Approximate degree lower bounds for oracle identification problems

Mark Bun* Nadezhda Voronova†

March 2023

Abstract

The approximate degree of a Boolean function is the minimum degree of real polynomial that approximates it pointwise. For any Boolean function, its approximate degree serves as a lower bound on its quantum query complexity, and generically lifts to a quantum communication lower bound for a related function.

We introduce a framework for proving approximate degree lower bounds for certain oracle identification problems, where the goal is to recover a hidden binary string $x \in \{0,1\}^n$ given possibly non-standard oracle access to it. Our lower bounds apply to decision versions of these problems, where the goal is to compute the parity of x. We apply our framework to the ordered search and hidden string problems, proving nearly tight approximate degree lower bounds of $\Omega(n/\log^2 n)$ for each. These lower bounds generalize to the weakly unbounded error setting, giving a new quantum query lower bound for the hidden string problem in this regime. Our lower bounds are driven by randomized communication *upper bounds* for the greater-than and equality functions.

1 Introduction

In an *oracle identification* problem, there is an unknown string $x \in \{0,1\}^n$. A query algorithm is given possibly non-standard oracle access to x, and its goal is to reconstruct x by making a minimal number of queries to this oracle. More specifically, an oracle identification problem is specified by a fixed family of Boolean functions $a_1, \ldots, a_N$. A query algorithm may inspect any value $a_i(x)$ of its choice at the cost of one query, and its goal is to determine x. Many influential problems in the study of quantum algorithms and complexity can be viewed as oracle identification problems, including van Dam's original oracle interrogation problem [vD98], the Bernstein-Vazirani problem [BV93], combinatorial group testing [AM14, Bel15], symmetric junta learning [Bel15], and more [BdW99, AIK$^+$04, AIK$^+$07, INRT12, CIG$^+$12, Kot14]. In this work, we study two such oracle identification problems:

Ordered Search. Consider the following abstraction of the problem of searching an ordered list of $N = 2^n$ elements. Given a list of N bits $a_i \in \{0,1\}$ under the promise that $a_0 \le a_1 \le \cdots \le a_{N-1}$, find the (binary encoding of the) minimum index $x \in \{0,1\}^n$ such that $a_x = 1$. Binary search

*Department of Computer Science, Boston University, Boston, MA 02215, USA. mbun@bu.edu. Supported in part by NSF awards CCF-1947889 and CNS-2046425 and a Sloan Research Fellowship.

†Department of Computer Science, Boston University, Boston, MA 02215, USA. voronova@bu.edu. Supported in part by NSF awards CCF-1947889 and CNS-2046425.

yields a deterministic algorithm making n queries, and it is not hard to see that this is optimal for randomized algorithms as well. As for quantum algorithms, it turns out that a constant-factor speedup is possible [FGGS99, CLP07, BH08], but a lower bound of $\Omega(n)$ holds in this model as well [BdW99, FGGS98, Amb99, HNS02, CL08]. Ordered search may be viewed as an oracle identification problem where the query algorithm is given oracle access to $a_0 = \mathsf{GT}_0(x), \ldots, a_{N-1} = \mathsf{GT}_{N-1}(x)$, where each "greater-than" function $\mathsf{GT}_i(x)$ evaluates to 1 if $i \geq x$ and to 0 otherwise.

Hidden String. In the hidden string problem, the goal is to reconstruct a hidden string $x \in \{0,1\}^n$ given information about the presence of absence of potential substrings of x. That is, the goal is to determine x given "substring oracle" access, i.e., oracle access to $a_s = \phi_s(x)$ for every binary string s of length at most n, where $\phi_s(x)$ evaluates to 1 iff s is a substring of x. Building on a classical query algorithm of Skiena and Sundaram [SS95], Cleve et al. [CIG$^+$12] gave a $3n/4 + o(n)$ quantum query algorithm for this problem, and proved a nearly matching quantum query lower bound of $\Omega(n/\log^2 n)$.

The state-of-the-art quantum query lower bounds for both problems are proved via the quantum adversary method, which in its modern formulation [HLS07a], characterizes the bounded-error quantum query complexity of every function up to a constant factor [Rei11]. The other major technique for proving quantum query lower bounds is the polynomial method [BBC$^+$01], which lower bounds the quantum query complexity of a function by lower bounding its *approximate degree*. The approximate degree of a Boolean function is the least degree of a real polynomial that approximates it pointwise to error 1/3. Since the acceptance probability of a T-query quantum algorithm is a polynomial of degree $2T$, the approximate degree of a function is always at most (half of its) quantum query complexity, but it can be much smaller [Amb06, ABK16, She20, BKT20].

In this work, we prove lower bounds of $\Omega(n/\log^2 n)$ on the approximate degree of (decision variants) of the ordered search and hidden string problems. These lower bounds are nearly optimal, as the known quantum (indeed, even classical) query algorithms for these problems automatically yield $O(n)$ upper bounds on their approximate degree. For the ordered search problem, Childs and Lee [CL08] explicitly posed the question of investigating approximate degree lower bounds to circumvent limitations of the adversary method. Meanwhile, our lower bound on the approximate degree of the hidden string problem implies a quantum query lower bound matching the state-of-the-art [CIG$^+$12].

Approximate degree is a fundamental measure of the complexity of Boolean functions that has been the subject of extensive study in its own right (see, e.g., [BT22] for a recent survey). And while nearly tight quantum query lower bounds for these problems were already known, we see two main quantum motivations for recovering these bounds via approximate degree. First, there are senses in which approximate degree is a more robust lower bound technique than the adversary method. For example, via Sherstov's pattern matrix method [She11], any approximate degree lower bound for a Boolean function f can be "lifted" to give the same quantum communication lower bound for a related two-party function F. Such a generic lifting result is not known for any other general quantum query lower bound technique. Moreover, variants of the polynomial method are capable of proving lower bounds against zero-, small-, and unbounded-error quantum algorithms [BBC$^+$01, BCdWZ99], as well as time-space tradeoffs [KSdW07]. Indeed, using the polynomial method, we give weakly-unbounded-error quantum query lower bounds for the hidden string problem (see Corollary 2) that significantly improve over the lower bound implied by the adversary method [CIG$^+$12].

Second, we believe that our approximate degree lower bounds shed additional light on what makes the ordered search and hidden string problems hard, and may be more transparent in this regard than existing adversary lower bounds. In particular, our lower bounds show that it is not only hard for quantum algorithms to reconstruct the hidden string x, but even to simply compute its parity (a decision problem). The other nearly tight lower bounds for the problems we consider appear to make essential use of the fact that the query algorithm needs to reconstruct all of x, and it isn't clear (at least to us) how to adapt them to hold for their decision variants. We believe that the technique we introduce, or at the very least the "indirect" method we use to prove our lower bounds, will be more broadly useful in understanding the approximate degree and quantum query complexity of other oracle identification problems.

1.1 Techniques

Here we give a brief summary of the ideas behind our lower bound for ordered search. A more detailed technical overview, including a discussion of how we apply our framework to the hidden string problem, appears in Section 2. Full proofs appear in Sections 3 and 4.

The first lower bound for quantum ordered search was given by Buhrman and de Wolf [BdW99], who actually showed an $\Omega(\sqrt{n})$ lower bound on its approximate degree. The starting point for the proof of our lower bound is their ingenious indirect argument, so let us review it here. Recall that the ability to solve ordered search on inputs $a_0 \leq a_1 \leq \cdots \leq a_{N-1}$ enables recovering the string $x \in \{0,1\}^n$, where $N = 2^n$, for which every $a_i = \mathsf{GT}_i(x)$. This, in particular, enables the evaluation of any "hard" Boolean function of x, e.g., its parity. In light of this, define the partial Boolean function $\mathsf{OS}_N(a_0, \ldots, a_{N-1}) := \mathsf{parity}(x)$ whenever there exists an x for which $a_i = \mathsf{GT}_i(x)$ for every i. Let $p : \{0,1\}^N \to \mathbb{R}$ be a polynomial of degree d approximating OS_N. It is known that every polynomial approximating parity must have degree $\Omega(n)$, so the goal now is to use this fact to prove a lower bound on the degree of p. To do so, we use the additional fact that the functions GT_i can each be approximated by a degree $O(\sqrt{n})$ polynomial q_i arising from, say, a variant of Grover search. By making p "robust to noise" in its input without increasing its degree [BNRdW07, She12a], we get that the composed polynomial $p(q_0(x), \ldots, q_{N-1}(x)) \approx \mathsf{parity}(x)$ and has degree $O(d\sqrt{n})$. Now the fact that the approximate degree of parity is $\Omega(n)$ implies that $d = \Omega(\sqrt{n})$.

In summary, the lower bound for OS_N follows from the fact that we can express the function $\mathsf{parity}(x) = \mathsf{OS}_N(\mathsf{GT}_0(x), \ldots, \mathsf{GT}_{N-1}(x))$, where we have a lower bound on the approximate degree of parity and an *upper bound* on the approximate degree of GT. However, the lower bound gets stuck at degree $\Omega(\sqrt{n})$ because the functions GT_i themselves require nontrivial degree $O(\sqrt{n})$ to approximate, and this is tight.

To get an improved lower bound of $\widetilde{\Omega}(n)$ on the approximate degree of OS_N, we introduce the following idea to make GT behave as if it were easier to approximate by low degree polynomials, while preserving the hardness of parity. Given an input $x \in \{0,1\}^n$, we redundantly encode x as a longer string $\mathcal{Y}(x) \in \{0,1\}^m$ for some $m = \mathrm{poly}(n)$. This encoding is chosen so that

- Access to $\mathcal{Y}(x)$ instead of just x itself makes each function $\mathsf{GT}_i(x)$ approximable by a much lower degree polynomial. That is, for every i, there exists a polynomial q_i of degree $\mathrm{polylog}(n)$ such that $q_i(\mathcal{Y}(x)) \approx \mathsf{GT}_i(x)$ for every x.
- Even with access to $\mathcal{Y}(x)$, the function $\mathsf{parity}(x)$ remains hard to approximate. That is, for every polynomial p of degree at most $n/\,\mathrm{polylog}(n)$, we have that $p(\mathcal{Y}(x))$ fails to approximate $\mathsf{parity}(x)$.

We can now obtain our improved lower bound by applying Buhrman and de Wolf's argument to the redundantly encoded inputs. Specifically, given a robust polynomial $p : \{0,1\}^N \to \mathbb{R}$ of degree d approximating OS_N, we would have $p(q_0(\mathcal{Y}(x)), \ldots, q_{N-1}(\mathcal{Y}(x))) \approx \mathsf{OS}_N(\mathsf{GT}_0(x), \ldots, \mathsf{GT}_{N-1}(x)) = \mathsf{parity}(x)$ for every x. Our upper bound on the degrees of the q_i's, together with our lower bound on the degree needed to approximate parity, imply that $d \operatorname{polylog} n \geq n/\operatorname{polylog} n$, and hence $d \geq \widetilde{\Omega}(n)$.

All that remains is to construct the appropriate encoding $\mathcal{Y}$. Our approach is inspired by Nisan's classic randomized *communication* protocol for computing the two-party greater-than function. The most helpful way to think about this protocol for our purposes is as follows. Suppose Alice and Bob hold strings $a, b \in \{0,1\}^n$ and their goal is to determine whether the natural number represented by a is at least that represented by b. They may do so by performing binary search to identify the minimum index j for which $a_j \neq b_j$, at which point the answer is determined by which of a_j or b_j is 1. Each step of this binary search can be conducted by testing the equality of a substring of a with a substring of b. Each equality test, in turn, may be performed (with high success probability) by comparing the inner products of a and b with a shared random string. The protocol requires $\log n$ steps of binary search, and each equality test should be repeated $O(\log \log n)$ times to achieve high success probability, giving an overall communication cost of $\widetilde{O}(\log n)$.

Now let us see how to turn this communication protocol into a polynomial approximating $\mathsf{GT}_i(x)$. Think of x as Bob's input to the communication protocol, and of Bob's role as passively computing an encoding $\mathcal{Y}(x)$ that consists of many inner products of x with random strings. Now thinking of i as Alice's input, she can compute $\mathsf{GT}_i(x)$ (with high probability) by repeatedly querying $\mathcal{Y}(x)$ at the locations that correspond to the appropriate inner products from the protocol described above. This results in a $\widetilde{O}(\log n)$ randomized query algorithm for computing $\mathsf{GT}_i(x)$ from $\mathcal{Y}(x)$, the success probability of which is a degree-$\widetilde{O}(\log n)$ polynomial in $\mathcal{Y}(x)$.

The final step is to argue that even given $\mathcal{Y}(x)$, consisting of many inner products of random strings with x, the parity function $\mathsf{parity}(x)$ remains hard to compute. To see why this is true, note that a single inner product of x with a random bit string is itself a parity on a random subset of indices. That is, $\mathcal{Y}(x) = (\mathsf{parity}(x|_{S_1}), \ldots, \mathsf{parity}(x|_{S_m}))$ for random subsets $S_1, \ldots, S_m \subseteq [n]$. The key observation then, is that a degree-d polynomial of these random parities is able to approximate the full $\mathsf{parity}(x)$ if and only if some degree-d polynomial of these random parities *exactly* computes $\mathsf{parity}(x)$, which in turn happens if and only if a symmetric difference of at most d of the sets $S_1, \ldots, S_m$ yields the entire set of indices $[n]$. As a result, as long as neither the degree d nor the number of random inner products m is too large, we obtain that $\mathsf{parity}(x)$ cannot be approximated using $\mathcal{Y}(x)$.[1]

1.2 Our results in detail

Recall that we introduce a framework that allows us to prove lower bounds on approximate degree, and hence quantum query complexity. It most naturally applies to decision versions of oracle identification problems, and extends to the "weakly unbounded error" setting of error approaching $1/2$. We summarize the results we prove using this framework in Table 1.

Ordered search. As mentioned, binary search yields a deterministic algorithm making n queries, which in turn yields a polynomial of degree n that exactly computes OS_{2^n}. To compute this function

[1] In fact, this argument shows that it is impossible to approximate $\mathsf{parity}(x)$ to bounded error, but even to represent it in sign. This corresponds to a *threshold degree* lower bound.

Problem	Model	Error	Previous work	This work
Ordered search	Approximate degree and quantum query complexity, decision version	Unbounded	$O(n - \log \frac{1}{\gamma})$, $\Omega(\sqrt{n} - \log \frac{1}{\gamma})$	$\Omega(\frac{n}{\log^2 n} - \log \frac{1}{\gamma})$
		Constant	$O(n)$, $\Omega(\sqrt{n})$	$\Omega(\frac{n}{\log^2 n})$
	Quantum query complexity, reconstruction version	Unbounded	$\Theta(n - \log \frac{1}{\gamma})$	$\Omega(\frac{n}{\log^2 n} - \log \frac{1}{\gamma})$
		Constant	$\Theta(n)$	$\Omega(\frac{n}{\log^2 n})$
Hidden string	Approximate degree and quantum query complexity, decision version	Unbounded	$O(n - \log \frac{1}{\gamma})$	$\Omega(\frac{n}{\log^2 n} - \log \frac{1}{\gamma})$
		Constant	$O(n)$	$\Omega(\frac{n}{\log^2 n})$
	Quantum query complexity, reconstruction version	Unbounded	$O(n - \log \frac{1}{\gamma})$, $\Omega(\gamma^2 \frac{n}{\log^2 n})$	$\Omega(\frac{n}{\log^2 n} - \log \frac{1}{\gamma})$
		Constant	$O(n)$	$\Omega(\frac{n}{\log^2 n})$

Table 1: Summary of our results and prior work.

with error probability $\frac{1}{2} - \gamma$ for some parameter $\gamma > 0$, there is an easy way to modify binary search to obtain an $O(n - \log \frac{1}{\gamma})$-query randomized algorithm (see Appendix A for details). This implies an upper bound of $O(n - \log \frac{1}{\gamma})$ on the approximate degree of OS_{2^n} with error parameter $1/2 - \gamma$.

Before this work, the best lower bound on approximate degree (for both bounded and unbounded error) was obtained by [BdW99] and was $\Omega(\sqrt{n} - \log \frac{1}{\gamma})$ for approximation to error $\frac{1}{2} - \gamma$. We significantly improve their result and obtain the following lower bound.

Theorem. For every natural number n and $0 < \gamma < 1/2$, every polynomial that approximates OS_{2^n} pointwise to error $\frac{1}{2} - \gamma$ requires degree

$$\Omega\left(\frac{n}{\log^2 n} - \log \frac{1}{\gamma}\right).$$

This result is restated as Theorem 12. It shows that it is hard to approximate the decision version of the ordered search problem OS_{2^n} (with parity as the predicate converting from reconstruction to decision problem) not only to constant error, but even to small advantage γ over random guessing. For instance, approximating OS_{2^n} with advantage $\gamma = 2^{-n^{0.99}}$ still requires degree $\Omega(\frac{n}{\log^2 n})$. Our lower bound is nearly tight in both the bounded and unbounded error regimes.

Query complexity of ordered search. Most previous work on the quantum query complexity of ordered search addressed the bounded error regime and the reconstruction version of the problem, where the goal is to output the entire string x, rather than a specific Boolean predicate applied to x. To our knowledge, the best prior lower bound for the decision version of ordered search with unbounded error follows from [BdW99] as described above and is $\Omega(\sqrt{n} - \log \frac{1}{\gamma})$. Note also that the $\Omega(n)$ lower bound of [Amb99], stated there for constant error, also generalizes to a tight lower bound $\Omega(n - \log \frac{1}{\gamma})$ for unbounded error, but it appears to hold only for the reconstruction version of ordered search.

Our application of the polynomial method implies a nearly tight quantum query lower bound that applies to the decision version of the problem.

Corollary 1. Every quantum algorithm that computes OS_{2^n} (decision version with parity) with probability of error at most $\frac{1}{2}-\gamma$ requires $\Omega(\frac{n}{\log^2 n}-\log\frac{1}{\gamma})$ queries.

Hidden string. The work of [SS95] yields a simple deterministic algorithm making $O(n)$ queries, which in turn yields a polynomial of degree $O(n)$ that exactly computes $\mathsf{HS}_{2^{n+1}-1}(\dots,\phi_s(x),\dots) := \mathsf{parity}(x)$ where $x\in\{0,1\}^n$ is the hidden string in question. Again, this algorithm can be modified to get a $O(n-\log\frac{1}{\gamma})$-query algorithm with error $1/2-\gamma$ (see Appendix A for details). This implies an upper bound $O(n-\log\frac{1}{\gamma})$ on the approximate degree of $\mathsf{HS}_{2^{n+1}-1}$.

We give the first lower bound on the approximate degree of the hidden string problem:

Theorem. For every natural number n and $0<\gamma<1/2$, every polynomial that approximates $\mathsf{HS}_{2^{n+1}-1}$ to error $\frac{1}{2}-\gamma$ requires degree

$$\Omega\left(\frac{n}{\log^2 n}-\log\frac{1}{\gamma}\right).$$

This result is restated as Corollary 21, and gives a nearly tight lower bound for approximating the decision version of $\mathsf{HS}_{2^{n+1}-1}$ to both constant and weakly unbounded error.

Query complexity of hidden string. Complementing the $O(n)$-query deterministic algorithm of [SS95], it turns out that a constant-factor speedup is possible for quantum algorithms [CIG+12]. As for lower bounds, the latter work shows a lower bound $\Omega(\frac{n}{\log^2 n})$ on reconstruction by adversary method. This lower bounds holds for bounded error, but does not generalize well to unbounded error regime. (By [BSS01, HLS07b], the same proof implies a lower bound of $\Omega(\gamma^2\frac{n}{\log^2 n})$ for solving the reconstruction version of hidden string with error $\frac{1}{2}-\gamma$.)

Our approximate degree lower bound recovers their lower bound for bounded error, and gives a significantly stronger lower bound for the weakly unbounded error regime, both for the decision version of the problem.

Corollary 2. Every quantum algorithm that computes $\mathsf{HS}_{2^{n+1}-1}$ (decision version with parity) with probability of error at most $\frac{1}{2}-\gamma$ requires $\Omega(\frac{n}{\log^2 n}-\log\frac{1}{\gamma})$ queries.

1.3 Further discussion

One of our initial motivations for studying the approximate degree of ordered search came from the preliminary version of Chattopadhyay et al. [CKLM17]. They showed that $\mathsf{OS}_N\circ\mathsf{IP}_m^N$ has randomized communication complexity $\Omega(\log N\cdot m)$, where IP_m is a two-party inner product (mod 2) gadget on m-bit inputs. This was done via an involved simulation argument, showing how a communication protocol for $\mathsf{OS}_N\circ\mathsf{IP}_m^N$ could be used to construct a randomized decision tree for OS_N. The techniques were specialized to the both the outer function and the inner function. Subsequent work [CFK+21] recovered this result using a generic simulation theorem. A direct application of Sherstov's pattern matrix method [She11] to our result yields a *quantum* communication lower bound of $\Omega(\log N/\log^2\log N)$ on $\mathsf{OS}_N\circ g^N$ even for a constant-sized gadget g.

Hoza [Hoz17] used ideas conceptually related to ours to nearly recover the known quantum query (but not approximate degree) lower bound for ordered search. Roughly, he used a Holevo-information argument to show that if an oracle identification problem specified by functions $a_1,\dots,a_N$

can be solved with T quantum queries, then $Q^*(A) \cdot T \gtrsim n$, where $A(i,x) = a_i(x)$ and Q^* is the bounded-error two-party quantum communication complexity with shared entanglement. His quantum query lower bound for ordered search follows directly from the fact that the quantum communication complexity of the two-party greater-than function GT on n-bit inputs is $O(\log n)$. However, without opening up the communication protocol for GT as we do, it is not clear how to recover an approximate degree lower bound from his construction.

The idea of indirectly proving approximate degree lower bounds by combining a lower bound for one problem with an upper bound for another also appears in [BBGK18]. They gave a tight lower bound on the approximate degree of any function of the form $f \circ g^n$ where f is an n-input symmetric function by combining a known lower bound for $\mathsf{parity} \circ g^n$ [She12b] with a quantum query and approximate degree upper bound for the combinatorial group testing problem [Bel15].

We believe it should be possible to extend our techniques to prove new lower bounds for other oracle identification problems. A family of special cases of oracle identification is captured by the symmetric junta learning problem [AM14]. Here, there is a symmetric function $h : \{0,1\}^k \to \{0,1\}$ and each f_S takes the form $f_S(x) = h(x|_S)$. An important instance of this problem is the combinatorial group testing problem, wherein one takes $h = \mathsf{OR}_k$. Belovs gave a tight upper bound of $O(\sqrt{k})$ [Bel15] for this problem. He also determined the query complexity for $h = \mathsf{EXACT-HALF}$ to be $\Theta(k^{1/4})$ and gave an upper bound of $O(k^{1/4})$ for $h = \mathsf{MAJ}$. These upper bounds were also (nearly) recovered algorithmically by Montanaro and Shao [MS20]. Despite its similarity to $\mathsf{EXACT} - \mathsf{HALF}$, no polynomial lower bound is known for the majority function MAJ.

In the counterfeit coin problem, there is a hidden string $x \in \{-1,1\}^n$ with Hamming weight at most k. A query is parameterized by a balanced (i.e., having an equal number of 1's and -1's) string $y \in \{-1,0,1\}^n$, and indicates whether $\langle x, y\rangle$ is zero or non-zero. Iwama et al. [INRT12] gave a quantum algorithm making $O(k^{1/4})$ queries and conjectured this is tight, but no lower bound is known. Note that the oracle here is quite similar to the $\mathsf{EXACT} - \mathsf{HALF}$ oracle.

2 Technical ideas

2.1 Our lower bound framework

We begin with a somewhat more abstract description of our framework for proving approximate degree lower bounds for oracle identification problems. The main idea is to provide additional information about the hidden input to an oracle identification problem so as to selectively affect the ability of quantum query algorithms and approximating polynomials to compute the functions we wish to understand.

Recall that an oracle identification problem is specified by a family of functions $a_1, \ldots, a_N$. Given query access to the values $a_1(x), \ldots, a_N(x)$, the goal in our decision problems is to compute the function $\mathsf{parity}(x)$. Suppose that we may identify $\mathsf{parity}(x) = f(a_1(x), \ldots, a_N(x))$ for some function f. If we can construct a function $\mathcal{Y}$ such that:

- Given $\mathcal{Y}(x)$, every function $a_i(x)$ can be computed by a low-degree polynomial, but
- Given $\mathcal{Y}(x)$, computing the parity of x requires a high-degree polynomial,

Then by combining these two statements, we see that the function $f(a_1, \ldots, a_N)$ itself requires a high-degree polynomial. We apply this framework taking f to be either the OS function or for

the "anchored hidden string" AHS function. The latter also implies a lower bound for the original (decisional) hidden string function HS described in the introduction.

In the following sections, we describe the main technical ideas that go into the proofs of our lower bounds. In order to provide more intuition about the structure of $\mathcal{Y}$, we describe the steps of constructing it for OS in detail before returning to the generalized framework.

2.2 Ordered search lower bound

First, notice that OS_N has the structure of an oracle identification problem since

$$\mathsf{OS}_N(\mathsf{GT}_{0^n}(x), \mathsf{GT}_{0^{n-1}1}(x), \ldots, \mathsf{GT}_{1^n}(x)) = \mathsf{parity}(x)$$

where $N = 2^n$ and $\mathsf{GT}_i(x) = 1$ if and only if $x \leq i$ where $i, x \in \{0,1\}^n$ if compared as numbers written in binary notation.

We want to show that there exists a function $\mathcal{Y}$ of x that we think of as revealing partial information about x such that:

- On one hand, for all $i \in \{0,1\}^n$ there is an algorithm that makes a small number of queries to $\mathcal{Y}$ and can identify the value of $\mathsf{GT}_i(x)$ with constant probability of success. Note that a query-efficient algorithm automatically gives rise to a low-degree approximating polynomial.
- On the other hand, approximating the value of $\mathsf{parity}(x)$ given $\mathcal{Y}$ with any probability of success requires a lot of queries to $\mathcal{Y}$. Let us denote this auxiliary problem by $\mathsf{PUR}(\mathcal{Y}) := \mathsf{parity}(x)$.

It is helpful to think of $\mathcal{Y}$ itself as an oracle, whose output is given to a polynomial or to a query algorithm, whose goal is then to compute some other function of x. We describe how we construct oracle $\mathcal{Y}$ through several attempts.

Let us first focus on constructing an oracle $\mathcal{Y}$ that meets the first condition. To do so, we can use the idea behind the $O(\log n \log\log n)$-bit communication protocol[2] for the two-party communication problem GT to obtain an efficient randomized query algorithm for every function GT_i. In the GT communication problem, Alice and Bob both get a string of n bits and the goal is to decide if the number represented by Alice's string is greater than the number represented by Bob's string.

In this randomized communication protocol for GT, Alice checks if the first halves of the inputs are equal and depending on the answer, she either recursively continues on the first halves of the inputs or the second halves. By doing so, she finds the most significant bit where the inputs differ. To perform each equality check, both Alice and Bob compute the inner products modulo 2 of each of the inputs with the same set of some α (publicly) random strings, Bob sends his values to Alice, and Alice compares these values to the values she obtained. If the original values were equal, then the inner products will be always equal, and otherwise, at least one pair of inner products will be unequal with high probability for sufficiently large α. This elementary operation (i.e., the ability to compute inner products with random strings) will be exactly what we want our oracle $\mathcal{Y}$ to be useful for.

[2] A more efficient $O(\log n)$-bit communication protocol is known and underlies our sharpest result for ordered search. We discuss it in Sections 2.4 and 3.

First attempt. We will eventually give a randomized construction of the oracle $\mathcal{Y}$, and to this end, think of it as taking as input both the hidden string x and a random input r. Let $\mathcal{Y}(r, x)$ be a function that takes a collection of m n-bit strings $r \in \times_{i\in[m]}(\{0,1\}^n)$ and $x \in \{0,1\}^n$, and outputs m bits, each representing the inner product of r_i with x: $(\mathcal{Y}(r, x))_i = \langle r_i, x\rangle$.

Our first attempt, however, will make no use of randomness at all. Let us consider $\mathcal{Y}(r, x)$ where r consists of all possible strings of length n. That is, the output of the oracle consists of $\langle x, r_i\rangle$ for every $r_i \in \{0,1\}^n$.

Let us now see how to construct a query algorithm C_i that, given oracle access to $\mathcal{Y}(r, x)$, computes $\mathsf{GT}_i(x)$ with high probability. This algorithm emulates Alice's side in the communication protocol, fixing her input to i. It samples random strings used in the communication protocol, computes the inner products of i with these random strings on its own, and asks the oracle (emulating Bob) for the inner products of x with the same random strings.

From the correctness of the communication protocol for GT we can conclude that for all $x, i \in \{0,1\}^n$

$$\Pr_{r_1\ldots,r_{\alpha\log n}}[C_i(\mathcal{Y}(r, x)) \neq \mathsf{GT}_i(x)] < \log n \cdot 2^{-\alpha}$$

where $r_1 \ldots, r_{\alpha\log n}$ are the strings that C_i sampled during the run, and r is a collection of all n-bit strings. The number of queries is $\alpha \log n$.

Thus we see that this oracle satisfies the first condition: it helps to compute the GT_i efficiently for every i and x. But now there is a problem with the second condition: $\mathsf{parity}(x) = \mathsf{PUR}(\mathcal{Y})$ can be computed easily since $\mathsf{parity}(x) = \mathsf{PUR}(\mathcal{Y}(r, x)) = \langle x, 1^n\rangle$. So there is a 1-query algorithm (and hence a degree-1 polynomial) that exactly computes $\mathsf{PUR}(\mathcal{Y}(r, x))$, violating our second condition.

Second attempt. Our goal now is to reduce the efficacy of the oracle $\mathcal{Y}$ in terms of how well it can be used by low-degree polynomials to approximate PUR. To do this, we instead consider a distribution over the potential oracles defined by the collection of strings used in the protocol. Let r denote a sequence of the random strings that could appear in one run of GT protocol described earlier. Let $\hat{\mathcal{R}}$ denote the set of all such sequences. This allows us to define a distribution of oracles $\mathcal{Y}[\hat{\mathcal{R}}](r, x)$, where $r \leftarrow \hat{\mathcal{R}}$, and for us to consider a deterministic query algorithm. Let $B_{(r,i)}$ be a deterministic algorithm that is given access to the $\mathcal{Y}[\hat{\mathcal{R}}](r, x)$ where $r \leftarrow \hat{\mathcal{R}}$ is chosen uniformly at random, and which has the realization of r and i hardcoded into it. This algorithm is able to emulate the communication protocol (and the algorithm C_i), but now each time it needs a random string, it uses one provided in r.

From the correctness of the communication protocol for GT we again can conclude that for all $x, i \in \{0,1\}^n$

$$\Pr_{r\leftarrow\hat{\mathcal{R}}}[B_{(r,i)}(\mathcal{Y}[\hat{\mathcal{R}}](r, x)) \neq \mathsf{GT}_i(x)] < \log n \cdot 2^{-\alpha}.$$

So, with high probability, $B_{(r,i)}$ computes $\mathsf{GT}_i(x)$ over the choice of the oracle $\mathcal{Y}[\hat{\mathcal{R}}](r, x)$ for $r \leftarrow \hat{\mathcal{R}}$.

Does this new oracle satisfy the second condition? Now an approximation to $\mathsf{PUR}[\hat{\mathcal{R}}](\mathcal{Y}[\hat{\mathcal{R}}](r, x))$ needs to approximate $\mathsf{parity}(x)$ when given a set of random parities from $\hat{\mathcal{R}}$. Indeed, we show this requires high degree, as a consequence of the fact that high degree polynomial is necessary to construct the full parity of x from random parities.

However, we need to add one more improvement to our structure. For every fixed i, x, the algorithm $B_{(r,i)}$ when run on $\mathcal{Y}[\hat{\mathcal{R}}](r, x)$ computes $\mathsf{GT}_i(x)$ with high probability over $r \leftarrow \hat{\mathcal{R}}$. But we need to switch quantifiers: we want an oracle that is "good" for all possible inputs for GT simultaneously and, unfortunately, our current construction doesn't give an algorithm computing $\mathsf{GT}_i(x)$ for all $i, x \in \{0,1\}^n$ using the same $r \leftarrow \hat{\mathcal{R}}$.

Third (and final) attempt. So, is there a way to fix the source of randomness so it works for all possible inputs? Inspired by Newman's theorem [New91] on simulating public randomness using private randomness in communication complexity, we show that there is. We show that by taking $t = O(\frac{n}{\delta^2})$ copies of $\hat{\mathcal{R}}$, denoted $\mathcal{R}_1, \mathcal{R}_2, \ldots \mathcal{R}_t$, we get a "good base" for the oracle. Consider a randomized algorithm $A_{(r,i)}$ that, given access to to $\mathcal{Y}[\mathcal{R}'](r, x)$ with $r \leftarrow \mathcal{R}' = \times_{j \in [t]} \mathcal{R}_j$, does the following:

- Sample $j \leftarrow [t]$ at random.
- Run $B_{(r,i)}$ using the set $\mathcal{R}_j$ as the source of randomness.

Following the argument underlying Newman's theorem, we show that this algorithm computes $\mathsf{GT}_i(x)$ with $\log n \cdot 2^{-\alpha} + \delta$ failure probability. It works for every i and x and it still makes only $\alpha \log n$ queries to the oracle. If we put $\delta = \frac{1}{12}$ and $\alpha = O(\log \log n)$ then the probability of this algorithm failing for some input pair is at most $\frac{1}{6}$ with only $\alpha \log n = O(\log n \log \log n)$ queries to the oracle, i.e.,

$$\Pr_{r \leftarrow \mathcal{R}'}[A_{(r,i)}(\mathcal{Y}[\mathcal{R}'](r, x)) \neq \mathsf{GT}_i(x)] < \frac{1}{6}.$$

This change also doesn't increase the "size" of the oracle (i.e., the number of queries it can answer) too much. This allows us to show that with high probability it is still impossible to combine the given partial parities to create the full parity using a low-degree polynomial, so the second condition is also satisfied. So there exists an oracle that allows computing the GT with low-degree polynomials but requires a high-degree polynomial to compute $\mathsf{parity}(x)$ which is exactly what allows us to prove the lower bound on the approximate degree of OS.

2.3 Technical ideas behind the parity lower bound

Our technique relies on a lower bound on the approximate degree of $\mathsf{parity}(x)$, or, more precisely, on the "Parity Under Randomness $\mathcal{R}$" function $\mathsf{PUR}[\mathcal{R}](\mathcal{Y}[\mathcal{R}](r, x))$ evaluates to $\mathsf{parity}(x)$ on input $\mathcal{Y}[\mathcal{R}](r, x)$. We, in fact, prove a more general statement lower bounding the approximate degree of $\mathsf{PUR}[\mathcal{R}]$ for a class of potential structures $\mathcal{R}$.

Specifically, we show that the parity function is hard, even to sign-represent, and even given access to $\mathcal{Y}[\mathcal{R}]$ consisting of inner products of x with random strings r_i where each bit of r_i is either fixed to zero or is an unbiased random bit. The only other restriction we need on $\mathcal{Y}[\mathcal{R}]$ is that its "size", i.e., the number of inner products it provides, is small. The bigger this number is, the worse our the lower bound becomes.

The proof idea is based on the hardness of sign-representing parity as described in [ABFR91], combined with the following combinatorial observation: given a set of n-bit strings (corresponding to samples from $\mathcal{R}$, and in turn to random inner products) where in every string each bit is either zero or is an unbiased random bit, with high probability no small subset of them adds up to the all-ones string (which corresponds to the parity function).

2.4 Improved ordered search and anchored hidden string lower bounds

Our generalized lower bound for approximating $\mathsf{PUR}[\mathcal{R}]$ allows us to obtain other lower bounds for oracle identification problems. For example, we give a slightly stronger lower bound for OS than what is implied by the discussion above. There is, in fact, a more efficient randomized communication protocol for GT that uses $O(\log n)$ bits of communication. It can be converted into randomized query algorithm and thus into a polynomial of degree $O(\log n)$. At the same time, this more efficient protocol is still based on computing equalities of substrings of inputs, and so the appropriate $\mathcal{Y}$ has a very similar structure to the one described above while still satisfying the conditions of the generalized lower bound for PUR. Moreover, the necessary "size" of $\mathcal{Y}$ barely blows up at all. Putting everything together gives our improved lower bound of $\Omega\left(\frac{n}{\log^2 n}\right)$ on the approximate degree of OS.

Using the same framework, we can also obtain a nearly tight lower bound on the approximate degree of the anchored hidden string problem AHS. In the anchored hidden string problem, the goal is to determine the parity of x given oracle access to $y_{i,s} = \phi_{i,s}(x)$ for every index i and every binary string s of length at most n, where $\phi_{i,s}(x) = 1$ iff the substring of x starting at index i matches s. This oracle identification problem has the right form for our framework since

$$\mathsf{AHS}_N((\phi_{i,s}(x))_{i\in[n],s\in\{0,1\}^{\leq n-i+1}}) = \mathsf{parity}(x).$$

Moreover, each function $\phi_{i,s}(x)$ simply computes the equality function of s with a substring of x of length $|s|$ starting from position i. As we have already seen, we can compute the equality function very efficiently given an oracle $\mathcal{Y}$ of the right random structure, and such a $\mathcal{Y}$ meets the conditions of our generalized lower bound for $\mathsf{PUR}[\mathcal{Y}]$. This directly implies a lower bound of $\Omega\left(\frac{n}{\log n}\right)$ on the approximate degree of AHS.

Finally, the last lower bound described in this work is on the approximate degree of HS. This lower bound follows via a reduction from AHS. This reduction was first introduced in [CIG+12] in the quantum query model, but it holds for polynomial approximation as well.

3 Ordered search and generalized lower bound

In this section we give the formal proof of our lower bound on the approximate degree of ordered search. We show how our framework is used for this function and prove the generalized lower bound on parity that we later reuse for the hidden string problem.

3.1 Preliminaries

Our lower bounds on the approximate degree of (a decision version) of ordered search and the hidden string problem require the following definition of polynomial approximations for promise problems.

Definition 3. Let $f : D \to \{0,1\}$ where $D \subseteq \{0,1\}^n$ for some $n \in \mathbb{N}$ be a partial Boolean function. For $\frac{1}{2} > \varepsilon > 0$, a polynomial $p : \{0,1\}^n \to \mathbb{R}$ is an ε-approximation to f if $|p(x) - f(x)| \leq \varepsilon$ for every $x \in D$ and $-\varepsilon \leq p(x) \leq 1 + \varepsilon$ for all $x \in \{0,1\}^n$. The ε-approximate degree of f, denoted $\widetilde{\deg}_\varepsilon(f)$ is the the least degree of a polynomial p that ε-approximates f. We use the convention $\widetilde{\deg}(f) = \widetilde{\deg}_{1/3}(f)$ to refer to the "approximate degree of f" without qualification.

That is, we require a polynomial approximation to a partial function defined on a domain D to approximate the function on D and remain bounded outside of D. Note that this is the type of approximation that arises from quantum query algorithms for promise problems.

We also formally define the ordered search function OS and the family of greater-than functions GT.

Definition 4. For all $i \in \{0,1\}^n$ define the function $\mathsf{GT}_i : \{0,1\}^n \to \{0,1\}$ to be the indicator of whether the value of the input is smaller than i: $\mathsf{GT}_i(x) = 1$ if and only if $x \leq i$ where i and x are compared as numbers written in binary notation.

Definition 5. The ordered search function $\mathsf{OS}_{2^n} : \{0^k 1^{2^n-k} \mid k \in [2^n]\} \to \{0,1\}$ is a partial function defined the following way: $\mathsf{OS}_{2^n}(0^k 1^{2^n-k}) = \mathsf{parity}(x)$ where $x \in \{0,1\}^n$ is the binary representation of k.

3.2 The notion of a *good base*.

In order to formally define the oracle, i.e. the source of additional information about the input, we introduce the notion of a "*good base*" for the oracle. A set $\mathcal{R}$, consisting of tuples of strings, is a *good base* if it's constructed as follows.

Let $\mathcal{R}'$ be a Cartesian product of m' subsets of $\{0,1\}^n$ where each subset $\mathcal{R}^\tau$ is itself defined by an n-bit string-template $\tau = \tau_1\tau_2\ldots\tau_n \in \{0,1\}^n$

$$\mathcal{R}^\tau = S_{\tau_1} S_{\tau_2} S_{\tau_3} \ldots S_{\tau_n}$$

where $S_0 = \{0\}$ and $S_1 = \{0,1\}$.

For example, if $\tau = 00100010$ then $\mathcal{R}^\tau = S_{\tau_1} S_{\tau_2} S_{\tau_3} \ldots S_{\tau_n} = S_0 S_0 S_1 S_0 S_0 S_0 S_1 S_0 = \{0\}\{0\}\{0,1\}\{0\}\{0\}\{0\}\{0,1\}\{0\} = \{00000000, 00000010, 00100000, 00100010\}$.

Let $\mathcal{B} = \{\mathbf{1}_1\} \times \{\mathbf{1}_2\} \times \ldots \times \{\mathbf{1}_n\}$ where $\mathbf{1}_j = 0^{i-j}10^{n-j}$ is the string that has the value 1 in j-th position and has the value 0 everywhere else. Let $\mathcal{R} = \mathcal{B} \times \mathcal{R}'$, and thus $\mathcal{R}$ is a Cartesian product of $m = n + m'$ subsets of $\{0,1\}^n$. Note that every $r \in \mathcal{R}$ is a m-tuple of n-bit strings:

$$r = (r_1, r_2, \ldots r_m) = (\mathbf{1}_1, \mathbf{1}_2, \ldots, \mathbf{1}_{n-1}, \mathbf{1}_n, r_{n+1}, r_{n+2}, \ldots, r_{n+m'})$$

where each r_j is a string of length n, the first n strings are fixed for all $r \in \mathcal{R}$, and the last m' strings are from some sets $\mathcal{R}^\tau$ each for some template τ. If $r \leftarrow \mathcal{R}$ is chosen u.a.r. then each $r_j, n < j \leq m'$ is chosen u.a.r. from some $\mathcal{R}^\tau$ and thus the subsequence of bits of r_j corresponding to ones in τ is a uniformly random string, and the subsequence of bits of r_j corresponding to zeros in τ is the all-zero string.

Any set $\mathcal{R}$ with the above structure will be called a *good base* of size m. Such an $\mathcal{R}$ is helpful for building our oracles as follows.

Let $\mathcal{Y}[\mathcal{R}] : \mathcal{R} \times \{0,1\}^n \to \{0,1\}^m$ be the following function: $(\mathcal{Y}[\mathcal{R}](r,x))_j = \langle r_j, x\rangle$ where r_j is an n-bit string from the collection $r \in \mathcal{R}$ and the inner product is taken modulo 2. Note that $\mathcal{Y}[\mathcal{R}]$ is parameterized by $\mathcal{R}$, so for each *good base* $\mathcal{R}$ the function $\mathcal{Y}[\mathcal{R}]$ will be different. We will omit the parameter $\mathcal{R}$ later in places where it is clear from context.

Notice the following properties of this function $\mathcal{Y}[\mathcal{R}](r,x)$ that hold whenever $\mathcal{R}$ is a *good base*:

- For every $r \in \mathcal{R}$, the values $\mathcal{Y}(r,x)$ completely determine x. Since the first n strings of r are $\mathbf{1}_1$, $\mathbf{1}_2$, …, $\mathbf{1}_{n-1}$, $\mathbf{1}_n$, the first n bits of $\mathcal{Y}(r,x)$ are exactly bits of x.

- Given $\mathcal{Y}(r,x)$ for $r \leftarrow \mathcal{R}$ and r itself, one can compute (with some probability of error) whether a subsequence of x specified by some pattern τ agrees with some fixed string s in those indices. To be more specific, if given $(\mathcal{Y}(r,x))_j = \langle r_j, x\rangle$ where r_j is sampled from $\mathcal{R}^\tau$ uniformly at random, and r_j itself, one can check whether the strings $x \wedge \tau$ (where $\wedge$ denotes bitwise AND) and $s \wedge \tau$ are equal for any $s \in \{0,1\}^n$ with one-sided error probability $\frac{1}{2}$.

So, $\mathcal{Y}[\mathcal{R}](r,x)$ could be used as an equality oracle for a fixed set of subsequences of x (predefined by $\mathcal{R}$) when r is chosen uniformly at random from $\mathcal{R}$. Thus, $\mathcal{Y}[\mathcal{R}](r,x)$ might give more information about x than x alone and might make some computations on x more efficient.

On the other hand, some functions of x remain "hard" even when given $\mathcal{Y}(r,x)$. We will later show that $\mathsf{parity}(x)$ remains hard to compute even with this additional information.

3.3 Approximating polynomials for GT_i

We start our proof by showing that for some *good base* $\mathcal{R}_{\mathsf{OS}}$ the oracle $\mathcal{Y}[\mathcal{R}_{\mathsf{OS}}]$ could be used to make the computation of GT functions more efficient.

Claim 6. There exists a *good base* $\mathcal{R}_{\mathsf{OS}}$ of size $m = O(n^2 \log\log n)$ such that if $r \leftarrow \mathcal{R}_{\mathsf{OS}}$ is sampled uniformly at random, then with probability at least $\frac{2}{3}$ over the choice of r there exists a family of 2^n polynomials $\{q_{(r,i)} : \{0,1\}^m \to \{0,1\} \mid i \in \{0,1\}^n\}$, each of degree at most $2\log n \log\log n$, such that given $\mathcal{Y}[\mathcal{R}_{\mathsf{OS}}](r,x)$ as the input, each polynomial $q_{(r,i)}(\mathcal{Y}[\mathcal{R}_{\mathsf{OS}}](r,x))$ approximates the corresponding $\mathsf{GT}_i(x)$ with error at most $\frac{1}{6}$. That is,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}\left[\exists i, x \in \{0,1\}^n : \left|q_{(r,i)}(\mathcal{Y}(r,x)) - \mathsf{GT}_i(x)\right| > \frac{1}{6}\right] < \frac{1}{3}.$$

Proof. This proof consists of two parts: constructing a *good base* $\mathcal{R}_{\mathsf{OS}}$ and showing that it actually helps to compute every GT_i.

Constructing the *good base* $\mathcal{R}_{\mathsf{OS}}$. We are going to construct $\mathcal{R}_{\mathsf{OS}}$ based on what random strings are useful in the communication protocol computing GT of two n-bit strings, x and i. Intuitively, in this protocol, we first need to check if the first half of i and x are equal using a randomized communication protocol for equality. To do that we need to compute and compare $\langle x, r\rangle$ and $\langle i, r\rangle$, for some number α of random strings r to be determined later, where each r is sampled from $\{0,1\}^{\frac{n}{2}}\{0\}^{\frac{n}{2}}$. If the computed values $\langle i, r\rangle = \langle x, r\rangle$ for all r we have considered, then we repeat this procedure on the second half of x and i, which corresponds to computing and comparing $\langle x, r\rangle$ and $\langle i, r\rangle$ for α random strings r sampled from $\{0\}^{\frac{n}{2}}\{0,1\}^{\frac{n}{4}}\{0\}^{\frac{n}{4}}$. If, on the other hand, the values were not equal then we repeat this procedure on the first half of x and i, which corresponds to computing and comparing $\langle x, r\rangle$ and $\langle i, r\rangle$ for α random strings r sampled from $\{0,1\}^{\frac{n}{4}}\{0\}^{\frac{3n}{4}}$. Since we want our oracle to be useful to emulate this procedure to compute $\mathsf{GT}_i(x)$, it should "contain" all the random strings used in this protocol.

Let $\hat{\mathcal{R}} = \mathcal{R}^{1^{n/2}0^{n/2}} \times \left(\mathcal{R}^{1^{n/4}0^{3n/4}} \times \mathcal{R}^{0^{n/2}1^{n/4}0^{n/4}}\right) \times \ldots \times \left(\bigtimes_{i=0}^{2^k-1} \mathcal{R}^{0^{2in/2^{k+1}}1^{n/2^{k+1}}0^{n-((2i+1)n/2^{k+1})}}\right) \times \ldots \times \left(\bigtimes_{i=0}^{n/2} \mathcal{R}^{0^{2i}1^1 0^{n-(2i+1)}}\right)$. See Figure 1 for an illustration.

This $\hat{\mathcal{R}}$ describes all the strings used as the source of randomness in the $O(\log n \log\log n)$ communication protocol for GT, but each of the strings appears in the structure only once instead of α times. So, we need to duplicate this structure α times to properly simulate the protocol.

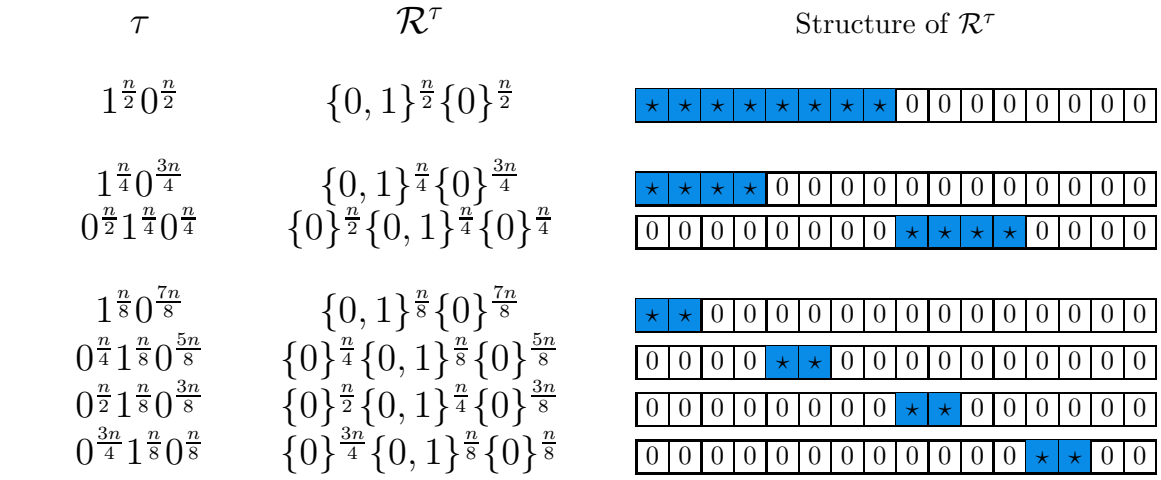

Figure 1: Structure of $\hat{\mathcal{R}}$. Blue cells with $\star$ represent indices in which either a 0 or a 1 could appear.

To finish the structure, we are going to add two other steps to the structure. First, we are going to have some number t of individual "prepackaged" copies to be determined later for the GT protocol. Let $\mathcal{R}_1 = \ldots = \mathcal{R}_t = \times_\alpha \hat{\mathcal{R}}$. Each of the copies has enough randomness and the right structure of that randomness to simulate one full run of the GT protocol. Let $\mathcal{R}' = \bigtimes_{j\in[t]} \mathcal{R}_j$ which allows us to handle t runs. Secondly, we want to be able to obtain the value of any specific index of x, so we add a set of "basis" strings to the structure: $\mathcal{B} = \{\mathbf{1}_1\} \times \{\mathbf{1}_2\} \times \ldots \times \{\mathbf{1}_n\} = \{10\ldots0\} \times \{010\ldots0\} \times \ldots \times \{00\ldots010\} \times \{00\ldots01\}$.

The final underlying structure of the oracle will be a Cartesian product of $\mathcal{R}'$ and $\mathcal{B}$: $\mathcal{R}_{\mathsf{OS}} = \mathcal{B} \times \mathcal{R}' = \mathcal{B} \times (\bigtimes_{j\in[t]} \mathcal{R}_j)$. See Figure 2 for an illustration.

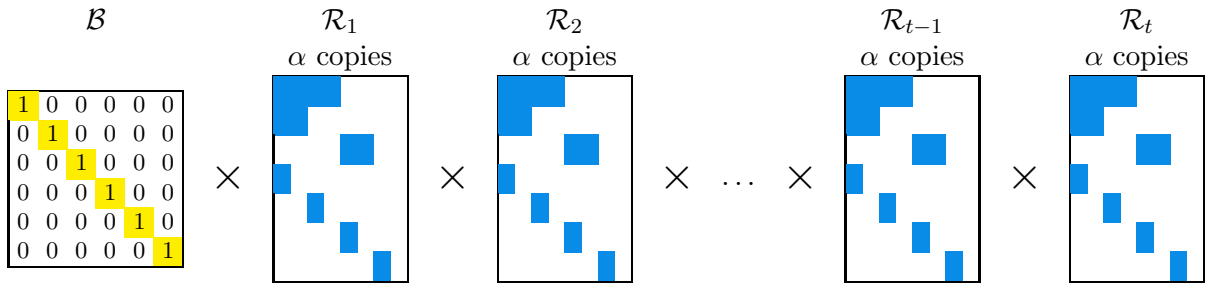

Figure 2: Structure of $\mathcal{R}_{\mathsf{OS}}$. Each $\mathcal{R}_j$ consist of α copies of $\hat{\mathcal{R}}$.

We also set the parameters to be $\alpha = 2\log(\log n), t = 250n\ln 2$. Notice that this set $\mathcal{R}_{\mathsf{OS}}$ is a *good base* by construction and has size $m = n + \alpha t n = n + cn^2\log(\log n)$ for some constant c.

Constructing the family of approximating polynomials. In order to prove this claim, we first describe a randomized query algorithm that computes $\mathsf{GT}_i(x)$ correctly for all i and x with high probability given $\mathcal{Y}[\mathcal{R}_{\mathsf{OS}}](r,x)$ as input. We then explain how to convert this query algorithm into a polynomial. The algorithm construction itself consists of two parts. In the first part, for all $j \in [t]$ we show the existence of a deterministic algorithm $B_{(r,i,j)}$ that, given $\mathcal{Y}(r,x)$, can compute $\mathsf{GT}_i(x)$ for every specific $x, i \in \{0,1\}^n$ with good probability over the choice of $r \leftarrow \mathcal{R}_{\mathsf{OS}}$, and this

algorithm is only going to use the parts of the input that correspond to $\mathcal{R}_j$ and $\mathcal{B}$. In the second part, we show that the algorithm $A_{(r,i)}$ that chooses a copy j to use randomly and runs $B_{(r,i,j)}$, computes $\mathsf{GT}_i(x)$ correctly for all i and x with high probability given $\mathcal{Y}(r,x)$ as input.

For all $i \in \{0,1\}^n, j \in [t], r \in \mathcal{R}_{\mathsf{OS}}$ let $B_{(r,i,j)}(\mathcal{Y}(r,x))$ be the following deterministic algorithm.

1. Set $\ell = 0, u = n/2$
2. While $\ell < u$:
3. Set $\tau = 0^\ell 1^{u-\ell} 0^{n-u}$
4. For all indices $v \in [m]$ corresponding to n-bit strings drawn from $\mathcal{R}^\tau$ within the j-th copy $\mathcal{R}_j$:
5. Compute $\langle i, r_v \rangle$ and compare it to $(\mathcal{Y}(r,x))_v = \langle x, r_v \rangle$.
6. If for all such v the inner products are equal, i.e., $\langle i, r_v \rangle = (\mathcal{Y}(r,x))_v$, then set $tmp = u, u = u + (u-\ell)/2, \ell = tmp$ and go step 2.
7. Otherwise, set $u = (u+\ell)/2$ and go step 2
8. Compare $i_\ell = \langle i, \mathbf{1}_\ell \rangle$ and $(\mathcal{Y}(r,x))_\ell = \langle x, \mathbf{1}_\ell \rangle = x_\ell$. If $x_\ell \leq i_\ell$ then accept. Otherwise, reject.

The last step is possible specifically because of $\mathcal{B}$ in the structure of $\mathcal{R}_{\mathsf{OS}}$: $r_\ell = \mathbf{1}_\ell$ for all $\ell \leq n$ and for all $r \in \mathcal{R}_{\mathsf{OS}}$. Notice that this algorithm emulates the randomized communication protocol for the GT communication problem.

In general, the algorithm emulates the randomized communication protocol for equality on the first half of the segment $[\ell, u + (u-\ell)]$ in x and i, and depending on the result it splits the inputs into smaller segments and continues recursively. In the end, if all the runs of equality protocols were correct, the algorithm finds and compares the most significant bit where x and i differ.

By [Nis93] we know that this algorithm computes $\mathsf{GT}_i(x)$ with probability at least $1-(\log n)2^{-\alpha} = 1-(\log n)2^{-2\log(\log n)} = 1 - \frac{1}{\log n} \geq \frac{11}{12}$ for sufficiently large n independently of the choice of $j \in [t]$. That is, for all $j \in [t]$ and for all $i, x \in \{0,1\}^n$,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}} [B_{(r,i,j)}(\mathcal{Y}(r,x)) = \mathsf{GT}_i(x)] \geq \frac{11}{12}.$$

This algorithm makes at most $\alpha \log n = 2 \log n \log\log n$ queries to the oracle $\mathcal{Y}(r,x)$. Note that this algorithm needs access to the specific r needed to compute every $\langle i, r_v \rangle$ and we enable this by "hardcoding" this r into the algorithm and creating a separate algorithm for each possible r.

We have shown that for every fixed $i, x \in \{0,1\}^n$ there are many $r \in \mathcal{R}_{\mathsf{OS}}$ that if used as a first input for the oracle $\mathcal{Y}$ allow $B_{(r,i,j)}$ to compute $\mathsf{GT}_i(x)$. Unfortunately, this is not enough: our algorithm should be universal, i.e., we want a single algorithm that with high probability over r succeeds on all i and x. On the other hand, $B_{(r,i,j)}$ only uses one fixed "package" of random strings, namely the j-th package.

Let $W(i,x,r,j)$ be the indicator that the j-th package of random strings in r defines a set of "bad" random strings for (i,x): $W(i,x,r,j) = 1$ if and only if $B_{(r,i,j)}(\mathcal{Y}(r,x)) \neq \mathsf{GT}_i(x)$. We established that $B_{(r,i,j)}(\mathcal{Y}(r,x))$ works well if given a random $r \leftarrow \mathcal{R}_{\mathsf{OS}}$ for every $j \in [t]$ and

the probability of this algorithm outputting an incorrect answer is at most $\frac{1}{12}$. So for all $i, x \in \{0,1\}^n, j \in [t]$, we have

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}[W(i,x,r,j) = 1] = \mathbb{E}_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}[W(i,x,r,j)] \leq \frac{1}{12}.$$

We can't immediately get a useful upper bound on the probability of $r \leftarrow \mathcal{R}$ working out for all i and x at the same time. To achieve this, we'll design a new algorithm that uses all t packages of random strings. Its construction and analysis are inspired by Newman's classic argument used for simulating public randomness by private randomness in communication protocols.

For all $i \in \{0,1\}^n, r \in \mathcal{R}_{\mathsf{OS}}$ let $A_{(r,i)}(\mathcal{Y}(r,x))$ be the following randomized algorithm:

- Choose $j \leftarrow [t]$ uniformly at random.
- Run $B_{(r,i,j)}(\mathcal{Y}(r,x))$.

Let us now analyse $A_{(r,i)}$. The number of queries that $A_{(r,i)}$ makes to the oracle is the same as $B_{(r,i,j)}$ which is $\alpha \log n = 2 \log n \log \log n$. We fix a pair (i,x) and evaluate the following probability.

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}\left[\Pr_{j \leftarrow [t]}[B_{(r,i,j)} \neq \mathsf{GT}_i(x)] > \frac{1}{6}\right] = \Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}\left[\frac{1}{t}\sum_{j \in [t]} W(i,x,r,j) > \frac{1}{6}\right].$$

We established that $\mathbb{E}_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}[W(i,x,r,j)] \leq \frac{1}{12}$ and so by Hoeffding's inequality,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}\left[\frac{1}{t}\sum_{j \in [t]} W(i,x,r,j) > \frac{1}{12} + \frac{1}{12}\right] \leq e^{-2\frac{t}{144}} \leq 2^{-\frac{500n}{144}}.$$

By a union bound over all possible $i, x \in \{0,1\}^n$,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}\left[\exists i, x \in \{0,1\}^n : \frac{1}{t}\sum_{j \in [t]} W(i,x,r,j) > \frac{1}{6}\right] \leq 2^{2n} 2^{-\frac{500n}{144}} \leq 2^{-n} < \frac{1}{3}.$$

Therefore, we have proven that

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}\left[\exists i, x \in \{0,1\}^n : \Pr_{j \leftarrow [t]}[A_{(r,i)}(\mathcal{Y}(r,x))) \neq \mathsf{GT}_i(x)] > \frac{1}{6}\right] < \frac{1}{3}.$$

The last step is to convert this family of query algorithms into a family of approximating polynomials. Let $q_{(r,i)}$ denote the acceptance probability of $A_{(r,i)}$. A standard argument (e.g., [BdW02, Theorem 15]) implies that this is a polynomial of degree at most $2 \log n \log \log n$ such that

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}}}\left[\exists i, x \in \{0,1\}^n : \left|q_{(r,i)}(\mathcal{Y}(r,x)) - \mathsf{GT}_i(x)\right| > \frac{1}{6}\right] < \frac{1}{3},$$

which is exactly what we were looking for. □

We successfully converted the most well-known communication protocol for GT that requires $O(\log n \log\log n)$ bits of communication into a family of polynomials of degree $O(\log n \log\log n)$ that approximates GT_i. It's known that there is a better communication protocol for GT that requires only $O(\log n)$ bits of communication, as observed by Nisan [Nis93]. The next claim establishes that this more efficient protocol can be converted into a family of polynomials as well.

Claim 7. There exists a *good base* $\mathcal{R}_{\mathsf{OS}++}$ of size $m = O(n^3 \log n)$ such that if $r \leftarrow \mathcal{R}_{\mathsf{OS}++}$ is sampled uniformly at random, then with probability at least $\frac{2}{3}$ over the choice of r there exists a family of polynomials $\{q_{(r,i)} : \{0,1\}^m \to \{0,1\} \mid i \in \{0,1\}^n\}$, each of degree at most $O(\log n)$, such that given $\mathcal{Y}(r,x)$ as the input, each polynomial $q_{(r,i)}(\mathcal{Y}[\mathcal{R}_{\mathsf{OS}++}](r,x))$ approximates the corresponding $\mathsf{GT}_i(x)$ with error at most $\frac{1}{6}$. That is,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}}\left[\exists i, x \in \{0,1\}^n : \left|q_{(r,i)}(\mathcal{Y}(r,x)) - \mathsf{GT}_i(x)\right| > \frac{1}{6}\right] < \frac{1}{3}.$$

The proof of Claim 7 is similar to the proof of Claim 6 and can be found in Appendix B.

3.4 General lower bound

To complete the framework and to obtain the lower bound for Ordered Search we need to show why computing the parity is hard even given $\mathcal{Y}[\mathcal{R}_{\mathsf{OS}}]$ or $\mathcal{Y}[\mathcal{R}_{\mathsf{OS}++}]$. We will show a stronger lower bound that would allow us to reuse this lower bound for other applications. Specifically, we will show that computing the parity of input x remains hard given $\mathcal{Y}[\mathcal{R}]$ for any *good base* $\mathcal{R}$ of small size.

3.4.1 Combinatorial claim

The hardness of parity in this model is based on the following statement. For every *good base* $\mathcal{R}$ of small size with high probability over the sample $r \leftarrow \mathcal{R}$ for every set of n-bit strings taken from the collection r of size at most $O(\frac{n}{\log n})$, the bitwise parity of these strings is not equal to the all-ones string.

Claim 8. For every *good base* $\mathcal{R}$ of size m with probability at least $\frac{2}{3}$ over the choice of $r \leftarrow \mathcal{R}$ for every set of elements $T \subseteq [m]$ of size at most $d = \frac{n}{4\log m} - 1$, the bitwise parity of n-bit strings r_i, $i \in T$ from the collection $r \leftarrow \mathcal{R}$ is not equal to the all-ones string:

$$\Pr_{r \leftarrow \mathcal{R}}\left[\forall T \subseteq [m], |T| \le d : \bigoplus_{i \in T} r_i \neq 1^n\right] \ge \frac{2}{3}.$$

Proof. Fix an arbitrary *good base* $\mathcal{R}$ of size m. Fix a set $T \subseteq [m]$ where $|T| \le d$. We want to bound the probability $\Pr_{r \leftarrow \mathcal{R}}[\bigoplus_{i \in T} r_i = 1^n]$ that for this r and for this T the strings corresponding to the indices in T sum up to the string of all ones. Fix a specific index $k \in [n]$. We compute the probability that index k is set to 1 in $\bigoplus_{i \in T} r_i$. To do this we need to understand how the candidate strings $r_i, i \in T$ can influence this value.

There are three possible scenarios for each index k:

- (Type I) There is at least one string $r_i \in \{0,1\}^n$ with $i \in T$ such that it is chosen from $\mathcal{R}^\tau$ where $\tau_k = 1$. Then in each such string, the bit at index k is sampled independently at random with probability $\frac{1}{2}$. Thus $\Pr_{r \leftarrow \mathcal{R}}[\langle \bigoplus_{i \in T} r_i, \mathbf{1}_k \rangle = 1] = \frac{1}{2}$.

- (Type II) There are no strings $r_i, i \in T$ such that r_i is chosen from $\mathcal{R}^\tau$ and $\tau_k = 1$, but there is $r_i, i \in T$ that is chosen from $\mathcal{B}$, such that $r_i = \mathbf{1}_k$. Then the value of $\langle \bigoplus_{i \in T} r_i, \mathbf{1}_k \rangle$ is one since there is exactly one string in this sum with the kth index value set to one. Thus $\Pr_{r \leftarrow \mathcal{R}}[\langle \bigoplus_{i \in T} r_i, \mathbf{1}_k \rangle = 1] = 1$.
- (Type III) There are no strings $r_i, i \in T$ such that r_i is chosen from $\mathcal{R}^\tau$ and $\tau_k = 1$, and there is no $r_i, i \in T$ that is chosen from $\mathcal{B}$, such that $r_i = \mathbf{1}_k$. Then for all strings r_i the index k is 0, so $\Pr_{r \leftarrow \mathcal{R}}[\langle \bigoplus_{i \in T} r_i, \mathbf{1}_k \rangle = 1] = 0$.

Index k	1	2	3	4	5	6	7	8
r_{i_1}	$\star$	$\star$	$\star$	$\star$	0	0	$\star$	0
r_{i_2}	$\star$	$\star$	$\star$	0	0	0	0	0
r_{i_3}	$\star$	0	$\star$	$\star$	0	0	$\star$	0
r_{i_4}	0	$\star$	0	0	0	0	0	0
r_{i_5}	0	0	1	0	0	0	0	0
r_{i_6}	0	0	0	0	1	0	0	0
Type	I	I	I	I	II	III	I	III
$\Pr_{r \leftarrow \mathcal{R}}[\langle \bigoplus_{i \in T} r_i, \mathbf{1}_k \rangle]$	1/2	1/2	1/2	1/2	1	0	1/2	0

Figure 3: Example of index types, $T = \{i_1, i_2, i_3, i_4, i_5, i_6\}$.

Notice that T fully defines the types of all indices and thus the values of $\langle \bigoplus_{i \in T} r_i, \mathbf{1}_k \rangle$ for k of types II and III don't depend on the choice of $r \leftarrow \mathcal{R}$. On the other hand, the values of indices of type I do depend on the choice of $r \leftarrow \mathcal{R}$. Each of them is either a parity of independent random bits or the negation of a parity of independent random bits which is fixed by T too. Thus they behave as independent bits themselves and therefore the values $\langle \bigoplus_{i \in T} r_i, \mathbf{1}_k \rangle$ are mutually independent for all indices k.

Denote by $n_{\mathrm{I}}, n_{\mathrm{II}}, n_{\mathrm{III}}$ the numbers of indices of each type. Notice that $n_{\mathrm{I}} + n_{\mathrm{II}} + n_{\mathrm{III}} = n$ and $n_{\mathrm{II}} \leq d$. Then in this notation

$$\Pr_{r \leftarrow \mathcal{R}} \left[\bigoplus_{i \in T} r_i = 1^n \right] = \left(\frac{1}{2} \right)^{n_{\mathrm{I}}} 1^{n_{\mathrm{II}}} 0^{n_{\mathrm{III}}}.$$

If there exists $k \in [n]$ of the third type, the probability $\Pr_{r \leftarrow \mathcal{R}}[\bigoplus_{i \in T} r_i = 1^n]$ becomes 0, so to upper bound the probability we may assume all the indices have one of the first two types. And, since $n_{\mathrm{II}} \leq d$, to maximize the value we assume that $n_{\mathrm{II}} = d$. Thus we have

$$\Pr_{r\leftarrow\mathcal{R}}\left[\bigoplus_{i\in T} r_i = 1^n\right] = \left(\frac{1}{2}\right)^{n-n_{\text{II}}} 1^{n_{\text{II}}} \leq 1^d \left(\frac{1}{2}\right)^{n-d} = 2^{-(n-d)}.$$

Since $d = \frac{n}{4\log m} - 1$ and $m \geq n$, we have $n - d > \frac{n}{2}$ for sufficiently large n. So for a fixed T,

$$\Pr_{r\leftarrow\mathcal{R}}\left[\bigoplus_{i\in T} r_i = 1^n\right] < 2^{-\frac{n}{2}}.$$

There are $\binom{m}{\leq d}$ ways to choose the set T, so by a union bound over the choice of T, the probability that for some set of size at most d the value $\bigoplus_{i\in T} r_i$ is equal to the string of all ones is

$$\Pr_{r\leftarrow\mathcal{R}}\left[\exists T \subseteq [m], |T| \leq d : \bigoplus_{i\in T} r_i = 1^n\right] \leq \binom{m}{\leq d} 2^{-\frac{n}{2}} = 2^{-\frac{n}{2}} \sum_{d'=0}^{d} \binom{m}{d'} \leq 2^{-\frac{n}{2}} \sum_{d'=0}^{d} m^{d'} \leq 2^{-\frac{n}{2}} m^{d+1}$$

$$= 2^{-\frac{n}{2}} m^{\frac{n}{4\log m}} = 2^{-\frac{n}{2}} 2^{\frac{n}{4\log m}\log m} = 2^{\frac{n}{4}-\frac{n}{2}} = 2^{-\frac{n}{4}} < \frac{1}{3}.$$

□

3.4.2 Lower bound on the degree of $\mathsf{PUR}[\mathcal{R}]$

For every *good base* $\mathcal{R}$ and for every fixed $r \in \mathcal{R}$ define the function $\mathsf{PUR}[\mathcal{R}]_r : D[\mathcal{R}]_r \to \{0,1\}$ where $D[\mathcal{R}]_r = \{\mathcal{Y}[\mathcal{R}](r,x) \mid x \in \{0,1\}^n\}$ is the subset of $\{0,1\}^m$ where each domain point corresponds to one specific $x \in \{0,1\}^n$ and is consistent with the fixed r. This function outputs the parity of the string encoded by the input: $\mathsf{PUR}[\mathcal{R}]_r(\mathcal{Y}(r,x)) = \mathsf{parity}(x)$. It is well defined since $\mathsf{parity}(x) = \bigoplus_{r_i\in\mathcal{B}}\langle x, r_i\rangle = \bigoplus_{i\in[n]}(\mathcal{Y}(r,x))_i$. Note that both $D[\mathcal{R}]_r$ and $\mathsf{PUR}[\mathcal{R}]$ are parameterized by $\mathcal{R}$ and, as with $\mathcal{Y}[\mathcal{R}]$, we will omit the parameter later in places where the parameter is clear from the context.

Our goal is to show that $\mathsf{PUR}[\mathcal{R}]$ is hard to approximate if $\mathcal{R}$ is a *good base* of small size. We do this by showing that for every *good base* $\mathcal{R}$ of size m if $r \leftarrow \mathcal{R}$ u.a.r. then every polynomial p of degree at most $d = O(\frac{n}{\log m})$ is completely uncorrelated with $\mathsf{PUR}[\mathcal{R}]_r(\mathcal{Y}(r,x))$ with high probability over the choice of r.

Theorem 9. For every *good base* $\mathcal{R}$ of size m if $r \leftarrow \mathcal{R}$ u.a.r. then with probability at least $\frac{2}{3}$ over the choice of r every polynomial $p : \{0,1\}^m \to \mathbb{R}$ of degree at most $d = \frac{n}{4\log m} - 1$ doesn't approximate $\mathsf{PUR}[\mathcal{R}]_r$:

$$\Pr_{r\leftarrow\mathcal{R}}\left[\forall \varepsilon < \frac{1}{2},\ \forall p, \deg(p) \leq d, \exists y \in D[\mathcal{R}]_r : |p(y) - \mathsf{PUR}[\mathcal{R}]_r(y)| > \varepsilon\right] \geq \frac{2}{3}$$

Note that Theorem 9 rules out approximating polynomials that may be unbounded outside of the domain of $\mathsf{PUR}[\mathcal{R}]_r$. That is, it asserts that there is no low-degree approximating polynomial even when that polynomial is permitted to take values outside of $[0,1]$ on points outside of the domain of PUR_r. Note also that since the lower bound applies for all $\varepsilon < 1/2$, it actually entails a threshold degree lower bound on computing $\mathsf{PUR}[\mathcal{R}]$.

Proof. Fix an arbitrary *good base* $\mathcal{R}$ of size m.

For convenience in this proof, let us change notation to consider polynomials approximations over $\{-1,1\}$ instead of over $\{0,1\}$. Define $\mathcal{Y}' : \mathcal{R} \times \{-1,1\}^n \to \{-1,1\}^m$ to be $(\mathcal{Y}'(r,x'))_i = 1 - 2(\mathcal{Y}(r,(\frac{1-x'_1}{2}, \frac{1-x'_2}{2}, \ldots, \frac{1-x'_n}{2})))_i = 1-2\langle x, r_i\rangle$ where r_i is the vector corresponding to ith component of $\mathcal{Y}(r,x)$ and $x \in \{0,1\}^n$ is the vector that corresponds to $x' \in \{-1,1\}^n$: $x_i = \frac{1-x'_i}{2}$ for all $i \in [n]$. Notice that this change of notation satisfies the following: if $a \in \{0,1\}$ and a' is the corresponding value in the new notation $a' \in \{-1,1\}$ then $a' = (-1)^a$.

Let's also rewrite PUR_r in this new notation. Let D'_r represent the domain of PUR'_r: $D'_r = \{\mathcal{Y}'(r,x') \mid x' \in \{-1,1\}^n\}$ and the function $\mathsf{PUR}'_r : D'_r \to \{-1,1\}$ be $\mathsf{PUR}'_r(\mathcal{Y}'_1, \mathcal{Y}'_2, \ldots, \mathcal{Y}'_m) = 1 - 2\mathsf{PUR}_r(\frac{1-\mathcal{Y}'_1}{2}, \frac{1-\mathcal{Y}'_2}{2}, \ldots, \frac{1-\mathcal{Y}'_m}{2})$.

Note that every polynomial $p' : \{-1,1\}^m \to \mathbb{R}$ that approximates PUR'_r to error ε can be converted by a linear transformation into a polynomial $p : \{0,1\}^m \to \mathbb{R}$ of the same degree that approximates PUR_r to error $\varepsilon/2$. So it suffices to prove that no polynomial p' of degree at most d approximates PUR'_r to error $\varepsilon < 1$.

Assume toward a contradiction that there is a polynomial p' of degree d that approximates PUR'_r. This means that there exists $\varepsilon < 1$ such that for all $y' \in D'_r$,

$$\left|p'(y') - \mathsf{PUR}'_r(y')\right| < \varepsilon.$$

Consider the following expression:

$$\begin{aligned}\frac{1}{2^n}\left|\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')(\mathsf{PUR}'_r(y') - p'(y'))\right| &\le \frac{1}{2^n}\left(\max_{y'\in D'_r}\left|p'(y') - \mathsf{PUR}'_r(y')\right|\right)\left(\sum_{y'\in D'_r}\left|\mathsf{PUR}'_r(y')\right|\right) \\ &< \frac{1}{2^n}\varepsilon|D'_r| = \varepsilon.\end{aligned} \tag{1}$$

The last equality holds because $\mathcal{Y}'(r,\cdot)$ is surjective, and hence $|D'_r| = 2^n$. On the other hand,

$$\begin{aligned}\frac{1}{2^n}\left|\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')(\mathsf{PUR}'_r(y') - p'(y'))\right| &= \frac{1}{2^n}\left|\left(\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')\mathsf{PUR}'_r(y')\right) - \left(\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')p'(y')\right)\right| \\ &= \frac{1}{2^n}\left||D'_r| - \left(\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')p(y')\right)\right|.\end{aligned} \tag{2}$$

We now show that with high probability the expression above is equal to $\frac{|D'_r|}{2^n}$.

Claim 10. With probability at least $\frac{2}{3}$ over the choice of $r \leftarrow \mathcal{R}$, for every polynomial $p' : \{-1,1\}^m \to \mathbb{R}$ of degree at most $d = \frac{n}{4\log m} - 1$ we have

$$\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')p'(y') = 0.$$

Proof. Fix a polynomial p' of degree at most $d = \frac{n}{4\log m} - 1$. By linearity it suffices to consider the case where p' is a monomial, $p'(y') = \prod_{j\in T} y'_j$ for some $T \subseteq [m], |T| \leq d$. So

$$\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')p'(y') = \sum_{x'\in\{-1,1\}^n} \left(\prod_{i\in[n]} (x'_i)\right)\left(\prod_{j\in T} (\mathcal{Y}'(r,x'))_j\right) = \sum_{x\in\{0,1\}^n} \left((-1)^{\langle x,1^n\rangle}\right)\left(\prod_{j\in T} (-1)^{\langle x,r_j\rangle}\right)$$

$$= \sum_{x\in\{0,1\}^n} (-1)^{\langle x,1^n\rangle}(-1)^{\sum_{j\in T}\langle x,r_j\rangle} = \sum_{x\in\{0,1\}^n} (-1)^{\langle x,1^n\rangle}(-1)^{\langle x,\bigoplus_{j\in T} r_j\rangle} = \sum_{x\in\{0,1\}^n} (-1)^{\langle x,1^n\oplus(\bigoplus_{j\in T} r_j)\rangle}$$

This expression is not zero if and only if $\bigoplus_{j\in T} r_j = 1^n$. By Claim 8 the probability that such T exists is at most $\frac{1}{3}$. So the probability over the choice of r for some polynomial $p' : \{-1,1\}^m \to \mathbb{R}$ of degree at most $d = \frac{n}{4\log m} - 1$ to have

$$\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')p'(y') \neq 0$$

is at most $\frac{1}{3}$. □

Combining expressions (1) and (2) and Claim 10, we have that with probability at least $\frac{2}{3}$,

$$\varepsilon > \frac{1}{2^n}\left|\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')(\mathsf{PUR}'_r(y') - p'(y'))\right| = \frac{1}{2^n}\left||D'_r| - \left(\sum_{y'\in D'_r} \mathsf{PUR}'_r(y')p'(y')\right)\right| = \frac{|D'_r|}{2^n} = 1.$$

And so $\varepsilon > 1$ which contradicts our assumption. Thus $\mathsf{PUR}'_r(\mathcal{Y}(r,x))$ cannot be approximated by a polynomial of degree at most $\frac{n}{4\log m} - 1$ with probability at least $\frac{2}{3}$ over the choice $r \leftarrow \mathcal{R}$ sampled uniformly at random. And therefore $\mathsf{PUR}_r(\mathcal{Y}(r,x))$ cannot be ε-approximated for every constant $\varepsilon < \frac{1}{2}$ with a polynomial of degree less than $\frac{n}{4\log m}$ with probability at least $\frac{2}{3}$ over the choice $r \leftarrow \mathcal{R}$ sampled uniformly at random for any *good base* $\mathcal{R}$ of size m. □

3.5 Lower bound for ordered search

Finally, we combine our general lower bound on the approximate degree of PUR with the upper bound on approximating GT_i to conclude our lower bound on the approximate degree of ordered search. We will use the statement of Claim 7 with a lower degree of polynomials approximating GT_i since, even though its proof is more complicated than the proof of the weaker bound, as it allows us to obtain a better lower bound on ordered search.

First, we apply Theorem 9 to obtain a lower bound on the approximate degree for $\mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]$.

Corollary 11. If $r \leftarrow \mathcal{R}_{\mathsf{OS}++}$ u.a.r. then with probability at least $\frac{2}{3}$ over the choice of r every polynomial $p : \{0,1\}^m \to \mathbb{R}$ of degree at most $d = \frac{n}{16\log n} - 1$ fails to approximate $\mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]_r$:

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{OS}++}}\left[\forall\varepsilon < \frac{1}{2},\ \forall p, \deg(p) \leq d, \exists y \in D[\mathcal{R}_{\mathsf{OS}++}]_r : |p(y) - \mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]_r(y)| > \varepsilon\right] \geq \frac{2}{3}.$$

Proof. The set $\mathcal{R}_{\mathsf{OS}++}$ is a *good base* and has size $m = O(n^3 \log n)$. By Theorem 9, with probability at least $\frac{2}{3}$ over the choice of r every polynomial $p : \{0,1\}^m \to \mathbb{R}$ of degree at most $\frac{n}{4 \log m} - 1$ fails to approximate $\mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]_r$. But since the size of $\mathcal{R}_{\mathsf{OS}++}$ is $m \leq n^4$ for sufficiently large n then every polynomial of degree at most $d = \frac{n}{16 \log n} - 1 = \frac{n}{4 \log n^4} - 1 \leq \frac{n}{4 \log m} - 1$ fails to approximate $\mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]_r$. □

By combining Claim 7 and Corollary 11, we obtain the following.

Theorem 12. The approximate degree of ordered search is

$$\widetilde{\deg}_{\frac{1}{2}-\gamma}(\mathsf{OS}_{2^n}) = \Omega\left(\frac{n}{\log^2 n} - \log\frac{1}{\gamma}\right)$$

where γ could depend on n, $0 < \gamma < \frac{1}{2}$.

Proof. Suppose OS_{2^n} can be $(\frac{1}{2} - \gamma)$-approximated by a bounded polynomial of degree d for some $\frac{1}{2} > \gamma > 0$. By [She12a, Theorem 1.1], for every $\delta > 0$, this polynomial can be converted to a polynomial p of degree $O(d + \log \frac{1}{\delta})$ that $(\frac{1}{2} - \gamma + \delta)$-approximates OS_{2^n} and is robust to noise in its inputs. That is,

$$|\mathsf{OS}_N(y) - p(y + \Delta)| < \frac{1}{2} - \gamma + \delta$$

for all $y \in \{0,1\}^N$, all $\Delta \in [-\frac{1}{6}, \frac{1}{6}]^N$, and $N = 2^n$. If we put $\delta = \frac{\gamma}{2}$, then p is a $(\frac{1}{2} - \frac{\gamma}{2})$-approximating polynomial for OS_{2^n} with degree $O\left(d + \log\left(\frac{1}{\gamma}\right)\right)$.

Note that $\mathsf{OS}_{2^n}(\mathsf{GT}_{0^n}(x), \mathsf{GT}_{0^{n-1}1}(x), \ldots, \mathsf{GT}_{1^n}(x)) = \mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]_r(\mathcal{Y}(r,x))$ for every $r \in \mathcal{R}_{\mathsf{OS}++}$. So by Claim 7, there exists a constant c such that the composed polynomial $p(q_{(r,0^n)}(\mathcal{Y}(r,x)), q_{(r,0^{n-1}1)}(\mathcal{Y}(r,x)), \ldots q_{(r,1^n)}(\mathcal{Y}(r,x)))$ has degree at most $\deg(p) \max_i(\deg(q_{(r,i)})) = c\left(d + \log\left(\frac{1}{\gamma}\right)\right) \log n$ and approximates $\mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]_r(\mathcal{Y}(r,x))$ to error $(\frac{1}{2} - \frac{\gamma}{2})$ with probability at least $\frac{2}{3}$ over the choice of $r \leftarrow \mathcal{R}_{\mathsf{OS}++}$. This holds because although the polynomials $q_{(r,i)}$ do not compute the functions GT_i exactly, but only approximate them with small error, the outer polynomial p is robust to this small error in the inputs. Note also that while the composed polynomial is bounded on the domain of PUR_r, it may be arbitrarily unbounded on points outside its domain.

On the other hand, by Claim 11, with probability at least $\frac{2}{3}$ over the choice of r, the function $\mathsf{PUR}[\mathcal{R}_{\mathsf{OS}++}]_r$ cannot be approximated to any error $(\frac{1}{2} - \frac{\gamma}{2}) \in (0, \frac{1}{2})$ by a polynomial in $\mathcal{Y}$ of degree less than $\frac{n}{16 \log n}$. By a union bound, with probability at least $1 - (1 - \frac{2}{3}) - (1 - \frac{2}{3}) = \frac{1}{3}$ both conditions on r hold simultaneously. Thus there exists $r \in \mathcal{R}_{\mathsf{OS}++}$ such that $p(q_{(r,0^n)}(\mathcal{Y}(r,x)), q_{(r,0^{n-1}1)}(\mathcal{Y}(r,x)), \ldots q_{(r,1^n)}(\mathcal{Y}(r,x)))$ approximates $\mathsf{PUR}_r(\mathcal{Y}(r,x))$ and $\mathsf{PUR}_r(\mathcal{Y}(r,x))$ cannot be approximated by a polynomial of degree less than $\frac{n}{16 \log n}$. So

$$c\left(d + \log\left(\frac{1}{\gamma}\right)\right) \log n \geq \frac{n}{16 \log n}.$$

And thus

$$d + \log\left(\frac{1}{\gamma}\right) \geq \frac{n}{16c \log^2 n},$$

so we conclude that

$$d = \Omega\left(\frac{n}{\log^2 n} - \log\left(\frac{1}{\gamma}\right)\right).$$

□

4 Anchored hidden string and hidden string

Now let us switch gears and consider the (anchored) hidden string problem, in which the goal is to reconstruct a string given information about the presence of specific substrings. In the decisional anchored hidden string (AHS) problem, the information given as input consists of whether each string s is a substring of the hidden input x starting at position i, for all valid i and s. The goal is then to compute $\mathsf{parity}(x)$.

In order to prove a lower bound for AHS we will follow the same outline as for OS. That is, first we will introduce a convenient set $\mathcal{R}_{\mathsf{AHS}}$ of collections of n-bit strings and show that oracle $\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}]$ providing the inner products of x with strings from the random sample from $\mathcal{R}_{\mathsf{AHS}}$ are useful for computing $\phi_{i,s}(x)$ for all possible queries (i, s) where $i \in [n], s \in \{0,1\}^{\leq n-i+1}$ and $\phi_{i,s}(x) = 1$ iff s is a substring of x starting from position i. After that, we will show that it is hard to compute $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]$, the parity of x using the oracle $\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}]$ with high probability. And finally, we will conclude that computing AHS is hard since composing an approximating polynomial for AHS with polynomials approximating $\phi_{i,s}(x)$ would allow us to approximate the $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]$ function.

4.1 Preliminaries

W define several functions in order to formalize the problem.

Throughout this section, we use the following notation $\{0,1\}^{\leq n}$ to denote the set of all bit strings of size at most n: $\{0,1\}^{\leq n} = \bigcup_{k=0}^{n} \{0,1\}^k$.

Definition 13. For all $s \in \{0,1\}^{\leq n}$ define the function $\chi_s : \{0,1\}^n \to \{0,1\}$ to be the indicator of whether the input string x has s as a substring: that is, there exists an integer i such that $x_{i+k-1} = s_k$ for all $1 \leq k \leq |s|$ then $\chi_s(x) = 1$ and otherwise $\chi_s(x) = 0$.

Definition 14. Define the "hidden string" function $\mathsf{HS}_N : \{0,1\}^N \to \{0,1\}$ be the partial function that takes $N = |\{0,1\}^{\leq n}| = 2^{n+1} - 1$ inputs, each corresponding to a substring $s \in \{0,1\}^{\leq n}$, and, given a collection of $\chi_s(x)$ for some fixed $x \in \{0,1\}^n$ as an input, outputs $\mathsf{parity}(x)$.

We will also consider a variation of this problem where the additional information is not only whether a specific substring is present in the hidden string, but if this substring is present at a specific location of the hidden string.

Definition 15. For all $i \in [n]$ and $s \in \{0,1\}^{\leq n-i+1}$ define the function $\phi_{i,s} : \{0,1\}^n \to \{0,1\}$ to be the indicator of whether the input string x has s as a substring starting from position i: that is, if $x_{i+k-1} = s_k$ for all $1 \leq k \leq |s|$ then $\phi_{i,s}(x) = 1$ and otherwise $\phi_{i,s}(x) = 0$.

Definition 16. Let the "anchored hidden string" function $\mathsf{AHS}_N : \{0,1\}^N \to \{0,1\}$ be the partial function that takes $N = |\{(i,s) \mid i \in [n], s \in \{0,1\}^{\leq n-i+1}\}| = 2^{n+2} - n - 4$ inputs, each corresponding to a pair of $i \in [n]$ and $s \in \{0,1\}^{\leq n-i+1}$, and, given a collection of $\phi_{i,s}(x)$ for some fixed $x \in \{0,1\}^n$ as an input, outputs $\mathsf{parity}(x)$.

4.2 Approximating polynomials for $\phi_{i,s}$

We start our proof by showing that for some *good base* $\mathcal{R}_{\mathsf{AHS}}$ the oracle $\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}]$ could be used to make the computation of the functions $\phi_{i,s}$ more efficient.

Claim 17. There exists a *good base* $\mathcal{R}_{\mathsf{AHS}}$ of size $m = O(n^3)$ such that if $r \leftarrow \mathcal{R}_{\mathsf{AHS}}$ is sampled uniformly at random, then with probability at least $\frac{2}{3}$ over the choice of r there exists a family of polynomials $\{q_{(r,i,s)} : \{0,1\}^m \to \{0,1\} \mid i \in [n], s \in \{0,1\}^{\leq n-i+1}\}$ of degree at most 4 such that given $\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}](r,x)$ as the input, each polynomial $q_{(r,i,s)}(\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}](r,x)$ approximates the corresponding $\phi_{i,s}(x)$ with error at most $\frac{1}{6}$. That is,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{AHS}}}\left[\exists i \in [n], x \in \{0,1\}^n, s \in \{0,1\}^{\leq n-i+1} : \left|q_{(r,i,s)}(\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}](r,x)) - \phi_{i,s}(x)\right| > \frac{1}{6}\right] < \frac{1}{3}.$$

Proof. The proof of this statement follows the same outline as the proof of Claim 6. First, we will construct a *good base* $\mathcal{R}_{\mathsf{AHS}}$, and then we will show (in two stages) how to compute $\phi_{i,s}$ given $\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}]$.

Constructing the *good base* $\mathcal{R}_{\mathsf{AHS}}$. Our base for the oracle should contain all the strings needed to check the equality with every substring of x, so let $\hat{\mathcal{R}} = \mathcal{R}^{1^n} \times \left(\mathcal{R}^{1^{n-1}0^1} \times \mathcal{R}^{0^1 1^{n-1}}\right) \times \left(\mathcal{R}^{1^{n-2}0^2} \times \mathcal{R}^{0^1 1^{n-1} 0^1} \times \mathcal{R}^{0^2 1^{n-2}}\right) \times \ldots \times \left(\bigtimes_{i=0}^{k} \mathcal{R}^{0^i 1^{i+k} 0^{n-i-k}}\right) \times \ldots \times \left(\bigtimes_{i=0}^{n-1} \mathcal{R}^{0^i 1^{i+1} 0^{n-(i+1)}}\right)$. See Figure 4 for an illustration.

τ	$\mathcal{R}^\tau$	Structure of $\mathcal{R}^\tau$
1111	$\{0,1\}^4$	⋆ ⋆ ⋆ ⋆
1110	$\{0,1\}^3\{0\}$	⋆ ⋆ ⋆ 0
0111	$\{0\}\{0,1\}^3$	0 ⋆ ⋆ ⋆
1100	$\{0,1\}^2\{0\}^2$	⋆ ⋆ 0 0
0110	$\{0\}\{0,1\}^2\{0\}$	0 ⋆ ⋆ 0
0011	$\{0\}^2\{0,1\}^2$	0 0 ⋆ ⋆
1000	$\{0,1\}\{0\}^3$	⋆ 0 0 0
0100	$\{0\}\{0,1\}\{0\}^2$	0 ⋆ 0 0
0010	$\{0\}^2\{0,1\}\{0\}$	0 0 ⋆ 0
0001	$\{0\}^3\{0,1\}$	0 0 0 ⋆

Figure 4: Structure of $\hat{\mathcal{R}}$ for $n = 4$. Blue cells with $\star$ represent places where either 0 or 1 values could be.

On top of this, we are going to add two other steps to the structure. First, we are going to have t individual "prepackaged" copies. Let $\mathcal{R}_1 = \ldots = \mathcal{R}_t = \times_\alpha \hat{\mathcal{R}}$, and $\mathcal{R}' = \bigtimes_{j \in [t]} \mathcal{R}_j$.

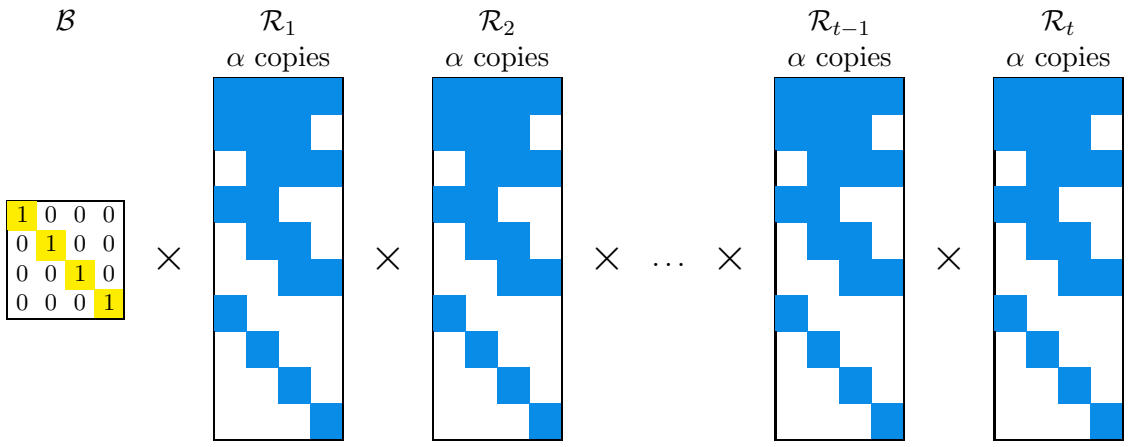

Figure 5: Structure of $\mathcal{R}_{\mathsf{AHS}}$ for $n = 4$. Each $\mathcal{R}_j$ consist of α copies of $\hat{\mathcal{R}}$.

Secondly, we add a set of "basis" strings to the structure: $\mathcal{B} = \{\mathbf{1}_1\} \times \{\mathbf{1}_2\} \times \ldots \times \{\mathbf{1}_{n-1}\} \times \{\mathbf{1}_n\} = \{10\ldots0\} \times \{010\ldots0\} \times \ldots \times \{00\ldots010\} \times \{00\ldots01\}$. The final underlying structure of the oracle will be a Cartesian product of all t copies and $\mathcal{B}$: $\mathcal{R} = \mathcal{B} \times \mathcal{R}' = \mathcal{B} \times (\bigtimes_{j\in[t]} \mathcal{R}_j)$. See Figure 5 for an illustration. We also set the parameters to be $\alpha = 4, t = 1000n \ln 2$. Notice that this set $\mathcal{R}_{\mathsf{AHS}}$ is a *good base* by construction, and has size $m = n + \alpha t \frac{n(n+1)}{2} = O(n^3)$.

Constructing the family of approximating polynomials. For all $i \in [n], s \in \{0,1\}^{\leq n-i+1}, j \in [t], r \in \mathcal{R}_{\mathsf{AHS}}$ let $B_{(r,i,s,j)}(\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}](r,x))$ be the following deterministic algorithm.

1. Set $\tau = 0^i 1^{|s|} 0^{n-|s|-i}$
2. Set $s' = 0^i s 0^{n-|s|-i}$ so s' is an n-bit string
3. For all $v \in [m]$ such that v corresponds to n-bit strings drawn from $\mathcal{R}^\tau$ within the jth copy $\mathcal{R}_j$:
4. Compute $\langle s', r_v\rangle$ and compare it to $(\mathcal{Y}(r,x))_v = \langle x, r_v\rangle$.
5. If for some v the inner products don't have the same value, $\langle i, r_v\rangle \neq (\mathcal{Y}(r,x))_v$, then reject.
6. Otherwise, accept.

This algorithm determines if s equal to the substring of x of length $|s|$ that starts at the ith position with probability at least $1 - 2^{-\alpha} = 1 - 2^{-4} \geq \frac{11}{12}$ independently of the choice of $j \in [t]$. That is, for all $j \in [t]$ and for all $i \in [n], s \in \{0,1\}^{n-i}, x \in \{0,1\}^n$

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}[B_{(r,i,s,j)}(\mathcal{Y}(r,x)) = \phi_{i,s}(x)] \geq \frac{11}{12}.$$

This algorithm makes at most $\alpha = 4$ queries to the oracle $\mathcal{Y}(r,x)$.

We have shown that for every fixed $i \in [n], s \in \{0,1\}^{\leq n-i+1}$, and $x \in \{0,1\}^n$ there are many $r \in \mathcal{R}_{\mathsf{AHS}}$ that if used as the first input for the oracle $\mathcal{Y}$ allow $B_{(r,i,s,j)}$ to compute $\phi_{i,s}(x)$.

Let $W(i,s,x,r,j)$ be the indicator that the j-th "package" of random strings in r defines a set of "bad" random strings for i,s,x: $W(i,s,x,r,j) = 1$ if and only if $B_{(r,i,s,j)}(\mathcal{Y}(r,x)) \neq \phi_{i,s}(x)$. We

established that $B_{(r,i,s,j)}(\mathcal{Y}(r,x))$ works well if given a random $r \leftarrow \mathcal{R}_{\mathsf{AHS}}$ for every $j \in [t]$ and the probability of this algorithm outputting an incorrect answer is at most $\frac{1}{12}$. So

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}[W(i,s,x,r,j)=1] = \mathbb{E}_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}[W(i,s,x,r,j)] \leq \frac{1}{12}.$$

Using the same transformation from before (described in detail in the proof of Claim 6), we design a new family of algorithms that succeeds on all i,s,x simultaneously. For all $i \in [n], s \in \{0,1\}^{\leq n-i+1}, r \in \mathcal{R}_{\mathsf{OS++}}$ let $A_{(r,i,s)}(\mathcal{Y}(r,x))$ be the following randomized algorithm:

- Choose $j \leftarrow [t]$ uniformly at random.
- Run $B_{(r,i,s,j)}(\mathcal{Y}(r,x))$.

The number of queries that $A_{(r,i,s)}$ makes to the oracle is the same as $B_{(r,i,s,j)}$ which is $\alpha = 4$. We fix arbitrary i,s,x and evaluate the following probability.

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}\left[\Pr_{j\leftarrow[t]}[B_{(r,i,s,j)} \neq \phi_{i,s}(x)] > \frac{1}{6}\right] = \Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}\left[\frac{1}{t}\sum_{j\in[t]} W(i,s,x,r,j) > \frac{1}{6}\right].$$

We established that $\mathbb{E}_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}[W(i,s,x,r,j)] \leq \frac{1}{12}$ and so by Hoeffding's inequality,

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}\left[\frac{1}{t}\sum_{j\in[t]} W(i,s,x,r,j) > \frac{1}{12}+\frac{1}{12}\right] \leq e^{-2\frac{t}{144}} \leq 2^{-\frac{2000n}{144}}.$$

And by a union bound,

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}\left[\exists i\in[n], s\in\{0,1\}^{\leq n-i+1}, x\in\{0,1\}^n : \frac{1}{t}\sum_{j\in[t]} W(i,s,x,r,j) > \frac{1}{6}\right] \leq 2^{2n+2}2^{-\frac{2000n}{144}} < \frac{1}{3},$$

since the number of possible pairs of $i \in [n]$ and $s \in \{0,1\}^{\leq n-i+1}$ is $\sum_{i=1}^{n}\sum_{|s|=1}^{n-i+1} 2^{|s|} \leq 2^{n+2}$. So, we proved that

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}\left[\exists i\in[n], s\in\{0,1\}^{\leq n-i+1}, x\in\{0,1\}^n : \Pr_{j\leftarrow[t]}[A_{(r,i,s)}(\mathcal{Y}(r,x))) \neq \phi_{i,s}(x)] > \frac{1}{6}\right] < \frac{1}{3}.$$

The last step is to convert this family of query algorithms into a family of approximating polynomials. Let $q_{(r,i,s)}$ denote the acceptance probability of $A_{(r,i,s)}$ which is a polynomial of degree at most $\alpha = 4$ such that

$$\Pr_{r\leftarrow\mathcal{R}_{\mathsf{AHS}}}\left[\exists i\in[n], s\in\{0,1\}^{\leq n-i+1}, x\in\{0,1\}^n : \left|q_{(r,i,s)}(\mathcal{Y}[\mathcal{R}_{\mathsf{AHS}}](r,x)) - \phi_{i,s}(x)\right| > \frac{1}{6}\right] < \frac{1}{3}$$

which is exactly what we were looking for.

□

4.3 Lower bounds for anchored hidden string and hidden string

Following the same framework, we can combine the general statement about the degree of PUR proven earlier with the upper bound for approximating $\phi_{i,s}$ to conclude our lower bound on the approximate degree of anchored hidden string.

First, we apply Theorem 9 to obtain a lower bound on approximate degree for $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]$.

Corollary 18. If $r \leftarrow \mathcal{R}_{\mathsf{AHS}}$ u.a.r. then with probability at least $\frac{2}{3}$ over the choice of r every polynomial $p : \{0,1\}^m \to \mathbb{R}$ of degree at most $d = \frac{n}{16 \log n} - 1$ fails to approximate $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]_r$:

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{AHS}}} \left[\forall \varepsilon < \frac{1}{2},\ \forall p, \deg(p) \leq d, \exists y \in D[\mathcal{R}_{\mathsf{AHS}}]_r : |p(y) - \mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]_r(y)| > \varepsilon \right] \geq \frac{2}{3}.$$

Proof. The set $\mathcal{R}_{\mathsf{AHS}}$ is a *good base* and has size $m = O(n^3)$. By Theorem 9, with probability at least $\frac{2}{3}$ over the choice of r every polynomial $p : \{0,1\}^m \to \mathbb{R}$ of degree at most $\frac{n}{4 \log m} - 1$ fails to approximate $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]_r$. But since the size of $\mathcal{R}_{\mathsf{AHS}}$ is $m \leq n^4$ for sufficiently large n then every polynomial of degree at most $d = \frac{n}{16 \log n} - 1 = \frac{n}{4 \log n^4} - 1 \leq \frac{n}{4 \log m} - 1$ fails to approximate $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]_r$. □

By combining Claim 17 and Corollary 18, we obtain the following.

Theorem 19. The approximate degree of AHS_N is

$$\widetilde{\deg}_{\frac{1}{2}-\gamma}(\mathsf{AHS}_N) = \Omega\left(\frac{n}{\log n} - \log \frac{1}{\gamma}\right)$$

where $N = 2^{n+2} - n - 4$ and γ could depend on n, $0 < \gamma < \frac{1}{2}$.

Proof. Suppose AHS_N can be $(\frac{1}{2} - \gamma)$-approximated by a bounded polynomial of degree d.

By the same argument as used in Theorem 12 we can conclude that there exists a polynomial p of degree $O(d + \log 1\gamma)$ that $(\frac{1}{2} - \frac{\gamma}{2})$-approximates AHS_N and is robust to noise. That is,

$$|\mathsf{AHS}_N(y) - p(y + \Delta)| < \frac{1}{2} - \frac{\gamma}{2}$$

for all $y \in \{0,1\}^N$ where $\Delta \in [-\frac{1}{6}, \frac{1}{6}]^N$.

Note that $\mathsf{AHS}_N((\phi_{i,s}(x))_{i \in [n], s \in \{0,1\}^{\leq n-i+1}}) = \mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]_r(\mathcal{Y}(r,x))$ for every $r \in \mathcal{R}_{\mathsf{AHS}}$. So by Claim 17 the polynomial $p(q_{(r,i,s)}(\mathcal{Y}(r,x)))$ of degree at most $\deg(p) \max_{i,s}(\deg(q_{(r,i,s)})) = 4c(d + \log \frac{1}{\gamma})$ for some constant c approximates $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]_r(\mathcal{Y}(r,x))$ to error $(\frac{1}{2} - \frac{\gamma}{2})$ with probability at least $\frac{2}{3}$ over the choice of $r \leftarrow \mathcal{R}_{\mathsf{AHS}}$. This holds because although the polynomials $q_{(r,i,s)}$ do not compute the functions $\phi_{i,s}$ exactly, but only approximate them with small error, the outer polynomial p is robust to this small error in the inputs. Note also that while the composed polynomial is bounded on the domain of PUR_r, it may be arbitrarily unbounded on points outside its domain.

On the other hand, by Claim 18, with probability at least $\frac{2}{3}$ over the choice of r, the function $\mathsf{PUR}[\mathcal{R}_{\mathsf{AHS}}]_r$ cannot be approximated by a polynomial in $\mathcal{Y}$ of degree less than $\frac{n}{16 \log n}$. By a union bound, with probability at least $(1 - (1 - \frac{2}{3}) - (1 - \frac{2}{3})) = \frac{1}{3}$ both conditions on r hold simultaneously. Thus there exists $r \in \mathcal{R}_{\mathsf{AHS}}$ such that $p(q_{(r,i,s)}(\mathcal{Y}(r,x)))$ approximates $\mathsf{PUR}_r(\mathcal{Y}(r,x))$ and $\mathsf{PUR}_r(\mathcal{Y}(r,x))$ cannot be approximated by a polynomial of degree less than $\frac{n}{16 \log n}$. So

$$4c(d + \log \frac{1}{\gamma}) \geq \frac{n}{16 \log n}.$$

And thus

$$d = \Omega\left(\frac{n}{\log n} - \log \frac{1}{\gamma}\right).$$

□

This lower bound on the approximate degree of AHS_N entails a lower bound on the approximate degree of HS_N. In [CIG+12] the authors gave a reduction between these two problems, showing that if there exists a quantum query algorithm for HS then there exists a quantum query algorithm for AHS with a small blow-up in the number of queries. Specifically, they showed how to compute AHS by applying a query algorithm for HS with a slightly bigger input, where each bit of the bigger input can be computed using a constant number of queries to the original AHS input. This argument works just as well to relate the approximate degrees of HS and AHS, giving the following statement.

Claim 20. If for every n', there is a polynomial of degree $d(n')$ approximating $\mathsf{HS}_{2^{n'+1}-1}$ to some error, then for every n there is a polynomial of degree $2d(10n \log n)$ approximating $\mathsf{AHS}_{2^{n+2}-n-4}$ to the same error.

This allows us to prove a lower bound for HS_N as well.

Corollary 21. The approximate degree of HS_N is

$$\widetilde{\deg}_{\frac{1}{2}-\gamma}(\mathsf{HS}_N) = \Omega\left(\frac{n}{\log^2 n} - \log\left(\frac{1}{\gamma}\right)\right)$$

where $N = 2^{n+1} - 1$ and γ could depend on n, $0 < \gamma < \frac{1}{2}$.

Proof. By Claim 20 if there exists a polynomial of degree $d(n')$ approximating $\mathsf{HS}_{2^{n'+1}-1}$ then there exists a polynomial of degree $2d(10n \log n)$ approximating $\mathsf{AHS}_{2^{n+2}-n-2}$. On the other hand, by Corollary 18 no polynomial of degree less than $\frac{cn}{\log n} - c \log\left(\frac{1}{\gamma}\right)$ can approximate $\mathsf{AHS}_{2^{n+2}-n-2}$ to error $\frac{1}{2} - \gamma$ for some constant c. Therefore,

$$2d(10n \log n) \geq \frac{cn}{\log n} - c \log \frac{1}{\gamma}.$$

Set $n' = 10n \log n$. Then

$$d(n') \geq \frac{cn}{2 \log n} - c \log \frac{1}{\gamma} = \frac{cn'}{20 \log^2 n} - c \log \frac{1}{\gamma} \geq \frac{cn'}{20 \log^2 n'} - c \log \frac{1}{\gamma}.$$

And thus

$$\widetilde{\deg}_{\frac{1}{2}-\gamma}(\mathsf{HS}_{2^{n'+1}-1}) = \Omega\left(\frac{n'}{\log^2 n'} - c \log \frac{1}{\gamma}\right).$$

□

Acknowledgments. We thank Arkadev Chattopadhyay for suggesting the problem of determining the approximate degree of ordered search, and Arkadev and Justin Thaler for many helpful conversations about it. We also thank the anonymous TQC 2023 reviewers for helpful suggestions on the presentation.

References

[ABFR91] James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. In *Proceedings of the twenty-third annual ACM symposium on Theory of Computing*, pages 402–409, 1991.

[ABK16] Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 863–876. ACM, 2016.

[AIK+04] Andris Ambainis, Kazuo Iwama, Akinori Kawachi, Hiroyuki Masuda, Raymond H. Putra, and Shigeru Yamashita. Quantum identification of boolean oracles. In Volker Diekert and Michel Habib, editors, *STACS 2004, 21st Annual Symposium on Theoretical Aspects of Computer Science, Montpellier, France, March 25-27, 2004, Proceedings*, volume 2996 of *Lecture Notes in Computer Science*, pages 105–116. Springer, 2004.

[AIK+07] Andris Ambainis, Kazuo Iwama, Akinori Kawachi, Rudy Raymond, and Shigeru Yamashita. Improved algorithms for quantum identification of boolean oracles. *Theor. Comput. Sci.*, 378(1):41–53, 2007.

[AM14] Andris Ambainis and Ashley Montanaro. Quantum algorithms for search with wildcards and combinatorial group testing. *Quantum Inf. Comput.*, 14(5-6):439–453, 2014.

[Amb99] A. Ambainis. A better lower bound for quantum algorithms searching an ordered list. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 352–357, 1999.

[Amb06] Andris Ambainis. Polynomial degree vs. quantum query complexity. *J. Comput. Syst. Sci.*, 72(2):220–238, 2006.

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald De Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001.

[BBGK18] Shalev Ben-David, Adam Bouland, Ankit Garg, and Robin Kothari. Classical lower bounds from quantum upper bounds. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 339–349. IEEE Computer Society, 2018.

[BCdWZ99] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. Bounds for small-error and zero-error quantum algorithms. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 358–368. IEEE Computer Society, 1999.

[BdW99] Harry Buhrman and Ronald de Wolf. A lower bound for quantum search of an ordered list. *Information Processing Letters*, 70(5):205–209, 1999.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[Bel15] Aleksandrs Belovs. Quantum algorithms for learning symmetric juntas via the adversary bound. *Comput. Complex.*, 24(2):255–293, 2015.

[BH08] Michael Ben-Or and Avinatan Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 221–230, 2008.

[BKT20] Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: Tight quantum query bounds via dual polynomials. *Theory Comput.*, 16:1–71, 2020.

[BNRdW07] Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory Comput. Syst.*, 40(4):379–395, 2007.

[BSS01] Howard Barnum, Michael Saks, and Mario Szegedy. Quantum decision trees and semidefinite programming. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2001.

[BT22] Mark Bun and Justin Thaler. Approximate degree in classical and quantum computing. *Found. Trends Theor. Comput. Sci.*, 15(3-4):229–423, 2022.

[BV93] Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 11–20. ACM, 1993.

[CFK+21] Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-to-communication lifting using low-discrepancy gadgets. *SIAM J. Comput.*, 50(1):171–210, 2021.

[CIG+12] Richard Cleve, Kazuo Iwama, François Le Gall, Harumichi Nishimura, Seiichiro Tani, Junichi Teruyama, and Shigeru Yamashita. Reconstructing strings from substrings with quantum queries. In *Scandinavian Workshop on Algorithm Theory*, pages 388–397. Springer, 2012.

[CKLM17] Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Composition and simulation theorems via pseudo-random properties. *Electron. Colloquium Comput. Complex.*, page 14, 2017.

[CL08] Andrew M. Childs and Troy Lee. Optimal quantum adversary lower bounds for ordered search. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 869–880. Springer, 2008.

[CLP07] Andrew M. Childs, Andrew J. Landahl, and Pablo A. Parrilo. Quantum algorithms for the ordered search problem via semidefinite programming. *Phys. Rev. A*, 75:032335, Mar 2007.

[FGGS98] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation for insertion into an ordered list, 1998.

[FGGS99] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Invariant quantum algorithms for insertion into an ordered list, 1999.

[FRPU94] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

[HLS07a] Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 526–535. ACM, 2007.

[HLS07b] Peter Hoyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 526–535, 2007.

[HNS02] Peter Høyer, Jan Neerbek, and Yaoyun Shi. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica*, 34(4):429–448, 2002.

[Hoz17] William M. Hoza. Quantum communication-query tradeoffs. *CoRR*, abs/1703.07768, 2017.

[INRT12] Kazuo Iwama, Harumichi Nishimura, Rudy Raymond, and Junichi Teruyama. Quantum counterfeit coin problems. *Theor. Comput. Sci.*, 456:51–64, 2012.

[Kot14] Robin Kothari. An optimal quantum algorithm for the oracle identification problem. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 482–493. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

[KSdW07] Hartmut Klauck, Robert Spalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM J. Comput.*, 36(5):1472–1493, 2007.

[MS20] Ashley Montanaro and Changpeng Shao. Quantum algorithms for learning graphs and beyond. *CoRR*, abs/2011.08611, 2020.

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991.

[Nis93] Noam Nisan. The communication complexity of threshold gates. In *Combinatorics, Paul Erdös is Eighty, number 1 in Bolyai Society Mathematical Studies*, pages 301–315, 1993.

[Rei11] Ben Reichardt. Reflections for quantum query algorithms. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 560–569. SIAM, 2011.

[She11] Alexander A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

[She12a] Alexander A Sherstov. Making polynomials robust to noise. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 747–758, 2012.

[She12b] Alexander A. Sherstov. Strong direct product theorems for quantum communication and query complexity. *SIAM J. Comput.*, 41(5):1122–1165, 2012.

[She20] Alexander A. Sherstov. Algorithmic polynomials. *SIAM J. Comput.*, 49(6):1173–1231, 2020.

[SS95] Steven Skiena and Gopalakrishnan Sundaram. Reconstructing strings from substrings. *J. Comput. Biol.*, 2(2):333–353, 1995.

[vD98] Wim van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 362–367. IEEE Computer Society, 1998.

A Upper bounds for the unbounded error regime

We describe upper bounds on the randomized (and hence, quantum) query complexities of the ordered search and hidden string problems in the setting of weakly unbounded error. Our algorithms are simple modifications of the corresponding deterministic algorithms. They show that the approximate degree and quantum query lower bounds we prove for these problems are nearly tight.

A.1 Ordered search

Reconstruction. We first describe a randomized query algorithm that computes ordered search (reconstruction version) with probability at least γ while making $O(n - \log \frac{1}{\gamma})$ queries.

To attempt to identify a hidden string x, the algorithm makes the first t queries of binary search and thus exactly identifies the first t bits of x. Then it samples the rest of the bits uniformly at random and outputs the resulting n-bit string. It succeeds in sampling the correct sequence of the last $(n-t)$ bits with probability at least $2^{-(n-t)}$. The upper bound follows by setting $t = n - \log \frac{1}{\gamma}$.

Decision. We now modify the algorithm above to compute the decision version of ordered search with probability at least $\frac{1}{2}+\gamma$, while making $O(n-\log\frac{1}{\gamma})$ queries.

The algorithm makes the first t queries of binary search to exactly identify the first t bits of x. Then it samples the rest of the bits uniformly at random, obtaining an n-bit candidate string x'. It then queries the input twice, on indices $x'-1$ and x', to check that $x \not\leq x'-1$ and $x \leq x'$. If both conditions hold, then $x'=x$ and the algorithm outputs outputs $\mathsf{parity}(x')=\mathsf{parity}(x)$. Otherwise, it outputs a random bit.

This algorithm succeeds when either it succeeds at identifying x (which happens with probability $2^{-(n-t)}$) or when it fails to identify x, but correctly guesses its parity, which happens with probability $\frac{1}{2}(1-2^{-(n-t)})$. Thus the algorithm succeeds with probability at least $\frac{1}{2}+2^{-(n-t)-1}$. The upper bound follows by setting $t=n+1-\log\frac{1}{\gamma}$.

A.2 Hidden string

Since our unbounded-error algorithm relies on the deterministic algorithm for the hidden string, we provide a sketch of that algorithm here.

Theorem. ([SS95]) There exists a deterministic query algorithm that computes hidden string (reconstruction version) using $O(n)$ queries.

Here is the sketch of the algorithm:

1. Let s be the empty string.
2. While either $s0$ or $s1$ is present in x:
3. Update s to be either $s0$ or $s1$, whichever is present
4. While either $0s$ or $1s$ is present in x:
5. Update s to be either $0s$ or $1s$, whichever is present

This algorithm makes at most $2n+2$ queries to the input and outputs the hidden string x.

Reconstruction. We now describe a randomized query algorithm that computes hidden string (reconstruction version) with probability at least γ while making $O(n-\log\frac{1}{\gamma})$ queries.

The algorithm makes the first t steps of the exact deterministic algorithm above and thus exactly identifies t bits of x. Then it samples the rest of the bits uniformly at random, samples a location among these $n-t$ bits in which to insert the identified substring of x, and outputs the resulting n-bit string. It succeeds in guessing the location for the substring with probability at least $\frac{1}{n-t+1}$ and it succeeds in guessing the correct values of the rest of the bits with probability at least $2^{-(n-t)}$. The upper bound follows by setting $t=2n-\log\frac{1}{\gamma}$.

Decision. We now describe a randomized query algorithm that computes hidden string (decision version) with probability at least $\frac{1}{2}+\gamma$ while making $O(n-\log\frac{1}{\gamma})$ queries.

The algorithm makes the first t steps of the exact deterministic algorithm above and thus exactly identifies t bits of x. Then it samples the rest of the bits uniformly at random, samples where to split this string of random bits to put the identified substring of x, thus getting an n-bit string x'.

It then queries if x' is a substring of x. Since $|x'| = |x|$, this is equivalent to checking if $x' = x$. If the answer is "yes", then the algorithm succeeded in identifying x and it outputs $\mathsf{parity}(x') = \mathsf{parity}(x)$. Otherwise, it samples a random bit and outputs it.

This algorithm succeeds when either it successfully identifies x (which happens with probability $\frac{1}{n-t}2^{-(n-t)}$) or when it fails to identify x but correctly guesses the value of its parity, which happens with probability $\frac{1}{2}(1 - \frac{1}{n-t}2^{-(n-t)})$. Thus the algorithm succeeds with probability at least $\frac{1}{2} + \frac{1}{n-t}2^{-(n-t)-1}$. The upper bound follows by taking $t = 2n + 1 - \log\frac{1}{\gamma}$.

B Proof of Claim 7

The communication protocol that we use to construct our polynomials itself is based on a result by [FRPU94] on algorithms in a noisy comparison model. To understand the protocol and to convert it to the family of polynomials later we need to open up the protocol and state their result.

The following is implicit in [FRPU94]:

Claim 22. Consider the following problem. There is an unknown "key" in $[n]$. Given the ability to ask questions of the type "is the unknown key greater than a?" for every $a \in [n]$ and get the correct answer with probability at least $\frac{3}{4}$ independently for each question, the algorithm's goal is to find the correct location in $(0, n]$ for the key while minimizing the number of questions it asks.

Then there exists an algorithm that finds the correct location in $(0, n]$ for the key by asking at most $c \log n$ questions for some constant c with probability at least $\frac{11}{12}$.

The algorithm basically performs a binary search for the correct location of the key with slight modifications. We are not going to present the algorithm in detail, but we are going to describe what questions the algorithm asks along the way. For each interval $(a, b]$ where $b - a > 1$, the algorithm seeks answers to the following questions: "is the key $> a$?", "is the key $> b$?" and "is the key $> \frac{a+b}{2}$?", each correct with probability $\frac{3}{4}$ independently from all other questions, and the algorithm needs to be able to ask each potential question of each type $c \log n$ times for some constant c. For each interval $(a, a+1]$ the algorithm seeks answers to the following questions: "is the key $> a$?" and "is the key $> a + 1$?", each correct with probability at least $\frac{3}{4}$ independently from all other questions, and the algorithm needs to be able to ask each potential question of each type $2c^2 \log^2 n$ times. Note that if the range for the key position is $(0, n]$ then questions "is the key > 0?" and "is the key $> n$?" can be omitted by the algorithm since it already knows the correct answer to them.

The efficient communication protocol for the GT heavily relies on the algorithm above. In the communication protocol, both Alice and Bob run the algorithm to find the most significant bit where their inputs differ. Each time the algorithm asks "is the position of the most significant bit where the inputs differ greater than a?" (or "is the key greater than a?"), Alice and Bob compute the equality of the first a bits of their inputs to error $\frac{1}{4}$ by computing $\alpha = 2$ inner products of their inputs with random strings from $\mathcal{R}^\tau$ where $\tau = 1^a 0^{n-a}$. See Figure 6 for an illustration. If their inner products are the same then the answer to the algorithm is "no, the key is not greater than a" and otherwise, it's "yes, the key is greater than a". And at the end of the protocol, Alice and Bob check who has a greater value in the most significant bit discovered during the procedure. Now we are ready to prove the claim.

Constructing the *good base* $\mathcal{R}_{\mathsf{OS}++}$. The construction is similar to the construction of $\mathcal{R}_{\mathsf{OS}}$. Let $\hat{\mathcal{R}}$ have the following structure.

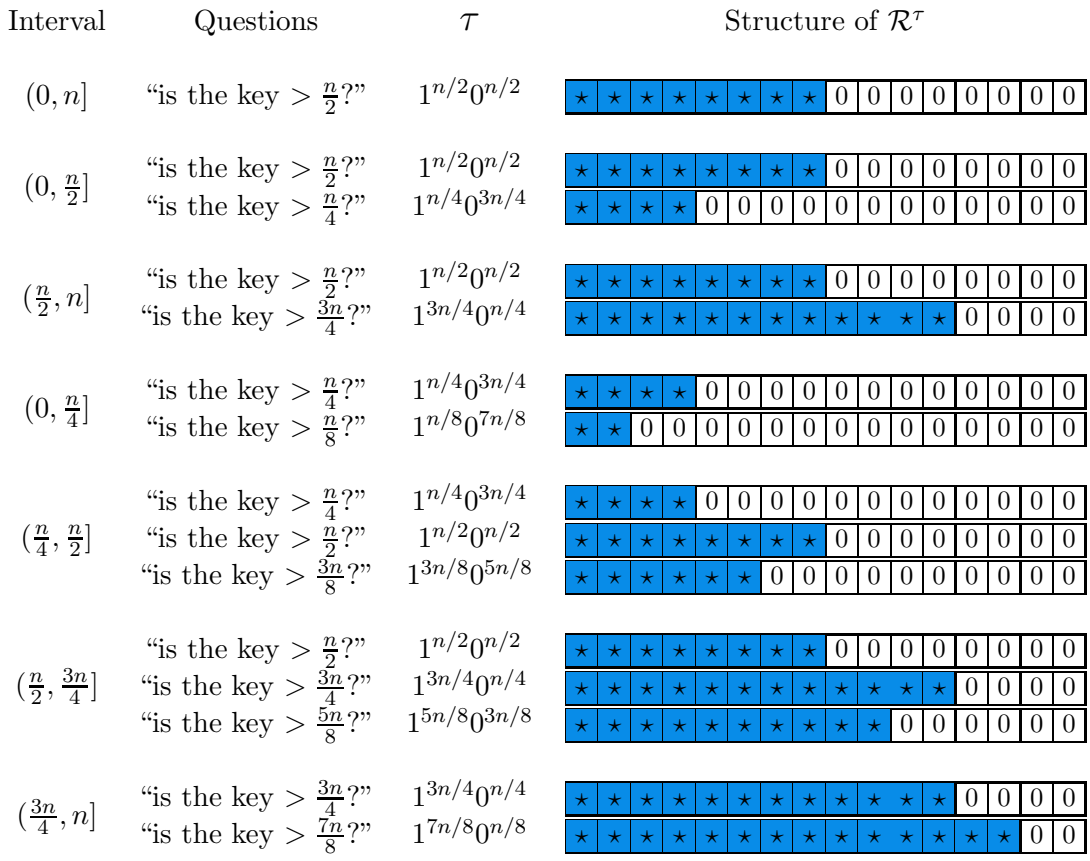

Figure 6: The structure of random strings used in the communication protocol.

$$
\begin{aligned}
\hat{\mathcal{R}} = {} & \mathcal{R}^{1^{n/2}0^{n/2}} \\
& \times \left(\mathcal{R}^{1^{n/2}0^{n/2}} \times \mathcal{R}^{1^{n/4}0^{3n/4}} \right) \times \left(\mathcal{R}^{1^{n/2}0^{n/2}} \times \mathcal{R}^{1^{3n/4}0^{n/4}} \right) \\
& \times \dots \\
& \times \left(\mathcal{R}^{1^{2}0^{n-2}} \times \mathcal{R}^{1^{1}0^{n-1}} \times \bigtimes_{a=1}^{(n-4)/2} \left(\mathcal{R}^{1^{2a}0^{n-2a}} \times \mathcal{R}^{1^{2a+2}0^{n-2a-2}} \times \mathcal{R}^{1^{2a+1}0^{n-2a-1}} \right) \times \mathcal{R}^{1^{n-2}0^{2}} \times \mathcal{R}^{1^{n-1}0^{1}} \right) \\
& \times \left(\bigtimes_{2c \log n} \left(\mathcal{R}^{1^{1}0^{n-1}} \times \bigtimes_{a=1}^{n-2} \left(\mathcal{R}^{1^{a}0^{n-a}} \times \mathcal{R}^{1^{a+1}0^{n-a-1}} \right) \times \mathcal{R}^{1^{n-1}0^{1}} \right) \right)
\end{aligned}
$$

See Figure 7 for an illustration. This $\hat{\mathcal{R}}$ describes all the strings as the source of randomness needed for the $O(\log n)$ communication protocol for GT, but each of the strings appears in the structure only once instead of $\alpha c \log n$ times. So, we need to duplicate this structure $\alpha c \log n$ times to properly simulate the protocol.

As in the proof of Claim 6, we are going to add two other steps to the structure. First, we are going to have t individual "prepackaged" copies for the GT protocol. Let $\mathcal{R}_1 = \ldots = \mathcal{R}_t =$

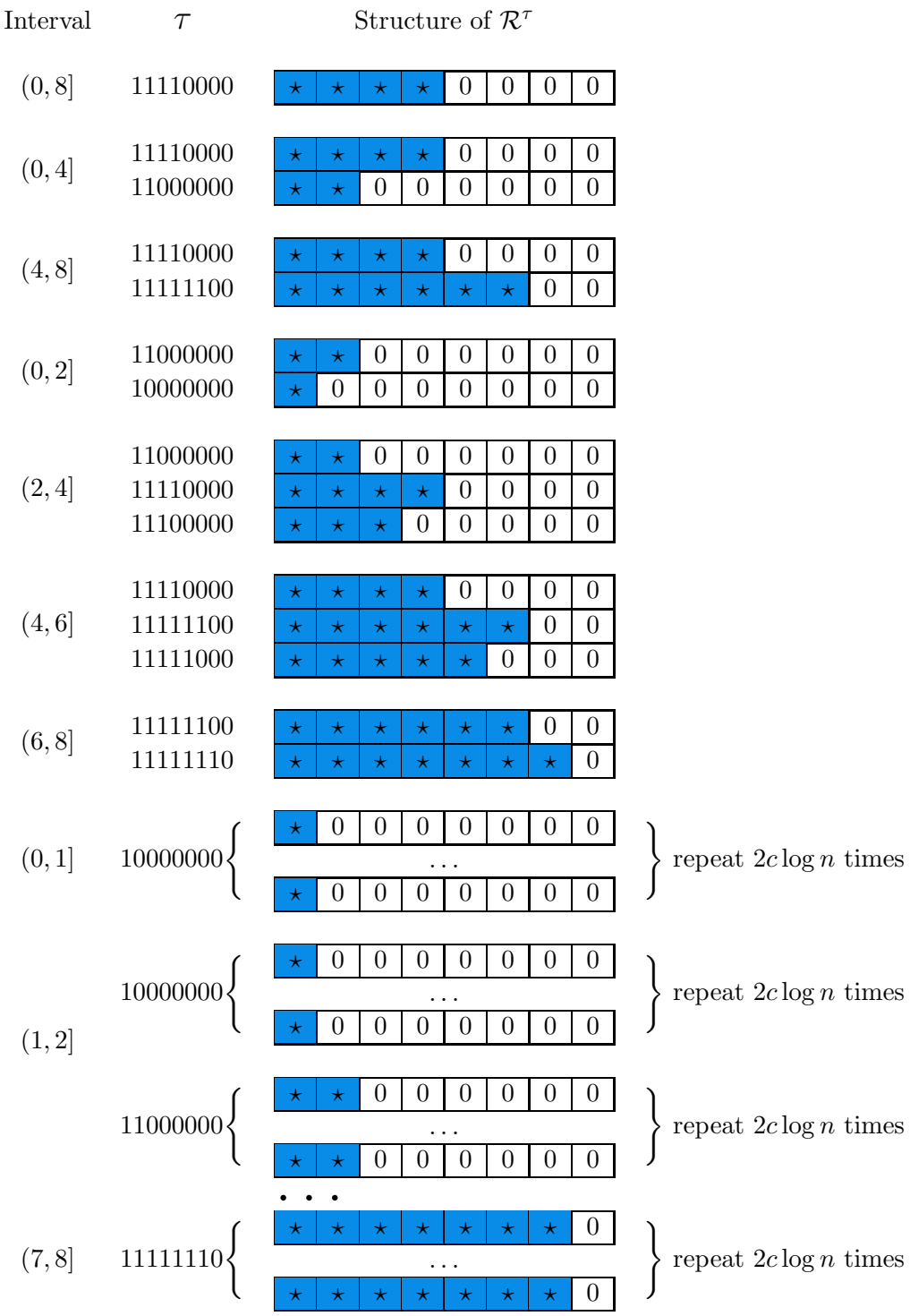

Figure 7: Structure of $\hat{\mathcal{R}}$. Blue cells with $\star$ represent indices where either a 0 or 1 could appear.

$\times_{\alpha c \log n} \hat{\mathcal{R}}$. Each of the copies has enough randomness and the right structure of that randomness to simulate one full run of the GT protocol. Let $\mathcal{R}' = \bigtimes_{j\in[t]} \mathcal{R}_j$, which allows us to handle t runs. Secondly, we add a set of "basis" strings to the structure: $\mathcal{B} = \{\mathbf{1}_1\} \times \{\mathbf{1}_2\} \times \ldots \{\mathbf{1}_{n-1}\} \times \{\mathbf{1}_n\} = \{10\ldots0\} \times \{010\ldots0\} \times \ldots \times \{00\ldots010\} \times \{00\ldots01\}$.

The final underlying structure of the oracle will be a Cartesian product of $\mathcal{R}'$ and $\mathcal{B}$: $\mathcal{R}_{\mathsf{OS}++} = \mathcal{B} \times \mathcal{R}' = \mathcal{B} \times (\bigtimes_{j\in[t]} \mathcal{R}_j)$. See Figure 8 for an illustration.

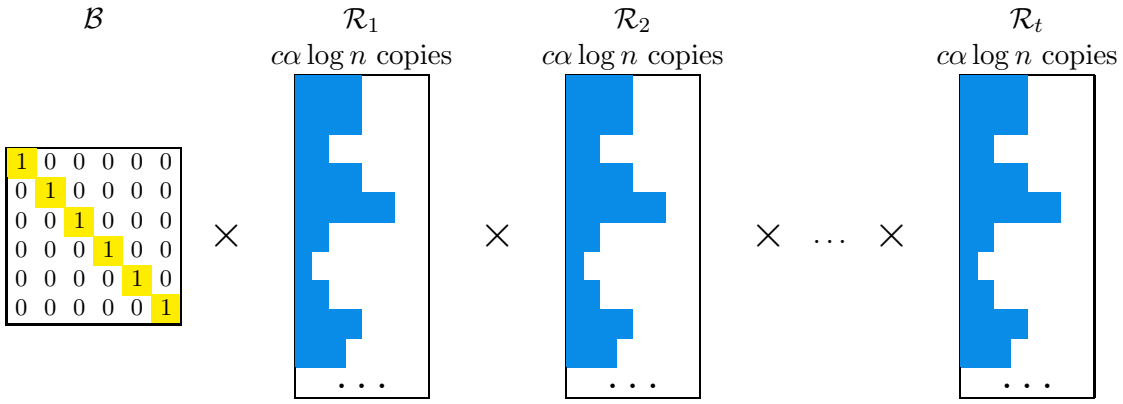

Figure 8: Structure of $\mathcal{R}_{\mathsf{OS}++}$. Each $\mathcal{R}_j$ consists of $c\alpha \log n$ copies of $\hat{\mathcal{R}}$.

We also set the parameters to be $\alpha = 2, t = 250n \ln 2$. Notice that this set $\mathcal{R}_{\mathsf{OS}++}$ is a *good base* by construction and has size $m \le n + 3\alpha t(cn \log n + 2cn \log^2 n) = O(n^3 \log^2 n)$.

Constructing the family of approximating polynomials. The construction will follow the same outline as the construction of polynomials for $\mathcal{R}_{\mathsf{OS}++}$. We construct a family of deterministic algorithms that work well for every fixed pair of inputs $i, x \in \{0,1\}^n$, and then construct a family of algorithms that work for all inputs with high probability at the same time, and then finally explain how to convert it to a family of approximating polynomials.

For all $i \in \{0,1\}^n, j \in [t], r \in \mathcal{R}_{\mathsf{OS}++}$ let $B_{(r,i,j)}(\mathcal{Y}(r,x))$ be the following deterministic algorithm.

1. Simulate the algorithm from Claim 22.
2. Each time the algorithm asks "is the key $> a$?":
3. Set $\tau = 1^a 0^{n-a}$.
4. For the α indices $v \in [m]$ corresponding to n-bit strings drawn from $\mathcal{R}^\tau$ within the jth copy $\mathcal{R}_j$ that were not already used prior to this step:
5. Compute $\langle i, r_v \rangle$ and compare it to $(\mathcal{Y}(r,x))_v = \langle x, r_v \rangle$.
6. If for all α indices v the inner products are the same then reply "yes, the key $> a$". Otherwise, reply "no, the key $\le a$".
7. Let k be the output of the algorithm.

8. Compare $i_k = \langle i, \mathbf{1}_k \rangle$ and $(\mathcal{Y}(r,x))_k = \langle x, \mathbf{1}_k \rangle = x_k$. If $x_k \leq i_k$ then accept. Otherwise, reject.

Notice that this algorithm emulates the $O(\log n)$ randomized communication protocol for GT communication problem.

This algorithm computes $\mathsf{GT}_i(x)$ with probability at least $\frac{11}{12}$ for all $j \in [t]$. That is, for all $j \in [t]$ and for all $i, x \in \{0,1\}^n$

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} [B_{(r,i,j)}(\mathcal{Y}(r,x)) = \mathsf{GT}_i(x)] \geq \frac{11}{12}.$$

This algorithm makes at most $3\alpha c \log n = 6c \log n$ queries to the oracle $\mathcal{Y}(r,x)$. Let $W(i,x,r,j)$ be the indicator that the j-th package of random strings in r defines a set of "bad" random strings for (i,x): $W(i,x,r,j) = 1$ if and only if $B_{(r,i,j)}(\mathcal{Y}(r,x)) \neq \mathsf{GT}_i(x)$. We established that $B_{(r,i,j)}(\mathcal{Y}(r,x))$ works well if given a random $r \leftarrow \mathcal{R}_{\mathsf{OS}++}$ for every $j \in [t]$ and the probability of this algorithm outputting an incorrect answer is at most $\frac{1}{12}$. So for all $i, x \in \{0,1\}^n, j \in [t]$

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} [W(i,x,r,j) = 1] = \mathbb{E}_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} [W(i,x,r,j)] \leq \frac{1}{12}.$$

For all $i \in \{0,1\}^n, r \in \mathcal{R}_{\mathsf{OS}++}$ let $A_{(r,i)}(\mathcal{Y}(r,x))$ be the following randomized algorithm:

- Choose $j \leftarrow [t]$ uniformly at random.
- Run $B_{(r,i,j)}(\mathcal{Y}(r,x))$.

The number of queries that $A_{(r,i)}$ makes to the oracle is the same as $B_{(r,i,j)}$ which is $3\alpha c \log n = 6c \log n$. We fix a pair (i,x) and evaluate the following probability.

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} \left[\Pr_{j \leftarrow [t]} [B_{(r,i,j)} \neq \mathsf{GT}_i(x)] > \frac{1}{6} \right] = \Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} \left[\frac{1}{t} \sum_{j \in [t]} W(i,x,r,j) > \frac{1}{6} \right].$$

We established that $\mathbb{E}_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} [W(i,x,r,j)] \leq \frac{1}{12}$ and so by Hoeffding's inequality,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} \left[\frac{1}{t} \sum_{j \in [t]} W(i,x,r,j) > \frac{1}{12} + \frac{1}{12} \right] \leq e^{-2\frac{t}{144}} \leq 2^{-\frac{500n}{144}}.$$

By a union bound over all possible $i, x \in \{0,1\}^n$,

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} \left[\exists i, x \in \{0,1\}^n : \frac{1}{t} \sum_{j \in [t]} W(i,x,r,j) > \frac{1}{6} \right] \leq 2^{2n} 2^{-\frac{500n}{144}} \leq 2^{-n} < \frac{1}{3}.$$

Therefore, we have proven that

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} \left[\exists i, x \in \{0,1\}^n : \Pr_{j \leftarrow [t]} [A_{(r,i)}(\mathcal{Y}(r,x))) \neq \mathsf{GT}_i(x)] > \frac{1}{6} \right] < \frac{1}{3}.$$

The last step is to convert this family of query algorithms into a family of approximating polynomials. Let $q_{(r,i)}$ denote the acceptance probability of $A_{(r,i)}$ which is a polynomial of degree at most $6c \log n$ such that

$$\Pr_{r \leftarrow \mathcal{R}_{\mathsf{OS}++}} \left[\exists i, x \in \{0,1\}^n : \left| q_{(r,i)}(\mathcal{Y}(r,x)) - \mathsf{GT}_i(x) \right| > \frac{1}{6} \right] < \frac{1}{3}$$

which is exactly what we were looking for.