Speed-ups for Quantum Algorithms with Easier Inputs

Shelby Kimmel, Jay-U Chung, Noel Anderson

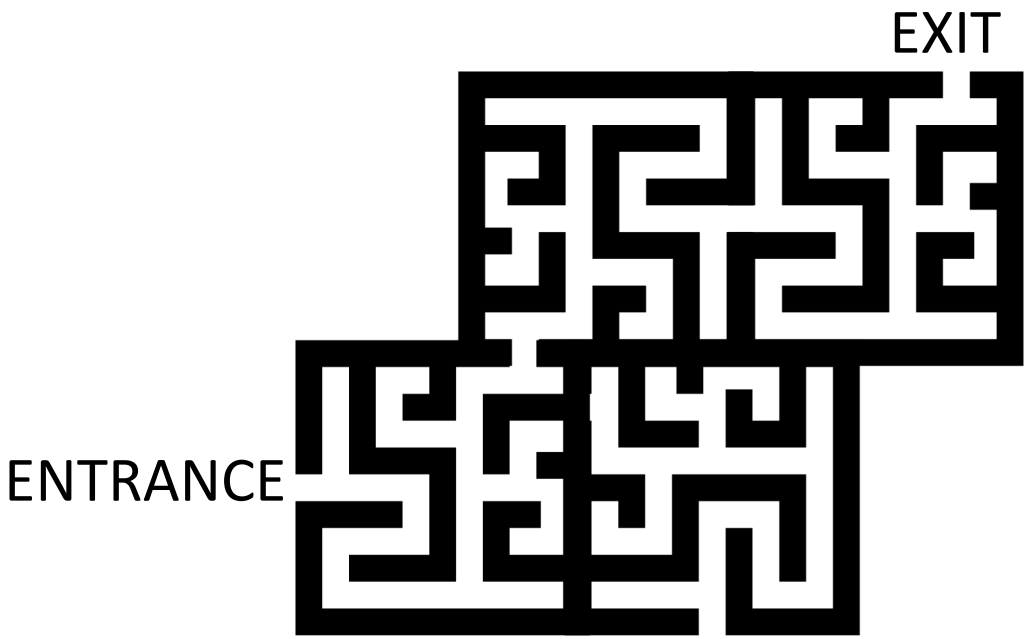
Middlebury College

- Noel Anderson, **SK**, and Jay-U Chung arXiv:2012.01276
- Kai DeLorenzo, **SK**, Teal Witter, arXiv:1904.05995 (TQC 2019)
- Michael Jarret, Stacey Jeffery, **SK**, Alvaro Piedrafita, arXiv:1804.10591 *(*ESA 2018*)*
- Stacey Jeffery, **SK**: arXiv:1704.00765 (Quantum vol 1 p 26)

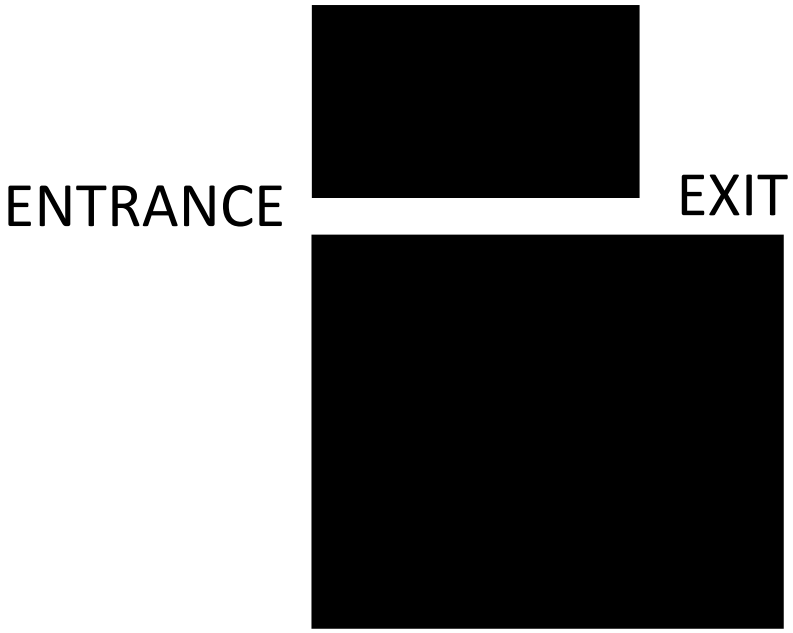
Easy vs Hard Instances

Path-Detection:

Harder:



Easier:



Easy vs Hard Instances: Classically

Ex: Path-Detection

Run search from ENTRANCE for time T (based on size of maze).

- If find EXIT, stop and output *YES,* otherwise continue
- If after time T don’t find EXIT, output *NO*

Easy vs Hard Instances: Classically

Ex: Path-Detection

Run search from ENTRANCE for time T (based on size of maze).

- If find EXIT, stop and output *YES,* otherwise continue
- If after time T don’t find EXIT, output *NO*

If easier: shorter run time

If harder: longer run time

Easy vs Hard Instances: Classically

Ex: Path-Detection

Run search from ENTRANCE for time T (based on size of maze).

- If find EXIT, stop and output *YES,* otherwise continue
- If after time T don't find EXIT, output *NO*

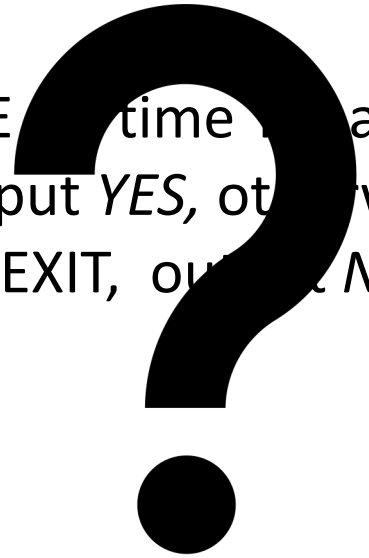
If easier: shorter run time

If harder: longer run time

Key properties:

1. Can check status mid-algorithm and continue running
2. Witness of completion (if find EXIT, convinced there is a path)

Easy vs Hard Instances: Quantumly



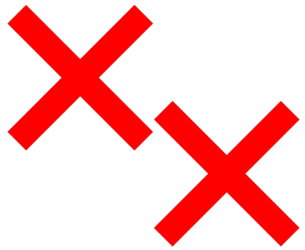
Run search from ENTRANCE for time T (based on size of maze).

- If find EXIT, stop and output *YES,* otherwise continue
- If after time T don't find EXIT, output *NO*

If easier: shorter run time
If harder: longer run time

Key properties:

1. Can check status mid-algorithm and continue running
2. Witness of completion (if find EXIT, convinced there is a path)



Our Result

For a large class of quantum algorithms that previously used worst-case complexity for all instances:

Our Result

For a large class of quantum algorithms that previously used worst-case complexity for all instances:

Create a modified algorithm:

- If hard instance:

Our Result

For a large class of quantum algorithms that previously used worst-case complexity for all instances:

Create a modified algorithm:

- If hard instance: (approximately) previous worst-case complexity

Our Result

For a large class of quantum algorithms that previously used worst-case complexity for all instances:

Create a modified algorithm:

- If hard instance: (approximately) previous worst-case complexity
- If easier instance:

Our Result

For a large class of quantum algorithms that previously used worst-case complexity for all instances:

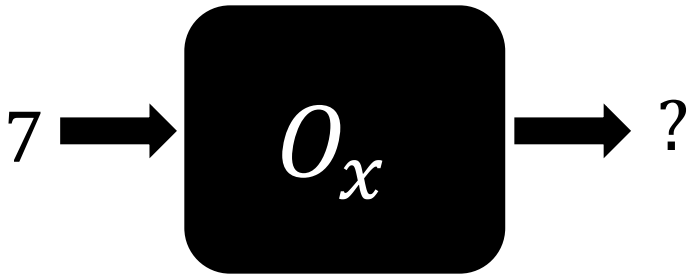
Create a modified algorithm:

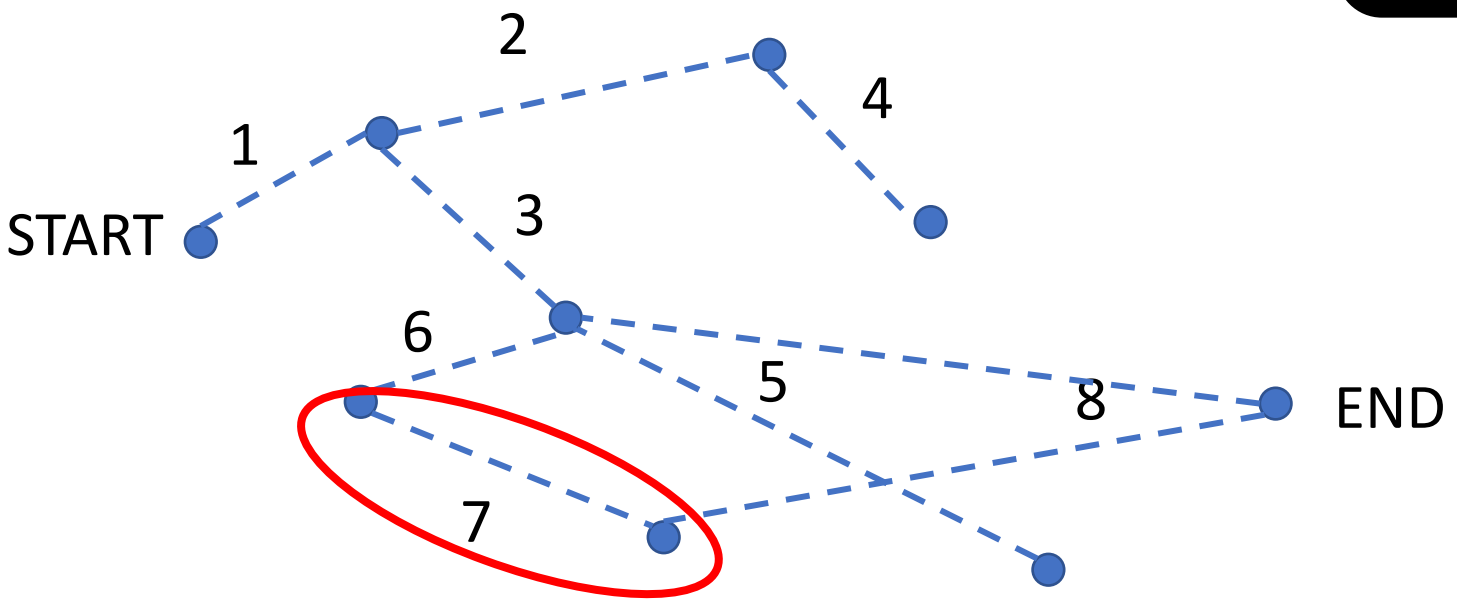
- If hard instance: (approximately) previous worst-case complexity
- If easier instance: better complexity

Talk Outline

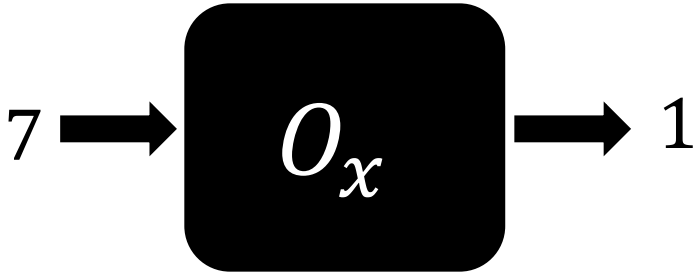
1. Oracle Model
2. Challenges:
    a. No check and continue
    b. No (easily accessible) witness of completion
3. Applications & Future Work

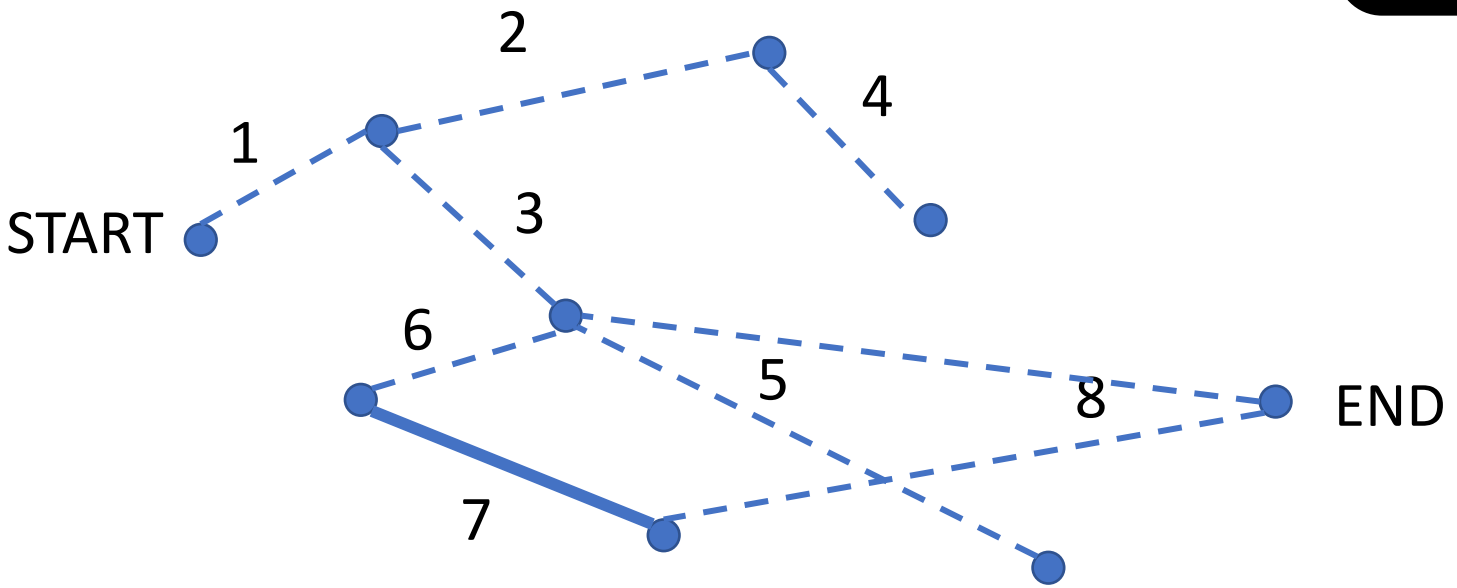
Oracle Model



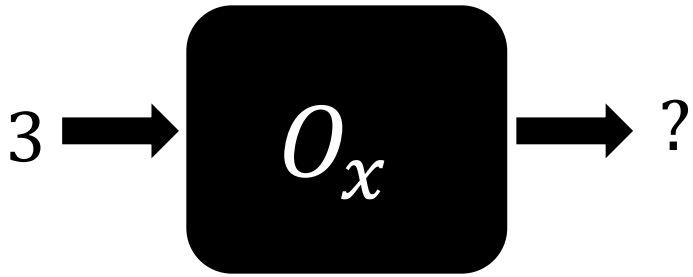


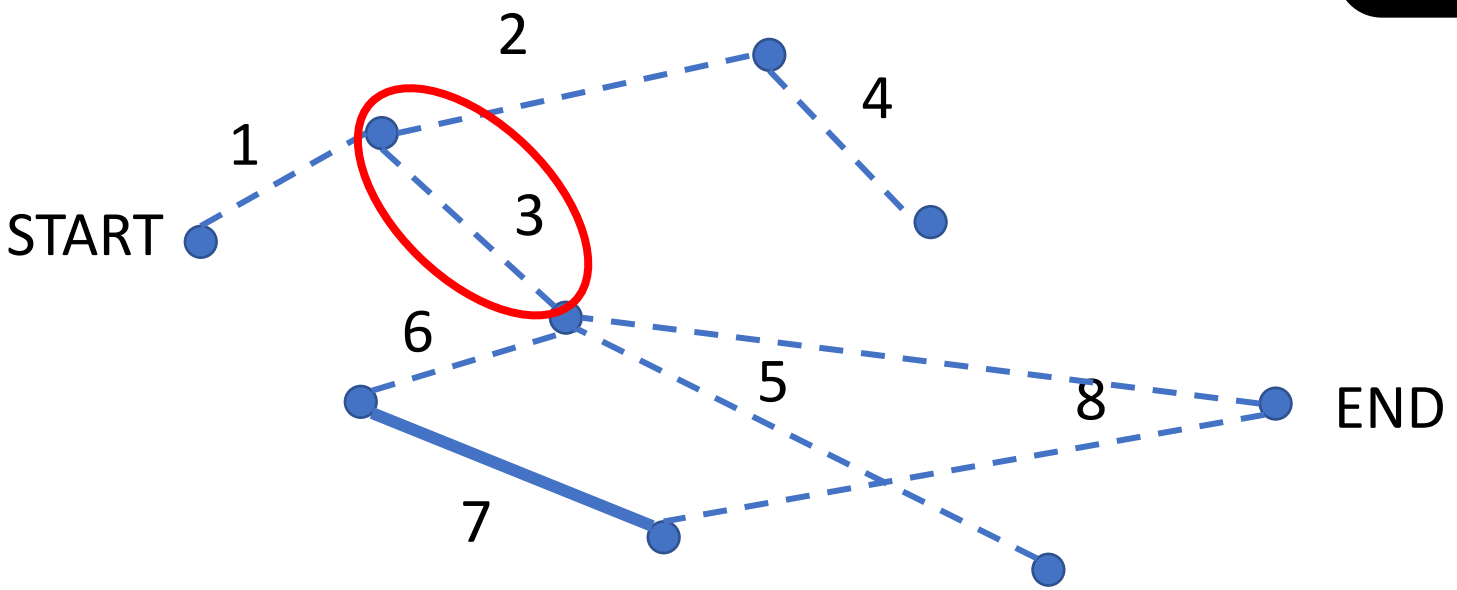
Oracle Model



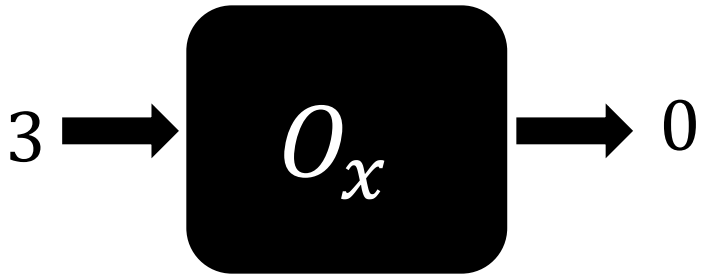


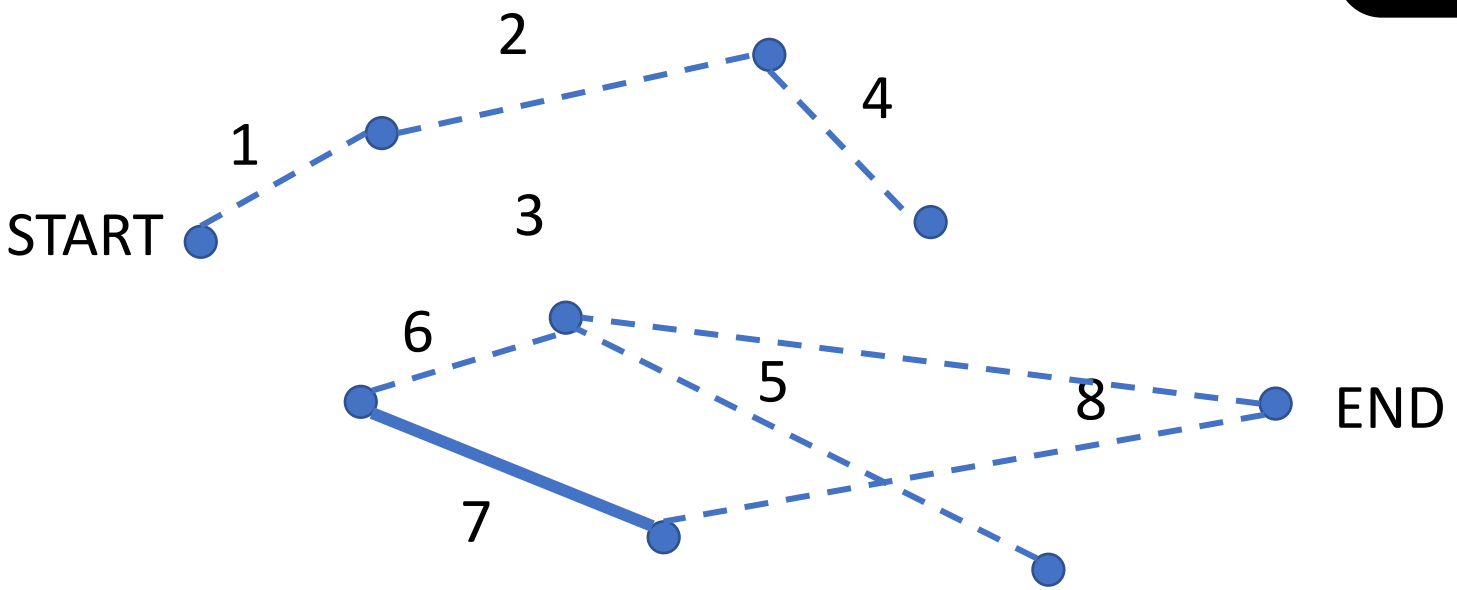
Oracle Model



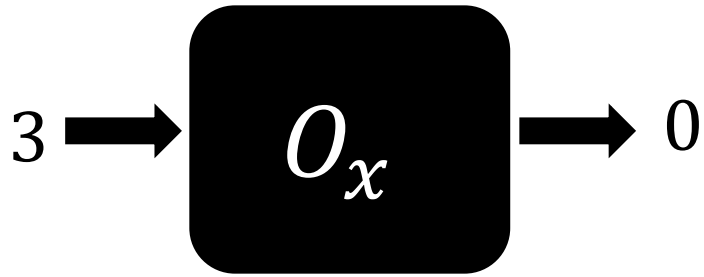


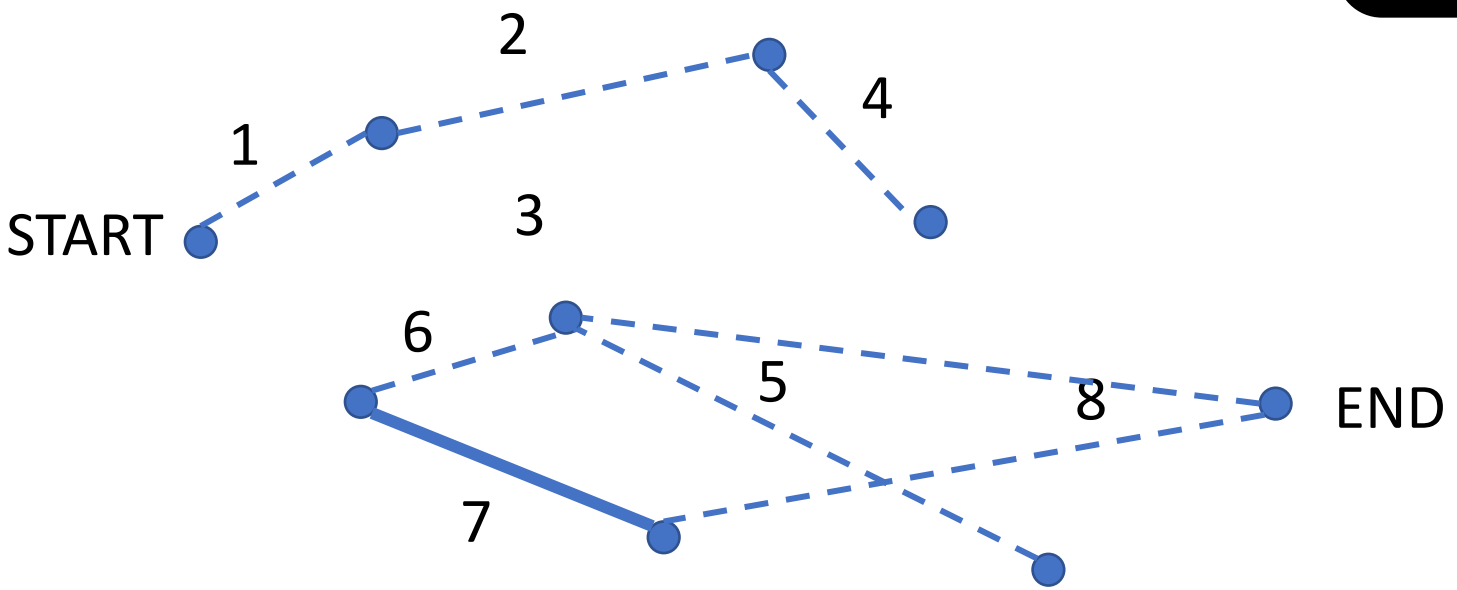
Oracle Model





Oracle Model





Goal: to solve our problem while querying as few times as possible

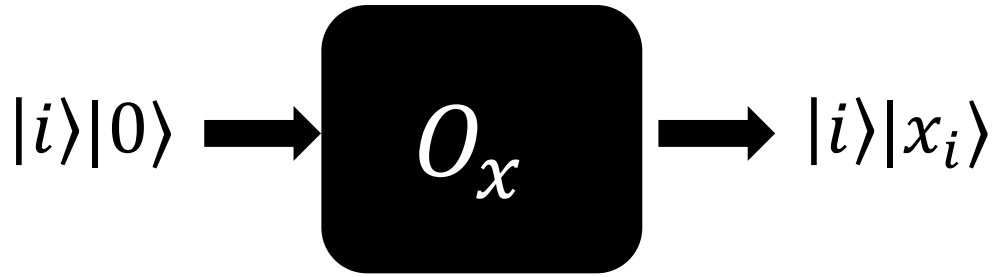
Quantum Oracle Model

Problem:

- Algorithm for $f: X \to \{0,1\}$ (e.g. is there a path?)

Input

O_x for instance $x \in X$
(e.g. edge positions)



Output:

$f(x)$, using as few queries as possible

- in worst case over X
- while using fewer queries on easier instances

# of queries – "runtime" – query complexity

Talk Outline

1. Oracle Model
2. Challenges:
   a. No check and continue ← **Easy**
   b. No (easily accessible) witness of completion ← **Harder**
3. Applications & Future Work

Quantum If/Else?

Classically: Easy

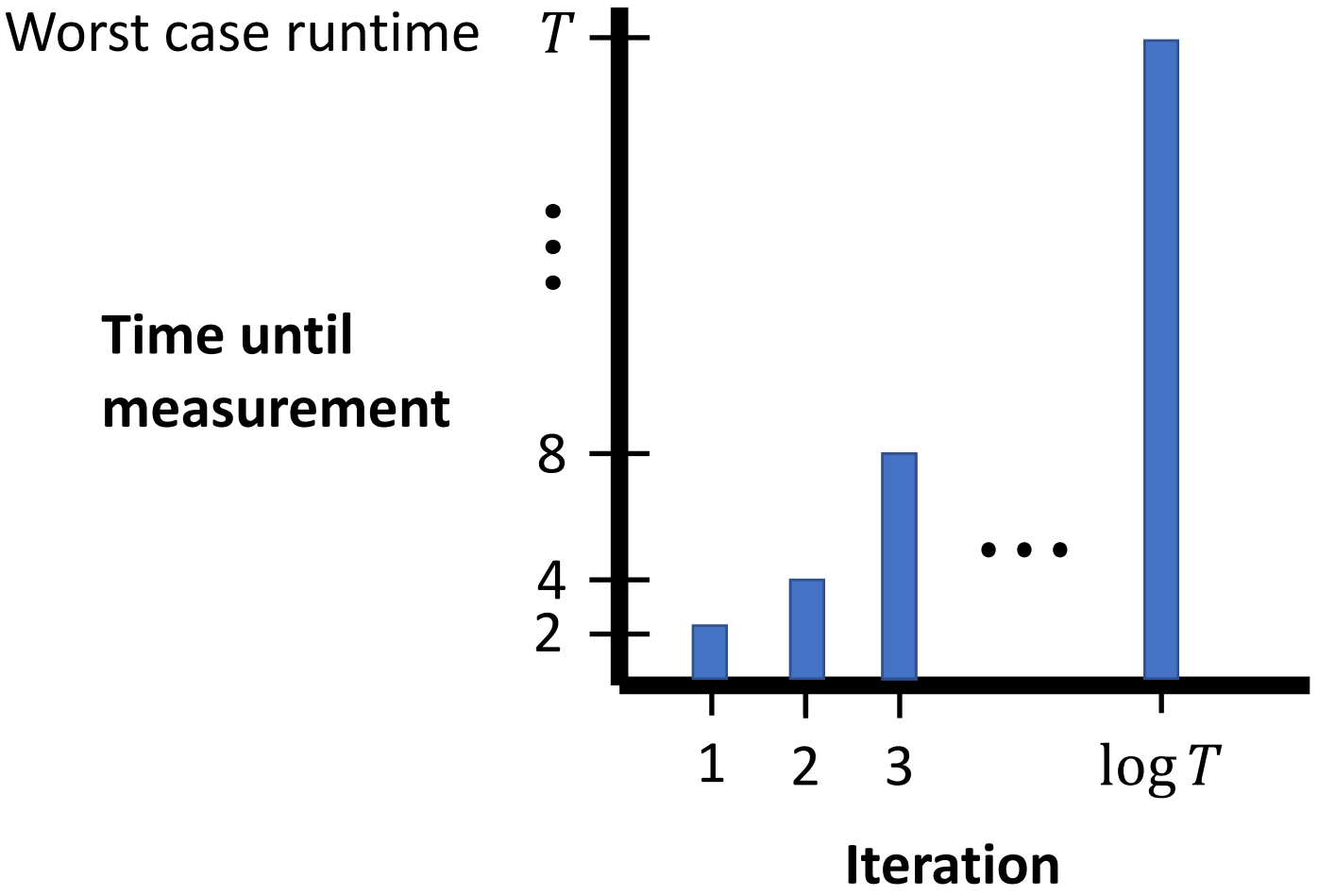
Quantum If/Else?

Classically: Easy

Quantumly:

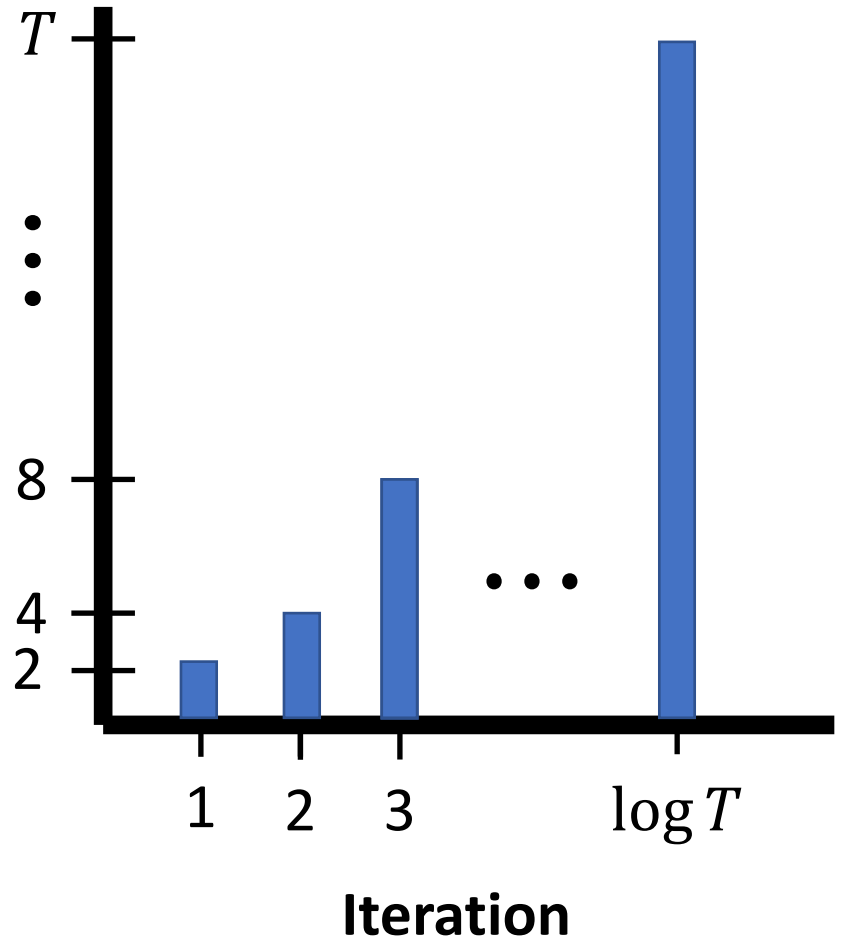
- if/else → measurement
- Measurement → collapse
- Collapse → computation ruined

Quantum If/Else?



Quantum If/Else?

Worst case runtime

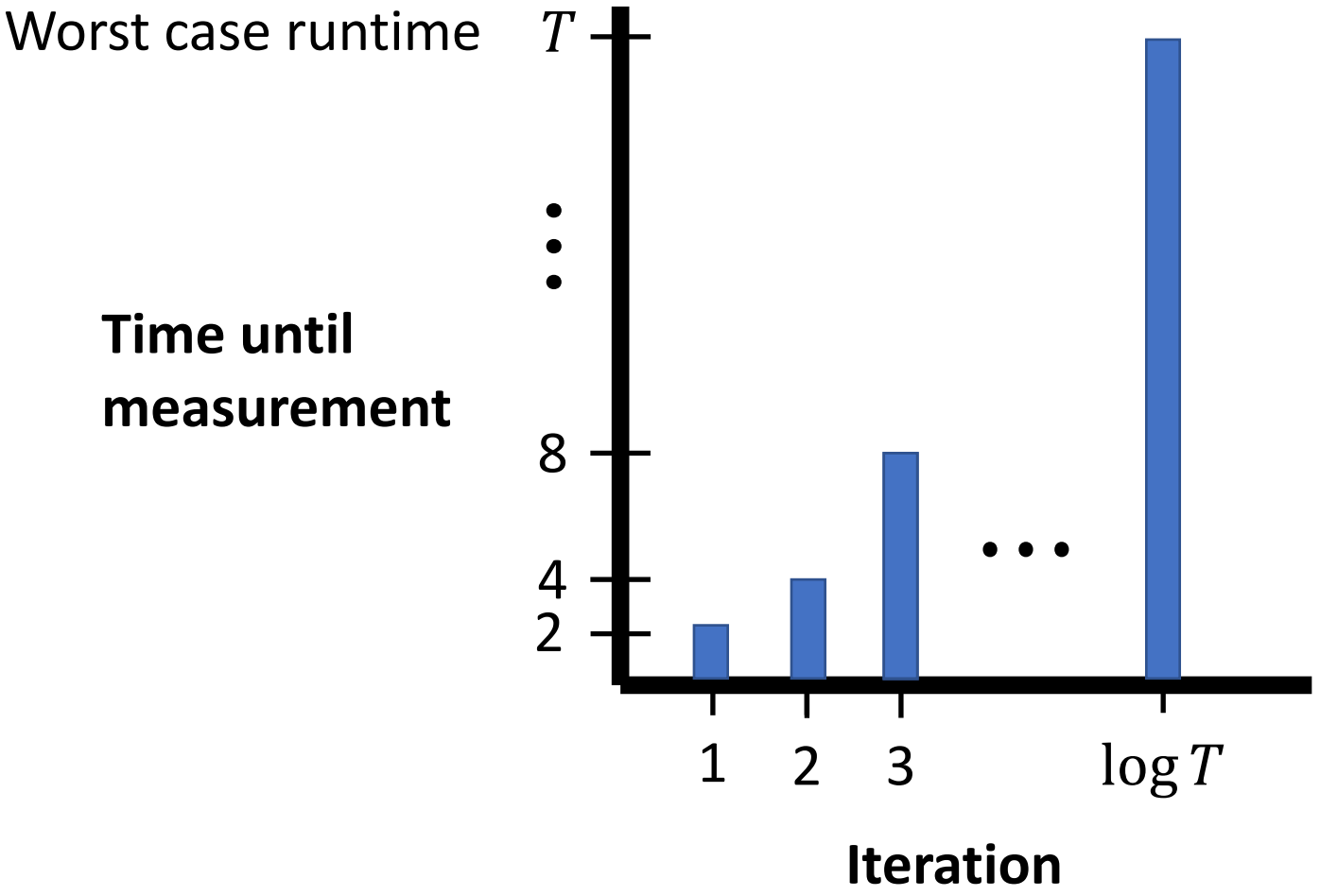


Total Runtime:

$O(T)$

(Geometric series)

Quantum If/Else?



Total Runtime:

$O(T)$

(Geometric series)

Runtime with Error Reduction:

$\tilde{O}(T)$

Talk Outline

1. Oracle Model
2. Challenges:
   a. No check and continue ← **Easy** ☑
   b. No (easily accessible) witness of completion ← **Harder**
3. Applications & Future Work

Our Result

Span program algorithms

For a large class of quantum algorithms that previously used worst-case runtime for all instances:

Create a modified algorithm:

- If worst-case instance: (approximately) previous worst case run time
- If easier instance: shorter run time

Span Program Algorithms

∀ Boolean function, ∃ span program algorithm:

- Query optimal for worst-case inputs

Reichardt 2009 FOCS, Reichardt 2011 SODA

Span Program Algorithms

∀ Boolean function, ∃ span program algorithm:

- Query optimal for worst-case inputs
- Not known how to get a speed-up for easier instances

Reichardt 2009 FOCS, Reichardt 2011 SODA

Span Program Algorithms

∀ Boolean function, ∃ span program algorithm:

- Query optimal for worst-case inputs
- Not known how to get a speed-up for easier instances*

*If don't know ahead of time that instance is easy

Reichardt 2009 FOCS, Reichardt 2011 SODA

Span Program Algorithms

∀ Boolean function, ∃ span program algorithm:

- Query optimal for worst-case inputs
- Not known how to get a speed-up for easier instances*

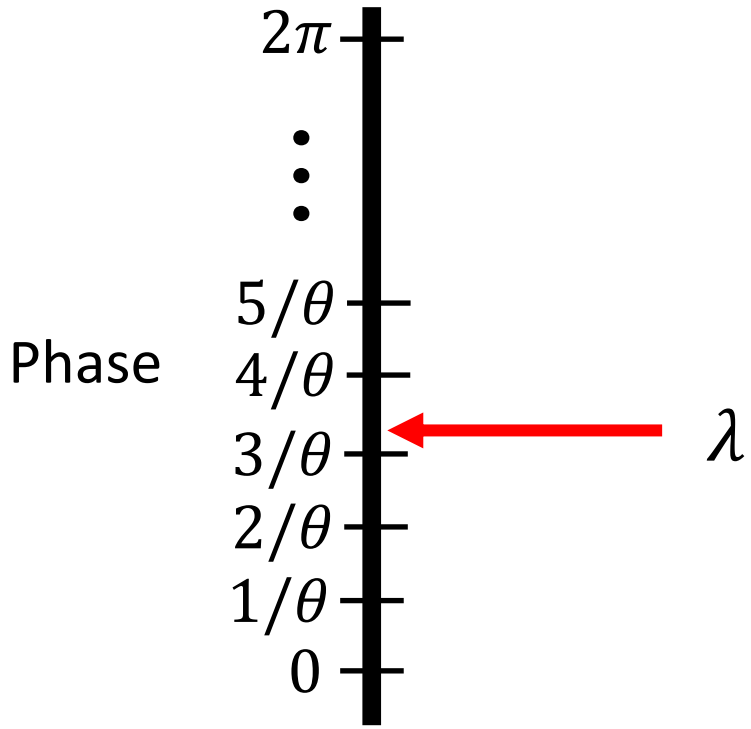
*If don't know ahead of time that instance is easy

Key subroutine: Phase estimation

Reichardt 2009 FOCS, Reichardt 2011 SODA

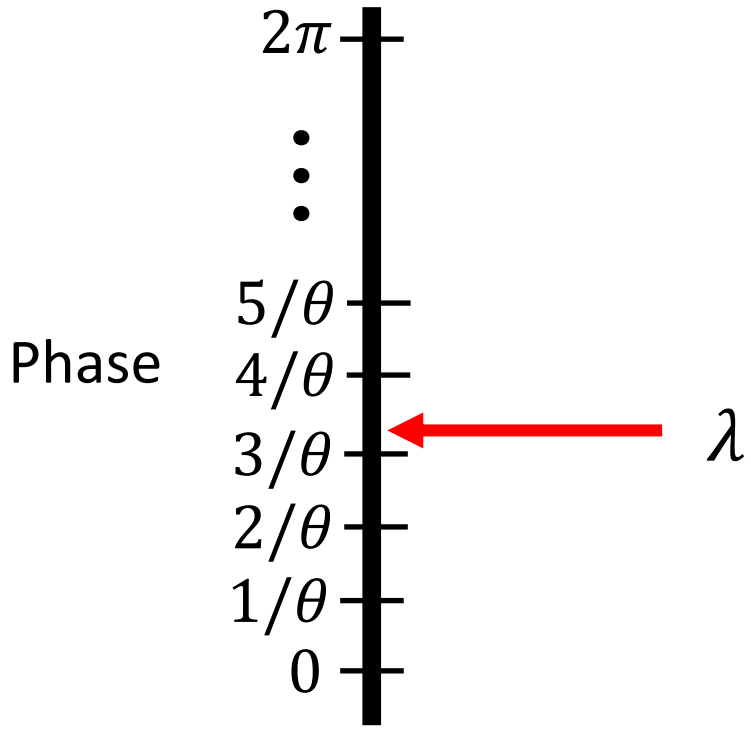
Phase Estimation

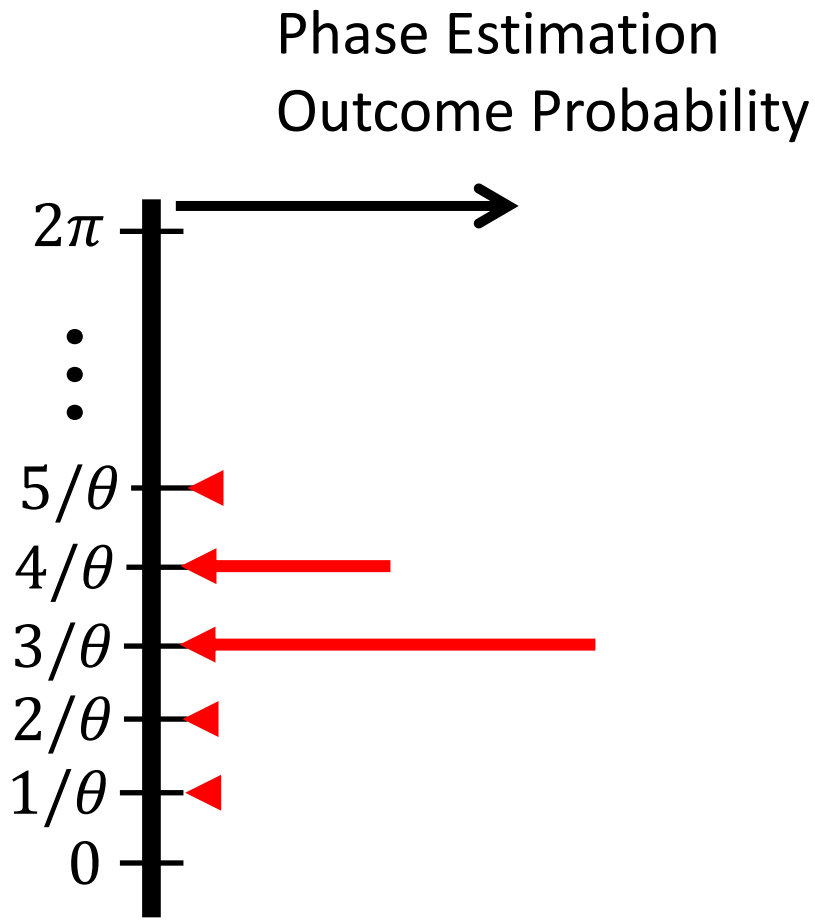
$$U|\psi\rangle = e^{i\lambda}|\psi\rangle$$



Phase Estimation

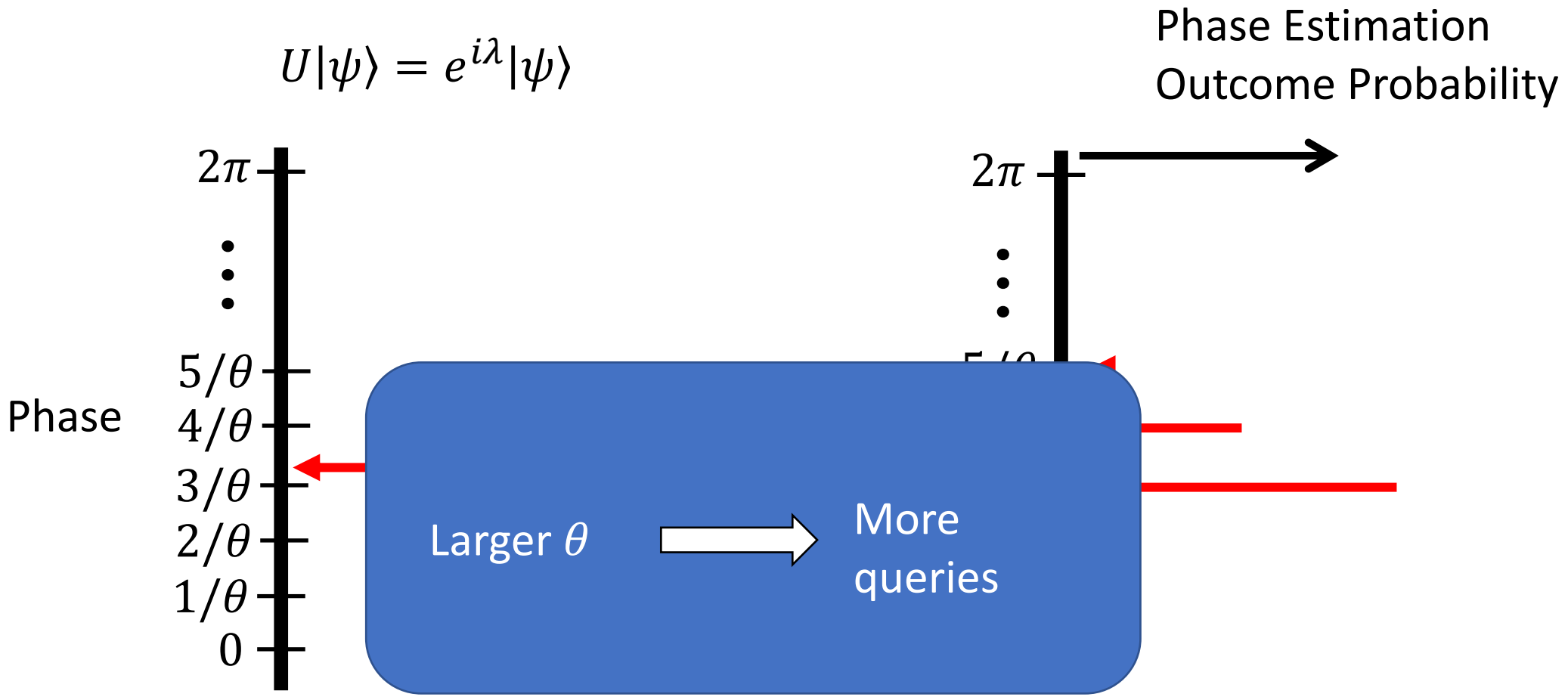
$$U|\psi\rangle = e^{i\lambda}|\psi\rangle$$





Phase Estimation

$$U|\psi\rangle = e^{i\lambda}|\psi\rangle$$



Phase Estimation for Span Programs

Phase estimation in span program algorithm

- If $f(x) = YES$,
  - Output phase = 0 w.h.p
- If $f(x) = NO$
  - Output phase ≠ 0 w.h.p

Phase Estimation for Span Programs

Phase estimation in span program algorithm

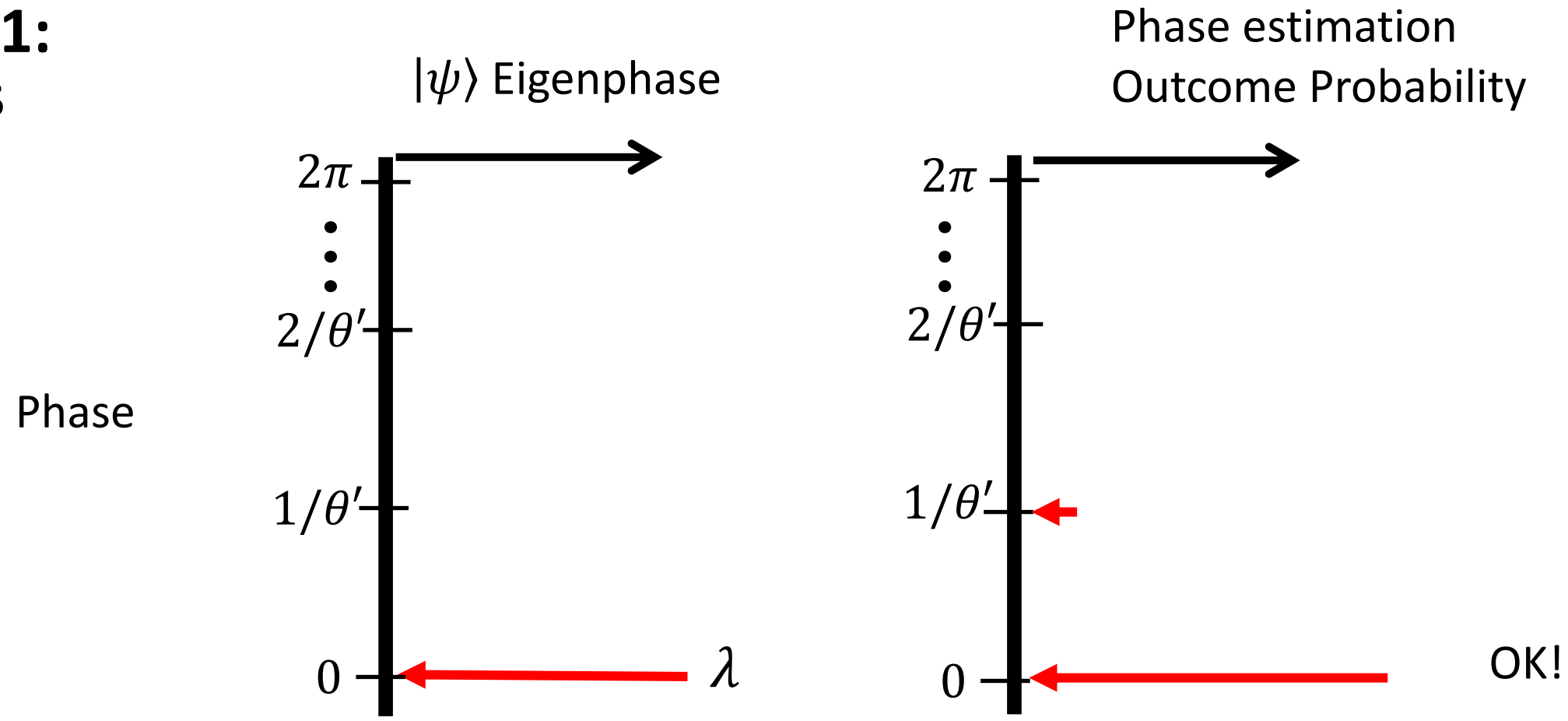
- If $f(x) = YES$,
  - Output phase = 0 w.h.p
- If $f(x) = NO$
  - Output phase ≠ 0 w.h.p

Strategy: Reduce precision

Reduce Precision (θ)

Case 1:

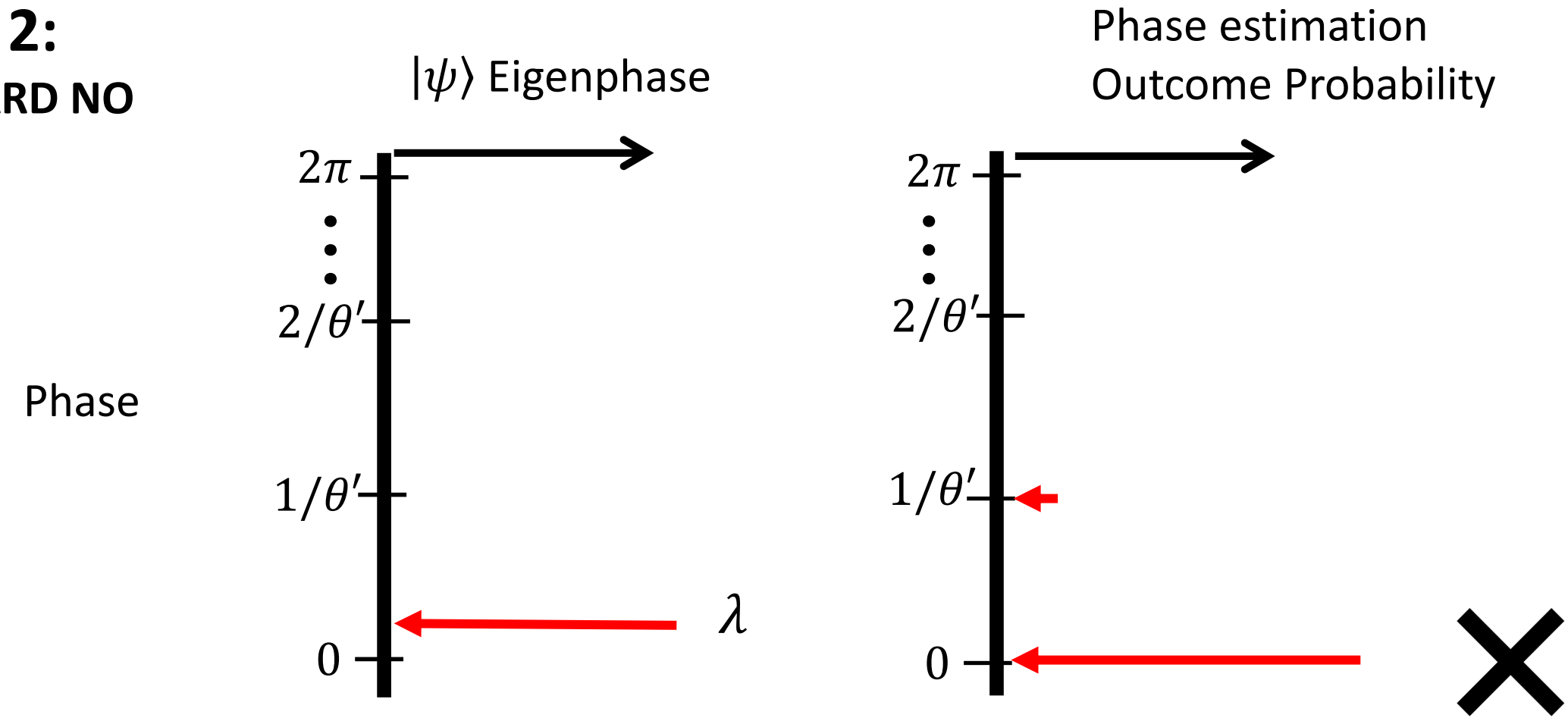
- **YES**



Reduce Precision (θ)

Case 2:

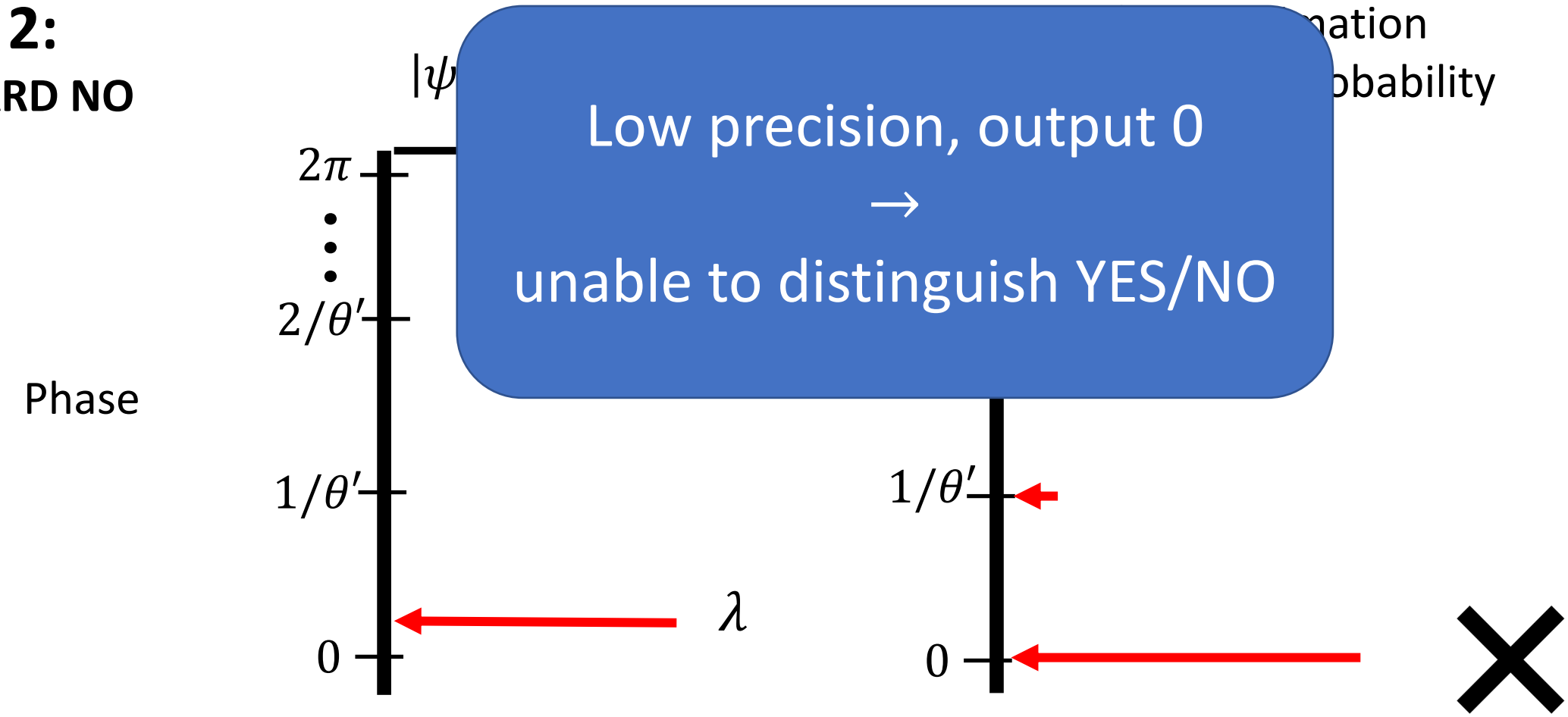
- **HARD NO**



Reduce Precision (θ)

Case 2:

- **HARD NO**



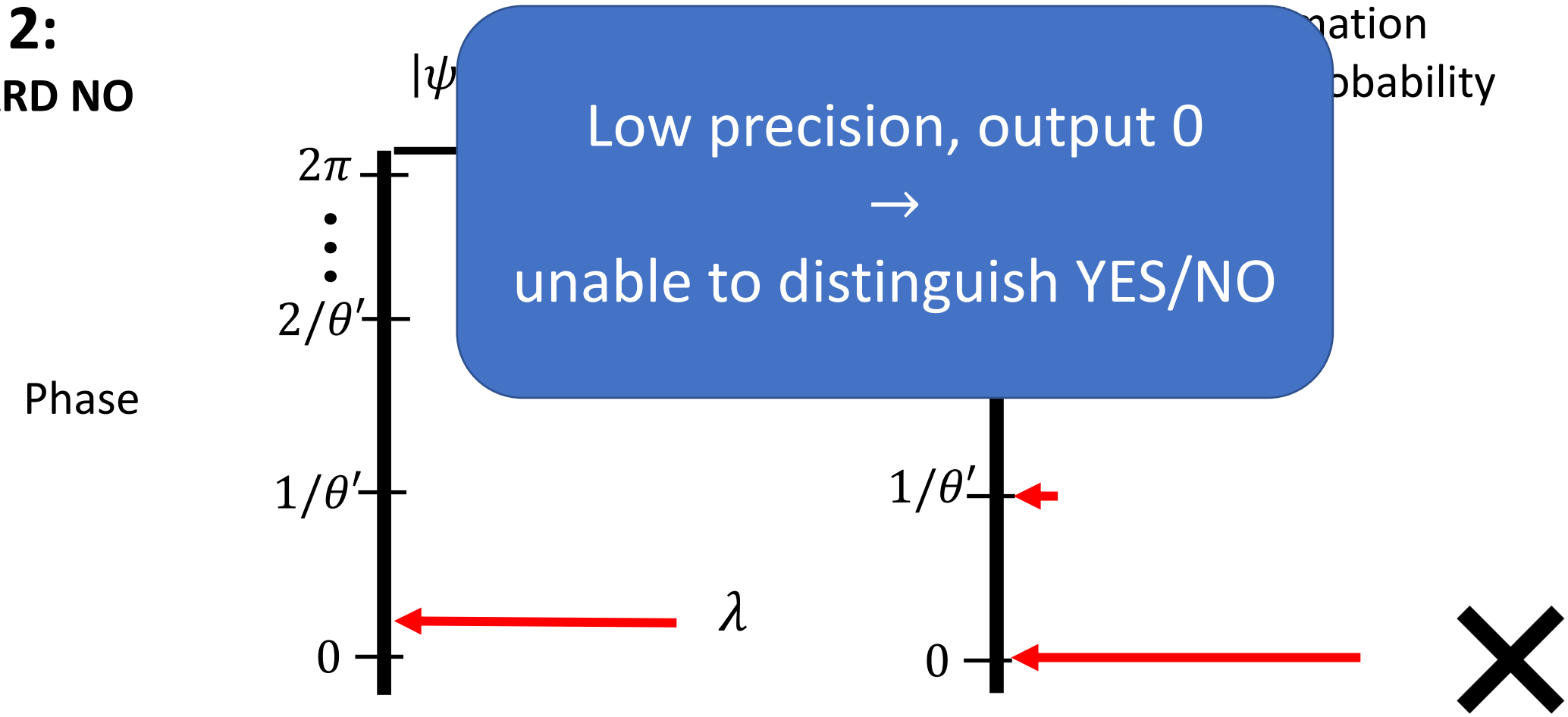
Reduce Precision (θ)

Eigenstate as witness?
Upcoming work

Reduce Precision (θ)

Case 2:

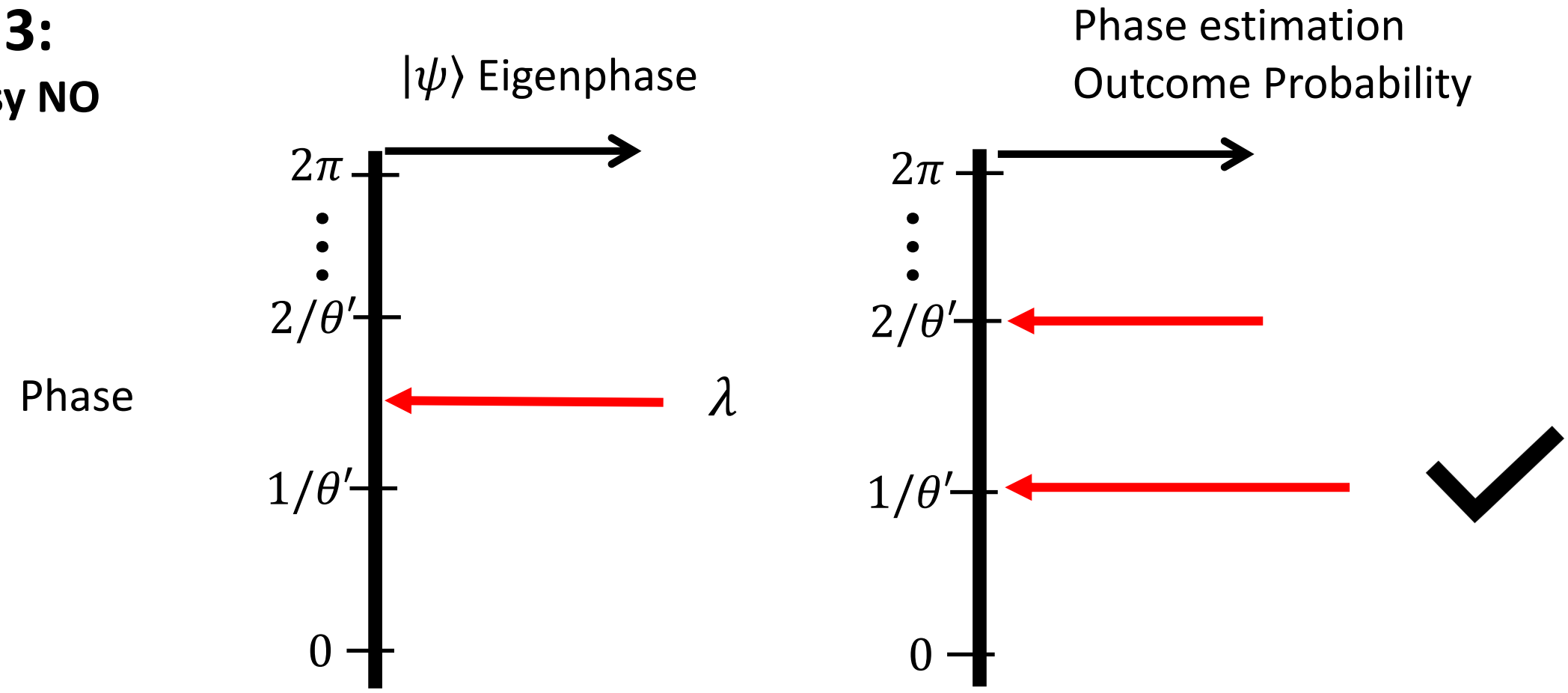
- **HARD NO**



Reduce Precision (θ)

Case 3:

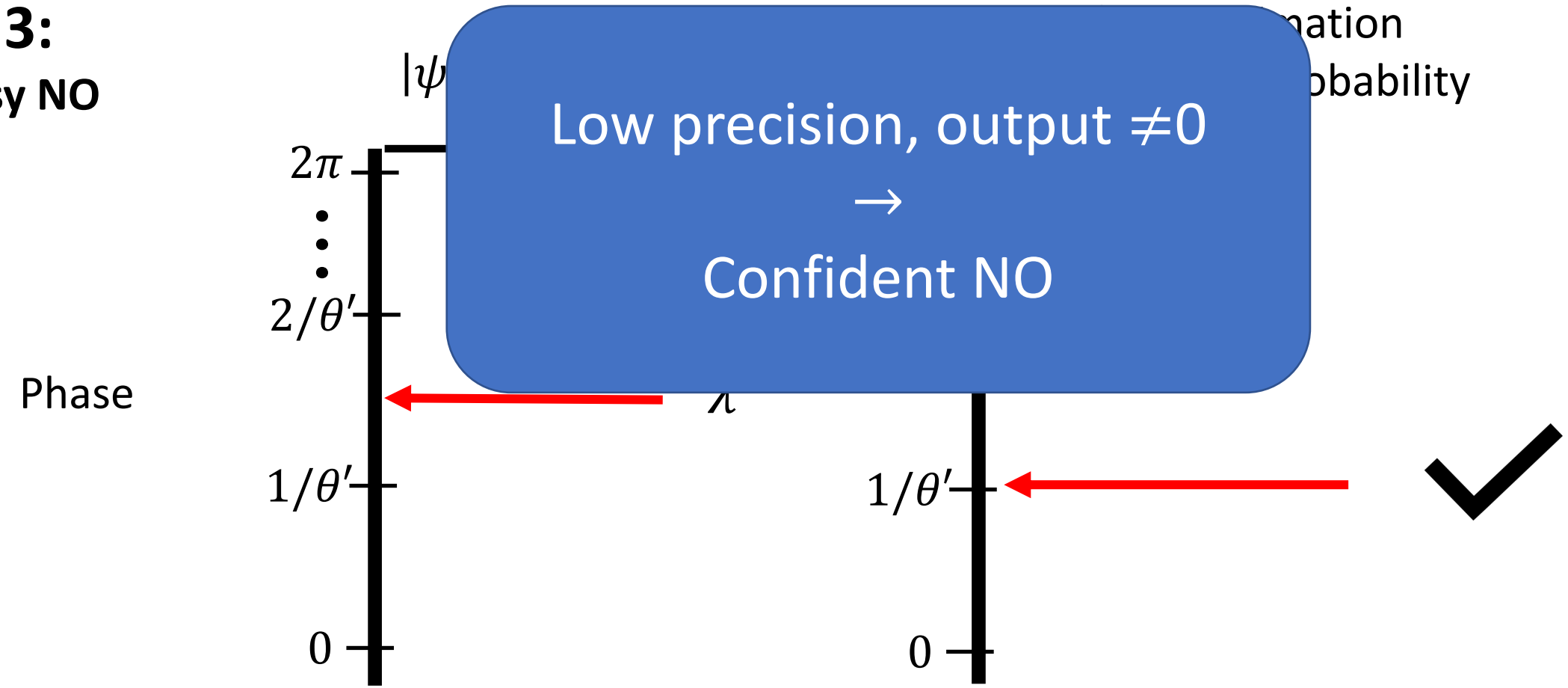
- **Easy NO**



Reduce Precision (θ)

Case 3:

- **Easy NO**



Reduce Precision (θ)

- Run span program phase estimation with exponentially increasing precision θ. Each iteration:
  - If 0 output, continue
  - If non-0 output, stop and output *NO*

Result: faster runtime for easy *NO* instances

Easy Yes Instances?

Design negated span program to exchange YES/NO instances

Result: Easy YES instance → easy NO instance

All Together

With exponentially increasing precision:

- Span program phase estimation
  - If non-0 phase, stop and output *NO*
- **Negated** span program phase estimation
  - If non-0 phase, stop and output *YES*

Result:

- faster for easy YES and NO instances
- Worst-case increases by only log factor (geometric scaling)

All Together

With exponentially increasing precision:

- Span program phase estimation
  - If non-0 phase, stop and output *NO*
- **Negated** span program phase estimation
  - If non-0 phase, stop and output *YES*

Result:

- faster for easy YES and NO instances
- Worst-case increases by only log factor (geometric scaling)

*Not exactly our algorithm :D

Performance

In span program, each instance $x \in X$ has witness size $w(x)$.

Original span program algorithm query complexity:

$$O\left(\sqrt{\left(\max_{x \in X : f(x)=YES} w(x)\right)\left(\max_{x \in X : f(x)=NO} w(x)\right)}\right)$$

Performance

In span program, each instance $x \in X$ has witness size $w(x)$.

Original span program algorithm query complexity:

$$O\left(\sqrt{\left(\max_{x\in X: f(x)=YES} w(x)\right)\left(\max_{x\in X: f(x)=NO} w(x)\right)}\right)$$

If promised that only have YES instances with $w(x) \leq w$

$$O\left(\sqrt{w\left(\max_{x\in X: f(x)=NO} w(x)\right)}\right)$$

Performance

In span program, each instance $x \in X$ has witness size $w(x)$.

Original span program algorithm query complexity:

$$O\left(\sqrt{\left(\max_{x\in X: f(x)=YES} w(x)\right)\left(\max_{x\in X: f(x)=NO} w(x)\right)}\right)$$

Our query complexity (no promise):

- If input instance x' is YES:

$$\tilde{O}\left(\sqrt{w(x')\left(\max_{x\in X: f(x)=NO} w(x)\right)}\right)$$

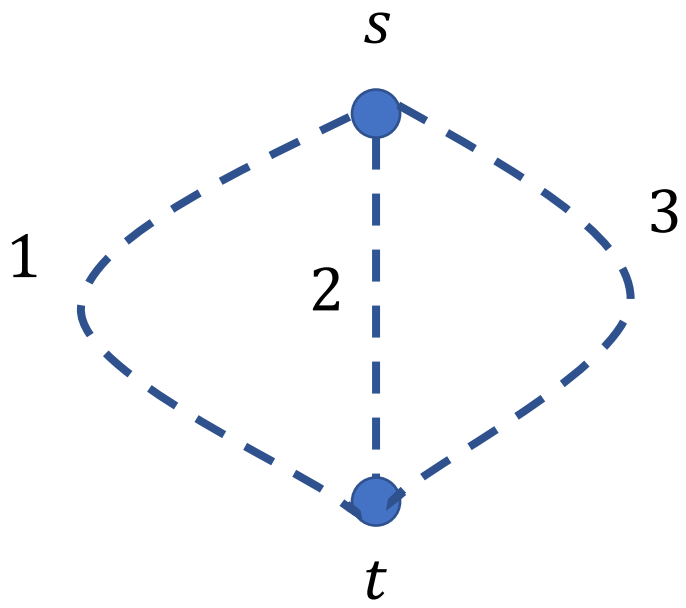
- If input instance x' is NO

$$\tilde{O}\left(\sqrt{w(x')\left(\max_{x\in X: f(x)=YES} w(x)\right)}\right)$$

Talk Outline

1. Oracle Model
2. Challenges:
   a. No check and continue ← **Easy**
   b. No (easily accessible) witness of completion ← **Harder**
3. Applications & Future Work

Path Detection



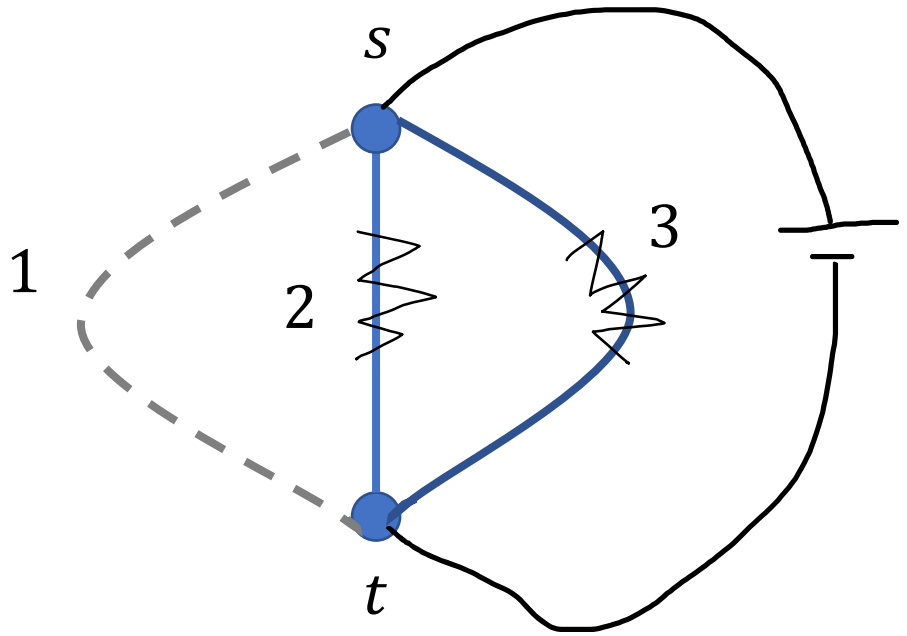
Path Detection

For each instance, calculate

- effective resistance (if path)

$x = 011$

$w(x) = 1/2$



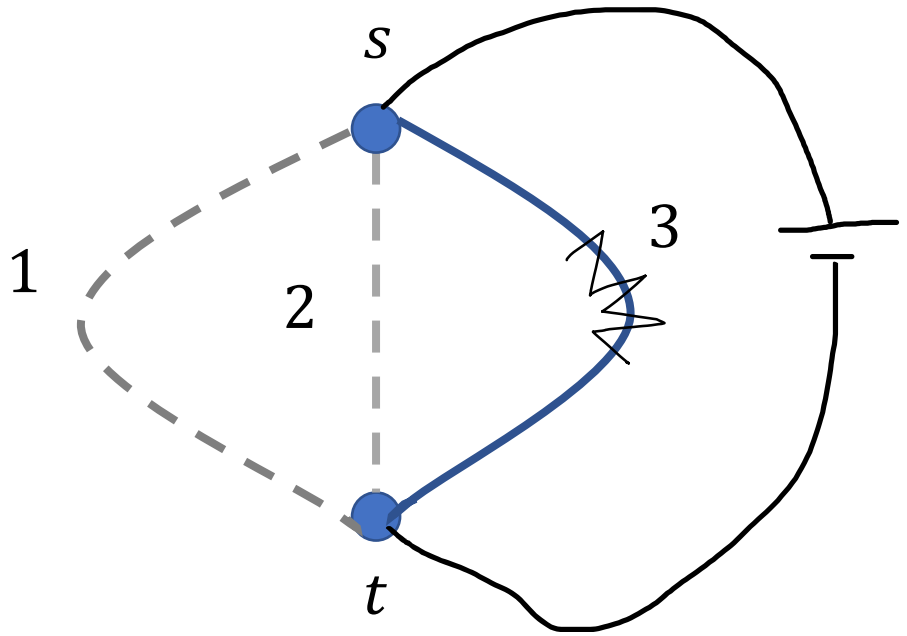
Path Detection

For each instance, calculate

- effective resistance (if path)

$x = 001$

$w(x) = 1$



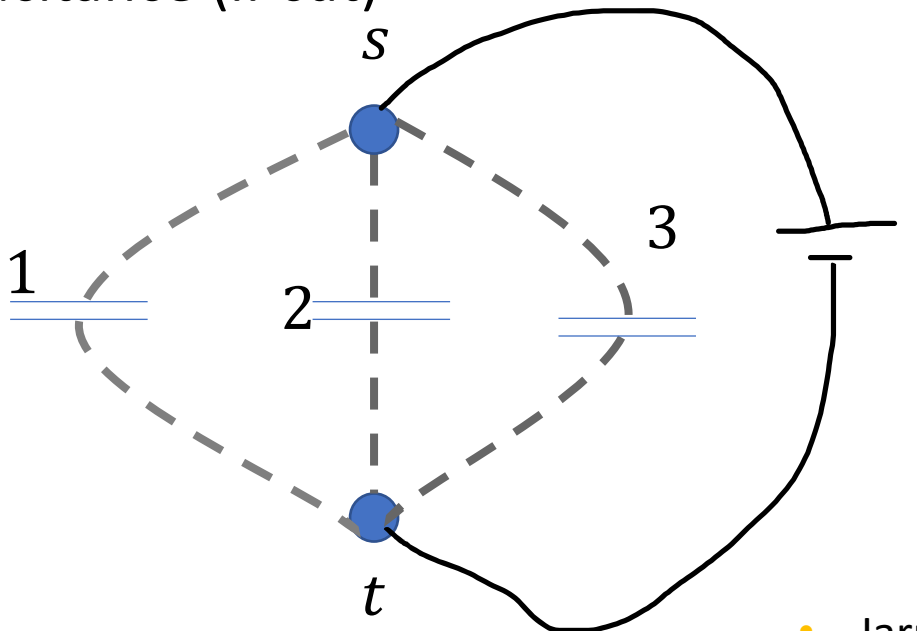
Path Detection

For each instance, calculate

- effective resistance (if path)
- effective capacitance (if cut)

$x = 000$

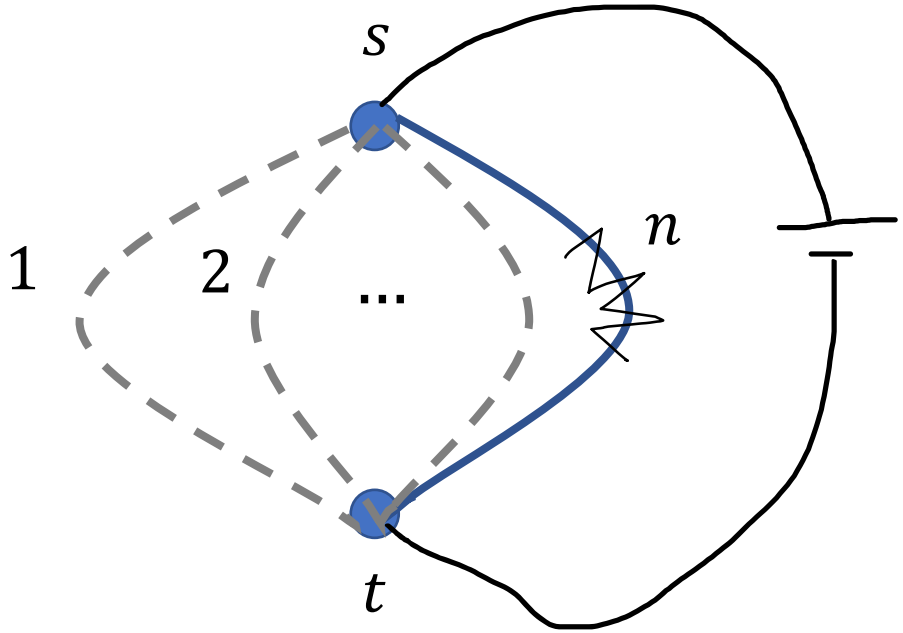
$w(x) = 3$



- Jarret, Jeffery, **SK**, Piedrafita, *(*ESA 2018*)*
- Reichard and Belovs, ESA 2012

Path Detection Applications

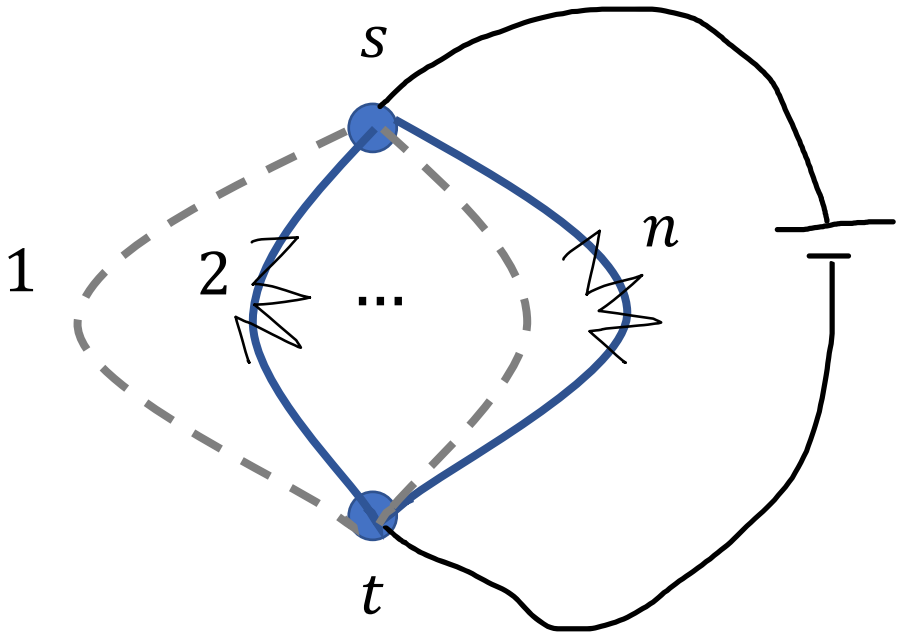
Grover's search!



Path Detection Applications

Grover's search!

m edges $\rightarrow w(x) = 1/m$
0 edges $\rightarrow w(x) = n$



If input instance x' is YES:

$$\tilde{O}\left(\sqrt{w(x')\left(\max_{x \in X : f(x)=NO} w(x)\right)}\right)$$

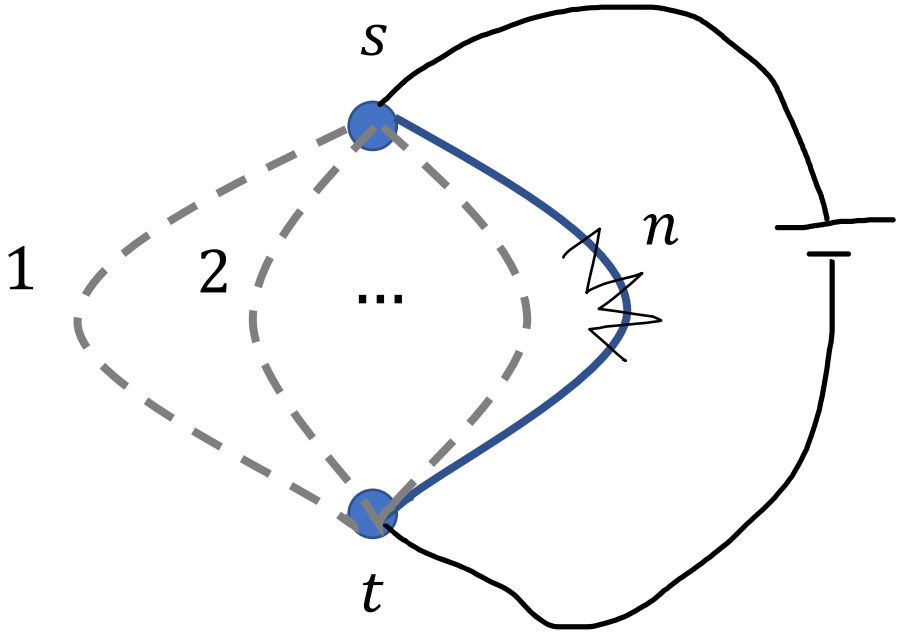
Path Detection Applications

Grover's search!

With our algorithm: if m of n items are marked, can solve in

$\tilde{O}\left(\sqrt{\frac{n}{m}}\right)$

queries without knowing m ahead of time

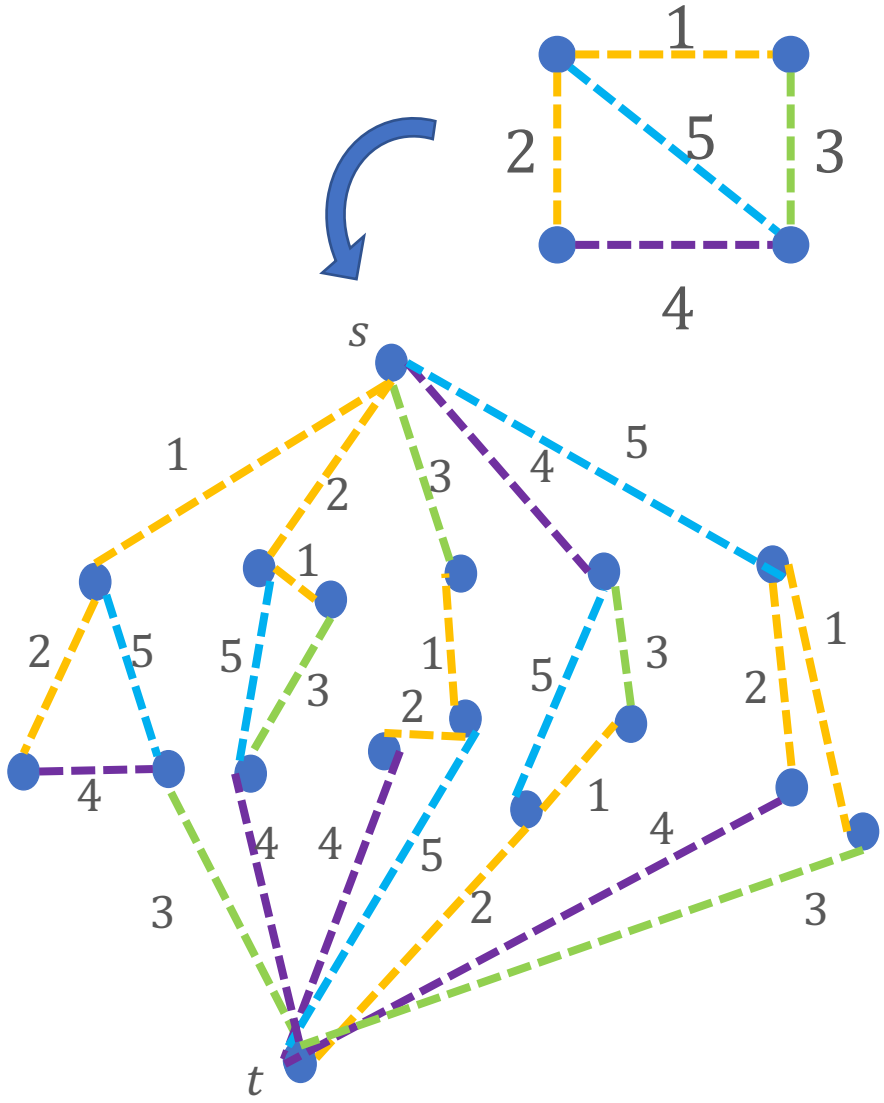


Matches: Boyer, Brassard, Hoyer, Tapp [1998] (Highly search specific)
Up to log factors: Yoder, Low, Chuang [2014] (Highly search specific)

Path Detection Applications

Cycle finding!

Can detect whether a cycle is present in $\tilde{O}\left(\frac{n^{3/2}}{\sqrt{c}}\right)$ where c is number of cycles, without knowing c ahead of time.

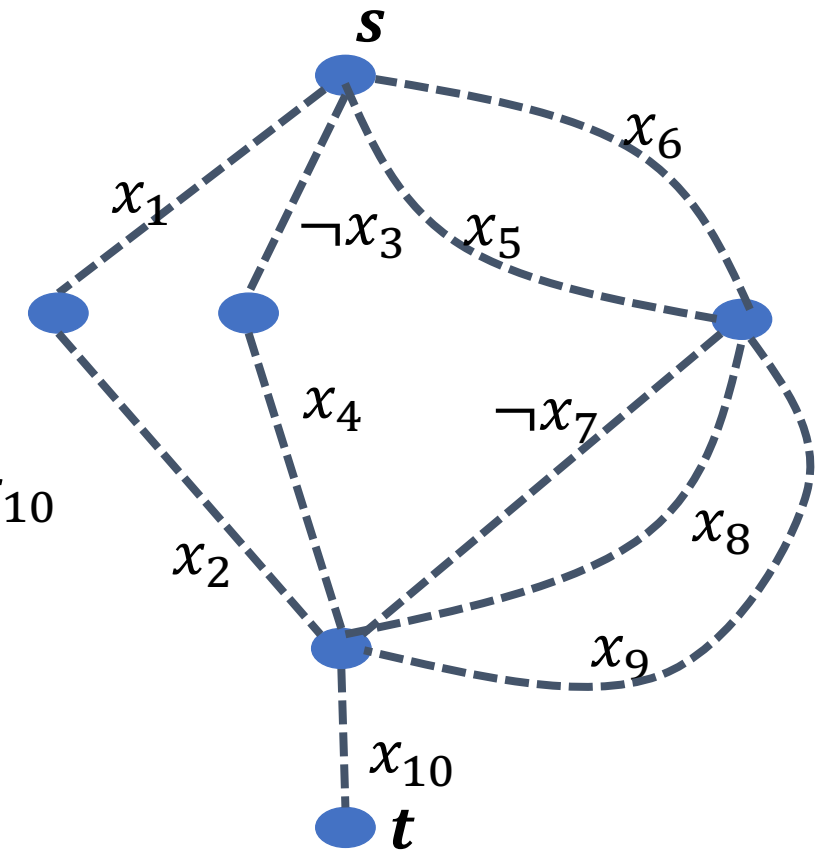


- DeLorenzo, **SK**, Witter, arXiv:1904.05995 (TQC 2019)

Path Detection Applications

Formula Evaluation

$$\Big((x_1 \wedge x_2) \vee (\neg x_3 \wedge x_4) \vee \big((x_5 \vee x_6) \wedge (\neg x_7 \vee x_8 \vee x_9)\big)\Big) \wedge x_{10}$$



Connection between easier formula instances and small effective resistance?

- Jeffery, **SK**: arXiv:1704.00765 (Quantum vol 1 p 26)

State generation extension

State Generation Problem:
Convert $|\rho_x\rangle$ to $|\sigma_x\rangle$ given access to O_x.

There is a span program-like algorithm that is nearly optimal for worst-case x.
Running faster on easier inputs?

Challenge:
Original algorithm has no measurements!

Our Result:
Use an auxiliary test to determine when can stop running.

Future/Current Work

- Opportunities for average case quantum vs. classical algorithms
  - Best classical algorithm on graphs with small effective resistance
- Get rid of log factors from error suppression? (Fixed-point methods)
- Generate witness states

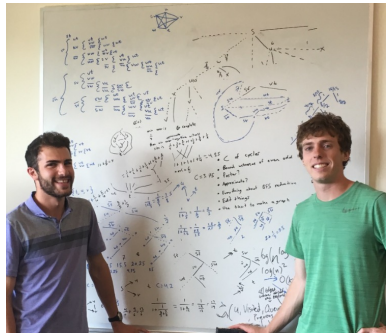
Thank you!

Middlebury College

ARO

Noel Anderson

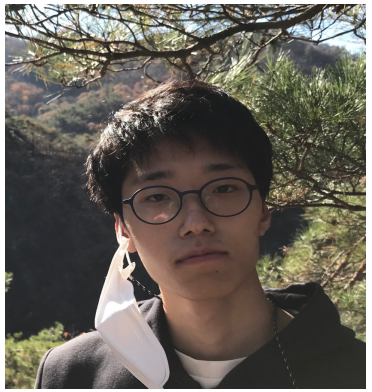
Stacey Jeffery



Teal Witter

Kai De Lorenzo

Michael Jarret



Jay-U Chung

Alvaro Piedrafita