Super-Polynomial Quantum Speedups for Boolean Evaluation Trees with Hidden Structure

Bohua Zhan[*], Shelby Kimmel[†], Avinatan Hassidim[‡]

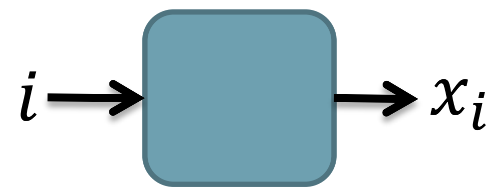
[*]Princeton University
[†]Massachusetts Institue of Technology
[‡] Bar Ilan University and Google

Super-Polynomial Quantum Speedups

Oracle Model

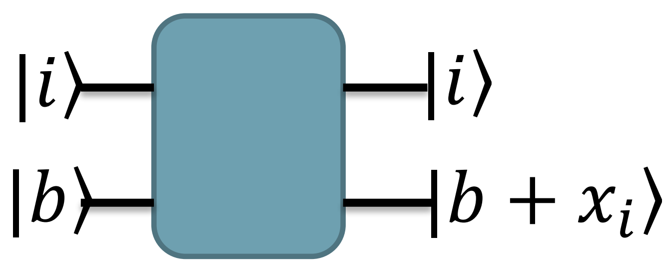
Goal: Determine the value of $f(x_1, \ldots, x_n)$ for a known function f, with an oracle for x

Classical Oracle



$R(f)$
(randomized bounded error query complexity)

Quantum Oracle



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

Only care about # of oracle calls (queries)

Super-Polynomial Quantum Speedups

Example of Super-Polynomial Speedup

Hidden Subgroup Problem:

Given: Group G

Promised: $\exists\, H \subseteq G$ s.t. $x_i = x_j \Leftrightarrow i, j \in G$ in same left coset of H

Problem: Determine H

$$Q(f) = O(\log|G|)$$

$$R(f) = \Omega(|G|)$$

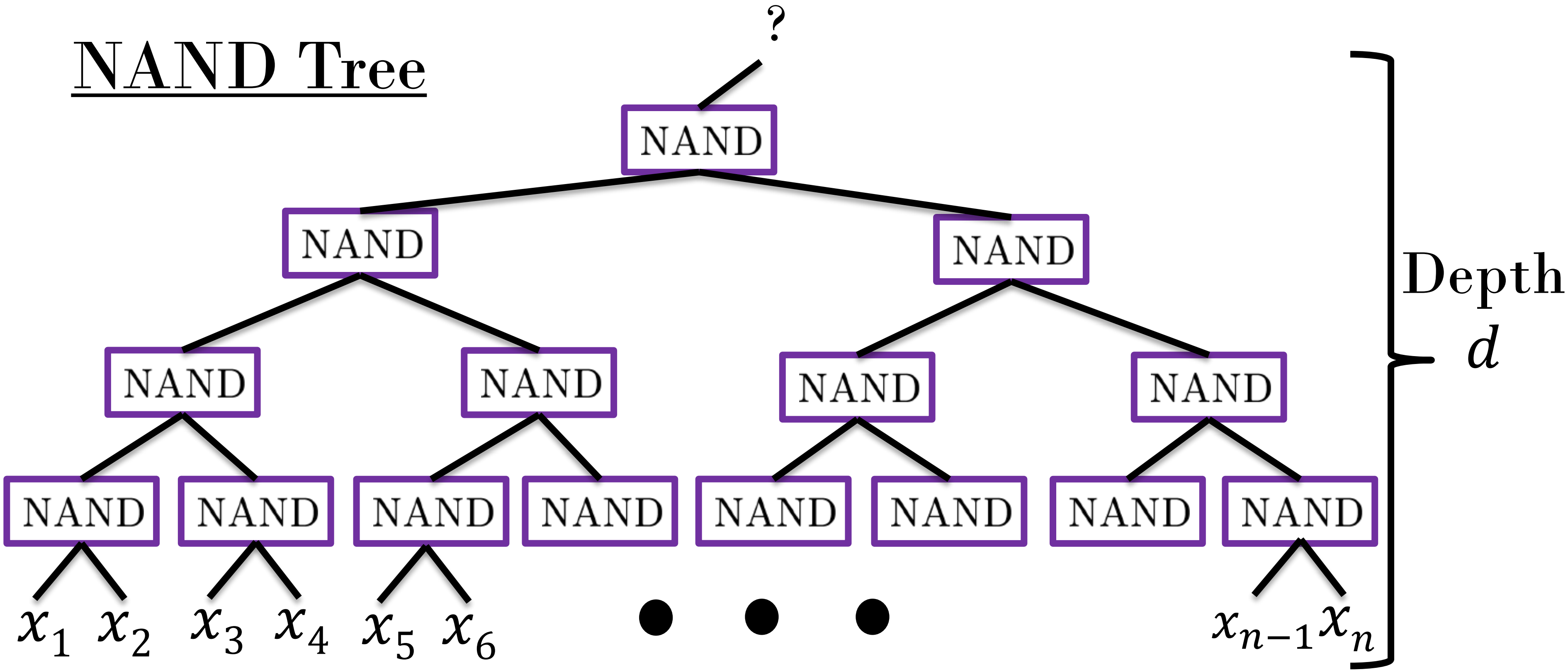
[Simon '94, Boneh and Lipton '95, Hallgren et al '03, Etttinger et al '04 ….]

Super-Polynomial Quantum Speedups for Boolean Evaluation Trees with Hidden Structure

Bohua Zhan*, Shelby Kimmel†, Avinatan Hassidim‡

*Princeton University
†Massachusetts Institue of Technology
‡ Bar Ilan University and Google

Boolean Evaluation Trees



$$\mathrm{Q}(\text{NAND TREE}) = O(2^{0.5d})$$ [Farhi et al '08]

$$R(\text{NAND TREE}) = \Omega(2^{0.753d})$$ [Saks and Widgerson '86]

Boolean Evaluation Trees

$$Q(\text{NAND TREE}) = O(2^{0.5d})$$

$$R(\text{NAND TREE}) = \Omega(2^{0.753d})$$

Fact: No super-poly speedups for total Boolean functions [Beals et. al 1998]

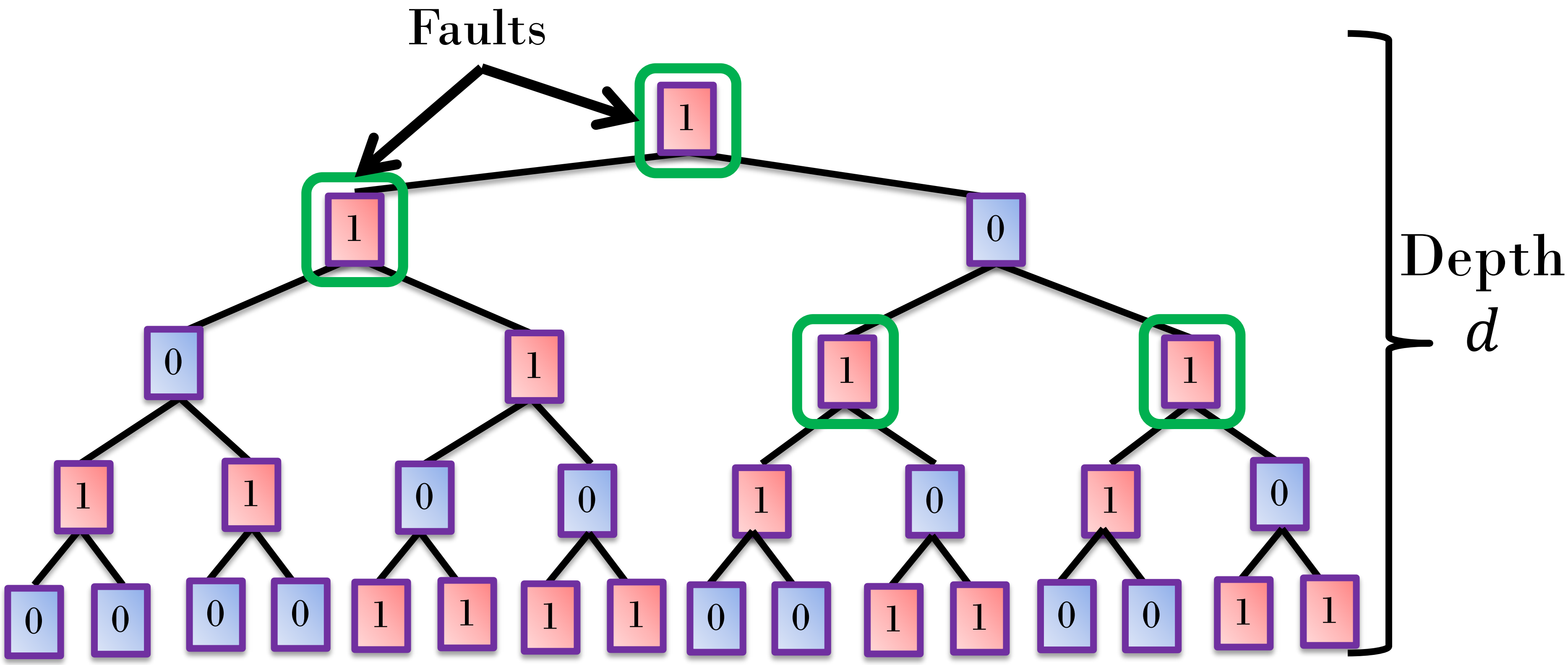
For a super-poly speed up in Boolean evaluation trees, need a promise on the input

Super-Polynomial Quantum Speedups
for Boolean Evaluation Trees with Hidden Structure

Bohua Zhan[*], Shelby Kimmel[†], Avinatan Hassidim[‡]

[*]Princeton University
[†]Massachusetts Institue of Technology
[‡] Bar Ilan University and Google

NAND Tree Hidden Structure

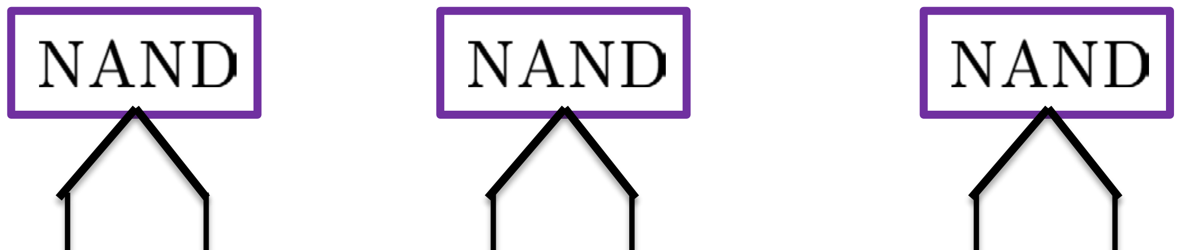


k =max # of faults on any path

NAND Tree Hidden Structure

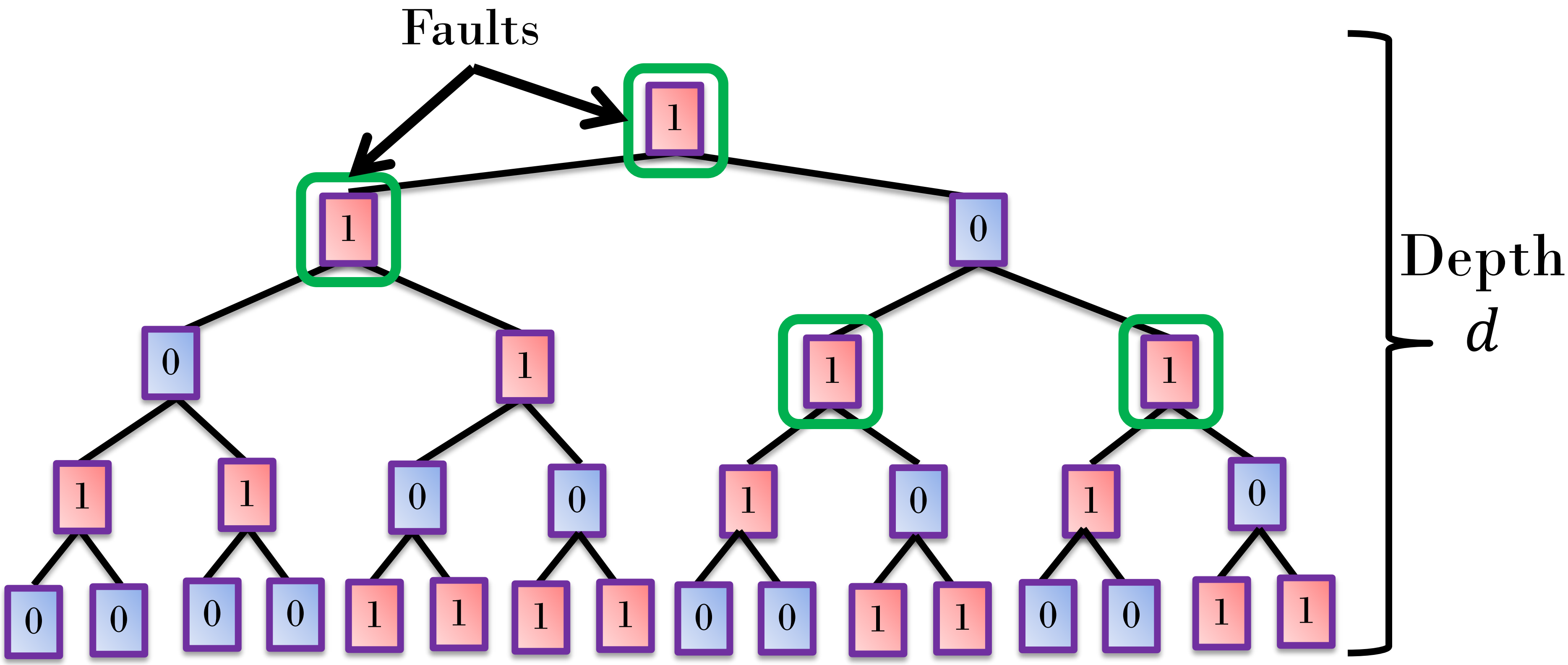
Input Affects Query Complexity

[Reichardt and Spalek '08]



	1 1	1 0	0 0
Our Algorithm:	FAST	SLOW	FAST
R&S Algorithm	MED	MED	FAST

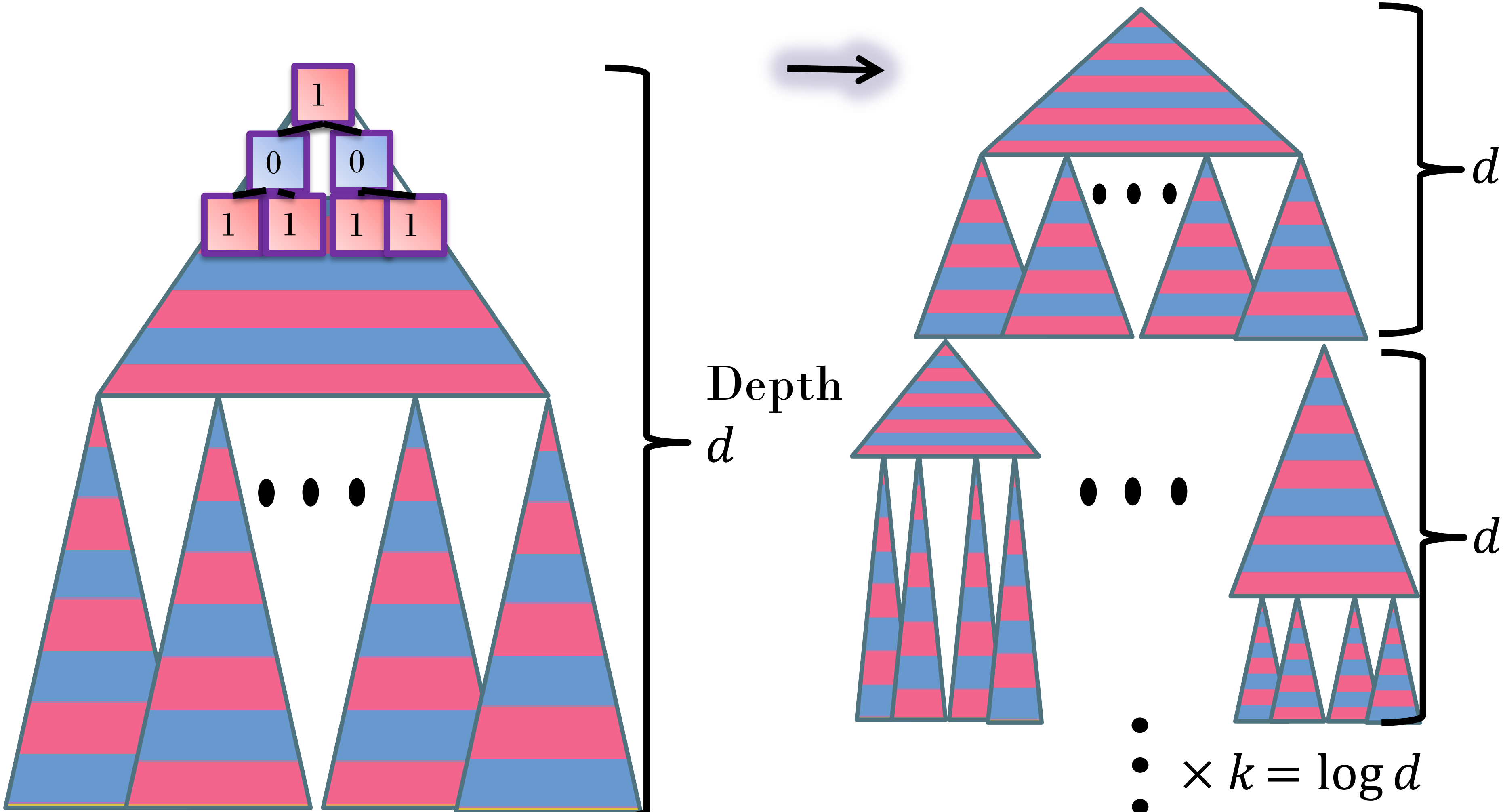
NAND Tree Hidden Structure



k =max # of faults on any path

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

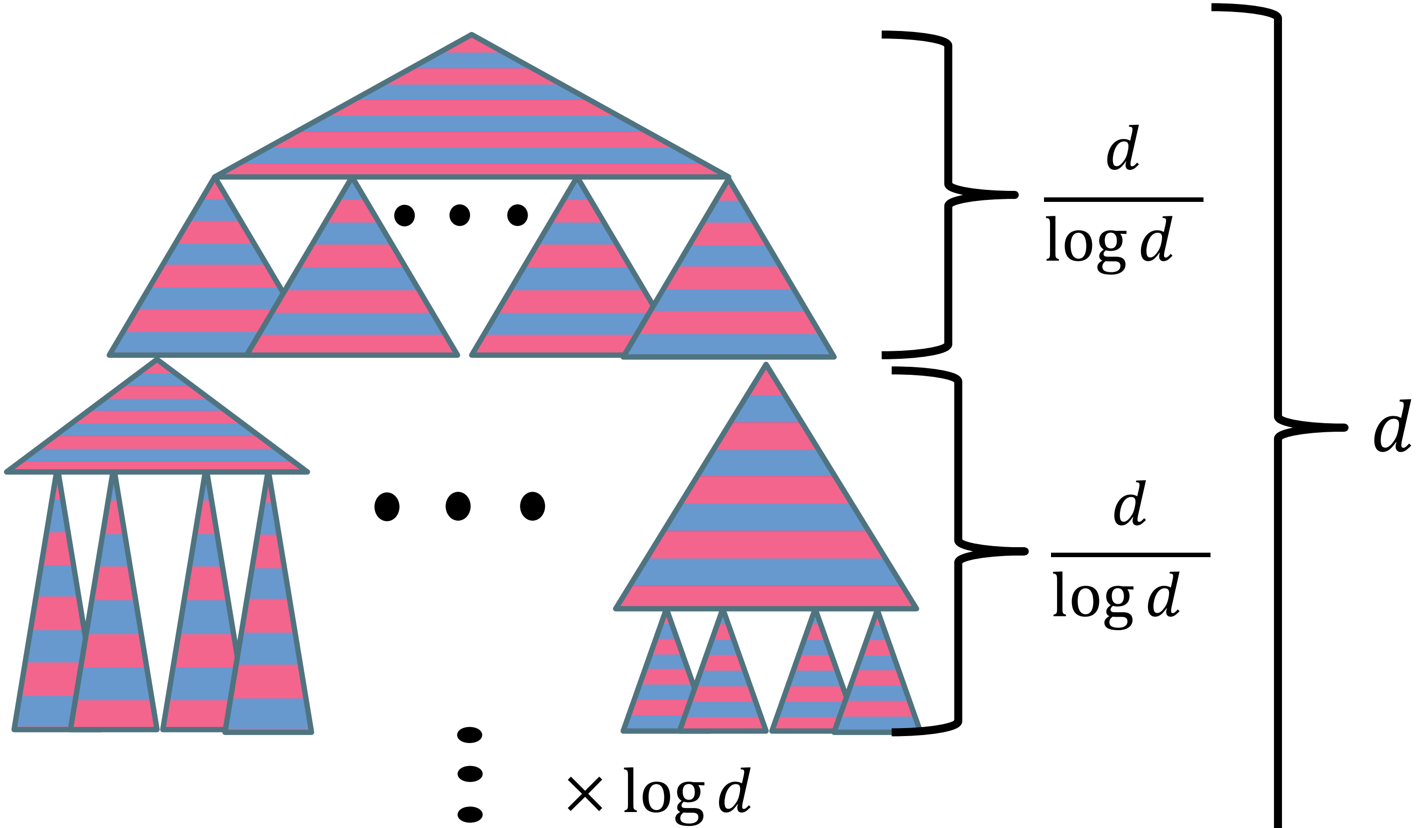
Classical Lower Bound



$R(f) = \Omega(\log d)$

$R(f) = \Omega(d^{\log \log d})$

Super-polynomial Separation



k =max # of faults on any path = $\log d$

$\mathrm{Q}(f) = O(d^3)$

$R(f) = d^{\Omega(\log\log d)}$

Extensions

- Not just NAND Trees
  - Majority Trees
  - Threshold Trees
  - “Direct” Trees

Conclusions and Future work

- We found super-polynomial speed up for many Boolean trees
- Hidden structure based on algorithm, can we do the same for other algorithms?
- Get rid of scaling with depth in quantum algorithm?
- Simplify classical proof?