

ABSTRACT

Title of Dissertation: QUANTUM SPEEDUPS:
STRUCTURE, DESIGN, AND APPLICATION

Daochen Wang
Doctor of Philosophy, 2023

Dissertation Directed by: Professor Andrew M. Childs
Department of Computer Science

Adjunct Associate Professor Carl A. Miller
Institute for Advanced Computer Studies

A quantum speedup occurs when a computational problem can be solved faster on a quantum computer than on a classical computer. This thesis studies the circumstances under which quantum speedups can arise from three perspectives.

First, we study the structure of the problem. We characterize how a problem's symmetries decide whether it admits a superpolynomial or polynomial quantum speedup. In particular, we show that the computation of any partial function of a hypergraph's adjacency matrix admits at most a polynomial speedup. Such functions are only weakly symmetric and subsequent work shows that if the symmetry is weakened even further, then a superpolynomial speedup can be possible. To complete the picture for graph problems, we also construct a partial function of a graph's adjacency *list* that admits an exponential speedup.

Second, we study classical paradigms for solving problems. We design a natural quantum query analogue of the divide and conquer paradigm, which we call the “quantum divide and conquer” framework. This framework allows us to quantumly speed up the solution of many problems that are classically solved using divide and conquer.

The main technical novelty of our framework is that it naturally combines quantum adversary and quantum algorithmic techniques. To showcase its usefulness, we apply it to obtain near-optimal quantum query complexities for four string problems: (i) recognizing regular languages; (ii) decision versions of String Rotation and String Suffix; and natural parameterized versions of (iii) Longest Increasing Subsequence and (iv) Longest Common Subsequence.

Third, we study ways of generalizing a problem known to admit a quantum speedup. We generalize the search problem of finding a marked item in a database to the case where the items are not either marked or unmarked, but are “partially marked”. The goal is to find the item that is the most heavily marked. We construct a quantum algorithm for this problem within the variable-time framework and prove its near-optimality by modifying the standard adversary method to work for oracles encoding probability distributions. Coincidentally, the problem studied is equivalent to the multi-armed bandits problem in reinforcement learning, which has many applications.

QUANTUM SPEEDUPS:
STRUCTURE, DESIGN, AND APPLICATION

by

Daochen Wang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:

Professor Alexander Barg, Dean's Representative

Professor Andrew M. Childs, Chair/Advisor

Adjunct Associate Professor Yi-Kai Liu

Adjunct Associate Professor Carl A. Miller, Advisor

Assistant Professor Xiaodi Wu

© Copyright by
Daochen Wang
2023

Acknowledgements

First and foremost, I thank my PhD advisors Andrew Childs and Carl Miller. Apart from serving as academic role models, they taught me the importance of asking the right questions, patiently supported my pursuit of high-risk but high-reward problems, and offered essential advice on my career development. I also thank Andrew for letting me be his TA for his advanced course on quantum algorithms. That experience increased my appreciation for the beauty of quantum algorithms and gave me a solid foundation for doing quantum algorithms research. I also thank Carl for being my guide to quantum cryptography and for selflessly providing so much detailed feedback on my work over the years. In particular, I thank Carl for helping me improve the mathematical rigour of this thesis.

The next person to thank would be Robin Kothari. I first met Robin at TQC 2019 and since then he has almost served as my third PhD advisor. I have lost count of the number of times that Robin amazed me by solving complex problems on the spot during our many research meetings. Alongside Robin, I thank Steve Brierley and Aarthi Sundaram for being exceptionally supportive mentors. I also thank the members of my thesis advisory committee who I have not already mentioned: Sasha Barg, Yi-Kai Liu, and Xiaodi Wu. In particular, I thank Sasha for his classes on discrete probability theory and error-correcting codes, Yi-Kai for his advice on how to approach lattice problems, and Xiaodi for sharing his unique perspective on quantum computing research. A less obvious person to thank who greatly impacted this thesis

is Ronald de Wolf. Ronald generously mailed his PhD thesis to me in book form at the start of my PhD. That book, by now well worn, probably explains why most of this thesis revolves around quantum query complexity.

For their profound insights and camaraderie, I thank my coauthors (not already mentioned) on papers that form this thesis: Shalev Ben-David, András Gilyén, Matt Kovacs-Deak, William Kretschmer, Tongyang Li, Supartha Podder, and Xuchen You. Likewise, I thank my coauthors (not already mentioned) on papers completed during my PhD that are not included in this thesis: Yusuf Alnawakhtha, Dong An, Earl Campbell, Ophelia Crawford, Honghao Fu, Oscar Higgott, Ashish Kapoor, Jin-Peng Liu, Atul Mantri, Thomas Parks, Martin Roetteler, Barnaby van Straaten, and Qi Zhao.

I'm fortunate to have been a member of the Joint Center for Quantum Information and Computer Science (QuICS). I thank everyone at QuICS for contributing to our excellent research environment. In particular, I thank QuICS members who participated in the reading groups on quantum information and lattice cryptography that I organised. Outside the academic, I thank Jessica Sadler and Andrea Svejda for their help with many administrative matters.

Finally, I thank my family for their unconditional support and encouragement.

Table of contents

Acknowledgements	ii
Table of contents	iv
Introduction	1
Quantum speedups	1
Overview	4
Preliminaries	8
1 Symmetries, graph properties, and quantum speedups	13
1.1 Introduction	14
1.2 Preliminaries	16
1.3 No exponential speedup in the adjacency matrix model	20
1.4 Example of exponential speedup in the adjacency list model	28
1.5 Discussion and open problems	45
2 Quantum divide and conquer	49
2.1 Introduction	50
2.2 Preliminaries	58
2.3 Framework	59
2.4 Applications	64
2.4.1 Recognizing regular languages	64
2.4.2 String Rotation and String Suffix	66
2.4.3 k -Increasing Subsequence	71
2.4.4 k -Common Subsequence	73
2.5 Discussion and open problems	82
3 Quantum exploration algorithms for multi-armed bandits	84
3.1 Introduction	84
3.2 Preliminaries	90
3.3 Quantum exploration algorithm	93
3.3.1 Variable-time quantum algorithm	94
3.3.2 Applying VTAA and VTAE to \mathcal{A}	101
3.3.3 Quantum algorithm for best-arm identification	103
3.3.4 Corollaries for the PAC and fixed-budget settings	109

3.4	Quantum lower bound	110
3.5	Discussion and open problems	117
Conclusion		120
Appendix		123
1	Adversary composition lemmas	123
1.1	Proof of Lemma 8	125
1.2	Proof of Lemma 9	129
2	Randomized search	130
Bibliography		134

Introduction

Quantum speedups

The promise of quantum computing lies in the existence of quantum speedups. A quantum speedup occurs when a computational problem can be solved faster on a quantum computer than on a classical computer. This thesis investigates the nature of quantum speedups.

A common misconception is that quantum computers can speed up the solution of any problem by simply trying all candidate solutions at the same time. This would mean that quantum computing can efficiently solve all problems in NP, which is contradicted by all the evidence since 1997 [BBBV97]. It is true that a quantum state on n quantum bits (or qubits) is mathematically described by a vector $u \in \mathbb{C}^{2^n}$ with $\|u\|_2 = 1$, which has a size that is exponential in n . However, the problem is that the information in u can only be obtained by making a measurement, which destroys most of the information in u . We can go further and say that the exponential size of u alone has no bearing on the question of quantum speedup. This is because a classical (randomized) computation manipulates a probability distribution on the set of n -bit strings, which is also mathematically described by an exponential-sized vector $v \in \mathbb{C}^{2^n}$, albeit with $\|v\|_1 = 1$ and non-negative real entries.

The main difference between classical and quantum computation lies in how u and v are manipulated. Classically, u is manipulated by a sequence of elementary

stochastic matrices chosen from a fixed universal set \mathcal{S} . Quantumly, v is manipulated by a sequence of elementary unitary matrices chosen from a fixed universal set \mathcal{U} . For example, one valid choice of \mathcal{S} consists of the coin-tossing matrix $C := \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$ and the (matrix representation of) the Toffoli gate.¹ One valid choice of \mathcal{U} consists of the Hadamard matrix $H := \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$ and the Toffoli gate [Shi03]. Framed this way, we see that quantum and classical computation differ essentially in the single minus sign in H . Since we can mimic the effect of C by H , it follows that efficient quantum computation subsumes efficient classical computation. On the other hand, it is not obvious at all why the presence of the minus sign in H should make quantum computation strictly *more* powerful, that is, yield quantum speedups.

The first rigorous evidence that quantum speedups exist came from the query (or black-box) setting [Deu85; DJ92; Sim94; Gro96; BV97; Chi+03]. In this setting, we consider a function $f: E \subseteq \Sigma^n \rightarrow \{0, 1\}$ and ask: what is the minimum number of queries to a black-box encoding the input $x \in E$ that suffices to compute $f(x)$ in the worst case? This question can be asked in three models of query computation: deterministic, randomized, and quantum. The answer is then known as the deterministic, randomized, and quantum query complexity of f , and is denoted $D(f)$, $R(f)$, and $Q(f)$, respectively. In the query model, a quantum speedup refers to when $Q(f)$ is strictly smaller than $R(f)$. Establishing a quantum speedup therefore entails upper bounding $Q(f)$ *and* lower bounding $R(f)$. One advantage of working in the query setting is that we have mathematical techniques to prove lower bounds on $D(f)$, $R(f)$, and $Q(f)$. In contrast, it is notoriously difficult to prove lower bounds on the total time it takes to solve a problem in the standard (non-query) model of computation, where total time refers to the total number of elementary stochastic or unitary matrices used.

Two important examples of f that admit a quantum speedup are OR and Simon's

¹Technically, these matrices need to be tensored with 2×2 identity matrices to act on \mathbb{C}^{2^n} .

function, defined below.

1. OR: $\{0, 1\}^n \rightarrow \{0, 1\}$ is defined by $\text{OR}(x) = 1$ if and only if the $x \neq 0^n$. This function has $R(\text{OR}) = \Omega(n)$ but Grover constructed a search algorithm² in [Gro96] that shows $Q(\text{OR}) = O(\sqrt{n})$. Since n is a polynomial function of \sqrt{n} , this is referred to as a polynomial quantum speedup.
2. Simon’s function f_{Simon} is defined when $n = 2^k$, $\Sigma = \{1, \dots, n\}$, and E consists of strings $x \in \Sigma^n$ that are either (i) a permutation of $(1, 2, \dots, n)$, or (ii) there exists some $0^k \neq a \in \{0, 1\}^k$ such that $x_{\text{str}(i)} = x_{\text{str}(i) \oplus a}$ for all $i \in \Sigma$ and $x_{\text{str}(i)} \neq x_{\text{str}(j)}$ for all $i, j \in \Sigma$ with $\text{str}(j) \notin \{\text{str}(i), \text{str}(i) \oplus a\}$, where $\text{str}: \Sigma \rightarrow \{0, 1\}^k$ maps i to the binary representation of $i - 1$ and \oplus denotes bitwise XOR. This function has $R(f_{\text{Simon}}) = \Omega(\sqrt{n})$ but Simon constructed an algorithm in [Sim94] that shows $Q(f_{\text{Simon}}) = O(\log(n))$. Since \sqrt{n} is a super-polynomial (in fact, exponential) function of $\log(n)$, this is referred to as a super-polynomial quantum speedup.

The query setting is interesting because a quantum speedup in this setting can sometimes be translated into an apparent³ quantum speedup in the standard model of computation. The main challenge is to instantiate the black-box encoding the input. Famously, Shor [Sho97] instantiated the black-box encoding the input to (a generalized version of) Simon’s function to obtain his algorithm that factors an n -digit integer in polynomial-time, i.e., time that is upper bounded by a polynomial function of n . As of early 2023, even the best-known *heuristic* classical algorithm for factoring (the general number field sieve [BLP93]) takes time $2^{\Omega(n^{1/3})}$, which cannot be upper bounded by a polynomial function of n .

²Note that computing $\text{OR}(x)$ is equivalent to searching for a 1 in x .

³As mentioned above, it is difficult to prove lower bounds on classical computation in the standard model of computation. By “apparent quantum speedup”, we mean that the best-known classical algorithm for a problem is less efficient than the best-known quantum algorithm for that problem.

Overview

Symmetries, graph properties, and quantum speedups. Two observations stand out when we compare Simon’s algorithm to Grover’s. First, Simon’s algorithm gives a super-polynomial (i.e., massive) speedup, whereas Grover’s gives a polynomial (i.e., modest) speedup. Second, Simon’s algorithm applies to f_{Simon} while Grover’s applies to OR, and the latter function appears to be much less “structured” than the former. The natural question to ask is whether these two observations are related. In [Chapter 1](#), we study the extent to which a function’s lack of structure means that it can only admit a modest quantum speedup.

One natural way to quantify the unstructuredness of a function is in terms of its symmetry. The more symmetrical a function is, the less structure it has since it is more insensitive to the input. Mathematically, symmetries are described by groups and we say a function $f: E \subseteq \Sigma^n \rightarrow \{0, 1\}$ is symmetric under a permutation group G on the set $\{1, \dots, n\}$ if $x^\sigma := (x_{\sigma(1)}, \dots, x_{\sigma(n)}) \in E$ and $f(x) = f(x^\sigma)$ for all $x \in E$ and $\sigma \in G$. Aaronson and Ambainis [[AA14](#)] showed that functions f symmetric under S_n that are further symmetric under any permutation of the symbols of Σ can admit at most a polynomial quantum speedup. Later, Chailloux [[Cha18](#)] showed that functions f symmetric only under S_n can admit at most a polynomial quantum speedup using a cryptographically-inspired proof based on results of Zhandry [[Zha12a](#)]. In the first chapter, we generalize Chailloux’s techniques to prove that there can only be a polynomial quantum speedup for deciding if a hypergraph satisfies some property when the hypergraph is specified by its adjacency matrix. Further work detailed in [[Ben+20](#)] goes on to establish a near-converse result: for any symmetry S that is “weaker” than the symmetry of a hypergraph, there exists an S -symmetric function that admits a super-polynomial quantum speedup. Taken together, this establishes a near-complete characterization of how symmetry relates to quantum speedup.

Returning to graph problems, we recall that the previous result held when the

input graph is specified in terms of its adjacency matrix. Another common way of specifying the input graph is in terms of its adjacency list. Perhaps surprisingly, we show that we can achieve an exponential quantum speedup for deciding a graph property in this case by a construction based on the glued-trees graph introduced in [Chi+03]. Further work detailed in [Ben+20] builds upon this construction using ideas from [Amb+17; ABK16] to give a graph property *testing* problem that yields an exponential speedup. In graph property testing, in addition to the input being promised to be in one of two sets, YES and NO, the NO set must consist of graphs that are ϵ -far away, under an appropriate notion of distance (see [GR02]), from any graph in YES.

Chapter 1 is based on [Ben+20]:

Shalev Ben-David, Andrew M. Childs, András Gilyén, William Kretschmer, Supartha Podder, and Daochen Wang. “Symmetries, Graph Properties, and Quantum Speedups”. In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 649–660. DOI: [10.1109/FOCS46700.2020.00066](https://doi.org/10.1109/FOCS46700.2020.00066). [arXiv:2006.12760](https://arxiv.org/abs/2006.12760)

Quantum divide and conquer. We have stressed that quantum speedups are subtle and there is no generic method of obtaining them. Therefore, we might be resigned to designing quantum algorithms for problems on a case-by-case basis. However, this is *not* the typical way a quantum algorithmic designer proceeds. Instead, the designer would first try to fit their problem into one of the main quantum algorithmic paradigms and design an algorithm within that paradigm. Some prominent examples of quantum algorithmic paradigms include: quantum walk search [Sze04], quantum dynamic programming [Amb+19], and quantum singular value transformation [GSLW19]. These paradigms are powerful because many problems can be solved using one of them.

In [Chapter 2](#), we develop the first quantum query analogue of one of the most powerful classical algorithmic paradigms, divide and conquer. Classically, a divide-and-conquer strategy divides a problem into parts, solves each part, and combines the partial solutions together. If part i takes time t_i , then the overall time taken is $\sum_i t_i$. The starting observation for quantum divide and conquer is that the overall quantum query complexity of a problem can be *smaller* than the sum of the partial quantum query complexities. For example, it could be that part i takes q_i queries yet the overall number of queries is of order $\sqrt{\sum_i q_i^2}$. This is a consequence of composition lemmas for a measure of complexity called the adversary quantity [[Amb02](#); [Rei09a](#); [Lee+11](#)] which is equal to the quantum query complexity up to a multiplicative constant. Next, as in classical divide and conquer, we interpret some of the q_i s as the quantum query complexity of a smaller *instance* of the initial problem and bound the remaining q_i s using algorithmic techniques. This yields a recurrence relation that we can then solve to obtain the overall quantum query complexity. A novel feature of the recurrence relation is that it is derived by a combination of algorithmic and adversary techniques. This means it is not obvious how to recover its solution using only one of these techniques.

To showcase the usefulness of our framework, we apply it to obtain near-optimal quantum query complexities for four string problems: (i) recognizing regular languages; (ii) decision versions of String Rotation and String Suffix; and natural parameterized versions of (iii) Longest Increasing Subsequence and (iv) Longest Common Subsequence. Using our framework, we obtain results for (i) and (ii) that slightly improve over those in [[AGS19](#); [AJ23](#)] in an easier way. Our results for (iii) and (iv) are new as we are the first to study these problems. Interestingly, the form of the recurrence relation arising from quantum divide and conquer allows us to apply novel combinatorial arguments for solving these problems that do not appear in the classical setting.

Chapter 2 is based on [Chi+22]:

Andrew M. Childs, Robin Kothari, Matt Kovacs-Deak, Aarthi Sundaram, and Daochen Wang. *Quantum divide and conquer*. 2022.

[arXiv:2210.06419](https://arxiv.org/abs/2210.06419)

Quantum exploration algorithms for multi-armed bandits. In Chapter 3, we present a specific application of quantum computing to the multi-armed bandit (MAB) problem, which lies at the theoretical root of the field of reinforcement learning (RL). Since RL is considered one of the most promising approaches to general artificial intelligence, even a modest speedup of its basic components can be significant. Classically, the exploration version of the MAB problem is formulated as follows. Given n biased coins with biases p_1, p_2, \dots, p_n , identify the i with the largest p_i using the fewest number of coin tosses. This i is known as the “best arm” of the MAB. Therefore, the problem is also known as the best-arm identification problem.

Another perspective on the MAB problem is that it naturally generalizes the search problem, which corresponds to the case when exactly one of the p_i s is equal to 1 and the rest are 0. Indeed, my original motivation for studying MAB was to probe the quantum speedups admitted by problems in the neighborhood of the search problem. In particular, could the introduction of certain p_i s inject the extra structure into the search problem that we know is necessary for a massive quantum speedup?

To formulate a quantum version of the MAB problem, we assume quantum query access to a coin-tossing oracle. To justify this assumption, we give a way of instantiating this oracle in the real world in scenarios where the outcome of tossing each coin can be simulated by a computer program.⁴ Letting Δ_i denote the difference between the largest and i th largest values in $\{p_j\}_{j=1}^n$ and $H := \sum_{i=2}^n \Delta_i^{-2}$, we construct a quantum algorithm that uses $\tilde{O}(\sqrt{H})$ queries to the oracle to identify the best

⁴Since the oracle can be instantiated in the real world, our algorithm can be applied in the real world. Note that we do not consider oracle instantiation in other chapters.

arm.⁵ Our algorithm is based on a quantization of the classical successive elimination strategy in [EMM02] using the variable-time amplitude amplification and estimation framework developed in [Amb12; CKS17; CGJ19]. We show that the bound is tight up to poly-logarithmic factors by a modified adversary method [Amb02]. The standard adversary method lower bounds quantum query complexity by the adversary quantity. In the MAB setting, we cannot directly use the adversary quantity since the standard adversary method assumes the input oracle encodes a bitstring, but here it encodes a probability distribution. Nevertheless, by modifying the proof of the standard adversary method, we are able to obtain our lower bound of $\Omega(\sqrt{H})$.⁶ Combining our lower bound with a classical upper bound of $\tilde{O}(H)$ [EMM02] gives a negative answer to the question of whether MAB can admit a massive quantum speedup for any choice of p_i s.

Chapter 3 is based on [WYLC21]:

Daochen Wang, Xuchen You, Tongyang Li, and Andrew M. Childs. “Quantum Exploration Algorithms for Multi-Armed Bandits”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.11 (2021), pp. 10102–10110.

[arXiv:2007.07049](https://arxiv.org/abs/2007.07049)

Preliminaries

Notation

\mathbb{N} denotes the set of positive integers. For $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, 2, \dots, n\}$. For a set X , we write $X^{n \times n}$ for the set of $n \times n$ matrices with elements in X . If X is a finite set, we write $x \leftarrow X$ to mean x is sampled uniformly at random from X .

⁵For simplicity, we assumed here that there is a unique best arm. It is easy to relax this assumption. The time complexity of our algorithm is also $\tilde{O}(\sqrt{H})$

⁶[Bel15] also generalizes the adversary method to work for more general input oracles. However, we found it difficult to apply its results to the MAB problem.

For two sets X, Y , we write $X \setminus Y$ for $\{x \in X \mid x \notin Y\}$. We write Y^X for the set of functions from X to Y . For $a, b \in [n]$, we write $a + b \pmod n$ to denote the unique integer $c \in \{0, \dots, n-1\}$ such that $c = a + b \pmod n$. For a vector space V , we write $\mathbf{1}_V$ for the identity operator on V . For $n \in \mathbb{N}$, we write $\mathbf{1}_n$ for the $n \times n$ identity matrix.

Let $f, g: \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$. We write $f = O(g)$ if there exists a $c > 0$ and an $x_0 \in \mathbb{R}_{\geq 0}$ such that $f(x) \leq cg(x)$ for all $x \geq x_0$. We write $f = \tilde{O}(g)$ if there exists a polynomial $p: \mathbb{R} \rightarrow \mathbb{R}$ and an $x_0 \in \mathbb{R}_{\geq 0}$ such that $f(x) \leq g(x) \cdot p(\log(g(x)))$ for all $x \geq x_0$. We write $f = \Omega(g)$ if there exists a constant c and an $x_0 \in \mathbb{R}_{\geq 0}$ such that $f(x) \geq cg(x)$ for all $x \geq x_0$. Note that $f = O(g) \iff g = \Omega(f)$. We write $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$.

Dirac's quantum notation. An N -dimensional quantum state is a vector in \mathbb{C}^N , i.e., $v = (v_1, \dots, v_N)^\top$ with $\|v\|_2 = 1$. The vector v is written in Dirac notation as $|v\rangle$ and called a “ket”. The complex conjugate transpose of $|v\rangle$ is written $\langle v|$ and called a “bra”, i.e., $\langle v| := v^\dagger$. The naming is based on the observation that a bra $\langle v|$ can be right-multiplied by a ket $|w\rangle$ to give the inner product “bracket” between v and w , i.e., $\langle v|w\rangle := \langle v| |w\rangle = v^\dagger w = \langle v, w\rangle \in \mathbb{C}$. Note that a $\langle v|$ can also be left-multiplied by $|w\rangle$ to give the outer product between w and v , i.e. $|w\rangle\langle v| = wv^\dagger \in \mathbb{C}^{N \times N}$. The *tensor product* of quantum states is their Kronecker product: if $|v\rangle \in \mathbb{C}^{N_1}$ and $|w\rangle \in \mathbb{C}^{N_2}$, then $|v\rangle |w\rangle := |v\rangle \otimes |w\rangle \in \mathbb{C}^{N_1} \otimes \mathbb{C}^{N_2}$. The *computational basis* of \mathbb{C}^N is the set of vectors $\{e^{(1)}, \dots, e^{(N)}\}$, where $e^{(i)} = (0, \dots, 1, \dots, 0)^\top$ is the all-zeros vector except with a 1 in the i th coordinate. We reserve the symbol $|i\rangle$ for $e^{(i)}$ and $\langle i|$ for $e^{(i)\dagger}$. Then, for example, $|v\rangle = \sum_{i=1}^N v_i |i\rangle$ and $\langle v| = \sum_{i=1}^N v_i^* \langle i|$. An *n -qubit quantum state* is a 2^n -dimensional quantum state. When describing quantum states on qubits, we reserve the symbol $|0\rangle$ for $(1, 0)^\top \in \mathbb{C}^2$ and the symbol⁷ $|1\rangle$ for

⁷The symbol $|1\rangle$ is technically overloaded since it could also mean $|i\rangle$ with $i = 1$, though it should always be clear from the context which meaning is being used.

$(0, 1)^\top \in \mathbb{C}^2$. Then, an n -qubit state can be written as $\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$, where $\alpha_x \in \mathbb{C}$ and $|x\rangle := |x_1\rangle |x_2\rangle \dots |x_n\rangle \in (\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$.

Query complexity

In this section, we mathematically define deterministic, randomized, and quantum query computation. Our exposition closely follows that in de Wolf's PhD thesis [Wol01]. Let $n \in \mathbb{N}$ and $f: E \subseteq \Sigma^n \rightarrow \{0, 1\}$, where E and Σ are finite sets. We fix a bijection $\phi: \Sigma \rightarrow [|\Sigma|]$ which is necessary for defining the quantum oracle. It is easy to see that the choice of the bijection ϕ does not affect our results.

A deterministic query computation (given⁸ f) is modelled by a deterministic decision tree (or, deterministic algorithm), i.e., a rooted ordered $|\Sigma|$ -ary tree, T . Each internal (i.e., non-leaf) node of T is labelled by an integer in $[n]$, each leaf of T is labelled by 0 or 1, and each edge of T is labelled by an element of Σ . Given $x \in \Sigma^n$, T can be naturally used to compute $T(x) \in \{0, 1\}$ as follows.

Set v_{current} to be the root vertex. Then, repeat the following until the label of a leaf is output:

1. If v_{current} is a leaf then output its label.
2. Otherwise, let $i \in [n]$ be the label of v_{current} and let v be the child of v_{current} such that the edge $\{v_{\text{current}}, v\}$ is labelled by x_i . Set $v_{\text{current}} = v$.

We say that T computes f if $T(x) = f(x)$ for all $x \in E$. The depth of T , $d(T)$, is defined to be the maximum length of a path from the root to a leaf in T . Then, the deterministic query complexity of f is defined as

$$D(f) := \min\{d(T) \mid T \text{ computes } f\}. \tag{1}$$

⁸The parameters of a deterministic, randomized, and quantum query computation depend on the domain and codomain of f . When f is clear from the context, we will often not specify it explicitly.

Randomized query computation is modelled by a randomized decision tree (or, randomized algorithm), i.e., a probability distribution \mathcal{T} over the set of all deterministic decision trees. For $\epsilon \in [0, 1/2)$, we say that \mathcal{T} computes f with two-sided error ϵ if $\Pr[T(x) \neq f(x) \mid T \leftarrow \mathcal{T}] \leq \epsilon$ for all $x \in E$. The depth of \mathcal{T} , $d(\mathcal{T})$, is defined to be $\max\{d(T) \mid \Pr(T \leftarrow \mathcal{T}) > 0\}$. Then, the randomized query complexity of f with two-sided error ϵ is defined as

$$R_\epsilon(f) := \min\{d(\mathcal{T}) \mid \mathcal{T} \text{ computes } f \text{ with two-sided error } \epsilon\}. \quad (2)$$

By convention, the randomized query complexity of f , $R(f)$, is defined to be $R_{1/3}(f)$. It is easy to see that there exists a constant $c > 0$ such that $R_\epsilon(f) < c \cdot R(f) \cdot \log(1/\epsilon)$ for all $\epsilon < 1/3$, which follows from error suppression using majority voting.

A quantum query computation is modelled by a quantum query algorithm. To define a quantum query algorithm, we use Dirac's quantum notation.

Let $d \geq 0$ and $r \geq 1$ be integers. A quantum query algorithm \mathcal{A} of depth d with r work qubits is specified by a sequence of d unitary matrices (U_0, U_1, \dots, U_d) acting on $\mathbb{C}^n \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r}$. To describe how \mathcal{A} computes, we first define the quantum oracle encoding $x \in \Sigma^n$, denoted O_x . O_x is a linear map defined by its action on the basis set $\{|i\rangle |b\rangle |w\rangle \mid i \in [n], b \in [|\Sigma|], w \in \{0, 1\}^r\}$ of $\mathbb{C}^n \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r}$ as follows:

$$\begin{aligned} O_x: \mathbb{C}^n \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r} &\rightarrow \mathbb{C}^n \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r} \\ |i\rangle |b\rangle |w\rangle &\mapsto |i\rangle |b + \phi(x_i) \pmod{|\Sigma|}\rangle |w\rangle. \end{aligned} \quad (3)$$

For $x \in \Sigma^n$, the final quantum state of \mathcal{A} on input x is defined as

$$|\psi(\mathcal{A}, x)\rangle := (U_d O_x) \dots (U_1 O_x) U_0 e_1, \quad (4)$$

where e_1 denotes the first standard basis vector of $\mathbb{C}^n \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r}$.

We denote the output distribution of \mathcal{A} on input x by $\Delta(\mathcal{A}, x)$ which is defined by

$$\Pr(a \leftarrow \Delta(\mathcal{A}, x)) = \left\| (\mathbf{1}_{\mathbb{C}^n \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes (r-1)} \otimes |a\rangle\langle a|} |\psi(\mathcal{A}, x)\rangle \right\|^2, \quad (5)$$

where $a \in \{0, 1\}$.

The quantum query complexity of f with two-sided error ϵ , $Q_\epsilon(f)$, is defined as the minimum $d \geq 0$ such that there exists a quantum algorithm \mathcal{A} of depth d with some number $r \geq 1$ of work qubits such that \mathcal{A} computes f with two-sided error ϵ .

By convention, the quantum query complexity of f , $Q(f)$, is defined to be $Q_{1/3}(f)$. Again, there exists a constant $c > 0$ such that $Q_\epsilon(f) < c \cdot Q(f) \cdot \log(1/\epsilon)$ for all $\epsilon < 1/3$, which follows from error suppression using majority voting.

Chapter 1

Symmetries, graph properties, and quantum speedups

Chapter summary. Quantum computers can sometimes exponentially outperform classical ones, but only for problems with sufficient structure. While it is well known that query problems with full permutation symmetry can have at most polynomial quantum speedup—even for partial functions—it is unclear how far this condition must be relaxed to enable exponential speedup. In particular, it is natural to ask whether exponential speedup is possible for (partial) *graph properties*, in which the input describes a graph and the output can only depend on its isomorphism class.

We show that the answer to this question depends strongly on the input model. In the adjacency matrix model, we prove that the bounded-error randomized query complexity R of any graph property \mathcal{P} has $R(\mathcal{P}) = O(Q(\mathcal{P})^6)$, where Q is the bounded-error quantum query complexity. This negatively resolves an open question of Montanaro and de Wolf in the adjacency matrix model. More generally, we prove $R(\mathcal{P}) = O(Q(\mathcal{P})^{3l})$ for any l -uniform hypergraph property \mathcal{P} in the adjacency matrix model. In direct contrast, in the adjacency list model for bounded-degree graphs, we exhibit a promise problem that shows an exponential separation between the randomized and quantum query complexities.

1.1 Introduction

Quantum computers offer the prospect of solving certain problems exponentially faster than is possible classically. The first concrete hint of this possibility came from the model of query complexity, where the input is provided by a black box and the computational cost is quantified as the number of queries to that box. In this model, there is an exponential quantum speedup between deterministic classical and quantum computation [DJ92], and even between bounded-error classical and quantum computation [Sim94]. Indeed, these algorithms directly motivated Shor’s algorithm for factoring, which replaces the black box with an efficiently computable function to give an (apparent) exponential speedup for an explicit problem [Sho97].

Since query complexity provides a useful testbed for understanding the potential power of quantum computers, it is natural to explore which problems allow for quantum speedup in this model. In the negative direction, it has been known for over twenty years that for total functions—i.e., query problems that are defined for any possible input string—quantum computers can offer at most a polynomial advantage [Bea+01]. More precisely, suppose the goal is to compute some known function

$$\mathcal{P}: S \rightarrow \{0,1\} \tag{1.1}$$

on a black-box input $x \in S \subseteq \Sigma^m$, where Σ is a finite set (the *input alphabet*). The domain S of \mathcal{P} is referred to as the *promise* on the input. When $S = \Sigma^m$, we say \mathcal{P} is *total*; otherwise we say it is *partial*. The deterministic, randomized, and quantum query complexities of \mathcal{P} are denoted $D(\mathcal{P}) \geq R(\mathcal{P}) \geq Q(\mathcal{P})$, respectively. For a total \mathcal{P} , Beals et al. show that $D(\mathcal{P}) = O(Q(\mathcal{P})^6)$ [Bea+01].

This result establishes that a promise is necessary to achieve superpolynomial quantum speedup. Thus it is natural to ask what kinds of promises can and cannot allow for a significant quantum advantage. In particular, if a query problem is highly

symmetric, then superpolynomial quantum speedup remains impossible, even if the problem is partial. Specifically, Aaronson and Ambainis show that if $\Sigma = \{0, 1\}$, S is closed under permutations of the input bits, and \mathcal{P} is invariant under those permutations, then $R(\mathcal{P}) = O(Q(\mathcal{P})^2)$ [AA14, Appendix]. Subsequently, Chailloux showed that $R(\mathcal{P}) = O(Q(\mathcal{P})^3)$ for any finite set Σ [Cha18] using a simpler proof.

On the other hand, with less than full permutation symmetry, it is unclear when quantum speedups are possible. A natural class of problems with significant symmetry, though much less than full permutation symmetry, is the class of *graph properties*. For such problems, the input describes a graph, and the output depends only on the isomorphism class of that graph. Thus the vertices can be permuted arbitrarily, but such a permutation induces a structured permutation on the edges, about which queries provide information. A graph property can be partial, i.e., there can be a promise that the input graph comes from a restricted family of (isomorphism classes of) graphs. In particular, partial graph properties arise in the setting of *graph property testing*, where the goal is to determine whether a given graph either has a certain property or is far from having that property.

Ambainis, Childs, and Liu studied quantum algorithms for graph property testing, giving polynomial quantum speedups for testing expansion and bipartiteness of bounded-degree graphs in the adjacency list model [ACL11]. Furthermore, they showed that at most a polynomial advantage is possible for testing expansion, and asked whether an exponential speedup is ever possible for graph property testing in the adjacency list model. In a subsequent survey, Montanaro and de Wolf asked the following more general question:

Open question ([MW16]). *Is there any graph property \mathcal{P} which admits an exponential quantum speed-up?*

This question takes different forms depending on the model of access to the graph. Indeed, we show that its answer depends strongly on which of two common input

models (formalised in [Section 1.2](#)) is used: the *adjacency matrix model* (in which the algorithm inputs a pair of vertices and the black box indicates whether they are adjacent) or the *adjacency list model* (in which the algorithm inputs a vertex and the black box returns its neighbours).

When the graph is specified in the adjacency matrix model, we prove in [Section 1.3](#) that exponential quantum speedup is impossible. In fact, we prove this not just for graph property testing, but for any partial graph property. Furthermore, we prove this for all l -uniform graph properties provided l is constant. Our proof is based on the framework of Chailloux [[Cha18](#)] which uses results of Zhandry [[Zha12a](#); [Zha15](#)].

In direct contrast, we show that exponential quantum speedup is possible for deciding graph properties in the adjacency list model. Specifically, in [Section 1.4](#) we design a promise on graphs with $\Theta(2^{2k})$ vertices such that the property \mathcal{P}_k that decides whether the graph has a vertex of degree 5 has $R(\mathcal{P}_k) = 2^{\Omega(k)}$ and $Q(\mathcal{P}_k) = \text{poly}(k)$. Our example is based on the glued-trees problem of Childs et. al. [[Chi+03](#)].

We conclude in [Section 1.5](#) with a brief discussion of subsequent work in [[Ben+20](#)] and some open problems.

1.2 Preliminaries

\mathbb{N} denotes the positive integers. Throughout this chapter, we let $n, d, l \in \mathbb{N}$, $1 \leq l \leq n$, and $M := \binom{n}{l}$. Recall that for $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, \dots, k\}$. For a finite set X , we write $\text{Sym}(X)$ for the set of permutations of the set X . S_n denotes $\text{Sym}([n])$. We write $\mathbb{P}(X)$ for the power set of X , i.e., the set of all subsets of X .

For a finite set V with $|V| \geq l$, we define the set of l -uniform hyperedges on V as

$$E_l(V) := \{e \subseteq V \mid |e| = l\}. \tag{1.2}$$

Note that $E_l(V)$ has size $\binom{|V|}{l}$. We write E_l for $E_l([n])$.

Then

Definition 1 (Hypergraphs). A simple l -uniform hypergraph G is a tuple (V, E) , where

1. V is a finite set with $|V| \geq l$, referred to as the vertex set of G .
2. E is a subset of $E_l(V)$.

A simple graph is a simple 2-uniform hypergraph.

In this chapter, we always focus on simple l -uniform hypergraphs on vertex set $[n]$ and so we often omit the adjectives. When it is clear what the vertex set of a graph is, we sometimes refer to a graph by its edge set. Similar results hold for hypergraphs that are not simple and we refer the reader to [Ben+20] for more details.

We consider the adjacency matrix model for specifying a hypergraph and the adjacency list model for specifying a graph. We will use Fig. 1.1 to illustrate these models.

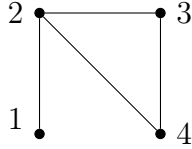


Figure 1.1: An illustration of a graph on vertex set $[4]$.

Adjacency matrix model. In the adjacency matrix model, we first fix a bijection between E_l and $[M]$. Then we model an l -uniform hypergraph by an M -bit string $x \in \{0, 1\}^M$ that indicates the presence (1) or absence (0) of each hyperedge. (We will explain why this bitstring is essentially equivalent to the adjacency matrix shortly.) For example, under the bijection

$$1 \leftrightarrow \{1, 2\}, 2 \leftrightarrow \{1, 3\}, 3 \leftrightarrow \{1, 4\}, 4 \leftrightarrow \{2, 3\}, 5 \leftrightarrow \{2, 4\}, 6 \leftrightarrow \{3, 4\}, \quad (1.3)$$

the graph in Fig. 1.1 is specified by

$$x = 100111. \tag{1.4}$$

Observe that the adjacency matrix A of the graph in Fig. 1.1 is given by

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \tag{1.5}$$

and 100111 corresponds to reading off the upper triangular part of this adjacency matrix row-by-row. Since A is symmetric, the lower triangular part of A contains no extra information. Therefore, 100111 is essentially equivalent to A .

Now, each permutation $\pi \in S_n$ of $[n]$ naturally induces a permutation $\tilde{\pi} \in S_M$ of hyperedges $[M]$ defined by

$$\tilde{\pi}(\{u_1, u_2, \dots, u_l\}) = \{\pi(u_1), \pi(u_2), \dots, \pi(u_l)\}. \tag{1.6}$$

Then, we say $\mathcal{P}: S \subseteq \{0, 1\}^M \rightarrow \{0, 1\}$ is an adjacency-matrix hypergraph property if

$$x \in S \implies x \circ \tilde{\pi} \in S \text{ and } \mathcal{P}(x) = \mathcal{P}(x \circ \tilde{\pi}) \text{ for all } \pi \in S_n, \tag{1.7}$$

where we have identified $\{0, 1\}^M$ with $\{0, 1\}^{[M]}$ in the natural way so that $x \in S$ is identified with the function from $[M]$ to $\{0, 1\}$ that maps i to x_i for all $i \in [M]$, and \circ denotes composition of functions.

Adjacency list model. The adjacency list model is an alternative model for specifying a graph given a bound on its maximum degree (i.e., the maximum number of neighbours a vertex has). In this model, a graph G on vertex set $[n]$ of maximum

degree d is modelled by a function $x: [n] \times [d] \rightarrow [n] \cup \{*\}$ with

$$(u, i) \mapsto \begin{cases} v \in [n] & \text{if } v \text{ is the } i\text{th neighbour of } u, \\ * & \text{if } u \text{ has fewer than } i \text{ neighbours.} \end{cases} \quad (1.8)$$

Note that x may be non-unique to the choice in the ordering of neighbours. For example, under the natural bijection between the set of functions from $[n] \times [d]$ to $[n] \cup \{*\}$ and the set of $n \times d$ matrices with entries in $[n] \cup \{*\}$, the graph in [Fig. 1.1](#) can be modelled by

$$x = \begin{bmatrix} 2 & * & * \\ 1 & 3 & 4 \\ 4 & 2 & * \\ 2 & 3 & * \end{bmatrix} \quad \text{or} \quad x = \begin{bmatrix} 2 & * & * \\ 4 & 1 & 3 \\ 2 & 4 & * \\ 3 & 2 & * \end{bmatrix} \quad (1.9)$$

among other possibilities. We write $\text{AdjList}(G)$ for the set of all functions from $[n] \times [d]$ to $[n] \cup \{*\}$ that model G .

Additional query complexity notation. We now introduce some convenient terminology related to quantum query algorithms. Let \mathcal{P} be a function

$$\mathcal{P} : S \subseteq \Sigma^m \rightarrow \{0, 1\}. \quad (1.10)$$

We use the word “estimate” as follows.

Definition 2. For $\alpha \in (0.5, 1]$, we say a randomized decision tree or quantum algorithm \mathcal{A} α -estimates \mathcal{P} if \mathcal{A} computes \mathcal{P} with two-sided error at most $1 - \alpha$. We say \mathcal{A} estimates \mathcal{P} if \mathcal{A} $(2/3)$ -estimates \mathcal{P} .

For a quantum query algorithm $\mathcal{A} = (U_0, U_1, \dots, U_q)$ that uses q quantum queries and r work qubits, we define the quantum circuit of \mathcal{A} on input $x \in \Sigma^m$ by the unitary

matrix Q_x acting on $\mathbb{C}^m \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r}$ defined by

$$Q_x := U_q O_x U_{q-1} \dots U_1 O_x U_0, \quad (1.11)$$

where O_x is the quantum oracle encoding $x \in \Sigma^m$, which we recall is a unitary matrix defined by

$$\begin{aligned} O_x: \mathbb{C}^m \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r} &\rightarrow \mathbb{C}^m \otimes \mathbb{C}^{|\Sigma|} \otimes (\mathbb{C}^2)^{\otimes r} \\ |i\rangle |b\rangle |w\rangle &\mapsto |i\rangle |b + \phi(x_i) \pmod{|\Sigma|}\rangle |w\rangle, \end{aligned} \quad (1.12)$$

Note that O_x always acts as the identity on the work register, $(\mathbb{C}^2)^{\otimes r}$.

1.3 No exponential speedup in the adjacency matrix model

The aim of this section will be to prove the following theorem.

Theorem 1. *For any l -uniform hypergraph property $\mathcal{P}: S \subseteq \{0, 1\}^M \rightarrow \{0, 1\}$, we have*

$$R(\mathcal{P}) = O(Q(\mathcal{P})^{3l}). \quad (1.13)$$

In particular, if $l = 2$, \mathcal{P} is a graph property and we have $R(\mathcal{P}) = O(Q(\mathcal{P})^6)$.

Our proof strategy closely follows [Cha18]. We construct a randomized decision tree that estimates \mathcal{P} using $O(q^{3l})$ queries from a quantum circuit that estimates \mathcal{P} using q queries.

Proof. Throughout this proof, we fix an arbitrary adjacency-matrix hypergraph property $\mathcal{P}: S \subseteq \{0, 1\}^M \rightarrow \{0, 1\}$. Let $q := Q(\mathcal{P})$.

By definition of q , there exists a q -query quantum circuit Q'_x of the form in Eq. (1.11) which estimates \mathcal{P} . Q'_x acts on $\mathbb{C}^M \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes r}$. By repeating Q'_x

three times, and outputting a bit according to majority vote, the probability of success can be boosted to at least $3 \cdot (2/3)^2 \cdot (1/3) + (2/3)^3 = 20/27$. Therefore, there exists a $3q$ -query quantum circuit

$$\mathbf{Q}_x = U_{3q} O_x U_{3q-1} \dots U_1 O_x U_0 \quad (1.14)$$

which $(20/27)$ -estimates \mathcal{P} . Note that $x \in \{0, 1\}^M$ describes the input graph.

Let $N := \sum_{i=1}^l \binom{n}{i}$. Recall from [Eq. \(1.2\)](#) that for $i \in [l]$, E_i denotes the set of i -uniform hyperedges on vertex set $[n]$. In particular, $|E_i| = \binom{n}{i}$. Choose l arbitrary bijections:

$$\begin{aligned} \Phi_1 &: \left\{ 1, \dots, \binom{n}{l} \right\} \rightarrow E_l, \\ \Phi_2 &: \left\{ \binom{n}{l} + 1, \dots, \binom{n}{l} + \binom{n}{l-1} \right\} \rightarrow E_{l-1}, \\ &\dots \\ \Phi_l &: \left\{ \sum_{i=2}^l \binom{n}{i} + 1, \dots, \sum_{i=2}^l \binom{n}{i} + \binom{n}{1} \right\} \rightarrow E_1. \end{aligned} \quad (1.15)$$

Note that the domain of Φ_1 is equal to $[M]$. Then define the map $\Phi: [N] \rightarrow \bigcup_{i=1}^l E_{l-(i-1)}$ by

$$\Phi(x) = \begin{cases} \Phi_1(x) & \text{if } x \in \text{Domain}(\Phi_1), \\ \Phi_2(x) & \text{if } x \in \text{Domain}(\Phi_2), \\ \dots & \\ \Phi_l(x) & \text{if } x \in \text{Domain}(\Phi_l). \end{cases} \quad (1.16)$$

Note that $\Phi([M]) = E_l$. The reason for doing this shall become clear later, in the second remark following our definition of D_s . Suffice it to say now that we shall need to consider functions that map hyperedges connecting l vertices to ones connecting

between 1 and l vertices.

For any function $F: [M] \rightarrow [N]$, we define the oracle $O_{x \circ F}$, acting on $\mathbb{C}^M \otimes \mathbb{C}^N \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes r}$ as follows.

1. Define oracle \tilde{O}_x on $\mathbb{C}^N \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes r}$ by

$$\tilde{O}_x : |i\rangle |y\rangle |w\rangle \mapsto \begin{cases} |i\rangle |y \oplus x(i)\rangle |w\rangle & \text{if } 1 \leq i \leq M, \\ |i\rangle |y\rangle |w\rangle & \text{if } M < i \leq N. \end{cases} \quad (1.17)$$

2. Define unitary O_F on $\mathbb{C}^M \otimes \mathbb{C}^N \otimes (\mathbb{C}^2)^{\otimes r}$ by

$$O_F : |i\rangle |j\rangle |w\rangle \mapsto |i\rangle |j + F(i) \pmod N\rangle |w\rangle. \quad (1.18)$$

3. Define oracle $O_{x \circ F}$ on $\mathbb{C}^M \otimes \mathbb{C}^N \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes r}$ by

$$O_{x \circ F} := (O_F^\dagger \otimes \mathbf{1}_2 \otimes \mathbf{1}_{2^r})(\mathbf{1}_M \otimes \tilde{O}_x)(O_F \otimes \mathbf{1}_2 \otimes \mathbf{1}_{2^r}). \quad (1.19)$$

Then, for $i \in [M]$ and $b \in \Sigma$, we have

$$O_{x \circ F} |i\rangle |0_{\mathbb{C}^N}\rangle |b\rangle |w\rangle = |i\rangle |0_{\mathbb{C}^N}\rangle |b'\rangle |w\rangle, \quad (1.20)$$

where

$$|b'\rangle := \begin{cases} |b \oplus (x \circ F)(i)\rangle & \text{if } 1 \leq F(i) \leq M, \\ |b\rangle & \text{if } M < F(i) \leq N. \end{cases} \quad (1.21)$$

For any unitary U acting on $\mathbb{C}^M \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes r}$, we let $U \otimes \mathbf{1}_N$ denote the unitary acting on $\mathbb{C}^M \otimes \mathbb{C}^N \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes r}$ that acts as U on $\mathbb{C}^M \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes r}$ and acts as the identity on \mathbb{C}^N . Then, we define

$$\mathbf{Q}_x(F) := (U_{3q} \otimes \mathbf{1}_N) O_{x \circ F} (U_{3q-1} \otimes \mathbf{1}_N) \dots (U_1 \otimes \mathbf{1}_N) O_{x \circ F} (U_0 \otimes \mathbf{1}_N), \quad (1.22)$$

where the unitaries U_i are as in [Eq. \(1.14\)](#). Note that $\mathbf{Q}_x(F)$ has $3q$ O_F gates and $3q$ O_F^\dagger gates giving a total of $6q$ gates that are each either O_F or O_F^\dagger .

If F is a permutation of $[M]$, i.e., F bijects its domain $[M]$ with $[M] \subseteq [N]$, then $\mathbf{Q}_x(F) = \mathbf{Q}_{x \circ F} \otimes \mathbf{1}_2$. If, in addition, F is a permutation of $[M]$ induced by a permutation of vertices $[n]$, then $\mathcal{P}(x) = \mathcal{P}(x \circ F)$ as \mathcal{P} is a graph property. Therefore, we have

$$\Pr(\mathbf{Q}_x(F) \text{ outputs } \mathcal{P}(x)) = \Pr(\mathbf{Q}_{x \circ F} \text{ outputs } \mathcal{P}(x \circ F)) \geq \frac{20}{27} \quad (1.23)$$

as \mathbf{Q}_x ($20/27$)-estimates \mathcal{P} .

The core argument of our proof is that $\mathbf{Q}_x(F)$ can behave similarly to \mathbf{Q}_x even when F has a limited range. This argument uses the following [Theorem 2](#) on a suitable family of distributions on functions from $[M]$ to $[N]$.

Theorem 2 ([\[Zha12a, Theorem VII.3\]](#)). *Let $k \in \mathbb{Z}_{\geq 0}$. For $s \in \mathbb{N} \cup \{\infty\}$, let \mathfrak{D}_s be a distribution on functions from $[M] \rightarrow [N]$. Suppose there exists $\delta \in \mathbb{Z}_{\geq 0}$ such that, for any $(d_1, e_1), \dots, (d_{2k}, e_{2k}) \in [M] \times [N]$, there exists a polynomial $p : \mathbb{R} \rightarrow \mathbb{R}$ of degree at most δ such that*

$$p(1/s) = \Pr_{F \leftarrow \mathfrak{D}_s}(F(d_i) = e_i \text{ for all } i \in [2k]) \quad (1.24)$$

for all $s \in \mathbb{N} \cup \{\infty\}$ (we take $1/\infty$ to mean 0). Then, for any $\{0, 1\}$ -output quantum circuit $\mathbf{Q}(F)$ that makes k quantum queries, each to either¹ O_F or O_F^\dagger , we have

$$\sum_{z \in \{0, 1\}} \left| \Pr_{F \leftarrow \mathfrak{D}_s}(\mathbf{Q}(F) \text{ outputs } z) - \Pr_{F \leftarrow \mathfrak{D}_\infty}(\mathbf{Q}(F) \text{ outputs } z) \right| \leq \frac{\pi^2 \delta^3}{3s} \quad (1.25)$$

¹In the statement of [Theorem 2](#), queries are made only to O_F . However, no adjustment to the resulting [Eq. \(1.25\)](#) is needed when queries can also be made to O_F^\dagger . This can be easily seen by examining the proof of [\[Zha12a, Theorem VII.1\]](#), given in Appendix B.1 of the Cryptology ePrint Archive version of [\[Zha12b\]](#), and noting that O_F^\dagger is simply the unitary on $\mathbb{C}^M \otimes \mathbb{C}^N \otimes (\mathbb{C}^2)^{\otimes r}$ acting by $|i\rangle |j\rangle |w\rangle \mapsto |i\rangle |j - F(i) \bmod N\rangle |w\rangle$.

for all $s \in \mathbb{N} \cup \{\infty\}$.

Note that the probabilities appearing in [Eq. \(1.25\)](#) take their natural meaning:

$$\Pr_{F \leftarrow \mathfrak{D}_s}(\mathbf{Q}(F) \text{ outputs } z) := \sum_F \Pr(\mathbf{Q}(F) \text{ outputs } z) \cdot \Pr_{F \leftarrow \mathfrak{D}_s}(F). \quad (1.26)$$

To utilise [Theorem 2](#), we define, for each $s \in \mathbb{Z}_{\geq 0}$, a distribution D_s on functions $f: [n] \rightarrow [n]$ and $F: [M] \rightarrow [N]$, where sampling is obtained by the following procedure.

1. Sample $g \leftarrow [s]^{[n]}$, where we recall that the notation $[s]^{[n]}$ denotes the set of functions from $[n]$ to $[s]$.
2. Let $S := \{g(x) : x \in [n]\}$. That is, S is the range of g .
3. Sample $h \leftarrow \text{Inj}([n]^S)$, where $\text{Inj}([n]^S)$ denotes the set of injective functions from S to $[n]$. ($\text{Inj}([n]^S) \neq \emptyset$ since $|S| \leq n$.)
4. Output $f = h \circ g$.
5. Output $F: [M] \rightarrow [N]$, defined by $F(x) = \Phi^{-1}(\{f(u_1), \dots, f(u_l)\})$, such that $\{u_1, \dots, u_l\} = \Phi_1^{-1}(x)$. (F is well defined since $\Phi_1: [M] \rightarrow E_l$ and $\Phi: [N] \rightarrow \cup_{i=1}^l E_{l-(i-1)}$ are bijections.)

For $s = \infty$, we further define a distribution D_∞ on functions $f: [n] \rightarrow [n]$ and $F: [M] \rightarrow [N]$, where sampling is obtained by the following procedure.

1. $f \leftarrow \text{Inj}([n]^{[n]})$, where $\text{Inj}([n]^{[n]})$ denotes the set of injective functions from $[n]$ to $[n]$. ($\text{Inj}([n]^{[n]})$ is the same as the set of bijective functions from $[n]$ to $[n]$.)
2. Output $F: [M] \rightarrow [N]$, defined by $F(x) = \Phi^{-1}(\{f(u_1), \dots, f(u_l)\})$, such that $\{u_1, \dots, u_l\} = \Phi_1(x)$.

Because \mathcal{P} is a graph property, any such F must satisfy $\mathcal{P}(x) = \mathcal{P}(x \circ F)$, cf. [Eq. \(1.7\)](#).

Intuitively, D_∞ can be thought of as the limit of D_s , with $s \in \mathbb{Z}_{\geq 0}$, as s tends to infinity.

We make two remarks. First, the distribution D_s on f is exactly the same as that defined in [Zha15, Sec. 3.1]. Second, in Step 5, because f may map multiple inputs to the same output, sets in $\Phi(F([M]))$ may have fewer than l elements. In fact, they may be any of the N sets in $\bigcup_{i=1}^l E_{l-(i-1)}$.

Now, the following Lemma 1 allows us to apply Theorem 2 to the distribution D_s on F . Our Lemma 1 uses and can be compared with [Zha15, Lemma 1].

Lemma 1. *Let $k \in \mathbb{N}$. Then, for any $(d_1, e_1), \dots, (d_k, e_k) \in [M] \times [N]$, there exists a polynomial $p : \mathbb{R} \rightarrow \mathbb{R}$ of degree at most $kl - 1$ such that*

$$p(1/s) = \Pr_{F \leftarrow D_s}(F(d_i) = e_i \text{ for all } i \in [k]) \quad (1.27)$$

for all $s \in \mathbb{N} \cup \{\infty\}$.

Proof. For each $i \in [k]$, write $d_i = \Phi_1^{-1}(\{u_1^{(i)}, \dots, u_{l_i}^{(i)}\})$ and $e_i = \Phi^{-1}(\{v_1^{(i)}, \dots, v_{l_i}^{(i)}\})$, where l_i is some integer in $[l]$. By definition, we have

$$F(d_i) = e_i \iff \{f(u_1^{(i)}), \dots, f(u_{l_i}^{(i)})\} = \{v_1^{(i)}, \dots, v_{l_i}^{(i)}\}. \quad (1.28)$$

Write $\{u_j\}_{j=1}^a = \bigcup_{i=1}^k \Phi(d_i)$ and $\{v_j\}_{j=1}^b = \bigcup_{i=1}^k \Phi(e_i)$, where $a \leq kl$ and $b \leq \sum_{i=1}^k l_i$. Then, the event $\{F : [M] \rightarrow [N] \mid F(d_i) = e_i \text{ for all } i \in [k]\}$ can be expressed as a disjoint union of events of the form

$$\{f : [n] \rightarrow [n] \mid f(u_1) = v_{j_1}, \dots, f(u_a) = v_{j_a}\}. \quad (1.29)$$

Therefore, $\Pr_{F \leftarrow D_s}(F(d_i) = e_i \text{ for all } i \in [k])$ is a sum of probabilities of the form

$$\Pr_{f \leftarrow D_s}(f(u_1) = v_{j_1}, \dots, f(u_a) = v_{j_a}). \quad (1.30)$$

But [Zha15, Lemma 1] shows that the probabilities in Eq. (1.30) can be represented

by a polynomial in $1/r$ of degree at most $a - 1 \leq kl - 1$. Hence the lemma. \square

Now, by setting the “ k ” in the statement of [Lemma 1](#) to $12q$, we see that for any $(d_1, e_1), \dots, (d_{12q}, e_{12q}) \in [M] \times [N]$, there exists a polynomial $p: \mathbb{R} \rightarrow \mathbb{R}$ of degree at most $12ql - 1$ such that

$$p(1/s) = \Pr_{F \leftarrow D_s}(F(d_i) = e_i \text{ for all } i \in [k]) \quad (1.31)$$

for all $s \in \mathbb{N} \cup \{\infty\}$.

Therefore, [Theorem 2](#) implies that for any $\{0, 1\}$ -output quantum circuit $Q(F)$ that makes $6q$ quantum queries, each to either O_F or O_F^\dagger , we have

$$\sum_{z \in \{0,1\}} |\Pr_{F \leftarrow D_s}(Q(F) \text{ outputs } z) - \Pr_{F \leftarrow D_\infty}(Q(F) \text{ outputs } z)| \leq \frac{\pi^2(12ql - 1)^3}{3s} \quad (1.32)$$

for all $s \in \mathbb{N} \cup \{\infty\}$.

But $Q_x(F)$ is a circuit that makes $6q$ queries, each to either O_F or O_F^\dagger . Therefore, by setting the “ s ” in [Eq. \(1.32\)](#) equal to

$$s_0 := \left\lceil \frac{\pi^2(12ql - 1)^3}{3} \cdot \frac{27}{2} \right\rceil, \quad (1.33)$$

we deduce

$$\sum_{z \in \{0,1\}} |\Pr_{F \leftarrow D_{s_0}}(Q_x(F) \text{ outputs } z) - \Pr_{F \leftarrow D_\infty}(Q_x(F) \text{ outputs } z)| \leq \frac{2}{27}. \quad (1.34)$$

We now define a randomized decision tree R that estimates \mathcal{P} as follows.

R. Given an input graph $x \in \{0, 1\}^M$, do the following.

1. Sample a random function $F: [M] \rightarrow [N]$ according to D_{s_0} .
2. Query bits $x(i)$ for those $i \leq M$ in the image of F , i.e., $i \in I := \text{Im}(F) \cap [M]$.
3. Output $z \in \{0, 1\}$ according to the output distribution of the quantum circuit $\mathbf{Q}_x(F)$.

First, by recalling from [Eq. \(1.21\)](#) the action of $O_{x \circ F}$, we see that the output distribution of $\mathbf{Q}_x(F)$ only depends on F and the values taken by $x(i)$ for $i \in I$. This output distribution can be *pre-computed* prior to executing R, and R simply draws from it in Step 3.

Second, let us prove the correctness of R.

Lemma 2. R estimates \mathcal{P} .

Proof. Let $x \in S \subseteq \{0, 1\}^M$ be an input graph, and write $z := \mathcal{P}(x)$.

Since any F drawn from D_∞ is a permutation of $[M]$ induced by a permutation of vertices $[n]$, [Eq. \(1.23\)](#) holds, i.e., $\Pr(\mathbf{Q}_x(F) \text{ outputs } z) \geq \frac{20}{27}$. So, by [Eq. \(1.26\)](#), we also have $\Pr_{F \leftarrow D_\infty}(\mathbf{Q}_x(F) \text{ outputs } z) \geq \frac{20}{27}$. So

$$\begin{aligned}
 P(\text{R outputs } z \text{ on input } x) &= \Pr_{F \leftarrow D_{s_0}}(\mathbf{Q}_x(F) \text{ outputs } z) \\
 &\geq \Pr_{F \leftarrow D_\infty}(\mathbf{Q}_x(F) \text{ outputs } z) - \frac{2}{27} \\
 &\geq \frac{20}{27} - \frac{2}{27} = \frac{2}{3},
 \end{aligned} \tag{1.35}$$

where the first equality follows from the definition of R and the first inequality follows from [Eq. \(1.34\)](#).

But $x \in S$ was arbitrary. Therefore, the last bound of [Eq. \(1.35\)](#) implies that R estimates \mathcal{P} . \square

Now, the query complexity, $\text{cost}(\text{R})$, of R is the size of the set $I = \text{Im}(F) \cap [M]$,

which is at most $\binom{s_0}{l}$. Then, by substituting in the expression for s_0 from [Eq. \(1.33\)](#), we deduce

$$R(\mathcal{P}) \leq \text{cost}(\mathbf{R}) \leq \binom{80000 q^3 l^3}{l} \leq (80000 q^3 l^3)^l = O(Q(\mathcal{P})^{3l}) \quad (1.36)$$

because $q = Q(\mathcal{P})$ by definition. Therefore, [Theorem 1](#) follows. □

1.4 Example of exponential speedup in the adjacency list model

We show that an exponential quantum query speedup exists in the adjacency list model by presenting an explicit graph property \mathcal{P}_k . The input to \mathcal{P}_k is promised to be what we call a “modified glued-trees graph (of depth $4k + 2$)” and \mathcal{P}_k outputs 1 if and only if that graph has a vertex of degree 5. We will show that $R(\mathcal{P}_k) = 2^{\Omega(k)}$ while $Q(\mathcal{P}_k) = \text{poly}(k)$.

We define modified glued-trees graphs using the glued-trees graphs in [\[Chi+03\]](#).

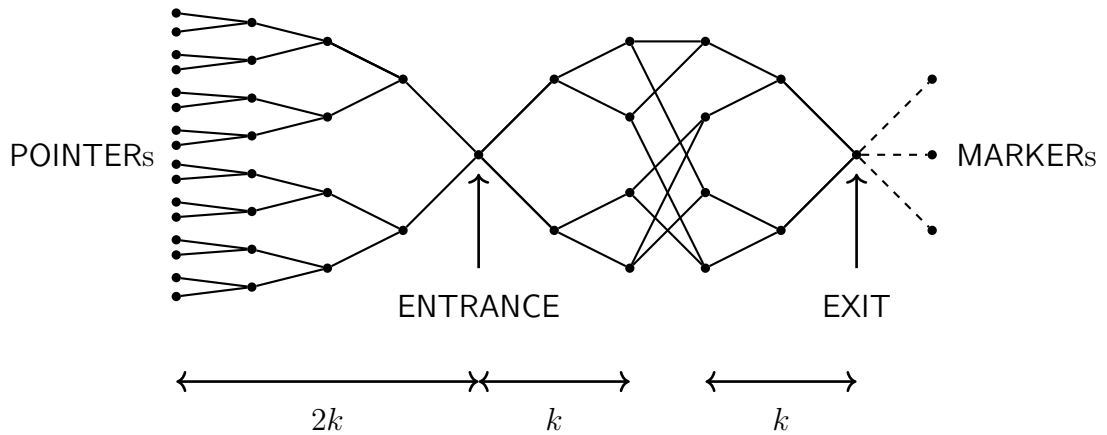


Figure 1.2: An illustration of a modified glued-trees graph in the case $k = 2$. This graph has a vertex of degree 5, i.e., has graph property \mathcal{P}_k , when the three MARKERS are connected to EXIT. It does not have \mathcal{P}_k when these MARKERS are isolated.

A glued-trees graph is defined as follows.

Definition 3 (Glued-trees graph). *Let $k \in \mathbb{N}$ and V be a finite set with $|V| = 2(2^{k+1} - 1)$. A graph G on vertex set V is called a glued-trees graph of depth $2k + 1$ if it can be specified as follows:*

1. *Let $L, R \subseteq V$ form a partition of V such that $|L| = |R| = \frac{|V|}{2}$.*
2. *Let B_L and B_R be perfect binary trees with 2^k leaves defined on vertex sets L and R respectively. Let $\text{Leaves}(B_L)$ and $\text{Leaves}(B_R)$ denote the set of leaves of B_L and B_R respectively.*
3. *Let C be a cycle on vertex set $\text{Leaves}(B_L) \cup \text{Leaves}(B_R)$ of the form*

$$C = \{(u_1, v_1), (v_1, u_2), (u_2, v_2), \dots, (u_{2^k}, v_{2^k}), (v_{2^k}, u_1)\}, \quad (1.37)$$

where $u_i \in \text{Leaves}(B_L)$ and $v_i \in \text{Leaves}(B_R)$ for all $i \in [2^k]$, and $u_i \neq u_j$ and $v_i \neq v_j$ for all $i, j \in [2^k]$ with $i \neq j$.

4. *Then $G = B_L \cup B_R \cup C$.*

For example, in [Fig. 1.2](#), the part of the graph between ENTRANCE and EXIT (inclusive) is a glued-trees graph of depth 5.

We now consider hidden glued-trees graphs. Each such graph consists of a glued-trees graph together with much more isolated vertices. The isolated vertices serve to “hide” the glued-trees graph. [\[CKS17\]](#) shows that, given a hidden glued-trees graph and a vertex of degree 2, it is hard for a few-query randomized algorithm to find another vertex of degree 2. We will later show that this task is easier than deciding \mathcal{P}_k and so deciding \mathcal{P}_k is hard for a few-query randomized algorithm. A hidden glued-trees graph is defined formally as follows.

Definition 4 (Hidden glued-trees graph). *Let $k \in \mathbb{N}$, V be a finite set with $|V| = 2(2^{k+1} - 1) + (2^{2k+1} - 2) + 3$. A graph G on vertex set V is called a hidden glued-trees graph of depth $2k + 1$ if it can be specified as follows:*

1. Let $V', V'' \subseteq V$ form a partition of V such that $|V'| = 2(2^{k+1} - 1)$, $|L| = 2^{2k+1} - 2$.
2. Let G' be a glued-trees graph on vertex set V' of depth k .
3. Let G'' be the graph on vertex set V'' with an empty set of edges.
4. Then $G = G' \cup G''$.

A modified glued trees graph G of depth $4k + 2$ is formed by a glued-trees graph G' of depth k , a depth- $2k$ perfect binary tree connected to a vertex of G' of degree 2, and three additional that are either (i) isolated or (ii) connected to the other vertex of G' of degree 2. In case (i), we call G a modified glued trees graph of of type 0 and G have no vertices of degree 5. In case (ii), we call G a modified glued trees graph of of type 1 and G has a vertex of degree 5. Deciding \mathcal{P}_k on G is then equivalent to deciding if G is of type 0 or 1. A modified glued-trees graph is defined formally as follows.

Definition 5 (Modified glued-trees graph). *Let $k \in \mathbb{N}$, V be a finite set with $|V| = 2(2^{k+1} - 1) + (2^{2k+1} - 2) + 3 = 2^{2k+1} + 2^{k+2} - 1$, and $b \in \{0, 1\}$. A graph G on vertex set V is called a type- b modified glued-trees graph of depth $4k + 2$ if it can be specified as follows:*

1. Let $V', L, R \subseteq V$ form a partition of V such that $|V'| = 2(2^{k+1} - 1)$, $|L| = 2^{2k+1} - 2$ and $|R| = 3$. Write $R = \{m_1, m_2, m_3\}$.
2. Let G' be a glued-trees graph on vertex set V' of depth k . Let $\{u, v\}$ denote the set of vertices in G' that have degree 2 (by the definition of a glued-trees graph, this set is of size 2).
3. Let B_L be a perfect binary tree with 2^{2k} leaves defined on vertex set $L \cup \{u\}$ with root u .
4. Then $G = B_L \cup G'$ if $b = 0$ or $G = B_L \cup G' \cup \{\{v, m_1\}, \{v, m_2\}, \{v, m_3\}\}$ if $b = 1$.

With respect to G , we call u the ENTRANCE, v the EXIT, any leaf of B_L a POINTER, and any vertex in $\{m_1, m_2, m_3\}$ a MARKER.

For example, Fig. 1.2 illustrates a modified glued-trees graph of depth 10. If the dashed lines are removed, then the graph is of type 0. If the dashed lines are made solid, then the graph is of type 1.

For $k \in \mathbb{N}$, set $n := 2(2^{k+1} - 1) + (2^{2k+1} - 2) + 3$. Let \tilde{Z}_k denote the set of all hidden glued-trees graphs of depth $2k + 1$ on vertex set $[n]$. Let \tilde{X}_k denote the set of all type-0 glued-trees graphs of depth $4k + 2$ on vertex set $[n]$. Let \tilde{Y}_k denote the set of all type-1 glued-trees graphs of depth $4k + 2$ on vertex set $[n]$. Let $\tilde{S}_k = \tilde{X}_k \cup \tilde{Y}_k$. Let

$$X_k := \bigcup_{G \in \tilde{X}_k} \text{AdjList}(G) \quad \text{and} \quad Y_k := \bigcup_{G \in \tilde{Y}_k} \text{AdjList}(G). \quad (1.38)$$

It is clear that $X_k \cap Y_k = \emptyset$ since graphs modelled by elements in X_k have maximum degree 4 whereas graphs modelled by elements in Y_k have maximum degree 5.

We define our promise set S_k to be $X_k \cup Y_k$ and the function $\mathcal{P}_k: S_k \rightarrow \{0, 1\}$ by $\mathcal{P}_k(x) = 0$ if $x \in X_k$ and $\mathcal{P}_k(x) = 1$ if $x \in Y_k$. \mathcal{P}_k is a graph property because X_k (resp. Y_k) remains invariant under vertex relabellings.

Theorem 3. *The randomized query complexity of $\mathcal{P}_k: S_k \rightarrow \{0, 1\}$ satisfies $R(\mathcal{P}_k) \geq \Omega(2^{k/6})$.*

Following [Chi+03], we prove the theorem by considering a sequence of games, Games A to E, that can be played and won or lost by a randomized algorithm that queries an oracle. We show that winning Game A is “easier” than deciding \mathcal{P}_k . We then show that each successive game from Game A to Game E is easier to win than the previous one. But [Chi+03] shows that any randomized algorithm that makes less than an exponential-in- k number of queries can only win Game E with a probability that decays exponentially with k . Therefore, deciding \mathcal{P}_k with probability greater than $2/3$ must take a randomized algorithm an exponential-in- k number of queries.

Formally, a game is defined as follows.

Definition 6 (Game). *A game is defined by the following data:*

1. An oracle set \mathcal{G} , which is a subset of the set of all functions from $[n]$ to Σ , where Σ is some finite set.
2. A winning condition $W: \mathcal{G} \rightarrow \mathbb{P}([n])$.
3. A restriction R , which is a subset of the set of all $|\Sigma|$ -ary deterministic decision trees with internal nodes labelled by $[n]$. We omit to define C unless C is a proper subset.

We say that a deterministic decision tree $T \in R$ wins game Γ on oracle $x \in \mathcal{G}$ if, given oracle x , T queries by an element in $W(x)$. Let \mathcal{T} be a randomized decision tree that is supported on deterministic decision trees in R . We define the *winning probability* of \mathcal{T} on game Γ by the probability that a deterministic decision tree sampled from \mathcal{T} wins game Γ .

For a graph G on vertex set $[n]$, we also define $\nu(G): [n] \rightarrow \mathbb{P}([n])$ to be the function such that $\nu(G)(i)$ equals the set of neighbours of vertex i in G . Note that ν is injective.

First, observe that a randomized algorithm that decides \mathcal{P}_k with high probability must also win the following game with high probability.

Game A.

1. Oracle set $S_k^A := \{\nu(G) \mid G \in \tilde{S}_k\}$.
2. Winning condition $W^A: S_k^A \rightarrow \mathbb{P}(n)$ defined by $W^A(\nu(G))$ is equal to the set containing the EXIT and all MARKERs of G .

Lemma 3. *Suppose there is a randomized decision tree \mathcal{A} of depth d that, given any oracle in S_k , correctly decides \mathcal{P}_k with probability $\geq \frac{1+p}{2}$ for some $0 \leq p \leq 1$. Then,*

there is a randomized decision tree \mathcal{A}^A of depth $\leq d$ that, given any oracle in S_k^A , wins *Game A* with probability at least $\geq p$.

Proof. First, note that one query to an oracle in S_k^A returns more information than one query to an oracle in S_k : the former returns all neighbours of a vertex, whereas the latter returns one neighbour of a vertex. Therefore, there exists a randomized decision tree \mathcal{A}^A of depth $d' \leq d$ that, given any oracle in S_k^A , correctly decides $\mathcal{P}_k^A: S_k^A \rightarrow \{0, 1\}$ with probability $\geq \frac{1+p}{2}$, where \mathcal{P}_k^A maps an $x := \nu(G) \in S_k^A$ to 0 if $G \in \tilde{X}_k$ and to 1 if $G \in \tilde{Y}_k$.

We proceed to show that, for any oracle $x \in S_k^A$, \mathcal{A}^A must query by an element of $W^A(x_G)$ with probability $\geq p$.

Fix $x := \nu(G) \in S_k^A$. If $G \in \tilde{X}_k$, let $\bar{G} \in \tilde{Y}_k$ be the same as G except with the MARKERS of G connected to the EXIT of G . If $G \in \tilde{Y}_k$, let $\bar{G} \in \tilde{X}_k$ be the same as G except with the MARKERS of G disconnected from the EXIT of G . Then, let $\bar{x} := \nu(\bar{G})$.

We consider the following subset of deterministic decision trees (for \mathcal{P}_k^A)²:

$$\Lambda_1 := \{T \mid T(x) = \mathcal{P}_k^A(x)\}, \quad (1.39)$$

$$\Lambda_2 := \{T \mid T(\bar{x}) = \mathcal{P}_k^A(\bar{x}) = 1 - \mathcal{P}_k^A(x)\}, \quad (1.40)$$

$$\Lambda_3 := \{T \mid T \text{ queries by an element of } W^A(x) \text{ given input } x\}. \quad (1.41)$$

Suppose that $T \notin \Lambda_3$, then we must have $T(x) = T(\bar{x})$ since, by the definition of \bar{x} , any query to x and \bar{x} by an $i \in [n] \setminus W^A(x)$ returns the same answer. Therefore

²In words, Λ_1 consists of deterministic decision trees that compute \mathcal{P}_k^A correctly on x , Λ_2 consists of deterministic decision trees that compute \mathcal{P}_k^A correctly on \bar{x} , and Λ_3 consists of deterministic decision trees that query by either the EXIT or a MARKER of x .

$T \in \Lambda_1 \cap \Lambda_2 \implies T \in \Lambda_3$ and so

$$\begin{aligned}
& \Pr[T \in \Lambda_3 \mid T \leftarrow \mathcal{A}^A] \\
& \geq \Pr[T \in \Lambda_1 \cap \Lambda_2 \mid T \leftarrow \mathcal{A}^A] \\
& = \Pr[T \in \Lambda_1 \mid T \leftarrow \mathcal{A}^A] + \Pr[T \in \Lambda_2 \mid T \leftarrow \mathcal{A}^A] - \Pr[T \in \Lambda_1 \cup \Lambda_2 \mid T \leftarrow \mathcal{A}^A] \\
& \geq \frac{1+p}{2} + \frac{1+p}{2} - 1 = p,
\end{aligned}$$

as desired. \square

Now consider:

Game B. Same as *Game A* except there exists $\alpha, \beta \in [n] - \{1\}$ with $\alpha \neq \beta$ such that the oracle set is defined as $S_k^B := \{\nu(G) \mid G \in \tilde{S}_k^B[\alpha, \beta]\}$, where $\tilde{S}_k^B[\alpha, \beta]$ consists of graphs $G \in \tilde{X}_k$ such that the ENTRANCE of G is 1 and the set of neighbours of 1 in the direction³ of the EXIT of G is $\{\alpha, \beta\}$.

Lemma 4. Suppose there is a randomized decision tree \mathcal{A}^A of depth d that, given any oracle in S_k^A , wins *Game A* with probability at least $\geq p$. Then there is a randomized decision tree $\mathcal{A}^B[\alpha, \beta]$ of depth d that, given any oracle in $S_k^B[\alpha, \beta]$, $\mathcal{A}^B[\alpha, \beta]$ wins *Game B* with probability at least $\geq p$.

Proof. Since $S_k^B[\alpha, \beta] \subseteq S_k^A$, it is clear that we can set $\mathcal{A}^B[\alpha, \beta]$ to be equal to \mathcal{A}^A . \square

Now consider:

Game C.

1. The oracle set $S_k^C[\alpha, \beta] := S_k^B[\alpha, \beta]$ is the same as that in *Game B*.
2. The winning condition $W^C : S_k^C \rightarrow \mathbb{P}([n])$ is the same as that in *Game B*.

³We say a neighbour v of 1 is in the direction of the EXIT of G if there is a path from v to EXIT that does not visit 1.

3. The restriction $R^C[\alpha, \beta]$ contains deterministic decision trees T such that each internal node η of T is labelled by an element in $\{\alpha, \beta\}$ or some $1 \neq j \in [n]$ such that j belongs to a set that labels an edge of T in the path from the root of T to η .

We can show that **Game C** is not much harder to win than **Game B**. More precisely:

Lemma 5. *Suppose there is a randomized decision tree $\mathcal{A}^B[\alpha, \beta]$ of depth $d \leq 2^k$ that, given any oracle in $S_k^C[\alpha, \beta]$, wins **Game B** with probability $\geq p$. Then, there is a randomized decision tree $\mathcal{A}^C[\alpha, \beta]$ of depth d that, for any graph $G \in \tilde{S}_k^B[\alpha, \beta]$, satisfies*

$$\begin{aligned} & \Pr[T \text{ wins } \mathbf{Game C} \text{ on } \nu(\pi(G)) \mid \pi \leftarrow \text{Sym}([n] \setminus \{1, \alpha, \beta\}), T \leftarrow \mathcal{A}^C[\alpha, \beta]] \\ & \geq p - O(d \cdot 2^{-k}), \end{aligned} \tag{1.42}$$

where $\pi(G)$ is the same as G but with its vertices permuted by π .

The intuition for **Lemma 5** is the following. To find the EXIT or a MARKER of a modified glued-trees graph $\pi(G)$, it does not help to query a “bad” vertex $v \notin \{\alpha, \beta\}$ that is not in a set that had been previously returned by the oracle. This is because, over the distribution $[\pi \leftarrow \text{Sym}([n] \setminus \{1, \alpha, \beta\})]$, v will lie in the depth- $2k$ perfect binary tree that is attached to the ENTRANCE with overwhelming probability. In that case, knowing the neighbours of v gives no information about where the EXIT or MARKERS are. To formalise this intuition, we let \mathcal{A}^C randomly sample a depth- $2k$ perfect binary tree which it uses to *simulate* the responses of the oracle (without querying it) whenever \mathcal{A}^B queries a bad vertex. Only when \mathcal{A}^B queries a good vertex does \mathcal{A}^C *really* query the oracle. We will argue that the simulated responses do not lead to an inconsistency from \mathcal{A}^B 's point of view with overwhelming probability. In the absence of an inconsistency, \mathcal{A}^B must find the EXIT or a MARKER with probability $\geq p$, and so must \mathcal{A}^C . Since \mathcal{A}^C only makes good queries, it is supported on deterministic decision trees in $R^C[\alpha, \beta]$ by definition.

Proof. For notational convenience, we write

1. \mathcal{A}^C for $\mathcal{A}^C[\alpha, \beta]$, \mathcal{A}^B for $\mathcal{A}^B[\alpha, \beta]$, and R^C for $R^C[\alpha, \beta]$.
2. X for $S_k^C[\alpha, \beta](= S_k^B[\alpha, \beta])$.
3. μ_X for the distribution on $S_k^C[\alpha, \beta]$ given by $[\nu(\pi(G)) \mid \pi \leftarrow \text{Sym}([n] \setminus \{1, \alpha, \beta\})]$.

We have, for all $x \in X$,

$$\begin{aligned} & \Pr[T \text{ wins Game B on } x \mid T \leftarrow \mathcal{A}^B] \\ &= \sum_T \Pr[T \leftarrow \mathcal{A}^B] \mathbb{1}[T \text{ wins Game B on } x] \geq p. \end{aligned} \tag{1.43}$$

Therefore,

$$\sum_{T,x} \Pr[x \leftarrow \mu_X] \Pr[T \leftarrow \mathcal{A}^B] \mathbb{1}[T \text{ wins Game B on } x] \geq p, \tag{1.44}$$

and so there exists a deterministic decision tree T^* in the support of \mathcal{A}^B such that⁴

$$\begin{aligned} & \Pr[T^* \text{ wins Game B on } x \mid x \leftarrow \mu_X] \\ &= \sum_x \Pr[x \leftarrow \mu_X] \mathbb{1}[T^* \text{ wins Game B on } x] \geq p. \end{aligned} \tag{1.45}$$

T^* does not have to belong to R^C (if it does, then the theorem follows immediately).

However, T^* must perform one of two Actions at each step (corresponding to each internal node):

- Action 1. Query the oracle by an element in $\{\alpha, \beta\}$ or some $1 \neq j \in [n]$ that belongs to a set that had been returned by the oracle during a previous instance of Action 1. (Note that this definition is recursive.)

⁴These first steps of the proof can be seen as writing out the “easy direction of Yao’s principle” [Yao77]. Yao’s principle is often used to reduce lower bounding randomized query complexity to lower bounding a distributional variant of deterministic query complexity.

Action 2. Query the oracle by an element in $[n]$ that does not satisfy the above condition.

Let Γ denote an arbitrary depth- $2k$ perfect binary tree on an abstract vertex set $\{\gamma_1, \gamma_2, \dots, \gamma_{2^{2k+1}-1}\}$ with γ_1 being the root. Label γ_1 by 1. Let $n' := 2^{2k+1} - 2$ denote the number of unlabelled vertices of Γ at the outset.

We define \mathcal{A}^C for winning **Game C** using T^* as follows. Initialize set VIEW to $\{1, \alpha, \beta\}$. Also initialize set REAL to $\{1, \alpha, \beta\}$ and set SIM to \emptyset (these will only be used in the analysis of \mathcal{A}^C). Then, \mathcal{A}^C performs one of the following two Actions at each step depending on T^* :

Action I. If T^* performs Action 1 and queries the oracle by $k \in [n]$. Then, \mathcal{A}^C also queries the oracle by k . Let K denote the set of neighbours of k returned by the oracle. \mathcal{A}^C sends K to T^* . Update the set VIEW to $\text{VIEW} \cup K$. Update the set REAL to $\text{REAL} \cup K$.

Action II. If T^* performs Action 2 and queries the oracle by $l \in [n]$. Let $\{\gamma_{i_1}, \dots, \gamma_{i_r}\}$ be the set of unlabelled vertices of Γ . (Note that $r \leq n' - |\text{SIM}|$.) Label $\{\gamma_{i_1}, \dots, \gamma_{i_r}\}$ by the following random procedure:

- (a) γ_{i_1} is labelled by $j_1 \leftarrow [n] \setminus \text{VIEW}$,
- (b) γ_{i_2} is labelled by $j_2 \leftarrow [n] \setminus (\text{VIEW} \cup \{j_1\})$,
- ...
- (c) γ_{i_r} is labelled by $j_r \leftarrow [n] \setminus (\text{VIEW} \cup \{j_1, \dots, j_{r-1}\})$.

Let $V := \{j_1, \dots, j_r\}$. There are two possible cases.

- (a) Case: $l \in V \cup \{1\}$. Let γ_i be the vertex labelled by l in Γ . Let $\text{nbrd}(\gamma_i)$ denote the set of neighbours of γ_i in Γ . Let L denote the set of labels of vertices in $\text{nbrd}(\gamma_i)$. Then \mathcal{A}^C *simulates* the oracle by sending $L \cup \{\alpha, \beta\}$ to T^* if $l = 1$ and sending L to T^* if $l \neq 1$.

Then, delete the labels of vertices in $\{\gamma_{i_1}, \dots, \gamma_{i_r}\} \setminus (\{\gamma_i\} \cup \text{nbnd}(\gamma_i))$.
 Update the set VIEW to $\text{VIEW} \cup \{l\} \cup L$. Update the set SIM to $\text{SIM} \cup \{l\} \cup L$.

(b) Case: $l \notin V \cup \{1\}$. Then, \mathcal{A}^C declares failure and sends $\{\perp\}$ to T^* .

This completes the description of \mathcal{A}^C .

Since \mathcal{A}^C does not really query the oracle in Action II, it satisfies the restriction R^C of **Game C**. Also note that \mathcal{A}^C makes at most d queries.

We proceed to show that \mathcal{A}^C satisfies [Eq. \(1.42\)](#). For the following calculations, it is useful to note that, at each step, $|\text{REAL}| \leq 3 + 2d \leq 5 \cdot 2^k$ (recall $d \leq 2^k$, the coefficient of 5 is loose but convenient), $|\text{SIM}| \leq 4d \leq 5 \cdot 2^k$, and VIEW is the disjoint union of SIM and REAL with $|\text{VIEW}| \leq 3 + 4d \leq 5 \cdot 2^k$.

At each step, one of the following failure events can occur:

1. If \mathcal{A}^C performs Action I at this step, the event E_1 where the set K has non-empty intersection with the set SIM (at that step). We have

$$\Pr(E_1 \mid \Gamma) \leq 2 \cdot \frac{|\text{SIM}|}{n - |\text{REAL}| - 1} \leq 2 \cdot \frac{5 \cdot 2^k}{2^{2k} - 5 \cdot 2^k} = O(2^{-k}), \quad (1.46)$$

where the probability is over the randomness of $[x \mid x \leftarrow \mu_X]$. The first inequality follows since $\Pr(E_1 \mid \Gamma)$ is upper bounded by the probability that a uniformly random size-2 subset of $[n] \setminus \text{REAL}$ intersects SIM. We consider size-2 subsets since K is of size at most 3 but one element in K must belong to REAL (by the definition of Action 1) so it cannot be in SIM. The uniformity of the subset distribution follows from the distribution on inputs.

2. If \mathcal{A}^C performs Action II at this step, the event E_2 where $l \notin V \cup \{1\}$. For $x \in X$,

we have

$$\begin{aligned}
\Pr(E_2 \mid x) &\leq 1 - \frac{n' - |\text{SIM}|}{n - |\text{VIEW}|} \\
&= \frac{2^{k+2} + 1 + |\text{SIM}| - |\text{VIEW}|}{n - |\text{VIEW}|} \\
&\leq \frac{2^{k+2} + 5 \cdot 2^k}{n - 5 \cdot 2^k} = O(2^{-k}),
\end{aligned} \tag{1.47}$$

where the probability is over the randomness of V . The first inequality follows since $\Pr(E_2 \mid x)$ is upper bounded by the probability that a uniformly random size- $(n' - |\text{SIM}|)$ subset of $[n] \setminus \text{VIEW}$ contains a particular element $l \in [n] \setminus \text{VIEW}$.

Now let μ^C denote the distribution on the sequence of sets $\vec{S} = (S_1, \dots, S_d) \in (\mathbb{P}([n]))^d$ that \mathcal{A}^C sends to T^* , where the distribution is over the randomness of V , Γ , and $[x \leftarrow \mu_X]$. Let μ^* denote the distribution on the sequence of sets $\vec{S} = (S_1, \dots, S_d) \in (\mathbb{P}([n]))^d$ that the oracle returns to T^* following the standard computation by T^* , where the distribution is over the randomness of $[x \leftarrow \mu_X]$. Let E denote the event that one of E_1 , E_2 , or E_3 occurs at some step $i \in [d]$ of \mathcal{A}^C . Then, we have

$$\Pr[\vec{S} \mid \vec{S} \leftarrow \mu^C, E^c] = \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^*], \tag{1.48}$$

for all $\vec{S} \in (\mathbb{P}([n]))^d$. This is because, for all steps $i \in [d]$ where \mathcal{A}^C performs Action II, the following distributions on depth- $2k$ binary trees are the same by design:

1. The distribution corresponding to the random labelling of Γ at the start of Action II, conditioned on S_1, \dots, S_{i-1} .
2. The distribution corresponding to taking the depth- $2k$ binary tree subgraph of $\pi(G)$, where $\pi \leftarrow \text{Sym}([n] \setminus \{1, \alpha, \beta\})$, conditioned on S_1, \dots, S_{i-1} .

But [Eqs. \(1.46\)](#) and [\(1.47\)](#) and the union bound (Boole's inequality) imply

$$\epsilon := \Pr[E] \leq O(d \cdot 2^{-k}), \tag{1.49}$$

where the probability is over the randomness of V , Γ , and $[x \leftarrow \mu_X]$.

Therefore, we can bound the total variation distance between μ^C and μ^* by

$$\begin{aligned}
& \text{TVD}(\mu^C, \mu^*) \\
&= \frac{1}{2} \sum_{\vec{S} \in (\mathbb{P}([n]))^d} \left| \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^C] - \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^*] \right| \\
&= \frac{1}{2} \sum_{\vec{S} \in (\mathbb{P}([n]))^d} \left| \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^C, E^c] \Pr[E^c] + \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^C, E] \Pr[E] - \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^*] \right| \\
&\leq \frac{1}{2} \epsilon + \frac{1}{2} \sum_{\vec{S} \in (\mathbb{P}([n]))^d} \left| \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^C, E^c] (1 - \epsilon) - \Pr[\vec{S} \mid \vec{S} \leftarrow \mu^*] \right| \\
&\leq \epsilon, \tag{1.50}
\end{aligned}$$

where the last inequality uses [Eq. \(1.48\)](#).

Say that a sequence of sets $\vec{S} = (S_1, \dots, S_d) \in (\mathbb{P}([n]))^d$ *wins* if there exists an $i \in [d]$ such that $|S_i| \in \{2, 0\}$. Now observe the following:

1. Querying an oracle $x \in X$ by the EXIT or a MARKER of x — which is how a classical algorithm wins [Games B](#) and [C](#) — is equivalent to the oracle returning a set of size 2 or 0, respectively.
2. \mathcal{A}^C can only send T^* a set S with $|S| \in \{2, 0\}$ if it performs Action I where it queries the oracle (if it performs Action II, $|S| \in \{1, 3, 4\}$).

Therefore

$$\begin{aligned}
& \Pr[T \text{ wins } \text{Game C} \text{ on } x \mid x \leftarrow \mu_X, T \leftarrow \mathcal{A}^C] \\
&= \Pr[\vec{S} \text{ wins} \mid \vec{S} \leftarrow \mu^C] \\
&\geq \Pr[\vec{S} \text{ wins} \mid \vec{S} \leftarrow \mu^*] - \text{TVD}(\mu^C, \mu^*) \\
&\geq \Pr[\vec{S} \text{ wins} \mid \vec{S} \leftarrow \mu^*] - \epsilon \tag{Eq. (1.50)} \\
&= \Pr[T^* \text{ wins } \text{Game B} \text{ on } x \mid x \leftarrow \mu_X] - \epsilon \\
&\geq p - O(d \cdot 2^{-k}), \tag{Eqs. (1.45) and (1.49)}
\end{aligned}$$

which completes the proof. \square

Now consider:

Game D.

1. The oracle set $S_k^D[\alpha, \beta] := S_k^C[\alpha, \beta]$ is the same as that in **Game C**.
2. The winning condition $W_D: S_k^D \rightarrow \mathbb{P}([n])$ is defined by $W_D(\nu(G))$ is equal to the singleton set containing the EXIT of G .
3. The restriction $R^D[\alpha, \beta] := R^C[\alpha, \beta]$ is the same as that in **Game C**.

To win **Game D**, an algorithm needs to find the EXIT. To win **Game C**, an algorithm only needs to find either the EXIT or a MARKER. However, since the algorithm in both cases is restricted to querying vertices connected to ENTRANCE, we can show that winning **Game C** implies winning **Game D**. Formally, we have

Lemma 6. *Suppose there is a randomized decision tree $\mathcal{A}^C[\alpha, \beta]$ of depth d that, for any graph $G \in \tilde{S}_k^B[\alpha, \beta]$, satisfies*

$$\Pr[T \text{ wins Game C on } \nu(\pi(G)) \mid \pi \leftarrow \text{Sym}([n] \setminus \{1, \alpha, \beta\}), T \leftarrow \mathcal{A}^C[\alpha, \beta]] \geq p,$$

where $\pi(G)$ is the same as G but with its vertices permuted by π . Then, there is a randomized decision tree $\mathcal{A}^D[\alpha, \beta]$ of depth d supported on deterministic decision trees in $R^C[\alpha, \beta]$ that, for any graph $G \in \tilde{S}_k^B[\alpha, \beta]$, satisfies

$$\Pr[T \text{ wins Game D on } \nu(\pi(G)) \mid \pi \leftarrow \text{Sym}([n] \setminus \{1, \alpha, \beta\}), T \leftarrow \mathcal{A}^D[\alpha, \beta]] \geq p.$$

Proof. Let $G \in \tilde{S}_k^B[\alpha, \beta]$, $\pi \in \text{Sym}([n] - \{1, \alpha, \beta\})$, and $T \in R^C[\alpha, \beta]$. Suppose T wins **Game C** on $\nu(\pi(G))$ so that it queries either the EXIT or a MARKER of $\pi(G)$. By the definition of $R^C[\alpha, \beta]$ and $\tilde{S}_k^B[\alpha, \beta]$, the vertices queried by T form a connected subgraph of $\pi(G)$ that includes vertex 1, i.e., the ENTRANCE of $\pi(G)$. Therefore, T

cannot query a MARKER of $\pi(G)$ because $\pi(G) \in \tilde{S}_k^B[\alpha, \beta] \subseteq \tilde{X}_k$ so any MARKER of $\pi(G)$ is disconnected from the ENTRANCE of $\pi(G)$. Therefore, T must query the EXIT of $\pi(G)$ and so win **Game D**. Therefore, the lemma follows by setting $\mathcal{A}^D[\alpha, \beta]$ to be equal to $\mathcal{A}^C[\alpha, \beta]$. \square

Now, we show that **Game D** is essentially the same as **Game E**, defined below.

Game E (Game 1 of [Chi+03], STOC version).

1. Oracle set $S_k^E := \{\nu(G) \mid G \in \tilde{S}_k^E\}$, where \tilde{S}_k^E consists of graphs $G \in \tilde{Z}_k$ where vertex 1 has degree 2.
2. Winning condition $W^E: S_k^E \rightarrow \mathbb{P}(n)$ is defined by $W^E(\nu(G))$ is equal to the singleton set $\{v\}$, where $v \neq 1$ and v has degree 2 in G .

Lemma 7. Suppose for any $\alpha, \beta \in [n]$ with $\alpha, \beta \neq 1$, there is a randomized decision tree $\mathcal{A}^D[\alpha, \beta]$ of depth d that, for any graph $H \in \tilde{S}_k^B[\alpha, \beta]$, satisfies

$$\Pr[T \text{ wins Game D on } \nu(\pi(H)) \mid \pi \leftarrow \text{Sym}([n] \setminus \{1, \alpha, \beta\}), T \leftarrow \mathcal{A}^D[\alpha, \beta]] \geq p,$$

where $\pi(G)$ is the same as G but with its vertices permuted by π . Then, there is a randomized decision tree \mathcal{A}^E of depth d that, for any graph $G \in \tilde{S}_k^E$, satisfies

$$\Pr[T \text{ wins Game E on } \nu(\sigma(G)) \mid \sigma \leftarrow \text{Sym}(\{2, \dots, n\}), T \leftarrow \mathcal{A}^E] \geq p.$$

Proof. We construct \mathcal{A}^E as follows. First \mathcal{A}^E queries the oracle by vertex 1. Suppose the oracle returns $\alpha, \beta \in [n] \setminus \{1\}$ with $\alpha \neq \beta$. Then \mathcal{A}^E runs $\mathcal{A}^D[\alpha, \beta]$. Clearly, the depth of \mathcal{A}^E is $d + 1$.

Now take $G \in \tilde{S}_k^E$. For $\alpha, \beta \in [n] \setminus 1$, we introduce the following notation:

1. Let $\mathcal{G}[\alpha, \beta]$ denote the set of graphs on vertex set $[n]$ such that vertex 1 has neighbours α and β .

2. Let $G[\alpha, \beta]$ denote an arbitrary graph that is isomorphic to G (i.e., the same as G after vertex relabelling) such that $G[\alpha, \beta] \in \mathcal{G}[\alpha, \beta]$.
3. Suppose $G[\alpha, \beta]$ is formed by a glued-trees graph $G'[\alpha, \beta]$ and a set of $2^{2k+1} + 1$ isolated vertices. Let $H[\alpha, \beta] \in \tilde{S}_k^B[\alpha, \beta]$ denote an arbitrary type-0 modified glued-trees graph that can be constructed using $G'[\alpha, \beta]$.

We also write c for $\frac{1}{\binom{n-1}{2}}$, Sym^E for $\text{Sym}(\{2, \dots, n\})$, and $\text{Sym}^D[\alpha, \beta]$ for $\text{Sym}([n] - \{1, \alpha, \beta\})$ for notational convenience.

Then

$$\begin{aligned}
& \Pr[T \text{ wins Game E on } \nu(\sigma(G)) \mid \sigma \leftarrow \text{Sym}^E, T \leftarrow \mathcal{A}^E] \\
&= \sum_{\{\alpha, \beta\}} \Pr[T \text{ wins Game E on } \nu(\sigma(G)) \mid \sigma(G) \in \mathcal{G}[\alpha, \beta], \sigma \leftarrow \text{Sym}^E, T \leftarrow \mathcal{A}^E] \\
&\quad \cdot \Pr[\sigma(G) \in \mathcal{G}[\alpha, \beta] \mid \sigma \leftarrow \text{Sym}^E] \\
&= c \sum_{\{\alpha, \beta\}} \Pr[T \text{ wins Game E on } \nu(\pi(G[\alpha, \beta])) \mid \pi \leftarrow \text{Sym}^D[\alpha, \beta], T \leftarrow \mathcal{A}^E] \\
&= c \sum_{\{\alpha, \beta\}} \Pr[T \text{ wins Game E on } \nu(\pi(G[\alpha, \beta])) \mid \pi \leftarrow \text{Sym}^D[\alpha, \beta], T \leftarrow \mathcal{A}^D[\alpha, \beta]] \\
&= c \sum_{\{\alpha, \beta\}} \Pr[T \text{ wins Game E on } \nu(\pi(H[\alpha, \beta])) \mid \pi \leftarrow \text{Sym}^D[\alpha, \beta], T \leftarrow \mathcal{A}^D[\alpha, \beta]] \\
&\geq c \sum_{\{\alpha, \beta\}} \Pr[T \text{ wins Game D on } \nu(\pi(H[\alpha, \beta])) \mid \pi \leftarrow \text{Sym}^D[\alpha, \beta], T \leftarrow \mathcal{A}^D[\alpha, \beta]] \\
&\geq c \sum_{\{\alpha, \beta\}} p = p.
\end{aligned}$$

Note that the fourth equality uses the fact that $\mathcal{A}^D[\alpha, \beta]$ is supported on deterministic decision trees in $R^D[\alpha, \beta]$ which have the same behaviour on $\nu(\pi(G[\alpha, \beta]))$ and $\nu(\pi(H[\alpha, \beta]))$. \square

With the above lemmas in place, we finally prove **Theorem 3**.

Proof of Theorem 3. From the sequence of lemmas above (**Lemmas 3 to 7**), we can

obtain a contradiction to $R(\mathcal{P}_k) \leq 2^{k/6} - 1$ as follows.

If there is a randomized decision tree of depth $d \leq 2^{k/6} - 1$ that, on each input $x \in S_k$, decides $\mathcal{P}_k(x)$ correctly with probability at least $2/3$, then the lemmas above imply that there is a randomized decision tree \mathcal{A} of depth $d' \leq d + 1 \leq 2^{k/6}$ such that, for any graph $G \in \tilde{S}_k^E$,

$$\begin{aligned} \Pr_G^E[\mathcal{A}] &:= \Pr[T \text{ wins Game E on } \nu(\sigma(G)) \mid \sigma \leftarrow \text{Sym}(\{2, \dots, n\}), T \leftarrow \mathcal{A}] \\ &\geq \frac{1}{3} - O(d \cdot 2^{-k}) \geq \frac{1}{3} - O(2^{-5k/6}). \end{aligned} \tag{1.51}$$

But [Chi+03, Theorem 6 (STOC version)]⁵ states that if $d' \leq 2^{k/6}$, then there exists a distribution μ on \tilde{S}_k^E such that

$$\mathbb{E}_{G \leftarrow \mu}[\Pr_G^E(\mathcal{A})] \leq O(2^{-k/6}). \tag{1.52}$$

Since Eq. (1.52) contradicts Eq. (1.51), we conclude that $R(\mathcal{P}_k) \geq \Omega(2^{k/6})$. \square

On the other hand, we show that \mathcal{P}_k can be solved by a quantum algorithm using $\text{poly}(k)$ queries.

Theorem 4. *The quantum query complexity of $\mathcal{P}_k: S_k \rightarrow \{0, 1\}$ satisfies $Q(\mathcal{P}_k) \leq \text{poly}(k)$.*

Proof. We describe a two-stage quantum algorithm that estimates \mathcal{P}_k using $\text{poly}(k)$ queries.

Stage 1. Keep querying a vertex uniformly at random until we hit a POINTER (distinguished by having degree 1). This occurs with high probability after a constant number of queries because the probability of querying a POINTER is

$$\frac{2^{2k}}{2 \cdot 2^{2k} + 2(2^{k+1} - 2)}, \tag{1.53}$$

⁵The following is a more precise statement of [Chi+03, Theorem 6 (STOC version)] that can be seen from its proof.

which is ≥ 0.49 for all $k \geq 10$. Then, we perform a classical random walk from the POINTER (never walking backwards to the previously queried vertex). If after k steps we reach another POINTER, we know that we made the wrong turn at the $(k/2)$ -th step (k must be even). So we simply proceed by taking the correct turn at the $(k/2)$ -th step. Continuing similarly, we can reach ENTRANCE (distinguished by having degree 4) after $O(k^2)$ queries [Chi+03]. Upon reaching ENTRANCE from one direction, we walk $2k$ steps in each of the three other directions (again never walking backwards). One of these directions will lead to another POINTER and we can eliminate that direction as a direction to reach EXIT. This concludes the first stage.

Stage 2. After the first stage, the problem becomes essentially the same as the original glued trees problem. Therefore, in the second stage, we run the quantum walk algorithm of [Chi+03]. This algorithm is able to query EXIT with high probability after $\text{poly}(k)$ queries. Note that EXIT is distinguished by having degree 2 or 5 when the input is in set X_k or Y_k , respectively.

Since this algorithm finds the EXIT using only $\text{poly}(k)$ queries, the result follows. \square

1.5 Discussion and open problems

We have shown that graph properties do not admit an exponential quantum query speedup in the adjacency matrix model, but that there is a graph property with exponential quantum speedup in the adjacency list model.

We emphasize that the result described in this chapter does not address the question of whether there can be an exponential separation for graph *property testing* in the adjacency list model. In property testing, the set of “no” inputs must include all

those that are ϵ -far away from the “yes” inputs. Our example in [Section 1.4](#) does not satisfy this condition.

This question is resolved affirmatively in follow-up work [\[Ben+20\]](#) by upgrading the example in [Section 1.4](#). The starting observation for the upgrade is that the entire structure of a modified glued-trees graph G can be efficiently tested by a randomized algorithm *if* the algorithm could efficiently decide whether a given vertex $v \in [n]$ is a “cycle vertex”, meaning that it lies on the cycle that defines the glued-trees subgraph of G . This is because the problem then reduces to testing perfect binary trees and testing cycles, for which there are efficient randomized algorithms. Then the key idea is to hide the information about whether a vertex is a cycle vertex or not in a way that can be efficiently found by a quantum algorithm but cannot be efficiently found by any randomized algorithm. This hiding is implemented using further copies of the modified glued-trees graph in a way that is reminiscent of the “pointer-functions” or “cheat-sheets” idea in [\[Amb+17; ABK16\]](#).

Another question is to more generally characterize how the quantum speedup admitted by a function $f: E \subseteq \Sigma^m \rightarrow \{0, 1\}$ relate to its symmetries, or more precisely, its automorphism group

$$\text{Aut}(f) := \{\sigma \in S_m \mid x \circ \sigma \in E \text{ and } f(x) = f(x \circ \sigma) \text{ for all } x \in E\}. \quad (1.54)$$

This question is addressed in follow-up work [\[Ben+20\]](#) where we show a near-complete characterization of the relationship between $\text{Aut}(f)$ and the quantum speedup admitted by f .

First, we show that when $H \leq S_m$ is a primitive subgroup⁶ of S_m with very small base size (in terms of m)⁷, then there exists an f with $\text{Aut}(f) = H$ such that f admits

⁶A subgroup $H \leq S_m$ is primitive if it is transitive and the only partitions $\mathcal{B} := \{B_1, \dots, B_k\}$ of $[m]$ preserved by H , i.e., $\{\pi(B_i) \mid i \in [k]\} = \mathcal{B}$ for all $\pi \in H$, are $\mathcal{B} = \{\{1\}, \dots, \{m\}\}$ and $\mathcal{B} = \{[m]\}$.

⁷A base of a subgroup $H \leq S_m$ is a set $B \subseteq [m]$ such that if $\pi \in H$ and $\pi(x) = x$ for all $x \in B$ then π is the identity element in H . The base size $b(H)$ of H is the minimal size of a base.

a massive quantum speedup (in the parameter m). But Liebeck [Lie84] shows that a primitive subgroup of S_m either (i) has very small base size or (ii) corresponds to the symmetries of a hypergraph. Therefore, given a primitive subgroup $H \leq S_m$, there are two possible cases:

1. H has very small base size. In this case, there exists an f with $\text{Aut}(f) = H$ such that the f admits a massive quantum speedup.
2. H corresponds to the symmetries of a hypergraph. In this case, for all f with $\text{Aut}(f) = H$, f admits a modest quantum speedup (as shown in this chapter).

The above constitutes a complete characterization of the relationship between primitive subgroups of S_m and quantum speedups. Using the fact that any subgroup of S_m can be “built from” primitive groups, we obtain a near-complete characterization of the relationship between arbitrary subgroups of S_m and quantum speedups, see [Ben+20, Corollary 43].

Despite the significant progress mentioned above, there remain several interesting open questions related to this chapter. We conclude by discussing them.

1. What is the largest k such that there exists an l -uniform (adjacency matrix) hypergraph property \mathcal{P} with $R(\mathcal{P}) = \Omega(Q(\mathcal{P})^k)$? Note that we showed $k \leq 3l$. We remark that the problem is open even for the case $l = 1$ of fully symmetric functions, where the best upper bound is $k \leq 3$ due to Chailloux [Cha18], and the best lower bound is $k \geq 2$ for the OR function [Gro96]. For larger l , it is at least possible to exhibit functions with a power separation of $k \geq l/2$ as explained in [Ben+20, Section 7 of arXiv v1].
2. Do similar results hold for “domain-symmetric” functions? We say a function $f: E \subseteq \Sigma^m \rightarrow \{0, 1\}$ is domain-symmetric under a subgroup $H \leq S_m$ if $x \circ \sigma \in E$ for all $\sigma \in H$ and $x \in E$. A symmetric function is domain-symmetric but the converse is clearly false. Nevertheless, [Ben16] shows that when f is domain-

symmetric under S_m , we have $R(f) = O(Q(f)^8 |\Sigma|^8 \log(|\Sigma|) + Q(f)^{12(|\Sigma|-1)})$, which is $O(Q(f)^{12(|\Sigma|-1)})$ if $|\Sigma|$ is constant.

3. Does there exist a super-polynomial quantum speedup for solving a *useful* graph problem in the adjacency list model? For example, it remains open whether there could be an exponential quantum speedup for testing bipartiteness [ACL11]. [LSZ21] shows that super-polynomial quantum speedups do exist for solving useful graph problems in the so-called *cut-query* model. In that model, we query an oracle encoding a graph G by a subset S of vertices and the oracle responds by returning the number of edges of G that have exactly one endpoint in S .

Chapter 2

Quantum divide and conquer

Chapter summary. The divide-and-conquer framework, used extensively in classical algorithm design, recursively breaks a problem of size n into smaller subproblems (say, a copies of size n/b each), along with some auxiliary work of cost $C^{\text{aux}}(n)$, to give a recurrence relation

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

for the classical complexity $C(n)$. We describe a quantum divide-and-conquer framework that, in certain cases, yields an analogous recurrence relation

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

that characterizes the quantum query complexity. We apply this framework to obtain near-optimal quantum query complexities for various string problems, such as (i) recognizing regular languages; (ii) decision versions of String Rotation and String Suffix; and natural parameterized versions of (iii) Longest Increasing Subsequence and (iv) Longest Common Subsequence.

2.1 Introduction

Classically, divide and conquer is a basic algorithmic framework that applies widely to many problems (see, e.g., [Hoa62; KO63; CT65; Str69; CW90; CLRS09; AW21]). This technique solves a problem by recursively solving smaller instances of the same problem together with some auxiliary problem. In this chapter, we develop a quantum analog of this framework that can provide quantum query complexity speedups relative to classical computation.

To motivate this investigation, consider a simple application of classical divide and conquer. Consider the problem of computing the OR function on n bits. The classical deterministic query complexity of this problem is well known to be n , since an algorithm must read all n bits in the worst case to check if any of the bits equals 1. This upper bound is trivial since any query problem on n bits can be solved by querying all n bits, but to illustrate the quantum idea, we first consider re-deriving this classical upper bound using divide and conquer.

It is easy to see that for an n -bit string x , $\text{OR}(x) = 1$ if and only if either $\text{OR}(x_{\text{left}}) = 1$ or $\text{OR}(x_{\text{right}}) = 1$, where x_{left} and x_{right} are the left and right halves of x . Thus we have divided the original problem into two smaller instances of the same problem, and given solutions to the two smaller instances, no further queries need to be made to solve the larger instance. Letting $C(n)$ denote the classical query complexity of this problem, this argument yields the recurrence relation

$$C(n) \leq 2C(n/2), \tag{2.1}$$

which, noting $C(1) = 1$, solves to $C(n) \leq n$, as desired.

Now notice that the last step that combines the solutions of the smaller instances to solve the larger instance is also an OR function, but on 2 bits, since $\text{OR}(x) = \text{OR}(x_{\text{left}}) \vee \text{OR}(x_{\text{right}})$. We know that the quantum query complexity of the OR function

on n bits is $O(\sqrt{n})$ due to Grover’s algorithm [Gro96], so we might be tempted to say that the OR of 2 bits should be faster to compute, potentially even quadratically faster. Thus we might like to write the following recurrence relation for the quantum query complexity $Q(n)$ of computing the OR function:

$$Q(n) \leq \sqrt{2} Q(n/2). \tag{2.2}$$

Here the quotes around the “ $\sqrt{2}$ ” convey that this is not really justified. First, the complexity of Grover’s algorithm is not exactly \sqrt{n} , but only \sqrt{n} up to a constant factor. Even if it were exactly \sqrt{n} , Grover’s algorithm is a bounded-error algorithm, so if used as a subroutine that calls itself many times, the error will become unbounded very quickly. And of course, it does not make sense to imagine that the query complexity is a non-integer.

Even though we have just argued that Equation (2.2) is questionable, if we solve this recurrence anyway, we find $Q(n) \leq \sqrt{n}$.¹ Up to a multiplicative constant, this is the correct answer for the quantum query complexity of the OR function on n bits!

This is not a coincidence. As we show in this chapter, this can be seen as a simple instance of a quantum algorithmic framework that we call “quantum divide and conquer.” More generally, a classical divide-and-conquer algorithm typically yields a recurrence relation of the form

$$C(n) \leq a C(n/b) + C^{\text{aux}}(n), \tag{2.3}$$

where $C^{\text{aux}}(n)$ is the classical query complexity of the auxiliary problem.

The main conceptual takeaway of this chapter is that, in many settings, the quantum query complexity equals (up to a multiplicative constant) the solution, $C_Q(n)$,

¹This may be familiar to readers who know the quantum adversary method. However, our divide-and-conquer framework generalizes this phenomenon to a scenario that cannot be captured using adversary machinery alone, as discussed below.

of the analogous recurrence relation,

$$C_Q(n) \leq \sqrt{a} C_Q(n/b) + C_Q^{\text{aux}}(n), \quad (2.4)$$

where the quantum query complexity of the auxiliary problem is $O(C_Q^{\text{aux}}(n))$. We show that this framework easily recovers and improves non-trivial quantum speedups that were previously known. Furthermore, we show it yields previously unknown quantum speedups that do not seem to be achievable by standard applications of techniques such as Grover search [Gro96], amplitude amplification and estimation [BHMT02], and quantum walk search [Amb07].

This chapter is organized as follows. We first introduce background material and describe how to use the quantum divide-and-conquer framework for certain scenarios in Section 2.3. Then we apply the framework to four string problems in Section 2.4, obtaining near-optimal quantum query complexities for each. Finally, we conclude by discussing some open problems in Section 2.5.

For the remainder of the introduction, we briefly overview our results, the techniques involved in the quantum divide and conquer framework, and related work.

Results. We apply quantum divide and conquer to the following decision problems, which we have ordered in roughly increasing difficulty. In all of these problems, we are given query access to a string $s \in \Sigma^a$ over some set Σ , i.e., query access to a function $f: \{1, \dots, a\} \rightarrow \Sigma$, with $f(i) = s_i$.

1. Recognizing regular languages. We recover the key algorithmic result in [AGS19] that is used to upper bound the quantum query complexity of recognizing star-free languages. (The latter result is, in turn, the key result used by [AGS19] to establish a quantum query complexity trichotomy for recognizing regular languages.) In particular, we show that $O(\sqrt{n \log n})$ quantum queries suffice to decide whether a string $x \in \{0, 1, 2\}^n$ contains a substring of the form 20^*2 , i.e., two 2s on either side

of an arbitrarily long string of 0s. Having established our framework, the analysis is simpler than that of [AGS19].

2. **String Rotation and String Suffix.** Let $x \in \Sigma^n$ and $i \in \{1, \dots, n\}$. In String Rotation, the problem is to decide whether the lexicographically minimal rotation of x starts at i . In String Suffix, the problem is to decide whether the lexicographically minimal suffix of x starts at i . These problems are natural decision versions of problems studied in [AJ23]. We obtain an $\tilde{O}(\sqrt{n})$ upper bound on the quantum query complexity for these problems, improving the previous best bound of $O(n^{1/2+o(1)})$, which follows from the algorithms of [AJ23] for the search versions. Furthermore, the analysis is again simpler using our framework.
3. **k -Increasing Subsequence (k -IS).** Given a string $x \in \Sigma^n$ over some ordered set Σ and an integer $k \geq 1$, the k -IS problem asks us to decide whether x has an increasing subsequence² of length at least k . We obtain a bound of $\tilde{O}(\sqrt{n})$ for any constant k . The k -IS problem is a natural parameterized version of Longest Increasing Subsequence (LIS), which is well-studied classically (see, e.g., [Fre75; AD99; SS10; RSSS19]). We consider this version because the quantum query complexity of LIS is easily seen to be³ $\Omega(n)$, so we cannot hope for a quantum speedup without a bound on k . Somewhat surprisingly, any constant bound on k allows for a quadratic speedup over classical algorithms.
4. **k -Common Subsequence (k -CS).** Given two strings $x, y \in \Sigma^n$ over some set Σ and integer $k \geq 1$, the k -CS problem asks us to decide whether x and y share a common subsequence of length at least k . We obtain a bound of $\tilde{O}(n^{2/3})$ for any constant k . The k -CS problem is a natural parametrized version of Longest

²Recall that a subsequence of a string is obtained by taking a subset of the elements without changing their order. Note that this is more general than a *substring*, in which the chosen elements must be consecutive.

³This follows by reduction from a threshold function such as majority [Bea+01]. Given a string $z \in \{0, 1\}^n$, define $x \in \{0, 1, \dots, n\}^n$ such that $x_i = iz_i$ for all $i \in \{1, \dots, n\}$. Then we can determine the Hamming weight $|z|$ of z by classically querying z_1 and adding $z_1 - 1$ to the length of the LIS of x .

Common Subsequence (LCS), which is well-studied classically (see, e.g., [WF74; LMS98; CLRS09; AKO10; ABW15; RSSS19]). Again, we consider this version because the quantum query complexity of LCS is easily seen to be⁴ $\Omega(n)$. Note that LCS is closely connected to Edit Distance (ED), which asks us to compute the number of insertions, deletions, and substitutions required to convert x into y . In fact, if substitutions were disallowed, the edit distance between x and y would equal $\text{ED}(x, y) = 2n - 2 \text{LCS}(x, y)$ (a proof can be found in [AV21, Lemma 6]).

All of our quantum query complexity upper bounds are tight up to logarithmic factors and represent quantum-over-classical speedups since the classical (randomized) query complexities of all problems considered are $\Omega(n)$.

Techniques. To derive the quantum recurrence relation in Equation (2.4), we employ the quantum adversary method as an upper bound on quantum query complexity [Rei11; Lee+11]. The adversary quantity $\text{Adv}(P(n))$ is a real number that can be associated with any query problem $P(n)$ of size n . For convenience, we write $\text{Adv}(n) := \text{Adv}(P(n))$ when P is clear from context. It is well-known that $\text{Adv}(n)$ is upper and lower bounded by the (two-sided bounded error) quantum query complexity, $Q(n)$, of $P(n)$ up to multiplicative constants.

If a problem can be expressed as the composition of smaller problems, then the adversary quantity of the larger problem can be expressed in terms of those of the smaller problems [Rei11; Lee+11]. For example, suppose $P(n)$ is the OR of a copies of $P(n/b)$ together with another problem $P^{\text{aux}}(n)$, corresponding to the auxiliary work performed in a divide-and-conquer algorithm. Then the composition theorem implies

$$\text{Adv}(n)^2 \leq a \text{Adv}(n/b)^2 + \text{Adv}(P^{\text{aux}}(n))^2. \quad (2.5)$$

⁴This also follows by reduction from a threshold function such as majority. Given a string $z \in \{0, 1\}^n$, define $x = 1^n$. Then we can determine the Hamming weight $|z|$ of z as the length of the longest common subsequence of z and x .

(For a more precise statement, see [Section 2.3](#).) Taking square roots⁵ and using the fact that $\text{Adv}(P^{\text{aux}}(n))$ is bounded by $O(Q^{\text{aux}}(n))$, where $Q^{\text{aux}}(n) := Q(P^{\text{aux}}(n))$, we obtain precisely [Equation \(2.4\)](#), i.e.,

$$\text{Adv}(n) \leq \sqrt{a} \text{Adv}(n/b) + O(Q^{\text{aux}}(n)). \quad (2.6)$$

We then bound $Q^{\text{aux}}(n)$ by constructing an explicit quantum algorithm for the auxiliary work and bounding its quantum query complexity. Finally, we solve [Equation \(2.6\)](#) either directly or by appealing to the Master Theorem [[BHS80](#)] to obtain a bound on $\text{Adv}(n)$ and hence a bound on $Q(n)$. Note that this bound is insensitive (up to a multiplicative constant) to the constant implicit in $O(Q^{\text{aux}}(n))$, whereas it is highly sensitive to the constant in front of $\text{Adv}(n/b)$.

We emphasize that the recurrence in [Equation \(2.6\)](#) involves *both* the adversary quantity and the quantum query complexity. At a high level, this is why the bound on $Q(n)$ resulting from solving [Equation \(2.6\)](#) does not follow directly from adversary techniques alone nor from quantum algorithmic techniques alone. Indeed, when we solve [Equation \(2.6\)](#) as a recurrence relation, we mix adversary and algorithmic techniques at each step of the recursion. In principle, it is possible to use [Equation \(2.6\)](#) to construct an explicit quantum algorithm for $P(n)$ because there is a constructive, complexity-preserving, and two-way correspondence between span programs—whose complexity is characterized by the adversary quantity—and quantum algorithms [[Rei09a](#); [CJOP20](#); [Jef22](#)]. However, this correspondence is involved.

This quantum divide-and-conquer framework allows us to use *classical* divide-and-conquer thinking to come up with a classical recurrence relation that we can easily convert to a corresponding quantum recurrence relation in the form of [Equation \(2.6\)](#). Bounding $Q(n)$ then reduces to bounding $Q^{\text{aux}}(n)$, which can be easier

⁵Note that while taking square roots gives a closer analogy with the classical recurrence, in some cases we can get a slightly tighter bound by instead directly solving the recurrence for $\text{Adv}(n)^2$.

than the original problem. In our applications to k -IS and k -CS, we bound $Q^{\text{aux}}(n)$ by using quantum algorithms for the $i < k$ versions of these problems, whose query complexities we can bound inductively. Note that the base cases ($k = 1$) of these problems are trivial, being equivalent to search and (bipartite) element distinctness, respectively.

The form of [Equation \(2.6\)](#) depends on the way we divide and conquer. Strikingly, in our application to k -CS, we find that an optimal (up to logarithmic factors) quantum query complexity can be derived by breaking the problem into *seven* parts (so that $b = 7$) but not fewer. Classically, the number of parts we break the problem into makes no difference as any choice leads to an $O(n)$ classical query complexity.

In the discussion above, we have assumed that the OR function relates $P(n)$ to $P(n/b)$ and $P^{\text{aux}}(n)$. However, this is only to simplify our exposition. In fact, we can use quantum divide and conquer for *any* function relating $P(n)$ to $P(n/b)$ and $P^{\text{aux}}(n)$. Functions that we use to obtain quantum speedups include combinations of OR and AND. In our application of quantum divide and conquer to k -CS, the function is a combination of SWITCH-CASE and OR. While SWITCH-CASE alone yields no quantum speedup, our analysis relies on an adversary composition theorem that we prove, which allows us to preserve the speedup yielded by OR. We remark that there are many other functions that could yield quantum speedups, but which we have not yet considered in applications. Examples include EQUAL (which decides if the subproblems all return the same answer or not) and EXACT_{2 of 3} (which decides if exactly 2 of 3 subproblems return 1)—see [[RŠ12](#), Table 1]—and their combinations with OR, AND, and SWITCH-CASE. We leave the consideration of such functions for future work.

Related work. The technique of using adversary composition theorems (and their relatives) to establish upper bounds on quantum query complexity has been applied

in several previous works, beginning with the setting of formula evaluation [Rei09b; RŠ12]. In [Kim13], an adversary composition theorem is used to bound the quantum query complexity of a function f given a bound on that of f composed with itself d times. More generally, the adversary quantity has been used to upper bound the quantum query complexity of problems including k -distinctness [Bel12a], triangle finding [Bel12b; LMS17], undirected st -connectivity [BR12], and learning symmetric juntas [Bel14]. In contrast to our work, these prior results typically bound the adversary quantity using only tools from the adversary world, such as span programs, semidefinite programming, and composition theorems, without constructing quantum algorithms to use as subroutines.

The first two application areas of quantum divide and conquer that we consider (recognizing regular languages and String Rotation and String Suffix) arise in [AGS19; AJ23] as discussed above. Turning to k -IS and k -CS, we first note that, despite the importance of these problems, there have been no significant works on their quantum complexities. Directly using Grover search [Gro96] over all possible positions of a 1-witness gives trivial $O(n)$ bounds as soon as $k \geq 2$. Directly using quantum walk search [Amb07] also yields highly sub-optimal bounds of $O(n^{k/(k+1)})$ for k -IS and $O(n^{2k/(2k+1)})$ for k -CS.

Classically, LIS and LCS are typically solved using dynamic programming. There are many recent works on quantum speedups for dynamic programming, e.g., [Amb+19; Abb19; GKMV21; Amb+20; KPV22]. The main idea of these works is to *classically* query a part of the input and to repeatedly use the result together with a quantum algorithm to solve many overlapping subproblems. However, we found it difficult to apply this idea to k -IS and k -CS.

Given the generality of the results in [AGS19] and the fact that we can recover some of its results, one might ask if the converse holds, i.e., if [AGS19] can recover our

results on k -IS and k -CS. Indeed, the results of [AGS19] do apply⁶ to k -IS and k -CS when the set size $|\Sigma|$ is *constant*. However, under that assumption, these problems can be easily solved by Grover search [Gro96] and quantum minimum finding [DH96] using $O(\sqrt{n})$ queries. Moreover, k -CS becomes qualitatively easier when $|\Sigma|$ is constant as its complexity drops to $O(\sqrt{n})$, down from the $\tilde{\Theta}(n^{2/3})$ bound that we establish when $|\Sigma|$ is unbounded. Note that $\tilde{\Theta}(n^{2/3})$ is not in the trichotomy of [AGS19], i.e., not one of $\Theta(1)$, $\tilde{\Theta}(\sqrt{n})$, or $\Theta(n)$.

2.2 Preliminaries

\mathbb{N} denotes the positive integers. We reserve $m, n \in \mathbb{N}$ for positive integers and Σ for a finite set. For a set X , we write $X^{m \times n}$ for the set of $m \times n$ matrices with entries in X . We write $[m] := \{1, \dots, m\}$. For any $A \in \mathbb{C}^{m \times n}$, $(A)_{ij}$ denotes the element in the i th row and j th column. $\|A\|$ denotes the spectral norm of A , i.e., the largest singular value of A . A^\top denotes the transpose of A . For any $A, B \in \mathbb{C}^{m \times m}$ of the same size, $A \circ B$ denotes their entrywise product, also known as their Hadamard product. For a function $f: D \rightarrow E$, let its Gram matrix F be defined as $(F)_{xy} := \delta_{f(x), f(y)}$ for all $x, y \in D$, where δ is the Kronecker delta function. For a length- n string $x \in \Sigma^n$, we write $x[i]$ for the i th element of x and $x[i..j]$ for the substring of x that ranges from its i th to j th elements (inclusive). We write $x(i..j) := x[i+1..j]$ and $x[i..j) := x[i..j-1]$. For $x \in \Sigma^m$ and $y \in \Sigma^n$, we write $\text{lcp}(x, y) := \max\{j: j \leq \min\{|x|, |y|\}, x[1..j] = y[1..j]\}$ for the length of their longest common prefix. When Σ is equipped with a total order, we say x is lexicographically smaller than y (denoted $x < y$) if either x is a proper prefix of y (i.e., $|x| < |y|$ and $x = y[1..|x|]$), or $l = \text{lcp}(x, y) < \min\{|x|, |y|\}$ and $x[l+1] < y[l+1]$. The notation $x > y$, $x \leq y$, and

⁶For example, k -IS can be expressed in first-order logic as $\bigvee_{a_1 < a_2 < \dots < a_k} \exists i_1, \dots, i_k (i_1 < \dots < i_k) \bigwedge_{j=1}^k \pi_{a_j}(i_j)$ where $\pi_a(i)$ indicates that the i th element is a . Expressibility in first-order logic is equivalent to being in a star-free language, which can be recognized with quantum query complexity $\tilde{\Theta}(\sqrt{n})$ by [AGS19].

$x \geq y$ are defined analogously.

2.3 Framework

In this section we explain how to apply the quantum divide-and-conquer framework in specific scenarios.

We write $Q(f)$ for $Q_{1/3}(f)$ (the quantum query complexity of f with failure probability at most $1/3$). A closely related quantity that is also used widely, and serves as both a lower and upper bound for the quantum query complexity of function computation, is the adversary quantity, $\text{Adv}(\cdot)$.

Definition 7 (Adversary quantity). *Let D , E , and Σ be finite sets and $n \in \mathbb{N}$ with $D \subseteq \Sigma^n$. Let $f: D \rightarrow E$ be a function with Gram matrix F and $\Delta = \{\Delta_1, \dots, \Delta_n\}$ with $(\Delta_j)_{x,y \in D} = 1 - \delta_{x_j, y_j}$. Then the adversary quantity for f ,*

$$\text{Adv}(f) := \min_{\Gamma \in \mathbb{R}^{|D| \times |D|}} \|\Gamma\| \tag{2.7}$$

$$\text{subject to: } \Gamma^\top = \Gamma, \Gamma \circ F = 0, \text{ and } \forall j \in [n], \|\Gamma \circ \Delta_j\| \leq 1. \tag{2.8}$$

The following result connecting $\text{Adv}(\cdot)$ and $Q(\cdot)$ will be useful.

Theorem 5 ([HLŠ07; Lee+11]). *There exist constants $c_1, c_2 > 0$ such that for any function $f: D \rightarrow E$, we have $c_1 \text{Adv}(f) \leq Q(f) \leq c_2 \text{Adv}(f)$.*

In this chapter, we write $Q(f) = \Theta(\text{Adv}(f))$ for the statement in [Theorem 5](#).

The adversary quantity for a composite function can be expressed in terms of the adversary quantities of its components. In this work, we specifically consider cases where (i) $g = f_1 \vee f_2$; (ii) $g = f_1 \wedge f_2$; or (iii) a SWITCH-CASE scenario on functions $f: D \rightarrow E$ and $\{g_s : D \rightarrow \{0, 1\} \mid s \in E\}$ defined as $h := (\text{IF } f(x) = s) \text{ THEN } h(x) = g_s(x)$ for $s \in E, x \in D$.

Lemma 8 (Adversary composition for OR and AND). *Let $a, b \in \mathbb{N}$ and let Σ be a finite set. Let $f_1: \Sigma^a \rightarrow \{0, 1\}$ and $f_2: \Sigma^b \rightarrow \{0, 1\}$. Let $g^\wedge, g^\vee: \Sigma^a \times \Sigma^b \rightarrow \{0, 1\}$ be such that $g^\wedge(x, y) = f_1(x) \wedge f_2(y)$ and $g^\vee(x, y) = f_1(x) \vee f_2(y)$. Then $\text{Adv}(g^\wedge)^2 = \text{Adv}(g^\vee)^2 \leq \text{Adv}(f_1)^2 + \text{Adv}(f_2)^2$.*

Moreover, suppose $a = b$ and $h: \Sigma^a \rightarrow \{0, 1\}$ satisfies $h(x) = f_1(x) \wedge f_2(x)$ for all $x \in \Sigma^n$. Let $f'_2: f_1^{-1}(1) \rightarrow \{0, 1\}$ be such that $f'_2(x) = f_2(x)$ for all $x \in f_1^{-1}(1)$. Then $\text{Adv}(h)^2 \leq \text{Adv}(f_1)^2 + \text{Adv}(f'_2)^2$.

Note that $f_1^{-1}(1)$ denotes the set $\{x \in \Sigma^a \mid f_1(x) = 1\}$. The intuitive reason for the “moreover” part of the lemma is that if $f_1(x) = 0$, we do not need to compute $f_2(x)$ to compute $h(x)$ hence we can restrict the domain of f_2 to $f_1^{-1}(1)$. The “moreover” part of the lemma is used in our proof of [Lemma 10](#).

[Lemma 8](#) follows from [[LMRŠ10](#); [Rei11](#)]. For completeness, we provide a proof in [Appendix 1](#). Note that the first part of the lemma still holds if f_1 and f_2 share variables.⁷

Lemma 9 (Adversary composition for SWITCH-CASE). *Let $a \in \mathbb{N}$ and let Σ, Λ be finite sets. Let $f: \Sigma^a \rightarrow \Lambda$. For $s \in \Lambda$, let $g_s: \Sigma^a \rightarrow \{0, 1\}$. Define $h: \Sigma^a \rightarrow \{0, 1\}$ by $h(x) = g_{f(x)}(x)$. Then*

$$\text{Adv}(h) \leq O(\text{Adv}(f)) + \max_{s \in \Lambda} \text{Adv}(g_s). \quad (2.9)$$

See [Appendix 1](#) for a proof of [Lemma 9](#).

A divide-and-conquer strategy for a problem of size n typically involves dividing the problem into a subproblems on smaller instances of size n/b for $b > 1$ and combining the solutions by solving another auxiliary problem P^{aux} . The classical cost for $P^{\text{aux}}(n)$ is denoted $C^{\text{aux}}(n)$. The classical cost of solving $P(n)$ is then described by

⁷For example, if $G: \Sigma^a \rightarrow \{0, 1\}$ with $G(x) := g(x, x) = f_1(x) \wedge f_2(x)$, then $\text{Adv}(G)^2 \leq \text{Adv}(g)^2 \leq \text{Adv}(f_1)^2 + \text{Adv}(f_2)^2$. Alternatively, this can be seen from the proof of the “moreover” part of the lemma.

the recurrence

$$C(n) \leq aC(n/b) + C^{\text{aux}}(n). \quad (2.10)$$

The situation is more subtle in the quantum case. Equation (2.10) bounds the cost of solving the size- n problem by summing the costs of solving each subproblem and the auxiliary problem. However, our quantum recurrence relations come from adversary composition theorems, which impose restrictions on how the size- n problem decomposes into subproblems and an auxiliary problem.

We consider decision problems that correspond to the computation of a Boolean function $f: \Sigma^n \rightarrow \{0, 1\}$, where Σ is a finite set, and we have oracle access to an input $x \in \Sigma^n$. While the most general form of quantum divide and conquer can compute f by breaking it into sub-functions and an auxiliary function (corresponding to the subproblems and an auxiliary problem) in many different ways, here we focus on two strategies.

Strategy 1. f is computed by an AND-OR formula involving an auxiliary function $f^{\text{aux}}: \Sigma^n \rightarrow \{0, 1\}$ and sub-functions $\{f_i: \Sigma^{n/b} \rightarrow \{0, 1\} \mid i \in [a]\}$, where $a, b \in \mathbb{N}$. To elaborate, such an AND-OR formula is a rooted binary tree T with $a + 1$ leaves, where the internal nodes are labelled by either \vee or \wedge and each leaf is uniquely labelled by one element in $\{f_1, \dots, f_a, f^{\text{aux}}\}$. We say f is computed by T if there exists subsets $S_1, \dots, S_a \subseteq [n]$ with $|S_i| = n/b$ for all $i \in [a]$, such that for all $x \in \Sigma^n$, $f(x)$ equals the value of T at the root computed in the obvious way when the leaf of T labelled f_i is set to $f_i(x|_{S_i})$ for all $i \in [a]$ and the leaf of T labelled f^{aux} is set to $f^{\text{aux}}(x)$. (The symbol $x|_{S_i}$ denotes the element in $\Sigma^{n/b}$ that is the restriction of x to positions in S_i .)

Strategy 2. f is computed sequentially as follows: (1) Compute an auxiliary func-

tion $f^{\text{aux}}: \Sigma^n \rightarrow \Lambda$, where Λ is a finite set, to obtain some $s \in \Lambda$; (2)
 Compute $g_s: \Sigma^n \rightarrow \{0, 1\}$.

Using the adversary composition theorems stated previously, the next corollary shows how to obtain quantum recurrence relations for $\text{Adv}(f)$ when f is computed according to [Strategy 1](#) or [Strategy 2](#).

Corollary 1 (Quantum divide-and-conquer strategies). *Let f , f^{aux} , $\{f_i \mid i \in [a]\}$, $\{f_i^{(s)} \mid i \in [a], s \in \Lambda\}$, and $\{g^{(s)} \mid s \in \Lambda\}$ be as defined in [Strategies 1 and 2](#). Then*

$$\text{Adv}(f)^2 \leq \sum_{i=1}^a \text{Adv}(f_i)^2 + O(Q(f^{\text{aux}}))^2 \quad \text{for } \text{Strategy 1}, \quad (2.11)$$

$$\text{Adv}(f) \leq O(Q(f^{\text{aux}})) + \max_{s \in \Lambda} \text{Adv}(g_s) \quad \text{for } \text{Strategy 2}. \quad (2.12)$$

Proof. For [Strategy 1](#), f^{aux} and $\{f_i \mid i \in [a]\}$ are all Boolean functions and f is computed as an AND-OR formula of these functions. Therefore, we can apply [Lemma 8](#) at each node of the rooted binary tree that defines the AND-OR formula to obtain

$$\begin{aligned} \text{Adv}(f)^2 &\leq \text{Adv}(f_1)^2 + \cdots + \text{Adv}(f_a)^2 + \text{Adv}(f^{\text{aux}})^2 \\ &\leq \text{Adv}(f_1)^2 + \cdots + \text{Adv}(f_a)^2 + O(Q(f^{\text{aux}}))^2 \quad (\text{using } \text{Theorem 5}). \end{aligned} \quad (2.13)$$

For [Strategy 2](#), $f^{\text{aux}}: \Sigma^n \rightarrow \Lambda$ is first computed to give an $s \in \Lambda$. Then, f is computed as $g^{(s)}$. Therefore, for all $x \in \Sigma^n$, $f(x)$ can be expressed using SWITCH-CASE as

$$f(x) = g_{f^{\text{aux}}(x)}(x). \quad (2.14)$$

Therefore, we can directly apply [Lemma 9](#) to obtain

$$\begin{aligned} \text{Adv}(f) &\leq O(\text{Adv}(f^{\text{aux}})) + \max_{s \in \Lambda} \text{Adv}(g_s) \\ &\leq O(Q(f^{\text{aux}})) + \max_{s \in \Lambda} \text{Adv}(g_s) \quad (\text{using } \text{Theorem 5}). \end{aligned}$$

The corollary follows. \square

While adversary composition theorems play a key role in decomposing the adversary quantity for f , one novel aspect of our framework lies in the switching from the adversary quantity $\text{Adv}(f^{\text{aux}})$ to the quantum query complexity $Q(f^{\text{aux}})$ when handling the auxiliary function. This switching means our framework employs both the adversary and quantum algorithms toolboxes.

We recall a few well-known quantum query complexity bounds that are used in subsequent sections, usually to bound the quantum query complexity of the auxiliary function f^{aux} .

Theorem 6 (Quantum query complexity bounds). *Let $m, n \in \mathbb{N}$ with $m \leq n$.*

- Unstructured search. *Given an $x \in \{0, 1\}^n$, finding an i such that $x_i = 1$ has quantum query complexity $O(\sqrt{n})$ [Gro96].*
- Minimum finding. *Given $x \in \Sigma^n$, where Σ is equipped with a total order, the quantum query complexity of finding an $i \in [n]$ such that x_i is the minimum (or maximum) element in $\{x_j \mid j \in [n]\}$ is $O(\sqrt{n})$ [DH96].*
- String matching. *Given two strings x, y with $|x| = n, |y| = m$, determining if y is a substring of x has quantum query complexity $\tilde{O}(\sqrt{m} + \sqrt{n})$ [RV03].*
- String comparison. *Given two strings $x, y \in \Sigma^n$, where Σ is equipped with a total order, determining whether $x < y$ (in lexicographic order) has quantum query complexity $O(\sqrt{n})$. This is because the problem reduces to minimum finding.*
- Bipartite element distinctness. *Given two strings x, y such that $|x| = n, |y| = m$, and $m \leq n$, finding $i \in [n]$ and $j \in [m]$ such that $x_i = y_j$ has quantum query complexity $O(n^{2/3})$ [Amb07].*

Finally, once we have the quantum recurrence relation for a problem along with a bound on the quantum query complexity of f^{aux} , we use the Master Theorem to

solve the recurrence to bound $\text{Adv}(f)$, and thereby $Q(f)$. This is just like in classical divide and conquer.

We state the Master Theorem below for convenience.

Theorem 7 (Master Theorem [BHS80; CLRS09]). *Let $A: \mathbb{N} \rightarrow \mathbb{R}$, $a, b, c > 0$, and $p \geq 0$. Suppose A satisfies the recurrence*

$$A(n) \leq a A(n/b) + O(n^c \log^p n). \quad (2.15)$$

Then

$$A(n) = \begin{cases} O(n^{\log_b a}) & \text{if } \log_b a > c, \\ O(n^{\log_b a} \log^{p+1} n) & \text{if } \log_b a = c, \\ O(n^c \log^p n) & \text{if } \log_b a < c. \end{cases} \quad (2.16)$$

2.4 Applications

In this section we apply the divide-and-conquer strategies described previously to various string problems. Note that we divide a problem on strings of length n into subproblems on strings of length n/b for $b > 1$. For notational convenience, we assume that $n = b^r$ for some $r \in \mathbb{N}$ so that $n/b^k \in \mathbb{N}$ for all $k \in [r]$. This is without loss of generality because when $n \neq b^r$, we can divide into subproblems of sizes $\lceil n/b \rceil$ or $\lfloor n/b \rfloor (\leq \lceil n/b \rceil)$. This will lead to recurrence relations with n/b replaced by $\lceil n/b \rceil$. But the asymptotics of the solutions to these modified recurrence relations do not change, see [CLRS09, Section 4.6.2].

2.4.1 Recognizing regular languages

We first consider the problem of recognizing regular languages to demonstrate how quantum divide and conquer allows one to prove [AGS19, Theorem 18], without

needing the intricacies of the original proof. In the following, we prove a special case of this result, mentioned in the abstract of [AGS19], that captures its essence. We do not prove [AGS19, Theorem 18] in its full generality only because that would require introducing significant terminology related to regular languages but would not offer further insight. Let $\Sigma = \{0, 1, 2\}$.

Problem ($\Sigma^*20^*2\Sigma^*$). *Given query access to a string $x \in \Sigma^n$, decide if $x \in \Sigma^*20^*2\Sigma^*$, i.e., if x contains a substring with two 2s on either side of an arbitrarily long string of 0s.*

Theorem 8. *The quantum query complexity of $\Sigma^*20^*2\Sigma^*$ is $O(\sqrt{n \log(n)})$.*

Proof. Let $f_n: \{0, 1, 2\}^n \rightarrow \{0, 1\}$ be the function defined by $f_n(x) = 1$ iff $x \in \Sigma^*20^*2\Sigma^*$. We proceed by upper bounding the quantum query complexity f_n .

Clearly, x contains the expression 20^*2 if and only if the expression is contained (i) entirely in the left half of the string; or (ii) entirely in the right half of the string; or (iii) it is partly contained in the left half and partly in the right half of the string. Therefore, we have

$$f_n(x) = f_{n/2}(x_{\text{left}}) \vee f_{n/2}(x_{\text{right}}) \vee g_n(x), \quad (2.17)$$

where $g_n: \Sigma^n \rightarrow \{0, 1\}$ is defined by

$$g_n(x) = \begin{cases} 1 & \text{if } x[i..j] \text{ is of the form } 20^*2 \text{ for some } i \leq n/2 < j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.18)$$

Clearly, g_n can be decided by the following algorithm using $O(\sqrt{n})$ queries:

1. Grover search for the maximal index $i \leq n/2$ such that $x_i = 2$ and the minimal index $j > n/2$ such that $x_j = 2$. If either fails to exist, output 0.

2. Decide whether $x[i + 1 \dots j - 1]$ equals 0^{j-i-1} by Grover search. Output 1 if yes and 0 if no.

Therefore, writing $a(n) := \text{Adv}(f_n)$, observing that [Equation \(2.17\)](#) follows [Strategy 1](#) and using [Corollary 1](#) gives $a(n)^2 \leq 2a(n/2)^2 + O(n)$, which solves to $a(n)^2 = O(n \log(n))$ by the Master Theorem ([Theorem 7](#)). Hence, $Q(f_n) = O(a(n)) = O(\sqrt{n \log(n)})$ by [Theorem 5](#). \square

2.4.2 String Rotation and String Suffix

Let Σ be equipped with a total order in this subsection. We consider the following problems.

Problems (String Rotation and String Suffix). *Let $n \in \mathbb{N}$ and $i \in [n]$. Given query access to a string $x \in \Sigma^n$, we define the following problems.*

1. *Minimal String Rotation. Decide if $x[i \dots n]x[1 \dots i - 1] \leq x[j \dots n]x[1 \dots j - 1]$ for all $j \in [n]$. (A string of the form $x[j \dots n]x[1 \dots j - 1]$ with $j \in [n]$ is called a string rotation of x .)*
2. *Minimal Suffix. Decide if $x[i \dots n] < x[j \dots n]$ (in lexicographic order) for all $j \in [n] \setminus \{i\}$.*

Remark. These problems are decision versions of the problems studied in [\[AJ23\]](#), which gives algorithms that output the positions of the minimal string rotation and minimal suffix.

As observed in [\[AJ23\]](#), both problems reduce to the following problem.

Problem (Minimal Length- l Substring). *Let $n, l \in \mathbb{N}$ with $l \leq n$. Given query access to strings $x \in \Sigma^n$ and $y \in \Sigma^l$, decide if all length- l substrings of x are lexicographically at least y .*

We prove the following lemma by quantum divide and conquer.

Lemma 10. *The quantum query complexity of Minimal Length- l Substring with $l = n/2$ is $\tilde{O}(\sqrt{n})$.*

Proof. We upper bound the quantum query complexity of the function $f_n: \Sigma^n \times \Sigma^{n/2} \rightarrow \{0, 1\}$ defined by $f_n(x, y) = 1$ iff all length- $(n/2)$ substrings of x are lexicographically at least y .

Observe that $f_n(x, y) = 1$ if and only if (a) all length- $(n/4)$ substrings contained in $x[1..^{3n/4}]$ are lexicographically at least $y[1..^{n/4}]$; and (b) all length- $(n/2)$ substrings that start with $y[1..^{n/4}]$ are lexicographically at least y . Also observe that all length- $(n/4)$ substrings contained in $x[1..^{3n/4}]$ are either contained in $x[1..^{n/2}]$ or contained in $x[n/4 + 1..^{3n/4}]$. Therefore, we deduce that

$$f_n(x, y) = \left(f_{n/2}(x[1..^{n/2}], y[1..^{n/4}]) \wedge f_{n/2}(x[n/4 + 1..^{3n/4}], y[1..^{n/4}]) \right) \wedge g_n(x, y),$$

where $g_n: S \subseteq \Sigma^n \times \Sigma^{n/2} \rightarrow \{0, 1\}$ is defined to have a domain S that consists of all pairs $(x, y) \in \Sigma^n \times \Sigma^{n/2}$ such that all length- $(n/4)$ substrings contained in $x[1..^{3n/4}]$ are lexicographically at least $y[1..^{n/4}]$, and

$$g_n(x, y) = \begin{cases} 1 & \text{if all length-}(n/2)\text{ substrings in } x \text{ starting with } y[1..^{n/4}] \\ & \text{are lexicographically at least } y \text{ (or do not exist),} \\ 0 & \text{otherwise.} \end{cases} \quad (2.19)$$

We can show that $Q(g_n) = \tilde{O}(\sqrt{n})$ by considering a 2-stage quantum algorithm \mathcal{A} of the following description given input $(x, y) \in S$.

1. Use the quantum string matching algorithm (see [Theorem 6](#)) and binary search to find the *first* and *last* positions where $y[1..^{n/4}]$ starts in $x[1..^{1+n/2}] = x[1..^{n/2}]$. Call these two positions u_1 and v_1 , respectively. This takes $\tilde{O}(\sqrt{n})$

queries. Similarly, use the quantum string matching algorithm and binary search to find the first and last positions where $y[1..n/4]$ starts in $x[n/4+1..1+3n/4] = x[n/4+1..3n/4]$. Call these two positions u_2 and v_2 , respectively. This takes another $\tilde{O}(\sqrt{n})$ queries. If none of u_1, v_1, u_2, v_2 exist, then output 1.

2. Use the quantum string comparison algorithm (see [Theorem 6](#)) to decide whether all the length- $(n/2)$ substrings in x starting at a position in $\{u_1, v_1, u_2, v_2\}$ is lexicographically at least y . If yes, then output 1; otherwise, output 0. This takes $\tilde{O}(\sqrt{n})$ queries.

Clearly, \mathcal{A} uses $\tilde{O}(\sqrt{n})$ queries. The correctness of \mathcal{A} follows from the following [Claim 1](#) and the definition of the domain S of g_n . To elaborate, the definition of S implies that the minimal length- $(n/2)$ substring of x must start with either (i) $y[1..n/4]$ or (ii) a length- $(n/4)$ string that is lexicographically greater than $y[1..n/4]$. In case (i), the correctness of \mathcal{A} follows from [Claim 1](#) with “ s ” set to x , “ λ ” set to $n/2$, “ a ” set to 1 or $n/4$, “ k ” set to $n/4$, and “ t ” set to $y[1..n/4]$. In case (ii), the correctness of \mathcal{A} follows from the last sentence in the first stage of \mathcal{A} .

Claim 1 (Exclusion rule). *Let $s \in \Sigma^n$ and λ be an integer with $n/2 \leq \lambda \leq n$, let*

$$I := \arg \min_{1 \leq i \leq n-\lambda+1} s[i..i+\lambda]. \quad (2.20)$$

denote the set of starting positions⁸ of the minimal length- λ substring of s . For integers $a \geq 1, k \geq 1$ such that $a+k \leq n-\lambda+1$, let t denote a length- k substring, let $J := \{i \in \{a, \dots, a+k\} \mid s[i..i+k] = t\}$ denote the set of starting positions of t in the string $s[a..a+2k]$. Then if $\{\min J, \max J\} \cap I = \emptyset$, we must have $J \cap I = \emptyset$.

Proof. It suffices to quote the proof of [[AJ23](#), Lemma 4.8] verbatim (replacing its letter l by λ). This is because the only property the proof requires of J is that its

⁸The elements of I must form an arithmetic progression by [[AJ23](#), Lemma 2.3].

elements form an arithmetic progression. But [AJ23, Lemma 2.3] shows that the elements of J indeed form an arithmetic progression. \square

Now, setting $a(n) := \text{Adv}(f_n)$, the “moreover” part of Lemma 8 gives $a(n)^2 \leq 2a(n/2)^2 + \text{Adv}(g_n)^2$. Since $Q(g_n) = \tilde{O}(\sqrt{n})$, Theorem 5 implies $a(n)^2 = 2a(n/2)^2 + \tilde{O}(n)$, which solves to $a(n)^2 = \tilde{O}(n)$ by the Master Theorem (Theorem 7). Hence, $Q(f_n) = O(a(n)) = \tilde{O}(\sqrt{n})$ by Theorem 5. \square

Remark. The analysis in [AJ23, Theorem 4.1] for Minimal Length- l Substring would yield a looser bound of $O(n^{1/2+o(1)})$ for its quantum query complexity. That analysis is more involved because it splits the problem into a non-constant number of parts, m . This is because quantum algorithms are recursively applied to obtain a recurrence of the form $Q(n) \leq \tilde{O}(\sqrt{m}) Q(n/m) + \tilde{O}(\sqrt{mn})$, which, due to the $\tilde{O}(\sqrt{m})$ coefficient in front of the $Q(n/m)$, requires m to be non-constant to solve to $Q(n) = O(n^{1/2+o(1)})$. In general, it can be difficult to split a problem into a non-constant number of parts by only relying on quantum algorithmic techniques (e.g., consider an AND-OR formula), so such an approach may not yield optimal upper bounds (even up to an $n^{o(1)}$ factor). We note that our analysis above is a quantum analogue of the *classical* divide-and-conquer analysis in [AJ23, Start of Section 4.2].

Theorem 9 below follows from Lemma 10 and the chain of reductions described in [AJ23, Propositions 4.2–4.5 and Theorem 4.6]. We provide a proof for completeness.

Theorem 9. *The quantum query complexities of Minimal String Rotation and Minimal Suffix are $\tilde{O}(\sqrt{n})$.*

Proof. Let f_{2n} be defined as in the proof of Lemma 10. We show how Minimal String Rotation and Minimal Suffix can be computed using f_{2n} as follows.

1. Minimal String Rotation can be computed by $f_{2n}(xx, x[i..n]x[1..i-1])$. This is because the set R containing all length- n substrings of xx is precisely the set of

all string rotations of x and, by definition, $f_{2n}(xx, x[i..n]x[1..i-1])$ equals 1 if and only if all strings in R are lexicographically at least $x[i..n]x[1..i-1]$.

2. Minimal Suffix can be computed by $f_{2n}(1x0^{n-1}, x[i..n]0^{i-1})$, where 0 is defined to be smaller than all elements in Σ and 1 is defined to be larger than all elements in Σ . To see this, we first make the following observation: for two strings s, t over Σ , if $|s| \leq |t|$ and r is a nonnegative integer, we have $s < t \iff s0^{|t|-|s|}0^r < t0^r$, where 0 denotes a symbol that is defined to be smaller than all elements in Σ . Then

- (a) If $x[i..n]$ is the minimal suffix of x then $x[i..n] < x[j..n]$ for all $j \in [n] \setminus \{i\}$. Therefore $x[i..n]0^{i-1} < x[j..n]0^{j-1}$ for all $j \in [n] \setminus \{i\}$ by the observation above. In addition, $x[i..n]0^{i-1} < 1x0^{n-1}$ since 1 is larger than all elements in Σ . Therefore $x[i..n]0^{i-1}$ is the minimal string among all $n+1$ substrings of $1x0^{n-1}$ and so $f_{2n}(1x0^{n-1}, x[i..n]0^{i-1}) = 1$.
- (b) If $f_{2n}(1x0^{n-1}, x[i..n]0^{i-1}) = 1$, then $x[i..n]0^{i-1} < x[j..n]0^{j-1}$ for all $j \in [n] \setminus \{i\}$. If $j < i$, then $x[i..n] < x[j..n]$ by the observation above. If $j > i$, then $x[i..n] < x[j..n]0^{j-i}$. If $x[j..n] < x[i..n]$, then $x[j..n]0^{j-i} < x[i..n]$ by the observation above which is a contradiction. So $x[j..n] \geq x[i..n]$ and so $x[j..n] > x[i..n]$ since these strings are not equal for $j > i$. Therefore, for all $j \in [n] \setminus \{i\}$, we have $x[i..n] < x[j..n]$, which means $x[i..n]$ is the minimal suffix of x .

The above establishes $f_{2n}(1x0^{n-1}, x[i..n]0^{i-1}) = 1$ if and only if $x[i..n]$ is the minimal suffix of x .

Since $Q(f_{2n}) = \tilde{O}(\sqrt{n})$ by [Lemma 10](#), the theorem follows. \square

Remark. The quantum query complexities of Maximal String Rotation and Maximal Suffix (defined in the obvious way) are also $\tilde{O}(\sqrt{n})$ because they easily reduce to Minimal String Rotation and Minimal Suffix.

2.4.3 k -Increasing Subsequence

Let Σ be equipped with a total order and fix k to be a constant in this subsection. We consider the following problem, which is a natural parametrized version of Longest Increasing Subsequence.

Problem (k -IS). Let $n \in \mathbb{N}$. Given query access to $x \in \Sigma^n$, decide if x has a k -IS, i.e., if there exist integers $i_1 < i_2 < \dots < i_k$ such that $x[i_1] < x[i_2] < \dots < x[i_k]$.

To solve k -IS, we solve a variant of it that we call k -Increasing Subsequence* (k -IS*), defined as follows. We consider k -IS* because it is more susceptible to recursion.

Problem (k -IS*). Let $n \in \mathbb{N}$ and let $*$ denote an element outside Σ . Given query access to $x \in (\Sigma \cup \{*\})^n$, decide if x has a k -IS*, i.e., if there exist integers $i_1 < i_2 < \dots < i_k$ such that $x[i_1] < x[i_2] < \dots < x[i_k]$ and $x[i_l] \neq *$ for all $l \in [k]$.

We now prove the main theorem of this subsection.

Theorem 10. The quantum query complexity of k -IS* is $O(\sqrt{n} \log^{3(k-1)/2}(n))$ and the quantum query complexity k -IS is also $O(\sqrt{n} \log^{3(k-1)/2}(n))$

Proof. Clearly, k -IS reduces to k -IS* without using additional queries. Therefore, it suffices to prove the theorem for k -IS*. We upper bound the quantum query complexity of the function $\text{LIS}_{k,n}: \Sigma^n \rightarrow \{0, 1\}$ defined by $\text{LIS}_{k,n}(x) = 1$ iff x contains a k -IS*.

We first make the following definitions. Define $a_k(n) := \text{Adv}(\text{LIS}_{k,n})$. Define $\text{LIS}_{(i,j),n}: \Sigma^n \rightarrow \{0, 1\}$ by $\text{LIS}_{(i,j),n}(x) = 1$ if and only if x contains an $(i + j)$ -IS* consisting of an i -IS*, $u_1 < u_2 < \dots < u_i$, and a j -IS*, $v_1 < v_2 < \dots < v_j$, such that $u_i \leq n/2 < v_1$. For $j \in \mathbb{N}$, define $\text{min-last}_{j,n}: \Sigma^n \rightarrow \Sigma$ by $\text{min-last}_{j,n}(x) = \min_l x[l]$, where the minimization is over those $l \in [n]$ that are the last index of some j -IS* in x . Similarly, define $\text{max-first}_{j,n}: \Sigma^n \rightarrow \Sigma$ by $\text{max-first}_{j,n}(x) = \max_l x[l]$, where the maximization is over those $l \in [n]$ that are the first index of some j -IS* in x .

Observe that the increasing subsequence can be (i) contained entirely in the left half of x ; or (ii) contained entirely in the right half of x ; or (iii) partially contained in both halves with i elements in the left and $k - i$ elements in the right for some $i \in [k - 1]$. Therefore, we have

$$\text{LIS}_{k,n}(x) = \text{LIS}_{k,n/2}(x_{\text{left}}) \vee \text{LIS}_{k,n/2}(x_{\text{right}}) \vee \bigvee_{i=1}^{k-1} \text{LIS}_{(i,k-i),n}(x). \quad (2.21)$$

Therefore, by [Corollary 1](#) for [Strategy 1](#), we obtain the quantum divide and conquer recurrence

$$a_k(n)^2 \leq 2a_k(n/2)^2 + \sum_{i=1}^{k-1} O(Q(\text{LIS}_{(i,k-i),n})^2). \quad (2.22)$$

Now,

$$Q(\text{LIS}_{(i,k-i),n}) \leq Q(\text{min-last}_{i,n/2}) + Q(\text{max-first}_{k-i,n/2}), \quad (2.23)$$

because $\text{LIS}_{(i,k-i),n}(x)$ can be computed by computing $\text{min-last}_{i,n/2}(x_{\text{left}})$ and $\text{max-first}_{k-i,n/2}(x_{\text{right}})$ and checking that $\text{min-last}_{i,n/2}(x_{\text{left}}) < \text{max-first}_{k-i,n/2}(x_{\text{right}})$.

Moreover, for any $j \in \mathbb{N}$, $\text{min-last}_{j,n}$ and $\text{max-first}_{j,n}$ can be computed by a “randomized search” that uses $O(\log(n))$ computations of $\text{LIS}_{j,n}$ and Grover search, which is similar to the approach taken for quantum minimum finding [[DH96](#)][—see \[Appendix 2\]\(#\)](#) for a proof. Therefore, for all $j, n \in \mathbb{N}$, we can use [Theorem 5](#) to obtain

$$Q(\text{min-last}_{j,n}), Q(\text{max-first}_{j,n}) \leq O((a_j(n) + \sqrt{n}) \log(n)). \quad (2.24)$$

Substituting the combination of [Equation \(2.23\)](#) and [Equation \(2.24\)](#) into [Equation \(2.22\)](#) shows that, for $k \geq 2$, we have

$$a_k(n)^2 \leq 2a_k(n/2)^2 + O\left(\sum_{i=1}^{k-1} (a_i(n)^2 + n) \log^2(n)\right). \quad (2.25)$$

But $\Omega(\sqrt{n}) \leq a_i(n) \leq O(a_{i+1}(n))$ for all $i \geq 1$, which follows by applying [Theorem 5](#) to the observations that (i) $\text{LIS}_{i,n}(x) = \text{LIS}_{i+1,n+1}(0x)$ for all x , where $0 \neq *$ is some element smaller than all elements in Σ , and (ii) computing $\text{LIS}_{1,n}$ is equivalent to searching for a $*$. Therefore, as k is a constant, we have

$$a_k(n)^2 \leq 2a_k(n/2)^2 + O(a_{k-1}(n)^2 \log^2(n)). \quad (2.26)$$

Finally, we prove $a_k(n) = O(\sqrt{n} \log^{3(k-1)/2}(n))$ by induction on $k \geq 1$. The base case ($k = 1$) follows by applying [Theorem 5](#) to the observation that computing $\text{LIS}_{1,n}$ is equivalent to searching for a $*$. For $k > 1$, [Equation \(2.26\)](#) and the inductive hypothesis for the $(k - 1)$ -th case give

$$a_k(n)^2 \leq 2a_k(n/2)^2 + O(n \log^{3k-4}(n)), \quad (2.27)$$

which solves to $a_k(n) = O(\sqrt{n} \log^{3(k-1)/2}(n))$, as required. The result follows from [Theorem 5](#). □

2.4.4 k -Common Subsequence

As our last application, we consider a parameterized version of the Longest Common Subsequence (LCS) problem. We fix k to be a constant in this subsection.

Problem (k -CS). *Let $n \in \mathbb{N}$. Given query access to strings $x, y \in \Sigma^n$, decide if they share a k -common subsequence (k -CS), i.e., if there exist integers $i_1 < i_2 < \dots < i_k$ and $j_1 < j_2 < \dots < j_k$ with $x[i_l] = y[j_l]$ for all $l \in [k]$.*

The main theorem of this subsection is the following.

Theorem 11. *The quantum query complexity of k -CS is $O(n^{2/3} \log^{k-1}(n))$.*

Define $\text{LCS}_{k,n} : \Sigma^n \times \Sigma^n \rightarrow \{0, 1\}$ by $\text{LCS}_{k,n}(x, y) = 1$ iff $x, y \in \Sigma^n$ share a k -CS. When $k = 1$ this problem is equivalent to the (bipartite) Element Distinctness prob-

lem [AS04; Amb07]. Aaronson and Shi showed that the quantum query complexity of Element Distinctness is at least $\Omega(n^{2/3})$ [AS04], while Ambainis proved the matching upper bound of $O(n^{2/3})$ [Amb07]. Since Element Distinctness reduces to k -CS (by prefixing $k - 1$ common elements to the input strings of k -CS), we see that our theorem gives a tight bound on the quantum query complexity of k -CS up to logarithmic factors.

We start by introducing some terminology. For $x, y \in \Sigma^n$, we say that $(i, j) \in [n] \times [n]$ is a collision of x and y iff $x[i] = y[j]$.

We fix some constant $m \in \mathbb{N}$ which we call the splitting factor. Given inputs $x, y \in \Sigma^n$, we consider splitting x and y each into m substrings of size n/m . We denote these substrings by

$$x^{(i)} := x\left(\frac{i-1}{m}n \dots \frac{i}{m}n\right) \quad \text{and} \quad y^{(i)} := y\left(\frac{i-1}{m}n \dots \frac{i}{m}n\right) \quad \text{for all } i \in [m]. \quad (2.28)$$

Definition 8 (Subproblems and signatures). *By a subproblem, we refer to a tuple $(i, j) \in [m] \times [m]$. By the (i, j) -subproblem of $(x, y) \in \Sigma^n \times \Sigma^n$, we refer to the tuple $(x^{(i)}, y^{(j)})$. To each $x, y \in \Sigma^n$ we associate an m^2 -bit signature $\mathbf{s} \in \mathcal{S} := \{0, 1\}^{m^2}$ such that $\mathbf{s}_{i,j} = 1$ iff there is a collision between $x^{(i)}$ and $y^{(j)}$. Let $\sigma_n: \Sigma^n \times \Sigma^n \rightarrow \mathcal{S}$ be the function that, given input (x, y) , returns its signature.*

It is clear that the signature can be computed by using Ambainis's algorithm for Element Distinctness [Amb07] m^2 times, once for each subproblem. Since m is a constant, we have the following.

Lemma 11. *The quantum query complexity of σ_n satisfies $Q(\sigma_n) = O(n^{2/3})$.*

Definition 9 (Compatible, relevant, and critical subproblems).

1. *We say that two subproblems (i_1, j_1) and (i_2, j_2) are compatible if $(i_1 < i_2$ and $j_1 < j_2)$ or $(i_1 > i_2$ and $j_1 > j_2)$.*

2. Given a signature $\mathbf{s} \in \mathcal{S}$, we say that the (i, j) subproblem is \mathbf{s} -relevant if $\mathbf{s}_{i,j} = 1$.
3. Given a signature $\mathbf{s} \in \mathcal{S}$, we say that an \mathbf{s} -relevant subproblem (i, j) is \mathbf{s} -critical if none of the subproblems compatible with (i, j) are \mathbf{s} -relevant.

Observe that if subproblems (i_1, j_1) and (i_2, j_2) are compatible then, for any $x, y \in \Sigma^n$, the union of any k_1 -CS of the (i_1, j_1) -subproblem of (x, y) and any k_2 -CS of the (i_2, j_2) -subproblem of (x, y) is a $(k_1 + k_2)$ -CS of (x, y) . A key intuition of our approach is the following. Given $x, y \in \Sigma^n$, to decide whether x, y share a k -CS using divide and conquer, we might want to consider recursing on the (i_1, j_1) -subproblem of (x, y) . Assume that we have obtained the signature \mathbf{s} of (x, y) , and we find that subproblem (i_1, j_1) is \mathbf{s} -relevant but not \mathbf{s} -critical, i.e., there exists an \mathbf{s} -relevant subproblem (i_2, j_2) which is compatible with (i_1, j_1) . In this setting, finding even a $(k - 1)$ -CS of the (i_1, j_1) -subproblem of (x, y) would suffice to conclude that $\text{LCS}_{k,n}(x, y) = 1$, since the collision in the (i_2, j_2) -subproblem of (x, y) can be combined with a $(k - 1)$ -CS in the (i_1, j_1) -subproblem of (x, y) to give a k -CS of (x, y) . This suggests that we should only recurse on those subproblems that are critical according to the signature of (x, y) .

The above discussion motivates the following definition.

Definition 10 (Simple and composite k -CS). *Let $x, y \in \Sigma^n$. We say that a k -CS of (x, y) is simple if it is also a k -CS of $(x^{(i)}, y^{(j)})$ for some $i, j \in [m]$. A k -CS of (x, y) which is not simple is called composite.*

We define $\text{LCS}_{k,n}^{\text{composite}}: \Sigma^n \times \Sigma^n \rightarrow \{0, 1\}$ by $\text{LCS}_{k,n}^{\text{composite}}(x, y) = 1$ iff x and y share a composite k -CS. Given a signature $\mathbf{s} \in \mathcal{S}$, define $h_{k,n}^{(\mathbf{s})}: \Sigma^n \times \Sigma^n \rightarrow \{0, 1\}$ by $h_{k,n}^{(\mathbf{s})}(x, y) = 1$ iff there exists an \mathbf{s} -critical subproblem (i, j) such that the (i, j) -subproblem of (x, y) has a k -CS.

By the above discussion, we have the following proposition.

Proposition 1. For all $x, y \in \Sigma^n$, we have

$$\text{LCS}_{k,n}(x, y) = \text{LCS}_{k,n}^{\text{composite}}(x, y) \vee h_{k,n}^{(\sigma_n(x,y))}(x, y). \quad (2.29)$$

Proof. If x and y do not share a k -CS, then both sides of the equation evaluate to false. If x and y share a composite k -CS then both sides of the equation evaluate to true.

Consider the remaining case when x and y share a simple k -CS but no composite k -CS. The left-hand side of Equation (2.29) evaluates to true. Let $\mathbf{s} := \sigma_n(x, y)$ be the signature of (x, y) . Then, the right-hand side of Equation (2.29) evaluates to $h_{k,n}^{(\mathbf{s})}(x, y)$. Since x and y share a simple k -CS, there exists a k -CS of (x, y) which is also a k -CS of $(x^{(i)}, y^{(j)})$ for some \mathbf{s} -relevant subproblem (i, j) . Then (i, j) must also be \mathbf{s} -critical, since otherwise there would be an \mathbf{s} -relevant subproblem (i', j') compatible with (i, j) , which implies that the (i, j) - and (i', j') -subproblems of (x, y) would together yield a composite k -CS of (x, y) , contradicting the case we are in. Therefore, $h_{k,n}^{(\mathbf{s})}(x, y)$ also evaluates to true as required. \square

Let $a_k(n)$ denote the adversary quantity of $\text{LCS}_{k,n}$. As $Q(\text{LCS}_{k,n}) = \Theta(a_k(n))$ (Theorem 5), we can prove Theorem 11 by upper bounding the adversary quantity $a_k(n)$ instead of the quantum query complexity $Q(\text{LCS}_{k,n})$. As a first step we establish upper bounds on the adversary quantities and quantum query complexities of the functions on the right-hand side of Equation (2.29).

Lemma 12. For any signature $\mathbf{s} \in \mathcal{S}$, the adversary quantity of the function $h_{k,n}^{(\mathbf{s})}$ satisfies

$$\text{Adv}(h_{k,n}^{(\mathbf{s})}) \leq \sqrt{2m-1} a_k(n/m). \quad (2.30)$$

Lemma 13. *The quantum query complexity of $\text{LCS}_{k,n}^{\text{composite}}$ satisfies*

$$Q(\text{LCS}_{k,n}^{\text{composite}}) \leq O(a_{k-1}(n) \log n). \quad (2.31)$$

We defer the proofs of [Lemmas 12](#) and [13](#) for now, and prove the following lemma assuming these results.

Lemma 14. *For $k \geq 2$, the adversary quantity $a_k(n)$ of $\text{LCS}_{k,n}$ satisfies the recurrence*

$$a_k(n) \leq \sqrt{2m-1} a_k(n/m) + O(a_{k-1}(n) \log n). \quad (2.32)$$

Proof. Recall that by [Proposition 1](#),

$$\text{LCS}_{k,n}(x, y) = \text{LCS}_{k,n}^{\text{composite}}(x, y) \vee h_{k,n}^{(\sigma_n(x,y))}(x, y). \quad (2.33)$$

This Boolean expression is an OR where one of the components, $h_{k,n}$, is a SWITCH-CASE conditioned on the value of the signature function $\sigma_n(x, y)$. This involves both of our strategies discussed in [Section 2.3](#). Therefore, by applying [Corollary 1](#),

$$a_k(n) = \text{Adv}(\text{LCS}_{k,n}) \leq O(Q(\text{LCS}_{k,n}^{\text{composite}})) + O(Q(\sigma_n)) + \max_{\mathbf{s} \in \mathcal{S}} \{\text{Adv}(h_{k,n}^{(\mathbf{s})})\}. \quad (2.34)$$

By [Lemmas 11](#) to [13](#),

$$a_k(n) \leq O(a_{k-1}(n) \log n) + O(n^{2/3}) + \sqrt{2m-1} a_k(n/m) \quad (2.35)$$

$$\leq \sqrt{2m-1} a_k(n/m) + O(a_{k-1}(n) \log n), \quad (2.36)$$

where the last line follows since $a_1(n) \in \Theta(n^{2/3})$, and $a_{k-1}(n) = \Omega(n^{2/3})$ since Element Distinctness reduces to $(k-1)$ -CS for $k \geq 2$ by prefixing $k-2$ common elements to the input strings for $(k-1)$ -CS. □

The above recurrence allows us to prove our main theorem.

Proof of Theorem 11. We proceed by induction on $k \geq 1$. In the base case ($k = 1$), the theorem holds since 1-CS is equivalent to bipartite Element Distinctness, which has quantum query complexity $O(n^{2/3})$ [Amb07]. Assume that $k \geq 2$. By Lemma 14 we have the following recurrence for the adversary quantity of $\text{LCS}_{k,n}$:

$$a_k(n) \leq \sqrt{2m-1} a_k(n/m) + O(a_{k-1}(n) \log n) \quad (2.37)$$

$$\leq \sqrt{2m-1} a_k(n/m) + O(n^{2/3} \log^{k-1}(n)) \quad (2.38)$$

where the second inequality follows by $a_{k-1}(n) \leq O(n^{2/3} \log^{k-2}(n))$, our inductive hypothesis. The result follows from the Master Theorem (Theorem 7) if $\log_m(\sqrt{2m-1}) < 2/3$. We achieve this by choosing $m = 7$. \square

We now prove Lemmas 12 and 13.

Proof of Lemma 12. Fix an arbitrary signature $\mathbf{s} \in \mathcal{S}$. Recall that the function $h_{k,n}^{(\mathbf{s})}(x, y)$ indicates whether there exists an \mathbf{s} -critical subproblem (i, j) such that the (i, j) -subproblem of (x, y) has a k -CS. Let $R := \{(i_l, j_l) : l \in [r]\}$ denote the set of \mathbf{s} -critical subproblems. Then

$$h_{k,n}^{(\mathbf{s})}(x, y) = \bigvee_{l \in [r]} \text{LCS}_{k,n/m}(x^{(i_l)}, y^{(j_l)}). \quad (2.39)$$

By the adversary composition lemma for OR (Lemma 8),

$$\text{Adv}(h_{k,n}^{(\mathbf{s})}) = \sqrt{r} a_k(n/m). \quad (2.40)$$

To conclude, it suffices to prove that $r \leq 2m - 1$. To this end, to each subproblem (i, j) we associate the (signed) slope $i - j$. Observe that there are only $2m - 1$ possible slope values, namely $\{0, \pm 1, \dots, \pm(m-1)\}$. Now notice that if two distinct

subproblems (i, j) and (i', j') have the same associated slope (i.e., $i - j = i' - j'$), then these subproblems are compatible. Therefore, for each slope, there can be at most one subproblem in R since the subproblems in R are s -critical. Therefore, $r \leq 2m - 1$. \square

Let $K_{b,b} = (U, V, E)$ denote the bipartite graph on vertex sets $U := \{u_1, \dots, u_b\}$ and $V := \{v_1, \dots, v_b\}$ where $E := \{\{u_i, v_j\} \mid i, j \in [b]\}$. A weighted subgraph of $K_{b,b}$ is a subgraph $G = (U', V', E')$ of $K_{b,b}$ together with a weight function $w: E' \rightarrow \mathbb{N}$. We say that edge $\{u_i, v_j\} \in E'$ has weight $w(\{u_i, v_j\})$. For notational convenience, we will write edge $\{u_i, v_j\} \in E'$ as simply⁹ $(i, j) \in [b] \times [b]$.

To prove [Lemma 13](#), we will consider the complete bipartite graph $K_{b,b}$ and weighted versions of its subgraphs. We will use the following definitions.

Definition 11 (Crossing and non-crossing edges of $K_{b,b}$). *We say two edges, (i_1, j_1) and (i_2, j_2) , of $K_{b,b}$ are crossing if either $(i_1 < i_2 \text{ and } j_1 > j_2)$ or $(i_1 > i_2 \text{ and } j_1 < j_2)$. Otherwise, we say the edges are non-crossing.*¹⁰

Definition 12 (Schematics and composite schematics). *A schematic is a weighted subgraph G of $K_{b,b}$ such that: (i) all edges in G are pair-wise non-crossing, (ii) the weight of each edge in G is a positive integer, and (iii) the sum of the weights of all edges in G equals k . A composite schematic is a schematic with at least two edges.*

Let the set of all possible composite schematics be denoted cSch . Clearly, the size of cSch , $|\text{cSch}|$, is a function of b and k only. Therefore, because b and k are constants, $|\text{cSch}|$ is also constant. The first property (non-crossing edges) of a schematic means that the edges (i_1, j_1) and (i_2, j_2) of a given schematic can be totally ordered by the relation “ \leq ” defined by $(i_1, j_1) \leq (i_2, j_2) \iff i_1 \leq i_2 \text{ and } j_1 \leq j_2$. We say (i_1, j_1) is to the *left* of (i_2, j_2) iff $(i_1, j_1) < (i_2, j_2)$, i.e., $(i_1, j_1) \leq (i_2, j_2)$ and $(i_1, j_1) \neq (i_2, j_2)$.

⁹A tuple in $[b] \times [b]$ was previously referred to as a subproblem. However, it is clearer to no longer think of them as subproblems. In particular, we will no longer consider whether these tuples are s -relevant or s -critical.

¹⁰The notion of non-crossing edges is slightly *different* from the notion of compatible subproblems defined previously. For example, $(1, 1)$ and $(1, 2)$ are non-crossing but are not compatible.

Let G be a given composite schematic. Since G has a finite number of edges and at least two edges, it must have a smallest edge and a *distinct* second-smallest edge, where “small” is defined with respect to \leq . We call these the left-most and second-left-most edges of G , respectively. The weight k_1 of the left-most edge G must satisfy $1 \leq k_1 \leq k - 1$ since the second-left-most edge also has a positive integer weight and the sum of the weights of all edges in G equals k .

We proceed to prove [Lemma 13](#).

Proof of Lemma 13. Consider the following quantum algorithm \mathcal{A} for computing $\text{LCS}_{k,n}^{\text{composite}}(x, y)$.

For each $G \in \text{cSch}$:

Let $(i_1, j_1) < (i_2, j_2)$ be the left-most and second-left-most edges of G , respectively. Let $1 \leq k_1 \leq k - 1$ be the weight of (i_1, j_1) . There are three possible cases.

Case 1: $i_1 < i_2$ and $j_1 < j_2$. Run a query-optimal quantum algorithm for k_1 -CS on input $(x^{(i_1)}, y^{(j_1)})$. Then run a query-optimal quantum algorithm for $(k - k_1)$ -CS on input $(x((i_2 - 1)n/b \dots n], y((j_2 - 1)n/b \dots n])$. Return 1 if both algorithms return 1.

Case 2: $i_1 = i_2$ and $j_1 < j_2$. Run a query-optimal quantum algorithm for k_1 -CS on input $(x_I, y^{(j_1)})$, where x_I is a substring of $x^{(i_1)}$ indexed by a set I . I is initially set to be $((i_1 - 1)n/b \dots i_1 n/b]$, so that $x_I = x^{(i_1)}$. If the algorithm returns 0 with this initial setting of I , then continue to the next $G \in \text{cSch}$. Else, perform a binary search by appropriately adjusting I to find the minimal $l \in ((i_1 - 1)n/b \dots i_1 n/b]$ such that $x((i_1 - 1)n/b \dots l]$ and $y^{(j_1)}$ share a k_1 -CS. Then, run a query-optimal quantum algorithm for $(k - k_1)$ -CS on input $(x(l \dots n], y((j_2 - 1)n/b \dots n])$ and return its output.

Case 3: $i_1 < i_2$ and $j_1 = j_2$. Run an algorithm analogous to that in Case 2.

Return 0.

By inspection, and using [Theorem 5](#), the query complexity of \mathcal{A} is of order

$$|\text{cSch}| \cdot \max_{1 \leq k_1 \leq k-1} (a_{k_1}(n) + a_{k-k_1}(n)) \cdot \log(n), \quad (2.41)$$

where the $\log(n)$ arises from the binary search¹¹ in Cases 2 and 3. For all $1 \leq i \leq k-1$, since i -CS reduces to $(k-1)$ -CS by appending $(k-1-i)$ common characters to the input strings, the quantum query complexity of i -CS must be of the same order as that of $(k-1)$ -CS. Combined with [Theorem 5](#), this shows that $\max_{1 \leq k_1 \leq k-1} (a_{k_1}(n) + a_{k-k_1}(n)) = O(a_{k-1}(n))$. Since $|\text{cSch}|$ is a constant, we see that the query complexity of \mathcal{A} is $O(a_{k-1}(n) \log(n))$.

It remains to argue that \mathcal{A} correctly computes $\text{LCS}_{k,n}^{\text{composite}}$. Let the input be $x, y \in \Sigma^n$. If x and y do not share a composite k -CS, it is easy to check that \mathcal{A} outputs 1 with low probability. Suppose x and y share a composite k -CS, $T := \{(u_1, v_1), \dots, (u_k, v_k)\}$ so that $u_l < u_{l+1}$, $v_l < v_{l+1}$, and $x[u_l] = y[v_l]$ for all l . In this case, we can associate T with a composite schematic G_T as follows.

For each $(i, j) \in [b] \times [b]$ do the following. Count the number N of (u_b, v_b) such that $u_b \in ((i-1)n/b .. in/b)$ and $v_b \in ((j-1)n/b .. jn/b)$. If $N = 0$, then do not assign edge (i, j) to G_T . If $N > 0$, then assign edge (i, j) with weight N to G_T .

It is straightforward to check that the G_T defined this way indeed satisfies the four conditions required for it to be a composite schematic. Moreover, it is straightforward

¹¹Since the quantum algorithms for k_1 -CS and $(k-k_1)$ -CS can be erroneous with bounded probability, one might naively expect an additional $\log \log(n)$ factor due to the need for error reduction since the algorithm for k_1 -CS is called $O(\log(n))$ times in the binary search. However, [\[FRPU94\]](#) shows that the $\log \log(n)$ factor can be removed by performing a variant of binary search.

to check that \mathcal{A} outputs 1 with high probability when $G = G_T$ in its for-loop. Hence the lemma. \square

2.5 Discussion and open problems

We conclude by discussing some open problems related to this chapter.

1. Can we obtain super-quadratic speedups using quantum divide and conquer? One candidate problem is the evaluation of a NAND tree under some promise on its input. We already know that a problem of this type admits a super-polynomial quantum speedup from [ZKH12] which uses adversary-related techniques to derive their quantum query upper bound. While the proof in [ZKH12] does not use adversary composition theorems as prescribed by the quantum divide and conquer framework, it may be possible to recast the proof so that it does use adversary composition theorems. However, our framework additionally allows for the use of quantum algorithmic techniques, which could potentially allow us to not only recover but extend the results in [ZKH12].
2. Can we apply quantum divide and conquer to search problems? In particular, is there a quantum divide-and-conquer algorithm for minimum finding? More precisely, let $\min_n: [m]^n \rightarrow [m]$ be defined by $\min_n(x) := \min_{i \in [n]} x_i$. We have $\min_n(x) = \min(\min_{n/2}(x_{\text{left}}), \min_{n/2}(x_{\text{right}}))$, where x_{left} and x_{right} are the left and right halves of x . Is it true that $\text{Adv}(\min_n) \leq \sqrt{2} \text{Adv}(\min_{n/2})$ for all $m \in \mathbb{N}$? Note that this inequality holds trivially for $m = 1$ and holds for $m = 2$ because \min_n is essentially the same as AND on n bits in that case. We want to prove this inequality not because we want to bound the quantum query complexity of \min_n (we know that to be $\Theta(\sqrt{n})$ from [DH96]) but because we want to use \min_n within the quantum divide and conquer framework.
3. Can we find applications of quantum divide and conquer using generalizations of

the two strategies presented in [Section 2.3](#) (i.e., using combining functions other than AND-OR formulas and SWITCH-CASE, such as those discussed in the introduction)? For example, can we draw inspiration from Strassen’s algorithm [[Str69](#)] which uses a highly novel divide-and-conquer strategy to multiply matrices fast. Concretely, we ask if a novel *quantum* divide-and-conquer strategy can be used to improve the quantum query complexity of verifying matrix products as studied in [[BŠ04](#)]. In this problem, we are given query access to three matrices A, B, C and asked to decide if $AB = C$. Note that unless we answer the second open problem, we may have to restrict attention to decision versions of matrix multiplication, such as verifying matrix products.

Chapter 3

Quantum exploration algorithms for multi-armed bandits

Chapter summary. Identifying the best arm of a multi-armed bandit is a central problem in reinforcement learning. We study a quantum computational version of this problem given query access to an oracle that encodes the reward probabilities of each arm as quantum amplitudes. Specifically, we show that we can find the best arm with fixed confidence using $\tilde{O}(\sqrt{\sum_{i=2}^n \Delta_i^{-2}})$ quantum queries, where Δ_i represents the difference between the mean reward of the best arm and the i^{th} -best arm. This algorithm, based on variable-time amplitude amplification and estimation, gives a quadratic speedup compared to the best possible classical result. We also prove a matching quantum lower bound (up to poly-logarithmic factors).

3.1 Introduction

The multi-armed bandit (MAB) model is one of the most fundamental settings in reinforcement learning. Introduced by Robbins [Rob52], it is customary to describe the model as consisting of a row of n slot machines (single-armed bandits) where each gives a random reward when its arm is pulled. The main problems in this model are concerned with how the total reward obtained can be maximized while minimizing

the number of arm pulls. Despite its simplicity, the MAB model already captures crucial theoretical issues in decision making under uncertainty such as the tradeoff between exploration and exploitation. Furthermore, it finds applications in many areas including economics, statistics, and operations research.

A key problem in the MAB model is known as *best-arm identification*, where the goal is to efficiently identify the arm with the largest expected reward. This problem captures a common difficulty in practical scenarios, where at unit cost, only partial information about the system of interest can be obtained. A real-world example is a recommendation system, where the goal is to find appealing items for users. For each recommendation, only feedback on the recommended item is obtained. In the context of machine learning, best-arm identification can be viewed as a high-level abstraction and core component of active learning, where the goal is to minimize the uncertainty of an underlying concept, and each step only reveals the label of the data point being queried. In this chapter, we study best-arm identification in multi-armed bandits, establishing quantum speedup.

Problem setup. We work in a standard multi-armed bandit setting [EMM02] in which the MAB has n arms, where arm $i \in [n] := \{1, \dots, n\}$ is a Bernoulli random variable taking value 1 with probability p_i and value 0 with probability $1 - p_i$. Each arm can therefore be regarded as a coin with *bias* p_i . As our algorithms and lower bounds are symmetric with respect to the arms, we assume without loss of generality that $p_1 \geq \dots \geq p_n$, and denote $\Delta_i := p_1 - p_i$ for all $i \in \{2, \dots, n\}$. We further assume that $p_1 > p_2$, i.e., the best arm is unique. Given a parameter $\delta \in (0, 1)$, our goal is to use as few queries as possible to determine the best arm with probability $\geq 1 - \delta$. This is known as the *fixed confidence setting*. We primarily characterize complexity in terms of the parameter

$$H := \sum_{i=2}^n \frac{1}{\Delta_i^2} \tag{3.1}$$

which arises in the analysis of classical MAB algorithms (as discussed below).

We consider a quantum version of best-arm identification in which we can access the arms *coherently*. This means we have access to a quantum oracle \mathcal{O} that acts as

$$\mathcal{O}: |i\rangle_I |0\rangle_B |0\rangle_J \mapsto |i\rangle_I (\sqrt{p_i} |1\rangle_B |v_i\rangle_J + \sqrt{1-p_i} |0\rangle_B |u_i\rangle_J), \quad (3.2)$$

where $|v_i\rangle$ and $|u_i\rangle$ are arbitrary states, for all $i \in [n]$. We have used standard Dirac notation which we review in [Section 3.2](#). Register I is the “index” register with n states that correspond to the n arms. Register B is the single-qubit “bandit” register with two states, $|1\rangle$ corresponding to a reward and $|0\rangle$ corresponding to no reward. Register J is a multi-qubit “junk” register. For convenience, we omit register labels when this causes no confusion. Compared to pulling an arm classically—which can be implemented by measuring the bandit register—the quantum oracle allows access to different arms in superposition, a necessary feature for quantum speedup. In real-world applications, we usually have junk when instantiating our oracle (see below). When deriving our results however, we will assume there is no junk (i.e., we set $|v_i\rangle = |u_i\rangle = 1$ for all $i \in [n]$ in [Eq. \(3.2\)](#)). This is without loss of generality as the algorithm we construct is insensitive to junk. The reason for this is that the amplitude estimation algorithm (see [\[BHMT02\]](#) and [Theorem 13](#)), which forms the base of the algorithm that we construct, is insensitive to junk.

Previous work on quantum algorithms for clustering [\[KLLP19; WKS15\]](#) and reinforcement learning [\[DTB16; DB18\]](#) has discussed how to instantiate an oracle similar to \mathcal{O} . In clustering, \mathcal{O} is created using the SWAP test where for each i , p_i encodes the distance between some fixed vector and the i^{th} vector in some collection. Our algorithm can be used to speed up the algorithms of [\[KLLP19; WKS15\]](#).

We now give another important example of instantiating \mathcal{O} . Consider a classical

Monte Carlo strategy for playing a two-player game such as Go.¹ At a given position, evaluate the quality of a next move i by uniformly randomly playing out games $x \in X(i)$, where $X(i)$ is the set of valid games from move i onwards, and querying a computer program f that computes a bit $f(i, x) \in \{0, 1\}$ indicating if game x is won (1) or lost (0). In the classical case, we obtain one sample of win or loss using one query to f . In the quantum case, we can also instantiate one query to the quantum oracle in Eq. (3.2) using just one query to f . To do this, we apply the circuit for f , made reversible in the usual way [NC10, Sec. 1.4.1], on the quantum state corresponding to uniformly random play as follows:

$$\begin{aligned}
 |i\rangle |0\rangle & \frac{1}{\sqrt{|X(i)|}} \sum_{x \in X(i)} |x\rangle \xrightarrow{f} |i\rangle \sum_{x \in X(i)} \frac{1}{\sqrt{|X(i)|}} |f(i, x)\rangle |x\rangle \\
 & = |i\rangle (\sqrt{p_i} |1\rangle |u_i\rangle + \sqrt{1 - p_i} |0\rangle |v_i\rangle),
 \end{aligned} \tag{3.3}$$

where $|u_i\rangle$ and $|v_i\rangle$ are some states, and p_i is the empirical probability that move i leads to a win. Our quantum algorithm then uses quadratically fewer calls to f compared with classical Monte Carlo search to find the best next move.

We stress that we do not need to know the p_i s to instantiate the quantum oracle above. We also remark that our algorithm does not apply to every MAB situation. For example, in clinical trials to identify the best drug, we cannot instantiate the quantum oracle because human participants, unlike computer programs, cannot be queried in superposition.

Our algorithm can also be adapted to work when the reward distributions are promised to have bounded variance (for example, if they are sub-Gaussian). The adaptation essentially follows by replacing amplitude estimation (introduced in Section 3.2) with quantum mean estimation [Mon15], which works on any distribution with bounded variance.

¹The strategy described here can be thought of as Monte Carlo tree search (MCTS) without tree expansion. We further discuss MCTS at the end of this chapter.

Contributions. In this chapter, we give a comprehensive study of best-arm identification using quantum algorithms. Specifically, we obtain the following main result:

Theorem 12. *Given a multi-armed bandit oracle \mathcal{O} and confidence parameter $\delta \in (0, 1)$, there exists a quantum algorithm that, with probability $\geq 1 - \delta$, outputs the best arm using $\tilde{O}(\sqrt{H})$ queries to \mathcal{O} . Moreover, this query complexity is optimal up to poly-logarithmic factors in n , δ , and Δ_2 .*

This represents a quadratic quantum speedup over what is possible classically. To establish [Theorem 12](#) we use more quantum algorithmic techniques that extend Grover’s algorithm, namely variable-time amplitude amplification (VTAA) [[Amb12](#); [CKS17](#)] and estimation (VTAE) [[CGJ19](#)]. We apply VTAA and VTAE on a variable-time quantum algorithm \mathcal{A} that we construct. \mathcal{A} outputs a state with labeled “good” and “bad” parts. Using that label, VTAA removes the bad part so that only the good part remains, and VTAE estimates the proportion of the good part. In our application, the good part is eventually the best-arm state. If we instead used standard amplitude amplification and estimation, which are more immediate corollaries of Grover’s algorithm, we would get a worse complexity of $\tilde{O}(\sqrt{n}/\Delta_2)$.

We emphasize that our quantum algorithm, like classical ones [[EMM02](#); [GGL12](#); [JMNB14](#); [KKS13](#); [MT04](#)], does not require any prior information about the p_i . Unlike classical algorithms, our algorithm has an initial stage dedicated solely to estimating p_1 and p_2 . Classical algorithms do not normally need to have such a stage because classical arm samples, stored in memory, contain information about *both* the location of the best arm as well as p_1 and p_2 . However, in the quantum setting, we cannot simply obtain many such classical samples as this would collapse the quantum state, preventing quantum speedup.

Related work. Classically, a naive algorithm for best-arm identification is to simply sample each arm the same number of times and output the arm with the best empiri-

cal bias [EMM02]. This algorithm has complexity $O(\frac{n}{\Delta_2} \log(\frac{n}{\delta}))$ but is sub-optimal for most multi-armed bandit instances. Therefore, classical research on best-arm identification [EMM02; GGL12; JMN14; KKS13; MT04] has primarily focused on proving bounds of the form $\tilde{O}(H)$ (recall that $H := \sum_{i=2}^n \frac{1}{\Delta_i^2}$), which can be shown to be almost tight for every instance. The first work to provide an algorithm with such complexity is [EMM02], giving $O(H \log(\frac{n}{\delta}) + \sum_{i=2}^n \Delta_i^{-2} \log(\Delta_i^{-1}))$. This was further improved to $O(H \log(\frac{1}{\delta}) + \sum_{i=2}^n \Delta_i^{-2} \log \log(\Delta_i^{-1}))$ by [GGL12; JMN14; KKS13], which is almost optimal [MT04], except for the additive term of $\sum_{i=2}^n \Delta_i^{-2} \log \log(\Delta_i^{-1})$. More recent work [CL15; CLQ17] has focused on bringing down even this additive term by tightening both the upper and lower bounds, leaving behind a gap only of the order $\sum_{i=2}^n \Delta_i^{-2} \log \log(\min\{n, \Delta_i^{-1}\})$.

Prior work on quantum machine learning focuses primarily on supervised [LMR14; LMR13; RML14; LCW19] and unsupervised learning [LMR13; WKS15; Ami+18; KLLP19]. [DTB17; DLWT17; Jer+21] give quantum algorithms for general reinforcement learning with provable guarantees, but do not consider the best-arm identification problem. The only directly comparable previous work on quantum algorithms for best-arm identification that we are aware of are [CDKR20] and [WKS15].² By applying Grover’s algorithm, [CDKR20] shows that quantum computers can find the best arm with confidence $p_1 / \sum_{i=1}^n p_i$ quadratically faster than classical ones. However, [CDKR20] does not show how to find the best arm with arbitrarily high confidence. In fact, there is a relatively simple quantum algorithm, analogous to the naive classical algorithm, that can achieve arbitrarily high confidence with quadratic speedup in terms of n/Δ_2^2 . This algorithm, which appears in [WKS15, Fig. 3], works by using the quantum minimum finding algorithm of Dürr and Høyer [DH96] on top of quantum amplitude estimation [BHMT02]. As in the classical case, we show that this simple quantum algorithm is suboptimal for most multi-armed bandit instances. Specifically,

²The authors of [WKS15] did not explicitly mention “best-arm identification” in their paper but part of their work studied exactly this.

we show that a quantum algorithm can achieve quadratic speedup in terms of the parameter H .

3.2 Preliminaries

For any $p \in [0, 1]$, we define the *coin state* in \mathbb{C}^2 as $|\text{coin } p\rangle := \sqrt{p}|1\rangle + \sqrt{1-p}|0\rangle$. Measuring $|\text{coin } p\rangle$ in the computational basis gives 1 with probability p , hence the name. Throughout this chapter, we reserve symbols ϵ, δ for numbers in $(0, 1)$.

The rest of this section reviews variable-time amplitude amplification (VTAA) and estimation (VTAE), which are essential components of our algorithm. VTAA and VTAE are procedures applied on top of so-called “variable-time” quantum algorithms, which are formally defined as follows.

Definition 13 (Variable-time quantum algorithm, cf. [Amb12, Section 3.3] and [CKS17, Section 5.1]). *Let \mathcal{A} be a unitary operator acting on $\mathcal{H} = \mathbb{C}^N$ and let $|0\rangle_{\mathcal{H}} \in \mathcal{H}$ denote the first standard basis vector. We say \mathcal{A} is a variable-time quantum algorithm if the following conditions hold:*

1. \mathcal{A} can be decomposed as $\mathcal{A} = \mathcal{A}_m \mathcal{A}_{m-1} \cdots \mathcal{A}_1$, where the \mathcal{A}_i s are unitary operators acting on \mathcal{H} .
2. \mathcal{H} is a tensor product $\mathcal{H} = \mathcal{H}_C \otimes \mathcal{H}_A$, where \mathcal{H}_C is a tensor product of m single-qubit registers denoted $\mathcal{H}_{C_1}, \mathcal{H}_{C_2}, \dots, \mathcal{H}_{C_m}$.
3. Each \mathcal{A}_j is a controlled unitary that acts on the registers $\mathcal{H}_{C_j} \otimes \mathcal{H}_A$ controlled on the first $j - 1$ qubits of \mathcal{H}_C being $|0\rangle$.
4. $\mathcal{A}|0\rangle_{\mathcal{H}}$ is perpendicular to $|0\rangle_C := |0\rangle_{C_1} |0\rangle_{C_2} \cdots |0\rangle_{C_m}$.

In each iteration of the variable-time algorithm we shall construct, we use a subroutine that we call *gapped amplitude estimation* (GAE). Standard amplitude estimation [BHMT02] performs phase estimation on a particular unitary, and GAE is

essentially the same as “gapped phase estimation” [CKS17, Lemma 22] of that unitary. We recall the standard technique of amplitude estimation [BHMT02]:

Theorem 13 (Amplitude estimation). *Suppose \mathcal{O}_p is a unitary operator acting on $\mathcal{H}_B = \mathbb{C}^2$ with $\mathcal{O}_p |0\rangle_B = |\text{coin } p\rangle_B$. Then there is a unitary operator $\text{AE}(\epsilon, \delta)$ acting on $\mathcal{H}_B \otimes \mathcal{H}_P = \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes a}$, for some $a \in \mathbb{N}$, making $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ queries to \mathcal{O}_p and \mathcal{O}_p^\dagger that on input $|\text{coin } p\rangle_B |0\rangle_P$ prepares a state of the form*

$$|\text{coin } p\rangle_B \sum_{p'} \alpha_{p'} (|p'\rangle |\psi_{p'}\rangle)_P, \quad (3.4)$$

where $|\psi_{p'}\rangle$ are some arbitrary states and

$$\sqrt{\sum_{p: |p'-p|>\epsilon} |\alpha_{p'}|^2} \leq \delta. \quad (3.5)$$

Theorem 13 implies the following corollary.

Corollary 2 (Gapped amplitude estimation). *Let $l \in (0, 1)$. Suppose \mathcal{O}_p is a unitary operator acting on $\mathcal{H}_B = \mathbb{C}^2$ with $\mathcal{O}_p |0\rangle = |\text{coin } p\rangle$. Then there is a unitary operator $\text{GAE}(\epsilon, \delta; l)$ acting on $\mathcal{H}_B \otimes \mathcal{H}_C \otimes \mathcal{H}_P = \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes a}$, for some $a \in \mathbb{N}$, making $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ queries to \mathcal{O}_p and \mathcal{O}_p^\dagger , that on input $|\text{coin } p\rangle_B |0\rangle_C |0\rangle_P$, prepares a state of the form*

$$|\text{coin } p\rangle_B (\beta_0 |0\rangle_C |\gamma_0\rangle_P + \beta_1 |1\rangle_C |\gamma_1\rangle_P), \quad (3.6)$$

where $|\gamma_0\rangle$ and $|\gamma_1\rangle$ are some arbitrary states and $\beta_0, \beta_1 \in [0, 1]$ satisfy $\beta_0^2 + \beta_1^2 = 1$ with $\beta_1 \leq \delta$ if $p \geq l - \epsilon$ and $\beta_0 \leq \delta$ if $p < l - 2\epsilon$.

Proof. We first run $\text{AE}(\epsilon/4, \delta)$ on registers B and P . Then, in register C , we output 0 if the value p' stored in (the relevant part of) register P satisfies

$$|p' - (l - \epsilon)| \leq |p' - (l - 2\epsilon)|, \quad (3.7)$$

and output 1 if p' satisfies

$$|p' - (l - \epsilon)| > |p' - (l - 2\epsilon)|. \quad (3.8)$$

This gives the desired unitary procedure. Without loss of generality, we can assume β_0 and β_1 are real by absorbing their phases into $|\gamma_0\rangle$ and $|\gamma_1\rangle$ respectively. \square

Theorem 14 (Variable-time amplitude amplification [Amb12; CKS17] and estimation [CGJ19]). *Let $\epsilon, \delta \in (0, 1)$. Let $\mathcal{A} = \mathcal{A}_m \cdots \mathcal{A}_1$ be a variable-time quantum algorithm acting on $\mathcal{H} = \mathbb{C}^N$ of the form $\mathcal{H}_C \otimes \mathcal{H}_F \otimes \mathcal{H}_W$. Let $|0\rangle_{\mathcal{H}}$ denote the first standard basis vector in \mathcal{H} . Suppose \mathcal{O} is a unitary operator acting on \mathcal{H} and each \mathcal{A}_i is specified by a sequence of unitary operators $(U_0^{(i)}, \dots, U_{d_i}^{(i)})$ such that $\mathcal{A}_i = (U_{d_i}^{(i)} \mathcal{O}) \cdots (U_1^{(i)} \mathcal{O}) U_0$ makes d_i queries to \mathcal{O} . Let*

$$t_j := \sum_{i=1}^j d_i, \quad w_j := \|\Pi_{C_j} \mathcal{A}_j \cdots \mathcal{A}_1 |0\rangle_{\mathcal{H}}\|^2, \quad \text{and} \quad t_{\text{avg}} := \sqrt{\sum_{j=1}^m w_j t_j^2}, \quad (3.9)$$

where Π_{C_j} denotes the projector onto $|1\rangle$ in \mathcal{H}_{C_j} . Let

$$p_{\text{succ}} := \|\Pi_F \mathcal{A}_m \cdots \mathcal{A}_1 |0\rangle_{\mathcal{H}}\|^2 \quad \text{and} \quad |\psi_{\text{succ}}\rangle := \frac{\Pi_F \mathcal{A}_m \cdots \mathcal{A}_1 |0\rangle_{\mathcal{H}}}{\|\Pi_F \mathcal{A}_m \cdots \mathcal{A}_1 |0\rangle_{\mathcal{H}}\|}, \quad (3.10)$$

where Π_F projects onto $|1\rangle$ in \mathcal{H}_F . Let

$$Q := t_m \log(t_m) + \frac{t_{\text{avg}}}{\sqrt{p_{\text{succ}}}} \log(t_m). \quad (3.11)$$

Then the following holds.

1. There exists a quantum algorithm making $O(Q)$ queries to \mathcal{O} that outputs the state $|\psi_{\text{succ}}\rangle$ with probability $\geq 1/2$ and a bit indicating whether it succeeds.
2. There exists a quantum algorithm making $O(\frac{Q}{\epsilon} \log^2(t_m) \log \log(\frac{t_m}{\delta}))$ queries to \mathcal{O} that outputs $p'_{\text{succ}} \in \mathbb{R}$ such that $(1 - \epsilon)p_{\text{succ}} \leq p'_{\text{succ}} \leq (1 + \epsilon)p_{\text{succ}}$ with probability

$$\geq 1 - \delta.$$

3.3 Quantum exploration algorithm

In this section, we construct a quantum algorithm for best-arm identification and analyze its performance. Specifically, we show the following theorem.

Theorem 15. *Given a multi-armed bandit oracle \mathcal{O} as defined in Eq. (3.2) and confidence parameter $\delta \in (0, 1)$, there exists a quantum algorithm that outputs the best arm with probability $\geq 1 - \delta$ using $\tilde{O}(\sqrt{H})$ queries to \mathcal{O} , where H is as defined in Eq. (3.1).*

Throughout this section, we regard the oracle \mathcal{O} as fixed, so we sometimes omit explicit reference to it. All logs have base 2.

Our construction proceeds in three steps, which we now describe at a high level.

First, in Section 3.3.1, we construct a variable-time quantum algorithm denoted \mathcal{A} (Algorithm 1) that is initialized in a uniform superposition state $|u\rangle := \frac{1}{\sqrt{n}} \sum_{i \in [n]} |i\rangle$ (since initially we have no information about which arm is the best). Given an input interval $I = [l_2, l_1]$, \mathcal{A} “flags” arm indices in $S'_{\text{right}} := \{i \in [n] : p_i \geq l_1\}$ with a bit $f = 1$ and those in $S'_{\text{left}} := \{i \in [n] : p_i \leq l_2\}$ with a bit $f = 0$. The flag bit f is written to a separate flag register F , so that the state (approximately) becomes $\frac{1}{\sqrt{n}} (\sum_{i \in S'_{\text{right}}} |i\rangle |1\rangle_F + \sum_{i \in S'_{\text{left}}} |i\rangle |0\rangle_F + \sum_{i \in S'_{\text{middle}}} |i\rangle |\psi_i\rangle_F)$ for some qubit states $|\psi_i\rangle$, where $S'_{\text{middle}} := [n] - (S'_{\text{left}} \cup S'_{\text{right}}) = \{i \in [n] : l_2 < p_i < l_1\}$. The flag bit f stored in the F register indicates whether VTAA (resp. VTAE), when applied on \mathcal{A} , should ($f = 1$) or should not ($f = 0$) amplify (resp. estimate) that part of the state.

Second, in Section 3.3.2, we apply VTAA and VTAE on \mathcal{A} to construct two new algorithms called **Amplify** and **Estimate**, respectively. **Amplify** produces a uniform superposition of all those is with F register in $|1\rangle$, while **Estimate** counts the number of such is . More precisely, **Estimate** (approximately) counts the number of indices in

S'_{right} , as their F register is in $|1\rangle$, plus some (unknown) fraction of indices in S'_{middle} as dictated by the fraction of $|1\rangle$ in the (unknown) states $|\psi_i\rangle$.

Third, in [Section 3.3.3](#), we use `Estimate` as a subroutine in `Locate` ([Algorithm 2](#)) to find l_2, l_1 such that $p_2 < l_2 < l_2 + \Delta_2/4 \leq l_1 < p_1$. Then, running `Amplify` with these l_2, l_1 in `BestArm` ([Algorithm 4](#)) gives the state $|1\rangle$ containing the best-arm index because only p_1 is to the right of l_2 . `Locate` is a type of binary search that counts the number of indices in S'_{right} using `Estimate`. There is a technical difficulty here because `Estimate` actually counts the number of indices in S'_{right} plus some fraction of indices in S'_{middle} . Trying to fix this by simply setting $l_2 = l_1$, so that $S'_{\text{middle}} = \emptyset$, does not work as it would increase the cost of `Estimate`. We overcome this difficulty via the `Shrink` subroutine ([Algorithm 3](#)) of `Locate`, which employs a technique from recent work on quantum ground state preparation [[LT20](#)].

3.3.1 Variable-time quantum algorithm

We first construct a variable-time quantum algorithm ([Algorithm 1](#)) that we call \mathcal{A} throughout. \mathcal{A} uses the following registers: input register I ; bandit register B ; clock register $C = (C_1, \dots, C_{m+1})$, where each C_i is a qubit; ancillary amplitude estimation register $P = (P_1, \dots, P_m)$, where each P_i has $O(m)$ qubits; and flag register F . We set $m := \lceil \log(1/(l_1 - l_2)) \rceil + 2$ as assigned in [Algorithm 1](#).

\mathcal{A} is indeed a variable-time quantum algorithm according to [Definition 13](#). This is because we can write $\mathcal{A} = \mathcal{A}_{m+1}\mathcal{A}_m \cdots \mathcal{A}_1\mathcal{A}_0$ as a product of $m+2$ sub-algorithms, where \mathcal{A}_0 is the initialization step ([Line 4](#)), \mathcal{A}_j consists of the operations in iteration j of the for loop ([Lines 6–9](#)) for $j \in [m]$, and \mathcal{A}_{m+1} is the termination step ([Lines 10–11](#)). The state spaces \mathcal{H}_C and \mathcal{H}_A in [Definition 13](#) correspond to the state spaces of the C register and the remaining registers of \mathcal{A} , respectively. \mathcal{A}_{m+1} ensures that [Condition 4](#) of [Definition 13](#) is satisfied.

Algorithm 1: $\mathcal{A}(\mathcal{O}, l_2, l_1, \alpha)$

Input: Oracle \mathcal{O} as in Eq. (3.2); $0 < l_2 < l_1 < 1$; approximation parameter $0 < \alpha < 1$.

- 1 $\Delta \leftarrow l_1 - l_2$
- 2 $m \leftarrow \lceil \log \frac{1}{\Delta} \rceil + 2$
- 3 $a \leftarrow \frac{\alpha}{2mn^{3/2}}$
- 4 Initialize state to $\frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle_I |\text{coin } p_i\rangle_B |0\rangle_C |0\rangle_P |1\rangle_F$
- 5 **for** $j = 1, \dots, m$ **do**
- 6 $\epsilon_j \leftarrow 2^{-j}$
- 7 **if** register I is in state $|i\rangle$ and registers C_1, \dots, C_{j-1} are in state $|0\rangle$ **then**
- 8 Apply $\text{GAE}(\epsilon_j, a; l_1)$ with \mathcal{O}_{p_i} on registers B, C_j , and P_j
- 9 Apply controlled-NOT gate with control on register C_j and target on register F
- 10 **if** registers C_1, \dots, C_m are in state $|0\rangle$ **then**
- 11 Flip the bit stored in register C_{m+1}

We define the following three sets that partition $[n]$:

$$S_{\text{left}} := \{i \in [n] : p_i < l_1 - \Delta/2\}, \quad (3.12)$$

$$S_{\text{middle}} := \{i \in [n] : l_1 - \Delta/2 \leq p_i < l_1 - \Delta/8\}, \quad (3.13)$$

$$S_{\text{right}} := \{i \in [n] : p_i \geq l_1 - \Delta/8\}. \quad (3.14)$$

Note that these sets play the roles of S'_{left} , S'_{middle} , and S'_{right} mentioned at the start of Section 3.3. They can be regarded as functions of (the input to) \mathcal{A} .

In the following, we say that two pure states $|\psi\rangle$ and $|\phi\rangle$ are ϵ -close if $\| |\psi\rangle - |\phi\rangle \| \leq \epsilon$, where $\|\cdot\|$ denotes the ℓ_2 -norm.

Lemma 15 (Correctness of \mathcal{A}). *The output state $|\phi(\mathcal{A})\rangle$ of \mathcal{A} is (α/n) -close to*

$$\begin{aligned} |\psi(\mathcal{A})\rangle &:= \frac{1}{\sqrt{n}} \sum_{S_{\text{right}}} |i\rangle_I |\text{coin } p_i\rangle_B |\psi_i\rangle_{C,P} |1\rangle_F \\ &\quad + \frac{1}{\sqrt{n}} \sum_{S_{\text{left}}} |i\rangle_I |\text{coin } p_i\rangle_B |\psi_i\rangle_{C,P} |0\rangle_F \\ &\quad + \frac{1}{\sqrt{n}} \sum_{S_{\text{middle}}} |i\rangle_I |\text{coin } p_i\rangle_B (\beta_{i,1} |\psi_{i,1}\rangle_{C,P} |1\rangle_F + \beta_{i,0} |\psi_{i,0}\rangle_{C,P} |0\rangle_F) \end{aligned}$$

for some $\beta_{i,1}, \beta_{i,0} \in \mathbb{C}$ and states $|\psi_i\rangle, |\psi_{i,j}\rangle$. In particular, we have $|p_{\text{succ}} - p'_{\text{succ}}| \leq \frac{2\alpha}{n}$ where $p_{\text{succ}} := \|\Pi_F |\phi(\mathcal{A})\rangle\|^2$ and $p'_{\text{succ}} := \|\Pi_F |\psi(\mathcal{A})\rangle\|^2 = \frac{1}{n} (|S_{\text{right}}| + \sum_{i \in S_{\text{middle}}} |\beta_{i,1}|^2)$.

At a high level, at iteration j , [Line 8](#) approximately identifies those $i \in S_{\text{left}}$ with $p_i \in [l_1 - 2\epsilon_j, l_1 - \epsilon_j)$ and stops computation on these i s by setting their associated C registers to $|1\rangle$. [Line 9](#) then flags these i s by setting their associated F registers to $|0\rangle$, indicating failure. The formal proof below is mainly concerned with bounding the error in the aforementioned approximation.

Our proof is similar to that presented in [[CKS17](#), Section 5.3]. For comparison, it may be helpful to note that our states $|i\rangle_I |\text{coin } p_i\rangle$ are analogous to the matrix eigenstates $|\lambda\rangle$ in [[CKS17](#)]. The controlled-NOT operation on [Line 9](#) of our [Algorithm 1](#) takes the place of the simulation subroutine called “ W ” in [[CKS17](#), Lemma 23], which is more elaborate.

Proof. Let $\mathcal{A}_{\text{main}} := \mathcal{A}_{m+1} \cdots \mathcal{A}_1$ denote the part of \mathcal{A} after initialization. We will prove the following claim.

Claim. For all $i \in [n]$, $\mathcal{A}_{\text{main}} |i\rangle_I |\text{coin } p_i\rangle_B |0\rangle_{C,P,F}$ is $(\frac{\alpha}{n^{3/2}})$ -close to

$$\begin{aligned} \text{Case } i \in S_{\text{middle}}: & \quad |i\rangle_I |\text{coin } p_i\rangle_B (\beta_{i,1} |\psi_i\rangle_{C,P} |1\rangle_F + \beta_{i,0} |\psi_{i,0}\rangle_{C,P} |0\rangle_F) \\ & \quad \text{for some } \beta_{i,1}, \beta_{i,0} \in \mathbb{C} \text{ and states } |\psi_i\rangle, |\psi_{i,j}\rangle; \end{aligned}$$

$$\text{Case } i \in S_{\text{right}}: \quad |i\rangle_I |\text{coin } p_i\rangle_B |\psi_i\rangle_{C,P} |1\rangle_F;$$

Case $i \in S_{\text{left}}$: $|i\rangle_I |\text{coin } p_i\rangle_B |\psi_i\rangle_{C,P} |0\rangle_F$.

If the claim holds, then

$$|\phi(\mathcal{A})\rangle = \mathcal{A} |0\rangle_{I,B,C,P,F} = \mathcal{A}_{\text{main}} \frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle_I |\text{coin } p_i\rangle_B |0\rangle_{C,P,F} \quad (3.15)$$

satisfies

$$\| |\phi(\mathcal{A})\rangle - |\psi(\mathcal{A})\rangle \| \leq \frac{1}{\sqrt{n}} \cdot n \cdot \frac{\alpha}{n^{3/2}} = \frac{\alpha}{n}. \quad (3.16)$$

Moreover,

$$\begin{aligned} |p_{\text{succ}} - p'_{\text{succ}}| &= \left| (\sqrt{p_{\text{succ}}} + \sqrt{p'_{\text{succ}}}) \cdot (\sqrt{p_{\text{succ}}} - \sqrt{p'_{\text{succ}}}) \right| \\ &= (\sqrt{p_{\text{succ}}} + \sqrt{p'_{\text{succ}}}) \cdot \left| \|\Pi_F |\phi(\mathcal{A})\rangle\| - \|\Pi_F |\psi(\mathcal{A})\rangle\| \right| \\ &\leq 2 \|\Pi_F (|\phi(\mathcal{A})\rangle - |\psi(\mathcal{A})\rangle)\| \\ &\leq 2 \frac{\alpha}{n}, \end{aligned}$$

which proves the lemma.

We now prove the claim by considering each of its cases separately.

Case $i \in S_{\text{middle}}$. The claim holds because $\beta_{i,1} |\psi_{i,1}\rangle_{C,P} |1\rangle_F + \beta_{i,0} |\psi_{i,0}\rangle_{C,P} |0\rangle_F$ can represent any state on registers C, P, F .

Case $i \in S_{\text{left}}$. Let $j \in [m-1]$ be such that $l_1 - 2\epsilon_j \leq p_i < l_1 - \epsilon_j$. Note that this j uniquely exists by the definition of S_{left} , m , and ϵ_j . Then the state of the algorithm after the $(j-1)$ st iteration of the for-loop on **Line 5** is $(2(j-1)a)$ -close to

$$|i\rangle_I |\text{coin } p_i\rangle_B |0\rangle_C |\gamma_0^1\rangle_{P_1} \cdots |\gamma_0^{j-1}\rangle_{P_{j-1}} |0\rangle_{P_j \cdots P_m} |1\rangle_F, \quad (3.17)$$

where, for each $k \in [j - 1]$, the state $|0\rangle_{C_k} |\gamma_0^k\rangle_{P_k}$ corresponds to the state “ $|0\rangle_C |\gamma_0\rangle$ ” in Eq. (3.6). Note that we incur an error of at most $2a$ at each iteration that comes from running $\text{GAE}(\epsilon_j, a; l_1)$ (cf. the case where $\beta_1 \leq a$ in Corollary 2). This error accumulates additively.

The state after the j^{th} iteration is $(2ja)$ -close to

$$\begin{aligned} & \beta_0 |i\rangle_I |\text{coin } p_i\rangle_B |0\rangle_C |\gamma_0^1\rangle_{P_1} \cdots |\gamma_0^j\rangle_{P_j} |0\rangle_{P_{j+1}\cdots P_m} |1\rangle_F \\ & + \beta_1 |i\rangle_I |\text{coin } p_i\rangle_B |\mathbf{j}\rangle_C |\gamma_0^1\rangle_{P_1} \cdots |\gamma_0^j\rangle_{P_j} |0\rangle_{P_{j+1}\cdots P_m} |1\rangle_F, \end{aligned} \quad (3.18)$$

where $\mathbf{j} := 0^{j-1}10^{m-j}$ denotes the unary representation of the integer j .

At the $(j + 1)$ st iteration, the part of the state in the second line of Eq. (3.18) is unchanged because its register C indicates “stop”, but the part in the first line of Eq. (3.18) changes to being $(2(j + 1)a)$ -close to

$$\beta_0 |i\rangle_I |\text{coin } p_i\rangle_B |\mathbf{j} + \mathbf{1}\rangle_C |\gamma_0^1\rangle_{P_1} \cdots |\gamma_0^j\rangle_{P_j} |\gamma_0^{j+1}\rangle_{P_{j+1}} |0\rangle_{P_{j+2}\cdots P_m} |0\rangle_F. \quad (3.19)$$

Hence, the state after the $(j + 1)$ st iteration is $(2(j + 1)a)$ -close to

$$\begin{aligned} & \beta_0 |i\rangle_I |\text{coin } p_i\rangle_B |\mathbf{j} + \mathbf{1}\rangle_C |\gamma_0^1\rangle_{P_1} \cdots |\gamma_0^j\rangle_{P_j} |\gamma_0^{j+1}\rangle_{P_{j+1}} |0\rangle_{P_{j+2}\cdots P_m} |0\rangle_F \\ & + \beta_1 |i\rangle_I |\text{coin } p_i\rangle_B |\mathbf{j}\rangle_C |\gamma_0^1\rangle_{P_1} \cdots |\gamma_0^j\rangle_{P_j} |0\rangle_{P_{j+1}\cdots P_m} |0\rangle_F. \end{aligned} \quad (3.20)$$

Since the C register of all parts of the state in Eq. (3.20) indicates “stop”, the remaining iterations $j + 2, \dots, m$ of \mathcal{A} do not alter it. Hence the final state of \mathcal{A} is $(2ma)$ -close to the state in Eq. (3.20), which is of the form

$$|i\rangle_I |\text{coin } p_i\rangle_B |\psi_i\rangle_{C,P} |0\rangle_F. \quad (3.21)$$

Note that $2ma = \frac{\alpha}{n^{3/2}}$, so the closeness of approximation is as claimed.

Case $i \in S_{\text{right}}$. In this case, there does not exist a $j \in [m-1]$ such that $l_1 - 2\epsilon_j \leq p_i < l_1 - \epsilon_j$. Thus a simplified version of the argument above, in which we do not have to consider different cases according to the iteration number, shows that the resulting state is $(2ma)$ -close to a state of the same form as Eq. (3.21) but with the F register remaining in state 1.

This completes the proof of the claim and therefore of the lemma. \square

We now prove the following lemma that bounds the complexity of \mathcal{A} . The proof is similar to that presented in [CKS17, Section 5.4].

Lemma 16 (Complexity of \mathcal{A}). $\mathcal{A}(\mathcal{O}, l_2, l_1, \alpha)$ has the following complexities:

1. The j^{th} stopping time t_j of $\mathcal{A}_j \mathcal{A}_{j-1} \cdots \mathcal{A}_0$ is of order $\sum_{k=1}^j \frac{1}{\epsilon_k} \log \frac{1}{a} \leq 2^{j+1} \log \frac{1}{a}$.

In particular, $t_{m+1} = O(\frac{1}{\Delta} \log \frac{1}{a})$.

2. The average stopping time squared, t_{avg}^2 , is of order

$$\frac{1}{n} \left(\frac{|S_{\text{right}}|}{\Delta^2} + \sum_{i \in S_{\text{left}} \cup S_{\text{middle}}} \frac{1}{(l_1 - p_i)^2} \right) \log^2 \left(\frac{1}{a} \right). \quad (3.22)$$

Proof. For the first claim, note first that \mathcal{A}_0 and \mathcal{A}_{m+1} use a constant number of queries (1 and 0, respectively), so we can ignore them. For $k \in [m]$, \mathcal{A}_k only uses queries to perform $\text{GAE}(\epsilon_k, d; l_1)$, which takes $O(\frac{1}{\epsilon_k} \log \frac{1}{a})$ queries. Therefore t_j , the number of queries in $\mathcal{A}_j \mathcal{A}_{j-1} \cdots \mathcal{A}_1$, is of order

$$\sum_{k=1}^j \frac{1}{\epsilon_k} \log \left(\frac{1}{a} \right) = \sum_{k=1}^j 2^k \log \left(\frac{1}{a} \right) \leq 2^{j+1} \log \left(\frac{1}{a} \right) \quad (3.23)$$

because $\epsilon_k = 2^{-k}$ by definition. In addition, we have $t_m = O(\frac{1}{\Delta} \log \frac{1}{a})$ because $m = \lceil \log \frac{1}{\Delta} \rceil + 2$ by definition. The first claim follows.

For the second claim, we have

$$t_{\text{avg}}^2 = \sum_{j=1}^m w_j t_j^2 = \sum_{j=1}^m \left\| \Pi_{C_j} \mathcal{A}_j \cdots \mathcal{A}_1 \frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle_I |\text{coin } p_i\rangle_B |0\rangle_C |0\rangle_P |1\rangle_F \right\|^2 t_j^2 \quad (3.24)$$

$$= \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m w_{i,j} t_j^2 \quad (3.25)$$

$$= \frac{1}{n} \sum_{i=1}^n \tau_i^2, \quad (3.26)$$

where $w_{i,j} := \left\| \Pi_{C_j} \mathcal{A}_j \cdots \mathcal{A}_1 |i\rangle_I |\text{coin } p_i\rangle_B |0\rangle_C |0\rangle_P |1\rangle_F \right\|^2 \in [0, 1]$ and $\tau_i := \sum_{j=1}^m w_{i,j} t_j^2$.

Note that $w_{i,j}$ can be thought of as the probability that \mathcal{A} stops at the end of iteration j if initialized with arm i ; τ_i^2 can be thought of as the squared average stopping time of \mathcal{A} if initialized with arm i .

For each fixed i , we consider τ_i^2 according to the following three cases.

Case $i \in S_{\text{right}}$. We have $\sum_{j=1}^m w_{i,j} = 1$, so $\tau_i^2 \leq t_m^2 = O(2^{2m} \log^2(\frac{1}{a})) = O(\frac{1}{\Delta^2} \log^2(\frac{1}{a}))$ because $m = \lceil \log \frac{1}{\Delta} \rceil + 2$ by definition.

Case $i \in S_{\text{middle}}$. We still have $\tau_i^2 = O(\frac{1}{\Delta^2} \log^2(\frac{1}{a}))$ as in the case $i \in S_{\text{right}}$, by exactly the same argument. But by the definition of S_{middle} , we have $l_1 - p_i \leq \Delta/2$, so we can also write $\tau_i^2 = O(\frac{1}{(l_1 - p_i)^2} \log^2(\frac{1}{a}))$.

Case $i \in S_{\text{left}}$. For $i \in S_{\text{left}}$, let $j \in [m-1]$ be such that $l_1 - 2\epsilon_j \leq p_i < l_1 - \epsilon_j$ as in the proof of [Lemma 15](#).

We know that after the $(j+1)$ st iteration, the state is $(m\alpha = \alpha/n)$ -close to the state in [Eq. \(3.20\)](#) on which the algorithm terminates. Therefore, the probability $w_{i,j+1}$ of terminating after the $(j+1)$ st iteration is $1 - O((\alpha/n)^2)$. It can also be seen that the probability $w_{i,j+r}$ of terminating after the $(j+r)$ th iteration is $(1 -$

$O((\alpha/n)^2) \cdot O((\alpha/n)^{2(r-1)})$. Hence

$$\tau_i^2 \leq t_{j+1}^2 + O\left(\sum_{r=2}^{m-j} \left(\frac{\alpha}{n}\right)^{2(r-1)} t_{j+r}^2\right) = O(t_{j+1}^2) = O\left(\frac{\log^2(\frac{1}{a})}{\epsilon_{j+1}}\right) = O\left(\frac{\log^2(\frac{1}{a})}{(l_1 - p_i)^2}\right),$$

where we used $\epsilon_{j+1} = \epsilon_j/2 \geq (l_1 - p_i)/4$ for the last inequality.

Substituting the above results into [Eq. \(3.26\)](#) shows that t_{avg}^2 is of order

$$\frac{1}{n} \left(\frac{|S_{\text{right}}|}{\Delta^2} + \sum_{i \in S_{\text{left}} \cup S_{\text{middle}}} \frac{1}{(l_1 - p_i)^2} \right) \cdot \log^2\left(\frac{1}{a}\right) \quad (3.27)$$

as desired. \square

3.3.2 Applying VTAA and VTAE to \mathcal{A}

In this subsection, we fix algorithm \mathcal{A} and its input parameters. We always assume that $|S_{\text{right}}| > 0$, so $|S_{\text{right}}| \geq 1$, which we need for some of the following results to hold. This is without loss of generality as we can always add an artificial arm 0 to \mathcal{O} with bias $p_0 = 1$, as we do on [Line 3](#) of [Algorithm 3](#).

We apply³ VTAA and VTAE ([Theorem 14](#)) on our variable-time quantum algorithm \mathcal{A} to prepare the state $|\psi_{\text{succ}}\rangle$ and estimate the probability p_{succ} , respectively. This gives two new algorithms **Amplify** and **Estimate** with the following performance guarantees.

Lemma 17 (Correctness and complexity of **Amplify**(\mathcal{A}, δ), **Estimate**($\mathcal{A}, \epsilon, \delta$)). *Let $\mathcal{A} = \mathcal{A}(\mathcal{O}, l_2, l_1, 0.01\delta)$. Then **Amplify**(\mathcal{A}, δ) uses $O(Q)$ queries to output an index $i \in S_{\text{right}} \cup S_{\text{middle}}$ with probability $\geq 1 - \delta$, and **Estimate**($\mathcal{A}, \epsilon, \delta$) uses $O(Q/\epsilon)$ queries to output an estimate r of p'_{succ} (defined in [Lemma 15](#)) such that*

$$(1 - \epsilon) \left(p'_{\text{succ}} - \frac{0.1}{n} \right) < r < (1 + \epsilon) \left(p'_{\text{succ}} + \frac{0.1}{n} \right) \quad (3.28)$$

³The state spaces \mathcal{H}_C , \mathcal{H}_F , and \mathcal{H}_W in [Theorem 14](#) correspond to the state spaces of the C , F , and remaining registers of \mathcal{A} , respectively.

with probability $\geq 1 - \delta$, where

$$Q = \left(\frac{1}{\Delta^2} + \frac{1}{|S_{\text{right}}|} \sum_{S_{\text{left}} \cup S_{\text{middle}}} \frac{1}{(l_1 - p_i)^2} \right)^{1/2} \text{poly} \left(\log \left(\frac{n}{\delta \Delta} \right) \right). \quad (3.29)$$

The lemma essentially follows from applying [Lemma 15](#) and [Lemma 16](#) to [Theorem 14](#). We give the details below.

Proof. We set the approximation parameter in \mathcal{A} to be $\alpha = c\delta$ for some constant $c < 0.05$ to be determined later. Then $\alpha < 0.05$.

We apply VTAA ([Theorem 14](#)) on \mathcal{A} . This gives an algorithm that outputs a state $|\psi_{\text{succ}}\rangle$ that is $(\frac{\alpha}{n} = \frac{c\delta}{n})$ -close to the (normalized) state proportional to

$$\begin{aligned} & \Pi_F |\psi(\mathcal{A})\rangle \\ &= \frac{1}{\sqrt{n}} \left(\sum_{i \in S_{\text{right}}} |i\rangle_I |\text{coin } p_i\rangle_B |\psi_i\rangle_{C,P} |1\rangle_F + \sum_{i \in S_{\text{middle}}} \alpha_{i,1} |i\rangle_I |\text{coin } p_i\rangle_B |\psi_{i,1}\rangle_{C,P} |1\rangle_F \right) \end{aligned}$$

with success probability at least $1/2$ and a bit indicating success or failure. Now, we repeat the entire procedure $O(\log \frac{1}{\delta})$ times to prepare $|\psi_{\text{succ}}\rangle$ at least once with probability $\geq 1 - \delta/2$. Once $|\psi_{\text{succ}}\rangle$ has been successfully prepared, as indicated by the algorithm, we measure its index register I . This procedure outputs an arm index in $S_{\text{right}} \cup S_{\text{middle}}$ with probability $\geq (1 - \delta/2) \cdot (1 - 2c\delta/n)$ which is $\geq 1 - \delta$ for $c \leq 1/4$ sufficiently small. So, as we also need $c < 0.05$, we choose $c = 0.01$. We call this procedure $\text{Amplify}(\mathcal{A}, \delta)$.

Let us consider the query complexity of $\text{Amplify}(\mathcal{A}, \delta)$. We have

$$t_{m+1} = O\left(\frac{1}{\Delta} \log\left(\frac{1}{a}\right)\right) = O\left(\frac{1}{\Delta} \log\left(n \log\left(\frac{1}{\Delta}\right)\right)\right) \quad (3.30)$$

because $a = \frac{\alpha}{2^{(\lceil \log(1/\Delta) \rceil + 2)n^{3/2}}}$ by definition. We also have

$$p_{\text{succ}} \geq p'_{\text{succ}} - \frac{2\alpha}{n} \geq \frac{|S_{\text{right}}|}{n} - \frac{0.1}{n} > \frac{|S_{\text{right}}|}{2n}, \quad (3.31)$$

where we used the assumption $|S_{\text{right}}| > 0$ for the last inequality. Lastly, t_{avg}^2 is of order given in Eq. (3.22) (reproduced in Eq. (3.27) above). Therefore, substituting all these bounds into Eq. (3.11) of Theorem 14, we see that $\text{Amplify}(\mathcal{A}, \delta)$ has query complexity of order

$$\begin{aligned} & \left(\frac{1}{\Delta^2} + \frac{1}{|S_{\text{right}}|} \sum_{i \in S_{\text{left}} \cup S_{\text{middle}}} \frac{1}{(l_1 - p_i)^2} \right)^{1/2} \\ & \times \log\left(\frac{n}{\delta} \log \frac{1}{\Delta}\right) \cdot \log\left(\frac{1}{\Delta} \log\left(\frac{n}{\delta} \log\left(\frac{1}{\Delta}\right)\right)\right) \cdot \log\left(\frac{1}{\delta}\right). \end{aligned} \quad (3.32)$$

We also apply VTAE (Theorem 14) with multiplicative accuracy ϵ and confidence δ on \mathcal{A} . This gives an algorithm, $\text{Estimate}(\mathcal{A}, \epsilon, \delta)$, that outputs an estimate r of p_{succ} with multiplicative accuracy ϵ (i.e., $|r - p_{\text{succ}}| < \epsilon p_{\text{succ}}$) with probability $\geq 1 - \delta$. Combining $|r - p_{\text{succ}}| < \epsilon p_{\text{succ}}$ with $|p_{\text{succ}} - p'_{\text{succ}}| \leq \frac{2\alpha}{n} < \frac{0.1}{n}$ gives

$$(1 - \epsilon)\left(p'_{\text{succ}} - \frac{0.1}{n}\right) < r < (1 + \epsilon)\left(p'_{\text{succ}} + \frac{0.1}{n}\right) \quad (3.33)$$

as claimed.

The query complexity of $\text{Estimate}(\mathcal{A}, \epsilon, \delta)$ is

$$\begin{aligned} & \text{Eq. (3.32)} \cdot \frac{1}{\epsilon} \log^2(t_{m+1}) \log\left(\log\left(\frac{t_{m+1}}{\delta}\right)\right) \\ & = \text{Eq. (3.32)} \cdot O\left(\frac{1}{\epsilon} \text{poly}\left(\log\left(\frac{n}{\delta\Delta}\right)\right)\right) \end{aligned} \quad (3.34)$$

according to Theorem 14 and Eq. (3.30). \square

3.3.3 Quantum algorithm for best-arm identification

In this subsection, we use Amplify and Estimate to construct three algorithms (Algorithms 2–4) that work together to identify the best arm following the outline at the start of Section 3.3.

Lemma 18 (Correctness and complexity of Algorithm 2). *Let $0 < \delta < 1$. Then*

Algorithm 2: Locate(\mathcal{O}, δ)

Input: Oracle \mathcal{O} as in Eq. (3.2); confidence parameter $0 < \delta < 1$.

- 1 $I_1, I_2 \leftarrow [0, 1]$
- 2 $\delta \leftarrow \delta/8$
- 3 **while** $\min I_1 - \max I_2 < 2|I_1|$ **do**
- 4 $I_1 \leftarrow \text{Shrink}(\mathcal{O}, 1, I_1, \delta)$
- 5 $I_2 \leftarrow \text{Shrink}(\mathcal{O}, 2, I_2, \delta)$
- 6 $\delta \leftarrow \delta/2$
- 7 **return** I_1, I_2

Algorithm 3: Shrink($\mathcal{O}, k, I, \delta$)

Input: Oracle \mathcal{O} as in Eq. (3.2); $k \in \{1, 2\}$; interval $I = [a, b]$; confidence parameter $0 < \delta < 1$.

- 1 $\epsilon \leftarrow (b - a)/5$
- 2 $\delta \leftarrow \delta/2$
- 3 Append arm $i = 0$ with bias $p_0 = 1$ to \mathcal{O} ; call the resulting oracle \mathcal{O}'
- 4 Construct variable-time quantum algorithms $\mathcal{A}_1, \mathcal{A}_2$:
- 5 $\mathcal{A}_1 \leftarrow \mathcal{A}(\mathcal{O}', l_2 = a + \epsilon, l_1 = a + 3\epsilon, 0.01\delta)$
- 6 $\mathcal{A}_2 \leftarrow \mathcal{A}(\mathcal{O}', l_2 = a + 2\epsilon, l_1 = a + 4\epsilon, 0.01\delta)$
- 7 $r_1 \leftarrow \text{Estimate}(\mathcal{A}_1, \epsilon = 0.1, \delta)$
- 8 $r_2 \leftarrow \text{Estimate}(\mathcal{A}_2, \epsilon = 0.1, \delta)$
- 9 $B_1 \leftarrow \mathbf{1}(r_1 > \frac{k+0.5}{n+1})$; $B_2 \leftarrow \mathbf{1}(r_2 > \frac{k+0.5}{n+1})$
- 10 **switch** (B_1, B_2) **do**
- 11 **case** $(0, 0)$: $I \leftarrow [a, a + 3\epsilon]$
- 12 **case** $(0, 1)$: $I \leftarrow [a + \epsilon, a + 4\epsilon]$
- 13 **case** $(1, 0)$: $I \leftarrow [a + \epsilon, a + 4\epsilon]$
- 14 **case** $(1, 1)$: $I \leftarrow [a + 2\epsilon, a + 5\epsilon = b]$
- 15 **return** I

the event $E := \{p_1 \in I_1 \text{ and } p_2 \in I_2 \text{ in all iterations of the while loop}\}$ holds with probability $\geq 1 - \delta$. When E holds, Algorithm 2 also satisfies the following for $k \in \{1, 2\}$:

1. its while loop (Line 3) breaks at or before the end of iteration $\lceil \log_{5/3}(\frac{1}{\Delta_2}) \rceil + 3$ and then returns I_k with $p_k \in I_k$ and $\min I_1 - \max I_2 \geq 2|I_1|$; during the while loop, we always have $|I_1| = |I_2| \geq \Delta_2/8$; and
2. it uses $O(\sqrt{H} \text{ poly}(\log(\frac{n}{\delta\Delta_2})))$ queries.

Proof. From the first claim of Lemma 19, we see that the probability of E^c is at most

Algorithm 4: BestArm(\mathcal{O}, δ)

Input: Oracle \mathcal{O} as in Eq. (3.2); confidence parameter $0 < \delta < 1$.

- 1 $\delta \leftarrow \delta/2$
- 2 $I_1, I_2 \leftarrow \text{Locate}(\mathcal{O}, \delta)$
- 3 $l_1 \leftarrow \min I_1$ (left endpoint of I_1)
- 4 $l_2 \leftarrow \max I_2$ (right endpoint of I_2)
- 5 Construct variable-time quantum algorithm \mathcal{A} :
- 6 $\mathcal{A} \leftarrow \mathcal{A}(\mathcal{O}, l_2, l_1, 0.01\delta)$
- 7 $i \leftarrow \text{Amplify}(\mathcal{A}, \delta)$
- 8 **return** i

$\frac{\delta}{4} \sum_{i=0}^{\infty} 2^{-i} = \delta/2$, where the geometric series arises because of Line 6. Henceforth, we assume E .

Consider the first claim. For given intervals I_2, I_1 , let us write

$$\text{gap}(I_2, I_1) := \min I_1 - \max I_2. \quad (3.35)$$

At the end of iteration $i \geq 1$ (i.e., after Line 6), we have $|I_k| = (3/5)^i$ by the first claim of Lemma 19. At the end of iteration $\lceil \log_{5/3}(\frac{1}{\Delta_2}) \rceil + 3$, we have $|I_k| < \Delta_2/4$, so $\text{gap}(I_2, I_1) > \Delta_2 - 2\Delta_2/4 = \Delta_2/2 > 2|I_1|$ because $p_k \in I_k$. Therefore the while loop must break at this point if it has not done so earlier. For the returned I_k , we clearly have $p_k \in I_k$ because E holds, and $\text{gap}(I_2, I_1) > 2|I_1|$ because the while loop has broken. During the while loop, because $|I_k|$ decreases from iteration to iteration, we always have $|I_k| \geq (3/5)^{\lceil \log_{5/3}(\Delta_2^{-1}) \rceil + 3} \geq \Delta_2/8$. Note that $|I_1| = |I_2|$ because, at each iteration of the while loop, the Shrink subroutine always shrinks intervals by the same factor of $3/5$ and $|I_1| = |I_2| = 1$ initially.

Now, consider the second claim. From the first claim, we know that the while loop breaks at or before the end of iteration $\lceil \log_{5/3}(\Delta_2^{-1}) \rceil + 3$, and we always have $1/\delta_i = O(2^{\log_{5/3}(\Delta_2^{-1})}/\delta) = O(\Delta_2^{-2}/\delta)$, where $\delta_i = \delta/2^{2+i}$ is the confidence parameter in Shrink at iteration i . Therefore, using the second claim of Lemma 19, the total

number of queries used is at most

$$O(\log(\Delta_2^{-1})) \cdot O\left(\sqrt{H} \cdot \text{poly}\left(\log\left(\frac{n}{\Delta_2} \cdot \frac{\Delta_2^{-2}}{\delta}\right)\right)\right), \quad (3.36)$$

which is $O(\sqrt{H} \cdot \text{poly}(\log(\frac{n}{\delta\Delta_2})))$ as desired. \square

Lemma 19 (Correctness and complexity of [Algorithm 3](#)). *Fix $k \in \{1, 2\}$, an interval $I = [a, b]$, and a confidence parameter $0 < \delta < 1$. Suppose that $p_k \in I$ and $|I| \geq \Delta_2/8$. Then [Algorithm 3](#)*

1. *outputs an interval J with $|J| = \frac{3}{5}|I|$ such that $p_k \in J$ with probability $\geq 1 - \delta$, and*
2. *uses $O(\sqrt{H} \text{poly}(\log(\frac{n}{\delta\Delta_2})))$ queries.*

Proof. Throughout, we fix $k \in \{0, 1\}$.

For the first claim, it is clear that $|J| = 3|I|/5$ because all the intervals appearing in [Lines 11–14](#) have length 3ϵ . Our proof that $p_k \in J$ with high probability is similar to that in [\[LT20, Section 4\]](#) (see its [Table 2](#) and [Algorithm 1](#)).

Let us write $x_j = a + j\epsilon$ for $j = 0, \dots, 5$, so that $x_0 = a$ and $x_5 = b$. Let E be the event that both [Estimates](#) in [Lines 7](#) and [8](#) return the correct result. The probability of E^c is at most δ so we restrict to the case of E in the following paragraph.

For $j \in \{1, 2\}$, we can use [Eq. \(3.28\)](#) in [Lemma 17](#) to see that if $p_k \leq x_j$, then $B_j = 0$ because $r_j \leq (1 + 0.1)(\frac{k}{n+1} + \frac{0.1}{n+1}) < \frac{k+0.5}{n+1}$, whereas if $p_k \geq x_{j+2}$, then $B_j = 1$ because $r_j \geq (1 - 0.1)(\frac{k+1}{n+1} - \frac{0.1}{n+1}) > \frac{k+0.5}{n+1}$. Here we use the fact $k \in \{1, 2\}$. By considering the contrapositive of the previous two if-then statements, we establish the first claim.

We now prove the second claim. Since we run [Estimate](#) with constant multiplicative error $\epsilon = 0.1$, its query complexity is of order [Eq. \(3.32\)](#), which is

$$\frac{1}{\Delta^2} + \frac{1}{|S_{\text{right}}|} \sum_{i \in S_{\text{left}} \cup S_{\text{middle}}} \frac{1}{(l_1 - p_i)^2} \quad (3.37)$$

up to polylogarithmic factors, where we recall that $\Delta = l_1 - l_2$. In addition, we recall

$$S_{\text{left}} \cup S_{\text{middle}} = \{i : p_i < l_1 - \Delta/8\} \quad (3.38)$$

from [Eq. \(3.13\)](#) and [Eq. \(3.14\)](#). Note that $|S_{\text{right}}| > 0$ because we appended an arm with bias $p_0 = 1$.

By assumption, $|I| \geq \Delta_2/8$. So, in view of [Lines 5](#) and [6](#), we have $\Delta = 2\epsilon = 2|I|/5 \geq \Delta_2/20$. Therefore $1/\Delta^2 = O(1/\Delta_2^2)$.

We also need to compare $p_1 - p_i$ with $l_1 - p_i$ for $i \in S_{\text{left}} \cup S_{\text{middle}}$. By definition, we have $p_i < l_1 - \Delta/8$, so $l_1 - p_i > \Delta/8$. Note that we also have $|p_k - l_1| \leq |I| = 5\Delta/2$ because $p_k \in I$ by assumption and $l_1 \in I$ by definition. If $k = 1$, this says $|p_1 - l_1| \leq 5\Delta/2$. If $k = 2$, this says $|p_2 - l_1| \leq 5\Delta/2$, but we can still bound

$$|p_1 - l_1| \leq \Delta_2 + |p_2 - l_1| \leq 20\Delta + 5\Delta/2 < 25\Delta. \quad (3.39)$$

So regardless of whether $k = 1$ or $k = 2$, we have that $|p_1 - l_1| < 25\Delta$. Therefore

$$\frac{p_1 - p_i}{l_1 - p_i} = 1 + \frac{p_1 - l_1}{l_1 - p_i} < 1 + \frac{25\Delta}{\Delta/8} = 201, \quad (3.40)$$

and so $1/(l_1 - p_i)^2 = O(1/(p_1 - p_i)^2)$. Hence we have established the second claim. \square

The following theorem is equivalent to [Theorem 15](#).

Theorem 16 (Correctness and complexity of [Algorithm 4](#)). *Let $0 < \delta < 1$. Then, with probability $\geq 1 - \delta$, [Algorithm 4](#)*

1. *outputs the best arm, and*
2. *uses $O(\sqrt{H} \text{poly}(\log(\frac{n}{\delta\Delta_2})))$ queries.*

Proof. Note that δ is halved at the beginning, on [Line 1](#). For the first claim, we know from the first claim of [Lemma 18](#) that, with probability $\geq 1 - \delta/2$, the two intervals I_k assigned on [Line 2](#) have $\min I_1 - \max I_2 \geq 2|I_1| \geq \Delta_2/4$ and $p_k \in I_k$. Assuming

this holds, we have $p_2 < l_2 < l_2 + \Delta_2/4 \leq l_1 < p_1$ for the endpoints l_k assigned in Lines 3 and 4. This means that the variable-time quantum algorithm \mathcal{A} defined on Line 6 has $S_{\text{right}} \cup S_{\text{middle}} = \{1\}$, so $\text{Amplify}(\mathcal{A}, \delta/2)$ returns index 1 with probability $\geq 1 - \delta/2$. Therefore, the overall probability of Algorithm 4 returning the best arm is at least $1 - \delta$.

The second claim follows from adding the number of queries used in $\text{Locate}(\mathcal{O}, \delta/2)$ (see Lemma 18) and $\text{Amplify}(\mathcal{A}, \delta/2)$ (see Lemma 17, using $l_1 - l_2 \geq \Delta_2/4$). \square

By establishing Theorem 16, we have established Theorem 15, the main claim of Section 3.3. As discussed previously, the main complexity measure of interest in the classical case is H , and we see that we get a quadratic speedup in terms of this parameter.

We can see that the poly-logarithmic factor has degree about 6 from Eq. (3.32), Eq. (3.34), and Eq. (3.36). It would be interesting to reduce this degree. A more fundamental challenge is to remove the variable n that appears in our log factors. In the classical case, n was already removed from log factors in early work [EMM02] by a procedure called “median elimination”. However, quantizing the median elimination framework is nontrivial, as the query complexity for outputting the $n/2$ smallest items among n elements is $\Theta(n)$ [Amb10, Theorem 1], exceeding our budget of $O(\sqrt{n})$.

As corollaries of our main results in the fixed confidence setting, we provide results on best-arm identification in the PAC (Probably Approximately Correct) and fixed-budget settings. In the (ϵ, δ) -PAC setting, the goal is to identify an arm i with $p_i \geq p_1 - \epsilon$ with probability $\geq 1 - \delta$. We provide an algorithm with query complexity $O(\sqrt{\min\{\frac{n}{\epsilon^2}, H\}} \text{poly}(\log(\frac{n}{\delta\Delta_2})))$. In the fixed-budget setting, the goal is to identify the best arm with high probability for a fixed number T of total queries (the budget). As a direct corollary of Theorem 15, when H is known in advance, there is an algorithm that returns the best arm with probability $\geq 1 - \exp(-\Omega(T/\sqrt{H}))$ by using a reduction similar to that from Monte Carlo to Las Vegas algorithms. See

Section 3.3.4 for details.

3.3.4 Corollaries for the PAC and fixed-budget settings

PAC setting. Another setting often considered in the classical literature is the (ϵ, δ) -PAC setting. The goal here is to identify an arm i with $p_i \geq p_1 - \epsilon$ with probability $\geq 1 - \delta$. Our best-arm identification algorithm can be modified to work in this setting as well. More precisely, we can modify `Locate` (Algorithm 2) by adding a breaking condition to the while loop when $|I_1|$ (or equivalently $|I_2|$) is smaller than ϵ . The resulting algorithm finds an ϵ -optimal arm with query complexity $O(\sqrt{\min\{\frac{n}{\epsilon^2}, H\}} \cdot \text{poly}(\log(\frac{n}{\delta\Delta_2})))$. Note that our modification means that the `Amplify` step in Algorithm 4 takes an input interval I with $|I| = l_1 - l_2 \in [\epsilon/2, \epsilon]$. The correctness and complexity follow directly from Lemma 15 and Lemma 17. For comparison, [EMM02] gave a classical PAC algorithm with complexity $O(\frac{n}{\epsilon^2} \log(\frac{n}{\delta}))$, which was later improved to $O(\sum_{i=1}^n \min\{\epsilon^{-2}, \Delta_i^{-2}\} \cdot \log(\frac{n}{\delta\Delta_2}))$ by [GGL12].

Fixed-budget setting. As mentioned near the end of Section 3.3, by using a reduction similar to that from Monte Carlo to Las Vegas algorithms, we can construct a fixed-budget algorithm from our fixed-confidence one. For completeness, we state and prove the following result:

Lemma 20 (Reduction to fixed confidence). *Let \mathcal{O} be a multi-armed bandit oracle. Suppose that for any $\delta \in (0, 1)$, we have an algorithm $\mathcal{A}_c(\delta)$ that with probability $\geq 1 - \delta$, terminates before using $T_c(\delta)$ queries to \mathcal{O} and returns the best-arm index $i^* = 1$. Suppose that we also know $T_c(\delta)$. Then, for any positive integer T , we can construct an algorithm $\mathcal{A}_b(T)$ that returns $i^* = 1$ with probability $\geq \min_{\delta \in (0, 1)} \exp(-\lfloor T/T_c(\delta) \rfloor D(\frac{1}{2} \parallel \delta))$ using at most T queries to \mathcal{O} , where $D(p \parallel q)$ is the relative entropy between Bernoulli random variables with bias p and q .*

Proof. Since $T_c(\delta)$ is known, consider the modified version of the fixed-confidence

algorithm where the algorithm is forced to halt and return some blank symbol “ \perp ” if the running time exceeds $T_c(\delta)$. We refer to the modified algorithm as $A'_c(\delta)$. $A'_c(\delta)$ returns the best-arm index $i^* = 1$ with probability $\geq 1 - \delta$ and returns some symbol in $\{2, \dots, n, \perp\}$ with probability $\leq \delta$.

For any T , we construct $\mathcal{A}_b(T)$ as follows. Pick some $\delta \in (0, 1)$, run $A'_c(\delta)$ $m := \lfloor T/T_c(\delta) \rfloor$ times, and take a majority vote over the outcomes. The failure probability can be upper bounded by the probability that i^* is observed fewer than $m/2$ times. The Chernoff bound upper bounds the latter probability by $\exp(-mD(\frac{1}{2}||\delta)) = \exp(-\lfloor T/T_c(\delta) \rfloor D(\frac{1}{2}||\delta))$. But δ was arbitrary, so we can take the δ that minimizes this upper bound. \square

As a direct corollary of [Theorem 15](#) and [Lemma 20](#), we see that when H (therefore T_c) is known in advance, for sufficiently large T , there is a quantum algorithm using at most T queries that returns the best-arm with probability $\geq 1 - \exp(-\Omega(T/\sqrt{H}))$.

3.4 Quantum lower bound

We also establish a lower bound for the quantum best-arm identification problem. Our lower bound shows that the algorithm of [Theorem 15](#) is optimal up to polylogarithmic factors.

Theorem 17. *Let $p \in (0, 1/2)$. For arbitrary biases $p_1, \dots, p_n \in [p, 1 - p]$, any quantum algorithm that identifies the best arm requires $\Omega(\sqrt{H})$ queries to the multi-armed bandit oracle \mathcal{O} .*

To prove this lower bound, we use the quantum adversary method to show quantum hardness of distinguishing oracles corresponding to the following n bandits. In the 1st bandit, we assign bias p_i to arm i for all i . In the x^{th} bandit for $x \in \{2, \dots, n\}$, we assign bias $p_1 + \eta$ to arm x and p_i to arm i for all $i \neq x$, where η is an appropriately

chosen parameter. This hard set of bandits is inspired by the proof of a corresponding classical lower bound [MT04, Theorem 5].

For completeness, we first describe the adversary method. The adversary method was historically used to show that the adversary *quantity* in Chapter 2 lower bounds quantum query complexity when the queries are made to an oracle encoding a *string*. We cannot directly employ the adversary quantity here because a best-arm identification algorithm makes queries to a multi-armed bandit oracle that encodes a *probability distributions* (see Eq. (3.2)).⁴

Suppose we have n multi-armed bandit oracles $\mathcal{O}_1, \dots, \mathcal{O}_n$ that correspond to n multi-armed bandits where the best arm is located at a different index in each. Suppose that we also have a best-arm identification algorithm \mathcal{A} that uses no more than T queries to identify the best arm with probability $\geq 1 - \delta$.

The basic quantum adversary method [Amb02; HŠ05] considers a quantity of the form

$$s_k := \sum_{x \neq y} w_{x,y} \langle \psi_x^{(k)} | \psi_y^{(k)} \rangle, \quad (3.41)$$

where $k \in [T]$, $x, y \in [n]$, $w_{x,y} \geq 0$, and $|\psi_x^{(k)}\rangle$ is the state of \mathcal{A} after the k^{th} query to the oracle \mathcal{O}_x .

At step $k = 0$, \mathcal{A} has made no queries to the oracle, so $|\psi_x^{(0)}\rangle$ must be the same for all x . Therefore $s_0 = \sum_{x \neq y} w_{x,y}$ as $\langle \psi_x^{(0)} | \psi_y^{(0)} \rangle = 1$.

At step $k = T$, \mathcal{A} must output the index of the best arm with probability $\geq 1 - \delta$. Since the location of the best arm is different for each \mathcal{O}_x , the states $|\psi_x^{(T)}\rangle$ must be distinguishable by a quantum measurement with probability $\geq 1 - \delta$. This means that $|\langle \psi_x^{(T)} | \psi_y^{(T)} \rangle| \leq 2\sqrt{\delta(1 - \delta)}$. Therefore $|s_T| \leq 2\sqrt{\delta(1 - \delta)} \cdot \sum_{x \neq y} w_{x,y}$.

⁴We remark that [Bel15] also treats a more general class of oracles, so it may be possible to prove Theorem 17 using its results. However, we give a self-contained proof using the formulation described above as this approach is straightforward in our case.

Combining the above observations, we have

$$|s_0 - s_T| \geq |s_0| - |s_T| \geq (1 - 2\sqrt{\delta(1-\delta)}) \cdot \sum_{x \neq y} w_{x,y}. \quad (3.42)$$

Hence, if we can upper bound $|s_{k+1} - s_k|$ by B for some constant B , we can deduce that

$$T \geq \frac{1 - 2\sqrt{\delta(1-\delta)}}{B} \cdot \sum_{x \neq y} w_{x,y}, \quad (3.43)$$

giving a lower bound on the query complexity.

With the setup in place, we proceed to prove [Theorem 17](#).

Proof of Theorem 17. We use the adversary method and consider the following n different multi-armed bandit oracles.

In the 1st bandit, we assign bias p_i to arm i . Let $\eta > 0$ be a constant to be determined later. In the x^{th} bandit, $x \in \{2, \dots, n\}$, we assign bias $p'_1 := p_1 + \eta$ to arm x and p_i to arm i for all $i \neq x$. A best-arm identification algorithm must output arm x on assignment x for all $x \in [n]$ with probability $\geq 1 - \delta$.

Following the adversary method, we consider the sum

$$s_k := \sum_{x>1} \frac{1}{\Delta_x'^2} \langle \psi_x^{(k)} | \psi_1^{(k)} \rangle \quad (3.44)$$

for $k \in \{0, \dots, T\}$, where $\Delta_x' := p'_1 - p_x$. Clearly

$$s_0 = \sum_{x>1} \frac{1}{\Delta_x'^2}. \quad (3.45)$$

We also have

$$s_T \leq \sum_{x>1} \frac{1}{\Delta_x'^2} \cdot 2\sqrt{\delta(1-\delta)}. \quad (3.46)$$

Next, we bound the difference $|s_{k+1} - s_k|$. For $i > 1$, let

$$A_i := \begin{pmatrix} \sqrt{1-p_i} & \sqrt{p_i} \\ \sqrt{p_i} & -\sqrt{1-p_i} \end{pmatrix}, \quad (3.47)$$

and let

$$A_1 := \begin{pmatrix} \sqrt{1-p'_1} & \sqrt{p'_1} \\ \sqrt{p'_1} & -\sqrt{1-p'_1} \end{pmatrix}, \quad (3.48)$$

where we recall $p'_1 = p_1 + \eta$ by definition.

Now, we write

$$|\psi_x^{(k)}\rangle = \sum_{z,i,b} \alpha_{x,z,i,b} |z, i, b\rangle \quad \text{and} \quad |\psi_1^{(k)}\rangle = \sum_{z,i,b} \alpha_{1,z,i,b} |z, i, b\rangle. \quad (3.49)$$

Then

$$|\psi_x^{(k+1)}\rangle = \mathcal{O}_x |\psi_x^{(k)}\rangle = \sum_{z,b} \alpha_{x,z,x,b} |z, x\rangle A_1 |b\rangle + \sum_{i \neq x} \sum_{z,b} \alpha_{x,z,i,b} |z, i\rangle A_i |b\rangle \quad (3.50)$$

and similarly

$$|\psi_1^{(k+1)}\rangle = \mathcal{O}_1 |\psi_1^{(k)}\rangle = \sum_{z,b} \alpha_{1,z,x,b} |z, x\rangle A_x |b\rangle + \sum_{i \neq x} \sum_{z,b} \alpha_{1,z,i,b} |z, i\rangle A_i |b\rangle. \quad (3.51)$$

Then

$$|s_{k+1} - s_k| \leq \sum_{x>1} \frac{1}{\Delta_x^2} \left| \langle \psi_x^{(k)} | \mathcal{O}_x^\dagger \mathcal{O}_1 | \psi_1^{(k)} \rangle - \langle \psi_x^{(k)} | \psi_1^{(k)} \rangle \right|. \quad (3.52)$$

Using [Eq. \(3.50\)](#) and [Eq. \(3.51\)](#), and after cancellations, we find that

$$\langle \psi_x^{(k)} | \mathcal{O}_x^\dagger \mathcal{O}_1 | \psi_1^{(k)} \rangle - \langle \psi_x^{(k)} | \psi_1^{(k)} \rangle = \sum_{z,b,b'} \alpha_{x,z,x,b}^* \alpha_{1,z,x,b'} \langle b | (A_1^\dagger A_x - \mathbb{I}) | b' \rangle. \quad (3.53)$$

With

$$\begin{aligned}
& \begin{pmatrix} u_x & v_x \\ -v_x & u_x \end{pmatrix} \\
& := A_1^\dagger A_x - \mathbb{I} \\
& = \begin{pmatrix} \sqrt{(1-p'_1)(1-p_x)} + \sqrt{p'_1 p_x} - 1 & \sqrt{(1-p'_1)p_x} - \sqrt{p'_1(1-p_x)} \\ -\sqrt{(1-p'_1)p_x} + \sqrt{p'_1(1-p_x)} & \sqrt{(1-p'_1)(1-p_x)} + \sqrt{p'_1 p_x} - 1 \end{pmatrix}, \tag{3.54}
\end{aligned}$$

we have

$$|s_{k+1} - s_k| \leq \sum_{x>1} \sum_{z,b} \frac{|u_x|}{\Delta_x'^2} |\alpha_{x,z,x,b}| |\alpha_{1,z,x,b}| + \sum_{x>1} \sum_{z,b \neq b'} \frac{|v_x|}{\Delta_x'^2} |\alpha_{x,z,x,b}| |\alpha_{1,z,x,b'}|. \tag{3.55}$$

Clearly, $|u_x| = 1 - \sqrt{(1-p'_1)(1-p_x)} - \sqrt{p'_1 p_x} \leq 1 - (1-p'_1) - p_x = p'_1 - p_x = \Delta'_x$.

It can also be seen that $|v_x| \leq \Delta'_x / c(p - \eta)$, where $c(x) := 2\sqrt{x(1-x)}$ is a monotone increasing function when $x \in [0, 1/2]$. For completeness, we prove the latter inequality as an auxiliary [Lemma 21](#) immediately after this proof.

We can establish the following bounds using the Cauchy-Schwarz inequality:

$$\begin{aligned}
\sum_{x>1} \sum_{z,b} \frac{|u_x|}{\Delta_x'^2} |\alpha_{x,z,x,b}| |\alpha_{1,z,x,b}| & \leq \sqrt{\sum_{x>1,z,b} \frac{|u_x|^2}{\Delta_x'^4} |\alpha_{x,z,x,b}|^2} \cdot \sqrt{\sum_{x>1,z,b} |\alpha_{1,z,x,b}|^2} \\
& \leq \sqrt{\sum_{x>1} \frac{1}{\Delta_x'^2}} \tag{3.56}
\end{aligned}$$

and

$$\begin{aligned}
\sum_{x>1} \sum_{z,b \neq b'} \frac{|u_x|}{\Delta_x'^2} |\alpha_{x,z,x,b}| |\alpha_{1,z,x,b'}| &= \sum_{b \neq b'} \sum_{x>1,z} \frac{|u_x|}{\Delta_x'^2} |\alpha_{x,z,x,b}| |\alpha_{1,z,x,b'}| \\
&\leq \sum_{b \neq b'} \sqrt{\sum_{x>1,z} \frac{|u_x|^2}{\Delta_x'^4} |\alpha_{x,z,x,b}|^2} \cdot \sqrt{\sum_{x>1,z} |\alpha_{1,z,x,b'}|^2} \quad (3.57) \\
&\leq \frac{2}{c(p-\eta)} \sqrt{\sum_{x>1} \frac{1}{\Delta_x'^2}}.
\end{aligned}$$

Therefore, we find that

$$|s_{k+1} - s_k| \leq \left(1 + \frac{2}{c(p-\eta)}\right) \sqrt{\sum_{x>1} \frac{1}{\Delta_x'^2}}. \quad (3.58)$$

Hence, from Eqs. (3.45), (3.46), and (3.58), we find that

$$T \geq \frac{1 - 2\sqrt{\delta(1-\delta)}}{1 + 2/c(p-\eta)} \sqrt{\sum_{x>1} \frac{1}{\Delta_x'^2}}. \quad (3.59)$$

We then set $\eta = p(p_1 - p_2)/2$. Now, it can be seen that

$$c(p-\eta) = c\left(\left(1 - \frac{p_1 - p_2}{2}\right)p\right) \geq c(p/2) \quad (3.60)$$

because $p \leq 1/2$ and $p_1 - p_2 \leq 1$. Moreover, for $x > 1$,

$$\Delta_x' = p_1 + \eta - p_x = \frac{p}{2}(p_1 - p_2) + (p_1 - p_x) \leq \left(1 + \frac{p}{2}\right)(p_1 - p_x) \leq \frac{5}{4}\Delta_x \quad (3.61)$$

because $p_x \leq p_2$ and $p \leq 1/2$. Therefore, we find that

$$T \geq \frac{4}{5} \cdot \frac{1 - 2\sqrt{\delta(1-\delta)}}{1 + 2/c(p/2)} \sqrt{\sum_{x>1} \frac{1}{\Delta_x^2}}, \quad (3.62)$$

and hence $T = \Omega\left(\sqrt{\sum_{i=2}^n \frac{1}{\Delta_i^2}}\right)$. □

The following lemma is used in the above proof.

Lemma 21. *Suppose that $p_1, p_2 \in [p, 1 - p]$ where $0 < p \leq 1/2$. Then*

$$|\sqrt{(1 - p_1)p_2} - \sqrt{(1 - p_2)p_1}| \leq \frac{|p_1 - p_2|}{2\sqrt{p(1 - p)}}, \quad (3.63)$$

and the term in the denominator is optimal.

Proof. Note that

$$\sqrt{(1 - p_1)p_2} - \sqrt{(1 - p_2)p_1} = \frac{(1 - p_1)p_2 - (1 - p_2)p_1}{\sqrt{(1 - p_1)p_2} + \sqrt{(1 - p_2)p_1}} \quad (3.64)$$

$$= \frac{-(p_1 - p_2)}{\sqrt{(1 - p_1)p_2} + \sqrt{(1 - p_2)p_1}}. \quad (3.65)$$

Therefore, it suffices to prove

$$\sqrt{(1 - p_1)p_2} + \sqrt{(1 - p_2)p_1} \geq 2\sqrt{p(1 - p)}. \quad (3.66)$$

Since $p_1, p_2 \in [p, 1 - p]$, we have

$$(p_1 - p)(p_1 - (1 - p)) \leq 0 \quad (3.67)$$

$$(p_2 - p)(p_2 - (1 - p)) \leq 0 \quad (3.68)$$

$$|2p_1 - 1| \leq 1 - 2p \quad (3.69)$$

$$|2p_2 - 1| \leq 1 - 2p. \quad (3.70)$$

Eqs. (3.67) and (3.68) are equivalent to

$$p_1 - p_1^2 \geq p(1 - p), \quad p_2 - p_2^2 \geq p(1 - p). \quad (3.71)$$

Eqs. (3.69) and (3.70) imply

$$4p_1p_2 - 2p_1 - 2p_2 + 1 = (2p_1 - 1)(2p_2 - 1) \leq (2p - 1)^2 = 4p^2 - 4p + 1, \quad (3.72)$$

which gives

$$p_1 + p_2 - 2p_1p_2 \geq 2p - 2p^2. \quad (3.73)$$

Now, we have

$$\left(\sqrt{(1-p_1)p_2} + \sqrt{(1-p_2)p_1} \right)^2 \quad (3.74)$$

$$= (1-p_1)p_2 + (1-p_2)p_1 + 2\sqrt{(1-p_1)p_2(1-p_2)p_1} \quad (3.75)$$

$$= p_1 + p_2 - 2p_1p_2 + 2\sqrt{p_1(1-p_1)}\sqrt{p_2(1-p_2)} \quad (3.76)$$

$$\geq 2p - 2p^2 + 2p(1-p) = (2\sqrt{p(1-p)})^2, \quad (3.77)$$

where the inequality comes from Eq. (3.71) and Eq. (3.73). Therefore, we have established Eq. (3.66). Note that this is optimal as taking $p_1 = p_2 = p$ makes the two sides in Eq. (3.66) equal. \square

3.5 Discussion and open problems

The MAB problem is the most basic problem in reinforcement learning (RL). In RL, there is an agent that interacts with an environment. The main problem in RL is to compute an optimal policy for an agent to follow so that it maximizes its total reward. The MAB problem corresponds to the simplest case of the RL problem where the environment is in a single fixed *state* that returns stochastic rewards according to some fixed distribution. In general, there are many states in an RL problem. Each state corresponds to a different multi-armed bandit and the agent's actions at each

state yield both a reward from the bandit and also moves it to a new state. My subsequent work [Wan+21] shows how the results in this chapter can be extended to compute a near-optimal policy for the agent. Like in this chapter, that work also gives a realistic way of instantiating the oracle used, see Appendix A of the arXiv version of [Wan+21].

We conclude by discussing some remaining open problems related to this chapter.

1. Do there exist biases $p_1, \dots, p_n \in [0, 1]$, that do not all lie within an interval $[p, 1-p]$ for some constant p , such that solving the best-arm identification problem with these p_i s gives a super-quadratic quantum speedup? Note that our quantum lower bound does not apply in this case. Alternatively, are there certain *summary statistics* of the p_i s that can be computed with a super-quadratic quantum speedup? More generally, can we generalize the search problem in other ways to obtain an exponential quantum speedup? Alternatively, can we generalize a problem known to admit an exponential quantum speedup (such as computing Simon’s function) so that its “usefulness” goes up, perhaps at the expense of a slight decrease in speedup? As I mentioned in the introduction to this chapter, these are the questions that originally motivated me to study the MAB problem.
2. Can we achieve a quantum speedup for Monte Carlo Tree Search (MCTS)? MCTS is an influential classical algorithm that enabled a computer program to beat the human Go champion for the first time [Sil+16]. In the introduction to this chapter, we described how an MAB algorithm can be used to implement a simple Monte Carlo strategy for playing a two-player game such as Go. Using that strategy, a player estimates how good a move is by the empirical probability that it leads to a win following *uniformly random* subsequent play by both players. However, the estimate obtained can be very inaccurate because, in reality, each player would play to its advantage. To avoid this, a player may instead use a *minimax* strategy where they decide a move is good if it leads to their win following *perfect* subsequent play

by both players. Unfortunately, implementing the minimax strategy requires an enormous computation. MCTS uses a strategy that interpolates between simple Monte Carlo and minimax. This chapter shows that the Monte Carlo strategy admits a quantum speedup. [AK17] shows that the minimax strategy also admits a quantum speedup. Therefore, we expect MCTS to admit a quantum speedup.

3. Can we meaningfully define the *regret* of a quantum algorithm for multi-armed bandits? Classically, the regret of an algorithm is the difference between the total reward the algorithm obtains and the maximum possible total reward that it could have obtained. Defining regret would allow us to study the exploration-exploitation tradeoff of MABs in the quantum setting. Since a quantum query to the oracle does not return a reward in the usual sense, it is not immediately obvious how regret should be defined. Recently, two works [Wan+22; LZ22] have suggested similar ways of defining regret in the quantum setting. These works assume that, at each step j where the quantum algorithm queries the MAB oracle, it must separately output the index of an arm $i_j \in [n]$ that it “would have liked to exploit”. The regret at step T can then be defined with respect to the classical sequence $(i_1, i_2, \dots, i_T) \in [n]^T$. Under this definition, [Wan+22; LZ22] both show that there are quantum algorithms with regret that scales logarithmically in T , which is much better than what is possible classically. It remains open to find a real-world scenario where this “counterfactual” definition of regret is relevant.

Conclusion

In this thesis, we have studied the nature of quantum speedups from three different perspectives. Each perspective gives the quantum algorithm designer concrete questions to ask when assessing whether a problem could yield any quantum speedup:

1. How symmetrical is the problem? If it is too symmetrical then we cannot hope for a massive quantum speedup. If it is a graph problem, is the input specified in the adjacency list or matrix model?
2. Can the problem be solved classically by divide and conquer? If so, is the relationship between the problem and its subproblems susceptible to speedup by quantum divide and conquer?
3. Is the problem a generalization of search (or of any other problem admitting quantum speedup)? Can we think of the problem as identifying the most heavily marked item from a list of partially marked items?

Finding a useful problem with a massive quantum speedup is the most pressing challenge facing quantum computing researchers. Surprisingly, despite the rapidly growing interest in the field, it is arguable whether the challenge has been answered affirmatively⁵. Technologically, it would be desirable if the challenge were answered affirmatively. Scientifically, it would be equally interesting if it were answered negatively. Indeed, the results in [Chapter 1](#) of this thesis point towards a tradeoff between

⁵Shor's factoring algorithm offers a massive quantum speedup but its main application appears to be breaking cryptosystems, which is the opposite of useful. Even the folklore massive speedup for simulating systems that are themselves quantum mechanical has recently been challenged [[Lee+22](#)].

a problem’s lack of structure and its quantum speedup. But, since a problem’s lack of structure can be seen as proxy for its wide-spread applicability and therefore usefulness, our results lean towards a negative answer. Nevertheless, in [Chapters 2 and 3](#) of this thesis, we have explored two ways of obtaining new quantum speedups. [Chapter 2](#) gives evidence that many problems that can be solved classically using divide and conquer may admit non-trivial quantum speedups. [Chapter 3](#) gives an example of how we can generalise an existing quantum speedup to give a modest but useful quantum speedup of real-world relevance. We hope that these and other approaches can help us find a massive quantum speedup for a useful problem in the future.

In this thesis, we have measured quantum speedup by how the *worst-case* quantum *query* complexity compares to the worst-case classical query complexity when queries are made to an oracle encoding a *classical* string or probability distribution. We must point out that this is not the only way to measure quantum speedup.

First, we can study quantum speedups in query complexity in the *average case*. Recently, Yamakawa and Zhandry [[YZ22](#)] constructed a (unfortunately useless) relation problem that a quantum algorithm can solve (in a verifiable way) using exponentially fewer queries than any classical algorithm on *average* over all inputs. One interesting aspect of their problem is that its input is not promised to have any structure unlike most previously known problems that admit a massive quantum speedup. (For example, the input to Simon’s function is promised to have a period.) The study of average-case quantum speedups is also tied to an intriguing conjecture of Aaronson and Ambainis [[AA14](#)] which posits that it is impossible to achieve a massive quantum speedup in the average case for solving a decision problem. Second, we focused on query complexity because it allows us to rigorously prove quantum speedups in a well-defined but idealised model. As quantum computing hardware evolves, I expect that *heuristic* quantum algorithms will proliferate and that some will yield heuristic but real quantum speedups. For example, [[LHLW23](#)] gives evidence that a physically

motivated quantum algorithm for gradient descent already demonstrates this kind of quantum speedup when implemented on currently available hardware. Given that some of the most impactful *classical* algorithms are heuristic (for example, those in deep learning), the same may hold for quantum algorithms. Third, we can study how the nature of quantum speedup changes when the queries are made to *quantum* objects, such as quantum states or channels. In this setting, we compare a classical algorithm that must measure immediately after each query against a quantum algorithm that is not subject to this restriction. Recent work has shown that a quantum algorithm can make exponentially fewer queries to the quantum object than any classical algorithm to compute certain properties of the quantum object [[HKP21](#); [ACQ22](#); [CCHL22](#)].

Appendix

1 Adversary composition lemmas

In this appendix, we prove two adversary compositions lemmas: [Lemma 8](#) in [Appendix 1.1](#) and [Lemma 9](#) in [Appendix 1.2](#).

Before proving these lemmas, we first introduce a quantity closely related to Adv , the filtered γ_2 norm, and state some facts about Adv and γ_2 that are used in our proofs.

For finite sets D_1 and D_2 , we write $A \in \mathbb{C}^{D_1 \times D_2}$ to mean that A is a complex $|D_1| \times |D_2|$ matrix with rows indexed by D_1 and columns indexed by D_2 .

Definition 14 (Filtered γ_2 norm). *Let D_1 and D_2 be finite sets and $n \in \mathbb{N}$. Let $A \in \mathbb{C}^{D_1 \times D_2}$ and $Z = \{Z_i \in \mathbb{C}^{D_1 \times D_2} \mid i \in [n]\}$. Define $\gamma_2(A|Z)$ by*

$$\gamma_2(A|Z) := \min_{\substack{d \in \mathbb{N}, \\ |u_{xj}\rangle, |v_{yj}\rangle \in \mathbb{C}^d}} \max \left\{ \max_{x \in D_1} \sum_{j=1}^n \| |u_{xj}\rangle \|^2, \max_{y \in D_2} \sum_{j=1}^n \| |v_{yj}\rangle \|^2 \right\} \quad (78)$$

$$\text{subject to: } \forall x \in D_1, y \in D_2, \quad \sum_{j=1}^n (Z_j)_{xy} \langle u_{xj} | v_{yj} \rangle = A_{xy}. \quad (79)$$

We can view $\text{Adv}(\cdot)$ as a special case of the filtered γ_2 norm. Let $f: D \rightarrow E$ be a function, where $D \subseteq \mathcal{D}^n$ and E are finite sets. Let F be the Gram matrix of f . Let $\Delta := \{\Delta_1, \dots, \Delta_n\}$ be a set of $|D| \times |D|$ matrices defined such that $(\Delta_j)_{x,y \in D} = 1 - \delta_{x_j, y_j}$ for all $j \in [n]$. Then it can be shown that $\text{Adv}(f) = \gamma_2(J - F|\Delta')$, where

$\Delta' := \{\Delta_j \circ (J - F) \mid j \in [n]\}$ and J is the all-ones matrix of the same size as F .

We now state five facts about Adv and γ_2 that are used in our proofs. The symbols Δ and J refer to the set of matrices defined above (appropriately sized) and the all-ones matrix (appropriately sized), respectively.

Fact 1 ([Rei09a, Theorem 6.2]). *Let $f: D \rightarrow \{0, 1\}$ be a Boolean function, where $D \subseteq \mathcal{D}^n$ is a finite set. Then*

$$\text{Adv}(f) = \min_{\substack{d \in \mathbb{N}, \\ \{u_{xj}\} \in \mathbb{C}^d}} \sqrt{A_0 \cdot A_1} \quad (80)$$

$$\text{subject to: } \forall b \in \{0, 1\}, \quad A_b = \max_{x \in f^{-1}(b)} \sum_{j=1}^n \| |u_{xj}\rangle \|^2. \quad (81)$$

$$\forall x, y \in D \text{ with } f(x) \neq f(y), \quad \sum_{j \in [n]: x_j \neq y_j} \langle u_{xj} | u_{yj} \rangle = 1. \quad (82)$$

Fact 2 ([Lee+11, Theorem 3.4]). *Let $f: D \rightarrow E$ be a function, where $D \subseteq \mathcal{D}^n$ and E are finite sets. Let F be the Gram matrix of f . Then*

$$\text{Adv}(f) \leq \gamma_2(J - F | \Delta) \leq 2(1 - 1/|E|) \text{Adv}(f). \quad (83)$$

Fact 3 (Triangle inequality [Lee+11, Lemma A.2, item 4]). *Let D_1, D_2 be finite sets and $n \in \mathbb{N}$. Let $A, B \in \mathbb{C}^{D_1 \times D_2}$ and $Y_j \in \mathbb{C}^{D_1 \times D_2}$ for all $j \in [n]$. Let $Z := \{Z_j \mid j \in [n]\}$. Then*

$$\gamma_2(A + B | Z) \leq \gamma_2(A | Z) + \gamma_2(B | Z). \quad (84)$$

Fact 4 ([Lee+11, Lemma A.2, item 10]). *Let D_1, D_2 be finite sets and $n \in \mathbb{N}$. Let $A, B \in \mathbb{C}^{D_1 \times D_2}$ and $Y_j, Z_j \in \mathbb{C}^{D_1 \times D_2}$ for all $j \in [n]$. Let $Y := \{Y_j \mid j \in [n]\}$ and $Z := \{Z_j \mid j \in [n]\}$. Then*

$$\gamma_2(A \circ B \mid \{Z_j \circ B \mid j \in [n]\}) \leq \gamma_2(A \mid Z) \leq \gamma_2(A \mid \{Z_j \circ B \mid j \in [n]\}) \cdot \gamma_2(B). \quad (85)$$

Fact 5 (Direct-sum property [Lee+11, Lemma A.2, item 12]). *Let C_1, C_2, D_1, D_2 be finite sets and $n \in \mathbb{N}$. Let $A \in \mathbb{C}^{C_1 \times C_2}$ and $Y_j \in \mathbb{C}^{C_1 \times C_2}$ for all $j \in [n]$. Let $B \in \mathbb{C}^{D_1 \times D_2}$ and $Z_j \in \mathbb{C}^{D_1 \times D_2}$ for all $j \in [n]$. Let $Y := \{Y_j \mid j \in [n]\}$ and $Z := \{Z_j \mid j \in [n]\}$. Then*

$$\gamma_2(A \oplus B \mid \{Y_j \oplus Z_j \mid j \in [n]\}) = \max\{\gamma_2(A \mid Y), \gamma_2(B \mid Z)\}. \quad (86)$$

1.1 Proof of Lemma 8

Lemma 8 (Adversary composition for OR and AND). *Let $a, b \in \mathbb{N}$ and let Σ be a finite set. Let $f_1: \Sigma^a \rightarrow \{0, 1\}$ and $f_2: \Sigma^b \rightarrow \{0, 1\}$. Let $g^\wedge, g^\vee: \Sigma^a \times \Sigma^b \rightarrow \{0, 1\}$ be such that $g^\wedge(x, y) = f_1(x) \wedge f_2(y)$ and $g^\vee(x, y) = f_1(x) \vee f_2(y)$. Then $\text{Adv}(g^\wedge)^2 = \text{Adv}(g^\vee)^2 \leq \text{Adv}(f_1)^2 + \text{Adv}(f_2)^2$.*

Moreover, suppose $a = b$ and $h: \Sigma^a \rightarrow \{0, 1\}$ satisfies $h(x) = f_1(x) \wedge f_2(x)$ for all $x \in \Sigma^n$. Let $f'_2: f_1^{-1}(1) \rightarrow \{0, 1\}$ be such that $f'_2(x) = f_2(x)$ for all $x \in f_1^{-1}(1)$. Then $\text{Adv}(h)^2 \leq \text{Adv}(f_1)^2 + \text{Adv}(f'_2)^2$.

Proof. It suffices to prove the first part of the lemma only for g^\vee since $g^\wedge(x, y) = f_1(x) \wedge f_2(y) = \neg((\neg f_1(x)) \vee (\neg f_2(x)))$ and $\text{Adv}(f) = \text{Adv}(\neg f)$ for any Boolean function $f: \Sigma^n \rightarrow \{0, 1\}$, where \neg denotes negation. For the rest of this proof, we write $g := g^\vee$, $a_1 := a$, and $a_2 := b$ for notational convenience.

For $i \in \{1, 2\}$, let $\{|u_{xj}^i\rangle \in \mathbb{C}^{d_i} \mid x \in \Sigma^{a_i}, j \in [a_i]\}$ be a minimum solution to the minimization problem in **Fact 1** that specifies $\text{Adv}(f_i)$. Then

$$\text{Adv}(f_i)^2 = A_0^i \cdot A_1^i, \quad \text{where } A_b^i := \max_{x \in f_i^{-1}(b)} \sum_{j=1}^{a_i} \||u_{xj}^i\rangle\|^2 \text{ for all } b \in \{0, 1\}, \quad (87)$$

$$\forall x, y \in \Sigma^{a_i} \text{ with } f_i(x) \neq f_i(y), \quad \sum_{j \in [a_i]: x_j \neq y_j} \langle u_{xj}^i | u_{yj}^i \rangle = 1. \quad (88)$$

By mapping $|u_{xj}^i\rangle \mapsto \sqrt{A_1^i} |u_{xj}^i\rangle$ for $x \in f_i^{-1}(0)$ and $|u_{xj}^i\rangle \mapsto |u_{xj}^i\rangle / \sqrt{A_1^i}$ for $x \in$

$f_i^{-1}(1)$, we can assume without loss of generality that

$$A_0^i = \text{Adv}(f_i)^2 \quad \text{and} \quad A_1^i = 1 \quad \text{for all } i \in \{1, 2\}. \quad (89)$$

Now, for $(x, x') \in \Sigma^{a_1} \times \Sigma^{a_2}$ and $k \in [a_1 + a_2]$, let $k' := k - a_1$ and define $|\lambda_{(x, x')k}\rangle \in \mathbb{C}^{d_1} \oplus \mathbb{C}^{d_2}$ according to the following table. For example, if $(f_1(x), f_2(x')) = (1, 0)$ and $k \leq a_1$, then $|\lambda_{(x, x')k}\rangle := |u_{xk}^1\rangle \oplus 0$.

$(f_1(x), f_2(x'))$	$k \leq a_1$	$k > a_1$
(0, 0)	$ u_{xk}^1\rangle \oplus 0$	$0 \oplus u_{x'k'}^2\rangle$
(0, 1)	$0 \oplus 0$	$0 \oplus u_{x'k'}^2\rangle$
(1, 0)	$ u_{xk}^1\rangle \oplus 0$	$0 \oplus 0$
(1, 1)	$ u_{xk}^1\rangle \oplus 0$	$0 \oplus 0$

(90)

Then it can be verified by inspection that

$$\begin{aligned} \forall (x, x'), (y, y') \in \Sigma^{a_1} \times \Sigma^{a_2} \text{ with } g(x, x') \neq g(y, y'), \\ \sum_{\substack{k \in [a_1 + a_2]: \\ (x, x')_k \neq (y, y')_k}} \langle \lambda_{(x, x')k} | \lambda_{(y, y')k} \rangle = 1. \end{aligned} \quad (91)$$

Moreover, from the definitions of g and $|\lambda_{(x, x')k}\rangle$, and [Equation \(89\)](#), we find

$$\begin{aligned} A_0 &:= \max_{(x, x') \in g^{-1}(0)} \sum_{k=1}^{a_1 + a_2} \left\| |\lambda_{(x, x')k}\rangle \right\|^2 = \text{Adv}(f_1)^2 + \text{Adv}(f_2)^2, \\ A_1 &:= \max_{(x, x') \in g^{-1}(1)} \sum_{k=1}^{a_1 + a_2} \left\| |\lambda_{(x, x')k}\rangle \right\|^2 = 1. \end{aligned} \quad (92)$$

But Equation (91) means that

$$\{|\lambda_{(x,x')k} \in \mathbb{C}^{d_1} \oplus \mathbb{C}^{d_2} \mid (x, x') \in \Sigma^{a_1} \times \Sigma^{a_2}, k \in [a_1 + a_2]\}$$
 (93)

is a feasible solution to the minimization problem in Fact 1 that specifies $\text{Adv}(g)$.

Therefore

$$\text{Adv}(g)^2 \leq A_0 \cdot A_1 = \text{Adv}(f_1)^2 + \text{Adv}(f_2)^2,$$
 (94)

which completes the proof of the first part of the lemma.

The proof of the “moreover” part is similar. Let $D := f_1^{-1}(1) \subseteq \Sigma^a$. Let $|u_{xj}^1\rangle$, A_0^1 , and A_1^1 be defined as above. Let $\{|u_{xj}^{2'}\rangle \in \mathbb{C}^{d_2'} \mid x \in D, j \in [a]\}$ be a minimum solution to the minimization problem in Fact 1 that specifies $\text{Adv}(f_2')$. Then

$$\text{Adv}(f_2')^2 = A_0^{2'} \cdot A_1^{2'}, \quad \text{where } A_b^{2'} := \max_{x \in f_2'^{-1}(b)} \sum_{j=1}^a \||u_{xj}^{2'}\rangle\|^2 \text{ for all } b \in \{0, 1\},$$
 (95)

$$\forall x, y \in D \text{ with } f_2'(x) \neq f_2'(y), \quad \sum_{j \in [a]: x_j \neq y_j} \langle u_{xj}^{2'} | u_{yj}^{2'} \rangle = 1.$$
 (96)

By a similar argument as above, we can assume without loss of generality that

$$A_0^1 = 1, \quad A_1^1 = \text{Adv}(f_1)^2, \quad A_0^{2'} = 1, \quad \text{and} \quad A_1^{2'} = \text{Adv}(f_2')^2.$$
 (97)

Now, for $x \in \Sigma^a$ and $k \in [a]$, define $|\mu_{xk}\rangle \in \mathbb{C}^{d_1} \oplus \mathbb{C}^{d_2'}$ according to the following

table.

$(f_1(x), f_2(x))$	$k \in [a]$
$(0, 0)$	$ u_{xk}^1\rangle \oplus 0$
$(0, 1)$	$ u_{xk}^1\rangle \oplus 0$
$(1, 0)$	$0 \oplus u_{xk}^{2'}\rangle$
$(1, 1)$	$ u_{xk}^1\rangle \oplus u_{xk}^{2'}\rangle$

(98)

Note that $|\mu_{xk}\rangle$ is well-defined since $|u_{xk}^{2'}\rangle$ only appear in rows with $f_1(x) = 1$.

Then it can be verified by inspection that

$$\forall x, y \in \Sigma^a \text{ with } h(x) \neq h(y), \quad \sum_{k \in [a]: x_k \neq y_k} \langle \mu_{xk} | \mu_{yk} \rangle = 1. \quad (99)$$

Moreover, from the definitions of h and $|\mu_{xk}\rangle$, and [Equation \(97\)](#), we find

$$\begin{aligned} B_0 &:= \max_{x \in h^{-1}(0)} \sum_{k=1}^a \|\mu_{xk}\|^2 \leq 1, \\ B_1 &:= \max_{x \in h^{-1}(1)} \sum_{k=1}^a \|\mu_{xk}\|^2 \leq \text{Adv}(f_1)^2 + \text{Adv}(f_2')^2. \end{aligned} \quad (100)$$

But [Equation \(99\)](#) means that

$$\{|\mu_{xk}\rangle \in \mathbb{C}^{d_1} \oplus \mathbb{C}^{d_2'} \mid x \in \Sigma^a, k \in [a]\} \quad (101)$$

is a feasible solution to the minimization problem in [Fact 1](#) that specifies $\text{Adv}(h)$.

Therefore

$$\text{Adv}(h)^2 \leq B_0 \cdot B_1 \leq \text{Adv}(f_1)^2 + \text{Adv}(f_2')^2, \quad (102)$$

which completes the proof of the “moreover” part of the lemma. □

1.2 Proof of Lemma 9

Lemma 9 (Adversary composition for SWITCH-CASE). *Let $a \in \mathbb{N}$ and let Σ, Λ be finite sets. Let $f: \Sigma^a \rightarrow \Lambda$. For $s \in \Lambda$, let $g_s: \Sigma^a \rightarrow \{0, 1\}$. Define $h: \Sigma^a \rightarrow \{0, 1\}$ by $h(x) = g_{f(x)}(x)$. Then*

$$\text{Adv}(h) \leq O(\text{Adv}(f)) + \max_{s \in \Lambda} \text{Adv}(g_s). \quad (2.9)$$

Proof. Define $h': \Sigma^a \rightarrow \Lambda \times \{0, 1\}$ by $h'(x) = (f(x), h(x)) = (f(x), g_{f(x)}(x))$. Let H' , F , and $\{G_s \mid s \in \Lambda\}$ be the Gram matrices for h' , f , and $\{g_s \mid s \in \Lambda\}$, respectively, each having size $|\Sigma|^a \times |\Sigma|^a$. Define $\Delta := \{\Delta_1, \dots, \Delta_a\}$, where each Δ_j is a $|\Sigma|^a \times |\Sigma|^a$ matrix with entries $(\Delta_j)_{xy} := 1 - \delta_{x_j, y_j}$ for all $x, y \in \Sigma^a$. For $m \in \mathbb{N}$, define J_m to be the all-ones matrix of size $m \times m$, and write $J := J_{|\Sigma|^a}$. We also write $\gamma_2(f) := \gamma_2(J - F | \Delta)$.

We have

$$\text{Adv}(h) \leq \text{Adv}(h') \quad (\text{Definition 7}) \quad (103)$$

$$\leq \gamma_2(J - H' | \Delta) \quad (\text{Fact 2}) \quad (104)$$

$$\leq \gamma_2(J - F | \Delta) + \gamma_2(F - H' | \Delta) \quad (\text{Fact 3}) \quad (105)$$

$$\leq O(\text{Adv}(f)) + \gamma_2(F - H' | \Delta) \quad (\text{Fact 2}). \quad (106)$$

We proceed to bound $\gamma_2(F - H' | \Delta)$. For $s \in \Lambda$, let $b_s := |f^{-1}(s)|$, where $f^{-1}(s) := \{x \in \Sigma^a \mid f(x) = s\}$. Note $\sum_{s \in \Lambda} b_s = |\Sigma|^a$. For $s \in \Lambda$, define a matrix \tilde{G}_s of size $b_s \times b_s$ by $(\tilde{G}_s)_{xy} = (G_s)_{xy}$ for all $x, y \in f^{-1}(s)$. For $s \in \Lambda$ and $j \in [a]$, define a matrix $\Delta_j^{(b_s)}$ of size $b_s \times b_s$ by $(\Delta_j^{(b_s)})_{xy} = 1 - \delta_{x_j, y_j}$ for all $x, y \in f^{-1}(s)$. Define $\Delta^{(b_s)} := \{\Delta_1^{(b_s)}, \dots, \Delta_a^{(b_s)}\}$.

By inspection, we have

$$H' = \bigoplus_{s \in \Lambda} \tilde{G}_s, \quad F = \bigoplus_{s \in \Lambda} J_{b_s}, \quad \text{and} \quad \Delta_j \circ F = \bigoplus_{s \in \Lambda} \Delta_j^{(b_s)}. \quad (107)$$

Therefore, we have

$$\gamma_2(F - H' \mid \Delta) \tag{108}$$

$$\leq \gamma_2(F - H' \mid \{\Delta_j \circ F \mid j \in [a]\}) \cdot \gamma_2(F) \quad (\text{Fact 4}) \tag{109}$$

$$\leq \gamma_2(F - H' \mid \{\Delta_j \circ F \mid j \in [a]\}) \quad (\gamma_2(F) \leq 1) \tag{110}$$

$$= \max_{s \in \Lambda} \gamma_2(J_{b_s} - \tilde{G}_s \mid \Delta^{(b_s)}) \quad (\text{Equation (107) and Fact 5}) \tag{111}$$

$$\leq \max_{s \in \Lambda} \gamma_2(J - G_s \mid \Delta) = \max_{s \in \Lambda} \text{Adv}(g_s). \tag{112}$$

The lemma follows. □

2 Randomized search

In this appendix, we equate the finite totally ordered set Σ with $[E]$, where $E := |\Sigma|$, for convenience of presentation. It is clear that our randomized search can be generalized to any finite totally ordered set Σ .

We claim that $\text{min-last}_{j,n}$ and $\text{max-first}_{j,n}$ can be computed by a randomized search that uses $O(\log(n))$ computations of $\text{LIS}_{j,n}$ and calls to Grover search. The claim follows from [Lemma 22](#), which describes the randomized search, with suitable instantiations of the oracles \mathcal{R} and \mathcal{O} in its statement.

Lemma 22 (Randomized search). *Let S be a multi-set of n integers and let $s \in S$. Let \mathcal{O} be an oracle that decides for any input $r \in \mathbb{Z}$ whether $s < r$, $s = r$, or $s > r$. Let \mathcal{R} be an oracle that, for any inputs $r_1, r_2 \in \mathbb{R} \cup \{+\infty, -\infty\}$, selects a uniformly random element u from the multi-set $\{a \in S \mid r_1 < a < r_2\}$. Then there exists a constant $c > 0$ and a randomized classical algorithm that calls each of \mathcal{O} and \mathcal{R} at most $c \cdot \log(n/\delta)$ times to output s with probability at least $1 - \delta$.*

Proof. We show that the following randomized classical algorithm satisfies the requirements of the lemma. The commands in square brackets are only used in the

analysis.

Set $r_1 = -\infty$ and $r_2 = \infty$. [Set $t = 0$ and $S_0 := S$.]

Repeat the following until success:

Use \mathcal{R} with input r_1 and r_2 to select a uniformly random element u from $\{a \in S \mid r_1 < a < r_2\}$.

Use \mathcal{O} to decide whether $s < u$, $s = u$, or $s > u$.

1. If $s = u$, output s .
2. If $s < u$, set $r_2 = u$. [Set $S_t := \{a \in S_{t-1} \mid a < u\}$.]
3. If $s > u$, set $r_1 = u$. [Set $S_t := \{a \in S_{t-1} \mid a > u\}$.]

[Set $t = t + 1$.]

For $t \geq 0$, write $X_t := |S_t|$, which is a random variable. Write $x_t := \mathbb{E}[X_t]$. Initially, $S_0 = S$ and $x_0 = n$.

Suppose that the rank of s in S_t is p (if S_t contains more than one s , then p can be taken as the rank of any). Then we have

$$\mathbb{E}[X_{t+1} | X_t] \leq \frac{1}{X_t} \left(\sum_{i=1}^{p-1} (X_t - i) + \sum_{i=p}^{X_t-1} i \right) \leq \frac{3}{4} X_t, \quad (113)$$

where the first inequality is due to the possibility of S containing duplicate elements (otherwise it would be equality) and the second inequality follows by considering the maximization of a quadratic polynomial in p . Taking the expectation of [Equation \(113\)](#) gives $x_{t+1} \leq \frac{3}{4} x_t$. Together with $x_0 = n$, this implies $x_t \leq (3/4)^t n$.

Therefore, by Markov's inequality, we have $\text{prob}(X_t > 1) \leq (3/4)^t n \leq \delta$ for $t \geq 10 \log(n/\delta)$. Therefore, since $s \in S_t$ for all $t \geq 0$ before the algorithm outputs s , if we force the algorithm to abort after $10 \log(n/\delta)$ steps, we will obtain s at some point with probability at least $1 - \delta$. □

We now instantiate the \mathcal{R} and \mathcal{O} in [Lemma 22](#) as follows.

1. *Instantiating the oracle \mathcal{R} .* This is Grover search over S for an element $x \in S$ with $r_1 < x < r_2$. Note that the Grover search does indeed return a uniformly random element from $\{x \in S \mid r_1 < x < r_2\}$ as required to apply [Lemma 22](#).
2. *Instantiating the oracle \mathcal{O} .* Let $x \in [E]^n$ and $u \in [E]$. Recall that $\text{LIS}_{j,n}(x)$ decides whether x contains a j -IS*. [Table 1](#) and [Table 2](#) show how an algorithm for computing $\text{LIS}_{j,n}$ can be used to instantiate \mathcal{O} for computing $\text{min-last}_{j,n}$ and $\text{max-first}_{j,n}$, respectively. In the table headings, “remove i ” is shorthand for “turn into $*$ when x is queried at position i ”. (Note that being able to do this removal operation is the reason why we considered k -IS* instead of k -IS, and why we say that k -IS* is more susceptible to recursion than k -IS.)

Remove all $i \in [n]$ such that $x[i] > u$ and compute $\text{LIS}_{j,n}(x)$	Remove all $i \in [n]$ such that $x[i] \geq u$ and compute $\text{LIS}_{j,n}(x)$	Conclusion
0	0	$\text{min-last}_{j,n}(x) > u$
0	1	Impossible
1	0	$\text{min-last}_{j,n}(x) = u$
1	1	$\text{min-last}_{j,n}(x) < u$

Table 1: Instantiating \mathcal{O} for computing $\text{min-last}_{j,n}$ using $\text{LIS}_{j,n}$.

Remove all $i \in [n]$ such that $x[i] < u$ and compute $\text{LIS}_{j,n}(x)$	Remove all $i \in [n]$ such that $x[i] \leq u$ and compute $\text{LIS}_{j,n}(x)$	Conclusion
0	0	$\text{max-first}_{j,n}(x) < u$
0	1	Impossible
1	0	$\text{max-first}_{j,n}(x) = u$
1	1	$\text{max-first}_{j,n}(x) > u$

Table 2: Instantiating \mathcal{O} for computing $\text{max-first}_{j,n}$ using $\text{LIS}_{j,n}$.

Remarks.

1. Our randomized search generalizes the technique of quantum minimum finding [[DH96](#)] because in quantum minimum finding, s is additionally promised to be the minimum value in S in [Lemma 22](#).

2. [Lemma 22](#) assumes that the oracles \mathcal{O} and \mathcal{R} are noiseless. In fact, the way we instantiate them means they can be erroneous with bounded probability. Naively, one might expect this to introduce an additional $\log \log(n)$ factor in the complexity of randomized search due to the need for error reduction since \mathcal{O} and \mathcal{R} are called $O(\log(n))$ times. However, techniques in [\[FRPU94\]](#) allow us to remove this $\log \log(n)$ factor.

Bibliography

- [AA14] Scott Aaronson and Andris Ambainis. “The Need for Structure in Quantum Speedups”. In: *Theory of Computing* 10.6 (2014), pp. 133–166. DOI: [10.4086/toc.2014.v010a006](https://doi.org/10.4086/toc.2014.v010a006). [arXiv:0911.0996](https://arxiv.org/abs/0911.0996).
- [ABK16] Scott Aaronson, Shalev Ben-David, and Robin Kothari. “Separations in Query Complexity Using Cheat Sheets”. In: *Proceedings of the 48th ACM Symposium on the Theory of Computing (STOC)*. 2016, pp. 863–876. DOI: [10.1145/2897518.2897644](https://doi.org/10.1145/2897518.2897644). [arXiv:1511.01937](https://arxiv.org/abs/1511.01937).
- [AGS19] Scott Aaronson, Daniel Grier, and Luke Schaeffer. “A Quantum Query Complexity Trichotomy for Regular Languages”. In: *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2019, pp. 942–965. DOI: [10.1109/FOCS.2019.00061](https://doi.org/10.1109/FOCS.2019.00061). [arXiv:1812.04219](https://arxiv.org/abs/1812.04219).
- [AS04] Scott Aaronson and Yaoyun Shi. “Quantum Lower Bounds for the Collision and the Element Distinctness Problems”. In: *Journal of the ACM* 51.4 (2004), pp. 595–605. DOI: [10.1145/1008731.1008735](https://doi.org/10.1145/1008731.1008735).
- [Abb19] Amir Abboud. “Fine-Grained Reductions and Quantum Speedups for Dynamic Programming”. In: *46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 132. 2019, 8:1–8:13. DOI: [10.4230/LIPIcs.ICALP.2019.8](https://doi.org/10.4230/LIPIcs.ICALP.2019.8).
- [ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. “Tight Hardness Results for LCS and Other Sequence Similarity Measures”. In: *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2015, pp. 59–78. DOI: [10.1109/FOCS.2015.14](https://doi.org/10.1109/FOCS.2015.14). [arXiv:1501.07053](https://arxiv.org/abs/1501.07053).
- [ACQ22] Dorit Aharonov, Jordan Cotler, and Xiao-Liang Qi. “Quantum algorithmic measurement”. In: *Nature Communications* 13.1 (2022), p. 887. DOI: [10.1038/s41467-021-27922-0](https://doi.org/10.1038/s41467-021-27922-0). [arXiv:2101.04634](https://arxiv.org/abs/2101.04634).
- [AJ23] Shyan Akmal and Ce Jin. “Near-Optimal Quantum Algorithms for String Problems”. In: *Algorithmica* (2023). DOI: [10.1007/s00453-022-01092-x](https://doi.org/10.1007/s00453-022-01092-x). [arXiv:2110.09696](https://arxiv.org/abs/2110.09696).

- [AV21] Shyan Akmal and Virginia Vassilevska Williams. “Improved Approximation for Longest Common Subsequence over Small Alphabets”. In: *48th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 198. 2021, 13:1–13:18. DOI: [10.4230/LIPIcs.ICALP.2021.13](https://doi.org/10.4230/LIPIcs.ICALP.2021.13). [arXiv:2105.03028](https://arxiv.org/abs/2105.03028).
- [AD99] David Aldous and Persi Diaconis. “Longest Increasing Subsequences: From Patience Sorting to the Baik-Deift-Johansson Theorem”. In: *Bulletin of the American Mathematical Society* 36 (1999), pp. 413–432. DOI: [10.1090/s0273-0979-99-00796-x](https://doi.org/10.1090/s0273-0979-99-00796-x).
- [AW21] Josh Alman and Virginia Vassilevska Williams. “A Refined Laser Method and Faster Matrix Multiplication”. In: *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2021, pp. 522–539. DOI: [10.1137/1.9781611976465.32](https://doi.org/10.1137/1.9781611976465.32). [arXiv:2010.05846](https://arxiv.org/abs/2010.05846).
- [Amb02] Andris Ambainis. “Quantum lower bounds by quantum arguments”. In: *Journal of Computer and System Sciences* 64.4 (2002), pp. 750–767. DOI: <https://doi.org/10.1006/jcss.2002.1826>. [arXiv:quant-ph/0002066](https://arxiv.org/abs/quant-ph/0002066).
- [Amb07] Andris Ambainis. “Quantum Walk Algorithm for Element Distinctness”. In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239. DOI: [10.1137/S0097539705447311](https://doi.org/10.1137/S0097539705447311). [arXiv:quant-ph/0311001](https://arxiv.org/abs/quant-ph/0311001).
- [Amb10] Andris Ambainis. “A new quantum lower bound method, with an application to a strong direct product theorem for quantum search”. In: *Theory of Computing* 6.1 (2010), pp. 1–25. [arXiv:quant-ph/0508200](https://arxiv.org/abs/quant-ph/0508200).
- [Amb12] Andris Ambainis. “Variable time amplitude amplification and quantum algorithms for linear algebra problems”. In: *Proceedings of the 29th Symposium on Theoretical Aspects of Computer Science (STACS)*. Vol. 14. 2012, pp. 636–647. DOI: [10.4230/LIPIcs.STACS.2012.636](https://doi.org/10.4230/LIPIcs.STACS.2012.636). [arXiv:1010.4458](https://arxiv.org/abs/1010.4458).
- [Amb+17] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. “Separations in Query Complexity Based on Pointer Functions”. In: *Journal of the ACM* 64.5 (2017). DOI: [10.1145/3106234](https://doi.org/10.1145/3106234). [arXiv:1506.04719](https://arxiv.org/abs/1506.04719).
- [Amb+19] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. “Quantum Speedups for Exponential-Time Dynamic Programming Algorithms”. In: *Proceedings of the 2019 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2019, pp. 1783–1793. DOI: [10.1137/1.9781611975482.107](https://doi.org/10.1137/1.9781611975482.107). [arXiv:1807.05209](https://arxiv.org/abs/1807.05209).
- [ACL11] Andris Ambainis, Andrew M. Childs, and Yi-Kai Liu. “Quantum Property Testing for Bounded-degree Graphs”. In: *Proceedings of the 14th International Workshop and 15th International Conference on Approximation, Randomization, and Combinatorial Optimization: Algorithms*

- and Techniques*. 2011, pp. 365–376. DOI: [10.1007/978-3-642-22935-0_31](https://doi.org/10.1007/978-3-642-22935-0_31). [arXiv:1012.3174](https://arxiv.org/abs/1012.3174).
- [AK17] Andris Ambainis and Martins Kokainis. “Quantum Algorithm for Tree Size Estimation, with Applications to Backtracking and 2-Player Games”. In: *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC)*. 2017, pp. 989–1002. DOI: [10.1145/3055399.3055444](https://doi.org/10.1145/3055399.3055444). [arXiv:1704.06774](https://arxiv.org/abs/1704.06774).
- [Amb+20] Andris Ambainis et al. “Quantum Lower and Upper Bounds for 2D-Grid and Dyck Language”. In: *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Vol. 170. 2020, 8:1–8:14. DOI: [10.4230/LIPIcs.MFCS.2020.8](https://doi.org/10.4230/LIPIcs.MFCS.2020.8). [arXiv:2007.03402](https://arxiv.org/abs/2007.03402).
- [Ami+18] Mohammad H. Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytsky, and Roger Melko. “Quantum Boltzmann machine”. In: *Physical Review X* 8.2 (2018). DOI: [10.1103/PhysRevX.8.021050](https://doi.org/10.1103/PhysRevX.8.021050). [arXiv:1601.02036](https://arxiv.org/abs/1601.02036).
- [AKO10] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. “Polylogarithmic Approximation for Edit Distance and the Asymmetric Query Complexity”. In: *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 377–386. DOI: [10.1109/FOCS.2010.43](https://doi.org/10.1109/FOCS.2010.43). [arXiv:1005.4033](https://arxiv.org/abs/1005.4033).
- [Bea+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. “Quantum lower bounds by polynomials”. In: *Journal of the ACM* 48.4 (2001), pp. 778–797. DOI: [10.1145/502090.502097](https://doi.org/10.1145/502090.502097). [arXiv:quant-ph/9802049](https://arxiv.org/abs/quant-ph/9802049).
- [Bel12a] Aleksandrs Belovs. “Learning-Graph-Based Quantum Algorithm for k -Distinctness”. In: *Proceedings of the 53rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 2012, pp. 207–216. DOI: [10.1109/FOCS.2012.18](https://doi.org/10.1109/FOCS.2012.18). [arXiv:1205.1534](https://arxiv.org/abs/1205.1534).
- [Bel12b] Aleksandrs Belovs. “Span Programs for Functions with Constant-Sized 1-Certificates”. In: *Proceedings of the 44th ACM Symposium on the Theory of Computing (STOC)*. 2012, pp. 77–84. DOI: [10.1145/2213977.2213985](https://doi.org/10.1145/2213977.2213985). [arXiv:1105.4024](https://arxiv.org/abs/1105.4024).
- [Bel14] Aleksandrs Belovs. “Quantum Algorithms for Learning Symmetric Juntas via Adversary Bound”. In: *Proceedings of the 29th Conference on Computational Complexity (CCC)*. 2014, pp. 22–31. DOI: [10.1109/CCC.2014.11](https://doi.org/10.1109/CCC.2014.11). [arXiv:1311.6777](https://arxiv.org/abs/1311.6777).
- [Bel15] Aleksandrs Belovs. *Variations on quantum adversary*. 2015. [arXiv:1504.06943](https://arxiv.org/abs/1504.06943).

- [BR12] Aleksandrs Belovs and Ben W. Reichardt. “Span Programs and Quantum Algorithms for st -Connectivity and Claw Detection”. In: *Algorithms – ESA 2012*. 2012, pp. 193–204. DOI: [10.1007/978-3-642-33090-2_18](https://doi.org/10.1007/978-3-642-33090-2_18). [arXiv:1203.2603](https://arxiv.org/abs/1203.2603).
- [Ben16] Shalev Ben-David. “The Structure of Promises in Quantum Speedups”. In: *11th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2016, September 27-29, 2016, Berlin, Germany*. 2016, 7:1–7:14. DOI: [10.4230/LIPIcs.TQC.2016.7](https://doi.org/10.4230/LIPIcs.TQC.2016.7). [arXiv:1409.3323](https://arxiv.org/abs/1409.3323).
- [Ben+20] Shalev Ben-David, Andrew M. Childs, András Gilyén, William Kretschmer, Supartha Podder, and Daochen Wang. “Symmetries, Graph Properties, and Quantum Speedups”. In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 649–660. DOI: [10.1109/FOCS46700.2020.00066](https://doi.org/10.1109/FOCS46700.2020.00066). [arXiv:2006.12760](https://arxiv.org/abs/2006.12760).
- [BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. “Strengths and Weaknesses of Quantum Computing”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1510–1523. DOI: [10.1137/S0097539796300933](https://doi.org/10.1137/S0097539796300933). [arXiv:quant-ph/9701001](https://arxiv.org/abs/quant-ph/9701001).
- [BHS80] Jon Louis Bentley, Dorothea Haken, and James B. Saxe. “A General Method for Solving Divide-and-Conquer Recurrences”. In: *SIGACT News* 12.3 (1980), pp. 36–44. DOI: [10.1145/1008861.1008865](https://doi.org/10.1145/1008861.1008865).
- [BV97] Ethan Bernstein and Umesh Vazirani. “Quantum Complexity Theory”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1411–1473. DOI: [10.1137/S0097539796300921](https://doi.org/10.1137/S0097539796300921).
- [BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. “Quantum amplitude amplification and estimation”. In: *Contemporary Mathematics* 305 (2002), pp. 53–74. DOI: [10.1090/conm/305/05215](https://doi.org/10.1090/conm/305/05215). [arXiv:quant-ph/0005055](https://arxiv.org/abs/quant-ph/0005055).
- [BLP93] J. P. Buhler, H. W. Lenstra, and Carl Pomerance. “Factoring integers with the number field sieve”. In: *The development of the number field sieve*. 1993, pp. 50–94. DOI: [0.1007/BFb0091539](https://doi.org/0.1007/BFb0091539).
- [BŠ04] Harry Buhrman and Robert Špalek. *Quantum Verification of Matrix Products*. 2004. [arXiv:quant-ph/0409035](https://arxiv.org/abs/quant-ph/0409035).
- [CDKR20] Balthazar Casalé, Giuseppe Di Molfetta, Hachem Kadri, and Liva Ralaivola. “Quantum bandits”. In: *Quantum Machine Intelligence 2.1* (2020), p. 11. DOI: [10.1007/s42484-020-00024-8](https://doi.org/10.1007/s42484-020-00024-8). [arXiv:2002.06395](https://arxiv.org/abs/2002.06395).
- [Cha18] André Chailloux. “A Note on the Quantum Query Complexity of Permutation Symmetric Functions”. In: *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*. Vol. 124. 2018, 19:1–19:7. DOI: [10.4230/LIPIcs.ITCS.2019.19](https://doi.org/10.4230/LIPIcs.ITCS.2019.19). [arXiv:1810.01790](https://arxiv.org/abs/1810.01790).

- [CGJ19] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. “The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation”. In: *46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 132. 2019, 33:1–33:14. DOI: [10.4230/LIPIcs.ICALP.2019.33](https://doi.org/10.4230/LIPIcs.ICALP.2019.33). [arXiv:1804.01973](https://arxiv.org/abs/1804.01973).
- [CL15] Lijie Chen and Jian Li. *On the optimal sample complexity for best arm identification*. 2015. [arXiv:1511.03774](https://arxiv.org/abs/1511.03774).
- [CLQ17] Lijie Chen, Jian Li, and Mingda Qiao. “Towards Instance Optimal Bounds for Best Arm Identification”. In: *Conference on Learning Theory*. 2017, pp. 535–592. [arXiv:1608.06031](https://arxiv.org/abs/1608.06031).
- [CCHL22] S. Chen, J. Cotler, H. Huang, and J. Li. “Exponential Separations Between Learning With and Without Quantum Memory”. In: *Proceedings of the 62nd IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 574–585. DOI: [10.1109/FOCS52979.2021.00063](https://doi.org/10.1109/FOCS52979.2021.00063). [arXiv:2111.05881](https://arxiv.org/abs/2111.05881).
- [Chi+03] Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. “Exponential Algorithmic Speedup by a Quantum Walk”. In: *Proceedings of the 35th ACM Symposium on the Theory of Computing (STOC)*. 2003, pp. 59–68. DOI: [10.1145/780542.780552](https://doi.org/10.1145/780542.780552). [arXiv:quant-ph/0209131](https://arxiv.org/abs/quant-ph/0209131).
- [Chi+22] Andrew M. Childs, Robin Kothari, Matt Kovacs-Deak, Aarthi Sundaram, and Daochen Wang. *Quantum divide and conquer*. 2022. [arXiv:2210.06419](https://arxiv.org/abs/2210.06419).
- [CKS17] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. “Quantum Algorithm for Systems of Linear Equations with Exponentially Improved Dependence on Precision”. In: *SIAM Journal on Computing* 46.6 (2017), pp. 1920–1950. DOI: [10.1137/16M1087072](https://doi.org/10.1137/16M1087072). [arXiv:1511.02306](https://arxiv.org/abs/1511.02306).
- [CT65] James W. Cooley and John W. Tukey. “An Algorithm for the Machine Calculation of Complex Fourier Series”. In: *Mathematics of Computation* 19.90 (1965), pp. 297–301. DOI: [10.2307/2003354](https://doi.org/10.2307/2003354).
- [CW90] Don Coppersmith and Shmuel Winograd. “Matrix multiplication via arithmetic progressions”. In: *Journal of Symbolic Computation* 9.3 (1990), pp. 251–280. DOI: [10.1016/S0747-7171\(08\)80013-2](https://doi.org/10.1016/S0747-7171(08)80013-2).
- [CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. Third. MIT Press, 2009.
- [CJOP20] Arjan Cornelissen, Stacey Jeffery, Maris Ozols, and Alvaro Piedrafito. “Span Programs and Quantum Time Complexity”. In: *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Vol. 170. 2020, 26:1–26:14. DOI: [10.4230/LIPIcs.MFCS.2020.26](https://doi.org/10.4230/LIPIcs.MFCS.2020.26). [arXiv:2005.01323](https://arxiv.org/abs/2005.01323).

- [Deu85] David Deutsch. “Quantum theory, the Church–Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (1985), pp. 97–117. DOI: [10.1098/rspa.1985.0070](https://doi.org/10.1098/rspa.1985.0070).
- [DJ92] David Deutsch and Richard Jozsa. “Rapid Solution of Problems by Quantum Computation”. In: *Proceedings of the Royal Society of London Series A* 439.1907 (1992), pp. 553–558. DOI: [10.1098/rspa.1992.0167](https://doi.org/10.1098/rspa.1992.0167).
- [DB18] Vedran Dunjko and Hans J. Briegel. “Machine learning & artificial intelligence in the quantum domain: a review of recent progress”. In: *Reports on Progress in Physics* 81.7 (2018), p. 074001. DOI: [10.1088/1361-6633/aab406](https://doi.org/10.1088/1361-6633/aab406). [arXiv:1709.02779](https://arxiv.org/abs/1709.02779).
- [DLWT17] Vedran Dunjko, Yi-Kai Liu, Xingyao Wu, and Jacob M. Taylor. *Exponential improvements for quantum-accessible reinforcement learning*. 2017. [arXiv:1710.11160](https://arxiv.org/abs/1710.11160).
- [DTB16] Vedran Dunjko, Jacob M. Taylor, and Hans J Briegel. “Quantum-enhanced machine learning”. In: *Physical Review Letters* 117.13 (2016). DOI: [10.1103/PhysRevLett.117.130501](https://doi.org/10.1103/PhysRevLett.117.130501). [arXiv:1610.08251](https://arxiv.org/abs/1610.08251).
- [DTB17] Vedran Dunjko, Jacob M. Taylor, and Hans J. Briegel. “Advances in quantum reinforcement learning”. In: *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics*. 2017, pp. 282–287. DOI: [10.1109/SMC.2017.8122616](https://doi.org/10.1109/SMC.2017.8122616). [arXiv:1811.08676](https://arxiv.org/abs/1811.08676).
- [DH96] Christoph Dürr and Peter Høyer. *A Quantum Algorithm for Finding the Minimum*. 1996. [arXiv:quant-ph/9607014](https://arxiv.org/abs/quant-ph/9607014).
- [EMM02] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. “PAC Bounds for Multi-armed Bandit and Markov Decision Processes”. In: *Computational Learning Theory*. 2002, pp. 255–270. DOI: [10.1007/3-540-45435-7_18](https://doi.org/10.1007/3-540-45435-7_18).
- [FRPU94] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. “Computing with Noisy Information”. In: *SIAM Journal on Computing* 23.5 (1994), pp. 1001–1018. DOI: [10.1137/S0097539791195877](https://doi.org/10.1137/S0097539791195877).
- [Fre75] Michael L. Fredman. “On computing the length of longest increasing subsequences”. In: *Discrete Mathematics* 11.1 (1975), pp. 29–35. DOI: [10.1016/0012-365X\(75\)90103-X](https://doi.org/10.1016/0012-365X(75)90103-X).
- [GGL12] Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. “Best arm identification: A unified approach to fixed budget and fixed confidence”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 3212–3220. [hal:00747005](https://hal.archives-ouvertes.fr/hal-00747005).
- [GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. “Quantum Singular Value Transformation and beyond: Exponential Improvements for Quantum Matrix Arithmetics”. In: *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*. 2019, pp. 193–204. DOI: [10.1145/3313276.3316366](https://doi.org/10.1145/3313276.3316366). [arXiv:1806.01838](https://arxiv.org/abs/1806.01838).

- [GKMV21] Adam Glos, Martins Kokainis, Ryuhei Mori, and Jevgēnijs Vihrovs. “Quantum Speedups for Dynamic Programming on n -Dimensional Lattice Graphs”. In: *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Vol. 202. 2021, 50:1–50:23. DOI: [10.4230/LIPIcs.MFCS.2021.50](https://doi.org/10.4230/LIPIcs.MFCS.2021.50). [arXiv:2104.14384](https://arxiv.org/abs/2104.14384).
- [GR02] Goldreich and Ron. “Property Testing in Bounded Degree Graphs”. In: *Algorithmica* 32.2 (2002), pp. 302–343. DOI: [10.1007/s00453-001-0078-7](https://doi.org/10.1007/s00453-001-0078-7).
- [Gro96] Lov K. Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the 28th ACM Symposium on the Theory of Computing (STOC)*. 1996, pp. 212–219. DOI: [10.1145/237814.237866](https://doi.org/10.1145/237814.237866). [arXiv:quant-ph/9605043](https://arxiv.org/abs/quant-ph/9605043).
- [Hoa62] C. A. R. Hoare. “Quicksort”. In: *The Computer Journal* 5.1 (1962), pp. 10–16. DOI: [10.1093/comjnl/5.1.10](https://doi.org/10.1093/comjnl/5.1.10).
- [HLŠ07] Peter Høyer, Troy Lee, and Robert Špalek. “Negative Weights Make Adversaries Stronger”. In: *Proceedings of the 39th ACM Symposium on the Theory of Computing (STOC)*. 2007, pp. 526–535. DOI: [10.1145/1250790.1250867](https://doi.org/10.1145/1250790.1250867). [arXiv:quant-ph/0611054](https://arxiv.org/abs/quant-ph/0611054).
- [HŠ05] Peter Høyer and Robert Špalek. “Lower Bounds on Quantum Query Complexity”. In: *Bulletin of the EATCS* 87 (2005), pp. 78–103. [arXiv:quant-ph/0509153](https://arxiv.org/abs/quant-ph/0509153).
- [HKP21] Hsin-Yuan Huang, Richard Kueng, and John Preskill. “Information-Theoretic Bounds on Quantum Advantage in Machine Learning”. In: *Physical Review Letters* 126 (19 2021). DOI: [10.1103/PhysRevLett.126.190505](https://doi.org/10.1103/PhysRevLett.126.190505). [arXiv:2101.02464](https://arxiv.org/abs/2101.02464).
- [JMNB14] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. “lil’UCB: An optimal exploration algorithm for multi-armed bandits”. In: *Conference on Learning Theory*. 2014, pp. 423–439. [arXiv:1312.7308](https://arxiv.org/abs/1312.7308).
- [Jef22] Stacey Jeffery. “Span Programs and Quantum Space Complexity”. In: *Theory of Computing* 18.11 (2022), pp. 1–49. DOI: [10.4086/toc.2022.v018a011](https://doi.org/10.4086/toc.2022.v018a011). [arXiv:1908.04232](https://arxiv.org/abs/1908.04232).
- [Jer+21] Sofiene Jerbi, Lea M. Trenkwalder, Hendrik Poulsen Nautrup, Hans J. Briegel, and Vedran Dunjko. “Quantum Enhancements for Deep Reinforcement Learning in Large Spaces”. In: *PRX Quantum* 2 (1 Feb. 2021), p. 010328. DOI: [10.1103/PRXQuantum.2.010328](https://doi.org/10.1103/PRXQuantum.2.010328). [arXiv:1910.12760](https://arxiv.org/abs/1910.12760).
- [KO63] Anatolij A. Karatsuba and Yu. Ofman. “Multiplication of Multidigit Numbers on Automata”. In: *Soviet Physics Doklady* 7 (1963), pp. 595–596.

- [KKS13] Zohar Karnin, Tomer Koren, and Oren Somekh. “Almost optimal exploration in multi-armed bandits”. In: *International Conference on Machine Learning*. 2013, pp. 1238–1246.
- [KLLP19] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. “q-means: A quantum algorithm for unsupervised machine learning”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 4136–4146. [arXiv:1812.03584](#).
- [Kim13] Shelby Kimmel. “Quantum Adversary (Upper) Bound”. In: *Chicago Journal of Theoretical Computer Science* 2013.4 (2013). DOI: [10.4086/cjtcs.2013.004](#). [arXiv:1101.0797](#).
- [KPV22] Vladislavs Kļevickis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. “Quantum Speedups for Treewidth”. In: *17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022)*. Vol. 232. 2022, 11:1–11:18. DOI: [10.4230/LIPIcs.TQC.2022.11](#). [arXiv:2202.08186](#).
- [LMS98] Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt. “Incremental String Comparison”. In: *SIAM Journal on Computing* 27.2 (1998), pp. 557–582. DOI: [10.1137/S0097539794264810](#).
- [Lee+22] Seunghoon Lee et al. *Is there evidence for exponential quantum advantage in quantum chemistry?* 2022. [arXiv:2208.02199](#).
- [LMS17] Troy Lee, Frédéric Magniez, and Miklos Santha. “Improved Quantum Query Algorithms for Triangle Detection and Associativity Testing”. In: *Algorithmica* 77.2 (2017), pp. 459–486. DOI: [10.1007/s00453-015-0084-9](#). [arXiv:1210.1014](#).
- [LMRŠ10] Troy Lee, Rajat Mittal, Ben W. Reichardt, and Robert Špalek. *An adversary for algorithms*. 2010. [arXiv:1011.3020v1](#).
- [Lee+11] Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. “Quantum Query Complexity of State Conversion”. In: *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*. 2011, pp. 344–353. DOI: [10.1109/FOCS.2011.75](#). [arXiv:1011.3020](#).
- [LSZ21] Troy Lee, Miklos Santha, and Shengyu Zhang. “Quantum algorithms for graph problems with cut queries”. In: *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2021, pp. 939–958. DOI: [10.1137/1.9781611976465.59](#). [arXiv:2007.08285](#).
- [LHLW23] Jiaqi Leng, Ethan Hickman, Joseph Li, and Xiaodi Wu. *Quantum Hamiltonian Descent*. 2023. [arXiv:2303.01471](#).
- [LCW19] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. “Sublinear quantum algorithms for training linear and kernel-based classifiers”. In: *International Conference on Machine Learning*. 2019, pp. 3815–3824. [arXiv:1904.02276](#).

- [LZ22] Tongyang Li and Ruizhe Zhang. *Quantum Speedups of Optimizing Approximately Convex Functions with Applications to Logarithmic Regret Stochastic Convex Bandits*. 2022. [arXiv:2209.12897](#).
- [Lie84] Martin W. Liebeck. “On minimal degrees and base sizes of primitive permutation groups”. In: *Archiv der Mathematik* 43.1 (1984), pp. 11–15. DOI: [10.1007/BF01193603](#).
- [LT20] Lin Lin and Yu Tong. “Near-optimal ground state preparation”. In: *Quantum* 4 (2020), p. 372. DOI: [10.22331/q-2020-12-14-372](#). [arXiv:2002.12508](#).
- [LMR13] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. *Quantum algorithms for supervised and unsupervised machine learning*. 2013. [arXiv:1307.0411](#).
- [LMR14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. “Quantum principal component analysis”. In: *Nature Physics* 10.9 (2014), p. 631. DOI: [10.1038/nphys3029](#). [arXiv:1307.0401](#).
- [MT04] Shie Mannor and John N. Tsitsiklis. “The sample complexity of exploration in the multi-armed bandit problem”. In: *Journal of Machine Learning Research* 5.Jun (2004), pp. 623–648.
- [Mon15] Ashley Montanaro. “Quantum speedup of Monte Carlo methods”. In: *Proceedings of the Royal Society A* 471.2181 (2015), p. 20150301. DOI: [10.1098/rspa.2015.0301](#). [arXiv:1504.06987](#).
- [MW16] Ashley Montanaro and Ronald de Wolf. *A Survey of Quantum Property Testing*. Graduate Surveys 7. 2016, pp. 1–81. DOI: [10.4086/toc.gs.2016.007](#). [arXiv:1310.2035](#).
- [NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: [10.1017/CB09780511976667](#).
- [RV03] H. Ramesh and V. Vinay. “String matching in $\tilde{O}(\sqrt{n} + \sqrt{m})$ quantum time”. In: *Journal of Discrete Algorithms* 1.1 (2003), pp. 103–110. DOI: [10.1016/S1570-8667\(03\)00010-8](#). [arXiv:quant-ph/0011049](#).
- [RML14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. “Quantum support vector machine for big data classification”. In: *Physical Review Letters* 113.13 (2014). DOI: [10.1103/PhysRevLett.113.130503](#). [arXiv:1307.0471](#).
- [Rei09a] Ben W. Reichardt. “Span Programs and Quantum Query Complexity: The General Adversary Bound Is Nearly Tight for Every Boolean Function”. In: *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2009, pp. 544–551. DOI: [10.1109/FOCS.2009.55](#). [arXiv:0904.2759](#).
- [Rei09b] Ben W. Reichardt. *Span-program-based quantum algorithm for evaluating unbalanced formulas*. 2009. [arXiv:0907.1622](#).

- [Rei11] Ben W. Reichardt. “Reflections for Quantum Query Algorithms”. In: *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2011, pp. 560–569. DOI: [10.5555/2133036.2133080](https://doi.org/10.5555/2133036.2133080). [arXiv:1005.1601](https://arxiv.org/abs/1005.1601).
- [RŠ12] Ben Reichardt and Robert Špalek. “Span-Program-Based Quantum Algorithm for Evaluating Formulas”. In: *Theory of Computing* 8.13 (2012), pp. 291–319. DOI: [10.4086/toc.2012.v008a013](https://doi.org/10.4086/toc.2012.v008a013). [arXiv:0710.2630](https://arxiv.org/abs/0710.2630).
- [Rob52] Herbert Robbins. “Some aspects of the sequential design of experiments”. In: *Bulletin of the American Mathematical Society* 58.5 (1952), pp. 527–535. DOI: [bams/1183517370](https://doi.org/10.2307/2371737).
- [RSSS19] Aviad Rubinfeld, Saeed Seddighin, Zhao Song, and Xiaorui Sun. “Approximation Algorithms for LCS and LIS with Truly Improved Running Times”. In: *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2019, pp. 1121–1145. DOI: [10.1109/FOCS.2019.00071](https://doi.org/10.1109/FOCS.2019.00071). [arXiv:2111.10538](https://arxiv.org/abs/2111.10538).
- [SS10] Michael Saks and C. Seshadhri. “Estimating the Longest Increasing Sequence in Polylogarithmic Time”. In: *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 458–467. DOI: [10.1109/FOCS.2010.51](https://doi.org/10.1109/FOCS.2010.51). [arXiv:1308.0626](https://arxiv.org/abs/1308.0626).
- [Shi03] Yaoyun Shi. “Both Toffoli and Controlled-NOT Need Little Help to Do Universal Quantum Computing”. In: *Quantum Information and Computation* 3.1 (2003), pp. 84–92. [arXiv:0205115](https://arxiv.org/abs/0205115).
- [Sho97] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* (1997). DOI: [10.1137/S0097539795293172](https://doi.org/10.1137/S0097539795293172). [arXiv:quant-ph/9508027](https://arxiv.org/abs/quant-ph/9508027).
- [Sil+16] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489. DOI: [10.1038/nature16961](https://doi.org/10.1038/nature16961).
- [Sim94] D. R. Simon. “On the Power of Quantum Computation”. In: *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1994, pp. 116–123. DOI: [10.1109/SFCS.1994.365701](https://doi.org/10.1109/SFCS.1994.365701).
- [Str69] Volker Strassen. “Gaussian elimination is not optimal”. In: *Numerische Mathematik* 13.4 (1969), pp. 354–356. DOI: [10.1007/BF02165411](https://doi.org/10.1007/BF02165411).
- [Sze04] Mario Szegedy. “Quantum speed-up of Markov Chain based algorithms”. In: *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2004, pp. 32–41. DOI: [10.1109/FOCS.2004.53](https://doi.org/10.1109/FOCS.2004.53).
- [WF74] Robert A. Wagner and Michael J. Fischer. “The String-to-String Correction Problem”. In: *Journal of the ACM* 21.1 (1974), pp. 168–173. DOI: [10.1145/321796.321811](https://doi.org/10.1145/321796.321811).

- [Wan+22] Zongqi Wan, Zhijie Zhang, Tongyang Li, Jialin Zhang, and Xiaoming Sun. *Quantum Multi-Armed Bandits and Stochastic Linear Bandits Enjoy Logarithmic Regrets*. 2022. [arXiv:2205.14988](#).
- [Wan+21] Daochen Wang, Aarthi Sundaram, Robin Kothari, Ashish Kapoor, and Martin Roetteler. “Quantum algorithms for reinforcement learning with a generative model”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. 18–24 Jul 2021, pp. 10916–10926. [arXiv:2112.08451](#).
- [WYLC21] Daochen Wang, Xuchen You, Tongyang Li, and Andrew M. Childs. “Quantum Exploration Algorithms for Multi-Armed Bandits”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.11 (2021), pp. 10102–10110. [arXiv:2007.07049](#).
- [WKS15] Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore. “Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning”. In: *Quantum Information and Computation* 15.3-4 (2015), pp. 316–356. [arXiv:1401.2142](#).
- [Wol01] Ronald de Wolf. “Quantum Computing and Communication Complexity”. PhD thesis. University of Amsterdam, 2001.
- [YZ22] Takashi Yamakawa and Mark Zhandry. “Verifiable Quantum Advantage without Structure”. In: *Proceedings of the 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 69–74. DOI: [10.1109/FOCS54457.2022.00014](#). [arXiv:2204.02063](#).
- [Yao77] Andrew Chi-Chih Yao. “Probabilistic computations: Toward a unified measure of complexity”. In: *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1977, pp. 222–227. DOI: [10.1109/SFCS.1977.24](#).
- [ZKH12] Bohua Zhan, Shelby Kimmel, and Avinandan Hassidim. “Super-Polynomial Quantum Speed-Ups for Boolean Evaluation Trees with Hidden Structure”. In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*. 2012, pp. 249–265. DOI: [10.1145/2090236.2090258](#). [arXiv:1101.0796](#).
- [Zha12a] Mark Zhandry. “How to Construct Quantum Random Functions”. In: *Proceedings of the 53rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 2012, pp. 679–687. DOI: [10.1109/FOCS.2012.37](#). [iacr:2012/182](#).
- [Zha12b] Mark Zhandry. “Secure Identity-Based Encryption in the Quantum Random Oracle Model”. In: *Advances in Cryptology – CRYPTO 2012*. 2012, pp. 758–775. [iacr:2012/076](#).
- [Zha15] Mark Zhandry. “A Note on the Quantum Collision and Set Equality Problems”. In: *Quantum Information and Computation* 15.7-8 (2015), pp. 557–567. [arXiv:1312.1027](#).