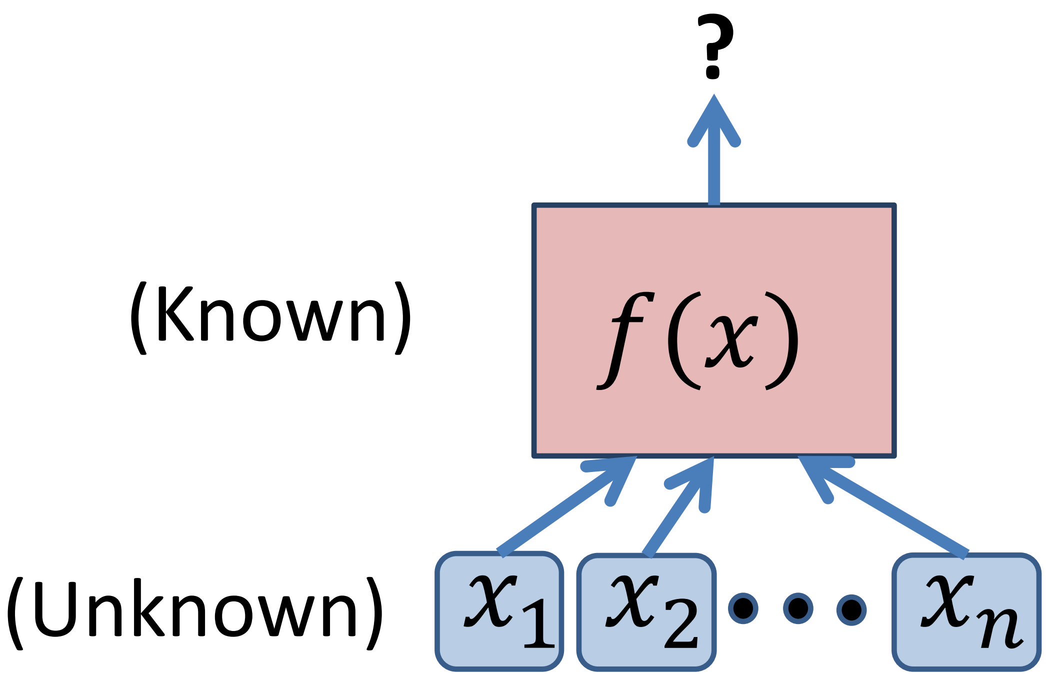
Quantum Adversary (Upper) Bound

Shelby Kimmel

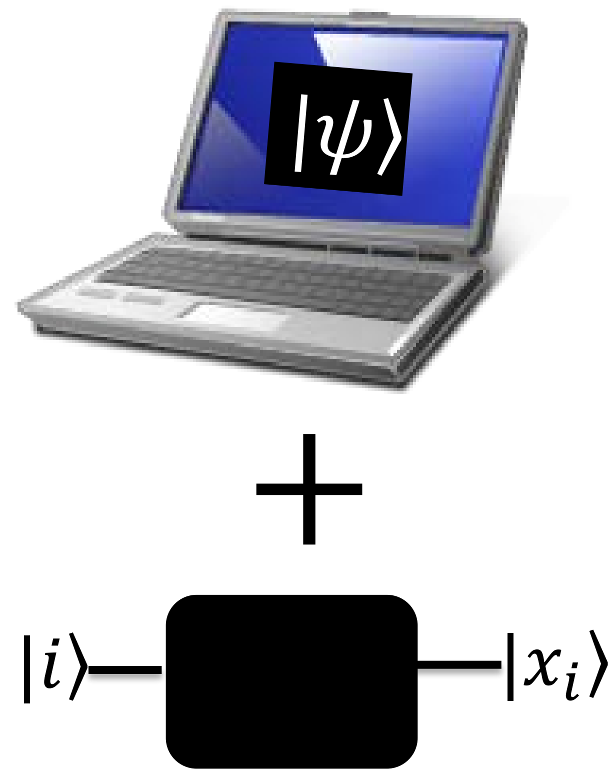
Massachusetts Institute of Technology

ICALP 2012

Goal: Understand Power of Quantum Computers

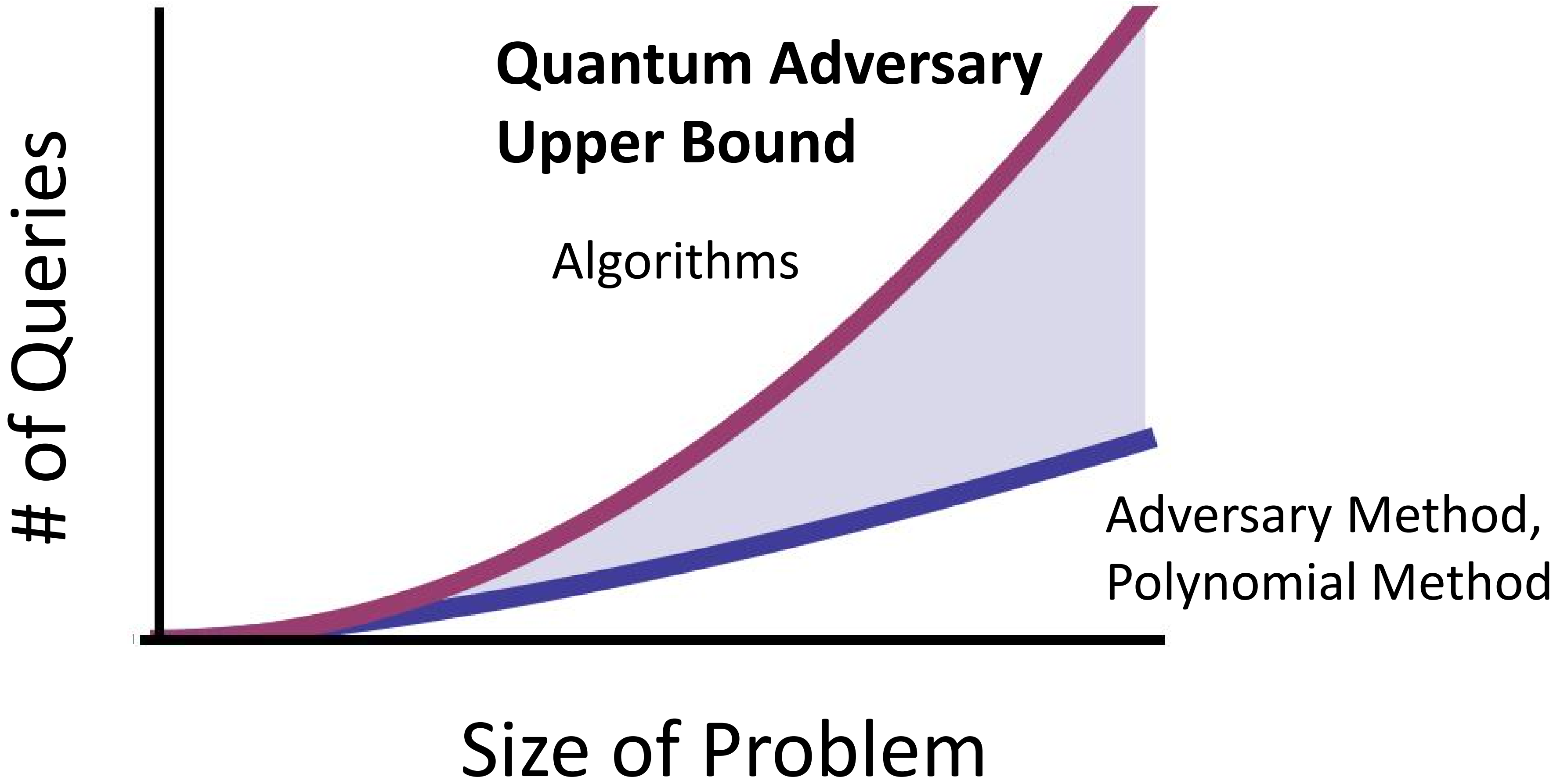


Tools



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

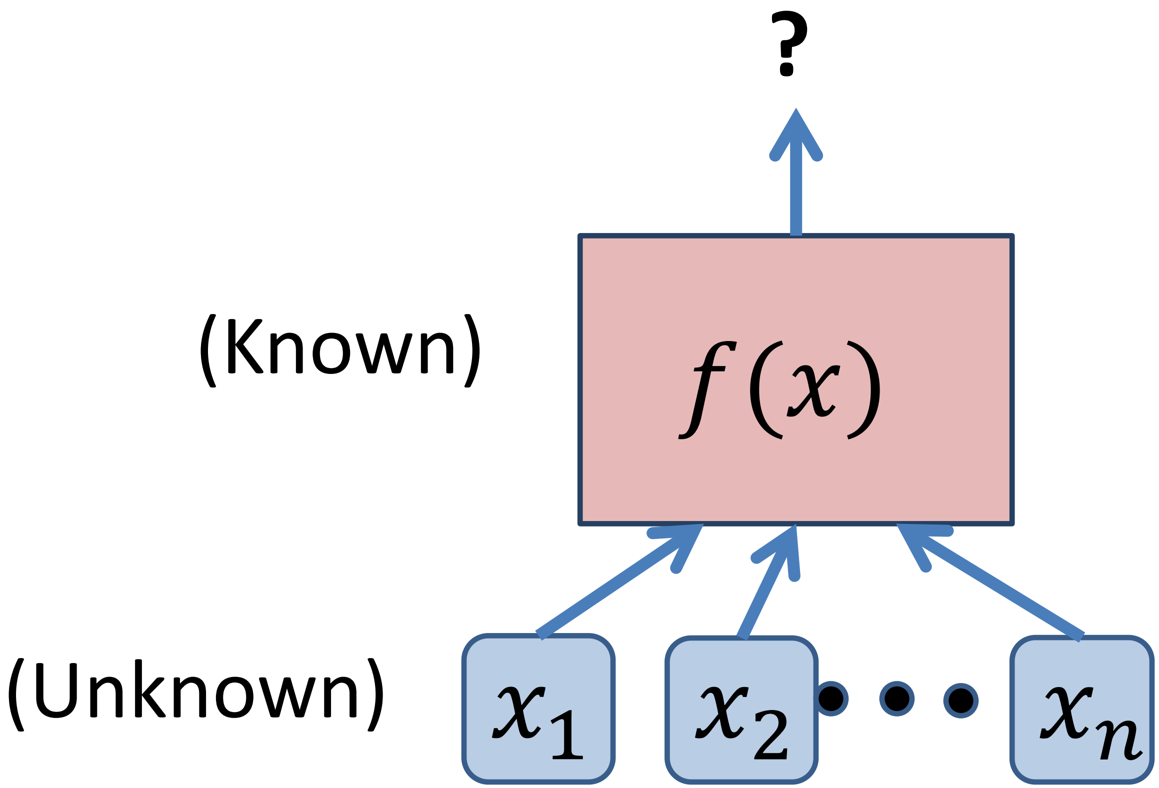
Goal: Understand Power of Quantum Computers



Outline

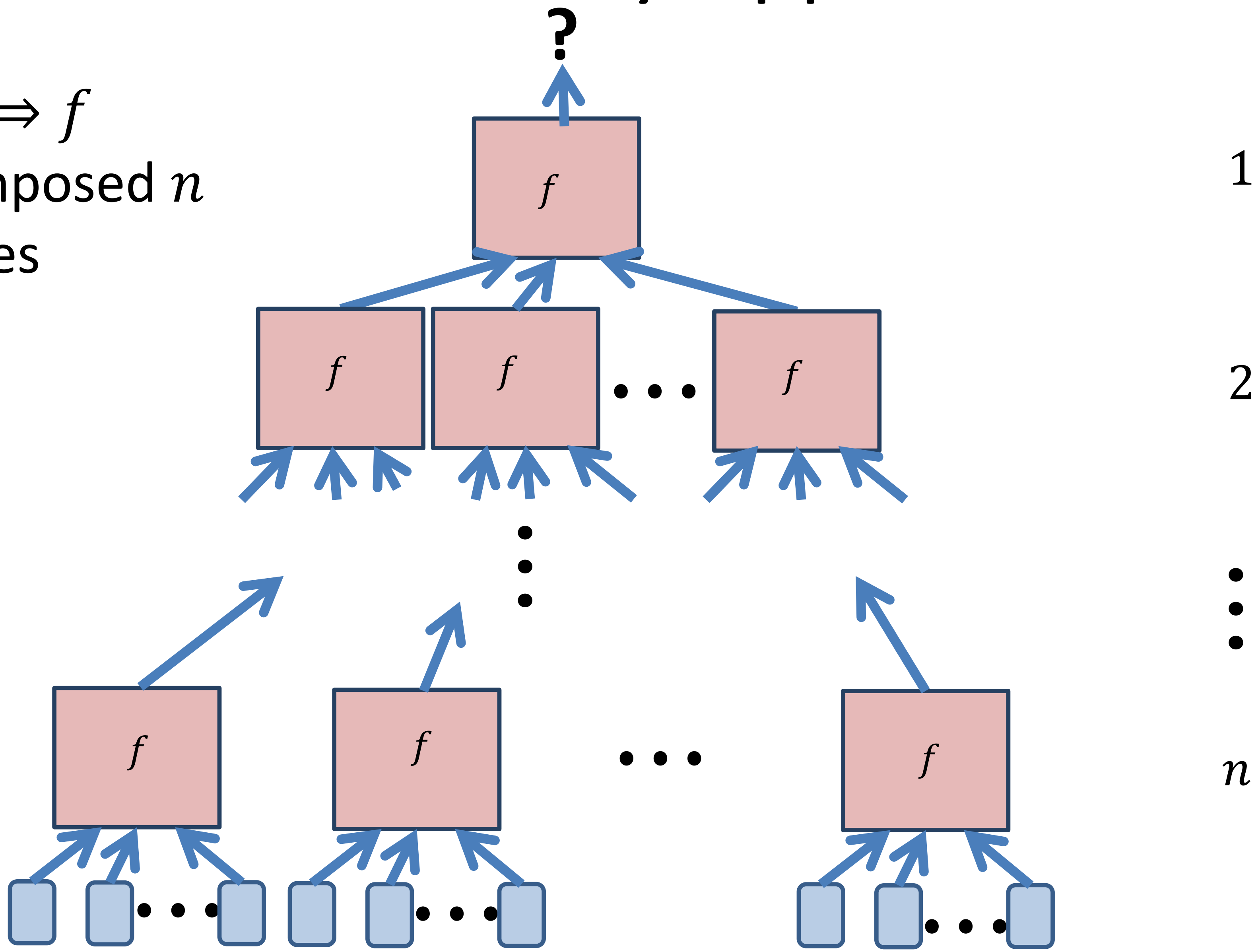
- Quantum Adversary Upper Bound
- Example: “1-Fault NAND Tree”
- Summary and Open Problems

Quantum Adversary Upper Bound



Quantum Adversary Upper Bound

$f^n \Rightarrow f$ composed n times



Quantum Adversary Upper Bound

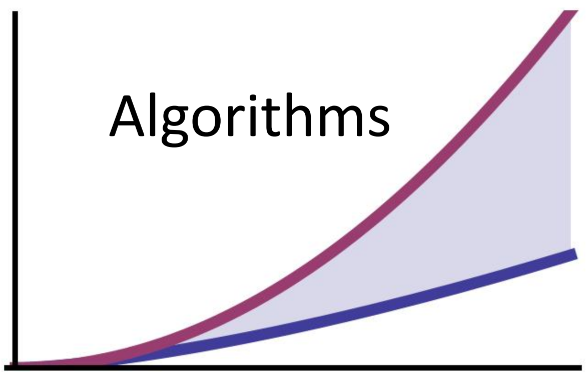
Let f be a Boolean function.

Create an algorithm for f^n, so learn $Q(f^n) = O(K)$.

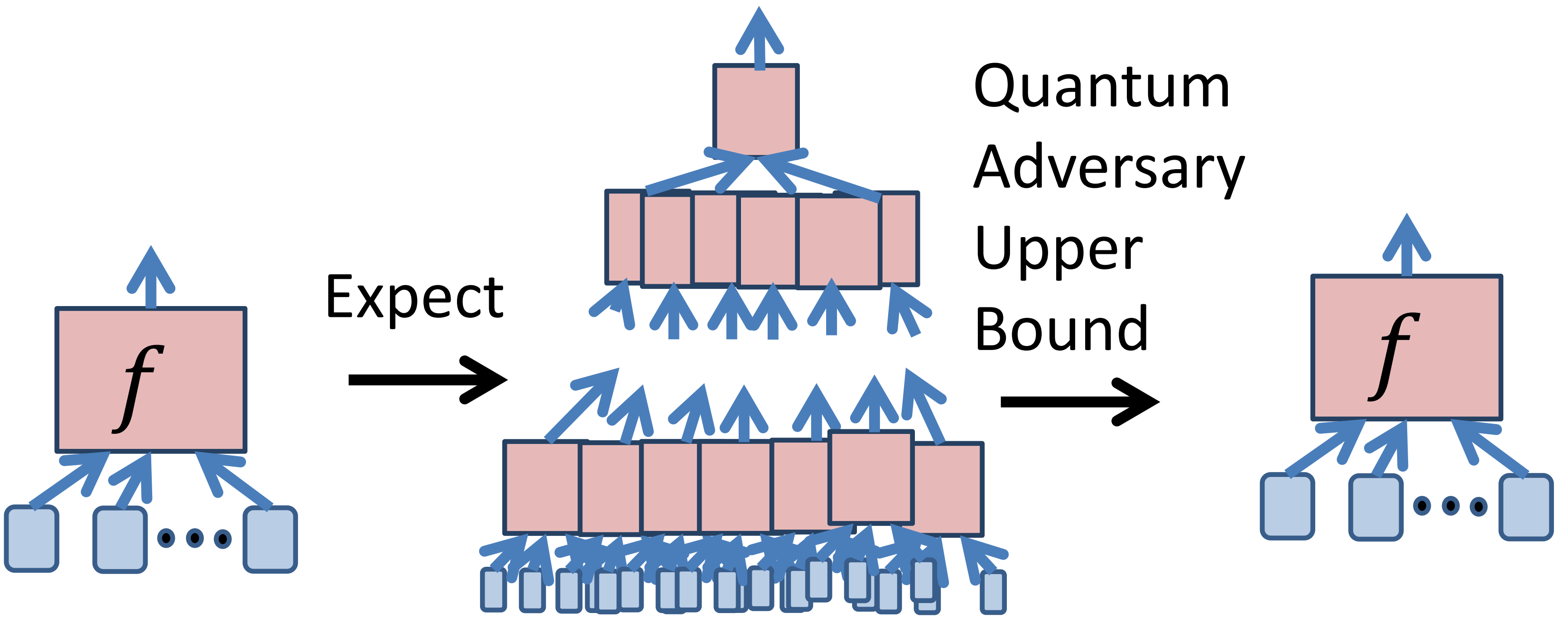
Then the quantum query complexity of f is $O(K^{1/n})$

Surprising:

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



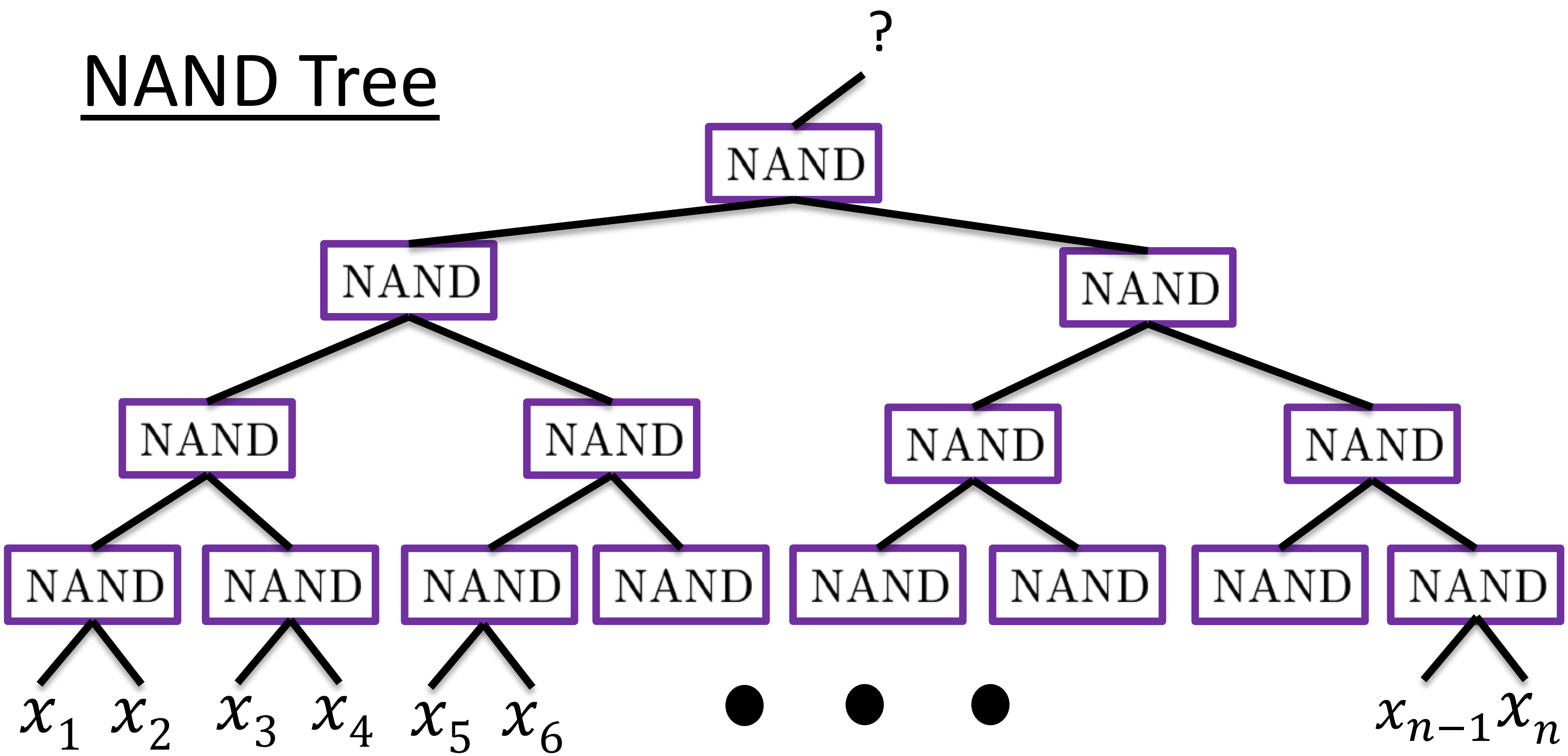
Quantum Adversary Upper Bound



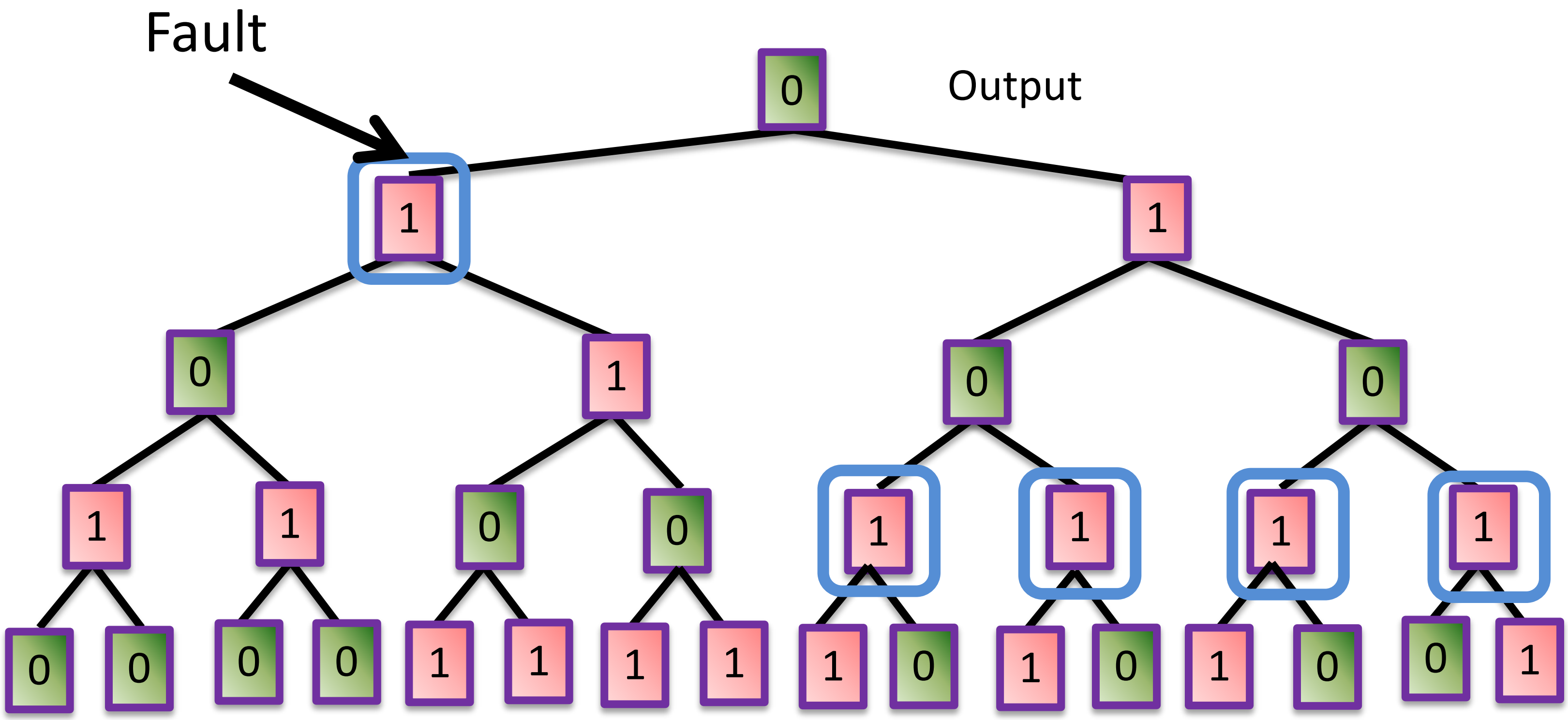
$Q(f) = O(T)$ $\qquad$ $Q(f^n) = O(T^n)$ $\qquad$ $Q(f) = O(T)$

1-Fault NAND Tree

NAND Tree



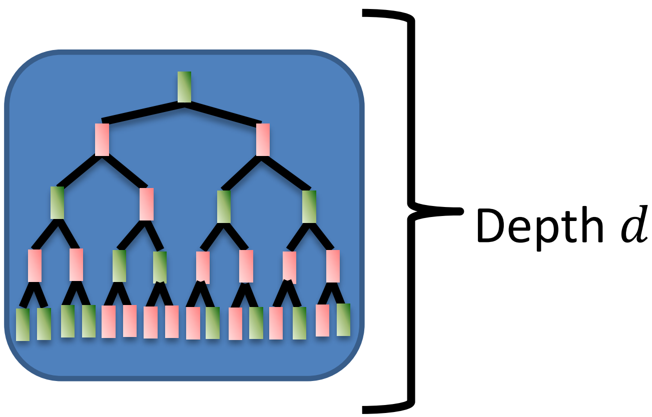
1-Fault NAND Tree



1-Fault NAND Tree

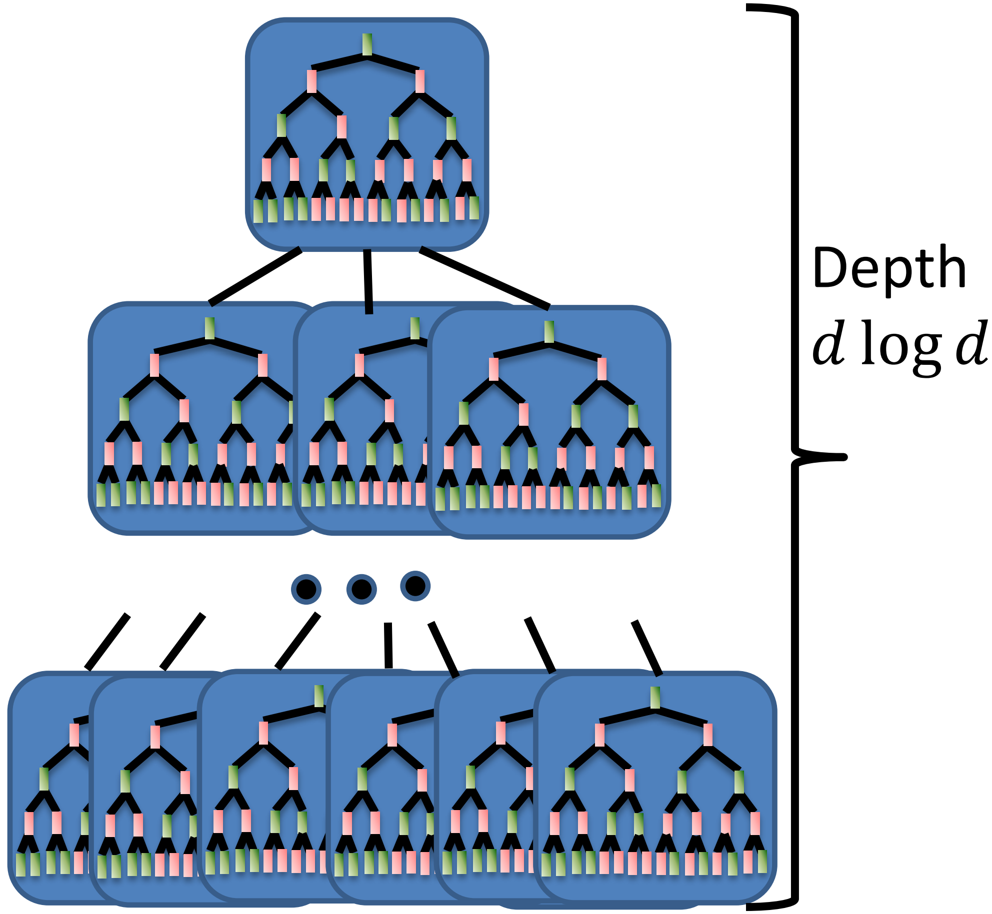
[Zhan, Hassidim, K. 2012]

1-Fault NAND Tree



$$Q(f) = O(d^2)$$

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



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

Quantum Adversary Upper Bound

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

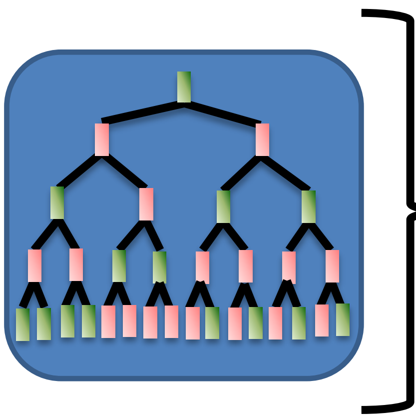
Quantum query complexity of $[1-\text{Fault NAND Tree}]^{\log d}$ is $O(d^3)$

Then the quantum query complexity of $[1-\text{Fault NAND Tree}]$ is $O\left(d^{3/\log d}\right) = \mathrm{O}\left(2^{3\log d/\log d}\right) = O(1)$

1-Fault NAND Tree

[Zhan, Hassidim, K. 2012]

1-Fault NAND Tree

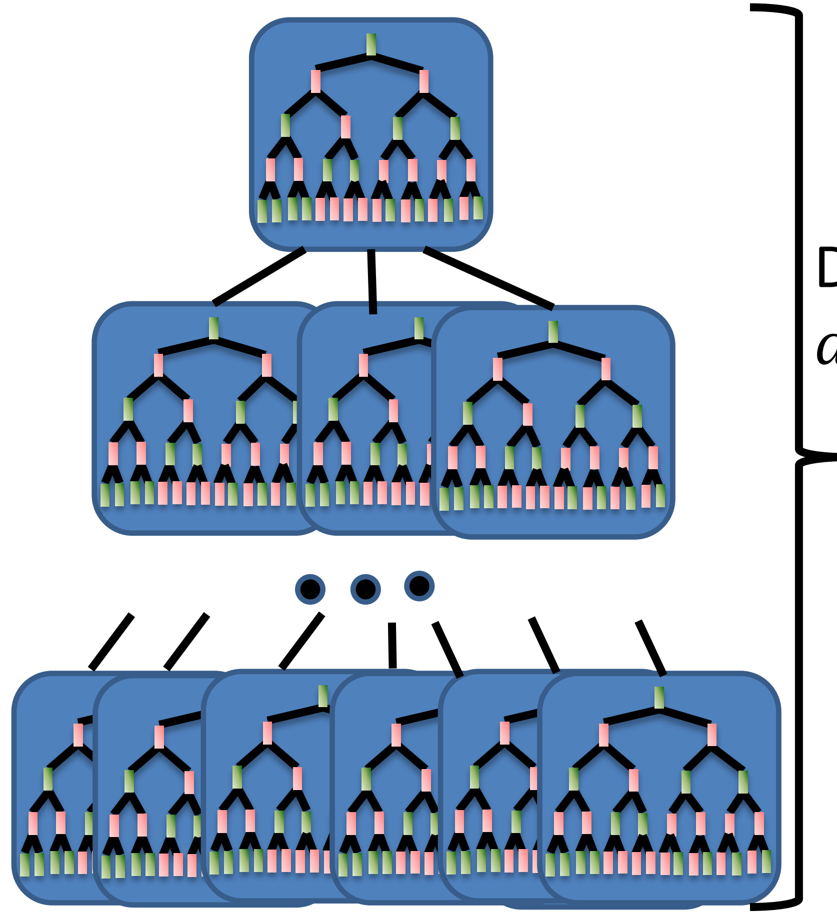


Depth d

$$Q(f) = O(d^2)$$

1-Fault NAND Tree

Composed $\log d$ times

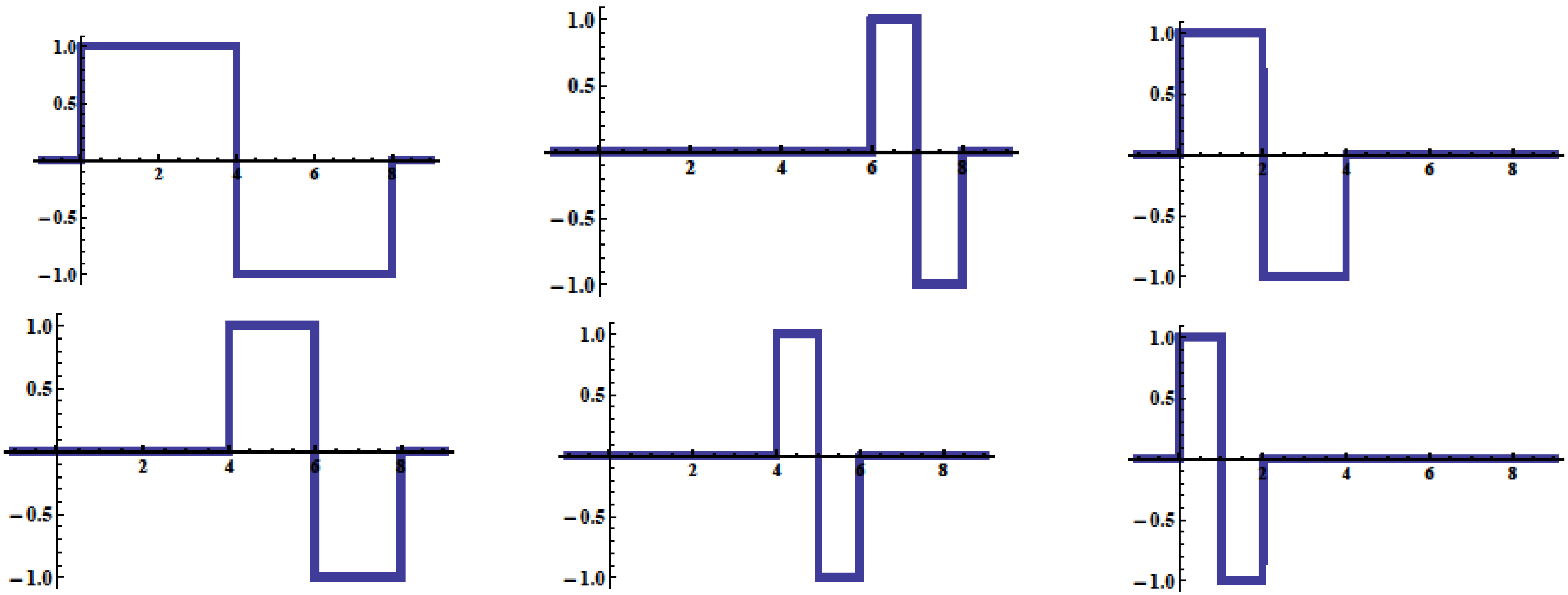


Depth $d \log d$

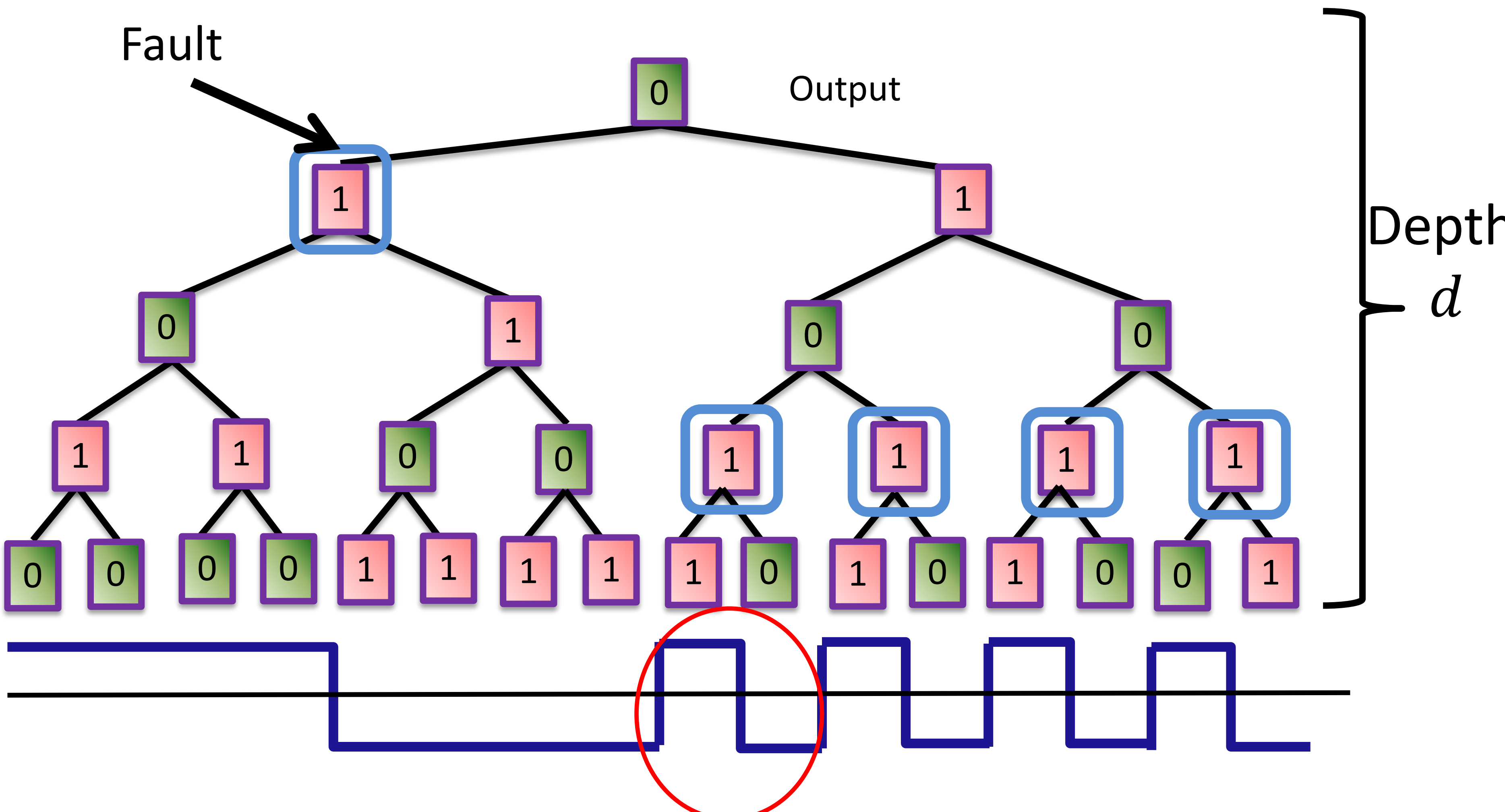
$$Q(f) = O(d^3)$$

Algorithm?

- Found a matching algorithm using span programs
- Found a related algorithm that uses quantum Haar Transform



1-Fault NAND Tree



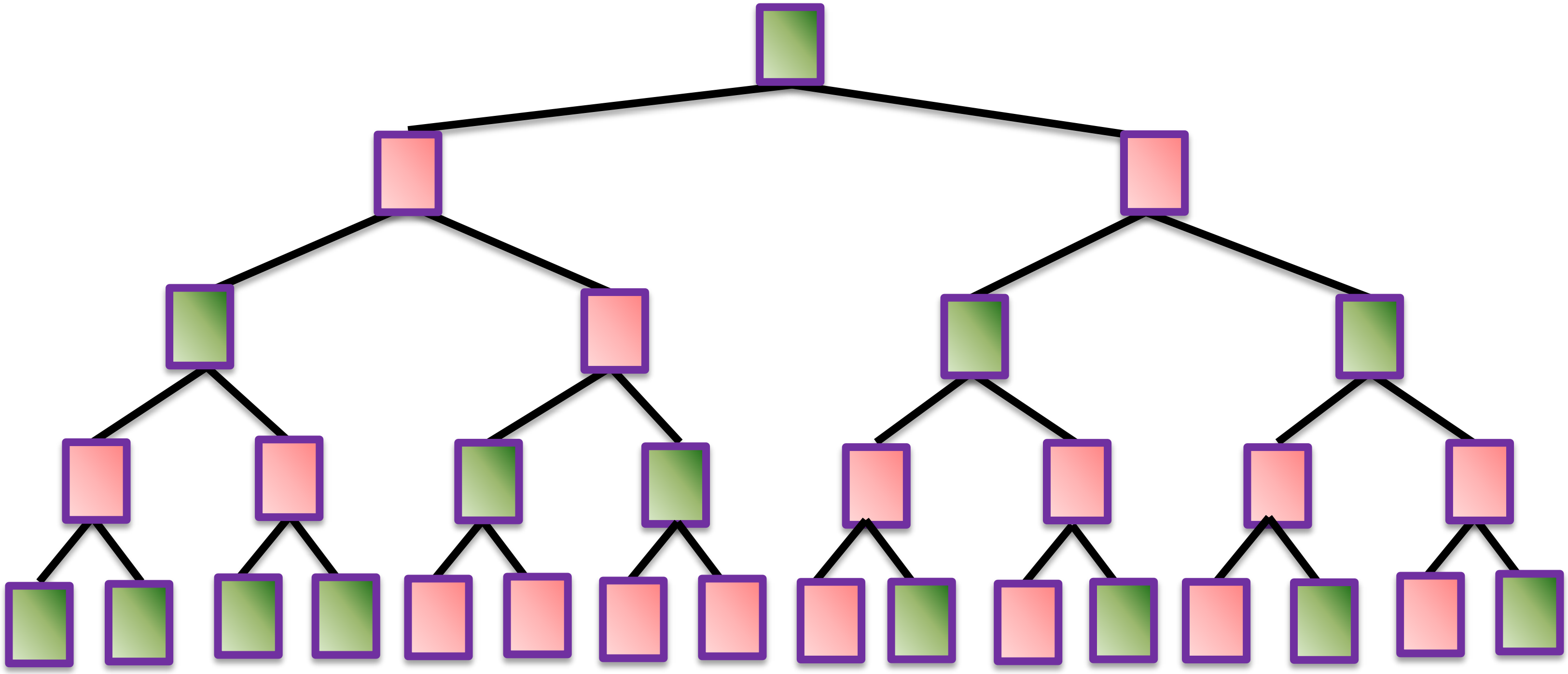
Summary and Open Questions

- 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

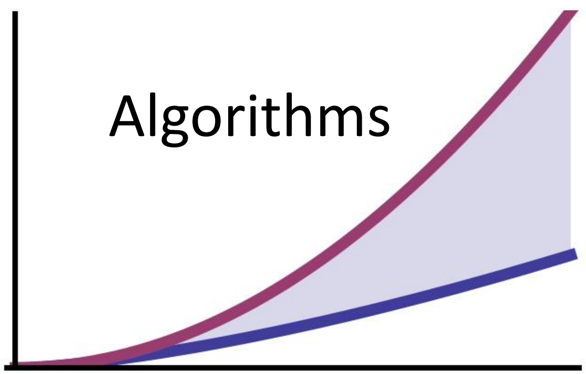
Let f be a Boolean function.

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

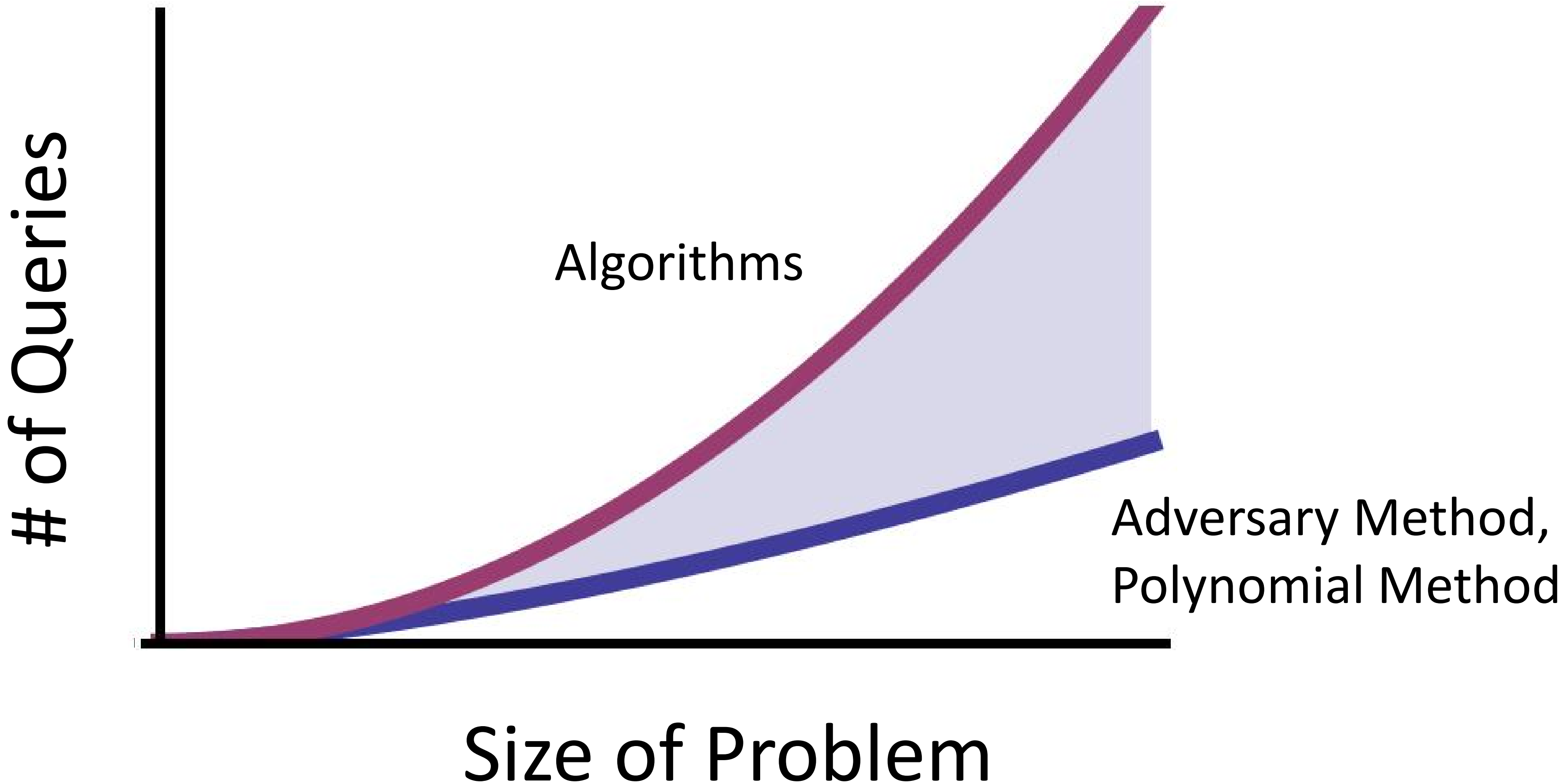
Then the quantum query complexity of f is $O(K^{1/n})$

Surprising:

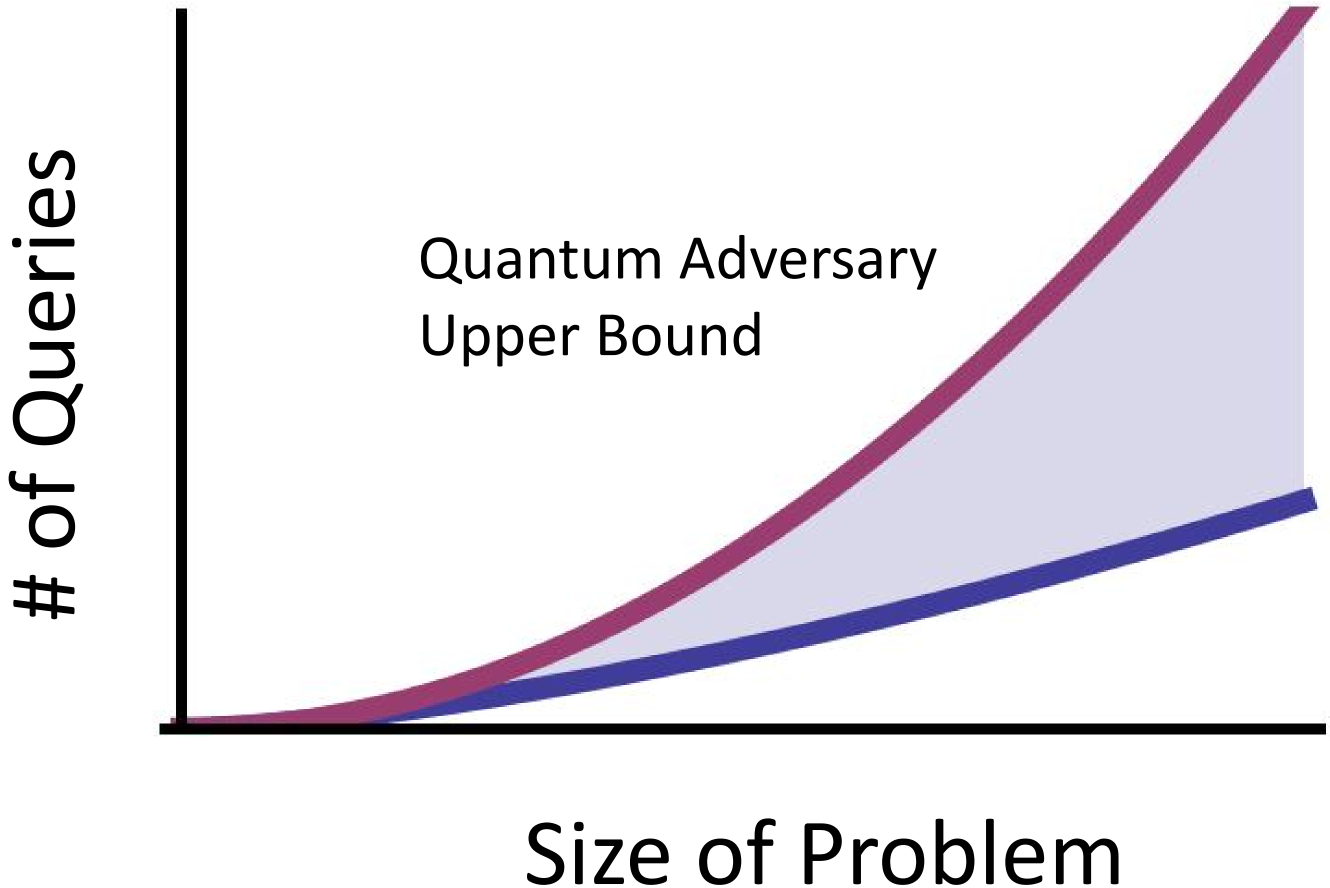
- Does not give algorithm for f
- This is a useful theorem!



Goal: Understand Power of Quantum Computers



New Tool



Depth d

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

[Farhi et al '08]

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

[Saks and Widgerson '86]