

# The Need for Structure in Quantum Speedups

Scott Aaronson\*  
MIT

Andris Ambainis†  
University of Latvia

## Abstract

Is there a general theorem that tells us when we can hope for exponential speedups from quantum algorithms, and when we cannot? In this paper, we make two advances toward such a theorem, in the black-box model where most quantum algorithms operate.

First, we show that for *any* problem that is invariant under permuting inputs and outputs (like the collision or the element distinctness problems), the quantum query complexity is at least the 9<sup>th</sup> root of the classical randomized query complexity. This resolves a conjecture of Watrous from 2002.

Second, inspired by recent work of O’Donnell et al. and Dinur et al., we conjecture that every bounded low-degree polynomial has a “highly influential” variable. Assuming this conjecture, we show that every  $T$ -query quantum algorithm can be simulated on *most* inputs by a poly( $T$ )-query classical algorithm, and that one essentially cannot hope to prove  $P \neq BQP$  relative to a random oracle.

## 1 Introduction

Perhaps the central lesson gleaned from fifteen years of quantum algorithms research is this:

*Quantum computers can offer superpolynomial speedups over classical computers, but only for certain “structured” problems.*

The key question, of course, is what we mean by “structured.” In the context of most existing quantum algorithms, “structured” basically means that we are trying to determine some global property of an extremely long sequence of numbers, *assuming* that the sequence satisfies some global regularity. As a canonical example, consider PERIOD-FINDING, the core of Shor’s algorithms for factoring and discrete logarithm [22]. Here we are given black-box access to an exponentially-long sequence of integers  $X = (x_1, \dots, x_N)$ ; that is, we can compute  $x_i$  for a given  $i$ . We are asked to find the *period* of  $X$ —that is, the smallest  $k > 0$  such that  $x_i = x_{i-k}$  for all  $i > k$ —promised that  $X$  is indeed periodic, with period  $k \ll N$ . The requirement of periodicity is crucial here: it is what lets us use the Quantum Fourier Transform to extract the information we want from a superposition of the form

$$\frac{1}{\sqrt{N}} \sum_{i=1}^N |i\rangle |x_i\rangle.$$

---

\*MIT. Email: aaronson@csail.mit.edu. This material is based upon work supported by the National Science Foundation under Grant No. 0844626. Also supported by a DARPA YFA grant and the Keck Foundation.

†Email: ambainis@lu.lv. Supported by University of Latvia Research Grant ZB01-100 and FP7 Marie Curie International Reintegration Grant (PIRG02-GA-2007-224886).

For other known quantum algorithms,  $X$  needs to be (for example) a cyclic shift of quadratic residues [12], or constant on the cosets of a hidden subgroup.

By contrast, the canonical example of an “unstructured” problem is the Grover search problem. Here we are given black-box access to an  $N$ -bit string  $x_1, \dots, x_N \in \{0, 1\}^N$ , and are asked whether there exists an  $i$  such that  $x_i = 1$ .<sup>1</sup> Grover [15] gave a quantum algorithm to solve this problem using  $O(\sqrt{N})$  queries [15], as compared to the  $\Omega(N)$  needed classically. However, this quadratic speedup is optimal, as shown by Bennett, Bernstein, Brassard, and Vazirani [7]. For other “unstructured” problems—such as computing the PARITY or MAJORITY of an  $N$ -bit string—quantum computers offer no asymptotic speedup at all over classical computers (see Beals et al. [6]).

Unfortunately, this “need for structure” has essentially limited the prospects for superpolynomial quantum speedups to those areas of mathematics that are liable to produce things like periodic sequences or sequences of quadratic residues.<sup>2</sup> *This is the fundamental reason why the greatest successes of quantum algorithms research have been in cryptography, and specifically in number-theoretic cryptography.* It helps to explain why we do not have a fast quantum algorithm to solve NP-complete problems (for example), or to break arbitrary one-way functions.

Given this history, the following problem takes on considerable importance:

**Problem 1 (Informal)** *For every “unstructured” problem  $f$ , are the quantum query complexity  $Q(f)$  and the classical randomized query complexity  $R(f)$  polynomially related?*

Despite its apparent vagueness, Problem 1 can be formalized in several natural and convincing ways—and under these formalizations, the problem has remained open for about a decade.

## 1.1 Formalizing the Problem

Let  $S \subseteq [M]^N$  be a collection of inputs, and let  $f : S \rightarrow \{0, 1\}$  be a function that we are trying to compute. In this paper, we assume for simplicity that the range of  $f$  is  $\{0, 1\}$ ; in other words, that we are trying to solve a decision problem. It will also be convenient to think of  $f$  as a function from  $[M]^N$  to  $\{0, 1, *\}$ , where  $*$  means ‘undefined’ (that is, that a given input  $X \in [M]^N$  is not in  $f$ ’s domain  $S$ ).

We will work in the well-studied *decision-tree model*. In this model, given an input  $X = (x_1, \dots, x_N)$ , an algorithm can at any time choose an  $i$  and receive  $x_i$ . We count only the number of queries the algorithm makes to the  $x_i$ ’s, ignoring other computational steps. Then the *deterministic query complexity* of  $f$ , or  $D(f)$ , is the number of queries made by an optimal deterministic algorithm on a worst-case input  $X \in S$ . The (bounded-error) *randomized query complexity*  $R(f)$  is the expected number of queries made by an optimal randomized algorithm that, for every  $X \in S$ , computes  $f(X)$  with probability at least  $2/3$ . The (bounded-error) *quantum query complexity*  $Q(f)$  is the same as  $R(f)$ , except that we allow quantum algorithms. Clearly  $Q(f) \leq R(f) \leq D(f) \leq N$  for all  $f$ . See Buhrman and de Wolf [11] for detailed definitions as well as a survey of these measures.

If  $S = [M]^N$ , then we say  $f$  is *total*, while if  $M = 2$ , then we say  $f$  is *Boolean*. The case of total  $f$  is relatively well-understood. Already in 1998, Beals et al. [6] showed the following:

**Theorem 2 (Beals et al. [6])**  $D(f) = O(Q(f)^6)$  for all total Boolean functions  $f : \{0, 1\}^N \rightarrow \{0, 1\}$ .

<sup>1</sup>A variant asks us to *find* an  $i$  such that  $x_i = 1$ , under the mild promise that such an  $i$  exists.

<sup>2</sup>Here we exclude BQP-complete problems, such as simulating quantum physics (the “original” application of quantum computers), approximating the Jones polynomial [4], and estimating a linear functional of the solution of a well-conditioned linear system [16].

Furthermore, it is easy to generalize Theorem 2 to show that  $D(f) = O(Q(f)^6)$  for *all* total functions  $f : [M]^N \rightarrow \{0, 1\}$ , not necessarily Boolean.<sup>3</sup> In other words, for total functions, the quantum query complexity is always at least the 6<sup>th</sup> root of the classical query complexity. The largest known gap between  $D(f)$  and  $Q(f)$  for a total function is quadratic, and is achieved by the OR function (because of Grover’s algorithm).

On the other hand, as soon as we allow non-total functions, we can get enormous gaps. Aaronson [2] recently gave a Boolean function  $f : S \rightarrow \{0, 1\}$  for which  $R(f) = \Omega(\sqrt{N})$ , yet  $Q(f) = O(1)$ . Other examples, for which  $R(f) = \Omega(\sqrt{N})$  and  $Q(f) = O(\log N \log \log N)$ , follow easily from Simon’s algorithm [23] and Shor’s algorithm [22]. Intuitively, these functions  $f$  achieve such large separations by being highly structured: that is, their domain  $S$  includes only inputs that satisfy a stringent promise, such as encoding a periodic function, or (in the case of [2]) encoding two Boolean functions, one of which is correlated with the Fourier transform of the other one.

By contrast with these highly-structured problems, consider the *collision problem*: that of deciding whether a sequence of numbers  $x_1, \dots, x_N \in [M]^N$  is one-to-one (each number appears once) or two-to-one (each number appears twice). Let  $\text{COL}(X) = 1$  if  $X$  is one-to-one and  $\text{COL}(X) = 2$  if  $X$  is two-to-one, promised that one of these is the case. Then  $\text{COL}(X)$  is not a total function, since its definition involves a promise on  $X$ . Intuitively, however, the collision problem seems much less “structured” than Simon’s and Shor’s problems. One way to formalize this intuition is as follows. Call a partial function  $f : [M]^N \rightarrow \{0, 1, *\}$  *permutation-invariant* if

$$f(x_1, \dots, x_N) = f(\tau(x_{\sigma(1)}), \dots, \tau(x_{\sigma(N)}))$$

for all inputs  $X \in [M]^N$  and all permutations  $\sigma \in S_N$  and  $\tau \in S_M$ . Then  $\text{COL}(X)$  is permutation-invariant: we can permute a one-to-one sequence and relabel its elements however we like, but it is still a one-to-one sequence, and likewise for a two-to-one sequence. Because of this symmetry, attempts to solve the collision problem using (for example) the Quantum Fourier Transform seem unlikely to succeed. And indeed, in 2002 Aaronson [1] proved that  $Q(\text{COL}) = \Omega(N^{1/5})$ : that is, the quantum query complexity of the collision problem is at most polynomially better than its randomized query complexity of  $\Theta(\sqrt{N})$ . The quantum lower bound was later improved to  $\Omega(N^{1/3})$  by Aaronson and Shi [3], matching an upper bound of Brassard, Høyer, and Tapp [10].

Generalizing boldly from this example, John Watrous (personal communication) conjectured that the randomized and quantum query complexities are polynomially related for *every* permutation-invariant problem:

**Conjecture 3 (Watrous 2002)**  $R(f) \leq Q(f)^{O(1)}$  for every partial function  $f : [M]^N \rightarrow \{0, 1, *\}$  that is permutation-invariant.

Let us make two remarks about Conjecture 3. First, the conjecture talks about *randomized* versus quantum query complexity, since in this setting, it is easy to find functions  $f$  for which  $R(f)$  and  $Q(f)$  are both tiny but  $D(f)$  is huge. As an example, consider the *Deutsch-Jozsa problem*: given a Boolean input  $(x_1, \dots, x_N)$ , decide whether the  $x_i$ ’s are all equal or whether half of them are 1 and the other half are 0, promised that one of these is the case.

---

<sup>3</sup>For Theorem 2 is proved by combining three ingredients:  $D(f) = O(C(f) \text{bs}(f))$ ,  $C(f) = O(\text{bs}(f)^2)$ , and  $\text{bs}(f) = O(Q(f)^2)$  (where  $C(f)$  is the *certificate complexity* of  $f$  and  $\text{bs}(f)$  is the *block sensitivity*). And all three ingredients go through with no essential change if we set  $M > 2$ , and define suitable  $M$ -ary generalizations of  $C(f)$  and  $\text{bs}(f)$ . (We could also convert the non-Boolean function  $f : [M]^N \rightarrow \{0, 1\}$  to a Boolean one, but then we would lose a factor of  $\log M$ .)

Second, if  $M = 2$  (that is,  $f$  is Boolean), then Conjecture 3 follows relatively easily from known results: indeed, we prove in Appendix 6 that  $R(f) = O(Q(f)^2)$  in that case. So the interesting case is when  $M \gg 2$ , as it is for the collision problem.

Conjecture 3 provides one natural way to formalize the idea that classical and quantum query complexities should be polynomially related for all “unstructured” problems. A different way is provided by the following conjecture, which we were aware of since about 1999:

**Conjecture 4 (folklore)** *Let  $Q$  be a quantum algorithm that makes  $T$  queries to a Boolean input  $X = (x_1, \dots, x_N)$ , and let  $\varepsilon > 0$ . Then there exists a deterministic classical algorithm that makes  $\text{poly}(T, 1/\varepsilon, 1/\delta)$  queries to the  $x_i$ ’s, and that approximates  $Q$ ’s acceptance probability to within an additive error  $\varepsilon$  on a  $1 - \delta$  fraction of inputs.*

Loosely speaking, while Conjecture 3 said that there was no property of a *symmetric* oracle string that quantum algorithms can evaluate superpolynomially faster than classical ones, Conjecture 4 says that there is no such property of a *random* oracle string.

Conjecture 4 would imply a far-reaching generalization of the result of Beals et al. [6] that  $D(f) = O(Q(f)^6)$  for all total Boolean functions  $f$ . In particular, define the  $\varepsilon$ -*approximate query complexity* of a Boolean function  $f : \{0, 1\}^N \rightarrow \{0, 1\}$ , or  $D_\varepsilon(f)$ , to be the minimum number of queries made by a deterministic algorithm that evaluates  $f$  on at least a  $1 - \varepsilon$  fraction of inputs  $X$ . Likewise, let  $Q_\varepsilon(f)$  be the minimum number of queries made by a quantum algorithm that evaluates  $f$  on at least a  $1 - \varepsilon$  fraction of inputs. Then Conjecture 4 implies that  $D_\varepsilon(f)$  and  $Q_\delta(f)$  are polynomially related for all Boolean functions  $f$  and all  $\varepsilon > \delta > 0$ . This would provide a quantum counterpart to a beautiful 2002 result of Smyth [24], who solved an old open problem of Steven Rudich by showing that  $D_\varepsilon(f) = O(C_{\varepsilon^{3/30}}(f)^2 / \varepsilon^3)$  for all  $\varepsilon > 0$  (where  $C_\delta(f)$  denotes the “ $\delta$ -approximate certificate complexity” of  $f$ ).

More dramatically, if Conjecture 4 holds, then *we basically cannot hope to prove  $P \neq \text{BQP}$  relative to a random oracle*. This would answer a question raised by Fortnow and Rogers [14] in 1998, and would contrast sharply with the situation for *non-random* oracles: we have had oracles relative to which  $P \neq \text{BQP}$ , and indeed  $\text{BQP} \not\subseteq \text{MA}$ , since the work of Bernstein and Vazirani [8] in the early 1990s. More precisely, under some suitable complexity assumption (such as  $P = P^{\#P}$ ), we would get  $\text{BQP}^A \subset \text{AvgP}^A$  with probability 1 for a random oracle  $A$ . Here  $\text{AvgP}$  is the class of languages for which there exists a polynomial-time algorithm that solves a  $1 - o(1)$  fraction of instances of size  $n$ . In other words, separating  $\text{BQP}$  from  $\text{AvgP}$  relative to a random oracle would be as hard as separating complexity classes in the unrelativized world. This would provide a quantum counterpart to a theorem of Impagliazzo and Rudich (credited in [17]), who used the powerful results of Kahn, Saks, and Smyth [17] to show that if  $P = \text{NP}$ , then  $\text{NP}^A \cap \text{coNP}^A \subset \text{ioAvgP}^A$  with probability 1 for a random oracle  $A$ .<sup>4</sup>

## 1.2 Our Results

Our main contribution in this paper is to prove Watrous’s conjecture, that randomized and quantum query complexities are polynomially related for every symmetric problem.

**Theorem 5** *Conjecture 3 holds. Indeed,  $R(f) = O(Q(f)^9 \text{polylog } Q(f))$  for every partial function  $f : [M]^N \rightarrow \{0, 1, *\}$  that is permutation-invariant.*

---

<sup>4</sup>Here  $\text{ioAvgP}$  means “average-case  $P$  for infinitely many input lengths  $n$ .” The reason Impagliazzo and Rudich only get a simulation in  $\text{ioAvgP}$ , rather than  $\text{AvgP}$ , has to do with the fact that Smyth’s result [24] only relates  $D_\varepsilon(f)$  to  $C_{\varepsilon^{3/30}}(f)$ , rather than  $D_{\varepsilon+\delta}(f)$  to  $C_\varepsilon(f)$  for all  $\delta > 0$ .

We conjecture that  $R(f)$  and  $Q(f)$  are polynomially related even for functions  $f$  satisfying *one* of the two symmetries: namely,  $f(x_1, \dots, x_N) = f(x_{\sigma(1)}, \dots, x_{\sigma(N)})$  for all  $\sigma \in S_N$ . We also conjecture that the exponent of 9 can be improved to 2: in other words, that Grover’s algorithm once again provides the optimal separation between the quantum and classical models.

While the proof of Theorem 5 is somewhat involved, it can be entirely understood by those unfamiliar with quantum computing: the difficulties lie in getting the problem into a form where existing quantum lower bound technology can be applied to it. Let us stress that it was not at all obvious *a priori* that existing quantum lower bounds would suffice here; that they did came as a surprise to us.

We first define and analyze a simple randomized algorithm, which tries to compute  $f(X)$  for a given  $X = (x_1, \dots, x_N)$  by estimating the multiplicity of each element  $x_i$ . Next, by considering where this randomized algorithm breaks down, we show that one can identify a “hard core” within  $f$ : roughly speaking, two input types  $\mathcal{A}^*$  and  $\mathcal{B}^*$ , such that the difficulty of distinguishing  $\mathcal{A}^*$  from  $\mathcal{B}^*$  accounts for a polynomial fraction of the entire difficulty of computing  $f$ . The rest of the proof consists of lower-bounding the quantum query complexity of distinguishing  $\mathcal{A}^*$  from  $\mathcal{B}^*$ . We do so using a hybrid argument: we develop a “chopping procedure” that gradually deforms  $\mathcal{A}^*$  and  $\mathcal{B}^*$  to make them more similar to each other, creating  $L = O(\log N)$  intermediate input types  $\mathcal{A}_0 = \mathcal{A}^*, \mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_L$  and  $\mathcal{B}_0 = \mathcal{B}^*, \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_L$  such that  $\mathcal{A}_L = \mathcal{B}_L$ . We then show that, for every  $\ell \in [L]$ , distinguishing  $\mathcal{A}_\ell$  from  $\mathcal{A}_{\ell-1}$  (or  $\mathcal{B}_\ell$  from  $\mathcal{B}_{\ell-1}$ ) requires many quantum queries, *either* by a reduction from Midrijanis’s quantum lower bound for the SET EQUALITY problem [18] (which was a nontrivial extension of Aaronson and Shi’s collision lower bound [3]), or *else* by an application of Ambainis’s general quantum adversary theorem [5].

Doing this hybrid argument in the “obvious” way produces a bound of the form  $R(f) \leq Q(f)^{O(1)}$  polylog  $N$ , which fails to imply a polynomial relationship between  $R(f)$  and  $Q(f)$  when  $Q(f) \leq (\log N)^{O(1)}$ . However, a more sophisticated hybrid argument gets rid of the polylog  $N$  factor.

Our second contribution is more exploratory, something we put forward in the hope of inspiring followup work. We study Conjecture 4, the one that stated that every  $T$ -query quantum algorithm can be simulated on *most* inputs using  $T^{O(1)}$  classical queries. We relate this conjecture to a fundamental open problem in Fourier analysis and approximation theory. Given a real polynomial  $p : \mathbb{R}^N \rightarrow \mathbb{R}$ , let

$$\text{Inf}_i [p] := \text{EX}_{X \in \{0,1\}^N} [|p(X) - p(X^i)|]$$

be the *influence* of the  $i^{\text{th}}$  variable, where  $X^i$  means  $X$  with the  $i^{\text{th}}$  bit flipped. Then we conjecture that *every bounded low-degree polynomial has a “highly influential” variable*. More precisely:

**Conjecture 6 (Bounded Polynomials Have Influential Variables)** *Let  $p : \mathbb{R}^N \rightarrow \mathbb{R}$  be a polynomial of degree  $d$ . Suppose that  $0 \leq p(X) \leq 1$  for all  $X \in \{0,1\}^N$ , and*

$$\text{E}_{X,Y \in \{0,1\}^N} [|p(X) - p(Y)|] \geq \varepsilon.$$

*Then there exists an  $i$  such that  $\text{Inf}_i [p] \geq (\varepsilon/d)^{O(1)}$ .*

We show the following:

**Theorem 7** *Assume Conjecture 6. Then*

(i) *Conjecture 4 holds.*

(ii)  $D_{\varepsilon+\delta}(f) \leq (Q_\varepsilon(f)/\delta)^{O(1)}$  for all Boolean functions  $f : \{0,1\}^N \rightarrow \{0,1\}$  and all  $\varepsilon, \delta > 0$ .

(iii) If  $P = P^{\#P}$ , then  $BQP^A \subset \text{AvgP}^A$  with probability 1 for a random oracle  $A$ .

The main evidence for Conjecture 6—besides the fact that all the Fourier analysis experts we asked were confident of it!—is that extremely similar statements have recently been proved. Firstly, O’Donnell, Saks, Schramm, and Servedio [19] proved an analogue of Conjecture 6 for *decision trees*, which are a special case of bounded real polynomials:

**Theorem 8 (O’Donnell et al. 2005 [19])** *Let  $f : \{0,1\}^N \rightarrow \{0,1\}$  be a Boolean function, and suppose  $\Pr[f = 1] \Pr[f = 0] \geq \varepsilon$ . Then there exists an  $i$  such that  $\text{Inf}_i[f] \geq 4\varepsilon/D(f)$ , where  $D(f)$  is the decision tree complexity of  $f$ .*

Unfortunately, Theorem 8 does not directly imply anything about our problem, even though Beals et al. [6] showed that  $D(f)$  and  $Q(f)$  are polynomially related for all total Boolean functions  $f$ . The reason is that the acceptance probability of a quantum algorithm need not approximate a total Boolean function.

The second piece of evidence for Conjecture 6 comes from a powerful result of Dinur, Friedgut, Kindler, and O’Donnell [13], which implies our conjecture, *except* with  $\text{Inf}_i[p] \geq \varepsilon^5/2^{O(d)}$  instead of  $\text{Inf}_i[p] \geq (\varepsilon/d)^{O(1)}$ . Let us state the special case of their result that is relevant for us:

**Theorem 9 (Dinur et al. 2006 [13])** *Let  $\varepsilon > 0$ , and let  $p : \mathbb{R}^N \rightarrow \mathbb{R}$  be a degree- $d$  polynomial such that  $0 \leq p(X) \leq 1$  for all  $X \in \{0,1\}^N$ . Then there exists a  $2^{O(d)}/\varepsilon^2$ -junta  $\tilde{p} : \mathbb{R}^N \rightarrow \mathbb{R}$  (that is, a polynomial depending on at most  $2^{O(d)}/\varepsilon^2$  variables) such that*

$$\mathbb{E}_{X \in \{0,1\}^N} \left[ (\tilde{p}(X) - p(X))^2 \right] \leq \varepsilon.$$

Even though Theorem 9 has an exponential rather than polynomial dependence on  $1/d$ , we observe that it *already* has a nontrivial consequence for quantum computation. Namely, it implies that any  $T$ -query quantum algorithm can be simulated on *most* inputs using  $2^{O(T)}$  classical queries.<sup>5</sup> Recall that the gaps between classical and quantum query complexities can be superexponential (and even  $N^{\Omega(1)}$  versus  $O(1)$ , as in the example of Aaronson [2]), so even an exponential upper bound is far from obvious.

## 2 Quantum Lower Bound for All Symmetric Problems

In this section we prove Theorem 5: that  $R(f) = O(Q(f)^9 \text{polylog } Q(f))$  for all permutation-symmetric  $f$ .

We start with a simple observation that is essential to everything that follows. Since  $f$  is symmetric, we can group the inputs  $X = (x_1, \dots, x_N)$  into equivalence classes that we call *types*.

**Definition 10** *Given an input  $X = (x_1, \dots, x_N) \in [M]^N$ , the type of  $X$  is a list of positive integers  $\mathcal{A} = (a_1, \dots, a_u)$  such that  $a_1 \geq \dots \geq a_u$  and  $a_1 + \dots + a_u = N$ , with each  $a_i$  recording the multiplicity of some integer in  $X$ . For convenience, we adopt the convention that  $a_i = 0$  for all  $i > u$ .*

---

<sup>5</sup>Indeed, in this case the classical queries are nonadaptive.

In other words, a type is just a partition (or *Young diagram*) that records the multiplicities of the input elements. For example, a one-to-one input has type  $a_1 = \dots = a_N = 1$ , while a two-to-one input has type  $a_1 = \dots = a_{N/2} = 2$ . We write  $X \in \mathcal{A}$  if  $X$  is of type  $\mathcal{A}$ . Clearly  $f(X)$  depends only on the type of  $X$ . Furthermore, given a quantum query algorithm  $Q$ , we can assume without loss of generality that  $\Pr [Q \text{ accepts } X]$  depends only on the type of  $X$ —since we can “symmetrize”  $Q$  (that is, randomly permute  $X$ ’s inputs and outputs) prior to running  $Q$ .

## 2.1 Randomized Upper Bound

Let  $X = (x_1, \dots, x_N)$  be an input. For each  $j \in [M]$ , let  $\kappa_j$  be the number of  $i$ ’s such that  $x_i = j$ . Then the first step is to give a classical randomized algorithm that estimates the  $\kappa_j$ ’s. This algorithm,  $\mathcal{S}_T$ , is an extremely straightforward sampling procedure (indeed, there is essentially nothing else that a randomized algorithm can do here).  $\mathcal{S}_T$  will make  $O(T^{1+c} \log T)$  queries, where  $T$  is a parameter and  $c \in (0, 1]$  is a constant that we will choose later to optimize the final bound.

```

Set  $U := 24T^{1+c} \ln T$ 
Choose  $U$  indices  $i_1, \dots, i_U \in [N]$  uniformly at random with replacement
Query  $x_{i_1}, \dots, x_{i_U}$ 
For each  $j \in [M]$ :
  Let  $z_j$  be the number of occurrences of  $j$  in  $(x_{i_1}, \dots, x_{i_U})$ 
  Output  $\tilde{\kappa}_j := \frac{N}{U} z_j$  as the estimate for  $\kappa_j$ 

```

We now analyze how well  $\mathcal{S}_T$  works.

**Lemma 11** *With probability  $1 - O(1/T)$ , we have  $|\tilde{\kappa}_j - \kappa_j| \leq \frac{N}{T} + \frac{\kappa_j}{T^c}$  for all  $j \in [M]$ .*

**Proof.** For each  $j \in [M]$ , we consider three cases. First suppose  $\kappa_j \geq N/T^{1-c}$ . Notice that  $z_j$  is a sum of  $U$  independent Boolean variables, and that  $\mathbb{E}[z_j] = \frac{U}{N} \mathbb{E}[\tilde{\kappa}_j] = \frac{U}{N} \kappa_j$ . So

$$\begin{aligned}
\Pr \left[ |\tilde{\kappa}_j - \kappa_j| > \frac{\kappa_j}{T^c} \right] &= \Pr \left[ \left| z_j - \frac{U}{N} \kappa_j \right| > \frac{U \kappa_j}{NT^c} \right] \\
&< 2 \exp \left( -\frac{U \kappa_j / N}{4T^{2c}} \right) \\
&< 2 \exp \left( -\frac{U}{4T^{1+c}} \right) \\
&= 2T^{-6},
\end{aligned}$$

where the second line follows from a Chernoff bound and the third from  $\kappa_j \geq N/T^{1-c}$ .

Second, suppose  $N/T^5 \leq \kappa_j < N/T^{1-c}$ . Then

$$\begin{aligned}
\Pr \left[ |\tilde{\kappa}_j - \kappa_j| > \frac{N}{T} \right] &= \Pr \left[ \left| z_j - \frac{U}{N} \kappa_j \right| > \frac{U}{T} \right] \\
&< 2 \exp \left( -\frac{U \kappa_j / N}{4} \left( \frac{N}{T \kappa_j} \right)^2 \right) \\
&< 2 \exp \left( -\frac{U}{4T^{1+c}} \right) \\
&= 2T^{-6}
\end{aligned}$$

where the second line follows from a Chernoff bound and the third from  $\kappa_j < N/T^{1-c}$ .

Third, suppose  $\kappa_j < N/T^5$ . Then

$$\begin{aligned} \Pr \left[ |\tilde{\kappa}_j - \kappa_j| > \frac{N}{T} \right] &= \Pr \left[ \left| z_j - \frac{U}{N} \kappa_j \right| > \frac{U}{T} \right] \\ &\leq \Pr [z_j \geq 2] \\ &\leq \binom{U}{2} \left( \frac{\kappa_j}{N} \right)^2 \\ &\leq \frac{U^2}{T^5} \left( \frac{\kappa_j}{N} \right) \\ &\leq \frac{\kappa_j}{TN} \end{aligned}$$

for all sufficiently large  $T$ , where the second line follows from  $\kappa_j < N/T^5$ , the third from the union bound, the fourth from  $\kappa_j < N/T^5$  (again), and the fifth from  $U \leq 24T^2 \ln T$ .

Notice that there are at most  $T^5$  values of  $j$  such that  $\kappa_j \geq N/T^5$ . So putting all three cases together,

$$\begin{aligned} \Pr \left[ \exists j : |\tilde{\kappa}_j - \kappa_j| > \frac{N}{T} + \frac{\kappa_j}{T^c} \right] &\leq \sum_{j:\kappa_j \geq N/T^{1-c}} \Pr \left[ |\tilde{\kappa}_j - \kappa_j| > \frac{\kappa_j}{T^c} \right] \\ &\quad + \sum_{j:N/T^5 \leq \kappa_j < N/T^{1-c}} \Pr \left[ |\tilde{\kappa}_j - \kappa_j| > \frac{N}{T} \right] \\ &\quad + \sum_{j:\kappa_j < N/T^5} \Pr \left[ |\tilde{\kappa}_j - \kappa_j| > \frac{N}{T} \right] \\ &\leq T^5 \left( \frac{2}{T^6} \right) + \sum_{j:\kappa_j < N/T^5} \frac{\kappa_j}{TN} \\ &= O \left( \frac{1}{T} \right). \end{aligned}$$

■

Now call  $\mathcal{A}$  a 1-type if  $f(X) = 1$  for all  $X \in \mathcal{A}$ , or a 0-type if  $f(X) = 0$  for all  $X \in \mathcal{A}$ . Consider the following randomized algorithm  $\mathcal{R}_T$  to compute  $f(X)$ :

Run  $\mathcal{S}_T$  to find an estimate  $\tilde{\kappa}_i$  for each  $\kappa_i$   
Sort the  $\tilde{\kappa}_i$ 's in descending order, so that  $\tilde{\kappa}_1 \geq \dots \geq \tilde{\kappa}_M$   
If there exists a 1-type  $\mathcal{A} = (a_1, a_2, \dots)$  such that  $|\tilde{\kappa}_i - a_i| \leq \frac{N}{T} + \frac{a_i}{T^c}$   
for all  $i$ , then output  $f(X) = 1$   
Otherwise output  $f(X) = 0$

Clearly  $\mathcal{R}_T$  makes  $O(T^{1+c} \log T)$  queries, just as  $\mathcal{S}_T$  does. We now give a sufficient condition for  $\mathcal{R}_T$  to succeed.

**Lemma 12** *Suppose that for all 1-types  $\mathcal{A} = (a_1, a_2, \dots)$  and 0-types  $\mathcal{B} = (b_1, b_2, \dots)$ , there exists an  $i$  such that  $|a_i - b_i| > \frac{2N}{T} + \frac{a_i + b_i}{T^c}$ . Then  $\mathcal{R}_T$  computes  $f$  with bounded probability of error, and hence  $R(f) = O(T^{1+c} \log T)$ .*

**Proof.** First suppose  $X \in \mathcal{A}$  where  $\mathcal{A} = (a_1, a_2, \dots)$  is a 1-type. Then by Lemma 11, with probability  $1 - O(1/T)$  we have  $|\tilde{\kappa}_i - a_i| \leq \frac{N}{T} + \frac{a_i}{T^c}$  for all  $i$  (it is easy to see that sorting the  $\tilde{\kappa}_i$ 's can only decrease the maximum difference). Provided this occurs,  $\mathcal{R}_T$  finds some 1-type close to  $(\tilde{\kappa}_1, \tilde{\kappa}_2, \dots)$  (possibly  $\mathcal{A}$  itself) and outputs  $f(X) = 1$ .

Second, suppose  $X \in \mathcal{B}$  where  $\mathcal{B} = (b_1, b_2, \dots)$  is a 0-type. Then with probability  $1 - O(1/T)$  we have  $|\tilde{\kappa}_i - b_i| \leq \frac{N}{T} + \frac{b_i}{T^c}$  for all  $i$ . Provided this occurs, by the triangle inequality, for every 1-type  $\mathcal{A} = (a_1, a_2, \dots)$  there exists an  $i$  such that

$$|\tilde{\kappa}_i - a_i| \geq |a_i - b_i| - |\tilde{\kappa}_i - b_i| > \frac{N}{T} + \frac{a_i}{T^c}.$$

Hence  $\mathcal{R}_T$  does *not* find a 1-type close to  $(\tilde{\kappa}_1, \tilde{\kappa}_2, \dots)$ , and it outputs  $f(X) = 0$ . ■

In particular, suppose we keep decreasing  $T$  until there exists a 1-type  $\mathcal{A}^* = (a_1, a_2, \dots)$  and a 0-type  $\mathcal{B}^* = (b_1, b_2, \dots)$  such that

$$|a_i - b_i| \leq \frac{3N}{T} + \frac{a_i + b_i}{T^c} \tag{1}$$

for all  $i$ , stopping as soon as that happens. Then Lemma 12 implies that we will still have  $R(f) = O(T^{1+c} \log T)$ . For the rest of the proof, we will fix that “almost as small as possible” value of  $T$  for which (1) holds, as well as the 1-type  $\mathcal{A}^*$  and the 0-type  $\mathcal{B}^*$  that  $\mathcal{R}_T$  “just barely distinguishes” from one another.

## 2.2 The Chopping Procedure

Given two sets of inputs  $A$  and  $B$  with  $A \cap B = \emptyset$ , let  $Q(A, B)$  be the minimum number of queries made by any quantum algorithm that accepts every  $X \in A$  with probability at least  $2/3$ , and accepts every  $Y \in B$  with probability at most  $1/3$ . Also, let  $Q_\varepsilon(A, B)$  be the minimum number of queries made by any quantum algorithm that accepts every  $X \in A$  with at least some probability  $p$ , and that accepts every  $Y \in B$  with probability at most  $p - \varepsilon$ . Then we have the following basic relation:

**Proposition 13**  $Q(A, B) = O\left(\frac{1}{\varepsilon} Q_\varepsilon(A, B)\right)$  for all  $A, B$  and all  $\varepsilon > 0$ .

**Proof.** This follows from standard amplitude estimation techniques (see Brassard et al. [9] for example). ■

The rest of the proof is going to consist of lower-bounding  $Q(\mathcal{A}^*, \mathcal{B}^*)$ , the quantum query complexity of distinguishing inputs of type  $\mathcal{A}^*$  from inputs of type  $\mathcal{B}^*$ . We do this via a hybrid argument. Let  $L := \lceil \log_2 N \rceil$ . At a high level, we will construct two sequences of types,  $\mathcal{A}_0, \dots, \mathcal{A}_L$  and  $\mathcal{B}_0, \dots, \mathcal{B}_L$ , such that

- (i)  $\mathcal{A}_0 = \mathcal{A}^*$ ,
- (ii)  $\mathcal{B}_0 = \mathcal{B}^*$ ,
- (iii)  $\mathcal{A}_L = \mathcal{B}_L$ , and
- (iv)  $Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1})$  and  $Q(\mathcal{B}_\ell, \mathcal{B}_{\ell-1})$  are large for every  $\ell \in [L]$ .

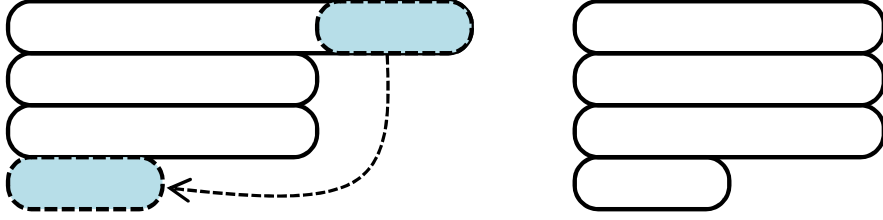


Figure 1: Chopping a row of  $\mathcal{A}_\ell$ 's Young diagram to make it more similar to  $\mathcal{B}_\ell$ .

Provided we can do this, it is not hard to see that we get the desired lower bound on  $Q(\mathcal{A}^*, \mathcal{B}^*)$ . For suppose a quantum algorithm distinguishes  $\mathcal{A}_0 = \mathcal{A}^*$  from  $\mathcal{B}_0 = \mathcal{B}^*$  with constant bias. Then by the triangle inequality, it must also distinguish *some*  $\mathcal{A}_\ell$  from  $\mathcal{A}_{\ell+1}$ , or *some*  $\mathcal{B}_\ell$  from  $\mathcal{B}_{\ell+1}$ , with reasonably large bias (say  $\Omega(1/\log N)$ ). And by Proposition 13, any quantum algorithm that succeeds with bias  $\varepsilon$  can be amplified, with  $O(1/\varepsilon)$  overhead, to an algorithm that succeeds with constant bias.

We now describe the procedure for creating the intermediate types  $\mathcal{A}_\ell$  and  $\mathcal{B}_\ell$ . Intuitively, we want to form  $\mathcal{A}_\ell$  from  $\mathcal{A}_{\ell-1}$ , and  $\mathcal{B}_\ell$  from  $\mathcal{B}_{\ell-1}$ , by “chopping the rows” of their respective Young diagrams, whenever a row of  $\mathcal{A}_\ell$  sticks out further than the corresponding row of  $\mathcal{B}_\ell$  or vice versa. This way, we can gradually make  $\mathcal{A}_\ell$  and  $\mathcal{B}_\ell$  more similar to one another. To describe how this works, it will be convenient to relax the notion of a type slightly. Let a *row-array* be a list of  $2N$  nonnegative integers  $(a_1, \dots, a_{2N})$ , not necessarily sorted, such that  $a_1 + \dots + a_{2N} = N$ . Note that every type  $(a_1, \dots, a_u)$  is also a row-array, if we adopt the convention that  $a_{u+1} = \dots = a_{2N} = 0$ . Also, every row-array  $(a_1, \dots, a_{2N})$  can be converted to a type  $\mathcal{A} = \text{type}(a_1, \dots, a_{2N})$  in a unique way, by simply sorting the  $a_i$ 's in descending order.

We construct the intermediate types  $\mathcal{A}_1, \mathcal{A}_2, \dots$  via the following procedure. In this procedure,  $(a_1, \dots, a_{2N})$  is a row-array that is initialized to  $\mathcal{A}^*$ , and  $\mathcal{B}^* = (b_1, b_2, \dots)$ .

```

let  $P$  be the first power of 2 greater than or equal to  $N$ 
for  $\ell := 1$  to  $L$ 
  for every  $i \in [2N]$  such that  $a_i - b_i \geq P/2^\ell$  and  $a_i > P/2^\ell$ 
    set  $a_i := a_i - P/2^\ell$ 
    find a  $j \in [2N]$  such that  $a_j = b_j = 0$ , and set  $a_j := P/2^\ell$ 
  set  $\mathcal{A}_\ell := \text{type}(a_1, \dots, a_{2N})$ 
next  $\ell$ 

```

The procedure to produce  $\mathcal{B}_1, \mathcal{B}_2, \dots$  is exactly the same, except with the roles of  $a$  and  $b$  reversed. The procedure is illustrated pictorially in Figure 1.

We start with some simple observations. First, by construction, this procedure halts after  $L = O(\log N)$  iterations. Second, within a given iteration  $\ell$ , no row  $i$  is ever chopped more than once—for if it could be, then it would have been chopped in a previous iteration. (This is just saying that the integer  $a_i - b_i$  can be written uniquely as a sum of powers of 2.) Third, initially  $a_i = b_i = 0$  for all  $N + 1 \leq i \leq 2N$ , and there are at most  $N - 1$  chopping steps throughout the whole procedure. That is why it is always possible to find a  $j \in [2N]$  such that  $a_j = b_j = 0$ .

Define

$$\|\mathcal{A} - \mathcal{B}\| := \frac{1}{2} \sum_{i=1}^N |a_i - b_i|.$$

Then it is not hard to see that  $\mathcal{A}_1, \mathcal{A}_2, \dots$  and  $\mathcal{B}_1, \mathcal{B}_2, \dots$  both evolve toward the same final configuration: namely,  $\|\mathcal{A}^* - \mathcal{B}^*\|$  singleton rows, together with one row of length  $\min\{a_i, b_i\}$  for each  $i$  such that  $\min\{a_i, b_i\} > 0$ . We therefore have the key fact that  $\mathcal{A}_L = \mathcal{B}_L$ .

Notice that  $\|\mathcal{A}_\ell - \mathcal{A}_{\ell-1}\| = rP/2^\ell$ , where  $r$  is number of rows that get chopped in the  $\ell^{\text{th}}$  iteration. We now prove an upper bound on  $\|\mathcal{A}_\ell - \mathcal{A}_{\ell-1}\|$  when  $\ell$  is small, which will be useful later.

**Lemma 14** *If  $\ell \leq (\log_2 T) - 2$ , then  $\|\mathcal{A}_\ell - \mathcal{A}_{\ell-1}\| \leq \frac{8N}{T^c}$ .*

**Proof.** Let  $C$  be the set of rows that are chopped in going from  $\mathcal{A}_{\ell-1}$  to  $\mathcal{A}_\ell$ ; then each row  $i \in C$  decreases in length by  $P/2^\ell$ . It follows that, if we let  $j = j(i)$  be the ‘‘ancestral row’’ in  $\mathcal{A}^* = (a_1, a_2, \dots)$  that  $i$  came from, we must have

$$\frac{P}{2^\ell} \leq |a_j - b_j| \leq \frac{3N}{T} + \frac{a_j + b_j}{T^c}.$$

Since  $\ell \leq (\log_2 T) - 2$ , the left inequality implies

$$|a_j - b_j| \geq \frac{4N}{T},$$

which combined with the right inequality yields

$$\frac{a_j + b_j}{T^c} \geq \frac{N}{T}.$$

Now let  $R := \bigcup_{i \in C} j(i)$  be the set of all ancestral rows. Then

$$\begin{aligned} \|\mathcal{A}_\ell - \mathcal{A}_{\ell-1}\| &\leq \sum_{i \in R} |a_i - b_i| \\ &\leq \sum_{i \in R} \left( \frac{3N}{T} + \frac{a_i + b_i}{T^c} \right) \\ &\leq 4 \sum_{i \in R} \frac{a_i + b_i}{T^c} \\ &\leq \frac{8N}{T^c}. \end{aligned}$$

■

### 2.3 Quantum Lower Bounds

Recall that we listed four properties that we needed the chopping procedure to satisfy. We have already seen that it satisfies properties (i)-(iii), so the remaining step is to show that it satisfies property (iv). That is, we need to lower-bound  $Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1})$ , the bounded-error quantum query complexity of distinguishing inputs of type  $\mathcal{A}_\ell$  from inputs of type  $\mathcal{A}_{\ell-1}$ . (Lower-bounding  $Q(\mathcal{B}_\ell, \mathcal{B}_{\ell-1})$  is exactly analogous.) To do this, it will be convenient to consider two cases: first, that forming  $\mathcal{A}_\ell$  involved chopping few elements of  $\mathcal{A}_{\ell-1}$ , and second, that it involved chopping

many elements. We will show that we “win either way,” by a different quantum lower bound in each case.

First consider the case that few elements were chopped. Here we prove a lower bound using Ambainis’s quantum adversary method [5], in its “general” form (the one used, for example, to lower-bound the quantum query complexity of inverting a permutation). For completeness, we now state Ambainis’s adversary theorem in the form we will need.

**Theorem 15 (Ambainis [5])** *Let  $A, B \subseteq [M]^N$  be two sets of inputs with  $A \cap B = \emptyset$ . Let  $R \subseteq A \times B$  be a relation on input pairs, such that for every  $X \in A$  there exists at least one  $Y \in B$  with  $(X, Y) \in R$  and vice versa. Given inputs  $X = (x_1, \dots, x_N)$  in  $A$  and  $Y = (y_1, \dots, y_N)$  in  $B$ , let*

$$q_{X,i} = \Pr_{Y \in B} [x_i \neq y_i \mid (X, Y) \in R],$$

$$q_{Y,i} = \Pr_{X \in A} [x_i \neq y_i \mid (X, Y) \in R].$$

*Suppose that  $q_{X,i}q_{Y,i} \leq \alpha$  for every  $(X, Y) \in R$  and every  $i \in [N]$  such that  $x_i \neq y_i$ . Then  $Q(A, B) = \Omega(1/\sqrt{\alpha})$ .*

Using Theorem 15, we can prove the following lower bound on  $Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1})$ .

**Lemma 16** *Let  $d = \|\mathcal{A}_\ell - \mathcal{A}_{\ell-1}\|$ , and assume  $d \leq N/2$ . Then  $Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1}) = \Omega(\sqrt{N/d})$ .*

**Proof.** Let  $\mathcal{A}_{\ell-1} = (a_1, a_2, \dots)$ , and let  $i(1), \dots, i(r)$  be the  $r$  rows in  $\mathcal{A}_{\ell-1}$  that get chopped. Recall that in going from  $\mathcal{A}_{\ell-1}$  to  $\mathcal{A}_\ell$ , we chop each row  $i(j)$  into a row of length  $a_{i(j)} - P/2^\ell$  and another row of length  $P/2^\ell$ , so that  $d = rP/2^\ell$ .

Now, given inputs  $X = (x_1, \dots, x_N)$  in  $\mathcal{A}_{\ell-1}$  and  $Y = (y_1, \dots, y_N)$  in  $\mathcal{A}_\ell$ , we set  $(X, Y) \in R$  if and only if one can transform  $X$  to  $Y$  in the following way:

- (1) Find distinct  $h_1, \dots, h_r \in [M]$  such that for each  $j \in [r]$ , there are exactly  $a_{i(j)}$  indices  $i \in [N]$  satisfying  $x_i = h_j$ .
- (2) For each  $j \in [r]$ , change exactly  $P/2^\ell$  of the  $x_i$ ’s that are equal to  $h_j$  to something else.
- (3) Swap the  $d$  elements of  $X$  that were changed in step (2) with any other  $d$  elements of  $X$ .

The procedure is illustrated pictorially in Figure 2. Note that we can reverse the procedure in a natural way to go from  $Y$  back to  $X$ :

- (1) Find distinct  $h_1, \dots, h_r \in [M]$  such that for each  $j \in [r]$ , there are exactly  $P/2^\ell$  indices  $i \in [N]$  satisfying  $y_i = h_j$ .
- (2) For each  $j \in [r]$ , change all of the  $y_i$ ’s that are equal to  $h_j$  to something else.
- (3) Swap the  $d$  elements of  $Y$  that were changed in step (2) with any other  $d$  elements of  $Y$ .

Fix any  $(X, Y) \in R$ , and let  $i \in [N]$  be any index such that  $x_i \neq y_i$ . Then applying Theorem 15, we claim that either  $q_{X,i} \leq \frac{d}{N-d}$  or  $q_{Y,i} \leq \frac{d}{N-d}$ . To see this, observe that either  $x_i$  is one of the “other  $d$  elements” in step (3) of the  $X \rightarrow Y$  conversion, in which case

$$\Pr_{Y' \in \mathcal{A}_\ell} [x_i \neq y'_i \mid (X, Y') \in R] \leq \frac{d}{N-d},$$

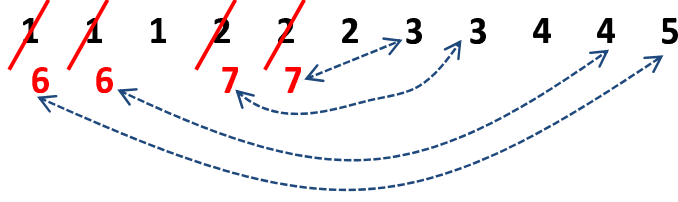


Figure 2: In this example,  $N = 11$ ,  $r = 2$ ,  $P/2^\ell = 2$ , and  $a_1 = a_2 = 3$ . So we transform  $X$  to  $Y$  by choosing  $h_1 = 1$  and  $h_2 = 2$ , changing any two elements equal to  $h_1$  and any two elements equal to  $h_2$ , and then swapping the four elements that we changed with four unchanged elements.

or else  $y_i$  is one of the “other  $d$  elements” in step (3) of the  $Y \rightarrow X$  conversion, in which case

$$\Pr_{X' \in \mathcal{A}_{\ell-1}} [x'_i \neq y_i \mid (X', Y) \in R] \leq \frac{d}{N-d}.$$

Hence

$$q_{X,i} q_{Y,i} \leq \frac{d}{N-d}.$$

So by Theorem 15,

$$Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1}) = \Omega\left(\frac{1}{\sqrt{q_{X,i} q_{Y,i}}}\right) = \Omega\left(\sqrt{\frac{N-d}{d}}\right) = \Omega\left(\sqrt{\frac{N}{d}}\right).$$

■

We now consider the case that many elements are chopped. Here we prove a lower bound by reduction from SET EQUALITY. Given two sequences of integers  $Y \in [M]^N$  and  $Z \in [M]^N$ , neither with any repeats, the SET EQUALITY problem is to decide whether  $Y$  and  $Z$  are equal as sets or disjoint as sets, promised that one of these is the case. SET EQUALITY is similar to the collision problem studied by Aaronson and Shi [3], but it lacks permutation symmetry, making it harder to prove a lower bound by the polynomial method. By combining the collision lower bound with Ambainis’s adversary method, Midrijanis [18] was nevertheless able to show the following:

**Theorem 17 (Midrijanis [18])**  $Q(\text{SET EQUALITY}) = \Omega\left((N/\log N)^{1/5}\right)$ .

We now use Theorem 17 to prove another lower bound on  $Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1})$ .

**Lemma 18** *Suppose  $\mathcal{A}_\ell$  was formed from  $\mathcal{A}_{\ell-1}$  by chopping  $r$  rows. Then*

$$Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1}) = \Omega\left(\left(\frac{r}{\log r}\right)^{1/5}\right).$$

**Proof.** We will show how to embed a SET EQUALITY instance of size  $r$  into the  $\mathcal{A}_\ell$  versus  $\mathcal{A}_{\ell-1}$  problem.

Let  $\mathcal{A}_{\ell-1} = (a_1, \dots, a_u)$ . Also, let  $i(1), \dots, i(r) \in [u]$  be the  $r$  rows that are chopped in going from  $\mathcal{A}_{\ell-1}$  to  $\mathcal{A}_\ell$ , and let  $j(1), \dots, j(u-r) \in [u]$  be the  $u-r$  rows that are *not* chopped. Recall that, in going from  $\mathcal{A}_{\ell-1}$  to  $\mathcal{A}_\ell$ , we chop each row  $i(k)$  into a row of length  $a_{i(k)} - P/2^\ell$  and another row of length  $P/2^\ell$ .

Now let  $Y = (y_1, \dots, y_r)$  and  $Z = (z_1, \dots, z_r)$  be an instance of SET EQUALITY. Then we construct an input  $X \in [M]^N$  as follows. First, for each  $k \in [r]$ , set  $a_{i(k)} - P/2^\ell$  of the  $x_i$ 's equal to  $y_k$ , and set  $P/2^\ell$  of the  $x_i$ 's equal to  $z_k$ . Next, let  $w_1, w_2, \dots \in [M]$  be a list of numbers that are guaranteed *not* to be in  $Y \cup Z$ . Then for each  $k \in [u - r]$ , set  $a_{j(k)}$  of the  $x_i$ 's equal to  $w_k$ .

It is easy to see that, if  $Y$  and  $Z$  are equal as sets, then  $X$  will have type  $\mathcal{A}_{\ell-1}$ , while if  $Y$  and  $Z$  are disjoint as sets, then  $X$  will have type  $\mathcal{A}_\ell$ . So in deciding whether  $X$  belongs to  $\mathcal{A}_\ell$  or  $\mathcal{A}_{\ell-1}$ , we also decide whether  $Y$  and  $Z$  are equal or disjoint. The lemma now follows from Theorem 17.  $\blacksquare$

## 2.4 Putting Everything Together

Let  $\mathcal{C}$  be a quantum query algorithm that distinguishes  $\mathcal{A}_0 = \mathcal{A}^*$  from  $\mathcal{B}_0 = \mathcal{B}^*$ , and assume  $\mathcal{C}$  is optimal: that is, it makes  $Q(\mathcal{A}^*, \mathcal{B}^*) \leq Q(f)$  queries. As mentioned earlier, we can assume that  $\Pr[\mathcal{C} \text{ accepts } X]$  depends only on the type of  $X$ . So let

$$\begin{aligned} p_\ell &:= \Pr[\mathcal{C} \text{ accepts } X \in \mathcal{A}_\ell], \\ q_\ell &:= \Pr[\mathcal{C} \text{ accepts } X \in \mathcal{B}_\ell]. \end{aligned}$$

Then by assumption,  $|p_0 - q_0| \geq 1/3$ . Since  $p_L = q_L$ , this implies that either  $|p_0 - p_L| \geq 1/6$  or  $|q_0 - q_L| \geq 1/6$ . Assume the former without loss of generality.

Now let  $\beta_\ell := \frac{1}{10\ell^2}$ , and observe that  $\sum_{\ell=1}^{\infty} \beta_\ell < \frac{1}{6}$ . By the triangle inequality, it follows that there exists an  $\ell \in [L]$  such that  $|p_\ell - p_{\ell-1}| \geq \beta_\ell$ . In other words, we get a  $Q(f)$ -query quantum algorithm that distinguishes  $\mathcal{A}_\ell$  from  $\mathcal{A}_{\ell-1}$  with bias  $\beta_\ell$ . By Proposition 13, this immediately implies

$$Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1}) = O\left(\frac{Q(f)}{\beta_\ell}\right)$$

or equivalently

$$Q(f) = \Omega\left(\frac{Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1})}{\ell^2}\right).$$

Now let  $d = \|\mathcal{A}_\ell - \mathcal{A}_{\ell-1}\|$ , and suppose  $\mathcal{A}_\ell$  was produced from  $\mathcal{A}_{\ell-1}$  by chopping  $r$  rows. Then  $d = rP/2^\ell \leq 2rN/2^\ell$ . So combining Lemmas 16 and 18, we find that

$$\begin{aligned} Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1}) &= \Omega\left(\max\left\{\sqrt{\frac{N}{d}}, \left(\frac{r}{\log r}\right)^{1/5}\right\}\right) \\ &= \Omega\left(\sqrt{\frac{2^\ell}{r}} + \left(\frac{r}{\log r}\right)^{1/5}\right) \\ &= \Omega\left(\frac{2^{\ell/7}}{\ell^{1/7}}\right) \end{aligned}$$

since the minimum occurs when  $r \approx 2^{5\ell/7} \ell^{2/7}$ . If  $\ell \leq (\log_2 T) - 2$ , then combining Lemmas 16 and 14, we also have the lower bound

$$Q(\mathcal{A}_\ell, \mathcal{A}_{\ell-1}) = \Omega\left(\sqrt{\frac{N}{8N/T^c}}\right) = \Omega(\sqrt{T^c}).$$

Hence

$$Q(f) = \begin{cases} \Omega\left(\frac{\sqrt{T^c}}{\ell^2}\right) & \text{if } \ell \leq (\log_2 T) - 2 \\ \Omega\left(\frac{1}{\ell^2} \cdot \frac{2^{\ell/7}}{\ell^{1/7}}\right) & \text{if } \ell > (\log_2 T) - 2 \end{cases}$$

Let us now make the choice  $c = 2/7$ , so that we get a lower bound of

$$Q(f) = \Omega\left(\frac{T^{1/7}}{\log^{15/7} T}\right)$$

in either case. Hence  $T = O(Q(f)^7 \log^{15} Q(f))$ . So by Lemma 12,

$$\begin{aligned} R(f) &= O(T^{1+c} \log T) \\ &= O(T^{9/7} \log T) \\ &= O(Q(f)^9 \log^{21} Q(f)). \end{aligned}$$

This completes the proof of Theorem 5.

### 3 Quantum Lower Bounds Under The Uniform Distribution

In this section, we consider the problems of  $\mathsf{P} \stackrel{?}{=} \mathsf{BQP}$  relative to a random oracle, and of simulating a  $T$ -query quantum algorithm on *most* inputs using  $T^{O(1)}$  classical queries. We show that these problems are connected to a fundamental conjecture about influences in low-degree polynomials.

Let  $p : \{0, 1\}^N \rightarrow [0, 1]$  be a real polynomial. Given a string  $X \in \{0, 1\}^N$ , let  $X^i$  denote  $X$  with the  $i^{\text{th}}$  bit flipped. The following notions will play important roles in this section: the  $L_1$ -variance  $\text{Vr}[p]$  of  $p$ , the *influence*  $\text{Inf}_i[p]$  of the  $i^{\text{th}}$  variable  $x_i$ , the *total influence*  $\text{SumInf}[p]$ , and the  $L_2$ -norm  $\|p\|_2^2$ .

$$\begin{aligned} \text{Vr}[p] &:= \mathbb{E}_{X, Y \in \{0, 1\}^N} [|p(X) - p(Y)|], \\ \text{Inf}_i[p] &:= \mathbb{E}_{X \in \{0, 1\}^N} [|p(X) - p(X^i)|], \\ \text{SumInf}[p] &:= \sum_{i=1}^N \text{Inf}_i[p], \\ \|p\|_2^2 &:= \mathbb{E}_{X \in \{0, 1\}^N} [p(X)^2]. \end{aligned}$$

Recall Conjecture 6, which stated that *bounded polynomials have influential variables*: that is, for every degree- $d$  polynomial  $p : \mathbb{R}^N \rightarrow \mathbb{R}$  such that  $0 \leq p(X) \leq 1$  for all  $X \in \{0, 1\}^N$ , there exists an  $i \in [N]$  such that  $\text{Inf}_i[p] \geq (\text{Vr}[p]/d)^{O(1)}$ . Assuming Conjecture 6, we will derive a number of consequences for quantum complexity theory.

To do so, we first need a lemma that slightly generalizes a result of Shi [21].

**Lemma 19** *Suppose a quantum algorithm makes  $T$  queries, and accepts the input  $X \in \{0, 1\}^N$  with probability  $p(X)$ . Then  $\text{SumInf}[p] = O(T)$ .*

**Proof.** Let  $|\psi_X\rangle$  be the final state of the quantum algorithm if the input is  $X$ , and let

$$E := \mathbb{E}_{X \in \{0,1\}^N, i \in [N]} \left[ \|\psi_X - \psi_{X^i}\|^2 \right].$$

Then Lemma 4.3 of Shi [21] implies that  $E \leq 2T/N$ . Hence

$$\text{SumInf}[p] = N \cdot \mathbb{E}_{X \in \{0,1\}^N, i \in [N]} [|p(X) - p(X^i)|] \leq N \cdot 2E \leq 4T.$$

■

We also need the following lemma of Beals et al. [6].

**Lemma 20 ([6])** *Suppose a quantum algorithm  $Q$  makes  $T$  queries to a Boolean input  $X \in \{0,1\}^N$ . Then  $Q$ 's acceptance probability is a real multilinear polynomial  $p(X)$ , of degree at most  $2T$ .*

We now prove our first consequence of Conjecture 6: namely, that it implies the folklore Conjecture 4.

**Theorem 21** *Suppose Conjecture 6 holds, and let  $\varepsilon, \delta > 0$ . Then given any quantum algorithm  $Q$  that makes  $T$  queries to a Boolean input  $X$ , there exists a deterministic classical algorithm that makes  $\text{poly}(T, 1/\varepsilon, 1/\delta)$  queries, and that approximates  $Q$ 's acceptance probability to within an additive constant  $\varepsilon$  on a  $1 - \delta$  fraction of inputs.*

**Proof.** Let  $p(X)$  be the probability that  $Q$  accepts input  $X = x_1 \dots x_N$ . Then Lemma 20 says that  $p$  is a real polynomial of degree at most  $2T$ . Assume Conjecture 6: that for every such  $p$ , there exists a variable  $i$  satisfying  $\text{Inf}_i[p] \geq q(\text{Vr}[p]/T)$ , for some fixed polynomial  $q$ . Under that assumption, we give a classical algorithm  $C$  that makes  $\text{poly}(T, 1/\varepsilon, 1/\delta)$  queries to the  $x_i$ 's, and that approximates  $p(X)$  on most inputs  $X$ .

```

set  $p_0 := p$ 
for  $j := 0, 1, 2, \dots$ :
  if  $\text{Vr}[p_j] \leq \varepsilon\delta$ 
    output  $p(X) \approx \mathbb{E}_{Y \in \{0,1\}^{N-j}} [p_j(Y)]$  and halt
  else
    find an  $i \in [N - j]$  such that  $\text{Inf}_i[p_j] > q(\varepsilon\delta/T)$ 
    query  $x_i$ , and let  $p_{j+1} : \mathbb{R}^{N-j} \rightarrow \mathbb{R}$  be the polynomial
      induced by the answer

```

If  $C$  halts, then by assumption  $\text{Vr}[p_i] \leq \varepsilon\delta$ . So clearly

$$\mathbb{E}_{X \in \{0,1\}^{N-j}} [|p_j(X) - \mathbb{E}[p_j]|] \leq \mathbb{E}_{X, Y \in \{0,1\}^{N-j}} [|p_j(X) - p_j(Y)|] \leq \varepsilon\delta$$

as well. By Markov's inequality, this implies

$$\Pr_{X \in \{0,1\}^{N-j}} [|p_j(X) - \mathbb{E}[p_j]| > \varepsilon] < \delta,$$

which proves  $C$ 's correctness.

On the other hand, suppose  $\text{Vr}[p_j] > \varepsilon\delta$ . Then by Conjecture 6, there exists an  $i$  such that

$$\text{Inf}_i[p_j] \geq q \left( \frac{\text{Vr}[p_j]}{T} \right) > q \left( \frac{\varepsilon\delta}{T} \right).$$

So if we query  $x_i$ , then we produce a new polynomial  $p_{j+1}$  such that

$$\text{SumInf}[p_{j+1}] < \text{SumInf}[p_j] - q \left( \frac{\varepsilon\delta}{T} \right).$$

But  $\text{SumInf}[p_0] = O(T)$  by Lemma 19. It follows that  $C$  halts after at most

$$\frac{\text{SumInf}[p_0]}{q(\varepsilon\delta/T)} \leq \left( \frac{T}{\varepsilon\delta} \right)^{O(1)}$$

iterations, each of which queries exactly one  $x_i$ . ■

An immediate corollary is the following:

**Corollary 22** *Suppose Conjecture 6 holds. Then  $D_{\varepsilon+\delta}(f) \leq (Q_\varepsilon(f)/\delta)^{O(1)}$  for all Boolean functions  $f$  and all  $\varepsilon, \delta > 0$ .*

**Proof.** Let  $Q$  be a quantum algorithm that evaluates  $f(X)$ , with bounded error, on a  $1-\varepsilon$  fraction of inputs  $X \in \{0,1\}^N$ . Let  $p(X) := \Pr[Q \text{ accepts } X]$ . Now run the classical simulation algorithm  $C$  from Theorem 21, to obtain an estimate  $\tilde{p}(X)$  of  $p(X)$  such that

$$\Pr_{X \in \{0,1\}^N} \left[ |\tilde{p}(X) - p(X)| \leq \frac{1}{10} \right] \geq 1 - \delta.$$

Output  $f(X) = 1$  if  $\tilde{p}(X) \geq \frac{1}{2}$  and  $f(X) = 0$  otherwise. By the theorem, this requires  $\text{poly}(T, 1/\delta)$  queries to  $X$ , and by the union bound it successfully computes  $f(X)$  on at least a  $1-\varepsilon-\delta$  fraction of inputs  $X$ . ■

We also get the following complexity consequence:

**Theorem 23** *Suppose Conjecture 6 holds. Then  $\text{P} = \text{P}^{\#\text{P}}$  implies  $\text{BQP}^A \subset \text{AvgP}^A$  with probability 1 for a random oracle  $A$ .*

**Proof.** Let  $Q$  be a polynomial-time quantum Turing machine that queries an oracle  $A$ , and assume  $Q$  decides some language  $L \in \text{BQP}^A$  with bounded error. Given an input  $x \in \{0,1\}^n$ , let  $p_x(A) := \Pr[Q^A(x) \text{ accepts}]$ . Then clearly  $p_x(A)$  depends only on some finite prefix  $B$  of  $A$ , of size  $N = 2^{\text{poly}(n)}$ . Furthermore, Lemma 20 implies that  $p_x$  is a polynomial in the bits of  $B$ , of degree at most  $\text{poly}(n)$ .

Assume Conjecture 6 as well as  $\text{P} = \text{P}^{\#\text{P}}$ . Then we claim that there exists a deterministic polynomial-time algorithm  $C$  such that for all  $Q$  and  $x \in \{0,1\}^n$ ,

$$\Pr_A \left[ |\tilde{p}_x(A) - p_x(A)| > \frac{1}{10} \right] < \frac{1}{n^3}, \tag{2}$$

where  $\tilde{p}_x(A)$  is the output of  $C$  given input  $x$  and oracle  $A$ . This  $C$  is essentially just the algorithm from Theorem 21. The key point is that we can implement  $C$  using not only  $\text{poly}(n)$  queries to  $A$ , but also  $\text{poly}(n)$  computation steps.

To prove the claim, let  $M$  be any of the  $2^{\text{poly}(n)}$  monomials in the polynomial  $p_j$  from Theorem 21, and let  $\alpha_M$  be the coefficient of  $M$ . Then notice that  $\alpha_M$  can be computed to  $\text{poly}(n)$  bits of precision in  $\mathbf{P}^{\#\mathbf{P}}$ , by the same techniques used to show  $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$  [8]. Therefore the expectation

$$\mathbb{E}_{Y \in \{0,1\}^{N-j}} [p_j(Y)] = \sum_M \frac{\alpha_M}{2^{|M|}}$$

can be computed in  $\mathbf{P}^{\#\mathbf{P}}$  as well. The other two quantities that arise in the algorithm— $\text{Vr}[p_j]$  and  $\text{Inf}_i[p_j]$ —can be computed in the second level of the counting hierarchy  $\mathbf{CH} = \mathbf{P}^{\#\mathbf{P}} \cup \mathbf{P}^{\#\mathbf{P}^{\#\mathbf{P}}} \cup \dots$  (since they involve an exponential sum inside of an absolute value sign, and another exponential sum outside of it). This means that finding an  $i$  such that  $\text{Inf}_i[p_j] > q(\varepsilon^2/T)$  is in the *third* level of the counting hierarchy. But under the assumption  $\mathbf{P} = \mathbf{P}^{\#\mathbf{P}}$ , the entire counting hierarchy collapses to  $\mathbf{P}$ . Therefore all of the computations needed to implement  $C$  take polynomial time.

Now let  $\delta_n(A)$  be the fraction of inputs  $x \in \{0,1\}^n$  such that  $|\tilde{p}_x(A) - p_x(A)| > \frac{1}{10}$ . Then by (2) together with Markov's inequality,

$$\Pr_A \left[ \delta_n(A) > \frac{1}{n} \right] < \frac{1}{n^2}.$$

Since  $\sum_{n=1}^{\infty} \frac{1}{n^2}$  converges, it follows that  $\delta_n(A) \leq \frac{1}{n}$  for all but finitely many values of  $n$ , with probability 1 over  $A$ . Assuming this occurs, we can simply hardwire the behavior of  $Q$  on the remaining  $n$ 's into our classical simulation procedure  $C$ . Hence  $L \in \text{AvgP}^A$ .

Since the number of  $\mathbf{BQP}^A$  languages is countable, the above implies that  $L \in \text{AvgP}^A$  for every  $L \in \mathbf{BQP}^A$  *simultaneously* (that is,  $\mathbf{BQP}^A \subset \text{AvgP}^A$ ) with probability 1 over  $A$ . ■

As a side note, suppose we had an extremely strong variant of Conjecture 6, one that implied something like

$$\Pr_A \left[ |\tilde{p}_x(A) - p_x(A)| > \frac{1}{10} \right] < \frac{1}{\exp(n)}.$$

in place of (2). Then we could eliminate the need for  $\text{AvgP}$  in Theorem 23, and show that  $\mathbf{P} = \mathbf{P}^{\#\mathbf{P}}$  implies  $\mathbf{P}^A = \mathbf{BQP}^A$  with probability 1 for a random oracle  $A$ .

We conclude this section with some unconditional results. These results will use Theorem 9 of Dinur et al. [13]: that for every degree- $d$  polynomial  $p : \mathbb{R}^N \rightarrow \mathbb{R}$  such that  $0 \leq p(X) \leq 1$  for all  $X \in \{0,1\}^N$ , there exists a polynomial  $\tilde{p}$  depending on at most  $2^{O(d)}/\varepsilon^2$  variables such that  $\|\tilde{p} - p\|_2^2 \leq \varepsilon$ .

Theorem 9 has the following simple corollary.

**Corollary 24** *Suppose a quantum algorithm  $Q$  makes  $T$  queries to a Boolean input  $X \in \{0,1\}^N$ . Then for all  $\alpha, \delta > 0$ , we can approximate  $Q$ 's acceptance probability to within an additive constant  $\alpha$ , on a  $1 - \delta$  fraction of inputs, by making  $\frac{2^{O(T)}}{\alpha^4 \delta^4}$  deterministic classical queries to  $X$ . (Indeed, the classical queries are nonadaptive.)*

**Proof.** Let  $p(X) := \Pr[Q \text{ accepts } X]$ . Then  $p$  is a degree- $2T$  real polynomial by Lemma 20. So by Theorem 9, there exists a polynomial  $\tilde{p}$ , depending on  $K = \frac{2^{O(T)}}{\alpha^4 \delta^4}$  variables  $x_{i_1}, \dots, x_{i_K}$ , such that

$$\mathbb{E}_{X \in \{0,1\}^N} \left[ (\tilde{p}(X) - p(X))^2 \right] \leq \alpha^2 \delta^2.$$

By Cauchy-Schwarz, then,

$$\mathbb{E}_{X \in \{0,1\}^N} [|\tilde{p}(X) - p(X)|] \leq \alpha \delta,$$

so by Markov's inequality

$$\Pr_{X \in \{0,1\}^N} [|\tilde{p}(X) - p(X)| > \alpha] < \delta.$$

Thus, our algorithm is simply to query  $x_{i_1}, \dots, x_{i_K}$ , and then output  $\tilde{p}(X)$  as our estimate for  $p(X)$ . ■

Likewise:

**Corollary 25**  $D_{\varepsilon+\delta}(f) \leq 2^{O(Q_\varepsilon(f))}/\delta^4$  for all Boolean functions  $f$  and all  $\varepsilon, \delta > 0$ .

**Proof.** Set  $\alpha$  to any constant less than  $1/6$ , then use the algorithm of Corollary 24 to simulate the  $\varepsilon$ -approximate quantum algorithm for  $f$ . Output  $f(X) = 1$  if  $\tilde{p}(X) \geq \frac{1}{2}$  and  $f(X) = 0$  otherwise. ■

Given an oracle  $A$ , let  $\text{BQP}^{A[\log]}$  be the class of languages decidable by a BQP machine able to make  $O(\log n)$  queries to  $A$ . Also, let  $\text{AvgP}_{\parallel}^A$  be the class of languages decidable, with probability  $1 - o(1)$  over  $x \in \{0,1\}^n$ , by a P machine able to make  $\text{poly}(n)$  parallel (nonadaptive) queries to  $A$ . Then we get the following unconditional variant of Theorem 23.

**Theorem 26** Suppose  $P = P^{\#P}$ . Then  $\text{BQP}^{A[\log]} \subset \text{AvgP}_{\parallel}^A$  with probability 1 for a random oracle  $A$ .

**Proof.** The proof is essentially the same as that of Theorem 23, except that we use Corollary 24 in place of Conjecture 6. In the proof of Corollary 24, observe that the condition

$$\mathbb{E}_{X \in \{0,1\}^N} [|\tilde{p}(X) - p(X)|] \leq \alpha\delta$$

implies

$$\mathbb{E}_{X \in \{0,1\}^N} [|p_\mu(X) - p(X)|] \leq \alpha\delta \tag{3}$$

as well, where  $p_\mu(X)$  equals the mean of  $p(Y)$  over all inputs  $Y$  that agree with  $X$  on  $x_{i_1}, \dots, x_{i_K}$ . Thus, given a quantum algorithm that makes  $T$  queries to an oracle string, the computational problem that we need to solve boils down to finding a subset of the oracle bits  $x_{i_1}, \dots, x_{i_K}$  such that  $K = \frac{2^{O(T)}}{\alpha^4 \delta^4}$  and (3) holds. Just like in Theorem 23, this problem is solvable in the counting hierarchy  $\text{CH} = P^{\#P} \cup P^{\#P^{\#P}} \cup \dots$ . So if we assume  $P = P^{\#P}$ , it is also solvable in P.

In Theorem 23, the conclusion we got was  $\text{BQP}^A \subset \text{AvgP}^A$  with probability 1 for a random oracle  $A$ . In our case, the number of classical queries  $K$  is exponential (rather than polynomial) in the number of quantum queries  $T$ , so we only get  $\text{BQP}^{A[\log]} \subset \text{AvgP}^A$ . On the other hand, since the classical queries are nonadaptive, we can strengthen the conclusion to  $\text{BQP}^{A[\log]} \subset \text{AvgP}_{\parallel}^A$ . ■

## 4 Open Problems

It would be nice to improve the  $R(f) = O(Q(f)^9 \text{polylog } Q(f))$  bound for all symmetric problems. As mentioned earlier, we conjecture that the right answer is  $R(f) = O(Q(f)^2)$ . Note that if one could tighten Midrijanis's quantum lower bound for SET EQUALITY [18] from  $\Omega((N/\log N)^{1/5})$  to  $\Omega(N^{1/3})$ , then an improvement to  $R(f) = O(Q(f)^7 \text{polylog } Q(f))$  would follow immediately. However, it seems better to avoid using SET EQUALITY altogether. After all, it is a curious feature of our proof that, to get a lower bound for all symmetric problems, we need to reduce from the

*non*-symmetric SET EQUALITY problem, which in turn is lower-bounded by a reduction from the symmetric collision problem!

We also conjecture that  $R(f) \leq Q(f)^{O(1)}$  for all partial functions  $f$  that are symmetric *only* under permuting the inputs (and not necessarily the outputs). Proving this seems to require a new approach.

It would be interesting to reprove the  $R(f) \leq Q(f)^{O(1)}$  bound using only the polynomial method, and not the adversary method. Or, to rephrase this as a purely classical question: for all  $X = (x_1, \dots, x_N)$  in  $[M]^N$ , let  $B_X$  be the  $N \times M$  matrix whose  $(i, j)^{th}$  entry is 1 if  $x_i = j$  and 0 otherwise. Then given a set  $S \subseteq [M]^N$  and a function  $f : S \rightarrow \{0, 1\}$ , let  $\widetilde{\deg}(f)$  be the minimum degree of a real polynomial  $p : \mathbb{R}^{MN} \rightarrow \mathbb{R}$  such that

- (i)  $0 \leq p(B_X) \leq 1$  for all  $X \in [M]^N$ , and
- (ii)  $|p(B_X) - f(X)| \leq \frac{1}{3}$  for all  $X \in S$ .

Then is it the case that  $R(f) \leq \widetilde{\deg}(f)^{O(1)}$  for all permutation-invariant functions  $f$ ?

On the random oracle side, the obvious problem is to prove Conjecture 6—thereby establishing that  $D_\varepsilon(f)$  and  $Q_\delta(f)$  are polynomially related, and all the other consequences shown in Section 3. Alternatively, one could look for some technique that was tailored to polynomials  $p$  that arise as the acceptance probabilities of quantum algorithms. In this way, one could conceivably solve  $D_\varepsilon(f)$  versus  $Q_\delta(f)$  and the other quantum problems, without settling the general conjecture about bounded polynomials.

## 5 Acknowledgments

We thank Andy Drucker, Ryan O’Donnell, and Ronald de Wolf for helpful discussions.

## References

- [1] S. Aaronson. Quantum lower bound for the collision problem. In *Proc. ACM STOC*, pages 635–642, 2002. quant-ph/0111102.
- [2] S. Aaronson. BQP and the polynomial hierarchy. arXiv:0910.4698, 2009.
- [3] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004.
- [4] D. Aharonov, V. Jones, and Z. Landau. A polynomial quantum algorithm for approximating the Jones polynomial. In *Proc. ACM STOC*, pages 427–436, 2006. quant-ph/0511096.
- [5] A. Ambainis. Quantum lower bounds by quantum arguments. *J. Comput. Sys. Sci.*, 64:750–767, 2002. Earlier version in ACM STOC 2000. quant-ph/0002066.
- [6] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. Earlier version in IEEE FOCS 1998, pp. 352–361. quant-ph/9802049.
- [7] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. quant-ph/9701001.

- [8] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. First appeared in ACM STOC 1993.
- [9] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In S. J. Lomonaco and H. E. Brandt, editors, *Quantum Computation and Information*, Contemporary Mathematics Series. AMS, 2002. quant-ph/0005055.
- [10] G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *ACM SIGACT News*, 28:14–19, 1997. quant-ph/9705002.
- [11] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Comput. Sci.*, 288:21–43, 2002.
- [12] W. van Dam, S. Hallgren, and L. Ip. Quantum algorithms for some hidden shift problems. *SIAM J. Comput.*, 36(3):763–778, 2006. Conference version in SODA 2003. quant-ph/0211140.
- [13] I. Dinur, E. Friedgut, G. Kindler, and R. O’Donnell. On the Fourier tails of bounded functions over the discrete cube. In *Proc. ACM STOC*, pages 437–446, 2006.
- [14] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *J. Comput. Sys. Sci.*, 59(2):240–252, 1999. cs.CC/9811023.
- [15] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. ACM STOC*, pages 212–219, 1996. quant-ph/9605043.
- [16] A. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations. *Phys. Rev. Lett.*, 15(150502), 2009. arXiv:0811.3171.
- [17] J. Kahn, M. Saks, and C. Smyth. A dual version of Reimer’s inequality and a proof of Rudich’s conjecture. In *Proc. IEEE Conference on Computational Complexity*, pages 98–103, 2000.
- [18] G. Midrijanis. A polynomial quantum query lower bound for the set equality problem. In *Proc. Intl. Colloquium on Automata, Languages, and Programming (ICALP)*, pages 996–1005, 2004. quant-ph/0401073.
- [19] R. O’Donnell, M. E. Saks, O. Schramm, and R. A. Servedio. Every decision tree has an influential variable. In *Proc. IEEE FOCS*, pages 31–39, 2005.
- [20] R. Paturi. On the degree of polynomials that approximate symmetric Boolean functions. In *Proc. ACM STOC*, pages 468–474, 1992.
- [21] Y. Shi. Lower bounds of quantum black-box complexity and degree of approximating polynomials by influence of Boolean variables. *Inform. Proc. Lett.*, 75(1-2):79–83, 2000. quant-ph/9904107.
- [22] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Earlier version in IEEE FOCS 1994. quant-ph/9508027.
- [23] D. Simon. On the power of quantum computation. In *Proc. IEEE FOCS*, pages 116–123, 1994.
- [24] C. D. Smyth. Reimer’s inequality and Tardos’ conjecture. In *Proc. ACM STOC*, pages 218–221, 2002.

## 6 Appendix: The Boolean Case

Given a partial Boolean function  $f : \{0, 1\}^N \rightarrow \{0, 1, *\}$ , call  $f$  *symmetric* if  $f(X)$  depends only on the Hamming weight  $|X| := x_1 + \dots + x_N$ . For completeness, in this appendix we prove the following basic fact:

**Theorem 27**  $R(f) = O(Q(f)^2)$  for every partial symmetric Boolean function  $f$ .

For *total* symmetric Boolean functions, Theorem 27 was already shown by Beals et al. [6], using an approximation theory result of Paturi [20]. Indeed, in the total case one even has  $D(f) = O(Q(f)^2)$ . So the new twist is just that  $f$  can be partial.

Abusing notation, let  $f(k) \in \{0, 1, *\}$  be the value of  $f$  on all inputs of Hamming weight  $k$  (where as usual,  $*$  means ‘undefined’). Then we have the following quantum lower bound:

**Lemma 28** Suppose that  $f(a) = 0$  and  $f(b) = 1$  or vice versa, where  $a < b$  and  $a \leq N/2$ . Then  $Q(f) = \Omega\left(\frac{\sqrt{bN}}{b-a}\right)$ .

**Proof.** This follows from a straightforward application of Ambainis’s adversary theorem (Theorem 15). Specifically, let  $A, B \subseteq \{0, 1\}^N$  be the sets of all strings of Hamming weights  $a$  and  $b$  respectively, and for all  $X \in A$  and  $Y \in B$ , put  $(X, Y) \in R$  if and only if  $X \preceq Y$  (that is,  $x_i \leq y_i$  for all  $i \in [N]$ ). Then

$$Q(f) = \Omega\left(\sqrt{\frac{N-a}{b-a} \cdot \frac{b}{b-a}}\right) = \Omega\left(\frac{\sqrt{bN}}{b-a}\right).$$

Alternatively, this lemma can be proved using the approximation theory result of Paturi [20], following Beals et al. [6]. ■

In particular, if we set  $\beta := \frac{b}{N}$  and  $\varepsilon := \frac{b-a}{N}$ , then  $Q(f) = \Omega(\sqrt{\beta/\varepsilon})$ . On the other hand, we also have the following randomized *upper* bound, which follows from a Chernoff bound (similar to Lemma 11):

**Lemma 29** Assume  $\beta > \varepsilon > 0$ . By making  $O(\beta/\varepsilon^2)$  queries to an  $N$ -bit string  $X$ , a classical sampling algorithm can estimate the fraction  $\beta := |X|/N$  of 1 bits to within an additive error  $\pm\varepsilon/3$ , with success probability at least (say)  $2/3$ .

So assume the function  $f$  is non-constant, and let

$$\gamma := \max_{f(a)=0, f(b)=1} \frac{\sqrt{bN}}{b-a}. \quad (4)$$

Assume without loss of generality that the maximum of (4) is achieved when  $a < b$  and  $a \leq N/2$ , if necessary by applying the transformations  $f(X) \rightarrow 1 - f(X)$  and  $f(X) \rightarrow f(N - X)$ . Now consider the following randomized algorithm to evaluate  $f$ , which makes  $T := O(\gamma^2)$  queries:

Choose indices  $i_1, \dots, i_T \in [N]$  uniformly at random with replacement  
 Query  $x_{i_1}, \dots, x_{i_T}$   
 Set  $k := \frac{N}{T}(x_{i_1} + \dots + x_{i_T})$   
 If there exists a  $b \in \{0, \dots, N\}$  such that  $f(b) = 1$  and  $|k - b| \leq \frac{\sqrt{bN}}{3\gamma}$   
 output  $f(X) = 1$

Otherwise output  $f(X) = 0$

By Lemma 29, the above algorithm succeeds with probability at least  $2/3$ , provided we choose  $T$  suitably large. Hence  $R(f) = O(\gamma^2)$ . On the other hand, Lemma 28 implies that  $Q(f) = \Omega(\gamma)$ . Hence  $R(f) = O(Q(f)^2)$ , completing the proof of Theorem 27.