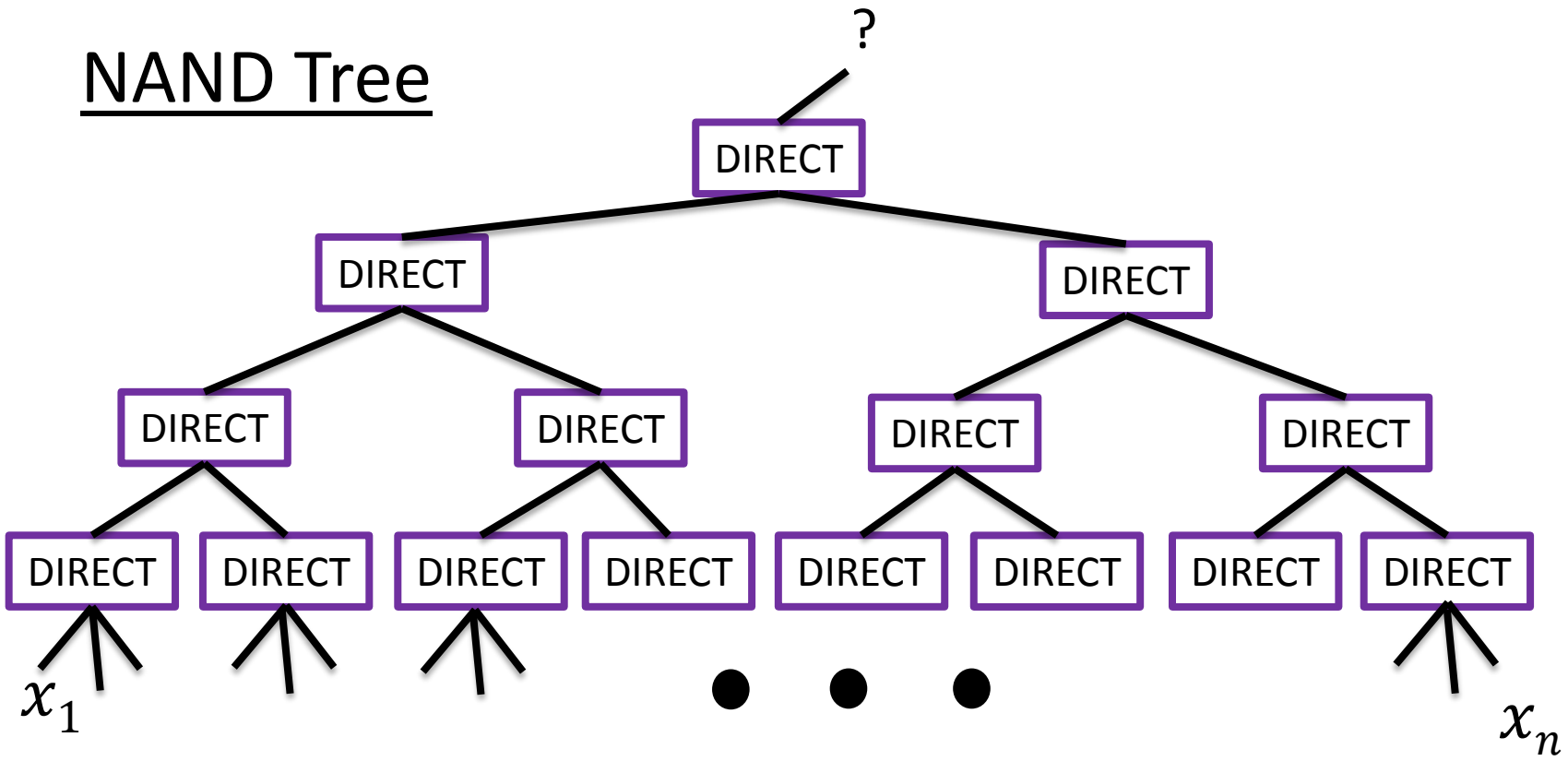
$$Q(f^k) = O(T)$$

$$ADV^{\pm}(f^k) = O(T)$$

$$ADV^{\pm}(f)^k = O(T)$$

$$ADV^{\pm}(f) = O(T^{1/k})$$

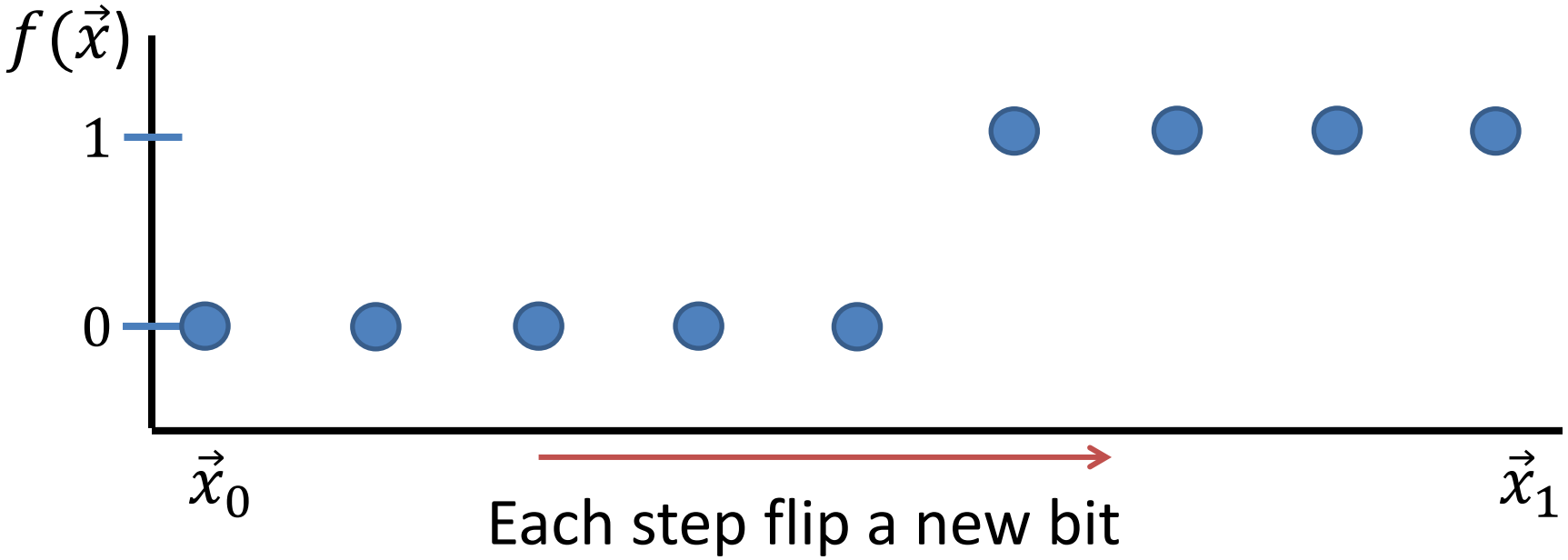
Extension: c-Fault Direct Tree



DIRECT → generalization of monotonic.

Direct Functions

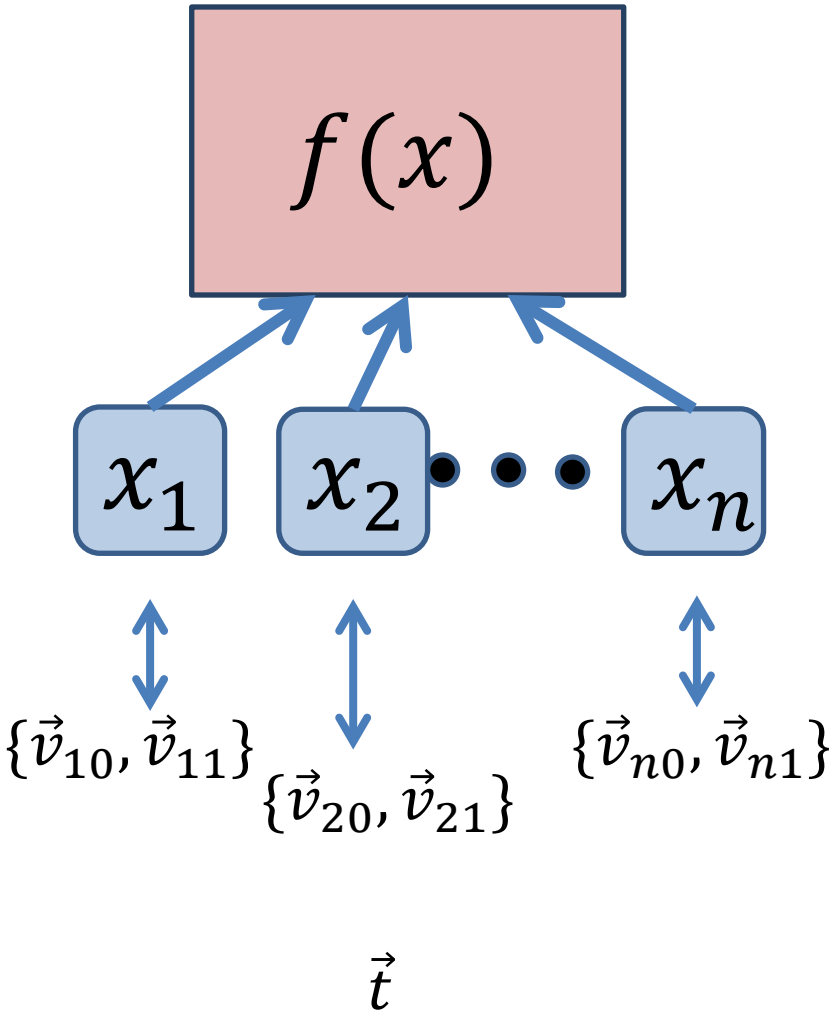
- Examples: Majority, NOT-Majority
- Generalization of monotonic



Algorithm?

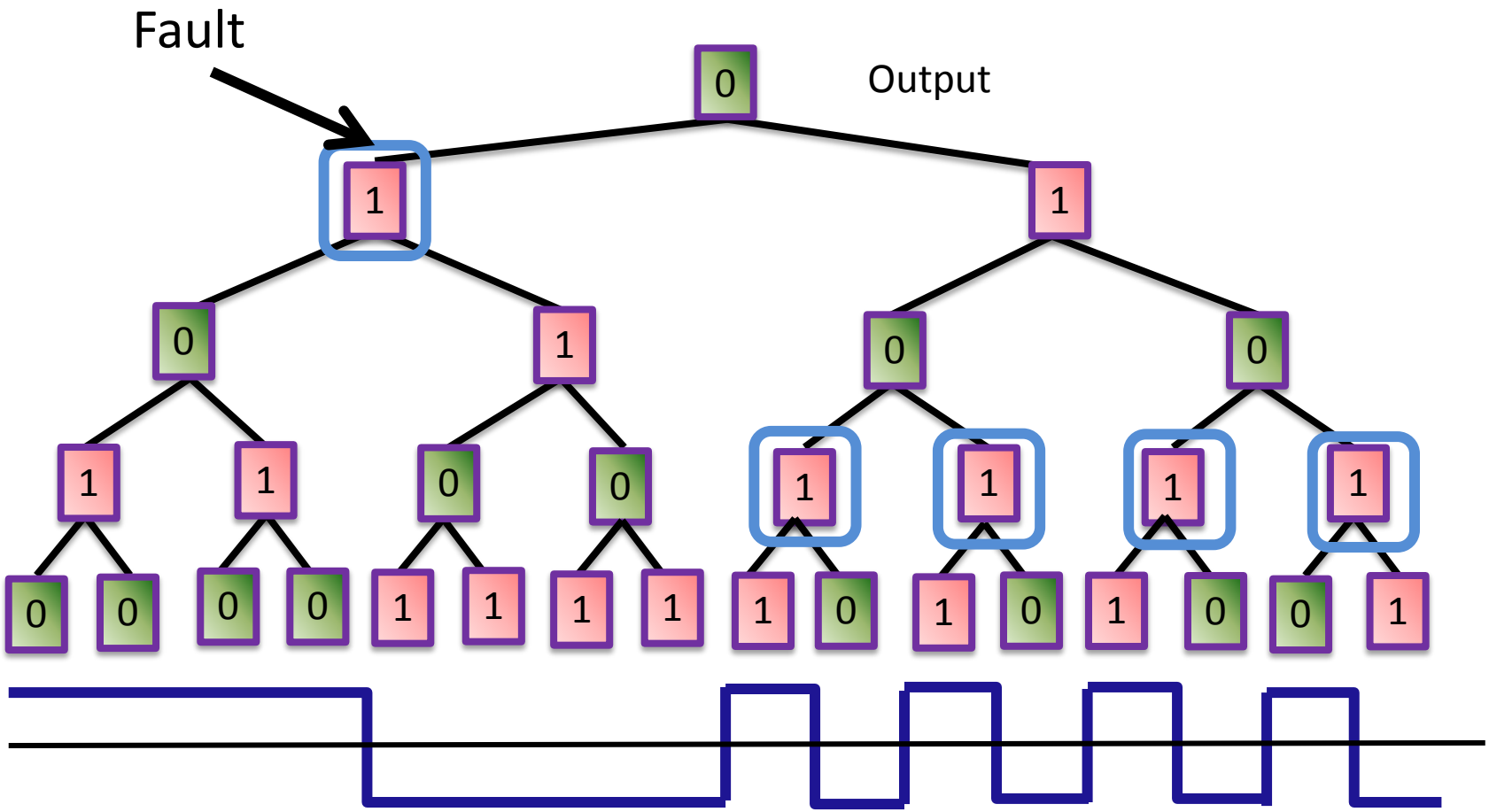
- For all c-Fault Direct Trees, constant query algorithms must exist.

Span Programs



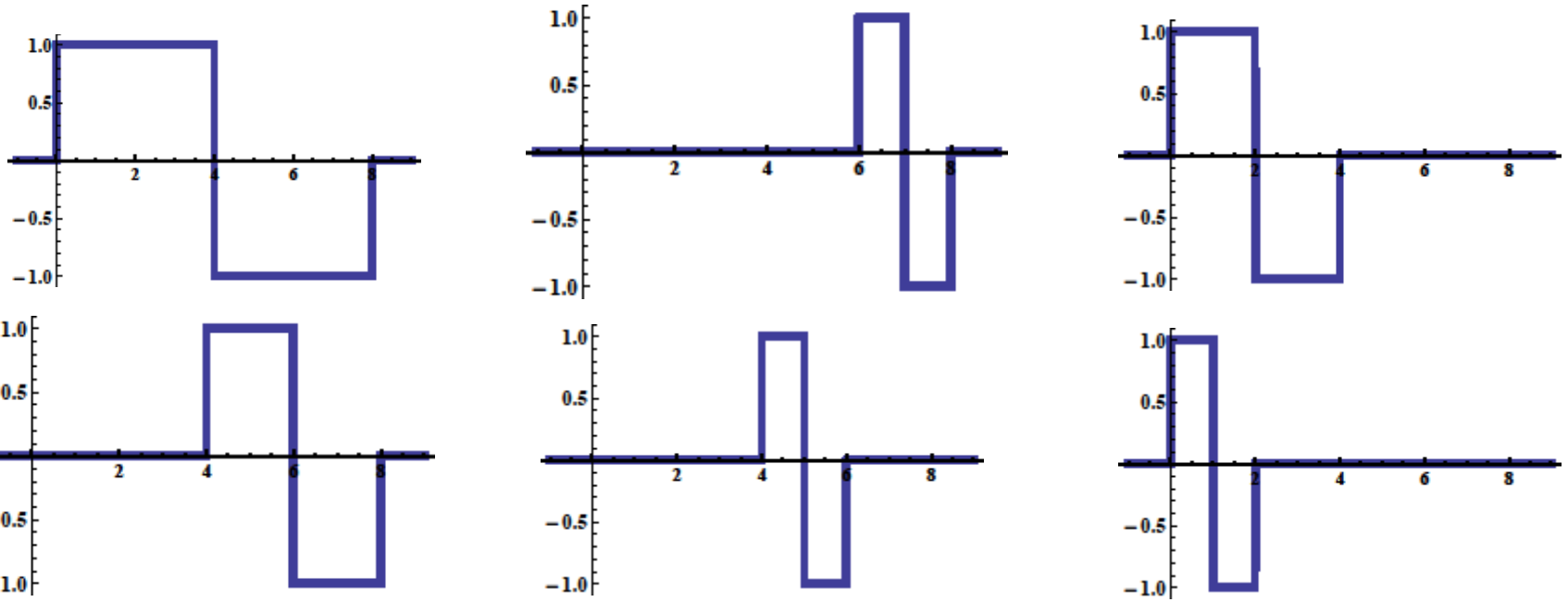
$f(\vec{x}_i) = 1$ iff
$\vec{t} \in SPAN\{\vec{v}_{1i}, \vec{v}_{2i}, \ldots, \vec{v}_{ni}\}$

1-Fault NAND Tree

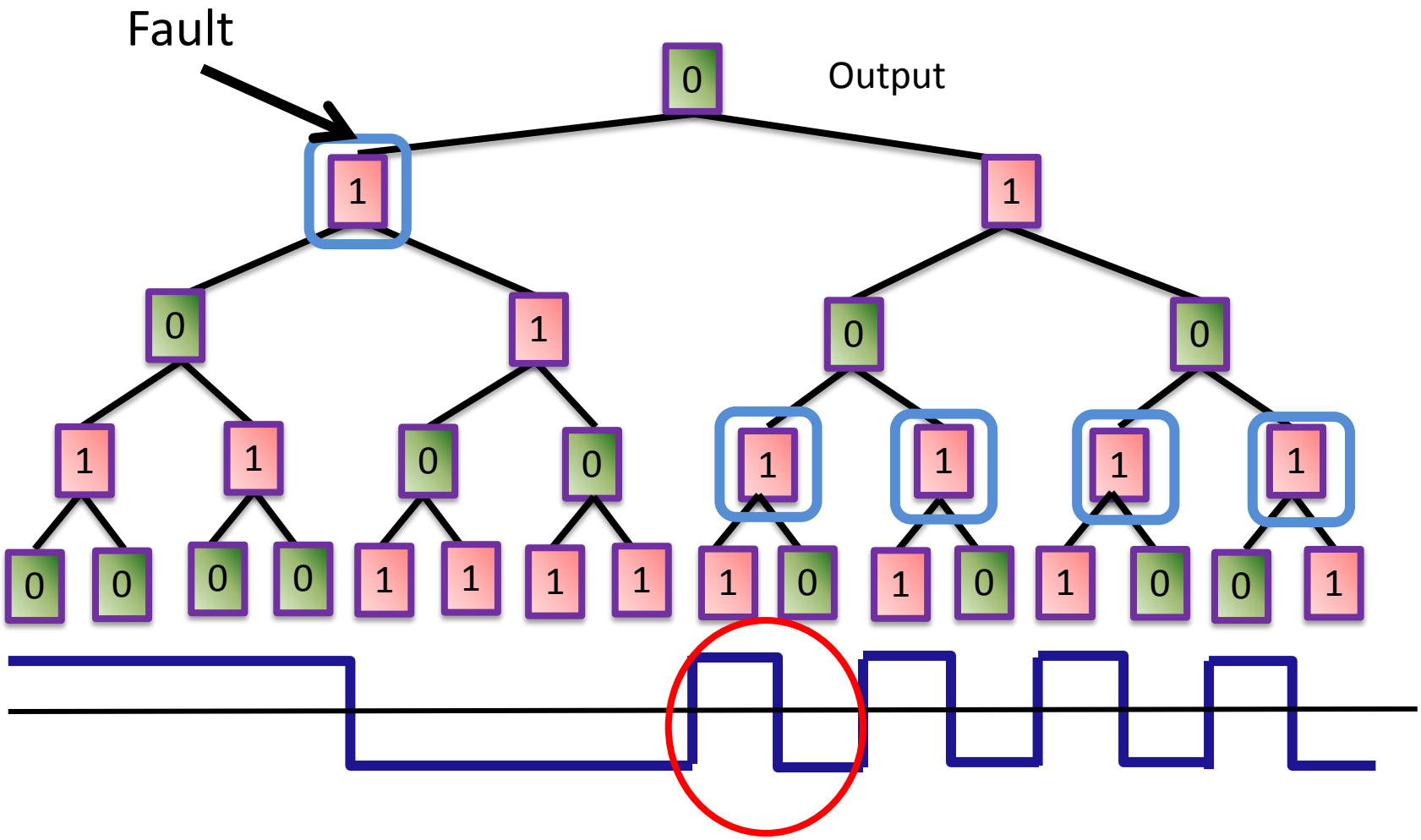


Haar Transform Algorithm

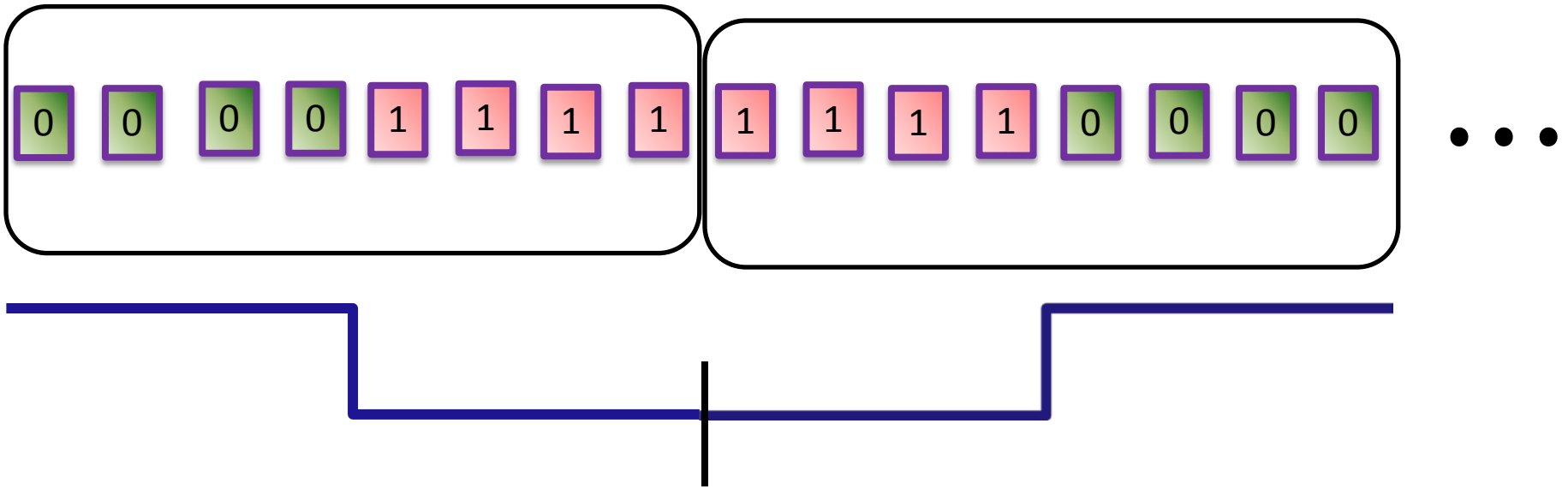
- Start in superposition: $\frac{1}{\sqrt{n}}\sum |i\rangle$.
- Apply Oracle. Phases= 
- Measure in Haar Basis



1-Fault NAND Tree



Period Finding



Summary and Open Questions

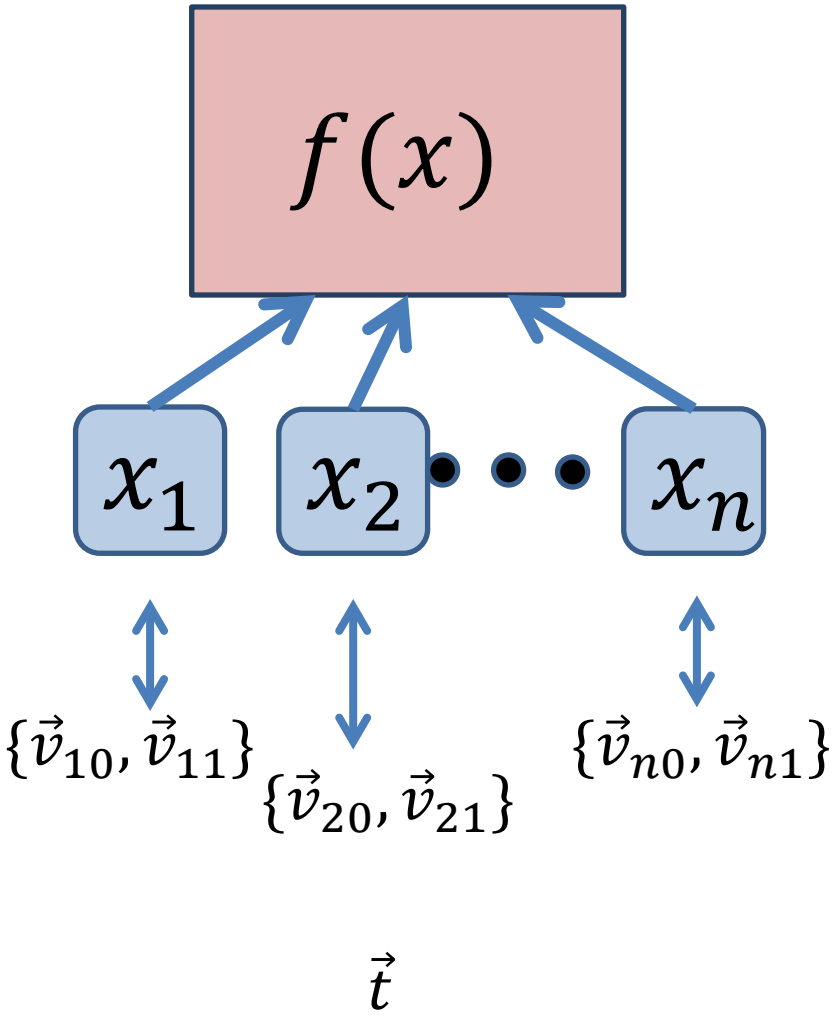
- Quantum adversary upper bound can prove the existence of quantum algorithms
  - 1-Fault NAND Tree
  - Other constant fault trees

- Are there other problems where the adversary upper bound will be useful?
- Do the matching algorithms have other applications?
- Can we take advantage of the structure of quantum algorithms to prove other similar results

Open Questions: Unique Result?

- Classically is it possible to prove the existence of an algorithm without creating it?
  - Probabilistic/Combinatorial algorithms can prove that queries exist that will give an optimal algorithm, but would need to do a brute-force search to find them [Grebinski and Kucherov, ‘97]

Span Programs



$f(\vec{x}_i) = 1$ iff
$\vec{t} \in SPAN\{\vec{v}_{1i}, \vec{v}_{2i}, \ldots, \vec{v}_{ni}\}$

NAND:

$$\vec{v}_{10} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \vec{v}_{20} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \vec{t} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

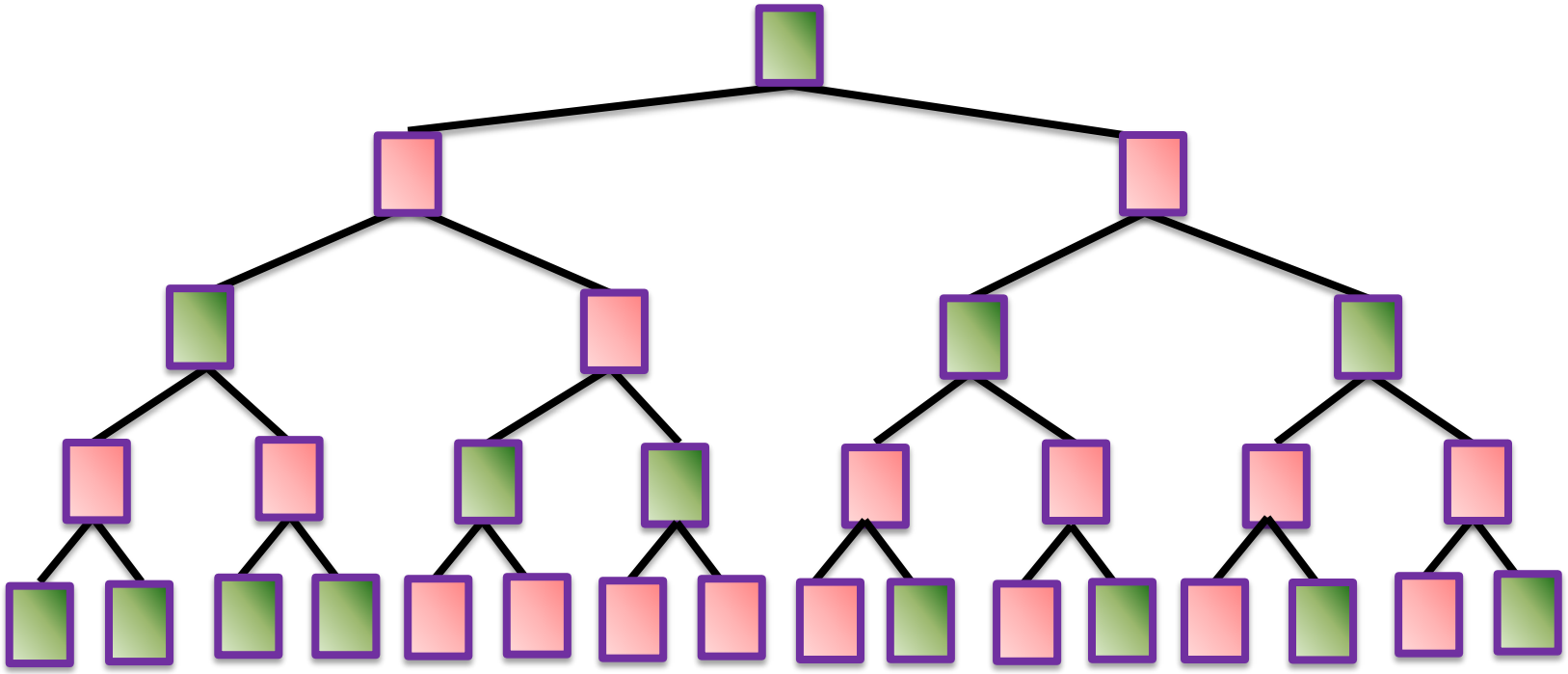
Summary and Open Questions

- Quantum adversary upper bound can prove the existence of quantum algorithms
  - 1-Fault NAND Tree
  - Other constant fault trees

- Are there other problems where this technique will be useful?
- Do the matching algorithms have other applications?
- Other Adversary SDP applications?

Smaller is not always easier

1-Fault NAND Tree



Quantum Adversary Upper Bound

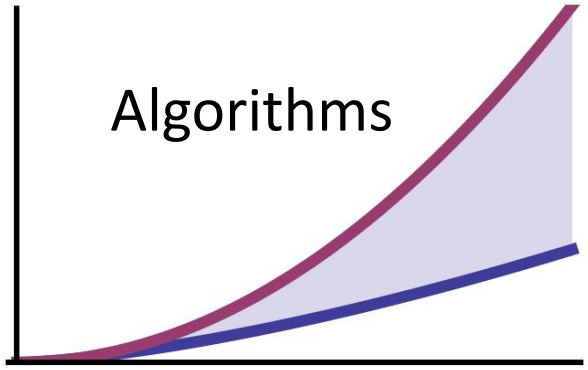
Let f be a Boolean function.

Let $Q(f^n)$, (the quantum query complexity of f^n), be $O(K)$.

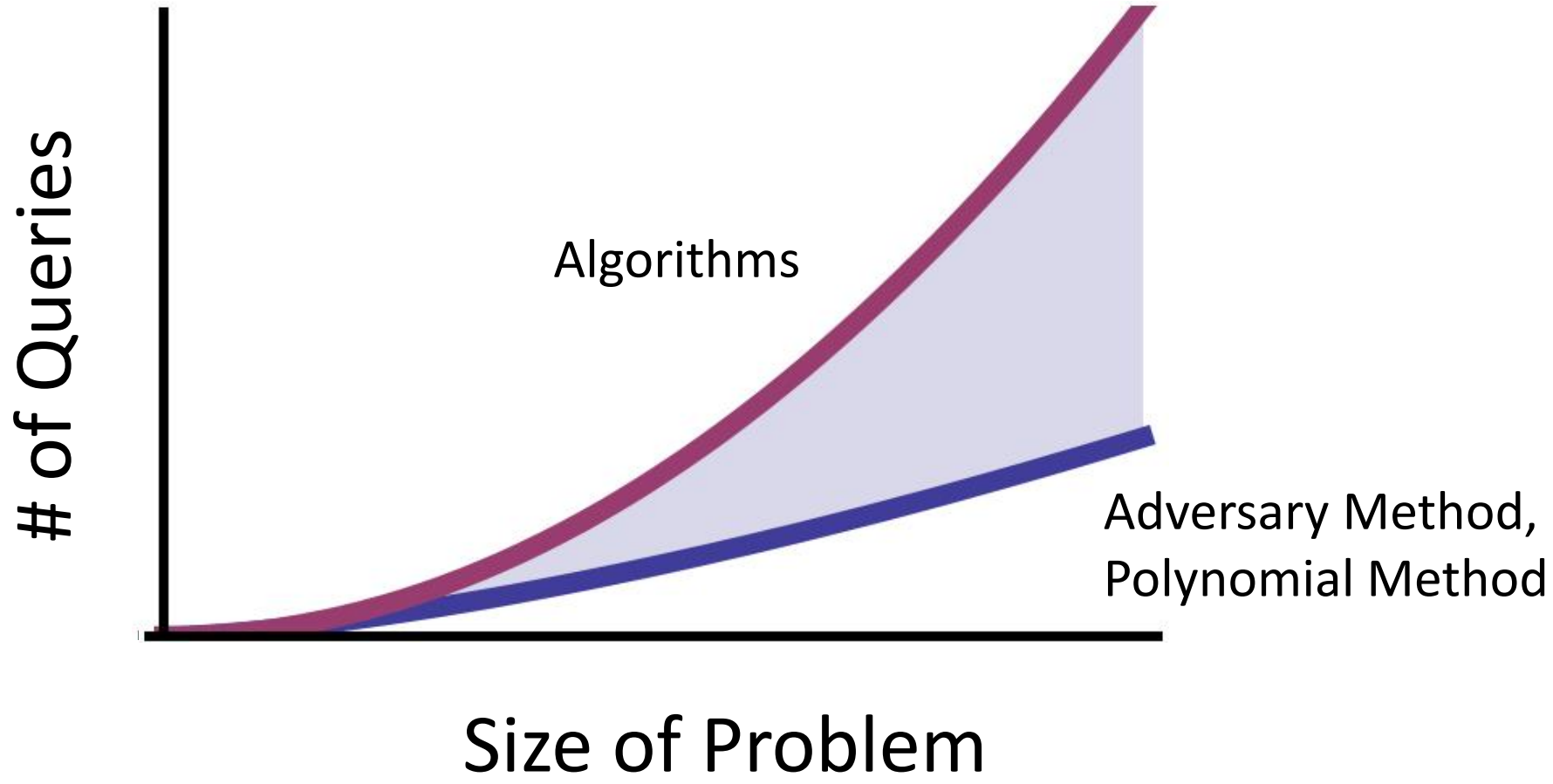
Then the quantum query complexity of f is $O(K^{1/n})$

Surprising:

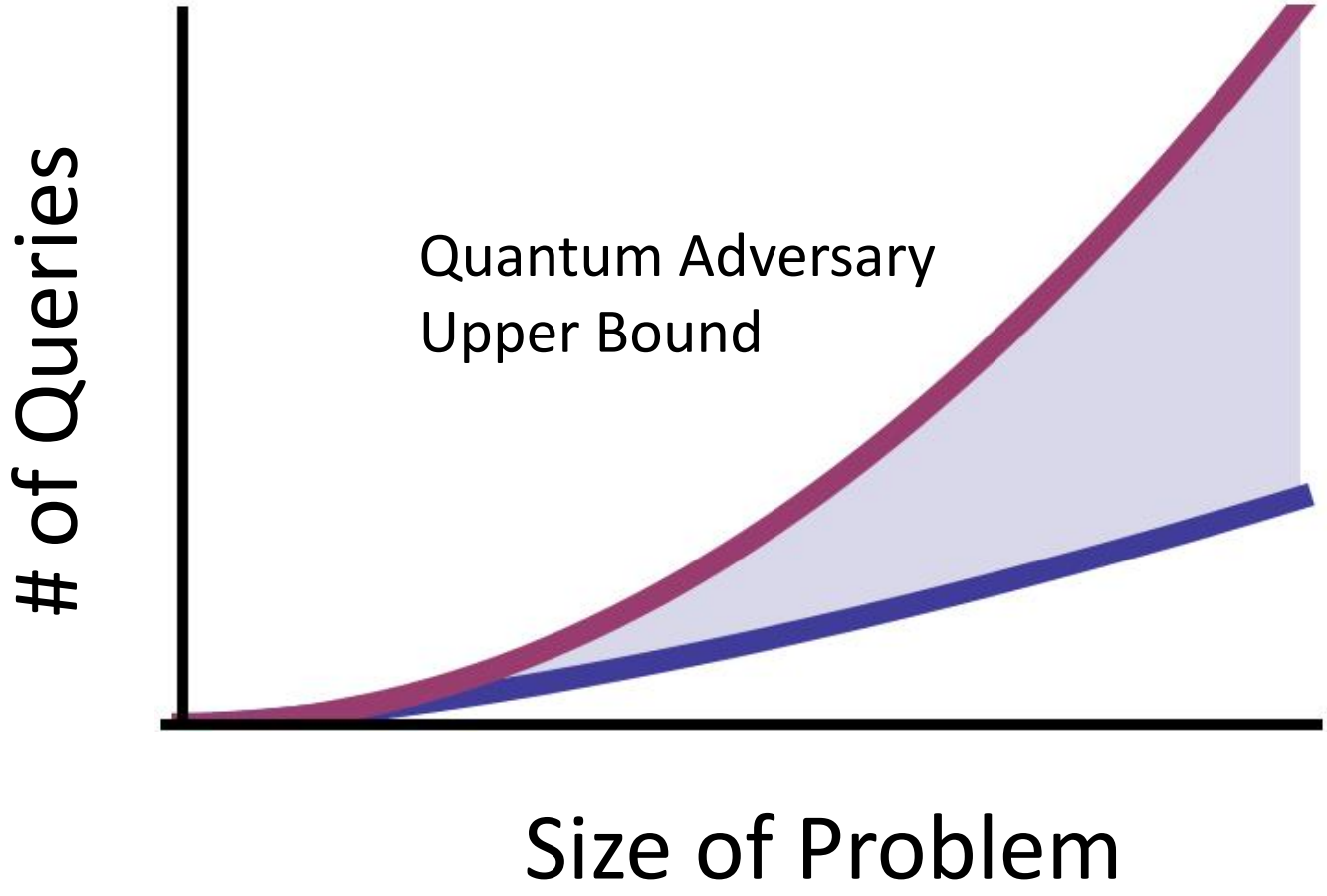
- Does not give algorithm for f
- This is a useful theorem!

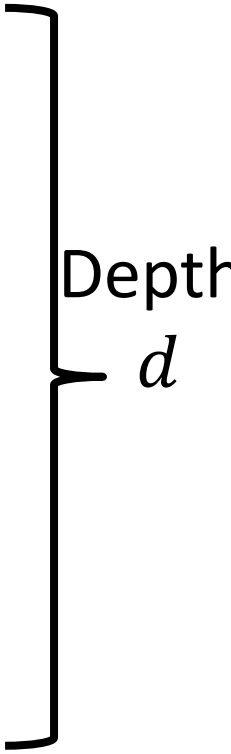


Goal: Understand Power of Quantum Computers



New Tool





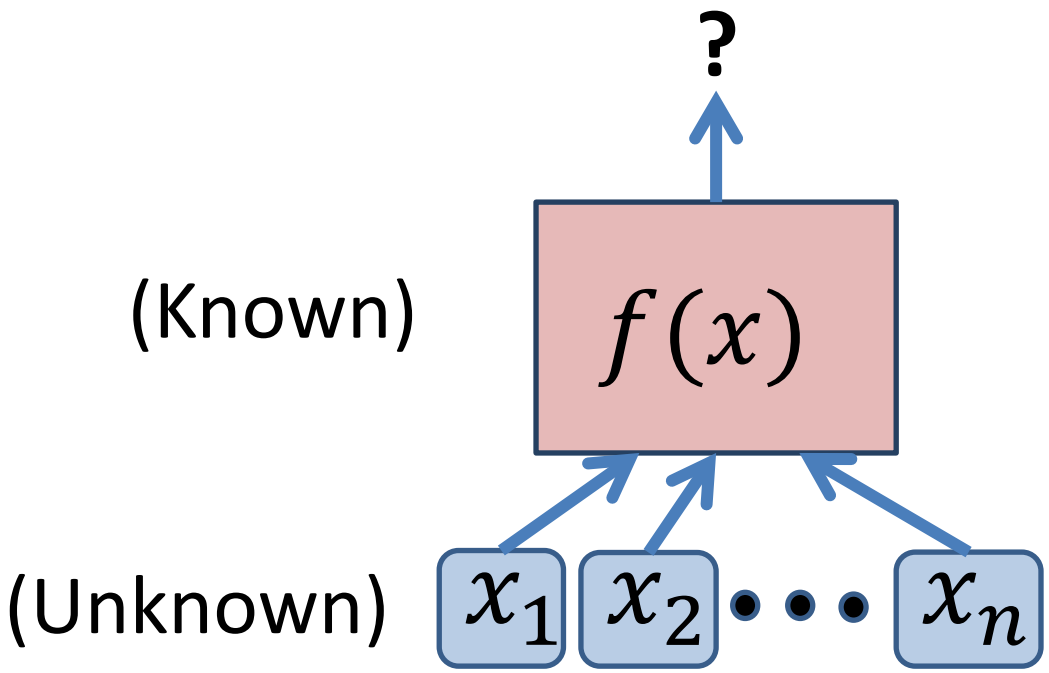
Quantum query complexity $= O(2^{0.5d})$

[Farhi et al '08]

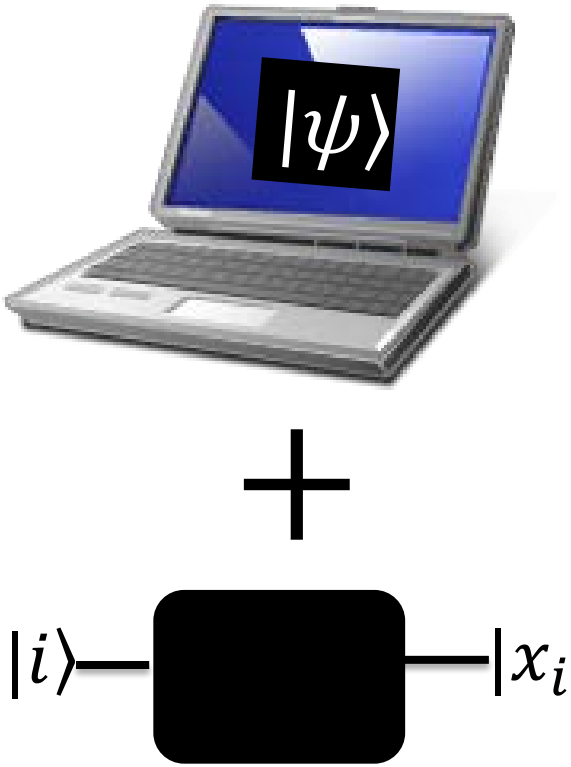
Randomized Classical Query Complexity$= \Omega(2^{0.753d})$

[Saks and Widgerson '86]

Goal: Understand Power of Quantum Computers



Tools



$Q(f)$ = Quantum Query Complexity = # of queries to black box needed to evaluate f w/ high probability