ABSTRACT

Title of Dissertation: QUANTUM QUERY ALGORITHMS: DESIGN, OPTIMALITY, COMPLEXITY

Máté Kovács-Deák
Doctor of Philosophy, 2025

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

In this thesis we investigate quantum query algorithms from three perspectives: design paradigms, optimality, and complexity.

First, we study how the divide and conquer paradigm—widely used in classical algorithm design—can be adapted to quantum algorithms. We leverage the quantum adversary method to develop a generic framework for designing quantum query algorithms using divide and conquer. We demonstrate the utility of our framework by proposing near-optimal quantum query algorithms for a variety of string processing problems, such as decision versions of the string rotation, string suffix, longest increasing subsequence, and longest common subsequence problems.

Next, we study translation-invariant quantum algorithms for the ordered search problem. On a classical computer, this problem is solved optimally by the binary search algorithm using $\lceil \log_2 n \rceil$ queries. Quantum computers offer only a constant-factor speedup: the quantum query complexity of solving this problem (with no error) is known to lie between approximately $0.221 \log_2 n$ and $0.433 \log_2 n$. We make progress towards closing this gap in two ways. First, we improve the above upper bound to approximately $0.390 \log_2 n$. Second, we prove that the class of translation-invariant algorithms, to which most algorithms proposed for this problem belong, is in fact optimal for ordered search. A consequence of our result is that no workspace is needed by optimal algorithms for this problem.

Finally, we consider the long-standing open question of whether the rational degree is polynomially related to the degree of a total Boolean function, a question that characterizes the power of exact, postselected quantum algorithms. We prove asymptotically tight bounds on the rational degree in terms of the degree for special classes of functions such as symmetric and unate functions. Additionally, we show that almost all Boolean functions on n variables have rational degree at least $n/2 - O(\sqrt{n})$. We also establish AND and OR composition lemmas for the rational degree and exhibit new polynomial separations between the rational degree and other well-studied complexity measures, such as sensitivity and spectral sensitivity.

Quantum Query Algorithms: Design, Optimality, Complexity

by

Máté Kovács-Deák

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
2025

Advisory Committee:
Professor Andrew M. Childs, Chair/Advisor
Professor Tamás Darvas, Dean's Representative
Professor William Gasarch
Professor Daniel Gottesman
Professor Runzhou Tao

Table of Contents

Chapter 1

Introduction

A model of computation is a formal framework that sets the scene for a systematic study of algorithms. It describes what it means to compute: what a computational problem is, a set of permitted operations (that serve as building blocks of algorithms), and a mechanism of assigning cost to these operations. This way, different algorithms solving the same problem can be analyzed and compared in terms of the total cost of the operations involved, i.e., their computational complexity.

The complexity of a problem is then defined as the complexity of the most efficient algorithm that solves the problem. Thus, to ascertain the complexity of a given problem a computer scientist needs to do two things. On the one hand, one needs to prove a lower bound by identifying fundamental barriers that no conceivable algorithm can surpass. On the other, one needs to construct an efficient algorithm, one that places a matching upper bound on the complexity of the problem under consideration.

For many problems, gaps persist between the best known upper and lower bounds. It is in this region where much of modern algorithmic research takes place. The tools available to researchers in narrowing these gaps depend on the computational model under consideration. For example, consider Turing machines. On the algorithms side, paradigms such as dynamic programming, greedy algorithms, and divide and conquer are widely used to design algorithms

[CLRS09]. For lower bounds, a variety of techniques have been devised, such as diagonalization, counting arguments, reductions, adversary arguments, etc. [AB09]

In the field of quantum computing, although an analog of the Turing machine model has been developed [Deu85], it is not widely adopted. Instead, most quantum algorithms are constructed and analyzed in the query complexity model [BdW02]. Unlike in the case of Turing machines, in this model the algorithm does not have direct access to the input, instead it must query it piece-by-piece. The emphasis is on the amount of information the algorithm needs to obtain about its input in order to solve the problem. Typically, the task is to compute the value of a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ on some input x. The only operations of nonzero cost are querying the bits of the input x.

The success of quantum query complexity is partly due to a number powerful techniques developed for proving lower bounds, such as the polynomial method [BBC+01] and the adversary method [Amb00; HLŠ07; LMR+11]. Another way to bound the query complexity of a problem is to study various other measures of Boolean complexity (such as sensitivity, certificate complexity, etc.), as many of these measures are polynomially related to the quantum query complexity [BdW02]. For proving upper bounds, techniques such as quantum walks [ADZ93; FG98; AAKV01; CCD+03], amplitude amplification [Gro96; BHMT02], and the quantum Fourier transform [Sho94] have proven to be widely applicable.

1.1 Preliminaries

Query Complexity

In the query complexity model we are usually interested in computing some Boolean function $f\colon \{0,1\}^n \to \{0,1\}$. Here, unlike in the Turing machine model, an algorithm tasked with computing $f(x)$ does not have direct access to the input $x \in \{0,1\}^n$. Instead, it must obtain x bit-by-bit by making queries to an oracle. All deterministic operations are permitted in between queries, which may depend on the result of earlier ones. An algorithm computes f if for any $x \in \{0,1\}^n$, on input x, the algorithm outputs $f(x)$ on termination. The query complexity of the algorithm is the maximal number of queries it makes on an input. The deterministic query complexity of f, denoted $\mathrm{D}(f)$ is the query complexity of the best performing (deterministic) algorithm that computes f.

To illustrate this concept, we consider three examples. First, let $\mathsf{DICT}_n : \{0,1\}^n \to \{0,1\}$ be the dictator function $\mathsf{DICT}_n(x) = x_1$. Querying the bit x_1 is sufficient and necessary to determine the value of this function, and thus $\mathrm{D}(\mathsf{DICT}_n) = 1$. Now suppose that we want to compute the Boolean OR of n bits, $\mathsf{OR}_n(x) = \vee_{j=1}^n x_j$. Clearly, an algorithm can terminate and output 1 whenever it queries a bit that is 1. However, it may be that after $n-1$ queries the algorithm still has not seen a 1, in which case it needs to make one more query to determine the value of f. Thus, $\mathrm{D}(\mathsf{OR}_n) = n$. Our final example is the so-called ordered search problem. Unlike the previous examples, this problem is represented by a partial function with a non-boolean codomain as follows. Let S be the set of all strings of the form $0^j 1^{2^n - j}$ for $j = 0, \ldots, 2^n - 1$. Define $\mathsf{OS}_{2^n} : S \to [2^n]$ by $\mathsf{OS}(0^j 1^{2^n-j}) = j + 1$. That is to say, f returns the coordinate of the first 1 in its input. It can be shown using tools from information theory that $\mathrm{D}(\mathsf{OS}_{2^n}) \geq n$. On the other hand, the binary search algorithm solves this problem using n queries [Knu98]. We conclude that $\mathrm{D}(\mathsf{OS}_{2^n}) = n$.

Allowing nondeterministic operations between queries leads to more powerful models, such as randomized and quantum variants. Of these, of primary interest in this thesis is the model of quantum query complexity, which we introduce below.

Quantum Query Algorithms

Since an idealized quantum computer manipulates its state by unitary evolution, we cannot rely on the map $j \mapsto x_j$ for the oracle. Instead, the oracle is usually implemented by the unitary map $O_x : \ |j\rangle |b\rangle \mapsto |j\rangle |x_j \oplus b\rangle$.

A k-query quantum algorithm $\mathscr{A}$ is specified by some initial state $|\psi_0\rangle$ and a sequence of k unitary operators $U_1, \ldots, U_k$. The final state of $\mathscr{A}$ given access to the oracle O_x is

$$|\psi_k^{(x)}\rangle = U_k O_x U_{k-1} \ldots U_1 O_x |\psi_0\rangle .$$

We say that $\mathscr{A}$ computes f with error ε if measuring the last qubit of $|\psi_k^{(x)}\rangle$ in the standard basis, we obtain $f(x)$ with probability at least $1 - \varepsilon$.

The exact quantum query complexity of f, denoted $\mathsf{Q}_E(f)$, is the minimal value of k for which a k-query algorithm computes f with error $\varepsilon = 0$. The bounded-error quantum query complexity (denoted $\mathsf{Q}(f)$) is defined similarly, except the error parameter is taken to be $\varepsilon = 1/3$.

A more powerful (albeit less realistic) model is the so-called postselected quantum query model. A postselected (quantum) algorithm $\mathscr{A}$ is essentially a regular quantum query algorithm, except the conditions under which $\mathscr{A}$ is considered successful are different. This time, two qubits of the final state are measured to obtain an output consisting of two bits $a, b \in \{0, 1\}$. We say that $\mathscr{A}$ computes f with error ϵ if $\Pr[a = 1] > 0$ and $\Pr[b = f(x) \mid a = 1] \geq 1 - \varepsilon$. The intuition is that we could use such an algorithm to compute $f(x)$ (with error probability ε) if we had the capability of forcing (postselecting) the measurement outcome $a = 1$. While the outcome $a = 1$ may be exceedingly unlikely, conditional on it occurring the bit b gives the right answer with high probability. The postselection query complexity $\mathsf{PostQ}_\varepsilon(f)$ of f is the minimal query complexity of a postselected algorithm that computes f with error ε. When $\varepsilon = 0$ we use $\mathsf{PostQ}_E(f)$ instead of $\mathsf{PostQ}_0(f)$ to denote the exact postselection query complexity.

Consider the earlier example of the Boolean OR function, $\mathsf{OR}_n : \{0,1\}^n \to \{0,1\}$. Among regular (bounded-error) quantum query algorithms, Grover's algorithm is asymptotically optimal,

and the quantum query complexity of OR_n is $\Theta(\sqrt{n})$. With postselection, we can do significantly better as the following construction of Mahadev and de Wolf demonstrates [MdW14]. Suppose that the initial state of an algorithm is

$$\delta |0\rangle |1\rangle + \sqrt{\frac{1-\delta^2}{n}} \sum_{j=1}^{n} |j\rangle |0\rangle$$

where $\delta > 0$ is a small constant. One query to the input oracle transforms this state into

$$\delta |0\rangle |1\rangle + \sqrt{\frac{1-\delta^2}{n}} \sum_{j=1}^{n} |j\rangle |x_j\rangle .$$

Projecting onto the subspace where the last qubit equals $|1\rangle$ we get a state proportional to

$$\delta |0\rangle |1\rangle + \sqrt{\frac{1-\delta^2}{n}} \sum_{j\,:\,x_j=1} |j\rangle |1\rangle .$$

Suppose that we measure the first qubit. When the input is $x = 0^n$, only the first term of the above expression is present and we measure 0. Otherwise, the second term is present and (provided that δ is sufficiently small) the measurement outcome will equal some $j \in [n]$ such that $x_j = 1$ with probability close to 1. Consequently, the postselected query complexity of OR_n for $\varepsilon > 0$ is 1.

Boolean Function Analysis

A Boolean function is a function $f\colon \Sigma^n \to \Sigma$ where Σ is a set of two elements. While $\Sigma = \{0, 1\}$ is a common choice, for now assume $\Sigma = \{-1, 1\}$. We refer to Σ^n as the Boolean hypercube.

A special class of Boolean functions is the set of 2^n parity functions χ_S indexed by subsets $S \subseteq \Sigma^n$. The value of χ_S on an input $x \in \Sigma^n$ is -1 if and only if the number of -1's in the entries of x indexed by S is odd. The parity functions form a basis of the space $\mathbb{R}^{\Sigma^n}$ of real-valued functions $p\colon \Sigma^n \to \mathbb{R}$ on the Boolean cube. With pointwise multiplication, $\mathbb{R}^{\Sigma^n}$ is a commutative $\mathbb{R}$-algebra. Mapping χ_S to the formal polynomial $X_S = \prod_{i \in S} X_i \in \mathbb{R}[X_1, \ldots, X_n]$ gives [Alo93; ODo14] an isomorphism of $\mathbb{R}$-algebras,

$$\mathbb{R}^{\Sigma^n} \simeq \mathbb{R}[X_1, \ldots, X_n]/\langle X_1^2 - X_1, \ldots, X_n^2 - X_n\rangle.$$

This essentially shows that given a real-valued function p on the hypercube, there exists a unique multilinear polynomial $\sum_{S \subseteq \Sigma^n} \hat{p}_S X_S$ that agrees on Σ^n with p. Thus, to each function $p\colon \Sigma^n \to \mathbb{R}$ we can associate a degree, by letting the degree of p, denoted $\deg(p)$, be the degree of the unique multilinear polynomial representing p.

The degree of a Boolean function f is just one among many complexity measures of f. Other notable examples are the approximate degree, sensitivity, certificate complexity, etc. The approximate degree $\widetilde{\deg}(f)$ is defined as the minimum degree of a multilinear polynomial that ε-approximates f (say for $\varepsilon = 1/3$) on the Boolean cube. A benefit of studying these measures is due to their many interconnections to query complexity. For instance, the degree and the approximate degree give lower bounds on the exact and bounded-error quantum query complexities [BBC+01]:

$$\frac{1}{2}\deg(f) \leq \mathrm{Q}_E(f) \qquad \text{and} \qquad \frac{1}{2}\widetilde{\deg}(f) \leq \mathrm{Q}(f).$$

Thanks to a web of inequalities like these, many key results concerning quantum query complexity have been established through other measures of Boolean complexity [NS94; ABK16; ABK+21].

The Rational Degree

As discussed above each Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is uniquely represented by a multilinear polynomial. We may also consider representations by rational functions. A rational function is a formal ratio p/q of two functions $p, q\colon \{0,1\}^n \to \mathbb{R}$ where q is required to be nonzero on the entire hypercube. The rational function p/q is a rational representation of f if

$$\forall x \in \{0,1\}^n : \qquad f(x) = \frac{p(x)}{q(x)}.$$

The degree of p/q is the larger of the degrees of p and q. Unlike polynomial representations, rational representations of f are not unique. The rational degree of f, denoted $\mathrm{rdeg}(f)$, is defined as the minimal degree of a rational representation of f.

How does the rational degree of f relate to its (polynomial) degree? While it is clear that $\mathrm{rdeg}(f) \leq \deg(f)$, it is a long standing open problem to find a lower bound of $\mathrm{rdeg}(f)$ in terms of $\deg(f)$. This question was first posed by Fortnow over 30 years ago [NS94], and in that time very little progress has been made on this matter.

One way to motivate this question is through the lens of quantum query complexity. Given $\varepsilon > 0$, let $\mathrm{rdeg}_\varepsilon(f)$ denote the minimal degree of a rational function that ε-approximates f on the Boolean cube. Mahadev and de Wolf showed that the rational degree is asymptotically equivalent to the postselection query complexity, $\frac{1}{2}\,\mathrm{rdeg}_\varepsilon(f) \leq \mathsf{PostQ}_\varepsilon(f) \leq \mathrm{rdeg}_\varepsilon(f)$ for any choice of $\varepsilon \in [0,1)$ [MdW14]. Thus, we can view Fortnow's question as one concerning the power of postselection—namely, how much smaller $\mathsf{PostQ}_E(f)$ can be than $\mathsf{Q}_E(f)$.

Consider our earlier example of the Boolean OR function on n bits. As we have seen OR_n can be computed (with nonzero error) by a postselected algorithm using a single query. An alternative way to see this is to consider the rational polynomial

$$\frac{\sum_{j=1}^n x_j}{\varepsilon + \sum_{j=1}^n x_j}.$$

This is a degree 1 rational function that approximates OR_n up to error ε. Indeed, when $x = 0^n$ this function equals 0. For $x \neq 0^n$, the value of this function is within distance ε of 1. What about exact representations? As we show in Section 4.4, $\mathrm{rdeg}(\mathsf{OR}_n) = n$.

Another intriguing perspective on this question is obtained by considering Hilbert's Nullstellensatz[1]. First, let us introduce one more complexity measure called the nondeterministic degree. A nondeterministic representation of a Boolean function f is a map $p\colon \{0,1\}^n \to \mathbb{R}$ such that $f(x) = 1$ if and only if $p(x) \neq 0$. The nondeterministic degree $\mathrm{ndeg}(f)$ of f is the minimal degree of a nondeterministic representation of f. de Wolf observed that the rational degree equals the greater of the nondeterministic degree of f and that of its negation $\bar{f}$:

$$\mathrm{rdeg}(f) = \max\{\mathrm{ndeg}(f), \mathrm{ndeg}(\bar{f})\}.$$

Now consider the sharp effective Nullstellensatz due to Kollár [Kol88]:

Theorem 1 (Sharp Effective Nullstellensatz)**.** *Let $p_1, \ldots, p_k \in \mathbb{R}[X_1, \ldots, X_n]$ be polynomials of degrees $d_1 \leq d_2 \leq \ldots \leq d_k$. If the polynomials have no common zeros in $\mathbb{C}^n$ then there exist polynomials $q_1, \ldots, q_k \in \mathbb{R}[X_1, \ldots, X_n]$ such that $\deg(p_i q_i) \leq \max(d_k, 3) \prod_{j=1}^{\min(n,k)-1} d_j$ and*

$$p_1 q_1 + \ldots + p_k q_k = 1.$$

Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, let p_1 and p_2 be nondeterministic representations of f and its negation respectively, of minimal degree. Replacing f with its negation we may assume that $\mathrm{ndeg}(f) \geq \mathrm{ndeg}(\bar{f})$. For simplicity of presentation, let us also assume that $\mathrm{ndeg}(f) \geq 3$. By the definition of nondeterministic representations, p_1 and p_2 have no common zeros in $\{0,1\}^n$. If p_1 and p_2 have no common zeros on the entirety of $\mathbb{C}^n$, then Theorem 1 implies that there exist polynomials $q_1, q_2 \in \mathbb{R}[X_1, \ldots, X_n]$ such that $p_1 q_1 + p_2 q_2 = 1$ and $\deg(p_1 q_1) \leq \mathrm{ndeg}(f)\,\mathrm{ndeg}(\bar{f})$. Notice that for any $x \in \{0,1\}^n$, $f(x) = 1$ implies $p_2(x) = 0$, and $f(x) = 0$ implies $p_1(x) = 0$.

[1] This previously unpublished observation is the result of an ongoing collaboration with Daochen Wang and Rain Zimin Yang.

Therefore, $p_1(x)q_1(x) = f(x)$ on the hypercube, and we have

$$\deg(f) \leq \deg(p_1 q_1) \leq \mathrm{ndeg}(f)\,\mathrm{ndeg}(\tilde{f}),$$

which is exactly the kind of relationship conjectured by de Wolf [dW00]. Thus, a resolution of Fortnow's question would follow from a discrete version of the sharp effective Nullstellensatz.

1.2 Overview

This thesis consists of three main chapters. Each of these chapters is standalone, covering a topic distinct from the others. A brief description of each chapter is given below. The contents of these chapters are essentially (save for some formatting changes) taken verbatim from the references indicated below.

Only a minimal amount of the necessary background is introduced in each of the chapters, and the reader is assumed to be familiar with basic concepts of quantum computing, query complexity and Boolean function analysis throughout this thesis. For background material on quantum computing see e.g., the textbook of Nielsen and Chuang [NC11]. For query complexity, the survey by Buhrman and de Wolf is good starting point [BdW02]. Finally, for the analysis and complexity of Boolean functions O'Donnell's textbook is a great resource [OD014].

Quantum Divide and Conquer. As mentioned in the introduction, many conceptual frameworks and design paradigms developed in the Turing machine framework have analogs in the query complexity model. Yet, despite decades of research, useful quantum algorithms with provable speedups over classical ones have proven to be elusive. While it is unclear why this is, it can be said that a quantum algorithm designer's toolkit is somewhat more limited than that available for classical algorithms. For example, consider the divide and conquer paradigm. Design and conquer is a versatile tool that underlies a diverse set of algorithms including binary search, the fast Fourier transform, and Strassen's matrix multiplication algorithm [CLRS09; CT65; Str69]. Informally,

a divide-and-conquer algorithm solves a problem by breaking it into smaller subproblems and solving these subproblems recursively to obtain a solution for the original problem. While this paradigm has found widespread use in classical algorithm design, it has not been well studied for quantum algorithms.

In Chapter 2, we present the first systematic study of this paradigm in the context of quantum query algorithms. We develop a generic framework for the design of quantum query algorithms using divide and conquer. We do this by leveraging the so-called quantum adversary method[2] [Amb00; Rei09a; LMR+11] to construct recurrence relations characterizing, informally speaking, the quantum query complexity of a problem in terms of the query complexities of its subproblems.

We demonstrate the utility of our framework by obtaining near-optimal quantum query algorithms for a variety of string processing problems, such as decisional variants of the string rotation, string suffix, longest increasing subsequence, and longest common subsequence problems. We also give a concise proof of one of the key lemmas underpinning the trichotomy theorem for the quantum query complexity of regular languages developed by Aaronson, Grier, and Schaeffer [AGS19].

Chapter 2 is based on the following publication:

> A. Childs, R. Kothari, M. Kovacs-Deak, A. Sundaram, and D. Wang. "Quantum Divide and Conquer". In: *ACM Transactions on Quantum Computing* 6.2 (Apr. 2025). doi:10.1145/3723884. arXiv:2210.06419

Translation-Invariant Algorithms for Ordered Search. Ordered search is a basic computational primitive with a wide range of applications. In its simplest form, this is the task of locating a target value within a sorted list of n numbers. Any classical deterministic algorithm requires $\lceil \log_2(n) \rceil$ queries to solve this problem, a bound attained by binary search [Knu98].

Unlike for unstructured search, quantum query algorithms can only offer a constant-factor improvement for this problem. Høyer, Neerbek and Shi gave the best known lower bound of $\frac{1}{\pi}(\ln n - 1) \approx 0.221 \log_2 n$ quantum queries [HNS02]. Most quantum algorithms proposed for this

[2] Though initially developed as a method for lower bounding the quantum query complexity, the adversary method can also be used to establish upper bounds.

problem belong to a class of algorithms called translation-invariant algorithms, developed by Farhi, Goldstone, Gutmann, and Sipser [FGGS99]. These algorithms are subject to two constraints: they may not use any workspace, and they must reflect the symmetries inherent in the problem. It is also in this framework that the best known algorithm to date—developed by Childs, Landahl and Parillo—was obtained, requiring $4\log_{605} n \approx 0.433 \log_2 n$ quantum queries [CLP07]. Therefore, a gap remains between the best known upper and lower bounds.

In Chapter 3, we make progress towards closing this gap in two ways. First, we use a linear programming approach to improve the upper bound to $5\log_{7265} n \approx 0.390 \log_2 n$. Second, we prove that the class of translation-invariant algorithms is in fact optimal for this problem. Our proof works by examining the semidefinite programming characterization of generic quantum algorithms developed by Barnum, Saks, and Szegedy [BSS03]. We show that, due to the symmetries of the ordered search problem, the Barnum-Saks-Szegedy SDP reduces to a generalization of the semidefinite programming characterization of translation-invariant algorithms developed by Childs, Landahl, and Parillo [CLP07].

Chapter 3 is based on the following paper:

J. Carolan, A. M. Childs, M. Kovacs-Deak, and L. Schaeffer. *Translation-Invariant Quantum Algorithms for Ordered Search are Optimal.* 2025. arXiv:2503.21090

The Rational Degree of Boolean Functions. For total functions, most Boolean complexity measures—whether combinatorial (e.g., sensitivity, certificate complexity), or algebraic (e.g., degree, approximate degree)—are known to be polynomially related to one another. One notable exception is the exact rational degree. In fact, it is not even known whether a bound such as $\mathrm{rdeg}(f) \geq \Omega(\log \deg(f))$ holds. In Chapter 4, we initiate a systematic study of the rational degree of Boolean functions. We establish tight lower bounds (in terms of the degree) on the rational degree for special classes of Boolean functions, such as symmetric and unate (a generalization of monotone) functions. We also prove that almost all Boolean functions on n bits satisfy $\mathrm{rdeg}(f) \geq n/2 - O(\sqrt{n})$.

Next, we prove that the nondeterministic degree composes exactly under AND, i.e., given two Boolean functions f and g on disjoint variables, $\mathrm{ndeg}(f \wedge g) = \mathrm{ndeg}(f) + \mathrm{ndeg}(g)$. Using this

composition lemma, we exhibit a degree-$3/2$ asymptotic separation between the rational degree and the spectral sensitivity of a total Boolean function.

In contrast to total functions, we exhibit large separations between the rational degree and the approximate degree of partial functions. On the one hand, we construct a partial function $f\colon \{0,1\}^n \to \{0,1\}$ with $\widetilde{\deg}(f) = O(1)$ and $\mathrm{rdeg}(f) = \Theta(n)$. On the other, we show that there is a partial function $g\colon \{0,1\}^n \to \{0,1\}$ with $\widetilde{\deg}(g) = \Theta(n)$ and $\mathrm{rdeg}(g) = O(1)$.

Chapter 4 is based on the following paper:

V. Iyer, S. Jain, R. Kothari, M. Kovacs-Deak, V. M. Kumar, L. Schaeffer, D. Wang, and M. Whitmeyer. *On the Rational Degree of Boolean Functions and Applications.* 2025. arXiv:2310.08004

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

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

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 the regular language $\Sigma^*20^*2\Sigma^*$ over the alphabet $\Sigma = \{0, 1, 2\}$; (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 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, $\mathsf{OR}(x) = 1$ if and only if either $\mathsf{OR}(x_{\text{left}}) = 1$ or $\mathsf{OR}(x_{\text{right}}) = 1$, 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

$$C(n) \leq 2\,C(n/2),$$

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 $\mathsf{OR}(x) = \mathsf{OR}(x_{\text{left}}) \vee \mathsf{OR}(x_{\text{right}})$. 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:

$$Q(n) \leq \text{“}\sqrt{2}\text{”}\, Q(n/2). \tag{2.1}$$

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 (2.1) 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

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

where $C^{\text{aux}}(n)$ is the classical query complexity of some 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, $C_Q(n)$, of the analogous recurrence relation,

$$C_Q(n) \leq \sqrt{a}\, C_Q(n/b) + O(C_Q^{\text{aux}}(n)), \tag{2.2}$$

where the quantum query complexity of the auxiliary problem is $C_Q^{\text{aux}}(n)$. We show that this framework easily recovers and improves nontrivial quantum speedups that were previously

[1] This may be familiar to readers who know the quantum adversary method. However, our divide-and-conquer framework generalizes this phenomenon to a scenario that is difficult to capture using adversary machinery alone, as discussed below.

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].

2.1.1 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 $s \in \Sigma^n$ over some set Σ, i.e., query access to a function $f\colon \{1, \dots, n\} \to \Sigma$, with $f(i) = s_i$.

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 is, in turn, a 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 arguably simpler than that of [AGS19].

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.

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 subsequence[2] 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 be[3] $\Omega(n)$, 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.

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 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 be[4] $\Omega(n)$.

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 $\mathsf{ED}(x, y) = 2n - 2\ \mathsf{LCS}(x, y)$ (a proof can be found in, e.g., [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 $\Omega(n)$.

[2] Recall 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 sub*string*, in which the chosen elements must be consecutive.

[3] This follows by reduction from a threshold function such as majority [BBC+01]. Given a string $z \in \{0,1\}^n$, define $x \in \{0, 1, \ldots, n\}^n$ such that $x_i = iz_i$ for all $i \in \{1, \ldots, n\}$. 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.

[4] This 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.

2.1.2 Techniques

To derive the quantum recurrence relation in Equation (2.2), we employ the quantum adversary method as an upper bound on quantum query complexity [Rei11; LMR+11]. The adversary quantity $\mathrm{Adv}(P(n))$ is a real number that can be associated with any query problem $P(n)$ of size n. For convenience, we write $\mathrm{Adv}(n) \coloneqq \mathrm{Adv}(P(n))$ when P is clear from context. It is well-known that $\mathrm{Adv}(n)$ 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; LMR+11]. For example, suppose $P(n)$ is the OR of a copies of $P(n/b)$ together with the solution to some auxiliary problem $P^{\mathrm{aux}}(n)$ that is not in the form of a subproblem. Then the composition theorem implies

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

(For a more precise statement, see Section 2.2.) Taking square roots[5] and using the fact that $\mathrm{Adv}(P^{\mathrm{aux}}(n))$ is bounded by $O(Q^{\mathrm{aux}}(n))$, where $Q^{\mathrm{aux}}(n) \coloneqq Q(P^{\mathrm{aux}}(n))$, we obtain an equation analogous to Equation (2.2),

$$\mathrm{Adv}(n) \leq \sqrt{a}\ \mathrm{Adv}(n/b) + O(Q^{\mathrm{aux}}(n)). \tag{2.3}$$

We then bound $Q^{\mathrm{aux}}(n)$ by constructing an explicit quantum algorithm for the auxiliary work and bounding its quantum query complexity. Finally, we solve Equation (2.3) either directly or by appealing to the Master Theorem [BHS80] to obtain a bound on $\mathrm{Adv}(n)$ and hence a bound on $Q(n)$. Note that this bound is insensitive (up to a multiplicative constant) to the constant implicit in $O(Q^{\mathrm{aux}}(n))$, whereas it is highly sensitive to the constant in front of $\mathrm{Adv}(n/b)$.

We emphasize that the recurrence in Equation (2.3) involves *both* the adversary quantity and

[5]Note 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 $\mathrm{Adv}(n)^2$.

the quantum query complexity. At a high level, this is why the bound on $Q(n)$ resulting from solving Equation (2.3) does not follow directly from adversary techniques alone nor from quantum algorithmic techniques alone. Indeed, when we solve Equation (2.3) as a recurrence relation, we mix adversary and algorithmic techniques at each step of the recursion. In principle, it is possible to use Equation (2.3) 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 (2.3). Bounding $Q(n)$ then reduces to bounding $Q^{\text{aux}}(n)$, which can be easier 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 (2.3) 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 $\Theta(n)$ 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 $\mathsf{EXACT}_{2\text{ of }3}$ (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.

2.1.3 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., [ABI+19; Abb19; GKMV21; ABI+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 apply 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 $\Theta(n)$.

Since the contents of this chapter were first released on arXiv in 2022, our results have attracted a number of follow-up works, including [Wan22; ABB+23; JP24]. In [Wan22], Wang improves the quantum query complexity of Minimal String Rotation by logarithmic factors. [ABB+23; JP24] both investigate the time complexity of quantum divide-and-conquer algorithms. In [ABB+23], Allcock, Bao, Belovs, Lee, and Santha improve the quantum query complexity of k-IS by logarithmic factors and give a time-efficient algorithm. They also give time-efficient quantum divide and conquer algorithms for a number of other problems. In [JP24], Jeffery and Pass formalize the notion of a multi-dimensional quantum walk and show how it can be used to implement certain quantum divide and conquer algorithms. In particular, they give a time-efficient implementation of one of our quantum divide-and-conquer strategies (see Strategy 1 below).

Organization. We first introduce background material and describe how to use the quantum divide-and-conquer framework for certain scenarios in Section 2.2. Then we apply the framework to four string problems in Section 2.3, obtaining near-optimal quantum query complexities for each. In particular, we consider recognizing regular languages in Section 2.3.1, String Rotation and String Suffix in Section 2.3.2, k-Increasing Subsequence in Section 2.3.3, and k-Common Subsequence in Section 2.3.4. Finally, we conclude by presenting some open problems in Section 2.4.

2.2 Framework

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

Let $\mathbb{N}$ denote the set of positive integers. We reserve $m, n \in \mathbb{N}$ for positive integers and Σ for a finite set. For $m \in \mathbb{N}$, we write $[m] := \{1, \ldots, m\}$. For any two matrices A and B of the same size, $A \circ B$ denotes their entrywise product, also known as their Hadamard product. For any matrix A, let $(A)_{ij}$ denote the element in the ith row and jth column. For a function $f\colon\ D \to E$, 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 ith element of x and $x[i\,..\,j]$ for the substring of x that ranges from its ith to jth elements inclusive (if $j < i$, then this is the empty string). We write $x(i\,..\,j] := x[i+1\,..\,j]$ and $x[i\,..\,j) := x[i\,..\,j-1]$. When Σ is equipped with a total order, and $x \in \Sigma^m$ and $y \in \Sigma^n$, we write inequalities between x and y (e.g., $x \leq y$) to refer to inequalities with respect to the lexicographic order.

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, $\mathrm{Adv}(\cdot)$.

Definition 2 (Adversary quantity)**.** Let D, E, and Σ be finite sets and $n \in \mathbb{N}$ with $D \subseteq \Sigma^n$. Let $f\colon\ D \to E$ be a function with Gram matrix F and $\Delta = \{\Delta_1, \ldots, \Delta_n\}$ with $(\Delta_j)_{x,y \in D} = 1 - \delta_{x_j, y_j}$. Then the *adversary quantity* for f is

$$\mathrm{Adv}(f) := \max\Big\{\|\Gamma\| \,:\, \forall j \in [n],\ \|\Gamma \circ \Delta_j\| \leq 1\Big\},$$

where the maximization is over $|D| \times |D|$ real, symmetric matrices Γ satisfying $\Gamma \circ F = 0$.

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

Theorem 3 ([HLŠ07; LMR+11]). *Let $f\colon D \to E$ be a function. Then*

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

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) $h(x) = f_1(x) \vee f_2(x)$;

(ii) $h(x) = f_1(x) \wedge f_2(x)$; or

(iii) $h(x) = g_{f(x)}(x)$ is SWITCH-CASE on functions $f\colon D \to \Lambda$ and $\{g_s : D \to \{0,1\} \mid s \in \Lambda\}$.

Lemma 4 (Adversary composition for OR and AND). *Let $a, b \in \mathbb{N}$ and let Σ be a finite set. Let $f_1 : \Sigma^a \to \{0,1\}$ and $f_2 : \Sigma^b \to \{0,1\}$. Suppose that $g^\wedge, g^\vee : \Sigma^a \times \Sigma^b \to \{0,1\}$ are 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*

$$\mathrm{Adv}(g^\wedge)^2 = \mathrm{Adv}(g^\vee)^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2)^2.$$

Moreover, suppose $a = b$ and $h^\wedge : \Sigma^a \to \{0,1\}$ satisfies $h^\wedge(x) = f_1(x) \wedge f_2(x)$ for all $x \in \Sigma^a$. Let f_2' denote the restriction $f_2' = f_2|_{f_1^{-1}(1)}$. Then $\mathrm{Adv}(h^\wedge)^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2')^2$.

Lemma 4 follows from [LMRŠ10; Rei11]. For completeness, we provide a proof in Section A.1. Note that the first part of the lemma still holds if f_1 and f_2 share variables.[6] The intuitive reason for the "moreover" part 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. See Section A.1 for a proof of Lemma 5.

Lemma 5 (Adversary composition for SWITCH-CASE). *Let $a \in \mathbb{N}$ and let Σ, Λ be finite sets. Let $f\colon \Sigma^a \to \Lambda$. For $s \in \Lambda$, let $g_s : \Sigma^a \to \{0,1\}$. Define $h : \Sigma^a \to \{0,1\}$ by $h(x) = g_{f(x)}(x)$. Then*

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

[6] For example, if $h : \Sigma^a \to \{0,1\}$ with $h(x) := g(x,x) = f_1(x) \wedge f_2(x)$, then $\mathrm{Adv}(h)^2 \leq \mathrm{Adv}(g)^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2)^2$.

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 solving them together with some auxiliary problem P^{aux} that is not in the form of a subproblem. 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 the recurrence

$$C(n) \leq a\,C(n/b) + C^{\text{aux}}(n). \tag{2.4}$$

The situation is more subtle in the quantum case. Equation (2.4) bounds the cost of solving the size-n problem by summing the costs of 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 $f\colon \Sigma^n \to \{0,1\}$, 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 subfunctions 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 an AND-OR formula involving an auxiliary function $f^{\text{aux}} : \Sigma^n \to \{0,1\}$ and subfunctions $\{f_i : \Sigma^{n/b} \to \{0,1\} \mid i \in [a]\}$, where $a, b \in \mathbb{N}$.

Strategy 2. f is computed sequentially as follows: (1) Compute an auxiliary function $f^{\text{aux}} : \Sigma^n \to \Lambda$, where Λ is a finite set, to obtain some $s \in \Lambda$; (2) s labels a set of subfunctions $\{f_i^{(s)} : \Sigma^{n/b} \to \{0,1\} \mid i \in [a]\}$, where $a, b \in \mathbb{N}$, as well as a Boolean function $g^{(s)} : \{0,1\}^a \to \{0,1\}$, such that $f(x)$ is computed by the function $g^{(s)} \circ \left(f_1^{(s)}, \ldots, f_a^{(s)}\right) : (\Sigma^{n/b})^a \to \{0,1\}$ via

$$g^{(s)} \circ \left(f_1^{(s)}, \ldots, f_a^{(s)}\right)(x^{(1)}, \ldots, x^{(a)}) := g^{(s)}(f_1^{(s)}(x^{(1)}), \ldots, f_a^{(s)}(x^{(a)})),$$

where $x^{(1)}, \ldots, x^{(a)}$ are a many (not necessarily disjoint) blocks of x of size n/b.

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

Corollary 6. *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*

$$\mathrm{Adv}(f)^2 \leq \sum_{i=1}^{a} \mathrm{Adv}(f_i)^2 + O(Q(f^{\mathrm{aux}}))^2 \qquad \text{for Strategy 1,} \tag{2.5}$$

$$\mathrm{Adv}(f) \leq O\big(Q(f^{\mathrm{aux}})\big) + \max_{s\in\Lambda} \mathrm{Adv}\big(g^{(s)} \circ \big(f_1^{(s)}, \ldots, f_a^{(s)}\big)\big) \qquad \text{for Strategy 2.} \tag{2.6}$$

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 directly apply Lemma 4 to obtain

$$\begin{aligned}\mathrm{Adv}(f)^2 &\leq \mathrm{Adv}(f_1)^2 + \cdots + \mathrm{Adv}(f_a)^2 + \mathrm{Adv}(f^{\mathrm{aux}})^2 \\ &\leq \mathrm{Adv}(f_1)^2 + \cdots + \mathrm{Adv}(f_a)^2 + O(Q(f^{\mathrm{aux}}))^2 \quad \text{(by Thm. 3).}\end{aligned}$$

For Strategy 2, $f^{\mathrm{aux}} : \Sigma^n \to \Lambda$ is first computed to give an $s \in \Lambda$. Then, f is computed as $g^{(s)} \circ \big(f_1^{(s)} \ldots, f_a^{(s)}\big)$. Therefore, $f(x)$ can be expressed using SWITCH-CASE as

$$f(x) = h_{f^{\mathrm{aux}}(x)}(x), \quad \text{where } h_s \coloneqq \big(g^{(s)} \circ \big(f_1^{(s)}, \ldots, f_a^{(s)}\big)\big) \text{ for all } s \in \Lambda.$$

Therefore, we can directly apply Lemma 5 to obtain

$$\begin{aligned}\mathrm{Adv}(f) &\leq O(\mathrm{Adv}(f^{\mathrm{aux}})) + \max_{s\in\Lambda} \mathrm{Adv}\big(g^{(s)} \circ \big(f_1^{(s)}, \ldots, f_a^{(s)}\big)\big) \\ &\leq O(Q(f^{\mathrm{aux}})) + \max_{s\in\Lambda} \mathrm{Adv}\big(g^{(s)} \circ \big(f_1^{(s)}, \ldots, f_a^{(s)}\big)\big) \quad \text{(by Thm. 3).}\end{aligned}$$

The corollary follows. □

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 $\mathrm{Adv}(f^{\mathrm{aux}})$ to the quantum query complexity $Q(f^{\mathrm{aux}})$ 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 7 (Quantum query complexity bounds)**.** *Let $m, n \in \mathbb{N}$ with $m \leq n$.*

- Unstructured search. *Given $x \in \{0,1\}^n$, finding an $i \in [n]$ such that $x_i = 1$ has quantum query complexity $O(\sqrt{n})$ [Gro96].*
- Minimum (maximum) finding. *Given $x \in \Sigma^n$ for an ordered set Σ, finding an $i \in [n]$ such that x_i is the minimum (maximum) symbol in x has quantum query complexity $O(\sqrt{n})$ [DH96].*
- String matching. *Given $x \in \Sigma^n$ and $y \in \Sigma^m$, determining if y is a substring of x has quantum query complexity $\tilde{O}(\sqrt{m} + \sqrt{n})$ [RV03].*
- String comparison. *Given $x, y \in \Sigma^n$ for an ordered set Σ, determining if $x < y$ has quantum query complexity $O(\sqrt{n})$. This is because the problem reduces to minimum finding.*
- Bipartite element distinctness. *Given $x \in \Sigma^n$ and $y \in \Sigma^m$, finding $i \in [n]$ and $j \in [m]$ such that $x_i = y_j$ has quantum query complexity $O(n^{2/3})$ [BDH+05; 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 $\mathrm{Adv}(f)$, and thereby $Q(f)$. This is just like in classical divide and conquer.

We state the Master Theorem below for convenience.

Theorem 8 (Master Theorem [BHS80])**.** *Let $A : \mathbb{N} \to \mathbb{R}$, $a, b, c > 0$, and $p \geq 0$. Suppose A satisfies the recurrence*

$$A(n) = a\, A(n/b) + O(n^c \log^p n).$$

Then

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

2.3 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 $b > 1$. Without loss of generality, we assume that $n = b^r$ for some $r \in \mathbb{N}$ so that $n/b^k \in \mathbb{N}$ for all $k \in [r]$.[7]

2.3.1 Recognizing Regular Languages

We first consider the problem of recognizing the regular language generated by the regular expression $\Sigma^* 20^* 2\Sigma^*$, where $\Sigma = \{0, 1, 2\}$. This problem is a key step in establishing the trichotomy theorem of Aaronson, Grier and Schaeffer for the quantum query complexity of regular languages [AGS19, Theorem 18].

Problem (Recognizing $\Sigma^* 20^* 2\Sigma^*$)**.** *Given query access to $x \in \Sigma^n$, 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 9. *The quantum query complexity of recognizing $\Sigma^* 20^* 2\Sigma^*$ is $O(\sqrt{n \log(n)})$.*

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

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

[7] When $n \neq b^r$, we can divide into subproblems of sizes $\lceil n/b \rceil$ or $\lfloor n/b \rfloor$ without affecting the validity of our arguments.

in the left half and partly in the right half of the string. Therefore, we have

$$f_n(x) = f_{n/2}(x_{\text{left}}) \vee f_{n/2}(x_{\text{right}}) \vee g_n(x), \tag{2.7}$$

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

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

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

1. Grover search for the maximal index $i \le n/2$ 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 if $x[i+1\,..\,j-1]$ equals 0^{j-i-1} by Grover search. Output 1 if yes and 0 if no.

Therefore, writing $a(n) \coloneqq \mathrm{Adv}(f_n)$, observing that Equation (2.7) follows Strategy 1 and using Corollary 6 gives $a(n)^2 \le 2a(n/2)^2 + O(n)$, which solves to $a(n)^2 = O(n\log(n))$ by the Master Theorem (Theorem 8). Hence, by Theorem 3 $Q(f_n) = O(a(n)) = O(\sqrt{n\log(n)})$. □

2.3.2 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 $n \in \mathbb{N}$. Given $i \in [n]$ and query access to a string $x \in \Sigma^n$, we define the following problems.*

(1) Minimal String Rotation. Decide if $x[i\,..\,n]x[1\,..\,i-1] \le x[j\,..\,n]x[1\,..\,j-1]$ for all $j \in [n]$.

(2) Minimal Suffix. Decide if $x[i\,..\,n] < x[j\,..\,n]$ 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 with l set to $n/2$. (See the proof of Theorem 12 for more details.)

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 $f_n : \Sigma^n \times \Sigma^{n/2} \to \{0,1\}$ 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

$$f_n(x,y) = \Big(f_{n/2}(x[1\,..\,{}^{n}/_2], y[1\,..\,{}^{n}/_4]) \wedge f_{n/2}(x[{}^{n}/_4 + 1\,..\,{}^{3n}/_4], y[1\,..\,{}^{n}/_4])\Big) \wedge g_n(x,y),$$

where $g_n : S \subseteq \Sigma^n \times \Sigma^{n/2} \to \{0,1\}$ 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

$$g_n(x,y) = \begin{cases} 1 & \text{if all length-}(n/2)\text{ substrings in } x \text{ starting with } y[1\,..\,{}^{n}/_4] \\ & \text{are lexicographically at least } y \text{ (or do not exist),} \\ 0 & \text{otherwise.} \end{cases}$$

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

(1) Use the quantum string matching algorithm (see Theorem 7) 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, 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 7) to decide whether all the length-$(n/2)$ substrings in x starting at a position in $\{u_1, v_1, u_2, v_2\}$ are lexicographically at least y. If yes, then output 1; otherwise, output 0. This takes $\tilde{O}(\sqrt{n})$ queries.

Clearly, $\mathscr{A}$ uses $\tilde{O}(\sqrt{n})$ queries. The correctness of $\mathscr{A}$ follows from the following Claim 11 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 $\mathscr{A}$ follows from Claim 11 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 $\mathscr{A}$ follows from the last sentence in the first stage of $\mathscr{A}$.

Claim 11 (Exclusion rule)**.** *Let $s \in \Sigma^n$ and λ be an integer with $n/2 \le \lambda \le n$, let*

$$I \coloneqq \operatorname*{arg\,min}_{1 \le i \le n-\lambda+1} s[i\,..\,i+l)$$

denote the set of starting positions[8] of the minimal length-λ substring of s. For integers $a \ge 1$, $k \ge 1$ such that $a+k \le n-\lambda+1$, let t denote a length-k substring, let $J \coloneqq \{i \in \{a, \dots, a+k\} \mid s[i\,..\,i+k) = t\}$ 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 elements form an arithmetic progression. But [AJ23, Lemma 2.3] shows that the elements of J indeed form an arithmetic progression. □

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

Now, the "moreover" part of Lemma 4 gives $a(n)^2 \leq 2a(n/2)^2 + \mathrm{Adv}(g_n)^2$, where we let $a(n) \coloneqq \mathrm{Adv}(f_n)$. Since $Q(g_n) = \tilde{O}(\sqrt{n})$, Theorem 3 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 8). Hence, $Q(f_n) = O(a(n)) = \tilde{O}(\sqrt{n})$ by Theorem 3. □

Remark. The analysis in [AJ23, Theorem 4.1] for Minimal Length-l Substring yields a bound of $O(n^{1/2+o(1)})$ for its quantum query and time complexity, which we now know is loose in terms of query complexity. That analysis is more involved because it splits the problem into a nonconstant 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 constant and logarithmic factors hidden in the $\tilde{O}(\sqrt{m})$ coefficient, requires m to be nonconstant in order to obtain $Q(n) = O(n^{1/2+o(1)})$. The hidden factors in $\tilde{O}(\sqrt{m})$ arise from doing Grover search with error probability $1/\mathrm{poly}(m)$, which our framework avoids. In general, it can be difficult to optimally solve a problem by splitting it into a nonconstant number of parts and applying quantum algorithmic techniques (e.g., consider an AND-OR formula) and such an approach could yield upper bounds that are loose by an $n^{\Omega(1)}$ 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 12 below follows from Lemma 10 and the chain of reductions described in [AJ23, Propositions 4.2–4.5 and Theorem 4.6]. We sketch a proof for completeness.

Theorem 12. *The quantum query complexities of Minimal String Rotation and Minimal Suffix are* $\tilde{O}(\sqrt{n})$.

Proof sketch. Let f_{2n} be defined as in the proof of Lemma 10. The theorem follows from two observations:

(1) Minimal String Rotation can be computed by $f_{2n}(xx, x[i\,..\,n]x[1\,..\,i-1])$.

(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. □

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.

2.3.3 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 $i_1 < i_2 < \cdots < i_k$ such that $x[i_1] < x[i_2] < \cdots < x[i_k]$.*

To solve k-IS, we solve a variant 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 $i_1 < i_2 < \cdots < i_k$ such that $x[i_1] < x[i_2] < \cdots < x[i_k]$ and $x[i_l] \neq *$ for all $l \in [k]$.*

We now prove the main theorem of this subsection.

Theorem 13. *The quantum query complexities of k-IS* is $O(\sqrt{n}\log^{3(k-1)/2}(n))$. Hence so is the complexity of k-IS.*

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 $\mathsf{LIS}_{k,n} : \Sigma^n \to \{0,1\}$ defined by $\mathsf{LIS}_{k,n}(x) = 1$ iff x contains a k-IS*.

We begin with a few definitions. Let $a_k(n) := \mathrm{Adv}(\mathsf{LIS}_{k,n})$. Define $\mathsf{LIS}_{(i,j),n} : \Sigma^n \to \{0,1\}$ by $\mathsf{LIS}_{(i,j),n}(x) = 1$ if and only if x contains an $(i+j)$-IS* consisting of an i-IS*, $u_1 < u_2 < \cdots < u_i$, and a j-IS*, $v_1 < v_2 < \cdots < v_j$, such that $u_i \leq n/2 < v_1$. For $j \in \mathbb{N}$, define $\mathsf{min\text{-}last}_{j,n} : \Sigma^n \to \Sigma$ by $\mathsf{min\text{-}last}_{j,n}(x) = \min_l x[l]$, where the minimization is over those $l \in [n]$ that are the last index of

some j-IS* in x. Similarly, define $\mathsf{max\text{-}first}_{j,n} : \Sigma^n \to \Sigma$ by $\mathsf{max\text{-}first}_{j,n}(x) = \max_l x[l]$, 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

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

Therefore, by Corollary 6 for Strategy 1, we obtain the quantum divide and conquer recurrence

$$a_k(n)^2 \leq 2a_k(n/2)^2 + \sum_{i=1}^{k-1} O(Q(\mathsf{LIS}_{(i,k-i),n})^2). \tag{2.8}$$

Now,

$$Q(\mathsf{LIS}_{(i,k-i),n}) \leq Q(\mathsf{min\text{-}last}_{i,n/2}) + Q(\mathsf{max\text{-}first}_{k-i,n/2}), \tag{2.9}$$

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

Moreover, for any $j \in \mathbb{N}$, $\mathsf{min\text{-}last}_{j,n}$ and $\mathsf{max\text{-}first}_{j,n}$ can be computed by a "randomized search" that uses $O(\log(n))$ computations of $\mathsf{LIS}_{j,n}$ and Grover search, which is similar to the approach taken for quantum minimum finding [DH96]—see Section A.2 for a proof. (The proof in Section A.2 also explains why k-IS* is more susceptible to recursion.) Therefore, for all $j, n \in \mathbb{N}$, we can use Theorem 3 to obtain

$$Q(\mathsf{min\text{-}last}_{j,n}), Q(\mathsf{max\text{-}first}_{j,n}) \leq O((a_j(n) + \sqrt{n}) \log(n)). \tag{2.10}$$

Substituting the combination of Equation (2.9) and Equation (2.10) into Equation (2.8) shows that, for $k \geq 2$, we have

$$a_k(n)^2 \leq 2a_k(n/2)^2 + O(a_{k-1}(n)^2 \log^2(n)), \tag{2.11}$$

where we used the fact that k is constant and $\Omega(\sqrt{n}) \leq a_i(n) \leq O(a_{i+1}(n))$ for all $i \geq 1$, which follows by applying Theorem 3 to the observations that (i) $\mathsf{LIS}_{i,n}(x) = \mathsf{LIS}_{i+1,n+1}(0x)$ for all x, where $0 \neq *$ is some element smaller than all elements in Σ, and (ii) computing $\mathsf{LIS}_{1,n}$ is equivalent to searching for a "$*$".

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 3 to the observation that computing $\mathsf{LIS}_{1,n}$ is equivalent to searching for a "$*$". For $k > 1$, Equation (2.11) and the inductive hypothesis for the $(k-1)$-th case give

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

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

2.3.4 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 $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ with $x[i_l] = y[j_l]$ for all $l \in [k]$.*

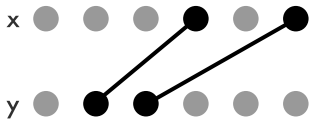
The main theorem of this subsection is the following.

Theorem 14. *The quantum query complexity of k-CS is $O(n^{2/3}\log^{k-1}(n))$.*

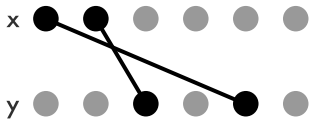
Define $\mathsf{LCS}_{k,n} : \Sigma^n \times \Sigma^n \to \{0,1\}$ by $\mathsf{LCS}_{k,n}(x,y) = 1$ iff $x, y \in \Sigma^n$ share a k-CS. When $k = 1$ this problem is equivalent to the (bipartite) Element Distinctness problem [AS04; Amb07]. Aaronson and Shi showed that the quantum query complexity of 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 $x[i] = y[j]$ (see Figure 2.1 for examples).



(a) A diagram showing two collisions, $(4, 2)$ and $(6, 3)$, of some 6-element strings x and y. Since these collisions are compatible, together they form a 2-CS of x and y.



(b) A diagram showing two collisions, $(1, 5)$ and $(2, 3)$, of some 6-element strings x and y. As these collisions are incompatible, they cannot both be part of a k-CS for any k.

Figure 2.1: Diagrams illustrating collisions of 6-element strings.

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

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

Definition 15 (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 $\sigma_n : \Sigma^n \times \Sigma^n \to \mathcal{S}$ 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 16. *The quantum query complexity of σ_n satisfies $Q(\sigma_n) = O(n^{2/3})$.*

Definition 17 (Compatible, relevant, and critical subproblems)**.**

(i) 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 $j_1 > j_2$).

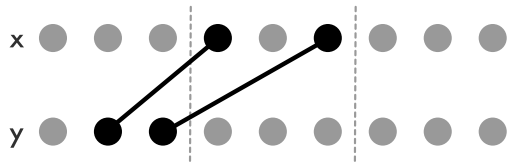
(ii) Given a signature $\mathbf{s} \in \mathcal{S}$, we say that the (i, j) subproblem is $\mathbf{s}$-*relevant* if $\mathbf{s}_{i,j} = 1$.

(iii) 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.

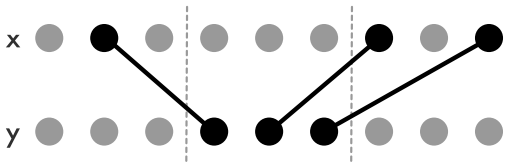
Observe that if subproblems (i_1, j_1) and (i_2, j_2) are compatible then, for any $x, y \in \Sigma^n$, the union of any k_1-CS of the (i_1, j_1)-subproblem of (x, y) and any k_2-CS of the (i_2, j_2)-subproblem of (x, y) is a $(k_1 + k_2)$-CS of (x, y). A key intuition of our approach is the following. 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 $\mathsf{LCS}_{k,n}(x, y) = 1$, 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 18 (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*.



(a) A diagram illustrating a 2-CS of some 9-element strings x and y. When the splitting factor is 3, this 2-CS is a simple witness of $\mathsf{LCS}_{2,9}(x, y) = 1$, since it is also a 2-CS of the $(2, 1)$-subproblem of (x, y).



(b) A diagram illustrating a 3-CS of some 9-element strings x and y. When the splitting factor is 3, this 3-CS is a composite witness of $\mathsf{LCS}_{3,9}(x, y) = 1$.

Figure 2.2: Diagrams illustrating simple and composite witnesses for different choices of k with respect to a splitting factor of $m = 3$.

We define $\mathsf{LCS}_{k,n}^{\text{composite}} : \Sigma^n \times \Sigma^n \to \{0,1\}$ by $\mathsf{LCS}_{k,n}^{\text{composite}}(x,y) = 1$ iff x and y share a composite k-CS (see Figure 2.2). Given a signature $\mathbf{s} \in \mathcal{S}$, define $h_{k,n}^{(\mathbf{s})} : \Sigma^n \times \Sigma^n \to \{0,1\}$ 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 19. *For all $x, y \in \Sigma^n$, we have*

$$\mathsf{LCS}_{k,n}(x,y) = \mathsf{LCS}_{k,n}^{\text{composite}}(x,y) \vee h_{k,n}^{(\sigma_n(x,y))}(x,y). \tag{2.12}$$

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 (2.12) evaluates to true. Let $\mathbf{s} := \sigma_n(x,y)$ be the signature of (x,y). Then, the right-hand side of Equation (2.12) 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. □

Let $a_k(n)$ denote the adversary quantity of $\mathsf{LCS}_{k,n}$. As $Q(\mathsf{LCS}_{k,n}) = \Theta(a_k(n))$ (Theorem 3), we can prove Theorem 14 by upper bounding the adversary quantity $a_k(n)$ instead of the quantum query complexity $Q(\mathsf{LCS}_{k,n})$. 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 (2.12).

Lemma 20. *For any signature $\mathbf{s} \in \mathcal{S}$, the adversary quantity of the function $h_{k,n}^{(\mathbf{s})}$ satisfies*

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

Lemma 21. *The quantum query complexity of* $\mathsf{LCS}_{k,n}^{\text{composite}}$ *satisfies*

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

We defer the proofs of Lemmas 20 and 21 for now, and prove the following lemma assuming these results.

Lemma 22. *For* $k \geq 2$*, the adversary quantity* $a_k(n)$ *of* $\mathsf{LCS}_{k,n}$ *satisfies the recurrence*

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

Proof. Recall that by Proposition 19,

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

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 2.2. Therefore, by applying Corollary 6,

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

By Lemmas 16, 20, and 21,

$$\begin{aligned} a_k(n) &\leq O(a_{k-1}(n) \log n) + O(n^{2/3}) + \sqrt{2m-1}\, a_k(n/m) \\ &\leq \sqrt{2m-1}\, a_k(n/m) + O(a_{k-1}(n) \log n), \end{aligned}$$

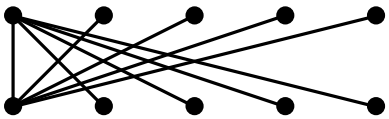
where the last line follows since $a_1(n) \in \Theta(n^{2/3})$, and $a_{k-1}(n) = \Omega(n^{2/3})$ since Element Distinctness reduces to $(k-1)$-CS for $k \geq 2$ by prefixing $k-2$ common elements to the inputs for $(k-1)$-CS. □

Proof of Theorem 14. We make use of the above recurrence, and proceed by induction on $k \geq 1$. In

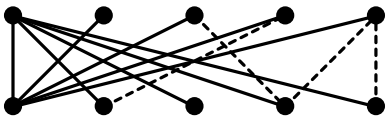
the base case ($k = 1$), the theorem holds since 1-CS is equivalent to bipartite Element Distinctness, which has quantum query complexity $O(n^{2/3})$ [Amb07]. Assume that $k \geq 2$. By Lemma 22 we have the following recurrence for the adversary quantity of $\mathsf{LCS}_{k,n}$:

$$\begin{aligned} a_k(n) &\leq \sqrt{2m-1}\, a_k(n/m) + O(a_{k-1}(n) \log n) \\ &\leq \sqrt{2m-1}\, a_k(n/m) + O(n^{2/3} \log^{k-1}(n)), \end{aligned}$$

where the second inequality follows by our inductive hypothesis, $a_{k-1}(n) \leq O(n^{2/3} \log^{k-2}(n))$. The result follows from the Master Theorem (Theorem 8) provided that $\log_m(\sqrt{2m-1}) < 2/3$. We achieve this by choosing $m = 7$. □



(a) For a splitting factor of $m = 5$, the maximum number of $\mathbf{s}$-critical subproblems for any signature $\mathbf{s}$ is 9. This diagram illustrates one signature $\mathbf{s}$ that attains this bound. Each edge (i, j) corresponds to an $\mathbf{s}$-critical subproblem.



(b) For each of the dotted edges (i, j), the above graph already contains a solid edge (p, q) of the same slope, i.e., with $i-j = p-q$. In fact, adding any more edges to the graph on the left will only decrease the number of $\mathbf{s}'$-critical subproblems for the resulting signature $\mathbf{s}'$.

Figure 2.3: Diagrams illustrating an example of a signature $\mathbf{s}$ for which the number of $\mathbf{s}$-critical subproblems is maximal. Here, the splitting factor is $m = 5$ and we represent a signature $\mathbf{s} \in \mathcal{S} = \{0,1\}^{m^2}$ by the subgraph of $K_{m,m}$ that has edge (i, j) iff $\mathbf{s}_{i,j} = 1$.

We now prove Lemmas 20 and 21.

Proof of Lemma 20. Fix an arbitrary signature $\mathbf{s} \in \mathcal{S}$. Recall that the function $h^{(\mathbf{s})}_{k,n}(x, y)$ 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 $R \coloneqq \{(i_l, j_l) : l \in [r]\}$ denote the set of $\mathbf{s}$-critical subproblems. Then

$$h^{(\mathbf{s})}_{k,n}(x, y) = \bigvee_{(i,j)\in R} \mathsf{LCS}_{k,n/m}(x^{(i)}, y^{(j)}).$$

By the adversary composition lemma for OR (Lemma 4),

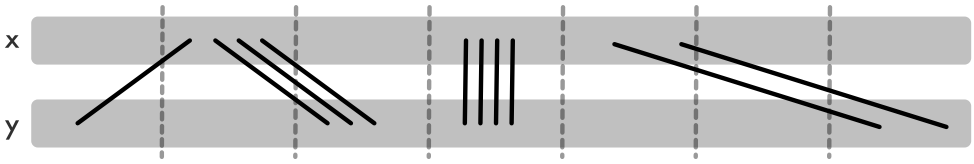
$$\mathrm{Adv}(h_{k,n}^{(\mathbf{s})}) = \sqrt{r}\, a_k(n/m).$$

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 $\{0, \pm 1, \ldots, \pm(m-1)\}$. 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 (see Figure 2.3). Therefore, for each slope, there can be at most one subproblem in R since the subproblems in R are $\mathbf{s}$-critical. Therefore, $r \leq 2m - 1$. □

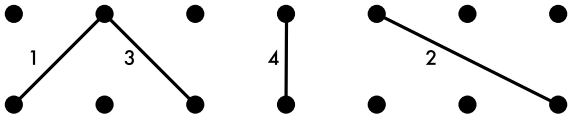
Proof of Lemma 21. Label the vertices in the two parts of the complete bipartite graph $K_{m,m}$ by u_i and v_i, respectively, for $i \in [m]$.

To any k-common subsequence S of some x and y, we assign a subgraph G_S of $K_{m,m}$ that is obtained by removing all edges (i, j) of $K_{m,m}$ for which S does not contain a collision of $x^{(i)}$ and $y^{(j)}$. Then the graph G_S has at least 2 edges if and only if S is a composite k-CS; otherwise G_S has a single edge only. We assign positive integer weights to the edges of G_S by letting the weight of the edge (u_i, v_j) be the number of collisions between the substrings $x^{(i)}$ and $y^{(j)}$ that are contained in S. Therefore, the total weight of G_S is k for any k-CS considered.

Placing the vertices of G_S on a square grid (say, u_i on $(i, 1)$ and v_j on $(j, 0)$ of the Euclidean plane) and drawing the edges as the line segments between the corresponding vertices, we see that by the definition of a k-CS, the edges of G_S meet only at vertices in this drawing (i.e., we have a planar embedding, see Figure 2.4b). This means that we can make the edge set of G_S a totally ordered set, by letting $(u_i, v_j) \preceq (u_{i'}, v_{j'})$ iff $i \leq i'$ and $j \leq j'$. Therefore, a unique minimal (i.e., leftmost) edge exists. Let (u_{i^*}, v_{j^*}) be this edge. We claim that at least one of u_{i^*} and v_{j^*} has degree 1. To see this, we can assume without loss of generality that $i^* \leq j^*$, as an analogous argument works for the other case. Suppose now that u_{i^*} has a neighbor other than v_{j^*}. Denoting this other neighbor by $v_{j'}$, the minimality of the edge (u_{i^*}, v_{j^*}) implies $j^* < j'$. Consider the vertex v_{j^*}, and



(a) A schematic diagram illustrating a 10-CS of some strings x and y. The dotted lines break up the strings x and y into $m = 7$ substrings of equal size.



(b) A diagram showing the graph G_S associated with the 10-CS seen in the above diagram, assuming a splitting factor of $m = 7$.

Figure 2.4: Diagrams illustrating a 10-CS S of some pair of strings, as well as the associated subgraph G_S of the complete bipartite graph $K_{m,m}$. Here, the splitting factor is $m = 7$.

suppose that v_{j^*} has another neighbor $u_{i'}$ for some i' with $i^* < i'$. Observe that in this case the edges $(u_{i^*}, v_{j'})$ and $(u_{i'}, v_{j^*})$ would be crossing in our drawing, contradicting our observation that there exists a planar embedding of G_S (see Figure 2.4b for an illustration).

To summarize, to any k-CS S we associate an edge-weighted undirected subgraph G_S of $K_{m,m}$ that satisfies the following properties:

(i) The weight of the edge (u_i, v_j) in G_S is the number of collisions of $x^{(i)}$ and $y^{(j)}$ that S contains.

(ii) The graph G_S has at least 2 edges if and only if S is a composite k-CS.

(iii) There is a unique leftmost edge (u_{i^*}, v_{j^*}) of G_S.

(iv) The leftmost edge of G_S is such that at least one of the vertices it is incident on has degree 1.

Let $f_{k,n}^{(i,j)}(x, y)$ denote the Boolean function that indicates if x and y have a composite k-CS for which (i, j) is the leftmost edge of the associated graph G_S. By Property (iii),

$$\mathsf{LCS}_{k,n}^{\text{composite}}(x, y) = \bigvee_{i,j \in [m]} f_{k,n}^{(i,j)}(x, y).$$

Therefore, an algorithm that decides $f_{k,n}^{(i,j)}(x, y)$ for every $i, j \in [m]$ can decide $\mathsf{LCS}_{k,n}^{\text{composite}}(x, y)$. To conclude, we give a quantum algorithm that decides $f_{k,n}^{(i,j)}$ using $O(a_{k-1}(n) \log n)$ queries, for

any $i, j \in [m]$. As there are only $O(1)$ choices of $i, j \in [m]$ to consider (since m is constant), an algorithm that iterates over each of these cases has the same asymptotic complexity.

Let S be a witness of $f_{k,n}^{(i,j)}(x, y) = 1$ for some $i, j \in [m]$ and consider the associated graph G_S. By Property (iv) above, at least one of the vertices u_i and v_j has degree 1 in G_S. Suppose that v_j has degree 1 (the other case can be treated analogously). Then by the properties of G_S we observed earlier, we see that S has the following structure (see Figure 2.4a for an example):

(1) By Property (ii), G_S has at least 2 edges. Therefore by Property (i), the weight of (u_i, v_j) is some $k_1 < k$. Therefore, S contains $k_1 \in [k-1]$ collisions of $x^{(i)}[1\,..\,p]$ and $y^{(j)}$, for some index $p \in [n/m]$.

(2) By Property (iv), and our assumption that v_j has degree 1, S does not contain any collisions between $y^{(j)}$ and any substring $x^{(i')}$ other than $x^{(i)}$. Therefore, none of the remaining $k - k_1$ collisions of S involve $y^{(j)}$. In other words, the remaining $k - k_1$ collisions of S are collisions of $x((\frac{i-1}{m}n + p)\,..\,n]$ and $y(\frac{j}{m}n\,..\,n]$.

The following quantum query algorithm decides if a witness S of the above structure exists, and returns false otherwise, using $O(a_{k-1}(n) \log n)$ queries:

(1) By combining a binary search over the indices of $x^{(i)}$ with a query-optimal quantum algorithm for k_1-CS, identify the minimal $p \in [n/m]$ for which $x^{(i)}[1\,..\,p]$ and $y^{(j)}$ have a k_1-CS. If no such p has been found, return false. This can be done using $O(a_{k_1}(n) \log n)$ queries.[9]

(2) Using a query-optimal algorithm for $(k - k_1)$-CS, decide if $x((\frac{i-1}{m}n + p)\,..\,n]$ and $y(\frac{j}{m}n\,..\,n]$ have a $(k - k_1)$-CS, and return the output. This can be done using $O(a_{k-k_1}(n))$ queries.

As $1 \le k_1 \le k - 1$, and $a_l(\cdot)$ is a nondecreasing[10] function of l, the total query complexity of the

[9] Since the quantum algorithm for 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.

[10] Recall that $(l-1)$-CS reduces to l-CS by prefixing a common element to the input strings of l-CS.

above algorithm is

$$O\big(a_{k_1}(n)\log n + a_{k-k_1}(n)\big) \leq O(a_{k-1}(n)\log n).$$

To decide $f_{k,n}^{(i,j)}$, we run the above algorithm for every possible choice of $k_1 \in [k-1]$. Since we arbitrarily assumed it was v_j that had degree 1, we run an analogous algorithm assuming that u_i has degree 1. Overall this does not affect the asymptotic query complexity of deciding $f_{k,n}^{(i,j)}$, which is

$$Q(f_{k,n}^{(i,j)}) \leq 2(k-1)\cdot O(a_{k-1}(n)\log n) \leq O(a_{k-1}(n)\log n).$$

The lemma follows. □

2.4 Open Problems

We conclude by describing some open problems related to the topics explored in this chapter.

- Can we find applications of quantum divide and conquer using generalizations of the two strategies presented in Section 2.2 (i.e., using combining functions other than AND-OR formulas and SWITCH-CASE, such as those discussed at the end of Section 2.1.2)?
- Can we obtain super-quadratic speedups using quantum divide and conquer?

Chapter 3

Translation-Invariant Algorithms for Ordered Search

Chapter summary. Ordered search is the task of finding an item in an ordered list using comparison queries. The best exact classical algorithm for this fundamental problem uses $\lceil \log_2 n \rceil$ queries for a list of length n. Quantum computers can achieve a constant-factor speedup, but the best possible coefficient of $\log_2 n$ for exact quantum algorithms is only known to lie between $(\ln 2)/\pi \approx 0.221$ and $4/\log_2 605 \approx 0.433$. We consider a special class of translation-invariant algorithms with no workspace, introduced by Farhi, Goldstone, Gutmann, and Sipser, that has been used to find the best known upper bounds. First, we show that any bounded-error, k-query quantum algorithm for ordered search can be implemented by a k-query algorithm in this special class. Second, we use linear programming to show that the best exact 5-query quantum algorithm can search a list of length 7265, giving an ordered search algorithm that asymptotically uses $5 \log_{7265} n \approx 0.390 \log_2 n$ quantum queries.

3.1 Introduction

Ordered search is a fundamental computational primitive with broad applications. The problem asks, given a sorted list of n numbers, to find the location of some specified target. In the classical

context, ordered search is solved by the standard binary search algorithm. In the decision tree model, any comparison-based algorithm for this problem requires at least $\log_2 n$ comparisons, because the output is $\log_2 n$ bits and each query returns a single bit. Binary search uses $\lceil \log_2 n \rceil$ comparisons, and therefore is asymptotically optimal, even up to the leading constant.

Quantum algorithms offer a constant-factor speedup, but the exact value of this constant factor remains elusive. Most upper bounds to date have been achieved by exhibiting an algorithm that uses a small number k of queries to search an m-element list exactly, and recursively applying that algorithm to obtain a $k \log_m n$ upper bound. Attention has focused on a special class of algorithms called *translation-invariant algorithms* introduced by Farhi, Goldstone, Gutmann, and Sipser [FGGS99]. These algorithms use no workspace, and are restricted to non-query operations that act diagonally in the Fourier basis, making them easier to analyze. [FGGS99] exhibited a 3-query translation-invariant algorithm for searching a 52-element list exactly, which can be recursively applied to give a $3 \log_{52} n \approx 0.526 \log_2 n$-query algorithm. Using a different technique called *pebbling*, Høyer, Neerbek, and Shi gave a $\log_3 n \approx 0.631 \log_2 n$-query algorithm [HNS02]. Brookes, Jacokes, and Landahl used gradient descent to exhibit a better translation-invariant algorithm, improving the upper bound to $4 \log_{550} n \approx 0.439 \log_2 n$ [BJL04]. Most recently, Childs, Landahl, and Parillo utilized a semidefinite programming formulation to numerically find a 4-query translation-invariant algorithm for exactly searching a list of size 605, giving the best known upper bound of $4 \log_{605} n \approx 0.433 \log_2 n$ [CLP07]. Lifting the requirement of an exact algorithm, Ben-Or and Hassidim gave a zero-error quantum algorithm using about $0.323 \log_2 n$ queries in expectation [BH07], also based on a translation-invariant algorithm. The bounded-error setting has not been studied in detail, as most known algorithms have been derived by composing (exact) quantum algorithms for small instances. (In particular, the error dependence of the coefficient of $\log_2 n$ is not well understood.)

Limitations on quantum algorithms for ordered search have been established through a sequence of lower bounds. Buhrman and de Wolf showed a lower bound of $\Omega\big(\sqrt{\log n}/\log\log n\big)$ through a reduction to parity [BdW99]. This was improved to $\Omega\big(\log n/\log\log n\big)$ by Farhi, Gold-

stone, Gutmann and Sipser [FGGS98], then to $\frac{1}{12}\log_2 n \approx 0.0833 \log_2 n$ by Ambainis [Amb99]. Høyer, Neerbek, and Shi used the adversary method to prove the best known quantum lower bound of $\frac{1}{\pi}(\ln n - 1) \approx 0.221 \log_2 n$ [Amb00; HNS02]. Childs and Lee then proved that this is the best possible adversary lower bound, up to an additive constant, by exactly characterizing the adversary quantity of ordered search [CL08].

Our Results

As mentioned above, most upper bounds for the ordered search problem have been obtained by considering the subclass of translation-invariant algorithms [FGGS98]. Our main result shows that there is no loss of generality in considering this subclass. This establishes translation-invariant quantum algorithms as a route towards settling this problem in both the exact and bounded-error settings.

Theorem 23. *Let $\varepsilon \geq 0$, and suppose that there is an ε-error quantum algorithm for the n-element ordered search problem using k queries. Then there exists an ε-error translation-invariant algorithm solving the same problem using the same number of queries.*

Our proof, which works by studying the symmetries of the ordered search problem, is organized as follows. In Section 3.3.1, we show that controlled access to the query oracle is unnecessary for this problem. In Section 3.3.2, we generalize a semidefinite programming characterization of exact (i.e., $\varepsilon = 0$) translation-invariant algorithms for the ordered search problem developed by Childs, Landahl, and Parillo to bounded-error algorithms [CLP07]. A key step in our proof is the observation that the variables of the SDP can be taken to be rank-1 matrices without loss of generality. Next, in Section 3.3.3 we analyze how a semidefinite program (SDP) of Barnum, Saks, and Szegedy that captures the quantum query complexity of an arbitrary problem interacts with the symmetries of the problem [BSS03]. Finally, in Section 3.3.4 we show that the symmetries of the ordered search problem can be used to reduce this program to our generalization of the SDP given by Childs, Landahl, and Parillo. It is worth emphasizing that the definition of a translation-invariant algorithm as in [FGGS99] assumes that the algorithm uses no workspace, in addition to

having translation symmetry. This means our proofs also establish that no workspace is needed by optimal algorithms for ordered search.

We show that the symmetries of the ordered search problem can be used to reduce this SDP to another one that characterizes ε-error translation-invariant quantum algorithms. It is worth emphasizing that the definition of a translation-invariant algorithm as in [FGGS99] assumes that the algorithm uses no workspace, in addition to having translation symmetry. This means our proofs also establish that no workspace is needed by optimal algorithms for ordered search.

Our second result is an improved upper bound of $5 \log_{7265} n \approx 0.390 \log_2 n$ for the exact ordered search problem, which follows from an exact 5-query quantum algorithm that can search a 7265-element list. To find this algorithm, we consider a linear programming relaxation of the polynomial characterization of [FGGS99], and develop a heuristic approach to finding the algorithm by solving instances of this linear program.

Theorem 24 (Informal)**.** *The largest list exactly searchable by a* 5*-query quantum algorithm is of size* 7265*.*

As before, a recursive application of this algorithm leads to the claimed $5 \log_{7265} n$ asymptotic upper bound.

3.2 Preliminaries

In this section we cover some of the technical background and basic notation used throughout this chapter.

3.2.1 Notation

We denote by J_n the $n \times n$ all-ones matrix. We use zero-based numbering when indexing matrices by the integers. Given a matrix A we denote its (i, j) entry by $A[i, j]$. We denote by $A \odot B$ the Hadamard (i.e., element-wise) product of two matrices A and B of the same dimensions. Given an

$n \times n$ square matrix A and an integer $-n < j < n$, we denote by $\mathrm{tr}_j(A)$ the trace along the jth sub- or superdiagonal of A. Concretely,

$$\mathrm{tr}_j(A) \coloneqq \begin{cases} \sum_{i=0}^{n-1-j} A[i, i+j], & j \geq 0 \\ \sum_{i=0}^{n-1+j} A[i-j, i], & j < 0. \end{cases}$$

We refer to the quantities $\mathrm{tr}_j(A)$ for $-n < j < n$ collectively as the *generalized traces* of the matrix A. For convenience, we let $\mathrm{tr}_j(A) \coloneqq 0$ for $|j| \geq n$. We denote by $\mathrm{Tr}_j(A)$ the jth cyclic trace of A, i.e.,

$$\mathrm{Tr}_j(A) \coloneqq \begin{cases} \mathrm{tr}_j(A) + \mathrm{tr}_{j-n}(A), & 0 \leq j < n \\ \mathrm{tr}_j(A) + \mathrm{tr}_{j+n}(A), & -n < j < 0. \end{cases}$$

Let $[n]$ denote the set $\{1, \dots, n\}$. We denote permutations $\sigma \in S_n$ using cycle notation. Given two permutations $\sigma, \tau \in S_n$ we write $\sigma\tau$ for the composition $\sigma \circ \tau$. To each $\sigma \in S_n$ we associate the permutation matrix P_σ obtained by permuting the rows of the identity matrix I_n according to σ. Because we use zero-based indexing throughout this chapter, we regard elements of the symmetric group S_n as permutations of the set $\{0, \dots, n-1\}$, rather than the more conventional choice of $[n]$.

Given an n-tuple of scalars $a = (a_0, \dots, a_{n-1})$ we denote by $\mathrm{toep}(a_0, \dots, a_{n-1})$ the $n \times n$ symmetric Toeplitz matrix with first row given by a. We denote the set of $n \times n$ positive semidefinite matrices by $\mathbf{S}_+^n$.

We denote by $\mathbb{C}_n[z]$ the vector space of complex polynomials in z that have degree at most n. For a polynomial $p(z) = \sum_j c_j z^j \in \mathbb{C}[z]$, we denote the l_2-norm of p by $\|p\|_2$, i.e.,

$$\|p\|_2 = \sqrt{\sum_j |c_j|^2}.$$

Whenever the degree of $p(z)$ is understood to be n, we denote the coefficient vector of $p(z)$ by $\mathbf{p} \coloneqq (\, c_0 \, \cdots \, c_n \,)^\top \in \mathbb{C}^{n+1}$.

For $b \in \{0,1\}$ we denote by b^j the string $b \dots b$ consisting of j identical bits. Given two bitstrings w_1 and w_2 we denote their concatenation by $w_1 w_2$.

Throughout this chapter we denote by T the cyclic translation operation

$$T|j\rangle = |j+1\rangle \qquad (\forall j \in \mathbb{Z}/2n)$$

where the arithmetic on the right-hand side is done modulo $2n$. As a matrix, T is just the permutation matrix corresponding to the cycle $(0\ 1\ \dots\ 2n-1)$, and acts by mapping $(v_0 \quad v_1 \quad \cdots \quad v_{2n-1})^\top$ to $(v_{2n-1} \quad v_0 \quad \cdots \quad v_{2n-2})^\top$. Abusing notation slightly, given a bitstring $w = w_0 w_1 \dots w_{2n-1}$ we write Tw as a shorthand for $w_{2n-1} w_0 \dots w_{2n-2}$.

3.2.2 The Ordered Search Problem

In the ordered search problem, we are tasked with finding the first element greater or equal to some target value t in a sorted list of numbers $a_0 \le a_1 \le \cdots \le a_{n-1}$, or the last element if no such value exists. We consider this problem in the so-called comparison model, where the list can only be accessed using comparisons that determine whether $a_i \ge t$. In this model, the problem can be framed as a query problem, where each instance is represented by a bitstring $x \in \{0,1\}^n$ such that $x_i = 1$ if and only if $a_i \ge t$. Given query access to x, the goal is to output the (zero-based) index of the first 1 in x.

Problem 25 (Ordered Search)**.** *Let $x = 0^j 1^{n-j}$ for some $j \in \{0, \dots, n-1\}$. Given query access to x, return j.*

Reference [FGGS99] considered a symmetrized variant of this problem by replacing the string x with

$$y = x\bar{x}$$

where $\bar{x}$ denotes the bitwise negation of x. The benefit of this change is that the symmetrized

input satisfies the translation equivariance

$$(T^j y)_i = y_{i-j} \qquad (\forall i, j \in \mathbb{Z}/2n)$$

where the arithmetic is done modulo $2n$. Note that we index bitstrings from 0, allowing us to do modular arithmetic for the indices as above.

Problem 26 (Ordered Search, symmetrized)**.** *Let $w = 1^n 0^n$ and $y = T^j w$ for some $j \in \mathbb{Z}/2n$. Given query access to y, return j* mod *n.*

It is clear that Problems 25 and 26 are equivalent, as black-box reductions turn one into the other with no query overhead. Note that in Problem 26 we only require the answer to be returned modulo n, while j ranges over $\mathbb{Z}/2n$. In the reduction from Problem 25 to Problem 26, the value of the shift modulo n is all that is required to solve the original ordered search instance. In fact, using the phase oracle defined in Section 3.2.3, it is impossible to distinguish a bitstring from its bitwise complement. This corresponds to distinguishing a shift j from a shift $(j + n)$ mod $2n$. The algorithms considered in this chapter use a phase oracle of this form.

3.2.3 Quantum Query Algorithms

Let $y = T^j w$ as in Problem 26. In the quantum query model, access to the Boolean oracle $j \mapsto y_j$ is customarily provided by a unitary *phase oracle* acting as

$$O_y |j\rangle = (-1)^{y_j} |j\rangle .$$

While the usual model allows for controlled queries (or equivalently, phase queries with the possibility of a null query), in Section 3.3.1 we show that for the ordered search problem we can restrict our attention (with no loss of generality) to algorithms without controlled query access.

A quantum query algorithm $\mathscr{A}$ that makes k queries to the oracle O_y is defined by a sequence $U_1, U_2, \ldots, U_k$ of unitary operators and an initial state $|\psi_0\rangle$. The final state of such an algorithm on

input y is

$$|\psi_k^{(y)}\rangle = U_k O_y U_{k-1} \dots U_1 O_y |\psi_0\rangle \,. \tag{3.1}$$

In the context of our problem, we say that $\mathscr{A}$ solves the ordered search problem *with error* ε if there exists a family $\Pi_0, \dots, \Pi_{n-1}$ of orthogonal projectors with pairwise orthogonal supports such that

$$\left\| \Pi_{j \bmod n} |\psi_k^{(T^j w)}\rangle \right\|^2 \geq 1 - \varepsilon \qquad (\forall j \in \{0, 1, \dots, 2n-1\}).$$

3.2.4 Translation-Invariant Algorithms

We work with the symmetrized version of the ordered search problem (Problem 26), which is invariant under cyclic shift. It is natural to expect that cyclic symmetry in the problem will lead to cyclic symmetry in the algorithm. In particular, by symmetrizing the solution of a semidefinite program for quantum query algorithms presented in Ref. [BSS03], one can show that any quantum query algorithm for a symmetric query problem can be chosen to be symmetric, in a certain sense.

Reference [FGGS99] focuses on "translation-invariant" quantum algorithms, but in a narrower sense: the unitaries $U_1, \dots, U_k$ of the algorithm act *only* on the index (from 0 to $2n-1$), and commute with the translation operator T. In other words, the algorithm cannot use any workspace as a side-effect of how the symmetry is defined. More formally, we consider the following class of algorithms.

Definition 27 (Translation-Invariant Algorithm)**.** A k-query quantum algorithm $\mathscr{A}$ for the ordered search problem defined by unitaries $U_1, U_2, \dots, U_k$, as in Equation (3.1), is said to be *translation-invariant* if it satisfies the following properties:

(i) $\mathscr{A}$ operates on a single register of dimension $2n$.

(ii) The initial state is the uniform superposition $|\psi_0\rangle = \frac{1}{\sqrt{2n}} \sum_{j=0}^{2n-1} |j\rangle$.

(iii) Conjugation by the translation operator T leaves the unitaries U_t invariant:

$$TU_tT^{-1} = U_t \qquad (\forall t \in [k]).$$

(iv) Measurement is made in the basis $\{|\phi_j\rangle \,:\, j = 0, \ldots, n-1\}$ where

$$|\phi_j\rangle := \frac{1}{\sqrt{2}}\big(|j\rangle + (-1)^k\,|j+n\rangle\big).$$

The measurement basis depends on the parity of k because in the momentum basis, all the mass shifts from even to odd states (and vice versa) under the query oracle.

Recall that one of our main results (Theorem 23) reduces any exact algorithm to a translation-invariant form. We stress that this is not just about symmetrizing the algorithm, which is possible with existing techniques, but *removing the workspace*. Regrettably, the established name "translation-invariant" for this kind of algorithm hides this aspect of our result.

3.2.5 Nonnegative Laurent Polynomials

A degree-d *Laurent polynomial* is a polynomial in z and z^{-1} of the form

$$q(z) = \sum_{j=-d}^{d} a_j z^j,$$

where $a_j \in \mathbb{C}$. The Laurent polynomial $q(z)$ is said to be *real-valued* if it is real-valued on the unit circle ($|z| = 1$). It is easy to see that $q(z)$ is real-valued if and only if $a_j = \bar{a}_{-j}$. We say that $q(z)$ is *symmetric* if $a_j = a_{-j}$. Thus, a symmetric real-valued Laurent polynomial can be parameterized by $d+1$ real parameters a_j as

$$q(z) = \frac{1}{2}\sum_{j=0}^{d} a_j(z^j + z^{-j}).$$

Lastly, we say that $q(z)$ is *nonnegative* if it is real-valued and nonnegative on the unit circle.

Besides their canonical expression as polynomials in z and z^{-1}, we consider two additional ways of representing nonnegative Laurent polynomials. The first is via the Fejér-Riesz theorem, which states that any nonnegative Laurent polynomial can be factorized as the square modulus of an ordinary polynomial [Fej16; Rie16].

Theorem 28 (Fejér-Riesz)**.** *Let $q(z)$ be a degree-d Laurent polynomial. Then $q(z)$ is nonnegative (on the unit circle) if and only if there exists a degree-d polynomial $p(z) \in \mathbb{C}[z]$ such that*

$$q(z) = |p(z)|^2 \quad (\forall z : \ |z| = 1).$$

Furthermore, $p(z)$ may be chosen so that all of its zeros are contained in the closed unit disk.

In this context, we refer to the polynomial $p(z)$ as a *spectral factor of* $q(z)$. In general there are several distinct spectral factors $p(z)$ for any nonnegative polynomial $q(z)$. However, $p(z)$ is unique (up to a complex phase) subject to the constraint that all of its zeros are in the closed unit disk [Fej16].

Another representation is the so-called *Gram matrix representation* [Dum07]. Recall that a Laurent polynomial $q(z) = \sum_{j=-d}^{d} a_j z^j$ is real-valued if and only if $a_j = \bar{a}_{-j}$ for all $j \in \{-d, \dots, d\}$. An equivalent condition is that there exists $(d+1) \times (d+1)$ Hermitian matrix Q such that

$$q(z) = \psi(z^{-1})^\top Q \psi(z).$$

where $\psi(z) = \begin{pmatrix} 1 & z & \cdots & z^d \end{pmatrix}^\top$. Equivalently, the coefficients of $q(z)$ are given by generalized traces of Q:

$$a_j = \mathrm{tr}_j(Q) \qquad (j \in \{-d, \dots, d\}).$$

Then, $q(z)$ is nonnegative if and only if it can be represented this way by a positive semidefinite matrix $Q \succeq 0$.

Lemma 29 ([Dumo7, Thm. 2.5]). *Let $q(z) = \sum_{j=-d}^{d} a_j z^j$ be a degree-d Laurent polynomial. Then $q(z)$ is nonnegative (on the unit circle) if and only if there exists a $(d+1)\times(d+1)$ Hermitian, positive semidefinite matrix Q such that $a_j = \mathrm{tr}_j(Q)$ for $j \in \{-d, \dots, d\}$.*

One example of a symmetric, nonnegative Laurent polynomial is the *Fejér kernel*

$$F_n(z) := \sum_{j=-(n-1)}^{(n-1)} \Big(1 - \frac{|j|}{n}\Big) z^j.$$

It is easy to check that the polynomial $p(z) := \frac{1}{\sqrt{n}}(1 + z + \cdots + z^{n-1})$ is a spectral factor of F_n. Moreover, the outer product $\mathbf{p}^*\mathbf{p}^\top = \frac{1}{n} J_n$ is a Gram matrix representation of F_n, where $\mathbf{p} \in \mathbb{C}^n$ is the column vector whose entries are the coefficients of $p(z)$.

Given a spectral factor $p(z)$ of a nonnegative Laurent polynomial $q(z)$, the rank-1 projector $\mathbf{p}^*\mathbf{p}^\top$ is a Gram matrix representation of the polynomial $q(z)$. A key step in this work exploits a partial converse of this fact.

Theorem 30. *Let $q(z) = \sum_{j=-(n-1)}^{n-1} a_j z^j$ be a nontrivial, nonnegative Laurent polynomial. Suppose that $Q \in \mathbf{S}_+^n$ is a Gram matrix representation of $q(z)$ for which the top left entry $Q[0,0]$ is maximal. Then Q has rank 1. Furthermore, $Q = \mathbf{p}^*\mathbf{p}^\top$ where $\mathbf{p} \in \mathbb{C}^n$ is the coefficient vector of a spectral factor $p(z)$ of $q(z)$.*

For completeness, we present the proof given in [Dumo7].

Proof. We first show that the rank of Q is 1. For each $-n < j < n$, the set of matrices $M \in \mathbb{C}^{n\times n}$ with $\mathrm{tr}_j M = a_j$ is closed. Thus so is the set of Gram matrix representations of $q(z)$, being the intersection of finitely many closed sets. Therefore, there exists a Gram representation $Q \in \mathbf{S}_+^n$ of $q(z)$ with maximal top left entry. As $q(z)$ is nontrivial, $Q \neq 0$. We may express Q as

$$Q = \begin{pmatrix} \alpha & v^\dagger \\ v & \widehat{Q} \end{pmatrix},$$

where α is a positive real scalar. Since Q is positive semidefinite, so are all of its conjugates. In particular,

$$\begin{pmatrix} 1 & 0 \\ -\frac{v}{\alpha} & I_{n-1} \end{pmatrix} Q \begin{pmatrix} 1 & 0 \\ -\frac{v}{\alpha} & I_{n-1} \end{pmatrix}^\dagger = \begin{pmatrix} \alpha & 0 \\ 0 & \widehat{Q} - \frac{vv^\dagger}{\alpha} \end{pmatrix} \succeq 0.$$

Thus the $(n-1) \times (n-1)$ submatrix $P \coloneqq \widehat{Q} - \frac{vv^\dagger}{\alpha}$ is positive semidefinite as well, and so is

$$Q' \coloneqq \begin{pmatrix} \alpha & v^\dagger \\ v & \frac{vv^\dagger}{\alpha} \end{pmatrix} + \begin{pmatrix} P & 0 \\ 0 & 0 \end{pmatrix}.$$

Since $Q' = Q + \operatorname{diag}(P, 0) - \operatorname{diag}(0, P)$, all of the generalized traces of Q and Q' are identical, i.e., Q' is another Gram representation of $q(z)$.

Now suppose for contradiction that $\operatorname{rank}(Q) \neq 1$. Then $P \neq 0$ and in particular P must have a strictly positive diagonal entry. Cyclically rotating the elements of P down and to the right if necessary, we may assume that $P[0,0] > 0$. Note that this change does not affect the fact that Q' is a Gram matrix representation of $q(z)$. Thus, Q' is a Gram representation of $q(z)$ with $Q'[0,0] = Q[0,0] + P[0,0] > Q[0,0]$, giving us the required contradiction.

The "furthermore" part follows by a straightforward calculation. Since Q is a positive semidefinite matrix of rank 1, $Q = \mathbf{p}^* \mathbf{p}^\top$ for some nonzero vector $\mathbf{p} = (b_0 \quad \cdots \quad b_{n-1})^\top$. Then the (i,j) entry of Q equals $b_i^* b_j$, and as Q represents $q(z)$,

$$a_j = \operatorname{tr}_j(Q) = \begin{cases} \sum_{i=0}^{n-1-j} b_i^* b_{i+j}, & j \geq 0 \\ \sum_{i=0}^{n-1+j} b_{i-j}^* b_i, & j < 0. \end{cases}$$

Finally, notice that these are precisely the coefficients of the Laurent polynomial

$$p(z)p\Big(\frac{1}{z^*}\Big)^*,$$

where $p(z) \coloneqq \sum_{j=0}^{n-1} b_j z^j$. □

3.2.6 Characterizing Algorithms by Polynomials

Farhi, Goldstone, Gutmann, and Sipser developed a polynomial characterization of translation-invariant quantum algorithms for the ordered search problem. The following result is implicit in their work, as summarized between equations (4.13) and (4.14) of [FGGS99].

Theorem 31. *There exists an ε-error, k-query translation-invariant algorithm for the n-element ordered search problem if and only if there exist polynomials $p_1, \ldots, p_k \in \mathbb{C}_{n-1}[z]$ such that*

$$p_0(z) \coloneqq \frac{1}{\sqrt{n}}\Big(1 + z + \cdots + z^{n-1}\Big) \tag{3.2}$$

$$|p_t(z)| = |p_{t-1}(z)| \qquad (\forall z :\ z^n = (-1)^t,\ t \in [k]) \tag{3.3}$$

$$|p_k(0)|^2 \geq 1 - \varepsilon \tag{3.4}$$

$$\|p_t\|_2 = 1 \qquad (t \in [k]). \tag{3.5}$$

For exact algorithms (corresponding to $\varepsilon = 0$), the authors considered the following equivalent characterization in terms of the nonnegative Laurent polynomials $q_t(z) \coloneqq p_t(z)p_t(1/z^*)^*$. The main benefit of this characterization is that we no longer need to deal with absolute values, as the constraints in Equation (3.3) are replaced by Equation (3.7).

Theorem 32 ([FGGS99]). *There exists an exact k-query translation-invariant quantum algorithm for the n-element ordered search problem if and only if the following program is feasible.*

Find symmetric, nonnegative Laurent polynomials $(q_t)_{t=0}^k$ of degree less than n s.t.

$$q_0 \equiv F_n \tag{3.6}$$

$$q_t(z) = q_{t-1}(z) \qquad (\forall z \text{ with } z^n = (-1)^t, 1 \le t \le k) \tag{3.7}$$

$$q_k \equiv 1 \tag{3.8}$$

$$\frac{1}{2\pi}\int_0^{2\pi} q_t(e^{i\theta})\, d\theta = 1 \qquad (0 \le t \le k). \tag{3.9}$$

A follow-up work of Childs, Landahl, and Parillo used the Gram matrix representation to turn the program in Theorem 32 into a semidefinite program that is feasible if and only if an exact, k-query translation invariant algorithm exists for the n-element ordered search problem [CLP07, Theorem 3]. One limitation of their program, inherited from Theorem 32, is that it only applies to the exact ($\varepsilon = 0$) case.

In this chapter, we extend the result of [CLP07] by constructing an SDP that characterizes the existence of ε-error algorithms for the ordered search problem for all choices of $\varepsilon > 0$.

Theorem 33. *There exists an ε-error translation-invariant, k-query quantum algorithm for the n-element ordered search problem if and only if the following semidefinite program is feasible:*

$$Q^{(0)} \coloneqq J_n/n \tag{3.10}$$

$$\operatorname{tr}_j Q^{(t)} + (-1)^t \operatorname{tr}_{j-n} Q^{(t)} = \operatorname{tr}_j Q^{(t-1)} + (-1)^t \operatorname{tr}_{j-n} Q^{(t-1)} \qquad (0 < j < n,\ t \in [k]) \tag{3.11}$$

$$Q^{(k)}[0,0] \ge 1 - \varepsilon \tag{3.12}$$

$$\operatorname{tr} Q^{(t)} = 1 \qquad (t \in [k]) \tag{3.13}$$

$$Q^{(t)} \in \mathbf{S}_+^n \qquad (t \in [k]) \tag{3.14}$$

where J_n denotes the $n \times n$ all-ones matrix.

We prove Theorem 33 in Section 3.3.2. We use this theorem to show that, informally speaking, (workspace-free) translation-invariant algorithms are in fact optimal for the ordered-search problem. The key idea is that we may without loss of generality assume that the variables of the SDP are all rank-1 matrices. While this is a nonconvex constraint in general, it can be assumed in this particular case, as can be shown using Theorem 30.

3.3 Translation-Invariant Quantum Algorithms Are Optimal

As discussed in the introduction, much of the prior research on quantum algorithms for the ordered search problem focused on the special class of algorithms that are (workspace-free and) translation-invariant (Definition 27). Indeed, the best known algorithms in the exact ($\varepsilon = 0$) setting use this framework [FGGS99; CLP07]. In this section we prove that this class of algorithms is in fact optimal in the sense that for any ε-error algorithm that solves the n-element ordered search problem, there exists an ε-error translation-invariant algorithm that solves the problem using the same number of queries.

3.3.1 Removing Controlled Access

We begin by showing that controlled query access is not necessary for the symmetrized ordered search problem. We use this to eliminate matrices corresponding to null queries when we consider a semidefinite programming characterization of quantum query complexity in Section 3.3.4.

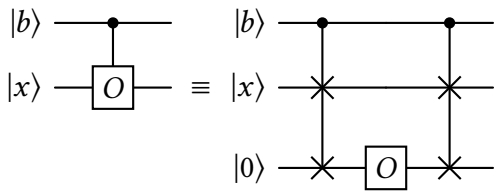
The key observation is that the problem asks for the shift modulo n, which means that the input x and its bitwise negation $\bar{x}$ correspond to the same output.

Proposition 34. *Any quantum algorithm for the symmetrized ordered search problem (Problem 26) that has controlled access to the input oracle can be simulated by an algorithm without controlled access with the same number of queries and success probability.*

Proof. Let w be as in Problem 26. Let $\mathscr{A}$ be an algorithm that makes controlled queries to the input oracle O. For an input string $s = T^j w$, a controlled query to O acts as

$$\text{ctrl-}O\,|b\rangle\,|x\rangle := (-1)^{b\cdot s_x}\,|b\rangle\,|x\rangle\,.$$

We modify $\mathscr{A}$ to replace each ctrl-O gate with the circuit below, which uses controlled-SWAPs to apply O to either an eigenstate (in this case $|0\rangle$) when $b = 0$, or to $|x\rangle$ when $b = 1$.



In other words, we map

$$\text{ctrl-}O\,|b\rangle\,|i\rangle \mapsto \underbrace{\text{ctrl-SWAP}\cdot(O\otimes I)\cdot\text{ctrl-SWAP}}_{O'}\,|b\rangle\,|0\rangle\,|x\rangle\,, \tag{3.15}$$

where

$$\text{ctrl-SWAP}\,|b\rangle\,|x\rangle\,|y\rangle := \begin{cases} |b\rangle\,|x\rangle\,|y\rangle\,, & \text{if } b = 0 \\ |b\rangle\,|y\rangle\,|x\rangle\,, & \text{if } b = 1. \end{cases}$$

For simplicity, we omit the $|0\rangle$ register of Equation (3.15) going forward. This register remains in the $|0\rangle$ state no matter the values of $|b\rangle$ and $|i\rangle$, so this loses no information. We can write the action of O' as

$$O'\,|b\rangle\,|i\rangle = (-1)^{x_i\cdot b}\underbrace{(-1)^{x_0\cdot(1-b)}}_{\text{Phase}}\,|b\rangle\,|i\rangle\,,$$

where the term labeled Phase is the only difference from the original oracle O. For any input of

the form $x \in 0^j 1^n 0^{n-j}$ with $j > 0$, the action of O' is the same as O (neglecting the ancilla register), because $x_0 = 0$ and therefore the Phase term is 1. On such inputs, $\mathscr{A}'$ outputs the same number as $\mathscr{A}$, namely j mod n.

The remaining inputs are of the form $x = 1^j 0^n 1^{n-j}$ for $j > 0$. Such an input is simply the negation of an input of the form $0^j 1^n 0^{n-j}$, and an algorithm with an uncontrolled phase oracle cannot distinguish the negation as the oracles only differ by a global phase. It follows that $\mathscr{A}'$ outputs j mod n on inputs of this form as well, which is the correct answer. □

3.3.2 An SDP Characterization of Translation-Invariant Algorithms

In this section we establish our semidefinite programming characterization of translation-invariant algorithms for the ordered search problem. Our starting point is the polynomial characterization of [FGGS99], which we restated below. In particular, we show that the SDP stated in our Theorem 33 is feasible if and only if the conditions in Theorem 31 are satisfied. The key idea is that we may without loss of generality assume that the variables of the SDP are all rank-1 matrices. While this is a nonconvex constraint in general, it can be assumed in this particular case, as can be shown using Theorem 30.

Theorem 31. *There exists an ε-error, k-query translation-invariant algorithm for the n-element ordered search problem if and only if there exist polynomials $p_1, \ldots, p_k \in \mathbb{C}_{n-1}[z]$ such that*

$$p_0(z) := \frac{1}{\sqrt{n}}\left(1 + z + \cdots + z^{n-1}\right) \tag{3.2}$$

$$|p_t(z)| = |p_{t-1}(z)| \qquad (\forall z :\ z^n = (-1)^t,\ t \in [k]) \tag{3.3}$$

$$|p_k(0)|^2 \geq 1 - \varepsilon \tag{3.4}$$

$$\|p_t\|_2 = 1 \qquad (t \in [k]). \tag{3.5}$$

We recall our characterization below.

Theorem 33. *There exists an ε-error translation-invariant, k-query quantum algorithm for the n-element ordered search problem if and only if the following semidefinite program is feasible:*

$$Q^{(0)} \coloneqq J_n/n \tag{3.10}$$

$$\operatorname{tr}_j Q^{(t)} + (-1)^t \operatorname{tr}_{j-n} Q^{(t)} = \operatorname{tr}_j Q^{(t-1)} + (-1)^t \operatorname{tr}_{j-n} Q^{(t-1)} \qquad (0 < j < n,\ t \in [k]) \tag{3.11}$$

$$Q^{(k)}[0,0] \geq 1 - \varepsilon \tag{3.12}$$

$$\operatorname{tr} Q^{(t)} = 1 \qquad (t \in [k]) \tag{3.13}$$

$$Q^{(t)} \in \mathbf{S}_+^n \qquad (t \in [k]) \tag{3.14}$$

where J_n denotes the $n \times n$ all-ones matrix.

This theorem is reminiscent of the characterization of exact ($\varepsilon = 0$) algorithms given by Childs, Landahl, and Parillo [CLP07], except that the condition for the last matrix in Equation (3.12) is different, allowing for a nonzero error probability. While this may seem like a minor change, establishing this generalization requires additional ingredients. In particular, our proof crucially relies on the fact that given a nonnegative Laurent polynomial, one can always choose a Gram matrix representation that has rank one (Theorem 30). We begin by establishing two lemmas.

Lemma 35. *The constraint in Equation (3.11) can be replaced by the following equation without affecting the feasibility of the SDP stated in Theorem 33:*

$$\sum_{j=-(n-1)}^{(n-1)} \operatorname{tr}_j(Q^{(t)}) z^j = \sum_{j=-(n-1)}^{(n-1)} \operatorname{tr}_j(Q^{(t-1)}) z^j \qquad (\forall z :\ z^n = (-1)^t,\ t \in [k]). \tag{3.16}$$

Proof. First observe that the variables $Q^{(1)}, \ldots, Q^{(k)} \in \mathbf{S}_+^n$ of the SDP can be assumed to be real symmetric matrices. Indeed, if $Q^{(1)}, \ldots, Q^{(k)}$ give a feasible solution, then it is easy to see that so do the complex conjugates $\overline{Q}^{(1)}, \ldots, \overline{Q}^{(k)}$ of these matrices. By convexity, the matrices $\widehat{Q}^{(t)} \coloneqq \left(Q^{(t)} + \overline{Q}^{(t)}\right)/2$ give another solution, and these are real symmetric. Note that the same argument works even if Equation (3.11) is replaced by Equation (3.16). Therefore we may restrict our attention to real symmetric matrices.

We show that under the assumption that $Q^{(1)}, \dots, Q^{(k)} \in \mathbf{S}^n_+$ are real symmetric matrices of trace 1, the constraints in Equations (3.11) and (3.16) are equivalent. For each $t \in \{0, \dots, k\}$, let $q_t(z)$ be the nonnegative Laurent polynomial represented by $Q^{(t)}$ through the Gram matrix representation. Since the matrix $Q^{(t)}$ is symmetric, so is the polynomial $q_t(z)$. Thus we may parameterize it as

$$q_t(z) = \frac{1}{2} \sum_{j=0}^{n-1} a_j^{(t)} (z^j + z^{-j}).$$

Equation (3.11) is equivalent to

$$a_j^{(t)} + (-1)^t a_{n-j}^{(t)} = a_j^{(t-1)} + (-1)^t a_{n-j}^{(t-1)} \qquad (0 < j < n,\ t \in [k]) \tag{3.17}$$

On the other hand, Equation (3.16) is equivalent to the following equation:

$$q_t(z) = q_{t-1}(z) \qquad (\forall z :\ z^n = (-1)^t,\ t \in [k]). \tag{3.18}$$

We claim that Equations (3.17) and (3.18) are equivalent under the assumption that the matrices $Q^{(t)}$ have trace one. This is essentially shown in Proposition 51, which is proven in Section 3.4.1. More precisely, Proposition 51 shows that Equation (3.18) is equivalent to

$$(I_n + (-1)^t V_n)(a^{(t)} - a^{(t-1)}) = 0 \quad (\forall t \in [k]) \tag{3.19}$$

where $V_n = \mathrm{diag}(I_1, X_{n-1})$, with X_{n-1} denoting the $(n-1) \times (n-1)$ permutation matrix with 1s on the anti-diagonal. This equation is almost identical to Equation (3.17), except that it requires $a_0^{(t)} = a_0^{(t-1)}$ whenever t is even. However, the constraint that $\mathrm{tr}\, Q^{(t)} = 1$ (for all $t \in [k]$) is equivalent to $a_0^{(t)} = 0$ (for all $t \in [k]$), and under this constraint Equation (3.19) is indeed equivalent to Equation (3.17). □

The next lemma is a corollary of Theorem 30.

Lemma 36. *Consider the SDP from Theorem 33 with the constraint in Equation (3.11) replaced by Equation (3.16). The variables $Q^{(t)}$ of this SDP can, without loss of generality, be taken to have rank* 1.

Proof. Suppose that $Q^{(1)}, \dots, Q^{(k)} \in \mathbf{S}_n^+$ is a feasible solution of the SDP. For each $t \in [k]$, we can associate to $Q^{(t)}$ a nonnegative Laurent polynomial $q_t(z)$ of degree $n-1$ through the Gram matrix representation. By Theorem 30 this polynomial is also represented by some rank-1 matrix $\widetilde{Q}^{(t)} \in \mathbf{S}_n^+$. In particular, $\widetilde{Q}^{(t)}$ is such that all of the generalized traces of $Q^{(t)}$ and $\widetilde{Q}^{(t)}$ are equal:

$$\operatorname{tr}_j \widetilde{Q}^{(t)} = \operatorname{tr}_j Q^{(t)} \quad (-n < j < n).$$

We conclude that the rank-1 matrices $\widetilde{Q}^{(1)}, \dots, \widetilde{Q}^{(k)}$ satisfy the constraints in Equations (3.13) and (3.16). Finally, Theorem 30 guarantees that the top left entry of $\widetilde{Q}^{(k)}$ is maximal among Gram matrix representations of $q_k(z)$, and thus the constraint in Equation (3.12) is also satisfied:

$$\widetilde{Q}^{(k)}[0,0] \geq Q^{(k)}[0,0] \geq 1 - \varepsilon.$$

□

We now have all the ingredients of the proof of Theorem 33.

Proof (Theorem 33). We show that for any $k, n \in \mathbb{N}$, the conditions of Theorem 31 are met if and only if the SDP stated in the theorem is satisfied. We consider the SDP with the constraint of Equation (3.11) replaced by Equation (3.16), as this change does not affect feasibility (as shown in Lemma 35).

We define a map $\mathbb{C}_{n-1}[z] \setminus \{0\} \to \mathbf{S}_+^n$ from the set of nontrivial polynomials of degree less than n to the set of $n \times n$ positive semidefinite matrices as follows. Given a nonzero polynomial $p(z) = \sum_{j=0}^{n-1} b_j z^j$, define the positive semidefinite matrix $Q \coloneqq \mathbf{p}^* \mathbf{p}^\top$, where $\mathbf{p}$ is the n-dimensional column vector whose jth entry is b_j. This is a surjection onto the set of rank-1 positive semidefinite matrices that maps two polynomials that differ by a complex phase to the same matrix.

Given a list $(p_t)_{t=1}^k$ of nonzero polynomials $p_t(z) = \sum_{j=0}^{n-1} b_j^{(t)} z^j$, let $(Q^{(t)})_{t=1}^k$ be their images under this mapping. We show that the polynomials $(p_t)_{t=1}^k$ satisfy the constraints of Theorem 31 if and only if the matrices $\left(Q^{(t)}\right)_{t=1}^k$ are a feasible solution of the SDP (with the constraint in Equation (3.11) replaced by Equation (3.16)). We examine the constraints of Theorem 31 one by one:

(i) For each $t \in \{0, \dots, k\}$ define the formal Laurent polynomial

$$q_t(z) := p_t(z) p_t\Big(\frac{1}{z^*}\Big)^*.$$

By construction $q_t(z) = |p_t(z)|^2$ for any z on the unit circle, and the constraint in Equation (3.3) is equivalent to

$$q_t(z) = q_{t-1}(z) \quad (\forall z : \ z^n = (-1)^t, \ 1 \le t \le k).$$

Note that $q_t(z)$ is nonnegative on the unit circle and $Q^{(t)}$ is a Gram matrix representation of it. In particular, we can express the above constraint in terms of the matrices $Q^{(t)}$ as

$$\sum_{j=-(n-1)}^{(n-1)} \mathrm{tr}_j(Q^{(t)}) z^j = \sum_{j=-(n-1)}^{(n-1)} \mathrm{tr}_j(Q^{(t-1)}) z^j \qquad (\forall z : \ z^n = (-1)^t, \ 1 \le t \le k).$$

This is just Equation (3.16), as needed.

(ii) Observe that the top left entry of $Q^{(k)}$ is the square modulus of the constant term of $p_k(z)$, i.e.,

$$Q^{(k)}[0,0] = \left|b_0^{(k)}\right|^2 = |p_k(0)|^2.$$

Thus, the constraint capturing the success probability (Equation (3.4)) is equivalent to Equation (3.12).

(iii) By construction,

$$\operatorname{tr} Q^{(t)} = \sum_{j=0}^{n-1} Q^{(t)}[j,j] = \sum_{j=0}^{n-1} \left|b_j^{(t)}\right|^2 = \|p_t\|_2^2.$$

Hence the normalization constraint (Equation (3.5)) is equivalent to Equation (3.13).

We have shown that the conditions in Theorem 31 are met if and only if the SDP (with Equation (3.11) replaced by Equation (3.16)) has a feasible solution that consists entirely of rank-1 matrices. This restriction, however, is with no loss of generality. Indeed, by Lemma 36, whenever this SDP is feasible, there exists a feasible solution that consists of rank-1 matrices only.

We conclude that the SDP stated in the theorem, with Equation (3.11) replaced by Equation (3.16), is feasible if and only if the conditions of Theorem 31 are met. By Lemma 35, the same is true of the original SDP as well. □

3.3.3 Symmetries and the Barnum-Saks-Szegedy SDP

In this section, we review a semidefinite programming characterization of quantum query complexity developed by Barnum, Saks, and Szegedy [BSS03], and study how symmetries of a problem manifest in terms of the solutions of this SDP.

For $X \subseteq \{0,1\}^n$, let $f\colon X \to Y$ be a function, k be a positive integer, and $\varepsilon \in [0, 1/2)$. Consider the task of deciding whether there exists a k-query quantum algorithm that computes f with error at most ε. To this end Barnum, Saks, and Szegedy defined the following a semidefinite program, and showed that the feasibility of this program determines the answer to the question.

Definition 37 ([BSS03, Section 3])**.** Let $X \subseteq \{0,1\}^n$, $f\colon X \to Y$ be a function, k be a positive integer, and $\varepsilon \in [0, 1/2)$.

- Let $J \coloneqq J_{|X|}$ denote the $|X| \times |X|$ all-ones matrix.
- For $i = 0, \ldots, n-1$, let E_i be the $|X| \times |X|$ matrix given by $E_i[x, x'] = (-1)^{x_i + x'_i}$.
- For each $y \in Y$, let Δ_y denote the diagonal matrix with $\Delta_y[x,x] = \delta_{f(x),y}$.

Let $P(f,k,\varepsilon)$ be the following semidefinite program:

$$J = \sum_{i=0}^{n-1} M_i^{(0)} + N^{(0)} \tag{3.20}$$

$$\sum_{i=0}^{n-1} E_i \odot M_i^{(t)} + N^{(t)} = \begin{cases} \sum_{i=0}^{n-1} M_i^{(t+1)} + N^{(t+1)}, & t = 0,\dots,k-2 \\ \sum_{y\in Y} \Gamma_y, & t = k-1 \end{cases} \tag{3.21}$$

$$\Delta_y \odot \Gamma_y \geq (1-\varepsilon)\Delta_y \quad (\forall y \in Y) \tag{3.22}$$

$$M_i^{(t)}, N^{(t)}, \Gamma_y \in \mathbf{S}_+^{|X|}. \tag{3.23}$$

Theorem 38 ([BSS03, Theorem 1])**.** *Let X, k, ε and $f\colon X \to Y$ be as in Definition 37. Then there exists a k-query quantum algorithm computing f with error at most ε if and only if the program $P(f,k,\varepsilon)$ is feasible.*

Remark 39. It is shown implicitly in Ref. [BSS03] that the matrices $G^{(t)} := \sum_{i=0}^{n-1} M_i^{(t)} + N^{(t)}$ for $t = 0,\dots,k-1$ can be taken without loss of generality to be Gram matrices of unit-norm vectors. Note that, as is the case for $t = 0$, these vectors are not necessarily distinct.

We will extensively use symmetries in our study of the ordered search problem. We formalize the notion of a symmetry of a function f as a 3-tuple of permutations of the input set, index set, and output set, that satisfy certain relations involving f.

Definition 40 (Symmetry of a query problem)**.** Let $X \subseteq \{0,1\}^n$ and let $f\colon X \to Y$ and $(E_i)_{i=0}^{n-1}$ be as in Definition 37. A symmetry of the function f is a 3-tuple (π,ρ,σ) of permutations of, respectively, the inputs X, query locations $\{0,\dots,n-1\}$, and outputs Y such that

$$E_{\rho(i)}[x,x'] = E_i[\pi(x),\pi(x')] \qquad (\forall i \in \{0,\dots,n-1\}, \forall x,x' \in X) \tag{3.24}$$

$$\sigma(f(x)) = f(\pi(x)) \qquad (\forall x \in X). \tag{3.25}$$

In this definition, π permutes the inputs, ρ permutes the query locations, and σ permutes the outputs. Next we describe a composition operation for symmetries.

Remark 41. Given a function $f\colon X \to Y$, let the matrices $(E_i)_{i=0}^{n-1}$ be as in Definition 40, and suppose that $(\pi_1, \rho_1, \sigma_1)$ and $(\pi_2, \rho_2, \sigma_2)$ are symmetries of f. Then

$$E_{\rho_1\rho_2(i)}[x, x'] = E_{\rho_2(i)}[\pi_1(x), \pi_1(x')] = E_i[\pi_2\pi_1(x), \pi_2\pi_1(x')] \quad \text{and}$$
$$\sigma_2\sigma_1(f(x)) = \sigma_2(f(\pi_1(x))) = f(\pi_2\pi_1(x)).$$

Thus we see that the symmetries of f form a group under the composition operation

$$(\pi_2, \rho_2, \sigma_2) \circ (\pi_1, \rho_1, \sigma_1) := (\pi_2 \circ \pi_1,\ \rho_1 \circ \rho_2,\ \sigma_2 \circ \sigma_1). \tag{3.26}$$

Note that in the second coordinate, the order of composition is reversed.

Next we show how symmetries transform the Δ matrices of Definition 37.

Proposition 42. *Suppose that (π, ρ, σ) is a symmetry of a function $f\colon X \to Y$, and let the matrices $\Delta_y \in \mathbb{R}^{|X|\times|X|}$ be defined as the diagonal matrices with $\Delta_y[x, x] = \delta_{f(x),y}$. Then for any $y \in Y$,*

$$\Delta_y = P_\pi^{-1}\Delta_{\sigma(y)}P_\pi,$$

where P_π is the permutation matrix associated with π.

Proof. First note that the matrix on the right-hand side is also diagonal. The equality of the diagonals can be verified as follows:

$$\begin{aligned}
\Delta_y[x, x] &= \delta_{f(x),y} && \text{(By definition)} \\
&= \delta_{\sigma(f(x)),\sigma(y)} && (\sigma \text{ is a bijection}) \\
&= \delta_{f(\pi(x)),\sigma(y)} && \text{(Equation (3.25))} \\
&= (P_\pi^{-1}\Delta_{\sigma(y)}P_\pi)[x, x].
\end{aligned}$$

□

Now we prove that symmetries generate new solutions from a given solution to the Barnum-Saks-Szegedy SDP.

Lemma 43. *Consider the program $P(f,k,\varepsilon)$, and suppose that (π,ρ,σ) is a symmetry of f. If the matrices $M_i^{(t)}, N^{(t)}, \Gamma_y$ constitute a solution to $P(f,k,\varepsilon)$, then so do matrices*

$$\widetilde{M}_i^{(t)} \coloneqq P^{-1} M_{\rho(i)}^{(t)} P \qquad \widetilde{N}^{(t)} \coloneqq P^{-1} N^{(t)} P \qquad \tilde{\Gamma}_y \coloneqq P^{-1} \Gamma_{\sigma(y)} P \tag{3.27}$$

where $P \coloneqq P_\pi$ is the permutation matrix corresponding to π.

Proof. First note that the initial constraint (Equation (3.20)) is satisfied since conjugation by a permutation matrix leaves the all-ones matrix invariant:

$$J = P^{-1} J P = P^{-1}\left(\sum_{i=1}^{n} M_i^{(0)} + N^{(0)}\right)P = \sum_{i=1}^{n} \widetilde{M}_i^{(0)} + \widetilde{N}^{(0)}.$$

Next, Equation (3.24) implies $E_i = P^{-1} E_{\rho(i)} P$. Therefore,

$$\begin{aligned} E_i \odot \widetilde{M}_i^{(t)} &= \left(P^{-1} E_{\rho(i)} P\right) \odot \left(P^{-1} M_{\rho(i)}^{(t)} P\right) \\ &= P^{-1}\left(E_{\rho(i)} \odot M_{\rho(i)}^{(t)}\right) P. \end{aligned}$$

Consider the forward step (Equation (3.21)), and notice that the conjugation by P commutes with the sums on both sides of the equation. The permutation ρ leaves the sums over i invariant, while the permutation σ leaves the sum over y invariant (for the case of $t = k-1$). Thus the forward step (Equation (3.21)) is also satisfied by the matrices defined in Equation (3.27).

Finally, by Proposition 42,

$$\begin{aligned} \Delta_y \odot \tilde{\Gamma}_y &= \left(P^{-1} \Delta_{\sigma(y)} P\right) \odot \left(P^{-1} \Gamma_{\sigma(y)} P\right) \\ &= P^{-1}\left(\Delta_{\sigma(y)} \odot \Gamma_{\sigma(y)}\right) P \\ &\geq (1-\varepsilon) P^{-1} \Delta_{\sigma(y)} P = (1-\varepsilon)\Delta_y. \qquad \text{(By Equation (3.22))} \end{aligned}$$

Therefore the final constraint, Equation (3.22), is also satisfied by the new matrices. □

By convexity, averaging over a group of symmetries gives another solution of $P(f, k, \varepsilon)$.

Corollary 44. *Suppose that the premise of Lemma 43 holds, and let H be a subgroup of the group of symmetries of f. For $t \in \{0, \dots, k-1\}$, $i \in \{0, \dots, n-1\}$, and $y \in Y$, let*

$$\widetilde{M}_i^{(t)} := \frac{1}{|H|} \sum P_\pi^{-1} M_{\rho(i)}^{(t)} P_\pi, \qquad \widetilde{N}^{(t)} := \frac{1}{|H|} \sum P_\pi^{-1} N^{(t)} P_\pi, \qquad \tilde{\Gamma}_y := \frac{1}{|H|} \sum P_\pi^{-1} \Gamma_{\sigma(y)} P_\pi.$$

where each summation ranges over $(\pi, \rho, \sigma) \in H$. Then these matrices satisfy $P(f, k, \varepsilon)$. Furthermore, this solution is invariant under the action of the elements of H, in the sense that

$$\forall (\pi, \rho, \sigma) \in H : \qquad \widetilde{M}_i^{(t)} = P_\pi^{-1} \widetilde{M}_{\rho(i)}^{(t)} P_\pi \qquad \widetilde{N}^{(t)} = P_\pi^{-1} \widetilde{N}^{(t)} P_\pi \qquad \tilde{\Gamma}_y = P_\pi^{-1} \tilde{\Gamma}_{\sigma(y)} P_\pi.$$

3.3.4 Translation-Invariant Algorithms

In this section we prove our main theorem, showing that the class of translation-invariant quantum algorithms (Definition 27) is optimal for the ordered search problem. We begin by studying the symmetries of the problem.

Recall that the possible inputs are the cyclic translates of the $2n$-bit string $w := 1^n 0^n$. The set of possible query locations is $\mathbb{Z}/2n$, while the set of possible outputs is $\mathbb{Z}/n$. Let $\tau = (0\ 1\ \dots\ (2n-1))$ and let $T = P_\tau$ denote the permutation matrix corresponding to τ. Then regarding the inputs as column vectors, multiplication by T permutes the set of inputs X, which we can express as $X = \{T^j w : j \in \mathbb{Z}/2n\}$. The function we need to compute is

$$f(T^j w) = j \bmod n.$$

In particular, we have $f(Tx) = f(x) + 1$, where the addition is done modulo n. Along these lines we identify a cyclic group of symmetries of f as follows.

Proposition 45. *The group of symmetries of the ordered search problem contains the cyclic subgroup generated by* $(T = P_\tau, \tau^{-1}, \mu)$*, where* $\tau = (0\ 1\ \dots\ (2n-1))$ *and* $\mu = (0\ 1\ \dots\ (n-1))$*.*

Here we abuse notation by taking the first coordinate of a symmetry to be a $2n \times 2n$ matrix T. We can identify T with a permutation of the inputs X by regarding the inputs as $2n$-dimensional column vectors. Then multiplication by T permutes the elements of X.

Proof. First observe that the group generated by (T, τ^{-1}, μ) with the group operation being composition (as in Equation (3.26)) is isomorphic to $\mathbb{Z}/2n$. This is because T and τ^{-1} have order $2n$ and μ has order $n \mid 2n$.

It remains to verify that (T, τ^{-1}, μ) is indeed a symmetry of f. First we verify that Equation (3.24) holds for $(\pi, \rho, \sigma) = (T, \tau^{-1}, \mu)$:

$$\begin{aligned} E_{\tau^{-1}(i)}[x, x'] &= E_{i-1}[x, x'] \\ &= (-1)^{x_{i-1}+x'_{i-1}} \\ &= (-1)^{(Tx)_i+(Tx')_i} \\ &= E_i[Tx, Tx'], \end{aligned}$$

where all arithmetic is done modulo $2n$.

As noted above, $f(Tx) = f(x) + 1$, and as such $\mu(f(x)) = f(x) + 1 = f(Tx)$, with arithmetic done modulo n this time. This is just Equation (3.25) for $(\pi, \rho, \sigma) = (T, \tau^{-1}, \mu)$. □

Averaging over this group of symmetries, we obtain solutions of a special form.

Corollary 46. *Consider the Barnum-Saks-Szegedy SDP (Definition 37) for the n-element symmetrized ordered search problem. The solution matrices* $M_i^{(t)}$*,* $N^{(t)}$*,* Γ_y *(for* $t \in \{0, \dots, k-1\}$*,* $i \in \{0, \dots, 2n-1\}$*, and* $y \in \{0, \dots, n-1\}$*) may be assumed without loss of generality to satisfy*

$$T^{-1} M_i^{(t)} T = M_{i+1}^{(t)}, \qquad T^{-1} N^{(t)} T = N^{(t)}, \qquad T^{-1} \Gamma_y T = \Gamma_{y-1},$$

where the subscript of $M_{i+1}^{(t)}$ is taken modulo $2n$, and the subscript of Γ_{y-1} is taken modulo n. Furthermore, the constant matrices E_i satisfy $E_{i+1} = T^{-1}E_iT$, where the subscript is taken modulo $2n$.

Proof. Let (T, τ^{-1}, μ) be as in Proposition 45, and let H denote the generated group of symmetries. Appealing to Corollary 44 for the group H, we have another SDP solution given by the matrices

$$\widetilde{M}_i^{(t)} := \frac{1}{2n}\sum_{j=0}^{2n-1} T^{-j}M_{i-j}^{(t)}T^j, \qquad \widetilde{N}^{(t)} := \frac{1}{2n}\sum_{j=0}^{2n-1} T^{-j}N^{(t)}T^j, \qquad \tilde{\Gamma}_y := \frac{1}{2n}\sum_{j=0}^{2n-1} T^{-j}\Gamma_{y+j}T^j.$$

Then

$$\begin{aligned} T^{-1}\widetilde{M}_i^{(t)}T &= \frac{1}{2n}\sum_{j=0}^{2n-1} T^{-(j+1)}M_{i-j}^{(t)}T^{(j+1)} \\ &= \frac{1}{2n}\sum_{j=0}^{2n-1} T^{-j}M_{i+1-j}^{(t)}T^j \qquad\qquad (\text{since } T^{2n} = I) \\ &= \widetilde{M}_{i+1}^{(t)}. \end{aligned}$$

The identities $T^{-1}\widetilde{N}^{(t)}T = \widetilde{N}^{(t)}$ and $T^{-1}\Gamma_y T = \Gamma_{y-1}$ can be verified analogously. Finally, the "furthermore" part follows directly from the definition of symmetries. □

From Proposition 34, we can restrict our attention to algorithms without controlled query access. In other words, we can drop the matrices $N^{(t)}$ corresponding to null queries.

Proposition 47. *The Barnum-Saks-Szegedy SDP characterizing ε-error quantum algorithms for the ordered search problem is equivalent to the following:*

$$\mathrm{Tr}_j\big(M^{(0)}\big) = 2 \qquad (0 \le j < 2n) \tag{3.28}$$

$$\mathrm{Tr}_j\big(E_0 \odot M^{(t)}\big) = \mathrm{Tr}_j\big(M^{(t+1)}\big) \qquad (0 \le j < 2n,\ 0 \le t < k) \tag{3.29}$$

$$M^{(k)}[0,0], M^{(k)}[n,n] \ge 1 - \varepsilon \tag{3.30}$$

$$M^{(t)} \in \mathbf{S}_+^{2n} \qquad (0 \le t \le k), \tag{3.31}$$

where

$$E_0 := \begin{pmatrix} J_n & -J_n \\ -J_n & J_n \end{pmatrix}.$$

The constraint $\mathrm{Tr}_j\big(M^{(0)}\big) = 2$ may seem somewhat unnatural. This is a result of the symmetrization of the problem (see the discussion preceding Problem 26), by which the size of the input strings is doubled. In Propositions 48 and 49 we show that the above program is equivalent to an analogous one where the matrices $M^{(t)} \in \mathbf{S}_+^{2n}$ are replaced by matrices $A^{(t)} \in \mathbf{S}_+^n$ of half the size, with the initial constraint becoming $\mathrm{Tr}\big(A^{(0)}\big) = 1$.

Proof. Let X be the set of all $2n$ cyclic translates of the bitstring $w := 1^n 0^n$, and let $Y = \mathbb{Z}/n$. Recall that the elements of X can be written as $T^j w$ where $T = P_\tau$ for $\tau = (0\ 1\ \ldots\ (2n-1))$ and $j \in \mathbb{Z}/2n$.

By Proposition 34 the null query matrices $N^{(t)}$ can be removed from the Barnum-Saks-Szegedy SDP (Definition 37) without loss of generality. We are left with

$$J_{2n} = \sum_{i=0}^{2n-1} M_i^{(0)} \tag{3.32}$$

$$\sum_{i=0}^{2n-1} E_i \odot M_i^{(t)} = \begin{cases} \sum_{i=0}^{2n-1} M_i^{(t+1)}, & t = 0, \ldots, k-2 \\ \sum_{y=0}^{n-1} \Gamma_y, & t = k-1 \end{cases} \tag{3.33}$$

$$\Delta_y \odot \Gamma_y \geq (1-\varepsilon)\Delta_y \quad (0 \leq y < n) \tag{3.34}$$

$$M_i^{(t)}, \Gamma_y \in \mathbf{S}_+^{2n}. \tag{3.35}$$

By Corollary 46 we can assume that the matrices $M_i^{(t)}$, Γ_y, and E_i satisfy $M_{i+1}^{(t)} = T^{-1} M_i^{(t)} T$, $T^{-1}\Gamma_y T = \Gamma_{y-1}$, and $E_{i+1} = T^{-1} E_i T$, respectively. In particular,

$$M_i^{(t)} = T^{-i} M_0^{(t)} T^i$$

$$E_i = T^{-i} E_0 T^i,$$

and in turn,

$$\begin{aligned} E_i \odot M_i^{(t)} &= \left(T^{-i} E_0 T^i\right) \odot \left(T^{-i} M_0^{(t)} T^i\right) \\ &= T^{-i}\left(E_0 \odot M_0^{(t)}\right) T^i. \end{aligned}$$

Hence the matrices $\sum_{i=0}^{2n-1} M_i^{(t)}$ and $\sum_{i=0}^{2n-1} E_i \odot M_i^{(t)}$ for $0 \le t < k$ are Toeplitz, in addition to being symmetric. More specifically, all of the entries of $\sum_{i=0}^{2n-1} M_i^{(t)}$ on the jth superdiagonal equal $\mathrm{Tr}_j(M_0^{(t)})$, while all the entries of $\sum_{i=0}^{2n-1} E_i \odot M_i^{(t)}$ on the jth superdiagonal equal $\mathrm{Tr}_j\left(E_0 \odot M_0^{(t)}\right)$:

$$\begin{aligned} \sum_{i=0}^{2n-1} M_i^{(t)} &= \mathrm{toep}\Big(\mathrm{Tr}_0\left(M_0^{(t)}\right), \dots, \mathrm{Tr}_{2n-1}\left(M_0^{(t)}\right)\Big) \\ \sum_{i=0}^{2n-1} E_i \odot M_i^{(t)} &= \mathrm{toep}\Big(\mathrm{Tr}_0\left(E_0 \odot M_0^{(t)}\right), \dots, \mathrm{Tr}_{2n-1}\left(E_0 \odot M_0^{(t)}\right)\Big). \end{aligned}$$

It follows by Equation (3.33) that $\sum_{y=0}^{n-1} \Gamma_y$ is Toeplitz as well. Since $T^{-1}\Gamma_y T = \Gamma_{y-1}$, we have

$$\begin{aligned} \sum_{y=0}^{n-1} \Gamma_y &= \sum_{y=0}^{n-1} T^y \Gamma_0 T^{-y} \\ &= \frac{1}{2}\mathrm{toep}\Big(\mathrm{Tr}_0(\Gamma_0), \mathrm{Tr}_1(\Gamma_0), \dots, \mathrm{Tr}_{2n-1}(\Gamma_0)\Big). \end{aligned}$$

Therefore the constraints in Equations (3.32) and (3.33) can be replaced by the following:

$$1 = \mathrm{Tr}_j\left(M_0^{(0)}\right) \qquad (0 \le j < 2n)$$

$$\mathrm{Tr}_j\left(E_0 \odot M_0^{(t)}\right) = \begin{cases} \mathrm{Tr}_j\left(M_0^{(t+1)}\right) & t = 0, \dots, k-2 \\ \frac{1}{2}\,\mathrm{Tr}_j(\Gamma_0), & t = k-1. \end{cases} \qquad (0 \le j < 2n)$$

Thus we eliminated the matrices $M_i^{(t)}$ and E_i for $i > 0$. To eliminate the matrices Γ_y (for $y > 0$) as well, we express the constraint

$$\Delta_y \odot \Gamma_y \geq (1-\varepsilon)\Delta_y \quad (0 \leq y < n) \tag{3.36}$$

in terms of Γ_0. To this end, by definition we have $\Delta_y = T^{-y}\Delta_0 T^y$, and accordingly,

$$\Delta_y \odot \Gamma_y = (T^{-y}\Delta_0 T^y) \odot (T^{-y}\Gamma_0 T^y) = T^{-y}(\Delta_0 \odot \Gamma_0)T^y.$$

Thus $\Delta_y \odot \Gamma_y \geq (1-\varepsilon)\Delta_y$ holds for all $y \in \{0, \dots, n-1\}$ if and only if

$$\Delta_0 \odot \Gamma_0 \geq (1-\varepsilon)\Delta_0.$$

In conclusion, we have the following SDP:

$$1 = \operatorname{Tr}_j\big(M_0^{(0)}\big) \qquad (0 \leq j < 2n) \tag{3.37}$$

$$\operatorname{Tr}_j\big(E_0 \odot M_0^{(t)}\big) = \begin{cases} \operatorname{Tr}_j\big(M_0^{(t+1)}\big) & t = 0, \dots, k-2 \\ \frac{1}{2}\operatorname{Tr}_j(\Gamma_0), & t = k-1 \end{cases} \qquad (0 \leq j < 2n) \tag{3.38}$$

$$\Gamma_0[0,0], \Gamma_0[n,n] \geq 1-\varepsilon \tag{3.39}$$

$$M_0^{(t)}, \dots, M_0^{(k-1)}, \Gamma_0 \in \mathbf{S}_+^{2n}. \tag{3.40}$$

This is equivalent to the claimed SDP, as can be seen by letting $M^{(t)} \coloneqq 2M_0^{(t)}$ for $0 \leq t < k$, and $M^{(k)} \coloneqq \Gamma_0$. □

In the next two propositions we show that the solution matrices $M^{(t)}$ can, without loss of

generality, be assumed to be of the form

$$M^{(t)} = \begin{pmatrix} A^{(t)} & (-1)^t A^{(t)} \\ (-1)^t A^{(t)} & A^{(t)} \end{pmatrix}.$$

Proposition 48. *Without loss of generality, the matrices $M^{(t)}$ in the above SDP (Equations (3.28) to (3.31)) may be assumed to have the form*

$$M^{(t)} = \begin{pmatrix} U^{(t)} & V^{(t)} \\ V^{(t)} & U^{(t)} \end{pmatrix},$$

where $U^{(t)} \in \mathbf{S}^n_+$ and $V^{(t)}$ is a symmetric $n \times n$ matrix. Moreover, for any such solution the main diagonals of $U^{(t)}$ and $(-1)^t V^{(t)}$ are identical.

Proof. We first establish the "moreover" part. Assume that the matrices $(M^{(t)})_{t=0}^k$ are a feasible solution of the prescribed form. Then by Equation (3.28) we have $1 = \operatorname{tr}(U^{(0)}) = \operatorname{tr}(V^{(0)})$. Equation (3.29) implies that $1 = \operatorname{tr}(U^{(t)}) = (-1)^t \operatorname{tr}(V^{(t)})$ for each $t \in \{0, \dots, k\}$. Since $M^{(t)}$ is positive semidefinite, we must have

$$-U^{(t)}[j,j] \leq V^{(t)}[j,j] \leq U^{(t)}[j,j] \qquad (\forall j \in \{0, \dots, n-1\}). \tag{3.41}$$

When t is odd, by the identity $\operatorname{tr}(U^{(t)}) = (-1)^t \operatorname{tr}(V^{(t)})$ the first inequality must be saturated. Similarly, when t is even the second inequality must be saturated. In other words, the identity $U^{(t)}[j,j] = (-1)^t V^{(t)}[j,j]$ holds as claimed.

We now prove the rest of the claim. Given an arbitrary solution $(\widehat{M}^{(t)})_{t=0}^k$ we construct another solution $(M^{(t)})_{t=0}^k$ that consists of matrices of the stated form. Suppose that the matrices $\widehat{M}^{(t)}$ are a feasible solution, and for each $t \in \{0, \dots, k\}$ write

$$\widehat{M}^{(t)} = \begin{pmatrix} A^{(t)} & B^{(t)} \\ C^{(t)} & D^{(t)} \end{pmatrix}.$$

Now define

$$\check{M}^{(t)} := \begin{pmatrix} D^{(t)} & C^{(t)} \\ B^{(t)} & A^{(t)} \end{pmatrix},$$

where we swap the rows and swap the columns of $\widehat{M}^{(t)}$. Clearly $\check{M}^{(t)} \succeq 0$ if and only if $\widehat{M}^{(t)} \succeq 0$. We claim that the set of matrices $\check{M}^{(t)}$ is another feasible solution. Given $j \in \{0, \dots, 2n-1\}$, define j' by letting $j' := j$ if $j \leq n$, and $j' := j - 2n$ otherwise. Then,

$$\mathrm{Tr}_j\big(\check{M}^{(t)}\big) = \mathrm{tr}_{j'}\big(D^{(t)}\big) + \mathrm{tr}_{j-n}\big(C^{(t)}\big) + \mathrm{tr}_{j'}\big(A^{(t)}\big) + \mathrm{tr}_{j-n}\big(B^{(t)}\big) = \mathrm{Tr}_j\big(\widehat{M}^{(t)}\big).$$

Similarly it is easy to see that $\mathrm{Tr}_j(E_0 \odot \check{M}^{(t)}) = \mathrm{Tr}_j(E_0 \odot \widehat{M}^{(t)})$ for $j \in \{0, \dots, 2n-1\}$. Thus, Equations (3.28) and (3.29) are satisfied by the matrices $\check{M}^{(t)}$. Equation (3.30) is also satisfied, since

$$\check{M}^{(k)}[0,0] = \widehat{M}^{(k)}[n,n] \quad \text{and} \quad \check{M}^{(k)}[n,n] = \widehat{M}^{(k)}[0,0].$$

We conclude that the matrices $\check{M}^{(t)}$ constitute another feasible solution. Then, by convexity so do the matrices $M^{(t)} := (\widehat{M}^{(t)} + \check{M}^{(t)})/2$, and these are of the desired form. □

Proposition 49. *Suppose that the SDP in Equations (3.28) to (3.31) has a solution $(\widehat{M}^{(t)})_{t=0}^{k}$ such that the matrices $\widehat{M}^{(t)}$ are of the form*

$$\widehat{M}^{(t)} = \begin{pmatrix} U^{(t)} & V^{(t)} \\ V^{(t)} & U^{(t)} \end{pmatrix}.$$

Then the matrices $M^{(t)} := \frac{1}{2}\big(\widehat{M}^{(t)} + (-1)^t \check{M}^{(t)}\big)$ give another solution where

$$\check{M}^{(t)} := \begin{pmatrix} V^{(t)} & U^{(t)} \\ U^{(t)} & V^{(t)} \end{pmatrix}.$$

Proof. First we show that the matrices $M^{(t)}$ are positive semidefinite. To see this, note that since $\widehat{M}^{(t)}$ is positive semidefinite, so is

$$\frac{1}{2}\begin{pmatrix} I_n & I_n \\ I_n & -I_n \end{pmatrix}^{\dagger} \widehat{M}^{(t)} \begin{pmatrix} I_n & I_n \\ I_n & -I_n \end{pmatrix} = \begin{pmatrix} U^{(t)} + V^{(t)} & 0 \\ 0 & U^{(t)} - V^{(t)} \end{pmatrix} \succeq 0.$$

In particular, the matrices $U^{(t)} + V^{(t)}$ and $U^{(t)} - V^{(t)}$ are both positive semidefinite. Thus so is $M^{(t)}$, being the Kronecker product of two positive semidefinite matrices:

$$M^{(t)} = \frac{1}{2}(U^{(t)} + (-1)^t V^{(t)}) \otimes \begin{pmatrix} 1 & (-1)^t \\ (-1)^t & 1 \end{pmatrix} \succeq 0.$$

It is easy to verify that $\operatorname{Tr}_j\big(\breve{M}^{(t)}\big) = \operatorname{Tr}_{j-n}\big(\widehat{M}^{(t)}\big)$ for $j \in \{0, \dots, 2n-1\}$. Analogously, $\operatorname{Tr}_j\big(E_0 \odot \breve{M}^{(t)}\big) = -\operatorname{Tr}_{j-n}\big(\widehat{M}^{(t)}\big)$. Then, by linearity

$$\operatorname{Tr}_j\big(E_0 \odot M^{(t)}\big) = \frac{1}{2}\Big(\operatorname{Tr}_j\big(E_0 \odot \widehat{M}^{(t)}\big) - (-1)^t \operatorname{Tr}_{j-n}\big(E_0 \odot \widehat{M}^{(t)}\big)\Big)$$

By Equation (3.29) this equals:

$$\begin{aligned} &= \frac{1}{2}\Big(\operatorname{Tr}_j\big(\widehat{M}^{(t+1)}\big) + (-1)^{t+1} \operatorname{Tr}_{j-n}\big(\widehat{M}^{(t+1)}\big)\Big) \\ &= \frac{1}{2}\Big(\operatorname{Tr}_j\big(\widehat{M}^{(t+1)}\big) + (-1)^{t+1} \operatorname{Tr}_j\big(\breve{M}^{(t+1)}\big)\Big) \\ &= \operatorname{Tr}_j\big(M^{(t+1)}\big). \end{aligned}$$

This shows that Equation (3.29) is satisfied by the matrices $M^{(t)}$. The initial constraint (Equation (3.28)) is also satisfied because $\operatorname{Tr}_j(\widehat{M}^{(0)}) = 2$ for all $j \in \{0, \dots, 2n-1\}$ by assumption, and therefore

$$\operatorname{Tr}_j\big(M^{(0)}\big) = \frac{1}{2}\Big(\operatorname{Tr}_j(\widehat{M}^{(0)}) + \operatorname{Tr}_j(\breve{M}^{(0)})\Big) = \frac{1}{2}\Big(\operatorname{Tr}_j(\widehat{M}^{(0)}) + \operatorname{Tr}_{j-n}(\widehat{M}^{(0)})\Big) = 2.$$

Finally, by Proposition 48 the main diagonal of $U^{(k)}$ is identical to the main diagonal of $(-1)^k V^{(k)}$, and consequently

$$\begin{aligned} M^{(k)}[0,0] &= \frac{1}{2}\Big(U^{(k)}[0,0] + (-1)^k V^{(k)}[0,0]\Big) \\ &= \frac{1}{2}\Big(U^{(k)}[0,0] + (-1)^{2k} U^{(k)}[0,0]\Big) \\ &= U^{(k)}[0,0] \geq 1 - \varepsilon. \end{aligned}$$

Analogously $M^{(k)}[n,n] \geq 1 - \varepsilon$, and the final constraint (Equation (3.30)) is satisfied as well. □

We can now present and prove our main result, which asserts that the class of translation-invariant quantum algorithms with no workspace (Definition 27) is optimal for the ordered search problem. We show this by reducing the Barnum-Saks-Szegedy SDP to our semidefinite programming characterization of translation-invariant algorithms for ordered search, which we restate below.

Theorem 33. *There exists an ε-error translation-invariant, k-query quantum algorithm for the n-element ordered search problem if and only if the following semidefinite program is feasible:*

$$Q^{(0)} \coloneqq J_n/n \tag{3.10}$$

$$\operatorname{tr}_j Q^{(t)} + (-1)^t \operatorname{tr}_{j-n} Q^{(t)} = \operatorname{tr}_j Q^{(t-1)} + (-1)^t \operatorname{tr}_{j-n} Q^{(t-1)} \qquad (0 < j < n,\ t \in [k]) \tag{3.11}$$

$$Q^{(k)}[0,0] \geq 1 - \varepsilon \tag{3.12}$$

$$\operatorname{tr} Q^{(t)} = 1 \qquad (t \in [k]) \tag{3.13}$$

$$Q^{(t)} \in \mathbf{S}^n_+ \qquad (t \in [k]) \tag{3.14}$$

where J_n denotes the $n \times n$ all-ones matrix.

We prove our main theorem by showing that for the ordered search problem the Barnum-Saks-Szegedy SDP is equivalent to the above semidefinite program.

Theorem 23. *Let $\varepsilon \geq 0$, and suppose that there is an ε-error quantum algorithm for the n-element ordered search problem using k queries. Then there exists an ε-error translation-invariant algorithm solving the same problem using the same number of queries.*

Proof. By Proposition 49, we can assume without loss of generality that the matrices $M^{(t)}$ in the SDP of Proposition 47 characterizing ε-error algorithms for the ordered search problem are of the form

$$M^{(t)} = \begin{pmatrix} A^{(t)} & (-1)^t A^{(t)} \\ (-1)^t A^{(t)} & A^{(t)} \end{pmatrix}.$$

Observe that $M^{(t)}$ is positive semidefinite if and only if $A^{(t)}$ is (in particular, $M^{(t)}$ is the tensor product of $A^{(t)}$ with a positive semidefinite matrix). Rewriting the constraints in terms of the submatrices $A^{(t)}$, we obtain the equivalent program

$$\operatorname{tr}_j A^{(0)} + \operatorname{tr}_{n-j} A^{(0)} = 1 \qquad (0 \leq j < n) \tag{3.42}$$

$$\operatorname{tr}_j A^{(t)} + (-1)^{t+1} \operatorname{tr}_{j-n} A^{(t)} = \operatorname{tr}_j A^{(t+1)} + (-1)^{t+1} \operatorname{tr}_{j-n} A^{(t+1)} \qquad (0 \leq j < n,\ 0 \leq t < k) \tag{3.43}$$

$$A^{(k)}[0,0] \geq 1 - \varepsilon \tag{3.44}$$

$$\operatorname{tr} A^{(t)} = 1 \qquad (0 \leq t \leq k) \tag{3.45}$$

$$A^{(t)} \in \mathbf{S}_+^n \qquad (0 \leq t \leq k) \tag{3.46}$$

In a proof deferred to the Appendix (Lemma 104), we show that Equation (3.42) has a unique positive semidefinite solution, namely $A^{(0)} = \frac{1}{n} J_n$. Replacing Equation (3.42) with this constant choice of $A^{(0)}$, we end up with the SDP stated in Theorem 33. Thus we have transformed the SDP characterizing all ε-error quantum algorithms for the ordered search problem into the SDP of Theorem 33 characterizing translation-invariant algorithms for this problem. □

As a consequence of this result, we see that the feasibility of ε-error k-query translation-invariant ordered search of an n-element list is monotonic in n.

Corollary 50. *If there is an ε-error k-query translation-invariant quantum algorithm for searching a list of length n, then there is also an ε-error k-query translation-invariant algorithm for a list of any length less than n.*

Proof. In general, algorithms for ordered search clearly satisfy this monotonicity, since an algorithm for a larger instance of Problem 25 (which is equivalent to Problem 26 as discussed in Section 3.2.2) can be applied to a smaller one by padding the input on the right with 1s. Since feasibility of translation-invariant algorithms is equivalent to feasibility of general algorithms, this property also holds for translation-invariant algorithms. □

Note that without establishing equivalence to general algorithms, this property is not obvious, since padding the input with 1s does not respect translation invariance. Indeed, for this reason Ref. [CLP07] was unable to definitively identify the largest list that could be searched exactly with a given number of queries. This result shows, for example, that the infeasibility of an exact translation-invariant algorithm searching a 606-element list with 4 queries (as shown in Ref. [CLP07]) implies the infeasibility of exactly searching an n-element list with 4 queries for any $n \geq 606$.

Another corollary of our result is that the algorithm of [BH07] can be made workspace-free. The implementation described by Ben-Or and Hassidim stores a list of $O(n)$ weights, one for each query index, whereas our result implies that there exists an algorithm with no workspace that achieves the same query complexity.

3.4 Finding Exact Translation-Invariant Algorithms with Linear Programming

In this section, we present a linear programming approach to identifying exact translation-invariant quantum algorithms for ordered search. Prior work in this direction utilized nonconvex optimization [FGGS99], gradient descent [BJL04], and semidefinite programming [CLP07]. In particular, Ref. [CLP07] used a semidefinite programming approach to exhibit an exact, 4-query translation-invariant algorithm for searching a 605-element list. The primary downside of such approaches is that the number of scalar variables an SDP solver needs to allocate to represent the matrices scales quadratically with the instance size n. In turn, n scales exponentially as a function of the number of queries k. This means that memory requirements quickly become prohibitive. Indeed, despite 18 years of improvements to computers since Ref. [CLP07], understanding the best 5-query algorithm still appears to be out of reach of SDP solvers.

While the exponential scaling of n with k is inevitable, we can improve the memory requirements of representing the program in Theorem 32 by considering a linear programming relaxation instead. This way, the memory required to represent the problem scales linearly with n, a quadratic improvement that allows us to go one step further than Ref. [CLP07]. Using this LP characterization, we show that the largest list searchable by an exact translation-invariant algorithm using $k = 5$ queries is of size $n = 7265$. By Theorem 23, this is the best one can do with any (not necessarily translation-invariant) 5-query algorithm.

We do this by exhibiting an explicit solution of the program in Theorem 32 for $n = 7265$, as well as an LP relaxation of this program that certifies infeasibility for $n = 7266$. By Corollary 50 we conclude that 7265 is the length of the longest list exactly searchable by a 5-query algorithm.

3.4.1 Eliminating Half of the Variables

Recall the polynomial characterization of [FGGS99] of exact translation-invariant algorithms:

Theorem 32 ([FGGS99]). *There exists an exact k-query translation-invariant quantum algorithm for the n-element ordered search problem if and only if the following program is feasible.*

Find symmetric, nonnegative Laurent polynomials $(q_t)_{t=0}^k$ of degree less than n s.t.

$$q_0 \equiv F_n \tag{3.6}$$

$$q_t(z) = q_{t-1}(z) \qquad (\forall z \text{ with } z^n = (-1)^t, 1 \le t \le k) \tag{3.7}$$

$$q_k \equiv 1 \tag{3.8}$$

$$\frac{1}{2\pi}\int_0^{2\pi} q_t(e^{i\theta})\, d\theta = 1 \qquad (0 \le t \le k). \tag{3.9}$$

Below we show that about half of the polynomial variables q_t in the above program are redundant. This observation allows for a more compact representation of the linear program (Definition 52) presented in the next section, leading to an improved runtime performance of our implementation.

Proposition 51. *For any $t \in [k-1]$ the constraint in Equation (3.7) involving q_t determines q_t in terms of q_{t-1} and q_{t+1}:*

$$a^{(t)} = \frac{1}{2}\left(a^{(t-1)} + a^{(t+1)}\right) + \frac{(-1)^t}{2} V_n \left(a^{(t-1)} - a^{(t+1)}\right), \tag{3.47}$$

where V_n is the $n \times n$ permutation matrix

$$V_n = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ \vdots & \vdots & \iddots & \iddots & 0 \\ 0 & 0 & 1 & \iddots & \vdots \\ 0 & 1 & 0 & \cdots & 0 \end{pmatrix}.$$

Proof. Recall that

$$q_t(z) = \frac{1}{2}\sum_{j=0}^{n-1} a_j^{(t)}(z^j + z^{-j}).$$

In Equation (3.7), we evaluate q_t at roots of either $z^n - 1$ or $z^n + 1$ depending on the parity of t. In other words, we want to evaluate q_t at powers of some primitive $2n$th root of unity ω. This can be expressed as a discrete Fourier transform:

$$\begin{pmatrix} q_t(\omega^0) \\ \vdots \\ q_t(\omega^{2n-1}) \end{pmatrix} = \frac{1}{2}F_{2n}\begin{pmatrix} 2a_0^{(t)} & a_1^{(t)} & \cdots & a_{n-1}^{(t)} & 0 & a_{n-1}^{(t)} & \cdots & a_1^{(t)} \end{pmatrix}^\top, \tag{3.48}$$

where F_{2n} is the $2n \times 2n$ discrete Fourier transform matrix.

Now let $\Xi := \mathrm{diag}(1, -1, 1, -1, \ldots, -1)$ be the $2n \times 2n$ matrix alternating 1 and -1 along the diagonal. When applied to a vector of evaluations of a polynomial at $\omega^0, \ldots, \omega^{2n-1}$, the matrix $\frac{1}{2}(I + \Xi)$ zeros out evaluations at roots of $z^n = -1$, while $\frac{1}{2}(I - \Xi)$ zeros out evaluations at roots of $z^n = +1$. It follows that, overloading notation by writing $q_t := \begin{pmatrix} q_t(\omega^0) & \ldots & q_t(\omega^{2n-1}) \end{pmatrix}^\top$, Equation (3.7) can be written

$$(I + (-1)^t\Xi)(q_t - q_{t-1}) = 0. \tag{3.49}$$

Observe that Ξ applies a phase ω^{nx} to the xth standard basis vector. Equivalently, $\chi := F_{2n}^{-1}\Xi F_{2n}$ acts by swapping the first and last n coordinates of a vector. Applying the inverse Fourier transform to both sides of Equation (3.49), we have

$$\begin{aligned} 0 &= F_{2n}^{-1}(I + (-1)^t\Xi)(q_t - q_{t-1}) \\ &= (I + (-1)^t\chi)F_{2n}^{-1}(q_t - q_{t-1}). \end{aligned}$$

Exchanging the two halves of the vector on the right-hand side of Equation (3.48) has the effect of

reversing $a_1^{(t)}, \ldots, a_{n-1}^{(t)}$, and exchanging $2a_0^{(t)}$ with 0. That is, Equation (3.49) is equivalent to

$$(I + (-1)^t V_n)(a^{(t)} - a^{(t-1)}) = 0. \tag{3.50}$$

Taking the difference of Equation (3.50) at t and $t+1$ yields

$$0 = (I + (-1)^t V_n)(a^{(t)} - a^{(t-1)}) - (I + (-1)^{t+1} V_n)(a^{(t+1)} - a^{(t)}),$$

so

$$a^{(t)} = \tfrac{1}{2}(a^{(t-1)} + a^{(t+1)}) + \tfrac{1}{2}(-1)^t V_n (a^{(t-1)} - a^{(t+1)})$$

as claimed. □

3.4.2 A Linear Programming Relaxation

We construct a linear program that mirrors Theorem 32, whose variables are the polynomial coefficients $a_j^{(t)}$ for $t = 0, 1, \ldots, k$ and $j = -(n-1), \ldots, n-1$. Except for polynomial nonnegativity, all the constraints in Theorem 32 are linear. Indeed, Equation (3.7) is linear since an evaluation of a polynomial is a linear function of the polynomial coefficients. Equation (3.9) is also linear, as it is equivalent to the constraint $a_0^{(t)} = 1$ (for each t).

To handle the nonnegativity constraint, we consider the following relaxation. Instead of requiring the polynomials to be nonnegative on their entire domain, we require that each polynomial exceeds a threshold β on some large but finite set of points G. We then let β be the objective of the LP to be maximized. Note that this procedure may not always result in positive polynomials. However, once a solution is found, one can certify the feasibility of the original program (Theorem 32) as follows. If the objective function is negative then the instance is unsolvable, since there are no polynomials that are all positive even at the points in G. On the other hand, if the produced polynomials are positive (which must be checked separately), then the instance is feasible.

We expand on this methodology below. First, we formally define and prove soundness of our construction.

Definition 52. Let G be a nonempty set of finitely many points on the unit circle. Let $\mathscr{L}(G)$ denote the following linear program, where the maximization is over symmetric real-valued Laurent polynomials $q_t(z) = \frac{1}{2}\sum_{j=0}^{n-1} a_j^{(t)}(z^j + z^{-j})$ for $1 \le t < k$:

$$\beta^* = \max \quad \beta \tag{3.51}$$

$$\text{s.t.} \quad q_0 \equiv F_n \tag{3.52}$$

$$q_t(z) = q_{t-1}(z) \qquad (\forall z \text{ with } z^n = (-1)^t, \forall t \in [k]) \tag{3.53}$$

$$q_k \equiv 1 \tag{3.54}$$

$$a_0^{(t)} = 2 \qquad (\forall t = 0, \dots, k) \tag{3.55}$$

$$q_t(z) \ge \beta \qquad (\forall z \in G, \forall t = 0, \dots, k) \tag{3.56}$$

Our first observation concerns the feasibility of $\mathscr{L}(G)$.

Corollary 53. *For any finite subset G of the unit circle, the program $\mathscr{L}(G)$ is feasible.*

Proof. Suppose k is even. Consider any assignment of the polynomials $q_2, q_4, \dots, q_{k-2}$ such that the respective constraints in Equation (3.55) are satisfied. By Proposition 51 the polynomials $q_1, q_3, \dots, q_{k-1}$ are determined through Equation (3.47) and the equality constraints in Equation (3.53) are all satisfied. Let β be the minimal evaluation of any one of the polynomials $q_0, \dots, q_k$ on the set G, which is finite since a Laurent polynomial can only have a pole at the origin. This gives a feasible solution.

The case of odd k requires more care. Suppose k is odd, and consider the constraint in Equation (3.53) for $t = 1$. By the proof of Proposition 51 this equation is equivalent to

$$(I - V_n)(a^{(1)} - a^{(0)}) = 0.$$

Since the matrix $I - V_n$ is singular, this equation has infinitely many solutions for $a^{(1)}$. Since the

above equation imposes no constraints on the coefficient $a_0^{(1)}$, we can fix a solution $a^{(1)}$ satisfying $a_0^{(1)} = 2$. Next, assign some values to the coefficients of the polynomials $q_3, q_5, \ldots, q_{k-2}$ such that the respective constraints in Equation (3.55) are satisfied. By Proposition 51 our choices determine the remaining polynomials $(q_2, q_4, \ldots, q_{k-1})$, and the equality constraints in Equation (3.53) are all satisfied. As before, β can be chosen to be the minimal value of the polynomials on the set G. □

The linear program $\mathscr{L}(G)$ provides an alternative way of certifying that there is no exact k-query translation-invariant algorithm for searching an n-element list. Indeed, if $\mathscr{L}(G)$ has a negative optimal objective value $\beta^* < 0$ then there do not exist nonnegative polynomials $q_0, \ldots, q_k$ satisfying the conditions of Theorem 32. The converse, however, is not true: $\mathscr{L}(G)$ having a positive objective value $\beta^* > 0$ does not imply that there exists an exact k-query translation-invariant algorithm for searching an n-element list, because the polynomial could be negative at points outside G.

We can overcome this limitation as follows. Suppose that for some choice of G, a solution consisting of the polynomials $q_0, \ldots, q_k$ attains a positive objective value $\beta > 0$. In the Chebyshev representation, the polynomials q_t can be represented as functions from $\mathbb{R} \to \mathbb{R}$ (see for example [Tre96, Section 8.3]). In this representation, the interval $[-1, 1]$ corresponds to the inputs on the unit circle. For each q_t in the Chebyshev basis, by considering the derivative q_t' we can check each of the local extrema of $q_t(x)$ within the interval $[-1, 1]$ for positivity, i.e., we can verify that $q_t'(x) = 0 \implies q_t(x) > 0$. If we do the same on the boundaries, this certifies that the polynomial is nonnegative on the unit circle. In this way, any solution returned by the linear program $\mathscr{L}(G)$ can be checked for positivity.

3.4.3 An Iterative Approach

As discussed above, the program $\mathscr{L}(G)$ can be used to certify that there is no exact, k-query translation-invariant quantum algorithm for searching an n-element list. However, this will only succeed for an appropriate choice of the set G. To this end, we use an iterative algorithm that heuristically constructs a sequence of increasing subsets $G_0 \subset G_1 \subset \cdots$ by solving the the

corresponding sequence $\mathscr{L}(G_0), \mathscr{L}(G_1), \ldots$ of linear programs. In this process, the set of constraints keeps increasing, and the feasible sets keep shrinking. While the process is not guaranteed to terminate, in practice we find that it does after a reasonable number of iterations. Algorithm 1 gives pseudocode for this procedure.

Algorithm 1 Finding exact translation-invariant algorithms

Let $G_0 \subset \{z \in \mathbb{C} : |z| = 1\}$ be such that $\mathscr{L}(G_0)$ is bounded
for $i = 0, 1, \ldots$ **do**
    Let β_i^* be the optimal value of $\mathscr{L}(G_i)$ and $q_0, \ldots, q_k$ be polynomials attaining that value
    if $\beta_i^* < 0$ **then return** G_i
    Let M be the set of minima[1] of $q_0, \ldots, q_k$ that are negative
    if $M = \emptyset$ **then**
        return $q_0, \ldots, q_k$
    else
        Let $G_{l+1} := G_l \cup M$

Proposition 54. *If Algorithm 1 returns some set G on Algorithm 1, then there does not exist an exact k-query translation-invariant algorithm for searching an n-element list. Conversely, if polynomials $q_0, \ldots, q_k$ are returned on Algorithm 1 then these polynomials witness the existence of an exact k-query translation-invariant algorithm for the n-element ordered search problem.*

Proof. Suppose some set G is returned on Algorithm 1. This occurs only if the optimal value of the linear program $\mathscr{L}(G)$ is negative. This in turn implies that there are no nonnegative polynomials satisfying the constraints in Equations (3.52) to (3.56) of the linear program $\mathscr{L}(G)$. Since $\mathscr{L}(G)$ is a relaxation of the program in Theorem 32, it follows that there does not exist an exact k-query translation-invariant algorithm for searching an n-element list.

Now suppose that polynomials $q_0, \ldots, q_k$ are returned on Algorithm 1 of the algorithm. These polynomials constitute a feasible solution of $\mathscr{L}(G)$ for some set G, so they satisfy Equations (3.52) to (3.55). The constraints in Equations (3.52) to (3.55) imply Equations (3.6) to (3.9), respectively. By construction the polynomials are symmetric and real-valued. They are also nonnegative (by

[1]More precisely, let M be the set of complex numbers on the unit circle corresponding to the minima of the Chebyshev representations of $q_0, \ldots, q_k$ restricted to $[-1, 1]$ that are negative, as well as any point corresponding to the boundary of $[-1, 1]$ at which a Chebyshev polynomial is negative.

Algorithm 1), as their Chebyshev representations are nonnegative in $[-1, 1]$. Thus, the polynomials $q_0, \ldots, q_k$ are a feasible solution for the program of Theorem 32, witnessing that there exists a k-query exact translation-invariant algorithm for the n-element ordered search problem. □

3.4.4 Computational Results

To identify the maximal value of n for which there exists an exact, translation-invariant algorithm for the n-element ordered search problem using $k = 5$ queries, we implemented Algorithm 1 and wrapped it with a binary search over n. Our program was built in Python using the CVXPY and MOSEK libraries for modeling and solving the linear program $\mathscr{L}(G)$ [DB16; AVDB18; ApS25].

Our algorithm ran on a machine with an Intel Xeon Silver 4216 2.10 GHz processor and 128 GB RAM. Our approach needed some intermittent manual oversight, primarily due to the scheduling limitations of the machine. In total the search took about a month (including downtime) and identified $n = 7265$ as the maximal instance size searchable with 5 queries. Our program returned both the witnessing polynomials $q_0, \ldots, q_5$ (depicted in Figure 3.1) for $n = 7265$, as well as a set G for which the linear program $\mathscr{L}(G)$ (with $n = 7266$) has a negative optimal value $\beta^* < 0$, certifying the nonexistence of 5-query algorithms for searching a list of 7266 elements. By Corollary 50, it follows that there is also no 5-query algorithm for searching a list of any length more than 7266.

Our code and data are available on GitHub at `https://github.com/jacarolan/ordered_search_public/releases/tag/v0.1.1`.

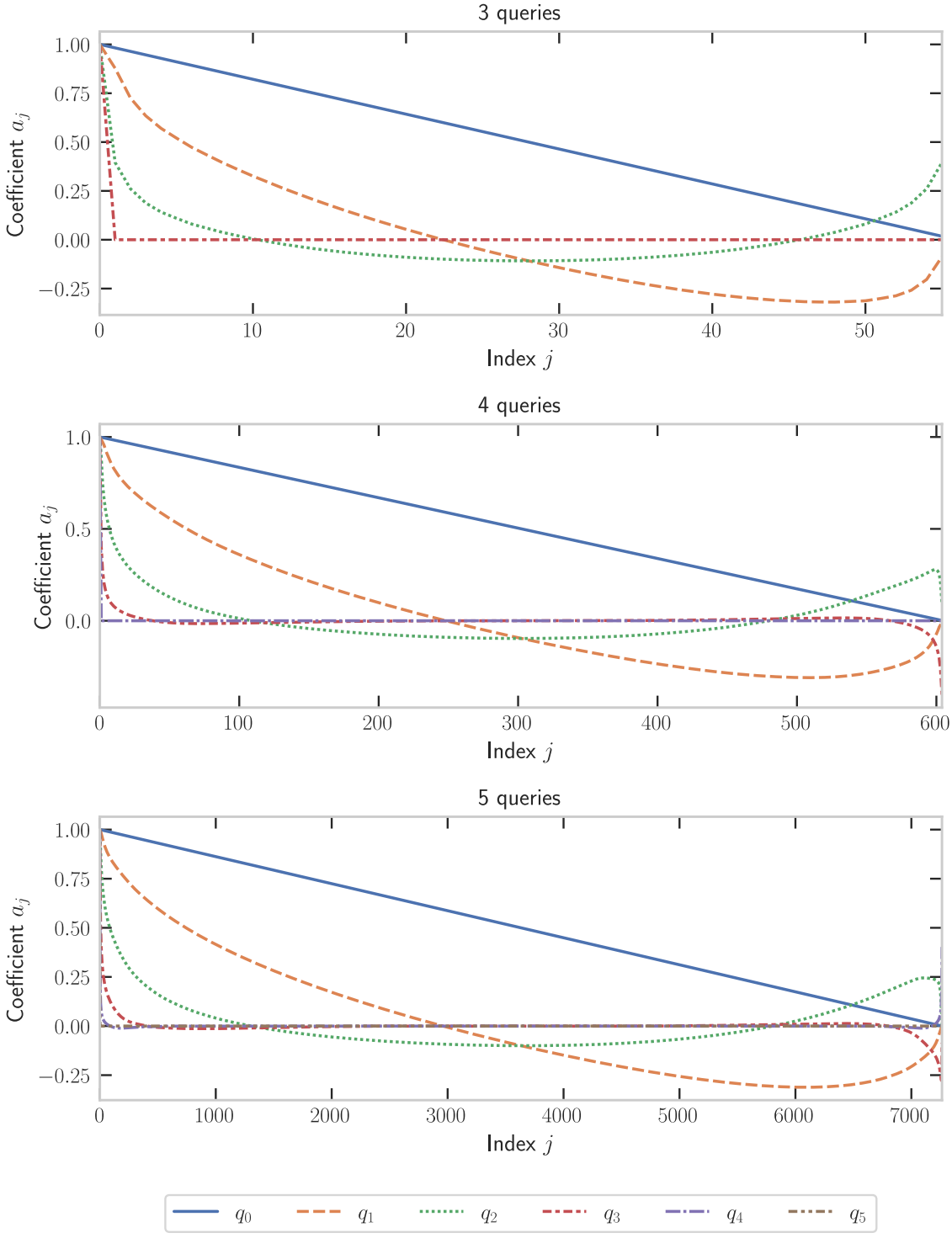


Figure 3.1: Coefficients of the Laurent polynomials corresponding to exact 3-, 4-, and 5-query algorithms. The corresponding instances sizes are 56, 605, and 7265, respectively. For each polynomial $q_t(z) = \sum_{j=-n}^{n} a_j^{(t)} z^j$ we only plot the coefficients $a_j^{(t)}$ for nonnegative indices j, as $a_j^{(t)} = a_{-j}^{(t)}$ for each of these polynomials.

3.5 Future Work

In this chapter we showed that translation-invariant algorithms are optimal among all quantum query algorithms for the ordered search problem, for all choices of the error parameter $\varepsilon \geq 0$. We also improved the best upper bound for exact algorithms from (approximately) $0.433 \log_2 n$ to $0.390 \log_2 n$ by exhibiting a 5-query algorithm that can search a list of 7265 elements exactly.

We mention some potential avenues for further research. First, while we narrowed the gap between upper and lower bounds for ordered search, the precise quantum query complexity of ordered search remains unknown. One could hope to give a better algorithm, or prove a stronger lower bound. On the algorithmic side, our results show that translation invariance can be taken without loss of generality. This motivates developing tools for better understanding translation-invariant algorithms. Alternatively, one could study bounded- or zero-error[2] algorithms through the lens of translation invariance. For instance, one could find the translation-invariant analog of the algorithm developed in [BH07].

On the lower bounds side, we know that the adversary method cannot yield better lower bounds than what is already known [CL08]. However, the error dependence of quantum lower bounds for ordered search has an unusual scaling. In particular, for general algorithms with success probability less than 0.89, the best known lower bound is $\frac{1}{12} \log_2 n$ [Amb99], which is a constant factor weaker than the $\frac{1}{\pi} \ln n$ bound for exact algorithms. Classically, a $\log_2 n - O(1)$ bound applies to any algorithm with constant success probability. It would be interesting to sharpen the error dependence of quantum lower bounds for ordered search.

[2] Zero error here refers to Las Vegas algorithms, which succeed with certainty after a randomized number of queries.

Chapter 4

The Rational Degree of Boolean Functions

Chapter summary. We study a natural complexity measure of Boolean functions known as the rational degree. Denoted $\mathrm{rdeg}(f)$, it is the minimal degree of a rational function that is equal to f on the Boolean hypercube. For total functions f, it is conjectured that $\mathrm{rdeg}(f)$ is polynomially related to the degree of f, $\deg(f)$. Towards this conjecture, we show that:

- Symmetric functions have rational degree at least $\Omega(\deg(f))$ and unate (a generalization of monotone) functions have rational degree at least $\sqrt{\deg(f)}$. We observe that both of these lower bounds are asymptotically tight.
- Read-once AC and TC formulae have rational degree at least $\Omega(\sqrt{\deg(f)})$. If these formulae contain parity gates, we show a lower bound of $\Omega(\deg(f)^{1/2d})$, where d is the depth.
- Almost every Boolean function on n variables has rational degree at least $n/2 - O(\sqrt{n})$.

In contrast, we exhibit partial functions that witness constant vs. $\Omega(n)$ separations between the rational and approximate degrees in both directions. As a consequence, we show that for quantum computers, postselection and bounded-error are incomparable resources in the black-box model.

In addition, we show AND and OR composition lemmas for the rational degree and exhibit new polynomial separations between the rational degree and other well-studied complexity measures, such as sensitivity and spectral sensitivity.

4.1 Introduction

Starting with the seminal work of Minsky and Papert [MP69], a long line of research has sought to relate various measures of Boolean function complexity. In [NS94], Nisan and Szegedy proved that the deterministic decision tree complexity $\mathrm{D}(f)$ of a Boolean function f is polynomially related to its degree $\deg(f)$. The same paper posed two open questions. One of them conjectures that the sensitivity and block sensitivity of a Boolean function are polynomially related. This conjecture was recently proven in a breakthrough by Huang [Hua19]. Huang's result brought sensitivity into a "happy flock" of complexity measures of total Boolean functions that are all polynomially related: sensitivity, degree, approximate degree, and notions of query complexity.

Another natural measure of Boolean function complexity is the minimal degree of the ratio of two polynomials which represents the function exactly, called the *rational degree* (denoted rdeg). However, rdeg is *not* known to be either polynomially related to or separated from the complexity measures mentioned above. In fact, this was the other open question posed over 30 years ago in the paper of Nisan and Szegedy (via personal communication with Fortnow) [NS94]. This question was reiterated by Aaronson *et al.* [ABK+21] yet very little progress has been made toward its resolution.

Question 55 (Fortnow [NS94])**.** *Does there exist $c > 1$ such that for all total Boolean functions f, $\deg(f) \leq O(\mathrm{rdeg}(f)^c)$?*

One of the motivations for Fortnow's question was complexity-theoretic: is the intersection of $\mathsf{C_{=}P}$ and $\mathsf{coC_{=}P}$ strictly contained in PP with respect to an oracle [For03]? $\mathsf{C_{=}P}$ and $\mathsf{coC_{=}P}$ are "counting classes" [AK] which we define later, and rational degree corresponds to the black-box version of their intersection.

The rational degree is also related to quantum query complexity. In particular, de Wolf defined the *nondeterministic degree* $\mathrm{ndeg}(f)$ of a Boolean function f as the minimal degree of a polynomial whose zero set is precisely the set of inputs on which f evaluates to false [dW00], and related it to the rational degree through the identity $\mathrm{rdeg}(f) = \max\{\mathrm{ndeg}(f), \mathrm{ndeg}(\bar{f})\}$. de Wolf also proved

that the nondeterministic degree $\mathrm{ndeg}(f)$ equals the *nondeterministic quantum query complexity*.

In the same manuscript, de Wolf stated the following conjecture which, together with the inequality $\deg(f) \leq \mathrm{D}(f)$, would resolve Fortnow's question in the affirmative with $c = 2$.

Conjecture 56 (de Wolf [dWoo])**.** *For all Boolean functions f,*

$$\mathrm{D}(f) \leq O(\mathrm{ndeg}(f)\,\mathrm{ndeg}(\bar{f})).$$

Mahadev and de Wolf showed [MdW14] an even tighter connection between the rational degree and quantum query complexity: denoting by $\mathrm{rdeg}_\epsilon(f)$ the minimum degree of a rational polynomial that ϵ-approximates f pointwise, they showed that $\mathrm{rdeg}_\epsilon(f)$ equals (up to a constant factor) the query complexity of an ε-error quantum algorithm that computes f with *postselection*, an operation that allows for projection onto an efficiently computable set of basis states. For partial functions, it can be shown that the rational degree gives a lower bound on the query complexity of algorithms with postselection, though the opposite direction is not known to be true. Furthermore, this result extends to the case of $\varepsilon = 0$, the so-called "exact" setting.

4.1.1 Our Results

We prove lower bounds on the rational degree for certain classes of total Boolean functions. We summarize our results, which are also depicted in Figure 4.1:

Sec. 4.3.1 For symmetric functions we show that $\mathrm{rdeg}(f) \geq (\deg(f) + 1)/3$. This lower bound is tight even in its constant factor, as witnessed by the "middle-third" (MT_n) function. Our technique generalizes to give a $\deg(f)/4$ lower bound for (possibly partial) functions which are constant on many Hamming slices.

Sec. 4.3.2 For the class of unate functions, which is a generalization of monotone functions, we prove that $\mathrm{rdeg}(f) = \mathrm{s}(f) \geq \sqrt{\deg(f)}$. This is also tight as witnessed by the $\mathsf{AND}_n \circ \mathsf{OR}_n$ function.

	Lower Bound	Upper Bound	
Symmetric Functions	$(\deg(f)+1)/3$	$\deg(f)(1+o(1))/3$	(MT_n)
Unate Functions	$\sqrt{\deg(f)}$	$\sqrt{\deg(f)}$	$(\mathsf{AND}_n \circ \mathsf{OR}_n)$
Read-once AC/TC	$\Omega(\sqrt{\deg(f)})$	$\sqrt{\deg(f)}$	$(\mathsf{AND}_n \circ \mathsf{OR}_n)$
Read-once depth-d AC[$\oplus$]	$\Omega(\deg(f)^{1/d})$	$\sqrt{\deg(f)}$	$(\mathsf{AND}_n \circ \mathsf{OR}_n)$
Read-once depth-d TC[$\oplus$]	$\Omega(\deg(f)^{1/2d})$	$\sqrt{\deg(f)}$	$(\mathsf{AND}_n \circ \mathsf{OR}_n)$
Almost all $f\colon \{0,1\}^n \to \{0,1\}$	$n/2 - O(\sqrt{n})$	n	

Figure 4.1: A table summarizing our lower bounds on the rational degree of total functions. The third column gives an example of a function that demonstrates the asymptotic tightness of our lower bound, where applicable.

Sec. 4.3.3 We employ the lower bound on symmetric and unate functions to show that for f computable by read-once depth-d AC and TC formulae,[1] $\mathrm{rdeg}(f) \geq \Omega(\sqrt{\deg(f)})$. This is tight, as witnessed by the $\mathsf{AND}_n \circ \mathsf{OR}_n$ function. For read-once TC formulae, we also give a standalone proof for a somewhat strengthened result in Section C.2. If we include parity gates, the lower bounds become $\Omega(\deg(f)^{1/d})$ for read-once AC[$\oplus$] and $\Omega(\deg(f)^{1/2d})$ for read-once TC[$\oplus$].

Sec. 4.3.4 Our final lower bound on total functions is extremal: we prove that almost all Boolean functions on n bits have rational degree at least $n/2 - O(\sqrt{n})$.

Next, we prove that the nondeterministic degree composes exactly with respect to AND. Leveraging this result, we:

Sec. 4.4 Establish AND and OR composition lemma for the rational degree.

Sec. 4.5 Exhibit an asymptotic separation between the rational degree and the spectral sensitivity of degree $3/2$.

In contrast to total functions, we exhibit a constant vs. $\Omega(n)$ separation between the rational and approximate degrees for partial functions in both directions.

[1] Read-once AC (resp. TC) are read-once formulae consisting of unbounded fan-in $\{\vee, \wedge, \neg\}$ gates (resp. arbitrary linear threshold gates). AC[$\oplus$] and TC[$\oplus$] refer to the classes with the addition of unbounded fan-in parity gates.

Sec. 4.6.1 We give a partial function $\textsc{MajOrNone}_n$ on n bits with constant quantum query complexity yet rational degree $\Omega(n)$. As a result, $\textsc{MajOrNone}_n$ has constant approximate degree and $\Omega(n)$ exact postselected quantum query complexity.

Sec. 4.6.2 On the other hand, we give a partial function $\textsc{Imbalance}_n$ on n bits with approximate degree $\Omega(n)$ yet constant rational degree. As a result, $\textsc{Imbalance}_n$ has constant exact postselected quantum query complexity and quantum query complexity $\Omega(n)$.

Now, employing the framework of standard complexity results such as [FSS81], we argue that postselection and bounded error are incomparable resources in the black-box setting. To formalise this, we define PostEQP as the class of decision problems which can be decided deterministically in polynomial time by quantum computers with access to postselection. In particular, there exist bidirectional oracle separations between PostEQP and BQP. Formally, combining the results of Corollaries 89 and 93 we get the following statement.

Corollary 57. *There exist oracles O_1 and O_2 such that*

$$\mathsf{BQP}^{O_1} \not\subseteq \mathsf{PostEQP}^{O_1} \quad \textit{yet} \quad \mathsf{PostEQP}^{O_2} \not\subseteq \mathsf{BQP}^{O_2}.$$

These complexity-theoretic consequences are summarized in Figure 4.2. As the figure illustrates, these are the strongest possible separations in the black-box model.

In addition to these consequences for PostEQP, our lower bound also shows that the answer to the relativization question that motivated Fortnow's question about the rational degree[2] is negative. That is, we show there is an oracle relative to which $\mathsf{C_=P} \cap \mathsf{coC_=P}$ is strictly contained in PP. We do this by exhibiting an oracle relative to which RP is not contained in PostEQP. Since $\mathsf{RP} \subseteq \mathsf{PP}$ and $\mathsf{C_=P} \cap \mathsf{coC_=P} \subseteq \mathsf{PostEQP}$ [dB15], the claim follows. The class $\mathsf{C_=P}$ is the set of languages decidable by an NP machine such that if the string is in the language, the number of accepting paths is *exactly* equal to the number of rejecting paths. Finally, to contextualize the

[2]We leave open whether rational degree is polynomially related to the degree for total functions.

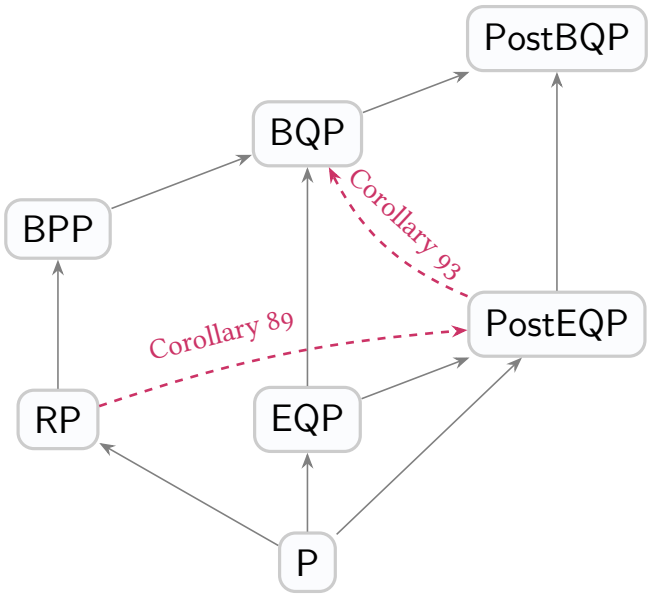


Figure 4.2: Relevant complexity classes. We are able to obtain the strongest possible oracle separations in this picture. An arrow $\mathsf{A} \to \mathsf{B}$ means $\mathsf{A} \subseteq \mathsf{B}$ relative to all oracles. A dashed arrow $\mathsf{A} \dashrightarrow \mathsf{B}$ means $\mathsf{A} \not\subseteq \mathsf{B}$ relative to some oracle.

power of PostEQP, we provide strong evidence that exact postselection can offer advantage over efficient classical computation, even in the nonrelativized setting.

Sec. 4.6.3 We show that PostEQP contains $\mathsf{NP} \cap \mathsf{coNP}$. We remark that $\mathsf{NP} \cap \mathsf{coNP}$ is not even believed to be contained in BPP.

Table 4.1 is a reproduction of [ABK+21, Table 1] showing the known relationships between various Boolean complexity measures. We extended this table with a column corresponding to the rational degree that shows the separations established in our work, and updated some of the other cells to reflect more recent developments.

Table 4.1: Best known separations between complexity measures for total functions (Reproduced from [ABK+21, Table 1] and updated with recent developments [BBG+22] as well as our results)

	D	R_0	R	C	RC	bs	s	λ	Q_E	deg	Q	$\widetilde{deg}$	rdeg
D		2, 2 [ABB+17]	2, 3 [ABB+17]	2, 2 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	3, 6 [BHT17]	4, 6 [ABB+17]	2, 3 [ABB+17]	2, 3 [GPW18]	4, 4 [ABB+17]	4, 4 [ABB+17]	2, ? $\wedge \circ \overline{\text{EH}}$
R_0	1, 1 $\oplus$		2, 2 [ABB+17]	2, 2 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	3, 6 [BHT17]	4, 6 [ABB+17]	2, 3 [ABB+17]	2, 3 [GJPW18]	3, 4 [ABB+17]	4, 4 [ABB+17]	2, ? $\wedge \circ \overline{\text{EH}}$
R	1, 1 $\oplus$	1, 1 $\oplus$		2, 2 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	3, 6 [BHT17]	4, 6 [ABB+17]	$\frac{3}{2}$, 3 [ABB+17]	2, 3 [GJPW18]	3, 4 [BS21] [SSW23]	4, 4 [ABB+17]	2, ? $\wedge \circ \overline{\text{EH}}$
C	1, 1 $\oplus$	1, 1 $\oplus$	1, 2 $\oplus$		2, 2 [GSS16]	2, 2 [GSS16]	3, 5 [BBG+22]	4, 6 [BBG+22]	1.15, 3 [Amb13]	2, 3 [BBG+22]	2, 4 $\wedge$	3, 4 [BBG+22]	2, ? $\wedge \circ \overline{\text{EH}}$
RC	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$		$\frac{3}{2}$, 2 [GSS16]	2, 4 [Rub95]	2, 4 $\wedge$	1.15, 2 [Amb13]	1.63, 2 [NW95]	2, 2 $\wedge$	2, 2 $\wedge$	2, ? $\wedge \circ \overline{\text{EH}}$
bs	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$		2, 4 [Rub95]	2, 4 $\wedge$	1.15, 2 [Amb13]	1.63, 2 [NW95]	2, 2 $\wedge$	2, 2 $\wedge$	2, ? $\wedge \circ \overline{\text{EH}}$
s	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$		2, 2 $\wedge$	1.15, 2 [Amb13]	1.63, 2 [NW95]	2, 2 $\wedge$	2, 2 $\wedge$	2, ? $\wedge \circ \overline{\text{EH}}$
λ	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$		1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	1.5, ? $\wedge \circ \overline{\text{EH}}$
Q_E	1, 1 $\oplus$	1.33, 2 $\bar{\wedge}$-tree	1.33, 3 $\bar{\wedge}$-tree	2, 2 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	2, 3 $\wedge \circ \vee$	3, 6 [BHT17]	4, 6 [ABK16]		2, 3 [ABK16]	2, 4 $\wedge$	4, 4 [ABK16]	2, ? $\wedge \circ \overline{\text{EH}}$
deg	1, 1 $\oplus$	1.33, 2 $\bar{\wedge}$-tree	1.33, 2 $\bar{\wedge}$-tree	2, 2 $\wedge \circ \vee$	2, 2 $\wedge \circ \vee$	2, 2 $\wedge \circ \vee$	2, 2 $\wedge \circ \vee$	2, 2 $\wedge$	1, 1 $\oplus$		2, 2 $\wedge$	2, 2 $\wedge$	2, ? $\wedge \circ \overline{\text{EH}}$
Q	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	2, 2 [ABK16]	2, 3 [ABK16]	2, 3 [ABK16]	3, 6 [BHT17]	4, 6 [ABK16]	1, 1 $\oplus$	2, 3 [ABK16]		4, 4 [ABK16]	1.5, ? $\wedge \circ \overline{\text{EH}}$
$\widetilde{deg}$	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$	2, 2 [BT20]	2, 2 [BT20]	2, 2 [BT20]	2, 2 [BT20]	2, 2 [BT20]	1, 1 $\oplus$	1, 1 $\oplus$	1, 1 $\oplus$		1.5, ? $\wedge \circ \overline{\text{EH}}$

- An entry a, b in the row M_1 and column M_2 roughly means that there exists a function g with $M_1(g) \geq M_2(g)^{a-o(1)}$, and for all total functions f, $M_1(f) \leq M_2(f)^{b+o(1)}$. For example, the $3, 4$ entry at row R and column Q means that the maximum possible separation between R and Q is at least cubic and at most quartic.
- The second row of each cell contains an example of a function that achieves the separation (or a citation to an example), where $\oplus$ = PARITY, $\wedge$ = AND, $\vee$ = OR, $\wedge \circ \vee$ = AND-OR, $\bar{\wedge}$-tree is the balanced NAND-tree function, and $\overline{\text{EH}}$ is the negation of the exact-half function defined in Section 4.5.
- Cells have a white background if the relationship is optimal and a gray background otherwise.
- The entries with green background in the column for rdeg are contributions of this work (see Proposition 86). Getting any polynomial upper bound (replacing "?" on the right with a constant) in any cell would resolve Question 55.

4.2 Preliminaries

In this section we review some of the notation and definitions used in this chapter. We denote by $[n]$ the set $\{1, 2, ..., n\}$. Given a function $f\colon S \to \mathbb{R}$ we denote by $\|f\|_1$ its 1-norm, $\|f\|_1 = \sum_{x \in S} |f(x)|$. For a bitstring $x \in \{0,1\}^n$, we denote by $|x|$ the Hamming weight of x: the number of indices equal to 1. If $x \in \{-1,1\}^n$ the Hamming weight is the number of bits that equal -1. For $x \in \{0,1\}^n$, $x^{i \to b}$ is the bitstring obtained by replacing the ith bit of x with b. Given $x \in \{0,1\}^n$ and some $S \subset [n]$ we denote by x^S the bitstring obtained by flipping all bits of x indexed by S.

4.2.1 Boolean Functions

A (total) Boolean function is any function $f\colon \Sigma^n \to \Sigma$ where Σ is some two-element set. We will refer to the set Σ^n as the Boolean hypercube. We will primarily work with the sets $\Sigma = \{0,1\}$ and $\Sigma = \{-1,1\}$, often swapping between them to make our analysis more presentable. While not all Boolean complexity measures are left invariant by this change of representation, all of the measures considered in this chapter are preserved.

We also consider restrictions of Boolean functions to proper subsets of the Boolean cube $D \subset \Sigma^n$. We refer to such functions $f\colon D \to \Sigma$ as *partial* Boolean functions. Given a Boolean function f, we denote its negation by $\bar{f}$.

We define an inner product on the space of functions $f\colon \{-1,1\}^n \to \mathbb{R}$ by letting

$$\langle f, g \rangle = 2^{-n} \sum_{x \in \{-1,1\}^n} f(x)g(x).$$

For each $S \subseteq [n]$ we define the *character function* χ_S *on* S as $\chi_S(x) = \prod_{i \in S} x_i$. The character functions χ_S form an orthonormal basis with respect to the above inner product. Thus each function over $\{-1,1\}^n$ can be uniquely expressed via its *Fourier representation*:

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \cdot \chi_S(x).$$

We refer to $\widehat{f}(S) = \langle f, \chi_S \rangle$ as the *Fourier coefficient of f at S*. We say an input $i \in [n]$ is *relevant* for f if x_i appears in the Fourier expansion of f. In other words, i is relevant for f if there is some x such that $f(x^{i \to 1}) \neq f(x^{i \to -1})$.

In this work, we consider some special classes of Boolean functions. For example, we say a function f is *symmetric* if it remains invariant under any permutation of the input variables. A Boolean function $f \colon \{0,1\}^n \to \{0,1\}$ is said to be *monotone* if $\forall x, y \in \{0,1\}^n$, $x \leq y$ implies $f(x) \leq f(y)$, where $x \leq y$ is taken pointwise. Finally, the notion of unateness generalizes monotonicity. We say a Boolean function $f \colon \{0,1\}^n \to \{0,1\}$ is *unate* in the coordinate i if either:

(1) For every $x \in \{0,1\}^n$, $f(x^{i \to 0}) \leq f(x^{i \to 1})$.

(2) For every $x \in \{0,1\}^n$, $f(x^{i \to 0}) \geq f(x^{i \to 1})$.

The function f itself is said to be unate if it is unate in every coordinate $i \in [n]$. Unate functions are exactly those functions that become monotone under negation of some subset of the input variables. While these properties were defined over the domain $\{0,1\}^n$, they have clear analogs for functions over the domain $\{-1,1\}^n$. As mentioned earlier, we will switch between the two bases to aid the presentation of our results.

We assume basic familiarity with notions in complexity such as Boolean formulae, but review some terminology. An AC formula is a formula with $\neg$ gates and unbounded fan-in $\vee$ and $\wedge$ gates. A TC formula is a formula where the gates are arbitrary linear threshold functions of unbounded fan-in. $\mathsf{AC}[\oplus]$ and $\mathsf{TC}[\oplus]$ are defined by allowing unbounded fan-in parity gates to AC and TC, respectively. Finally, a formula is *read-once* if every variable feeds into at most one gate. Recall that every gate in a formula has fan-out at most 1.

For a comprehensive introduction to the analysis of Boolean functions see [Sak93; ODo14].

4.2.2 Polynomials

As described previously, each real-valued function on the Boolean cube is represented uniquely by a formal multilinear polynomial. We define the *degree* of f as $\deg(f) = \max\{|S| : \widehat{f}(S) \neq 0\}$. We extend this notion to polynomials that pointwise approximate f.

Definition 58. Let $D \subseteq \{-1,1\}^n$ and $f\colon D \to \{-1,1\}$. A function $p : \{-1,1\}^n \to \mathbb{R}$ is said to ε-approximate f if:

(1) $\forall x \in D, |p(x) - f(x)| \leq \varepsilon$.

(2) $\forall x \in \{-1,1\}^n, |p(x)| \leq 1$.

The *ϵ-approximate degree* of f, denoted $\widetilde{\deg}_\epsilon(f)$, is defined as the minimum degree of any polynomial that ϵ-approximates f. If ε is not specified, we define the *approximate degree* of f, denoted $\widetilde{\deg}(f)$, to be $\widetilde{\deg}_{1/3}(f)$.

In this chapter, we are primarily concerned with representations of f via rational polynomials. This gives rise to a measure known as *rational degree*, which is formally defined as follows.

Definition 59. Let $D \subseteq \{-1,1\}^n$ and $f\colon D \to \{-1,1\}$. If $p, q\colon D \to \mathbb{R}$ are such that

$$\forall x \in D, \quad \left|f(x) - \frac{p(x)}{q(x)}\right| \leq \epsilon,$$

we say that p/q is an ϵ-approximate rational representation of f. The degree of p/q is defined to be the maximum of the degrees of p and q. The *ϵ-approximate rational degree* of f, denoted $\mathrm{rdeg}_\epsilon(f)$, is the minimum degree of an ϵ-approximate rational representation of f. The *rational degree* $\mathrm{rdeg}(f)$ of f is the minimum degree of an exact (i.e., $\varepsilon = 0$) rational representation of f.

In the case of $\varepsilon = 0$, we drop the "approximate" and refer to p/q as a rational representation of f. Unlike in the definition of approximate degree, there is no requirement for an approximate rational representation to be bounded outside of D. Whether or not such a boundedness condition

is imposed matters significantly for the degree (see [BKT20]) but not for the rational degree, as we outline in Section C.1.

4.2.3 Sensitivity and Certificate Complexity

We now define some useful combinatorial measures of Boolean function complexity. Let f be a Boolean function, $x \in \{-1,1\}^n$, and $B \subseteq [n]$. We say that B is a sensitive block of f at x if $f(x) \neq f(x^B)$ where x^B denotes the bitstring obtained by flipping all bits of x indexed by B. We define, and denote by $\mathrm{bs}_f(x)$, the *block sensitivity of f at x* as the maximum number of disjoint blocks that are all sensitive at x. By restricting our attention to sensitive blocks that are singletons we obtain the analogous notion of the *sensitivity of f at x*, denoted $\mathrm{s}_f(x)$. The *block sensitivity* of f is defined as $\mathrm{bs}(f) = \max_{x\in\{-1,1\}^n} \mathrm{bs}_f(x)$. Similarly the *sensitivity* of f is defined as $\mathrm{s}(f) = \max_{x\in\{-1,1\}^n} \mathrm{s}_f(x)$. For $b \in \{0,1\}$, we also write $\mathrm{s}^{(b)}(f) = \max_{x\in f^{-1}(b)} \mathrm{s}_f(x)$.

A partial assignment is some function $\rho : [n] \to \{-1, 1, \star\}$. We define, and denote by $|\rho|$, the size of the partial assignment ρ as cardinality of the set $\{i \in [n] : \rho(i) \neq \star\}$. We say that a partial assignment ρ is *consistent* with some $x \in \{-1,1\}^n$ if $x_i = \rho(i)$ for all i with $\rho(i) \neq \star$. Given a Boolean function f we denote by $f|_\rho$ the restriction of f to the set of inputs $x \in \{-1,1\}^n$ that are consistent with ρ. Given $b \in \{-1,1\}$, we say that a partial assignment ρ is a *b-certificate for f* if $f|_\rho(x) = b$ for all $x \in \mathrm{Dom}(f|_\rho)$. The *$b$-certificate complexity* of f is defined as

$$\mathrm{C}_b(f) = \max_{x\in f^{-1}(b)} \min\{|\rho| : \rho \text{ is a } b\text{-certificate for } f \text{ consistent with } x\}.$$

The *certificate complexity of f* is defined as $\mathrm{C}(f) = \max_{b\in\{-1,1\}} \mathrm{C}_b(f)$.

4.2.4 Sign and Nondeterministic Degree

Given a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ we say that a function $p : \{-1,1\}^n \to \mathbb{R}$ is a *sign representation of f* if $\mathrm{sgn}(p(x)) = f(x)$ for all $x \in \{-1,1\}^n$ and $p(x) \neq 0$ on the entire hypercube. The *sign degree* of f is the minimum degree of a sign representation of f. Alon [Alo93] and

Anthony [Ant95] have shown that all but a negligible fraction of n-bit Boolean functions have sign degree at least $n/2$. Later, O'Donnell and Servedio proved [OS08] that almost every n-bit Boolean function has sign degree at most $n/2 + O(\sqrt{n \log n})$.

A less common but somewhat similar notion is that of a *nondeterministic representation* introduced by de Wolf [dW00]. In this context, it is customary to consider Boolean functions using the $\{0,1\}^n \to \{0,1\}$ representation.

We say that $p\colon \{0,1\}^n \to \mathbb{R}$ is a nondeterministic representation for $f\colon \{0,1\}^n \to \{0,1\}$ if $p(x) = 0$ if and only if $f(x) = 0$. The nondeterministic degree of f is the minimal degree of a nondeterministic representation of f. An easy calculation establishes the following relationship between the rational and nondeterministic degrees

$$\mathrm{rdeg}(f) = \max\{\mathrm{ndeg}(f), \mathrm{ndeg}(\bar{f})\}.$$

Note that this implies $\mathrm{rdeg}(f) = \mathrm{rdeg}(\bar{f})$. As mentioned in the introduction de Wolf conjectured that $\mathrm{D}(f) \leq O(\mathrm{ndeg}(f)\,\mathrm{ndeg}(\bar{f}))$ for all total Boolean functions. By showing that $\mathrm{ndeg}(f) \leq \mathrm{C}_1(f)$, de Wolf also established the inequality $\mathrm{rdeg}(f) \leq \mathrm{C}(f)$ [dW00].

4.2.5 Quantum Query Complexity and Postselection

We assume basic familiarity with concepts in quantum information. While we briefly review some of these, we refer the reader to, e.g., [NC11] for background.

Consider a Boolean function $f\colon D \to \{0,1\}$ where $D \subseteq \{0,1\}^n$. We say an ε-error quantum algorithm computes f if for all $x \in D$ it outputs a bit $a(x)$, corresponding to the measurement outcome of the first qubit, such that $\Pr[a(x) = f(x)] \geq 1 - \varepsilon$. BQP is the class of problems that have efficient (polynomial-time) quantum algorithms with error $1/3$ and EQP is the analogous class of exact algorithms. We also define complexity classes corresponding to quantum algorithms augmented with the power of *postselection*.

Definition 60. PostBQP is the set of languages $L \subset \{0,1\}^\star$ for which there exists a polynomial time quantum algorithm that, on input $x \in \{0,1\}^\star$, outputs two bits $a(x), b(x)$ (corresponding to the measurement results of the first and second qubits, respectively) such that:

(1) $\Pr[b(x) = 1] > 0$.

(2) If $x \in L$, then $\Pr[a(x) = 1 | b(x) = 1] \geq 2/3$.

(3) If $x \notin L$, then $\Pr[a(x) = 1 | b(x) = 1] \leq 1/3$.

PostEQP is the analogous class for *exact* algorithms with postselection: that is, the language defined by replacing the $2/3$ and $1/3$ above with 1 and 0, respectively.

Each of these computational complexity classes has an associated query measure. Formally, we say a function has query access to a string w if it has black-box access to a unitary U such that $U|i\rangle|b\rangle = |i\rangle|b \oplus w_i\rangle$. When the input w encodes the truth table of a Boolean function f, we will often write this as $U|x\rangle|b\rangle = |x\rangle|b \oplus f(x)\rangle$, where $x \in \{0,1\}^n$. The maximal number of calls an algorithm makes on some input to the unitary U is its *query complexity*. By $\mathsf{Q}_\varepsilon(f)$ and $\mathsf{Q}_E(f)$ we denote the query complexities of ε-error and exact quantum algorithms, respectively. $\mathsf{PostQ}_\varepsilon(f)$ and $\mathsf{PostQ}_E(f)$ are defined analogously for quantum algorithms with postselection. For simplicity of notation, $\mathsf{Q}(f)$ and $\mathsf{PostQ}(f)$ are understood to correspond to $\varepsilon = 1/3$.

A seminal result by Beals *et al.* establishes a lower bound on the quantum query complexity in terms of the degree and the approximate degree [BBC+01]. Formally, we have $\mathsf{Q}_\varepsilon(f) \geq \widetilde{\deg}_\varepsilon(f)/2$ for all (possibly partial) Boolean functions f. As a special case, $\mathsf{Q}_E(f) \geq \deg(f)/2$. This result gave rise to the so-called *polynomial method* for quantum query lower bounds. Similarly, it was shown by Mahadev and de Wolf that $\mathsf{PostQ}_\varepsilon(f) = \Theta(\mathrm{rdeg}_\varepsilon(f))$ and $\mathsf{PostQ}_E(f) = \mathrm{rdeg}(f)$ for total functions f [MdW14]. It is not difficult to extend this result to partial functions (see Section C.1). Nonetheless, it is surprising that the result does still hold for partial functions since the analogous result for quantum query complexity and approximate degree was recently shown to be false [AB23].

4.3 Lower Bounds on the Rational Degree

In this section, we establish lower bounds on the rational degree of special classes of Boolean functions. Our results constitute progress towards better understanding the relationship between the rational degree and the degree of total Boolean functions.

First, we establish the asymptotically tight lower bound $\mathrm{rdeg}(f) \geq (\deg(f)+1)/3$ for symmetric functions. Next, we prove that the rational degree equals the sensitivity for monotone functions, which implies that $\sqrt{\deg(f)} \leq \mathrm{rdeg}(f)$ for such functions. This lower bound is also tight. These results become key in lower bounding the rational degree of read-once Boolean formulae. Finally, we show that almost all n-bit Boolean functions have rational degree at least $n/2 - O(\sqrt{n})$.

4.3.1 Symmetric Functions

Our first lower bound is for (possibly partial) functions which are constant on a large number of Hamming slices. The following lemma will be used in Theorem 88 to obtain a constant vs. $\Omega(n)$ separation between the approximate degree and the rational degree of a partial function.

Lemma 61. *Given $f\colon D \to \{0,1\}^n$, where $D \subseteq \{0,1\}$, define*

$$\begin{aligned} S_0 &:= \{k \in \{0,1,\ldots,n\} : |x| = k \implies f(x) = 0\}, \\ S_1 &:= \{k \in \{0,1,\ldots,n\} : |x| = k \implies f(x) = 1\}. \end{aligned} \tag{4.1}$$

Then $\mathrm{rdeg}(f) \geq \frac{1}{2}\max(|S_0|, |S_1|)$.

In the above definition of S_0 and S_1, we require that D contains the entirety of the relevant Hamming slices. We will use the Minsky-Papert symmetrization technique, which assigns to a function $p\colon \{0,1\}^n \to \mathbb{R}$ a univariate function by letting $P(k) := \mathbb{E}_{|x|=k}[p(x)]$ [MP69].

Proof. Since $\mathrm{rdeg}(f) = \mathrm{rdeg}(\bar{f})$, we can assume without loss of generality that $|S_0| \geq |S_1|$. It suffices to show that $\mathrm{rdeg}(f) \geq |S_0|$. Let $f = p/q$ be a rational representation of f. We will also

assume that p is nonnegative, which we can do with at most a factor 2 overhead in the degree. Indeed, if $f = p'/q'$ where p' and q' can take on negative values, we can take $f = p/q$ where $p = p'q'$ and $q = q'^2$, increasing the degree of the rational representation by a factor of 2. Since f takes values in $\{0, 1\}$ and q is clearly positive, p must always be nonnegative.

Applying the Minsky-Papert symmetrization technique to $p(x)$, we obtain a univariate function $P(k)$ such that $\deg(p) \geq \deg(P)$ and $P(k) = 0$ for any $k \in S_0$. On the other hand, there exists at least one $k \in [n]$ such that $P(k) \neq 0$, since f is nonconstant and nonnegative. Thus $\deg(p) \geq \deg(P) \geq |S_0|$. Recall that we also incurred a multiplicative blowup of 2 in the degree of the representation by assuming that p is nonnegative. This results in a factor of $1/2$ in the lower bound. Since this argument holds for every rational representation of f, the result follows. □

For total Boolean functions, we use a slightly more careful version of the argument for Lemma 61 to show the next proposition.

Proposition 62. *If $f\colon \{0,1\}^n \to \{0,1\}$ is symmetric and nonconstant then*

$$\mathrm{rdeg}(f) \geq (\deg(f) + 1)/3.$$

Proof. Since $\deg(f)$ is always at most n, it suffices to show that $\mathrm{rdeg}(f) \geq (n+1)/3$. Define S_0 and S_1 as in Equation (4.1). Since f is total and symmetric, we have $|S_0| + |S_1| = n + 1$.

We analyze f according to the following four cases.

(1) $f(0^n) = 0$ and $f(1^n) = 0$. In this case, $\mathrm{ndeg}(\bar{f}) \geq |S_1|$. Importantly, there is no factor of $1/2$ since the symmetrization of any nondeterministic polynomial for $\bar{f}$ is guaranteed to be nonzero at inputs 0 and n. In addition, $\mathrm{ndeg}(f) \geq |S_0|/2$ by considering the symmetrization of the square of a nondeterministic representation for f (like in the proof of Lemma 61). Therefore $\mathrm{rdeg}(f) \geq \max(|S_0|/2, |S_1|) = \max(|S_0|/2, n + 1 - |S_0|)$. This quantity is minimized when $|S_0| = \frac{2(n+1)}{3}$, from which we obtain a lower bound of $\mathrm{rdeg}(f) \geq (n+1)/3$.

(2) $f(0^n) = 0$ and $f(1^n) = 1$. In this case, regardless of whether we consider f or $\bar{f}$, we will have at least one Hamming slice that is nonzero upon symmetrization. As such, $\mathrm{rdeg}(f) \geq \max(|S_0|, |S_1|)$ and so $\mathrm{rdeg}(f) \geq (n+1)/2$.

(3) $f(0^n) = 1$ and $f(1^n) = 0$. We obtain a lower bound $\mathrm{rdeg}(f) \geq \max(|S_0|, |S_1|) \geq (n+1)/2$ in the exact same way as in case 2.

(4) $f(0^n) = 1$ and $f(1^n) = 1$. We follow the same reasoning as case 1, giving us the lower bound $\mathrm{rdeg}(f) \geq \max(|S_0|, |S_1|/2) \geq (n+1)/3$.

In all cases, we have $\mathrm{rdeg}(f) \geq (n+1)/3$, which gives the proposition. □

In fact, Proposition 62 can be tight even in its constant factor as witnessed by the following function. For a positive integer n that is divisible by 3, let $\mathsf{MT}_n : \{0,1\}^n \to \{0,1\}$ be the Boolean function such that $\mathsf{MT}_n(x) = 1$ iff $|x| \in \{n/3, n/3+1, \ldots, 2n/3\}$. (The name MT abbreviates "middle third".) We will show in the next proposition that $\mathrm{rdeg}(\mathsf{MT}_n) \leq n/3 + 1$. But the polynomial degree of any nonconstant symmetric Boolean function is at least $n - O(n^{0.548}) = n(1 - o(1))$ as shown by von Zur Gathen and Roche [vZR97]. Therefore, $\mathrm{rdeg}(\mathsf{MT}_n) \leq \frac{1}{3}(1 + o(1)) \cdot \deg(\mathsf{MT}_n)$.

Proposition 63. $\mathrm{rdeg}(\mathsf{MT}_n) \leq n/3 + 1$.

It is easy to see that $\mathrm{ndeg}(\overline{\mathsf{MT}_n}) \leq n/3 + 1$. The harder part of this proposition is proving $\mathrm{ndeg}(\mathsf{MT}_n) \leq n/3$, which we do via the following sequence of three, increasingly stronger, lemmas. The first of these contains the key dimension-counting idea.

Lemma 64. *For every $k \in \{n/3, \ldots, 2n/3\}$, there exists $z \in \{0,1\}^n$ with $|z| = k$ and a function $q : \{0,1\}^n \to \mathbb{R}$ of degree at most $n/3$ such that q is nonzero at z but is zero at all $x \in \{0,1\}^n$ with $|x| \in \{0, 1, \ldots, n/3 - 1\} \cup \{2n/3 + 1, \ldots, n\}$.*

Proof. We prove this by a dimension-counting argument based on the following elementary fact whose easy proof we defer to the end of this subsection.

Fact 65. *For every positive integer n that is divisible by 3,*

$$\sum_{i \in \{0,1,\ldots,n/3-1\} \cup \{2n/3+1,\ldots,n\}} \binom{n}{i} < \sum_{i \in \{0,1,\ldots,n/3\}} \binom{n}{i}.$$

From this fact and dimension counting, we deduce there must exist a *nonzero* multilinear polynomial q of degree at most $n/3$ that vanishes at all $x \in \{0,1\}^n$ with $|x| \in \{0,1,\ldots,n/3-1\} \cup \{2n/3+1,\ldots,n\}$. It remains to argue that q is nonzero at some $z \in \{0,1\}^n$ with $|z| = k$.

Suppose for contradiction that q is zero at all $z \in \{0,1\}^n$ with $|z| = k$. Since q is nonzero (as a polynomial) and multilinear, q must be nonzero at some point $y \in \{0,1\}^n$ with $|y| \in \{n/3,\ldots,2n/3\}$. Let $l := |y|$, and note that $l \neq k$ by supposition. We assume that $k > l$ as the argument in the case $k < l$ is analogous.

We obtain a contradiction by symmetrizing a "translated version" of q as follows. Assume that y is of the form $1^l 0^{n-l}$. It is easy to see that the following argument makes this assumption without loss of generality. Consider the polynomial

$$\tilde{q}(x_{l+1},\ldots,x_n) := q(1,\ldots,1,x_{l+1},\ldots,x_n).$$

Clearly, $\deg(\tilde{q}) \leq \deg(q) \leq n/3$. But

$$\begin{aligned} &\tilde{q}(0^l) \neq 0, \\ &\tilde{q}(x) = 0 \qquad\qquad \text{for all } x \in \{0,1\}^n \text{ with } |x| \in \{k-l\} \cup \{2n/3+1-l,\ldots,n-l\}, \end{aligned}$$

where the not-equal-to uses $\tilde{q}(0^l) = q(y) \neq 0$ and the equalities use the fact that q vanishes at all $x \in \{0,1\}^n$ with $|x| \in \{2n/3+1,\ldots,n\}$.

Therefore, $\tilde{q}$ is a nonzero univariate polynomial vanishing at $n/3+1$ points, so the standard symmetrization argument implies $\deg(\tilde{q}) \geq n/3+1$, which contradicts $\deg(\tilde{q}) \leq n/3$. □

The preceding lemma can be strengthened to:

Lemma 66. *For every $z \in \{0,1\}^n$ with $|z| \in \{n/3, \dots, 2n/3\}$ there exists a function $q : \{0,1\}^n \to \mathbb{R}$ of degree at most $n/3$ such that q is nonzero at z but is zero at all $x \in \{0,1\}^n$ such that $|x| \in \{0,1,\dots,n/3-1\} \cup \{2n/3+1,\dots,n\}$.*

Proof. By the preceding lemma (Lemma 64), there exists a polynomial p of degree at most $n/3$ such that p is nonzero at some $w \in \{0,1\}^n$ with $|w| = |z|$ but is zero at all $x \in \{0,1\}^n$ with $|x| \in \{0,1,\dots,n/3-1\} \cup \{2n/3+1,\dots,n\}$.

Now, we simply observe that the following polynomial q satisfies the conditions of the lemma:

$$q(x_1,\dots,x_n) = p(x_{i_1},\dots,x_{i_n}),$$

where $i_1,\dots,i_n$ is any permutation of $[n]$ that maps $\{i \in [n] \mid z_i = 1\}$ to $\{i \in [n] \mid w_i = 1\}$, which have the same size since $|w| = |z|$. □

The preceding lemma can be further strengthened to:

Lemma 67. *There exists a function $q : \{0,1\}^n \to \mathbb{R}$ of degree at most $n/3$ that is zero at all $x \in \{0,1\}^n$ with $|x| \in \{0,1,\dots,n/3-1\} \cup \{2n/3+1,\dots,n\}$ but is nonzero at all $z \in \{0,1\}^n$ with $|z| \in \{n/3,\dots 2n/3\}$.*

Proof. By Lemma 66, for each $z \in \{0,1\}^n$ with $n/3 \le |z| \le 2n/3$ there exist a function $q_z : \{0,1\}^n \to \mathbb{R}$ of degree at most $n/3$ such that $q_z(z) \neq 0$ and $q_z(x) = 0$ at every $x \in \{0,1\}^n$ with $|x| \in \{0,1,\dots,n/3-1\} \cup \{2n/3+1,\dots,n\}$. Let S be the set of all such functions for all choices of z with $n/3 \le |z| \le 2n/3$. Then, by Lemma 82 the linear span of S contains a function q with the desired properties. □

Proposition 63 is a corollary of the lemmas above.

Proof of Proposition 63. Since $\mathrm{rdeg}(\mathrm{ndeg}(\mathsf{MT}_n)) = \max(\mathrm{ndeg}(\mathsf{MT}_n), \mathrm{ndeg}(\overline{\mathsf{MT}_n}))$, it suffices to show $\mathrm{ndeg}(\mathsf{MT}_n) \le n/3$ and $\mathrm{ndeg}(\overline{\mathsf{MT}_n}) \le n/3+1$. The first inequality is just a rephrasing of Lemma 67.

To see the second inequality, observe that the following polynomial p is nonzero at every $x \in \{0,1\}^n$ with $|x| \in \{0,1,\dots,n/3-1\} \cup \{2n/3+1,\dots,n\}$ and is zero at every $x \in \{0,1\}^n$ with $|x| \in \{n/3,\dots,2n/3\}$:

$$p(x) := \prod_{i \in \{n/3,\dots,2n/3\}} (x_1 + \cdots + x_n - i).$$

Therefore, p nondeterministically represents $\overline{\mathsf{MT}_n}$. But the degree of p is clearly $n/3+1$. Therefore, $\mathrm{ndeg}(\overline{\mathsf{MT}_n}) \leq n/3+1$. □

For completeness, we prove the fact used in the proof of Lemma 64.

Proof of Fact 65. By symmetry of the binomial coefficients, the fact is equivalent to

$$\sum_{i \in \{0,1,\dots,n/3-1\}} \tbinom{n}{i} < \tbinom{n}{n/3}.$$

Now, for every $i \in \{0,1,\dots,n/3-1\}$, we have

$$\frac{\binom{n}{i+1}}{\binom{n}{i}} = \frac{n-i}{i+1},$$

which decreases with i and is at least $(n-(n/3-1))/(n/3) = 2+3/n > 2$ and so

$$\sum_{i \in \{0,1,\dots,n/3-1\}} \tbinom{n}{i} < (1/2 + 1/4 + \cdots + 1/2^{n/3})\tbinom{n}{n/3} < \tbinom{n}{n/3},$$

as claimed. □

4.3.2 Monotone and Unate Functions

In this subsection we prove that $\mathrm{rdeg}(f) = \mathrm{s}(f)$ for monotone and unate Boolean functions f. We note that for monotone functions it suffices to prove that $\mathrm{s}(f) \leq \mathrm{rdeg}(f)$. This is because the certificate complexity of a monotone Boolean function f equals its sensitivity $\mathrm{C}(f) = \mathrm{s}(f)$ [Nis91]. Combining this with the fact that $\mathrm{rdeg}(f) \leq C(f)$ (via [dWoo]) we can already conclude the other inequality.

Claim 68. *For monotone Boolean functions* $f\colon \{0,1\}^n \to \{0,1\}$, $s(f) \leq \mathrm{rdeg}(f)$.

Our proof is similar to the proof that for all monotone Boolean functions $s(f) \leq \deg(f)$ as presented in [BdW02, Proposition 4].

Proof. Suppose without loss of generality that f is monotone increasing. We prove the claim by showing that

$$\mathrm{s}^{(0)}(f) \leq \mathrm{ndeg}(\bar{f}) \quad \text{and} \quad \mathrm{s}^{(1)}(f) \leq \mathrm{ndeg}(f).$$

We only prove the first inequality as the second can be proven analogously. Let x be such that $\mathrm{s}^{(0)}(f) = \mathrm{s}_f(x)$. All sensitive variables must be 0 in x since f is monotone increasing. Moreover, setting any sensitive variable to 1 changes the value of f from 0 to 1. Therefore, fixing all variables in x except for the $\mathrm{s}^{(0)}(f)$ many sensitive variables yields the OR_m function on $m := \mathrm{s}^{(0)}(f)$ variables. Since $\mathrm{ndeg}(\overline{\mathsf{OR}}_m) \geq m$, $\mathrm{ndeg}(\bar{f}) \geq \mathrm{s}^{(0)}(f)$. □

We remark that Claim 68 cannot be extended to all Boolean functions, as evidenced by the Kushilevitz function $K_m : \{0,1\}^{6^m} \to \{0,1\}$. The degree of K_m is only 3^m, while its sensitivity is maximal [NW95]. Since monotone functions satisfy $\mathrm{s}(f) = \mathrm{C}(f)$ and $\sqrt{\deg(f)} \leq \mathrm{s}(f)$ [Nis91], we have the following corollary.

Corollary 69. *For a monotone* f, $\mathrm{rdeg}(f) = \mathrm{s}(f)$. *Therefore,* $\sqrt{\deg(f)} \leq \mathrm{rdeg}(f)$.

This result also extends easily to unate functions. We first show that negating inputs does not change the rational degree.

Claim 70. *Let* $f\colon \{0,1\}^n \to \{0,1\}$ *be a Boolean function and let* $S \subseteq [n]$. *Define the Boolean function* $f' : \{0,1\}^n \to \{0,1\}$ *by letting* $f'(x) = f(x^S)$. *Then* $\mathrm{rdeg}(f) = \mathrm{rdeg}(f')$.

Proof. Suppose that p/q is a rational representation of f. Define a polynomial p' by taking every $i \in S$ and replacing every instance of x_i in the multilinear expansion of p with $1 - x_i$, and define q' similarly. This operation does not change the degree: $\deg(p) = \deg(p')$ and $\deg(q) = \deg(q')$. Furthermore, $f'(x) = p'(x)/q'(x)$, and the result follows. □

This fact allows for a simple proof that unate functions have large rational degree.

Corollary 71. *For unate Boolean functions f, $\sqrt{\deg(f)} \leq \mathrm{rdeg}(f)$.*

Proof. Every unate Boolean function is equivalent to a monotone function under a negation of some subset of the input variables. Thus the claim follows by Corollary 69 and Claim 70. □

Note that these bounds are tight, as witnessed by the AND-of-ORs function, $f = \mathsf{AND}_n \circ \mathsf{OR}_n$, on n^2 bits, which has $\mathrm{rdeg}(f) \leq \mathrm{C}(f) = n = \sqrt{\deg(f)}$. Writing $f(x) = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} x_{ij}$, then an explicit rational representation of f of degree n is

$$\frac{\prod_{i=1}^{n}(\sum_{j=1}^{n} x_{ij})}{\prod_{i=1}^{n}(\sum_{j=1}^{n} x_{ij}) + \sum_{i=1}^{n}\prod_{j=1}^{n}(1 - x_{ij})}.$$

4.3.3 Read-Once Formulae

In this section, we demonstrate lower bounds on read-once AC, $\mathsf{AC}[\oplus]$, TC, and $\mathsf{TC}[\oplus]$ formulae. Our result is actually slightly stronger, and holds for any read-once formula where the gates are functions of high rational degree. We begin by proving a composition result for the rational degree.

Lemma 72. *Let $f\colon \{0,1\}^n \to \{0,1\}$ and $g_i : \{0,1\}^{n_i} \to \{0,1\}$ be Boolean functions where every variable in each function is relevant. Suppose that the function $h : \{0,1\}^{\sum n_i} \to \{0,1\}$ is given by $h(x^1, \ldots, x^n) = f(g_1(x^1), \ldots, g_n(x^n))$. Then*

$$\mathrm{rdeg}(h) \geq \max\{\mathrm{rdeg}(f), \mathrm{rdeg}(g_1), \ldots, \mathrm{rdeg}(g_n)\}.$$

Proof. Since every variable is relevant for each g_i, we know there exist restrictions ρ^i to all but 1 variable in each x^i such that $g_i|_{\rho^i}(x^i) = x^i_{k_i}$ or $(1 - x^i_{k_i})$ for some $1 \leq k_i \leq n_i$. Considering the restriction $\rho = \rho^1 \cup \cdots \cup \rho^n$, it is evident that $h|_\rho(x) = f(x^1_{k_1}, \ldots, x^n_{k_n})$ up to negations. Therefore,

$$\mathrm{rdeg}(h) \geq \mathrm{rdeg}(h|_\rho) \geq \mathrm{rdeg}(f). \tag{4.2}$$

Now pick an arbitrary i. Since, by assumption, every variable of f is relevant, there exists an assignment $x_j = z_j$ for all $j \neq i$ such that $f(z_1, \dots z_{i-1}, x_i, z_{i+1}, \dots, z_n) = x_i$ or $\overline{x_i}$. Since g_i is nonconstant, it follows that there exists an assignment to the variables $(x^j)_{j \neq i}$ such that each $g_j(x^j)$ is fixed to z_j.

Let τ be the restriction induced by this partial assignment. Then

$$h|_\tau(x) = f(z_1, \dots z_{i-1}, g_i(x^i), z_{i+1}, \dots z_n) = g_i(x^i) \text{ or } \overline{g_i(x^i)}.$$

Consequently,

$$\operatorname{rdeg}(h) \geq \operatorname{rdeg}(h|_\tau) \geq \operatorname{rdeg}(g_i). \tag{4.3}$$

Combining Equations (4.2) and (4.3) gives the desired result. □

Next, we show how this composition result can be applied to rational degree lower bounds for read-once formulae.

Lemma 73. *Suppose that f can be written as a read-once formula with symmetric gates where the maximum branching factor of any node is w. Then* $\operatorname{rdeg}(f) = \Omega(w)$.

Proof. Let p/q be a rational representation of f. Due to Claim 70, we can assume without loss of generality that only the literals x_i, appear in the formula and not $\overline{x_i}$.

Now consider the node G with branching factor w. Let F be the subformula with top gate G and let $F_1, \dots, F_w$ be the read-once subformulae below G. Each F_i is nonconstant, which implies the existence of a restriction ρ_i of all but 1 variable in each V_i such that toggling the sole live variable (say x_{k_i}) toggles the value of F_i (i.e., $F_i|_{\rho_i} = x_k$ or $\overline{x_k}$ for some k). As the V_i are disjoint (as f is read-once), these restrictions together define a unified restriction ρ such that $F|_\rho(x) = G(x_{k_1}, \dots, x_{k_w})$ up to negations. Inductively using Lemma 72 on the formula $f|_\rho$ by starting at the top node and

going down the path to G, it follows that

$$\mathrm{rdeg}(f) \geq \mathrm{rdeg}(f|_\rho) \geq \mathrm{rdeg}(F|_\rho) = \mathrm{rdeg}(G) \geq w/2,$$

where the last inequality follows from Lemma 61. □

We now prove rational degree lower bounds on read-once formulae with symmetric gates.

Corollary 74. *Suppose that f can be written as a depth-d read-once formula with symmetric gates. Then* $\mathrm{rdeg}(f) = \Omega(\deg(f)^{1/d})$.

Proof. The result follows from Lemma 73 and a simple contradiction argument: if all nodes have branching factor strictly less than $n^{1/d}$ then there must be strictly fewer than n literals. Note that $n \geq \deg(f)$ so the lower bound $\Omega(n^{1/d})$ is stronger. □

In particular, this result holds for read-once $\mathsf{AC}[\oplus]$ formulae. This lower bound is tight for $d = 2$, as witnessed by the AND-of-ORs function $f = \mathsf{AND}_{\sqrt{n}} \circ \mathsf{OR}_{\sqrt{n}}$. Indeed, in this case $\mathrm{rdeg}(f) \leq \mathrm{C}(f) = n^{1/2} = \sqrt{\deg(f)}$.

A similar, albeit weaker result can be shown for read-once formulae where the gates are symmetric *or* unate.

Corollary 75. *Let f be written as a depth-d read once formula where every gate is symmetric and/or unate. Then* $\mathrm{rdeg}(f) = \Omega(\deg(f)^{1/2d})$.

Proof. The proof is similar to that of Corollary 74. Consider any gate in the formula and let w be its branching factor. If this node corresponds to a symmetric or unate gate then we have $\mathrm{rdeg}(f) = \Omega(w)$ (via Lemma 61) or $\mathrm{rdeg}(f) = \Omega(\sqrt{w})$ (via Corollary 71), respectively. By contradiction, at least one node has branching factor at least $\deg(f)^{1/d}$. The result follows by applying Lemma 73. □

Thus, in particular, read-once $\mathsf{TC}[\oplus]$ formulae have rational degree at least $\Omega(\deg(f)^{1/2d})$, as they consist of linear threshold gates, which are unate, and parity gates, which are symmetric.

Observe that if we restrict our attention to read-once AC and TC (that is, if we exclude parity gates), the lower bounds in Corollary 74 and Corollary 75 improve to $\sqrt{n}$ via Corollary 71. This is because the composition of unate functions is unate, and read-once AC and TC are entirely comprised of unate gates.

Corollary 76. *For any read-once* TC *formula f on m disjoint variables, $\sqrt{m} \leq \mathrm{rdeg}(f)$.*

Proof. We know that f is composed entirely of unate gates. By a simple inductive argument, the composition of unate functions is unate. Thus we deduce that f is unate, and since the degree of f is m, the result follows from Corollary 71. □

In Section C.2, we prove a structural result, Proposition 109, showing that any read-once TC formula on m disjoint variables must restrict to either an AND or OR function on at least $\sqrt{m}$ disjoint variables (up to negation). Proposition 109 immediately implies Corollary 76 but is not implied by it and so may be of independent interest.

It can be shown that there exist arbitrary-depth read-once formulae with rational degree at most $\sqrt{n}$ (see Figure 4.3). The inverse dependence on depth in Corollary 74 and Corollary 75 is somewhat unintuitive, and we conjecture that these results are not tight for $d > 2$.

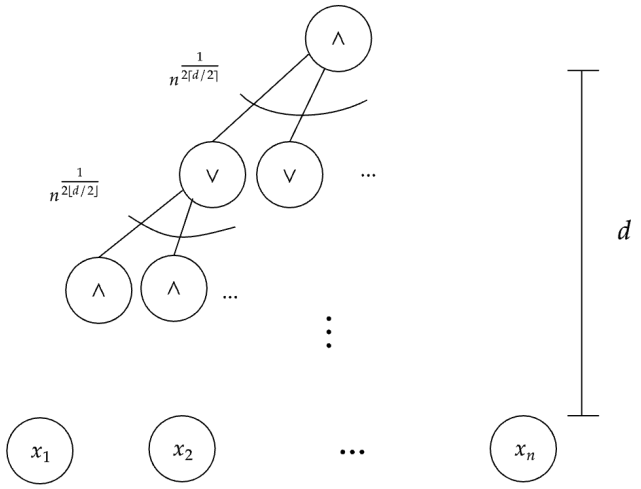


Figure 4.3: A depth-d AND-OR tree with certificate complexity $\sqrt{n}$, and thus rational degree at most $\sqrt{n}$. Indeed, setting all input wires to 1 for AND functions and a single input wire to 1 for OR functions along a single path to root gives a 1 certificate, and setting all input wires to 0 for OR functions and a single input wire to 0 for AND functions gives a 0-certificate. One can easily verify that both of these certificates are of size $\sqrt{n}$.

4.3.4 Random Functions

In this section we prove that all but a negligible fraction of Boolean functions $f\colon \{-1,1\}^n \to \{-1,1\}$ have rational degree at least $n/2 - O(\sqrt{n})$.

As mentioned in the introduction Alon [Alo93] and Anthony [Ant95] used counting arguments to show that all but a negligible fraction of n-bit Boolean functions have sign degree at least $n/2$. Given nondeterministic representations $p(x)$ of f and $q(x)$ of $\bar{f}$, the function $p(x)^2 - q(x)^2$ gives a sign representation of f. Thus, $2\,\mathrm{rdeg}(f) \geq \mathrm{sdeg}(f)$, and it follows that almost all n-bit Boolean functions have rational degree at least $n/4$. To show the tighter result for the rational degree, we restate a variant of the function counting theorem used by Anthony [Ant95].

Given a finite set X and a mapping $\phi\colon X \to \mathbb{R}^d$, a ϕ-separable dichotomy of X is a partition of X into subsets $X^+ \cup X^-$ such that there exists some $w \in \mathbb{R}^d$ for which $w \cdot \phi(x) > 0$ for all $x \in X^+$ and $w \cdot \phi(x) < 0$ for all $x \in X^-$.

Theorem 77 (Function counting theorem, [Cov65])**.** *Let $\phi\colon S \to \mathbb{R}^d$. Let $X = \{x_1, \ldots, x_N\}$ be a subset of S. If a ϕ-surface (i.e., a set of the form $\{x \in S : w \cdot \phi(x) = 0\}$ for some $w \in \mathbb{R}^d$) contains a set of points $Y = \{y_1, y_2, \ldots, y_k\} \subseteq S$, where $\phi(y_i)$ are linearly independent for all i, and where the projection of the set of vectors $\phi(x_1), \ldots, \phi(x_N)$ onto the orthogonal complement of the span of the $\phi(y_i)$'s is in general position, then there are $C(N, d-k)$ many ϕ-separable dichotomies of X, where*

$$C(N,d) = 2\sum_{i=0}^{d-1} \binom{N-1}{i}.$$

We consider the following adaptation of the above theorem. Consider a set of N points $S = \{v_1, \ldots, v_n\}$ in $\mathbb{R}^D$. Given a 2-coloring of the points $f\colon [N] \to \{-1,1\}$, we say that the coloring f is separable by two hyperplanes if there exist hyperplanes $H_j = \{v : \alpha_j \cdot v = 0\}$ for $j = 1, 2$ such that

$$\forall i \in [N] : \quad f(i) = \mathrm{sgn}((\alpha_1 \cdot v_i)(\alpha_2 \cdot v_i)).$$

Corollary 78. *Given N points in $\mathbb{R}^M$, the number of 2-colorings $f\colon [N] \to \{-1,1\}$ that are separable by two hyperplanes is at most $C(N,M)^2$.*

Proof. Let $S \subset \mathbb{R}^M$ be given and suppose that f is a coloring that is separated by the hyperplanes H_1 and H_2. Then there exist colorings f_1, f_2 that are separated by the hyperplanes H_1 and H_2 respectively. Since there are at most $C(N,M)$ choices for each of f_1 and f_2, the number of such colorings f is bounded by $C(N,M)^2$. □

Lemma 79. *Let $m \leq n$ be positive integers. The number of Boolean functions $f\colon \{-1,1\}^n \to \{-1,1\}$ with rational degree at most m is at most $C(2^n, \binom{n}{\leq m})^2$.*

Proof. Let $M = \binom{n}{\leq m}$. For each $x \in \{-1,1\}^n$ define $v_x \in \mathbb{C}^M$ by letting $(v_x)_S = \chi_S(x)$ for $S \subseteq [n]$, $|S| \leq m$. Suppose p/q is a rational representation of f of degree at most m. Then

$$f(x) = \operatorname{sgn}(p(x)/q(x)) = \operatorname{sgn}(p(x)q(x)) = \operatorname{sgn}((\widehat{p}\cdot v_x)(\widehat{q}\cdot v_x)).$$

Thus the coloring given by f is separated by the hyperplanes defined by $\widehat{p}$ and $\widehat{q}$. The result follows by Corollary 78. □

We now state and prove our extremal lower bound on the rational degree.

Corollary 80. *All but a negligible fraction of Boolean functions on n variables have rational degree at least $n/2 - O(\sqrt{n})$.*

Proof. Write $m = n/2 - \sqrt{cn}$ for some constant c, and let $N = 2^n$. Then by Corollary 78 there are at most $C\big(N, \binom{n}{\leq m}\big)^2$ Boolean functions on n variables of rational degree less than m. By the Chernoff bound $\binom{n}{\leq n/2-\lambda} < 2^n e^{-2\lambda^2/n}$ and the Hamming bound, we have that the proportion of Boolean functions with rational degree strictly less m is bounded above by

$$\frac{C\big(N, \binom{n}{\leq m}\big)^2}{2^N} \leq \frac{C(N, Ne^{-2c})^2}{2^N} \leq O\big(2^{N(2h_2(e^{-2c})-1)}\big),$$

where $h_2(\cdot)$ denotes the binary entropy function. Solving the inequality $h_2(e^{-2c}) < 1/2$ numerically we find that for $c \geq 1.104$, the above bound tends to 0. □

4.4 Composition Lemmas for the Rational Degree

Typically, the first composition lemma one proves for a Boolean complexity measure is exact composition with respect to XOR. However, this fails for the nondeterministic degree, e.g., $\mathrm{ndeg}(x_1 \oplus x_2) = 1 \neq 2 = \mathrm{ndeg}(x_1) + \mathrm{ndeg}(x_2)$. Instead, we find that the nondeterministic degree composes exactly with respect to AND.

In this section, we consider Boolean functions $f\colon \{0,1\}^m \to \{0,1\}$, and $g\colon \{0,1\}^n \to \{0,1\}$ for some $m, n \in \mathbb{N}$. By $f \wedge g$ and $f \vee g$ we denote the functions $\{0,1\}^m \times \{0,1\}^n \to \{0,1\}$ given by $(f \wedge g)(x,y) = f(x) \wedge g(y)$ and $(f \vee g)(x,y) = f(x) \vee g(y)$, respectively. We also write $\mathbb{R}[X]$, $\mathbb{R}[Y]$, and $\mathbb{R}[X,Y]$ for the rings of formal polynomials $\mathbb{R}[X_1, \ldots, X_m]$, $\mathbb{R}[Y_1, \ldots, Y_n]$, and $\mathbb{R}[X_1, \ldots X_m, Y_1, \ldots, Y_n]$, respectively.

Proposition 81. *Let f and g be nonconstant Boolean functions on disjoint variables. Then*

$$\mathrm{ndeg}(f \wedge g) = \mathrm{ndeg}(f) + \mathrm{ndeg}(g).$$

Proof. Let $k \coloneqq \mathrm{ndeg}(f)$ and $l \coloneqq \mathrm{ndeg}(g)$. It is clear that $\mathrm{ndeg}(f \wedge g) \leq k + l$ by considering the product of nondeterministic polynomials for f and g. We proceed to prove the lower bound.

Let $P(X,Y) \in \mathbb{R}[X,Y]$ be a nondeterministic representation of $f \wedge g$. Since we will lower bound the degree of P, we may assume P is multilinear without loss of generality.

Then, write

$$\begin{aligned} P(X,Y) &= \sum_{S \subseteq [m]} p_S(Y) \prod_{i \in S} X_i \\ &= \sum_{S \subseteq [m] \,:\, |S| \geq k} p_S(Y) \prod_{i \in S} X_i + \sum_{S \subseteq [m] \,:\, |S| < k} p_S(Y) \prod_{i \in S} X_i. \end{aligned}$$

Let $y \in g^{-1}(0)$. Then $P(X, y) \in \mathbb{R}[X]$ satisfies $P(x, y) = 0$ for all $x \in \{0,1\}^m$. This means that $P(X, y)$ is the zero polynomial by the uniqueness of exact multilinear polynomial representations of Boolean functions. Therefore $p_S(y) = 0$ for all $S \subseteq [m]$.

Let $y \in g^{-1}(1)$. Then $P(X, y)$ is a nondeterministic representation of f. Therefore, since $\mathrm{ndeg}(f) = k > 0$, at least one entry of the vector $(p_S(y))_{S\subseteq[m],|S|\geq k}$ must be nonzero.

We now appeal to the following lemma with $\mathscr{D}$ set to $g^{-1}(1)$ and the f_i functions given by (functions induced by) the set $\{p_S : S \subseteq [m], |S| \geq k\}$. We defer the short proof of this lemma to after this proof.

Lemma 82. *Let $n \in \mathbb{N}$. Let $\mathscr{D}$ be a finite nonempty set. Let $f_1, \ldots, f_n : \mathscr{D} \to \mathbb{R}$. Suppose that for all $y \in \mathscr{D}$, there exists $i \in [n]$ such that $f_i(y) \neq 0$. Then there exist $\alpha_1, \ldots, \alpha_n > 0$ such that, for all $y \in \mathscr{D}$*

$$(\alpha_1 f_1 + \cdots + \alpha_n f_n)(y) \coloneqq \alpha_1 f_1(y) + \cdots + \alpha_n f_n(y) \neq 0.$$

By Lemma 82, for every $S \subseteq [m]$ with $|S| \geq k$ there exists $\alpha_S > 0$ such that the polynomial

$$Q(Y) \coloneqq \sum_{S\subseteq[m]\,:\,|S|\geq k} \alpha_S \cdot p_S(Y) \in \mathbb{R}[Y]$$

satisfies $y \in g^{-1}(1) \implies Q(y) \neq 0$.

On the other hand, we have already observed that $y \in g^{-1}(0)$ implies $p_S(y) = 0$ for all $S \subseteq [m]$. Therefore, $y \in g^{-1}(0) \implies Q(y) = 0$.

We conclude that $Q(Y)$ nondeterministically represents g. Therefore, the degree of Q is at least l. Thus, there must exist $S^* \subseteq [m]$ with $|S^*| \geq k$ such that $\deg(p_{S^*}(Y)) \geq l$.

But $l > 0$ since g is nonconstant so the degree of the term

$$p_{S^*}(Y) \prod_{i\in S^*} X_i$$

is at least[3] $k + l$. Therefore, P has degree at least $k + l$, as required. □

[3]Note that if $l = 0$, $p^*_S(Y)$ could be the zero polynomial in which case the degree of the term here is also zero.

Proof of Lemma 82. Since $\mathscr{D}$ is finite, for all $i \in [n]$, there exists $0 < b_i < B_i$ such that $f_i(\mathscr{D}) \setminus \{0\} \subseteq (-B_i, -b_i) \cup (b_i, B_i)$. Now define $\alpha_1 = 1$ and for $i = 2, \ldots, n$, define $\alpha_i > 0$ by

$$\alpha_i b_i := \sum_{j=1}^{i-1} \alpha_j B_j + 1 > \sum_{j=1}^{i-1} \alpha_j B_j.$$

It is straightforward to verify that $(\alpha_1 f_1 + \cdots + \alpha_n f_n)(y) \neq 0$ for all $y \in \mathscr{D}$. □

We also show the following composition lemma for the nondeterministic degree with respect to OR. (Note that this lemma also does not hold with respect to XOR since

$$\mathrm{ndeg}((x_1 \oplus x_2) \oplus x_3) = 2 \neq 1 = \max(\mathrm{ndeg}(x_1 \oplus x_2), \mathrm{ndeg}(x_3)).$$

Indeed, we were unable to find a "clean" composition lemma for the nondeterministic or rational degree with respect to XOR.)

Proposition 83. *Let f and g be nonconstant Boolean functions on disjoint inputs. Then*

$$\mathrm{ndeg}(f \vee g) = \max(\mathrm{ndeg}(f), \mathrm{ndeg}(g)).$$

Proof. Let $k := \mathrm{ndeg}(f)$ and $l := \mathrm{ndeg}(g)$. By restriction it is clear that $\mathrm{ndeg}(f \vee g)$ is no less than $\max(\mathrm{ndeg}(f), \mathrm{ndeg}(g))$. We proceed to prove the upper bound.

Suppose $p(X) \in \mathbb{R}[X]$ and $q(Y) \in \mathbb{R}[Y]$ are nondeterministic polynomials for f and g respectively such that $\deg(p(X)) = k$ and $\deg(q(Y)) = l$. Let $\alpha, \beta \in \mathbb{R} \setminus \{0\}$ be such that

$$\forall x \in \{0,1\}^m : \ |\alpha p(x)| \leq 1;$$

$$\forall x \in \{0,1\}^n \text{ s.t. } q(x) \neq 0 : \ |\beta q(x)| \geq 100.$$

Then it easy to verify that $\alpha p(X) + \beta q(Y) \in \mathbb{R}[X, Y]$ is a nondeterministic polynomial for $f \vee g$. Since $\deg(\alpha p(X) + \beta q(Y)) \leq \max(k, l)$, the proposition follows. □

We have the following corollary for the rational degree of the AND and OR of Boolean functions.

Corollary 84. *Let f and g be nonconstant Boolean functions on disjoint variables. Then*

$$\mathrm{rdeg}(f \wedge g) = \max(\mathrm{ndeg}(f) + \mathrm{ndeg}(g), \mathrm{ndeg}(\overline{f}), \mathrm{ndeg}(\overline{g})),$$
$$\mathrm{rdeg}(f \vee g) = \max(\mathrm{ndeg}(\overline{f}) + \mathrm{ndeg}(\overline{g}), \mathrm{ndeg}(f), \mathrm{ndeg}(g)).$$

Proof. This follows from De Morgan's laws, the identity $\mathrm{rdeg}(h) = \max(\mathrm{ndeg}(h), \mathrm{ndeg}(\overline{h}))$, and Propositions 81 and 83. □

4.5 Rational Degree versus Spectral Sensitivity

The spectral sensitivity $\lambda(f)$ is a complexity measure of a Boolean function f first defined formally by Aaronson *et al.* [ABK+21], although it has implicitly appeared earlier in Huang's proof of the Sensitivity Theorem [Hua19]. Aaronson and his coauthors suggested [ABK+21] that it may be possible to establish a lower bound on the rational degree $\mathrm{rdeg}(f)$ by relating it to the spectral sensitivity $\lambda(f)$. In this direction, we show that $\lambda(f) \leq \mathrm{rdeg}(f)$ is false by exhibiting a degree $3/2$ separation between these two measures. The spectral sensitivity is defined formally as follows.

Definition 85. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. The sensitivity graph of f is defined as the subgraph $G_f = (V, E)$ of the Boolean hypercube with $V = \{0,1\}^n$ and edge set $E = \{(x, y) : |x \oplus y| = 1, f(x) \neq f(y)\}$. The spectral sensitivity of f is defined as the spectral norm of the adjacency matrix of G_f, i.e., $\lambda(f) = \|A_f\|$, where A_f denotes the adjacency matrix of G_f.

For an even positive integer n, let $\mathsf{EH}_n : \{0,1\}^n \to \{0,1\}$ be the Boolean function such that $\mathsf{EH}_n(x) = 1$ iff $|x| = n/2$. As we show below, composing AND_n with the negation of EH_n gives a degree-$3/2$ separation between the spectral sensitivity and the rational degree.

Proposition 86. *The function $f = \mathsf{AND}_n \circ \overline{\mathsf{EH}_n}$ satisfies $\lambda(f) = \Theta(n^{3/2})$, $\mathrm{rdeg}(f) = \Theta(n)$ and $\mathrm{s}(f) = \Theta(n^2)$, $\deg(f) = \Theta(n^2)$.*

Proof. Since the spectral sensitivity behaves multiplicatively under composition (c.f. [ABK+21, Thm. 29]) we have $\lambda(f) = \lambda(\mathsf{AND}_n) \cdot \lambda(\overline{\mathsf{EH}_n})$. It is not hard to see that $\lambda(\mathsf{AND}_n) = \sqrt{n}$ and $n/2 \leq \lambda(\overline{\mathsf{EH}_n}) \leq n$. Thus $\lambda(f) = \Theta(n^{3/2})$.

On the other hand, by our composition theorem (Proposition 81) $\mathrm{ndeg}(f) = n \cdot \mathrm{ndeg}(\overline{\mathsf{EH}_n})$. By De Morgan's laws and Proposition 83 $\mathrm{ndeg}(\bar{f}) = \mathrm{ndeg}(\mathsf{OR}_n \circ \mathsf{EH}_n) = \mathrm{ndeg}(\mathsf{EH}_n)$. So it remains to identify the nondeterministic degrees of $\overline{\mathsf{EH}_n}$ and EH_n. To this end, observe that $q(x) = \sum_{i=1}^{n} x_i - n/2$ is a nondeterministic representation of $\overline{\mathsf{EH}_n}$ with degree 1, which must also be degree-optimal since $\overline{\mathsf{EH}_n}$ is nonconstant. Therefore, $\mathrm{ndeg}(\overline{\mathsf{EH}_n}) = 1$ and so $\mathrm{ndeg}(f) = n$. Moreover, we trivially have $\mathrm{ndeg}(\mathsf{EH}_n) \leq n$ so $\mathrm{ndeg}(\bar{f}) \leq n$. Therefore, $\mathrm{rdeg}(f) = \max(\mathrm{ndeg}(f), \mathrm{ndeg}(\bar{f})) = n$.

Finally, notice that the following input has $\approx n^2/2$ sensitive bits: for every instance of $\overline{\mathsf{EH}}$ we consider an input with $n/2 - 1$ many ones. Flipping any 0 in this input flips the value of f from 1 to 0. Moreover, it is known that the degree composes and both AND and EH have maximal degree, therefore we can conclude that $\deg(f) = \Theta(n^2)$. □

We remark that $f = \mathsf{AND}_n \circ \mathsf{EH}_n$ also separates $\mathrm{rdeg}(f)$ from $\min\{\mathsf{C}(f), \deg(f)\}$ quadratically. It is clear that $\mathrm{rdeg}(g) \leq \min\{\mathsf{C}(g), \deg(g)\}$ for any function g, and $\mathsf{C}(g)$ and $\deg(g)$ can be separated quadratically from each other, so it is not surprising that $\mathrm{rdeg}(g)$ can be quadratically smaller than $\mathsf{C}(g)$ or $\deg(g)$ separately. However, as this example shows, it can be quadratically separated from both simultaneously.

4.6 Applications in Complexity Theory

In this section we construct two functions that demonstrate constant vs. $\Omega(n)$ separations between the approximate degree and the rational degree in each direction. These examples in turn give bidirectional oracle separations between BQP and $\mathsf{PostEQP}$. We conclude the section by giving evidence that exact quantum computation with postselection gives advantage over bounded-error randomized algorithms, providing context to our results.

4.6.1 Postselection can be a Weak Resource

In this subsection, we give an oracle which witnesses that BQP $\not\subseteq$ PostEQP (in fact, even RP $\not\subseteq$ PostEQP). This is accomplished by constructing a partial function with constant 1-sided error randomized query complexity but maximal PostQ_E. In fact, this problem also demonstrates that the rational degree can be much higher than the approximate degree for partial functions.

Problem 87 (Majority or None)**.** *The* $\textsc{MajOrNone}_n$ *function is defined as a partial Boolean function on the set of bitstrings* $x \in \{0,1\}^n$ *that have Hamming weight either* 0 *or at least* $n/2$*. The function* $\textsc{MajOrNone}_n$ *takes value* 0 *in the former case, and takes value* 1 *otherwise.*

Theorem 88. *The* $\textsc{MajOrNone}_n$ *function can be decided by a quantum algorithm using constantly many queries, yet its rational degree is at least* $\Omega(n)$*. Consequently,* $\textsc{MajOrNone}_n$ *witnesses the following separations:*

$$\widetilde{\deg}(\textsc{MajOrNone}_n) \leq O(1) \quad \textit{yet} \quad \mathrm{rdeg}(\textsc{MajOrNone}_n) \geq \Omega(n),$$
$$\mathsf{Q}(\textsc{MajOrNone}_n) \leq O(1) \quad \textit{yet} \quad \mathsf{PostQ}_E(\textsc{MajOrNone}_n) \geq \Omega(n).$$

Proof. The $\textsc{MajOrNone}_n$ function even has constant RP query complexity. Indeed, we may simply query a constant number of random bits and output 1 if any of them are 1. Therefore, $\textsc{MajOrNone}_n$ has constant quantum query complexity, which in turn implies a constant approximate degree. We show via a rational degree lower bound that $\mathsf{PostQ}_E(\textsc{MajOrNone}_n) = \Omega(n)$. In particular, we show that $\mathrm{rdeg}(\textsc{MajOrNone}_n) = \Omega(n)$. Using the notation of Lemma 61, for $\textsc{MajOrNone}_n$ we have $|S_1| \geq n/2$, giving us

$$\mathsf{PostQ}_E(\textsc{MajOrNone}_n) \geq \mathrm{rdeg}(\textsc{MajOrNone}_n) = \Omega(n). \qquad \square$$

The complexity classes Q and PostQ_E are the query complexity equivalents of BQP and PostEQP, respectively. As such, our constant vs. $\Omega(n)$ separation between these complexity measures gives an oracle separation of BQP and PostEQP.

Corollary 89. *There exists an oracle O such that* $\mathsf{RP}^O \not\subseteq \mathsf{PostEQP}^O$. □

4.6.2 Postselection can be a Strong Resource

On the other hand, we give an oracle which witnesses $\mathsf{PostEQP} \not\subseteq \mathsf{BQP}$. We do this by constructing a promise problem f for which $\mathsf{PostQ}_E(f) = O(1)$ but $\mathsf{Q}_\varepsilon(f) = \Omega(n)$. This problem also witnesses an $\Omega(n)$ vs. constant separation between the approximate degree and the rational degree of a partial function.

Problem 90 (Imbalance)**.** *Let $n = 4m + 2$ for some positive integer m. Define the functions $L, R : \{-1,1\}^n \to \mathbb{R}$ as $L(x) = x_1 + x_2 + \ldots + x_{2m+1}$ and $R(x) = x_{2m+2} + \ldots + x_{4m+2}$. Then the* Imbalance $: \{-1,1\}^n \to \mathbb{R}$ *function is defined as* $\textsc{Imbalance}(x) = \frac{L(x)}{R(x)}$.

Note that we assumed n is not divisible by 4 to ensure that the denominator $R(x)$ cannot be 0.

Problem 91 (Boolean Imbalance)**.** *Let m and n be as in the above problem. We define the Boolean Imbalance* BI_n *function as the restriction of* Imbalance *to the union $S_- \cup S_+$ where we let*

$$S_+ := \{(x_L, x_R) : \ |x_L| = |x_R| = m\},$$
$$S_- := \{(x_L, x_R) : \ |x_L| + |x_R| = 2m + 1 \text{ and } |x_L|, |x_R| \geq m\}.$$

Note that $\mathrm{BI}_n(x) = 1$ for any $x \in S_+$ since the numerator and denominator evaluate to the same quantity. On the other hand, for any $x \in S_-$ we have that $\mathrm{BI}_n(x) = -1$ since both $L(x)$ and $R(x)$ must be ± 1 but they must be different.

In Section C.1 we show that the identity $\mathsf{PostQ}_\varepsilon(f) \leq \mathrm{rdeg}_\varepsilon(f) \leq 2\mathsf{PostQ}_\varepsilon(f)$ (established by Mahadev and de Wolf) holds for partial functions as well. Thus, $\mathsf{PostQ}_E(\mathrm{BI}_n) \leq 2$. In contrast, we show below that $\mathrm{rdeg}(\mathrm{BI}_n) \geq \Omega(n)$.

Lemma 92. *The* BI_n *function can be decided by a postselected quantum algorithm using only 2 queries, yet its rational degree is at least* $\Omega(n)$*. Consequently,* BI_n *witnesses the following separations:*

$$\mathrm{rdeg}(\mathrm{BI}_n) \leq O(1) \quad \textit{yet} \quad \widetilde{\deg}(\mathrm{BI}_n) \geq \Omega(n),$$
$$\mathsf{PostQ}_E(\mathrm{BI}_n) \leq O(1) \quad \textit{yet} \quad \mathsf{Q}(\mathrm{BI}_n) \geq \Omega(n).$$

Proof. Note that BI_n is defined on inputs of Hamming weight $2m$ and $2m+1$. Nayak and Wu showed that any function that is constant on Hamming slices $l, l+1$ but flips its value has approximate degree $\Omega(\max\{l, n-l\})$ [NW99]. In this case, since the function value flips on Hamming weights $2m, 2m+1$ we get a lower bound of $\Omega(\max\{2m+1, 2m\}) = \Omega(n)$. □

Finally, just like in the previous section, this separation between complexity measures allows us to construct an oracle relative to which PostEQP is not contained in BQP.

Corollary 93. *There exists an oracle* O *such that* $\mathsf{PostEQP}^O \not\subseteq \mathsf{BQP}^O$. □

Our constant vs. $\Omega(n)$ separation between the rational degree and the approximate degree gives an oracle separation of PostEQP and BQP. Combined with Corollary 89, this tells us that exact postselection and bounded error are "incomparable" resources in the black-box model: one is not stronger than the other.

4.6.3 Postselection and Nondeterminism

To conclude the section, we provide more context to our results by giving evidence that exact quantum computation with postselection gives advantage over efficient classical computation.

Claim 94. $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{PostEQP}$.

Proof. Let $L \in \mathsf{NP} \cap \mathsf{coNP}$. Since $L \in \mathsf{NP}$, there is an efficient algorithm M_1 and a polynomial p_1 such that for every $x \in L$, there exists $u_1 \in \{0,1\}^{p_1(|x|)}$ such that $M_1(x, u_1) = 1$ and for every $x \notin L$ and $u \in \{0,1\}^{p_1(|x|)}$ we have $M_1(x, u) = 0$. Similarly since $L \in \mathsf{coNP}$ there is an efficient

algorithm M_2 and polynomial p_2 such that for every $x \notin L$, there exists $u_2 \in \{0,1\}^{p_2(|x|)}$ such that $M_2(x, u_2) = 1$ and for every $x \in L$ and $u_2 \in \{0,1\}^{p_2(|x|)}$ we have $M_2(x, u_2) = 0$.

Now, given x, our quantum computer can generate a uniform superposition over all the possible certificates for both M_1 and M_2 (concatenated together), and postselect on the event that either $M(x, u_1) = 1$ or $M_2(x, u_2) = 1$. Then, the quantum algorithm can measure all registers and simulate both $M_1(x, u_1)$ and $M_2(x, u_2)$ and see which one outputs 1. By definition, only one of M_1 and M_2 will accept, and whichever one accepts tells us if $x \in L$ or not. □

It is widely believed that $\mathsf{NP} \cap \mathsf{coNP}$ is not contained in P or even BPP. As such, there is reason to believe that exact quantum algorithms with postselection can offer advantage over efficient classical computation.

4.7 Open Questions

In this chapter, we considered the problem of lower bounding the rational degree of Boolean functions in terms of their degree as multilinear polynomials. While we could not answer this question in its full generality, we showed that the square root of the degree lower bounds the rational degree for both symmetric and unate Boolean functions. We conjecture that this lower bound extends to all total Boolean functions.

Conjecture 95. *For all Boolean functions* $f\colon \{0,1\}^n \to \{0,1\}$, $\sqrt{\deg(f)} \leq \mathrm{rdeg}(f)$.

Answering this conjecture in the affirmative would place rational degree within a plethora of Boolean function complexity measures all of which are polynomially related. Recall that for partial functions, we have constant vs. $\Omega(n)$ separations between the rational and approximate degrees in each direction.

We showed that a hypothetical total function that witnesses any such separation must lack a certain level of structure: in particular, it cannot be symmetric, unate, or expressible by a low-depth read-once Boolean formula. In this direction, an easier question is whether there are other classes of functions (e.g., transitive-symmetric, etc.) for which the rational degree cannot be separated from the degree.

We also proved that almost all Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ have rational degree at least $n/2 - O(\sqrt{n})$. As mentioned in the preliminaries O'Donnell and Servedio proved [OSo8] that almost all Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ have sign degree at most $n/2 + O(\sqrt{n \log n})$. It would be interesting to know if a similar result can be established for the rational degree.

Conjecture 96. *All but a negligible fraction of Boolean functions* $f\colon \{0,1\}^n \to \{0,1\}$ *have rational degree at most* $n/2 + o(n)$.

Chapter 5

Conclusion

In this thesis we investigated quantum query algorithms from three different angles: design, optimality, and complexity. First, we investigated how divide-and-conquer strategies can be used to design new algorithms. Next, we saw how the optimality of the class of translation-invariant algorithms can be established for the ordered search problem. Lastly, we investigated the rational degree of Boolean functions, a measure that captures the complexity of exact, postselected quantum query algorithms. We provide a brief summary of the topics discussed.

In Chapter 2, we proposed a generic framework for developing new quantum algorithms using divide-and-conquer strategies. We demonstrated the utility of our framework by proposing near-optimal quantum algorithms for a number of string processing problems. In Chapter 3, we improved the best known upper bound on the (fine-grained) query complexity of solving the ordered search problem without error. By analyzing the symmetries inherent to the problem, we also proved that the class of translation-invariant algorithms introduced by Farhi, Goldstone, Gutmann, and Sipser [FGGS99] is optimal for the ordered search problem, in both the exact and bounded-error settings. Finally, in Chapter 4 we initiated a systematic study of the rational degree of Boolean functions. We established a variety of results in this chapter. For example, we gave sharp lower bounds (in terms of the degree) on the rational degree of symmetric and unate functions. We also showed that almost all n-bit Boolean functions have rational degree $n/2 - O(\sqrt{n})$. Additionally,

by proving that the nondeterministic degree composes exactly under AND, we constructed a degree-3/2 separation between the rational degree and the spectral sensitivity.

Looking beyond the scope of this thesis, I state two conjectures that provide a natural starting point for a continuation of this work. First, recall from Chapter 2 the improved upper bound of $O\left(\sqrt{n \log n}\right)$ on the complexity of deciding membership in the (star-free) regular language $\Sigma^* 20^* 2 \Sigma^*$. I expect that further improvements are possible, and that the query complexity in fact matches the lower bound $\Omega\left(\sqrt{n}\right)$. Moreover, I conjecture that the quantum query complexity of deciding membership in any star-free language is also $\Theta\left(\sqrt{n}\right)$. Second, consider the improved upper bound on the quantum query complexity of (exact) ordered search from Chapter 3. Here, too, I conjecture that the best known lower bound is essentially tight, i.e., that the quantum query complexity of exact, n-element ordered search is $\frac{1}{\pi} \ln n + O(1)$.

APPENDICES

Appendix A

A.1 Adversary composition lemmas

In this appendix, we prove two adversary compositions lemmas: Lemma 4 in Section A.1 and Lemma 5 in Section A.1. 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 97 (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

$$\gamma_2(A|Z) := \min_{\substack{d \in \mathbb{N}, \\ |u_{xj}\rangle, |v_{yj}\rangle \in \mathbb{C}^d}} \max\left\{ \max_{x \in D_1} \sum_{j=1}^{n} \| |u_{xj}\rangle \|^2, \max_{y \in D_2} \sum_{j=1}^{n} \| |v_{yj}\rangle \|^2 \right\}$$
$$\text{subject to:} \quad \forall x \in D_1, y \in D_2, \quad \sum_{j=1}^{n} (Z_j)_{xy} \langle u_{xj} | v_{yj} \rangle = A_{xy}.$$

We can view Adv$(\cdot)$ as a special case of the filtered γ_2 norm. Let $f\colon D \to E$ be a function, where $D \subseteq \mathscr{D}^n$ and E are finite sets. Let F be the Gram matrix of f. Let $\Delta := \{\Delta_1, \ldots, \Delta_n\}$ 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 $\mathrm{Adv}(f) = \gamma_2(J - F|\Delta')$, 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 98 ([Rei09a, Theorem 6.2]). *Let $f\colon D \to \{0,1\}$ be a Boolean function, where $D \subseteq \mathscr{D}^n$ is a finite set. Then*

$$\begin{aligned}\mathrm{Adv}(f) = \min_{\substack{d\in\mathbb{N},\\ |u_{xj}\rangle\in\mathbb{C}^d}} & \sqrt{A_0 \cdot A_1} \\ \text{subject to:}\quad \forall b \in \{0,1\},\quad A_b &= \max_{x\in f^{-1}(b)} \sum_{j=1}^{n} \| \, |u_{xj}\rangle \, \|^2. \\ \forall x,y \in D \text{ with } f(x)\neq f(y),\quad & \sum_{j\in[n]\colon x_j\neq y_j} \langle u_{xj}|u_{yj}\rangle = 1.\end{aligned}$$

Fact 99 ([LMR+11, Theorem 3.4]). *Let $f\colon D \to E$ be a function, where $D \subseteq \mathscr{D}^n$ and E are finite sets. Let F be the Gram matrix of f. Then*

$$\mathrm{Adv}(f) \leq \gamma_2(J - F|\Delta) \leq 2(1 - 1/|E|)\,\mathrm{Adv}(f).$$

Fact 100 (Triangle inequality [LMR+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*

$$\gamma_2(A + B|Z) \leq \gamma_2(A|Z) + \gamma_2(B|Z).$$

Fact 101 ([LMR+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*

$$\gamma_2(A \circ B \mid \{Z_j \circ B \mid j \in [n]\}) \leq \gamma_2(A \mid Z) \leq \gamma_2(A \mid \{Z_j \circ B \mid j \in [n]\}) \cdot \gamma_2(B).$$

Fact 102 (Direct-sum property [LMR+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 $Y \coloneqq \{Y_j \mid j \in [n]\}$ and $Z \coloneqq \{Z_j \mid j \in [n]\}$. Then*

$$\gamma_2(A \oplus B \mid \{Y_j \oplus Z_j \mid j \in [n]\}) = \max\{\gamma_2(A|Y), \gamma_2(B|Z)\}.$$

Proof of Lemma 4.

Lemma 4 (Adversary composition for OR and AND). *Let $a, b \in \mathbb{N}$ and let Σ be a finite set. Let $f_1 : \Sigma^a \to \{0,1\}$ and $f_2 : \Sigma^b \to \{0,1\}$. Suppose that $g^\wedge, g^\vee : \Sigma^a \times \Sigma^b \to \{0,1\}$ are 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*

$$\mathrm{Adv}(g^\wedge)^2 = \mathrm{Adv}(g^\vee)^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2)^2.$$

Moreover, suppose $a = b$ and $h^\wedge : \Sigma^a \to \{0,1\}$ satisfies $h^\wedge(x) = f_1(x) \wedge f_2(x)$ for all $x \in \Sigma^a$. Let f_2' denote the restriction $f_2' = f_2|_{f_1^{-1}(1)}$. Then $\mathrm{Adv}(h^\wedge)^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2')^2$.

Proof. It suffices to prove 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 $\mathrm{Adv}(f) = \mathrm{Adv}(\neg f)$ for any Boolean function $f : \Sigma^n \to \{0,1\}$, where $\neg$ denotes negation. For the rest of this proof, we write $g \coloneqq g^\vee$, $a_1 \coloneqq a$, and $a_2 \coloneqq b$ for notational convenience.

For $i \in \{1,2\}$, let $\{|u^i_{xj}\rangle \in \mathbb{C}^{d_i} \mid x \in \Sigma^{a_i}, j \in [a_i]\}$ be a minimum solution to the minimization problem in Fact 98 that specifies $\mathrm{Adv}(f_i)$. Then

$$\mathrm{Adv}(f_i)^2 = A^i_0 \cdot A^i_1, \quad \text{where } A^i_b \coloneqq \max_{x \in f_i^{-1}(b)} \sum_{j=1}^{a_i} \| |u^i_{xj}\rangle \|^2 \text{ for all } b \in \{0,1\},$$

$$\forall x, y \in \Sigma^{a_i} \text{ with } f_i(x) \neq f_i(y), \quad \sum_{j \in [a_i] \,:\, x_j \neq y_j} \langle u^i_{xj} | u^i_{yj} \rangle = 1.$$

By mapping $|u^i_{xj}\rangle \mapsto \sqrt{A^i_1}\,|u^i_{xj}\rangle$ for $x \in f_i^{-1}(0)$ and $|u^i_{xj}\rangle \mapsto |u^i_{xj}\rangle / \sqrt{A^i_1}$ for $x \in f_i^{-1}(1)$, we can assume without loss of generality that

$$A^i_0 = \mathrm{Adv}(f_i)^2 \quad \text{and} \quad A^i_1 = 1 \quad \text{for all } i \in \{1,2\}. \tag{A.1}$$

Now, given some $(x, x') \in \Sigma^{a_1} \times \Sigma^{a_2}$ 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 $k \le a_1$, then $|\lambda_{(x,x')k}\rangle := |u^1_{xk}\rangle \oplus 0$.

$(\mathbf{f_1(x)}, \mathbf{f_2(x')})$	$\mathbf{k \le a_1}$	$\mathbf{k > a_1}$
$(0,0)$	$\lvert u^1_{xk}\rangle \oplus 0$	$0 \oplus \lvert u^2_{x'k'}\rangle$
$(0,1)$	$0 \oplus 0$	$0 \oplus \lvert u^2_{x'k'}\rangle$
$(1,0)$	$\lvert u^1_{xk}\rangle \oplus 0$	$0 \oplus 0$
$(1,1)$	$\lvert u^1_{xk}\rangle \oplus 0$	$0 \oplus 0$

Then it can be directly verified that

$$\forall (x,x'),(y,y') \in \Sigma^{a_1} \times \Sigma^{a_2} \text{ with } g(x,x') \neq g(y,y') : \sum_{\substack{k \in [a_1+a_2]: \\ (x,x')_k \neq (y,y')_k}} \langle \lambda_{(x,x')k} | \lambda_{(y,y')k} \rangle = 1. \tag{A.2}$$

Moreover, from the definitions of g and $|\lambda_{(x,x')k}\rangle$, and Equation (A.1), we find

$$A_0 := \max_{(x,x') \in g^{-1}(0)} \sum_{k=1}^{a_1+a_2} \| \, |\lambda_{(x,x')k}\rangle \, \|^2 = \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2)^2,$$
$$A_1 := \max_{(x,x') \in g^{-1}(1)} \sum_{k=1}^{a_1+a_2} \| \, |\lambda_{(x,x')k}\rangle \, \|^2 = 1.$$

But Equation (A.2) means that

$$\left\{|\lambda_{(x,x')k}\rangle \in \mathbb{C}^{d_1} \oplus \mathbb{C}^{d_2} \mid (x,x') \in \Sigma^{a_1} \times \Sigma^{a_2}, k \in [a_1 + a_2]\right\}$$

is a feasible solution to the minimization problem in Fact 98 that specifies $\mathrm{Adv}(g)$. Therefore

$$\mathrm{Adv}(g)^2 \leq A_0 \cdot A_1 = \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2)^2, \tag{A.3}$$

which completes the proof of the first part of the lemma.

The proof of the "moreover" part is similar. Let $D \coloneqq f_1^{-1}(1) \subseteq \Sigma^a$. Let $|u^1_{xj}\rangle$, A^1_0, and A^1_1 be defined as above. Let $\{|u^{2\prime}_{xj}\rangle \in \mathbb{C}^{d_2'} \mid x \in D, j \in [a]\}$ be a minimum solution to the minimization problem in Fact 98 that specifies $\mathrm{Adv}(f_2')$. Then

$$\mathrm{Adv}(f_2')^2 = A^{2\prime}_0 \cdot A^{2\prime}_1, \quad \text{where } A^{2\prime}_b \coloneqq \max_{x \in f_2'^{-1}(b)} \sum_{j=1}^{a} \| |u^{2\prime}_{xj}\rangle \|^2 \text{ for all } b \in \{0,1\},$$

$$\forall x, y \in D \text{ with } f_2'(x) \neq f_2'(y), \quad \sum_{j \in [a] \,:\, x_j \neq y_j} \langle u^{2\prime}_{xj} | u^{2\prime}_{yj} \rangle = 1.$$

By a similar argument as above, we can assume without loss of generality that

$$A^1_0 = 1, \quad A^1_1 = \mathrm{Adv}(f_1)^2, \quad A^{2\prime}_0 = 1, \quad \text{and} \quad A^{2\prime}_1 = \mathrm{Adv}(f_2')^2. \tag{A.4}$$

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.

$(\mathbf{f_1(x)}, \mathbf{f_2(x)})$	$\mathbf{k \in [a]}$
$(0,0)$	$\lvert u^1_{xk}\rangle \oplus 0$
$(0,1)$	$\lvert u^1_{xk}\rangle \oplus 0$
$(1,0)$	$0 \oplus \lvert u^{2\prime}_{xk}\rangle$
$(1,1)$	$\lvert u^1_{xk}\rangle \oplus \lvert u^{2\prime}_{xk}\rangle$

Note that $|\mu_{xk}\rangle$ is well-defined since $|u^{2\prime}_{xk}\rangle$ only appear in rows with $f_1(x) = 1$.

Then it can be verified by inspection that

$$\forall x, y \in \Sigma^a \text{ with } h^\wedge(x) \neq h^\wedge(y), \quad \sum_{k \in [a] \,:\, x_k \neq y_k} \langle \mu_{xk} | \mu_{yk} \rangle = 1. \tag{A.5}$$

Moreover, from the definitions of $h^\wedge$ and $|\mu_{xk}\rangle$, and Equation (A.4), we find

$$\begin{aligned} B_0 &:= \max_{x \in h^{-1}(0)} \sum_{k=1}^{a} \| \, |\mu_{xk}\rangle \, \|^2 \leq 1, \\ B_1 &:= \max_{x \in h^{-1}(1)} \sum_{k=1}^{a} \| \, |\mu_{xk}\rangle \, \|^2 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2')^2. \end{aligned}$$

But Equation (A.5) means that

$$\left\{ |\mu_{xk}\rangle \in \mathbb{C}^{d_1} \oplus \mathbb{C}^{d_2'} \mid x \in \Sigma^a, k \in [a] \right\}$$

is a feasible solution to the minimization problem in Fact 98 that specifies $\mathrm{Adv}(h^\wedge)$. Therefore

$$\mathrm{Adv}(h^\wedge)^2 \leq B_0 \cdot B_1 \leq \mathrm{Adv}(f_1)^2 + \mathrm{Adv}(f_2')^2,$$

which completes the proof of the "moreover" part of the lemma. □

Proof of Lemma 5

Lemma 5 (Adversary composition for SWITCH-CASE). *Let $a \in \mathbb{N}$ and let Σ, Λ be finite sets. Let $f\colon \Sigma^a \to \Lambda$. For $s \in \Lambda$, let $g_s\colon \Sigma^a \to \{0,1\}$. Define $h\colon \Sigma^a \to \{0,1\}$ by $h(x) = g_{f(x)}(x)$. Then*

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

Proof. Define $h'\colon \Sigma^a \to \Lambda \times \{0,1\}$ 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

$$\begin{aligned}
\mathrm{Adv}(h) &\le \mathrm{Adv}(h') && \text{(Definition 2)} \\
&\le \gamma_2(J - H' \mid \Delta) && \text{(Fact 99)} \\
&\le \gamma_2(J - F \mid \Delta) + \gamma_2(F - H' \mid \Delta) && \text{(Fact 100)} \\
&\le O(\mathrm{Adv}(f)) + \gamma_2(F - H' \mid \Delta) && \text{(Fact 99).}
\end{aligned}$$

We proceed to bound $\gamma_2(F - H' \mid \Delta)$. 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 $\left(\Delta_j^{(b_s)}\right)_{xy} = 1 - \delta_{x_j, y_j}$ 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

$$H' = \bigoplus_{s\in\Lambda} \tilde{G}_s, \quad F = \bigoplus_{s\in\Lambda} J_{b_s}, \quad \text{and} \quad \Delta_j \circ F = \bigoplus_{s\in\Lambda} \Delta_j^{(b_s)}. \tag{A.6}$$

Therefore, we have

$$\begin{aligned}
\gamma_2(F - H' \mid \Delta) &\le \gamma_2(F - H' \mid \{\Delta_j \circ F \mid j \in [a]\}) \cdot \gamma_2(F) && \text{(Fact 101)} \\
&\le \gamma_2(F - H' \mid \{\Delta_j \circ F \mid j \in [a]\}) && (\gamma_2(F) \le 1) \\
&= \max_{s\in\Lambda} \gamma_2(J_{b_s} - \tilde{G}_s \mid \Delta^{(b_s)}) && \text{(Eq. A.6 \& Fact 102)} \\
&\le \max_{s\in\Lambda} \gamma_2(J - G_s \mid \Delta) = \max_{s\in\Lambda} \mathrm{Adv}(g_s).
\end{aligned}$$

The lemma follows. □

A.2 Randomized search

In this appendix, we equate the totally ordered set Σ with $\mathbb{Z}$ for convenience of presentation. It is clear that our randomized search can be generalized to any totally ordered set Σ.

We claim that $\mathsf{min\text{-}last}_{j,n}$ and $\mathsf{max\text{-}first}_{j,n}$ can be computed by a randomized search that uses $O(\log(n))$ computations of $\mathsf{LIS}_{j,n}$ and calls to Grover search. The claim follows from Lemma 103, which describes the randomized search, with the oracles $\mathscr{R}$ and $\mathscr{O}$ in its statement instantiated as follows.

(1) *Instantiating the oracle $\mathscr{R}$.* This is Grover search with marking function $r_1 < \cdot < r_2$. Note that Grover search does indeed return a uniformly random marked item as required to apply Lemma 103.

(2) *Instantiating the oracle $\mathscr{O}$.* Let $x \in \mathbb{Z}^n$ and $u \in \mathbb{Z}$. Recall that $\mathsf{LIS}_{j,n}(x)$ decides whether x contains a j-IS*. Table A.1 and Table A.2 show how an algorithm for computing $\mathsf{LIS}_{j,n}$ can be used to instantiate $\mathscr{O}$ for computing $\mathsf{min\text{-}last}_{j,n}$ and $\mathsf{max\text{-}first}_{j,n}$, 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 $\mathbf{i} \in [\mathbf{n}]$ such that $x[\mathbf{i}] > \mathbf{u}$ and compute $\mathsf{LIS}_{\mathbf{j},\mathbf{n}}(\mathbf{x})$	**Remove all $\mathbf{i} \in [\mathbf{n}]$ such that $\mathbf{x}[\mathbf{i}] \geq \mathbf{u}$ and compute $\mathsf{LIS}_{\mathbf{j},\mathbf{n}}(\mathbf{x})$**	**Conclusion**
0	0	$\mathsf{min\text{-}last}_{j,n}(x) > u$
0	1	Impossible
1	0	$\mathsf{min\text{-}last}_{j,n}(x) = u$
1	1	$\mathsf{min\text{-}last}_{j,n}(x) < u$

Table A.1: Instantiating $\mathscr{O}$ for computing $\mathsf{min\text{-}last}_{j,n}$ using $\mathsf{LIS}_{j,n}$.

Remove all $\mathbf{i} \in [\mathbf{n}]$ such that $\mathbf{x}[\mathbf{i}] < \mathbf{u}$ and compute $\mathsf{LIS}_{\mathbf{j},\mathbf{n}}(\mathbf{x})$	**Remove all $\mathbf{i} \in [\mathbf{n}]$ such that $\mathbf{x}[\mathbf{i}] \leq \mathbf{u}$ and compute $\mathsf{LIS}_{\mathbf{j},\mathbf{n}}(\mathbf{x})$**	**Conclusion**
0	0	$\mathsf{max\text{-}first}_{j,n}(x) < u$
0	1	Impossible
1	0	$\mathsf{max\text{-}first}_{j,n}(x) = u$
1	1	$\mathsf{max\text{-}first}_{j,n}(x) > u$

Table A.2: Instantiating $\mathcal{O}$ for computing $\mathsf{max\text{-}first}_{j,n}$ using $\mathsf{LIS}_{j,n}$.

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 103.

(2) Lemma 103 assumes that the oracles $\mathcal{O}$ and $\mathscr{R}$ 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 $\mathscr{R}$ are called $O(\log(n))$ times. However, techniques in [FRPU94] allow us to remove this $\log\log(n)$ factor.

Lemma 103 (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 $\mathscr{R}$ 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 $\mathscr{R}$ 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 $\mathscr{R}$ with input r_1 and r_2 to select an element u from $\{a \in S \mid r_1 < a < r_2\}$ uniformly at random.

Use $\mathscr{O}$ 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 there are multiple ss, then p can be taken as the rank of any). Then we have

$$\mathbb{E}[X_{t+1}|X_t] \leq \frac{1}{X_t}\Big(\sum_{i=1}^{p-1}(X_t - i) + \sum_{i=p}^{X_t-1} i\Big) \leq \frac{3}{4}X_t, \tag{A.7}$$

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 (A.7) 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 $\mathrm{prob}(X_t > 1) \leq (3/4)^t n \leq \delta$ 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 $1 - \delta$. □

Appendix B

B.1 Uniqueness of the Initial Matrix

Lemma 104. *The semidefinite program*

$$\operatorname{tr}_l A + \operatorname{tr}_{n-l} A = 1 \quad (0 \le l < n) \tag{B.1}$$

$$A \in \mathbf{S}_+^n \tag{B.2}$$

has the unique solution $A = \frac{1}{n} J_n$.

Proof. Since A is positive semidefinite, all of its diagonal entries (being principal minors) are nonnegative. For distinct $i, j \in \{0, \dots, n-1\}$, by considering the principal minor

$$0 \le \begin{vmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{vmatrix} = a_{ii}a_{jj} - a_{ij}^2,$$

we see that $a_{ij} \le \sqrt{a_{ii}a_{jj}}$. By the AM-GM inequality,

$$a_{ij} \le \sqrt{a_{ii}a_{jj}} \tag{B.3}$$

$$\le \frac{a_{ii} + a_{jj}}{2}. \tag{B.4}$$

Then by Equation (B.1),

$$\begin{aligned}
1 &= \operatorname{tr}_l A + \operatorname{tr}_{n-l} A \\
&= \sum_{j=0}^{n-1-l} a_{j,l+j} + \sum_{j=0}^{l-1} a_{j,n-l+j} \\
&\leq \frac{1}{2}\left(\sum_{j=0}^{n-l-1} (a_{jj} + a_{l+j,l+j}) + \sum_{j=0}^{l-1} (a_{jj} + a_{n-l+j,n-l+j}) \right) \\
&= \operatorname{tr} A \\
&= 1.
\end{aligned}$$

We conclude that Equations (B.3) and (B.4) hold with equality for all distinct $i, j \in \{0, \dots, n-1\}$. Therefore $a_{ii} = a_{jj} = 1/n$, and in particular $a_{ij} = 1/n$ as well. □

Appendix C

C.1 The Rational Degree and the Postselected Query Complexity

In this appendix, we show that the main theorem of [MdW14], stated for total Boolean functions, in fact also holds for partial ones. We do so simply by observing that the proof given in [MdW14] also works for partial Boolean functions.

Theorem 105. *For any (possibly partial) Boolean function f on n variables and $\varepsilon \in [0, 1/2)$,* $\mathrm{rdeg}_\varepsilon(f) = \Theta(\mathsf{PostQ}_\varepsilon(f))$.

First we show that rational degree lower bounds quantum query complexity with postselection.

Lemma 106. *Let $D \subseteq \{0,1\}^n$ and consider $f\colon D \to \{0,1\}$. Then* $\mathrm{rdeg}_\varepsilon(f) \leq 2\mathsf{PostQ}_\varepsilon(f)$.

Proof. Suppose there is a T-query postselected ε-error algorithm for f. As in Definition 60 let $a(x)$ be the random variable corresponding to the measurement outcome of the output qubit and $b(x)$ the random variable corresponding to the measurement outcome of the postselected qubit. We have that

$$\Big|\Pr[a(x) = 1 | b(x) = 1] - f(x)\Big| \leq \varepsilon.$$

Now, by [BBC+01], the coefficients of the state of the algorithm after T queries are polynomials in $x_1, \ldots, x_n$ of degree at most T. Thus the probabilities $p(x) := \Pr[a(x) \wedge b(x) = 1]$ and

$q(x) \coloneqq \Pr[b(x) = 1]$ are polynomials of degree at most $2T$. Thus we have

$$\left|\frac{p(x)}{q(x)} - f(x)\right| = \left|\Pr[a(x) = 1 | b(x) = 1] - f(x)\right| \leq \varepsilon,$$

which gives us the desired rational approximation of f. □

Now, we show that, up to a constant factor, the rational degree upper bounds quantum query complexity with postselection. The proof is Fourier analytic, and so we switch to the $\{-1, 1\}$ basis.

Lemma 107. *Let $D \subseteq \{-1, 1\}^n$ and consider $f\colon D \to \{-1, 1\}$. Then $\mathsf{PostQ}_\varepsilon(f) \leq \mathrm{rdeg}_\varepsilon(f)$.*

Proof. Suppose $f\colon D \to \{-1, 1\}^n$ has an ε-approximate rational representation p/q such that $\max\{\deg(p), \deg(q)\} = d$. Considering the Fourier expansions of p, q, we can construct a state proportional to

$$\sum_{S \subseteq [n]} \widehat{p}(S) |0\rangle |S\rangle + \widehat{q}(S) |1\rangle |S\rangle,$$

where $|S\rangle$ is the basis state that corresponds to the indicator bitstring for the set S. Then, using $\max\{\deg(p), \deg(q)\} = d$ queries to x, we obtain the state (up to normalization)

$$\sum_{S \subseteq [n]} \widehat{p}(S)\chi_S(x) |0\rangle |S\rangle + \widehat{q}(S)\chi_S(x) |1\rangle |S\rangle.$$

Now we apply an n-qubit Hadamard transform to the second register, obtaining a state proportional to

$$|0\rangle \left(\sum_{S \subseteq [n]} \widehat{p}(S) |0^n\rangle + \ldots \right) + |1\rangle \left(\sum_{S \subseteq [n]} \widehat{q}(S) |0^n\rangle + \ldots \right),$$

after which we postselect on the second register being equal to 0^n. This gives us a state proportional to

$$|0\rangle \left(\sum_{S \subseteq [n]} \widehat{p}(S) |0^n\rangle \right) + |1\rangle \left(\sum_{S \subseteq [n]} \widehat{q}(S) |0^n\rangle \right) = \big(p(x) |0\rangle + q(x) |1\rangle \big) |0^n\rangle.$$

After discarding the second register and normalizing, we are left with

$$\frac{p(x)}{p(x)^2 + q(x)^2} |0\rangle + \frac{q(x)}{p(x)^2 + q(x)^2} |1\rangle.$$

We measure this state in the Hadamard basis and interpret the result as having value in $\{-1, 1\}$. If $f(x) = -1$, then $\left|p(x)/q(x) + 1\right| \leq \varepsilon$. The probability of obtaining $+1$ is

$$\frac{(q(x) + p(x))^2}{2(p(x)^2 + q(x)^2)} = \frac{(1 + p(x)/q(x))^2}{2(p^2(x)/q^2(x) + 1)} \leq \frac{\varepsilon^2}{2((-1+\varepsilon)^2 + 1)} \leq \varepsilon.$$

A similar calculation shows that given $f(x) = 1$, the probability that we obtain -1 is less than ε. Thus, we conclude that there is an ε-error d-query postselected algorithm for f. □

Note that in the proof of Lemma 106 we get a rational polynomial which is bounded outside of the promise. As a consequence, we have that imposing this boundedness condition only increases the ϵ-approximate rational degree by at most a factor of 2.

C.2 Read-Once TC Formulae

In this appendix, we prove a structural result on read-once TC formulae. Recall that TC formulae are constructed from threshold gates. For convenience in our proof, we will consider the gate set as consisting of the NOT gate and all nonconstant monotone increasing Threshold gates. A nonconstant monotone increasing Threshold gate is a gate that computes a threshold function of the form $\mathrm{Thr}_k : \{0,1\}^m \to \{0,1\}$ where $m \in \mathbb{N}$, $k \in \{1, \ldots, m\}$ and $\mathrm{Thr}_k(x) = 1$ if and only if $|x| \geq k$.

More specifically, we show that any read-once TC formula on m disjoint variables restricts to either an AND or OR function on at least $\sqrt{m}$ distinct variables up to negating some input bits. As such, we make the following definition.

Definition 108 (AND- and OR-dimension)**.** Let $f : \{0,1\}^n \to \{0,1\}$. The AND-dimension of f, denoted $\dim_\wedge(f)$, is the largest positive integer m such that there exists a restriction ρ of f such that $f|_\rho : \{0,1\}^m \to \{0,1\}$ is equal to the AND function up to negating some input bits. The OR-dimension of f, denoted $\dim_\vee(f)$, is defined similarly using the OR function instead.

We now prove Proposition 109.

Proposition 109. *Suppose* $f\colon \{0,1\}^n \to \{0,1\}$ *is a read-once* TC *formula on* m *disjoint variables of* any *depth, then* $\dim_\wedge(f) \cdot \dim_\vee(f) \geq m$. *In particular,*

$$\mathrm{rdeg}(f) \geq \max(\dim_\wedge(f), \dim_\vee(f)) \geq \sqrt{m}.$$

Proof. The "in particular" part, follows from the first part by observing that Lemma 61 and Claim 70 imply $\mathrm{rdeg}(f) \geq \max(\dim_\wedge(f), \dim_\vee(f))$. Therefore $\mathrm{rdeg}(f) \geq \sqrt{m}$.

We proceed to prove the first part. The proof is by induction on the depth d of the tree T defining the read-once formula.

The proposition holds in the base case $d = 1$ since either

(1) $m = 1$ and f is the dictator or its negation, and $\max(\dim_\wedge(f), \mathrm{ndeg}(f)) = 1$; or

(2) $\mathrm{Thr}_k\colon \{0,1\}^m \to \{0,1\}$ is a restriction of f, so $\dim_\wedge(f) \geq k$ (by restricting $m-k$ bits of Thr_k to 0) and $\dim_\vee(f) \geq m-k+1$ (by restricting $k-1$ bits of Thr_k to 1). So in this case $\max(\dim_\wedge(f), \dim_\vee(f)) \geq k(m-k+1) \geq m$ as $k \in \{1,\ldots,m\}$.

For the induction step, assume that T has depth $d > 1$, and the proposition holds for all read-once formulae of depth less than d. There are again two cases:

(1) The root is labeled by the NOT gate. In this case, f can be computed by taking the NOT of an read-once formula on m distinct variables of depth $d-1$. By the inductive hypothesis, that read-once formula computes a function $f'\colon \{0,1\}^n \to \{0,1\}$ with $\dim_\wedge(f') \cdot \dim_\vee(f') \geq m$. Since f is the NOT of f', De Morgan's laws and the definition of the AND- and OR-dimensions imply that $\dim_\wedge(f) \geq \dim_\vee(f')$ and $\dim_\vee(f) \geq \dim_\wedge(f')$. Therefore $\dim_\wedge(f) \cdot \dim_\vee(f) \geq m$, which establishes the induction step.

(2) The root is labeled by a Threshold gate that computes $\mathrm{Thr}_k\colon \{0,1\}^a \to \{0,1\}$. Let $T_1,\ldots,T_a$ denote the a read-once formulae feeding into the root. Each T_i is on some number $m_i \geq 1$ of distinct variables and of depth at most $d-1$. T_j and T_k do not share variables for any $j \neq k$ and $\sum_{i=1}^a m_i = m$. Then each T_i can be viewed as computing a nonconstant function

$f_i : \{0,1\}^{m_i} \to \{0,1\}$ such that $\mathrm{Thr}_k(f_1, \ldots, f_a) : \{0,1\}^m \to \{0,1\}$ is a restriction of f. By the inductive hypothesis, $\dim_\wedge(f_i) \cdot \dim_\vee(f_i) \geq m_i$.

Let $A_1 \geq \ldots \geq A_a$ denote the multiset $\{\dim_\wedge(f_i) : i \in [a]\}$ sorted in descending order. Similarly, let $O_1 \geq \ldots \geq O_a$ denote the multiset $\{\dim_\vee(f_i) : i \in [a]\}$ in descending order. Since the f_i functions are all nonconstant, we can appropriately restrict f as explained in the base case to deduce

$$\dim_\wedge(f) \geq A_1 + \cdots + A_k \quad \text{and} \quad \dim_\vee(f) \geq O_1 + \cdots + O_{a-k+1}. \tag{C.1}$$

We now observe the following fact whose short proof we defer to after this proof.

Fact 110. *Let $n \in \mathbb{N}$. Let $x_1, x_2, \ldots, x_n \geq 0$ and $y_1, y_2, \ldots, y_n \geq 0$. Suppose $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_n$ are the x_is sorted in descending order and $\tilde{y}_1 \geq \tilde{y}_2 \geq \cdots \geq \tilde{y}_n \geq 0$ are the y_is sorted in descending order. Then, for any $k \in [n]$,*

$$(\tilde{x}_1 + \tilde{x}_2 + \cdots + \tilde{x}_k)(\tilde{y}_1 + \tilde{y}_2 + \cdots + \tilde{y}_{n-k+1}) \geq x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

Therefore

$$\begin{aligned}
\dim_\wedge(f) \cdot \dim_\vee(f) &\geq \sum_{i=1}^{a} \dim_\wedge(f_i) \cdot \dim_\vee(f_i) && \text{(Equation (C.1) and Fact 110)} \\
&\geq \sum_{i=1}^{a} m_i && \text{(Inductive hypothesis)} \\
&= m,
\end{aligned}$$

which establishes the induction step.

The proposition follows. □

Proof of Fact 110. It suffices to prove that

$$(\tilde{x}_1 + \tilde{x}_2 + \cdots + \tilde{x}_k)(\tilde{y}_1 + \tilde{y}_2 + \cdots + \tilde{y}_{n-k+1}) \geq \tilde{x}_1\tilde{y}_1 + \tilde{x}_2\tilde{y}_2 + \cdots + \tilde{x}_n\tilde{y}_n \tag{C.2}$$

from which the lemma follows by the rearrangement inequality.

To prove Equation (C.2), it suffices to prove it for any $k \leq \lceil n/2 \rceil$ by the symmetry between the $\tilde{x}_i$s and $\tilde{y}_i$s. We may also assume $k \geq 2$ since the case $k = 1$ is clearly true.

For $2 \leq k \leq \lceil n/2 \rceil$, we have

$$\begin{aligned}
&(\tilde{x}_1 + \tilde{x}_2 + \cdots + \tilde{x}_k)(\tilde{y}_1 + \tilde{y}_2 + \cdots + \tilde{y}_{n-k+1}) \\
\geq &(\tilde{x}_1 + \tilde{x}_2)(\tilde{y}_1 + \tilde{y}_2 + \cdots + \tilde{y}_{\lfloor n/2 \rfloor+1}) && (\tilde{x}_i, \tilde{y}_i \geq 0) \\
\geq &\tilde{x}_1(\tilde{y}_1 + \tilde{y}_2 + \cdots + \tilde{y}_{\lfloor n/2 \rfloor+1}) + \tilde{x}_{\lfloor n/2 \rfloor+2}(\tilde{y}_1 + \tilde{y}_2 + \cdots + \tilde{y}_{\lfloor n/2 \rfloor+1}) \\
\geq &\tilde{x}_1\tilde{y}_1 + \tilde{x}_2\tilde{y}_2 + \cdots + \tilde{x}_n\tilde{y}_n,
\end{aligned}$$

as required. □

Bibliography

[ABK16] S. Aaronson, S. Ben-David, and R. Kothari. "Separations in query complexity using cheat sheets". In: *STOC'16—Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, New York, 2016, pp. 863–876. doi:10.1145/2897518.2897644.

[ABK+21] S. Aaronson, S. Ben-David, R. Kothari, S. Rao, and A. Tal. "Degree vs. Approximate Degree and Quantum Implications of Huang's Sensitivity Theorem". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 1330–1342. doi:10.1145/3406325.3451047. arXiv:2010.12629.

[AGS19] S. Aaronson, D. Grier, and L. 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. arXiv:1812.04219.

[AK] S. Aaronson and G. Kuperberg. *Complexity Zoo*. https://complexityzoo.net/Complexity_Zoo.

[AS04] S. Aaronson and Y. Shi. "Quantum Lower Bounds for the Collision and the Element Distinctness Problems". In: *J. ACM* 51.4 (2004), pp. 595–605. doi:10.1145/1008731.1008735.

[Abb19] A. Abboud. "Fine-Grained Reductions and Quantum Speedups for Dynamic Programming". In: *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). 2019, 8:1–8:13. doi:10.4230/LIPIcs.ICALP.2019.8.

[ABW15] A. Abboud, A. Backurs, and V. V. 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. arXiv:1501.07053.

[AVDB18] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. "A rewriting system for convex optimization problems". In: *Journal of Control and Decision* 5.1 (2018), pp. 42–60. doi:10.1080/23307706.2017.1397554. arXiv:1709.04494.

[AAKV01] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. "Quantum walks on graphs". In: *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*. STOC '01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 50–59. doi:10.1145/380752.380758.

[ADZ93] Y. Aharonov, L. Davidovich, and N. Zagury. "Quantum random walks". In: *Phys. Rev. A* 48 (2 Aug. 1993), pp. 1687–1690. doi:10.1103/PhysRevA.48.1687.

[AJ23] S. Akmal and C. Jin. "Near-Optimal Quantum Algorithms for String Problems". In: *Algorithmica* 85.8 (Aug. 1, 2023), pp. 2260–2317. doi:10.1007/s00453-022-01092-x. arXiv:2110.09696.

[AV21] S. Akmal and V. Vassilevska Williams. "Improved Approximation for Longest Common Subsequence over Small Alphabets". In: *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 198. Leibniz International Proceedings in Informatics (LIPIcs). 2021, 13:1–13:18. doi:10.4230/LIPIcs.ICALP.2021.13. arXiv:2105.03028.

[AD99] D. Aldous and P. 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.

[ABB+23] J. Allcock, J. Bao, A. Belovs, T. Lee, and M. Santha. *On the quantum time complexity of divide and conquer*. 2023. arXiv:2311.16401.

[AW21] J. Alman and V. V. Williams. "A Refined Laser Method and Faster Matrix Multiplication". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2021, pp. 522–539. doi:10.1137/1.9781611976465.32. arXiv:2010.05846.

[Alo93] N. Alon. *Personal communication to M. Saks*. Reported in M. Saks, Slicing the Hypercube. 1993.

[Amb99] A. Ambainis. "A Better Lower Bound for Quantum Algorithms Searching an Ordered List". In: *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*. FOCS '99. IEEE Computer Society, 1999, p. 352. doi:10.1109/SFFCS.1999.814606. arXiv:quant-ph/9902053.

[Amb00] A. Ambainis. "Quantum lower bounds by quantum arguments". In: *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*. STOC '00. Association for Computing Machinery, 2000, pp. 636–643. doi:10.1145/335305.335394. arXiv:quant-ph/0002066.

[Amb07] A. Ambainis. "Quantum Walk Algorithm for Element Distinctness". In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239. doi:10.1137/S0097539705447311. arXiv:quant-ph/0311001.

[Amb13] A. Ambainis. "Superlinear advantage for exact quantum algorithms". In: *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC)*. 2013, pp. 891–900. doi:10.1145/2488608.2488721.

[ABB+17] A. Ambainis, K. Balodis, A. Belovs, T. Lee, M. Santha, and J. Smotrovs. "Separations in query complexity based on pointer functions". In: *J. ACM* 64.5 (2017), Art. 32, 24. doi:10.1145/3106234.

[ABI+20] A. Ambainis, K. Balodis, J. Iraids, K. Khadiev, V. Kļevickis, K. Prūsis, Y. Shen, J. Smotrovs, and J. Vihrovs. "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. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 8:1–8:14. doi:10.4230/LIPIcs.MFCS.2020.8. arXiv:2007.03402.

[ABI+19] A. Ambainis, K. Balodis, J. Iraids, M. Kokainis, K. Prūsis, and J. 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. arXiv:1807.05209.

[AB23] A. Ambainis and A. Belovs. "An Exponential Separation between Quantum Query Complexity and the Polynomial Degree". In: *Proceedings of the 38th Annual Conference on Computational Complexity (CCC)*. 2023. doi:10.4230/LIPIcs.CCC.2023.24. arXiv:2301.09218.

[AKO10] A. Andoni, R. Krauthgamer, and K. 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. arXiv:1005.4033.

[Ant95] M. Anthony. "Classification by polynomial surfaces". In: *Discrete Applied Mathematics* 61.2 (1995), pp. 91–103. doi:10.1016/0166-218X(94)00008-2.

[ApS25] M. ApS. *MOSEK Optimizer API for Python 11.0.7*. 2025.

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. 1st. USA: Cambridge University Press, 2009. doi:10.5555/1540612.

[BBG+22] K. Balodis, S. Ben-David, M. Göös, S. Jain, and R. Kothari. "Unambiguous DNFs and Alon-Saks-Seymour". In: *Proceedings of the 2021 IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 116–124. doi:10.1109/FOCS52979.2021.00020.

[BS21] N. Bansal and M. Sinha. "k-Forrelation optimally separates quantum and classical query complexity". In: *Proceedings of the 53rd ACM SIGACT Symposium on Theory of Computing (STOC)*. 2021, pp. 1303–1316. doi:10.1145/3406325.3451040.

[BSS03] H. Barnum, M. Saks, and M. Szegedy. "Quantum Query Complexity and Semi-Definite Programming". In: *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.* 2003, pp. 179–193. doi:10.1109/CCC.2003.1214419.

[BBC+01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. d. Wolf. "Quantum lower bounds by polynomials". In: *Journal of the ACM* 48.4 (2001), pp. 778–797. arXiv:quant-ph/9802049.

[dB15] N. de Beaudrap. *On exact counting and quasi-quantum complexity*. 2015. arXiv:1509.07789.

[Bel12a] A. 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. arXiv:1205.1534.

[Bel12b] A. Belovs. "Span Programs for Functions with Constant-Sized 1-Certificates". In: *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*. 2012, pp. 77–84. doi:10.1145/2213977.2213985. arXiv:1105.4024.

[Bel14] A. Belovs. "Quantum Algorithms for Learning Symmetric Juntas via Adversary Bound". In: *Proceedings of the 29th IEEE Conference on Computational Complexity (CCC)*. 2014, pp. 22–31. doi:10.1109/CCC.2014.11. arXiv:1311.6777.

[BR12] A. Belovs and B. 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. arXiv:1203.2603.

[BHT17] S. Ben-David, P. Hatami, and A. Tal. "Low-sensitivity functions from unambiguous certificates". In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. Vol. 67. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, 28:1–28:23. doi:10.4230/LIPIcs.ITCS.2017.28.

[BH07] M. Ben-Or and A. Hassidim. *Quantum Search in an Ordered List via Adaptive Learning*. 2007. arXiv:quant-ph/0703231.

[BHS80] J. L. Bentley, D. Haken, and J. 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.

[BHMT02] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. "Quantum amplitude amplification and estimation". In: *Contemporary Mathematics* 305 (2002), pp. 53–74. doi:10.1090/conm/305/05215. arXiv:quant-ph/0005055.

[BJL04] E. M. Brookes, M. B. Jacokes, and A. J. Landahl. *An improved quantum algorithm for searching an ordered list*. 2004.

[BdW99] H. Buhrman and R. de Wolf. "A lower bound for quantum search of an ordered list". In: *Information Processing Letters* 70.5 (1999), pp. 205–209. doi:10.1016/S0020-0190(99)00069-1.

[BdW02] H. Buhrman and R. de Wolf. "Complexity measures and decision tree complexity: a survey". In: *Theoretical Computer Science* 288.1 (2002), pp. 21–43. doi:10.1016/S0304-3975(01)00144-X.

[BDH+05] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. "Quantum Algorithms for Element Distinctness". In: *SIAM Journal on Computing* 34.6 (2005), pp. 1324–1330. doi:10.1137/S0097539702402780. arXiv:quant-ph/0007016.

[BKT20] M. Bun, R. Kothari, and J. Thaler. "The Polynomial Method Strikes Back: Tight Quantum Query Bounds via Dual Polynomials". In: *Theory of Computing* 16.10 (2020), pp. 1–71. doi:10.4086/toc.2020.v016a010.

[BT20] M. Bun and J. Thaler. "A nearly optimal lower bound on the approximate degree of AC^0". In: *SIAM J. Comput.* 49.4 (2020), pp. 59–96. doi:10.1137/17M1161737.

[CCKS25] J. Carolan, A. M. Childs, M. Kovacs-Deak, and L. Schaeffer. *Translation-Invariant Quantum Algorithms for Ordered Search are Optimal*. 2025. arXiv:2503.21090.

[CKK+25] A. Childs, R. Kothari, M. Kovacs-Deak, A. Sundaram, and D. Wang. "Quantum Divide and Conquer". In: *ACM Transactions on Quantum Computing* 6.2 (Apr. 2025). doi:10.1145/3723884. arXiv:2210.06419.

[CCD+03] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman. "Exponential algorithmic speedup by a quantum walk". In: *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*. New York, NY, USA: Association for Computing Machinery, 2003, pp. 59–68. doi:10.1145/780542.780552.

[CLP07] A. M. Childs, A. J. Landahl, and P. A. Parrilo. "Quantum Algorithms for the Ordered Search Problem via Semidefinite Programming". In: *Physical Review A* 75.3 (2007). doi:10.1103/physreva.75.032335. arXiv:quant-ph/0608161.

[CL08] A. M. Childs and T. Lee. "Optimal Quantum Adversary Lower Bounds for Ordered Search". In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2008, pp. 869–880. doi:10.1007/978-3-540-70575-8_71. arXiv:0708.3396.

[CT65] J. W. Cooley and J. 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.

[CW90] D. Coppersmith and S. 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.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. en. Third. MIT Press, 2009.

[CJOP20] A. Cornelissen, S. Jeffery, M. Ozols, and A. Piedrafita. "Span Programs and Quantum Time Complexity". In: *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*. Vol. 170. Leibniz International Proceedings in Informatics (LIPIcs). 2020, 26:1–26:14. doi:10.4230/LIPIcs.MFCS.2020.26. arXiv:2005.01323.

[Cov65] T. M. Cover. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14.3 (1965), pp. 326–334. doi:10.1109/PGEC.1965.264137.

[Deu85] D. Deutsch. "Quantum theory, the Church-Turing principle and the universal quantum computer". In: *Proceedings of the Royal Society of London Series A* 400.1818 (July 1985), pp. 97–117. doi:10.1098/rspa.1985.0070.

[DB16] S. Diamond and S. Boyd. "CVXPY: A Python-embedded modeling language for convex optimization". In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5. arXiv:1603.00943.

[Dum07] B. Dumitrescu. *Positive Trigonometric Polynomials and Signal Processing Applications*. Springer Publishing Company, Incorporated, 2007. doi:10.5555/1512941.

[DH96] C. Dürr and P. Høyer. *A Quantum Algorithm for Finding the Minimum*. 1996. arXiv:quant-ph/9607014.

[FGGS98] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. *A Limit on the Speed of Quantum Computation for Insertion into an Ordered List*. 1998. arXiv:quant-ph/9812057.

[FGGS99] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. *Invariant Quantum Algorithms for Insertion into an Ordered List*. 1999. arXiv:quant-ph/9901059.

[FG98] E. Farhi and S. Gutmann. "Quantum computation and decision trees". In: *Phys. Rev. A* 58 (2 Aug. 1998), pp. 915–928. doi:10.1103/PhysRevA.58.915.

[FRPU94] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. "Computing with Noisy Information". In: *SIAM Journal on Computing* 23.5 (1994), pp. 1001–1018. doi:10.1137/S0097539791195877.

[Fej16] L. Fejér. "Über trigonometrische Polynome." In: *Journal für die reine und angewandte Mathematik* 146 (1916), pp. 53–82.

[For03] L. Fortnow. *Computational Complexity - Rational Functions and Decision-Tree Complexity*. https://blog.computationalcomplexity.org/2003/11/rational-functions-and-decision-tree.html. 2003.

[Fre75] M. 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.

[FSS81] M. Furst, J. B. Saxe, and M. Sipser. "Parity, circuits, and the polynomial-time hierarchy". In: *22nd Annual Symposium on Foundations of Computer Science*. 1981, pp. 260–270. doi:10.1109/SFCS.1981.35.

[GSS16] J. Gilmer, M. Saks, and S. Srinivasan. "Composition limits and separating examples for some boolean function complexity measures". In: *Combinatorica* 36.3 (2016), pp. 265–311. doi:10.1007/s00493-014-3189-x.

[GKMV21] A. Glos, M. Kokainis, R. Mori, and J. 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. Leibniz International Proceedings in Informatics (LIPIcs). 2021, 50:1–50:23. doi:10.4230/LIPIcs.MFCS.2021.50. arXiv:2104.14384.

[GJPW18] M. Göös, T. S. Jayram, T. Pitassi, and T. Watson. "Randomized communication versus partition number". In: *ACM Trans. Comput. Theory* 10.1 (2018), Art. 4, 20. doi:10.1145/3170711.

[GPW18] M. Göös, T. Pitassi, and T. Watson. "Deterministic communication vs. partition number". In: *SIAM J. Comput.* 47.6 (2018), pp. 2435–2450. doi:10.1137/16M1059369.

[Gro96] L. K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC)*. Philadelphia, Pennsylvania, USA, 1996, pp. 212–219. doi:10.1145/237814.237866. arXiv: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.

[HNS02] P. Høyer, J. Neerbek, and Y. Shi. "Quantum complexities of ordered searching, sorting, and element distinctness". In: *Algorithmica* 34.4 (2002). Preliminary version in ICALP 2001, pp. 429–448. doi:10.1007/s00453-002-0976-3. arXiv:quant-ph/0102078.

[HLŠ07] P. Høyer, T. Lee, and R. Špalek. "Negative Weights Make Adversaries Stronger". In: *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*. San Diego, California, USA, 2007, pp. 526–535. doi:10.1145/1250790.1250867. arXiv:quant-ph/0611054.

[Hua19] H. Huang. "Induced subgraphs of hypercubes and a proof of the sensitivity conjecture". In: *Ann. of Math. (2)* 190.3 (2019), pp. 949–955. doi:10.4007/annals.2019.190.3.6. arXiv:1907.00847.

[IJK+25] V. Iyer, S. Jain, R. Kothari, M. Kovacs-Deak, V. M. Kumar, L. Schaeffer, D. Wang, and M. Whitmeyer. *On the Rational Degree of Boolean Functions and Applications*. 2025. arXiv:2310.08004.

[Jef22] S. Jeffery. "Span Programs and Quantum Space Complexity". In: *Theory of Computing* 18.11 (2022), pp. 1–49. doi:10.4086/toc.2022.v018a011. arXiv:1908.04232.

[JP24] S. Jeffery and G. Pass. *Multidimensional Quantum Walks, Recursion, and Quantum Divide & Conquer*. 2024. arXiv:2401.08355.

[KO63] A. A. Karatsuba and Y. Ofman. "Multiplication of Multidigit Numbers on Automata". In: *Soviet Physics Doklady* 7 (1963), pp. 595–596.

[Kim13] S. 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] V. Kļevickis, K. Prūsis, and J. Vihrovs. "Quantum Speedups for Treewidth". In: *17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022)*. Vol. 232. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 11:1–11:18. doi:10.4230/LIPIcs.TQC.2022.11. arXiv:2202.08186.

[Knu98] D. E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., 1998. doi:10.5555/280635.

[Kol88] J. Kollár. "Sharp Effective Nullstellensatz". In: *Journal of the American Mathematical Society* 1.4 (1988), pp. 963–975. doi:10.2307/1990996.

[LMS98] G. M. Landau, E. W. Myers, and J. P. Schmidt. "Incremental String Comparison". In: *SIAM J. Comput.* 27.2 (1998), pp. 557–582. doi:10.1137/S0097539794264810.

[LMS17] T. Lee, F. Magniez, and M. 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] T. Lee, R. Mittal, B. W. Reichardt, and R. Špalek. *An adversary for algorithms*. 2010. arXiv:1011.3020v1.

[LMR+11] T. Lee, R. Mittal, B. W. Reichardt, R. Špalek, and M. 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.

[MdW14] U. Mahadev and R. de Wolf. "Rational Approximations and Quantum Algorithms with Postselection". In: *Quantum Info. Comput.* 15.3–4 (2015), pp. 295–307. doi:10.48550/ARXIV.1401.0912. arXiv:1401.0912.

[MP69] M. Minsky and S. Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969.

[NW99] A. Nayak and F. Wu. "The quantum query complexity of approximating the median and related statistics". In: *Annual ACM Symposium on Theory of Computing (Atlanta, GA, 1999)*. ACM, New York, 1999, pp. 384–393. doi:10.1145/301250.301349.

[NC11] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. USA: Cambridge University Press, 2011.

[Nis91] N. Nisan. "CREW PRAMs and Decision Trees". In: *SIAM Journal on Computing* 20.6 (1991), pp. 999–1007. doi:10.1137/0220062.

[NS94] N. Nisan and M. Szegedy. "On the degree of Boolean functions as real polynomials". In: *computational complexity* 4.4 (1994), pp. 301–313. doi:10.1007/BF01263419.

[NW95] N. Nisan and A. Wigderson. "On rank vs. communication complexity". In: *Combinatorica* 15.4 (Dec. 1, 1995), pp. 557–565. doi:10.1007/BF01192527.

[ODo14] R. O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, New York, 2014, pp. xx+423. doi:10.1017/CBO9781139814782. arXiv:2105.10386.

[OSo8] R. O'Donnell and R. A. Servedio. "Extremal properties of polynomial threshold functions". In: *Journal of Computer and System Sciences* 74.3 (2008). Computational Complexity 2003, pp. 298–312. doi:10.1016/j.jcss.2007.06.021.

[RVo3] 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.

[RŠ12] B. Reichardt and R. 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. arXiv:0710.2630.

[Rei09a] B. 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] B. W. Reichardt. *Span-program-based quantum algorithm for evaluating unbalanced formulas*. 2009. arXiv:0907.1622.

[Rei11] B. W. Reichardt. "Reflections for Quantum Query Algorithms". In: *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. San Francisco, California, 2011, pp. 560–569. doi:10.5555/2133036.2133080. arXiv:1005.1601.

[Rie16] F. Riesz. "Über ein Problem des Herrn Carathéodory". In: *Journal für die reine und angewandte Mathematik* 146 (1916), pp. 83–87.

[RSSS19] A. Rubinstein, S. Seddighin, Z. Song, and X. 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. arXiv:2111.10538.

[Rub95] D. Rubinstein. "Sensitivity vs. block sensitivity of Boolean functions". In: *Combinatorica* 15.2 (1995), pp. 297–299.

[SS10] M. 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. arXiv:1308.0626.

[Sak93] M. E. Saks. “Slicing the hypercube”. In: *Surveys in Combinatorics, 1993*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1993, pp. 211–256. doi:10.1017/CBO9780511662089.009.

[SSW23] A. A. Sherstov, A. A. Storozhenko, and P. Wu. “An optimal separation of randomized and quantum query complexity”. In: *SIAM J. Comput.* 52.2 (2023), pp. 525–567. doi:10.1137/22M1468943.

[Sho94] P. Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*. 1994, pp. 124–134. doi:10.1109/SFCS.1994.365700.

[Str69] V. Strassen. “Gaussian elimination is not optimal”. In: *Numerische Mathematik* 13.4 (Aug. 1, 1969), pp. 354–356. doi:10.1007/BF02165411.

[Tre96] L. N. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*. Unpublished text, available at http://people.maths.ox.ac.uk/trefethen/pdetext.html. 1996.

[WF74] R. A. Wagner and M. J. Fischer. “The String-to-String Correction Problem”. In: *J. ACM* 21.1 (1974), pp. 168–173. doi:10.1145/321796.321811.

[Wan22] Q. Wang. *A Note on Quantum Divide and Conquer for Minimal String Rotation*. 2022. arXiv:2210.09149.

[dW00] R. de Wolf. “Characterization of non-deterministic quantum query and quantum communication complexity”. In: *Proceedings of the 15th Annual Conference on Computational Complexity (CCC)*. IEEE Computer Society, 2000, pp. 271–278. doi:10.1109/CCC.2000.856758. arXiv:cs/0001014.

[vZR97] J. von Zur Gathen and J. R. Roche. “Polynomials with two values”. In: *Combinatorica* 17.3 (Sept. 1, 1997), pp. 345–362. doi:10.1007/BF01215917.