Quantum Algorithms, Lower Bounds, and Time-Space Tradeoffs

Robert Špalek

Quantum Algorithms, Lower Bounds, and Time-Space Tradeoffs

ILLC Dissertation Series DS-2006-04

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: `illc@science.uva.nl`
homepage: `http://www.illc.uva.nl/`

Quantum Algorithms, Lower Bounds, and Time-Space Tradeoffs

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.mr. P.F. van der Heijden
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Aula der Universiteit
op donderdag 7 september 2006, te 12.00 uur

door

Robert Špalek

geboren te Žilina, Slowakije.

Cover design by Roman Kozubík.
Printed and bound by PrintPartners Ipskamp.

ISBN-10: 90–5776–155–6
ISBN-13: 978–90–5776–155–3

The highest technique is to have no technique.

Bruce Lee

Contents

List of Figures

Acknowledgments

First, I would like to thank my advisor Harry Buhrman. After supervising my Master's Thesis, he offered me a PhD position. He was always a good source of interesting new problems, and let me work on topics that I liked. Second, I want to thank my co-advisor Ronald de Wolf. He spent much time in discussions with me, proofread all my papers and always had plenty of comments, refined my style, corrected my English, and drank many bottles of good wine with me.

I thank my coauthors: Andris Ambainis (2×), Harry Buhrman, Peter Høyer (3×), Hartmut Klauck, Troy Lee, Mario Szegedy, and Ronald de Wolf (2×), for the wonderful time spent in our research. It was very exciting to collaborate on frontier research. I also thank the members of my thesis committee: Pavel Pudlák, Lex Schrijver, Mario Szegedy, and Paul Vitányi.

A lot of my results was done in different time-zones. I thank the following people for their hospitality, for fruitful scientific discussions, or both. My first steps overseas went to the Rutgers University in New Jersey in the autumn 2004, and I thank André Madeira, Martin Pál, Peter Richter, Xiaoming Sun, and Mario Szegedy and his family, among others for carrying me everywhere in their cars. In the summer 2005, I visited two research groups in Canada. I thank Scott Aaronson, Andris Ambainis, Elham Kashefi, and Ashwin Nayak from the University of Waterloo, and Dmitry Gavinsky, Peter Høyer, Mehdi Mhalla, Barry Sanders, Andrew Scott, Jonathan Walgate, and John Watrous from the University of Calgary for a wonderful time, and the quantum physics group in Calgary for stunning trips to the Rocky Mountains. At the end of 2005, I visited LRI in Orsay, and spent a great week in Paris thanks to Julia Kempe, Sophie Laplante, Frédéric Magniez, and Miklos Santha. I thank Ben Reichardt and Umesh Vazirani for discussions at UC Berkeley during my round-the-world trip in 2006. I learned a lot during my one-month stay at the quantum information group of the University of Queensland in Brisbane, and I thank Juliet Aylmer, Sean Barrett, Aggie Branczyk, Jennifer Dodd, Andrew Doherty, Mark Dowling, Mile Gu, Nick Menicucci, Michael Nielsen, Roman Orus, Peter

Rohde, and Guifré Vidal, for a warm welcome in their group and a lot of social activities.

Back in Amsterdam, I thank Troy Lee and Falk Unger, who often went climbing with me, brought me into salsa, and generally helped my last 4 years to become one long neverending party. I also thank all other friends from our hallway: Manuel Ballester, Harry Buhrman, Rudi Cilibrasi, Tim van Erven, Mart de Graaf, Peter Grünwald, Hartmut Klauck, Wouter Koolen-Wijkstra, Michal Koucký, Zvi Lotker, Steven de Rooij, Hein Röhrig, Caspar Treijtel, John Tromp, Paul Vitányi, Stephanie Wehner, and Ronald de Wolf, for interesting discussions we had over the years.

With the risk of forgetting someone, I thank Luis Antunes, Dave Bacon, Piotr Berman, Amit Chakrabarti, Matthias Christandl, Scott Diehl, Chris Dürr, Steve Fenner, Lance Fortnow, Fred Green, Sean Hallgren, Johan Håstad, Steve Homer, Karol Horodecki, Piotr Indyk, Valentine Kabanets, Marek Karpinski, Eike Kiltz, Jan Krajíček, Jan Kratochvíl, Dieter van Melkebeek, Chris Moore, Mike Mosca, Ilan Newman, Pavel Pudlák, Oded Regev, Pranab Sen, Jiří Sgall, Yaoyun Shi, Mikkel Thorup, Kolya Vereshchagin, Andreas Winter, and Shengyu Zhang, for discussions and inspiration.

Finally, I thank my parents, sister, family, and friends for support and distraction from science.

Robert Špalek
July 2006, Amsterdam

Chapter 1

Quantum Computation

A little more than a century ago, before the birth of quantum physics, the physical description of the world was, as we would say today, classical. Every particle had a defined position and velocity, and one could, in principle, predict the state of a closed physical system at any future moment from the complete description of its current state. Although the physical description of the world seemed nearly complete, some experiments were still incompatible with the physical theories of that age. In fact, it turned out that the whole foundation of physics had to be changed to fully explain these phenomena.

Old quantum theory The first such phenomenon was *black body radiation*, studied from 1850. According to classical physics, the amount of energy radiated out of an ideal black body in thermal equilibrium would be infinite. This follows from summing up the contributions of all possible wave forms. This obvious inconsistency was resolved by Planck in 1900 [Pla01], who postulated that energy is *quantized*, that is it comes in discrete chunks instead of a continuous range of energies. In 1905, Einstein explained the *photoelectric effect* [Ein05] by postulating that light also comes in *quanta* called *photons*; he later got the Nobel prize for it. The photoelectric effect is an emission of electrons from matter upon the absorption of a photon with wavelength smaller than a certain threshold; if the wavelength is too big, no electron is emitted. In 1913, Bohr explained the spectral lines of the hydrogen atom [Boh13], again using quantization, and in 1924 de Broglie stated that also particles can exhibit wave characteristics and vice versa [Bro24]; we call this *wave-particle duality.* These theories, though successful, were strictly phenomenological, without formal justification for quantization. Shortly afterward, mathematical formalisms arose that possessed quantization as an inherent feature.

Modern quantum physics In 1925, Heisenberg developed *matrix mechanics* and Schrödinger developed *wave mechanics*; these two approaches were later shown to be equivalent. A few years later, Dirac and von Neumann formulated the rigorous basis for quantum mechanics as the theory of linear operators on Hilbert spaces. Here physical observables correspond to eigenvalues of these operators and thus may be discrete.

Quantum physics satisfactorily explains both quantization and wave-particle duality. It also exhibits two new interesting phenomena, both later verified in the laboratory. The *uncertainty principle*, formulated by Heisenberg in 1927 [Hei27], is a phenomenon that two consecutive measurements on the same object (say position and velocity) possess a fundamental limitation on accuracy. *Entanglement* is a phenomenon that certain quantum states of two objects have to be described with reference to each other even if they are spatially separated. Quantum physics, however, does not contradict classical physics. The *correspondence principle*, formulated by Bohr in 1923, states that quantum systems reduce to classical in the limit of large quantum numbers.

Quantum mechanics is needed to explain certain macroscopic systems such as superconductors and superfluids. Despite initial reluctance on the part of many physicists, quantum physics turned out to be one of the most successful physical theories ever. The whole world nowadays is dependent on electronics and computers. They are based on semiconductors that could not be understood let alone built without quantum physics.

Quantum physics has several interpretations. Being built on the same basis, they do not differ in predictions, but some of them are more suitable to explain certain effects. The *Copenhagen interpretation*, largely due to Bohr in 1927, is the most widely accepted one among physicists. According to it, quantum mechanics provides probabilistic results because the physical universe is itself probabilistic rather than deterministic. The *many-worlds interpretation*, formulated by Everett in 1956, instead postulates that all the possibilities described by quantum theory simultaneously occur in a *multiverse* composed of independent parallel universes. In this thesis, we will stick to the Copenhagen interpretation, because it nicely describes the process of quantum measurement.

Double-slit experiment To illustrate quantum mechanics, consider the following simple experiment, first performed by Young in 1801. It is outlined in Figure 1.1. We send a beam of coherent light through a double-slit to a light sensitive film. When the film is developed, we see a pattern of light and dark fringes, called the *interference pattern*. This pattern is well explained by the *wave theory* of light formulated by Huygens, as follows. When a wave goes through the double-slit, two waves similar to the waves on the water level are formed. These two waves interfere: in the light spots constructively and in the dark spots destructively. This explains why the film is lit more on some places.

There is another theory of light, the *corpuscular theory*, formulated by Newton and supported by Einstein that claims that light is quantized—there is a smallest

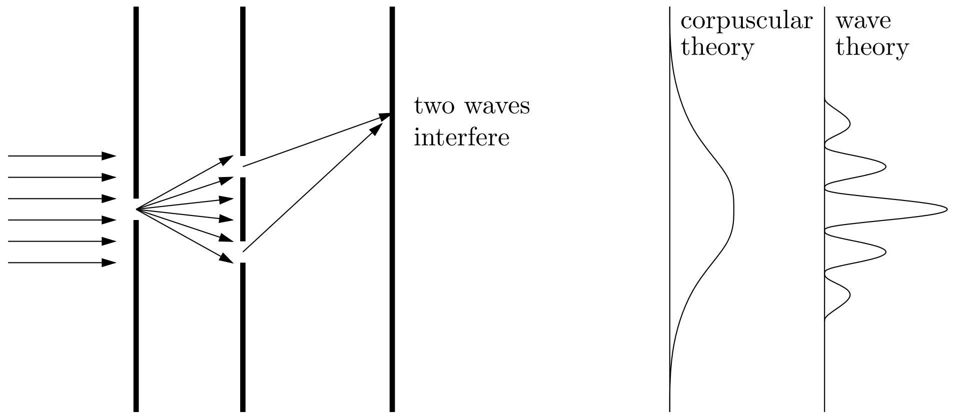


Figure 1.1: Double-slit experiment

amount of light, a photon, that one can produce. This photon behaves like a particle, a little ball that has its own trajectory. Consider thus the following thought experiment. We attenuate the light source such that at every moment, at most one photon is produced. Assuming the corpuscular theory, this photon goes along one concrete trajectory chosen at random. The final position should then be distributed according to a bell-like distribution with one maximum in the middle. Let us send many single photons one after another so that the film is exposed enough, and develop the film. One would expect the picture to contain the bell-like pattern. However, it turns out that the interference pattern arises instead. It seems as if each of the individual photons behaves like a wave and interferes with itself.

Since it is very difficult to produce single photons, this experiment has been so far just a thought experiment. However, one can perform a similar experiment with other particles. In 1961, Jönsson performed it with electrons, and in 1974 Merli et al. performed it in the setting “one electron at a time”, both observing the interference pattern described above. Interestingly, if one puts a detector behind the slits that measures which trajectory the electron has chosen, the interference pattern disappears. None of this could be explained with classical physics!

Postulates of quantum physics Quantum physics postulates that a photon (and also any other particle) is a kind of wave. It does not possess a concrete position, but instead a so-called *wave function* assigns amplitudes to all possible positions. An amplitude somehow corresponds to the probability of measuring the photon at a given position. The wave function evolves in time according to *Schrödinger's equation*. This equation is linear, as is its solution, and hence a *principle of superposition* holds—any linear combination of solutions to the equation is also a solution. When a system is in a linear combination, called *superposition*, of two states, then any future state is the same linear combination of the two individual evolved states. An observer, however, does not have access to the wave function and can only *observe* the system, which usually causes a disturbance. There are systems, for which quantum physics cannot, even in principle, predict the outcome of a measurement with certainty. This profound result was unacceptable for many physicists, including Einstein who initially helped to develop the theory.

Quantum computation Let us move from continuous quantum systems (such as position or velocity of a particle) to discrete ones, for example electron *spin* (a non-classical feature of electron corresponding to presence of angular momentum) or polarization of light (which can be horizontal or vertical, or a superposition of both). Consider a system of n such particles. Label the classical states of each particle by 0 and 1. There are 2^n possible classical states of the system and each has assigned a quantum amplitude. Since simulating this quantum system involves computation of exponentially many amplitudes, classical computers seem to be of little help and the best way to understand a quantum system is building it and experimenting with it. It was Feynman who first proposed [Fey82, Fey85] that since computers based on quantum physics can easily simulate quantum systems, they may be more powerful than classical computers. Deutsch then laid the foundations of quantum computing [Deu85] by defining a universal quantum Turing machine.

The real interest in quantum computing started in 1994, when Shor discovered a polynomial-time quantum algorithm for factoring [Sho97], a problem for which no fast classical algorithm is known (it may exist though). Since several widely used cryptosystems rely on the intractability of this problem, even a slight possibility of building a quantum computer is a threat. Two years later, Grover discovered a universal quantum search algorithm [Gro96] that is quadratically faster than the best classical algorithm. Thanks to its universality, quantum computers can solve many problems that include some form of search faster. Since that time, much progress has been made in various areas, including but not limited to quantum cryptography, quantum error correction, and quantum communication. Several books on these topics have been written; let us just mention the excellent textbook [NC00] by Nielsen and Chuang that covers the foundations in much more detail than presented in this chapter.

Structure of the thesis In this thesis, we explore fast quantum algorithms and also lower bounds, that is proofs of optimality of the algorithms. In the rest of Chapter 1, we introduce the basic principles of quantum computing and present several quantum algorithms, including the analysis of their running time. In Chapter 2, we introduce two basic lower-bound techniques: the quantum adversary method and the polynomial method. In Chapter 3, we present new faster quantum algorithms for matching and network flows, based on quantum search.

In Chapter 4, we present a new faster quantum algorithm for matrix verification, based on quantum random walks. In Chapter 5, we present new results unifying and extending the quantum adversary method. In Chapter 6 and Chapter 7, we prove several direct product theorems, and we apply them in Chapter 8, where we prove several new and tight time-space tradeoffs for quantum computation.

1.1 Postulates of quantum mechanics

1.1.1 State space

Information Classical information is measured in bits. A *bit* is a unit of information that represents a *bi*nary outcome $b \in \{0,1\}$. If we want to store more information, we can split it into several bits. For example, an outcome with 256 possible values can be stored in 8 bits, which is often called a *byte*, because there are $2^8 = 256$ possible combinations of the values of 8 bits. Bits are not the only way of measuring information. We could instead decide that the basic unit is a *trit* that represents a *ter*nary outcome $t \in \{0,1,2\}$. However, the theory would stay the same, hence we rather stick to the (simpler) bit.

One qubit A quantum variant of a bit is called a *quantum bit*, or simply *qubit*. If a qubit represents one of the two classical outcomes $b \in \{0,1\}$, we say that it is in a *computational basis state* and denote it by $|b\rangle$. The term $|b\rangle$ is a column vector with one 1 at the b^{th} position and zeroes everywhere else, that is

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \text{ and } \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \quad .$$

As we have already said, a quantum computer may not only be in a computational basis state, but also in a linear combination of those; we call it a *superposition* state. A general state of a qubit φ can thus be expressed as

$$|\varphi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix}\alpha_0\\\alpha_1\end{pmatrix} \quad ,$$

where $\alpha_0, \alpha_1 \in \mathbb{C}$ are so-called *quantum amplitudes*. The amplitude α_x is related to the probability of the qubit being in the state x. A vector like $|\varphi\rangle$ is called a *ket*-vector.

Scalar product Let us define a scalar product on states so that we can define a norm and measure distances between states. For a ket-vector $|\varphi\rangle$, let $\langle\varphi| = |\varphi\rangle^\dagger = \overline{|\varphi\rangle}^T$ denote the complex conjugate of the transpose of $|\varphi\rangle$. We call it a *bra*-vector and it is used as a linear operator. The *scalar product* of $|\varphi\rangle, |\psi\rangle$ is the usual matrix product $\langle\varphi|\cdot|\psi\rangle$, often shortened as $\langle\varphi|\psi\rangle$, which looks like a bracket and that is where the names *bra-* and *ket-* come from. The norm of a quantum state $|\varphi\rangle$ is defined as $\|\varphi\| = \sqrt{\langle\varphi|\varphi\rangle}$. Quantum mechanics requires that physical quantum states are *normalized*, hence the amplitudes of a qubit satisfy

$$1 = \|\varphi\|^2 = \langle\varphi|\varphi\rangle = (\overline{\alpha_0}\langle 0| + \overline{\alpha_1}\langle 1|)(\alpha_0|0\rangle + \alpha_1|1\rangle) = \overline{\alpha_0}\alpha_0 + \overline{\alpha_1}\alpha_1 = |\alpha_0|^2 + |\alpha_1|^2 \ .$$

It will often be convenient to work with unnormalized states. Quantum bits are elements of the two-dimensional complex Hilbert space $\mathcal{H} = \mathbb{C}^2$. In a finite dimension, a *Hilbert space* is a vector space with a scalar product. The computational basis $\{|0\rangle, |1\rangle\}$ is an *orthonormal basis* for $\mathcal{H}$, that is every basis vector is normalized and the scalar product between different basis vectors is 0.

Quantum phase The amplitude α_x of a basis state $|x\rangle$ can be decomposed as $\alpha_x = |\alpha_x| \cdot e^{i\theta}$ for some real θ, that is as a product of the absolute value of the amplitude and a complex unit; the latter is called a *quantum phase.* The quantum phase is something new that classical randomized computers do not have. We say that two states $|\varphi\rangle, |\psi\rangle$ differ by a *global phase* $e^{i\theta}$, if $|\varphi\rangle = e^{i\theta}|\psi\rangle$. If the quantum states are parts of a larger quantum system, for example $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ versus $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\theta}|1\rangle)$, we call the quantum phase a *relative phase.*

There are two important one-qubit states that have their own names: the *plus state* and the *minus state.* The plus state is a *uniform superposition* of both basis states, that is a superposition with equal amplitudes of all states, and the minus state differs by the relative phase of $|1\rangle$ as follows.

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \text{ and } \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \ .$$

The *Hadamard basis* $\{|+\rangle, |-\rangle\}$ is an orthonormal basis of $\mathcal{H}$ different from the computational basis. This means that the quantum phase is as important a quantity as the absolute value of the amplitude.

Many qubits A classical computer with n-bit memory is at every moment in exactly one of 2^n possible states. We can index them by n-bit strings $x \in \{0,1\}^n$ or by integers $0, 1, \ldots, 2^n - 1$. Again, a quantum computer can be in a superposition of these states. We describe it formally as follows.

Let a_1, a_2 be two vectors of dimensions n_1, n_2. The *tensor product* $a = a_1 \otimes a_2$ is the vector of dimension $n = n_1 n_2$ such that

$$a[i_1 n_2 + i_2] = a_1[i_1] a_2[i_2] \ . \tag{1.1}$$

If the vectors are quantum states, we often write $|x\rangle|y\rangle$, $|x, y\rangle$, or even $|xy\rangle$ instead of the longer $|x\rangle \otimes |y\rangle$. For example, if $x = x_1 \ldots x_n$ is an n-bit string, then $|x\rangle = |x_1\rangle \otimes \cdots \otimes |x_n\rangle$ is a column vector with one 1 at the position whose binary representation is x and zeroes everywhere else.

Let $\mathcal{S}_1, \mathcal{S}_2$ be two Hilbert spaces spanned by vectors $\{|s_{1,i}\rangle\}_i$ and $\{|s_{2,j}\rangle\}_j$. The *direct sum* $\mathcal{S} = \mathcal{S}_1 \oplus \mathcal{S}_2$ is the Hilbert space spanned by $\{|s_{1,i}\rangle\}_i \cup \{|s_{2,j}\rangle\}_j$. For example $\mathcal{H} = \mathbb{C} \oplus \mathbb{C}$. The *tensor product* $\mathcal{S} = \mathcal{S}_1 \otimes \mathcal{S}_2$ is the Hilbert space spanned by $\{|s_{1,i}\rangle \otimes |s_{2,j}\rangle\}_{i,j}$. For example, $\mathcal{H} \otimes \mathcal{H}$ is spanned by $\{|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle\}$, which we shorten as $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Let $\mathcal{S}^{\otimes 1} = \mathcal{S}$ and $\mathcal{S}^{\otimes(n+1)} = \mathcal{S} \otimes \mathcal{S}^{\otimes n}$ denote a *tensor power* of $\mathcal{S}$. A state of an n-qubit quantum computer is an element of $\mathcal{H}^{\otimes n}$, which is spanned by all n-bit strings $\{|x\rangle : x \in \{0,1\}^n\}$.

Entanglement If a state of a composite quantum system $|\varphi\rangle \in \mathcal{S}_1 \otimes \mathcal{S}_2$ can be written as $|\varphi\rangle = |\varphi_1\rangle|\varphi_2\rangle$ with $|\varphi_1\rangle \in \mathcal{S}_1$ and $|\varphi_2\rangle \in \mathcal{S}_2$, we call it a *product state*, otherwise we call it an *entangled state*. An entangled state is similar to a probability distribution of two random variables that are not independent.

All computational basis states are by definition product states, but there are states which cannot be written as a direct product of two states. Perhaps the simplest example is the following quantum state. It is called a *singlet* or an *EPR-pair* after the inventors Einstein, Podolsky, and Rosen who first showed [EPR35] its peculiar properties. It can be expressed as

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad . \tag{1.2}$$

We will say more about this state in the section about quantum measurement, in Example 1.1.3. The EPR-pair is at the heart of a physical experiment designed by Bell [Bel64] that provably cannot be explained by classical physics, and that thus gives strong indications in favor of the validity of quantum physics. Most quantum states are entangled. This follows from a counting argument: the Hilbert space on n qubits has dimension 2^n whereas product states can be described using just $2n$ complex parameters.

1.1.1. Remark. In this thesis, we mostly use numbers, roman letters, and English names for computational basis states ($|2\rangle, |i\rangle, |\text{good}\rangle$), and Greek letters and punctuation for superposition quantum states ($|\varphi\rangle, |+\rangle$), unless stated otherwise.

1.1.2 Evolution

Linear evolution Physical operations are linear on quantum states. If the original state is denoted by $|\varphi\rangle$ and the state after applying the operation U is denoted by $\mathsf{U}|\varphi\rangle$, then

$$\mathsf{U}(\alpha|\varphi\rangle + \beta|\psi\rangle) = \alpha\mathsf{U}|\varphi\rangle + \beta\mathsf{U}|\psi\rangle \quad .$$

Therefore it is sufficient to specify the effect of U on all basis states from some fixed basis, and the effect on all superposition states follows from linearity. If $\{|x\rangle\}_{x=1}^N$ is an orthonormal basis for the Hilbert space and $\mathsf{U} : |x\rangle \to \sum_{y=1}^N u_{y,x}|y\rangle$

for some complex coefficients $u_{y,x}$, then U is represented in this basis by the $N \times N$ matrix

$$\mathsf{U} = \begin{pmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,N} \\ u_{2,1} & u_{2,2} & \dots & u_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N,1} & u_{N,2} & \dots & u_{N,N} \end{pmatrix} = \sum_{x,y} u_{y,x} |y\rangle\langle x| \ .$$

This matrix representation illuminates why we have chosen the syntax $\mathsf{U}|\varphi\rangle$ for the state of the quantum system after applying the operation—the state is determined by the matrix-vector product of U and $|\varphi\rangle$.

Unitary operations Physical quantum states are normalized. It is therefore reasonable to require that a quantum operation does not violate this constraint, but it preserves the norm of the quantum state.

$$\|\mathsf{U}|\varphi\rangle\| = \|\varphi\|$$

A norm-preserving linear operator is called a *unitary operator*. Elementary linear algebra implies that every unitary operator U satisfies $\mathsf{U}^\dagger \mathsf{U} = \mathsf{U}\mathsf{U}^\dagger = \mathsf{I}$, where the *adjoint operator* $\mathsf{U}^\dagger = \overline{\mathsf{U}^T}$ is the complex conjugate of the transpose of U and I is the identity. It follows that unitary operators

- are *reversible* (that is they have an inverse) and $\mathsf{U}^{-1} = \mathsf{U}^\dagger$,
- are diagonalizable and all their eigenvalues are complex units,
- preserve scalar product, because $\langle\varphi|\mathsf{U}^\dagger \cdot \mathsf{U}|\psi\rangle = \langle\varphi|\psi\rangle$,
- map orthonormal bases to orthonormal bases.

Since unitary operators are reversible, quantum computers cannot perform any irreversible operations, such as storing a number into memory or removing an unused register. The most common ways to load a number are to add it modulo some fixed number or to *xor* it bitwise, that is add it bitwise modulo 2.

Elementary quantum gates Let us present a few simple one-qubit gates. We represent them in the computational basis. The simplest ones are the *Pauli operators* $\mathsf{I}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}$.

$$\mathsf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathsf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \mathsf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \mathsf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \ . \tag{1.3}$$

I is the identity operator that does nothing, X is the *bit flip*, and Z is the *phase flip*. The Pauli operators form a group if we ignore the global phase. Other useful one-qubit operators are

$$\mathsf{R}_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}, \qquad \mathsf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \ . \tag{1.4}$$

R_θ is the *phase shift by angle* θ and H is the *Hadamard gate.* The Hamadard gate is very remarkable. It is *self-inverse*, that is $\mathsf{H}^2 = \mathsf{I}$, and it maps the computational basis $\{|0\rangle, |1\rangle\}$ to the Hadamard basis $\{|+\rangle, |-\rangle\}$ and vice-versa. Very roughly, one can think of H as a reversible *coin flip operator*, because it replaces a computational basis state by a superposition of both states, on the other hand one can still recover the original state by another application of H. As we will see, it is very often used in quantum algorithms.

Two important two-qubit operators are the controlled-NOT operator, which maps $|x\rangle|y\rangle \to |x\rangle|x \oplus y\rangle$, where $x \oplus y = (x + y) \bmod 2$, and the controlled-Z operator, which maps $|x\rangle|y\rangle \to (-1)^{xy}|x\rangle|y\rangle$. In matrix form, they are respectively

$$\text{controlled-NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \text{controlled-Z} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} .$$

An important three-qubit operator is the Toffoli gate, which maps $|x, y\rangle|z\rangle \to |x, y\rangle|z \oplus xy\rangle$, that is it computes the logical AND of the first two bits.

Many qubits The quantum gates presented above act on 1–3 qubits. If we apply a quantum gate on a subset of bits in a many-qubit quantum system, then the overall quantum operation can be expressed in terms of a *tensor product.* If $\mathsf{U}_1, \mathsf{U}_2$ are operators, then $\mathsf{U} = \mathsf{U}_1 \otimes \mathsf{U}_2$ maps $|\varphi\rangle \otimes |\psi\rangle \to \mathsf{U}_1|\varphi\rangle \otimes \mathsf{U}_2|\psi\rangle$. For example, if we have a 4-qubit system and apply the Hadamard transform on the second qubit and the controlled-NOT operator on the last two qubits, then the quantum operation is $\mathsf{I} \otimes \mathsf{H} \otimes \text{controlled-NOT}$. In matrix form, the tensor product of two matrices is analogous to the tensor product of two vectors defined by equation (1.1).

We can apply several gates in parallel on different qubits. For example, if we apply the Hadamard gate on each qubit in an n-qubit system, the operator maps

$$\begin{aligned} |x\rangle = |x_1 \dots x_n\rangle \xrightarrow{\mathsf{H}^{\otimes n}} (\mathsf{H}|x_1\rangle) \otimes \cdots \otimes (\mathsf{H}|x_n\rangle) &= \bigotimes_{i=1}^{n} \mathsf{H}|x_i\rangle \\ &= \frac{1}{2^{n/2}} \bigotimes_{i=1}^{n} \sum_{y_i \in \{0,1\}} (-1)^{y_i x_i} |y_i\rangle \\ &= \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} \bigotimes_{i=1}^{n} (-1)^{y_i x_i} |y_i\rangle \\ &= \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{\sum_{i=1}^n y_i x_i} |y\rangle = \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \bullet y} |y\rangle \ , \end{aligned} \tag{1.5}$$

where $x \bullet y$ is the *bitwise scalar product* of x and y.

1.1.3 Measurement

We have described what quantum states are like and what kind of operations one can apply on them. If all quantum systems were closed and did not interact with the environment, then the description would be complete. However, we often want to look at the quantum state at the end of the computation and read the outcome. By reading the outcome we perform a (unitary) interaction on the joint system computer–user, but this operation is not unitary on the subsystem containing only the computer.

One can describe the interaction in two ways: as a probabilistic measurement or using the density matrix formalism. We use both of them in this thesis, because some aspects of quantum computing are better understood in terms of a measurement and some in terms of a density matrix.

Projective measurement The simplest kind of measurement is a projective measurement. It can be described as follows. The Hilbert space of the quantum system is split into a direct sum of orthogonal subspaces $\mathcal{S} = \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_m$. Each subspace corresponds to one possible outcome of the measurement. Decompose the measured quantum state $|\varphi\rangle \in \mathcal{S}$ as

$$|\varphi\rangle = \sum_{i=1}^{m} \alpha_i |\varphi_i\rangle \enspace ,$$

where $\alpha_i \in \mathbb{C}$ and $|\varphi_i\rangle \in \mathcal{S}_i$ is a normalized quantum state. This decomposition is unique modulo quantum phases of the sub-states $|\varphi_i\rangle$ thanks to the orthogonality of the subspaces. If the quantum state $|\varphi\rangle$ is measured using this projective measurement, then the outcome is a random variable $I \in \{1, 2, \dots, m\}$ with

$$\Pr\left[I = i\right] = |\alpha_i|^2 \enspace .$$

This illuminates why we want quantum states to be normalized—the probabilities have to sum up to 1. After the measurement, if the outcome was i, then the quantum state collapses to $|\varphi_i\rangle$.

Measuring often disturbs the quantum state. Roughly speaking, the more information we get, the more the quantum state gets collapsed. The only way to not collapse an *arbitrary* quantum state is to have only one subspace $\mathcal{S} = \mathcal{S}_1$, that is to not measure at all. On the other hand, if we have a promise that the quantum state lies in a restricted set of possible quantum states, then it may be possible to configure the subspaces such that the collapsed state is equal to the original state. For example, if we know that the quantum state lies in some orthonormal basis, then we simply measure in this basis and obtain full information while not disturbing the state at all.

A convenient way of representing a projective measurement is in terms of projectors. A *projector* is a linear operator P such that $\mathsf{P}^2 = \mathsf{P}$. Projectors only have eigenvalues 1 and 0, and they thus have two eigenspaces: the subspace that stays untouched and the subspace that is mapped to the zero vector. We represent the i^{th} subspace $\mathcal{S}_i$ by the projector to this subspace, denoted by $\mathsf{P}_i = \Pi_{\mathcal{S}_i}$, and require that

$$\mathsf{P}_i^2 = \mathsf{P}_i, \text{ and} \qquad \sum_{i=1}^{m} \mathsf{P}_i = \mathsf{I} \ .$$

If a state $|\varphi\rangle$ is measured by the projective measurement $\{\mathsf{P}_i\}_{i=1}^m$, then the probability of obtaining the outcome i is $\|\mathsf{P}_i|\varphi\rangle\|^2 = \langle\varphi|\mathsf{P}_i|\varphi\rangle$ and the quantum state then collapses to $\mathsf{P}_i|\varphi\rangle / \|\mathsf{P}_i|\varphi\rangle\|$.

1.1.2. Example. The measurement of all qubits of an n-qubit system in the computational basis is described by the projective measurement $\{|i\rangle\langle i| : i \in \{0,1\}^n\}$. Measuring nothing at all is described by the projective measurement $\{\mathsf{I}_n\}$, where I_n is the identity operator on n qubits. The measurement of the 1st qubit of an n-qubit system in the Hadamard basis is described by the projective measurement $\{|+\rangle\langle+| \otimes \mathsf{I}_{n-1},\ |-\rangle\langle-| \otimes \mathsf{I}_{n-1}\}$.

1.1.3. Example. (EPR-pair) Let us measure the first qubit of the singlet $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ from equation (1.2) in the computational basis. The projectors are $\{|00\rangle\langle 00| + |01\rangle\langle 01|,\ |10\rangle\langle 10| + |11\rangle\langle 11|\}$. The quantum state hence collapses to $|00\rangle$ with probability $\frac{1}{2}$ and to $|11\rangle$ with probability $\frac{1}{2}$. The interesting thing is that both qubits collapse to the same value, but we cannot tell beforehand which one. If we measure any of these two qubits first, we get a completely random outcome and that outcome determines the value of the other qubit when measured next. The measurement on the first qubit thus influences the state of the second qubit that can be physically or even causally separated (the collapse of the wave function is immediate and there is no limit such as the speed of light). This was the reason why Einstein strongly argued [EPR35] for many decades against quantum physics. It, however, turns out that no information can be transferred by measuring entangled states, hence the general theory of relativity is not contradicted. Note that this behavior only occurs with entangled states. Elementary linear algebra implies that if a sub-state of a product state is measured in an arbitrary basis, the other sub-state is not influenced.

POVM measurement Projective measurements are not the most general measurements. A very popular and elegant generalization is the *POVM measurement*, named after Positive Operator Valued Measure. An operator E is called *self-adjoint* or *Hermitian*, if $\mathsf{E}^\dagger = \mathsf{E}$, and it is called *positive semidefinite*, denoted by $\mathsf{E} \succeq 0$, if $\langle v|\mathsf{E}|v\rangle \geq 0$ for every vector v. Positive semidefinite operators are Hermitian and all their eigenvalues are non-negative. A POVM measurement is described by a list of positive semidefinite operators $\{\mathsf{E}_i\}_{i=1}^m$ such that

$$\mathsf{E}_i \succeq 0, \text{ and} \qquad \sum_{i=1}^m \mathsf{E}_i = \mathsf{I} \ .$$

The difference between a projective and POVM measurement is that E_i may not be a projector, but it can have non-negative eigenvalues smaller than 1. The probability of obtaining the outcome i is $\langle\varphi|\mathsf{E}_i|\varphi\rangle = \mathrm{Tr}\,(\mathsf{E}_i \cdot |\varphi\rangle\langle\varphi|)$. If we care how the quantum state looks like after the measurement, we need to specify the measurement operators in more detail. In particular, we have to choose the *Cholesky decomposition* $\mathsf{E}_i = \mathsf{M}_i^\dagger \mathsf{M}_i$; it always exists. The state then collapses to $\mathsf{M}_i|\varphi\rangle / \|\mathsf{M}_i|\varphi\rangle\|$. The probabilities, however, do not depend on the decomposition.

The elegant mathematical formulation of POVM measurements in terms of semidefinite programs allows solving several measurement related problems by techniques of semidefinite optimization.

1.1.4 Density operator formalism

The formalism of quantum measurements is a bit inconvenient for some problems, because it uses the concept of randomness to determine the state of the quantum system after a measurement. It is often useful to have an object that completely describes the quantum system and manipulate it deterministically. Classically, for example, such an object may be a probability distribution. If the evolution of a system is not fully deterministic, but depends on the outcome of a random process, we simply track the *probabilities* of possible states instead of *concrete* states during time.

Density operator Let us show how to combine probability distributions with quantum states. Let an n-qubit quantum system be in one of the states $|\varphi_i\rangle$, with respective probabilities p_i. We call $\{p_i, |\varphi_i\rangle\}_{i=1}^m$ an *ensemble of pure states.* The *density operator* for the system is

$$\rho = \sum_{i=1}^m p_i |\varphi_i\rangle\langle\varphi_i| \ . \tag{1.6}$$

For example, if the system is in the computational basis state $|x\rangle$, then $\rho = |x\rangle\langle x|$ contains exactly one 1 on the main diagonal and it contains 0 everywhere else. If the system is in an arbitrary quantum state $|\varphi\rangle$ with certainty, then $\rho = |\varphi\rangle\langle\varphi|$ is a rank-1 Hermitian operator. Finally, if the system is in a probabilistic mixture of computational basis states described by the probability distribution p, then $\rho = \sum_{x=0}^{2^n-1} p_x |x\rangle\langle x|$ is a diagonal operator whose diagonal represents the probability distribution. If $\rho = |\varphi\rangle\langle\varphi|$ for some $|\varphi\rangle$, we call it a *pure state*, otherwise we call it a *mixed state.*

Evolution and measurement Suppose that a quantum system ρ undergoes a unitary evolution U. Since $|\varphi\rangle$ is mapped to $\mathsf{U}|\varphi\rangle$, the density operator is mapped as

$$\rho \xrightarrow{\mathsf{U}} \sum_{i=1}^{m} p_i \mathsf{U}|\varphi_i\rangle\langle\varphi_i|\mathsf{U}^\dagger = \mathsf{U}\rho\mathsf{U}^\dagger \ .$$

Suppose that we apply a POVM measurement (or a simpler projective measurement) specified by positive semidefinite operators $\mathsf{E}_j \succeq 0$ with Cholesky decomposition $\mathsf{E}_j = \mathsf{M}_j^\dagger \mathsf{M}_j$ such that $\sum_j \mathsf{E}_j = \mathsf{I}$. It is not hard to see that the probability of obtaining the outcome j is $\mathrm{Tr}\,(\mathsf{E}_j\rho)$. The density operator after observing j and *remembering the result* is

$$\rho_j = \frac{\mathsf{M}_j \rho \mathsf{M}_j^\dagger}{\mathrm{Tr}\,(\mathsf{E}_j \rho)} \ .$$

Imagine that a quantum system is prepared in the state ρ_j with probability q_j. The density operator is then $\rho = \sum_j q_j \rho_j$. It follows that if we measure the quantum system ρ and *forget the result*, then the density operator is

$$\rho \xrightarrow{\mathsf{M}} \sum_j \mathrm{Tr}\,(\mathsf{E}_j\rho)\rho_j = \sum_j \mathsf{M}_j \rho \mathsf{M}_j^\dagger \ .$$

Properties of the density operator It is immediate from equation (1.6) that ρ is a positive semidefinite operator with trace 1. The *trace* of an operator is the sum of its diagonal entries; it does not depend on the basis. The rules of quantum evolution and measurement imply that two quantum ensembles that lead to an equal density operator ρ are indistinguishable. For example, the uniform mixture of $|0\rangle, |1\rangle$ is the same as the uniform mixture of $|+\rangle, |-\rangle$, because $|0\rangle\langle 0| + |1\rangle\langle 1| = \mathsf{I} = |+\rangle\langle +| + |-\rangle\langle -|$. We cannot say, when we get an ensemble, whether someone flipped a coin and picked a state at random from the computational basis or from the Hadamard basis. Hence it is not important what states we use to define a quantum ensemble and the only thing that matters is the density operator.

Reduced density operator Perhaps the most important application of the density operator is the description of subsystems of a composite system. Let ρ_{AB} be the state of a composite system AB. Then the *reduced density operator for system A* is

$$\rho_A = \mathrm{Tr}_B\,(\rho_{AB}) \ ,$$

where Tr_B the *partial trace* over system B. The partial trace is the (unique) linear operator such that

$$\mathrm{Tr}_B\,(\rho_A \otimes \rho_B) = \rho_A \cdot \mathrm{Tr}\,(\rho_B)$$

for all operators ρ_A, ρ_B. The usual trace operator is a special case of the partial trace when we trace out the whole system $\mathrm{Tr} = \mathrm{Tr}_{AB}$.

One may wonder why the reduced density operator ρ_A should be defined this way. The answer is that ρ_A is the only density operator that gives correct probabilities for all measurements on the subsystem A alone, meaning the same probabilities as if the measurement was done on the joint system AB.

1.1.4. Example. Let us calculate the reduced density operator for the 1st qubit of the singlet (1.2). The density matrix of the joint system is $\rho_{AB} = \frac{1}{2}(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|)$. If we trace out the second qubit, the reduced density operator is $\rho_A = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) = \frac{1}{2} \cdot \mathsf{I}$, that is the *completely mixed state*.

Purification *Purification* is a mathematical procedure that allows us to associate pure states with mixed states. Given a density operator ρ_A, we introduce a new fictitious quantum system R, called the *reference system*, such that the quantum state of the joint system AR is a pure state and the reduced density matrix for system A is exactly ρ_A. This trick allows us to handle mixed states using the same techniques as pure states.

The purification can be done as follows. Suppose the density operator ρ_A has rank ℓ. We find the eigenvectors of ρ_A, denote them by $|\varphi_i\rangle$, and write $\rho_A = \sum_{i=1}^{\ell} p_i |\varphi_i\rangle\langle\varphi_i|$. Let R be an ℓ-dimensional Hilbert space with computational basis $|i\rangle$. It is not hard to verify that

$$\rho_A = \mathrm{Tr}_R\left(|\varphi\rangle\langle\varphi|\right), \text{ where } \qquad |\varphi\rangle = \sum_{i=1}^{\ell} \sqrt{p_i}|\varphi_i\rangle_A |i\rangle_R \ .$$

1.2 Models of computation

The laws of quantum physics allow us to perform any unitary operation or POVM measurement *in principle*. This does not mean that all operations can be done *efficiently*, that is using a small number of elementary quantum gates. By a simple counting argument, there are many more operations than short sequences of elementary quantum gates, hence to compute or even approximate some operations one needs (exponentially) many elementary quantum gates.

In this section we present models for classical and quantum computation. There are two basic models: the Turing machine and the circuit model. The circuit model is much more convenient for quantum algorithms, hence we will stick to it.

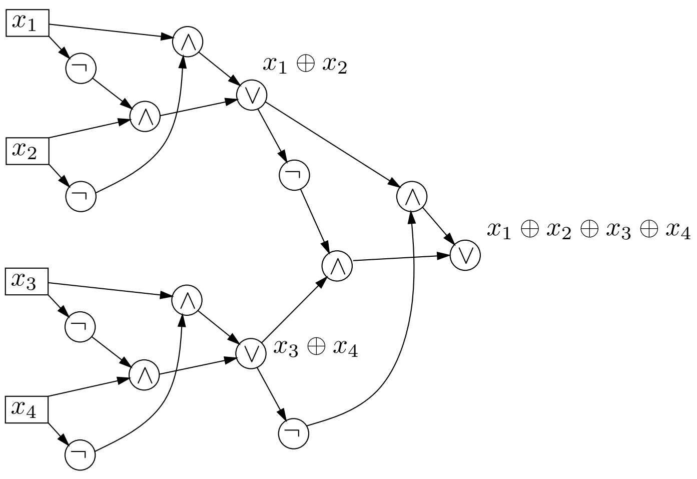


Figure 1.2: Classical circuit computing the parity of 4 bits

1.2.1 Quantum circuits

Classical circuits A circuit is represented by an acyclic oriented graph. The leaves are labeled by input bits and the other vertices are labeled by gates from the set $\{\neg, \wedge, \vee\}$. A computation is naturally induced by the graph as follows. We sort the vertices into a *topological order*, that is every vertex is placed after all its ancestors; if the graph is acyclic, then such an order always exists. Then the values of all vertices are computed in this sorted order. A vertex labeled by an input bit gets the value of the input bit, and a vertex labeled by a gate gets the value of the logical operation on the values of its ancestors. The outcome of the computation is the list of values of all vertices without descendants. A classical circuit computing the parity of 4 bits is outlined in Figure 1.2.

If all gates have at most d inputs, we say that the circuit has *fan-in* d. The *fan-out* is the maximal number of outgoing edges. Two important complexity measures are associated to a circuit: the *size* is the number of gates, and the *depth* is the length of the longest path. The above set of gates is *universal*, that means any function $f : \{0,1\}^n \to \{0,1\}^m$ is computed by a finite circuit composed of these gates.

Reversible circuits *Reversible computation* is a slightly restricted model of computation, where each individual computational step must be reversible, that is the state of the computation before any step can be computed from the state after applying the step. It is clear that reversible computers can only compute reversible functions.

We model reversible computation by reversible circuits as follows. When the input size is n, the memory is of fixed size $m \geq n$ and it is initialized to the state $x_1 \dots x_n 0 \dots 0$. One computational step consists of applying a gate from the set $\{\neg, \mathsf{controlled\text{-}NOT}, \mathsf{Toffoli}\}$ on a subset of bits. Every bit is used exactly once, the fan-in and fan-out of each gate are equal, and each elementary gate is reversible. The controlled-NOT gate flips the target bit if the value of the source bit is 1. The Toffoli gate flips the target bit if the logical AND of the two source bits is 1. We use reversible circuits with constant fan-in, and the logical AND of n bits can thus be computed using $O(\log n)$ operations.

The set of gates above is universal, that means any permutation $f : \{0,1\}^n \to \{0,1\}^n$ is computed by a finite circuit composed of these gates. The restriction of reversibility is not as big a handicap as one might think, because there are several known classical tricks that turn any computation into a reversible one at little cost—at most quadratic slowdown and twice bigger memory consumption. One of them is called *uncomputation* and it works as follows. If A tries to overwrite a register, we use a fresh ancilla register instead. The computation thus leaves a lot of garbage behind. We clean it by running the computation of A backwards, which is possible since no data was overwritten.

$$|x\rangle|0\rangle|0\rangle \xrightarrow{A} |x\rangle|\text{garbage}\rangle|0\rangle \xrightarrow{\text{copy}} |x\rangle|\text{garbage}\rangle|\text{outcome}\rangle \xrightarrow{A^{-1}} |x\rangle|0\rangle|\text{outcome}\rangle \ .$$

Since we copied the outcome into an ancilla register, we end up with the input, the outcome, and an array of fresh zero bits. Any classical computation of a function $f_1 : \{0,1\}^n \to \{0,1\}^m$ can thus be turned into a reversible computation of the permutation $f_2 : \{0,1\}^{n+m} \to \{0,1\}^{n+m}$ such that $f_2 : |x\rangle|y\rangle \to |x\rangle|y \oplus f_1(x)\rangle$. For details and tradeoffs, see [Ben89].

Quantum circuits Quantum computation can be modelled by quantum circuits that are a straightforward extension of reversible circuits. We simply extend classical reversible gates $\{\neg, \mathsf{controlled\text{-}NOT}, \mathsf{Toffoli}\}$ by linearity to unitary operators and add a few purely quantum gates, in particular the Hadamard gate H and all phase shifts R_θ. It follows that every reversible circuit is automatically a valid quantum circuit.

An example quantum circuit which is very different from any classical circuit is outlined in Figure 1.3. Classically, computing the parity of n bits is hard even if we allow $\{\wedge, \vee\}$-gates with unbounded fan-in and fan-out, that is one needs a circuit of logarithmic depth [Raz87, Smo87]. In the quantum case, unbounded fan-out, that is the mapping

$$(\alpha|0\rangle + \beta|1\rangle)|0\dots0\rangle \to \alpha|0\dots0\rangle + \beta|1\dots1\rangle$$

implies unbounded parity in constant depth. The trick behind the circuit is that when the controlled-NOT gate is applied in the Hadamard basis, the control qubit and the target qubit get swapped.

Adding H, $\{\mathsf{R}_\theta\}_{\theta\in[0,2\pi]}$ is sufficient to get the full power of quantum computing. The Pauli operators do not have to be added, because X is the gate $\neg$, $\mathsf{Z} = \mathsf{R}_\pi$,

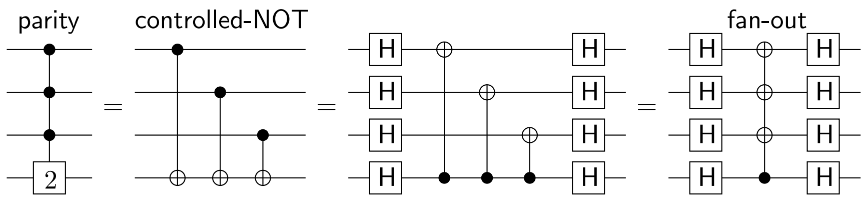


Figure 1.3: Quantum circuit computing the parity of n bits

and Y can be expressed using X and R_θ. The controlled-Z can be expressed using the controlled-NOT and the Hadamard gate. The set of gates above is universal, that is every unitary operator on n qubits is computed by a finite quantum circuit composed of these gates. In fact, there exist smaller sets of gates that are universal. For example, unlike the classical case, the Toffoli gate is not necessary and it can be computed using only one-qubit gates and the controlled-NOT [BBC+95]. There even exist *finite* universal sets of gates where we only allow phase shifts R_θ with angle θ taken from a finite set, such that the gates can *approximate* efficiently any unitary operator within any precision [ADH97].

1.2.2 Quantum query complexity

Oracle model Many quantum algorithms are developed for the so-called *oracle model*, also called the *black-box* model, in which the input is given as an *oracle* so that the only knowledge we can gain about the input is in asking queries to the oracle. The input is a finite string $x \in \Sigma^n$ of some length n, where $x = x_1 x_2 \dots x_n$ and Σ is the *input alphabet.* The goal is to compute some function $f : \Sigma^n \to \Sigma'$ of the input x, where Σ' is the *output alphabet.* We associate letters from the input alphabet with numbers $\{0, 1, \dots, |\Sigma| - 1\}$. Some of the functions we consider are *Boolean*, that is $\Sigma = \Sigma' = \{0, 1\}$, some not. We use the shorthand notation $[n] = \{1, 2, \dots, n\}$.

As our measure of complexity in this model, we use query complexity. The query complexity of an algorithm A computing a function f is the number of queries used by A. The *query complexity* of f is the minimum query complexity among all algorithms computing f.

An alternative measure of complexity would be to use the *time complexity* which counts the number of basic operations used by an algorithm, that is the number of queries plus the number of (bounded fan-in) elementary gates. The time complexity is always at least as large as the query complexity, and thus a lower bound on the query complexity is also a lower bound on the time complexity. For most existing quantum algorithms, including Grover's search [Gro96], the time complexity is within polylogarithmic factors of the query complexity. A notorious exception is the so-called Hidden Subgroup Problem which has polynomial query complexity [EHK04], yet polynomial *time* algorithms are known only for some instances of the problem.

One query The oracle model in the classical setting is called the model of *decision trees* [BBC$^+$01]. A classical query consists of an index $i \in [n]$, and the answer of the number x_i. There is a natural way of modelling a query so that it is reversible. The input is a pair (i, b), where $i \in [n]$ is an index and $b \in \Sigma$ a number. The output is the pair $(i, (b + x_i) \bmod |\Sigma|)$. There are (at least) two natural ways of generalizing a query to the quantum setting, in which we require all operations to be unitary. The first way is to consider a quantum query as a unitary operator that takes two inputs $|i\rangle|b\rangle$, where $i \in [n]$ and $b \in \Sigma$, and outputs $|i\rangle|(b + x_i) \bmod |\Sigma|\rangle$. The oracle is then simply just a linear extension of the reversible query given above. We allow the oracle to take some arbitrary ancilla state $|z\rangle$ as part of the input that is untouched by the query.

$$\mathsf{O}'_x|i, b, z\rangle = |i, (b + x_i) \bmod |\Sigma|, z\rangle \tag{1.7}$$

The ancilla $|z\rangle$ contains any additional information currently part of the quantum state that is not involved in the query.

The second way is to consider a quantum query as a unitary operator O_x that takes two inputs $|j\rangle$ and $|b\rangle$, and outputs $e^{\frac{2\pi i}{|\Sigma|} x_j b}|j\rangle|b\rangle$, where $j \in [n]$ and $b \in \Sigma$. Here we use the letter j to denote the index, because $i = \sqrt{-1}$ denotes the complex unit. Again, we allow the query to take some arbitrary ancilla state $|z\rangle$ that is untouched by the query. We say that the oracle is *computed in the phases* by O_x. Both operators O'_x and O_x and unitary. The two operators are equivalent in that one query to either oracle can be simulated by a superposition query to the other oracle preceded and followed by a basis change (in particular by the Fourier transform). Hence the choice of O_x or O'_x only influences the query complexity and the time complexity by a multiplicative constant. Though the first way is possibly the more intuitive, we often adopt the second way as it is very convenient.

$$\mathsf{O}_x|j, b, z\rangle = e^{\frac{2\pi i}{|\Sigma|} x_j b}|j, b, z\rangle$$

We mostly work with binary input alphabet $\Sigma = \{0, 1\}$ and then the oracle O_x can be expressed in the following simpler form.

$$\mathsf{O}_x|i, b, z\rangle = \begin{cases} |i, b, z\rangle & b = 0, \text{ also called } \textit{non-query} \\ (-1)^{x_i}|i, b, z\rangle & b = 1, \text{ also called } \textit{query} \end{cases} \tag{1.8}$$

We may think of one query as a one-round exchange of information between two parties, the algorithm and the oracle. In the classical binary setting, the algorithm sends an index $i \in [n]$ to the oracle, and the oracle responds with one bit of information, namely x_i. In the quantum setting, the algorithm sends the $\log_2(n)$ qubits $|i\rangle$ to the oracle O_x, and the oracle responds with $(-1)^{x_i}|i\rangle$. The algorithm and oracle thus exchange a total number of $2\log_2(n)$ qubits, and thus, a quantum query to O_x can convey up to $2\log_2(n)$ classical bits of information about the oracle by Holevo's theorem [Hol73, CDNT98]. The *super-dense coding* [BW92] achieves this upper bound.

Information theoretically, a function $f : \{0,1\}^n \to \{0,1\}^{\log_2(n)}$ that outputs at most $\log_2(n)$ bits, can potentially be solved by a constant number of queries to the oracle. An example of such a problem is the Deutsch-Jozsa problem [DJ92], which is to distinguish balanced Boolean functions from constant functions. (A function f is constant if $f(x) = f(y)$ for all inputs x, y, and it is balanced if it is not constant and $|f^{-1}(f(x))| = |f^{-1}(f(y))|$ for all inputs x, y.) The deterministic complexity of the Deutsch-Jozsa problem is $\lfloor \frac{n}{2} \rfloor + 1$ and its randomized complexity is constant.

Quantum algorithms A *quantum algorithm* in the oracle model starts in a state that is independent of the oracle. For convenience, we choose the state $|\vec{0}\rangle$ in which all qubits are initialized to 0. It then evolves by applying arbitrary unitary operators U to the system, alternated with queries O_x to the oracle x, followed by a measurement of the final state, the outcome of which is the result of the computation. In symbols, a quantum algorithm A that uses t queries computes the final state

$$|\psi_x^t\rangle = \mathsf{U}_t \mathsf{O}_x \mathsf{U}_{t-1} \cdots \mathsf{U}_1 \mathsf{O}_x \mathsf{U}_0 |\vec{0}\rangle \ ,$$

which is then measured. If the algorithm computes some function $f : \Sigma^n \to \Sigma'$, we measure the $\log|\Sigma'|$ leftmost bits of the final state $|\psi_x^t\rangle$, producing some outcome r.

Without loss of generality, we assume that quantum algorithms only perform one final measurement at the end of the computation. If a measurement is needed in the middle, it can be *postponed* and the following quantum operations that depend on the outcome of that measurement are replaced by *controlled* quantum operations, like the $\mathsf{controlled\text{-}NOT}$.

Query complexity The success probability p_x of A on input $x \in \Sigma^n$ is the probability that $r = f(x)$. For *total* functions $f : \Sigma^n \to \Sigma'$, we define the *success probability* of A as the minimum of p_x over all $x \in \Sigma^n$. For *partial* functions $f : S \to \Sigma'$, where $S \subseteq \Sigma^n$, we take the minimum over S only. A quantum algorithm A has *error* at most ϵ if the success probability of A is at least $1 - \epsilon$. We say that an algorithm A for a Boolean function has *1-sided error* if its success probability is 1 on all zero-inputs (or on all one-inputs), otherwise we call the error *2-sided.*

1.2.1. Definition. For an $\varepsilon \geq 0$, let $Q_\epsilon(f)$ denote the minimum query complexity of any quantum algorithm that computes f with 2-sided error at most ϵ, and as is common, let $Q_2(f) = Q_{\frac{1}{3}}(f)$ denote the *2-sided bounded error quantum query complexity* with $\epsilon = \frac{1}{3}$. Let $Q_E(f) = Q_0(f)$ denote the *exact quantum query complexity* of f.

Sometimes we run a classical algorithm (possibly using quantum subroutines) whose query complexity depends on random choices made by the algorithm. In such a case, it makes sense to talk about the *expected query complexity* on an input, which is the average query complexity over all random choices. Note that we do not define this concept for quantum algorithms, where we only use one measurement at the end of the computation and thus have no information whatsoever about the progress of the computation.

1.3 Quantum search

One of the two most important quantum algorithms is Grover search [Gro96] invented by Lov Grover in 1996. The algorithm can be used for searching any unstructured database and it is quadratically faster than the best classical algorithm. Several improved variants of the basic algorithm are known and we describe the most important ones here [BBHT98, BHMT02].

1.3.1 Searching ones in a bit string

For simplicity, let us restrict our attention to the following simple problem.

1.3.1. Problem. (Unordered search) *Decide whether a given n-bit input string x contains a one, and if it does, find the position of some.*

Assume for simplicity that n is a power of 2; this allows us to implement the algorithm in a very simple way. The famous quantum algorithm Grover search that optimally solves this problem is outlined in Figure 1.4.

1.3.2. Theorem ([Gro96, BBHT98]). *Let $w = |x|$ be the number of ones in the input string. If the number of iterations can be written as $t = (1+2k+\varepsilon)\frac{\pi}{4}\sqrt{\frac{n}{w}}$ for some integer $k \geq 0$ and real $|\varepsilon| \leq \frac{1}{2}$, then* Grover search (n, t) *finds a one with probability at least $\frac{1}{2}$.*

Proof. The algorithm starts in a uniform superposition over all input bits. We show that after performing the Grover iteration t times, the quantum state will be close to a uniform superposition over all ones, and hence a measurement will yield an index of some one with high probability. The analysis is based on the fact that in the Grover iteration, the n-dimensional quantum state actually lies in a two-dimensional real subspace $\mathcal{S}$ spanned by the uniform superposition over all zeroes $|\psi_0\rangle = \frac{1}{\sqrt{n-w}} \sum_{i:x_i=0} |i\rangle$ and the uniform superposition over all ones $|\psi_1\rangle = \frac{1}{\sqrt{w}} \sum_{i:x_i=1} |i\rangle$.

GROVER SEARCH (input string of length $n = 2^\ell$, number of iterations t) returns some i such that $x_i = 1$.

1. Let i denote a $\log n$-bit number that indexes the input bits. Put the quantum register into the superposition
$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle \ .$$
2. Repeat t times the following so-called *Grover iteration* G:
   (a) **Phase flip.** Multiply the quantum phase of $|i\rangle$ by -1 if $x_i = 1$.
   $$\mathsf{O}_x : |i\rangle \to (-1)^{x_i}|i\rangle$$
   (b) **Diffusion.** Apply the *diffusion operator* as follows.
   - Apply the Hadamard gate on each qubit of $|i\rangle$.
   - Multiply the quantum phase of $|i\rangle$ by -1 if $i \neq 0$.
   - Apply the Hadamard gate on each qubit of $|i\rangle$.
   $$\mathsf{D} : |i\rangle \to -|i\rangle + \frac{2}{\sqrt{n}}|\psi\rangle \tag{1.9}$$
3. Measure $|i\rangle$ in the computational basis and return it.

Figure 1.4: Quantum algorithm GROVER SEARCH [Gro96]

First, the initial state of the algorithm can be written as

$$|\psi\rangle = \sqrt{\frac{n-w}{n}}|\psi_0\rangle + \sqrt{\frac{w}{n}}|\psi_1\rangle = \cos\alpha|\psi_0\rangle + \sin\alpha|\psi_1\rangle \ ,$$
$$\text{where } \sin\alpha = \langle\psi|\psi_1\rangle = \sqrt{\frac{w}{n}} \ , \tag{1.10}$$

and hence it is in $\mathcal{S}$. Second, the phase flip computed by the oracle O_x maps $|\psi_0\rangle \to |\psi_0\rangle$ and $|\psi_1\rangle \to -|\psi_1\rangle$ and hence it preserves $\mathcal{S}$. Third, the diffusion operator can be decomposed as $\mathsf{D} = \mathsf{H}^{\otimes \log n} \cdot \mathsf{O}_0 \cdot \mathsf{H}^{\otimes \log n}$, because

$$|i\rangle \xrightarrow{\mathsf{H}^{\otimes \log n}} \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} (-1)^{i \bullet j}|j\rangle \qquad \text{by equation (1.5)}$$
$$\xrightarrow{\mathsf{O}_0} -\frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} (-1)^{i \bullet j}|j\rangle + \frac{2}{\sqrt{n}}|0\rangle \qquad \text{using } 0 \bullet j = 0$$

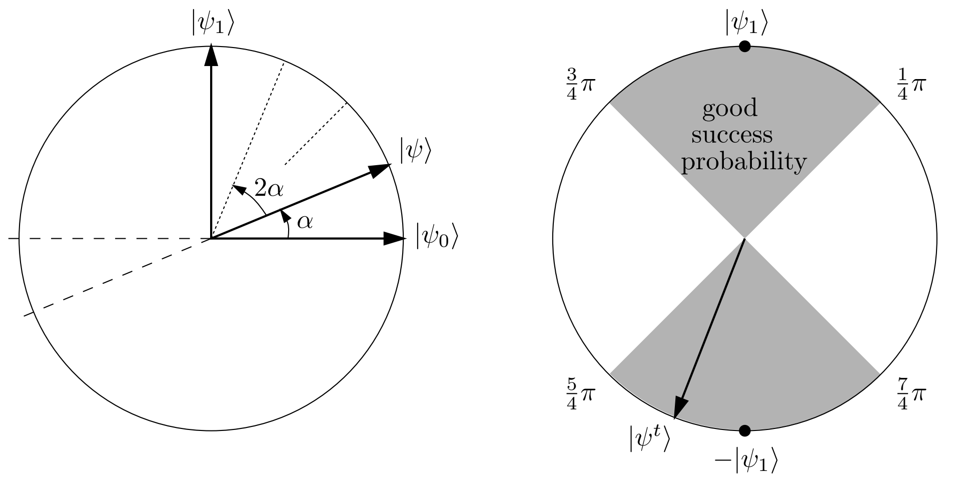


Figure 1.5: The Grover iteration G drawn as a rotation in two dimensions

$$\xrightarrow{\mathsf{H}^{\otimes \log n}} -|i\rangle + 2|\psi\rangle \cdot \langle\psi|i\rangle \ . \qquad \text{H is self-inverse and } \langle\psi|i\rangle = \tfrac{1}{\sqrt{n}}$$

Here $i \bullet j$ denotes the bitwise scalar product of i and j. We conclude that

$$\mathsf{D} = -\mathsf{I} + 2|\psi\rangle\langle\psi| \ ,$$

and it maps

$$\begin{aligned} |\psi_0\rangle &\to -|\psi_0\rangle + 2\cos\alpha\,(\cos\alpha|\psi_0\rangle + \sin\alpha|\psi_1\rangle) = \cos 2\alpha|\psi_0\rangle + \sin 2\alpha|\psi_1\rangle \\ |\psi_1\rangle &\to -|\psi_1\rangle + 2\sin\alpha\,(\cos\alpha|\psi_0\rangle + \sin\alpha|\psi_1\rangle) = \sin 2\alpha|\psi_0\rangle - \cos 2\alpha|\psi_1\rangle \ , \end{aligned}$$

and hence it also preserves $\mathcal{S}$. The Grover iteration G can thus be written in the basis $\{|\psi_0\rangle, |\psi_1\rangle\}$ as a unitary matrix

$$\mathsf{G} = \mathsf{D}\cdot\mathsf{O}_x = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} \cos 2\alpha & -\sin 2\alpha \\ \sin 2\alpha & \cos 2\alpha \end{pmatrix} \ .$$

The operator G is a rotation by an angle of 2α in the two-dimensional subspace $\mathcal{S}$, so its t-th power is $\mathsf{G}^t = \left(\begin{smallmatrix} \cos 2t\alpha & -\sin 2t\alpha \\ \sin 2t\alpha & \cos 2t\alpha \end{smallmatrix}\right)$. We visualize the Grover iteration in Figure 1.5. The phase flip is a reflection around the vector $|\psi_0\rangle$, the diffusion is a reflection around the vector $|\psi\rangle$, and their composition is hence a rotation of the plane spanned by $|\psi_0\rangle$ and $|\psi\rangle$ by twice the angle α between the two vectors.

The initial state of the algorithm is $|\psi\rangle$ and the desired final state is $|\psi_1\rangle$. The total angle one has to rotate by is thus $A = \frac{\pi}{2} - \alpha$. It follows that the optimal number of Grover iterations is $t = \frac{A}{2\alpha} = \frac{\pi}{4\alpha} - \frac{1}{2} = \frac{\pi}{4}\sqrt{\frac{n}{w}} - \frac{1}{2} - O(\sqrt{\frac{w}{n}})$, because $\alpha = \arcsin\sqrt{\frac{w}{n}} = \sqrt{\frac{w}{n}} + O((\frac{w}{n})^{3/2})$ by expanding the power series. However, the algorithm has good success probability in a much wider range of t. Let

$$|\psi^t\rangle = \mathsf{G}^t|\psi\rangle = \cos((2t+1)\alpha)\cdot|\psi_0\rangle + \sin((2t+1)\alpha)\cdot|\psi_1\rangle$$

```
GENERALIZED GROVER SEARCH ($n$-bit input string $x$)
returns some $i$ such that $x_i = 1$, or "zero string".
  1. Let $\lambda = \sqrt[4]{4/3} \doteq 1.07$ and $M = \log_\lambda(20\sqrt{n})$.
  2. For $m = 0, 1, \dots, M$, do the following:
       • Pick $t$ uniformly at random from $\{0, 1, \dots, \lambda^m\}$.
       • Run GROVER SEARCH $(x, t)$ and return $i$ if $x_i = 1$.
  3. Return "zero string".
```

Figure 1.6: Quantum algorithm Generalized Grover search [BBHT98]

denote the quantum state after t Grover iterations. If the *overlap* between $|\psi^t\rangle$ and $|\psi_1\rangle$ is at least $\frac{1}{\sqrt{2}}$, that is if $|\langle\psi^t|\psi_1\rangle| = |\sin((2t+1)\alpha)| \geq \frac{1}{\sqrt{2}}$, then the final measurement yields an i from the superposition over all ones with probability at least $\frac{1}{2}$, hence the success probability is at least $\frac{1}{2}$. The overlap is at least $\frac{1}{\sqrt{2}}$ for any angle from the intervals

$$A \in [\tfrac{1}{4}\pi, \tfrac{3}{4}\pi] \cup [\tfrac{5}{4}\pi, \tfrac{7}{4}\pi] \ ;$$

in the first interval the final state is close to $|\psi_1\rangle$ and in the second interval the final state is close to $-|\psi_1\rangle$. Since the rotation is periodical, one can add any integer multiple of 2π without changing the success probability. Expressing $t = \frac{A}{2\alpha}$ as $t = (k\pi + \frac{\pi}{2} + \varepsilon\frac{\pi}{2})/(2\sqrt{\frac{w}{n}})$, where $k \in \mathbb{Z}$ and $|\varepsilon| \leq \frac{1}{2}$ finishes the proof. □

Unknown number of solutions It may seem that to apply Grover search, one needs to know beforehand a good estimate of the *Hamming weight* of the input string, that is the number of ones. This is not a problem, because one can run the algorithm with $t \in \{1, \lambda, \lambda^2, \lambda^3, \dots, n\}$ for some fixed $\lambda > 1$. At least one guess will be close enough and the algorithm will have constant success probability. By refining this idea a bit further (so that we do not gain an additional logarithmic factor), we get the following algorithm.

1.3.3. Theorem (Quantum search [BBHT98]). *The algorithm* Generalized Grover search *stops after $O(\sqrt{n})$ queries. If the Hamming weight of the input string is $|x| \geq 1$, then the algorithm finds a one with probability at least $1 - \frac{1}{(4|x|)^2} \geq \frac{15}{16}$ and its expected query complexity is $O(\sqrt{n/|x|})$.*

Proof. The algorithm is outlined in Figure 1.6. It runs Grover search several times with exponentially increasing numbers of iterations.

Assume that x is not a zero string. Let $\mu = \log_\lambda(10\sqrt{n/|x|})$. The crucial observation is that once $m \geq \mu$, that is when the value of λ^m exceeds more than 12 times the optimal number of Grover iterations $\frac{\pi}{4}\sqrt{n/|x|}$, the total rotation after λ^m iterations is $\lambda^m \alpha \geq 3 \cdot 2\pi$. Since t is actually picked uniformly at random from $\{0, 1, \dots, \lambda^m\}$, the final state $|\psi^t\rangle$ ends in a random position distributed almost uniformly around the circle. With probability $\frac{1}{2}$, the state lies in the good region where Grover search has success probability at least $\frac{1}{2}$. Hence in *every* round after $m \geq \mu$, the success probability is at least $\frac{1}{4}$, that is the error probability is at most $p \leq \frac{3}{4}$. (With more care, the expected error probability is $\frac{1}{2}$.)

Let J be the random variable denoting the number of rounds after which the algorithm finishes. The expected query complexity can be expressed as a sum

$$\begin{aligned} T &= \sum_{j=0}^{M} \Pr[J = j] \cdot \sum_{m=0}^{j} O(\lambda^j) = \sum_{j=0}^{M} \Pr[J \geq j] \cdot O(\lambda^j) \\ &\leq \sum_{j=0}^{\mu-1} 1 \cdot O(\lambda^j) + \sum_{j=\mu}^{\infty} p^{j-\mu} O(\lambda^j) = O(\lambda^\mu) \cdot \left(1 + \sum_{j=0}^{\infty} (p\lambda)^j\right) \\ &\leq O(\lambda^\mu) = O(\sqrt{n/|x|}) \ , \end{aligned}$$

where the last inequality holds because $p\lambda \leq \frac{3}{4} \cdot 1.07 \leq 0.81$. The number of rounds is limited by $M + 1$, hence the total query complexity is at most $O(\sqrt{n})$. If there is at least one 1 in the input string, then it will be missed with probability at most p^k, where

$$k = M - \mu = \log_\lambda \frac{20\sqrt{n}}{10\sqrt{n/|x|}} = \frac{\log_\lambda(4|x|)}{2} = \frac{\log_p(4|x|)}{2\log_p \lambda} = -2\log_p(4|x|)$$

is the number of iterations after $\lambda^m \geq 10\sqrt{n/|x|}$. We use that $\lambda = p^{-1/4}$. The success probability is hence at least $1 - p^k = 1 - \frac{1}{(4|x|)^2}$. □

Running time We show later in Chapter 2 that the query complexity of unordered search is $\Omega(\sqrt{n})$ and thus Grover search uses an asymptotically optimal number of queries. The running time of Grover search is, however, $O(\log n)$ times bigger than its query complexity, because $2\log n$ Hadamard transforms and a logical OR of $\log n$ qubits are computed in each diffusion step. Grover showed [Gro02] how to reduce the running time to $O(\sqrt{n}\log\log n)$ while increasing the query complexity only by a constant factor, using techniques introduced in the rest of this chapter. Most algorithms presented in this thesis are efficient, that is their running time is within a polylogarithmic factor of their query complexity.

Exact quantum search Quantum search algorithms can be made exact, that is the error probability can be made zero, if we know exactly the number of ones in the input. From the proof of Theorem 1.3.2, if the number of Grover iterations is exactly $t = \frac{A}{2\alpha} = \frac{\pi}{4\alpha} - \frac{1}{2}$, where $\alpha = \arcsin \sqrt{|x|/n}$, then the final state is exactly $|\psi_1\rangle$ and the measurement yields 1 with certainty. For example, if $|x| = \frac{n}{4}$, then $\alpha = \arcsin \frac{1}{2} = \frac{\pi}{6}$ and the optimal number of iterations is just $t = \frac{6}{4} - \frac{1}{2} = 1$. We can always compute t first, but the problem is what to do when the optimal t is not an integer, because then neither $\lfloor t \rfloor$ nor $\lceil t \rceil$ iterations yield exactly $|\psi_1\rangle$.

The solution is to perform $\lfloor t \rfloor$ normal iterations and one last *slower iteration.* We change the last Grover iteration such that quantum phases get multiplied by $e^{i\theta_1}$ for some angle θ_1 instead of -1 in the phase flip step and, similarly, by $e^{i\theta_2}$ in the step O_0. It turns out that for every $\alpha' \in [0, \alpha]$ there exist θ_1, θ_2 such that the slower Grover iteration rotates by angle $2\alpha'$.

1.3.4. Corollary (Exact quantum search [BHMT02]). *For every $w \in [1, n]$, there is a quantum algorithm that finds a one with certainty in an n-bit string containing exactly w ones, and uses $O(\sqrt{n/w})$ queries.*

Finding all ones If we want to find all ones in the input string instead of just one of them, we can iterate Generalized Grover search. We keep a list of all found ones in the memory and modify the oracle such that it flips the phase of $|i\rangle$ only if $x_i = 1$ *and* the index i is not yet in the list. At the beginning, the quantum search proceeds quite fast thanks to the existence of many solutions, and it gets slower at the end when only a few of them are left. The expected query complexity of finding all ones in a string of Hamming weight $|x|$ is

$$\sum_{k=1}^{|x|} O\left(\sqrt{\frac{n}{|x| - k + 1}}\right) = \sum_{k=1}^{|x|} O\left(\sqrt{\frac{n}{k}}\right) = O(\sqrt{n|x|}) \ . \tag{1.11}$$

We need to take into account additional $O(\sqrt{n})$ queries for the final check that there are no more solutions.

1.3.5. Corollary ([BBC+01]). *There is a quantum algorithm that finds all ones in an n-bit input string with probability at least $\frac{8}{9}$. It worst-case query complexity is $O((|x|+1)\sqrt{n})$ and its expected query complexity is $O(\sqrt{n(|x|+1)})$.*

Proof. As stated above, we run Generalized Grover search while it succeeds in finding ones. The worst-case bound holds because there are at most $|x| + 1$ iterations, each with $O(\sqrt{n})$ queries. The bound on the expected query complexity follows from Theorem 1.3.3 and (1.11). The algorithm succeeds if none of the $|x|$ ones is missed. By the same Theorem 1.3.3, when there are w remaining ones, the error probability is at most $\frac{1}{(4w)^2}$. By the union bound, the total error probability is at most $\frac{1}{16} \sum_{w=1}^{|x|} \frac{1}{w^2} \leq \frac{\pi^2}{16 \cdot 6} < \frac{1}{9}$. □

1.3.2 Searching an unstructured database

Finding ones in a bit string is a rather specific problem. However, using a simple trick, GROVER SEARCH can be used to search any unstructured database. Let Σ be any finite alphabet, for example $[n]$. Consider an input consisting of a string $x \in \Sigma^n$ and an item $y \in \Sigma$. The problem is to decide whether there exists an index $i \in [n]$ such that $x_i = y$. We define the following oracle P_x.

$$\mathsf{P}_x : |i\rangle|y\rangle \to \begin{cases} -|i\rangle|y\rangle & x_i = y \\ |i\rangle|y\rangle & \text{otherwise} \end{cases}$$

If our new oracle P_x is used instead of the original oracle O_x, then GROVER SEARCH implements searching for y in the string x.

The oracle P_x can be computed using O'_x as follows. Assume we have a clean ancilla register z. First, we load x_i into z using O'_x. Then we compare y with z and flip the quantum phase if they are equal. Finally, we unload x_i, that is clear the contents of the register z by another application of O'_x. The computation only takes 2 queries and time $O(\log n)$ for the comparison of two numbers.

1.3.3 Amplitude amplification

Quantum computers offer substantial speedup in a yet more general setting. Assume we have a subroutine that performs some computational task with success probability $p > 0$, and another subroutine that verifies deterministically whether the first subroutine succeeded. Classically, to amplify the success probability to a constant, we repeat the subroutines c/p times for some constant c. The probability that the computation has always failed is $(1-p)^{c/p} \leq e^{-c}$. We call this process *probability amplification*, because for $p \ll 1$, the success probability after t iterations gets amplified to roughly tp. This probability amplification is optimal as there is no algorithm that runs faster for all black-box subroutines. Quantum computers, however, perform quadratically better and they can amplify any subroutine using just $O(\sqrt{1/p})$ applications. This process is called *amplitude amplification*, because the quantum amplitudes of the solutions get amplified in a similar way as the probabilities in the classical case.

1.3.6. THEOREM (AMPLITUDE AMPLIFICATION [BHMT02]). *Let S be a subset of $[n]$ and let O_S be a quantum algorithm that flips the phase of $|i\rangle$ if $i \in S$; the elements of S are solutions we are looking for. Let A be a quantum algorithm that does not perform any measurements and that computes some superposition of elements $|\psi\rangle = \sum_{i=1}^n \alpha_i |i\rangle$. Let $p = \sum_{i \in S} |\alpha_i|^2$ denote the probability that we obtain an element from S if we measure $|\psi\rangle$ in the computational basis. Then there is a quantum algorithm that uses $O(\sqrt{1/p})$ applications of $\mathsf{A}, \mathsf{A}^{-1}, \mathsf{O}_S$, and finds an element of S with constant probability.*

PROOF (SKETCH). The algorithm is similar to GROVER SEARCH, but it uses the subroutines A and A^{-1} instead of the Hadamard transform on each qubit. The final state after t iterations is

$$|\psi^t\rangle = (\mathsf{A} \cdot \mathsf{O}_0 \cdot \mathsf{A}^{-1} \cdot \mathsf{O}_S)^t \cdot \mathsf{A}|0\rangle \ ,$$

where $\mathsf{A}|0\rangle = |\psi\rangle$ is the superposition computed by the algorithm and O_0 flips the phase of $|i\rangle$ if $i \neq 0$. Here we see why measurements are not allowed in A; we need to be able to reverse the computation.

By the same arguments as in Theorem 1.3.2, the quantum state stays in a two-dimensional real subspace $\mathcal{S}$ spanned by $|\varphi_0\rangle$ and $|\varphi_1\rangle$, where $|\varphi_0\rangle$ is the projection of $|\psi\rangle$ onto the bad subspace spanned by the set of basis states $|i\rangle$ for which $i \notin S$, and $|\varphi_1\rangle$ is the projection of $|\psi\rangle$ onto the good subspace. Note that $p = \|\varphi_1\|^2$. Normalize $|\psi_1\rangle = \frac{|\varphi_1\rangle}{\|\varphi_1\|}$. Then $\langle\psi|\psi_1\rangle = \frac{\langle\psi|\varphi_1\rangle}{\|\varphi_1\|} = \frac{\langle\varphi_1|\varphi_1\rangle}{\|\varphi_1\|} = \|\varphi_1\| = \sqrt{p}$. A simple computation yields that the optimal number of iterations is $t = \frac{\pi}{4\alpha} - \frac{1}{2}$, where $\alpha = \arcsin\langle\psi|\psi_1\rangle$ like in equation (1.10), which is roughly $\frac{\pi}{4\langle\psi|\psi_1\rangle} = \frac{\pi}{4}\sqrt{\frac{1}{p}}$. □

Let us show a non-trivial application of quantum amplitude amplification.

1.3.7. PROBLEM (ELEMENT DISTINCTNESS). *The input is a sequence of n numbers $x_1, \dots, x_n \in [n]$. A* collision *is a pair $(i, j) \in [n] \times [n]$ such that $i \neq j$ and $x_i = x_j$. The problem is to decide whether there is a collision in x.*

The classical complexity of element distinctness is $\Omega(n)$. The best known algorithm sorts the input numbers and then it looks for a collision in another round. Quantum computers cannot sort faster than $O(n \log n)$, however they can solve element distinctness polynomially faster.

1.3.8. THEOREM ([BDH+01]). *There is a bounded-error quantum algorithm for element distinctness that uses $O(n^{3/4})$ queries and runs in time $O(n^{3/4} \log^2 n)$.*

PROOF. The algorithm is outlined in Figure 1.7. The first step uses $O(\sqrt{n})$ queries. Sorting in the second step does not use any query. The GROVER SEARCH in the third step uses $O(\sqrt{n})$ queries, because each Grover iteration uses 2 queries to load and unload the input number. In each iteration, $O(\log n)$ rounds of classical binary search within the contents of the memory are performed, however they use no further queries. Hence the first three steps together use $O(\sqrt{n})$ queries. Assume that there is a collision (i, j) in x. The probability that we have picked the right interval I containing at least one of i, j is at least $\frac{1}{|I|} = \frac{1}{\sqrt{n}}$. The success probability of GROVER SEARCH is at least $\frac{1}{2}$, hence the success probability of the first three steps is $p \geq \frac{1}{2}\frac{1}{\sqrt{n}}$. Amplitude amplification hence takes $O(\sqrt{1/p}) = O(n^{1/4})$ rounds and the total query complexity is $O(n^{3/4})$.

The running time is $O(\log^2 n)$ times bigger than the query complexity, because both sorting and binary searching in the first three steps take $O(\sqrt{n} \log n)$ operations on $\log n$-bit numbers. □

```
OLDER ELEMENT DISTINCTNESS ($n$ input numbers $x_1, \dots, x_n$)
returns whether there are $i \neq j$ such that $x_i = y_i$.

  1. Pick a random number $s \in [\sqrt{n}]$. Let $I$ denote the interval
     $[(s-1)\sqrt{n}+1, s\sqrt{n}]$. Read the numbers indexed by $I$.

  2. Sort classically the numbers in $I$ according to their $x$-value.

  3. For a specific $j \in [n]$, one can check if there is an $i \in I$ such that
     $x_i = x_j$ using classical binary search on $I$. Use GENERALIZED
     GROVER SEARCH to find out whether there is a $j \in [n]$ for which
     such an $i \in I$ exists.

  4. Apply quantum amplitude amplification on the first three steps of
     the algorithm.
```

Figure 1.7: Quantum algorithm OLDER ELEMENT DISTINCTNESS [BDH+01]

1.4 Quantum random walks

A quantum walk is a quantum variant of a classical random walk on a graph. Instead of flipping a coin at random and walking based on the outcome of the coin flip, we put the coin into a quantum superposition and thus walk in a superposition of all directions. Quantum effects, such as negative interference, cause the properties of quantum walks to be different from those of random walks.

1.4.1 Element distinctness

In the previous section, we defined the element distinctness problem and presented a fast quantum algorithm for it based on quantum search. In this section, we show a fundamental quantum algorithm based on quantum walks that solves this problem even faster; in fact it is optimal. This algorithm is novel and quite universal, and a whole branch of quantum algorithms for several important and natural problems are inferred from it. We will see more examples later.

1.4.1. DEFINITION. The *Johnson graph* $J(n,k)$ is defined as follows: its vertices are all subsets of $[n]$ of size k, and two vertices are connected by an edge if and only if their symmetric difference is 2, that is one can get one subset from the other one by replacing one number.

Ambainis's quantum walk algorithm ELEMENT DISTINCTNESS [Amb04] is outlined in Figure 1.8. It walks on the Johnson graph $J(n,k)$ for $k = n^{2/3}$. A graph vertex corresponds to a subset of k input numbers loaded into the memory and one step of the walk corresponds to replacing one number by another one.

Element distinctness (n input numbers $x_1, \dots, x_n$)
returns whether there are $i \neq j$ such that $x_i = y_i$.

1. Let $k = n^{2/3}$ denote the number of elements kept in the memory in each basis state. Put the quantum register into the superposition
$$\frac{1}{\sqrt{\binom{n}{k}}} \frac{1}{\sqrt{n-k}} \sum_{\substack{S \subseteq [n] \\ |S| = k}} |S\rangle \sum_{y \in [n] - S} |y\rangle \ .$$
Read the numbers indexed by S into a third register.
2. Repeat $t_1 = \Theta(n/k)$ times the following:
   (a) **Phase flip O_S.** Multiply the quantum phase of $|S\rangle$ by -1 if there is a collision inside S.
   (b) **Diffusion D'.** Apply $t_2 = \Theta(\sqrt{k})$ steps of quantum walk by running t_2 times One Step ($|S\rangle|y\rangle$).
3. Measure S and verify classically whether there is a collision in the observed set S.

One Step (subset $|S\rangle$ of size k, index $|y\rangle$ outside S)
is a quantum subroutine applying a unitary operation on its arguments.

4. Diffuse $|y\rangle$ by mapping $|S\rangle|y\rangle \to |S\rangle \left(-|y\rangle + \frac{2}{n-k} \sum_{y' \notin S} |y'\rangle\right)$.
5. Insert y into the set S and read x_y by querying the input.
6. Diffuse $|y\rangle$ by mapping $|S\rangle|y\rangle \to |S\rangle \left(-|y\rangle + \frac{2}{k+1} \sum_{y' \in S} |y'\rangle\right)$.
7. Erase x_y by querying the input and remove y from the set S.

Figure 1.8: Quantum algorithm Element distinctness [Amb04]

At first glance, ELEMENT DISTINCTNESS resembles GROVER SEARCH. At the beginning, the quantum register is put into the uniform superposition of all basis states. Then a sequence of phase flips and diffusions is performed. Note that if a set S is picked uniformly at random, then the probability that it contains a collision is roughly $p = (k/n)^2$ (assuming there is exactly one collision), and the number of iterations is proportional to $\sqrt{1/p} = n/k$ exactly like in GROVER SEARCH. There is, however, one crucial difference between the algorithms and that is the implementation of the diffusion step. An ideal diffusion operator maps

$$\mathsf{D} : |S\rangle \to -|S\rangle + \frac{2}{\sqrt{\binom{n}{k}}} \cdot \frac{1}{\sqrt{\binom{n}{k}}} \sum_{\substack{S' \subseteq [n] \\ |S'|=k}} |S'\rangle \ ,$$

exactly like in equation (1.9). Had we performed exactly this operator D, we could apply Theorem 1.3.2 and conclude that the algorithm finds a collision with a constant probability. We do not do this for a simple reason: the ideal diffusion operator D costs k queries to x, because it can replace a subset S by another subset S' consisting of elements outside S. This means that the total number of queries of such an algorithm would be at least $\frac{n}{k} \cdot k \geq n$, which is worse than that of OLDER ELEMENT DISTINCTNESS!

New diffusion operator This difficulty can be overcome as follows. We define a different diffusion operator D' that can be computed using only $O(\sqrt{k})$ queries and prove that D' behaves equally well as the ideal diffusion operator D for the purpose of quantum search. D' is computed by t_2 applications of the quantum subroutine ONE STEP. The quantum walk uses an additional register $|y\rangle$ called a *coin flip register*. It determines which number is to be added to or deleted from S, and the ideal diffusion operator (1.9) is applied on $|y\rangle$ in between these operations. This is the second difference from GROVER SEARCH, which does not need to use any coin flip register. Basically, all discrete-time quantum walks use coin flip registers.

1.4.2. THEOREM ([AMB04]). *Let $x_1, \dots, x_n$ contain either no collision or exactly one collision. Then* ELEMENT DISTINCTNESS *uses $O(n^{2/3})$ queries and it has constant success probability.*

PROOF (SKETCH). The algorithm has already been described in Figure 1.8. Its total query complexity is $O(k + \frac{n}{k}(0 + \sqrt{k})) = O(k + \frac{n}{\sqrt{k}})$. We minimize it by setting $k = n^{2/3}$ and thus use $O(n^{2/3})$ queries.

Assume that there is a collision (i, j). It remains to prove that the subset S at the end contains the collision with probability at least a constant. We analyze the eigenspaces of the unitary operator $\mathsf{G} = \mathsf{D}' \cdot \mathsf{O}_S$. The analysis is feasible because, like in GROVER SEARCH, the quantum state stays in a constant-dimensional subspace $\mathcal{S}$. Here $\mathcal{S}$ has 5 dimensions and it is spanned by $\{|\psi_1\rangle, |\psi_2\rangle, |\psi_3\rangle, |\psi_4\rangle, |\psi_5\rangle\}$, where $|\psi_\ell\rangle$ is the uniform superposition of states of type ℓ as follows. Each state $|S\rangle|y\rangle$ has a type assigned according to the following table:

$$\begin{array}{lll} \text{type} = 1 \text{ if} & |S \cap \{i,j\}| = 0 & y \notin \{i,j\} \\ \quad\quad\; 2 & |S \cap \{i,j\}| = 0 & y \in \{i,j\} \\ \quad\quad\; 3 & |S \cap \{i,j\}| = 1 & y \notin \{i,j\} \\ \quad\quad\; 4 & |S \cap \{i,j\}| = 1 & y \in \{i,j\} \\ \quad\quad\; 5 & |S \cap \{i,j\}| = 2 & y \notin \{i,j\} \end{array} \tag{1.12}$$

For example $|\psi_1\rangle = \sum_{S:|S\cap\{i,j\}|=0} \sum_{y \notin S \cup \{i,j\}} |S\rangle|y\rangle$. Note that there is no 6th subspace, because when both $i, j \in S$, they cannot be simultaneously equal to y. It is straightforward to verify that both O_S and D' preserve $\mathcal{S}$.

The operator One Step has 5 different eigenvalues. One of them is 1 with the uniform superposition being the eigenvector. The other ones are $e^{\pm i\theta_j}$ for $j = 1, 2$ with $\theta_j = (2\sqrt{j} + o(1))\frac{1}{\sqrt{k}}$. Let us perform $t_2 = \lceil \frac{\pi}{3\sqrt{2}} \sqrt{k} \rceil$ quantum walk steps in the diffusion operator; this means that the eigenvalues are raised to the power of t_2. Hence $\mathsf{D}' = \text{One Step}^{t_2}$ has eigenvalues 1 and $e^{\pm i(\frac{2\pi}{3}\sqrt{\frac{j}{2}}+o(1))}$ for $j = 1, 2$. The latter eigenvalues are bounded away from 1; they are $e^{i\theta}$ with $\theta \in [c, 2\pi - c]$ for a constant $c > 0$ independent of n, k. Recall that the ideal diffusion operator D of Grover search in the proof of Theorem 1.3.2 has the same properties: it has one eigenvalue 1 with the uniform superposition being the eigenvector, and one eigenvalue -1 (which is bounded away from 1).

Ambainis showed [Amb04, Lemma 3] that these two properties are sufficient. He proved that any operator like this can be used instead of the ideal diffusion operator D and the quantum search will still behave like Grover search. It follows that $t_1 = \Theta(\sqrt{1/p}) = \Theta(n/k)$ iterations are enough to find the collision with constant probability. □

Running time The running time of Element distinctness can be much bigger than its query complexity. The phase flip costs 0 queries, because all elements from S are read in the memory, but the computation takes some time. Two simple approaches fail: naive comparison of all pairs takes time $O(k^2)$ and sorting takes time $O(k \log k)$, hence the total running time would be $\Omega(n)$, which is slower than Older Element distinctness. One may try to store the elements in a balanced search tree, such as a red-black tree [Bay72], that takes time $O(\log k)$ per operation. This approach is fast, but it fails too for a different reason. The same set of elements can be encoded by many trees of different shapes depending on the order of insertions and deletions. We need to have a unique encoding for the sake of the quantum interference. The correct solution is to use a random hash function and compare all pairs within the same collision group.

1.4.3. Corollary ([Amb04]). *There is a bounded-error quantum algorithm for element distinctness with 0 or 1 collision running in time $O(n^{2/3} \log^c n)$ for some $c > 0$.*

Many collisions It remains to solve element distinctness in the case when there might be more than one collision. The proof of Theorem 1.4.2 does not work in this setting— the subspace $\mathcal{S}$ can have much more than 5 dimensions because the states cannot be classified into a constant number of cases like in equation (1.12). A simple trick of Valiant and Vazirani [VV86] helps here. If there are roughly w collisions, we first pick a random subset $T \subseteq [n]$ of size $n/\sqrt{w}$ and then run Element distinctness on T. With constant probability, the number of collisions inside T is 1 and hence the algorithm has constant success probability. If we do not know the value of w, then we probe all values from an exponentially decreasing sequence like in Theorem 1.3.3. The running time is decreased when there are many collisions.

1.4.2 Walking on general graphs

The quantum algorithm Element distinctness from the previous section walks on a Johnson graph and the analysis in Theorem 1.4.2 is specific for this graph. There are many efficient classical algorithms that walk on different graphs, such as grids or hypercubes, therefore it is interesting to investigate whether quantum computing is of any help in the general case. It turns out that quantum walks converge quadratically faster on any undirected regular graph.

Let us define a few concepts from the classical theory of random walks on graphs.

1.4.4. Definition. Let $G = (V, E)$ be an undirected d-regular graph, that is a graph where each vertex has degree d. The *normalized adjacency matrix* A is defined as $A[i, j] = \frac{1}{d}$ if $(i, j) \in E$, and 0 otherwise. A has principal eigenvalue 1 with the all-ones vector being the eigenvector. The *spectral gap* of G, denoted by δ_G, is the difference between 1 and the second largest eigenvalue of A.

The spectral gap is nonzero if and only if G is connected. The spectral gap is proportional to the inverse of the *mixing time* of a random walk on the graph, that is the number of random walk steps after which the probability distribution on the vertices is close to uniform regardless of the starting vertex.

Many practical algorithms can be modelled as instances of the following abstract problem.

1.4.5. Problem. *Let G be a fixed graph on a vertex set V. Some subset of vertices $M \subseteq V$ are marked. The input is a bit string of vertex marks. The promise is that $|M|$ is either zero or at least $\varepsilon|V|$. The task is to decide which is the case.*

Szegedy [Sze04] solved this problem by the algorithm Decide Marked Vertices outlined in Figure 1.9. The algorithm is a quantum walk algorithm. This means that if it queries a vertex x at some moment, then only neighbors of x

Decide Marked Vertices (undirected graph $G = (V, E)$, marked vertices $M \subseteq V$, maximal number of iterations t)
returns whether $M \neq \emptyset$.

1. **Initialization.** Put the quantum register into the superposition

$$|\psi\rangle = \frac{1}{\sqrt{|V|}} \sum_{x \in V} |x\rangle|\psi_x\rangle, \text{ where} \qquad |\psi_x\rangle = \frac{1}{\sqrt{d}} \sum_{y:(x,y)\in E} |y\rangle$$

   is a uniform superposition of all neighbors of x.

2. Pick a number of iterations $\ell \in \{1, 2, \ldots, t\}$ uniformly at random. Repeat ℓ times the following:

   (a) **Phase flip.** Multiply the phase of $|x\rangle$ by -1 if $x \in M$.

   (b) **Diffusion.** Perform one step of quantum walk on G. Map

$$|x\rangle|y\rangle \to |x\rangle \otimes \mathsf{D}_x|y\rangle$$

   and, then, map

$$|x\rangle|y\rangle \to \mathsf{D}_y|x\rangle \otimes |y\rangle \ ,$$

   where $|x\rangle, |y\rangle$ are computational basis states, and

$$\mathsf{D}_x = -\mathsf{I} + 2|\psi_x\rangle\langle\psi_x| \ . \tag{1.13}$$

3. Return "non-empty" if the final quantum state has small scalar product with the uniform superposition $|\psi\rangle$. The scalar product is estimated using Lemma 1.4.7.

Figure 1.9: Quantum algorithm Decide Marked Vertices [Sze04]

can be queried next time. The algorithm follows the same line as other quantum search algorithms: it starts in a fixed superposition and it interleaves phase flips and diffusion steps. However, the structure of its coin flip register is a bit more general than that of ELEMENT DISTINCTNESS. Instead of using one additional register of dimension equal to the degree of the graph, the vertex register is doubled. It always holds that these two registers contain two adjacent vertices. At each step, one of the two registers is used as a control register and the other one as a coin flip register, and the roles of the two registers get exchanged after each application. The diffusion operator D_x defined by equation (1.13) maps

$$\mathsf{D}_x : |y\rangle \to \begin{cases} -|y\rangle & (x,y) \notin E \\ -|y\rangle + \frac{2}{\sqrt{d}}|\psi_x\rangle & (x,y) \in E \end{cases},$$

that is it diffuses the neighbors of x like equation (1.9) and leaves other vertices untouched up to the phase. It follows that D_x can be implemented efficiently using only two transitions along an edge in E: we first move from the old neighbor y back into x and then we move to the new neighbor y'. It is simple to prove that D_x on a Johnson graph exactly computes the diffusion operator of ELEMENT DISTINCTNESS up to an isomorphism between quantum registers, hence DECIDE MARKED VERTICES is not really a new quantum algorithm on this graph. However, as we have said above, DECIDE MARKED VERTICES works well even for more general graphs.

1.4.6. THEOREM ([SZE04]). *Let $G = (V, E)$ be an undirected regular graph with spectral gap δ_G and let $M \subseteq V$ be a subset of vertices such that either $|M| = 0$ or $|M| \geq \varepsilon|V|$. Let $t_0 = \Theta(1/\sqrt{\delta_G \varepsilon})$. For every $t \geq t_0$,* DECIDE MARKED VERTICES *(G, M, t) decides whether M is non-empty with 1-sided error $\gamma \in [\frac{1}{2}, \frac{7}{8}]$. The algorithm never makes an error when M is empty.*

PROOF (SKETCH). If there is no marked vertex, then the algorithm only performs the diffusions and both of them are identity on $|\psi\rangle = \frac{1}{\sqrt{|V|}} \sum_{x \in V} |x\rangle|\psi_x\rangle = \frac{1}{\sqrt{|V|}} \sum_{y \in V} |\psi_y\rangle|y\rangle$. This means that the scalar product at the end is 1. On the other hand Szegedy proved [Sze04, Lemma 7, Lemma 10, and Corollary 2] that, if there are many marked vertices, then after t_0 iterations the scalar product is at most a constant (smaller than 1) with constant probability. We estimate the scalar product by a *swap test* as follows. At the beginning, we put a one-qubit register $|z\rangle$ into the superposition $|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ and condition the whole algorithm by $|z\rangle$. At the end, we apply the Hadamard operation on $|z\rangle$ and measure it in the computational basis. By Lemma 1.4.7, we obtain 0 with certainty in the first case and 1 with constant probability in the second case. □

1.4.7. Lemma (Hadamard test [BCWW01]). *Let $|\varphi\rangle$ and $|\psi\rangle$ be two quantum states, let $|X\rangle = \frac{\sqrt{2}}{2}(|0,\varphi\rangle + |1,\psi\rangle)$ be their superposition with a control qubit, and let $|Y\rangle = (\mathsf{H} \otimes \mathsf{I})|X\rangle$ be the state after applying the Hadamard operation on the control qubit. If the control qubit of $|Y\rangle$ is measured in the computational basis, then* $\Pr[Y=1] = \frac{1}{2}(1 - \Re\langle\varphi|\psi\rangle)$.

Proof. We write

$$|Y\rangle = \frac{|0,\varphi\rangle + |1,\varphi\rangle}{2} + \frac{|0,\psi\rangle - |1,\psi\rangle}{2} = |0\rangle\frac{|\varphi\rangle + |\psi\rangle}{2} + |1\rangle\frac{|\varphi\rangle - |\psi\rangle}{2} \quad ,$$

hence

$$\begin{aligned}\Pr[Y=1] &= \left\|\frac{|\varphi\rangle - |\psi\rangle}{2}\right\|^2 = \frac{(\langle\varphi| - \langle\psi|)(|\varphi\rangle - |\psi\rangle)}{4} \\ &= \frac{\langle\varphi|\varphi\rangle + \langle\psi|\psi\rangle - \langle\varphi|\psi\rangle - \langle\psi|\varphi\rangle}{4} = \frac{1 - \Re\langle\varphi|\psi\rangle}{2} \quad .\end{aligned}$$

□

Application to element distinctness Let us show how to solve element distinctness using the general quantum walk algorithm. As before, we walk on the Johnson graph $J(n,k)$ (see Definition 1.4.1). A vertex is marked if and only if the corresponding subset contains a collision; this can be tested by only looking at the elements of the subset. If the quantum register is in state $|S\rangle$, then the numbers indexed by S are read in the memory. In the initialization step we need to read them all, hence it takes $O(k)$ queries. The phase flip on the other hand takes 0 queries. One step of the quantum walk takes 2 queries, one for an insertion and one for a deletion, because edges only connect subsets that differ by replacing one number.

1.4.8. Lemma ([Knu03]). *The spectral gap of $J(n,k)$ is $\delta_J = \frac{n}{(n-k)k}$.*

If there is at least one collision, then the fraction of marked vertices is at least $\varepsilon = (\frac{k}{n})^2$. By Theorem 1.4.6, $t_0 = \Theta(\sqrt{1/\delta_J\varepsilon}) = \Theta(\frac{n}{\sqrt{k}})$ iterations of the quantum walk are sufficient. The total number of queries is thus $O(k + \frac{n}{\sqrt{k}})$, which gives query complexity $O(n^{2/3})$ for $k = n^{2/3}$. This complexity equals the complexity of Element distinctness.

Comparison between the two quantum walk algorithms On a Johnson graph, Szegedy's quantum walk algorithm Decide Marked Vertices is just an extended version of Ambainis's algorithm Element distinctness with a slightly different coin-flip register. The main differences between these two algorithms are: (1) Szegedy's walk performs a phase flip after *every* diffusion step, whereas Ambainis's walk only does it once in every $\Theta(\sqrt{k})$ steps, (2) Szegedy's walk only decides whether there exists a solution, whereas Ambainis's walk actually *finds* a solution. On the other hand, Ambainis's walk has only been analyzed for the case of at most one collision, whereas Szegedy's walk is known to converge in every case. Moreover, Szegedy's versatile coin-flip register allows us to apply it on any graph. These two quantum walks are thus incomparable and there are algorithms (such as triangle finding [MSS05]) that exploit the faster speed of Ambainis's walk and that thus would not work fast with Szegedy's walk. It is an interesting open problem whether these two algorithms can be merged.

Walking on Markov chains Let us mention briefly a useful generalization of DECIDE MARKED VERTICES coming from the same paper [Sze04]. Theorem 1.4.6 also works for all symmetric ergodic Markov chains. A *Markov chain* is roughly speaking an undirected graph with weights on the edges that represent transition probabilities. This allows walking on non-regular graphs as long as we tune the weights such that the transition probabilities are symmetric. Thanks to its universality, this version can be regarded as a quantum walk version of amplitude amplification Theorem 1.3.6. It is not known whether quantum walks offer any speedup for directed graphs.

Finally, let us compare the performance of quantum random walks with classical random walks and quantum search. Problem 1.4.5 can be solved by classical random walks in $O(1/\delta_G \varepsilon)$ iterations. Ordinary quantum search converges faster in $O(1/\delta_G \sqrt{\varepsilon})$ iterations, and, finally, quantum random walks converge fastest in $O(1/\sqrt{\delta_G \varepsilon})$ iterations.

1.5 Quantum counting

In several algorithms, one needs to estimate the number of solutions before searching for them, for example for the sake of the initialization of data structures. In this section, we present an elegant quantum algorithm for this problem. It combines GROVER SEARCH and the quantum Fourier transform, which is the heart of the famous polynomial-time quantum factoring algorithm [Sho97] invented by Peter Shor in 1994.

1.5.1. DEFINITION. For an integer t and a real number $\beta \in \mathbb{R}$, let $|\varphi_t(\beta)\rangle = \frac{1}{\sqrt{t}} \sum_{y=0}^{t-1} e^{2\pi i \beta y} |y\rangle$. The *quantum Fourier transform* with respect to modulus t maps $\mathsf{F}_t : |x\rangle \to |\varphi_t(\frac{x}{t})\rangle$.

Coppersmith showed [Cop94] how to compute F_{2^ℓ} by a quantum circuit of depth and size both $O(\ell^2)$ using no ancilla qubits. It is folklore that the depth can be improved to $O(\ell)$. Hales and Hallgren showed [HH99] how to approximate F_t by $\mathsf{F}_{t'}$ with polynomially small error for any $t' \geq t^3$. By concatenating these two constructions, F_t for any integer t can be approximated in depth $O(\log t)$,

Quantum Counting (input string x, number of queries t)
returns an estimate of $w = |x|$.

1. Initialize two registers to the state $\mathsf{F}_t|0\rangle \otimes |\psi\rangle$.
2. If the first register is $|y\rangle$, apply y Grover iterations on the second register, that is map $|y\rangle|\psi\rangle \to |y\rangle \otimes \mathsf{G}^y|\psi\rangle$.
3. Apply F_t^{-1} to the first register.
4. Measure the first register and denote the outcome by $\tilde{y}$.
5. Return $\tilde{w} = n \cdot \sin^2(\pi \frac{\tilde{y}}{t})$.

Figure 1.10: Quantum algorithm Quantum Counting [BHMT02]

which is good enough for our purposes. Cleve and Watrous [CW00] later showed how to approximate F_t in depth $O(\log\log t)$, however we do not need their better construction here.

1.5.1 General counting algorithm

The algorithm is outlined in Figure 1.10.

1.5.2. Theorem ([BHMT02]). *For any integers $k, t > 0$, the algorithm* Quantum Counting (x, t) *outputs an estimate $\tilde{w}$ of $w = |x|$ such that*

$$|\tilde{w} - w| \le 2\pi k \frac{\sqrt{w(n-w)}}{t} + \pi^2 k^2 \frac{n}{t^2}$$

with probability at least $\frac{8}{\pi^2}$ when $k = 1$, and with probability greater than $1 - \frac{1}{2(k-1)}$ for $k \ge 2$. If $w = 0$, then $\tilde{w} = 0$ with certainty, and if $w = n$ and t is even, then $\tilde{w} = n$ with certainty.

Proof. Following the proof of Theorem 1.3.2, the Grover iteration can be expressed in the basis $\{|\psi_0\rangle, |\psi_1\rangle\}$ as $\mathsf{G} = \left(\begin{smallmatrix} \cos 2\alpha & -\sin 2\alpha \\ \sin 2\alpha & \cos 2\alpha \end{smallmatrix}\right)$, where $\sin^2 \alpha = \frac{w}{n}$. G has eigenvectors $|\psi_\pm\rangle = \frac{|\psi_0\rangle \pm i|\psi_1\rangle}{\sqrt{2}}$ with eigenvalues $e^{\mp 2i\alpha}$. The starting vector can be decomposed as $|\psi\rangle = \cos\alpha|\psi_0\rangle + \sin\alpha|\psi_1\rangle = \frac{e^{-i\alpha}|\psi_+\rangle + e^{i\alpha}|\psi_-\rangle}{\sqrt{2}}$. Hence, after the first step, the quantum state is

$$|\varphi_t(0)\rangle|\psi\rangle = \frac{1}{\sqrt{2t}} \sum_{y=0}^{t-1} |y\rangle \left(e^{-i\alpha}|\psi_+\rangle + e^{i\alpha}|\psi_-\rangle\right) \quad ,$$

which gets mapped in the second step to

$$\begin{aligned}
&\to \frac{1}{\sqrt{2t}} \sum_{y=0}^{t-1} |y\rangle \left(e^{-i\alpha} \mathsf{G}^y |\psi_+\rangle + e^{i\alpha} \mathsf{G}^y |\psi_-\rangle \right) \\
&= \frac{e^{-i\alpha}}{\sqrt{2t}} \sum_{y=0}^{t-1} e^{-2i\alpha y} |y\rangle |\psi_+\rangle + \frac{e^{i\alpha}}{\sqrt{2t}} \sum_{y=0}^{t-1} e^{2i\alpha y} |y\rangle |\psi_-\rangle \\
&= \frac{e^{-i\alpha}}{\sqrt{2}} |\varphi_t(-\tfrac{\alpha}{\pi})\rangle |\psi_+\rangle + \frac{e^{i\alpha}}{\sqrt{2}} |\varphi_t(\tfrac{\alpha}{\pi})\rangle |\psi_-\rangle \ .
\end{aligned}$$

Assume for a moment that $y = \frac{\alpha}{\pi} t$ is an integer. Let us pretend that we measure the second register in the $|\psi_\pm\rangle$ basis and the outcome is $|\psi_-\rangle$. Then the first register is in the state $|\varphi_t(\frac{y}{t})\rangle$ and the inverse of the quantum Fourier transform F_t^{-1} maps it to $|y\rangle$. We know that $w = n \cdot \sin^2 \alpha = n \cdot \sin^2(\pi \frac{y}{t})$, which is exactly what the algorithm computes as $\tilde{w}$ in its last step. If the value of the second register was $|\psi_+\rangle$, then the computation of $\tilde{w}$ gives the same value thanks to $\sin^2(-\alpha) = \sin^2 \alpha$. This means that we do not have to measure the second register at all and the outcome will be right anyway.

In reality it is very unlikely that $y = \frac{\alpha}{\pi} t$ would be an integer, hence it remains to prove that in such a case the inverse of the quantum Fourier transform gives a good approximation $\tilde{y}$ of y. Let $\tilde{\alpha} = \pi \frac{\tilde{y}}{t}$ be the approximation of the true angle α. For any two numbers $\beta_0, \beta_1 \in \mathbb{R}$, let $d(\beta_0, \beta_1) = \min_{z \in \mathbb{Z}} (z + \beta_1 - \beta_0)$; thus $2\pi d(\beta_0, \beta_1)$ is the length of the shortest arc on the unit circle going from $e^{2\pi i \beta_0}$ to $e^{2\pi i \beta_1}$. By the following Lemma 1.5.3, with probability at least $1 - \frac{1}{2(k-1)}$, $d(\alpha, \tilde{\alpha}) \leq \frac{k}{t}\pi$. Using Lemma 1.5.4 with $\varepsilon = \frac{k}{t}\pi$, and $w = n \cdot \sin^2 \alpha$, the result easily follows. □

1.5.3. Lemma ([BHMT02, Theorem 11]). *Let Y be the discrete random variable corresponding to the classical result of measuring $\mathsf{F}_t^{-1} |\varphi_t(\beta)\rangle$. If βt is an integer, then* $\Pr[Y = \beta t] = 1$, *otherwise, letting* $\Delta = d(\beta, \frac{\tilde{y}}{t})$,

- $\Pr[Y = \tilde{y}] \leq \frac{1}{(2t\Delta)^2}$,
- *for any $k > 1$ we also have* $\Pr[\Delta \leq \frac{k}{t}] \geq 1 - \frac{1}{2(k-1)}$,
- *and, in the case $k = 1$ and $t > 2$,* $\Pr[\Delta \leq \frac{1}{t}] \geq \frac{8}{\pi^2}$.

Proof (sketch). First, $\Pr[Y = \tilde{y}] = |\langle \tilde{y} | (\mathsf{F}_t^{-1} |\varphi_t(\beta)\rangle) |^2 = |\langle \varphi_t(\frac{\tilde{y}}{t}) | \varphi_t(\beta) \rangle|^2$ which, after some manipulations, is equal to $\frac{\sin^2(t\Delta\pi)}{M^2 \sin^2(\Delta\pi)} \leq \frac{1}{(2t\Delta)^2}$. The second and the third part follow by summing probabilities from the first point and finding the minimum. □

1.5.4. Lemma ([BHMT02, Lemma 7]). *Let $a = \sin^2 \alpha$ and $\tilde{a} = \sin^2 \tilde{\alpha}$ with $\alpha, \tilde{\alpha} \in [0, 2\pi]$. Then*

$$|\tilde{\alpha} - \alpha| \leq \varepsilon \Longrightarrow |\tilde{a} - a| \leq 2\varepsilon\sqrt{a(1-a)} + \varepsilon^2 \ .$$

Proof. For $\varepsilon \geq 0$, using standard trigonometric identities, we obtain

$$\begin{aligned} \sin^2(\alpha + \varepsilon) - \sin^2 \alpha &= \sqrt{a(1-a)} \sin 2\varepsilon + (1 - 2a) \sin^2 \varepsilon \ , \\ \sin^2 \alpha - \sin^2(\alpha - \varepsilon) &= \sqrt{a(1-a)} \sin 2\varepsilon + (2a - 1) \sin^2 \varepsilon \ . \end{aligned}$$

The inequality follows directly from $\sin \varepsilon \leq \varepsilon$. □

1.5.2 Estimating the Hamming weight of a string

If we want to estimate $|x|$ within a few standard deviations, we can run Quantum Counting with $t = \sqrt{n}$ queries and $k = 1$.

1.5.5. Corollary. *For an n-bit string x,* Quantum Counting $(x, \sqrt{n})$ *outputs an estimate $\tilde{w}$ of $w = |x|$ such that $|\tilde{w} - w| \leq 2\pi\sqrt{\min(w, n-w)} + 10$ with probability at least $8/\pi^2$.*

In several algorithms, one needs to decide whether some string contains at least $t > 0$ ones. This problem is called a *t-threshold function*, denoted by Thr_t. Formally, $\mathrm{Thr}_t(x) = 1$ if and only if $|x| \geq t$. One call to Quantum Counting with $t = O(\sqrt{t(n-t+1)})$ queries gives a correct outcome with constant probability. Beals et al. [BBC+01] showed that this is optimal; we will present a different proof in Corollary 2.4.3. Quantum Counting is not the only way to compute this function. One could repeatedly use Grover search like in Corollary 1.3.5, however Quantum Counting has two significant advantages: the strong upper bound holds for the worst-case running time instead of for the expected running time, and the computation only uses $O(\log n)$ qubits instead of potentially $\Omega(n)$ qubits to remember all ones found so far.

1.5.6. Corollary ([BHMT02]). *For every n, t, there is a quantum algorithm that computes the t-threshold function on an n-bit string using $O(\sqrt{t(n-t+1)})$ queries with constant success probability.*

1.6 Summary

We have explained the basic concepts of quantum mechanics: superposition, evolution, and measurement. We have introduced the model of quantum circuits and defined the quantum query complexity. We have presented one of the two major quantum algorithms Grover search that searches quickly any unordered database, and several its variations. We have analyzed its success probability and running time. We have sketched the model of quantum random walks and outlined two important quantum walk algorithms. Finally, we have presented algorithm QUANTUM COUNTING that approximates the number of solutions.

Chapter 2

Quantum Lower Bounds

This chapter is based on the following survey:

[HŠ05] P. Høyer and R. Špalek. Lower bounds on quantum query complexity. *Bulletin of the European Association for Theoretical Computer Science*, 87:78–103, October 2005.

2.1 Introduction

In this chapter, we offer an introduction to the study of limitations on the power of quantum computers. Can quantum computers really be more powerful than traditional computers? What can quantum computers not do? What proof techniques are used for proving bounds on the computational power of quantum computers? This is a highly active area of research and flourishing with profound and beautiful theorems. Though deep, it is fortunately also an accessible area, based on basic principles and simple concepts, and one that does not require specialized prior knowledge. One aim of this chapter is to show this by providing a fairly complete introduction to the two most successful methods for proving lower bounds on quantum computations, the adversary method and the polynomial method. The text is biased towards the adversary method since it is often used in this thesis and it yields very strong lower bounds. This text can be supplemented by the excellent survey of Buhrman and de Wolf [BW02] on decision tree complexities, published in 2002 in the journal Theoretical Computer Science.

The rest of the chapter is organized as follows. We discuss very basic principles used in proving quantum lower bounds in Section 2.2 and use them to establish our first lower-bound method, the adversary method, in Section 2.3. We discuss how to apply the method in Section 2.4, and its limitations in Section 2.5. We then give an introduction to the second method, the polynomial method, in Section 2.6. We compare the two methods in Section 2.7, discuss some open problems in Section 2.8, and give a few final remarks in Section 2.9.

We demonstrate the methods on a running example, and for this, we use one of the most basic algorithmic questions one may think of: that of searching an ordered set.

2.1.1. Problem. (Ordered search) *In the oracle model, the input to ordered search is an n-bit string $x = x_1 \dots x_n$. We are promised that $x_i \leq x_{i+1}$ for all $1 \leq i < n$ and that $x_n = 1$, and the goal is to find the leftmost 1, that is $f(x)$ is the index $i \in [n]$ for which $x_i = 1$ and no index $j < i$ exists with $x_j = 1$.*

2.1.2. Question. Can one implement ordered search significantly faster on a quantum computer than applying a standard $\Theta(\log n)$ binary search algorithm?

The classical query complexity of ordered search is $\lceil \log_2(n) \rceil$ and is achieved by standard binary search. The quantum query complexity of ordered search is at most $0.45 \log_2 n$, due to the work of high school student M. B. Jacokes in collaboration with Landahl and Brookes [JLB05] (See also [FGGS99, BW98, HNS02]). Using the adversary method, we show that their algorithm is within a factor of about two of being optimal.

2.2 Distinguishing hard inputs

The first quantum lower bound using adversary arguments was given by Bennett, Bernstein, Brassard, and Vazirani in [BBBV97]. They show that any quantum query algorithm can be sensitive to at most quadratically many oracle bits, which implies a lower bound of $\Omega(\sqrt{n})$ for unordered search (Problem 1.3.1) and thus proves that the $O(\sqrt{n})$ algorithm Grover search [Gro96] is optimal. Interestingly, the lower bound of Bennett et al. was proved in 1994, well before Grover defined his search problem. In 2000, Ambainis [Amb02] found an important generalization of the method and coined it *adversary arguments*.

A constructive interpretation of basic adversary arguments is in terms of *distinguishability*. We will thus not be concerned with computing the function f, but merely interested in distinguishing oracles. Consider some algorithm A that computes some function $f : \Sigma^n \to \Sigma'$ in the oracle model, and consider two inputs $x, y \in \Sigma^n$ for which $f(x) \neq f(y)$. Since A computes f, it must in particular be capable of distinguishing between oracle x and oracle y. For a given problem we try to identify *pairs of oracles* that are hard to *distinguish*. If we can identify hard input pairs, we may derive a good lower bound. However, a caveat is that using only the very hardest input pairs does not yield good lower bounds for some problems, and we are thus naturally led to also consider less hard input pairs. A remedy is to use *weights* that capture the hardness of distinguishing each pair of oracles, and to do so, we define a matrix Γ of dimension $2^n \times 2^n$ that takes non-negative real values,

$$\Gamma : \Sigma^n \times \Sigma^n \to \mathbb{R}_0^+ \ . \tag{2.1}$$

We require that Γ is symmetric and that $\Gamma[x, y] = 0$ whenever $f(x) = f(y)$. We say that Γ is a *spectral adversary matrix for f* if it satisfies these two conditions. The symmetry condition on Γ states that we are concerned with distinguishing *between* any two inputs x, y. We are not concerned with distinguishing x *from* y, nor distinguishing y *from* x. We discuss this subtlety further in Section 2.4 below when considering alternative definitions of weighted adversary arguments. The spectral adversary matrix Γ allows us to capture both total and partial functions, as well as non-Boolean functions. Since we are only concerned with distinguishability, once we have specified the entries of Γ, we may safely ignore the underlying function f.

Weighted adversary arguments were first used by Høyer, Neerbek, and Shi in [HNS02] to prove a lower bound of $\Omega(\log n)$ for ordered search and a lower bound $\Omega(n \log n)$ for sorting. Barnum and Saks [BS04] used weighted adversary arguments to prove a lower bound of $\Omega(\sqrt{n})$ for read-once formulae, and introduced the notion Γ that we adapt here. Barnum, Saks, and Szegedy extended their work in [BSS03] and derived a general lower bound on the query complexity of f in terms of spectral properties of matrix Γ. Their lower bound has a very elegant and short formulation, a basic proof, and captures important properties of adversary methods, and we shall thus adapt much of their terminology.

As discussed above, the key to prove a good lower bound is to pick a good adversary matrix Γ. For our running example of ordered search, which is a partial non-Boolean function, we use the following weights.

2.2.1. Example. (Ordered Search) The weight on the pair (x, y) is the inverse of the Hamming distance of x and y,

$$\Gamma^{\text{search}}[x, y] = \begin{cases} \frac{1}{|f(x)-f(y)|} & \text{if } x \text{ and } y \text{ are valid and distinct inputs to } f \\ & (f(x) \text{ is the position of the leftmost 1 in } x) \\ 0 & \text{otherwise} \ , \end{cases}$$

The larger the Hamming distance between x and y, the easier it is to distinguish them, and the smaller weight is assigned to the pair.

We have to choose how to measure distinguishability. The possibly simplest measure is to use inner products. Two quantum states are distinguishable with certainty if and only if they are orthogonal, and they can be distinguished with high probability if and only if their inner product has small absolute value.

2.2.2. Fact. ([BV97]) Suppose we are given one of two known states $|\psi_x\rangle, |\psi_y\rangle$. There exists a measurement that correctly determines which of the two states we are given with error probability at most ϵ if and only if $|\langle\psi_x|\psi_y\rangle| \le \epsilon'$, where $\epsilon' = 2\sqrt{\epsilon(1-\epsilon)}$.

Since a unitary operator is just a change of basis, it does not change the inner product between any two quantum states, and thus the inner product can only change as a consequence of queries to the oracle.

2.3 Adversary lower bounds

Adversary lower bounds are of information theoretical nature. A basic idea in adversary lower bounds is to upper-bound the amount of information that can be learned in a single query. If little information can be learned in any one query, then many queries are required. We use spectral properties of Γ to put an upper bound on the amount of information the algorithm learns about the oracle.

Let A be some quantum algorithm that computes some function f with bounded 2-sided error. For every integer $t \geq 0$ and every oracle x, let

$$|\psi_x^t\rangle = \mathsf{U}_t \mathsf{O}_x \cdots \mathsf{U}_1 \mathsf{O}_x \mathsf{U}_0 |0\rangle$$

denote the quantum state after t queries to the oracle. To measure the progress of the algorithm, we define similarly to [Amb02, HNS02, BS04, BSS03] a weight function

$$W^t = \sum_{x,y} \Gamma[x,y] \delta_x \delta_y \cdot \langle \psi_x^t | \psi_y^t \rangle \enspace ,$$

where δ is a fixed principal eigenvector of Γ, that is a normalized eigenvector corresponding to the largest eigenvalue of Γ, and where δ_x denotes the x^{th} entry of δ.

The algorithm starts in a quantum state $|\psi_x^0\rangle = \mathsf{U}_0|0\rangle$ which is independent of the oracle x, and thus the total initial weight is

$$W^0 = \sum_{x,y} \Gamma[x,y] \delta_x \delta_y = \delta^T \Gamma \delta = \lambda(\Gamma) \enspace ,$$

where δ^T denotes the transpose of δ and $\lambda(\Gamma)$ denotes the spectral norm of Γ. The final state of the algorithm after t queries is $|\psi_x^t\rangle$ if the oracle is x, and it is $|\psi_y^t\rangle$ if the oracle is y. If $f(x) \neq f(y)$, we must have that $|\langle \psi_x^t | \psi_y^t \rangle| \leq \epsilon'$ by Fact 2.2.2, and hence $W^t \leq \epsilon' W^0$. If the total weight can decrease by at most Δ by each query, the algorithm requires $\Omega(\frac{W^0}{\Delta})$ queries to the oracle.

Following Barnum, Saks, and Szegedy [BSS03], we upper-bound Δ by the largest spectral norm of the matrices Γ_i, defined by

$$\Gamma_i[x,y] = \begin{cases} \Gamma[x,y] & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \enspace , \end{cases} \tag{2.2}$$

for each $1 \leq i \leq n$. The theorem of [BSS03] is here stated (and proved) in a slightly more general form than in [BSS03] so that it also applies to non-Boolean functions. Our proof aims at emphasizing distinguishability and differs from the original.

2.3.1. Theorem (Spectral method [BSS03]). *For any partial function $f : S \to \Sigma'$ with domain $S \subseteq \Sigma^n$, and any adversary matrix Γ for f,*

$$Q_2(f) = \Omega\left(\frac{\lambda(\Gamma)}{\max_i \lambda(\Gamma_i)}\right) \enspace .$$

Proof. We prove that the drop in total weight $W^t - W^{t+1}$ by the $t+1$st query is upper-bounded by the largest eigenvalue of the matrices Γ_i.

For each $i \in [n]$ and $b \in \Sigma$, let $\mathsf{P}_{i,b} = \sum_{z \geq 0} |i,b;z\rangle\langle i,b;z|$ denote the projection onto the subspace querying the ith oracle number with multiplicative factor b, and let $\mathsf{P}_i = \sum_{b \in \Sigma} \mathsf{P}_{i,b}$. Let $\beta_{x,i} = \|\mathsf{P}_i|\psi_x^t\rangle\|$ denote the absolute value of the amplitude of querying the ith number in the $t+1$st query, provided the oracle is x. Note that $\sum_{i=1}^n \mathsf{P}_i = \mathsf{I}$ and $\sum_{i=1}^n \beta_{x,i}^2 = 1$ for any oracle x, since the algorithm queries one of the n numbers $x_1, \dots, x_n$. The $t+1$st query changes the inner product by at most the overlap between the projections of the two states onto the subspace that corresponds to indices i on which x_i and y_i differ,

$$
\begin{aligned}
\left|\langle\psi_x^t|\psi_y^t\rangle - \langle\psi_x^{t+1}|\psi_y^{t+1}\rangle\right| &= \left|\langle\psi_x^t|(\mathsf{I} - \mathsf{O}_x^\dagger \mathsf{O}_y)|\psi_y^t\rangle\right| \\
&= \left|\langle\psi_x^t|(\mathsf{I} - \mathsf{O}_x^\dagger \mathsf{O}_y)\Big(\sum_{i\in[n],\, b\in\Sigma} \mathsf{P}_{i,b}\Big)|\psi_y^t\rangle\right| \\
&= \left|\sum_{i\in[n],\, b\in\Sigma} \underbrace{\Big(1 - e^{\frac{2\pi\sqrt{-1}}{|\Sigma|}(y_i - x_i)b}\Big)}_{=\Delta_i[b,b]} \langle\psi_x^t|\mathsf{P}_{i,b}|\psi_y^t\rangle\right| \\
&\leq \sum_{i:x_i\neq y_i} \left|\langle\psi_x^t|\sum_{b\in\Delta} \Delta_i[b,b]\mathsf{P}_{i,b}|\psi_y^t\rangle\right| \qquad \begin{pmatrix}\Delta_i \text{ is diagonal} \\ \lambda(\Delta_i) \leq 2\end{pmatrix} \\
&= \sum_{i:x_i\neq y_i} \left|\langle\psi_x^t|\mathsf{P}_i \cdot \Delta_i \cdot \mathsf{P}_i|\psi_y^t\rangle\right| \leq 2 \sum_{i:x_i\neq y_i} \beta_{x,i}\beta_{y,i} \ . \qquad (2.3)
\end{aligned}
$$

The bigger the amplitudes of querying the numbers i on which x_i and y_i differ, the larger the drop in the inner product can be.

Define an auxiliary $|S|$-dimensional vector $a_i[x] = \delta_x \beta_{x,i}$ and note that

$$
\sum_{i=0}^n \|a_i\|^2 = \sum_{i=0}^n \sum_x \delta_x^2 \beta_{x,i}^2 = \sum_x \delta_x^2 \sum_{i=0}^n \beta_{x,i}^2 = \sum_x \delta_x^2 = 1 \ .
$$

The drop in the total weight is upper-bounded by

$$
\begin{aligned}
\left|W^t - W^{t+1}\right| &= \left|\sum_{x,y} \Gamma[x,y]\delta_x\delta_y \big(\langle\psi_x|\psi_y\rangle - \langle\psi'_x|\psi'_y\rangle\big)\right| \\
&\leq 2 \sum_{x,y} \sum_i \Gamma_i[x,y]\delta_x\delta_y \cdot \beta_{x,i}\beta_{y,i} \\
&= 2 \sum_i a_i^T \Gamma_i a_i \\
&\leq 2 \sum_i \lambda(\Gamma_i)\|a_i\|^2 \\
&\leq 2 \max_i \lambda(\Gamma_i) \cdot \sum_i \|a_i\|^2 \\
&= 2 \max_i \lambda(\Gamma_i) \ .
\end{aligned}
$$

The first inequality bounds the drop in inner product for a specific pair and follows from equation (2.3). The second inequality follows from the spectral norm of Γ. The second and third inequalities state that the best possible query distributes the amplitude of the query according to the largest principal eigenvector of the query matrices Γ_i. □

2.3.2. Example. Returning to our example of ordered search, for $n = 4$, the adversary matrix with respect to the ordered basis $(0001, 0011, 0111, 1111)$ is given by

$$\Gamma^{\text{search}(4)} = \begin{bmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{3} \\ 1 & 0 & 1 & \frac{1}{2} \\ \frac{1}{2} & 1 & 0 & 1 \\ \frac{1}{3} & \frac{1}{2} & 1 & 0 \end{bmatrix} .$$

The spectral norm is easily seen to be lower-bounded by the sum of the entries in the first row, $\lambda(\Gamma^{\text{search}(4)}) \geq 1 + \frac{1}{2} + \frac{1}{3}$. In general, $\lambda(\Gamma^{\text{search}})$ is lower-bounded by the *harmonic number* $H_{n-1} = \sum_{k=1}^{n-1} \frac{1}{k}$, which is at least $\ln(n)$. By a simple calculation, the spectral norm of the query matrices $\lambda(\Gamma_i^{\text{search}})$ is maximized when $i = \lfloor n/2 \rfloor$, in which case it is upper-bounded by the spectral norm of the infinite *Hilbert matrix* $[1/(r+s-1)]_{r,s\geq 1}$, which is π. We thus reprove the lower bound of $(1-\epsilon')\frac{\ln(n)}{\pi}$ for ordered search given in [HNS02].

2.4 Applying the spectral method

The spectral method is very appealing in that it has a simple formulation, a basic proof, and gives good lower bounds for many problems. However, estimating the spectral norm of a matrix may be difficult in general, so let us present a few variants of the spectral method that are easier to apply.

The first general quantum lower bound using adversary arguments was introduced by Ambainis in [Amb02]. It can be derived from the spectral method by applying simple bounds on the spectral norm of Γ and each Γ_i. By definition, the numerator $\lambda(\Gamma)$ is lower-bounded by $\frac{1}{\|d\|^2} d^T \Gamma d$ for any non-negative vector d, and by the following lemma by Mathias, the denominator $\lambda(\Gamma_i)$ is upper-bounded by the product of a row-norm and a column-norm.

2.4.1. Lemma ([Mat90]). *Let $M, N \geq 0$ be non-negative rectangular matrices of the same dimension, and let $G = M \circ N$ be the entry-wise product of M and N. Let $\lambda(G) = \max_{v,w} \frac{v^T G w}{\|v\| \cdot \|w\|}$ denote the spectral norm of G. Then*

$$\lambda(G) \leq \max_{x,y} r_x(M)\, c_y(N) \ ,$$

where $r_x(M)$ is the ℓ_2-norm of the x^{th} row in M, and $c_y(N)$ is the ℓ_2-norm of the y^{th} column in N.

Proof. Let v, w be a pair of singular vectors of G corresponding to the principal singular value, and let $\|v\| = \|w\| = 1$. Then $\lambda(G) = v^T G w$. Using $G[x,y] = M[x,y] \cdot N[x,y]$ and one Cauchy-Schwarz inequality,

$$\begin{aligned}
\lambda(G) = v^T G w &= \sum_{x,y} G[x,y] v_x w_y \\
&= \sum_{x,y} M[x,y] v_x \cdot N[x,y] w_y \\
&\le \sqrt{\sum_{x,y} M[x,y]^2 v_x^2} \cdot \sqrt{\sum_{x,y} N[x,y]^2 w_y^2} \\
&= \sqrt{\sum_x r_x(M)^2 v_x^2} \cdot \sqrt{\sum_y c_y(N)^2 w_y^2} \\
&\le \sqrt{\max_x r_x(M)^2 \sum_x v_x^2} \cdot \sqrt{\max_y c_y(N)^2 \sum_y w_y^2} \\
&= \max_{x,y} r_x(M) c_y(N) \ ,
\end{aligned}$$

which we had to prove. □

Applying these two bounds, we can obtain Ambainis's lower bound in [Amb02] as follows. We refer to the method as an *unweighted adversary method* since it considers only two types of inputs: easy inputs and hard inputs. We construct an adversary matrix Γ that corresponds to a uniform distribution over the hard input pairs.

2.4.2. Theorem (Unweighted method [Amb02]). *Let $f : S \to \{0,1\}$ be a partial Boolean function with domain $S \subseteq \{0,1\}^n$, and let $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$ be subsets of (hard) inputs. Let $R \subseteq A \times B$ be a relation, and set $R_i = \{(x,y) \in R : x_i \neq y_i\}$ for each $1 \le i \le n$. Let m, m' denote the minimal number of ones in any row and any column in relation R, respectively, and let ℓ, ℓ' denote the maximal number of ones in any row and any column in any of the relations R_i, respectively. Then $Q_2(f) = \Omega(\sqrt{mm'/\ell\ell'})$.*

Proof. Since f is Boolean, any adversary matrix can by written as $\Gamma = \left(\begin{smallmatrix} 0 & G \\ G^T & 0 \end{smallmatrix}\right)$ for some rectangular matrix $G : A \times B \to \mathbb{R}^+$, and $\lambda(\Gamma) = \lambda(G)$. Define column vectors v, w as $v_x = r_x(R)$ and $w_y = c_y(R)$, and an adversary matrix G as $G[x,y] = \frac{1}{v_x w_y}$ if $(x,y) \in R$, and 0 otherwise. Then $\lambda(G) \ge \frac{v^T G w}{\|v\| \cdot \|w\|} = \frac{|R|}{|R|} = 1$. For each G_i, we apply Lemma 2.4.1 with $M_i[x,y] = \frac{1}{v_x}$ and $N_i[x,y] = \frac{1}{w_y}$ if $(x,y) \in R_i$, and they are both 0 otherwise. Then $r_x(M_i) \le \sqrt{\ell / v_x^2} \le \sqrt{\ell/m}$ and $c_y(N_i) \le \sqrt{\ell'/w_y^2} \le \sqrt{\ell'/m'}$. Hence, by Lemma 2.4.1, $\lambda(G_i) \le \max_{x,y} r_x(M) c_y(N) \le \sqrt{\ell\ell'/mm'}$, and the result follows from Theorem 2.3.1. □

2.4.3. Corollary. *The threshold function* Thr_t*, defined by* $\mathrm{Thr}_t(x) = 1$ *if and only if* $|x| \geq t$*, has quantum query complexity* $\Omega(\sqrt{t(n-t+1)})$*.*

Proof. This result was first proven using the polynomial method in [BBC+01]; here we reprove it using the adversary method. Let $A = \{x : |x| = t-1\}$, $B = \{y : |x| = t\}$, and $R = \{(x,y) : x \in A, y \in B, |y-x| = 1\}$. Then every zero-input x is connected with $n-t+1$ one-inputs y, and every one-input y is connected with t zero-inputs x. On the other hand, if we set a restriction that the two inputs must differ in some fixed i^{th} bit, then the number of connections of any input is either 0 or 1. By Theorem 2.4.2, $Q_2(\mathrm{Thr}_t) = \Omega(\sqrt{(n-t+1)t/(1 \cdot 1)}) = \Omega(\sqrt{t(n-t+1)})$. □

The unweighted adversary method is very simple to apply as it requires only to specify a set R of hard input pairs. It gives tight lower bounds for many computational problems, including inverting a permutation [Amb02], computing any symmetric function and counting (lower bound first proven by the polynomial method in [NW99, BBC+01], upper bound in [BHMT02]), constant-level AND-OR trees (lower bound in [Amb02], upper bounds in [BCW98, HMW03]), and various graph problems [DHHM04]. For some computational problems, the hardness does however not necessarily rely only on a few selected hard instances, but rather on more global properties of the inputs. Applying the unweighted method on ordered search would for instance only yield a constant lower bound. In these cases, we may apply the following weighted variant of the method, due to Ambainis [Amb03] and Zhang [Zha05].

2.4.4. Theorem (Weighted method [Amb03, Zha05]). *Let* $S \subseteq \Sigma^n$ *and let* $f : S \to \Sigma'$ *be a function. Let* w, w' *denote a weight scheme as follows:*

- *Every pair* $(x,y) \in S^2$ *is assigned a non-negative weight* $w(x,y) = w(y,x)$ *that satisfies* $w(x,y) = 0$ *whenever* $f(x) = f(y)$*.*
- *Every triple* $(x,y,i) \in S^2 \times [n]$ *is assigned a non-negative weight* $w'(x,y,i)$ *that satisfies* $w'(x,y,i) = 0$ *whenever* $x_i = y_i$ *or* $f(x) = f(y)$*, and the inequality* $w'(x,y,i)w'(y,x,i) \geq w^2(x,y)$ *for all* x,y,i *with* $x_i \neq y_i$*.*

Then

$$Q_2(f) = \Omega\left(\min_{\substack{x,y,i\\ w(x,y)>0\\ x_i \neq y_i}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,i)}}\right),$$

where $wt(x) = \sum_y w(x,y)$ *and* $v(x,i) = \sum_y w'(x,y,i)$ *for all* $x \in S$ *and* $i \in [n]$*.*

At first glance, the weighted method may look rather complicated, both in its formulation and use, though it is not. We first assign weights to pairs (x,y) of inputs for which $f(x) \neq f(y)$, as in the spectral method. We require the weights to be symmetric so that they represent the difficulty in distinguishing *between x and y.*

We then afterwards assign weights $w'(x, y, i)$ that represent the difficulty in distinguishing *x from y by querying index i.* The harder it is to distinguish x from y by index i, compared to distinguishing y from x by index i, the more weight we put on (x, y, i) and the less on (y, x, i), and vice versa.

To quantify this, define $t(x, y, i) = w'(x, y, i)/w'(y, x, i)$. Then $t(x, y, i)$ represents the relative amount of information we learn about input pairs (x, z) compared to the amount of information we learn about input pairs (u, y) for an average u, z, by querying index i. If we, by querying index i, learn little about x compared to y, we let $t(x, y, i)$ be large, and otherwise small. Suppose we query an index i for which $x_i \neq y_i$. Then we learn whether the oracle is x or y. However, at the same time, we also learn whether the oracle is x or z for any other pair (x, z) for which $x_i \neq z_i$ and $f(x) \neq f(z)$; and similarly, we learn whether the oracle is u or y for any other pair (u, y) for which $u_i \neq y_i$ and $f(u) \neq f(y)$. The less information querying index i provides about pairs (x, z) compared to pairs (u, y), the larger we choose $t(x, y, i)$. Having thus chosen $t(x, y, i)$, we set $w'(x, y, i) = w(x, y)\sqrt{t(x, y, i)}$ and $w'(y, x, i) = w(x, y)/\sqrt{t(x, y, i)}$.

We show next that the weighted method yields a lower bound of $\Omega(\log n)$ for the ordered search problem. This proves that the weighted method is strictly stronger than the unweighted method. We show in Section 5.3 that the weighted method is equivalent to the spectral method. The weighted method yields strong (though not necessarily tight) lower bounds for read-once formula [BS04] and iterated functions [Amb03]. Aaronson [Aar04b], Santha and Szegedy [SS04], and Zhang [Zha06] use adversary arguments to prove lower bounds for local search.

2.4.5. Example. To apply the weighted method on ordered search, we pick weights $w(x, y) = \Gamma^{\text{search}}[x, y]\, \delta_x \delta_y$; they are exactly the coefficients of the weight function W^t for the input pair (x, y). Now, consider $t(x, y, i)$ with $f(x) \leq i < f(y)$ so that $x_i \neq y_i$. By querying index i, we also distinguish between x and z for each of the $f(y) - i$ inputs z with $i < f(z) \leq f(y)$, and we learn to distinguish between u and y for each of the $i - f(x) + 1$ inputs u with $f(x) \leq f(u) \leq i$. We thus choose to set

$$t(x, y, i) = \frac{|f(x) - i| + 1}{|f(y) - i| + 1} \ .$$

Plugging these values into the weighted method yields a lower bound of $\Omega(\log n)$ for ordered search.

2.5 Limitations of the spectral method

The spectral method and the weighted adversary method bound the amount of information that can be learned in any one query. They do not take into account that the amount of information that can be learned in the j^{th} query might differ from the amount of information that can be learned in the k^{th} query.

In 1999, Zalka [Zal99] successfully managed to capture the amount of information that can be learned in each individual query for a restricted version of unordered search. In this restricted version, we are promised that the input x is either the zero-string (so $|x| = 0$) or exactly one entry in x is one (so $|x| = 1$), and the goal is to determine which is the case. By symmetry considerations, Zalka demonstrates that Grover search saturates some improved inequalities (which are similar to equation (2.3)) and hence is optimal, even to within an additive constant.

Since current adversary methods do not capture the amount of information the algorithm currently knows, we may simply assume that the algorithm already knows every number of the oracle and that it tries to prove so. This motivates a study of the relationship between the best bound achievable by the spectral method and the certificate complexity.

2.5.1. Definition. For any function f, the *adversary bound for f* is

$$\mathrm{Adv}(f) = \max_{\Gamma} \frac{\lambda(\Gamma)}{\max_i \lambda(\Gamma_i)} \enspace ,$$

where Γ ranges over all adversary matrices for f, that is Γ is non-negative and symmetric, and $\Gamma[x,y] = 0$ whenever $f(x) = f(y)$.

2.5.2. Definition. A *certificate* for an input $x \in \Sigma^n$, is a subset $C \subseteq [n]$ of input indices such that for any other input y in the domain of f that may be obtained from x by flipping some of the indices not in C, we have that $f(x) = f(y)$. The certificate complexity $\mathrm{C}_x(f)$ of input x is the size of a smallest certificate for x. The *certificate complexity* $\mathrm{C}(f)$ of a function f is the maximum certificate complexity among its inputs. We also define the z-certificate complexity $\mathrm{C}_z(f)$ when taking the maximum only over inputs that map to z.

The spectral theorem can never yield a lower bound better than a quantity that can be expressed in terms of certificate complexity.

2.5.3. Theorem ([LM04, Zha05, ŠS06], Section 5.4). *Let $f : S \to \{0,1\}$ be any Boolean function. If f is partial, that is if $S \subsetneq \{0,1\}^n$, then the adversary bound* $\mathrm{Adv}(f)$ *is at most* $\min\left\{\sqrt{\mathrm{C}_0(f)n}, \sqrt{\mathrm{C}_1(f)n}\right\}$. *If f is total, that is if $S = \{0,1\}^n$, then the bound is limited by $\sqrt{\mathrm{C}_0(f)\mathrm{C}_1(f)}$.*

The spectral adversary method is also not suitable for proving lower bounds for problems related to *property testing*. If function $f : S \to \Sigma'$ is a partial function with domain $S \subseteq \Sigma^n$ such that every pair of inputs that evaluate to different function values has Hamming distance at least εn, then the spectral theorem does not yield a lower bound better than $1/\varepsilon$ [LLS06].

The certificate complexity of a function $f : \{0,1\}^n \to \Sigma'$ is itself polynomially related to the block sensitivity of the function.

2.5.4. DEFINITION. An input $x \in \{0,1\}^n$ is *sensitive* to a block $B \subseteq [n]$ if $f(x) \neq f(x^B)$, where x^B denotes the input obtained by flipping the bits in x with indices from B. The block sensitivity $bs_x(f)$ of input x is the maximum number of disjoint blocks $B_1, B_2, \dots, B_k \subseteq [n]$ on which x is sensitive. The *block sensitivity* $bs(f)$ of f is the maximum block sensitivity of any of its inputs. We also define the z-block sensitivity $bs_z(f)$ when taking the maximum only over inputs that map to z.

For any Boolean function $f : \{0,1\}^n \to \{0,1\}$, the certificate complexity is upper-bounded by $\mathrm{C}(f) \leq bs_0(f) bs_1(f)$, and so is the spectral adversary bound. Conversely, $\mathrm{Adv}(f) \geq \sqrt{bs(f)}$ by a zero-one valued adversary matrix Γ, as follows: Let $x' \in \{0,1\}^n$ be an input that achieves the block sensitivity of f, and let $B_1, B_2, \dots, B_k \subseteq [n]$ be disjoint blocks on which x' is sensitive, where $k = bs(f)$. Set $\Gamma(f)[x, x^B] = 1$ if and only if $x = x'$ and B is one of the k blocks B_i, and then close Γ under transposition. Then $\lambda(\Gamma) = \sqrt{k}$ and $\max_i \lambda(\Gamma_i) = 1$, and thus

$$\sqrt{bs(f)} \leq \mathrm{Adv}(f) \leq bs_0(f) bs_1(f) \ .$$

A useful property of the adversary method is that it composes for Boolean functions. Consider a function on kn bits of the form $h = f \circ (g, \dots, g)$, where $f : \{0,1\}^k \to \{0,1\}$ and $g : \{0,1\}^n \to \{0,1\}$ are Boolean functions. A composition theorem states the complexity of function h in terms of the complexities of f and g. Barnum and Saks [BS04] use composition properties to prove a query lower bound of $\Omega(\sqrt{n})$ for any read-once formula, Ambainis [Amb03] proves a composition lower bound for iterated Boolean functions, and Laplante, Lee, and Szegedy [LLS06] prove a matching upper bound.

2.5.5. THEOREM (COMPOSITION THEOREM [AMB03, LLS06]). *For every pair of Boolean functions $f : \{0,1\}^k \to \{0,1\}$ and $g : \{0,1\}^n \to \{0,1\}$,*

$$\mathrm{Adv}(f \circ (g, \dots, g)) = \mathrm{Adv}(f) \cdot \mathrm{Adv}(g) \ .$$

2.6 Polynomial lower bounds

There are essentially two different methods known for proving lower bounds on quantum computations. The historically first method is the adversary method we discuss above. It was introduced in 1994 by Bennett, Bernstein, Brassard, and Vazirani, and published in 1997 in the SIAM Journal on Computing, in a special section that contains some of the most outstanding papers on quantum computing. The second method was introduced shortly after, in 1998, by Beals, Buhrman, Cleve, Mosca, and de Wolf [BBC$^+$01], and implicitly used by Fortnow and Rogers in [FR99]. Their approach is algebraic and follows earlier very successful work on classical lower bounds via polynomials (see for instance Nisan and Szegedy [NS94] on real polynomials, and Beigel's 1993 survey [Bei93] and Regan's 1997 survey [Reg97] on polynomials modulo 2). We first establish that any partial Boolean function can be represented by a real-valued polynomial.

2.6.1. Definition. Let $f : S \to \{0,1\}$ be a partial Boolean function with domain $S \subseteq \{0,1\}^n$, and let p be a real-valued n-variable polynomial $p : \mathbb{R}^n \to \mathbb{R}$ such that $0 \leq p(x) \leq 1$ for all $x \in \{0,1\}^n$. We say that p *represents* f if $p(x) = f(x)$ for all $x \in S$, and p *approximates* f if $|p(x) - f(x)| \leq \frac{1}{3}$ for all $x \in S$. The *degree* of f, denoted $\deg(f)$, is the minimal degree of a polynomial representing f. The *approximate degree* of f, denoted $\widetilde{\deg}(f)$, is the minimal degree of a polynomial approximating f.

The crux in [BBC$^+$01] is in showing that any quantum algorithm A computing some function f gives rise to some polynomial p_{A} that represents or approximates f.

2.6.2. Theorem ([BBC$^+$01]). *Let A be a quantum algorithm that computes a partial Boolean function $f : S \to \{0,1\}$ with domain $S \subseteq \{0,1\}^n$, using at most t queries to the oracle O'_x. Then there exists an n-variate real-valued multilinear polynomial $p_{\mathsf{A}} : \mathbb{R}^n \to \mathbb{R}$ of degree at most $2t$, which equals the acceptance probability of A.*

Proof. In this theorem, we use the oracle O'_x which is equivalent to the oracle O_x, since it allows for simple formulations. We first rewrite the action of O'_x as

$$\mathsf{O}'_x|i,b;z\rangle = (1-x_i)|i,b;z\rangle + x_i|i,b\oplus 1;z\rangle \ , \tag{2.4}$$

where we define $x_i = 0$ for $i = 0$ so that we can simulate a non-query by querying x_i with $i = 0$. Suppose we apply O'_x on some superposition $\sum_{i,b,z} \alpha_{i,b,z}|i,b;z\rangle$ where each amplitude $\alpha_{i,b,z}$ is an n-variate complex-valued polynomial in x of degree at most j. Then, by equation (2.4), the resulting state $\sum_{i,b,z} \beta_{i,b,z}|i,b;z\rangle$ is a superposition where each amplitude $\beta_{i,b,z}$ is an n-variate complex-valued polynomial in x of degree at most $j+1$. By induction, after t queries, each amplitude can be expressed as a complex-valued polynomial in x of degree at most t. The probability that the final measurement yields the outcome 1, corresponding to accepting the input, is obtained by summing the absolute values of some of the amplitudes squared. The square of any of the absolute amplitudes can be expressed as a real-valued polynomial p_{A} in x of degree at most $2t$. The theorem follows. □

The above theorem states that to any quantum algorithm A computing a Boolean function $f : S \to \{0,1\}$ with domain $S \subseteq \{0,1\}^n$, we can associate an

n-variate polynomial $p_{\mathsf{A}} : \mathbb{R}^n \to \mathbb{R}$ that expresses the acceptance probability of the algorithm on any given input. If algorithm A is exact, that is if A always stops and outputs the correct answer, then $p_{\mathsf{A}}(x) = f(x)$ for all $x \in S$, and thus p_{A} represents f. If A has bounded error, then $0 \leq p_{\mathsf{A}}(x) \leq 1/3$ if $f(x) = 0$ and $2/3 \leq p_{\mathsf{A}}(x) \leq 1$ if $f(x) = 1$, and thus p_{A} approximates f. Furthermore, since the acceptance probability of any algorithm on any input is between 0 and 1, it holds that $0 \leq p(x) \leq 1$ for every $x \in \{0,1\}^n$. The degree of p_{A} is at most twice the number of queries used by algorithm A. Consequently, the degree of a function is a lower bound on the quantum query complexity, up to a factor of two.

2.6.3. Corollary (Polynomial method [BBC$^+$01]). *Let f be any Boolean function. Then $Q_E(f) \geq \deg(f)/2$ and $Q_2(f) \geq \widetilde{\deg}(f)/2$.*

2.7 Applying the polynomial method

The challenge in applying the polynomial method lies in the dimensionality of the input. Typically, the method is applied by first identifying a univariate or bivariate polynomial that captures essential properties of the problem, and then proving a lower bound on the degree of that polynomial. The second part is typically reasonably straightforward since polynomials have been studied for centuries and much is known about their degrees. The possibly simplest nontrivial example is when f is the threshold function Thr_t defined by $\mathrm{Thr}_t(x) = 1$ if and only if $|x| \geq t$. It is easy to see that $\deg(\mathrm{Thr}_t) = \Theta(n)$ for all nontrivial threshold functions, and thus $Q_E(\mathrm{Thr}_t) = \Omega(n)$. Paturi [Pat92] shows that $\widetilde{\deg}(\mathrm{Thr}_t) = \Theta\big(\sqrt{t(n-t+1)}\big)$, and we thus readily get that $Q_2(\mathrm{Thr}_t) = \Omega\big(\sqrt{t(n-t+1)}\big)$, which is tight by Quantum Counting [BHMT02, BBC$^+$01]. This degree argument extends to any symmetric function f by writing f as a sum of threshold functions. The same tight lower bounds for symmetric functions can also be obtained by the unweighted adversary method (see Corollary 2.4.3).

For general non-symmetric functions, the polynomial method is, however, significantly harder to apply. For problems that are "close" to being symmetric, we can sometimes succeed in constructing a univariate or bivariate polynomial that yields a non-trivial lower bound. The first and, in our view, most important such a result was obtained by Aaronson in [Aar02] in which he proves a lower bound of $\Omega(n^{1/5})$ on any bounded-error quantum algorithm for the collision problem.

The collision problem is a non-Boolean promise problem. The oracle is an n-tuple of positive integers between 1 and m, which we think of as a function $X : [n] \to [m]$. We model the oracle O''_X so that a query to the i^{th} entry of the oracle returns the integer $X(i)$. Specifically, O''_X takes as input $|i, r; z\rangle$ and outputs $|i, r \oplus X(i); z\rangle$ where $0 \leq r < 2^k$ for $k = \lceil \log_2(m+1) \rceil$, and $r \oplus X(i)$ denotes bitwise addition modulo 2. We are promised that either X is a one-to-one function, or X is two-to-one, and the goal is to determine which is the case.

The result of Aaronson was soon improved by Shi [Shi02] to $\Omega(n^{1/4})$ for general functions $X : [n] \to [m]$, and to $\Omega(n^{1/3})$ in the case the range is larger than the domain by a constant factor, $m \geq \frac{3}{2}n$. The lower bound of Aaronson and Shi appears as a joint article [AS04]. Finally, Kutin [Kut05] and Ambainis [Amb05b] independently found remedies for the technical limitations in Shi's proof, yielding an $\Omega(n^{1/3})$ lower bound for all functions, which is tight by an algorithm that uses Grover search on subsets by Brassard, Høyer, and Tapp [BHT97].

The best lower bound for the collision problem that can be obtained using the adversary method is only a constant, since any one-to-one function is of large Hamming distance to any two-to-one function. Koiran, Nesme, and Portier [KNP05] use the polynomial method to prove a lower bound of $\Omega(\log n)$ for Simon's problem [Sim97], which is tight [Sim97, BH97]. Simon's problem is a partial Boolean function having properties related to finite Abelian groups. Also for this problem, the best lower bound that can be obtained using the adversary method is a constant.

In contrast, for any *total* Boolean function $f : \{0,1\}^n \to \{0,1\}$, the adversary and polynomial method are both polynomially related to block sensitivity,

$$\sqrt{bs(f)/6} \leq \widetilde{\deg}(f) \leq \deg(f) \leq bs^3(f)$$
$$\sqrt{bs(f)} \leq \mathrm{Adv}(f) \leq bs^2(f) \ .$$

Beals et al. [BBC$^+$01] showed that $\deg(f) \leq bs^3(f)$, and it follows from Nisan and Szegedy [NS94] that $6\widetilde{\deg}(f)^2 \geq bs(f)$. Buhrman and de Wolf [BW02] provide an excellent survey of these and other complexity measures of Boolean functions.

The polynomial lower bound is known to be inferior to the weighted adversary method for some total Boolean functions. In [Amb03], Ambainis gives a Boolean function $f : \{0,1\}^4 \to \{0,1\}$ on four bits, which can be described as "the four input bits are sorted" [LLS06], for which $\deg(f) = 2$ and for which there exists an adversary matrix Γ^f satisfying $\lambda(\Gamma^f)/\max_i \lambda(\Gamma^f_i) = 2.5$. We compose the function with itself and obtain a Boolean function $f_2 = f \circ (f,f,f,f) : \{0,1\}^{16} \to \{0,1\}$ defined on 16 bits for which $\deg(f_2) = 4$, and for which $\lambda(\Gamma^{f_2})/\max_i \lambda(\Gamma^{f_2}_i) = 2.5^2$, by Theorem 2.5.5. Iterating d times, yields a function f_d on $n = 4^d$ bits of degree $\deg(f_d) = 2^d$, with adversary lower bound $2.5^d = \deg(f_d)^{1.32\ldots}$, by the composition theorem. Thus the constructed function f_d is an example of an iterated function of low degree and high quantum query complexity. It is the currently biggest known gap between the polynomial method and the adversary method for a total function. Another iterated total function for which the adversary methods yield a lower bound better than the degree, is the function described as "all three input bits are equal" [Amb03].

The polynomial method is very suitable when considering quantum algorithms computing functions with error ϵ that is sub-constant, whereas the adversary method is not formulated so as to capture such a fine-grained analysis. Buhrman, Cleve, de Wolf, and Zalka [BCWZ99] show that any quantum algorithm for unordered search that succeeds in finding an index i for which $x_i = 1$ with probability at least $1 - \epsilon$, provided one exists, requires $\Omega(\sqrt{n \log(1/\epsilon)})$ queries to the oracle, and show that this is tight. A possibly more familiar example is the following one.

2.7.1. Example. Any polynomial approximating the parity function with any positive bias $\epsilon > 0$ (as opposed to bias $\frac{1}{6}$ where $\frac{1}{6} = \frac{2}{3} - \frac{1}{2}$) has degree n, since any such polynomial gives rise to a univariate polynomial of no larger degree with n roots. Hence, any quantum algorithm computing the parity function with arbitrary small bias $\epsilon > 0$ requires $n/2$ queries to the oracle [BBC+01, FGGS98], which is tight.

A useful property of representing polynomials is that they compose. If p is a polynomial representing a function f, and polynomials $q_1, q_2, \ldots, q_k$ represent functions $g_1, \ldots, g_k$, then $p \circ (q_1, \ldots, q_k)$ represents $f \circ (g_1, \ldots, g_k)$, when well-defined. This composition property does not hold for approximating polynomials: if each sub-polynomial q_i takes the value 0.8, say, then we cannot say much about the value $p(0.8, \ldots, 0.8)$ since the value of p on non-integral inputs is not restricted by the definition of being an approximating polynomial. To achieve composition properties, we require that the polynomials are insensitive to small variations of the input bits. Buhrman, Newman, Röhrig, and de Wolf give in [BNRW05] a definition of such polynomials, and refer to them as being robust.

2.7.2. Definition. (Robust polynomials [BNRW05]) An approximate n-variate polynomial p is *robust* on $S \subseteq \{0,1\}^n$ if $|p(y) - p(x)| \leq \frac{1}{3}$ for every $x \in S$ and $y \in \mathbb{R}^n$ such that $|y_i - x_i| \leq \frac{1}{3}$ for every $i = 1, \ldots, n$. The *robust degree* of a Boolean function $f : S \to \{0,1\}$, denoted $\mathrm{rdeg}(f)$, is the minimal degree of a robust polynomial approximating f.

Robust polynomials compose by definition. Buhrman et al. [BNRW05] show that the robust degree of any total function $f : \{0,1\}^n \to \{0,1\}$ is $O(n)$ by giving a classical algorithm that uses a quantum subroutine for unordered search [Gro96] which is tolerant to errors, due to Høyer, Mosca, and de Wolf [HMW03]. Buhrman et al. [BNRW05] also show that $\mathrm{rdeg}(f) \in O(\widetilde{\deg}(f) \log \widetilde{\deg}(f))$ by giving a construction for turning any approximating polynomial into a robust polynomial at the cost of at most a logarithmic factor in the degree of f. This implies that for any composite function $h = f \circ (g, \ldots, g)$, we have $\widetilde{\deg}(h) \in O(\widetilde{\deg}(f)\widetilde{\deg}(g) \log \widetilde{\deg}(f))$. It is not known whether this is tight. Neither is it known if the approximate degree of h can be significantly smaller than the product of the approximate degrees of f and g. The only known lower bound on the approximate degree of h is the trivial bound $\Omega(\widetilde{\deg}(f) + \widetilde{\deg}(g))$.

An AND-OR tree of depth two is the composed function $f \circ (g, \ldots, g)$ in which the outer function f is the logical AND of $\sqrt{n}$ bits, and the inner function g is the logical OR of $\sqrt{n}$ bits. By the unweighted adversary method, computing AND-OR trees of depth two requires $\Omega(\sqrt{n})$ queries. Høyer, Mosca, and de Wolf [HMW03] give a bounded-error quantum algorithm that uses $O(\sqrt{n})$ queries, which thus is tight. The existence of that algorithm implies that there exists an approximating polynomial for AND-OR tree of depth two of degree $O(\sqrt{n})$. No other characterization of an approximating polynomial for AND-OR trees of depth two of degree $O(\sqrt{n})$ is currently known. The best known lower bound on the approximate degree of AND-OR trees of depth two is $\Omega(n^{1/3})$, up to logarithmic factors in n, by a reduction [Amb05b] from the element distinctness problem on $\sqrt{n}$ integers [AS04].

2.8 Challenges

There is a range of problems for which we do not currently know tight quantum query bounds. One important example is binary AND-OR trees of logarithmic depth. A binary AND-OR tree on $n = 4^d$ variables is obtained by iterating the function $f(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ in total d times. The classical query complexity for probabilistic algorithms is $\Theta(n^{0.753})$ (see [Sni85] for a zero-error algorithm, [SW86] for a matching lower bound, and [San95] for a similar lower bound for all bounded-error algorithms). No better bounded-error quantum algorithm is known. The best known lower bound on the quantum query complexity is $\Omega(\sqrt{n})$ by embedding the parity function on $\sqrt{n}$ bits and noting that the parity function has linear query complexity, which can be shown by either method.

In Chapter 4, we present a quantum algorithm for verifying whether a product of two $n \times n$ matrices is equal to a third one that runs in time $O(n^{5/3})$. The best known lower bound is $\Omega(n^{3/2})$ by the unweighted adversary method, and we conjecture that it is not tight. Magniez, Santha, and Szegedy give in [MSS05] a quantum algorithm for determining if a graph on n vertices contains a triangle which uses $O(n^{1.3})$ queries to the adjacency matrix. The best known lower bound is $\Omega(n)$ by the unweighted adversary method, and has also been conjectured not to be tight [Amb03].

None of the adversary lower bounds above can be strengthened due to the limitation by the certificate complexity. The polynomial method might give a better bound, however coming up with such a bound is difficult due to the high dimensionality of the problem.

2.9 Summary

We have been focusing on two methods for proving lower bounds on quantum query complexity: the adversary method and the polynomial method. Adversary lower bounds are in general easy to compute, but are limited by the certificate complexity. Known lower bounds are constructed by identifying hard input pairs, finding weights accordingly, and computing either the spectral norm of some matrices, or applying the weighted method. Polynomial lower bounds may yield stronger bounds, but are often hard to prove. Known lower bounds by the polynomial methods are constructed by identifying symmetries within the problem, reducing the number of input variables to one or two, and proving a lower bound on the degree of the reduced polynomial.

Barnum, Saks, and Szegedy give in [BSS03] a third lower-bound method that exactly characterizes the quantum query complexity in terms of a semidefinite program, but this strength turns out also to be its weakness: it is very hard to apply and every known lower bound obtained by the method can also be shown by one of the other two methods. A significant body of work has been conducted on lower bounds on communication complexity. We refer to de Wolf's excellent survey [Wol02] as a possible starting point.

Part I

Algorithms

Chapter 3
Matching and Network Flows

This chapter is based on the following paper:

[AŠ06] A. Ambainis and R. Špalek. Quantum algorithms for matching and network flows. In *Proceedings of 23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 172–183, 2006. Lecture Notes in Computer Science 3884.

3.1 Introduction

Matching Finding a matching in a graph is the following combinatorial optimization problem. We are given an undirected graph. If the graph is bipartite, one can think of its vertices as sets of boys on the left and girls on the right, and edges denote whether they like each other. A *matching* is a set of edges such that each vertex is connected to at most one of its neighbors, called a *mate*. The task is to find a matching of maximal size. A matching is *perfect* if each vertex has a mate.

Let n denote the number of vertices and let m denote the number of edges. The simplest classical algorithm based on *augmenting paths* runs in time $O(n^3)$ [Edm65, Gab76]. If the graph is bipartite, then the same simple algorithm finds a maximal matching in faster time $O(n^{5/2})$ [HK73]. Finding a bipartite matching can be reduced to finding a maximal flow in a directed unit network, and one can achieve running time $O(\min(n^{2/3}m, m^{3/2}))$ [ET75] using algorithms presented further in this section. The fastest known algorithm for general sparse graphs by Micali and Vazirani [MV80] runs in time $O(\sqrt{n}m)$. Recently, Mucha and Sankowski published a new randomized algorithm [MS04] based on matrix multiplication that finds a maximal matching in general graphs in time $O(n^\omega)$, where $2 \le \omega \le 2.38$ is the exponent of the best matrix multiplication algorithm.

Network flows Network flows is one of the most studied problems in computer science. We are given a directed graph with two designated vertices: a *source* and a *sink*. Each edge is assigned a *capacity*. One can think of the graph as a pipe network where we want to push as much water from the source to the sink as possible. A *network flow* is an assignment of flows to the edges such that the flow going through each edge is less than or equal to the capacity of the edge, and the flow is conserved in each vertex, that is the total incoming and outgoing flow are equal for each vertex except for the source and the sink. The *size* of a flow is the total flow leaving the source. The task is to find a flow of maximal size.

After the pioneering work of Ford and Fulkerson [FF56], many algorithms were proposed. For networks with real capacities, the fastest algorithms run in time $O(n^3)$ [Kar74, MKM78]. They are based on augmenting the flow by *blocking flows* in *layered residual networks* [Din70]. Each augmentation takes time $O(n^2)$ and the number of iterations is at most n, because the depth of the residual network is increased in each step. If the network is sparse, one can achieve faster time $O(nm(\log n)^2)$ [GN79].

If all capacities are integers bounded by u, then the maximal flow can be found by a simple *capacity scaling* algorithm in time $O(nm \log u)$ [EK72, Din70]. The fastest known algorithm runs in time $O(\min(n^{2/3}m, m^{3/2}) \log(n^2/m) \log u)$ [GR98]. For unit networks with $u = 1$, the logarithmic factor is not necessary and a simple combination of the capacity scaling algorithm and the blocking-flow algorithm runs in time $O(\min(n^{2/3}m, m^{3/2}))$ [ET75]. For undirected unit networks, there is an algorithm running in time $O(n^{3/2}\sqrt{m})$ [GR99], the fastest known deterministic algorithm runs in time $O(n^{7/6}m^{2/3})$, and the fastest known probabilistic algorithm runs in time $O(n^{20/9})$ [KL98].

Our results In this chapter, we analyze the quantum time complexity of these problems. We use Grover search to speed up searching for an edge. We present quantum algorithms for finding a maximal bipartite matching in time

$$O(n\sqrt{m}\log^2 n) \ ,$$

a maximal non-bipartite matching in time

$$O(n^2(\sqrt{m/n} + \log n)\log^2 n) \ ,$$

and a maximal flow in an integer network in time

$$O(\min(n^{7/6}\sqrt{m} \cdot u^{1/3}, \sqrt{num})\log^2 n)$$

when capacities are at most $u \leq n^{1/4}$. A similar approach was successfully applied by Dürr et al. [DHHM04] to the following graph problems: connectivity, strong connectivity, minimum spanning tree, and single source shortest paths. Our bipartite matching algorithm is polynomially faster than the best classical algorithm when $m = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$, and the network flows algorithm is polynomially faster when $m = \Omega(n^{1+\varepsilon})$ and u is small. Our non-bipartite matching algorithm is worse than the best known classical algorithm [MV80].

There is an $\Omega(n^{3/2})$ quantum adversary lower bound for the bipartite matching problem [BDF$^+$04, Zha05]. Since the bipartite matching problem is a special case of the other problems studied in this chapter, this implies an $\Omega(n^{3/2})$ quantum lower bound for all problems in this chapter.

The structure of this chapter is as follows. In Section 3.3, we present a quantum algorithm for computing a layered network from a given network. It is used as a tool in almost all our algorithms. In Section 3.4, we present a simple quantum algorithm for bipartite matching. In Section 3.5, we show how to quantize the classical algorithm for non-bipartite matching. In Section 3.6, we present a quantum algorithm for network flows.

3.2 Preliminaries

Graph theory A very good book about network flows is the book by Ahuja, Magnanti, and Orlin [AMO93]. It, however, does not contain most of the newest algorithms that we compare our algorithms to. In this chapter, we only use the following concepts. A *layered network* is a network whose vertices are ordered into layers such that the edges only go from the i^{th} layer to the $i + 1^{\text{st}}$ layer. For a given network and some flow, the *residual network* is a network whose capacities denote residual capacities of the edges with respect to the flow. When an edge has a capacity c and carries a flow f, then its residual capacity is either $c - f$ or $c + f$ depending on the direction. An *augmenting path* in a network is a path from the source to the sink whose residual capacity is bigger than 0. An *augmenting path* for the matching problem is a path that starts and ends in an unmatched vertex, and consists of alternated non-edges and edges of the current matching. A *blocking flow* in a layered residual network is a flow whose size cannot be increased by monotone operations, that is by only increasing the flow along some edges. A blocking flow may not be maximal; in such a case one has to decrease the flow along some edges to get out of a local maximum. A *cut* in a network is a subset of edges such that there is no path from the source to the sink if we remove these edges. The size of a cut is the sum of the capacities of its edges. Flows and cuts are dual to each other, and the size of a maximal flow is equal to the size of a minimal cut.

Black-box model Let us define our computational model. Let V be a set of vertices and let $E \subseteq \binom{V}{2}$ be a set of edges. V is fixed and E is a part of the input. Let $n \geq 1$ denote the number of vertices and let m denote the number of edges. We assume that $m \geq n - 1$, since one can eliminate zero-degree vertices in classical time $O(n)$. We consider the following two *black-box* models for accessing directed graphs:

- *Adjacency* model: the input is specified by an $n \times n$ Boolean matrix A, where $A[v, w] = 1$ if and only if $(v, w) \in E$.

```
FIND LAYERED SUBGRAPH (fixed vertex set $V$, input edge set $E$ and
starting vertex $a$) returns layer numbers $\ell : V \to \mathbb{N}_0$.

  1. Set $\ell(a) = 0$ and $\ell(x) = \infty$ for $x \neq a$.
     Create a one-entry queue $W = \{a\}$.

  2. While $W \neq \emptyset$, do the following:

       • take the first vertex $x$ from $W$,
       • find by GROVER SEARCH all its neighbors $y$ with $\ell(y) = \infty$,
         set $\ell(y) := \ell(x) + 1$, and append $y$ to $W$,
       • and remove $x$ from $W$.
```

Figure 3.1: Quantum algorithm Find Layered Subgraph

- *List* model: the input is specified by n arrays $\{N_v : v \in V\}$ of length $1 \leq d_v \leq n$. Each entry of an array is either a number of a neighbor or a *hole*, and all neighbors are included in the list without repetition, that is $\{N_v[i] : i = 1, \dots, d_v\} - \{\text{hole}\} = \{w : (v, w) \in E\}$.

Logarithmic factors Our algorithms are classical at a higher level, and they use quantum algorithms as subroutines. Quantum subroutines used in our algorithms may output an incorrect answer with a small constant probability. Our algorithms use a polynomial number n^c of these subroutines. Because of that, we have to repeat each quantum subroutine $O(\log n)$ times, to make sure that the probability of an incorrect answer is at most $\alpha = n^{-c+1}$. Then the probability that some quantum subroutine in our algorithm outputs an incorrect answer is at most $1/n$. This increases the running time of all our algorithms by a logarithmic factor. Furthermore, the running time of Grover search is bigger that its query complexity by another logarithmic factor.

3.2.1. Remark. For simplicity, we often omit logarithmic factors in the proofs, but we state them correctly in the statements of our theorems.

3.3 Finding a layered subgraph

We are given a connected directed black-box graph $G = (V, E)$ and a starting vertex $a \in V$, and we want to assign layers $\ell : V \to \mathbb{N}_0$ to its vertices such that $\ell(a) = 0$ and $\ell(y) = 1 + \min_{x:(x,y)\in E} \ell(x)$ otherwise. The quantum algorithm described in Figure 3.1 based on Dijkstra's algorithm computes layer numbers for all vertices.

3.3.1. Theorem. Find Layered Subgraph *runs in time $O(n^{3/2}\log^2 n)$ in the adjacency model and in time $O(\sqrt{nm}\log^2 n)$ in the list model.*

Proof. The algorithm is a quantum implementation of breadth-first search. The initialization costs time $O(n)$. Every vertex is processed at most once. Recall that we omit logarithmic factors. In the adjacency model, every vertex contributes by time at most $O(\sqrt{n})$, because finding a vertex from its ancestor costs time at most $O(\sqrt{n})$ and discovering that a vertex has no descendant costs the same.

In the list model, processing a vertex v costs time $O(\sqrt{n_v d_v} + \sqrt{d_v + 1})$, where n_v is the number of vertices inserted into W when processing v. Let q be the total number of processed vertices. Since $\sum_v n_v \leq q \leq n$ and $\sum_v (d_v + 1) \leq m + q = O(m)$, the total running time is upper-bounded by the Cauchy-Schwarz inequality as follows:

$$\sum_v \sqrt{n_v d_v} \leq \sqrt{\sum_v n_v}\sqrt{\sum_v d_v} = O(\sqrt{nm}) \ ,$$

and $\sum_v \sqrt{d_v + 1} \leq \sqrt{q}\sqrt{m+q} = O(\sqrt{nm})$ is upper-bounded in the same way. □

3.4 Bipartite matching

We are given an undirected bipartite black-box graph $G = (V_1, V_2, E)$ and we want to find a maximal matching among its vertices. This can be done classically in time $O(n^{5/2})$ [HK73] by the algorithm from Figure 3.2.

The algorithm is correct because (1) a matching is maximal if and only if there exists no augmenting path [HK73], and (2) the minimal length of an augmenting path is increased by at least one after every iteration. Classically, the construction of H and the depth-first search both cost $O(n^2)$. The maximal number of iterations is $O(\sqrt{n})$ due to the following statement.

3.4.1. Lemma. [HK73] *If M_1 and M_2 are two matchings of size s_1 and s_2 with $s_1 < s_2$, then there exist $s_2 - s_1$ vertex-disjoint augmenting paths in M_1.*

Proof (sketch). Consider the symmetric difference X of the two matchings. X consists of isolated vertices, vertex disjoint circles of even length, and vertex disjoint paths. There must be at least $s_2 - s_1$ paths of odd length with one more edge from M_2 than from M_1. These are the augmenting paths. □

Let s be the size of the maximal matching M in G, and let s_i be the size of the matching M_i computed after the i^{th} iteration. Let j be the number of the last iteration with $s_j < s - \sqrt{n}$. The total number of iterations is at most $j + 1 + \sqrt{n}$, because the algorithm finds at least one augmenting path in every

Find Bipartite Matching (fixed vertex sets V_1, V_2, input edge set E) returns a maximal matching M.

1. Set M to the empty matching.
2. Let a, b denote two new vertices not present in V_1 or V_2.
   Let $G' = (V', E')$ denote the following directed graph.

   $$\begin{aligned} V' &= V_1 \cup V_2 \cup \{a, b\} \\ E' &= \{(a, x) : x \in V_1, x \in W\} \\ &\quad \cup \{(x, y) : x \in V_1, y \in V_2, (x, y) \in E, (x, y) \notin M\} \\ &\quad \cup \{(y, x) : x \in V_1, y \in V_2, (x, y) \in E, (x, y) \in M\} \\ &\quad \cup \{(y, b) : y \in V_2, y \in W\} \ , \end{aligned}$$

   where W is the set of vertices that are unmatched.

   Find a maximal (with respect to inclusion) set S of vertex-disjoint augmenting paths of minimal length. This is done as follows:

   (a) Construct $H =$ Find Layered Subgraph (V', E', a).

   (b) Perform depth-first search for vertex-disjoint paths from a to b in H. Each such path corresponds to an augmenting path in M. They all have the same length and this length is minimal, because H is a layered graph. Figure 3.3 illustrates an example vertex disjoint set of augmenting paths.
3. Augment the matching M by S.
4. If $S \neq \emptyset$, go back to step 2, otherwise output the matching M.

Figure 3.2: Classical algorithm [HK73] Find Bipartite Matching

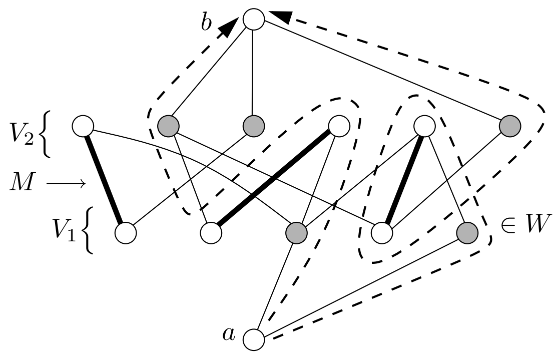


Figure 3.3: Vertex-disjoint augmenting paths in an example layered graph

iteration. On the other hand, by Lemma 3.4.1, there exist $s - s_j > \sqrt{n}$ vertex-disjoint augmenting paths in M_j. Since all augmenting paths in the j^{th} iteration are of length at least $j+2$ (each iteration adds at least 1 to the minimal length of an augmenting path, and there are two more edges going from and to vertices a, b), it must be that $j < \sqrt{n}$, otherwise the paths could not be disjoint. We conclude that the total number of iterations is at most $2\sqrt{n}$.

3.4.2. Theorem (Bipartite matching). *There exists a quantum algorithm that finds a maximal bipartite matching in time $O(n^2 \log^2 n)$ in the adjacency model and $O(n\sqrt{m} \log^2 n)$ in the list model.*

Proof. Recall that we omit logarithmic factors. We present a quantum algorithm that finds all augmenting paths in one iteration in time $O(n^{3/2})$, resp. $O(\sqrt{nm})$. Since the number of iterations is $O(\sqrt{n})$, the upper bound on the running time follows. Our algorithm works similarly to the classical one; it also computes the layered graph H and then searches in it.

The intermediate graph G' is not pre-computed, but it is generated on-line from the input graph G and the current matching M. One edge query is answered using a constant number of queries as follows. The restriction of G' to $V_1 \times V_2$ is a subgraph of G with some edges removed; here we exploit the fact that lists of neighbors can contain holes. We also add two new vertices a and b, add one list of neighbors of a with holes of length n, and at most one neighbor b to every vertex from V_2. Find Layered Subgraph computes H from G' fast by Theorem 3.3.1. It remains to show how to find the augmenting paths.

This is simple once we have assigned layer numbers to all vertices. We find a maximal set of vertex-disjoint paths from a to b by depth-first search. A descendant of a vertex is found by Grover search over all *unmarked* vertices with a layer number that is bigger by one. All vertices are unmarked in the beginning. When we find a descendant of some vertex, we mark it and continue backtracking. Either the vertex will become a part of an augmenting path, or it does not belong to any and hence it need not be probed again. Each vertex is thus visited at most once.

In the adjacency model, every vertex costs time $O(\sqrt{n})$ to be found and time $O(\sqrt{n})$ to discover that it does not have any descendant. In the list model, a vertex v costs time $O(\sqrt{n_v d_v} + \sqrt{d_v})$, where n_v is the number of unmarked vertices found from v. The sum over all vertices is upper-bounded like in the proof of Theorem 3.3.1. Note that during the construction of G' from G, $\sum_v d_v$ is increased by at most $2n$. $\square$

3.5 Non-bipartite matching

We are given an undirected black-box graph $G = (V, E)$ and we want to find a maximal matching among its vertices. There is a classical algorithm [Edm65, Gab76] running in total time $O(n^3)$ in n iterations of time $O(n^2)$.

Each iteration consists of searching for one augmenting path. The algorithm Find Augmenting Path performs breadth-first search from some unmatched vertex a. The algorithm is described in Figure 3.4 and its progress on an example graph is outlined in Figure 3.5. Find Augmenting Path browses paths that consist of alternated non-edges and edges of the current matching. For each vertex, we store a pointer *mate* that either points to a mate of the vertex if it is matched, or it is set to *nil*. Let us call a vertex v *even* if we have found such an alternating path of even length from a to v; otherwise we call it *odd*. Newly reached vertices are considered odd. For each even vertex v, we store two pointers *link* and *link2* for constructing the alternating path back, and a pointer *first* pointing at the last odd vertex on this path. Basically, if *link2* is nil, then v was discovered by the quantum search and *link* points to the previous even vertex on this path; it has always distance 2. If *link2* is not nil, then v became even by collapsing an odd-length circle and (*link*, *link2*) is the edge that caused the collapse.

If no augmenting path has been found from some vertex, then there will be none even later after more iterations of the algorithm. Hence it suffices to search for an augmenting path from each vertex exactly once.

3.5.1. Theorem (Non-bipartite matching). *There exists a quantum algorithm that finds a maximal non-bipartite matching in time $O(n^{5/2}\log^2 n)$ in the adjacency model and $O(n^2(\sqrt{m/n}+\log n)\log^2 n)$ in the list model.*

Proof. Recall that we omit logarithmic factors. The quantum algorithm iteratively augments the current matching by single augmenting paths, like the classical algorithm. An augmenting path is found by the quantum version of Find Augmenting Path that uses Grover search in faster time $O(n^{3/2})$, resp. $O(n(\sqrt{m/n}+\log n))$. This implies the upper bound on the total running time, since there are n vertices and each of them is used as the starting vertex a at most once. Let us prove the time bound for the list model.

Let q denote the number of even vertices found by Find Augmenting Path. For every even vertex v, we perform the following 3 quantum searches:

1. We look for an unmatched neighbor of v in time $O(\sqrt{d_v})$.
2. Let o_v be the number of odd neighbors of v whose mate is also odd. We find them all in total time $O(\sqrt{o_v d_v})$.
3. Let b_v be the number of bridges that are found during processing v. We find all even neighbors of v whose pointer *first* is different from v's pointer *first* in time $O(\sqrt{b_v d_v})$.

Clearly $\sum_v o_v \le q \le n$ and $\sum_v b_v \le q \le n$, and, since $\sum_v d_v \le m$, by the Cauchy-Schwarz inequality, the total time spent in all quantum searches is $O(\sqrt{nm})$.

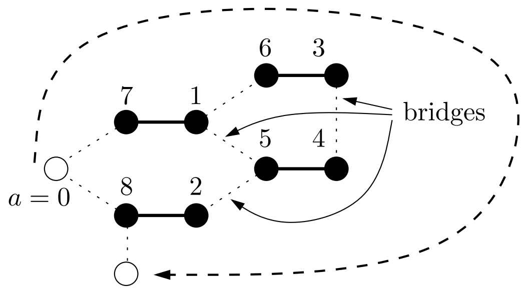
Find Augmenting Path (fixed vertex set V, input edge set E, matching *mate*, unmatched vertex a) returns an augmenting path from a.

1. Create a one-entry queue of even vertices $W = \{a\}$.
2. Take the first vertex v from W and delete it from W.
3. If there is an unmatched vertex w connected to v, then construct the augmenting path $a \to v$, add the edge (v, w) to it, and exit.

   A subpath $g \to h$ for arbitrary even vertices g, h on the augmenting path is constructed recursively using the pointers of h as follows:

   - If *link2* is nil, then *link* points at the last even vertex on the subpath. Construct thus subpath $g \to$ *link* and add 2 edges (*link*, *mate*) and (*mate*, h) to it.
   - Otherwise h was reached via a bridge, *link* points at h's side of the bridge, and *link2* at the other side. Construct subpath $g \to$ *link2*, then construct subpath $h \to$ *link* (it goes in the opposite direction), and join the two paths via (*link*, *link2*).
4. For every odd vertex w connected to v, do the following:
   - Let w be connected to a mate w'. If w' is even, do nothing.
   - Otherwise mark w' as even, append it to W, and set its pointers as follows: *link* to v, *link2* to nil, and *first* to w.
5. For every even vertex w connected to v, do the following:
   - If the pointers *first* of v and w are equal, do nothing.
   - Otherwise v and w lie on a circle of odd length, and the edge (v, w) is a bridge via which odd vertices turn to even. Find the nearest common odd ancestor p of v and w using the link-list of pointers *first*. Collapse the circle as follows:
     - Mark all odd vertices between v and p as even, append them to W, and set their pointers as follows: *link* to v, *link2* to w, and *first* to p.
     - Do the opposite for all odd vertices between w and p.
     - Finally, rewrite to p all pointers *first* pointing at the odd vertices that have just become even.
6. If W is empty, then there is no augmenting path from a and we quit, otherwise go back to step 2.

Figure 3.4: Classical algorithm [Edm65, Gab76] Find Augmenting Path



- Thick solid lines denote the current matching and dotted lines are the remaining edges of the graph. Unmatched vertices are white.
- The vertices are numbered in the order in which they have turned even. The augmenting path is 0, 7, 1, 6, 3, 4, 5, 2, 8, and the final unmatched vertex.
- Vertices 5, 6, 7, and 8 have been reached via the 3 highlighted bridges.

Figure 3.5: Progress of Find Augmenting Path on an example graph

Let us estimate the running time of collapsing one circle. Let p_1 be the length of the link-list of pointers *first* from one side of the bridge to the nearest common ancestor, let p_2 be the other one, and let $p = \max(p_1, p_2)$. The nearest common ancestor is found in time $O(p \log p)$ as follows. We maintain two balanced binary trees for each link-list, add vertices synchronously one-by-one, and search for every newly inserted vertex in the opposite tree until we find a collision. Let r_v be the number of odd vertices collapsed during processing v. It holds that $r_v = p_1 + p_2 = \Theta(p)$ and $\sum_v r_v \leq q$. Hence the total time spent in collapsing circles is $O(q \log q)$. Immediate rewriting of the pointers *first* of even vertices inside collapsed circles would be too slow. We instead maintain aside a union tree of these pointers, and for every odd vertex converted to even, we just append its subtree to the node of the nearest common ancestor. The total time spent in doing this is $O(q \log q)$.

The augmenting path has length at most n and it is constructed in linear time. We conclude that the total running time of finding an augmenting path is $O(\sqrt{nm} + n \log n) = O(n(\sqrt{m/n} + \log n))$, and it is $O(\sqrt{nm})$ for $m \geq n(\log n)^2$. The running time in the adjacency model is equal to the running time in the list model for $m = n^2$, and it is $O(n^{3/2})$. □

Our quantum algorithm is slower than the fastest known classical algorithm by Micali and Vazirani [MV80] running in total time $O(\sqrt{n}m)$. Their algorithm is quite complicated and it actually took a few years to prove it correct. It may be interesting to apply our techniques to that algorithm.

3.6 Integer network flows

We are given a black-box directed network with integer capacities bounded by u, and we want to find a maximal flow from the source to the sink. We present a fast quantum algorithm based on GROVER SEARCH and combining three classical techniques: blocking flows in layered residual networks [Din70], searching single augmenting paths [EK72], and switching between them at a good time [ET75].

3.6.1. LEMMA. [ET75] *Consider an integer network with capacities bounded by u, and a flow whose layered residual network has depth k. Then the size of the residual flow is at most* $\min((2n/k)^2, m/k) \cdot u$.

PROOF.

(1) There exist layers V_ℓ and $V_{\ell+1}$ that both have less than $2n/k$ vertices. This is because if for every $i = 0, 1, \dots, k/2$, at least one of the layers V_{2i}, V_{2i+1} had size at least $2n/k$, then the total number of vertices would exceed n. Since V_ℓ and $V_{\ell+1}$ form a cut, the residual flow has size at most $|V_\ell| \cdot |V_{\ell+1}| \cdot u \leq (2n/k)^2 u$.

(2) For every $i = 0, 1, \dots, k-1$, the layers V_i and V_{i+1} form a cut. These cuts are disjoint and they together have at most m edges. Hence at least one of them has at most m/k edges, and the residual flow has thus size at most $O(mu/k)$. □

3.6.2. THEOREM (NETWORK FLOWS). *Let $u \leq n^{1/4}$. There exists a quantum algorithm that finds a maximal network flow with integer capacities bounded by u in time $O(n^{13/6} \cdot u^{1/3} \log^2 n)$ in the adjacency model and $O(\min(n^{7/6}\sqrt{m} \cdot u^{1/3}, \sqrt{num}) \log^2 n)$ in the list model.*

PROOF. The algorithm iteratively augments the flow by blocking flows in layered residual networks until the depth of the network exceeds $k = \min(n^{2/3}u^{1/3}, \sqrt{mu})$. Then it switches to searching augmenting paths while there are some. Our algorithm uses classical memory of size $O(n^2)$ to store the current flow and its direction for every edge of the network, and a 1-bit status of each vertex. Its first subroutine FIND BLOCKING FLOW is outlined in Figure 3.6.

The layered subgraph of G' is actually never explicitly constructed in the memory. When we need to find a neighbor of a vertex v in it, we search among the neighbors of v in G' conditioned on that their layer number is bigger by one; arrays with holes are used here to mask out unwanted vertices. Layer numbers are assigned to the vertices of G' fast by Theorem 3.3.1. When the flow F is augmented by μ along a path ρ, no update of data structures is needed. Saturated edges will be automatically omitted at the next query, because the edges of G' and their capacities are computed on-line (in constant time per query).

```
FIND BLOCKING FLOW (fixed vertex set $V$, input edge set $E$, capacities,
current flow, source, and sink) returns a blocking flow $F$.

  1. Run FIND LAYERED SUBGRAPH $(V, E', \text{source})$ to compute layer
     numbers of vertices of the residual network $G' = (V, E')$. The
     capacity of each edge in $E'$ is equal to the original capacity plus
     or minus the current flow depending on the direction. Only edges
     with nonzero capacities are included in $E'$.

  2. Mark all vertices in $V$ as enabled. Set $F$ to the empty flow.

  3. Run depth-first search using GROVER SEARCH as a subroutine, and
     find a path $\rho$ in the layered subgraph of $G'$ from the source to the
     sink that only goes through enabled vertices. If there is no such
     path, exit. During back-tracking, disable all vertices from which
     there is no path to the sink.

  4. Compute the minimal capacity $\mu$ of an edge on $\rho$.
     Augment $F$ by $\mu$ along $\rho$.

  5. Go back to step 3.
```

Figure 3.6: Quantum algorithm FIND BLOCKING FLOW

Let us compute how much time the algorithm spends in total in a vertex v during the search the augmenting paths. Let a_v denote the number of augmenting paths going through v and let $e_{v,i}$ denote the number of outgoing edges from v at the moment when there are still i remaining augmenting paths in comparison to the final set that will be output by the algorithm. The capacity of every edge is at most u, hence $e_{v,i} \geq \lceil i/u \rceil$. We will again omit logarithmic factors. The time spent in quantum searches leading to an augmenting path in v is thus at most

$$\sum_{i=1}^{a_v} \sqrt{\frac{d_v}{e_{v,i}}} \leq \sqrt{u} \cdot \sum_{i=1}^{a_v} \sqrt{\frac{d_v}{i}} = O(\sqrt{u a_v d_v}) \enspace .$$

Let c_v denote the number of enabled vertices found from v that do not lie on an augmenting path and are thus disabled. The time spent in quantum searches for these vertices is at most $O(\sqrt{c_v d_v})$. Furthermore, it takes additional time $O(\sqrt{d_v + 1})$ to discover that there is no augmenting path from v, and in this case v is disabled and never visited again.

Let j denote the depth of the residual network and let b_j be the size of its blocking flow. The total number of augmenting paths going through vertices in any given layer is at most b_j. We conclude that $\sum_v a_v \leq j b_j$. We also know that $\sum_v c_v \leq n$. Since $\sum_v d_v \leq m$, by the Cauchy-Schwarz inequality, the total time spent by finding one blocking flow is

$$\sum_v (\sqrt{u a_v d_v} + \sqrt{c_v d_v} + \sqrt{d_v + 1}) \le \sqrt{u}\sqrt{\sum_v a_v}\sqrt{\sum_v d_v} + 2\sqrt{nm}$$
$$= O(\sqrt{j m b_j u} + \sqrt{nm}) \ .$$

Our algorithm performs at most

$$k = \min(n^{2/3} u^{1/3}, \sqrt{mu})$$

iterations of finding the blocking flow in total time at most $\sqrt{mu} \cdot \sum_{j=1}^k \sqrt{j b_j} + k\sqrt{nm}$. Let us assume that the algorithm has not finished, and estimate the size of the residual flow and thus upper-bound the number of augmenting paths that need to be found. The algorithm constructs in the k^{th} iteration a layered network of depth bigger than k. By Lemma 3.6.1, the residual flow has size $O(\min((n/k)^2, m/k) \cdot u) = O(k)$, hence the algorithm terminates in $O(k)$ more iterations. From this point on, the algorithm only looks for one augmenting path in each layered network, hence its complexity drops to $O(\sqrt{nm})$ per iteration, omitting the factor $\sqrt{b_j u}$. The total running time is thus at most

$$O\Big(\sqrt{mu} \cdot \sum_{j=1}^k \sqrt{j b_j} + k\sqrt{nm}\Big) + O(k\sqrt{nm}) \ .$$

Let us prove that $\sum_j \sqrt{j b_j} = O(k^{3/2})$. We split the sequence $[1, k]$ into $\log k$ intervals $S_i = [2^i, 2^{i+1} - 1]$ of length 2^i. By Lemma 3.6.1, the residual flow after $\ell = k/2^i$ iterations is at most $O(\min((n/k)^2 \cdot 2^{2i}, m/k \cdot 2^i) \cdot u) \le O(2^{2i} k) = O((k/\ell)^2 \ell) = O(k^2/\ell)$. Since the total size of all blocking flows cannot exceed the residual flow, $\sum_{j=\ell}^{2\ell-1} b_j = O(k^2/\ell)$. By applying the Cauchy-Schwarz inequality independently on each block, we get

$$\sum_{j=1}^k \sqrt{j b_j} = \sum_{i=0}^{\log k} \sum_{j=2^i}^{2^{i+1}-1} \sqrt{j b_j} \le \sum_{i=0}^{\log k} \sqrt{2^i \cdot 2^{i+1}} \sqrt{\sum_{j=2^i}^{2^{i+1}-1} b_j}$$
$$\le \sqrt{2} \sum_{i=0}^{\log k} 2^i \sqrt{k^2/2^i} = \sqrt{2} \cdot k \sum_{k=0}^{\log k} 2^{i/2} = O(k^{3/2}) \ .$$

The total running time is thus $O(k\sqrt{m}(\sqrt{ku} + \sqrt{n}))$. Now, $ku \le n$, because we assume that $u \le n^{1/4}$ and $ku = \min(n^{2/3} u^{4/3}, \sqrt{m} \cdot u^{3/2}) \le n^{2/3} n^{1/3} = n$. The running time is therefore $O(k\sqrt{nm}) = O(\min(n^{7/6}\sqrt{m} \cdot u^{1/3}, \sqrt{n}um))$. The time for the adjacency model follows from setting $m = n^2$ and it is $O(n^{13/6} \cdot u^{1/3})$. □

It is actually simpler to compute a similar (weaker) upper bound on the running time of the network flows algorithm in the more general setting without assuming $u \leq n^{1/4}$. One obtains $O(\min(n^{7/6}\sqrt{m}, \sqrt{n}m) \cdot u \log n)$ for arbitrary u by setting $k = \min(n^{2/3}, \sqrt{m})$. It would be interesting to apply techniques of Goldberg and Rao [GR98] to improve the multiplicative factor in Theorem 3.6.2 from poly(u) to $\log u$. That would make our algorithm useful in the more realistic setting of polynomially large capacities. If $m = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$ and u is small, then our algorithm is polynomially faster than the best classical algorithm. For constant u and $m = O(n)$, it is slower by at most a polylogarithmic factor. The speedup is biggest for dense networks with $m = \Omega(n^2)$.

3.6.3. Theorem. *Any bounded-error quantum algorithm for network flows with integer capacities bounded by $u = n$ has quantum query complexity $\Omega(n^2)$.*

Proof. Consider the following layered graph with $m = \Theta(n^2)$ edges. The vertices are ordered into 4 layers: the first layer contains the source, the second and the third layer contain $p = \frac{n}{2} - 1$ vertices each, and the last layer contains the sink. The source and the sink are both connected to all vertices in the neighboring layer by p edges of full capacity n. The vertices in the second and third layer are connected by either $\frac{p^2}{2}$ or $\frac{p^2}{2} + 1$ edges of capacity 1 chosen at random. The edges between these two layers form a minimal cut. Now, deciding whether the maximal flow is $\frac{p^2}{2}$ or $\frac{p^2}{2} + 1$ allows us to compute the majority on p^2 bits. By Corollary 2.4.3, majority requires $\Omega(p^2) = \Omega(n^2)$ queries, hence the same lower bound also holds for the computation of a maximal flow. $\square$

3.7 Summary

We have presented two quantum algorithms for the problem of finding a maximal matching in a graph. If the graph is bipartite, then our quantum algorithm is polynomially faster than the best classical algorithm. In the non-bipartite case, there exists a more complicated classical algorithm that is faster than our quantum algorithm. It is possible that one can use similar techniques to quantize that algorithm too.

We have shown how to find quantumly a maximal flow in an integer network. If the capacities are upper-bounded by a polynomial of small degree, then our quantum algorithm is polynomially faster than the best classical algorithm. There is a classical algorithm whose complexity depends on the logarithm of the maximal capacity instead of on some polynomial function. If one could quantize that algorithm, then quantum computers would outperform classical ones in a much wider range of capacities.

Chapter 4

Matrix Verification

This chapter is based on the following paper:

[BŠ06] H. Buhrman and R. Špalek. Quantum verification of matrix products. In *Proceedings of 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 880–889, 2006.

4.1 Introduction

The computational complexity of matrix multiplication is the subject of extensive study. Matrix multiplication is the central algorithmic part of many applications, like for example solving linear systems of equations and computing the transitive closure of a graph. A fast algorithm for matrix multiplication thus implies a fast algorithm for a variety of computational tasks. Strassen [Str69] was the first to show that surprisingly two $n \times n$ matrices can be multiplied in time $O(n^\omega)$ for $\omega < 3$. His result was improved by many subsequent papers. The best known bound to date is an algorithm with $\omega \approx 2.376$ by Coppersmith and Winograd [CW90]. It is a major open problem to determine the true value of ω. Freivalds showed [Fre79] that *verifying* whether the product of two $n \times n$ matrices is equal to a third can be done with high probability in time proportional to n^2. We refer to this latter problem as *matrix verification.*

We study the computational complexity of matrix multiplication and verification on a quantum computer. The first to study matrix verification in the quantum mechanical setting were Ambainis, Buhrman, Høyer, Karpinski, and Kurur [ABH+02] who used a clever recursive version of Grover search to verify whether two $n \times n$ matrices equal a third in time $O(n^{7/4})$, thereby improving the optimal classical bound of Freivalds.

In this chapter we construct a bounded-error quantum algorithm for the matrix verification problem that runs in time $O(n^{5/3})$. Suppose we are verifying whether $AB = C$. An entry (i, j) in C is called *wrong* if $C[i, j] \neq$

$\sum_{k=1}^{n} A[i,k] \cdot B[k,j]$. When the number of wrong entries in C is $w \leq \sqrt{n}$, our algorithm runs in expected time

$$O\left(\frac{n^{5/3}}{w^{1/3}}\right) .$$

For $w = \sqrt{n}$ we have a matching lower bound. For larger values of w, the algorithm may run faster, but its performance depends on the concrete pattern of wrong entries. We use our fast quantum matrix verification algorithm as a building block to construct a quantum algorithm for computing the actual matrix product $A{\cdot}B$. When the number of nonzero entries in the final product is $w \leq \sqrt{n}$, our algorithm runs in expected time

$$O(n^{5/3} w^{2/3} \log n) ,$$

that is substantially faster than any classical method. For larger values of w, the classical algorithm by Indyk [Ind05] running in time $\tilde{O}(n^2 + nw)$ takes over.

Our algorithm uses the quantum random walk formalism by Szegedy [Sze04] that he developed as a generalization of the quantum random walk technique of Ambainis [Amb04], see Section 1.4.1 and Section 1.4.2. Ambainis used a quantum random walk to obtain an optimal quantum algorithm for the element distinctness problem. If one were to adapt that method directly to the setting of matrix verification, one does obtain an $O(n^{5/3})$ algorithm in terms of queries to the input. However that algorithm still requires $\Omega(n^2)$ time, because it computes several times a matrix product of sub-matrices. This costs no additional queries, because the sub-matrices are loaded in the memory, but it takes additional time. Using random vectors, we improve the running time of the quantum algorithm to $O(n^{5/3})$.

We perform a quantum random walk on the product of two Johnson graphs, analyze its spectral gap, and estimate that in our setting enough of the vertices are marked if $AB \neq C$. See Section 4.3 for a detailed description of our matrix verification algorithm and Section 4.4 for the analysis of its running time. In Section 4.5, we introduce a combinatorial tool to analyze the behavior of our algorithm when many of the entries are wrong. In Section 4.6, we describe our matrix multiplication algorithm, and in Section 4.7, we discuss products of Boolean matrices.

4.2 Previous algorithm for matrix verification

Ambainis, Buhrman, Høyer, Karpinski, and Kurur [ABH$^+$02] discovered a quantum algorithm for matrix verification running in time $O(n^{7/4})$. Since it was never published, we briefly sketch it in Figure 4.1. It uses Freivalds's trick [Fre79] with multiplying the matrices by a random vector. The trick works as follows.

Older Matrix Verification (input $n \times n$ matrices A, B, C)
returns whether $AB \neq C$.

1. Partition the matrices B and C into $\sqrt{n}$ blocks of $\sqrt{n}$ columns each. Let B_i and C_i be the individual sub-matrices of size $n \times \sqrt{n}$. It holds that $AB = C$ if and only if $AB_i = C_i$ for every i.
2. The verification of $AB_i = C_i$ is done with bounded error in time $O(n^{3/2})$ by the following subroutine V_i:
   (a) Choose a random vector x of dimension $\sqrt{n}$.
   (b) Multiply both sides of the equation by x from the right side. Compute classically $y = B_i x$ and $z = C_i x$ in time $O(n^{3/2})$.
   (c) Verify the matrix-vector product $Ay = z$ by Grover search. Search for a row j such that $(Ay - z)_j \neq 0$. Quantum search over n rows cost $O(\sqrt{n})$ iterations and a verification of one row takes time n.
3. If we pick a block i at random, the probability that we have picked a block with a wrong entry is at least $1/\sqrt{n}$. Apply amplitude amplification on top of this procedure, and verify all $\sqrt{n}$ blocks with bounded error using $O(n^{1/4})$ calls to V_i.

Figure 4.1: Quantum algorithm Older Matrix Verification [ABH+02]

Consider matrices A, B of the same dimension over some field. If $A = B$, then $Av = Bv$ for every vector v. On the other hand, if $A \neq B$ and v is chosen uniformly at random, then $Av \neq Bv$ with probability at least one half. Therefore matrix verification can be solved with constant 1-sided error in time $O(n^2)$. The quantum algorithm Older Matrix Verification uses this trick and two recursive applications of Grover search.

It seems that one cannot improve on this algorithm using only Grover search. Our algorithm is based on quantum walks and achieves better running time. It is inspired by Ambainis's quantum walk algorithm Element Distinctness [Amb04] from Section 1.4.1.

4.2.1. Remark. The running time of all algorithms presented in this chapter is polylogarithmic in the size of the field the matrices are over. We assume that the field has constant size and hide the dependency in the big-O constant. If the field contains polynomially many elements, then the running time is longer by a polylogarithmic factor.

4.3 Algorithm for matrix verification

Let A, B, C be $n \times n$ matrices over some integral domain. An *integral domain* is a commutative ring with identity and no divisors of 0. Any field, such as $\mathbb{R}$ or $\mathbb{GF}(2)$, is an integral domain and so is $\mathbb{N}$. *Matrix verification* is deciding whether $AB = C$. We construct an efficient quantum walk algorithm for this problem. It is described in Figure 4.2 and its expected running time is analyzed in Section 4.4 and Section 4.5.

4.3.1. Definition. For a set $R \subseteq [n]$, let $A|_R$ denote the $|R| \times n$ sub-matrix of A restricted to the rows from R. Analogously, for a set $S \subseteq [m]$, let $A|^S$ denote the $n \times |S|$ sub-matrix of A restricted to the columns from S. These two notation can be combined, for example $A|_R^S$.

The intuition behind the algorithm is the following: The subroutine Verify Once performs a quantum walk on subsets R of rows of A and subsets S of columns of B, multiplied by phase flips whenever $C|_R^S$ contains at least one wrong entry. It then estimates the scalar product of the superposition computed by the quantum walk and the uniform superposition. If $AB = C$, then the phase flip is never performed and the diffusion on a uniform superposition is equal to identity. Hence the superposition computed by the quantum walk stays uniform, and the scalar product is 1. On the other hand, if $AB \neq C$ and k is sufficiently large, then for the number of quantum walk steps ℓ drawn uniformly from $\{1, 2, \dots, k\}$, with high probability, the quantum walk converges to a superposition almost orthogonal to the uniform superposition. Since the optimal value of k is not known beforehand, the main routine Matrix Verification tries a sequence of exponentially increasing k. We also use the following trick to improve the performance of the algorithm. Instead of checking all of $A|_R$ and $B|^S$, we multiply the matrices in Verify Once from both sides by random vectors p, q. This allows us to achieve both better running time and smaller space complexity.

4.4 Analysis of the algorithm

In this section, we prove that Matrix Verification has 1-sided bounded error and estimate its expected running time depending on the set of wrong entries. We use the formalism of Szegedy's quantum walk described in Section 1.4.2.

4.4.1 Multiplication by random vectors

Let Verify Full denote a modified version of Verify Once that does not use the random vectors p, q, but instead reads all sub-matrices into the memory and verifies $A|_R \cdot B|^S = C|_R^S$ in the phase-flip step (6a). Verify Full has the same query complexity as Verify Once, but its space complexity and running time

```
MATRIX VERIFICATION (input $n \times n$ matrices $A, B, C$)
returns whether $AB \neq C$.

1. Take any $1 < \lambda < \frac{8}{7}$, for example $\lambda = \frac{15}{14}$.
2. For $i = 0, 1, \ldots, \log_\lambda(n^{2/3}) + 9$, repeat 16 times the following:
    • Run VERIFY ONCE $(2 \cdot \lambda^i)$.
    • If it returns 1, then return "not equal".
3. Return "equal".

VERIFY ONCE (number of rows $k$) returns 1 when $AB \neq C$ is detected.

4. Pick a number of iterations $\ell$ uniformly at random between 1 and $k$. Pick a random row vector $p$ and a random column vector $q$ of dimension $n$.
5. **Initialization.** Put the quantum register into the superposition $\sum_{\substack{R \subseteq [n] \\ |R|=k}} \sum_{\substack{S \subseteq [n] \\ |S|=k}} |R\rangle|S\rangle$. Think of $R$ as a subset of the rows of $A$ and $S$ as a subset of the columns of $B$. Compute vectors $a_R = p|^R \cdot A|_R$ and $b_S = B|^S \cdot q|_S$, and number $c_{R,S} = p|^R \cdot C|_R^S \cdot q|_S$ in time $2kn + k^2$. Add a one-qubit *control register* set to $|+\rangle$. The quantum state is
       $$|+\rangle \sum_{R,S} |R, a_R\rangle\, |S, b_S\rangle\, |c_{R,S}\rangle \ .$$
6. **Quantum walk.** Conditioned on the value of the control register, perform $\ell$ iterations of the following:
    (a) **Phase flip.** Multiply the quantum phase of $|R\rangle|S\rangle$ by $-1$ if $a_R \cdot b_S \neq c_{R,S}$. The scalar product is verified in time $n$ using no queries.
    (b) **Diffusion.** Perform one step of quantum walk on $(R, S)$, that is exchange one row, and then one column. The update of $a_R$, $b_S$, and $c_{R,S}$ costs $2n$ queries to $A$, $2n$ queries to $B$, and $4k$ queries to $C$.
7. Apply the Hadamard transform on the control register, measure it in the computational basis, and return the outcome.
```

Figure 4.2: Quantum algorithm MATRIX VERIFICATION

are bigger. Although the phase flip costs no additional queries, the time needed to compute classically $A|_R \cdot B|^S$ is at least kn, whereas the scalar product $a_R \cdot b_S$ is computed in time n.

The multiplication by p, q complicates the analysis. For example, if there is exactly one wrong entry and the multiplication is over $\mathbb{GF}(2)$, then the wrong entry is completely hidden by a multiplication by zero with probability $\frac{3}{4}$. However, we prove that 16 independent calls to Verify Once have a better success probability than one call to Verify Full.

Let $R, S \subseteq [n]$ denote subsets of rows of A and columns of B, and let

$$W = \{(i,j) \,|\, (AB - C)[i,j] \neq 0\}$$

be the *set of wrong entries* of the matrix product. We *mark* (R, S) if and only if $C|_R^S$ contains a wrong entry, formally $A|_R \cdot B|^S \neq C|_R^S$, or equivalently $W \cap R \times S \neq \emptyset$. Let

$$\varepsilon(W, k) = \Pr_{R,S}\left[(R,S) \text{ is marked}\right]$$

be the *fraction of marked pairs* for $|R| = |S| = k$.

We call (R, S, p, q) *revealing* if and only if $p|^R \cdot (A|_R \cdot B|^S) \cdot q|_S \neq p|^R \cdot C|_R^S \cdot q|_S$, because it reveals the existence of a wrong entry in the matrix product. The condition is equivalent to $(p|^R \cdot A|_R) \cdot (B|^S \cdot q|_S) = a_R \cdot b_S \neq c_{R,S}$ due to the associativity of matrix multiplication. The performance of the algorithm depends on the *fraction of revealing pairs*

$$\zeta_{p,q}(W, k) = \Pr_{R,S}\left[(R, S, p, q) \text{ is revealing}\right] ,$$

where $|R| = |S| = k$. By the above example, not every marked pair is revealing. However, the fraction of revealing pairs is expected to be not too far from the fraction of marked pairs.

4.4.1. Lemma. *Let G be an integral domain with $g \geq 2$ elements, let (R, S) be marked, and let p, q be picked uniformly at random from G^n. The probability τ that a quadruple (R, S, p, q) is revealing is $\tau \geq (1 - \frac{1}{g})^2 \geq \frac{1}{4}$.*

Proof. Let $D = AB - C$, that is the wrong entries correspond to the nonzero entries of D. Assume that (R, S) is marked and pick any $D[i_0, j_0] \neq 0$. Now, (R, S, p, q) is not revealing if and only if

$$0 = \sum_{i \in R, j \in S} p_i q_j D[i,j] = \sum_{\substack{i \in R, j \in S \\ D[i,j] \neq 0}} p_i q_j D[i,j] = p_{i_0}(q_{j_0} D[i_0, j_0] + c_1) + c_2 q_{j_0} + c_3 ,$$

where c_1, c_2, c_3 are some constants depending on D and other coordinates of p, q. Fix these constants and pick p_{i_0}, q_{j_0} at random from G. Let $\tau' = \Pr\left[q_{j_0} D[i_0, j_0] + \right.$

$c_1 = 0]$. Since G is an integral domain with g elements, $\tau' \leq \frac{1}{g}$. For every q_{j_0} such that $q_{j_0} D[i_0, j_0] + c_1 \neq 0$, the equality is equivalent to $c_4 p_{i_0} + c_5 = 0$ for other constants $c_4 \neq 0$ and c_5, which is again satisfied by at most 1 value of $p_{i_0} \in G$. Hence the probability of having equality is at most

$$\tau' \cdot 1 + (1 - \tau') \cdot \frac{1}{g} = \frac{1}{g} + \tau' \cdot \frac{g-1}{g} \leq \frac{1}{g} + \frac{g-1}{g^2} = \frac{2g-1}{g^2} \ .$$

The probability that (R, S, p, q) is revealing is thus $\tau \geq 1 - \frac{2g-1}{g^2} = (1 - \frac{1}{g})^2 \geq \frac{1}{4}$ and equality holds when $g = 2$. □

4.4.2. Claim. Let $0 \leq X \leq 1$ and $\mathrm{E}[X] \geq \alpha$. Then $\Pr[X \geq \beta] \geq \alpha - \beta$.

Proof. Decompose the expected value of X conditioned on $X \geq \beta$: $\mathrm{E}[X] = \mathrm{E}[X | X < \beta] \cdot (1 - \Pr[X \geq \beta]) + \mathrm{E}[X | X \geq \beta] \cdot \Pr[X \geq \beta]$. Rearrange, plug in $0 \leq X \leq 1$, and obtain $\Pr[X \geq \beta] = \frac{\mathrm{E}[X] - \mathrm{E}[X|X<\beta]}{\mathrm{E}[X|X \geq \beta] - \mathrm{E}[X|X<\beta]} \geq \frac{\alpha - \beta}{1-0} = \alpha - \beta$. □

4.4.3. Lemma. *Let p, q be picked uniformly at random. Then* $\Pr[\zeta_{p,q}(W,k) \geq \frac{1}{8}\varepsilon(W,k)] \geq \frac{1}{8}$.

Proof. Consider the following Boolean random variables: $V_{R,S,p,q} = 1$ if and only if (R, S, p, q) is revealing. Let $v_{p,q}$ be the *fraction of marked sets that are also revealing* when p, q multiply the equation, formally $v_{p,q} = \mathrm{E}_{\text{marked}\,(R,S)}[V_{R,S,p,q}]$. By Lemma 4.4.1, for every marked (R, S), $\mathrm{E}_{p,q}[V_{R,S,p,q}] \geq \frac{1}{4}$. It follows that $\mathrm{E}_{\text{marked}\,(R,S)}[\mathrm{E}_{p,q}[V_{R,S,p,q}]] \geq \frac{1}{4}$ and so $\mathrm{E}_{p,q}[\mathrm{E}_{\text{marked}\,(R,S)}[V_{R,S,p,q}]] = \mathrm{E}_{p,q}[v_{p,q}] \geq \frac{1}{4}$. By Claim 4.4.2, when p, q is picked uniformly at random, $\Pr[v_{p,q} \geq \frac{1}{8}] \geq \frac{1}{8}$. Hence in this "good" case, $\zeta_{p,q}(W,k) \geq \frac{1}{8}\varepsilon(W,k)$. □

The constant probability $\frac{1}{8}$ of picking "good" random vectors is compensated by a constant number of repetitions. Using the following lemma, it is sufficient to analyze the error of the algorithm as if it is performing Verify Full in each step.

4.4.4. Lemma. *Let $AB \neq C$. The probability that* Verify Once $(2k)$ *outputs 1 at least once in 16 independent trials, each time with new random p, q, is bigger than the success probability of one call to* Verify Full (k).

Proof. By Lemma 4.4.3, the success probability of Verify Once is at least $\frac{\tau}{8}$, where τ is the success probability of Verify Once given that it guesses "good" vectors p, q with $\zeta(W,k) \geq \frac{1}{8}\varepsilon(W,k)$. By Theorem 1.4.6, $\tau = 1 - \gamma \in [\frac{1}{8}, \frac{1}{2}]$ for every sufficiently large $k \geq k_0$. The factor $\frac{1}{8}$ between $\varepsilon(W,k)$ and $\zeta(W,k)$ is compensated by taking a twice bigger k_0; this follows from equations (4.1) and (4.2) in the next section. The success probability of 16 independent trials is at least $1 - (1 - \frac{\tau}{8})^{16} \geq 1 - (e^{-\tau})^2 \geq 1 - (1 - 0.64\tau)^2 \geq 1.28\tau - 0.4\tau^2 \geq \tau$, because $1 - x \leq e^{-x} \leq 1 - 0.64x$ for $x \in [0,1]$, and $\tau < 0.7$. □

4.4.2 Analysis of Matrix Verification

In this section, we analyze the expected running time of the main algorithm. We need the following two statements.

4.4.5. Definition. The *strong product graph* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted by $G = G_1 \times G_2$, is defined as follows: $G = (V, E)$, $V = V_1 \times V_2$, and $((g_1, g_2), (g_1', g_2')) \in E$ if and only if $(g_1, g_1') \in E_1$ and $(g_2, g_2') \in E_2$. Walking on a strong product graph corresponds to walking simultaneously and independently on two graphs.

4.4.6. Lemma. *Let $1 \le k \le \frac{n}{2}$. The spectral gap of $G_k = J(n,k) \times J(n,k)$ is $\delta(G_k) = \Theta(1/k)$.*

Proof. By Lemma 1.4.8, the spectral gap of the Johnson graph $J(n,k)$ is $\frac{n}{(n-k)k}$, which is $\Theta(1/k)$ for $1 \le k \le \frac{n}{2}$. The eigenvalues of the normalized adjacency matrix (see Definition 1.4.4) of a product graph are exactly all products of one eigenvalue of one graph and one eigenvalue of the other graph. Since the largest eigenvalue of each graph is one and the spectral gap is the difference between the largest two eigenvalues, it immediately follows that $\delta(J \times J) = \delta(J)$. We conclude that $\delta(G_k) = \Theta(1/k)$. □

4.4.7. Definition. A set of matrix entries $W \subseteq [n]^2$ is called an *independent set* if and only if it contains at most one item in every row and column.

4.4.8. Lemma (Fraction of marked pairs). *Let*

$$\rho(W) = \max(w', \min(|W|, \sqrt{n})) \ ,$$

where w' is the largest possible size of a subset of W that is an independent set. For every W and $k \le n^{2/3}/\rho(W)^{1/3}$, it holds that

$$\varepsilon(W, k) = \Omega\Big(\frac{k^2}{n^2}\rho(W)\Big) \ . \tag{4.1}$$

The main Lemma 4.4.8 is proved using simple combinatorics in Section 4.5.

4.4.9. Theorem. Matrix Verification *always returns "equal" if $AB = C$. If $AB \neq C$, then it returns "not equal" with probability at least $\frac{2}{3}$. Its worst-case running time is $O(n^{5/3})$, its expected running time is $O(n^{5/3}/\rho(W)^{1/3})$, and its space complexity is $O(n)$.*

Proof. We compute an upper bound on the expected number of iterations of Verify Full. By Lemma 4.4.4, the number of calls to Verify Once achieving the same success probability is less than 16, if we double the value of the parameter k that we pass to Verify Once in comparison with what we would pass to Verify Full. Verify Once walks ℓ quantum steps on the strong product graph of two Johnson graphs $G_k = J(n,k) \times J(n,k)$. The marked vertices of G_k correspond to marked pairs (R,S), those are pairs such that $A|_R \cdot B|^S \neq C|_R^S$. The initialization costs time $O(kn)$, a phase flip costs time n, and one step of the quantum walk costs time $4n+4k = O(n)$. The running time of Verify Once is thus $O((k+\ell)n) = O(kn)$. The scalar product of the two quantum distributions is estimated using Lemma 1.4.7.

Let $W \neq \emptyset$. By Theorem 1.4.6, Verify Once recognizes a wrong matrix product with bounded error for every $k \geq k_0$, where $k_0 = \Theta(1/\sqrt{\delta(G_{k_0})\,\varepsilon(W,k_0)})$. Note that the same k is used twice: once as an upper bound on the number of iterations ℓ, and once as the number of rows/columns read into memory, which determines $\delta(G_k)$ and $\varepsilon(W,k)$. Plug in $\delta(G_k) = \Theta(\frac{1}{k})$ by Lemma 4.4.6 and $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}\rho(W))$ by Lemma 4.4.8. We get that $k_0 = O(n/\sqrt{k_0\rho(W)})$, which means that the optimal k_0 is at most $O(n^{2/3}/\rho(W)^{1/3})$. Hence for every

$$k = \Omega\left(\frac{n^{2/3}}{\rho(W)^{1/3}}\right) , \tag{4.2}$$

Verify Once makes only small 1-sided error. The algorithm Matrix Verification does not know $\rho(W)$ and k_0, but it instead tries different values of k from the exponentially increasing sequence λ^i.

Let $T_i = O(\lambda^i n)$ denote the running time of the ith call to Verify Once, and let L be the number of the last call. The expected running time can be written as a sum

$$\mathrm{E}\,[T] = \sum_{j=0}^{\infty}\left(\sum_{i=0}^{j} T_i\right)\cdot \Pr\,[L=j] = \sum_{i=0}^{\infty} T_i \cdot \Pr\,[L \geq i] \ .$$

Each call after $k \geq k_0$ fails with probability $\gamma \leq \frac{7}{8}$, so

$$\begin{aligned}\mathrm{E}\,[T] = \sum_{i=0}^{\infty}\lambda^i n \cdot \Pr\,[L\geq i] &\leq \sum_{i=0}^{(\log_\lambda k_0)-1}\lambda^i n\cdot 1 + \sum_{i=\log_\lambda k_0}^{\infty}\lambda^i n\cdot\gamma^{i-\log_\lambda k_0}\\ &\leq O(k_0 n)\left(1+\sum_{i=0}^{\infty}(\lambda\gamma)^i\right) = O\left(\frac{n^{5/3}}{\rho(W)^{1/3}}\right) ,\end{aligned}$$

because $\lambda\gamma < \frac{8}{7}\cdot\frac{7}{8} = 1$. The probability that a wrong product is never recognized is at most $\gamma^9 < \frac{1}{3}$, where 9 is the number of calls after $k \geq n^{2/3}$.

Matrix Verification never makes an error when $AB = C$. In this case, the phase flip is never performed. The diffusion is equal to identity on the uniform superposition, hence the whole quantum walk in Verify Once does nothing and the control register stays untouched. Finally, Matrix Verification always terminates when $k \geq \lambda^9 n^{2/3}$, hence its worst-case running time is $O(n^{5/3})$. □

4.4.3 Comparison with other quantum walk algorithms

Matrix Verification resembles a few other quantum algorithms. The first quantum algorithm of this type was the quantum walk algorithm for element distinctness [Amb04]. The same technique was subsequently successfully applied to triangle finding [MSS05] and group commutativity testing [MN05]. The triangle algorithm walks on the Johnson graph $J(n, k)$ and the commutativity algorithm walks on a product of two Johnson graphs. The analysis of Ambainis [Amb04] relies on the fact that the quantum state stays in a constant-dimensional subspace. This constraint is satisfied if there is at most one solution; then the subsets can be divided into a constant number of cases. In the non-promise version, the number of cases is, however, not constant. The latter papers [Amb04, MSS05] solve the non-promise case by running the algorithm on a random subset of the input. With high probability, there is exactly one solution in the subset; this trick originally comes from Valiant and Vazirani [VV86]. Since it is not known whether this technique can be used in more than one dimension, we solve the non-promise version of matrix verification using the more general quantum walk by Szegedy [Sze04] instead of the original one by Ambainis [Amb04].

Had we not multiplied the equation by random vectors from both sides, the running time of the algorithm would be much bigger. It seems that this slowdown "time $\gg$ #queries" is a typical property of algorithms using quantum walks: the quantum algorithm for element distinctness [Amb04] needs to use random hash functions to remedy it, and it is open whether triangle finding [MSS05] can be improved this way.

Theorem 1.4.6 by Szegedy [Sze04] allows us to analyze directly the general case with many wrong entries. On the other hand, the algorithm Verify Once obtained by it is a bit slower than the original Ambainis walk [Amb04, MSS05]. First, Verify Once only solves the decision version of the problem and it does not find the actual position of a wrong entry. We will resolve it by binary search. Second, Verify Once does a phase flip after every step of quantum walk instead of doing it once per block of steps. For both element distinctness and matrix verification, the additional cost is subsumed by the cost of the quantum walk, but this is not the case for triangle finding.

4.5 Fraction of marked pairs

In this section, we address the following combinatorial problem.

4.5.1. PROBLEM. *Given an $n \times n$ Boolean matrix W and two integers $1 \leq r, s \leq n$, what is the probability $\varepsilon(W, r, s)$ that a random $r \times s$ sub-matrix of W contains a 1? Or, equivalently: Given a bipartite graph on n, n vertices, what is the probability that a randomly chosen induced subgraph with r, s vertices contains at least one edge?*

It is simple to prove that $\varepsilon(W, r, s)$ is monotone in all its three parameters. As we have seen in Theorem 4.4.9, the expected running time of MATRIX VERIFICATION depends on the fraction of marked pairs, which is $\varepsilon(W, k, k)$, also denoted there by $\varepsilon(W, k)$. Our algorithm only tries balanced choices $r = s = k$. Since the initialization costs $O((r + s)n)$, setting one of the variables smaller would not decrease the query complexity, but it would decrease the success probability of VERIFY ONCE. Let us compute ε when W contains exactly one 1:

$$\varepsilon(W, r, s) = \binom{n-1}{r-1}\binom{n-1}{s-1} \Big/ \binom{n}{r}\binom{n}{s} = \frac{rs}{n^2} \; .$$

With this bound, monotonicity, and Theorem 4.4.9, we conclude that MATRIX VERIFICATION finds the correct answer with bounded error in time $O(n^{5/3})$. The rest of this section contains a more detailed analysis of the expected running time of the algorithm for larger W. Unfortunately, we are only able to prove weak lower bounds on ε for general W. However, if one improves them, then we automatically get an improved upper bound on the running time of the same algorithm.

The average probability over matrices W with t ones is $\mathrm{E}_{W:|W|=t}\left[\varepsilon(W, r, s)\right] = \Omega(t\frac{rs}{n^2})$ (Lemma 4.5.2). We prove the same bound for all W with $|W| \leq \sqrt{n}$ (Lemma 4.5.3), when the ones in W form an independent set (Lemma 4.5.4), or when the ones in W form a rectangle (again Lemma 4.5.3). However, the latter rectangle bound only holds for a limited class of r, s, which does not include the balanced case $r = s = k$ in the range used by our algorithm. As a consequence, if the ones in W form a full row or a column, our algorithm is slower than what one might expect from this formula. We, however, show that in this case our algorithm is optimal (Theorem 4.5.5); this is the only known tight lower bound for our algorithm. The main Lemma 4.4.8, which we prove here, is a corollary of Lemma 4.5.3 and Lemma 4.5.4.

4.5.1 Special cases

The following lemma is not needed for the analysis of MATRIX VERIFICATION. We include it, because it gives an insight into what may be the true bound on $\varepsilon(W, r, s)$ for most W.

4.5.2. LEMMA. *Let $rs \leq \frac{n^2}{t}$. Then $\mathrm{E}_{W:|W|=t}\left[\varepsilon(W, r, s)\right] = \Omega(t\frac{rs}{n^2})$.*

Proof. Consider the following random variables: $V_{R,S,W} = 1$ if and only if $W \cap R \times S \neq \emptyset$. Then for every pair (R,S), it holds that $\mathrm{E}_{W:|W|=t}\left[V_{R,S,W}\right] = \Pr_W\left[W \cap R \times S \neq \emptyset\right]$ and, when $|R| = r$ and $|S| = s$,

$$\begin{aligned}\Pr_{W:|W|=t}\left[W \cap R \times S \neq \emptyset\right] &= 1 - \frac{n^2 - rs}{n^2} \cdot \frac{n^2 - rs - 1}{n^2 - 1} \cdots \frac{n^2 - rs - t + 1}{n^2 - t + 1} \\ &\geq 1 - \left(\frac{n^2 - rs}{n^2}\right)^t \geq 1 - e^{-\frac{rs}{n^2}t} = \Omega\left(t\frac{rs}{n^2}\right) ,\end{aligned}$$

because $1 - x \leq e^{-x}$ and, on any fixed interval $x \in [0, A]$, also $e^{-x} \leq 1 - \frac{1-e^{-A}}{A}x$. The claim is now proved using standard arguments. Since $\forall R, S : \mathrm{E}_W\left[V_{R,S,W}\right] \geq t\frac{rs}{n^2}$, also $\mathrm{E}_{R,S}\left[\mathrm{E}_W\left[V_{R,S,W}\right]\right] \geq t\frac{rs}{n^2}$. Exchange the order of summation and obtain $\mathrm{E}_W\left[\mathrm{E}_{R,S}\left[V_{R,S,W}\right]\right] = \mathrm{E}_W\left[\varepsilon(W, r, s)\right] \geq t\frac{rs}{n^2}$. □

4.5.3. Lemma. *Let w be the number of nonzero rows of W, and let v be the maximal number of nonzero entries in a row. Then for every $r \leq \frac{n}{w}$ and $s \leq \frac{n}{v}$, $\varepsilon(W, r, s) = \Omega(|W|\frac{rs}{n^2})$.*

Proof. Let Z denote the event "$W \cap R \times S \neq \emptyset$". For $j = 0, 1, \ldots, w$, let Z_j denote the event "$W \cap R \times [n]$ has exactly j nonempty rows". Since the events $\{Z_j\}$ are disjoint and $\sum_{j=0}^{w} \Pr\left[Z_j\right] = 1$, we decompose the probability

$$\Pr\left[Z\right] = \sum_{j=0}^{w} \Pr\left[Z_j\right] \cdot \Pr\left[Z \mid Z_j\right] \geq \sum_{j=1}^{w} \Pr\left[Z_j\right] \cdot \Pr\left[Z \mid Z_1\right] = (1 - \Pr\left[Z_0\right]) \cdot \Pr\left[Z \mid Z_1\right] ,$$

because $\Pr\left[Z \mid Z_j\right] \geq \Pr\left[Z \mid Z_1\right]$ for $j \geq 1$; when we replace an empty row by a nonempty, the probability can only increase. Now, $\Pr\left[Z_0\right] = \Pr\left[W \cap R \times [n] = \emptyset\right] = \frac{n-w}{n} \cdot \frac{n-w-1}{n-1} \cdots \frac{n-w-r+1}{n-r+1} \leq (\frac{n-w}{n})^r = (1 - \frac{w}{n})^r \leq e^{-\frac{rw}{n}}$, because $1 - x \leq e^{-x}$. For every constant A, it holds that $e^{-x} \leq 1 - \frac{1-e^{-A}}{A}x$ on $x \in [0, A]$. By our assumption, $r \leq \frac{n}{w}$, hence $\frac{rw}{n} \leq 1$ and $e^{-\frac{rw}{n}} \leq 1 - \frac{1-e^{-1}}{1}\frac{rw}{n} = 1 - \alpha\frac{rw}{n}$ for $\alpha = 1 - e^{-1}$. We conclude that $1 - \Pr\left[Z_0\right] \geq \alpha\frac{rw}{n}$.

To lower-bound the other term, we decompose Z_1. For $i = 1, 2, \ldots, n$, let Y_i denote the event "$W \cap R \times [n]$ has the ith row nonempty and all other rows are empty". Let w_i be the number of entries in the ith row of W. Since $\{Y_i\}$ are disjoint and $Y_1 \cup \cdots \cup Y_n = Z_1$,

$$\Pr\left[Z \mid Z_1\right] = \sum_{i:w_i \neq 0} \Pr\left[Z \mid Y_i\right] \cdot \Pr\left[Y_i \mid Z_1\right] = \frac{1}{w} \sum_{i:w_i \neq 0} \Pr\left[Z \mid Y_i\right] .$$

$\Pr\left[Z \mid Y_i\right]$ is easy to evaluate, since the ith row of W contains w_i entries and S is picked uniformly at random. Let W_i be the ith row of W. By the same arguments as above, $\Pr\left[Z \mid Y_i\right] = \Pr\left[W_i \cap S \neq \emptyset\right] = 1 - \Pr\left[W_i \cap S = \emptyset\right] \geq 1 - e^{-\frac{sw_i}{n}}$. By our assumption, $s \leq \frac{n}{w_i}$, hence $e^{-\frac{sw_i}{n}} \leq 1 - \alpha\frac{sw_i}{n}$ and $\Pr\left[Z \mid Y_i\right] \geq \alpha\frac{sw_i}{n}$.

Put both lower bounds together and obtain

$$\Pr[Z] \geq \alpha \frac{rw}{n} \cdot \frac{1}{w} \sum_i \alpha \frac{sw_i}{n} = \alpha^2 \frac{rs}{n^2} \sum_i w_i = \Omega\left(|W| \frac{rs}{n^2}\right) ,$$

which is what we had to prove. □

4.5.4. Lemma. *Let W have at most one entry in every row and column. Then for every r, s satisfying $rs \leq n^{4/3}/|W|^{2/3}$, $\varepsilon(W, r, s) = \Omega(|W| \frac{rs}{n^2})$.*

Proof. Let $w = |W|$. If $w \leq \sqrt{n}$, then the result follows from Lemma 4.5.3 as follows. We are promised that $\min(r, s) \leq n^{2/3}/w^{1/3}$, which is at most $\frac{n}{w}$ for $w \leq \sqrt{n}$. The parameters in that lemma for our set W are $w = |W|$ and $v = 1$, hence the bound $\Omega(|W| \frac{rs}{n^2})$ we have to prove holds for all $r \leq \frac{n}{w}$ and all s. If $r \leq s$, then we are done. If $r > s$, then we observe that there is nothing special about rows in Lemma 4.5.3 and that the lemma holds also for columns. Hence our bound holds also for all $s \leq \frac{n}{w}$ and all r.

Let us assume that $w > \sqrt{n}$. Let Z denote the event "$W \cap R \times S \neq \emptyset$" and, for $j = 0, 1, \ldots, r$, let Z_j denote the event "$W \cap R \times [n]$ has exactly j nonempty rows". Then

$$\begin{aligned} 1 - \Pr[Z \mid Z_j] = \Pr[W \cap R \times S = \emptyset \mid Z_j] &= \frac{n-s}{n} \cdot \frac{n-s-1}{n-1} \cdots \frac{n-s-j+1}{n-j+1} \\ &\leq \left(\frac{n-s}{n}\right)^j \leq e^{-\frac{sj}{n}} . \end{aligned}$$

Since $j \leq r$ and $w > \sqrt{n}$, we get that $js \leq rs \leq n^{4/3}/w^{2/3} \leq n^{4/3}/(\sqrt{n})^{2/3} = n$. Hence $\frac{sj}{n} \leq 1$ and by upper-bounding the exponential we get that $\Pr[Z \mid Z_j] \geq 1 - e^{-\frac{sj}{n}} \geq 1 - (1 - \alpha \frac{sj}{n}) = \alpha \frac{sj}{n}$ for $\alpha = 1 - e^{-1}$. Now, $\{Z_j\}$ are disjoint and $\sum_{j=0}^r \Pr[Z_j] = 1$, hence we decompose the probability

$$\Pr[Z] = \sum_{j=0}^r \Pr[Z \mid Z_j] \cdot \Pr[Z_j] \geq \sum_{j=0}^r \alpha \frac{sj}{n} \Pr[Z_j] = \alpha \frac{s}{n} \sum_{j=0}^r j \cdot \Pr[Z_j] = \alpha \frac{s}{n} \cdot \mathrm{E}[Y] ,$$

where Y is the number of nonempty rows. There are r rows among n in R and we pick w entries without returning uniformly at random. An application of $\mathrm{E}[Y] = \frac{rw}{n}$ completes the proof. □

4.5.2 Proof of the main lemma

Proof of Lemma 4.4.8 Let us prove that $\varepsilon(W, k) = \Omega(\frac{k^2}{n^2} \min(|W|, \sqrt{n}))$. First, assume that $|W| \leq \sqrt{n}$, and let us verify the restrictions on $r = s = k$. For every $t \le \sqrt{n}$ it holds that $n^{2/3}/t^{1/3} \le n/t$. Hence if $|W| \le \sqrt{n}$, then for every $k \le n^{2/3}/|W|^{1/3}$ it holds that $k \le n/|W|$ and, since $v, w \le |W|$, also $k \le \frac{n}{w}$ and $k \le \frac{n}{v}$. Hence the lower bound $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}|W|)$ given by Lemma 4.5.3 holds for every k in the range required by Lemma 4.4.8. Now, if $|W| > \sqrt{n}$, the bound follows from the monotonicity of $\varepsilon(W,k)$ in W.

Lemma 4.5.4 says that $\varepsilon(W',k) = \Omega(\frac{k^2}{n^2}|W'|)$ for every independent W' and k in the range required by Lemma 4.4.8. The bound on W follows from the monotonicity of $\varepsilon(W,k)$ in W. If we put these two bounds together, we obtain that $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}\rho(W))$, as desired. □

4.5.3 The bound is tight

The bound cannot be strengthened to $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}|W|)$ for a general W in the full range of k. We show that *no* quantum algorithm can be fast if the n ones in W form a full row.

4.5.5. Theorem. *Any bounded-error quantum algorithm distinguishing a correct matrix product and a matrix product with one wrong row has query complexity* $\Omega(n^{3/2})$.

Proof. We reduce OR of n parities of length n to matrix verification. Let

$$z = (x_{1,1} \oplus \dots \oplus x_{1,n}) \vee \dots \vee (x_{n,1} \oplus \dots \oplus x_{n,n}) \ .$$

By the unweighted adversary method (Theorem 2.4.2), computing z requires $\Omega(n^{3/2})$ quantum queries, and the lower bound holds even if we promise that at most one of the parities is equal to 1. Since $z = 1$ if and only if $\exists i : 0 \neq \bigoplus_{\ell=1}^n x_{i,\ell}$, one can reduce this problem to the matrix verification $AB = C$ over $\mathbb{GF}(2)$, where $A[i,j] = x_{i,j}$, $B[i,j] = 1$, and $C[i,j] = 0$. The promise is transformed into that at most one row is wrong. □

A straightforward calculation shows that $\rho(W)$ for a W that has all n wrong entries in one row can be at most $O(\sqrt{n})$ if we want the bound on ε to hold for all k in the interval required by Lemma 4.4.8. We have a lower bound $\Omega(n^{3/2})$ for this problem and we want to find a value of $\rho(W)$ such that the verification can be done in time $O(n^{5/3}/\rho(W)^{1/3})$. It follows that $\rho(W) = O(\sqrt{n})$.

4.6 Algorithm for matrix multiplication

Let $m \ge n^{2/3}$. One can modify the algorithm Matrix Verification to verify the product $A_{n\times m}B_{m\times n} = C_{n\times n}$ in time proportional to $n^{2/3}m$. The quantum walk stays the same and only the inner scalar products have dimension m instead of n. Using the new algorithm and binary search, we construct a quantum algorithm that outputs the position of a wrong entry. By iterating this and correcting the wrong entries, we compute the matrix product $AB = C$ whenever a good approximation to C is known. The algorithms are described in Figure 4.3.

4.6.1. Theorem. Find Wrong Entry *has 1-sided polynomially small error, worst-case running time $O(n^{2/3}m \log n)$, and its expected running time is* $\mathrm{O}(n^{2/3}m \log n/\rho(W)^{1/3})$ *when the set of wrong entries is W.*

Proof. Assume that $AB \neq C$. Let W^ℓ be the set of wrong entries in the ℓ^{th} recursion level of the binary search. From the definition of $\rho(W)$, if $\rho(W^\ell) = \rho$, then $\rho(W^\ell_{i,j}) \geq \frac{\rho}{4}$ for at least one quadrant $i, j \in \{1, 2\}$. Find Wrong Entry descends into the first quadrant it finds a solution in, hence it chooses $W^{\ell+1} = W^\ell_{i,j}$ with high probability and then $(\frac{n}{2})^2/\rho(W^{\ell+1}) \leq n^2/\rho(W^\ell)$. There are $\log n$ levels of recursion. Hence its expected running time is at most

$$\sum_{\ell=1}^{\log n} 4\sqrt[3]{\frac{(n/2^\ell)^2}{\rho(W^\ell)}}m \leq \sum_{\ell=1}^{\log n} 4\sqrt[3]{\frac{n^2}{\rho(W)}}m = O\left(\frac{n^{2/3}m}{\rho(W)^{1/3}} \log n\right) ,$$

as claimed. By Theorem 4.4.9, the probability that a wrong matrix product is not recognized in one iteration is at most $\frac{1}{3}$. The probability that it is not recognized in $O(\log n)$ iterations is $1/\mathrm{poly}(n)$. If $AB = C$, then the first iteration of binary search is repeated $O(\log n)$ times and the worst-case running time is $O(n^{2/3}m \log n)$. □

4.6.2. Remark. It might be that the position of the wrong entry can be obtained from just one call to Matrix Verification in the same way as in the quantum walk algorithm Element Distinctness—by measuring the subsets R, S instead of performing the scalar product test (Lemma 1.4.7) on the control register. However, this is only known to follow from Theorem 1.4.6 for exactly one wrong entry, that is when $|W| = 1$ [Sze04, Section 10]. The logarithmic factor in the total running time is necessary for having a polynomially small error.

One can always start by guessing $C = 0$, hence the following bound holds.

4.6.3. Theorem. *Let $m \geq n^{2/3}$. The matrix product $A_{n\times m}B_{m\times n} = C_{n\times n}$ can be computed with polynomially small error probability in expected time*

$$T_M \leq O(1) \cdot \begin{cases} m \log n \cdot n^{2/3}w^{2/3} & \text{if } 1 \leq w \leq \sqrt{n} \\ m \log n \cdot \sqrt{n}w & \text{if } \sqrt{n} \leq w \leq n \\ m \log n \cdot n\sqrt{w} & \text{if } n \leq w \leq n^2 \ , \end{cases} \tag{4.3}$$

where W is the set of nonzero entries of C and $w = |W|$.

```
MATRIX MULTIPLICATION (input matrices $A_{n\times m}, B_{m\times n}$)
returns $C_{n\times n} = AB$.

 1. Initialize $C = 0$.
 2. Run FIND WRONG ENTRY $(A, B, C)$.
    If it returns "equal", return $C$.
 3. Otherwise let $(r, c)$ be the wrong position. Recompute $C[r, c]$. Find
    and recompute all wrong entries in the $r^{\text{th}}$ row of $C$ using GROVER
    SEARCH. Find and recompute all wrong entries in the $c^{\text{th}}$ column
    of $C$ using GROVER SEARCH.
 4. Go back to step 2.

FIND WRONG ENTRY (input matrices $A_{n\times m}, B_{m\times n}, C_{n\times n}$)
returns a position $(r, c)$ if $C[r, c] \neq (AB)[r, c]$, or "equal" if $AB = C$.

 5. If $n = 1$, verify the scalar product and exit.
 6. Let $A_1, A_2$ denote the top and bottom half of $A$,
    let $B_1, B_2$ denote the left and right half of $B$, and
    let $C_{1,1}, C_{1,2}, C_{2,1}, C_{2,2}$ denote the four quadrants of $C$.
 7. Repeat at most $O(\log n)$ times the following:
    Run in parallel MATRIX VERIFICATION $(A_i, B_j, C_{i,j})$ for $i, j \in$
    $\{1, 2\}$. As soon as one of them returns "not equal", stop the other
    branches of computation and cancel the loop.
 8. If MATRIX VERIFICATION always output "equal", then return
    "equal".
 9. Let $C_{i,j} \neq A_i B_j$ be the wrong sub-matrix found. Let $(r', c') =$
    FIND WRONG ENTRY $(A_i, B_j, C_{i,j})$.
10. If $i = 1$, set $r = r'$, otherwise set $r = r' + \frac{n}{2}$.
    If $j = 1$, set $c = c'$, otherwise set $c = c' + \frac{n}{2}$.
    Return $(r, c)$.
```

Figure 4.3: Quantum algorithm MATRIX MULTIPLICATION

PROOF. Finding all r_ℓ wrong entries in the ℓ^{th} row is done by iterating GROVER SEARCH. By Corollary 1.3.5, it takes time $\sum_{i=1}^{r_\ell} \sqrt{\frac{n}{i}} m = O(\sqrt{nr_\ell} m)$, where the m-dimensional scalar products are computed on-line. We ensure that there are no wrong entries left with probability polynomially close to one in additional time $O(\sqrt{n} m \log n)$. Let us condition the rest of the analysis on the event that all quantum searches indeed find all ones.

Let w' be the largest possible size of an independent subset of W. Clearly, MATRIX MULTIPLICATION finishes in at most w' iterations, because the algorithm in every iteration first finds a wrong entry and then recomputes all wrong entries of C in the same row and the same column. If the number of iterations was more than w', then the starting wrong entries of these iterations would form an independent set larger than w'. The total running time is the sum of the time spent in FIND WRONG ENTRY

$$T_F \le \sum_{\ell=1}^{w'} \frac{n^{2/3} m \log n}{\ell^{1/3}} = O((nw')^{2/3} m \log n) \ ,$$

and the time T_G spent in GROVER SEARCH. By applying the Cauchy-Schwarz inequality several times,

$$\begin{aligned} T_G \le \sum_{\ell=1}^{w'} (\sqrt{nr_\ell} m + \sqrt{nc_\ell} m) \log n &= m\sqrt{n} \log n \left(\sum_{\ell=1}^{w'} 1 \cdot \sqrt{r_\ell} + \sum_{\ell=1}^{w'} 1 \cdot \sqrt{c_\ell} \right) \\ &\le m\sqrt{n} \log n \sqrt{\sum_{\ell=1}^{w'} 1} \left(\sqrt{\sum_{\ell=1}^{w'} r_\ell} + \sqrt{\sum_{\ell=1}^{w'} c_\ell} \right) \\ &= O(m\sqrt{n} \log n \sqrt{w'} \sqrt{w}) \ . \end{aligned}$$

The algorithm has bounded error, because both FIND WRONG ENTRY and iterated quantum searches have polynomially small error. Put the bounds together and obtain

$$T_M = T_F + T_G \le m \log(n) \sqrt{n} \sqrt{w'} \cdot (n^{1/6} (w')^{1/6} + \sqrt{w}) \ .$$

Evaluate separately three cases $w \in [1, \sqrt{n}]$, $w \in [\sqrt{n}, n]$, and $w \in [n, n^2]$, use that $w' \le w$ and $w' \le n$, and obtain inequality (4.3), which we had to prove. □

Let us compare our algorithm to the fastest known classical algorithms. Let $w = |W|$ be the number of nonzero entries of C and let $\kappa = \sqrt{n}/(\log n)^{3/2}$. Our algorithm computes sparse matrix products with $w = o(\kappa)$ in time $o(nm)$. Classically, *dense* square matrix products can be computed in time $O(n^{2.376})$ [CW90], which is faster than our algorithm for $w = \Omega(n^{0.876})$, and dense rectangular matrix products can be computed in time $O(n^{1.844+o(1)} m^{0.533} + n^{2+o(1)})$ [Cop97]. Recently, Indyk discovered a classical algorithm for sparse rectangular matrix products running in time $\tilde{O}(m(n+w))$ [Ind05], which is faster than ours for $w = \Omega(\kappa)$. The fastest known algorithm for sparse square *input* matrices runs in time $O(n^{1.2}z^{0.7} + n^{2+o(1)})$ [YZ04], where A, B each have at most z nonzero elements.

4.7 Boolean matrix verification

The algorithm Verify Once relies on the fact that arithmetical operations are over some integral domain. If the matrices are over the Boolean algebra $\{\vee, \wedge\}$, then the multiplication by random vectors from both sides does not work. However, Boolean matrix products can be verified even faster by the following simple algorithm.

4.7.1. Theorem. *There exists a quantum Boolean matrix verification algorithm running in time $O(n\sqrt{m})$ and space $O(\log n + \log m)$.*

Proof. The condition that three given matrices form a valid product can be written as an AND-OR tree with an AND of n^2 equalities, each being an OR of m products. There is a bounded-error quantum algorithm [HMW03] running in time $O(\sqrt{n^2 m}) = O(n\sqrt{m})$ and space $O(\log(n^2 m))$. □

Using the technique from Theorem 1.3.3, one can speed up the verification to time $O(n\sqrt{m/t})$, if the number of wrong entries t is known beforehand. If t is unknown, then the verification can be done in *expected time* $O(n\sqrt{m/t})$ and the worst-case time stays $O(n\sqrt{m})$. The Boolean matrix product with t nonzero entries can thus be computed in expected time $O(n\sqrt{tm})$.

4.8 Summary

We have presented two new quantum algorithms based on quantum random walks, one for matrix verification and one for matrix multiplication. Let us close with a few open problems.

It would be interesting to strengthen the lower bound on the fraction $\varepsilon(W, k)$ of marked pairs and thus also the upper bound on the running time of Matrix Verification when there are many wrong entries. As we have shown, this cannot be done in full generality, but perhaps one can show a stronger bound in terms of the density of wrong entries.

Can one prove a better lower bound on matrix verification than $\Omega(n^{3/2})$? This lower bound is tight when there are $\sqrt{n}$ wrong entries. Is the true bound higher with only one wrong entry? Due to the small certificate complexity of this problem, one cannot prove such a bound using the adversary method (see Theorem 2.5.3), but it might be provable by the polynomial method [BBC+01].

Part II

Lower Bounds

Chapter 5
Adversary Lower Bounds

This chapter is based on the following papers:

[ŠS06]	R. Špalek and M. Szegedy. All quantum adversary methods are equivalent. *Theory of Computing*, 2(1):1–18, 2006. Earlier version in *Proceedings of 32nd International Colloquium on Automata, Languages and Programming*, pages 1299-1311, 2005. Lecture Notes in Computer Science 3580.
[HŠ05]	P. Høyer and R. Špalek. Lower bounds on quantum query complexity. *Bulletin of the European Association for Theoretical Computer Science*, 87:78–103, October 2005.
[HLŠ05]	P. Høyer, T. Lee, and R. Špalek. Tight adversary bounds for composite functions. Technical report. quant-ph/0509067, 2005.

5.1 Introduction

In this chapter, we examine the adversary bound of a function. This quantity was known before under many different names as a (very versatile) lower bound on the quantum query complexity of the function [BSS03, Amb03, LM04, Zha05]. We show that all these lower bounds are equal and hence we can use the name *adversary bound* for all of them. The adversary bound is actually an interesting quantity on its own. It has both a primal and a dual representation and so it can be easily lower-bounded and upper-bounded, it possesses tight composition properties, and perhaps most remarkably it is useful even beyond quantum computing. Indeed, Laplante, Lee, and Szegedy [LLS06] showed that its square is a lower bound on the formula size of the function. It can be shown that its logarithm is a lower bound on the circuit depth with bounded fan-in [LŠ05]. We also investigate limitations of the adversary bound, and show that it is limited if the function has small certificates.

History The intuition behind adversary lower bounds has already been explained in Section 2.2. Let us review here the history of the bounds. Bennett, Bernstein, Brassard, and Vazirani [BBBV97] showed a tight lower bound for quantum search, and called it a *hybrid method.* A generalization of this method that is applicable to all functions, the original *unweighted* version of the adversary bound, was invented by Ambainis [Amb02]. This version is very simple to apply as it only requires to specify a hard set of input pairs; see Theorem 2.4.2. Despite its simplicity it gives tight lower bounds for many computational problems. The first attempt to allow *weights* on the input pairs was due to Høyer, Neerbek, and Shi [HNS02] who obtained a tight lower bound on binary search that could not be obtained by the unweighted method. Barnum and Saks used similar arguments [BS04] for read-once formulas. The two latter bounds were, however, specialized to narrow classes of functions. Shortly afterwards, Barnum, Saks, and Szegedy [BSS03] and Ambainis [Amb03] independently discovered a weighted version of the adversary bound applicable to all functions; see Theorem 2.3.1 and Theorem 2.4.4. Barnum et al. got their bound as a byproduct of their characterization of the quantum query complexity in terms of semidefinite programs. Ambainis used his bound to separate the polynomial degree and the quantum query complexity; he exhibited iterated functions with low exact degree and high weighted adversary bound. Laplante and Magniez [LM04] then came up with a Kolmogorov complexity version of the dual of the adversary bound, and finally Zhang [Zha05] generalized Ambainis's weighted bound.

Relations between the methods The primordial hybrid method [BBBV97] is a special case of the unweighted adversary method [Amb02], which is a special case of the weighted adversary method [Amb03], which can be further generalized into the strong weighted method [Zha05]. The early extensions of the unweighted adversary method allowing weights [HNS02, BS04] are also special cases of the general weighted adversary method. Laplante and Magniez [LM04] showed by direct constructions that the Kolmogorov complexity method is at least as strong as any of the methods above.

We extend their work and prove that the spectral method, the weighted method, and the generalization of the weighted method are at least as strong as the Kolmogorov complexity method, allowing us to conclude that all these methods [BSS03, Amb03, LM04, Zha05] are equivalent. We also propose the following simple combinatorial version of the Kolmogorov complexity method that is stated using only algebraic terms and that is also equivalent with the methods above. Having such a variety of representations of the same method shows that the adversary method is very robust, and it captures fundamental properties of functions.

5.1.1. Theorem (Minimax method [LM04, ŠS06]). *Let $f : S \to \Sigma'$ be a function with domain $S \subseteq \Sigma^n$, and let A be a bounded-error quantum algorithm* *for f. Let $p : S \times [n] \to \mathbb{R}_0^+$ be a set of $|S|$ probability distributions over the input variables such that $p_x(i)$ denotes the average square amplitude of querying the i^{th} input variable on input x, where the average is taken over the whole computation of A. Let*

$$\mathrm{MM}_p(f) = \max_{x,y:f(x)\neq f(y)} \frac{1}{\sum_{i:x_i\neq y_i} \sqrt{p_x(i)\, p_y(i)}} \ .$$

Then the query complexity of algorithm A satisfies $Q_{\mathsf{A}} \geq \mathrm{MM}_p(f)$.

The adversary methods introduced in Chapter 2, such as the spectral method or the unweighted method, satisfy the property that if we plug in some matrix or relation, we get a lower bound that is valid for all quantum algorithms. The minimax method is principally different. A lower bound computed by the minimax theorem holds for one particular algorithm A, and it may not hold for some other and better algorithm. However, we may obtain a universal lower bound that holds for *every* bounded error algorithm by simply taking the minimum of the bound $\mathrm{MM}_p(f)$ over all possible sets of probability distributions p.

5.1.2. Definition. Let $f : S \to \Sigma'$ be a function with domain $S \subseteq \Sigma^n$. The *minimax bound for f* is defined as

$$\mathrm{MM}(f) = \min_p \mathrm{MM}_p(f) \ ,$$

where $p : S \times [n] \to \mathbb{R}_0^+$ ranges over all sets of probability distributions.

The spectral adversary bound from Definition 2.5.1 and the minimax bound from Definition 5.1.2 are in a *primal-dual relation.* We show below in Section 5.3 that the best lower bound that can be obtained by any adversary matrix Γ equals the smallest bound that can be obtained by a set of probability distributions p. Primal methods are used for obtaining concrete lower bounds and dual methods are used for proving limitations of the method.

Non-Boolean functions The early variants of the adversary method were stated only for Boolean functions, those are functions of the form $f : S \to \{0, 1\}$ with domain $S \subseteq \{0, 1\}^n$. It turned out that these were just technical limitations of the definitions and proofs. Laplante and Magniez [LM04] were the first to state their Kolmogorov complexity bound for functions with arbitrary input and output alphabets. It is quite straightforward to generalize the other variants of the adversary method to this setting. In fact, the proof of the spectral adversary lower bound we have given in Theorem 2.3.1 [HŠ05] holds for all functions $f : S \to \Sigma'$ with domain $S \subseteq \Sigma^n$, where Σ, Σ' are arbitrary finite alphabets. Our equivalence result from Section 5.3 also holds in the generalized setting.

Limitations of the method The lower bounds obtained by the adversary method are limited as follows. Consider a Boolean function f. Szegedy [Sze03] observed that the weighted adversary method is limited by $\min(\sqrt{\mathrm{C}_0 n}, \sqrt{\mathrm{C}_1 n})$, where C_0 is the zero-certificate complexity of f and C_1 is the one-certificate complexity of f (see Definition 2.5.2 above). Laplante and Magniez proved the same limitation for the Kolmogorov complexity method [LM04], which implies that all other adversary methods are also bounded. Finally, this bound was improved to $\sqrt{\mathrm{C}_0 \mathrm{C}_1}$ for total f by Zhang [Zha05] and independently by us.

Here we present a new simple proof of the limitation of the quantum adversary method that also holds for non-Boolean functions. Let us order the letters from the output alphabet Σ' by their certificate complexities such that $\mathrm{C}_0 \geq \mathrm{C}_1 \geq \cdots \geq \mathrm{C}_{|\Sigma'|-1}$. Then all adversary lower bounds are at most $2\sqrt{\mathrm{C}_1 n}$ for partial f and $\sqrt{\mathrm{C}_0 \mathrm{C}_1}$ for total f.

Composition properties The adversary methods yields good, and often tight, lower bounds on many computational problems. Ambainis shows in particular that the weighted method yields good (though not necessarily tight) lower bounds on iterated Boolean functions. We say that a function is iterated if it is defined iteratively from a base function $f : \{0,1\}^k \to \{0,1\}$ by $f^1 = f$ and $f^{d+1} = f \circ (f^d, \dots, f^d)$ for $d > 1$. An example iterated function is the binary AND-OR tree, where the base function $f : \{0,1\}^4 \to \{0,1\}$ is $f(x_1, x_2, x_3, x_4) = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$.

Ambainis [Amb03] shows that if we have a good adversary lower bound on the base function, we get a good adversary lower bound on the iterated functions: if $\mathrm{Adv}(f) = a$, then $\mathrm{Adv}(f^d) \geq a^d$, where $\mathrm{Adv}(f)$ was defined in Definition 2.5.1 above. Laplante, Lee, and Szegedy [LLS06] show that this is tight by proving a matching upper bound, $\mathrm{Adv}(f^d) \leq a^d$. We thus conclude that the adversary method possesses the following composition property.

5.1.3. THEOREM ([AMB03, LLS06]). *For any function $f : \{0,1\}^k \to \{0,1\}$,*

$$\mathrm{Adv}(f^d) = \mathrm{Adv}(f)^d \ .$$

A natural possible generalization of Theorem 5.1.3 is to consider composed functions that can be written in the form

$$h = f \circ (g_1, \dots, g_k) \ . \tag{5.1}$$

One may think of h as a two-level decision tree with the top node being labelled by a function $f : \{0,1\}^k \to \{0,1\}$, and each of the k internal nodes at the bottom level being labelled by a function $g_i : \{0,1\}^{n_i} \to \{0,1\}$. We do not require that the domains of the inner functions g_i have the same size. An input $x \in \{0,1\}^n$ to h is a bit string of length $n = \sum_i n_i$, which we think of as being comprised of k parts, $x = (x^1, x^2, \dots, x^k)$, where $x^i \in \{0,1\}^{n_i}$. We may evaluate h on input x by first computing the k bits $g_i(x^i)$, and then evaluating f on input $(g_1(x^1), \dots, g_k(x^k))$.

It is plausible, and not too difficult to prove, that if $a_1 \leq \mathrm{Adv}(f) \leq a_2$ and $b_1 \leq \mathrm{Adv}(g_i) \leq b_2$ for all i, then $a_1 b_1 \leq \mathrm{Adv}(h) \leq a_2 b_2$. In particular, if the adversary bounds of the sub-functions g_i are equal (that is $\mathrm{Adv}(g_i) = \mathrm{Adv}(g_j)$ for all i, j), then we can state an exact expression for the adversary bound on h in terms of the adversary bounds of its sub-functions,

$$\mathrm{Adv}(h) = \mathrm{Adv}(f) \cdot \mathrm{Adv}(g) \ ,$$

where g is any of the sub-functions g_i. However, it is not so clear what the exact adversary bound of h is when the adversary bounds of the sub-functions g_i differ.

Adversary bound with costs To formulate a composition theorem for arbitrary cases when the functions g_i may have different adversary bounds, we require as an intermediate step, a new generalization of the adversary method. For any function $f : \{0,1\}^k \to \{0,1\}$, and any vector $\alpha \in \mathbb{R}_+^k$ of dimension k of positive reals, we define below a quantity $\mathrm{Adv}_\alpha(f)$ that generalizes the adversary bound $\mathrm{Adv}(f)$.

5.1.4. Definition. Let $f : S \to \{0,1\}$ be a Boolean function with domain $S \subseteq \{0,1\}^n$, and $\alpha = (\alpha_1, \ldots, \alpha_n)$ a string of positive reals. The *adversary bound for f with costs α* is

$$\mathrm{Adv}_\alpha(f) = \max_\Gamma \min_i \left\{ \alpha_i \frac{\lambda(\Gamma)}{\lambda(\Gamma_i)} \right\} \ ,$$

where Γ ranges over all adversary matrices for f.

One may choose to think of α_i as expressing the cost of querying the ith input bit x_i, for example due to the fact that x_i is equal to the parity of $2\alpha_i$ new input bits, or, alternatively, as if each query to x_i reveals only a fraction of $\frac{1}{\alpha_i}$ bits of information about x_i. When all costs $\alpha_i = a$ are equal, the new adversary bound $\mathrm{Adv}_\alpha(f)$ reduces to $a \cdot \mathrm{Adv}(f)$, the product of a and the standard adversary bound $\mathrm{Adv}(f)$. The adversary bound $\mathrm{Adv}_\alpha(f)$ with cost vector α may not in itself be a query complexity lower bound for f when the costs α_i differ. It is, however, still a very useful quantity for give exact expressions on composed functions on the form given in (5.1) above.

5.1.5. Theorem. (Composition Theorem [BS04, Amb03, LLS06, ŠS06]) *Let h be a composite function of the form $h = f \circ (g_1, \ldots, g_k)$, where $f : \{0,1\}^k \to \{0,1\}$ and $g_i : \{0,1\}^{n_i} \to \{0,1\}$ are (possibly partial) Boolean functions. Let $\alpha \in \mathbb{R}_+^n$ be any cost vector. Then*

$$\mathrm{Adv}_\alpha(h) = \mathrm{Adv}_\beta(f) \ ,$$

where $\alpha = (\alpha^1, \ldots, \alpha^k)$ is a k-tuple of strings $\alpha^i \in \mathbb{R}_+^{n_i}$ and $\beta = (\beta_1, \ldots, \beta_k)$ is a k-tuple of adversary bounds $\beta_i = \mathrm{Adv}_{\alpha^i}(g_i)$.

By this theorem, to compute the actual adversary bound $\mathrm{Adv}(h)$, it suffices to first compute all inner adversary bounds $\beta_i = \mathrm{Adv}(g_i)$ separately, and then compute the outer adversary bound $\mathrm{Adv}_\beta(f)$ with costs β. We thus split the problem of computing an adversary bound into a number of smaller disjoint subproblems. The main limitation of our approach is that it applies only when the sub-functions g_i act on disjoint subsets of the input bits. Our theorem generalizes in particular the adversary lower bound of $\Omega(\sqrt{n})$ in [BS04] on read-once formulas of size n over the basis $\{\wedge, \vee\}$. We assume throughout this section on composition that the functions f, g_i, and h are Boolean. We do not assume that h is total—if $h = f \circ (g_1, \dots, g_k)$ is partial, the domain of f, and of each sub-function g_i, is given by the possible inputs that may be generated to them via valid inputs to h.

Structure of the chapter In Section 5.3, we prove the equivalence of most variants of the adversary method, in Section 5.4, we prove the limitation on the adversary method, and in Section 5.5, we prove the composition theorem.

5.2 Preliminaries

Kolmogorov complexity Kolmogorov complexity describes how difficult it is to produce a given finite string in terms of the length of the shortest program that writes the string to the output. An excellent book about Kolmogorov complexity is the book [LV97] by Li and Vitányi. Deep knowledge of Kolmogorov complexity is not necessary to understand this chapter. Some results on the relation between various classical forms of the quantum adversary method and the Kolmogorov complexity method are taken from Laplante and Magniez [LM04], and the others just use basic techniques. We just use the following definitions.

5.2.1. Definition. A *universal Turing machine* is a Turing machine that can simulate any other Turing machine. Consider a universal Turing machine M. The *Kolmogorov complexity of x given y with respect to M*, denoted by $C_M(x|y)$, is the length of the shortest program that prints x if it gets y on the input. Formally, $C_M(x|y) = \min\{\mathrm{length}(p) : M(p, y) = x\}$.

A set of finite strings $S \subseteq \{0,1\}^*$ is called *prefix-free* if none of its elements is a proper prefix of another element. Prefix-free sets are *self-delimiting*, that is one can concatenate any strings $s_1, s_2, \dots, s_k$ from a prefix-free set S into $s = s_1 s_2 \dots s_k$ and it will always be possible to uniquely determine the individual strings from s by reading left to right. Let M be a universal Turing machine with a prefix-free domain, that is a machine that never stops on an input $p = p_1 p_2$ if it stops on the input p_1, and p_2 is some string. We call M a *prefix-free universal Turing machine*.

We are interested in the Kolmogorov complexity of a string when programs are taken from some prefix-free domain. Naturally, such a complexity measure depends on the concrete choice of M. We say that a prefix-free machine M is a *Kolmogorov minimal element*, if for any other prefix-free machine M', there is a constant c such that

$$C_M(x|y) \leq C_{M'}(x|y) + c \qquad \text{for all strings } x, y \ .$$

In some sense, the Kolmogorov minimal element is an optimal prefix-free machine. The *invariance theorem* [LV97] says that there exists a Kolmogorov minimal element. The invariance theorem makes it possible to talk about the prefix-free complexity of an infinite *family of strings* without reference to the concrete machine M. The exact choice of the machine M only affects the result by an additive constant. However, the following definition does not make much sense for any *single string*, because the hidden additive constant can be as big as the actual complexity.

5.2.2. Definition. Let M be a Kolmogorov minimal element for the set of prefix-free universal Turing machines. The *prefix-free Kolmogorov complexity of x given y* is $K(x|y) = C_M(x|y)$.

Semidefinite programming Semidefinite programming is an extremely useful generalization of linear programming, where one allows non-linear constraints on input variables in terms of requiring that some matrix must be positive semidefinite. Most of the results and techniques of linear programming generalize to the semidefinite setting. In the section on the equivalence of adversary bounds, we use a lot the duality theory of semidefinite programming [Lov00]. There are various forms of the duality principle in the literature. We use a semidefinite extension of Farkas's lemma [Lov00, Theorem 3.4].

5.2.3. Definition. $A \geq B$ denotes the *entry-wise comparison*, formally $\forall x, y : A[x, y] \geq B[x, y]$, and $C \succeq D$ denotes that $C - D$ is *positive semidefinite*, formally $\forall v : v^T(C - D)v \geq 0$.

5.3 Equivalence of adversary bounds

In this section we present several equivalent quantum adversary methods. We categorize these methods into two groups. Some of them solve conditions on the *primal* of the quantum system [BSS03]: these are the spectral, weighted, strong weighted, and generalized spectral adversary; and some of them solve conditions on the *dual*: these are the Kolmogorov complexity bound, minimax, and the semidefinite version of minimax. Primal methods are mostly used to give lower bounds on the query complexity, while we can use the duals to prove limitations of the method.

The primal methods known before, that is the spectral, weighted, and strong weighted adversary, were originally stated only for Boolean functions. The generalization to the more general non-Boolean case is straightforward and hence we state them here in the generalized form.

5.3.1. THEOREM. *Let $f : S \to \Sigma'$ be a function with domain $S \subseteq \Sigma^n$. For an $\varepsilon \in (0, \frac{1}{2})$, let $Q_\varepsilon(f)$ denote the ε-error quantum query complexity of f. Then*

$$\begin{aligned}\tfrac{Q_\varepsilon(f)}{1-2\sqrt{\varepsilon(1-\varepsilon)}} \geq \mathrm{Adv}(f) &= \mathrm{SA}(f) = \mathrm{WA}(f) = \mathrm{SWA}(f) \\ &= \mathrm{MM}(f) = \mathrm{SMM}(f) = \mathrm{GSA}(f) = \Theta(\mathrm{KA}(f)) \ ,\end{aligned}$$

where SA, WA, SWA, MM, SMM, GSA, *and* KA *are lower bounds given by the following methods.*

- **Spectral adversary [BSS03] and Theorem 2.3.1.** *Let D_i, F denote $|S| \times |S|$ zero-one valued matrices that satisfy $D_i[x, y] = 1$ if and only if $x_i \neq y_i$ for $i \in [n]$, and $F[x, y] = 1$ if and only if $f(x) \neq f(y)$. Let Γ be an adversary matrix for f, that is an $|S| \times |S|$ non-negative symmetric matrix such that $\Gamma \circ F = \Gamma$; here $\circ$ denotes the entry-wise product of two matrices, not the composition of two functions. Then*

$$\mathrm{SA}(f) = \max_\Gamma \frac{\lambda(\Gamma)}{\max_i \lambda(\Gamma \circ D_i)} \ . \tag{5.2}$$

- **Weighted adversary [Amb03].**[1] *Let w, w' denote a weight scheme as follows:*

  - *Every pair $(x, y) \in S^2$ is assigned a non-negative weight $w(x, y) = w(y, x)$ that satisfies $w(x, y) = 0$ whenever $f(x) = f(y)$.*
  - *Every triple $(x, y, i) \in S^2 \times [n]$ is assigned a non-negative weight $w'(x, y, i)$ that satisfies $w'(x, y, i) = 0$ whenever $x_i = y_i$ or $f(x) = f(y)$, and $w'(x, y, i)w'(y, x, i) \geq w^2(x, y)$ for all x, y, i such that $x_i \neq y_i$ and $f(x) \neq f(y)$.*

  For all x, i, let $wt(x) = \sum_y w(x, y)$ and $v(x, i) = \sum_y w'(x, y, i)$. Then

$$\mathrm{WA}(f) = \max_{w,w'} \min_{\substack{x,y,\ i,j \\ f(x)\neq f(y) \\ v(x,i)v(y,j)>0}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,j)}} \ . \tag{5.3}$$

[1]We use a different formulation [LM04] than in the original Ambainis papers [Amb02, Amb03]. In particular, we omit the relation R on which the weights are required to be nonzero, and instead allow zero weights. It is simple to prove that both formulations are equivalent.

- **Strong weighted adversary [Zha05].** *Let w, w' denote a weight scheme as above. Then*

$$\mathrm{SWA}(f) = \max_{w,w'} \min_{\substack{x,y,i \\ w(x,y)>0 \\ x_i \neq y_i}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,i)}} \ . \tag{5.4}$$

- **Kolmogorov complexity [LM04].**[2] *Let $\sigma \in \{0,1\}^*$ denote a finite binary string. Then*

$$\mathrm{KA}(f) = \min_{\sigma} \max_{\substack{x,y \\ f(x)\neq f(y)}} \frac{1}{\sum_{i:x_i\neq y_i} \sqrt{2^{-K(i|x,\sigma)-K(i|y,\sigma)}}} \ . \tag{5.5}$$

- **Minimax over probability distributions [LM04].** *Let $p : S \times [n] \to \mathbb{R}$ denote a set of probability distributions, that is $p_x(i) \geq 0$ and $\sum_{i=1}^n p_x(i) = 1$ for every $x \in S$. Then*

$$\mathrm{MM}(f) = \min_{p} \max_{\substack{x,y \\ f(x)\neq f(y)}} \frac{1}{\sum_{i:x_i\neq y_i} \sqrt{p_x(i)\, p_y(i)}} \tag{5.6}$$

$$= 1 \Big/ \max_{p} \min_{\substack{x,y \\ f(x)\neq f(y)}} \sum_{i:x_i\neq y_i} \sqrt{p_x(i)\, p_y(i)} \ . \tag{5.7}$$

- **Semidefinite version of minimax.** *Let D_i, F be matrices as in the spectral adversary. Then* $\mathrm{SMM}(f) = 1/\mu_{\max}$*, where $\mu_{\max}$ is the maximal solution of the following semidefinite program:*

$$\begin{array}{ll} \text{maximize} & \mu \\ \text{subject to} & \forall i: \quad R_i \succeq 0 \\ & \sum_i R_i \circ I = I \\ & \sum_i R_i \circ D_i \geq \mu F \ . \end{array} \tag{5.8}$$

- **Generalized spectral adversary.** *Let D_i, F be matrices as in the spectral adversary. Then* $\mathrm{GSA}(f) = 1/\mu_{\min}$*, where $\mu_{\min}$ is the minimal solution of the following semidefinite program:*

$$\begin{array}{lr} \text{minimize} & \mu = \mathrm{Tr}\,\Delta \\ \text{subject to} & \Delta \text{ is diagonal} \\ & Z \geq 0 \\ & Z \cdot F = 1 \\ & \forall i: \ \Delta - Z \circ D_i \succeq 0 \ . \end{array} \tag{5.9}$$

[2]We use a different formulation than Laplante and Magniez [LM04]. They minimize over all algorithms A computing f and substitute $\sigma =$ source code of A, whereas we minimize over all finite strings σ. Our way is equivalent. One can easily argue that any finite string σ can be "embedded" into any algorithm B as follows. Let C be the source code of B with appended comment σ that is never executed. Now, the programs B and C are equivalent, and $K(x|\sigma) \leq K(x|C) + O(1)$ for every x.

Proof. The statement is a corollary of several equivalence statements proved in the following subsections.

- $\mathrm{SA}(f) = \mathrm{WA}(f) = \mathrm{SWA}(f)$ by Theorem 5.3.4 and Theorem 5.3.5,
- $\mathrm{MM}(f) = \mathrm{SMM}(f)$ by Theorem 5.3.7,
- $\mathrm{SMM}(f) = \mathrm{GSA}(f)$ by Theorem 5.3.8,
- $\mathrm{GSA}(f) = \mathrm{SA}(f)$ by Theorem 5.3.9,
- $\mathrm{KA}(f) = \Theta(\mathrm{WA}(f))$ by Theorem 5.3.10 and Theorem 5.3.11.

This justifies the definition of the new quantity, the *adversary bound of f*, that is equal to all these bounds. Finally, one has to prove that $\mathrm{Adv}(f)$ is a quantum query lower bound. Historically, Ambainis [Amb03] was the first to prove that $Q_\varepsilon(f) \geq (1 - 2\sqrt{\varepsilon(1-\varepsilon)})\,\mathrm{WA}(f)$ for every Boolean f. Laplante and Magniez proved [LM04] that $Q_2(f) = \Omega(\mathrm{KA}(f))$ for general f. In Theorem 2.3.1, we presented an alternative proof of the spectral adversary bound that holds for all non-Boolean functions. □

5.3.1 Equivalence of spectral and weighted adversary

In this section, we give a linear-algebraic proof that the spectral bound [BSS03] and the (strong) weighted bound [Amb03, Zha05] are equal. The proof has three steps. First, we show that the weighted bound is at least as good as the spectral bound. Second, using a small combinatorial lemma, we show that the spectral bound is at least as good as the strong weighted bound. The strong weighted bound is always at least as good as the weighted bound, because every term in the minimization of (5.4) is included in the minimization of (5.3): if $w(x, y) > 0$ and $x_i \neq y_i$, then $f(x) \neq f(y)$ and both $w'(x, y, i) > 0$ and $w'(y, x, i) > 0$. The generalization of the weighted adversary method thus does not make the bound stronger, however its formulation is easier to use.

Spectral norm of a non-negative matrix First, let us state two useful statements upper-bounding the spectral norm of a entry-wise product of two non-negative matrices. Let $r_x(M)$ denote the *ℓ_2-norm* of the x^{th} row of M and let $c_y(M)$ denote the *ℓ_2-norm* of the y^{th} column of M. Formally,

$$r_x(M) = \sqrt{\sum_y M[x,y]^2}\ , \qquad \text{Also, let} \quad r(M) = \max_x r_x(M)\ ,$$

$$c_y(M) = \sqrt{\sum_x M[x,y]^2}\ . \qquad\qquad c(M) = \max_y c_y(M)\ .$$

5.3.2. Lemma ([Mat90]). *Let $M, N \geq 0$ be non-negative rectangular matrices of the same dimension, and let $S = M \circ N$ be the entry-wise product of M and N. Then*

$$\lambda(S) \leq r(M)c(N) = \max_{x,y} r_x(M)\; c_y(N) \enspace .$$

Furthermore, for every $S \geq 0$ there exist $M, N \geq 0$ such that $S = M \circ N$ and $r(M) = c(N) = \sqrt{\lambda(S)}$. If $S \geq 0$ is a symmetric square matrix, then there exist matrices $M, N \geq 0$ giving the optimal bound that satisfy $M = N^T$.

Proof. The first half of this lemma has already been stated and proved as Lemma 2.4.1 in the introductory chapter on lower bounds. Let us prove its second half here. Given a non-negative matrix S, let v, w be a pair of singular vectors of S corresponding to the principal singular value of S, such that $\|v\| = \|w\| = 1$. Thus $Sw = \lambda(S)v$, $S^T v = \lambda(S)w$, and $\lambda(S) = v^T S w$. Since S is non-negative, both v, w must also be non-negative. Define

$$M[x,y] = \sqrt{S[x,y] \cdot \frac{w_y}{v_x}} \qquad \text{and} \qquad N[x,y] = \sqrt{S[x,y] \cdot \frac{v_x}{w_y}} \enspace . \tag{5.10}$$

Note that if $v_x = 0$, then $S[x,y]w_y = 0$ for all y, because $v_x = \frac{1}{\lambda(S)} \sum_y S[x,y]w_y$ and all numbers are non-negative. Hence divisions by zero only occur in the form $\frac{0}{0}$ and we define them to be 0. Analogously, divisions by w_y are also safe.

Then all row norms of M are the same and they are equal to

$$r_x(M) = \sqrt{\sum_y M[x,y]^2} = \sqrt{\frac{\sum_y S[x,y]w_y}{v_x}} = \sqrt{\lambda(S)\frac{v_x}{v_x}} = \sqrt{\lambda(S)} \enspace .$$

The same argument gives that $c_y(N) = \sqrt{\lambda(S)}$. Finally, if S is symmetric, then the principal singular vectors coincide with the principal eigenvector, that is $v = w$ and $Sv = \lambda(S)v$. A quick look at equation (5.10) reveals that the two optimal matrices we have just defined satisfy $M[x,y] = N[y,x]$. □

We strengthen Mathias's lemma such that the maximum can be taken over a smaller set of pairs x, y that satisfy $S[x,y] > 0$.

5.3.3. Lemma. *Let $M, N \geq 0$ be non-negative matrices and let $S = M \circ N$ be a matrix that is not identically zero. Then*

$$\lambda(S) \leq \max_{\substack{x,y \\ S[x,y]>0}} r_x(M)c_y(N) \enspace .$$

Proof. Define an abbreviation

$$B(M,N) = \max_{\substack{x,y \\ S[x,y]>0}} r_x(M)c_y(N) \enspace .$$

Without loss of generality, we assume that $M[x, y] = 0 \Leftrightarrow N[x, y] = 0$, otherwise the term on the right-hand side gets bigger and the inequality is easier to satisfy. Let us prove the existence of matrices M', N' with $B(M', N') = r(M')c(N')$ such that

$$M \circ N = M' \circ N' \qquad \text{and} \qquad B(M, N) = B(M', N') \ . \tag{5.11}$$

We then apply Lemma 5.3.2 and obtain

$$\lambda(S) = \lambda(M \circ N) = \lambda(M' \circ N') \le r(M')c(N') = B(M', N') = B(M, N) \ .$$

Take as M', N' any pair of matrices that satisfies (5.11) and the following two constraints:

- $r(M')c(N')$ is minimal, that is there is no pair M'', N'' satisfying equation (5.11) and giving a smaller value of the product $r(M'')c(N'')$,
- and, among those, the set R of maximum-norm rows of M' and the set C of maximum-norm columns of N' are both minimal (in the same sense).

Let (r, c) be any *maximal entry*, that is let $S[r, c] > 0$ and $r_r(M')c_c(N') = B(M', N')$. Let $\overline{R}$ denote the *complement* of R and let $S|_R^C$ denote the *sub-matrix* of S indexed by $R \times C$. Then one of the following cases happens:

1. $(r, c) \in R \times C$: Then $B(M', N') = r(M')c(N')$ and we are done.

   If $(r, c) \notin R \times C$, then we know that $S|_R^C = 0$ is a zero sub-matrix.

2. $(r, c) \in R \times \overline{C}$: Then $S|_{\overline{R}}^C = 0$, otherwise we get a contradiction with one of the minimality assumptions as follows. If $S[x, y] \neq 0$ for some $(x, y) \in \overline{R} \times C$, multiply $M'[x, y]$ by $1 + \varepsilon$ and divide $N'[x, y]$ by $1 + \varepsilon$ for some small $\varepsilon > 0$ such that the norm of the x^{th} row of M' is still smaller than $r(M')$. Now, we have either deleted the y^{th} column from C or, if $|C| = 1$, decreased $c(N')$. Both cases are a contradiction.

   Finally, if $S|_{\overline{R}}^C = 0$, then $c(N') = 0$ due to $S|_R^C = 0$ and the fact that C are the maximum-norm columns. Hence S is the zero matrix, which is also a contradiction.

3. $(r, c) \in \overline{R} \times C$: This case is similar to the previous case.

4. $(r, c) \in \overline{R} \times \overline{C}$: First, note that $S|_R^c = 0$, otherwise (r, c) would not be a maximal entry. Now we divide all entries in $M'|_R^{\overline{C}}$ by $1 + \varepsilon$ and multiply all entries in $N'|_R^{\overline{C}}$ by $1 + \varepsilon$ for some small $\varepsilon > 0$ such that the sets R, C of maximum-norm rows and columns do not change. Since $S|_R^C = 0$, we know that $S|_R^{\overline{C}} \neq 0$, otherwise S would be the zero matrix due to $r(M') = 0$. Hence there is a nonzero number in every row of $M'|_R^{\overline{C}}$. We have preserved $B(M', N')$ and $c(N')$, and decreased $r(M')$, which is a contradiction.

We conclude that the only consistent case is $(r, c) \in R \times C$. Then $B(M', N') = r(M')c(N')$, and hence $\lambda(S) \leq B(M, N)$. $\Box$

Reducing spectral adversary to weighted adversary Now we use Mathias's bound to prove that the weighted adversary gives at least as good a bound as the spectral adversary.

5.3.4. Theorem. $\mathrm{SA}(f) \leq \mathrm{WA}(f)$.

Proof. Let Γ be a non-negative symmetric matrix with $\Gamma \circ F = \Gamma$ as in equation (5.2) that gives the optimal spectral bound. Assume without loss of generality that $\lambda(\Gamma) = 1$. Let δ be the principal eigenvector of Γ, that is $\Gamma\delta = \delta$. Define the following weight scheme:

$$w(x, y) = w(y, x) = \Gamma[x, y]\delta_x\delta_y \ .$$

Furthermore, for every i, using the last part of Lemma 5.3.2, decompose $\Gamma_i = \Gamma \circ D_i$ into a entry-wise product of two non-negative matrices $\Gamma_i = M_i \circ M_i^T$ such that $r(M_i) = \sqrt{\lambda(\Gamma_i)}$. Define w' as follows:

$$w'(x, y, i) = M_i[x, y]^2\delta_x^2 \ .$$

Let us verify that w, w' is a weight scheme. First, $w(x, y) = w'(x, y, i) = 0$ if $f(x) = f(y)$, and also $w'(x, y, i) = 0$ if $x_i = y_i$. Furthermore, if $f(x) \neq f(y)$ and $x_i \neq y_i$, then $w'(x, y, i)w'(y, x, i) = (M_i[x, y]\,\delta_x)^2(M_i[y, x]\,\delta_y)^2 = (\Gamma_i[x, y]\,\delta_x\delta_y)^2 = w(x, y)^2$. Finally, let us compute the bound (5.3) given by the weight scheme.

$$wt(x) = \sum_y w(x, y) = \delta_x \sum_y \Gamma[x, y]\delta_y = \delta_x\,(\Gamma\delta)_x = \delta_x^2 \ ,$$
$$\frac{v(x, i)}{wt(x)} = \frac{\sum_y w'(x, y, i)}{wt(x)} = \frac{\sum_y M_i[x, y]^2\delta_x^2}{\delta_x^2} = r_x(M_i)^2 \leq r(M_i)^2 = \lambda(\Gamma_i) \ .$$

The weighted adversary lower bound (5.3) is thus at least

$$\mathrm{WA}(f) \geq \min_{\substack{x,y,\ i,j \\ f(x)\neq f(y) \\ v(x,i)v(y,j)>0}} \sqrt{\frac{wt(x)wt(y)}{v(x, i)v(y, j)}}$$
$$\geq \min_{i,j} \frac{1}{\sqrt{\lambda(\Gamma_i) \cdot \lambda(\Gamma_j)}} = \frac{\lambda(\Gamma)}{\max_i \lambda(\Gamma_i)} = \mathrm{SA}(f) \ .$$

Hence the weighted adversary gives at least as strong a bound as the spectral adversary (5.2). $\Box$

Reducing strong weighted adversary to spectral adversary Now we use our stronger version of Mathias's bound to prove that the spectral adversary gives at least as good a bound as the strong weighted adversary.

5.3.5. Theorem. SWA$(f) \le$ SA(f).

Proof. Let w, w' be a weight scheme as in equation (5.3) that gives the optimal strong weighted bound (5.4). Define the following symmetric matrix Γ on $S \times S$:

$$\Gamma[x,y] = \frac{w(x,y)}{\sqrt{wt(x)wt(y)}} \ .$$

We also define column vector δ on S such that $\delta_x = \sqrt{wt(x)}$. Let $W = \sum_x wt(x)$. Then

$$\lambda(\Gamma) \ge \frac{\delta^T \Gamma \delta}{\|\delta\|^2} = \frac{W}{W} = 1 \ .$$

Define the following matrix on the index set $S \times S$:

$$M_i[x,y] = \sqrt{\frac{w'(x,y,i)}{wt(x)}} \ .$$

Every weight scheme satisfies $w'(x,y,i)w'(y,x,i) \ge w^2(x,y)$ for all x, y, i such that $x_i \ne y_i$. Hence

$$M_i[x,y] \cdot M_i[y,x] = \frac{\sqrt{w'(x,y,i)w'(y,x,i)}}{\sqrt{wt(x)wt(y)}} \ge \frac{w(x,y) \cdot D_i[x,y]}{\sqrt{wt(x)wt(y)}} = \Gamma_i[x,y] \ .$$

This means that $\Gamma_i \le M_i \circ M_i^T$ and, thanks to the non-negativity, it also holds that $\lambda(\Gamma_i) \le \lambda(M_i \circ M_i^T)$. By Lemma 5.3.3 and using $c_y(M^T) = r_y(M)$,

$$\begin{aligned}\lambda(\Gamma_i) \le \max_{\substack{x,y \\ \Gamma_i[x,y]>0}} r_x(M) r_y(M) &= \max_{\substack{x,y \\ w(x,y)>0 \\ x_i \ne y_i}} \sqrt{\sum_k \frac{w'(x,k,i)}{wt(x)} \sum_\ell \frac{w'(y,\ell,i)}{wt(y)}} \\ &= \max_{\substack{x,y \\ w(x,y)>0 \\ x_i \ne y_i}} \sqrt{\frac{v(x,i)v(y,i)}{wt(x)wt(y)}} \ .\end{aligned}$$

The spectral adversary lower bound (5.2) is thus at least

$$\mathrm{SA}(f) \ge \frac{\lambda(\Gamma)}{\max_i \lambda(\Gamma_i)} \ge \min_i \min_{\substack{x,y \\ w(x,y)>0 \\ x_i \ne y_i}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,i)}} = \mathrm{SWA}(f) \ .$$

Hence the spectral adversary gives at least as strong a bound as the strong weighted adversary (5.4). □

5.3.6. Remark. The strength of the obtained reduction depends on which statement is used for upper-bounding the spectral norm of Γ_i.

- Lemma 5.3.3 has given us $\mathrm{SWA}(f) \leq \mathrm{SA}(f)$.
- Lemma 5.3.2 would give a weaker bound $\mathrm{WA}(f) \leq \mathrm{SA}(f)$.
- Høyer, Neerbek, and Shi used an explicit expression for the norm of the Hilbert matrix to get an $\Omega(\log n)$ lower bound for ordered search [HNS02]. Their method is thus a special case of the spectral method.
- As we have shown in Theorem 2.4.2, both versions of the original unweighted adversary method [Amb02] are obtained by using a spectral matrix Γ corresponding to a zero-one valued weight scheme w, the lower bound $\lambda(\Gamma) \geq d^T \Gamma d / \|d\|^2$, and Lemma 5.3.2, resp. Lemma 5.3.3.

5.3.2 Equivalence of primal and dual adversary bounds

Equivalence of minimax and generalized spectral adversary Here we prove that the minimax bound is equal to the generalized spectral bound. We first remove the reciprocal by taking the max-min bound. Second, we write this bound as a semidefinite program. An application of duality theory of semidefinite programming finishes the proof.

5.3.7. Theorem. $\mathrm{MM}(f) = \mathrm{SMM}(f)$.

Proof. Let p be a set of probability distributions as in equation (5.7). Define $R_i[x,y] = \sqrt{p_x(i)\, p_y(i)}$. Since p_x is a probability distribution, we get that $\sum_i R_i$ must have all ones on the diagonal. The condition

$$\min_{\substack{x,y \\ f(x) \neq f(y)}} \sum_{i : x_i \neq y_i} R_i[x,y] \geq \mu$$

may be rewritten

$$\forall x, y : f(x) \neq f(y) \Longrightarrow \sum_{i : x_i \neq y_i} R_i[x,y] \geq \mu \ ,$$

which is to say $\sum_i R_i \circ D_i \geq \mu F$. Each matrix R_i should be an outer product of a non-negative vector with itself: $R_i = r_i r_i^T$ for a column vector $r_i[x] = \sqrt{p_x(i)}$. We have, however, replaced that condition by $R_i \succeq 0$ to get the semidefinite program (5.8). Since $r_i r_i^T \succeq 0$, the program (5.8) is a relaxation of the condition of (5.7) and $\mathrm{SMM}(f) \leq \mathrm{MM}(f)$.

Let us show that every solution R_i of the semidefinite program can be changed to an at least as good rank-1 solution R'_i. Take a Cholesky decomposition $R_i = X_i X_i^T$. Define a column-vector $q_i[x] = \sqrt{\sum_j X_i[x,j]^2}$ and a rank-1 matrix $R'_i =$

$q_i q_i^T$. It is not hard to show that all R'_i satisfy the same constraints as R_i. First, R'_i is positive semidefinite. Second, $R'_i[x,x] = \sum_j X_i[x,j]^2 = R_i[x,x]$, hence $\sum_i R'_i \circ I = I$. Third, by the Cauchy-Schwarz inequality,

$$\begin{aligned} R_i[x,y] &= \sum_j X_i[x,j] X_i[y,j] \\ &\le \sqrt{\sum_k X_i[x,k]^2} \sqrt{\sum_\ell X_i[y,\ell]^2} = q_i[x] q_i[y] = R'_i[x,y] \ , \end{aligned}$$

hence $\sum_i R'_i \circ D_i \ge \sum_i R_i \circ D_i \ge \mu F$. We conclude that $\mathrm{MM}(f) \le \mathrm{SMM}(f)$. $\square$

The equivalence of the semidefinite version of minimax and the generalized spectral adversary is proved using the duality theory of semidefinite programming. We use the duality principle [Lov00, Theorem 3.4], which is a semidefinite version of Farkas's lemma.

5.3.8. Theorem. $\mathrm{SMM}(f) = \mathrm{GSA}(f)$.

Proof. Let us compute the dual of a semidefinite program without converting it to/from the standard form, but using Lagrange multipliers. Take the objective function μ of the semidefinite version of minimax (5.8) and add negative penalty terms for violating the constraints. Let $A \cdot B = \mathrm{Tr}\,(AB^T)$ denote the scalar product of A and B.

$$\begin{aligned} &\mu + \sum_i Y_i \cdot R_i + D \cdot \Big(\sum_i R_i \circ I - I \Big) + Z \cdot \Big(\sum_i R_i \circ D_i - \mu F \Big) \\ &\quad \text{for } Y_i \succeq 0, \text{ unconstrained } D, \text{ and } Z \ge 0 \\ &= \sum_i R_i \cdot \Big(Y_i + D \circ I + Z \circ D_i \Big) + \mu \Big(1 - Z \cdot F \Big) - D \cdot I \ . \end{aligned}$$

Its dual system is formed by the constraints on Y_i, D, and Z plus the requirements that both expressions in the parentheses on the third line are zero. The duality principle [Lov00, Theorem 3.4] says that any primal solution is smaller than or equal to any dual solution. Moreover, if any of the two systems has a strictly feasible solution, then the maximal primal solution $\mu_{\max}$ equals the minimal dual solution $\mu_{\min}$.

Since $Y_i \succeq 0$ only appears once, we remove it by requiring that $D \circ I + Z \circ D_i \preceq 0$. We substitute $\Delta = -D \circ I$ and obtain $\Delta - Z \circ D_i \succeq 0$. The objective function is $-D \cdot I = \mathrm{Tr}\,\Delta$. We have obtained the generalized spectral adversary (5.9). Let us prove its strong feasibility. Assume that the function f is not constant, hence $F \ne 0$. Take Z a uniform probability distribution over nonzero entries of F and a large enough constant Δ. This is a strictly feasible solution. We conclude that $\mu_{\max} = \mu_{\min}$. $\square$

Equivalence of generalized spectral and spectral adversary Here we prove that the generalized spectral adversary bound is equal to the spectral adversary bound. The main difference between them is that the generalized method uses an arbitrary positive diagonal matrix Δ as a new variable instead of the identity matrix I.

5.3.9. Theorem. $\mathrm{GSA}(f) = \mathrm{SA}(f)$.

Proof. Let $\operatorname{diag}(A)$ denote the column vector containing the *main diagonal* of A. Let Z, Δ be a solution of (5.9). First, let us prove that $\Delta \succ 0$. Since both $Z \geq 0$ and $D_i \geq 0$, it holds that $\operatorname{diag}(-Z \circ D_i) \leq 0$. We know that $\Delta - Z \circ D_i \succeq 0$, hence $\operatorname{diag}(\Delta - Z \circ D_i) \geq 0$, and $\operatorname{diag}(\Delta) \geq 0$ follows. Moreover, $\operatorname{diag}(\Delta) > 0$ unless Z contains an empty row (together with the corresponding column), in which case we delete them and continue. The spectral matrix Γ will then contain zeroes in this row and column, too. Second, since positive semidefinite real matrices are symmetric, $\Delta - Z \circ D_i \succeq 0$ implies that $Z \circ D_i$ is symmetric (for every i). For every $x \neq y$ there is an index i such that $x_i \neq y_i$, hence Z must be also symmetric.

Take column vector $a = \operatorname{diag}(\Delta^{-1/2})$ and rank-1 matrix $A = aa^T \succeq 0$. It is simple to prove that $A \circ X \succeq 0$ for every matrix $X \succeq 0$. Since $\Delta - Z \circ D_i \succeq 0$, also $A \circ (\Delta - Z \circ D_i) = I - Z \circ D_i \circ A \succeq 0$ and hence $\lambda(Z \circ D_i \circ A) \leq 1$. Now, define the spectral adversary matrix

$$\Gamma = Z \circ F \circ A \ .$$

Since $0 \leq Z \circ F \leq Z$, it follows that

$$\lambda(\Gamma \circ D_i) = \lambda(Z \circ F \circ A \circ D_i) \leq \lambda(Z \circ D_i \circ A) \leq 1 \ .$$

It remains to show that $\lambda(\Gamma) \geq 1/\operatorname{Tr}\Delta$. Let $b = \operatorname{diag}(\sqrt{\Delta})$ and $B = bb^T$. Then $A \circ B$ is the all-ones matrix, and

$$1 = Z \cdot F = \Gamma \cdot B = b^T \Gamma b \leq \lambda(\Gamma) \cdot \|b\|^2 = \lambda(\Gamma) \cdot \operatorname{Tr}\Delta \ .$$

It is obvious that Γ is symmetric, $\Gamma \geq 0$, and $\Gamma \circ F = \Gamma$. The bound (5.2) given by Γ is bigger than or equal to $1/\operatorname{Tr}\Delta$, hence $\mathrm{SA}(f) \geq \mathrm{GSA}(f)$.

For the other direction, let Γ be a non-negative symmetric matrix satisfying $\Gamma \circ F = \Gamma$. Let δ be its principal eigenvector with $\|\delta\| = 1$. Assume without loss of generality that $\lambda(\Gamma) = 1$ and let $\mu = \max_i \lambda(\Gamma_i)$. Take $A = \delta\delta^T$, $Z = \Gamma \circ A$, and $\Delta = \mu I \circ A$. Then $Z \cdot F = \Gamma \cdot A = \delta^T \Gamma \delta = 1$ and $\operatorname{Tr}\Delta = \mu$. For every i, $\lambda(\Gamma_i) \leq \mu$, hence $\mu I - \Gamma \circ D_i \succeq 0$. It follows that $0 \preceq A \circ (\mu I - \Gamma \circ D_i) = \Delta - Z \circ D_i$. The semidefinite program (5.9) is satisfied and hence its optimum is $\mu_{\min} \leq \mu$. We conclude that $\mathrm{GSA}(f) \geq \mathrm{SA}(f)$. □

5.3.3 Equivalence of minimax and Kolmogorov adversary

In this section, we prove the last equivalence. We use the results of Laplante and Magniez, who proved [LM04] that the Kolmogorov complexity bound is asymptotically lower-bounded by the weighted adversary bound and upper-bounded by the minimax bound. The upper bound is implicit in their paper, because they did not state the minimax bound as a separate theorem.

5.3.10. Theorem. [LM04, Theorem 2] $\mathrm{KA}(f) = \Omega(\mathrm{WA}(f))$.

The proof of Theorem 5.3.10 uses the symmetry of information [LV97].

5.3.11. Theorem. $\mathrm{KA}(f) = O(\mathrm{MM}(f))$.

Proof. Take a set of probability distributions p as in equation (5.6). The query information lemma [LM04, Lemma 3] says that $K(i|x,p) \leq \log \frac{1}{p_x(i)} + O(1)$ for every x, i such that $p_x(i) > 0$. This is true, because any i of nonzero probability can be encoded in $\lceil \log \frac{1}{p_x(i)} \rceil$ bits using the Shannon-Fano code of distribution p_x, and the Shannon-Fano code is a prefix-free code. Rewrite the inequality as $p_x(i) = O(2^{-K(i|x,p)})$. The statement follows, because the set of all strings σ in (5.5) includes among others also the descriptions of all sets of probability distributions p. $\square$

5.3.12. Remark. The hidden constant in the equality $\mathrm{KA}(f) = \Theta(\mathrm{WA}(f))$ depends on the choice of the universal Turing machine and the prefix-free set.

5.4 Limitation of adversary bounds

In this section, we show that there are limits that none of these quantum adversary methods can go beyond. Recall Definition 2.5.2 of a certificate of a function for an input, and of the h-certificate complexity $\mathrm{C}_h(f)$ of a function f, often shortened as just C_h. We need the following additional definition.

5.4.1. Definition. Let $\mathrm{Cert}(f, x)$ denote the lexicographically first certificate among the smallest certificates of f for input x.

It is actually not important for us that we take the lexicographically first certificate, any other certificate would work as well, however the discussion gets simpler if we can talk about a concrete certificate instead of a set of certificates.

5.4.2. Theorem. *Let $f : S \to \Sigma'$ be a function with domain $S \subseteq \Sigma^n$. Let the output alphabet be $\Sigma' = \{0, 1, \dots, |\Sigma'| - 1\}$ with letters $h \in \Sigma'$ ordered by their h-certificate complexities such that $\mathrm{C}_0 \geq \mathrm{C}_1 \geq \dots \geq \mathrm{C}_{|\Sigma'|-1}$. Then the max-min bound (5.7) is upper-bounded by $\mathrm{MM}(f) \leq 2\sqrt{\mathrm{C}_1 n}$. If f is total, that is if $S = \Sigma^n$, then $\mathrm{MM}(f) \leq \sqrt{\mathrm{C}_0 \mathrm{C}_1}$.*

PROOF. The following simple argument is due to Ronald de Wolf. We exhibit two sets of probability distributions p such that

$$\mathrm{MM}_p(f) = \min_{\substack{x,y \\ f(x)\neq f(y)}} \sum_{i: x_i \neq y_i} \sqrt{p_x(i)\, p_y(i)} \geq \frac{1}{2\sqrt{\mathrm{C}_1 n}} \ \text{, resp.}\ \frac{1}{\sqrt{\mathrm{C}_0 \mathrm{C}_1}} \ .$$

The max-min bound (5.7) is equal to $\mathrm{MM}(f) = 1/\max_p \mathrm{MM}_p(f)$ and the statement follows.

Let f be partial. For every $x \in S$, distribute one half of the probability uniformly over $\mathrm{Cert}(f,x)$, and one half of the probability uniformly over all input variables. Formally, $p_x(i) = \frac{1}{2n} + \frac{1}{2|\mathrm{Cert}(f,x)|}$ if $i \in \mathrm{Cert}(f,x)$, and $p_x(i) = \frac{1}{2n}$ otherwise. Take any x, y such that $f(x) \neq f(y)$. Assume that $\mathrm{C}_{f(x)} \leq \mathrm{C}_{f(y)}$, and take the $f(x)$-certificate $C = \mathrm{Cert}(f,x)$. Since $y|_C \neq x|_C$, there is a $j \in C$ such that $x_j \neq y_j$. Now we lower-bound the sum of (5.7).

$$\begin{aligned}\sum_{i: x_i \neq y_i} \sqrt{p_x(i)\, p_y(i)} &\geq \sqrt{p_x(j)\, p_y(j)} \\ &\geq \sqrt{\frac{1}{|2\mathrm{Cert}(f,x)|} \cdot \frac{1}{2n}} \geq \frac{1}{2\sqrt{\mathrm{C}_{f(x)} n}} \geq \frac{1}{2\sqrt{\mathrm{C}_1 n}} \ .\end{aligned}$$

Since this inequality holds for any x, y such that $f(x) \neq f(y)$, also $\mathrm{MM}_p(f) \geq 1/2\sqrt{\mathrm{C}_1 n}$. Take the reciprocal and conclude that $\mathrm{MM}(f) \leq 2\sqrt{\mathrm{C}_1 n}$.

For Boolean output alphabet $\Sigma' = \{0,1\}$, we can prove a twice stronger bound $\mathrm{MM}(f) \leq \sqrt{\mathrm{C}_1 n}$ as follows. Define p as a uniform distribution over $\mathrm{Cert}(f,x)$ for all one-inputs, and a uniform distribution over all input variables for all zero-inputs. The same computation as above gives the bound.

If f is total, then we can do even better. For every $x \in \Sigma^n$, distribute the probability uniformly over $\mathrm{Cert}(f,x)$. Formally, $p_x(i) = \frac{1}{|\mathrm{Cert}(f,x)|}$ if $i \in \mathrm{Cert}(f,x)$, and $p_x(i) = 0$ otherwise. Take any x, y such that $f(x) \neq f(y)$, and let $C = \mathrm{Cert}(f,x) \cap \mathrm{Cert}(f,y)$. There must exist a $j \in C$ such that $x_j \neq y_j$, otherwise we could find an input z that is consistent with both certificates. (That would be a contradiction, because f is total and hence $f(z)$ has to be defined and be equal to both $f(x)$ and $f(y)$.) After we have found a j, we lower-bound the sum of (5.7) by $1/\sqrt{\mathrm{C}_{f(x)}\mathrm{C}_{f(y)}}$ in the same way as above. Since $\sqrt{\mathrm{C}_{f(x)}\mathrm{C}_{f(y)}} \leq \sqrt{\mathrm{C}_0 \mathrm{C}_1}$, the bound follows. □

Some parts of the following statement have been observed for individual methods by Szegedy [Sze03], Laplante and Magniez [LM04], and Zhang [Zha05]. This statement rules out all adversary attempts to prove good lower bounds for problems with small certificate complexity, such as element distinctness [AS04], binary AND-OR trees [BS04, HMW03], triangle finding [MSS05], or verification of matrix products (see Chapter 4).

5.4.3. Corollary. *Quantum adversary lower bounds are smaller than or equal to* $\min(\sqrt{\mathrm{C}_0 n}, \sqrt{\mathrm{C}_1 n})$ *for partial Boolean functions and* $\sqrt{\mathrm{C}_0 \mathrm{C}_1}$ *for total Boolean functions.*

5.5 Composition of adversary bounds

In this section we prove Theorem 5.1.5, that is the composition theorem for adversary bounds. Unlike in the previous sections, we are only able to prove this statement for Boolean functions. We first generalize the adversary bound by introducing a cost of querying an input variable, and sketch the proof of an equivalence theorem for this bound. Then we express the spectral norm of a composite adversary matrix in terms of spectral norms of the individual adversary matrices. Finally, we use this expression to prove that the spectral bound composes, which, together with the fact that the minimax bound composes, implies that our composite bound is tight.

5.5.1 Adversary bound with costs

5.5.1. Definition. Let $f : S \to \{0,1\}$ be a Boolean function with domain $S \subseteq \{0,1\}^n$, and let $\alpha \in \mathbb{R}^n_+$ be a *cost vector*. The *spectral bound for f with costs α* is

$$\mathrm{SA}_\alpha(f) = \max_\Gamma \min_i \left\{ \alpha_i \frac{\lambda(\Gamma)}{\lambda(\Gamma_i)} \right\} ,$$

where Γ ranges over all adversary matrices for f, that is Γ is non-negative and symmetric, and $\Gamma[x,y] = 0$ if $f(x) = f(y)$. The *minimax bound for f with costs α* is

$$\mathrm{MM}_\alpha(f) = \min_p \max_{\substack{x,y \\ f(x) \neq f(y)}} \frac{1}{\sum_{i: x_i \neq y_i} \sqrt{p_x(i) p_y(i)}/\alpha_i} ,$$

where $p : S \times [n] \to [0,1]$ ranges over all sets of $|S|$ probability distributions over input bits, that is, $p_x(i) \geq 0$ and $\sum_{i=1}^n p_x(i) = 1$ for every $x \in S$.

These two bounds are natural generalizations of the spectral adversary bound (Definition 2.5.1) and the minimax bound (Definition 5.1.2)—we obtain the original bounds by setting $\alpha_i = 1$ for all i. The two new bounds are equal, and thus we give this common quantity a name, the *adversary bound of f with costs α*, denoted by $\mathrm{Adv}_\alpha(f)$.

5.5.2. Theorem. *For every function $f : S \to \{0,1\}$ with domain $S \subseteq \{0,1\}^n$, and for every $\alpha \in \mathbb{R}^n_+$,*

$$\mathrm{Adv}_\alpha(f) = \mathrm{SA}_\alpha(f) = \mathrm{MM}_\alpha(f) .$$

Proof (sketch). We start with the minimax bound with costs, substitute $q_x(i) = p_x(i)/\alpha_i$, and rewrite the condition $\sum_i p_x(i) = 1$ into $\sum_i \alpha_i q_x(i) = 1$. Using similar arguments as in Theorem 5.3.7, we rewrite the bound as a semidefinite program, compute its dual, and after a few simplifications like in Theorem 5.3.9, get the spectral bound with costs. $\square$

5.5.2 Spectral norm of a composite spectral matrix

In this section we construct an adversary matrix for a composite function and compute its spectral norm. We later show that this construction is in fact optimal given that the adversary matrices of the inner functions used in this construction are all optimal. This construction is an extension of the construction that Ambainis used to establish the composition theorem for iterated functions [Amb03].

5.5.3. Definition. Let $h = f \circ (g_1, \dots, g_k)$ be a composed function, like in equation (5.1). An input $x \in \{0,1\}^n$ to h is a bit string of length $n = \sum_i n_i$, which we think of as being comprised of k parts, $x = (x^1, x^2, \dots, x^k)$, where $x^i \in \{0,1\}^{n_i}$. We may evaluate h on input x by first computing the k bits $\tilde{x}_i = g_i(x^i)$, and then evaluating f on input $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_k)$.

5.5.4. Definition. Let Γ_f be an adversary matrix for f and let δ_f be a principal eigenvector of Γ_f such that $\|\delta_f\| = 1$. Similarly, for each $i = 1, \dots, k$, let Γ_{g_i} be an adversary matrix for g_i and let δ_{g_i} be a principal eigenvector of Γ_{g_i} such that $\|\delta_{g_i}\| = 1$. Let Γ_h be the matrix

$$\Gamma_h[x, y] = \Gamma_f[\tilde{x}, \tilde{y}] \cdot \Gamma_g[x, y] \ , \tag{5.12}$$

where

$$\Gamma_g = \bigotimes_{i=1}^{k} \Gamma^\star_{g_i}, \qquad \Gamma^\star_{g_i} = \Gamma^0_{g_i} + \Gamma^1_{g_i}, \qquad \Gamma^0_{g_i} = \lambda(\Gamma_{g_i}) I, \qquad \Gamma^1_{g_i} = \Gamma_{g_i} \ , \tag{5.13}$$

and I is the identity matrix.

It is simple to prove that Γ_h is an adversary matrix for h, that is Γ_h is a non-negative symmetric matrix and $\Gamma_h[x, y] = 0$ if $h(x) = h(y)$.

5.5.5. Theorem. *The matrix Γ_h has spectral norm $\lambda(\Gamma_h) = \lambda(\Gamma_f) \cdot \prod_{i=1}^k \lambda(\Gamma_{g_i})$ and principal eigenvector $\delta_h[x] = \delta_f[\tilde{x}] \cdot \prod_i \delta_{g_i}[x^i]$.*

Proof. Evaluate $d^T \Gamma_h d$ for a general column vector d.

$$\begin{aligned} d^T \Gamma_h d &= \sum_{x,y} \Gamma_h[x,y] \cdot d_x d_y \\ &= \sum_{x,y} \Gamma_f[\tilde{x},\tilde{y}] \cdot \Gamma_g[x,y] \cdot d_x d_y \\ &= \sum_{x',y'} \Gamma_f[x',y'] \sum_{\substack{x:\tilde{x}=x' \\ y:\tilde{y}=y'}} \Gamma_g[x,y] \cdot d_x d_y \ . \end{aligned}$$

Fix any pair x', y' of k-bit strings and look at the bit string $z = x' \oplus y'$. Set $\Gamma_g^z = \bigotimes_{i=1}^k \Gamma_{g_i}^{z_i}$, that is instead of taking the product of sums $\Gamma_{g_i}^\star = \Gamma_{g_i}^0 + \Gamma_{g_i}^1$ as in Γ_g in equation (5.13) we admit exactly one term for each i. Note that $\lambda(\Gamma_g^z) = \prod_{i=1}^k \lambda(\Gamma_{g_i}^{z_i}) = \prod_i \lambda(\Gamma_{g_i})$. We prove that $\Gamma_g = \Gamma_g^z$ on the rectangle $(x,y) : \tilde{x} = x', \tilde{y} = y'$, because the omitted term in $\Gamma_{g_i}^\star$ is zero for each i. There are two cases. First, if $z_i = 0$, then $\tilde{x}_i = \tilde{y}_i$, that is $g_i(x^i) = g_i(y^i)$, and hence the omitted term is $\Gamma_{g_i}[x^i, y^i] = 0$, because Γ_{g_i} is an adversary matrix for g_i. Second, if $z_i = 1$, then $\tilde{x}_i \neq \tilde{y}_i$, hence it must be that $x^i \neq y^i$ and so the omitted term is $I[x^i, y^i] = 0$.

For a bit string $a \in \{0,1\}^k$, let $d^{\restriction a}[x] = d_x$ if $\tilde{x} = a$, and 0 otherwise. Define a column vector $d'[a] = \|d^{\restriction a}\|$ of dimension 2^k. It holds that

$$\|d\|^2 = \sum_{x \in \{0,1\}^n} |d_x|^2 = \sum_{a \in \{0,1\}^k} \sum_{x:\tilde{x}=a} |d_x|^2 = \sum_{a \in \{0,1\}^k} \|d^{\restriction a}\|^2 = \sum_{a \in \{0,1\}^k} d'[a]^2 = \|d'\|^2 \ .$$

We prove the equality in two steps. First, we prove that $\lambda(\Gamma_h) \leq \lambda(\Gamma_f) \cdot \prod_{i=1}^k \lambda(\Gamma_{g_i})$. This follows from the fact that for any vector d,

$$\begin{aligned} d^T \Gamma_h d &= \sum_{a,b} \Gamma_f[a,b] \sum_{\substack{x:\tilde{x}=a \\ y:\tilde{y}=b}} \Gamma_g[x,y] \cdot d_x d_y \\ &= \sum_{a,b} \Gamma_f[a,b] \sum_{x,y} \Gamma_g^{a \oplus b}[x,y] \cdot d^{\restriction a}[x] d^{\restriction b}[y] \qquad \text{by the above paragraph} \\ &\leq \sum_{a,b} \Gamma_f[a,b] \cdot \lambda(\Gamma_g^{a \oplus b}) \cdot \|d^{\restriction a}\| \cdot \|d^{\restriction b}\| \\ &= \prod_{i=1}^k \lambda(\Gamma_{g_i}) \cdot \sum_{a,b} \Gamma_f[a,b] \cdot d'[a] d'[b] \\ &\leq \prod_{i=1}^k \lambda(\Gamma_{g_i}) \cdot \lambda(\Gamma_f) \cdot \|d'\|^2 \\ &= \lambda(\Gamma_f) \cdot \prod_{i=1}^k \lambda(\Gamma_{g_i}) \cdot \|d\|^2 \ . \end{aligned}$$

Second, we prove that the vector $\delta_h[x] = \delta_f[\tilde{x}] \cdot \prod_i \delta_{g_i}[x^i]$ achieves this upper bound and so is a principal eigenvector of Γ_h. For a bit $c \in \{0,1\}$, let $\delta_{g_i}^{\restriction c}[x] = \delta_{g_i}[x]$ if $g_i(x) = c$, and 0 otherwise. Note that $\delta_{g_i} = \delta_{g_i}^{\restriction 0} + \delta_{g_i}^{\restriction 1}$, $\Gamma_{g_i}\delta_{g_i} = \lambda(\Gamma_{g_i})\delta_{g_i}$, and $\Gamma_{g_i}\delta_{g_i}^{\restriction c} = \lambda(\Gamma_{g_i})\delta_{g_i}^{\restriction 1-c}$. The last equality holds because $\Gamma_{g_i}[x,y] \neq 0$ only if $g_i(x) \neq g_i(y)$, that is after rearranging the inputs such that zero-inputs come first, Γ_{g_i} is of the form $\left(\begin{smallmatrix} 0 & G_i \\ G_i^T & 0 \end{smallmatrix}\right)$ for some non-negative matrix G_i. This is exactly the place where we need that the composed functions are Boolean. If the functions are over alphabets of size bigger than 2, then we cannot say much about how the vectors $\delta_{g_i}^{\restriction c}$ are mapped. It also follows that $\|\delta_{g_i}^{\restriction 0}\|^2 = \|\delta_{g_i}^{\restriction 1}\|^2 = \frac{\|\delta_{g_i}\|^2}{2} = \frac{1}{2}$.

Let us first evaluate the ℓ_2-norm of δ_h.

$$\begin{aligned}
\|\delta_h\|^2 &= \sum_x \delta_h[x]^2 = \sum_x \delta_f[\tilde{x}]^2 \prod_i \delta_{g_i}[x^i]^2 \\
&= \sum_a \delta_f[a]^2 \sum_{x:\tilde{x}=a} \prod_i \delta_{g_i}[x^i]^2 \\
&= \sum_a \delta_f[a]^2 \prod_i \sum_{x^i:\tilde{x}_i=a_i} \delta_{g_i}[x^i]^2 \\
&= \sum_a \delta_f[a]^2 \prod_i \|\delta_{g_i}^{\restriction a_i}\|^2 \\
&= \frac{1}{2^k} \sum_a \delta_f[a]^2 = \frac{1}{2^k} \ .
\end{aligned}$$

We already know that

$$\begin{aligned}
{\delta_h}^T \Gamma_h \delta_h &= \sum_{a,b} \Gamma_f[a,b] \sum_{\substack{x:\tilde{x}=a \\ y:\tilde{y}=b}} \Gamma_g^{a\oplus b}[x,y] \cdot \delta_h[x]\delta_h[y] \\
&= \sum_{a,b} \Gamma_f[a,b]\delta_f[a]\delta_f[b] \sum_{\substack{x:\tilde{x}=a \\ y:\tilde{y}=b}} \prod_{i=1}^k \Gamma_{g_i}^{a_i\oplus b_i}[x^i,y^i]\delta_{g_i}[x^i]\delta_{g_i}[y^i] \ .
\end{aligned}$$

Swap the inner summation and multiplication as follows:

$$\begin{aligned}
\sum_{\substack{x:\tilde{x}=a \\ y:\tilde{y}=b}} \prod_{i=1}^k \Gamma_{g_i}^{a_i\oplus b_i}[x^i,y^i]\delta_{g_i}[x^i]\delta_{g_i}[y^i] &= \prod_i \sum_{\substack{x^i:\tilde{x}_i=a_i \\ y^i:\tilde{y}_i=b_i}} \Gamma_{g_i}^{a_i\oplus b_i}[x^i,y^i]\delta_{g_i}[x^i]\delta_{g_i}[y^i] \\
&= \prod_i \sum_{x^i,y^i} \Gamma_{g_i}^{a_i\oplus b_i}[x^i,y^i]\delta_{g_i}^{\restriction a_i}[x^i]\delta_{g_i}^{\restriction b_i}[y^i] \\
&= \prod_i (\delta_{g_i}^{\restriction a_i})^T \Gamma_{g_i}^{a_i\oplus b_i} \delta_{g_i}^{\restriction b_i} \\
&= \prod_i \left(\lambda(\Gamma_{g_i}) \cdot \|\delta_{g_i}^{\restriction a_i}\|^2\right) = \frac{1}{2^k} \prod_i \lambda(\Gamma_{g_i}) \ ,
\end{aligned}$$

which is a constant. The penultimate equality holds since $\Gamma_{g_i}^{a_i \oplus b_i} \delta_{g_i}^{\restriction b_i} = \lambda(\Gamma_{g_i}) \delta_{g_i}^{\restriction a_i}$.

Substitute the rewritten expression back into the total sum. Take the constant out of the sum.

$$\begin{aligned} {\delta_h}^T \Gamma_h \delta_h &= \frac{1}{2^k} \prod_i \lambda(\Gamma_{g_i}) \cdot \sum_{a,b} \Gamma_f[a,b] \delta_f[a] \delta_f[b] \\ &= \frac{1}{2^k} \prod_i \lambda(\Gamma_{g_i}) \cdot \lambda(\Gamma_f) \cdot \|\delta_f\|^2 \\ &= \lambda(\Gamma_f) \cdot \prod_i \lambda(\Gamma_{g_i}) \cdot \|\delta_h\|^2 \ . \end{aligned}$$

It follows that $\lambda(\Gamma_h) \geq \lambda(\Gamma_f) \cdot \prod_{i=1}^k \lambda(\Gamma_{g_i})$, witnessed by δ_h. Together with the matching upper bound we conclude that $\lambda(\Gamma_h) = \lambda(\Gamma_f) \prod_i \lambda(\Gamma_{g_i})$. □

5.5.3 Composition properties

In this section, we prove that both the spectral bound with costs and the minimax bound with costs compose. Then we apply the duality theorem and conclude that the composite bound is tight. In our composition theorems, we use the following cost vectors:

- $\alpha = (\alpha^1, \dots, \alpha^k)$ is a k-tuple of vectors $\alpha^i \in \mathbb{R}_+^{n_i}$, which are arbitrary cost vectors for the inner functions g_i,
- $\beta = (\beta_1, \dots, \beta_k)$ is a k-tuple of numbers $\beta_i = \mathrm{Adv}_{\alpha^i}(g_i)$, which are adversary bounds for the inner functions with the prescribed costs. β forms a cost vector for the outer function.

Composition of the spectral bound Let D_i denote the zero-one valued matrix defined by $D_i[x,y] = 1$ if and only if $x_i \neq y_i$. Let $A \circ B$ denote the entry-wise product of matrices A and B, that is, $(A \circ B)[x,y] = A[x,y] \cdot B[x,y]$. To avoid confusion with subscripts, we write $\Gamma \circ D_i$ instead of Γ_i in this section.

5.5.6. Lemma. *$\mathrm{SA}_\alpha(h) \geq \mathrm{SA}_\beta(f)$.*

Proof. Let Γ_f be an optimal spectral matrix saturating the spectral bound

$$\mathrm{SA}_\beta(f) = \min_{i=1}^{k} \left\{ \beta_i \frac{\lambda(\Gamma_f)}{\lambda(\Gamma_f \circ D_i)} \right\} \ .$$

Similarly, for each $i = 1, \dots, k$, let Γ_{g_i} be an optimal spectral matrix saturating the spectral bound

$$\mathrm{SA}_{\alpha^i}(g_i) = \min_{j=1}^{n_i} \left\{ \alpha_j^i \frac{\lambda(\Gamma_{g_i})}{\lambda(\Gamma_{g_i} \circ D_j)} \right\} \ .$$

Due to the maximization over all matrices Γ, the spectral bound of the composite function h is at least

$$\mathrm{SA}_\alpha(h) \geq \min_{\ell=1}^{n} \left\{ \alpha_\ell \frac{\lambda(\Gamma_h)}{\lambda(\Gamma_h \circ D_\ell)} \right\} \ , \tag{5.14}$$

where Γ_h depends on Γ_f and Γ_{g_i} according to Definition 5.5.4. Recall that

$$\Gamma_h[x,y] = \Gamma_f[\tilde{x},\tilde{y}] \cdot \prod_{i=1}^{k} \Gamma^\star_{g_i}[x^i,y^i] \ .$$

Theorem 5.5.5 gives an exact expression for $\lambda(\Gamma_h)$. We now compute $\lambda(\Gamma_h \circ D_\ell)$ for $\ell = 1,\dots,n$. Let the ℓth input bit be the jth bit in the ith block. Recall the notation $x_\ell = (x^i)_j = x^i_j$. We prove that

$$(\Gamma_h \circ D_\ell)[x,y] = (\Gamma_f \circ D_i)[\tilde{x},\tilde{y}] \cdot (\Gamma_{g_i} \circ D_j)^\star[x^i,y^i] \cdot \prod_{e \neq i} \Gamma^\star_{g_e}[x^e,y^e] \ .$$

There are the following cases:

1. $x_\ell = y_\ell$: the left side is zero due to $(\Gamma_h \circ D_\ell)[x,y] = 0$.
   (a) $\tilde{x}_i = \tilde{y}_i$: the right side is zero due to $(\Gamma_f \circ D_i)[\tilde{x},\tilde{y}] = 0$.
   (b) $\tilde{x}_i \neq \tilde{y}_i$: the right side is zero due to $(\Gamma_{g_i} \circ D_j)^\star[x^i,y^i] = 0$. That holds because $(\Gamma_{g_i} \circ D_j)[x^i,y^i] = 0$ since $x^i_j = y^i_j$, and $I[x^i,y^i] = 0$ since $x^i \neq y^i$.
2. $x_\ell \neq y_\ell$, $\tilde{x}_i = \tilde{y}_i$: the left side is zero due to $\Gamma^\star_{g_i}[x^i,y^i] = 0$. That holds because $\Gamma_{g_i}[x^i,y^i] = 0$ since $g_i(x^i) = g_i(y^i)$, and $I[x^i,y^i] = 0$, since $x^i \neq y^i$. The right side is also zero due to $(\Gamma_f \circ D_i)[\tilde{x},\tilde{y}] = 0$.
3. $x_\ell \neq y_\ell$, $\tilde{x}_i \neq \tilde{y}_i$: both sides are equal, because all multiplications by D_i, D_j, D_ℓ are multiplications by 1.

Since $\Gamma_h \circ D_\ell$ has the same structure as Γ_h, by Theorem 5.5.5,

$$\lambda(\Gamma_h \circ D_\ell) = \lambda(\Gamma_f \circ D_i) \cdot \lambda(\Gamma_{g_i} \circ D_j) \cdot \prod_{e \neq i} \lambda(\Gamma_{g_e}) \ .$$

By dividing the two spectral norms from equation (5.14),

$$\frac{\lambda(\Gamma_h)}{\lambda(\Gamma_h \circ D_\ell)} = \frac{\lambda(\Gamma_f)}{\lambda(\Gamma_f \circ D_i)} \cdot \frac{\lambda(\Gamma_{g_i})}{\lambda(\Gamma_{g_i} \circ D_j)} \ .$$

Since the spectral adversary maximizes over all Γ_h, we conclude that

$$\begin{aligned}
\mathrm{SA}_\alpha(h) &\geq \min_{\ell=1}^{n} \frac{\lambda(\Gamma_h)}{\lambda(\Gamma_h \circ D_\ell)} \cdot \alpha_\ell \\
&= \min_{i=1}^{k} \min_{j=1}^{n_i} \frac{\lambda(\Gamma_f)}{\lambda(\Gamma_f \circ D_i)} \cdot \frac{\lambda(\Gamma_{g_i})}{\lambda(\Gamma_{g_i} \circ D_j)} \cdot \alpha^i_j \\
&= \min_{i=1}^{k} \frac{\lambda(\Gamma_f)}{\lambda(\Gamma_f \circ D_i)} \cdot \mathrm{SA}_{\alpha^i}(g_i) \\
&= \min_{i=1}^{k} \frac{\lambda(\Gamma_f)}{\lambda(\Gamma_f \circ D_i)} \cdot \beta_i \\
&= \mathrm{SA}_\beta(f) \ ,
\end{aligned}$$

which we had to prove. □

Composition of the minimax bound

5.5.7. Lemma. $\mathrm{MM}_\alpha(h) \leq \mathrm{MM}_\beta(f)$.

Proof. Let p^f and p^{g_i} for $i = 1, \dots, k$ be optimal sets of probability distributions saturating the minimax bounds

$$\begin{aligned}
\mathrm{MM}_\beta(f) &= \max_{\substack{x,y \\ f(x)\neq f(y)}} \frac{1}{\sum_{i:x_i\neq y_i} \sqrt{p^f_x(i) p^f_y(i)}/\beta_i} \ , \\
\mathrm{MM}_{\alpha^i}(g_i) &= \max_{\substack{x,y \\ g_i(x)\neq g_i(y)}} \frac{1}{\sum_{j:x_j\neq y_j} \sqrt{p^{g_i}_x(j) p^{g_i}_y(j)}/\alpha^i_j} \ .
\end{aligned}$$

Define the set of probability distributions p^h as $p^h_x(\ell) = p^f_{\tilde{x}}(i) p^{g_i}_{x^i}(j)$, where the ℓth input bit is the jth bit in the ith block. This construction was first used by Laplante, Lee, and Szegedy [LLS06]. We claim that $\mathrm{MM}_\alpha(h) \leq \mathrm{MM}_\beta(f)$. The minimax bound is a minimization problem. If we plug in p^h, we get an upper bound on $\mathrm{MM}_\alpha(h)$ as follows:

$$\begin{aligned}
\mathrm{MM}_\alpha(h) &\leq \max_{\substack{x,y \\ h(x)\neq h(y)}} \frac{1}{\sum_{\ell:x_\ell\neq y_\ell} \sqrt{p^h_x(\ell) p^h_y(\ell)}/\alpha_\ell} \\
&= 1 \Big/ \min_{\substack{x,y \\ h(x)\neq h(y)}} \sum_{\ell:x_\ell\neq y_\ell} \sqrt{p^f_{\tilde{x}}(i) p^f_{\tilde{y}}(i)} \sqrt{p^{g_i}_{x^i}(j) p^{g_i}_{y^i}(j)}/\alpha^i_j \\
&= 1 \Big/ \min_{\substack{\tilde{x},\tilde{y} \\ f(\tilde{x})\neq f(\tilde{y})}} \sum_i \sqrt{p^f_{\tilde{x}}(i) p^f_{\tilde{y}}(i)} \min_{\substack{x^i,y^i \\ g_i(x^i)=\tilde{x}_i \\ g_i(y^i)=\tilde{y}_i}} \sum_{j:x^i_j\neq y^i_j} \sqrt{p^{g_i}_{x^i}(j) p^{g_i}_{y^i}(j)}/\alpha^i_j
\end{aligned}$$

$$
\begin{aligned}
&\leq 1 \Big/ \min_{\substack{\tilde{x},\tilde{y} \\ f(\tilde{x})\neq f(\tilde{y})}} \sum_{i:\tilde{x}_i\neq\tilde{y}_i} \sqrt{p^f_{\tilde{x}}(i)p^f_{\tilde{y}}(i)} \min_{\substack{x^i,y^i \\ g_i(x^i)\neq g_i(y^i)}} \sum_{j:x^i_j\neq y^i_j} \sqrt{p^{g_i}_{x^i}(j)p^{g_i}_{y^i}(j)}/\alpha^i_j \\
&= 1 \Big/ \min_{\substack{\tilde{x},\tilde{y} \\ f(\tilde{x})\neq f(\tilde{y})}} \sum_{i:\tilde{x}_i\neq\tilde{y}_i} \sqrt{p^f_{\tilde{x}}(i)p^f_{\tilde{y}}(i)} \;/\; \mathrm{MM}_{\alpha^i}(g_i) \\
&= 1 \Big/ \min_{\substack{\tilde{x},\tilde{y} \\ f(\tilde{x})\neq f(\tilde{y})}} \sum_{i:\tilde{x}_i\neq\tilde{y}_i} \sqrt{p^f_{\tilde{x}}(i)p^f_{\tilde{y}}(i)} \;/\; \beta_i \\
&= \mathrm{MM}_\beta(f) \enspace ,
\end{aligned}
$$

where the second inequality follows from that fact that we have removed $i : \tilde{x}_i = \tilde{y}_i$ from the sum. We conclude that $\mathrm{MM}_\alpha(h) \leq \mathrm{MM}_\beta(f)$. □

5.5.8. Remark. Laplante, Lee, and Szegedy [LLS06] proved a similar bound in a stronger setting where the sub-functions g_i can act on the same input bits. They did not allow costs of input bits. Their setting is, however, not relevant for our result, because the composition bound for $\mathrm{SA}_\alpha(h)$ does not hold in this setting.

The composite bound is tight

5.5.9. Theorem. *Let h be a composite function of the form $h = f \circ (g_1, \dots, g_k)$, and let $\alpha = \mathbb{R}^n_+$ be any cost vector. Then*

$$\mathrm{Adv}_\alpha(h) = \mathrm{Adv}_\beta(f) \enspace ,$$

where $\alpha = (\alpha^1, \dots, \alpha^k)$, $\beta = (\beta_1, \dots, \beta_k)$, and $\beta_i = \mathrm{Adv}_{\alpha^i}(g_i)$.

Proof. By Theorem 5.5.2,

$$
\begin{aligned}
\mathrm{Adv}_\alpha(h) &= \mathrm{SA}_\alpha(h) = \mathrm{MM}_\alpha(h) \\
\mathrm{Adv}_\beta(f) &= \mathrm{SA}_\beta(f) = \mathrm{MM}_\beta(f) \enspace .
\end{aligned}
$$

By Lemma 5.5.6, $\mathrm{SA}_\alpha(h) \geq \mathrm{SA}_\beta(f)$. By Lemma 5.5.7, $\mathrm{MM}_\alpha(h) \leq \mathrm{MM}_\beta(f)$. We conclude that $\mathrm{Adv}_\alpha(h) = \mathrm{Adv}_\beta(f)$. □

5.6 Summary

We have cleaned up the forest of quantum adversary methods and shown that all of them are equivalent. We have generalized the adversary method to non-Boolean functions and proved a limitation on the best achievable bound in terms of the certificate complexity of the function. The adversary bound can be expressed as an optimal solution of a semidefinite program. Thanks to the duality of semidefinite programming, one can efficiently find both upper- and lower-bounds on the adversary bound. We have found an exact expression for the adversary bound for a composite function in terms of adversary bounds for the individual composed functions.

Chapter 6

Direct Product Theorems

This chapter is based on the following papers:

[KŠW04] H. Klauck, R. Špalek, and R. de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. In *Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 12–21, 2004. To appear in SIAM Journal on Computing.

[AŠW06] A. Ambainis, R. Špalek, and R. de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. In *Proceedings of 38th Annual ACM Symposium on Theory of Computing*, pages 618–633, 2006.

6.1 Introduction

For every reasonable model of computation one can ask the following fundamental question.

6.1.1. Question. How do the resources that we need for computing k independent instances of a function f scale with the resources needed for one instance and with k?

Here the notion of *resource* needs to be specified. It could refer to time, space, queries, communication etc. Similarly we need to define what we mean by *computing* f, for instance whether we allow the algorithm some probability of error, and whether this probability of error is average-case or worst-case.

Direct product theorems In this chapter we consider two kinds of resources, queries and communication, and allow our algorithms some error probability. An algorithm is given k inputs $x^1, \dots, x^k$, and has to output the vector of k answers $f(x^1), \dots, f(x^k)$. The issue is how the algorithm can optimally distribute its resources among the k instances it needs to compute. We focus on the relation between the total amount T of resources available and the best-achievable success probability σ (which could be average-case or worst-case). Intuitively, if every algorithm with t resources must have some constant error probability when computing one instance of f, then for computing k instances we expect a constant error on each instance and hence an exponentially small success probability for the k-vector as a whole. Such a statement is known as a *weak direct product theorem*:

$$\text{If } T \approx t, \text{ then } \sigma \leq 2^{-\Omega(k)}.$$

Here "$T \approx t$" informally means that T is not much smaller than t. However, even if we give our algorithm roughly kt resources, on average it still has only t resources available per instance. So even here we expect a constant error per instance and an exponentially small success probability overall. Such a statement is known as a *strong direct product theorem*:

$$\text{If } T \approx kt, \text{ then } \sigma \leq 2^{-\Omega(k)}.$$

Strong direct product theorems, though intuitively very plausible, are generally hard to prove and sometimes not even true. Shaltiel [Sha01] exhibits a general class of examples where strong direct product theorems fail. This applies for instance to query complexity, communication complexity, and circuit complexity. In his examples, success probability is taken under the uniform probability distribution on inputs. The function is chosen such that for most inputs, most of the k instances can be computed quickly and without any error probability. This leaves enough resources to solve the few hard instances with high success probability. Hence for his functions, with $T \approx tk$, one can achieve average success probability close to 1.

Accordingly, we can only establish direct product theorems in special cases. Examples are Nisan et al.'s [NRS94] strong direct product theorem for *decision forests*, Parnafes et al.'s [PRW97] direct product theorem for *forests of communication protocols*, Shaltiel's strong direct product theorems for *fair decision trees* and his discrepancy bound for communication complexity [Sha01]. Shaltiel's result for discrepancy was recently strengthened to a strong direct product theorem for the *corruption measure* under product distributions on the inputs by Beame et al. [BPSW05]. There also has been recent progress on the related issue of direct sum results, see for example [CSWY01, BJKS02b, BJKS02a] and the references therein. A *direct sum theorem* states that computing k instances with overall error ε requires roughly k times as many resources as computing one instance with error ε. Clearly, strong direct product theorems alway imply direct sum results, since they state the same resource lower bounds even for algorithms whose overall error is allowed to be exponentially close to 1, rather than at most ε.

In the quantum case, much less work has been done. Aaronson [Aar04a, Theorem 10] established a direct product result for the unordered search problem that lies in between the weak and the strong theorems. Our main contributions in this chapter are strong direct product theorems for the OR function in various settings.

Classical query complexity First consider the case of classical randomized algorithms. Let OR_n denote the n-bit OR function, and let $f^{(k)}$ denote k independent instances of a function f. Any randomized algorithm with less than, say, $n/2$ queries will have a constant error probability when computing OR_n. Hence we expect an exponentially small success probability when computing $\mathrm{OR}_n^{(k)}$ using $\ll kn$ queries. We prove this in Section 6.2.

6.1.2. PROPOSITION. (DPT FOR CLASSICAL QUERY COMPLEXITY)
Every randomized algorithm for $\mathrm{OR}_n^{(k)}$ using $T \leq \alpha kn$ queries has worst-case success probability $\sigma \leq 2^{-\Omega(k)}$ (formally, there exists a small constant $\alpha > 0$ such that the statement holds for all k, n; we will skip these quantifiers later).

For simplicity we have stated this result with σ being *worst-case success probability*, but the statement is also valid for the *average success probability* under a hard k-fold product distribution that is implicit in our proof.

This statement for OR actually implies a somewhat weaker DPT for all total Boolean functions f, via the notion of block sensitivity $bs(f)$; see Definition 2.5.4. Using techniques of Nisan and Szegedy [NS94], we can embed $\mathrm{OR}_{bs(f)}$ in f (with the promise that the weight of the OR's input is 0 or 1), while on the other hand we know that the classical bounded-error query complexity $R_2(f)$ is upper-bounded by $bs(f)^3$ [BBC⁺01]. This implies the following.

6.1.3. PROPOSITION. *For every total Boolean f: Every randomized algorithm for $f^{(k)}$ using $T \leq \alpha k R_2(f)^{1/3}$ queries has success probability $\sigma \leq 2^{-\Omega(k)}$.*

This theorem falls short of a true strong direct product theorem in having $R_2(f)^{1/3}$ instead of $R_2(f)$ in the resource bound. However, the other two main aspects of a DPT remain valid: the linear dependence of the resources on k and the exponential decay of the success probability.

Quantum query complexity Next we turn our attention to quantum algorithms. Buhrman et al. [BNRW05] actually proved that roughly k times the resources for one instance suffices to compute $f^{(k)}$ with success probability close to 1, rather than exponentially small: $Q_2(f^{(k)}) = O(kQ_2(f))$, where $Q_2(f)$ denotes the quantum bounded-error query complexity of f (such a result is not known to hold in the classical world). For instance, $Q_2(\mathrm{OR}_n) = \Theta(\sqrt{n})$ by algorithm GROVER SEARCH, so $O(k\sqrt{n})$ quantum queries suffice to compute $\mathrm{OR}_n^{(k)}$

with high success probability. In Section 6.3 we show that if we make the number of queries slightly smaller, the best-achievable success probability suddenly becomes exponentially small.

6.1.4. Proposition. (DPT for quantum query complexity)
Every quantum algorithm for $\mathrm{OR}_n^{(k)}$ *using* $T \leq \alpha k\sqrt{n}$ *queries has success probability* $\sigma \leq 2^{-\Omega(k)}$.

Our proof uses the polynomial method [BBC$^+$01] and is completely different from the classical proof. The polynomial method was also used by Aaronson [Aar04a] in his proof of a weaker quantum direct product theorem for the search problem, mentioned above. Our proof takes its starting point from his proof, analyzing the degree of a single-variate polynomial that is 0 on $\{0, \ldots, k-1\}$, at least σ on k, and between 0 and 1 on $\{0, \ldots, kn\}$. The difference between his proof and ours is that we partially factor this polynomial, which gives us some nice extra properties over Aaronson's approach of differentiating the polynomial, and we use a strong result of Coppersmith and Rivlin [CR92]. In both cases (different) extremal properties of Chebyshev polynomials finish the proofs.

Again, using block sensitivity we can obtain a weaker result for all total Boolean functions.

6.1.5. Proposition. *For every total Boolean* f*: Every quantum algorithm for* $f^{(k)}$ *using* $T \leq \alpha k Q_2(f)^{1/6}$ *queries has success probability* $\sigma \leq 2^{-\Omega(k)}$.

We also prove a slightly stronger direct product theorem for threshold functions. This result is worse than the DPT for OR in applying only to *1-sided error* quantum algorithms, those are algorithms whose 1-bits in the k-bit output vector are always correct; but it's better in giving a much stronger upper bound on the success probability. In Chapter 7, we develop a new version of the quantum adversary method and prove a proper DPT for all threshold functions, however we have not been able to prove it using the polynomial method. Let $\mathrm{Thr}_{t,n}$ denote the t-threshold function on n bits.

6.1.6. Proposition. *Every 1-sided error quantum algorithm for* $\mathrm{Thr}_{t,n}^{(k)}$ *using* $T \leq \alpha k Q_2(\mathrm{Thr}_{t,n})$ *queries has success probability* $\sigma \leq 2^{-\Omega(kt)}$.

A similar theorem can be proven for the k-fold t-search problem, where in each of k inputs of n bits, we want to find at least t ones. The different error bounds $2^{-\Omega(kt)}$ and $2^{-\Omega(k)}$ for 1-sided and 2-sided error algorithms intuitively say that imposing the 1-sided error constraint makes deciding each of the k individual t-threshold problems as hard as actually *finding* t ones in each of the k inputs.

Quantum communication complexity The third and last setting where we establish a strong direct product theorem is quantum communication complexity. Suppose Alice has an n-bit input x and Bob has an n-bit input y. These x and y represent sets, and $\mathrm{Disj}_n(x, y) = 1$ if and only if those sets are disjoint. Note that Disj_n is the negation of $\mathrm{OR}_n(x \wedge y)$, where $x \wedge y$ is the n-bit string obtained by bitwise AND-ing x and y. In many ways, Disj_n has the same central role in communication complexity as OR_n has in query complexity. In particular, it is *co-NP complete* [BFS86]. The communication complexity of Disj_n has been well studied: it takes $\Theta(n)$ bits of communication in the classical world [KS92, Raz92] and $\Theta(\sqrt{n})$ in the quantum world [BCW98, HW02, AA03, Raz03]. For the case where Alice and Bob want to compute k instances of Disjointness, we establish a strong direct product theorem in Section 6.4.

6.1.7. Proposition. (DPT for quantum communication complexity) *Every quantum protocol for* $\mathrm{Disj}_n^{(k)}$ *using* $T \leq \alpha k \sqrt{n}$ *qubits of communication has success probability* $\sigma \leq 2^{-\Omega(k)}$.

Our proof uses Razborov's [Raz03] lower-bound technique to translate the quantum protocol to a polynomial, at which point the polynomial results established for the quantum query DPT take over. We can obtain similar results for other symmetric predicates. The same bound was later obtained independently by Beame et al. [BPSW05, Corollary 9] for classical protocols under a specific input distribution, as a corollary of their strong direct product theorem for corruption. We conjecture that the optimal result in the classical case has a communication bound of αkn rather than $\alpha k\sqrt{n}$, but cannot prove this.

The stronger direct product theorem for 1-sided error threshold functions can also be translated to the communication setting.

Parity of the outcomes One may also consider algorithms that compute the *parity* of the k outcomes instead of the vector of k outcomes. This issue has been well studied, particularly in circuit complexity, and generally goes under the name of *XOR lemmas* [Yao82, GNW95]. In this chapter we focus mostly on the vector version, but we can prove similar strong bounds for the parity version. In particular, we state a classical strong XOR lemma in Section 6.2.3 and can get similar strong XOR lemmas for the quantum case using the technique of Cleve et al. [CDNT98, Section 3]. They show how the ability to compute the parity of any subset of k bits with probability $1/2 + \varepsilon$, suffices to compute the full k-vector with probability $4\varepsilon^2$. Hence our strong quantum direct product theorems imply strong quantum XOR lemmas.

6.2 Classical DPT for OR

In this section we prove a strong direct product theorem for classical randomized algorithms computing k independent instances of OR_n. By Yao's principle [Yao77], it is sufficient to prove it for deterministic algorithms under a fixed hard input distribution.

6.2.1 Non-adaptive algorithms

We first establish a strong direct product theorem for non-adaptive algorithms. We call an algorithm *non-adaptive* if, for each of the k input blocks, the maximum number of queries in that block is fixed before the first query. Note that this definition is non-standard in fixing only the *number* of queries in each block, rather than fixing all queried *indices* in advance. Let $\mathrm{Suc}_{t,\mu}(f)$ be the success probability of the best algorithm for f under μ that queries at most t input bits.

6.2.1. Lemma. *Let $f : \{0,1\}^n \to \{0,1\}$ and μ be an input distribution. Every non-adaptive deterministic algorithm for $f^{(k)}$ under μ^k using $T \le kt$ queries has success probability $\sigma \le \mathrm{Suc}_{t,\mu}(f)^k$.*

Proof. The proof has two steps. First, we prove by induction that non-adaptive algorithms for $f^{(k)}$ under general product distribution $\mu_1 \times \cdots \times \mu_k$ that spend t_i queries in the i^{th} input x^i have success probability at most $\prod_{i=1}^k \mathrm{Suc}_{t_i,\mu_i}(f)$. Second, we argue that, when $\mu_i = \mu$, the value is maximal for $t_i = t$.

Following [Sha01, Lemma 7], we prove the first part by induction on $T = t_1 + \cdots + t_k$. If $T = 0$, then the algorithm has to guess k independent random variables $x^i \sim \mu_i$. The probability of success is equal to the product of the individual success probabilities, that is $\prod_{i=1}^k \mathrm{Suc}_{0,\mu_i}(f)$.

For the induction step $T \Rightarrow T+1$: pick some $t_i \neq 0$ and consider two input distributions $\mu'_{i,0}$ and $\mu'_{i,1}$ obtained from μ_i by fixing the queried bit x^i_j (the j^{th} bit in the i^{th} input). By the induction hypothesis, for each value $b \in \{0,1\}$, there is an optimal non-adaptive algorithm A_b that achieves the success probability $\mathrm{Suc}_{t_i-1,\mu'_{i,b}}(f) \cdot \prod_{j\neq i} \mathrm{Suc}_{t_j,\mu_j}(f)$. We construct a new algorithm A that calls A_b as a subroutine after it has queried x^i_j with b as an outcome. A is optimal and it has success probability

$$\left(\sum_{b=0}^{1} \Pr_{\mu_i}[x^i_j = b] \cdot \mathrm{Suc}_{t_i-1,\mu'_{i,b}}(f)\right) \cdot \prod_{j\neq i} \mathrm{Suc}_{t_j,\mu_j}(f) = \prod_{i=1}^{k} \mathrm{Suc}_{t_i,\mu_i}(f) \ .$$

Since we are dealing with non-adaptive algorithms here, symmetry reasons imply: if all k instances x^i are independent and identically distributed, then the optimal distribution of queries $t_1 + \cdots + t_k = kt$ is uniform, that is $t_i = t$. (Note that counterexamples to the analogous property for the general non-adaptive case, like the ones given in Remark 6.2.2 below, do not apply here.) In such a case, the algorithm achieves the success probability $\mathrm{Suc}_{t,\mu}(f)^k$. $\square$

6.2.2 Adaptive algorithms

In this section we prove a similar statement also for adaptive algorithms.

6.2.2. Remark. The strong direct product theorem is not always true for adaptive algorithms. Following [Sha01], define $h(x) = x_1 \vee (x_2 \oplus \cdots \oplus x_n)$. Clearly $\mathrm{Suc}_{\frac{2}{3}n,\mu}(h) = \frac{3}{4}$ for μ uniform. By the Chernoff bound, $\mathrm{Suc}_{\frac{2}{3}nk,\mu^k}(h^{(k)}) = 1 - 2^{-\Omega(k)}$, because approximately half of the blocks can be solved using just 1 query and the unused queries can be used to answer exactly also the other half of the blocks.

However, the strong direct product theorem is valid for $\mathrm{OR}_n^{(k)}$ under ν^k, where $\nu(0^n) = \frac{1}{2}$ and $\nu(e_i) = \frac{1}{2n}$ for e_i an n-bit string that contains a 1 only at the i^{th} position. It is simple to prove that $\mathrm{Suc}_{\alpha n,\nu}(\mathrm{OR}_n) = \frac{\alpha+1}{2}$. Non-adaptive algorithms for $\mathrm{OR}_n^{(k)}$ under ν^k with αkn queries thus have $\sigma \leq (\frac{\alpha+1}{2})^k = 2^{-\log_2(\frac{2}{\alpha+1})k}$. We can achieve any $\gamma < 1$ by choosing α sufficiently small. Here we prove that adaptive algorithms cannot be much better. Without loss of generality, we assume:

1. The adaptive algorithm is deterministic. By Yao's principle [Yao77], if there exists a randomized algorithm with success probability σ under some input distribution, then there exists a deterministic algorithm with success probability σ under that distribution.

2. Whenever the algorithm finds a 1 in some input block, it stops querying that block.

3. The algorithm spends the same number of queries in all blocks where it does not find a 1. This is optimal due to the symmetry between the blocks (we omit the straightforward calculation that justifies this). It implies that the algorithm spends at least as many queries in each *empty* input block as in each *non-empty* block.

6.2.3. Lemma. *If there is an adaptive T-query algorithm A computing $\mathrm{OR}_n^{(k)}$ under ν^k with success probability σ, then there is a non-adaptive $3T$-query algorithm A' computing $\mathrm{OR}_n^{(k)}$ with success probability $\sigma - 2^{-\Omega(k)}$.*

Proof. Let Z be the number of empty blocks. $\mathrm{E}[Z] = \frac{k}{2}$ and, by the Chernoff bound, $\delta = \Pr[Z < \frac{k}{3}] = 2^{-\Omega(k)}$. If $Z \geq \frac{k}{3}$, then A spends at most $\frac{3T}{k}$ queries in each empty block. Define non-adaptive A' that spends $\frac{3T}{k}$ queries in *each* block.

Then A' queries all the positions that A queries, and maybe some more. Compare the overall success probabilities of A and A':

$$\begin{aligned}\sigma_A &= \Pr\left[Z < \tfrac{k}{3}\right] \cdot \Pr\left[A \text{ succeeds} \mid Z < \tfrac{k}{3}\right] \\ &\quad + \Pr\left[Z \geq \tfrac{k}{3}\right] \cdot \Pr\left[A \text{ succeeds} \mid Z \geq \tfrac{k}{3}\right] \\ &\leq \delta \cdot 1 + \Pr\left[Z \geq \tfrac{k}{3}\right] \cdot \Pr\left[A' \text{ succeeds} \mid Z \geq \tfrac{k}{3}\right] \\ &\leq \delta + \sigma_{A'} \ .\end{aligned}$$

We conclude that $\sigma_{A'} \geq \sigma_A - \delta$. (*Remark.* By replacing the $\frac{k}{3}$-bound on Z by a βk-bound for some $\beta > 0$, we can obtain arbitrary $\gamma < 1$ in the exponent $\delta = 2^{-\gamma k}$, while the number of queries of A' becomes T/β.) □

Combining the two lemmas establishes the following theorem.

6.2.4. Theorem (DPT for OR). *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that every randomized algorithm for $\mathrm{OR}_n^{(k)}$ using $T \leq \alpha k n$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

6.2.3 A bound for the parity of the outcomes

Here we give a strong direct product theorem for the *parity* of k independent instances of OR_n. The parity is a Boolean variable, hence we can always guess it with probability at least $\frac{1}{2}$. However, we prove that the advantage (instead of the success probability) of our guess must be exponentially small.

Let X be a random bit with $\Pr[X = 1] = p$. We define the *advantage of X* by $\mathrm{Advantage}(X) = |2p - 1|$. Note that a uniformly distributed random bit has advantage 0 and a bit known with certainty has advantage 1. It is well known that if $X_1, \dots, X_k$ are independent random bits, then $\mathrm{Advantage}(X_1 \oplus \cdots \oplus X_k) = \prod_{i=1}^k \mathrm{Advantage}(X_i)$. Compare this with the fact that the probability of guessing correctly the complete vector $(X_1, \dots, X_k)$ is the product of the individual probabilities.

We have proved a lower bound for the computation of $\mathrm{OR}_n^{(k)}$ (vector of OR's). By the same technique, replacing the success probability by the advantage in all claims and proofs, we can also prove a lower bound for the computation of $\mathrm{OR}_n^{\oplus k}$ (parity of OR's).

6.2.5. Theorem (DPT for parity of OR's). *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that every randomized algorithm for $\mathrm{OR}_n^{\oplus k}$ using $T \leq \alpha k n$ queries has advantage $\tau \leq 2^{-\gamma k}$.*

6.2.4 A bound for all functions

Here we show that the strong direct product theorem for OR actually implies a weaker direct product theorem for all functions. In this weaker version, the success probability of computing k instances still goes down exponentially with k, but we need to start from a polynomially smaller bound on the overall number of queries.

Recall Definition 2.5.4 of block sensitivity. Block sensitivity is closely related to deterministic and bounded-error classical query complexity as follows.

6.2.6. Theorem ([Nis91, BBC+01]). *$R_2(f) = \Omega(bs(f))$ for all f, $D(f) \leq bs(f)^3$ for all total Boolean f.*

Nisan and Szegedy [NS94] showed how to embed a $bs(f)$-bit OR function (with the promise that the input has weight at most 1) into f. Combined with our strong direct product theorem for OR, this implies a direct product theorem for all functions in terms of their block sensitivity.

6.2.7. Theorem. *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that for every f, every classical algorithm for $f^{(k)}$ using $T \leq \alpha k bs(f)$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

This is optimal whenever $R_2(f) = \Theta(bs(f))$, which is the case for most functions. For total functions, the gap between $R_2(f)$ and $bs(f)$ is not more than cubic [BBC+01].

6.2.8. Corollary. *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that for every total Boolean f, every classical algorithm for $f^{(k)}$ using $T \leq \alpha k R_2(f)^{1/3}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

6.3 Quantum DPT for OR

In this section we prove strong direct product theorems for quantum algorithms computing k independent instances of OR, and for quantum algorithms computing k independent instances of 1-sided error threshold functions. Our proofs rely on the polynomial method of [BBC+01]; see Section 2.6.

6.3.1 Bounds on polynomials

We use three results about polynomials, also used in [BCWZ99]. The first is by Coppersmith and Rivlin [CR92, p. 980] and gives a general bound for polynomials bounded by 1 at integer points.

6.3.1. Theorem (Coppersmith & Rivlin [CR92]). *Every polynomial p that has degree $d \leq n$ and absolute value*

$$|p(i)| \leq 1 \text{ for all integers } i \in [0, n] \ ,$$

satisfies

$$|p(x)| < ae^{bd^2/n} \text{ for all real } x \in [0,n] \ ,$$

where $a, b > 0$ are universal constants (no explicit values for a and b are given in [CR92]).

The other two results concern the Chebyshev polynomials T_d, defined by (see for example [Riv90]):

$$T_d(x) = \frac{1}{2}\left(\left(x+\sqrt{x^2-1}\right)^d + \left(x-\sqrt{x^2-1}\right)^d\right) \ .$$

T_d has degree d and its absolute value $|T_d(x)|$ is bounded by 1 if $x \in [-1,1]$. On the interval $[1,\infty)$, T_d exceeds all others polynomials with those two properties ([Riv90, p.108] and [Pat92, Fact 2]).

6.3.2. Theorem. *If q is a polynomial of degree d such that $|q(x)| \leq 1$ for all $x \in [-1,1]$ then $|q(x)| \leq |T_d(x)|$ for all $x \geq 1$.*

Paturi [Pat92, before Fact 2] proved the following:

6.3.3. Lemma (Paturi [Pat92]). *$T_d(1+\mu) \leq e^{2d\sqrt{2\mu+\mu^2}}$ for all $\mu \geq 0$.*

Proof. For $x = 1+\mu$: $T_d(x) \leq (x+\sqrt{x^2-1})^d = (1+\mu+\sqrt{2\mu+\mu^2})^d \leq (1+2\sqrt{2\mu+\mu^2})^d \leq e^{2d\sqrt{2\mu+\mu^2}}$ (using that $1+z \leq e^z$ for all real z). □

Key lemma The following key lemma is the basis for our direct product theorems for the OR function.

6.3.4. Lemma. *Suppose p is a degree-D polynomial such that for some $\delta \geq 0$*

$$\begin{aligned} &p(i) \in [-\delta,\delta] && \text{for all } i \in \{0,\dots,k-1\} \\ &p(k) = \sigma && \\ &p(i) \in [-\delta,1+\delta] && \text{for all } i \in \{0,\dots,N\} \ . \end{aligned}$$

Then for every integer $1 \leq C < N-k$ and $\mu = \dfrac{2C}{N-k-C}$ we have

$$\begin{aligned}\sigma \leq a\left(1+\delta+\frac{\delta(2N)^k}{(k-1)!}\right)\cdot& \\ \cdot\exp\left(\frac{b(D-k)^2}{(N-k-C)} + 2(D-k)\sqrt{2\mu+\mu^2} - k\ln\left(\tfrac{C}{k}\right)\right) &+ \delta k 2^{k-1} \ ,\end{aligned}$$

where a, b are the constants given by Theorem 6.3.1.

Before establishing this gruesome bound, let us reassure the reader by noting that we will apply this lemma with δ either 0 or negligibly small, $D = \alpha\sqrt{kN}$ for sufficiently small α, and $C = ke^{\gamma+1}$, giving

$$\sigma \leq \exp\left((b\alpha^2 + 4\alpha e^{\gamma/2+1/2} - 1 - \gamma)k\right) \leq e^{-\gamma k} \leq 2^{-\gamma k} \ .$$

Proof of Lemma 6.3.4. Divide p with remainder by $\prod_{j=0}^{k-1}(x-j)$ to obtain

$$p(x) = q(x)\prod_{j=0}^{k-1}(x-j) + r(x) \ ,$$

where $d = \deg(q) = D - k$ and $\deg(r) \leq k-1$. We know that $r(x) = p(x) \in [-\delta, \delta]$ for all $x \in \{0, \dots, k-1\}$. Decompose r as a linear combination of polynomials e_i, where $e_i(i) = 1$ and $e_i(x) = 0$ for $x \in \{0, \dots, k-1\} - \{i\}$.

$$r(x) = \sum_{i=0}^{k-1} p(i)e_i(x) = \sum_{i=0}^{k-1} p(i) \prod_{\substack{j=0 \\ j\neq i}}^{k-1} \frac{x-j}{i-j}$$

We bound the values of r for all real $x \in [0, N]$ by

$$\begin{aligned}
|r(x)| &\leq \sum_{i=0}^{k-1} \frac{|p(i)|}{i!(k-1-i)!} \prod_{\substack{j=0 \\ j\neq i}}^{k-1} |x-j| \\
&\leq \frac{\delta}{(k-1)!} \sum_{i=0}^{k-1} \binom{k-1}{i} N^k \leq \frac{\delta(2N)^k}{(k-1)!} \ , \\
|r(k)| &\leq \delta k 2^{k-1} \ .
\end{aligned}$$

This implies the following about the values of the polynomial q:

$$\begin{aligned}
|q(k)| &\geq \frac{\sigma - \delta k 2^{k-1}}{k!} \ , \\
|q(i)| &\leq \frac{(i-k)!}{i!}\left(1 + \delta + \frac{\delta(2N)^k}{(k-1)!}\right) && \text{for } i \in \{k, \dots, N\} \ .
\end{aligned}$$

In particular,

$$|q(i)| \leq C^{-k}\left(1 + \delta + \frac{\delta(2N)^k}{(k-1)!}\right) = A \qquad \text{for } i \in \{k+C, \dots, N\} \ .$$

Theorem 6.3.1 implies that there are constants $a, b > 0$ such that

$$|q(x)| \leq A \cdot a e^{bd^2/(N-k-C)} = B \qquad \text{for all real } x \in [k+C, N] \ .$$

We now divide q by B to normalize it, and re-scale the interval $[k+C,N]$ to $[1,-1]$ to get a degree-d polynomial t satisfying

$$\begin{aligned} |t(x)| &\leq 1 && \text{for all } x \in [-1,1] \ , \\ t(1+\mu) &= \frac{q(k)}{B} && \text{for } \mu = \frac{2C}{N-k-C} \ . \end{aligned}$$

Since t cannot grow faster than the degree-d Chebyshev polynomial, we get

$$t(1+\mu) \leq T_d(1+\mu) \leq e^{2d\sqrt{2\mu+\mu^2}} \ .$$

Combining our upper and lower bounds on $t(1+\mu)$, we obtain

$$\frac{(\sigma - \delta k 2^{k-1})/k!}{C^{-k}\left(1+\delta+\frac{\delta(2N)^k}{(k-1)!}\right) a e^{bd^2/(N-k-C)}} \leq e^{2d\sqrt{2\mu+\mu^2}} \ .$$

Rearranging gives the bound. □

Extended key lemma For the threshold function, we need the following more general version of the key lemma. It analyzes polynomials that are 0 on the first m integer points, and that significantly "jump" a bit later. We do not merge these two key lemmas for the sake of simplicity of the proofs. The second lemma allows for a more variable jump in the polynomial, on the other hand it requires the polynomial to be exactly zero at small integer values. The statement actually holds even when the values of the polynomial at small integer values are exponentially smaller than σ, rather than equal to 0. We do not need such a generalized version here, hence we rather simplify the statement.

6.3.5. Lemma. *Suppose E,N,m are integers satisfying $10 \leq E \leq \frac{N}{2m}$, and let p be a degree-D polynomial such that*

$$\begin{aligned} p(i) &= 0 && \text{for all } i \in \{0,\dots,m-1\} \\ p(8m) &= \sigma \\ p(i) &\in [0,1] && \text{for all } i \in \{0,\dots,N\} \ . \end{aligned}$$

Then $\sigma \leq 2^{O(D^2/N + D\sqrt{Em/N} - m\log E)}$.

Proof. Divide p by $\prod_{j=0}^{m-1}(x-j)$ to obtain

$$p(x) = q(x)\prod_{j=0}^{m-1}(x-j) \ ,$$

where $d = \deg(q) = D - m$. This implies the following about the values of the polynomial q:

$$\begin{aligned} |q(8m)| &\geq \frac{\sigma}{(8m)^m} \ , \\ |q(i)| &\leq \frac{1}{((E-1)m)^m} && \text{for } i \in \{Em, \dots, N\} \ . \end{aligned}$$

Theorem 6.3.1 implies that there are constants $a, b > 0$ such that

$$|q(x)| \leq \frac{a}{((E-1)m)^m} e^{bd^2/(N-Em)} = B \qquad \text{for all real } x \in [Em, N] \ .$$

We now divide q by B to normalize it, and re-scale the interval $[Em, N]$ to $[1, -1]$ to get a degree-d polynomial t satisfying

$$\begin{aligned} |t(x)| &\leq 1 && \text{for all } x \in [-1, 1] \ , \\ t(1+\mu) &= \frac{q(8m)}{B} && \text{for } \mu = \frac{2(E-8)m}{N-Em} \ . \end{aligned}$$

Since t cannot grow faster than the degree-d Chebyshev polynomial, we have

$$t(1+\mu) \leq e^{2d\sqrt{2\mu+\mu^2}} \ .$$

Combining our upper and lower bounds on $t(1+\mu)$ gives

$$\frac{\sigma}{(8m)^m} \cdot \frac{((E-1)m)^m}{ae^{O(d^2/N)}} \leq e^{O(d\sqrt{Em/N})} \ ,$$

which implies the lemma. □

6.3.2 Consequences for quantum algorithms

The previous result about polynomials implies a strong tradeoff between queries and success probability for quantum algorithms that have to find k ones in an N-bit input. A *1-sided error k-threshold algorithm with success probability σ* is an algorithm on N-bit input x, that outputs 0 with certainty if $|x| < k$, and outputs 1 with probability at least σ if $|x| = k$.

6.3.6. Theorem. *For every $\gamma > 0$, there exists an $\alpha > 0$ such that every 1-sided error quantum k-threshold algorithm using $T \leq \alpha\sqrt{kN}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

Proof. Fix $\gamma > 0$ and consider a 1-sided error T-query k-threshold algorithm. By Theorem 2.6.2, its acceptance probability is an N-variate polynomial of degree $D \leq 2T \leq 2\alpha\sqrt{kN}$ and can be symmetrized to a single-variate polynomial p with the following properties:

$$\begin{aligned} p(i) &= 0 && \text{if } i \in \{0, \dots, k-1\} \\ p(k) &\geq \sigma && \\ p(i) &\in [0,1] && \text{for all } i \in \{0, \dots, N\} \ . \end{aligned}$$

Choosing $\alpha > 0$ sufficiently small and $\delta = 0$, the result follows from Lemma 6.3.4. □

Unordered search The above statement implies a strong direct product theorem for k instances of the n-bit search problem. For each such instance, the goal is to find the index of a 1-bit among the n inputs bits of the instance (or to report that none exists). We already studied this problem in the introductory section as Problem 1.3.1.

6.3.7. Theorem (DPT for Search). *For every $\gamma > 0$, there exists an $\alpha > 0$ such that every quantum algorithm for* $\text{Search}_n^{(k)}$ *using $T \leq \alpha k\sqrt{n}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

Proof. Set $N = kn$, fix a $\gamma > 0$ and a T-query algorithm A for $\text{Search}_n^{(k)}$ with success probability σ. Now consider the following algorithm that acts on an N-bit input x.

1. Apply a random permutation π to x.
2. Run A on $\pi(x)$.
3. Query each of the k positions that A outputs, and return 1 if and only if at least $\frac{k}{2}$ of those bits are 1.

This uses $T + k$ queries. We will show that it is a 1-sided error $\frac{k}{2}$-threshold algorithm. First, if $|x| < \frac{k}{2}$, it always outputs 0. Second, consider the case $|x| = \frac{k}{2}$. The probability that π puts all $\frac{k}{2}$ ones in distinct n-bit blocks is

$$\frac{N}{N} \cdot \frac{N-n}{N-1} \cdots \frac{N-\frac{k}{2}n}{N-\frac{k}{2}} \geq \left(\frac{N-\frac{k}{2}n}{N}\right)^{k/2} = 2^{-k/2} \ .$$

Hence our algorithm outputs 1 with probability at least $\sigma 2^{-k/2}$. Choosing α sufficiently small, the previous theorem implies $\sigma 2^{-k/2} \leq 2^{-(\gamma+1/2)k}$, hence $\sigma \leq 2^{-\gamma k}$. □

Our bounds are quite precise for $\alpha \ll 1$. We can choose $\gamma = 2\log_2(\frac{1}{\alpha}) - O(1)$ and ignore some lower-order terms to get roughly $\sigma \leq \alpha^{2k}$. On the other hand, it is known that Grover search with $\alpha\sqrt{n}$ queries on an n-bit input has success probability roughly α^2 [BBHT98]. Doing such a search on all k instances gives overall success probability α^{2k}.

The OR function The unordered search problem and OR are closely related.

6.3.8. Theorem (DPT for OR). *There are $\alpha, \gamma > 0$ such that every quantum algorithm for $\mathrm{OR}_n^{(k)}$ using $T \leq \alpha k\sqrt{n}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

Proof. An algorithm A for $\mathrm{OR}_n^{(k)}$ with success probability σ can be used to build an algorithm A' for $\mathrm{Search}_n^{(k)}$ with slightly worse success probability.

1. Run A on the original input and remember which blocks contain a 1.
2. Run simultaneously (at most k) binary searches on the nonzero blocks. Iterate this $s = 2\log_2(\frac{1}{\alpha})$ times. Each iteration is computed by running A on the parts of the blocks that are known to contain a 1, halving the remaining instance size each time.
3. Run the exact version of Grover search from Corollary 1.3.4 on each of the remaining parts of the instances to look for a one there; each remaining part has size $n/2^s$.

This new algorithm A' uses $(s+1)T + \frac{\pi}{4}k\sqrt{n/2^s} = O(\alpha\log_2(\frac{1}{\alpha})k\sqrt{n})$ queries. With probability at least σ^{s+1}, A succeeds in all iterations, in which case A' solves $\mathrm{Search}_n^{(k)}$. By the previous theorem, for every $\gamma' > 0$ of our choice we can choose $\alpha > 0$ such that

$$\sigma^{s+1} \leq 2^{-\gamma' k} ,$$

which implies the theorem with $\gamma = \frac{\gamma'}{s+1}$. □

Choosing our parameters carefully, we can actually show that for every $\gamma < 1$ there is an $\alpha > 0$ such that $\alpha k\sqrt{n}$ queries give success probability $\sigma \leq 2^{-\gamma k}$. Clearly, $\sigma = 2^{-k}$ is achievable without any queries by random guessing.

Threshold functions with 1-sided error Let $\mathrm{Thr}_{t,n}$ denote the t-threshold function on n bits. Theorem 6.3.6 uses the key lemma to give a (tight) lower bound for one instance of $\mathrm{Thr}_{k,N}$ with 1-sided error. Here we use the extended key lemma to obtain a strong direct product theorem for this problem. Note that we are changing the names of the variables here. In a way the following statement is a direct product theorem of direct product theorems. We say that an algorithm for $\mathrm{Thr}_{t,n}^{(k)}$ has *1-sided error* if the ones in its k-bit output vector are always correct.

6.3.9. Theorem. (DPT for 1-sided error Threshold)
There are $\alpha, \gamma > 0$ such that every 1-sided error quantum algorithm for $\mathrm{Thr}_{t,n}^{(k)}$ using $T \leq \alpha k Q_2(\mathrm{Thr}_{t,n})$ queries has success probability $\sigma \leq 2^{-\gamma kt}$.

Proof. We assume without loss of generality that $t \leq \frac{n}{20}$, the other cases can easily be reduced to this. Corollary 1.5.6 and Corollary 2.4.3 imply that $Q_2(\mathrm{Thr}_{t,n}) = \Theta(\sqrt{tn})$. Consider a quantum algorithm A with $T \leq \alpha k\sqrt{tn}$ queries that computes $\mathrm{Thr}_{t,n}^{(k)}$ with success probability σ. Roughly speaking, we use A to solve one big threshold problem on the total input, and then invoke the extended key lemma to upper-bound the success probability.

Define a new quantum algorithm B on an input x of $N = kn$ bits as follows. B runs A on a random permutation $\pi(x)$, and then outputs 1 if and only if the k-bit output vector has at least $\frac{k}{2}$ ones.

Let $m = \frac{kt}{2}$. Note that if $|x| < m$, then B always outputs 0 because the 1-sided error output vector must have fewer than $\frac{k}{2}$ ones. Now suppose $|x| = 8m = 4kt$. Call an n-bit input block *full* if $\pi(x)$ contains at least t ones in that block. Let F be the random variable counting how many of the k blocks are full. We claim that $\Pr[F \geq \frac{k}{2}] \geq \frac{1}{9}$. To prove this, observe that the number B of ones in one fixed block is a random variable distributed according to a hyper-geometric distribution ($4kt$ balls into N boxes, n of which count as success) with expectation $\mu = 4t$ and variance $V \leq 4t$. Using Chebyshev's inequality we bound the probability that this block is not full.

$$\Pr[B < t] \leq \Pr[|B - \mu| > 3t] \leq \Pr\left[|B - \mu| > \frac{3\sqrt{t}}{2}\sqrt{V}\right] < \frac{1}{(3\sqrt{t}/2)^2} \leq \frac{4}{9}$$

Hence the probability that the block is full ($B \geq t$) is at least $\frac{5}{9}$. This is true for each of the k blocks, so using linearity of expectation we have

$$\frac{5k}{9} \leq \mathrm{E}[F] \leq \Pr[F \geq \tfrac{k}{2}] \cdot k + (1 - \Pr[F \geq \tfrac{k}{2}]) \cdot \frac{k}{2} \ .$$

This implies $\Pr[F \geq \frac{k}{2}] \geq \frac{1}{9}$, as claimed. But then on all inputs with $|x| = 8m$, B outputs 1 with probability at least $\frac{\sigma}{9}$.

Algorithm B uses $\alpha k\sqrt{tn}$ queries. By Theorem 2.6.2 and symmetrization, the acceptance probability of B is a single-variate polynomial p of degree $D \leq 2\alpha k\sqrt{tn}$ such that

$$\begin{aligned} p(i) &= 0 && \text{for all } i \in \{0, \dots, m-1\} \\ p(8m) &\geq \tfrac{\sigma}{9} \\ p(i) &\in [0,1] && \text{for all } i \in \{0, \dots, N\} \ . \end{aligned}$$

The result now follows by applying Lemma 6.3.5 with $N = kn$, $m = \frac{kt}{2}$, $E = 10$, and α a sufficiently small positive constant. □

6.3.3 A bound for all functions

As in Section 6.2.4, we can extend the strong direct product theorem for OR to a slightly weaker theorem for all total functions. Block sensitivity is closely related to bounded-error quantum query complexity.

6.3.10. Theorem ([BBC$^+$01]). *$Q_2(f) = \Omega(\sqrt{bs(f)})$ for all f, $D(f) \leq bs(f)^3$ for all total Boolean f.*

By embedding an OR of size $bs(f)$ in f, we obtain the following:

6.3.11. Theorem. *There are $\alpha, \gamma > 0$ such that for every f, every quantum algorithm for $f^{(k)}$ using $T \leq \alpha k\sqrt{bs(f)}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

This is close to optimal whenever $Q_2(f) = \Theta(\sqrt{bs(f)})$. For total functions, the gap between $Q_2(f)$ and $\sqrt{bs(f)}$ is no more than a 6th power [BBC$^+$01].

6.3.12. Corollary. *There are $\alpha, \gamma > 0$ such that for every total Boolean f, every quantum algorithm for $f^{(k)}$ using $T \leq \alpha k Q_2(f)^{1/6}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

6.4 Quantum DPT for Disjointness

In this section we establish a strong direct product theorem for quantum communication complexity, specifically for protocols that compute k independent instances of the Disjointness problem. Our proof relies crucially on the beautiful technique that Razborov introduced to establish a lower bound on the quantum communication complexity of (one instance of) Disjointness [Raz03]. It allows us to translate a quantum communication protocol to a single-variate polynomial that represents, roughly speaking, the protocol's acceptance probability as a function of the size of the intersection of x and y. Once we have this polynomial, the results from Section 6.3.1 suffice to establish a strong direct product theorem.

6.4.1 Razborov's technique

Razborov's technique relies on the following linear algebraic notions. The *spectral norm* $\lambda(A)$ of a matrix A is its largest singular value σ_1. The *trace inner product* (a.k.a. Hilbert-Schmidt inner product) between A and B is $\langle A, B\rangle = \mathrm{Tr}\,(A^\dagger B)$. The *trace norm* is $\|A\|_{tr} = \max\{|\langle A, B\rangle| : \lambda(B) = 1\} = \sum_i \sigma_i$, the sum of all singular values of A. The *Frobenius norm* is $\|A\|_F = \sqrt{\sum_{ij} |A[i,j]|^2} = \sqrt{\sum_i \sigma_i^2}$. The following lemma is implicit in Razborov's paper.

6.4.1. Lemma ([Raz03]). *Consider a Q-qubit quantum communication protocol on N-bit inputs x and y, with acceptance probabilities denoted by $P(x,y)$. Define $P(i) = \mathrm{E}_{|x|=|y|=\frac{N}{4}, |x \wedge y|=i}[P(x,y)]$, where the expectation is taken uniformly over all x,y that each have weight $\frac{N}{4}$ and that have intersection i. For every $d \leq \frac{N}{4}$ there exists a degree-d polynomial q such that $|P(i) - q(i)| \leq 2^{-d/4+2Q}$ for all $i \in \{0,\ldots,\frac{N}{8}\}$.*

Proof. We only consider the $\mathcal{N} = \binom{N}{N/4}$ strings of weight $\frac{N}{4}$. Let P denote the $\mathcal{N} \times \mathcal{N}$ matrix of the acceptance probabilities on these inputs. We know from Yao and Kremer [Yao93, Kre95] that we can decompose P as a matrix product $P = AB$, where A is an $\mathcal{N} \times 2^{2Q-2}$ matrix with each entry at most 1 in absolute value, and similarly for B. Note that $\|A\|_F, \|B\|_F \leq \sqrt{\mathcal{N} 2^{2Q-2}}$. Using Hölder's inequality we have

$$\|P\|_{tr} \leq \|A\|_F \cdot \|B\|_F \leq \mathcal{N} 2^{2Q-2} \ .$$

Let μ_i denote the $\mathcal{N} \times \mathcal{N}$ matrix corresponding to the uniform probability distribution on $\{(x,y) : |x \wedge y| = i\}$. These *combinatorial matrices* have been well studied [Knu03]. Note that $\langle P, \mu_i \rangle$ is the expected acceptance probability $P(i)$ of the protocol under that distribution. One can show that the different μ_i commute, so they have the same eigenspaces $E_0, \ldots, E_{N/4}$ and can be simultaneously diagonalized by some orthogonal matrix U. For $t \in \{0,\ldots,\frac{N}{4}\}$, let $(UPU^T)_t$ denote the block of UPU^T corresponding to E_t, and $a_t = \mathrm{Tr}\left((UPU^T)_t\right)$ be its trace. Then we have

$$\sum_{t=0}^{N/4} |a_t| \leq \sum_{j=1}^{\mathcal{N}} \left|(UPU^T)[j,j]\right| \leq \|UPU^T\|_{tr} = \|P\|_{tr} \leq \mathcal{N} 2^{2Q-2} \ ,$$

where the second inequality is a property of the trace norm.

Let $\lambda_{i,t}$ be the eigenvalue of μ_i in eigenspace E_t. It is known [Raz03, Section 5.3] that $\lambda_{i,t}$ is a degree-t polynomial in i, and that $|\lambda_{i,t}| \leq 2^{-t/4}/\mathcal{N}$ for $i \leq \frac{N}{8}$ (the factor $\frac{1}{4}$ in the exponent is implicit in Razborov's paper). Consider the high-degree polynomial p defined by

$$p(i) = \sum_{t=0}^{N/4} a_t \lambda_{i,t} \ .$$

This satisfies

$$p(i) = \sum_{t=0}^{N/4} \mathrm{Tr}\left((UPU^T)_t\right) \lambda_{i,t} = \langle UPU^T, U\mu_i U^T \rangle = \langle P, \mu_i \rangle = P(i) \ .$$

Let q be the degree-d polynomial obtained by removing the high-degree parts of p.

$$q(i) = \sum_{t=0}^{d} a_t \lambda_{i,t}$$

Then P and q are close on all integers i between 0 and $\frac{N}{8}$.

$$|P(i) - q(i)| = |p(i) - q(i)| = \left| \sum_{t=d+1}^{N/4} a_t \lambda_{i,t} \right| \leq \frac{2^{-d/4}}{\mathcal{N}} \sum_{t=0}^{N/4} |a_t| \leq 2^{-d/4+2Q} \quad ,$$

which concludes the proof. □

6.4.2 Consequences for quantum protocols

Combining Razborov's technique with our polynomial bounds we obtain:

6.4.2. Theorem (DPT for Disjointness). *There are $\alpha, \gamma > 0$ such that every quantum protocol for $\text{Disj}_n^{(k)}$ using $Q \leq \alpha k \sqrt{n}$ qubits of communication has success probability $p \leq 2^{-\gamma k}$.*

Proof (sketch). By doing the same trick with $s = 2\log_2(\frac{1}{\alpha})$ rounds of binary search as for Theorem 6.3.8, we can tweak a protocol for $\text{Disj}_n^{(k)}$ to a protocol that satisfies, with $P(i)$ defined as in Lemma 6.4.1, $N = kn$, and $\sigma = p^{s+1}$:

$$\begin{aligned} P(i) &= 0 && \text{if } i \in \{0, \ldots, k-1\} \\ P(k) &\geq \sigma && \\ P(i) &\in [0,1] && \text{for all } i \in \{0, \ldots, N\} \ . \end{aligned}$$

A subtlety: instead of an exact version of distributed Grover search we use an exact version of the $O(\sqrt{n})$-qubit Disjointness protocol of [AA03]; the [BCW98]-protocol would lose a $\log n$-factor. Lemma 6.4.1, using $d = 12Q$, then gives a degree-d polynomial q that differs from P by at most $\delta \leq 2^{-Q}$ on all $i \in \{0, \ldots, \frac{N}{8}\}$. This δ is sufficiently small to apply Lemma 6.3.4, which in turn upper-bounds σ and hence p. □

This technique also gives strong direct product theorems for symmetric predicates other than Disj_n. As mentioned in the introduction, the same bound was later obtained independently by Beame et al. [BPSW05, Corollary 9] for classical protocols.

6.5 Summary

We have proved strong direct product theorems for the OR function in the following settings: randomized query complexity, quantum query complexity, and quantum communication complexity. We have also proved even stronger direct product theorems for 1-sided error threshold functions, and weaker direct product theorems for all total Boolean functions. Let us conclude with two major open problems.

First, we would like to know whether a strong direct product theorem holds in the query and communication setting for all Boolean functions if we consider worst-case error probability (Shaltiel [Sha01] disproved this for *average-case* error probability). Second, does a strong direct product theorem hold for the Disjointness function in the setting of classical randomized communication complexity?

Chapter 7

A New Adversary Method

This chapter is based on part of the following paper:[1]

[AŠW06] A. Ambainis, R. Špalek, and R. de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. In *Proceedings of 38th Annual ACM Symposium on Theory of Computing*, pages 618–633, 2006.

7.1 Introduction

Quantum query lower bounds In Chapter 2, we introduced two lower-bound methods for quantum query complexity: the adversary method [Amb02, Amb03] and the polynomial method [BBC⁺01]. We gave several examples showing that these two methods are in general incomparable. On the one hand, the adversary method proves stronger bounds than the polynomial method for certain iterated functions [Amb03], and also gives tight lower bounds for constant-depth AND-OR trees [Amb02, HMW03], where we do not know how to analyze the polynomial degree.

On the other hand, the polynomial method works well for analyzing zero-error or low-error quantum algorithms [BBC⁺01, BCWZ99] and gives optimal lower bounds for the collision problem and element distinctness [AS04]. As we showed in Section 5.4, the adversary method fails for the latter problem (and also for other problems like triangle-finding), because the best bound provable with it is $\sqrt{\mathrm{C}_0\mathrm{C}_1}$. Here C_0 and C_1 are the certificate complexities of the function on zero-inputs and one-inputs. In the case of element distinctness and triangle-finding, one of these complexities is constant. Hence the adversary method in its present

[1]Most of the contents of this chapter are due to Ambainis. Our main contribution is extending his direct product theorem that only held for *implicit thresholds* $t \leq \sqrt{n}$ to the full range $t \leq \frac{n}{2}$. In particular, Claim 7.6.1 gives an exact expression for the norm of a projected quantum state that holds for all thresholds, whereas the original version of the paper by Ambainis used different arguments to prove a bound that only held for small thresholds.

form(s) can prove at most an $\Omega(\sqrt{n})$ bound, where n is the input size, while the true bound is $\Theta(n^{2/3})$ [AS04, Amb04] in the case of element distinctness and the best known algorithm for triangle-finding costs $O(n^{13/20})$ [MSS05]. A second limitation of the adversary method is that it cannot deal well with the case where there are many different possible outputs, and a success probability much smaller than $\frac{1}{2}$ would still be considered good.

A new adversary method In this chapter we describe a new version of the adversary method that does not suffer from the second limitation, and possibly also not from the first—though we have not found an example yet where the new method breaks through the $\sqrt{\mathrm{C}_0\mathrm{C}_1}$ barrier.

Very roughly speaking, the new method works as follows. We view the algorithm as acting on a 2-register state space $\mathcal{H}_A \otimes \mathcal{H}_I$. Here the actual algorithm's operations take place in the first register, while the second contains (a superposition of) the inputs. In particular, the query operation on $\mathcal{H}_A$ is now conditioned on the basis states in $\mathcal{H}_I$. We start the analysis with a superposition of zero-inputs and one-inputs in the input register, and then track how this register evolves as the computation moves along. Let ρ_t be the state of this register (tracing out the $\mathcal{H}_A$-register) after making the t^{th} query. By employing symmetries in the problem's structure, such as invariances of the function under certain permutations of its input, we can decompose the input space into orthogonal subspaces $S_0, \ldots, S_m$. We can decompose the state accordingly.

$$\rho_t = \sum_{i=0}^{m} p_{t,i}\sigma_i \quad ,$$

where σ_i is a density matrix in subspace S_i. Thus the t^{th} state can be fully described by a probability distribution $p_{t,0}, \ldots, p_{t,m}$ that describes how the input register is distributed over the various subspaces. Crucially, only some of the subspaces are *good*, meaning that the algorithm will only work if most of the amplitude is concentrated in the good subspaces at the end of the computation. At the start of the computation, hardly any amplitude will be in the good subspaces. If we can show that in each query, not too much amplitude can move from the bad subspaces to the good subspaces, then we get a lower bound on the number of queries that have to be done.

This idea was first introduced by Ambainis in [Amb05a] and used there to reprove our strong direct product theorem for the OR function from Section 6.3. In this chapter we extend it and use it to prove direct product theorems for all symmetric functions.

Direct product theorems for symmetric functions Consider an algorithm that simultaneously needs to compute k independent instances of a function f (denoted $f^{(k)}$). Direct product theorems deal with the optimal tradeoff between the resources and success probability of such algorithms. We examined in Section 6.2 and Section 6.3 the case where the resource is query complexity and f is the OR function, and proved an optimal DPT both for classical algorithms and for quantum algorithms.

Here we generalize these results to the case where f can be any *symmetric function*, that is a function depending only on the Hamming weight $|x|$ of its input. In the case of classical algorithms the situation is quite simple. Every n-bit symmetric function f has classical bounded-error query complexity $R_2(f) = \Theta(n)$ and block sensitivity $bs(f) = \Theta(n)$, hence an optimal classical DPT follows immediately from Theorem 6.2.4. Classically, all symmetric functions essentially cost the same in terms of query complexity. This is different in the quantum world. For instance, the OR function has bounded-error quantum query complexity $Q_2(\mathrm{OR}) = \Theta(\sqrt{n})$, while parity needs $\frac{n}{2}$ quantum queries; see Example 2.7.1. If f is the t-threshold function ($\mathrm{Thr}_{t,n}(x) = 1$ if and only if $|x| \geq t$), then $Q_2(\mathrm{Thr}_{t,n}) = \Theta(\sqrt{t(n-t+1)})$; see Corollary 1.5.6 for an upper bound and Corollary 2.4.3 for a lower bound, also for the OR function with $t = 1$.

Our main result is an essentially optimal quantum DPT for all symmetric functions.

7.1.1. Theorem. (Quantum DPT for symmetric functions)
There is a constant $\alpha > 0$ such that for every symmetric f and every positive integer k: Every 2-sided error quantum algorithm for $f^{(k)}$ using $T \leq \alpha k Q_2(f)$ queries has success probability $\sigma \leq 2^{-\Omega(k)}$.

Our new direct product theorem generalizes the polynomial-based DPT for the OR function from Theorem 6.3.8, but our current proof uses the above-mentioned version of the adversary method. The new DPT should also be contrasted to the DPT for threshold functions from Theorem 6.3.9, which gives a much stronger bound on the success probability, but only holds for 1-sided error algorithms.

7.2 Quantum DPT for symmetric functions

The main result of this chapter is the proof of Theorem 7.1.1 above. In this section we give an outline of the proof. Most of the proofs of technical claims are deferred to the following sections.

Implicit threshold Let us first say something about $Q_2(f)$ for a symmetric function $f : \{0,1\}^n \to \{0,1\}$. Let t denote the smallest non-negative integer such that f is constant on the interval $|x| \in [t, n-t]$. We call this value t the *implicit threshold* of f. For instance, functions like OR and AND have $t = 1$, while parity and majority have $t = \frac{n}{2}$. If f is the t-threshold function with $t \leq \frac{n}{2}$, then the implicit threshold is just the threshold. The implicit threshold is related to the parameter $\Gamma(f)$ introduced by Paturi [Pat92] via $t = \frac{n}{2} - \frac{\Gamma(f)}{2} \pm 1$. It characterizes the bounded-error quantum query complexity of f: $Q_2(f) = \Theta(\sqrt{tn})$ [BBC+01]. Hence our resource bound in the above theorem will be $\alpha k \sqrt{tn}$ for some small constant $\alpha > 0$.

We actually prove a stronger statement, applying to any Boolean function f (total or partial) for which $f(x) = 0$ if $|x| = t-1$ and $f(x) = 1$ if $|x| = t$.

Input register Let A be an algorithm that computes k instances of this weight-$(t-1)$ versus weight-t problem. We recast A into a different form, using a register that stores the input $x^1, \dots, x^k$. Let $\mathcal{H}_A$ be the Hilbert space on which A operates. Let $\mathcal{H}_I$ be an $(\binom{n}{t-1} + \binom{n}{t})^k$-dimensional Hilbert space whose basis states correspond to inputs $(x^1, \dots, x^k)$ with Hamming weights $|x^1| \in \{t-1, t\}, \dots, |x^k| \in \{t-1, t\}$. We transform A into a sequence of transformations on a Hilbert space $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_I$. A non-query transformation U on $\mathcal{H}_A$ is replaced with $\mathsf{U} \otimes \mathsf{I}$ on $\mathcal{H}$. A query is replaced by a transformation O that is equal to $\mathsf{O}_{x^1,\dots,x^k} \otimes \mathsf{I}$ on the subspace consisting of states of the form $|s\rangle_A \otimes |x^1 \dots x^k\rangle_I$. This is an extension of the standard model of quantum query complexity from Section 1.2.2 that allows superposition inputs as opposed to allowing only computational basis inputs.

The starting state of the algorithm on Hilbert space $\mathcal{H}$ is $|\varphi_0\rangle = |\psi_{start}\rangle_A \otimes |\psi_0\rangle_I$ where $|\psi_{start}\rangle$ is the starting state of A as an algorithm acting on $\mathcal{H}_A$ and $|\psi_0\rangle = |\psi_{one}\rangle^{\otimes k}$ is a tensor product of k copies of the state $|\psi_{one}\rangle$ in which half of the amplitude is on $|x\rangle$ with $|x| = t$, the other half is on $|x\rangle$ with $|x| = t-1$, and any two states $|x\rangle$ with the same $|x|$ have equal amplitudes:

$$|\psi_{one}\rangle = \frac{1}{\sqrt{2\binom{n}{t}}} \sum_{x:|x|=t} |x\rangle + \frac{1}{\sqrt{2\binom{n}{t-1}}} \sum_{x:|x|=t-1} |x\rangle \; .$$

Let $|\varphi_d\rangle$ be the state of the algorithm A, as a sequence of transformations on $\mathcal{H}$, after the dth query. Let ρ_d be the mixed state in $\mathcal{H}_I$ obtained from $|\varphi_d\rangle$ by tracing out the $\mathcal{H}_A$ register.

Subspaces of the input register We define two decompositions of $\mathcal{H}_I$ into a direct sum of subspaces. We have $\mathcal{H}_I = (\mathcal{H}_{one})^{\otimes k}$ where $\mathcal{H}_{one}$ is the input Hilbert space for one instance, with basis states $|x\rangle$, $x \in \{0,1\}^n$, $|x| \in \{t-1, t\}$. Let

$$|\psi^0_{i_1,\dots,i_j}\rangle = \frac{1}{\sqrt{\binom{n-j}{t-1-j}}} \sum_{\substack{x_1,\dots,x_n: \\ x_1+\dots+x_n=t-1, \\ x_{i_1}=\dots=x_{i_j}=1}} |x_1 \dots x_n\rangle$$

and let $|\psi^1_{i_1,\dots,i_j}\rangle$ be a similar state with $x_1 + \dots + x_n = t$ instead of $x_1 + \dots + x_n = t-1$. Let $T_{j,0}$ (resp. $T_{j,1}$) be the space spanned by all states $|\psi^0_{i_1,\dots,i_j}\rangle$ (resp. $|\psi^1_{i_1,\dots,i_j}\rangle$) and let $S_{j,a} = T_{j,a} \cap T^{\perp}_{j-1,a}$. For a subspace S, we use Π_S to denote the projector onto S. Let $|\tilde{\psi}^a_{i_1,\dots,i_j}\rangle = \Pi_{T^\perp_{j-1,a}}|\psi^a_{i_1,\dots,i_j}\rangle$. For $j < t$, let $S_{j,+}$ be the subspace spanned by the states

$$\frac{|\tilde{\psi}^0_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^0_{i_1,\dots,i_j}\|} + \frac{|\tilde{\psi}^1_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^1_{i_1,\dots,i_j}\|}$$

and $S_{j,-}$ be the subspace spanned by

$$\frac{|\tilde{\psi}^0_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^0_{i_1,\dots,i_j}\|} - \frac{|\tilde{\psi}^1_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^1_{i_1,\dots,i_j}\|} \ .$$

For $j = t$, we define $S_{t,-} = S_{t,1}$ and there is no subspace $S_{t,+}$. Thus $\mathcal{H}_{one} = \bigoplus_{j=0}^{t-1}(S_{j,+} \oplus S_{j,-}) \oplus S_{t,-}$. Let us try to give some intuition. In the spaces $S_{j,+}$ and $S_{j,-}$, we may be said to "know" the positions of j of the ones. In the *good* subspaces $S_{j,-}$ we have distinguished the zero-inputs from one-inputs by the relative phase, while in the *bad* subspaces $S_{j,+}$ we have not distinguished them. Accordingly, the algorithm is doing well on this one instance if most of the state sits in the good subspaces $S_{j,-}$.

For the space $\mathcal{H}_I$ (representing k independent inputs for our function) and $r_1, \dots, r_k \in \{+, -\}$, we define

$$S_{j_1,\dots,j_k,r_1,\dots,r_k} = S_{j_1,r_1} \otimes S_{j_2,r_2} \otimes \dots \otimes S_{j_k,r_k} \ .$$

Let $\mathcal{S}_{m-}$ be the direct sum of all $S_{j_1,\dots,j_k,r_1,\dots,r_k}$ such that exactly m of the signs $r_1, \dots, r_k$ are equal to $-$. Then $\mathcal{H}_I = \bigoplus_m \mathcal{S}_{m-}$. This is the first decomposition.

The above intuition for one instance carries over to k instances: the more minuses the better for the algorithm. Conversely, if most of the input register sits in S_{m-} for low m, then its success probability will be small. More precisely, in Section 7.3 we prove the following lemma.

7.2.1. LEMMA (MEASUREMENT IN BAD SUBSPACES). *Let ρ be the reduced density matrix of $\mathcal{H}_I$. If the support of ρ is contained in $\mathcal{S}_{0-} \oplus \mathcal{S}_{1-} \oplus \dots \oplus \mathcal{S}_{m-}$, then the probability that measuring $\mathcal{H}_A$ gives the correct answer is at most* $\frac{\sum_{m'=0}^{m}\binom{k}{m'}}{2^k}$.

Note that this probability is exponentially small in k for, say, $m = \frac{k}{3}$. The following consequence of this lemma is proven in Section 7.4.

7.2.2. COROLLARY (TOTAL SUCCESS PROBABILITY). *Let ρ be the reduced density matrix of $\mathcal{H}_I$. The probability that measuring $\mathcal{H}_A$ gives the correct answer is at most*

$$\frac{\sum_{m'=0}^{m}\binom{k}{m'}}{2^k} + 4\sqrt{\operatorname{Tr}\Pi_{(\mathcal{S}_{0-}\oplus\mathcal{S}_{1-}\oplus\dots\oplus\mathcal{S}_{m-})^\perp}\rho} \ .$$

$\lvert\psi^a_{i_1,\ldots,i_j}\rangle$	uniform superposition of states with $\lvert x\rvert = t-1+a$ and with j fixed bits set to 1
$T_{j,a}$	spanned by $\lvert\psi^a_{i_1,\ldots,i_j}\rangle$ for all j-tuples i
$S_{j,a} = T_{j,a} \cap T^{\perp}_{j-1,a}$	that is, we remove the lower-dimensional subspace
$\lvert\tilde{\psi}^a_{i_1,\ldots,i_j}\rangle$	projection of $\lvert\psi^a_{i_1,\ldots,i_j}\rangle$ onto $S_{j,a}$
$S_{j,\pm}$	spanned by $\frac{\lvert\tilde{\psi}^0\rangle}{\lVert\tilde{\psi}^0\rVert} \pm \frac{\lvert\tilde{\psi}^1\rangle}{\lVert\tilde{\psi}^1\rVert}$
$R_j = S_{j,+}$	for $j < \frac{t}{2}$ ...bad
$R_{t/2}$	direct sum of $S_{j,+}$ for $j \geq t/2$, and all $S_{j,-}$ ...good
$\mathcal{S}_{m-} = \bigoplus_{\substack{\lvert r\rvert=m \\ j}} \bigotimes_{i=1}^{k} S_{j_i,r_i}$	where $\lvert r\rvert$ is the number of minuses in $r = r_1,\ldots,r_k$
$\mathcal{R}_m = \bigoplus_{\lvert j\rvert_1=m} \bigotimes_{i=1}^{k} R_{j_i}$	where $\lvert j\rvert_1$ is the sum of all entries in $j = j_1,\ldots,j_k$
$\mathcal{R}'_j = \bigoplus_{m\geq j} \mathcal{R}_m$	
$\lvert\psi^{a,b}_{i_1,\ldots,i_j}\rangle$	uniform superposition of states with $\lvert x\rvert = t-1+a$, with j fixed bits set to 1, and $x_1 = b$
$T_{j,a,b}$	spanned by $\lvert\psi^{a,b}_{i_1,\ldots,i_j}\rangle$ for all j-tuples i
$S_{j,a,b} = T_{j,a,b} \cap T^{\perp}_{j-1,a,b}$	that is, we remove the lower-dimensional subspace
$\lvert\tilde{\psi}^{a,b}_{i_1,\ldots,i_j}\rangle$	projection of $\lvert\psi^{a,b}_{i_1,\ldots,i_j}\rangle$ into $S_{j,a,b}$
$S^{\alpha,\beta}_{j,a}$	spanned by $\alpha\frac{\lvert\tilde{\psi}^{a,0}\rangle}{\lVert\tilde{\psi}^{a,0}\rVert} + \beta\frac{\lvert\tilde{\psi}^{a,1}\rangle}{\lVert\tilde{\psi}^{a,1}\rVert}$

Figure 7.1: States and subspaces used in the adversary-type DPT

Second decomposition To define the second decomposition into subspaces, we express $\mathcal{H}_{one} = \bigoplus_{j=0}^{t/2} R_j$ with $R_j = S_{j,+}$ for $j < t/2$ and

$$R_{t/2} = \bigoplus_{j\geq t/2} S_{j,+} \oplus \bigoplus_{j\geq 0} S_{j,-} \ .$$

Intuitively, all subspaces except for $R_{t/2}$ are bad for the algorithm, since they equal the *bad* subspaces $S_{j,+}$. Let $\mathcal{R}_\ell$ be the direct sum of all $R_{j_1} \otimes \cdots \otimes R_{j_k}$ satisfying $j_1 + \cdots + j_k = \ell$. Then $\mathcal{H}_I = \bigoplus_{\ell=0}^{tk/2} \mathcal{R}_\ell$. This is the second decomposition.

Intuitively, the algorithm can only have good success probability if for most of the k instances, most of the input register sits in $R_{t/2}$. Aggregated over all k instances, this means that the algorithm will only work well if most of the k-input register sits in $\mathcal{R}_\ell$ for ℓ large, meaning fairly close to $kt/2$. Our goal below is to show that this cannot happen if the number of queries is small.

Let $\mathcal{R}'_j = \bigoplus_{\ell=j}^{tk/2} \mathcal{R}_\ell$. Note that $\mathcal{S}_{m-} \subseteq \mathcal{R}'_{tm/2}$ for every m: $\mathcal{S}_{m-}$ is the direct sum of subspaces $S = S_{j_1,r_1} \otimes \cdots \otimes S_{j_k,r_k}$ having m minuses among $r_1,\ldots,r_k$; each such minus-subspace sits in the corresponding $R_{t/2}$ and hence $S \subseteq \mathcal{R}'_{tm/2}$.

This implies

$$(\mathcal{S}_{0-} \oplus \mathcal{S}_{1-} \oplus \cdots \oplus \mathcal{S}_{(m-1)-})^{\perp} \subseteq \mathcal{R}'_{tm/2} \ .$$

Accordingly, if we prove an upper bound on $\operatorname{Tr} \Pi_{\mathcal{R}'_{tm/2}} \rho_T$, where T is the total number of queries, this bound together with Corollary 7.2.2 implies an upper bound on the success probability of A.

Potential function To bound $\operatorname{Tr} \Pi_{\mathcal{R}'_{tm/2}} \rho_T$, we consider the following *potential function*

$$P(\rho) = \sum_{m=0}^{tk/2} q^m \operatorname{Tr} \Pi_{\mathcal{R}_m} \rho \ ,$$

where $q = 1 + \frac{1}{t}$. Then for every d

$$\operatorname{Tr} \Pi_{\mathcal{R}'_{tm/2}} \rho_d \leq P(\rho_d) q^{-tm/2} = P(\rho_d) e^{-(1+o(1))m/2} \ . \tag{7.1}$$

$P(\rho_0) = 1$, because the initial state $|\psi_0\rangle$ is a tensor product of the states $|\psi_{one}\rangle$ on each copy of $\mathcal{H}_{one}$ and $|\psi_{one}\rangle$ belongs to $S_{0,+}$, hence $|\psi_0\rangle$ belongs to $\mathcal{R}_0$. In Section 7.7 we prove

7.2.3. Lemma (Change of the potential). *There is a constant C such that*

$$P(\rho_{j+1}) \leq \left(1 + \frac{C}{\sqrt{tn}}(q^{t/2} - 1) + \frac{C\sqrt{t}}{\sqrt{n}}(q - 1)\right) P(\rho_j) \ .$$

Since $q = 1 + \frac{1}{t}$, Lemma 7.2.3 means that $P(\rho_{j+1}) \leq (1 + \frac{C\sqrt{e}}{\sqrt{tn}})P(\rho_j)$ and $P(\rho_j) \leq (1 + \frac{C\sqrt{e}}{\sqrt{tn}})^j \leq e^{\frac{2Cj}{\sqrt{tn}}}$. By equation (7.1), for the final state after T queries we have

$$\operatorname{Tr} \Pi_{\mathcal{R}'_{tm/2}} \rho_T \leq e^{\frac{2CT}{\sqrt{tn}} - (1+o(1))\frac{m}{2}} \ .$$

We take $m = \frac{k}{3}$. Then if $T \leq \frac{1}{8C} m \sqrt{tn}$, this expression is exponentially small in k. Together with Corollary 7.2.2, this implies Theorem 7.1.1.

7.3 Measurement in bad subspaces

In this section we prove Lemma 7.2.1.

The measurement of $\mathcal{H}_A$ decomposes the state in the $\mathcal{H}_I$ register as follows:

$$\rho = \sum_{a_1,\ldots,a_k \in \{0,1\}} p_{a_1,\ldots,a_k} \sigma_{a_1,\ldots,a_k} \ ,$$

with $p_{a_1,\ldots,a_k}$ being the probability of the measurement giving $(a_1, \ldots, a_k)$ (where $a_j = 1$ means the algorithm outputs—not necessarily correctly—that $|x^j| = t$ and

$a_j = 0$ means $|x^j| = t-1$) and $\sigma_{a_1,\ldots,a_k}$ being the density matrix of $\mathcal{H}_I$, conditional on this outcome of the measurement. Since the support of ρ is contained in $\mathcal{S}_{0-} \oplus \cdots \oplus \mathcal{S}_{m-}$, the support of the states $\sigma_{a_1,\ldots,a_k}$ is also contained in $\mathcal{S}_{0-} \oplus \cdots \oplus \mathcal{S}_{m-}$. The probability that the answer $(a_1, \ldots, a_k)$ is correct is equal to

$$\operatorname{Tr} \Pi_{\otimes_{j=1}^k \oplus_{l=0}^{t-1+a_j} S_{l,a_j}} \sigma_{a_1,\ldots,a_k} \quad . \tag{7.2}$$

We show that, for any $\sigma_{a_1,\ldots,a_k}$ with support contained in $\mathcal{S}_{0-} \oplus \cdots \oplus \mathcal{S}_{m-}$, (7.2) is at most $\frac{\sum_{m'=0}^m \binom{k}{m'}}{2^k}$.

For brevity, we now write σ instead of $\sigma_{a_1,\ldots,a_k}$. A measurement with respect to $\otimes_{j=1}^k \oplus_l S_{l,a_j}$ and its orthogonal complement commutes with a measurement with respect to the collection of subspaces

$$\otimes_{j=1}^k (S_{l_j,0} \oplus S_{l_j,1}) \quad ,$$

where $l_1, \ldots, l_k$ range over $\{0, \ldots, t\}$. Therefore

$$\operatorname{Tr} \Pi_{\otimes_{j=1}^k \oplus_l S_{l,a_j}} \sigma = \sum_{l_1,\ldots,l_k} \operatorname{Tr} \Pi_{\otimes_{j=1}^k \oplus_l S_{l,a_j}} \Pi_{\otimes_{j=1}^k (S_{l_j,0} \oplus S_{l_j,1})} \sigma \quad .$$

Hence to bound (7.2) it suffices to prove the same bound with

$$\sigma' = \Pi_{\otimes_{j=1}^k (S_{l_j,0} \oplus S_{l_j,1})} \sigma$$

instead of σ. Since

$$\left(\otimes_{j=1}^k (S_{l_j,0} \oplus S_{l_j,1})\right) \cap \left(\otimes_{j=1}^k (\oplus_l S_{l,a_j})\right) = \otimes_{j=1}^k S_{l_j,a_j} \quad ,$$

we have

$$\operatorname{Tr} \Pi_{\otimes_{j=1}^k (\oplus_l S_{l,a_j})} \sigma' = \operatorname{Tr} \Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \sigma' \quad . \tag{7.3}$$

We prove this bound for the case when σ' is a pure state: $\sigma' = |\psi\rangle\langle\psi|$. Then equation (7.3) is equal to

$$\|\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \psi\|^2 \quad . \tag{7.4}$$

The bound for mixed states σ' follows by decomposing σ' as a mixture of pure states $|\psi\rangle$, bounding (7.4) for each of those states and then summing up the bounds.

We have

$$(\mathcal{S}_{0-} \oplus \cdots \oplus \mathcal{S}_{m-}) \cap (\bigotimes_{j=1}^k (S_{l_j,0} \oplus S_{l_j,1})) = \bigoplus_{\substack{r_1,\ldots,r_k \in \{+,-\},\\ |\{i: r_i = -\}| \le m}} \bigotimes_{j=1}^k S_{l_j,r_j} \quad .$$

We express

$$|\psi\rangle = \sum_{\substack{r_1,\ldots,r_k \in \{+,-\},\\ |\{i: r_i = -\}| \le m}} \alpha_{r_1,\ldots,r_k} |\psi_{r_1,\ldots,r_k}\rangle \quad ,$$

with $|\psi_{r_1,\ldots,r_k}\rangle \in \otimes_{j=1}^k S_{l_j,r_j}$. Therefore

$$\begin{aligned} \|\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \psi\|^2 &\leq \left(\sum_{r_1,\ldots,r_k} |\alpha_{r_1,\ldots,r_k}| \cdot \|\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \psi_{r_1,\ldots,r_k}\| \right)^2 \\ &\leq \sum_{r_1,\ldots,r_k} \|\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \psi_{r_1,\ldots,r_k}\|^2 \ , \end{aligned} \tag{7.5}$$

where the second inequality uses Cauchy-Schwarz and

$$\|\psi\|^2 = \sum_{r_1,\ldots,r_k} |\alpha_{r_1,\ldots,r_k}|^2 = 1 \ .$$

7.3.1. CLAIM. $\|\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \psi_{r_1,\ldots,r_k}\|^2 \leq \frac{1}{2^k}$.

PROOF. Let $|\varphi_i^{j,0}\rangle$, $i \in [\dim S_{l_j,0}]$ form a basis for the subspace $S_{l_j,0}$. Define a map $\mathsf{U}_j : S_{l_j,0} \to S_{l_j,1}$ by $\mathsf{U}_j|\tilde{\psi}^0_{i_1,\ldots,i_{l_j}}\rangle = |\tilde{\psi}^1_{i_1,\ldots,i_{l_j}}\rangle$. Then U_j is a multiple of a unitary transformation: $\mathsf{U}_j = c_j\mathsf{U}'_j$ for some unitary U'_j and a constant c_j. (This follows from Claim 7.5.2 below.)

Let $|\varphi_i^{j,1}\rangle = \mathsf{U}'_j|\varphi_i^{j,0}\rangle$. Since U'_j is a unitary transformation, the states $|\varphi_i^{j,1}\rangle$ form a basis for $S_{l_j,1}$. Therefore

$$\bigotimes_{j=1}^k |\varphi_{i_j}^{j,a_j}\rangle \tag{7.6}$$

is a basis for $\otimes_{j=1}^k S_{l_j,a_j}$. Moreover, the states

$$|\varphi_i^{j,+}\rangle = \frac{1}{\sqrt{2}}|\varphi_i^{j,0}\rangle + \frac{1}{\sqrt{2}}|\varphi_i^{j,1}\rangle, \quad |\varphi_i^{j,-}\rangle = \frac{1}{\sqrt{2}}|\varphi_i^{j,0}\rangle - \frac{1}{\sqrt{2}}|\varphi_i^{j,1}\rangle$$

are a basis for $S_{l_j,+}$ and $S_{l_j,-}$, respectively. Therefore

$$|\psi_{r_1,\ldots,r_k}\rangle = \sum_{i_1,\ldots,i_k} \alpha_{i_1,\ldots,i_k} \bigotimes_{j=1}^k |\varphi_{i_j}^{j,r_j}\rangle \ . \tag{7.7}$$

The inner product between $\otimes_{i=1}^k |\varphi_{i'_j}^{j,a_j}\rangle$ and $\otimes_{j=1}^k |\varphi_{i_j}^{j,r_j}\rangle$ is

$$\prod_{j=1}^k \langle \varphi_{i_j}^{j,r_j} | \varphi_{i'_j}^{j,a_j}\rangle \ .$$

Note that $r_j \in \{+,-\}$ and $a_j \in \{0,1\}$. The terms in this product are $\pm\frac{1}{\sqrt{2}}$ if $i'_j = i_j$ and 0 otherwise. This means that $\otimes_{j=1}^k |\varphi_{i_j}^{j,r_j}\rangle$ has inner product $\pm\frac{1}{2^{k/2}}$ with $\otimes_{i=1}^k |\varphi_{i_j}^{j,a_j}\rangle$ and inner product 0 with all other basis states (7.6). Therefore,

$$\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \otimes_{j=1}^k |\varphi_{i_j}^{j,r_j}\rangle = \pm\frac{1}{2^{k/2}} \otimes_{i=1}^k |\varphi_{i_j}^{j,a_j}\rangle \ .$$

Together with equation (7.7), this means that

$$\|\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \psi_{r_1,\dots,r_k}\| \le \frac{1}{2^{k/2}} \|\psi_{r_1,\dots,r_k}\| = \frac{1}{2^{k/2}} \ .$$

Squaring both sides completes the proof of the claim. □

Since there are $\binom{k}{m'}$ tuples $(r_1, \dots, r_k)$ with $r_1, \dots, r_k \in \{+, -\}$ and $|\{i : r_i = -\}| = m'$, Claim 7.3.1 together with equation (7.5) implies

$$\|\Pi_{\otimes_{j=1}^k S_{l_j,a_j}} \psi\|^2 \le \frac{\sum_{m'=0}^{m} \binom{k}{m'}}{2^k} \ .$$

7.4 Total success probability

In this section we prove Corollary 7.2.2.

Let $|\psi\rangle$ be a purification of ρ in $\mathcal{H}_A \otimes \mathcal{H}_I$. Let

$$|\psi\rangle = \sqrt{1-\delta}|\psi'\rangle + \sqrt{\delta}|\psi''\rangle \ ,$$

where $|\psi'\rangle$ is in the subspace $\mathcal{H}_A \otimes (\mathcal{S}_{0-} \oplus \mathcal{S}_{1-} \oplus \cdots \oplus \mathcal{S}_{m-})$ and $|\psi''\rangle$ is in the subspace $\mathcal{H}_A \otimes (\mathcal{S}_{0-} \oplus \mathcal{S}_{1-} \oplus \cdots \oplus \mathcal{S}_{m-})^{\perp}$. Then $\delta = \operatorname{Tr} \Pi_{(\mathcal{S}_{0-} \oplus \cdots \oplus \mathcal{S}_{m-})^{\perp}} \rho$.

The success probability of A is the probability that, if we measure both the register $\mathcal{H}_A$ containing the result of the computation and $\mathcal{H}_I$, then we get $a_1, \dots, a_k$ and $x^1, \dots, x^k$ such that x^j contains $t - 1 + a_j$ ones for every $j \in \{1, \dots, k\}$.

Consider the probability of getting $a_1, \dots, a_k \in \{0, 1\}$ and $x^1, \dots, x^k \in \{0, 1\}^n$ with this property, when measuring $|\psi'\rangle$ (instead of $|\psi\rangle$). By Lemma 7.2.1, this probability is at most $\frac{\sum_{m'=0}^{m} \binom{k}{m'}}{2^k}$. We have

$$\|\psi - \psi'\| \le (1 - \sqrt{1-\delta})\|\psi'\| + \sqrt{\delta}\|\psi''\| = (1 - \sqrt{1-\delta}) + \sqrt{\delta} \le 2\sqrt{\delta} \ .$$

We now apply the following lemma by Bernstein and Vazirani.

7.4.1. Lemma ([BV97]). *For any states $|\psi\rangle$ and $|\psi'\rangle$ and any measurement M, the variational distance between the probability distributions obtained by applying M to $|\psi\rangle$ and $|\psi'\rangle$ is at most* $2\|\psi - \psi'\|$.

Hence the success probability of A is at most

$$\frac{\sum_{m'=0}^{m} \binom{k}{m'}}{2^k} + 4\sqrt{\delta} = \frac{\sum_{m'=0}^{m} \binom{k}{m'}}{2^k} + 4\sqrt{\operatorname{Tr} \Pi_{(\mathcal{S}_{0-} \oplus \cdots \oplus \mathcal{S}_{m-})^{\perp}} \rho} \ .$$

7.5 Subspaces when asking one query

Let $|\psi_d\rangle$ be the state of $\mathcal{H}_A \otimes \mathcal{H}_I$ after d queries. Write

$$|\psi_d\rangle = \sum_{i=0}^{kn} a_i |\psi_{d,i}\rangle \ ,$$

with $|\psi_{d,i}\rangle$ being the part in which the query register contains $|i\rangle$. Let $\rho_{d,i} = \mathrm{Tr}_{\mathcal{H}_A} |\psi_{d,i}\rangle\langle\psi_{d,i}|$. Then

$$\rho_d = \sum_{i=0}^{kn} a_i^2 \rho_{d,i} \ . \tag{7.8}$$

Because of

$$\mathrm{Tr}\, \Pi_{\mathcal{R}_m} \rho_d = \sum_{i=0}^{kn} a_i^2 \,\mathrm{Tr}\, \Pi_{\mathcal{R}_m} \rho_{d,i} \ ,$$

we have $P(\rho_d) = \sum_{i=0}^{kn} a_i^2 P(\rho_{d,i})$. Let ρ'_d be the state after the d^{th} query and let $\rho'_d = \sum_{i=0}^{kn} a_i^2 \rho'_{d,i}$ be a decomposition similar to equation (7.8). Lemma 7.2.3 follows by showing

$$P(\rho'_{d,i}) \leq \left(1 + \frac{C}{\sqrt{tn}}(q^{t/2} - 1) + \frac{C\sqrt{t}}{\sqrt{n}}(q-1)\right) P(\rho_{d,i}) \tag{7.9}$$

for each i. For $i = 0$, the query does not change the state if the query register contains $|i\rangle$. Therefore, $\rho'_{d,0} = \rho_{d,0}$ and $P(\rho'_{d,0}) = P(\rho_{d,0})$. This means that equation (7.9) is true for $i = 0$. To prove the $i \in \{1, \ldots, kn\}$ case, it suffices to prove the $i = 1$ case (because of symmetry).

Subspaces of $\rho_{d,1}$ Let $|\psi^{a,b}_{i_1,\ldots,i_j}\rangle$ (with $a, b \in \{0, 1\}$ and $i_1, \ldots, i_j \in \{2, \ldots, n\}$) be the uniform superposition over basis states $|b, x_2, \ldots, x_n\rangle$ (of $\mathcal{H}_{one}$) with $b + x_2 + \cdots + x_n = t - 1 + a$ and $x_{i_1} = \cdots = x_{i_j} = 1$. Let $T_{j,a,b}$ be the space spanned by all states $|\psi^{a,b}_{i_1,\ldots,i_j}\rangle$ and let $S_{j,a,b} = T_{j,a,b} \cap T^{\perp}_{j-1,a,b}$. Let $|\tilde{\psi}^{a,b}_{i_1,\ldots,i_j}\rangle = \Pi_{T^{\perp}_{j-1,a,b}} |\psi^{a,b}_{i_1,\ldots,i_j}\rangle$.

Let $S^{\alpha,\beta}_{j,a}$ be the subspace spanned by all states

$$\alpha \frac{|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle}{\||\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|} + \beta \frac{|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle}{\||\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|} \ . \tag{7.10}$$

7.5.1. Claim. Let $\alpha_a = \sqrt{\frac{n-(t-1+a)}{n-j}} \||\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|$ and $\beta_a = \sqrt{\frac{(t-1+a)-j}{n-j}} \||\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|$. Then (i) $S^{\alpha_a,\beta_a}_{j,a} \subseteq S_{j,a}$ and (ii) $S^{\beta_a,-\alpha_a}_{j,a} \subseteq S_{j+1,a}$.

Proof. For part (i), consider the states $|\psi^{a}_{i_1,\ldots,i_j}\rangle$ in $T_{j,a}$, for $1 \notin \{i_1, \ldots, i_j\}$. We have

$$|\psi^{a}_{i_1,\ldots,i_j}\rangle = \sqrt{\frac{n-(t-1+a)}{n-j}} |\psi^{a,0}_{i_1,\ldots,i_j}\rangle + \sqrt{\frac{(t-1+a)-j}{n-j}} |\psi^{a,1}_{i_1,\ldots,i_j}\rangle \tag{7.11}$$

because among the states $|x_1 \dots x_n\rangle$ with $|x| = t-1+a$ and $x_{i_1} = \dots = x_{i_j} = 1$, a $\frac{n-(t-1+a)}{n-j}$ fraction have $x_1 = 0$ and the rest have $x_1 = 1$. The projections of these states to $T^{\perp}_{j-1,a,0} \cap T^{\perp}_{j-1,a,1}$ are

$$\sqrt{\tfrac{n-(t-1+a)}{n-j}}|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\rangle + \sqrt{\tfrac{(t-1+a)-j}{n-j}}|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\rangle$$

which, by equation (7.10) are exactly the states spanning $S^{\alpha_a,\beta_a}_{j,a}$. Furthermore, we claim that

$$T_{j-1,a} \subseteq T_{j-1,a,0} \oplus T_{j-1,a,1} \subseteq T_{j,a} \quad . \tag{7.12}$$

The first containment is true because $T_{j-1,a}$ is spanned by the states $|\psi^a_{i_1,\dots,i_{j-1}}\rangle$ which either belong to $T_{j-2,a,1} \subseteq T_{j-1,a,1}$ (if $1 \in \{i_1, \dots, i_{j-1}\}$) or are a linear combination of states $|\psi^{a,0}_{i_1,\dots,i_{j-1}}\rangle$ and $|\psi^{a,1}_{i_1,\dots,i_{j-1}}\rangle$ (by equation (7.11)), which belong to $T_{j-1,a,0}$ and $T_{j-1,a,1}$. The second containment follows because the states $|\psi^{a,1}_{i_1,\dots,i_{j-1}}\rangle$ spanning $T_{j-1,a,1}$ are the same as the states $|\psi^a_{1,i_1,\dots,i_{j-1}}\rangle$ which belong to $T_{j,a}$, and the states $|\psi^{a,0}_{i_1,\dots,i_{j-1}}\rangle$ spanning $T_{j-1,a,0}$ can be expressed as linear combinations of $|\psi^a_{i_1,\dots,i_{j-1}}\rangle$ and $|\psi^a_{1,i_1,\dots,i_{j-1}}\rangle$ which both belong to $T_{j,a}$.

The first part of (7.12) now implies

$$S^{\alpha_a,\beta_a}_{j,a} \subseteq T^{\perp}_{j-1,a,0} \cap T^{\perp}_{j-1,a,1} \subseteq T^{\perp}_{j-1,a} \quad .$$

Also, $S^{\alpha_a,\beta_a}_{j,a} \subseteq T_{j,a}$, because $S^{\alpha_a,\beta_a}_{j,a}$ is spanned by the states

$$\Pi_{T^{\perp}_{j-1,a,0} \cap T^{\perp}_{j-1,a,1}}|\psi^a_{i_1,\dots,i_j}\rangle = |\psi^a_{i_1,\dots,i_j}\rangle - \Pi_{T_{j-1,a,0} \oplus T_{j-1,a,1}}|\psi^a_{i_1,\dots,i_j}\rangle$$

and $|\psi^a_{i_1,\dots,i_j}\rangle$ belongs to $T_{j,a}$ by the definition of $T_{j,a}$ and $\Pi_{T_{j-1,a,0} \oplus T_{j-1,a,1}}|\psi^a_{i_1,\dots,i_j}\rangle$ belongs to $T_{j,a}$ because of the second part of (7.12). Therefore, $S^{\alpha_a,\beta_a}_{j,a} \subseteq T_{j,a} \cap T^{\perp}_{j-1,a} = S_{j,a}$.

For part (ii), we have

$$S^{\alpha_a,\beta_a}_{j,a} \subseteq S_{j,a,0} \oplus S_{j,a,1} \subseteq T_{j,a,0} \oplus T_{j,a,1} \subseteq T_{j+1,a} \quad ,$$

where the first containment is true because $S^{\alpha_a,\beta_a}_{j,a}$ is spanned by linear combinations of vectors $|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\rangle$ (which belong to $S_{j,a,0}$) and vectors $|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\rangle$ (which belong to $S_{j,a,1}$) and the last containment is true because of the second part of equation (7.12). Now let

$$|\psi\rangle = \beta_a \frac{|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\|} - \alpha_a \frac{|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\|} \tag{7.13}$$

be one of the vectors spanning $S^{\beta_a,-\alpha_a}_{j,a}$. To prove that $|\psi\rangle$ is in $S_{j+1,a} = T_{j+1,a} \cap T^{\perp}_{j,a}$, it remains to prove that $|\psi\rangle$ is orthogonal to $T_{j,a}$. This is equivalent to proving that $|\psi\rangle$ is orthogonal to each of the vectors $|\psi^a_{i'_1,\dots,i'_j}\rangle$ spanning $T_{j,a}$. We distinguish two cases (note that $1 \notin \{i_1, \dots, i_j\}$):

1. $1 \in \{i'_1, \dots, i'_j\}$.

   For simplicity, assume $1 = i'_j$. Then $|\psi^a_{i'_1,\dots,i'_j}\rangle$ is the same as $|\psi^{a,1}_{i'_1,\dots,i'_{j-1}}\rangle$, which belongs to $T_{j-1,a,1}$. By definition, the vector $|\psi\rangle$ belongs to $T^\perp_{j-1,a,0} \cap T^\perp_{j-1,a,1}$ and is therefore orthogonal to $|\psi^{a,1}_{i'_1,\dots,i'_{j-1}}\rangle$.

2. $1 \notin \{i'_1, \dots, i'_j\}$.

   We will prove this case by induction on $\ell = |\{i'_1, \dots, i'_j\} - \{i_1, \dots, i_j\}|$.

   In the base step ($\ell = 0$), we have $\{i'_1, \dots, i'_j\} = \{i_1, \dots, i_j\}$. Since $|\psi\rangle$ belongs to $T^\perp_{j-1,a,0} \cap T^\perp_{j-1,a,1}$, it suffices to prove $|\psi\rangle$ is orthogonal to the projection of $|\psi^a_{i_1,\dots,i_j}\rangle$ to $T^\perp_{j-1,a,0} \cap T^\perp_{j-1,a,1}$ which, by the discussion after equation (7.11), equals

$$\alpha_a \frac{|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\|} + \beta_a \frac{|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\rangle}{\||\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\|} \ . \tag{7.14}$$

   From equations (7.13) and (7.14), we see that the inner product of the two states is $\alpha_a\beta_a - \beta_a\alpha_a = 0$.

   For the inductive step ($\ell \geq 1$), assume $i'_j \notin \{i_1, \dots, i_j\}$. Up to a constant multiplicative factor, we have

$$|\psi^a_{i'_1,\dots,i'_{j-1}}\rangle = \sum_{i' \notin \{i'_1,\dots,i'_{j-1}\}} |\psi^a_{i'_1,\dots,i'_{j-1},i'}\rangle \ .$$

   Because $|\psi^a_{i'_1,\dots,i'_{j-1}}\rangle$ is in $T_{j-1,a,0} \oplus T_{j-1,a,1}$, we have

$$\sum_{i' \notin \{i'_1,\dots,i'_{j-1}\}} \langle\psi^a_{i'_1,\dots,i'_{j-1},i'}|\psi\rangle = \langle\psi^a_{i'_1,\dots,i'_{j-1}}|\psi\rangle = 0 \ . \tag{7.15}$$

   As proven in the previous case, $\langle\psi^a_{i'_1,\dots,i'_{j-1},1}|\psi\rangle = 0$. Moreover, by the induction hypothesis we have $\langle\psi^a_{i'_1,\dots,i'_{j-1},i'}|\psi\rangle = 0$ whenever $i' \in \{i_1, \dots, i_j\}$. Therefore equation (7.15) reduces to

$$\sum_{i' \notin \{i'_1,\dots,i'_{j-1},i_1,\dots,i_j,1\}} \langle\psi^a_{i'_1,\dots,i'_{j-1},i'}|\psi\rangle = 0 \ . \tag{7.16}$$

   By symmetry, the inner products in this sum are the same for every i'. Hence they are all 0, in particular for $i' = i'_j$.

We conclude that $|\psi\rangle$ is orthogonal to the subspace $T_{j,a}$ and therefore $|\psi\rangle$ is in $S_{j+1,a} = T_{j+1,a} \cap T^\perp_{j,a}$. □

7.5.2. CLAIM. The maps $\mathsf{U}_{01} : S_{j,0,0} \to S_{j,0,1}$, $\mathsf{U}_{10} : S_{j,0,0} \to S_{j,1,0}$ and $\mathsf{U}_{11} : S_{j,0,0} \to S_{j,1,1}$ defined by $\mathsf{U}_{ab}|\tilde{\psi}^{0,0}_{i_1,\dots,i_j}\rangle = |\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\rangle$ are multiples of unitary transformations: $\mathsf{U}_{ab} = c_{ab}\mathsf{U}'_{ab}$ for some unitary U'_{ab} and some constant c_{ab}.

PROOF. We define $\mathsf{M} : T_{j,0,0} \to T_{j,0,1}$ by

$$\mathsf{M}|0x_2 \dots x_n\rangle = \sum_{\ell : x_\ell = 1} |1x_2 \dots x_{\ell-1} 0 x_{\ell+1} \dots x_n\rangle \ .$$

Note that M does not depend on j. We claim

$$\begin{aligned} \mathsf{M}|\tilde{\psi}^{0,0}_{i_1,\dots,i_j}\rangle &= c|\tilde{\psi}^{0,1}_{i_1,\dots,i_j}\rangle \\ \mathsf{M}^\dagger|\tilde{\psi}^{0,1}_{i_1,\dots,i_j}\rangle &= c'|\tilde{\psi}^{0,0}_{i_1,\dots,i_j}\rangle \end{aligned} \tag{7.17}$$

for some constants c and c' that may depend on n, t and j but not on $i_1, \dots, i_j$.

To prove that, we need to prove two things. First, we claim that

$$\mathsf{M}|\psi^{0,0}_{i_1,\dots,i_j}\rangle = c|\psi^{0,1}_{i_1,\dots,i_j}\rangle + |\psi'\rangle \ , \tag{7.18}$$

where $|\psi'\rangle \in T_{j-1,0,1}$ (note that $1 \notin \{i_1, \dots, i_j\}$). Equation (7.18) follows by

$$\begin{aligned} \mathsf{M}|\psi^{0,0}_{i_1,\dots,i_j}\rangle &= \frac{1}{\sqrt{\binom{n-j-1}{t-1-j}}} \sum_{\substack{x:|x|=t-1, x_1=0 \\ x_{i_1}=\dots=x_{i_j}=1,}} \mathsf{M}|x\rangle \\ &= \frac{1}{\sqrt{\binom{n-j-1}{t-1-j}}} \sum_{\substack{x:|x|=t-1, x_1=0 \\ x_{i_1}=\dots=x_{i_j}=1}} \sum_{\ell : x_\ell=1} |1x_2 \dots x_{\ell-1} 0 x_{\ell+1} \dots x_n\rangle \\ &= \frac{n-t+1}{\sqrt{\binom{n-j-1}{t-1-j}}} \sum_{\substack{y:|y|=t-1, y_1=1 \\ y_{i_1}=\dots=y_{i_j}=1}} |y\rangle + \frac{1}{\sqrt{\binom{n-j-1}{t-1-j}}} \sum_{\ell=1}^{j} \sum_{\substack{y:|y|=t-1, y_1=1, y_{i_\ell}=0 \\ y_{i_1}=\dots=y_{i_j}=1}} |y\rangle \\ &= \frac{n-t-j+1}{\sqrt{\binom{n-j-1}{t-1-j}}} \sum_{\substack{y:|y|=t-1, y_1=1 \\ y_{i_1}=\dots=y_{i_j}=1}} |y\rangle + \frac{1}{\sqrt{\binom{n-j-1}{t-1-j}}} \sum_{\ell=1}^{j} \sum_{\substack{y:|y|=t-1, y_1=1 \\ y_{i_1}=\dots=y_{i_{\ell-1}}=1 \\ y_{i_{\ell+1}}=\dots=y_{i_j}=1}} |y\rangle \\ &= (n-t-j+1)\sqrt{\tfrac{t-1-j}{n-t+1}}|\psi^{0,1}_{i_1,\dots,i_j}\rangle + \sqrt{\tfrac{n-j}{n-t+1}} \sum_{\ell=1}^{j} |\psi^{0,1}_{i_1,\dots,i_{\ell-1},i_{\ell+1},\dots,i_j}\rangle \ . \end{aligned}$$

This proves (7.18), with $|\psi'\rangle$ equal to the second term.

Second, for every j, $\mathsf{M}(T_{j,0,0}) \subseteq T_{j,0,1}$ and $\mathsf{M}(T^\perp_{j,0,0}) \subseteq T^\perp_{j,0,1}$. The first statement follows from equation (7.18), because the subspaces $T_{j,0,0}$, $T_{j,0,1}$ are spanned by the states $|\psi^{0,0}_{i_1,\dots,i_j}\rangle$ and $|\psi^{0,1}_{i_1,\dots,i_j}\rangle$, respectively, and $T_{j-1,0,1} \subseteq T_{j,0,1}$. To prove the second statement, let $|\psi\rangle \in T^{\perp}_{j,0,0}$, $|\psi\rangle = \sum_x a_x|x\rangle$. We would like to prove $\mathsf{M}|\psi\rangle \in T^{\perp}_{j,0,1}$. This is equivalent to $\langle\psi^{0,1}_{i_1,\dots,i_j}|\mathsf{M}|\psi\rangle = 0$ for all $i_1,\dots,i_j$. We have

$$\begin{aligned}\langle\psi^{0,1}_{i_1,\dots,i_j}|\mathsf{M}|\psi\rangle &= \frac{1}{\sqrt{\binom{n-j-1}{t-j-2}}}\sum_{\substack{y:|y|=t-1,y_1=1\\ y_{i_1}=\dots=y_{i_j}=1}}\langle y|\mathsf{M}|\psi\rangle\\ &= \frac{1}{\sqrt{\binom{n-j-1}{t-j-2}}}\sum_{\substack{x:|x|=t-1,x_1=0\\ x_{i_1}=\dots=x_{i_j}=1}}\sum_{\substack{\ell:x_\ell=1\\ \ell\notin\{i_1,\dots,i_j\}}} a_x\\ &= \frac{t-1-j}{\sqrt{\binom{n-j-1}{t-j-2}}}\sum_{\substack{x:|x|=t-1,x_1=0\\ x_{i_1}=\dots=x_{i_j}=1}} a_x = 0 \ .\end{aligned}$$

The first equality follows by writing out $\langle\psi^{0,1}_{i_1,\dots,i_j}|$, the second equality follows by writing out M. The third equality follows because, for every x with $|x| = t-1$ and $x_{i_1} = \dots = x_{i_j} = 1$, there are $t-1-j$ more $\ell \in [n]$ satisfying $x_\ell = 1$. The fourth equality follows because $\sum_{\substack{x:|x|=t-1,x_1=0\\ x_{i_1}=\dots=x_{i_j}=1}} a_x$ is a constant times $\langle\psi^{0,0}_{i_1,\dots,i_j}|\psi\rangle$, and $\langle\psi^{0,0}_{i_1,\dots,i_j}|\psi\rangle = 0$ because $|\psi\rangle \in T^{\perp}_{j,0,0}$.

To deduce equation (7.17), we write

$$|\psi^{0,0}_{i_1,\dots,i_j}\rangle = |\tilde{\psi}^{0,0}_{i_1,\dots,i_j}\rangle + \Pi_{T_{j-1,0,0}}|\psi^{0,0}_{i_1,\dots,i_j}\rangle \ .$$

Since $\mathsf{M}(T_{j-1,0,0}) \subseteq T_{j-1,0,1}$ and $\mathsf{M}(T^{\perp}_{j-1,0,0}) \subseteq T^{\perp}_{j-1,0,1}$,

$$\mathsf{M}|\tilde{\psi}^{0,0}_{i_1,\dots,i_j}\rangle = \Pi_{T^{\perp}_{j-1,0,1}}\mathsf{M}|\psi^{0,0}_{i_1,\dots,i_j}\rangle = c\Pi_{T^{\perp}_{j-1,0,1}}|\psi^{0,1}_{i_1,\dots,i_j}\rangle = c|\tilde{\psi}^{0,1}_{i_1,\dots,i_j}\rangle \ ,$$

with the second equality following from (7.18) and $|\psi'\rangle \in T_{j-1,0,1}$. This proves the first half of (7.17). The second half follows similarly. Therefore

$$\langle\tilde{\psi}^{0,0}_{i_1,\dots,i_j}|\mathsf{M}^\dagger\mathsf{M}|\tilde{\psi}^{0,0}_{i'_1,\dots,i'_j}\rangle = c\cdot c'\langle\tilde{\psi}^{0,0}_{i_1,\dots,i_j}|\tilde{\psi}^{0,0}_{i'_1,\dots,i'_j}\rangle \ .$$

Hence M is a multiple of a unitary transformation. By equation (7.17), $\mathsf{U}_{01} = \mathsf{M}/c$ and, therefore, U_{01} is also a multiple of a unitary transformation.

Next, we define M by $\mathsf{M}|0x_2\dots x_n\rangle = |1x_2\dots x_n\rangle$. Then M is a unitary transformation from the space spanned by $|0x_2\dots x_n\rangle$, $x_2+\dots+x_2 = t-1$, to the space spanned by $|1x_2\dots x_n\rangle$, $1+x_2+\dots+x_n = t$. We claim that $\mathsf{U}_{11} = \mathsf{M}$. To prove that, we first observe that

$$\begin{aligned}\mathsf{M}|\psi^{0,0}_{i_1,\dots,i_j}\rangle &= \frac{1}{\sqrt{\binom{n-j-1}{t-j-1}}}\sum_{\substack{x_2,\dots,x_n:\\ x_{i_1}=\dots=x_{i_j}=1}}\mathsf{M}|0x_2\dots x_n\rangle\\ &= \frac{1}{\sqrt{\binom{n-j-1}{t-j-1}}}\sum_{\substack{x_2,\dots,x_n:\\ x_{i_1}=\dots=x_{i_j}=1}}|1x_2\dots x_n\rangle = |\psi^{1,1}_{i_1,\dots,i_j}\rangle \ .\end{aligned}$$

Since $T_{j,a,b}$ is defined as the subspace spanned by all $|\psi^{a,b}_{i_1,\ldots,i_j}\rangle$, this means that $\mathsf{M}(T_{j,0,0}) = T_{j,1,1}$ and similarly $\mathsf{M}(T_{j-1,0,0}) = T_{j-1,1,1}$. Since M is unitary, this implies $\mathsf{M}(T^{\perp}_{j-1,0,0}) = T^{\perp}_{j-1,1,1}$ and

$$\mathsf{M}|\tilde{\psi}^{0,0}_{i_1,\ldots,i_j}\rangle = \mathsf{M}\Pi_{T^{\perp}_{j-1,0,0}}|\psi^{0,0}_{i_1,\ldots,i_j}\rangle = \Pi_{T^{\perp}_{j-1,1,1}}|\psi^{1,1}_{i_1,\ldots,i_j}\rangle = |\tilde{\psi}^{1,1}_{i_1,\ldots,i_j}\rangle \ .$$

Finally, we have $\mathsf{U}_{10} = \mathsf{U}''_{10}\mathsf{U}_{11}$, where U''_{10} is defined by $\mathsf{U}''_{10}|\tilde{\psi}^{1,1}_{i_1,\ldots,i_j}\rangle = |\tilde{\psi}^{1,0}_{i_1,\ldots,i_j}\rangle$. Since U_{11} is unitary, it suffices to prove that U''_{10} is a multiple of a unitary transformation and this follows similarly to U_{01} being a multiple of a unitary transformation. □

Let $|\psi_{00}\rangle$ be an arbitrary state in $S_{j,0,0}$ for some $j \in \{0,\ldots,t-1\}$. Define $|\psi_{ab}\rangle = \mathsf{U}'_{ab}|\psi_{00}\rangle$ for $ab \in \{01, 10, 11\}$.

7.5.3. Claim. Let $\alpha'_a = \sqrt{\frac{n-(t-1+a)}{n-j}}\|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|$, $\beta'_a = \sqrt{\frac{(t-1+a)-j}{n-j}}\|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|$, $\alpha_a = \frac{\alpha'_a}{\sqrt{(\alpha'_a)^2+(\beta'_a)^2}}$, $\beta_a = \frac{\beta'_a}{\sqrt{(\alpha'_a)^2+(\beta'_a)^2}}$. Then

1. $|\phi_1\rangle = \alpha_0|\psi_{00}\rangle + \beta_0|\psi_{01}\rangle + \alpha_1|\psi_{10}\rangle + \beta_1|\psi_{11}\rangle$ belongs to $S_{j,+}$;
2. $|\phi_2\rangle = \beta_0|\psi_{00}\rangle - \alpha_0|\psi_{01}\rangle + \beta_1|\psi_{10}\rangle - \alpha_1|\psi_{11}\rangle$ belongs to $S_{j+1,+}$;
3. Any linear combination of $|\psi_{00}\rangle$, $|\psi_{01}\rangle$, $|\psi_{10}\rangle$ and $|\psi_{11}\rangle$ which is orthogonal to $|\phi_1\rangle$ and $|\phi_2\rangle$ belongs to $S_- = \bigoplus_{j=0}^{t} S_{j,-}$.

Proof. Let $i_1,\ldots,i_j$ be j distinct elements of $\{2,\ldots,n\}$. As shown in the beginning of the proof of Claim 7.5.1,

$$\begin{aligned}|\tilde{\psi}^{a}_{i_1,\ldots,i_j}\rangle &= \sqrt{\tfrac{n-(t-1+a)}{n-j}}|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle + \sqrt{\tfrac{(t-1+a)-j}{n-j}}|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle \\ &= \alpha'_a\frac{|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|} + \beta'_a\frac{|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|} \ .\end{aligned}$$

This means that $\|\tilde{\psi}^{a}_{i_1,\ldots,i_j}\| = \sqrt{(\alpha'_a)^2+(\beta'_a)^2}$ and

$$\frac{|\tilde{\psi}^{a}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a}_{i_1,\ldots,i_j}\|} = \alpha_a\frac{|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|} + \beta_a\frac{|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|} \ .$$

Since the states $|\tilde{\psi}^{0}_{i_1,\ldots,i_j}\rangle$ span $S_{j,0}$, $|\psi_{00}\rangle$ is a linear combination of states $\frac{|\tilde{\psi}^{0,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{0,0}_{i_1,\ldots,i_j}\|}$. By Claim 7.5.2, the states $|\psi_{ab}\rangle$ are linear combinations of $\frac{|\tilde{\psi}^{a,b}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,b}_{i_1,\ldots,i_j}\|}$ with the same coefficients. Therefore, $|\phi_1\rangle$ is a linear combination of

$$\alpha_0\frac{|\tilde{\psi}^{0,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{0,0}_{i_1,\ldots,i_j}\|} + \beta_0\frac{|\tilde{\psi}^{0,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{0,1}_{i_1,\ldots,i_j}\|} + \alpha_1\frac{|\tilde{\psi}^{1,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{1,0}_{i_1,\ldots,i_j}\|} + \beta_1\frac{|\tilde{\psi}^{1,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{1,1}_{i_1,\ldots,i_j}\|} = \frac{|\tilde{\psi}^{0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{0}_{i_1,\ldots,i_j}\|} + \frac{|\tilde{\psi}^{1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{1}_{i_1,\ldots,i_j}\|} \ ,$$

each of which, by definition, belongs to $S_{j,+}$.

Let $i_1, \ldots, i_j$ be distinct elements of $\{2, \ldots, n\}$. We claim

$$\frac{|\tilde{\psi}^a_{1,i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^a_{1,i_1,\ldots,i_j}\|} = \beta_a \frac{|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|} - \alpha_a \frac{|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|} \ . \tag{7.19}$$

By Claim 7.5.1, the right hand side of (7.19) belongs to $S_{j+1,a}$. We need to show that it is equal to $|\tilde{\psi}^a_{1,i_1,\ldots,i_j}\rangle$. We have

$$\begin{aligned} |\tilde{\psi}^a_{1,i_1,\ldots,i_j}\rangle &= \Pi_{T^\perp_{j,a}} |\psi^a_{1,i_1,\ldots,i_j}\rangle = \Pi_{T^\perp_{j,a}} |\psi^{a,1}_{i_1,\ldots,i_j}\rangle \\ &= \Pi_{T^\perp_{j,a}} \Pi_{T^\perp_{j-1,a,1}} |\psi^{a,1}_{i_1,\ldots,i_j}\rangle = \Pi_{T^\perp_{j,a}} |\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle \ , \end{aligned}$$

where the third equality follows from $T_{j-1,a,1} \subseteq T_{j,a}$. This is because the states $|\psi^{a,1}_{i_1,\ldots,i_{j-1}}\rangle$ spanning $T_{j-1,a,1}$ are the same as the states $|\psi^a_{1,i_1,\ldots,i_{j-1}}\rangle$ in $T_{j,a}$. Write

$$|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle = c_1|\delta_1\rangle + c_2|\delta_2\rangle \ ,$$

where

$$\begin{aligned} |\delta_1\rangle &= \alpha_a \frac{|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|} + \beta_a \frac{|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|} \ , \\ |\delta_2\rangle &= \beta_a \frac{|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\|} - \alpha_a \frac{|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|} \ . \end{aligned}$$

By Claim 7.5.1, we have $|\delta_1\rangle \in S_{j,a} \subseteq T_{j,a}$, $|\delta_2\rangle \in S_{j+1,a} \subseteq T^\perp_{j,a}$. Therefore, $\Pi_{T^\perp_{j,a}} |\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle = c_2|\delta_2\rangle$ and

$$\frac{|\tilde{\psi}^a_{1,i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^a_{1,i_1,\ldots,i_j}\|} = |\delta_2\rangle = \beta_a \frac{|\tilde{\psi}^{a,0}_{i_1,\ldots,i_j}\rangle}{\|\psi^{a,\tilde{0}}_{i_1,\ldots,i_j}\|} - \alpha_a \frac{|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{a,1}_{i_1,\ldots,i_j}\|} \ ,$$

proving (7.19).

Similarly to the argument for $|\phi_1\rangle$, equation (7.19) implies that $|\phi_2\rangle$ is a linear combination of

$$\beta_0 \frac{|\tilde{\psi}^{0,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{0,0}_{i_1,\ldots,i_j}\|} - \alpha_0 \frac{|\tilde{\psi}^{0,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{0,1}_{i_1,\ldots,i_j}\|} + \beta_1 \frac{|\tilde{\psi}^{1,0}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{1,0}_{i_1,\ldots,i_j}\|} - \alpha_1 \frac{|\tilde{\psi}^{1,1}_{i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^{1,1}_{i_1,\ldots,i_j}\|} = \frac{|\tilde{\psi}^0_{1,i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^0_{1,i_1,\ldots,i_j}\|} + \frac{|\tilde{\psi}^1_{1,i_1,\ldots,i_j}\rangle}{\|\tilde{\psi}^1_{1,i_1,\ldots,i_j}\|}$$

and each of those states belongs to $S_{j+1,+}$.

To prove the third part of Claim 7.5.3, we observe that any vector orthogonal to $|\phi_1\rangle$ and $|\phi_2\rangle$ is a linear combination of

$$|\phi_3\rangle = \alpha_0|\psi_{00}\rangle + \beta_0|\psi_{01}\rangle - \alpha_1|\psi_{10}\rangle - \beta_1|\psi_{11}\rangle \ ,$$

which, in turn, is a linear combination of vectors

$$\frac{|\tilde{\psi}^0_{i_1,\dots,i_j}\rangle}{\|\tilde{\psi}^0_{i_1,\dots,i_j}\|} - \frac{|\tilde{\psi}^1_{i_1,\dots,i_j}\rangle}{\|\tilde{\psi}^1_{i_1,\dots,i_j}\|} \ ,$$

and

$$|\phi_4\rangle = \beta_0|\psi_{00}\rangle - \alpha_0|\psi_{01}\rangle - \beta_1|\psi_{10}\rangle + \alpha_1|\psi_{11}\rangle \ ,$$

which is a linear combination of vectors

$$\frac{|\tilde{\psi}^0_{1,i_1,\dots,i_j}\rangle}{\|\tilde{\psi}^0_{1,i_1,\dots,i_j}\|} - \frac{|\tilde{\psi}^1_{1,i_1,\dots,i_j}\rangle}{\|\tilde{\psi}^1_{1,i_1,\dots,i_j}\|} \ .$$

This means that we have $|\phi_3\rangle \in S_{j,-}$ and $|\phi_4\rangle \in S_{j+1,-}$. □

7.6 Norms of projected basis states

7.6.1. Claim. Let $j < t/2$ and $x^{\underline{j}} = x(x-1)\cdots(x-j+1)$.

1. $\|\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\| = \sqrt{\dfrac{(n-t-a+b)^{\underline{j}}}{(n-j)^{\underline{j}}}}$.
2. $\|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\| \geq \dfrac{1}{\sqrt{2}}\|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\|$.
3. $\dfrac{\|\tilde{\psi}^{0,0}_{i_1,\dots,i_j}\| \cdot \|\tilde{\psi}^{1,1}_{i_1,\dots,i_j}\|}{\|\tilde{\psi}^{0,1}_{i_1,\dots,i_j}\| \cdot \|\tilde{\psi}^{1,0}_{i_1,\dots,i_j}\|} = 1 + O\left(\dfrac{1}{t}\right)$.

Proof. Define $t_a = t-1+a$. We calculate the vector

$$|\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\rangle = \Pi_{T^\perp_{j-1,a,b}}|\psi^{a,b}_{i_1,\dots,i_j}\rangle \ .$$

Both vector $|\psi^{a,b}_{i_1,\dots,i_j}\rangle$ and subspace $T_{j-1,a,b}$ are fixed by

$$\mathsf{U}_\pi|x\rangle = |x_{\pi(1)}\dots x_{\pi(n)}\rangle$$

for any permutation π that fixes 1 and maps $\{i_1,\dots,i_j\}$ to itself. Hence $|\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\rangle$ is fixed by any such U_π as well. Therefore, the amplitude of $|x\rangle$ with $|x| = t_a$, $x_1 = b$ in $|\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\rangle$ only depends on $|\{i_1,\dots,i_j\} \cap \{i : x_i = 1\}|$, so $|\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\rangle$ is of the form

$$|\upsilon_{a,b}\rangle = \sum_{m=0}^{j} \kappa_m \sum_{\substack{x:|x|=t_a, x_1=b \\ |\{i_1,\dots,i_j\}\cap\{i:x_i=1\}|=m}} |x\rangle \ .$$

To simplify the following calculations, we multiply $\kappa_0, \ldots, \kappa_j$ by the same constant so that $\kappa_j = 1/\sqrt{\binom{n-j-1}{t_a-j-b}}$. Then $|\tilde{\psi}^{a,b}_{i_1,\ldots,i_j}\rangle$ remains a multiple of $|\upsilon_{a,b}\rangle$ but may no longer be equal to $|\upsilon_{a,b}\rangle$.

$\kappa_0, \ldots, \kappa_{j-1}$ should be such that the state is orthogonal to $T_{j-1,a,b}$ and, in particular, orthogonal to the states $|\psi^{a,b}_{i_1,\ldots,i_\ell}\rangle$ for all $\ell \in \{0, \ldots, j-1\}$. By writing out $\langle \upsilon_{a,b}|\psi^{a,b}_{i_1,\ldots,i_\ell}\rangle = 0$,

$$\sum_{m=\ell}^{j} \kappa_m \binom{n-j-1}{t_a-m-b}\binom{j-\ell}{m-\ell} = 0 \ . \tag{7.20}$$

To show that, we first note that $|\psi^{a,b}_{i_1,\ldots,i_\ell}\rangle$ is a uniform superposition of all $|x\rangle$ with $|x| = t_a$, $x_1 = b$, $x_{i_1} = \cdots = x_{i_\ell} = 1$. If we want to choose x subject to those constraints and also satisfying $|\{i_1, \ldots, i_j\} \cap \{i : x_i = 1\}| = m$, then we have to set $x_i = 1$ for $m-\ell$ different $i \in \{i_{\ell+1}, \ldots, i_j\}$ and for $t_a - m - b$ different $i \notin \{1, i_1, \ldots, i_j\}$. This can be done in $\binom{j-\ell}{m-\ell}$ and $\binom{n-j-1}{t_a-m-b}$ different ways, respectively.

By solving the system of equations (7.20), starting from $\ell = j-1$ and going down to $\ell = 0$, we get that the only solution is

$$\kappa_m = (-1)^{j-m} \frac{\binom{n-j-1}{t_a-j-b}}{\binom{n-j-1}{t_a-m-b}} \kappa_j \ . \tag{7.21}$$

Let $|\upsilon'_{a,b}\rangle = \frac{|\upsilon_{a,b}\rangle}{\|\upsilon_{a,b}\|}$ be the normalized version of $|\upsilon_{a,b}\rangle$. Then

$$\begin{aligned} |\tilde{\psi}^{a,b}_{i_1,\ldots,i_j}\rangle &= \langle \upsilon'_{a,b}|\psi^{a,b}_{i_1,\ldots,i_j}\rangle |\upsilon'_{a,b}\rangle \ , \\ \|\tilde{\psi}^{a,b}_{i_1,\ldots,i_j}\| &= \langle \upsilon'_{a,b}|\psi^{a,b}_{i_1,\ldots,i_j}\rangle = \frac{\langle \upsilon_{a,b}|\psi^{a,b}_{i_1,\ldots,i_j}\rangle}{\|\upsilon_{a,b}\|} \ . \end{aligned} \tag{7.22}$$

We have

$$\langle \upsilon_{a,b}|\psi^{a,b}_{i_1,\ldots,i_j}\rangle = 1 \ , \tag{7.23}$$

because $|\psi^{a,b}_{i_1,\ldots,i_j}\rangle$ consists of $\binom{n-j-1}{t_a-j-b}$ basis states $|x\rangle$, $x_1 = b$, $x_{i_1} = \cdots = x_{i_j} = 1$, each having amplitude $1/\sqrt{\binom{n-j-1}{t_a-j-b}}$ in both $|\upsilon_{a,b}\rangle$ and $|\psi^{a,b}_{i_1,\ldots,i_j}\rangle$. Furthermore,

$$\begin{aligned} \|\upsilon_{a,b}\|^2 &= \sum_{m=0}^{j} \binom{j}{m}\binom{n-j-1}{t_a-m-b}\kappa_m^2 \\ &= \sum_{m=0}^{j} \binom{j}{m} \frac{\binom{n-j-1}{t_a-j-b}^2}{\binom{n-j-1}{t_a-m-b}} \kappa_j^2 \\ &= \sum_{m=0}^{j} \binom{j}{m} \frac{\binom{n-j-1}{t_a-j-b}}{\binom{n-j-1}{t_a-m-b}} \end{aligned}$$

$$= \sum_{m=0}^{j} \binom{j}{m} \frac{(t_a - m - b)!(n - t_a + m - j - 1 + b)!}{(t_a - j - b)!(n - t_a - 1 + b)!}$$

$$= \sum_{m=0}^{j} \binom{j}{m} \frac{(t_a - m - b)^{\underline{j-m}}}{(n - t_a - 1 + b)^{\underline{j-m}}} \quad . \tag{7.24}$$

Here the first equality follows because there are $\binom{j}{m}\binom{n-j-1}{t_a-m-b}$ vectors x such that $|x| = t_a$, $x_1 = b$, $x_i = 1$ for m different $i \in \{i_1, \dots, i_j\}$ and $t_a - m$ different $i \notin \{1, i_1, \dots, i_j\}$, the second equality follows from equation (7.21) and the third equality follows from our choice $\kappa_j = 1/\sqrt{\binom{n-j-1}{t_a-j-b}}$.

From equations (7.22), (7.23), and (7.24), $\|\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\| = \frac{1}{\sqrt{A_{a,b}}}$ where $A_{a,b} = \sum_{m=0}^{\infty} C_{a,b}(m)$ and

$$C_{a,b}(m) = \binom{j}{m} \frac{(t_a - m - b)^{\underline{j-m}}}{(n - t_a - 1 + b)^{\underline{j-m}}} \quad .$$

The terms with $m > j$ are zero because $\binom{j}{m} = 0$ for $m > j$.

We compute the combinatorial sum $A_{a,b}$ using hyper-geometric series [GKP98, Section 5.5]. Since

$$\frac{C_{a,b}(m+1)}{C_{a,b}(m)} = \frac{(m-j)(m+n-t_a-j+b)}{(m+1)(m-t_a+b)}$$

is a rational function of m, $A_{a,b}$ is a hyper-geometric series and its value is

$$A_{a,b} = \sum_{m=0}^{\infty} C_{a,b}(m) = C_{a,b}(0) \cdot F\left(\begin{matrix} -j,\ n - t_a - j + b \\ -t_a + b \end{matrix} \middle| 1\right) \quad .$$

We apply Vandermonde's convolution $F\left(\begin{smallmatrix} -j,\ x \\ y \end{smallmatrix}\middle|1\right) = (x-y)^{\underline{j}}/(-y)^{\underline{j}}$ [GKP98, Equation 5.93 on page 212], which holds for every integer $j \geq 0$, and obtain

$$A_{a,b} = \frac{(t_a - b)^{\underline{j}}}{(n - t_a - 1 + b)^{\underline{j}}} \cdot \frac{(n-j)^{\underline{j}}}{(t_a - b)^{\underline{j}}} = \frac{(n-j)^{\underline{j}}}{(n - t_a - 1 + b)^{\underline{j}}} \quad .$$

This proves the first part of the claim, that $\|\tilde{\psi}^{a,b}_{i_1,\dots,i_j}\| = \sqrt{\frac{(n-t_a-1+b)^{\underline{j}}}{(n-j)^{\underline{j}}}}$.

The second part of the claim follows because

$$\frac{\|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\|}{\|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\|} = \sqrt{\frac{(n - t_a - 1)^{\underline{j}}}{(n - t_a)^{\underline{j}}}} = \sqrt{1 - \frac{j}{n - t_a}} \geq \sqrt{1 - \frac{n/4}{n/2}} = \frac{1}{\sqrt{2}} \quad ,$$

because $j \leq t_a/2$, and $t_a \leq n/2$.

For the third part,

$$\frac{A_{1,0}A_{0,1}}{A_{0,0}A_{1,1}} = \frac{((n-t)^{\underline{j}})^2}{(n-t+1)^{\underline{j}}(n-t-1)^{\underline{j}}} = \frac{(n-t)(n-t-j+1)}{(n-t+1)(n-t-j)}$$
$$= 1 + \frac{j}{(n-t+1)(n-t-j)} \ ,$$

which is $1+\Theta(j/n^2) = 1 + O(\frac{1}{t})$ for $t \le n/2$ and $j \le t/2$. The expression in the third part of the claim is the square root of this value, hence it is $1 + O(\frac{1}{t})$. □

7.6.2. Claim. If $j < t/2$, then $\beta_a \le \sqrt{\dfrac{2t}{n}}$.

Proof. Define $t_a = t - 1 + a$. By Claim 7.6.1, $\|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\| \ge \frac{1}{\sqrt{2}}\|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\|$. That implies

$$\alpha'_a = \frac{\sqrt{n-t_a}}{\sqrt{n-j}}\|\tilde{\psi}^{a,0}_{i_1,\dots,i_j}\| \ge \frac{1}{\sqrt{2}}\frac{\sqrt{n-t_a}}{\sqrt{t_a-j}}\frac{\sqrt{t_a-j}}{\sqrt{n-j}}\|\tilde{\psi}^{a,1}_{i_1,\dots,i_j}\| = \frac{\sqrt{n-t_a}}{\sqrt{2(t_a-j)}}\beta'_a$$

and hence

$$\sqrt{(\alpha'_a)^2 + (\beta'_a)^2} \ge \beta'_a\sqrt{\frac{n-t_a}{2(t_a-j)}+1} = \beta'_a\frac{\sqrt{n+t_a-2j}}{\sqrt{2(t_a-j)}} \ .$$

Then, using $j \le \frac{t_a}{2}$,

$$\beta_a = \frac{\beta'_a}{\sqrt{(\alpha'_a)^2+(\beta'_a)^2}} \le \frac{\sqrt{2(t_a-j)}}{\sqrt{n+t_a-2j}} \le \sqrt{\frac{2t}{n}} \ ,$$

which we had to prove. □

7.6.3. Claim. If $j < t/2$, then $|\alpha_0\beta_1 - \alpha_1\beta_0| = O\left(\dfrac{1}{\sqrt{tn}}\right)$.

Proof. We first estimate

$$\frac{\alpha_0\beta_1}{\alpha_1\beta_0} = \frac{\alpha'_0\beta'_1}{\alpha'_1\beta'_0} = \frac{\sqrt{(n-t+1)(t-j)}}{\sqrt{(n-t)(t-1-j)}} \cdot \frac{\|\tilde{\psi}^{0,0}_{i_1,\dots,i_j}\|\|\tilde{\psi}^{1,1}_{i_1,\dots,i_j}\|}{\|\tilde{\psi}^{1,0}_{i_1,\dots,i_j}\|\|\tilde{\psi}^{0,1}_{i_1,\dots,i_j}\|} \ .$$

By Claim 7.6.1, we have

$$\frac{\alpha'_0\beta'_1}{\alpha'_1\beta'_0} = \left(1 + O\left(\frac{1}{t}\right)\right)\frac{\sqrt{(n-t+1)(t-j)}}{\sqrt{(n-t)(t-1-j)}} \ .$$

Since $\frac{\sqrt{t-j}}{\sqrt{t-1-j}} = \sqrt{1 + \frac{1}{t-1-j}} = 1+O(\frac{1}{t-1-j}) = 1+O(\frac{1}{t})$ and, similarly, $\frac{\sqrt{n-t+1}}{\sqrt{n-t}} = 1 + O(\frac{1}{n-t}) = 1 + O(\frac{1}{t})$, we have shown that $\frac{\alpha_0\beta_1}{\alpha_1\beta_0}$ is of order $1 + O(\frac{1}{t})$. We thus have

$$|\alpha_0\beta_1 - \beta_0\alpha_1| = O\left(\frac{1}{t}\right)|\beta_0\alpha_1| = O\left(\frac{1}{t}\cdot\sqrt{\frac{t}{n}}\right) = O\left(\frac{1}{\sqrt{tn}}\right) \ ,$$

thanks to Claim 7.6.2 and the fact that $|\alpha_1| \leq 1$. □

7.7 Change of the potential function

Proof of Lemma 7.2.3 We first analyze the case when $\rho_{d,1}$ belongs to the subspace $\mathcal{H}_4$ spanned by $|\psi_{ab}\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_k\rangle$, where $|\psi_2\rangle, \ldots, |\psi_k\rangle$ are some vectors from subspaces $R_{j_2}, \ldots, R_{j_k}$ for some $j_2, \ldots, j_k$, $|\psi_{00}\rangle$ is an arbitrary state in $S_{j,0,0}$ for some $j \in \{0, \ldots, t-1\}$, $|\psi_{ab}\rangle = \mathsf{U}'_{ab}|\psi_{00}\rangle$ for $ab \in \{01, 10, 11\}$, and U'_{ab} are unitaries from Claim 7.5.2.

We pick an orthonormal basis for $\mathcal{H}_4$ that has $|\phi_1\rangle$ and $|\phi_2\rangle$ from Claim 7.5.3 as its first two vectors. Let $|\phi_3\rangle$ and $|\phi_4\rangle$ be the other two basis vectors. We define

$$|\chi_i\rangle = |\phi_i\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_k\rangle \ . \tag{7.25}$$

By Claim 7.5.3, $|\chi_1\rangle$ belongs to $S_{j,+} \otimes R_{j_2} \otimes \cdots \otimes R_{j_k}$ which is contained in $\mathcal{R}_{\min(j,t/2)+j_2+\cdots+j_k}$. Similarly, $|\chi_2\rangle$ belongs to $\mathcal{R}_{\min(j+1,t/2)+j_2+\cdots+j_k}$ and $|\chi_3\rangle$, $|\chi_4\rangle$ belong to $\mathcal{R}_{t/2+j_2+\cdots+j_k}$. If $j < t/2$, this means that

$$\begin{aligned} P(\rho_{d,1}) = {} & q^{j_2+\cdots+j_k} \cdot \Big(q^j\langle\chi_1|\rho_{d,1}|\chi_1\rangle + q^{j+1}\langle\chi_2|\rho_{d,1}|\chi_2\rangle \\ & + q^{\frac{t}{2}}\langle\chi_3|\rho_{d,1}|\chi_3\rangle + q^{\frac{t}{2}}\langle\chi_4|\rho_{d,1}|\chi_4\rangle\Big) \ . \end{aligned} \tag{7.26}$$

If $j \geq t/2$, then $|\chi_1\rangle$, $|\chi_2\rangle$, $|\chi_3\rangle$, $|\chi_4\rangle$ are all in $\mathcal{R}_{t/2+j_2+\cdots+j_k}$. This means that $P(\rho_{d,1}) = q^{t/2+j_2+\cdots+j_k}$ and it remains unchanged by a query.

We define $\gamma_\ell = \langle\chi_\ell|\rho_{d,1}|\chi_\ell\rangle$. Since the support of $\rho_{d,1}$ is contained in the subspace spanned by $|\chi_\ell\rangle$, we have $\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 = \operatorname{Tr}\rho_{d,1} = 1$. This means that equation (7.26) can be rewritten as

$$\begin{aligned} P(\rho_{d,1}) &= q^{j+j_2+\cdots+j_k}\gamma_1 + q^{j+j_2+\cdots+j_k+1}\gamma_2 + q^{t/2+j_2+\cdots+j_k}(\gamma_3+\gamma_4) \\ &= q^{t/2+j_2+\cdots+j_k} + q^{j_2+\cdots+j_k}(q^{j+1} - q^{t/2})(\gamma_1+\gamma_2) \\ &\quad + q^{j_2+\cdots+j_k}(q^j - q^{j+1})\gamma_1 \ . \end{aligned} \tag{7.27}$$

$P(\rho'_{d,1})$ can be also expressed in a similar way, with $\gamma'_j = \langle\chi_j|\rho'_{d,1}|\chi_j\rangle$ instead of γ_j. By combining equations (7.27) for $P(\rho_{d,1})$ and $P(\rho'_{d,1})$, we get

$$\begin{aligned} P(\rho'_{d,1}) - P(\rho_{d,1}) = {} & q^{j+j_2+\cdots+j_k}(q^{t/2-j} - q)(\gamma_1 + \gamma_2 - \gamma'_1 - \gamma'_2) \\ & + q^{j+j_2+\cdots+j_k}(q-1)(\gamma_1 - \gamma'_1) \ . \end{aligned}$$

Therefore, it suffices to bound $|\gamma_1+\gamma_2-\gamma_1'-\gamma_2'|$ and $|\gamma_1-\gamma_1'|$. W.l.o.g. we can assume that $\rho_{d,1}$ is a pure state $|\varphi\rangle\langle\varphi|$. Let

$$|\varphi\rangle = (a|\psi_{00}\rangle + b|\psi_{01}\rangle + c|\psi_{10}\rangle + d|\psi_{11}\rangle) \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_k\rangle \ .$$

Then the state after a query is

$$|\varphi'\rangle = (a|\psi_{00}\rangle - b|\psi_{01}\rangle + c|\psi_{10}\rangle - d|\psi_{11}\rangle) \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_k\rangle$$

and we have to bound

$$\gamma_\ell - \gamma_\ell' = |\langle\chi_\ell|\varphi\rangle|^2 - |\langle\chi_\ell|\varphi'\rangle|^2$$

for $\ell \in \{1,2\}$. For $\ell = 1$, we have

$$\langle\chi_1|\varphi\rangle = a\alpha_0 + b\beta_0 + c\alpha_1 + d\beta_1 \ .$$

The expression for φ' is similar, with minus signs in front of $b\beta_0$ and $d\beta_1$. Therefore,

$$\big||\langle\chi_1|\varphi\rangle|^2 - |\langle\chi_1|\varphi'\rangle|^2\big| \leq 4|a||b|\alpha_0\beta_0 + 4|c||d|\alpha_1\beta_1 + 4|a||d|\alpha_0\beta_1 + 4|b||c|\alpha_1\beta_0 \ . \tag{7.28}$$

Since $|a|$, $|b|$, $|c|$, $|d|$ are all at most $\|\varphi\| = 1$ and α_0, α_1 are less than 1, equation (7.28) is at most $8\beta_0 + 8\beta_1$. By Claim 7.6.2, we have

$$|\gamma_1 - \gamma_1'| \leq 8\beta_0 + 8\beta_1 \leq 16\sqrt{\frac{2t}{n}} \ .$$

We also have

$$\begin{aligned} |\gamma_1+\gamma_2-\gamma_1'-\gamma_2'| &= \big||\langle\chi_1|\varphi\rangle|^2 + |\langle\chi_2|\varphi\rangle|^2 - |\langle\chi_1|\varphi'\rangle|^2 - |\langle\chi_2|\varphi'\rangle|^2\big| \\ &\leq 4|a||d||\alpha_0\beta_1 - \alpha_1\beta_0| + 4|b||c||\alpha_1\beta_0 - \alpha_0\beta_1| \\ &\leq 8|\alpha_0\beta_1 - \alpha_1\beta_0| \leq \frac{8C}{\sqrt{tn}} \ , \end{aligned}$$

where C is the big-O constant from Claim 7.6.3. By taking into account that $P(\rho_{d,1}) \geq q^{j+j_2+\cdots+j_k}$,

$$\begin{aligned} P(|\varphi'\rangle\langle\varphi'|) - P(|\varphi\rangle\langle\varphi|) &\leq \left((q^{t/2-j} - q)\frac{8C}{\sqrt{tn}} + (q-1)\frac{16\sqrt{2t}}{\sqrt{n}}\right) P(|\varphi\rangle\langle\varphi|) \\ &\leq \left((q^{t/2} - 1)\frac{8C}{\sqrt{tn}} + (q-1)\frac{16\sqrt{2t}}{\sqrt{n}}\right) P(|\varphi\rangle\langle\varphi|) \ . \end{aligned} \tag{7.29}$$

This proves Lemma 7.2.3 for the case when the support of $\rho_{d,1}$ is contained in $\mathcal{H}_4$. (If $\rho_{d,1}$ is a mixed state, we just express it as a mixture of pure states $|\varphi\rangle$. The bound for $\rho_{d,1}$ follows by summing equations (7.29) for every $|\varphi\rangle$.)

For the general case, we divide the entire state space $\mathcal{H}_I$ into 4-dimensional subspaces. To do that, we first subdivide $\mathcal{H}_I$ into subspaces

$$(S_{j,0,0} \oplus S_{j,0,1} \oplus S_{j,1,0} \oplus S_{j,1,1}) \otimes R_{j_2} \otimes \cdots \otimes R_{j_k} \ . \tag{7.30}$$

Let states $|\psi_{1,i}^{0,0}\rangle$, $i \in [\dim S_{j,0,0}]$ form a basis for $S_{j,0,0}$ and let $|\psi_{1,i}^{a,b}\rangle = \mathsf{U}'_{ab}|\psi_{1,i}^{0,0}\rangle$ for $(a,b) \in \{(0,1),(1,0),(1,1)\}$, where the U'_{ab} are the unitaries from Claim 7.5.2. Then the $|\psi_{1,i}^{a,b}\rangle$ form a basis for $S_{j,a,b}$.

Let $|\psi_{l,i}\rangle$, $i \in [\dim R_{j_l}]$, form a basis for R_{j_l}, $l \in \{2,\dots,k\}$. We subdivide (7.30) into 4-dimensional subspaces $H_{i_1,\dots,i_k}$ spanned by

$$|\psi_{1,i_1}^{a,b}\rangle \otimes |\psi_{2,i_2}\rangle \otimes \cdots \otimes |\psi_{k,i_k}\rangle \ ,$$

where a, b range over $\{0,1\}$. Let $\mathcal{H}_{all}$ be the collection of all $H_{i_1,\dots,i_k}$ obtained by subdividing all subspaces (7.30). We claim that

$$P(\rho) = \sum_{H \in \mathcal{H}_{all}} P(\Pi_H \rho) \ . \tag{7.31}$$

Equation (7.31) together with equation (7.29) implies Lemma 7.2.3. Since $P(\rho)$ is defined as a weighted sum of traces $\operatorname{Tr} \Pi_{\mathcal{R}_m} \rho$, we can prove equation (7.31) by showing

$$\operatorname{Tr} \Pi_{\mathcal{R}_m} \rho_{d,1} = \sum_{H \in \mathcal{H}_{all}} \operatorname{Tr} \Pi_{\mathcal{R}_m} \Pi_H \rho_{d,1} \ . \tag{7.32}$$

To prove (7.32), we define a basis for $\mathcal{H}_I$ by first decomposing $\mathcal{H}_I$ into subspaces $H \in \mathcal{H}_{all}$, and then for each subspace, taking the basis consisting of $|\chi_1\rangle$, $|\chi_2\rangle$, $|\chi_3\rangle$ and $|\chi_4\rangle$ defined by equation (7.25). By Claim 7.5.3, each of the basis states belongs to one of the subspaces $\mathcal{R}_m$. This means that each $\mathcal{R}_m$ is spanned by some subset of this basis.

The left hand side of (7.32) is equal to the sum of squared projections of $\rho_{d,1}$ to basis states $|\chi_j\rangle$ that belong to $\mathcal{R}_m$. Each of the terms $\operatorname{Tr} \Pi_{\mathcal{R}_m} \Pi_H \rho_{d,1}$ on the right hand side is equal to the sum of squared projections to basis states $|\chi_j\rangle$ that belong to $\mathcal{R}_m \cap H$. Summing over all H gives the sum of squared projections of $\rho_{d,1}$ to all $|\chi_j\rangle$ that belong to $\mathcal{R}_m$. Therefore, the two sides of (7.32) are equal. □

7.8 Summary

We have described a new version of the adversary method for quantum query lower bounds, based on analyzing the subspaces of the problem we want to lower-bound. We have proved a new quantum direct product theorem for all symmetric functions. The result is tight up to a constant factor.

Chapter 8

Time-Space Tradeoffs

This chapter is based on the following papers:

[KŠW04] H. Klauck, R. Špalek, and R. de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. In *Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 12–21, 2004. To appear in SIAM Journal on Computing.

[AŠW06] A. Ambainis, R. Špalek, and R. de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. In *Proceedings of 38th Annual ACM Symposium on Theory of Computing*, pages 618–633, 2006.

8.1 Introduction

A *time-space tradeoff* is a relation between the running time and the space complexity of an algorithm. Basically, the more memory we allow the algorithm to use the faster it could possibly run. Such tradeoffs between the two main computational resources are well known classically for problems like sorting, element distinctness, hashing, etc. Our direct product theorems from Chapter 6 and Chapter 7, apart from answering a fundamental question about the computational models of (quantum) query complexity and communication complexity, also imply a number of new and optimal time-space tradeoffs.

Sorting First, we consider the tradeoff between the time T and space S that a quantum circuit needs for *sorting* N numbers. Classically, it is well known that $TS = \Omega(N^2)$ and that this tradeoff is achievable [Bea91]. The classical lower bound holds for the most general model of branching programs, and the optimal classical algorithm works in the (more restrictive) circuit model. In the quantum case, Klauck [Kla03] constructed a bounded-error quantum algorithm that runs in time $T = O((N \log N)^{3/2}/\sqrt{S})$ for all $(\log N)^3 \le S \le \frac{N}{\log N}$. He also claimed a lower bound $TS = \Omega(N^{3/2})$, which would be close to optimal for small S but not for large S. Unfortunately there is an error in the proof presented in [Kla03] (Lemma 5 appears to be wrong). Here we use our quantum DPT for one threshold function from Theorem 6.3.6 to prove the tradeoff $T^2S = \Omega(N^3)$ in the circuit model. Our lower-bound technique does not work for branching programs. This tradeoff is tight up to polylogarithmic factors.

Boolean matrix multiplication Secondly, we consider time-space tradeoffs for the problems of *Boolean matrix-vector product* and *Boolean matrix product.* In the first problem there are an $N \times N$ matrix A and a vector b of dimension N, and the goal is to compute the vector $c = Ab$, where

$$c_i = \bigvee_{j=1}^{N} (A[i,j] \wedge b_j) \quad .$$

In the setting of time-space tradeoffs, the matrix A is fixed and the input is the vector b. In the problem of matrix multiplication two matrices have to be multiplied with the same type of Boolean product, and both are inputs.

Time-space tradeoffs for Boolean matrix-vector multiplication have been analyzed in an average-case scenario by Abrahamson [Abr90], whose results give a worst-case lower bound of $TS = \Omega(N^{3/2})$ for classical algorithms. He conjectured that a worst-case lower bound of $TS = \Omega(N^2)$ holds. Using our classical DPT for OR from Theorem 6.2.4 we are able to confirm this, that is there is a matrix A, such that computing Ab requires $TS = \Omega(N^2)$. We also show a lower bound of $T^2S = \Omega(N^3)$ for this problem in the quantum case using the quantum DPT for OR from Theorem 6.3.8. Both bounds are tight (the second within a logarithmic factor) if T is taken to be the number of queries to the inputs. We also get a lower bound of $T^2S = \Omega(N^5)$ for the problem of multiplying two matrices in the quantum case. This bound is close to optimal for small S; it is open whether it is close to optimal for large S.

Communication-space tradeoffs Research on classical communication-space tradeoffs has been initiated by Lam et al. [LTT92] in a restricted setting, and by Beame et al. [BTY94] in a general model of space-bounded communication complexity. In the setting of communication-space tradeoffs, players Alice and Bob are modeled as space-bounded circuits, and we are interested in the communication cost when given particular space bounds.

We consider communication-space tradeoffs for the above problems of Boolean matrix multiplication. For the problem of computing the matrix-vector product Alice receives the matrix A (now an input) and Bob the vector b. Beame et al. gave tight lower bounds for example for the matrix-vector product and matrix product over $\mathbb{GF}(2)$, but stated the complexity of Boolean matrix-vector multiplication as an open problem. Using our quantum DPT for Disjointness from Theorem 6.4.2 we are able to show that any quantum protocol for this problem satisfies $C^2S = \Omega(N^3)$. This is tight within a polylogarithmic factor. We also get a lower bound of $C^2S = \Omega(N^5)$ for computing the product of two matrices, which again is tight.

Note that no classical lower bounds for these problems were known previously, and that finding better classical lower bounds than these remains open. The possibility to show good quantum bounds comes from the deep relation between quantum protocols and polynomials implicit in Razborov's lower bound technique [Raz03].

Evaluating solutions to systems of linear inequalities Furthermore, we apply the quantum DPT for symmetric functions from Theorem 7.1.1 to the following problem. Let A be a fixed $N \times N$ matrix of non-negative integers. Our inputs are column vectors $x = (x_1, \ldots, x_N)$ and $b = (b_1, \ldots, b_N)$ of non-negative integers. We are interested in the system

$$Ax \geq b$$

of N linear inequalities, and want to find out which of these inequalities hold (we could also mix $\geq$, $=$, and $\leq$, but omit that for ease of notation). Note that if A and x are Boolean and $b = (t, \ldots, t)$, this gives N overlapping t-threshold functions. Note that the output is an N-bit vector. Again, we want to analyze the tradeoff between the time T and space S needed to solve this problem. Lower bounds on T will be in terms of query complexity. For simplicity we omit polylogarithmic factors in the following discussion.

In the classical world, the optimal tradeoff is $TS = N^2$, independent of the values in b. The upper bounds are for deterministic algorithms, and the lower bounds are for 2-sided error algorithms and they follow from our tradeoffs for Boolean matrix multiplication. In the quantum world the situation is more complex. Let us put an upper bound $\max\{b_i\} \leq t$. We have the following two regimes for 2-sided error quantum algorithms:

- *Quantum regime.* If $S \leq \frac{N}{t}$, then the optimal tradeoff is $T^2S = tN^3$.
- *Classical regime.* If $S > \frac{N}{t}$, then the optimal tradeoff is $TS = N^2$.

In the quantum regime, quantum algorithms performs better than classical, whereas in the classical regime, they both perform equally well. Our lower bounds hold even for the constrained situation where b is fixed to the all-t vector, A and x are Boolean, and A is sparse in having only $O(\frac{N}{S})$ nonzero entries in each row.

Stronger lower bound for 1-sided error algorithms Since our quantum DPT for 1-sided error threshold functions from Theorem 6.3.9 is stronger that the 2-sided DPT from Theorem 7.1.1 by an extra factor of t in the exponent, we obtain the following stronger lower bound for 1-sided error algorithms:

- If $t \leq S \leq \frac{N}{t^2}$, then the tradeoff for 1-sided error algorithms is $T^2 S \geq t^2 N^3$.
- If $S > \frac{N}{t^2}$, then the optimal tradeoff for 1-sided error algorithms is $TS = N^2$.

We do not know whether the lower bound in the first case is optimal (probably it is not), but note that it is stronger than the optimal bounds that we have for 2-sided error algorithms. This is the first separation of 2-sided and 1-sided error algorithms in the context of quantum time-space tradeoffs. Strictly speaking, there is a quadratic gap for OR, but space $\log n$ suffices for the fastest 1-sided and 2-sided error algorithms so there is no real tradeoff in that case.

8.1.1. Remark. The time-space tradeoffs for 2-sided error algorithms for $Ax \geq b$ similarly hold for a system of N equalities, $Ax = b$. The upper bound clearly carries over, while the lower holds for equalities as well, because our DPT from Theorem 7.1.1 holds even under the promise that the input has weight t or $t-1$. In contrast, the stronger 1-sided error time-space tradeoff does not automatically carry over to systems of equalities, because we do not know how to prove the DPT from Theorem 6.3.9 under this promise.

8.2 Preliminaries

Circuit model For investigating time-space tradeoffs we use the circuit model. A circuit accesses its input via an oracle like a query algorithm; see Section 1.2.2. Time corresponds to the number of gates in the circuit. We will, however, usually consider the number of queries to the input, which is obviously a lower bound on time. A quantum circuit uses space S if it works with S qubits only. We require that the outputs are made at predefined gates in the circuit, by writing their value to some extra qubits that may not be used later on. Similar definitions are made for classical circuits.

Communicating quantum circuits In the model of quantum communication complexity, two players Alice and Bob compute a function f on distributed inputs x and y. The complexity measure of interest in this setting is the amount of communication. The players follow some predefined protocol that consists of local unitary operations, and the exchange of qubits. The communication cost of a protocol is the maximal number of qubits exchanged for any input. In the standard model of communication complexity, Alice and Bob are computationally unbounded entities, but we are also interested in what happens if they have bounded memory, that is they work with a bounded number of qubits. To this end we model Alice and Bob as communicating quantum circuits, following Yao [Yao93].

A pair of communicating quantum circuits is actually a single quantum circuit partitioned into two parts. The allowed operations are local unitary operations and access to the inputs that are given by oracles. Alice's part of the circuit may use oracle gates to read single bits from her input, and Bob's part of the circuit may do so for his input. The communication C between the two parties is simply the number of wires carrying qubits that cross between the two parts of the circuit. A pair of communicating quantum circuits uses space S, if the whole circuit works on S qubits.

In the problems we consider, the number of outputs is much larger than the memory of the players. Therefore we use the following output convention. The player who computes the value of an output sends this value to the other player at a predetermined point in the protocol. In order to make the models as general as possible, we furthermore allow the players to do local measurements, and to throw qubits away as well as pick up some fresh qubits. The space requirement only demands that at any given time no more than S qubits are in use in the whole circuit.

A final comment regarding upper bounds: Buhrman et al. [BCW98] showed how to run a query algorithm in a distributed fashion with small overhead in the communication. In particular, if there is a T-query quantum algorithm computing N-bit function f, then there is a pair of communicating quantum circuits with $O(T \log N)$ communication that computes $f(x \wedge y)$ with the same success probability. We refer to the book of Kushilevitz and Nisan [KN97] for more on communication complexity in general, and to the surveys [Kla00, Buh00, Wol02] for more on its quantum variety.

8.3 Time-space tradeoff for sorting

We will now use our strong direct product theorem to get near-optimal time-space tradeoffs for quantum circuits for sorting. In our model, the numbers $a_1, \ldots, a_N$ that we want to sort can be accessed by means of queries, and the number of queries lower-bounds the actual time taken by the circuit. The circuit has N output gates and in the course of its computation outputs the N numbers in sorted (say, descending) order, with success probability at least $\frac{2}{3}$.

8.3.1. Theorem. *Every bounded-error quantum circuit for sorting N numbers that uses T queries and space S satisfies $T^2 S = \Omega(N^3)$.*

Proof. We *slice* the circuit along the time-axis into $L = T/\alpha\sqrt{SN}$ slices, each containing $\frac{T}{L} = \alpha\sqrt{SN}$ queries. Each such slice has a number of output gates. Consider any slice. Suppose it contains output gates $i, i+1, \ldots, i+k-1$, for $i \leq \frac{N}{2}$, so it is supposed to output the i^{th} up to $i+k-1^{\text{st}}$ largest elements of its input. We want to show that $k = O(S)$. If $k \leq S$ then we are done, so assume $k > S$. We can use the slice as a k-threshold algorithm on $\frac{N}{2}$ bits, as follows. For an $\frac{N}{2}$-bit input x, construct a sorting input by taking $i-1$ copies of the number 2, the $\frac{N}{2}$ bits in x, and $\frac{N}{2}-i+1$ copies of the number 0, and append their position behind the numbers.

Consider the behavior of the sorting circuit on this input. The first part of the circuit has to output the $i-1$ largest numbers, which all start with 2. We condition on the event that the circuit succeeds in this. It then passes on an S-qubit state (possibly mixed) as the starting state of the particular slice we are considering. This slice then outputs the k largest numbers in x with probability at least $\frac{2}{3}$. Now, consider an algorithm that runs just this slice, starting with the completely mixed state on S-qubits, and that outputs 1 if it finds k numbers starting with 1, and outputs 0 otherwise. If $|x| < k$ this new algorithm always outputs 0 (note that it can verify finding a 1 since its position is appended), but if $|x| = k$ then it outputs 1 with probability at least $\sigma \geq \frac{2}{3} \cdot 2^{-S}$, because the completely mixed state has overlap 2^{-S} with the *good* S-qubit state that would have been the starting state of the slice in the run of the sorting circuit. On the other hand, the slice has only $\alpha\sqrt{SN} < \alpha\sqrt{kN}$ queries, so by choosing α sufficiently small, Theorem 6.3.6 implies $\sigma \leq 2^{-\Omega(k)}$. Combining our upper and lower bounds on σ gives $k = O(S)$. Thus we need $L = \Omega(\frac{N}{S})$ slices, so $T = L\alpha\sqrt{SN} = \Omega(N^{3/2}/\sqrt{S})$. □

As mentioned, our tradeoff is achievable up to polylogarithmic factors [Kla03]. Interestingly, the near-optimal algorithm uses only a polylogarithmic number of qubits and otherwise just classical memory. For simplicity we have shown the lower bound for the case when the outputs have to be made in their natural ordering only, but we can show the same lower bound for any ordering of the outputs that does not depend on the input using a slightly different proof.

8.4 Time-space tradeoffs for matrix products

In this section we use our direct product theorems for OR to get near-optimal time-space tradeoffs for Boolean matrix multiplication.

8.4.1 Construction of a hard matrix

Using the probabilistic method, we construct a hard matrix A, and use it in the proof of the lower bound.

8.4.1. FACT. For every $k = o(N/\log N)$, there exists an $N \times N$ Boolean matrix A, such that all rows of A have weight $\frac{N}{2k}$, and every set of k rows of A contains a set R of $\frac{k}{2}$ rows with the following property: each row in R contains at least $n = \frac{N}{6k}$ ones that occur in no other row of R.

PROOF. We pick A randomly by setting $\frac{N}{2k}$ random positions in each row to 1. We want to show that with positive probability for all sets of k rows $A|_{i_1}, \ldots, A|_{i_k}$ many of the rows $A|_{i_j}$ contain at least $\frac{N}{6k}$ ones that are not ones in any of the $k-1$ other rows.

This probability can be bounded as follows. We will treat the rows as subsets of $\{1, \ldots, N\}$. A row $A|_j$ is called *bad* with respect to $k-1$ other rows $A|_{i_1}, \ldots, A|_{i_{k-1}}$, if $\left|A|_j - \bigcup_\ell A|_{i_\ell}\right| \leq \frac{N}{6k}$. For fixed $i_1, \ldots, i_{k-1}$, the probability that some $A|_j$ is bad with respect to the $k-1$ other rows is at most $e^{-\Omega(N/k)}$ by the Chernoff bound and the fact that k rows can together contain at most $\frac{N}{2}$ elements. Since $k = o(N/\log N)$ we may assume this probability is at most N^{-10}.

Now fix any set $I = \{i_1, \ldots, i_k\}$. The probability that for $j \in I$ it holds that $A|_j$ is bad with respect to the other rows is at most N^{-10}, and this also holds, if we condition on the event that some other rows are bad, since this condition makes it only less probable that another row is also bad. So for any fixed $J \subset I$ of size $\frac{k}{2}$ the probability that all rows in J are bad is at most N^{-5k}, and the probability that there exists such J is at most

$$\binom{k}{k/2} N^{-5k} \ .$$

Furthermore the probability that there is a set I of k rows for which $\frac{k}{2}$ are bad is at most

$$\binom{N}{k}\binom{k}{k/2} N^{-5k} < 1 \ .$$

So there is an A as required and we may fix one. □

8.4.2 Boolean matrix products

First we show a lower bound on the time-space tradeoff for Boolean matrix-vector multiplication on classical machines.

8.4.2. THEOREM. *There is an $N \times N$ matrix A such that every bounded-error classical circuit for computing the Boolean matrix-vector product Ab that uses T queries and space $S = o(N/\log N)$ satisfies $TS = \Omega(N^2)$.*

The bound is tight if T measures queries to the input.

PROOF. Fix $k = O(S)$ large enough. Take a hard matrix A from Fact 8.4.1.

Now suppose we are given a circuit with space S that computes the Boolean product between the rows of A and b in some order. We again proceed by slicing the circuit into $L = \frac{T}{\alpha N}$ slices, each containing $\frac{T}{L} = \alpha N$ queries. Each such slice has a number of output gates. Consider any slice. Suppose it contains output gates $i_1 < \ldots < i_k \leq \frac{N}{2}$, so it is supposed to output $\bigvee_{\ell=1}^{N} (A[i_j, \ell] \wedge b_\ell)$ for all i_j with $1 \leq j \leq k$.

Such a slice starts on a classical value of the *memory* of the circuit, which is in general a probability distribution on S bits (if the circuit is randomized). We replace this probability distribution by the uniform distribution on the possible values of S bits. If the original circuit succeeds in computing the function correctly with probability at least $\frac{1}{2}$, then so does the circuit slice with its outputs, and replacing the initial value of the memory by a uniformly random one decreases the success probability to no less than $\frac{1}{2} \cdot 2^{-S}$.

If we now show that any classical circuit with αN queries that produces the outputs $i_1, \ldots, i_k$ can succeed only with exponentially small probability in k, we get that $k = O(S)$, and hence $\frac{T}{\alpha N} \cdot O(S) \geq N$, which gives the claimed lower bound for the time-space tradeoff.

Each set of k outputs corresponds to k rows of A, which contain $\frac{N}{2k}$ ones each. Thanks to the construction of A there are $\frac{k}{2}$ rows among these, such that $\frac{N}{6k}$ of the ones in each such row are in position where none of the other contains a one. So we get $\frac{k}{2}$ sets of $\frac{N}{6k}$ positions that are unique to each of the $\frac{k}{2}$ rows. The inputs for b will be restricted to contain ones only at these positions, and so the algorithm naturally has to solve $\frac{k}{2}$ independent OR problems on $n = \frac{N}{6k}$ bits each. By Theorem 6.2.4, this is only possible with αN queries if the success probability is exponentially small in k. $\square$

An absolutely analogous construction can be done in the quantum case. Using circuit slices of length $\alpha\sqrt{NS}$ and Theorem 6.3.8 we can prove the following:

8.4.3. THEOREM. *There is an $N \times N$ matrix A such that every bounded-error quantum circuit for computing the Boolean matrix-vector product Ab that uses T queries and space $S = o(N/\log N)$ satisfies $T^2 S = \Omega(N^3)$.*

Note that this is tight within a logarithmic factor (that is needed to improve the success probability of GROVER SEARCH).

8.4.4. THEOREM. *Every bounded-error classical circuit for computing the $N \times N$ Boolean matrix product AB that uses T queries and space S satisfies $TS = \Omega(N^3)$.*

While this is near-optimal for small S, it is probably not tight for large S, a likely tight tradeoff being $T^2 S = \Omega(N^6)$. It is also no improvement compared to Abrahamson's average-case bounds [Abr90].

PROOF. Suppose that $S = o(N)$, otherwise the bound is trivial, since time N^2 is always needed. We can proceed similar to the proof of Theorem 8.4.2. We slice the circuit so that each slice has only αN queries. Suppose a slice makes k outputs. We are going to restrict the inputs to get a direct product problem with k instances of size $\frac{N}{k}$ each, hence a slice with αN queries has exponentially small success probability in k and can thus produce only $O(S)$ outputs. Since the overall number of outputs is N^2 we get the tradeoff $TS = \Omega(N^3)$.

Suppose a circuit slice makes k outputs, where an output labeled (i,j) needs to produce the vector product of the i^{th} row $A|_i$ of A and the j^{th} column $B|^j$ of B. We may partition the set $\{1,\dots,N\}$ into k mutually disjoint subsets $U(i,j)$ of size $\frac{N}{k}$, each associated to an output (i,j).

Assume that there are ℓ outputs $(i,j_1),\dots,(i,j_\ell)$ involving $A|_i$. Each such output is associated to a subset $U(i,j_t)$, and we set $A|_i$ to zero on all positions that are not in any of these subsets, and to one on all positions that are in one of these. When there are ℓ outputs $(i_1,j),\dots,(i_\ell,j)$ involving $B|^j$, we set $B|^j$ to zero on all positions that are not in any of the corresponding subsets, and allow the inputs to be arbitrary on the other positions.

If the circuit computes on these restricted inputs, it actually has to compute k instances of OR of size $n=\frac{N}{k}$ in B, for it is true that $A|_i$ and $B|^j$ contain a single block of size $\frac{N}{k}$ in which $A|_i$ contains only ones, and $B|^j$ *free* input bits, if and only if (i,j) is one of the k outputs. Hence the strong direct product theorem is applicable. □

The application to the quantum case is analogous.

8.4.5. Theorem. *Every bounded-error quantum circuit for computing the $N\times N$ Boolean matrix product AB that uses T queries and space S satisfies $T^2S=\Omega(N^5)$.*

If $S=O(\log N)$, then N^2 applications of Grover search can compute AB with $T=O(N^{2.5}\log N)$. Hence our tradeoff is near-optimal for small S. We do not know whether it is optimal for large S.

8.5 Communication-space tradeoffs

In this section we use the strong direct product theorem for quantum communication from Theorem 6.4.2 to prove communication-space tradeoffs. We later show that these are close to optimal.

8.5.1. Theorem. *Every bounded-error quantum protocol for computing the N-dimensional Boolean matrix-vector product in which Alice and Bob have bounded space S satisfies $C^2S=\Omega(N^3)$.*

Proof. In a protocol, Alice receives a matrix A, and Bob a vector b as inputs. Given a circuit that multiplies these with communication C and space S, we again proceed to slice it. This time, however, a slice contains a limited amount of communication. Recall that in communicating quantum circuits the communication corresponds to wires carrying qubits that cross between Alice's and Bob's circuits. Hence we may cut the circuit after $\alpha\sqrt{NS}$ qubits have been communicated and so on. Overall there are $C/\alpha\sqrt{NS}$ circuit slices. Each starts with an initial state that may be replaced by the completely mixed state at the cost of decreasing the success probability to $\frac{1}{2} \cdot 2^{-S}$. We want to employ the direct product theorem for quantum communication complexity to show that a protocol with the given communication has success probability at most exponentially small in the number of outputs it produces, and so a slice can produce at most $O(S)$ outputs. Combining these bounds with the fact that N outputs have to be produced gives the tradeoff.

To use the direct product theorem we restrict the inputs in the following way. Suppose a protocol makes k outputs. We partition the vector b into k blocks of size $\frac{N}{k}$, and each block is assigned to one of the k rows of A for which an output is made. This row is made to contain zeroes outside of the positions belonging to its block, and hence we arrive at a problem where Disjointness has to be computed on k instances of size $\frac{N}{k}$. With communication $\alpha\sqrt{kN}$, the success probability must be exponentially small in k due to Theorem 6.4.2. Hence $k = O(S)$ is an upper bound on the number of outputs produced. $\Box$

8.5.2. Theorem. *Every bounded-error quantum protocol for computing the N-dimensional Boolean matrix product in which Alice and Bob have bounded space S satisfies $C^2S = \Omega(N^5)$.*

Proof. The proof uses the same slicing approach as in the other tradeoff results. Note that we can assume that $S = o(N)$, since otherwise the bound is trivial. Each slice contains communication $\alpha\sqrt{NS}$, and as before a direct product result showing that k outputs can be computed only with success probability exponentially small in k leads to the conclusion that a slice can only compute $O(S)$ outputs. Therefore $(C/\alpha\sqrt{NS}) \cdot O(S) \geq N^2$, and we are done.

Consider a protocol with $\alpha\sqrt{NS}$ qubits of communication. We partition the universe $\{1, \dots, N\}$ of the Disjointness problems to be computed into k mutually disjoint subsets $U(i,j)$ of size $\frac{N}{k}$, each associated to an output (i,j), which in turn corresponds to a row/column pair $A|_i$, $B|^j$ in the input matrices A and B. Assume that there are ℓ outputs $(i,j_1), \dots, (i,j_\ell)$ involving $A|_i$. Each output is associated to a subset of the universe $U(i,j_t)$, and we set $A|_i$ to zero on all positions that are not in one of these subsets. Then we proceed analogously with the columns of B.

If the protocol computes on these restricted inputs, it has to solve k instances of Disjointness of size $n = \frac{N}{k}$ each, since $A|_i$ and $B|^j$ contain a single block of size $\frac{N}{k}$ in which both are not set to 0 if and only if (i,j) is one of the k outputs. Hence Theorem 6.4.2 is applicable. $\Box$

We now want to show that these tradeoffs are not too far from optimal.

8.5.3. Theorem. *There is a bounded-error quantum protocol for computing the Boolean product between an $N \times N$ matrix and an N-dimensional vector that uses* *space S and communication $C = O((N^{3/2}\log^2 N)/\sqrt{S})$. There is a bounded-error quantum protocol for computing the Boolean product between two $N \times N$ matrices that uses space S and communication $C = O((N^{5/2}\log^2 N)/\sqrt{S})$.*

Proof. We begin by showing a protocol for the following scenario: Alice gets S vectors $x_1, \ldots, x_S$ of N bits each, Bob gets an N-bit vector y, and they want to compute the S Boolean inner products between these vectors. The protocol uses space $O(S)$.

In the following, we interpret Boolean vectors as sets. The main idea is that Alice can compute the union z of the x_i, and then Alice and Bob can find an element in the intersection of z and y using the protocol for the Disjointness problem described in [BCW98]. Alice then marks all x_i that contain this element and removes them from z.

A problem with this approach is that Alice cannot store z explicitly, since it might contain much more than S elements. Alice may, however, store the indices of those sets x_i for which an element in the intersection of x_i and y has already been found, in an array of length S. This array and the input given as an oracle work as an implicit representation of z.

Now suppose at some point during the protocol the intersection of z and y has size k. Then Alice and Bob can find one element in this intersection within $O(\sqrt{N/k})$ rounds of communication in which $O(\log N)$ qubits are exchanged each. Furthermore in $O(\sqrt{Nk})$ rounds all elements in the intersection can be found. So if $k \le S$, then all elements are found within communication $O(\sqrt{NS}\log N)$ and the problem can be solved completely. On the other hand, if $k \ge S$, finding one element costs $O(\sqrt{N/S}\log N)$, but finding such an element removes at least one x_i from z, and hence this has to be done at most S times, giving the same overall communication bound.

It is not hard to see that this process can be implemented with space $O(S)$. The protocol from [BCW98] is a distributed Grover search that itself uses only space $O(\log N)$. Bob can work as in this protocol. For each search, Alice has to start with a superposition over all indices in z. This superposition can be computed from her oracle and her array. In each step she has to apply the Grover iteration. This can also be implemented from these two resources.

To get a protocol for matrix-vector product, the above procedure is repeated $\frac{N}{S}$ times, hence the communication is $O(\frac{N}{S}\sqrt{NS}\log^2 N)$, where one logarithmic factor stems from improving success probability to $\frac{1}{\text{poly}(N)}$.

For the product of two matrices, the matrix-vector protocol may be repeated N times. □

These near-optimal protocols use only $O(\log N)$ qubits, and otherwise just classical memory.

8.6 Time-space tradeoff for linear inequalities

Let A be a fixed $N \times N$ matrix of non-negative integers and let x, b be two input vectors of N non-negative integers smaller or equal to t. A *matrix-vector product with upper bound*, denoted by $y = (Ax)_{\leq b}$, is a vector y such that $y_i = \min((Ax)[i], b_i)$. An *evaluation of a system of linear inequalities* $Ax \geq b$ is the N-bit vector of the truth values of the individual inequalities. Here we present a quantum algorithm for matrix-vector product with upper bound that satisfies time-space tradeoff $T^2 S = O(tN^3(\log N)^5)$. We then use our direct product theorems to show this is close to optimal.

8.6.1 Classical algorithm

It is easy to prove that matrix-vector products with upper bound t can be computed by a classical algorithm with $TS = O(N^2 \log t)$. Let $S' = \frac{S}{\log t}$ and divide the matrix A into $(\frac{N}{S'})^2$ blocks of size $S' \times S'$ each. The output vector is evaluated row-wise as follows:

1. Clear S' counters, one for each row, and read the upper bounds b_i.
2. For each block, read S' input variables, multiply them by the corresponding sub-matrix of A, and update the counters, but do not let them grow larger than b_i.
3. Output the counters.

The space used is $O(S' \log t) = O(S)$ and the total query complexity is $T = O(\frac{N}{S'} \cdot \frac{N}{S'} \cdot S') = O(N^2 (\log t)/S)$.

8.6.2 Quantum algorithm

The quantum algorithm Bounded Matrix Product works in a similar way and it is outlined in Figure 8.1. We compute the matrix product in groups of $S' = \frac{S}{\log N}$ rows, read input variables, and update the counters accordingly. The advantage over the classical algorithm is that we use faster Grover search and Quantum Counting for finding nonzero entries.

The u^{th} row is called *open* if its counter has not yet reached b_u. The subroutine Small Matrix Product maintains a set of open rows $U \subseteq \{1, \dots, S'\}$ and counters $0 \leq y_u \leq b_u$ for all $u \in U$. We process the input x in blocks, each containing between $S' - O(\sqrt{S'})$ and $2S' + O(\sqrt{S'})$ nonzero numbers at the positions j where $A[u, j] \neq 0$ for some $u \in U$. The length ℓ of such a block is first found by Quantum Counting and the nonzero input numbers are then found by Grover search. For each such number, we update all counters y_u and close all rows that have exceeded their threshold b_u.

```
BOUNDED MATRIX PRODUCT (fixed matrix $A_{N\times N}$, threshold $t$, input
vectors $x$ and $b$ of dimension $N$) returns output vector $y = (Ax)_{\leq b}$.

For $i = 1, 2, \ldots, \frac{N}{S'}$, where $S' = \frac{S}{\log N}$:

  1. Run SMALL MATRIX PRODUCT on the $i^{\text{th}}$ block of $S'$ rows of $A$.

  2. Output the $S'$ obtained results for those rows.

--------------------------------------------------------------------------------

SMALL MATRIX PRODUCT (fixed matrix $A_{S'\times N}$, input vectors $x_{N\times 1}$ and
$b_{S'\times 1}$) returns $y_{S'\times 1} = (Ax)_{\leq b}$.

  3. Initialize $y := (0, 0, \ldots, 0)$, $p := 1$, $U := \{1, \ldots, S'\}$, and read $b$.
     Let $a_{1\times N}$ denote an on-line computed $N$-bit row-vector with $a_j = 1$
     if $A[u, j] = 1$ for some $u \in U$, and $a_j = 0$ otherwise.

  4. While $p \leq N$ and $U \neq \emptyset$, do the following:

     (a) Let $\tilde{c}_{p,k}$ denote an estimate of $c_{p,k} = \sum_{j=p}^{p+k-1} a_j x_j$; we es-
         timate it by computing the median of $O(\log N)$ calls to
         QUANTUM COUNTING $((a_p x_p) \ldots (a_{p+k-1} x_{p+k-1}), \sqrt{k})$.
           • Initialize $k = S'$.
           • While $p + k - 1 < N$ and $\tilde{c}_{p,k} < S'$, double $k$.
           • Find by binary search the maximal $\ell \in [\frac{k}{2}, k]$ such that
             $p + \ell - 1 \leq N$ and $\tilde{c}_{p,\ell} \leq 2S'$.
     (b) Use GROVER SEARCH to find the set $J$ of all positions $j \in$
         $[p, p + \ell - 1]$ such that $a_j x_j > 0$.
     (c) For all $j \in J$, read $x_j$, and then do the following for all $u \in U$:
           • Increase $y_u$ by $A[u, j] x_j$.
           • If $y_u \geq b_u$, then set $y_u := b_u$ and remove $u$ from $U$.
     (d) Increase $p$ by $\ell$.

  5. Return $y$.
```

Figure 8.1: Quantum algorithm BOUNDED MATRIX PRODUCT

8.6.1. Theorem. Bounded Matrix Product *has bounded error, space complexity $O(S)$, and query complexity $T = O(N^{3/2}\sqrt{t} \cdot (\log N)^{5/2}/\sqrt{S})$.*

Proof. The proof is a generalization of Theorem 8.5.3. The space complexity of Small Matrix Product is $O(S' \log N) = O(S)$, because it stores a subset $U \subseteq \{1, \ldots, S'\}$, integer vectors y, b of dimension S' with numbers at most $t \le N$, the set J of size $O(S')$ with numbers at most N, and a few counters. Let us estimate its query complexity.

Consider the i^{th} block found by Small Matrix Product; let p_i be its left column, let ℓ_i be its length, and let U_i be the set of open rows at the beginning of processing of this block. The scalar product c_{p_i,ℓ_i} is estimated by Quantum Counting with $\sqrt{\ell_i}$ queries. Finding a proper ℓ_i requires $O(\log \ell_i)$ iterations. Let r_i be the number of rows closed during processing of this block and let s_i be the total number added to the counters for other (still open) rows in this block. The numbers ℓ_i, r_i, s_i are random variables. If we instantiate them at the end of the quantum subroutine, the following inequalities hold:

$$\sum_i \ell_i \le N, \quad \sum_i r_i \le S', \text{ and } \quad \sum_i s_i \le tS' \ .$$

The iterated Grover search finds ones for two purposes: closing rows and increasing counters. Since each $b_i \le t$, the total cost in the i^{th} block is at most $\sum_{j=1}^{r_i t} O(\sqrt{\ell_i/j}) + \sum_{j=1}^{s_i} O(\sqrt{\ell_i/j}) = O(\sqrt{\ell_i r_i t} + \sqrt{\ell_i s_i})$ by Corollary 1.3.5. By the Cauchy-Schwarz inequality, the total number of queries that Small Matrix Product spends in Grover search is at most

$$\begin{aligned}\sum_{i=1}^{\#\text{blocks}} (\sqrt{\ell_i r_i t} + \sqrt{\ell_i s_i}) &\le \sqrt{\sum_i \ell_i}\sqrt{t\sum_i r_i} + \sqrt{\sum_i \ell_i}\sqrt{\sum_i s_i} \\ &\le \sqrt{N}\sqrt{tS'} + \sqrt{N}\sqrt{tS'} = O(\sqrt{NS't}) \ .\end{aligned}$$

The error probability of Grover search can be made polynomially small in a logarithmic overhead. It remains to analyze the outcome and error probability of Quantum Counting. Let $c_i = c_{p_i,\ell_i} \in [S', 2S']$. By Corollary 1.5.5, Quantum Counting with $\sqrt{\ell_i}$ queries gives an estimate $\tilde{c}$ such that

$$|\tilde{c} - c_i| = O\left(\sqrt{\min(c_i, \ell_i - c_i)} + 1\right) = O(\sqrt{c_i} + 1) = O(\sqrt{S'})$$

with probability at least $\frac{8}{\pi^2} \approx 0.8$. We do it $O(\log N)$ times and take the median, hence we obtain an estimate $\tilde{c}$ of c_i with accuracy $O(\sqrt{S'})$ with polynomially small error probability. The result of Quantum Counting is compared with the given threshold, that is with S' or $2S'$. Binary search for $\ell \in [\frac{k}{2}, k]$ costs another factor of $\log k \le \log N$. By the Cauchy-Schwarz inequality, the total number of queries spent in the QUANTUM COUNTING is at most $(\log N)^2$ times

$$\sum_i \sqrt{\ell_i} \le \sqrt{\sum_i \ell_i}\sqrt{\sum_i 1} \le \sqrt{N}\sqrt{\#\text{blocks}} \le \sqrt{N}\sqrt{S'+t} \le \sqrt{NS't} \ ,$$

because in every block the algorithm closes a row or adds $\Theta(S')$ in total to the counters. The number of closed rows is at most S' and the number S' can be added at most t times.

The total query complexity of SMALL MATRIX PRODUCT is thus $O(\sqrt{NS't} \cdot (\log N)^2)$ and the total query complexity of BOUNDED MATRIX PRODUCT is $\frac{N}{S'}$-times bigger. The overall error probability is at most the sum of the individual polynomially small error probabilities of the different subroutines, hence it can be kept below $\frac{1}{3}$. $\square$

8.6.3 Matching quantum lower bound

Here we use our direct product theorems to lower-bound the quantity T^2S for T-query, S-space quantum algorithms for systems of linear inequalities. The lower bound even holds if we fix b to the all-t vector $\vec{t}$ and let A and x be Boolean.

8.6.2. THEOREM. *Let $S \le \min(O(\frac{N}{t}), o(\frac{N}{\log N}))$. There exists an $N \times N$ Boolean matrix A such that every 2-sided error quantum algorithm for evaluating a system $Ax \ge \vec{t}$ of N inequalities that uses T queries and space S satisfies $T^2S = \Omega(tN^3)$.*

PROOF. The proof is a generalization of Theorem 8.4.3.

Using Fact 8.4.1, fix a hard matrix A for $k = cS$, for some constant c to be chosen later. Consider a quantum circuit with T queries and space S that solves the problem with success probability at least $\frac{2}{3}$. We slice the quantum circuit into disjoint consecutive slices, each containing $Q = \alpha\sqrt{tNS}$ queries, where α is the constant from our direct product theorem (Theorem 7.1.1). The total number of slices is $L = \frac{T}{Q}$. Together, these disjoint slices contain all N output gates. Our aim below is to show that with sufficiently small constant α and sufficiently large constant c, no slice can produce more than k outputs. This will imply that the number of slices is $L \ge \frac{N}{k}$, hence

$$T = LQ \ge \frac{\alpha N^{3/2}\sqrt{t}}{c\sqrt{S}} \ .$$

Now consider any slice. It starts with an S-qubit state that is delivered by the previous slice and depends on the input, then it makes Q queries and outputs some ℓ results that are jointly correct with probability at least $\frac{2}{3}$. Suppose, by way of contradiction, that $\ell \ge k$. Then there exists a set of k rows of A such that our slice produces the k corresponding results (t-threshold functions) with probability at least $\frac{2}{3}$. By the above Fact 8.4.1, some set R of $\frac{k}{2}$ of those rows has the following property: each row from R contains a set of $n = \frac{N}{6k} = \Theta(\frac{N}{S})$ ones that do not occur in any of the $\frac{k}{2} - 1$ other rows of R. By setting all other $N - \frac{kn}{2}$ bits of x to 0, we naturally get that our slice, with the appropriate S-qubit starting state, solves $\frac{k}{2}$ independent t-threshold functions $\mathrm{Thr}_{t,n}$ on n bits each. (Note that we need $t \leq \frac{n}{2} = O(\frac{N}{S})$; this follows from our assumption $S = O(\frac{N}{t})$ with appropriately small constant in the $O(\cdot)$.) Now we replace the initial S-qubit state by the completely mixed state, which has overlap 2^{-S} with every S-qubit state. This turns the slice into a stand-alone algorithm solving $\mathrm{Thr}_{t,n}^{(k/2)}$ with success probability

$$\sigma \geq \frac{2}{3} \cdot 2^{-S} \quad .$$

But this algorithm uses only $Q = \alpha\sqrt{tNS} = O(\alpha k\sqrt{tn})$ queries, so our direct product theorem (Theorem 7.1.1) with sufficiently small constant α implies

$$\sigma \leq 2^{-\Omega(k/2)} = 2^{-\Omega(cS/2)} \quad .$$

Choosing c a sufficiently large constant (independent of this specific slice), our upper and lower bounds on σ contradict. Hence the slice must produce fewer than k outputs. □

It is easy to see that the case $S \geq \frac{N}{t}$ (equivalently, $t \geq \frac{N}{S}$) is at least as hard as the $S = \frac{N}{t}$ case, for which we have the lower bound $T^2S = \Omega(tN^3) = \Omega(N^4/S)$, hence $TS = \Omega(N^2)$. But that lower bound matches the *classical* deterministic upper bound up to a logarithmic factor and hence is essentially tight also for quantum. We thus have two different regimes for space: for small space, a quantum computer is faster than a classical one in evaluating solutions to systems of linear inequalities, while for large space it is not.

A similar slicing proof using Theorem 6.3.9 (with each slice of $Q = \alpha\sqrt{NS}$ queries producing at most $\frac{S}{t}$ outputs) gives the following lower bound on time-space tradeoffs for 1-sided error algorithms.

8.6.3. Theorem. *Let $t \leq S \leq \min(O(\frac{N}{t^2}), o(\frac{N}{\log N}))$. There exists an $N \times N$ Boolean matrix A such that every 1-sided error quantum algorithm for evaluating a system $Ax \geq \vec{t}$ of N inequalities that uses T queries and space S satisfies $T^2S = \Omega(t^2N^3)$.*

Note that our lower bound $\Omega(t^2N^3)$ for 1-sided error algorithms is higher by a factor of t than the best upper bound for 2-sided error algorithms. This lower bound is probably not optimal. If $S > \frac{N}{t^2}$, then the essentially optimal classical tradeoff $TS = \Omega(N^2)$ takes over.

8.7 Summary

Using the strong direct product theorems from the previous two chapters, we have derived a series of near-optimal classical and quantum time-space tradeoffs, and quantum communication-space tradeoffs for sorting, Boolean matrix multiplication, and evaluating solutions to systems of linear inequalities. In the latter problem, we have exhibited a phase-transition between classical and quantum regimes, and separated 1-sided and 2-sided error quantum algorithms.

Let us mention some open problems. The first is to determine tight time-space tradeoffs for Boolean matrix product on both classical and quantum computers. Second, regarding communication-space tradeoffs for Boolean matrix-vector and matrix product, we did not prove any classical bounds that were better than our quantum bounds. Klauck [Kla04] proved classical tradeoffs $CS^2 = \Omega(N^3)$ and $CS^2 = \Omega(N^2)$ for Boolean matrix product and matrix-vector product, respectively, by means of a *weak* direct product theorem for Disjointness. A classical *strong* direct product theorem for Disjointness (with communication bound αkn instead of our current $\alpha k\sqrt{n}$) would imply optimal tradeoffs, but we do not know how to prove this at the moment. Finally, it would be interesting to get any lower bounds on time-space or communication-space tradeoffs for decision problems in the quantum case, for example for element distinctness (see Section 1.4.1) or matrix verification (see Chapter 4). We currently cannot show quantum time-space tradeoffs for *any* decision problem.

Bibliography

[AA03] S. Aaronson and A. Ambainis. Quantum search of spatial regions. In *Proc. of 44th IEEE FOCS*, pages 200–209, 2003. 6.1, 6.4.2

[Aar02] S. Aaronson. Quantum lower bound for the collision problem. In *Proc. of 34th ACM STOC*, pages 635–642, 2002. 2.7

[Aar04a] S. Aaronson. Limitations of quantum advice and one-way communication. In *Proc. of 19th IEEE Complexity*, pages 320–332, 2004. 6.1, 6.1

[Aar04b] S. Aaronson. Lower bounds for local search by quantum arguments. In *Proc. of 36th ACM STOC*, pages 465–474, 2004. 2.4

[ABH$^+$02] A. Ambainis, H. Buhrman, P. Høyer, M. Karpinski, and P. Kurur. Quantum matrix verification. Unpublished Manuscript, 2002. (document), 4.1, 4.2, 4.1

[Abr90] K. Abrahamson. A time-space tradeoff for Boolean matrix multiplication. In *Proc. of 31st IEEE FOCS*, pages 412–419, 1990. 8.1, 8.4.2

[ADH97] L. M. Adleman, J. DeMarrais, and M. A. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997. 1.2.1

[Amb02] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. Earlier version in STOC'00. 2.2, 2.3, 2.4, 2.4, 2.4.2, 2.4, 5.1, 5.1, 1, 5.3.6, 7.1

[Amb03] A. Ambainis. Polynomial degree vs. quantum query complexity. In *Proc. of 44th IEEE FOCS*, pages 230–239, 2003. 2.4, 2.4.4, 2.4, 2.5, 2.5.5, 2.7, 2.8, 5.1, 5.1, 5.1, 5.1, 5.1.3, 5.1.5, 5.3.1, 1, 5.3, 5.3.1, 5.5.2, 7.1

[Amb04] A. Ambainis. Quantum walk algorithm for element distinctness. In *Proc. of 45th IEEE FOCS*, pages 22–31, 2004. (document), 1.4.1, 1.8, 1.4.2, 1.4.1, 1.4.3, 4.1, 4.2, 4.4.3, 7.1

[Amb05a] A. Ambainis. A new quantum lower bound method, with an application to strong direct product theorem for quantum search. quant-ph/0508200, 2005. 7.1

[Amb05b] A. Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1:37–46, 2005. 2.7, 2.7

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice-Hall, 1993. 3.2

[AS04] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problem. *Journal of the ACM*, 51(4):595–605, 2004. 2.7, 2.7, 5.4, 7.1

[AŠ06] A. Ambainis and R. Špalek. Quantum algorithms for matching and network flows. In *Proc. of 23rd STACS*, pages 172–183, 2006. LNCS 3884. 3

[AŠW06] A. Ambainis, R. Špalek, and R. de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. In *Proc. of 38th ACM STOC*, pages 618–633, 2006. 6, 7, 8

[Bay72] R. Bayer. Symmetric binary B-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1:290–306, 1972. 1.4.1

[BBBV97] H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. 2.2, 5.1, 5.1

[BBC$^+$95] A. Barenco, C. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995. 1.2.1

[BBC$^+$01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. 1.2.2, 1.3.5, 1.5.2, 2.4, 2.6, 2.6, 2.6.2, 2.6.3, 2.7, 2.7.1, 4.8, 6.1, 6.1, 6.2.6, 6.2.4, 6.3, 6.3.10, 6.3.3, 7.1, 7.2

[BBHT98] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. Earlier version in Physcomp'96. (document), 1.3, 1.3.2, 1.6, 1.3.3, 6.3.2

[BCW98] H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proc. of 30th ACM STOC*, pages 63–68, 1998. 2.4, 6.1, 6.4.2, 8.2, 8.5

[BCWW01] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902, 2001. 1.4.7

[BCWZ99] H. Buhrman, R. Cleve, R. de Wolf, and Ch. Zalka. Bounds for small-error and zero-error quantum algorithms. In *Proc. of 40th IEEE FOCS*, pages 358–368, 1999. 2.7, 6.3.1, 7.1

[BDF$^+$04] A. Berzina, A. Dubrovsky, R. Freivalds, L. Lace, and O. Scegulnaja. Quantum query complexity for some graph problems. In *Proc. of 30th SOFSEM*, pages 140–150, 2004. 3.1

[BDH$^+$01] H. Buhrman, Ch. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. In *Proc. of 16th IEEE Complexity*, pages 131–137, 2001. (document), 1.3.8, 1.7

[Bea91] P. Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM Journal on Computing*, 20(2):270–277, 1991. Earlier version in STOC'89. 8.1

[Bei93] R. Beigel. The polynomial method in circuit complexity. In *Proc. of 8th IEEE Structure in Complexity Theory Conf.*, pages 82–95, 1993. 2.6

[Bel64] J. S. Bell. On the Einstein-Podolsky-Rosen paradox. *Physics*, 1:195–200, 1964. 1.1.1

[Ben89] C. H. Bennett. Time/space tradeoffs for reversible computation. *SIAM Journal of Computing*, 4(18):766–776, 1989. 1.2.1

[BFS86] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proc. of 27th IEEE FOCS*, pages 337–347, 1986. 6.1

[BH97] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem. In *Proc. of 5th Israeli Symp. on Theory of Computing and Systems*, pages 12–23, 1997. 2.7

[BHMT02] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74. 2002. (document), 1.3, 1.3.4, 1.3.6, 1.10, 1.5.2, 1.5.3, 1.5.4, 1.5.6, 2.4, 2.7

[BHT97] G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *SIGACT News*, 28:14–19, 1997. 2.7

[BJKS02a] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *Proc. of 43rd IEEE FOCS*, pages 209–218, 2002. 6.1

[BJKS02b] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. Information theory methods in communication complexity. In *Proc. of 17th IEEE Complexity*, pages 93–102, 2002. 6.1

[BNRW05] H. Buhrman, I. Newman, H. Röhrig, and R. de Wolf. Robust quantum algorithms and polynomials. In *Proc. of 22nd STACS*, pages 593–604, 2005. LNCS 3404. 2.7, 2.7.2, 2.7, 6.1

[Boh13] N. Bohr. On the constitution of atoms and molecules. *Philosophical Magazine*, 26:1–25, 476–502, 857–875, 1913. 1

[BPSW05] P. Beame, T. Pitassi, N. Segerlind, and A. Wigderson. A strong direct product lemma for corruption and the multiparty NOF communication complexity of disjointness. In *Proc. of 20th IEEE Complexity*, pages 52–66, 2005. 6.1, 6.1, 6.4.2

[Bro24] L. de Broglie. *Researches on the quantum theory.* PhD thesis, Paris, 1924. 1

[BS04] H. Barnum and M. Saks. A lower bound on the quantum query complexity of read-once functions. *Journal of Computer and Systems Sciences*, 69(2):244–258, 2004. 2.2, 2.3, 2.4, 2.5, 5.1, 5.1, 5.1.5, 5.1, 5.4

[BŠ06] H. Buhrman and R. Špalek. Quantum verification of matrix products. In *Proc. of 17th ACM-SIAM SODA*, pages 880–889, 2006. 4

[BSS03] H. Barnum, M. Saks, and M. Szegedy. Quantum decision trees and semidefinite programming. In *Proc. of 18th IEEE Complexity*, pages 179–193, 2003. 2.2, 2.3, 2.3, 2.3.1, 2.9, 5.1, 5.1, 5.1, 5.3, 5.3.1, 5.3.1

[BTY94] P. Beame, M. Tompa, and P. Yan. Communication-space tradeoffs for unrestricted protocols. *SIAM Journal on Computing*, 23(3):652–661, 1994. Earlier version in FOCS'90. 8.1

[Buh00] H. Buhrman. Quantum computing and communication complexity. *EATCS Bulletin*, 70:131–141, 2000. 8.2

[BV97] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. Earlier version in STOC'93. 2.2.2, 7.4.1

[BW92] C. H. Bennett and S. J. Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69(20):2881–2884, 1992. 1.2.2

[BW98] H. Buhrman and R. de Wolf. Lower bounds for quantum search and derandomization. quant-ph/9811046, 18 Nov 1998. 2.1

[BW02] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002. 2.1, 2.7

[CDNT98] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp. Quantum entanglement and the communication complexity of the inner product function. In *Proc. of 1st NASA QCQC conference*, pages 61–74. Springer, 1998. LNCS 1509. 1.2.2, 6.1

[Cop94] D. Coppersmith. An approximate Fourier transform useful in quantum factoring. IBM technical report RC19642, quant-ph/0201067, 1994. 1.5

[Cop97] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13:42–49, 1997. 4.6

[CR92] D. Coppersmith and T. J. Rivlin. The growth of polynomials bounded at equally spaced points. *SIAM Journal on Mathematical Analysis*, 23(4):970–983, 1992. 6.1, 6.3.1, 6.3.1

[CSWY01] A. Chakrabarti, Y. Shi, A. Wirth, and A. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proc. of 42nd IEEE FOCS*, pages 270–278, 2001. 6.1

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9:251–280, 1990. Earlier version in STOC'87. 4.1, 4.6

[CW00] R. Cleve and J. Watrous. Fast parallel circuits for the quantum Fourier transform. In *Proc. of 41st IEEE FOCS*, pages 526–536, 2000. 1.5

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle, and the universal quantum Turing machine. *Proceedings of the Royal Society, London*, A400:97–117, 1985. 1

[DHHM04] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quatum query complexity of some graph problems. In *Proc. of 31st ICALP*, pages 481–493, 2004. LNCS 3142. 2.4, 3.1

[Din70] E. A. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady*, 11:1277–1280, 1970. 3.1, 3.6

[DJ92] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society, London*, A439:553–558, 1992. 1.2.2

[Edm65] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. (document), 3.1, 3.5, 3.4

[EHK04] M. Ettinger, P. Høyer, and E. Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004. 1.2.2

[Ein05] A. Einstein. On a heuristic viewpoint concerning the production and transformation of light. *Annalen der Physik*, 17:132–148, 1905. 1

[EK72] J. Edmonds and R. M. Karp. Theoretical improvement in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972. 3.1, 3.6

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 47:777–780, 1935. 1.1.1, 1.1.3

[ET75] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4:507–518, 1975. 3.1, 3.1, 3.6, 3.6.1

[Fey82] R. Feynman. Simulating physics with computers. *Internation Journal of Theoretical Physics*, 21(6/7):467–488, 1982. 1

[Fey85] R. Feynman. Quantum mechanical computers. *Optics News*, 11:11–20, 1985. 1

[FF56] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. 3.1

[FGGS98] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation in determining parity. *Physical Review Letters*, 81:5442–5444, 1998. 2.7.1

[FGGS99] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. Invariant quantum algorithms for insertion into an ordered list. quant-ph/9901059, 1999. 2.1

[FR99] L. Fortnow and J. D. Rogers. Complexity limitations on quantum computation. *Journal of Computer and Systems Sciences*, 59(2):240–252, 1999. 2.6

[Fre79] R. Freivalds. Fast probabilistic algorithms. In *Proc. of 8th Symp. on Math. Foundations of Computer Science*, pages 57–69. Springer Verlag, 1979. LNCS 74. 4.1, 4.2

[Gab76] H. N. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *Journal of the ACM*, 23(2):221–234, 1976. (document), 3.1, 3.5, 3.4

[GKP98] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics.* Addison-Wesley, 1998. 7.6

[GN79] Z. Galil and A. Naamad. Network flow and generalized path compression. In *Proc. of 11th ACM STOC*, pages 13–26, 1979. 3.1

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR lemma. Technical report, ECCC TR–95–050, 1995. Available at http://www.eccc.uni-trier.de/eccc/. 6.1

[GR98] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *Journal of the ACM*, 45(5):783–797, 1998. 3.1, 3.6

[GR99] A. V. Goldberg and S. Rao. Flows in undirected unit capacity networks. *SIAM Journal on Discrete Mathematics*, 12(1):1–5, 1999. 3.1

[Gro96] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. of 28th ACM STOC*, pages 212–219, 1996. (document), 1, 1.2.2, 1.3, 1.3.2, 1.4, 2.2, 2.7, 5.6, 5.6

[Gro02] L. Grover. Tradeoffs in the quantum search algorithm. quant-ph/0201152, 2002. 1.3.1

[Hei27] W. Heisenberg. Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. *Zeitschrift für Physik*, 43:172–198, 1927. English version: J. A. Wheeler and H. Zurek, Quantum Theory and Measurement, Princeton Univ. Press, pages 62–84, 1983. 1

[HH99] L. Hales and S. Hallgren. Quantum Fourier sampling simplified. In *Proc. of 31st ACM STOC*, pages 330–338, 1999. 1.5

[HK73] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973. (document), 3.1, 3.4, 3.4, 3.4.1, 3.2

[HLŠ05] P. Høyer, T. Lee, and R. Špalek. Tight adversary bounds for composite functions. quant-ph/0509067, 2005. 5

[HMW03] P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. In *Proc. of 30th ICALP*, pages 291–299, 2003. LNCS 2719. 2.4, 2.7, 4.7, 5.4, 7.1

[HNS02] P. Høyer, J. Neerbek, and Y. Shi. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica*, 34(4):429–448, 2002. Special issue on Quantum Computation and Cryptography. 2.1, 2.2, 2.3, 2.3.2, 5.1, 5.1, 5.3.6

[Hol73] A. S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problems in Information Transmission*, 9:177–183, 1973. 1.2.2

[HŠ05] P. Høyer and R. Špalek. Lower bounds on quantum query complexity. *EATCS Bulletin*, 87:78–103, October, 2005. 2, 5, 5.1

[HW02] P. Høyer and R. de Wolf. Improved quantum communication complexity bounds for disjointness and equality. In *Proc. of 19th STACS*, pages 299–310. Springer, 2002. LNCS 2285. 6.1

[Ind05] P. Indyk. Output-sensitive algorithm for matrix multiplication. Manuscript, 2005. 4.1, 4.6

[JLB05] M. B. Jacokes, A. J. Landahl, and E. Brookes. An improved quantum algorithm for searching an ordered list. Manuscript, 2005. 2.1

[Kar74] A. V. Karzanov. Determining the maximal flow in a network by the method of preflows. *Soviet Mathematics Doklady*, 15:434–437, 1974. 3.1

[KL98] D. R. Karger and M. S. Levine. Finding maximum flows in undirected graphs seems easier than bipartite matching. In *Proc. of 30th ACM STOC*, pages 69–78, 1998. 3.1

[Kla00] H. Klauck. Quantum communication complexity. In *Proc. of Workshop on Boolean Functions and Applications at 27th ICALP*, pages 241–252, 2000. 8.2

[Kla03] H. Klauck. Quantum time-space tradeoffs for sorting. In *Proc. of 35th ACM STOC*, pages 69–76, 2003. 8.1, 8.3

[Kla04] H. Klauck. Quantum and classical communication-space tradeoffs from rectangle bounds. In *Proc. of 24th FSTTCS*, pages 384–395, 2004. 8.7

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997. 8.2

[KNP05] P. Koiran, V. Nesme, and N. Portier. A quantum lower bound for the query complexity of Simon's problem. In *Proc. of 32nd ICALP*, pages 1287–1298, 2005. LNCS 3580. 2.7

[Knu03] D. E. Knuth. Combinatorial matrices. In *Selected Papers on Discrete Mathematics*, volume 106 of *CSLI Lecture Notes*. Stanford University, 2003. 1.4.8, 6.4.1

[Kre95] I. Kremer. Quantum communication. Master's thesis, Hebrew University, Computer Science Department, 1995. 6.4.1

[KS92] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992. Earlier version in Structures'87. 6.1

[KŠW04] H. Klauck, R. Špalek, and R. de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. In *Proc. of 45th IEEE FOCS*, pages 12–21, 2004. 6, 8

[Kut05] S. Kutin. Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1:29–36, 2005. 2.7

[LLS06] S. Laplante, T. Lee, and M. Szegedy. The quantum adversary method and classical formula size lower bounds. *Computational Complexity*, 15:163–196, 2006. Earlier version in Complexity'05. 2.5, 2.5, 2.5.5, 2.7, 5.1, 5.1, 5.1.3, 5.1.5, 5.5.3, 5.5.8

[LM04] S. Laplante and F. Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. In *Proc. of 19th IEEE Complexity*, pages 294–304, 2004. 2.5.3, 5.1, 5.1, 5.1, 5.1.1, 5.1, 5.1, 5.2, 1, 5.3.1, 5.3.1, 2, 5.3, 5.3.3, 5.3.10, 5.3.3, 5.4

[Lov00] L. Lovász. Semidefinite programs and combinatorial optimization. `http://research.microsoft.com/users/lovasz/semidef.ps`, 2000. 5.2, 5.3.2, 5.3.2

[LŠ05] S. Laplante and R. Špalek. Adversary type circuit depth lower bounds. Unpublished manuscript, 2005. 5.1

[LTT92] T.W. Lam, P. Tiwari, and M. Tompa. Trade-offs between communication and space. *Journal of Computer and Systems Sciences*, 45(3):296–315, 1992. Earlier version in STOC'89. 8.1

[LV97] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer, Berlin, second edition, 1997. 5.2, 5.2, 5.3.3

[Mat90] R. Mathias. The spectral norm of a nonnegative matrix. *Linear Algebra and its Applications*, 139:269–284, 1990. 2.4.1, 5.3.2

[MKM78] V. M. Malhotra, P. Kumar, and S. N. Maheshwari. An $O(V^3)$ algorithm for finding the maximum flows in networks. *Information Processing Letters*, 7(6):277–278, 1978. 3.1

[MN05] F. Magniez and A. Nayak. Quantum complexity of testing group commutativity. In *Proc. of 32nd ICALP*, pages 1312–1324, 2005. LNCS 3580. 4.4.3

[MS04] M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In *Proc. of 45th IEEE FOCS*, pages 248–255, 2004. 3.1

[MSS05] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. In *Proc. of 16th ACM-SIAM SODA*, pages 1109–1117, 2005. 1.4.2, 2.8, 4.4.3, 5.4, 7.1

[MV80] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proc. of 21st IEEE FOCS*, pages 17–27, 1980. 3.1, 3.1, 3.5

[NC00] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000. 1

[Nis91] N. Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. Earlier version in STOC'89. 6.2.6

[NRS94] N. Nisan, S. Rudich, and M. Saks. Products and help bits in decision trees. In *Proc. of 35th IEEE FOCS*, pages 318–329, 1994. 6.1

[NS94] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994. Earlier version in STOC'92. 2.6, 2.7, 6.1, 6.2.4

[NW99] A. Nayak and F. Wu. The quantum query complexity of approximating the median and related statistics. In *Proc. of 31st ACM STOC*, pages 384–393, 1999. 2.4

[Pat92] R. Paturi. On the degree of polynomials that approximate symmetric Boolean functions. In *Proc. of 24th ACM STOC*, pages 468–474, 1992. 2.7, 6.3.1, 6.3.1, 6.3.3, 7.2

[Pla01] M. Planck. On the law of distribution of energy in the normal spectrum. *Annalen der Physik*, 4:553, 1901. 1

[PRW97] I. Parnafes, R. Raz, and A. Wigderson. Direct product results and the GCD problem, in old and new communication models. In *Proc. of 29th ACM STOC*, pages 363–372, 1997. 6.1

[Raz87] A. A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\&, \oplus\}$. *Math. Notes Acad. Sci. USSR*, 41(4):333–338, 1987. 1.2.1

[Raz92] A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992. 6.1

[Raz03] A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya of the Russian Academy of Science, mathematics*, 67(1):159–176, 2003. English version in quant-ph/0204025. 6.1, 6.1, 6.4, 6.4.1, 8.1

[Reg97] K. Regan. Polynomials and combinatorial definitions of languages. In *Complexity Theory Retrospective II*, pages 261–293. Springer-Verlag, 1997. 2.6

[Riv90] T. J. Rivlin. *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory.* Wiley-Interscience, second edition, 1990. 6.3.1

[San95] M. Santha. On the Monte Carlo decision tree complexity of read-once formulae. *Random Structures and Algorithms*, 6(1):75–87, 1995. 2.8

[Sha01] R. Shaltiel. Towards proving strong direct product theorems. In *Proc. of 16th IEEE Complexity*, pages 107–119, 2001. 6.1, 6.2.1, 6.2.2, 6.5

[Shi02] Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. In *Proc. of 43rd IEEE FOCS*, pages 513–519, 2002. 2.7

[Sho97] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. Earlier version in FOCS'94. 1, 1.5, 5.6, 5.6

[Sim97] D. R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. 2.7

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. of 19th ACM STOC*, pages 77–82, 1987. 1.2.1

[Sni85] M. Snir. Lower bounds on probabilistic decision trees. *Theoretical Computer Science*, 38:69–82, 1985. 2.8

[SS04] M. Santha and M. Szegedy. Quantum and classical query complexities of local search are polynomially related. In *Proc. of 36th ACM STOC*, pages 494–501, 2004. 2.4

[ŠS06] R. Špalek and M. Szegedy. All quantum adversary methods are equivalent. *Theory of Computing*, 2(1):1–18, 2006. Earlier version in ICALP'05. 2.5.3, 5, 5.1.1, 5.1.5

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969. 4.1

[SW86] M. Saks and A. Wigderson. Probabilistic Boolean decision trees and the complexity of evaluating games trees. In *Proc. of 27th IEEE FOCS*, pages 29–38, 1986. 2.8

[Sze03] M. Szegedy. On the quantum query complexity of detecting triangles in graphs. quant-ph/0310107, 2003. 5.1, 5.4

[Sze04] M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proc. of 45th IEEE FOCS*, pages 32–41, 2004. (document), 1.4.2, 1.9, 1.4.6, 1.4.2, 1.4.2, 4.1, 4.4.3, 4.6.2

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986. 1.4.1, 4.4.3

[Wol02] R. de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–353, 2002. 2.9, 8.2

[Yao77] A. C-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. of 18th IEEE FOCS*, pages 222–227, 1977. 6.2, 1

[Yao82] A. C-C. Yao. Theory and applications of trapdoor functions. In *Proc. of 23rd IEEE FOCS*, pages 80–91, 1982. 6.1

[Yao93] A. C-C. Yao. Quantum circuit complexity. In *Proc. of 34th IEEE FOCS*, pages 352–360, 1993. 6.4.1, 8.2

[YZ04] R. Yuster and U. Zwick. Fast sparse matrix multiplication. In *Proc. of 12th RSA*, pages 604–615, 2004. 4.6

[Zal99] Ch. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60:2746–2751, 1999. 2.5

[Zha05] S. Zhang. On the power of Ambainis's lower bounds. *Theoretical Computer Science*, 339(2–3):241–256, 2005. Earlier version in ICALP'04. 2.4, 2.4.4, 2.5.3, 3.1, 5.1, 5.1, 5.1, 5.1, 5.3.1, 5.3.1, 5.4

[Zha06] S. Zhang. New upper and lower bounds for randomized and quantum local search. In *Proc. of 38th ACM STOC*, pages 634–643, 2006. 2.4

List of symbols

$A_{n\times m}$	matrix of dimensions $n \times m$
$A\|_R, A\|^S, A\|_R^S$	sub-matrix indexed by rows R (resp. columns C, resp. both), Definition 4.3.1
$A \geq 0$	A has non-negative entries
$A \succeq 0$	A is positive semidefinite, Definition 5.2.3
AB	standard matrix product
$A \cdot B$	scalar product of two matrices as vectors
$A \circ B$	entry-wise product of two matrices
$\mathrm{Adv}(f)$	adversary bound for f, Definition 2.5.1
$\mathrm{Adv}_\alpha(f)$	adversary bound for f with costs α, Definition 5.1.4
$bs(f)$	block sensitivity of f, Definition 2.5.4
$\mathrm{C}_z(f), \mathrm{C}(f)$	z-certificate complexity of f (resp. total), Definition 2.5.2
$c_x(M), c(M)$	ℓ_2-norm of the x^{th} column (maximal column), Section 5.3.1
$\deg(f), \widetilde{\deg}(f)$	degree and approximate degree of f, Definition 2.6.1
D_i	zero-one matrix that is 1 iff $x_i \neq y_i$
$\delta_G, \delta(G)$	spectral gap of graph G, Definition 1.4.4
Disj_n	disjointness communication problem on n bits, above Proposition 6.1.7
e_i	n-bit string with exactly one 1 at position i
F	zero-one matrix that is 1 iff $f(x) \neq f(y)$
$f \circ g$	composition of functions f and g
$f^{(k)}$	k independent instances of f
$\Gamma, \Gamma^f, \Gamma_f$	adversary matrix (for function f), equation (2.1)
Γ_i	adversary matrix on entries that differ in the i^{th} bit, equation (2.2)
$\mathcal{H}$	Hilbert space
H	Hadamard operator, equation (1.4)
I	identity
$J(n,k)$	Johnson graph on n vertices, Definition 1.4.1
$K(x\|y)$	prefix-free Kolmogorov complexity of x given y, Definition 5.2.2
$\lambda(M)$	spectral norm of M, that is the largest eigenvalue of M, Lemma 2.4.1

$[n]$	$\{1,2,\ldots,n\}$
$\mathbb{N}$	natural numbers
$O(f(x))$	at most constant $\cdot f(x)$
$\Omega(f(x))$	at least constant $\cdot f(x)$
$\mathsf{O}_x, \mathsf{O}'_x$	oracle operators on input x, equations (1.8) and (1.7)
OR_n	OR function on n bits
Π_S	projector into subspace S
$Q_E(f)$	exact quantum query complexity of f
$Q_\varepsilon(f), Q_2(f)$	ε-error quantum query complexity of f (resp. $\frac{1}{3}$-error), Definition 1.2.1
$\mathbb{R}$	real numbers
$R_2(f)$	bounded-error classical query complexity of f, above Proposition 6.1.3
$r_x(M), r(M)$	ℓ_2-norm of the x^{th} row (maximal row), Section 5.3.1
$\overline{S}$	set complement of S
$\Theta(f(x))$	roughly constant $\cdot f(x)$
$\mathrm{Thr}_t, \mathrm{Thr}_{t,n}$	t-threshold function (on n bits), Corollary 2.4.3
$\mathrm{Tr}\, A$	trace of A
U_i	unitary operator
$\|x\|$	Hamming weight of x, that is the number of ones
$[x, y]$	closed real interval of numbers between x and y
$\mathsf{X}, \mathsf{Y}, \mathsf{Z}$	Pauli operators, equation (1.3)

Samenvatting

Computers zijn fysieke objecten en dus onderhevig aan de wetten van de natuur. Hoewel de meeste computers tegenwoordig zijn opgebouwd uit halfgeleiders die onderhevig zijn aan quantummechanische effecten, zijn hun berekeningen volledig klassiek—op elk moment is de computer in een specifieke klassieke toestand en de stappen van de berekening zijn volledig deterministisch. *Quantum computers* daarentegen zijn computers die juist quantumeffecten proberen te gebruiken voor hun berekeningen. In tegenstelling tot klassieke computers kan een quantum computer op elk moment in een *superpositie* van verschillende klassieke toestanden tegelijk zijn, en verschillende berekeningen in parallel uitvoeren.

In 1994 vond Peter Shor [Sho97] een efficiënt quantum algoritme voor het factoriseren van grote getallen, een probleem waarvoor geen efficiënt klassiek algoritme bekend is. In 1996 vond Lov Grover [Gro96] een ander belangrijk quantum algoritme, dat een ongeordende database kwadratisch sneller kan doorzoeken dan het best mogelijke klassieke algoritme. Daarna is het vakgebied van de quantum computing sterk gegroeid. Andere belangrijke ontdekkingen zijn ook gedaan in aanverwante gebieden zoals quantum cryptografie, informatietheorie, error-correcting codes, en communicatie complexiteit.

In dit proefschrift onderzoeken we de kracht van quantum computers. We presenteren enkele nieuwe quantum algoritmes, verbeteren technieken om ondergrenzen te bewijzen, en bewijzen de eerste time-space tradeoffs voor quantum computers.

Deel I: Algoritmes

Het quantum zoekalgoritme van Grover is breed toepasbaar en kan als subroutine worden gebruikt in vele klassieke algoritmes. Wij passen het toe op de volgende graafproblemen: het vinden van een maximale *matching* en het vinden van een maximale *stroom* (flow) in een integer netwerk. In beide gevallen is ons quantum algoritme polynomiaal sneller dan het beste bekende klassieke algoritme.

We houden ons ook bezig met de vraag of het product van twee gegeven $n \times n$ matrices gelijk is aan een derde gegeven matrix. Dit is een makkelijker probleem dan het berekenen van het product van de twee matrices zelf, aangezien er een klassiek algoritme bestaat voor het verificatie probleem in tijd $O(n^2)$, terwijl het beste bekende algoritme voor het vermenigvuldigen van twee matrices $O(n^{2.376})$ tijd gebruikt. Ons quantum algoritme voor de verificatie loopt in tijd $O(n^{5/3})$ en is dus polynomiaal sneller dan het beste klassieke algoritme.

Deel II: Ondergrenzen

Een *ondergrens* is een bewijs dat een bepaald probleem niet sneller kan worden opgelost dan een bepaalde waarde. Er zijn twee globale technieken om ondergrenzen te bewijzen voor quantum computers. De *adversary methode* is gebaseerd op de complexiteit van het onderscheiden van inputs die verschillende outputs moeten geven. Deze methode is redelijk eenvoudig te gebruiken en geeft vaak optimale ondergrenzen, bijvoorbeeld voor Grovers zoekprobleem. De *polynomiale methode* drukt de kans dat een quantum algoritme een bepaalde output geeft uit als een polynoom van lage graad; ondergrenzen op de graad van bepaalde approximerende polynomen impliceren dan ondergrenzen voor quantum algoritmes. Deze methode is over het algemeen moeilijker toepasbaar dan de eerste, maar geeft voor sommige problemen sterkere ondergrenzen.

Vele varianten van de adversary methode waren bekend. Wij ruimen dit woud aan varianten op, en bewijzen dat er in feite één adversary methode is, en dat alle bestaande varianten daarvan slechts verschillende equivalente formuleringen zijn. We bewijzen ook een beperking op de ondergrens die met de adversary methode bewezen kan worden voor een functie, in termen van de certificaat-complexiteit van die functie. Dit laat zien dat de meeste van de bekende quantum ondergrenzen die niet optimaal zijn, niet verder verbeterd kunnen worden met de adversary methode. We geven ook precieze formules voor de adversary ondergrens voor functies die de compositie zijn van andere functies.

We bewijzen optimale *direct product stellingen* voor verschillende functies. Een DPS zegt dat de complexiteit van het berekenen van k onafhankelijke instanties van een probleem k maal zo hoog is als de complexiteit van één instantie, zelfs als de succeskans van het algoritme voor de k instanties slechts exponentieel klein hoeft te zijn. Stellingen van dit type klinken meestal wel plausibel, maar zijn vaak moeilijk te bewijzen.

Naast het beantwoorden van een fundamentele vraag over complexiteit, hebben DPS ook toepassingen voor time-space tradeoffs. Een *time-space tradeoff* geeft de relatie die er bestaat tussen de tijd en de geheugenruimte die nodig zijn om een probleem op te lossen. We gebruiken de polynomiale methode om om een DPS te bewijzen voor de OR functie, en passen dit toe om een optimale time-space tradeoff te bewijzen voor het probleem van het sorteren van n getallen ($T^2S = \Theta(n^3)$ voor quantum computers, versus $TS = \Theta(n^2)$ voor klassieke computers) en voor het probleem van het vermenigvuldigen van twee Booleaanse matrices. We bewijzen ook enkele communicatie-geheugenruimte tradeoffs. Dit zijn de eerste tradeoffs die ontdekt zijn voor quantum computers. Verder ontwikkelen we een nieuwe generalisering van de adversary methode in termen van de analyse van de deelruimtes van density matrices, en gebruiken dit om een nieuwe DPS te bewijzen voor alle *symmetrische* functies (dit zijn functies die alleen afhangen van het aantal enen in de input). Deze DPS impliceert een optimale time-space tradeoff voor het evalueren van een gegeven oplossing voor een stelsel van lineaire vergelijkingen, die polynomiaal beter is dan de klassieke tradeoff in het geval er weinig geheugenruimte is.

Abstract

Computers are physical objects and thus obey the laws of physics. Although most computers nowadays are built of semiconductors governed by quantum effects, their computation is purely classical—at every instant the computer is in a classical state and the computational steps are fully deterministic. *Quantum computers*, on the other hand, are computers that exploit quantum effects to do computation. Unlike classical computers, they can be at any moment in a *superposition* of many classical states and perform several computations is parallel.

In 1994, Peter Shor [Sho97] discovered a ground-breaking polynomial time quantum algorithm for factoring integers, a task for which no efficient classical algorithm is known. In 1996, Lov Grover [Gro96] discovered another important quantum algorithm that searches an unordered database quadratically faster than the best classical algorithm. Since then, the field of quantum computing has significantly matured. Other important discoveries have also been made in related fields, such as quantum cryptography, information theory, error-correction codes, and communication complexity.

In this thesis, we investigate the power of quantum computers. We present a few new quantum algorithms, improve some quantum lower-bound techniques, and prove the first known quantum time-space tradeoffs.

Part I: Algorithms

The quantum search algorithm by Grover is very versatile and can be used as a subroutine in many classical algorithms. We apply it on the following graph problems: finding a maximal *matching* and finding a maximal *flow* in an integer network. In both cases we obtain a quantum algorithm that is polynomially faster than the best known classical algorithm.

We address the question of verifying whether a product of two $n \times n$ matrices is equal to a third one. This is easier than computing the actual matrix product, as there is a classical algorithm that runs in time $O(n^2)$, whereas the best known matrix multiplication algorithm runs in time $O(n^{2.376})$. Our quantum algorithm for matrix verification is polynomially faster and it runs in time $O(n^{5/3})$.

Part II: Lower Bounds

A *lower bound* is a proof that some task cannot be computed faster than some value. There are two basic techniques for proving quantum lower bounds. The *adversary method* is based on the complexity of distinguishing inputs that lead to different outcomes. It is easy to use and often gives tight bounds, for example for the unordered search problem. The *polynomial method* expresses the acceptance probability of a quantum algorithm as a low-degree polynomial, and then bounds on the degree of approximate polynomials imply lower bounds on quantum computation. It is generally harder to apply, but gives stronger bounds for some problems.

Many variants of the adversary method were known. We clean up the forest and prove that there is just one method and that all known variants are just different formulations. We prove a limitation on the adversary bound in terms of the certificate complexity of the function, which implies that most of the best known lower bounds that are not known to be tight cannot be further improved by this method. We give tight formulas for adversary bounds of composite functions.

We prove tight *direct product theorems* for several functions. A DPT states that the complexity of computing k independent instances of a problem is k times bigger than the complexity of computing one instance, even if we are willing to only succeed with exponentially small probability. A statement of this type sounds plausible, however it is very hard to prove in general.

Besides answering a fundamental question about computation, DPT's have applications to time-space tradeoffs. A *time-space tradeoff* is a relation between the running time and space complexity of an algorithm. We use the polynomial method to obtain a DPT for the OR function, and apply it to get tight quantum time-space tradeoffs for sorting (in particular $T^2S = \Theta(n^3)$, whereas $TS = \Theta(n^2)$ classically) and Boolean matrix multiplication, and also several communication-space tradeoffs. These are the first such tradeoffs known for quantum computation. We develop a new generalization of the adversary method in terms of analysis of subspaces of density matrices, and use it to get a DPT for all *symmetric* functions (functions that only depend on the number of ones in the input). This DPT implies a time-space tradeoff for the evaluation of solutions of systems of linear inequalities, which is tight and polynomially better than classical when the space is small.