Quantum divide and conquer

arXiv: 2210.06419, QIP 2023

Andrew M. Childs
University of Maryland

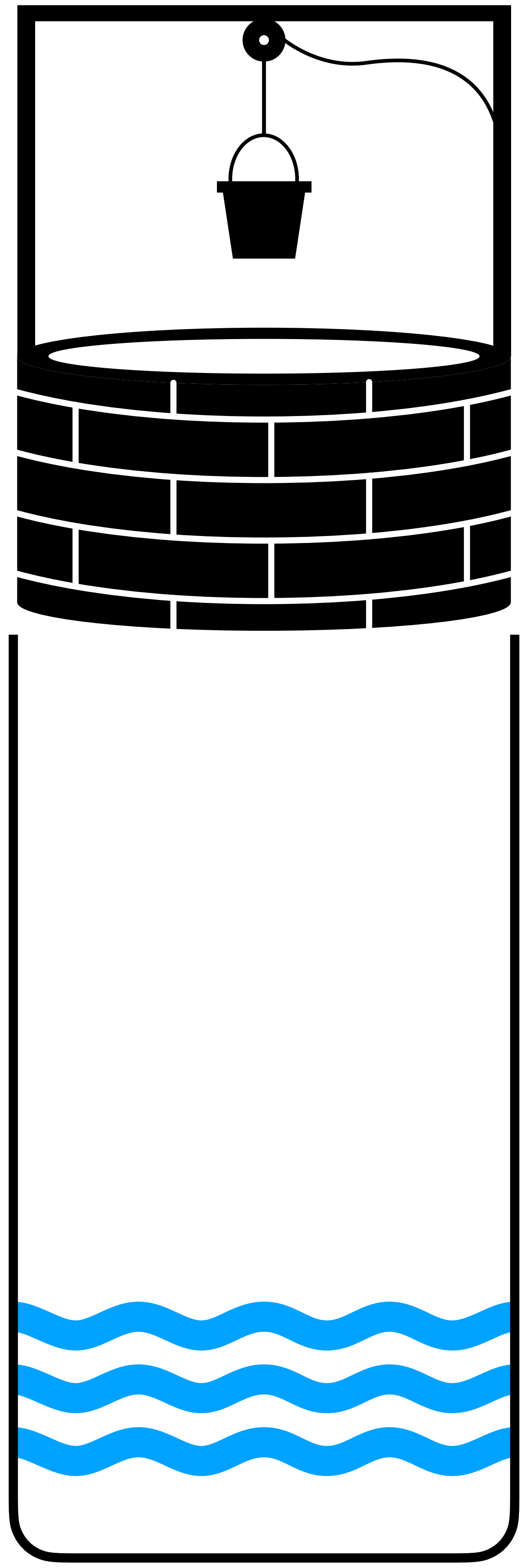
Robin Kothari
Microsoft
(→Google)

Matt Kovacs-Deak
University of Maryland

Aarthi Sundaram
Microsoft

Daochen Wang, University of British Columbia

Quantum BC Seminar: 12th March 2024



Application

Design

Structure

Quantum speedups

Examples of exponential quantum speedups

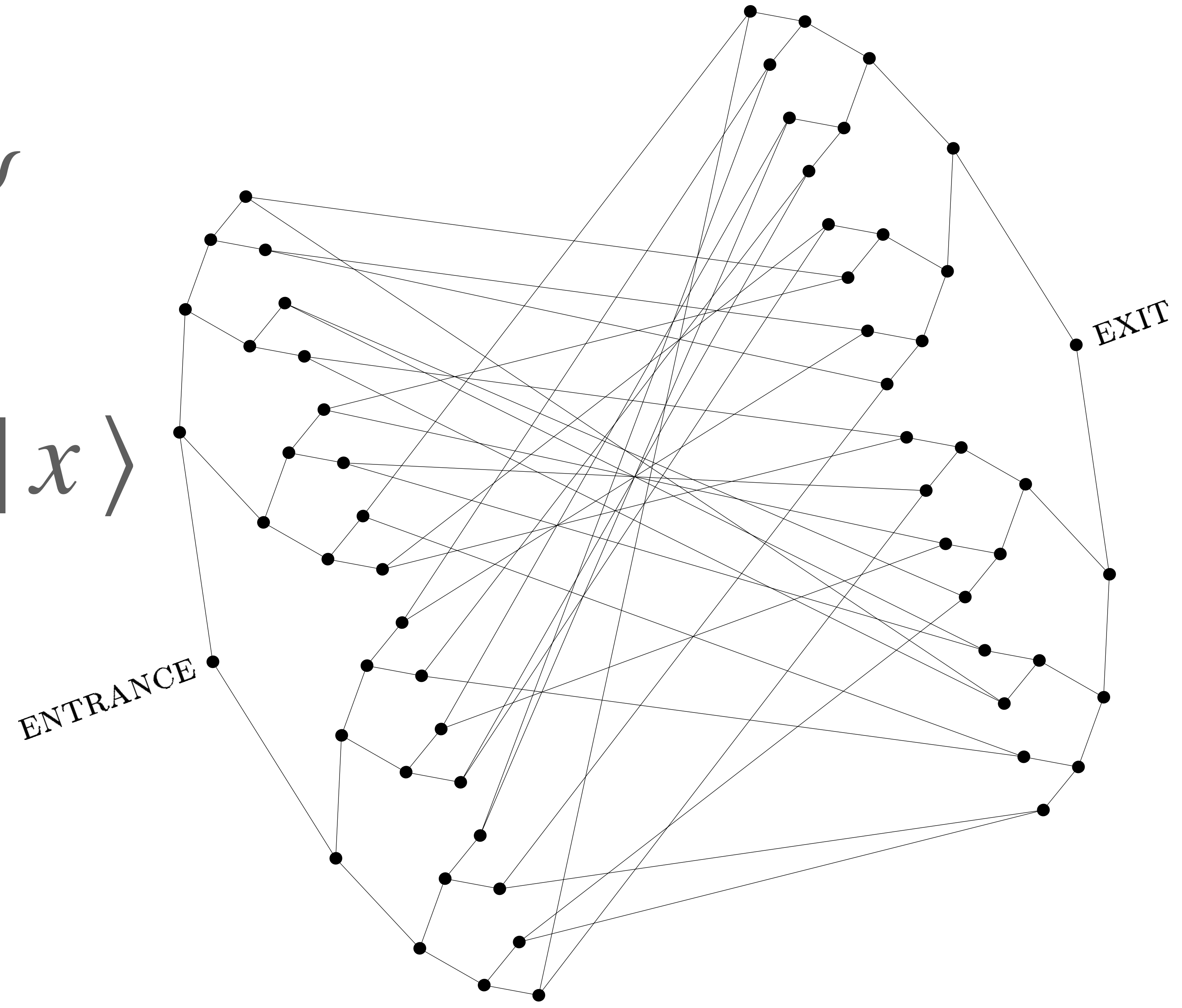
factoring $$30743126349163 = 4210601 \times 7301363$$

solving Pell's equation $x^2 - dy^2 = 1$, $d > 0$ non-square positive integer

simulating quantum dynamics $$i\hbar \frac{d}{dt}\psi = H\psi$$

solving linear systems quantumly $$Ax = b \rightarrow |x\rangle$$

EXIT-finding in glued-trees



Examples of polynomial quantum speedups

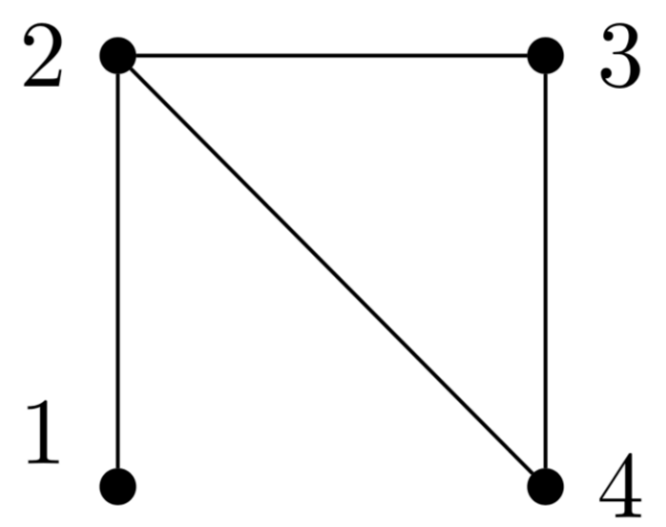
unstructured search (e.g., SAT) $(u_1 \vee \neg u_4 \vee u_3) \wedge (u_5 \vee \neg u_2 \vee \neg u_3) \wedge (u_1 \vee \neg u_2)$

maximum finding $\max_x f(x)$

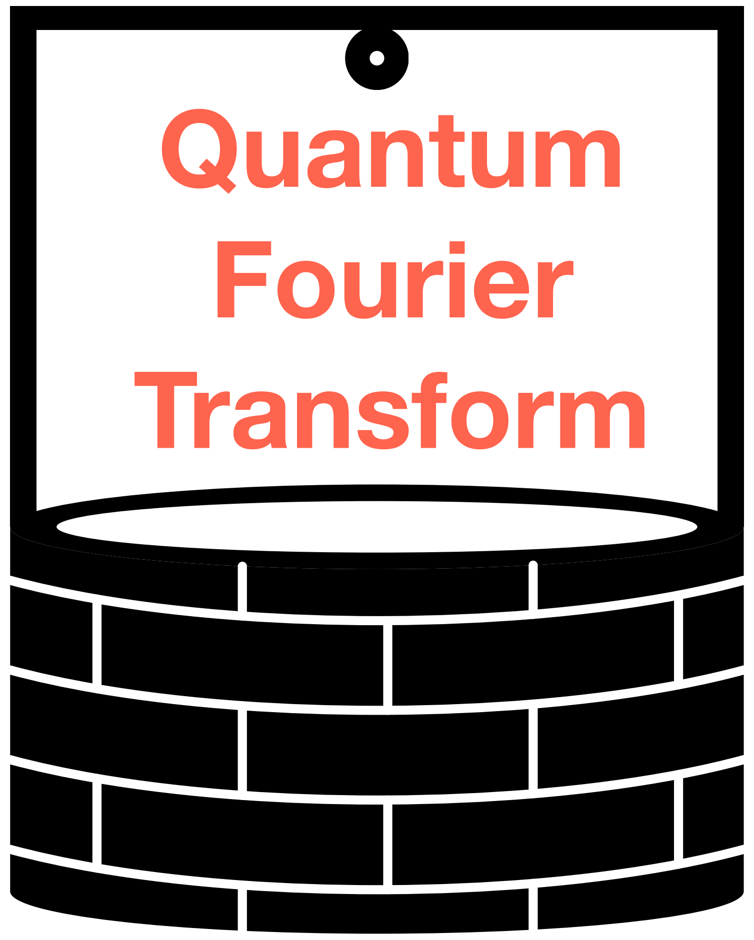
Monte-Carlo mean estimation $\mathbb{E}[f(X)] \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$

element distinctness $1\ 8\ 2\ 9\ 4\ 5\ 7\ 8\ 3\ 6 \rightarrow$ "not distinct!"

triangle detection

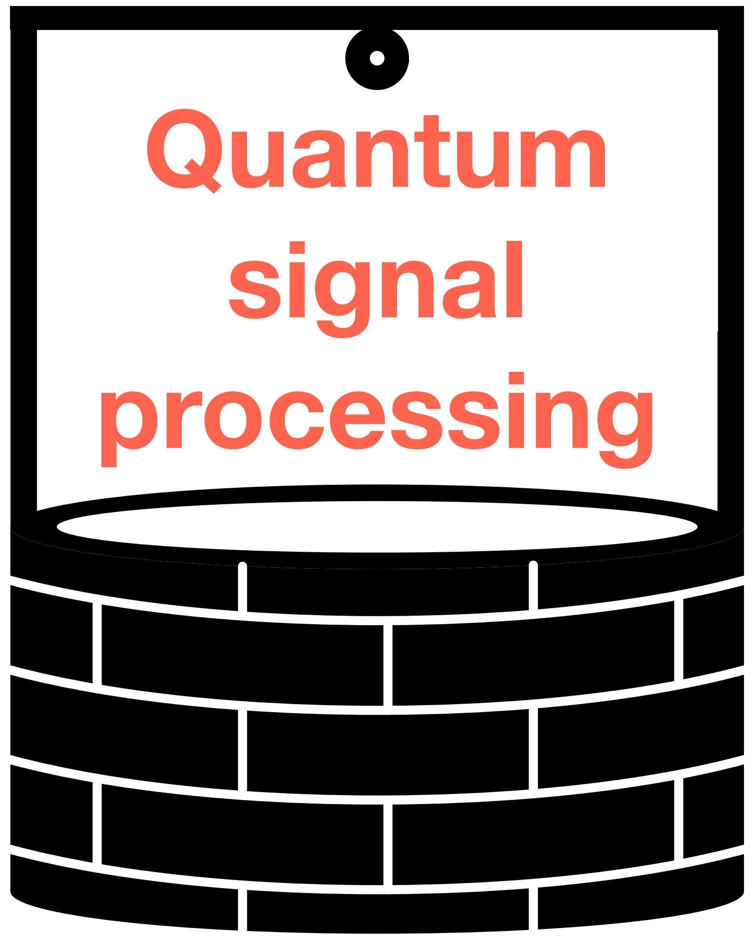


Frameworks for quantum algorithm design



factoring

solving Pell's equation



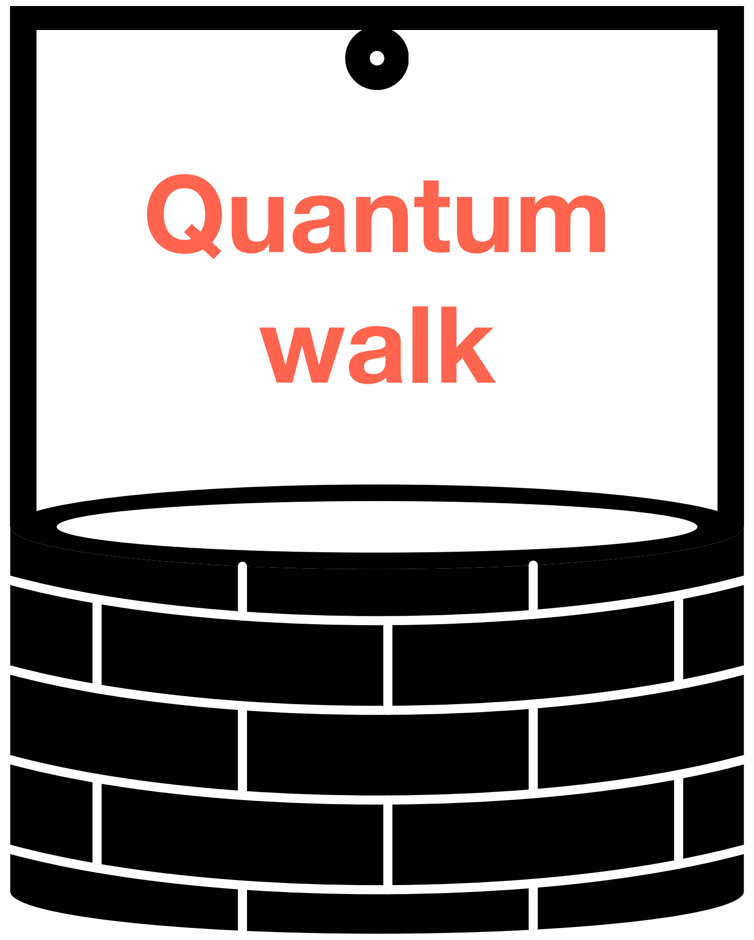
simulating quantum dynamics

solving linear systems quantumly

unstructured search

maximum finding

Monte-Carlo mean estimation



element distinctness

triangle detection

EXIT-finding in glued-trees

Divide and conquer

1. Divide a problem into subproblems

2. Recursively solve each subproblem

3. Combine the solutions of the subproblems to solve the full problem

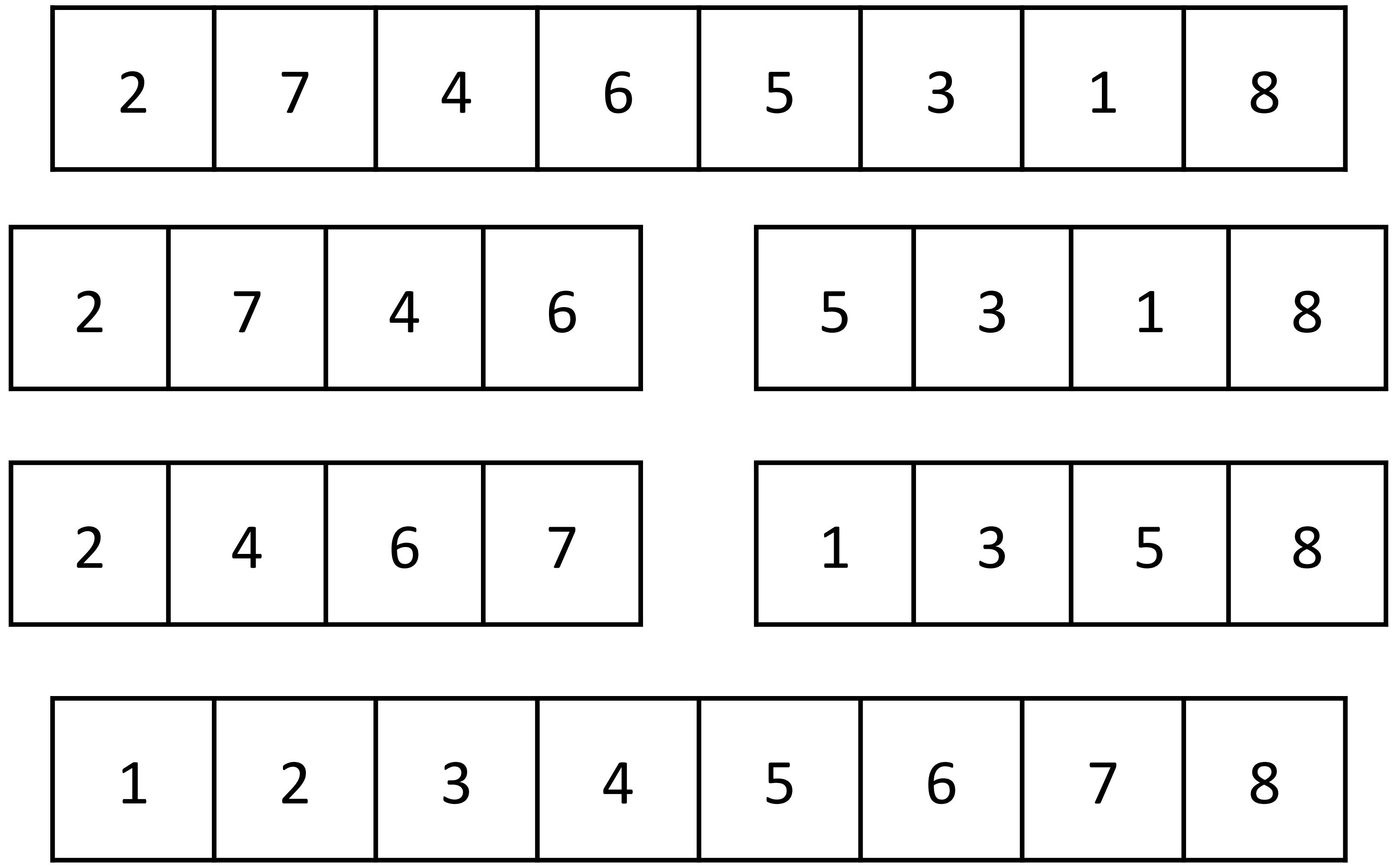
Merge sort

Recurrence:

Cost of solving auxiliary problem

$$C(n) = 2C(n/2) + O(n) \implies C(n) = O(n \log n)$$

Cost of solving subproblem



From classical to quantum divide and conquer

$x = x_1 x_2 \ldots x_n \in \{0,1\}^n$ unknown bitstring

Question: is there a bit of x that is equal to 1?
Answer denoted: $\mathrm{OR}(x)$

Divide and conquer: $\mathrm{OR}(x) = \mathrm{OR}(\mathrm{OR}(x_{\mathrm{left}}), \mathrm{OR}(x_{\mathrm{right}}))$

Classical: $C(n) \leq 2C(n/2) \to C(n) \leq n$

Quantum: $C(n) \leq \sqrt{2}C(n/2) \to C(n) \leq \sqrt{n}$?

From classical to quantum divide and conquer

Divide a problem of size n into a instances of size n/b each

- Typical classical divide-and-conquer recurrence:

$$C(n) \leq aC(n/b) + C^{\mathrm{aux}}(n)$$

- Corresponding quantum divide-and-conquer recurrence:

$$C(n) \leq \sqrt{a}C(n/b) + C_Q^{\mathrm{aux}}(n)$$

Query complexity

Let $f\colon \Sigma^n \to \{0,1\}$, suppose an algorithm $\mathscr{A}$ computes $f(x)$ correctly with probability $\geq 2/3$ for all $x \in \Sigma^n$

How many queries to (the oracle encoding) input x does $\mathscr{A}$ need to make?

Answer denoted $D(f)$, $R(f)$, and $Q(f)$, when $\mathscr{A}$ is deterministic, randomized, and quantum, respectively

Quantum speedup $\iff$ $Q(f) < R(f)$

Classical query

$i \mapsto x_i$

Quantum query

$|i\rangle|a\rangle \mapsto |i\rangle|a+x_i\rangle$

Example: $\mathrm{OR}_n\colon \{0,1\}^n \to \{0,1\}$; $R(\mathrm{OR}_n) = \Theta(n)$ and $Q(\mathrm{OR}_n) = \Theta\left(\sqrt{n}\right)$

Adversary quantity

Every $f\colon \Sigma^n \to \{0,1\}$ is associated with an adversary quantity

$$\mathrm{Adv}(f) = \max_{\Gamma} \frac{\|\Gamma\|}{\max_{i\in[n]} \|\Gamma_i\|},$$

max over $|\Sigma|^n \times |\Sigma|^n$ real symmetric matrices Γ with $f(x) = f(y) \implies \Gamma_{xy} = 0$ and

$$(\Gamma_i)_{xy} = \begin{cases} \Gamma_{xy} & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \end{cases}$$

Adversary quantity

Theorem [Høyer, Lee, Špalek 07; Lee, Mittal, Reichardt, Špalek 10]

$$Q(f) = \Theta(\mathrm{Adv}(f))$$

Composition theorems

AND: if $g(x, y) = f_1(x) \wedge f_2(y)$, then $\mathrm{Adv}(g)^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2)^2$ [LMRŠ 10]

OR: if $g(x, y) = f_1(x) \vee f_2(y)$, then $\mathrm{Adv}(g)^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2)^2$ [LMRŠ 10]

SWITCH-CASE: if $h(x) = g_{f(x)}(x)$, then $\mathrm{Adv}(h) \leq 2\mathrm{Adv}(f) + \max_s \mathrm{Adv}(g_s)$ [our work]

Quantum divide and conquer

AND-OR: suppose f is computed as $f_1 \square f_2 \square \ldots \square f_a \square f_{\text{aux}}$, $\square \in \{\wedge, \vee\}$

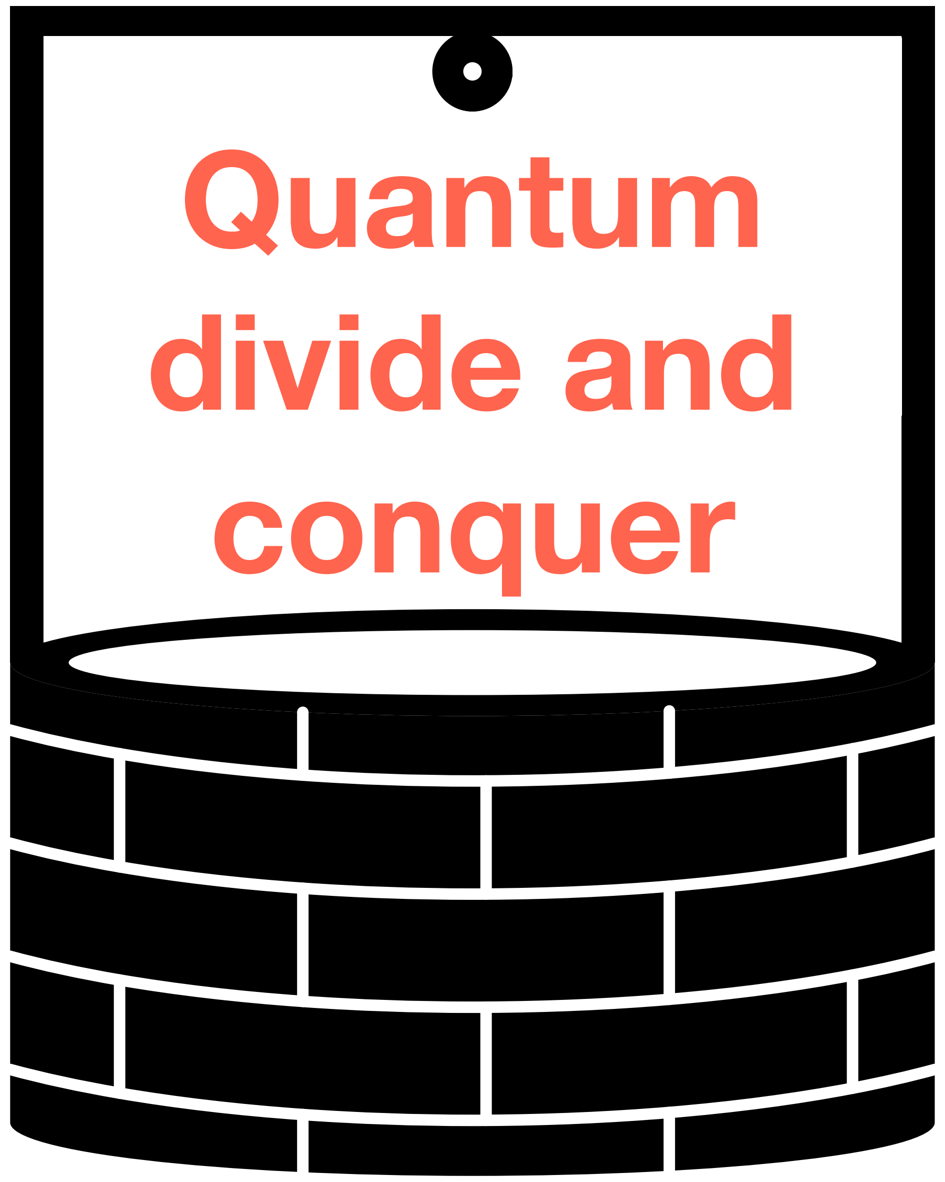
$$\mathrm{Adv}(f)^2 \leq O\big(Q(f_{\text{aux}})^2\big) + \sum_{i=1}^{a} \mathrm{Adv}(f_i)^2$$

SWITCH-CASE: Suppose f is computed by first computing $s = f_{\text{aux}}(x)$ and then some function $g_s(x)$, then

$$\mathrm{Adv}(f) \leq O\big(Q(f_{\text{aux}})\big) + \max_s \mathrm{Adv}(g_s)$$

$\rightarrow$ **Divide and conquer recurrences in the quantum setting**

Applications



- Recognizing regular languages [Aaronson, Grier, Schaeffer 19]
- String rotation [Akmal, Jin 22]
- Longest increasing subsequence New!
- Longest common subsequence New!

Recognizing regular languages

Let $\Sigma = \{0,1,2\}, f_n : \Sigma^n \to \{0,1\}$ such that $f_n(x) = 1$ iff $x \in \Sigma^*20^*2\Sigma^*$

02002110 ✅ 02102112 ⛔

Observation. Let $g_n(x) = \left(x_{\text{left}} \in \Sigma^*20^*\right) \wedge \left(x_{\text{right}} \in 0^*2\Sigma^*\right)$, then

$$f_n(x) = f_{n/2}(x_{\text{left}}) \vee f_{n/2}(x_{\text{right}}) \vee g_n(x)$$

Let $a(n) = \mathrm{Adv}(f_n)$, then $a(n)^2 \leq 2a^2(n/2) + O\left(Q(g_n)^2\right)$

But $Q(g_n) = O\left(\sqrt{n}\right)$, so $a(n) = O\left(\sqrt{n \log n}\right)$

Longest common subsequence

k-common subsequence (k-CS). Given $x, y \in \Sigma^n$, do x and y share a subsequence of length k?

E	i	n	s	t	e	i	n	$k \leq 4$ ✅
E	n	t	w	i	n	e	d	$k > 4$ ⛔

- $R(k\text{-CS}) = \Theta(n)$ for $k \geq 1$
- $Q(1\text{-CS}) = \Theta\left(n^{2/3}\right)$ ← bipartite element distinctness [Aaronson, Shi 04; Ambainis 03]
- $Q(k\text{-CS}) = O\left(n^{2k/(2k+1)}\right)$ ← using [Ambainis 03]

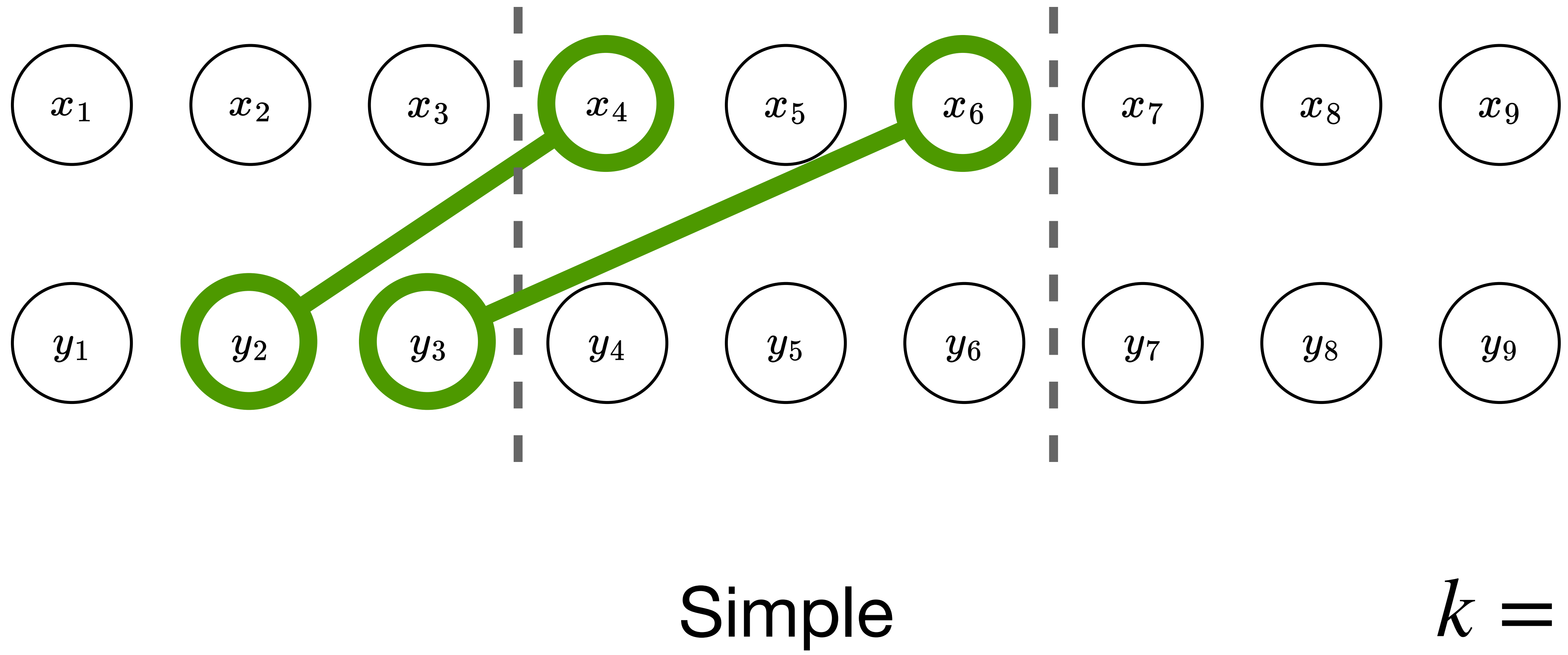
Can we do better?

Simple and composite k-CS

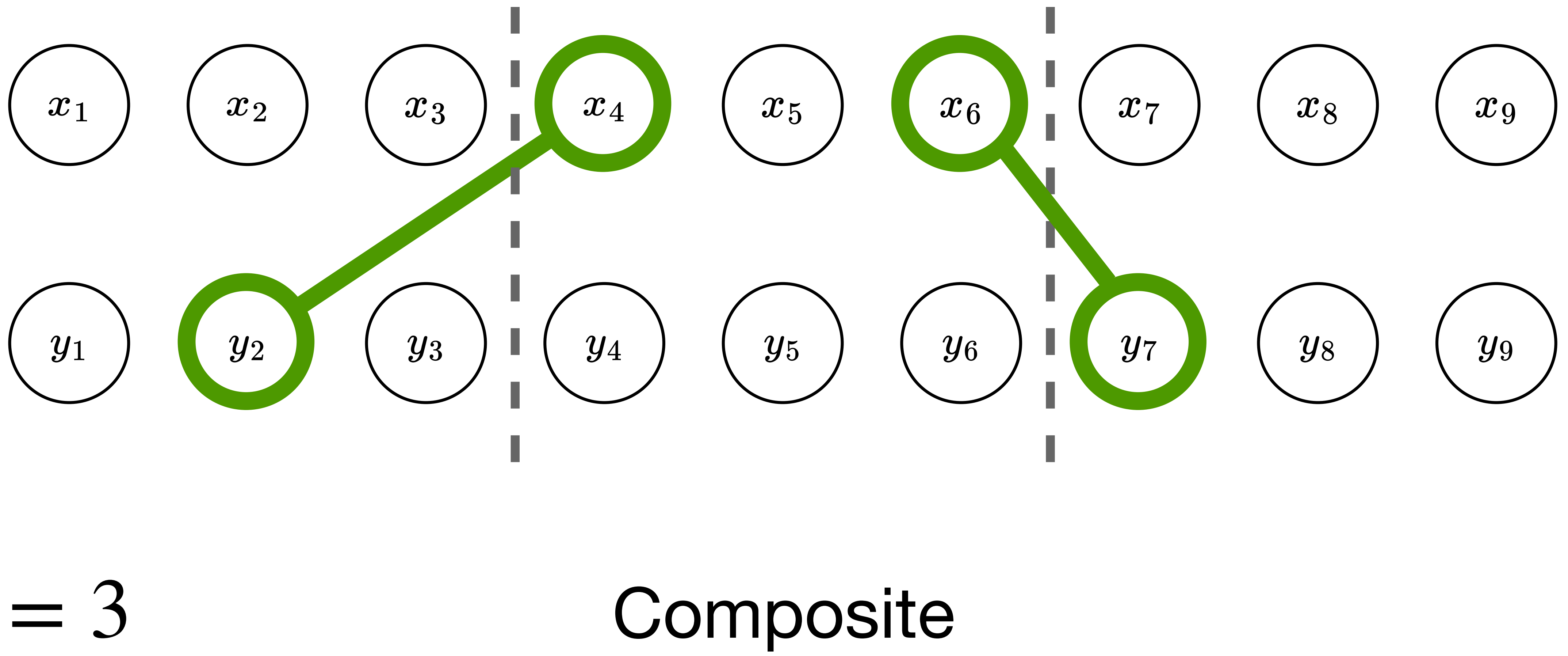
Theorem. Let $a_k(n)$ = adversary quantity for k-CS on input length n. Then $a_k(n) = O\left(n^{2/3}\log^{k-1} n\right)$

Divide the two input strings x and y into m parts each. Then, a k-CS can either be **simple** or **composite**

- A simple k-CS is a k-CS formed by symbols within a *single* part of x and a *single* part of y
- A composite k-CS is any k-CS that is not simple

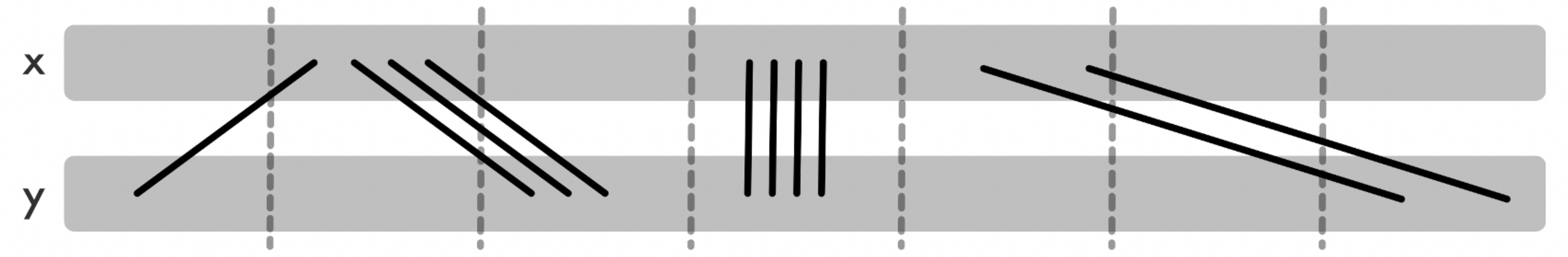


$k = 2,\ m = 3$



Detecting composite k-CS

$k = 10,\ m = 7$

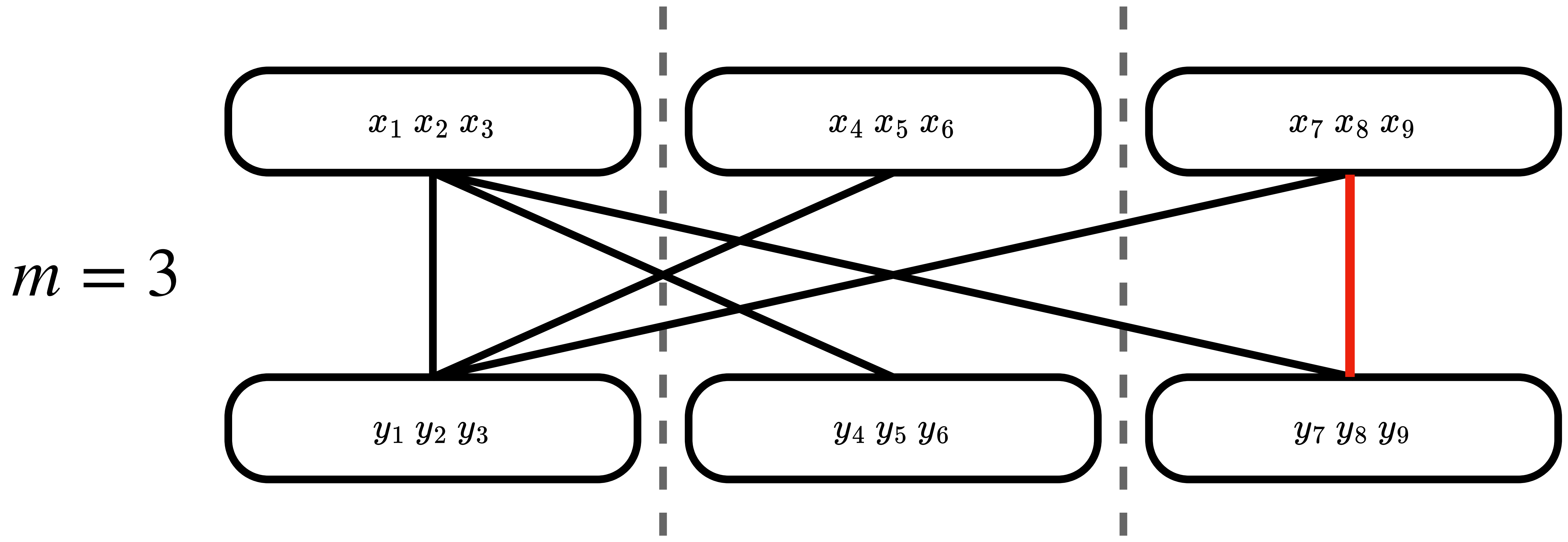


Only a constant number of possible configurations!

Cost: $O\left(\sum_{j=1}^{k-1} a_j(n) \log n\right)$

Detecting simple k-CS

Only need to detect if there exists a simple k-CS between $\leq 2m-1$ pairs of length-(n/m) substrings!



Line between parts = parts share common symbol

Cost of computing lines = $m^2 \cdot O\left(n^{2/3}\right) = O(n^{2/3})$

Cost: $O(n^{2/3}) + \sqrt{2m-1}\, a_k(n/m)$

Putting it together

Theorem. Let $a_k(n)$ = adversary quantity for k-CS on input length n. Then $a_k(n) = O\left(n^{2/3} \log^{k-1} n\right)$

Proof.

- Detecting composite k-CS costs: $O\left(\sum_{j=1}^{k-1} a_j(n) \log(n) \right)$

- Detecting simple k-CS costs: $O(n^{2/3}) + \sqrt{2m-1} \cdot a_k(n/m)$

Induction on $k \implies a_k(n) \leq \sqrt{2m-1}\, a_k(n/m) + O(n^{2/3} \log^{k-1} n)$

Solves to $a_k(n) = O\left(n^{2/3} \log^{k-1} n\right)$, provided $\log_m\left(\sqrt{2m-1}\right) < 2/3 \iff m \geq 7$

Conclusion

Framework for designing quantum query algorithms using divide-and-conquer intuition

Applications:

- Simpler analysis for recognizing regular languages and string rotation with tighter bounds
- Quantum algorithm for k-IS using $\tilde{O}\left(\sqrt{n}\right)$ queries
- Quantum algorithm for k-CS using $\tilde{O}\left(n^{2/3}\right)$ queries

Open questions

- Can we find **other applications** of quantum divide and conquer using combining functions other than AND-OR and SWITCH-CASE?
- Can we obtain **super-quadratic speedups** using quantum divide and conquer?
- What about **time complexity**? Follow-up works by [Allcock, Bao, Belovs, Lee, Santha 23] and [Jeffery, Pass 24] partly resolve this