Title: Quantum Adversary (Upper) Bound

Date: Nov 20, 2013 04:00 PM

URL: http://pirsa.org/13110090

Abstract: I discuss a technique - the quantum adversary upper bound - that uses the structure of quantum algorithms to gain insight into the quantum query complexity of Boolean functions. Using this bound, I show that there must exist an algorithm for a certain Boolean formula that uses a constant number of queries. Since the method is non-constructive, it does not give information about the form of the algorithm. After describing the technique and applying it to a class of functions, I will outline quantum algorithms that match the non-constructive bound.

Quantum Adversary (Upper) Bound

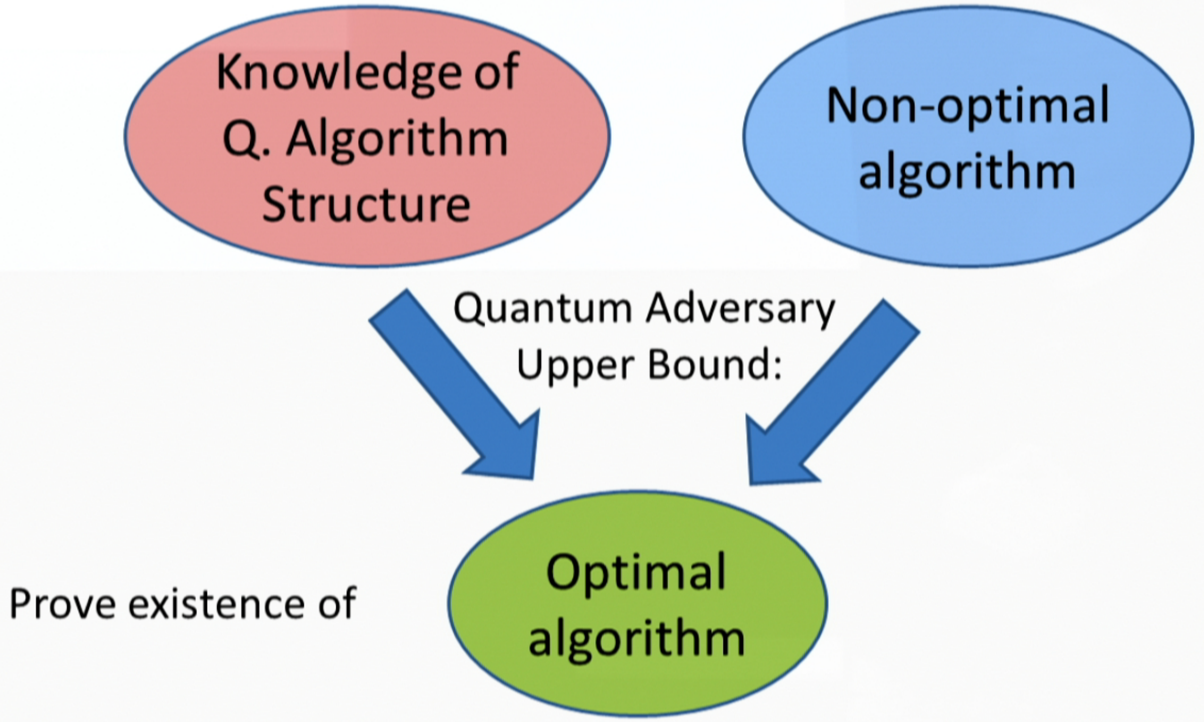
Shelby Kimmel
Center for Theoretical Physics,
Massachusetts Institute of Technology

Perimeter Institute
Nov. 20, 2013

Big Goal:

Design new quantum algorithms

Result

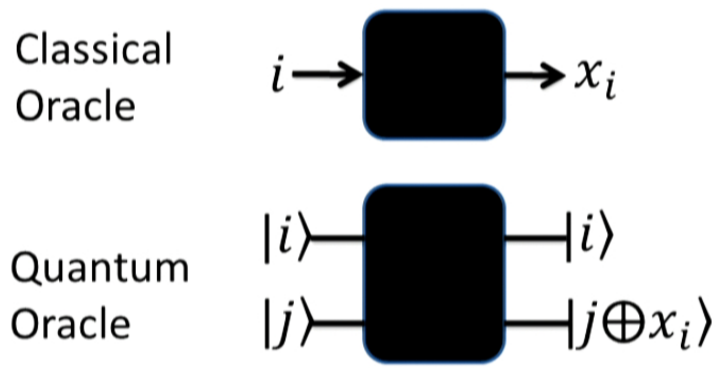

Knowledge of Q. Algorithm Structure
Non-optimal algorithm
Quantum Adversary Upper Bound:
Prove existence of
Optimal algorithm

Outline

- Oracle Model and Query Complexity
- Quantum Adversary (Upper) Bound
- Application
  - Prove existence of optimal algorithm using Quantum Adversary (Upper) Bound
  - Find explicit optimal algorithm

Oracle Model

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



$Q(f)$
(quantum bounded error query complexity)

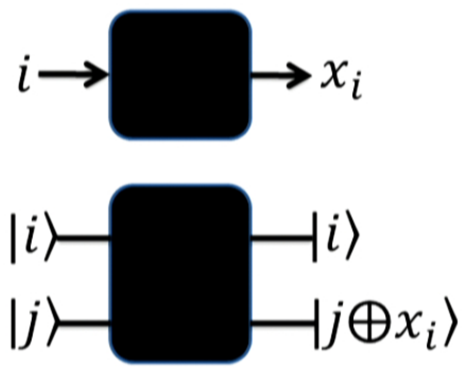
Only care about # of oracle calls (queries)

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

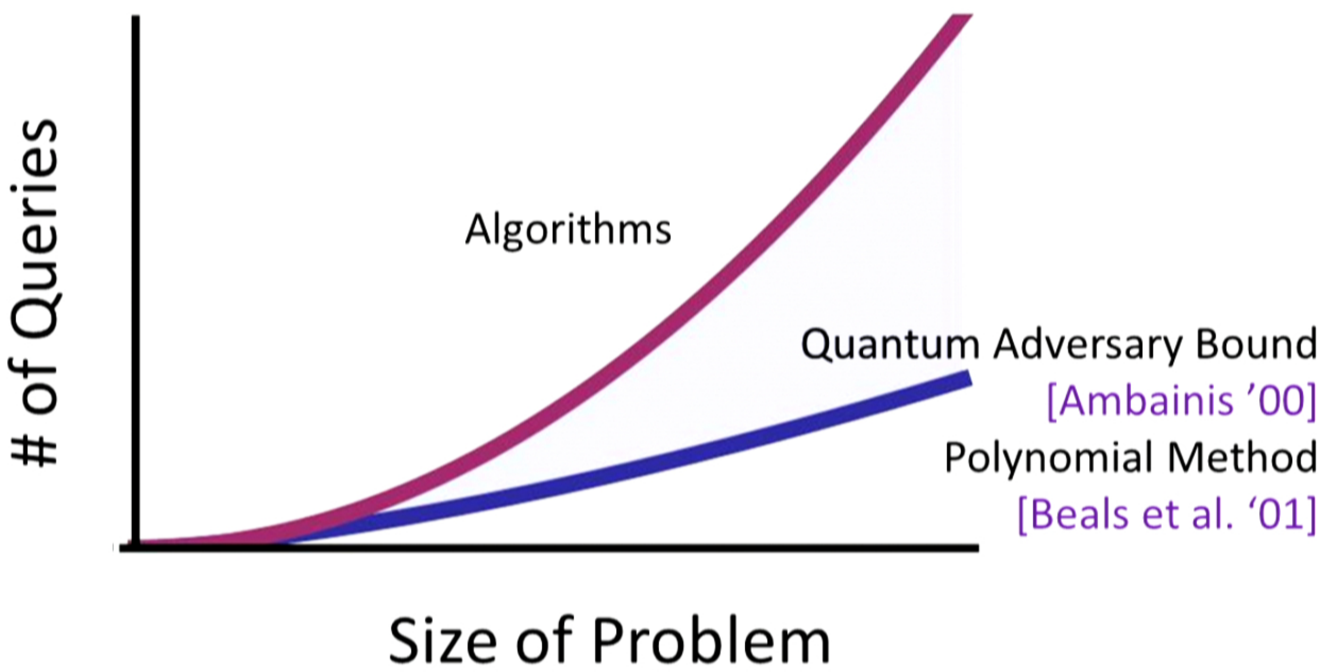
Quantum Oracle



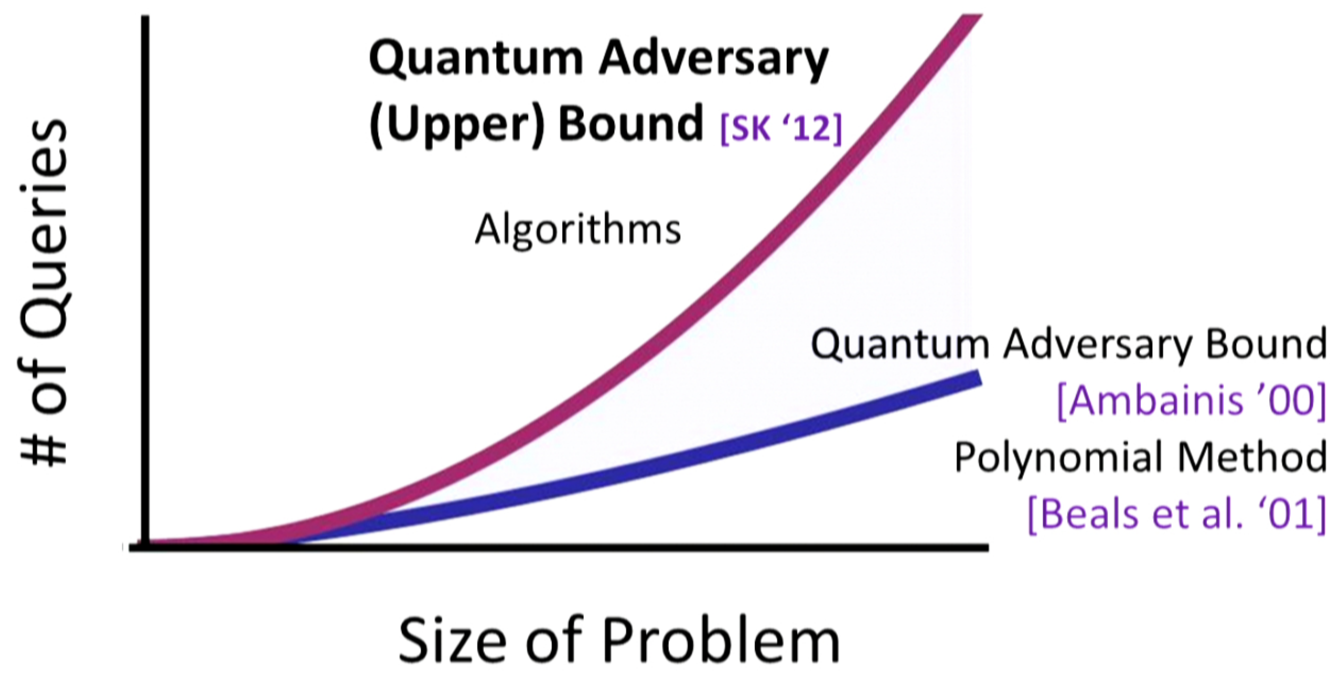
$\mathrm{Q}(f)$
(quantum bounded error query complexity)

Only care about # of oracle calls (queries)

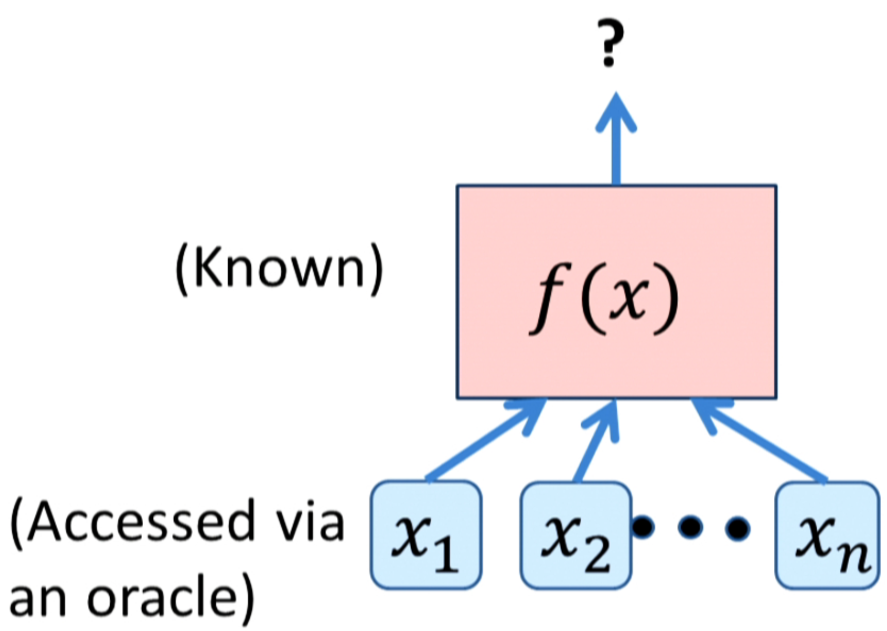
Query Complexity



Query Complexity

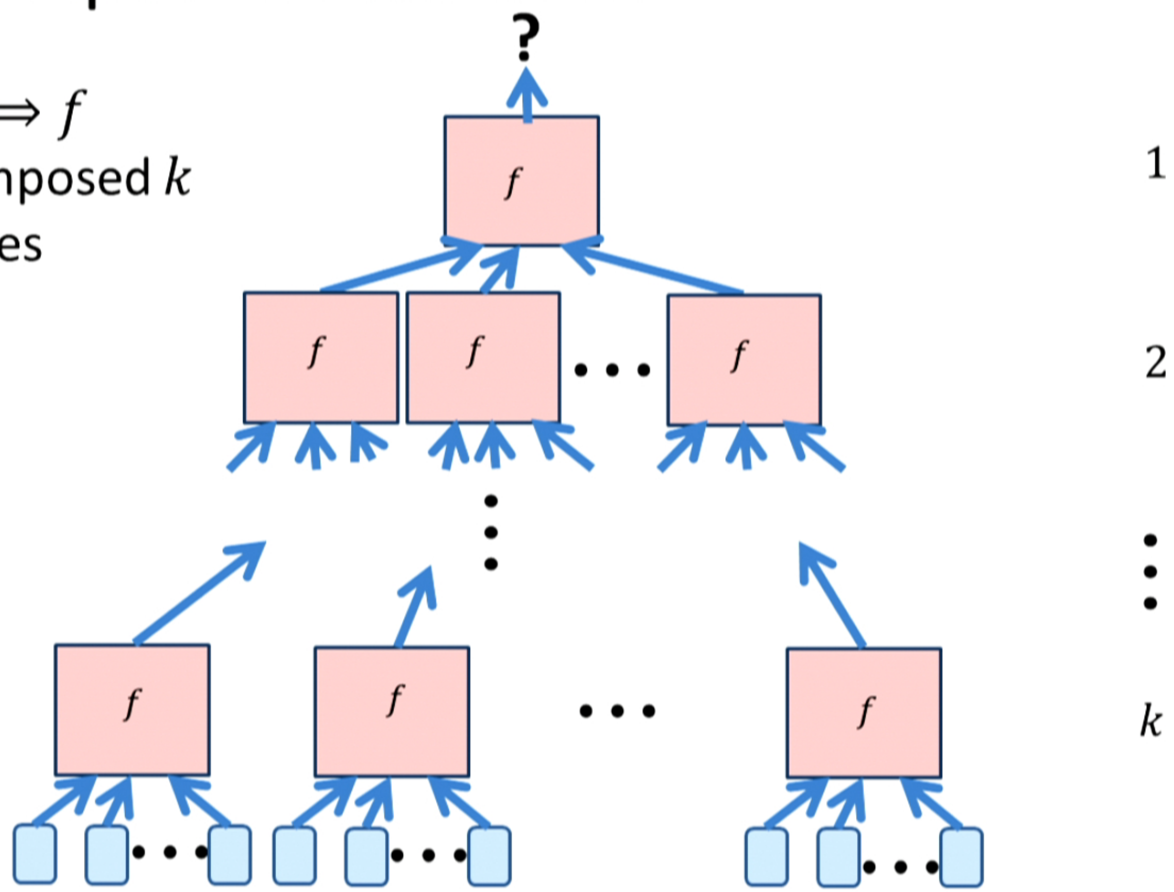


Composed Functions



Composed Functions

$f^k \Rightarrow f$ composed k times



Quantum Adversary Upper Bound

[SK '12]

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 $Q(f)$ is upper bounded by $T^{1/k}$.

($Q(f)$ = quantum query complexity of f)

Quantum Adversary Upper Bound

[SK ’12]

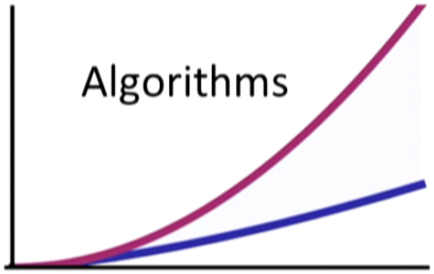
Let f be a Boolean function.

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

Then $Q(f)$ is upper bounded by $T^{1/k}$.

Surprising:

- Does not give algorithm for f



Quantum Adversary Upper Bound

[SK '12]

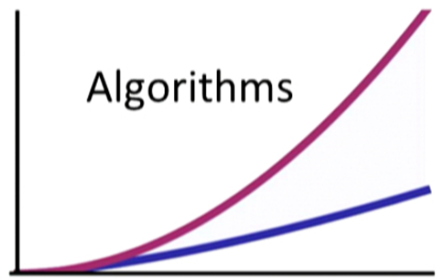
Let f be a Boolean function.

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

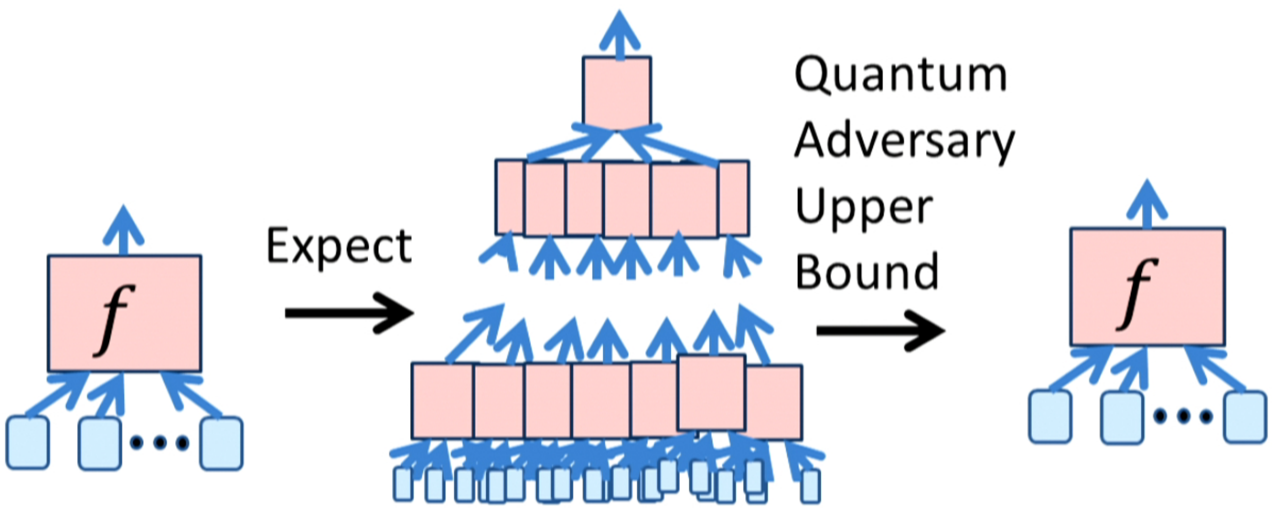
Then $Q(f)$ is upper bounded by $T^{1/k}$.

Surprising:

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



Quantum Adversary Upper Bound

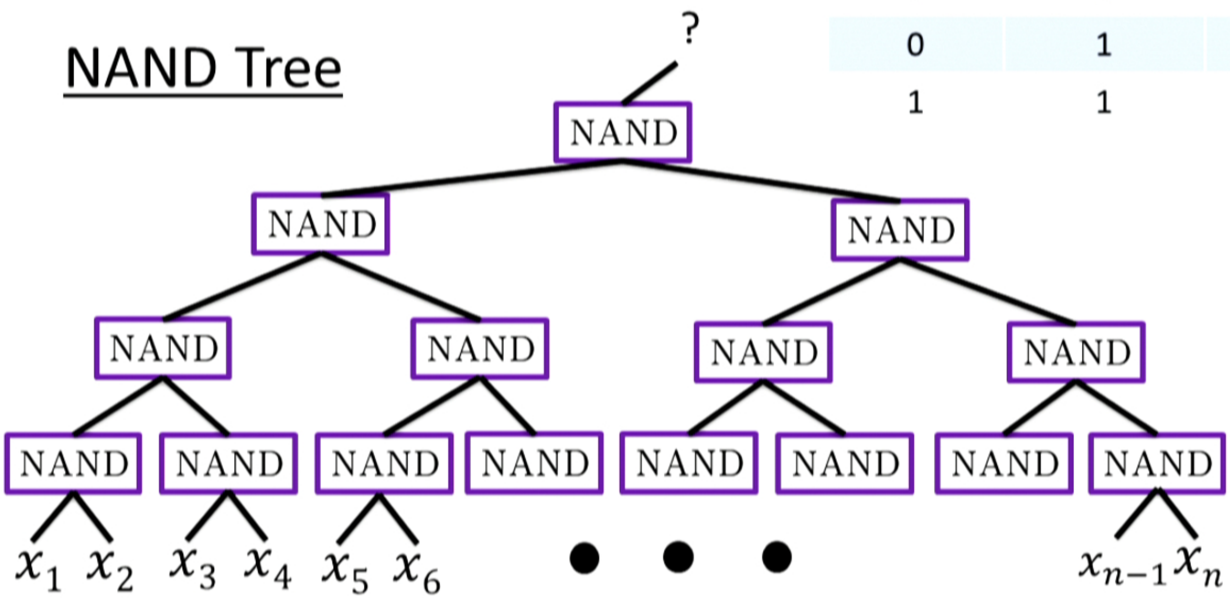


$Q(f) = O(T)$ $\quad$ $Q(f^k) = O(T^k)$ $\quad$ $Q(f) = O(T)$

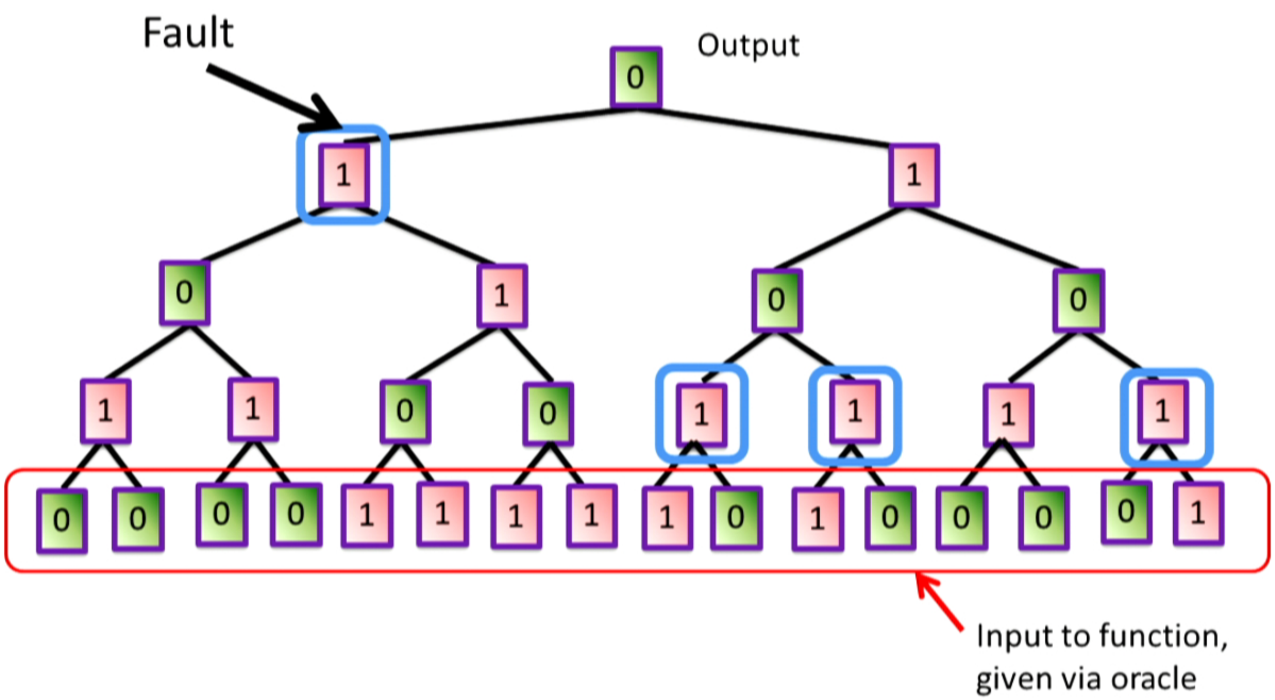
Example: 1-Fault NAND Tree

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

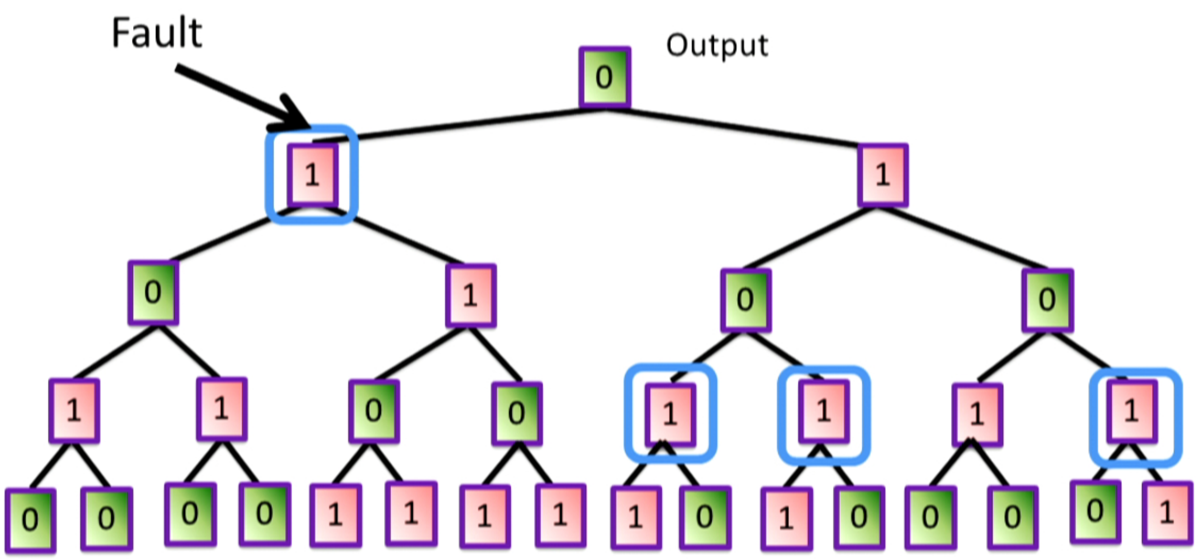
NAND Tree



Example: 1-Fault NAND Tree



Example: 1-Fault NAND Tree



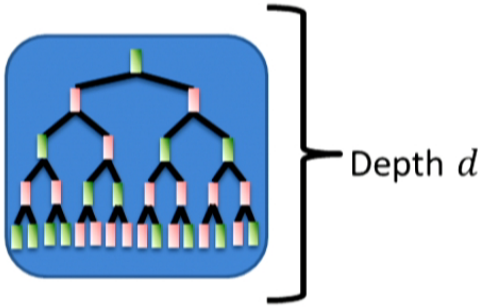
Another view point: 1-Fault NAND Tree is a game tree where the players are promised that they will only have to make one critical decision in the game.

Example: 1-Fault NAND Tree

[Zhan, Hassidim, SK `12]

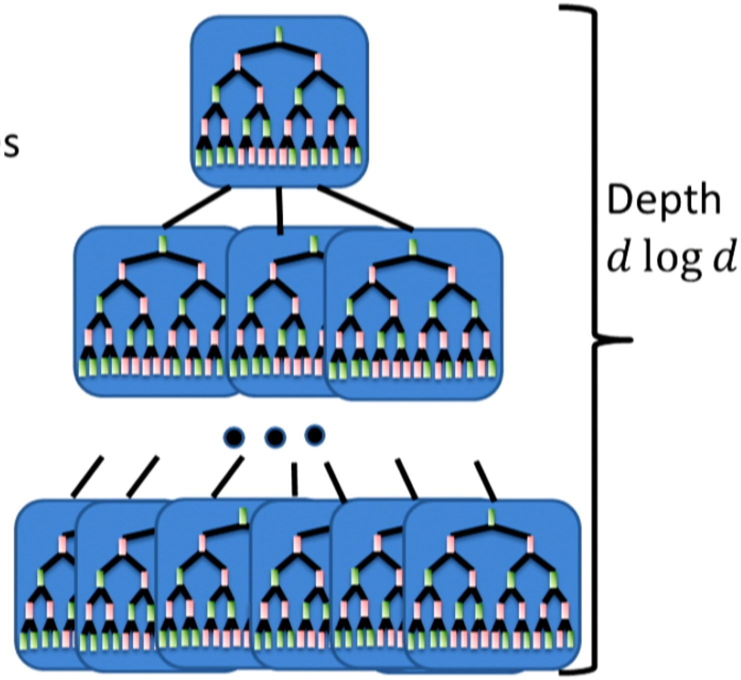
We found algorithm for k-fault tree using $(2^k \times depth^2)$ queries

1-Fault NAND Tree



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

$[\text{1–Fault NAND Tree}]^{\log d}$



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

Quantum Adversary Upper Bound

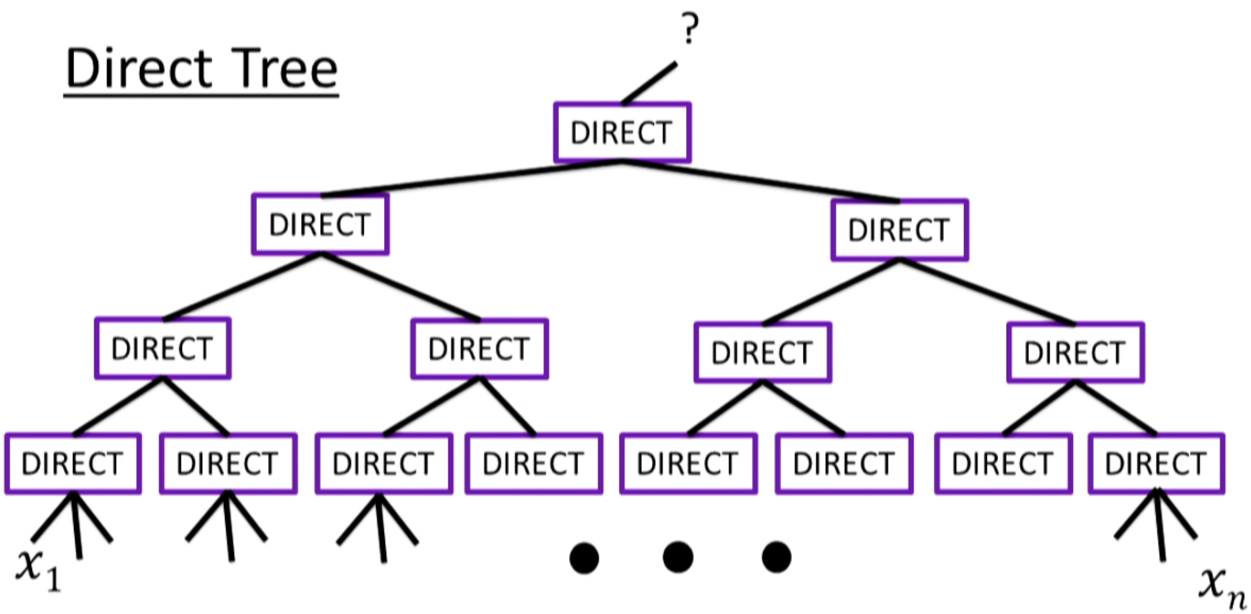
$1-$Fault NAND Tree is a Boolean function

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 $O\left(d^{3/\log d}\right) = \mathrm{O}\left(2^{3\log d/\log d}\right) = O(1)$

Extension: c-Fault Direct Tree

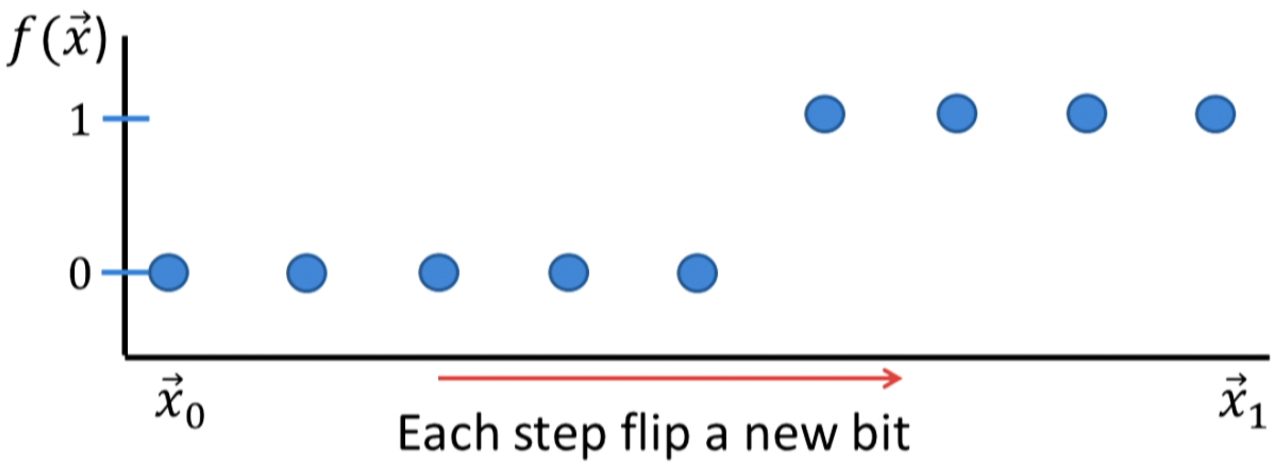
Direct Tree



DIRECT → generalization of monotonic.

Direct Functions

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



Proving Quantum Adversary Upper Bound

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

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

Proof [SK '11]:

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

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

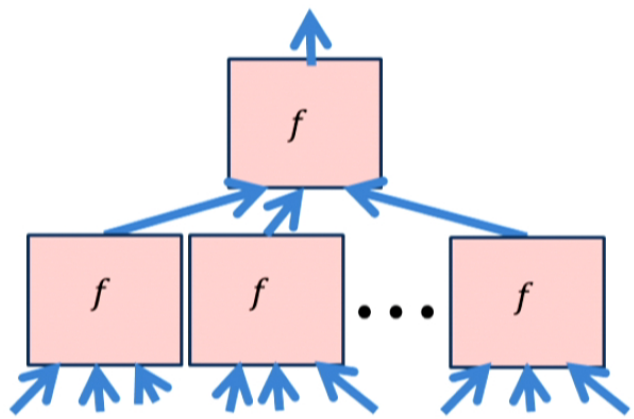
$$ADV^{\pm}(f)^k = O(T)$$

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

Proving Quantum Adversary Upper Bound

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

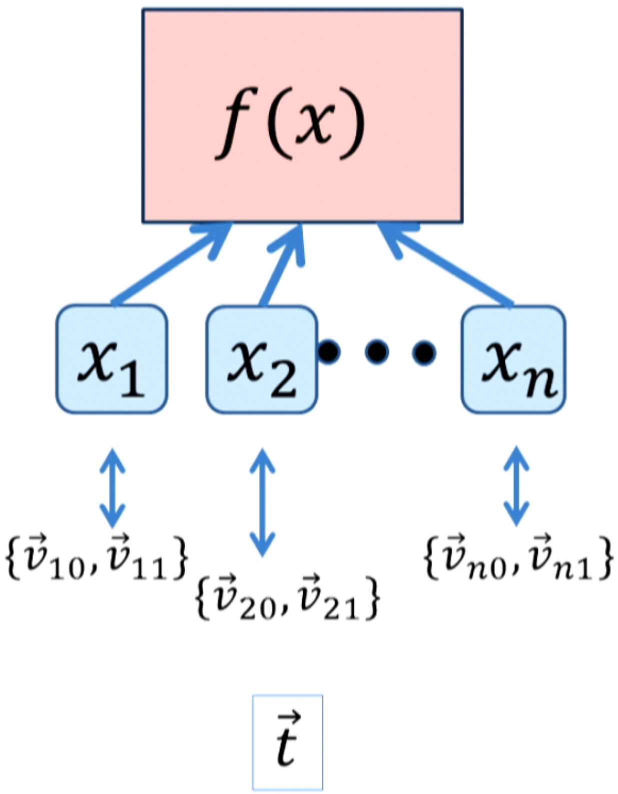
- Given a matrix that maximizes objective function of SDP of $ADV^{\pm}(f)$, construct a matrix satisfying the SDP for f^k
- When f is partial, set entries corresponding to non-valid inputs to 0. Need to check that things go through



Matching Algorithm?

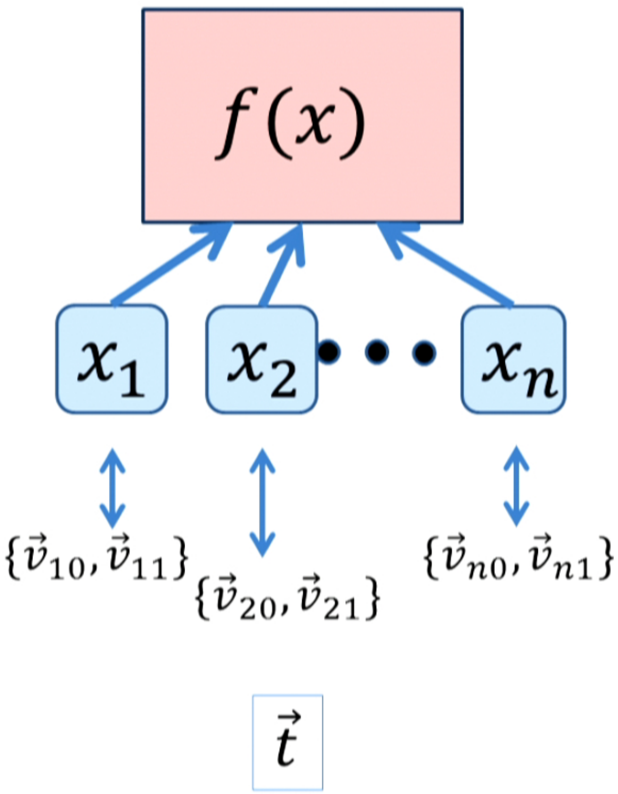
- For all c-Fault Direct Trees, O(1) query algorithms must exist.
- Can we find them?

Method 1: Span Programs [Zhan, Hassidim, SK '12]



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

Method 1: Span Programs [Zhan, Hassidim, SK '12]



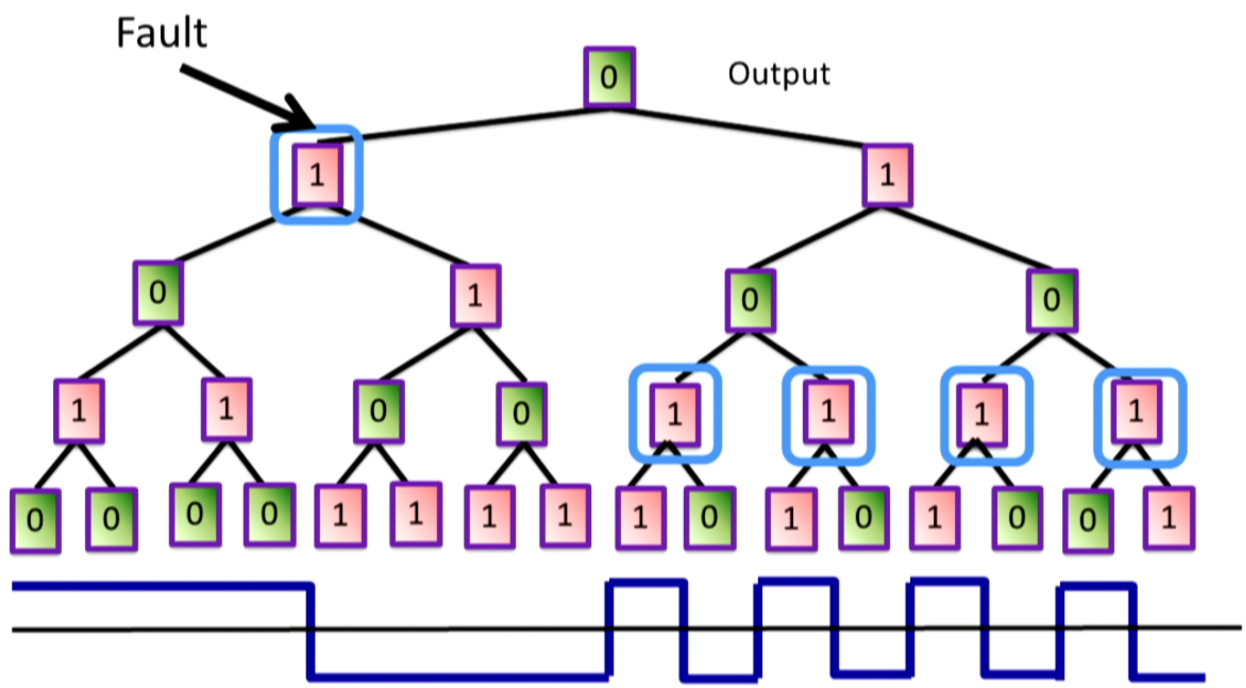
$f(\vec{x}_i) = 1$ iff
$\vec{t} \in SPAN\{\vec{v}_{1i}, \vec{v}_{2i}, \ldots, \vec{v}_{ni}\}$

AND:

$$\vec{v}_{11} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \vec{v}_{21} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \vec{t} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

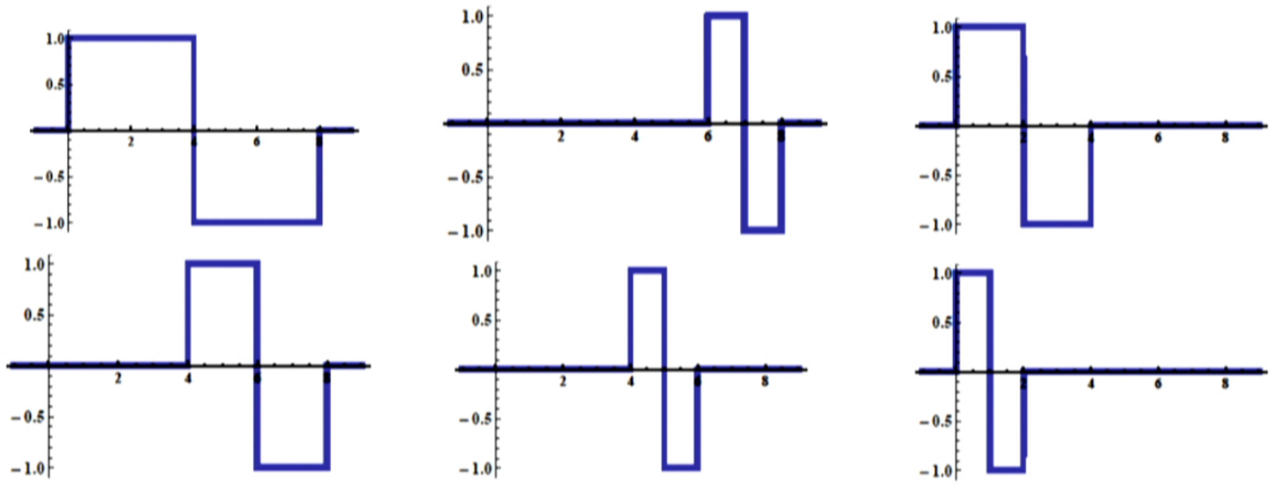
All other: $\binom{0}{0}$

Method 2: Haar Transform



Method 2: Haar Transform

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



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

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]