Compressed Oracles and Adversary Methods

Ronak Ramachandran

April 2026

1 Introduction

This writeup is meant to be an informal introduction to two quantum query complexity lower bound techniques, with a focus on intuition over mathematical rigor.

This was written for a course project for *CS 395T: Quantum Protocols* taught by Dr. William Kretschmer at UT Austin. My original goal was to use a new explicit reduction from compressed oracles to multiplicative adversaries [JZ25] to re-express the compressed oracle lower bound of [TW26] as an adversary matrix. I instead mostly summarize [JZ25] in an attempt to distill out the intuition behind their main result.

2 Preliminaries

For any M, we use $[M]$ to denote $\{0, \dots, M-1\}$ (more precisely, we mean $\mathbb{Z}_M$, the integers mod M). Recall the quantum Fourier transform over the integers mod M. Let $\omega := e^{\frac{2\pi i}{M}}$. Then, for $y \in [M]$,

$$|\hat{y}\rangle := \mathrm{QFT}_M |y\rangle = \sum_{z \in [M]} \omega^{yz} |z\rangle . \tag{1}$$

3 Quantum Query Complexity

The goal of every algorithm is to produce some output given some input.

Example 3.1 (Or)**.** Given input $x \in \{0,1\}^N$, output "True" if x has any 1's and "False" otherwise.

Example 3.2 (UnstructuredSearch)**.** Given input $x \in \{0,1\}^N$, output i such that $x_i = 1$.

Example 3.3 (Collision)**.** Given input $f : [2N] \to [N]$, output distinct $x, x' \in [2N]$ such that $f(x) = f(x')$.

It is often helpful to think of inputs as functions f from "queries" to "responses." For instance, in UnstructuredSearch (and also Or), queries are of the form $i \in [N]$ and responses are of the form $x_i \in [M]$ where $M = 2$. In fact, in all cases we will consider, the set of all possible queries will be $[N]$, and the set of all possible responses will be $[M]$ for some N and M. At the same time, it can also be helpful to view our inputs as strings of length N over the alphabet $[M]$, so we might write $f \in [M]^{[N]}$.

We want to know how quickly algorithms can solve problems like these. We can prove lower bounds on the amount of time required to solve a problem by proving lower bounds on the number of queries that must be answered before a correct output can be produced, known as the **query complexity** of the problem.

Quantum algorithms can make queries in superposition, so queries to an input f are captured by a unitary $\mathcal{O}_f$ that we call a quantum oracle to f. For $x \in [N]$ and $y \in [M]$ stored in registers X and Y, $\mathcal{O}_f$ performs the map

$$|x\rangle_{\mathsf{X}} |y\rangle_{\mathsf{Y}} \xrightarrow{\mathcal{O}_f} |x\rangle_{\mathsf{X}} |y + f(x)\rangle_{\mathsf{Y}} . \tag{2}$$

Note that addition in the Y register is implicitly mod M. When $y = 0$, $\mathcal{O}_f$ has the effect of writing the response to our query x in the Y register. However, a more interesting fact is also true: when Y contains a Fourier basis state $|\hat{y}\rangle$, $\mathcal{O}_f$ has the effect of merely applying a phase that depends on x, y, and f.

$$|x\rangle_{\mathsf{X}} |\hat{y}\rangle_{\mathsf{Y}} \xrightarrow{\mathcal{O}_f} \omega^{-yf(x)} |x\rangle_{\mathsf{X}} |\hat{y}\rangle_{\mathsf{Y}} . \tag{3}$$

Proof.

$$\begin{aligned} |x\rangle_{\mathsf{X}} |\hat{y}\rangle_{\mathsf{Y}} &= |x\rangle_{\mathsf{X}} \sum_{z\in[M]} \omega^{yz} |z\rangle_{\mathsf{Y}} \\ &\xrightarrow{\mathcal{O}_f} |x\rangle_{\mathsf{X}} \sum_{z\in[M]} \omega^{yz} |z + f(x)\rangle_{\mathsf{Y}} \\ &= \omega^{-yf(x)} |x\rangle_{\mathsf{X}} \sum_{z\in[M]} \omega^{y(z+f(x))} |z + f(x)\rangle_{\mathsf{Y}} \\ &= \omega^{-yf(x)} |x\rangle_{\mathsf{X}} |\hat{y}\rangle_{\mathsf{Y}} . \end{aligned}$$

□

This is known as a phase query.

Without loss of generality, an arbitrary quantum query algorithm acts on a "query" register X, "response" register Y, and "workspace" register W, alternating between arbitrary unitaries U_i that don't depend on the input and applications of $\mathcal{O}_f$ on X and Y.

$$U_T \mathcal{O}_f U_{T-1} \mathcal{O}_f \dots U_2 \mathcal{O}_f U_1 \mathcal{O}_f U_0 |0\rangle_{\mathsf{W}} |0\rangle_{\mathsf{X}} |0\rangle_{\mathsf{Y}} \tag{4}$$

To succeed, the desired output must exist somewhere in the workspace register after the algorithm terminates.

4 Lower Bound Techniques

When proving query lower bounds, we first define some distribution $\mathcal{I}$ over inputs (often the uniform distribution over all inputs) for which solving our problem is hard. Then, we try to prove that no algorithm using fewer than T queries could produce a valid output on $\mathcal{I}$ with probability higher than $1 - \varepsilon$.

Instead of sampling an input f from $\mathcal{I}$ before running some quantum algorithm, it will be helpful to imagine a purification of this process. More precisely, we will introduce a new "input" register F that would collapse to some input f drawn from $\mathcal{I}$ if it were measured, but we will not allow the algorithm to measure it or directly interact with it. If $\mathcal{I}(f)$ is the probability of sampling f from $\mathcal{I}$, our new register F would initially contain

$$|\mathcal{I}\rangle_{\mathsf{F}} := \sum_{f\in[M]^{[N]}} \sqrt{\mathcal{I}(f)} |f\rangle_{\mathsf{F}} . \tag{5}$$

The only way the algorithm will be allowed to interact with this input register will be through queries. We will replace all queries to $\mathcal{O}_f$ with queries to a "purified" version $\mathcal{O}$ that uses the contents of register F to decide which $\mathcal{O}_f$ to apply to registers X and Y:

$$|x\rangle_{\mathsf{X}} |y\rangle_{\mathsf{Y}} |f\rangle_{\mathsf{F}} \xrightarrow{\mathcal{O}} |x\rangle_{\mathsf{X}} |y + f(x)\rangle_{\mathsf{Y}} |f\rangle_{\mathsf{F}} . \tag{6}$$

Importantly, the algorithm's view will not change: introducing register F will have no effect on the reduced state of the system on registers X, Y, and W at any point of the algorithm. In other words, the algorithm cannot distinguish between a world where the purifying register F exists and a world where f was sampled from $\mathcal{I}$ ahead of time, so we might as well imagine register F exists and we never collapse its contents.

The key insight behind both the compressed oracle method and the adversary method is that making queries affects register F. In fact, because the unitaries U_i only act on X, Y, and W, making queries is the

only way an algorithm affects register F. This lets us quantify how helpful queries can be for solving a problem by quantifying how much the contents of register F can change with each query.

Let ρ_t be the reduced state of register F after the t^{th} query. If we define "progress" as some property of ρ_t, and if we show "progress" starts small, increases very little with each query, but must eventually be high for our algorithm to succeed, then we prove many queries are necessary for our algorithm to succeed.

In the case of the adversary method, we measure progress by the expected value of our choice of observable Γ applied to ρ_t. This observable is called an "adversary matrix," and designing a good one is the central difficulty of applying the adversary method. In the case of the compressed oracle method, we view ρ_t as a superposition over some databases, and we measure progress by how much amplitude ρ_t has in the span of databases with enough information to produce our desired output. Within this view, the result of [JZ25], that the compressed oracle method is a special case of the adversary method, seems natural: the specific progress measure considered by the compressed oracle method can also be viewed as some observable Γ.

4.1 The Adversary Method

The initial state of the input register F is a pure state $|\mathcal{I}\rangle$ that is unentangled with the rest of our system. Because the workspace register must eventually contain our desired output (which depends on the value of the input f), by the end of the algorithm, registers W and F must be highly entangled. This motivates defining some notion of progress associated with "becoming less like $|\mathcal{I}\rangle$."

The adversary method defines this measure of progress W^t to be the expected value of a positive semi-definite observable Γ on register F with minimum eigenvalue 1 on eigenstate $|\mathcal{I}\rangle$.

$$W^t := \text{Tr}(\Gamma \rho_t) \tag{7}$$

We pick some "cutoff" eigenvalue $\lambda > 1$ of Γ and deem eigenspaces of eigenvalue less than λ "bad" and eigenspaces of higher eigenvalue "good." We want it to be the case that the algorithm must transform the state in F from bad to good for it to succeed with high probability. Note that if any "bad" state had high amplitude in a subspace of inputs that share the same desired output, then this goal would be doomed, because such a state would have low expectation for the observable Γ, but an algorithm that reached such a state would know what output to produce. For this reason, we define η to upper bound the amplitude of every "bad" vector within every subspace of inputs that share the same desired output.

We then get what's called the "Multiplicative Adversary Method" named so because in Eq. (9) below, we bound the multiplicative factor by which progress increases with each query (the additive adversary method is very similar, except it bounds $W^{t+1} - W^t$ instead of $\frac{W^{t+1}}{W^t}$).

For any T-query quantum algorithm that succeeds on $\mathcal{I}$ with probability $1 - \varepsilon$,

$$1)\ W^0 = 1 \tag{8}$$

$$2)\ \frac{W^{t+1}}{W^t} \leq \max_{\substack{x\in[N]\\ y\in[M]}} ||\mathcal{O}_{x,y}^\dagger \Gamma^{1/2} \mathcal{O}_{x,y} \Gamma^{-1/2}||^2 \tag{9}$$

$$3)\ W^T \geq 1 + (\lambda - 1)(\sqrt{1-\varepsilon} - \sqrt{\eta})^2 \tag{10}$$

where $\mathcal{O}_{x,y}$ denotes the action of $\mathcal{O}$ in the subspace where X is set to x and Y is set to y.

If the right hand side of Eq. (9) is a and the right hand side of Eq. (10) is b, then we see that T, the number of queries, must be big enough so that $1 \cdot a^T \geq b$. In other words, $T \geq \log_a(b) = \frac{\log(b)}{\log(a)}$. In other words, if we choose Γ and λ right, then we get a lower bound on quantum query complexity. For a problem $\mathscr{P}$,

$$\mathsf{MADV}(\mathscr{P}) = \max_{\Gamma,\lambda} \frac{\log\left(1 + (\lambda-1)(\sqrt{1-\varepsilon} - \sqrt{\eta})^2\right)}{\log\left(\max_{\substack{x\in[N]\\ y\in[M]}} ||\mathcal{O}_{x,y}^\dagger \Gamma^{1/2} \mathcal{O}_{x,y} \Gamma^{-1/2}||^2\right)} \tag{11}$$

See [JZ25, Theorem 3.3] for a more in-depth mathematical treatment of this bound.

Multiplicative Ladder Adversary Method Multiplicative ladder adversary matrices are a strict subclass of the adversary matrices allowed in the general multiplicative adversary method. This special case was introduced in [JZ25] to help reduce the compressed oracle method to the adversary method, but almost all known adversary lower bounds happen to be multiplicative ladder adversary matrices.

Unlike the general multiplicative adversary method, the multiplicative ladder adversary method has a per-step progress bound (like Eq. (9)) that is dependent on how many queries have been made. To explain this change, we need to define some subspaces. Let $|f_Q\rangle$ be the superposition over all f consistent with a list Q of query-response pairs, weighted according to some input distribution $\mathcal{I}$. Then let $\Pi_{\leq t}$ be the projection onto the span of $|f_Q\rangle$ where the number of query-response pairs in Q is $\leq t$. You can think of these spaces as those reachable from $|\mathcal{I}\rangle$ after t queries.

The additional constraints on multiplicative ladder adversary matrices are:

1. The eigenspaces of Γ have eigenvalues $1, \kappa, \kappa^2, \kappa^3, \ldots$ for some $\kappa > 1$.
   We call the κ^i eigenspace the i^{th} rung of the ladder, and let Λ_i be the projector to that eigenspace.
2. Γ and $\Pi_{\leq t}$ simultaneously diagonalize for all t's.
3. Applying $\mathcal{O}$ can only move an eigenvector up or down by at most one rung.
   That is, for any $|\psi_i\rangle$ on rung i and $|\psi_j\rangle$ on rung j such that $|i-j|>1$, $\langle\psi_i|\,\mathcal{O}\,|\psi_j\rangle = 0$.

The new version of Eq. (9) becomes:

$$2)\ \frac{W^{t+1}}{W^t} \leq \max_{\substack{i \\ x\in[N] \\ y\in[M]}} \left(1 + \frac{\kappa-1}{\sqrt{\kappa}} ||\Lambda_{i+1}\Pi_{\leq t+1}\mathcal{O}_{x,y}\Pi_{\leq t}\Lambda_i||\right)^2 \tag{12}$$

This special case is much more natural to think about: Γ is defined so states that are more unreachable with few queries have exponentially increasing eigenvalues.

With per-step progress depending on what step the algorithm is in, we don't get as nice a closed form for our quantum query complexity bound, but we can still say that for a problem $\mathscr{P}$, $\mathsf{MLADV}(\mathscr{P})$ is the smallest T such that the progress per step multiplies to more than the total progress needed to succeed:

$$\prod_{t=1}^{T} \max_{\substack{i \\ x\in[N] \\ y\in[M]}} \left(1 + \frac{\kappa-1}{\sqrt{\kappa}} ||\Lambda_{i+1}\Pi_{\leq t+1}\mathcal{O}_{x,y}\Pi_{\leq t}\Lambda_i||\right)^2 \geq 1 + (\lambda-1)(\sqrt{1-\varepsilon} - \sqrt{\eta})^2 \tag{13}$$

See [JZ25, Corollary 4.4] for a more in-depth mathematical treatment of this bound.

4.2 The Compressed Oracle Method

The key insight of the compressed oracle method, first introduced by [Zha19], is that in the Fourier basis, queries to the oracle result in some history of past queries getting recorded in the input register. This lets us view the input register as a database of some previous queries the algorithm has made (more accurately, a superposition over such databases). So long as the database does not contain enough information to determine the correct output for our problem, we can be confident our algorithm does not have enough information to succeed.

To explain what we mean, we will need the following neat fact:

Fact 4.1. *If, for all $a, b \in [M]$, O performs the map*

$$|a\rangle |b\rangle \xrightarrow{O} |a+b\rangle |b\rangle , \tag{14}$$

then in the Fourier basis, O performs the map

$$|\hat{a}\rangle |\hat{b}\rangle \xrightarrow{O} |\hat{a}\rangle |\widehat{b-a}\rangle . \tag{15}$$

In other words, copying information from the second register to the first looks like copying information from the first register to the second in the Fourier basis. A commonly used special case of this fact is that the CNOT gate switches direction when conjugated by Hadamard gates.

Proof.

$$\begin{aligned} |\hat{a}\rangle |\hat{b}\rangle &= \sum_{c,d\in[M]} \omega^{ac+bd} |c\rangle |d\rangle \\ &\xrightarrow{O} \sum_{c,d\in[M]} \omega^{ac+bd} |c+d\rangle |d\rangle \\ &= \sum_{c,d\in[M]} \omega^{a(c+d)+(b-a)d} |c+d\rangle |d\rangle \\ &= |\hat{a}\rangle |\widehat{b-a}\rangle . \end{aligned}$$

□

This gives us a new way to view the action of a query to $\mathcal{O}$. Imagine the entire evaluation table of f is written in register F. That is, imagine the register F is decomposed into the tensor product of registers $\mathsf{F}_0, \ldots, \mathsf{F}_{N-1}$ each containing the evaluation of f on that input:

$$|f\rangle_{\mathsf{F}} = |f(0)\rangle_{\mathsf{F}_0} |f(1)\rangle_{\mathsf{F}_1} |f(2)\rangle_{\mathsf{F}_2} \ldots |f(x)\rangle_{\mathsf{F}_x} \ldots |f(N-1)\rangle_{\mathsf{F}_{N-1}} , \tag{16}$$

then, within the subspace where X contains some specific query $|x\rangle$, $\mathcal{O}$ acts as the identity on all registers except Y and F_x and performs the map

$$|y\rangle_{\mathsf{Y}} |f(x)\rangle_{\mathsf{F}_x} \to |y+f(x)\rangle_{\mathsf{Y}} |f(x)\rangle_{\mathsf{F}_x} . \tag{17}$$

by Fact 4.1, this means $\mathcal{O}$ maps

$$|\hat{y}\rangle_{\mathsf{Y}} |\widehat{f(x)}\rangle_{\mathsf{F}_x} \to |\hat{y}\rangle_{\mathsf{Y}} |\widehat{f(x)-y}\rangle_{\mathsf{F}_x} . \tag{18}$$

In other words, in the Fourier basis, instead of queries copying information from the input register to the algorithm's registers, queries record the algorithm's query in the input register! To take advantage of this, we will often express the response register Y and the input registers $\{\mathsf{F}_x\}_{x\in[N]}$ in the Fourier basis.

The Uniform Distribution. When $\mathcal{I}$ is the uniform distribution over all possible inputs, the initial state of the purifying register F is

$$|\text{Uniform}\rangle = \sum_{f\in[M]^{[N]}} \frac{1}{\sqrt{M^N}} |f\rangle_{\mathsf{F}} = |\hat{0}\rangle_{\mathsf{F}} . \tag{19}$$

We can think of this state as an empty database ready to record any queries the algorithm might make. Just before the algorithm's first query (after applying U_0) the state of F is still $|\hat{0}\rangle_{\mathsf{F}}$. If we interpret the state of registers Y and F as a superposition over Fourier basis states, then in each branch of the superposition, applying $\mathcal{O}$ adds a single element to the database. Specifically, in the $|x\rangle_{\mathsf{X}} |\hat{y}\rangle_{\mathsf{Y}} |\hat{0}\rangle_{\mathsf{F}}$ branch, the state in F_x goes from $|\hat{0}\rangle_{\mathsf{F}_x}$ to $|\widehat{-y}\rangle_{\mathsf{F}_x}$. Similarly, after the t^{th} query, in every branch of superposition, the database will contain at most t entries.

The compressed oracle method gets its name from the fact that the databases can be stored and manipulated succinctly by only storing non-zero entries. This has cryptographic applications (it lets quantum computers efficiently implement quantum random oracles by lazy sampling), but I believe it has no relevance to proving query lower bounds, so in this writeup I will not "compress" the contents of the F register.

When is the compressed oracle method useful? The compressed oracle is most applicable to prove lower bounds against quantum algorithms trying to solve search problems where the goal is to output k input-output pairs $\{(x_1, f(x_1)), \ldots, (x_T, f(x_T))\}$ that satisfy some property $\mathcal{P}$, and the algorithm wants to succeed on the **uniform** distribution over all inputs. Like the multiplicative adversary method, it is capable of proving lower bounds even against algorithms that are allowed to succeed with low probability.

Let $\Pi_{\leq t}^{\text{good}}$ (acting on F) be the projector to the subspace spanned by all t-element databases that contain k elements satisfying $\mathcal{P}$, and let $\Pi_{\leq t}^{\text{bad}}$ project to t-element databases that lack k elements satisfying $\mathcal{P}$. By the time the algorithm finishes, register F must have a large projection onto the "good" subspace. However, if we can show that any individual query cannot shift much amplitude from the "bad" to the "good" subspace, then we get a lower bound on query complexity. If we need our algorithm to succeed with probability $1-\varepsilon$, the lower bound given by the compressed oracle method, $\mathsf{Comp}(\mathscr{P})$, is the smallest T such that

$$\sqrt{\frac{k}{M}} + \sum_{t=1}^{T} \max_{\substack{x\in[N]\\ y\in[M]}} ||\Pi_{\leq t+1}^{\text{good}} \mathcal{O}_{x,y} \Pi_{\leq t}^{\text{bad}}|| \geq \sqrt{1-\varepsilon} \tag{20}$$

Intuitively, what this is saying is that before any queries, there is at most a $\frac{k}{M}$ chance of success, and each step moves no more than $\max_{\substack{x\in[N]\\ y\in[M]}} ||\Pi_{\leq t+1}^{\text{good}} \mathcal{O}_{x,y} \Pi_{\leq t}^{\text{bad}}||$ amplitude to the "good" subspace.

Collision Lower Bound. For example, for Collision, we're looking for two query-response pairs with matching responses, so $k = 2$. Classically, if you've made t queries, and you haven't seen a collision yet, then there's a $\frac{t}{M}$ chance of seeing a collision when you pick the $t+1^{\text{th}}$ random query. By essentially the same argument, we find that

$$\max_{\substack{x\in[N]\\ y\in[M]}} ||\Pi_{\leq t+1}^{\text{good}} \mathcal{O}_{x,y} \Pi_{\leq t}^{\text{bad}}|| \leq \sqrt{\frac{t}{M}}. \tag{21}$$

Rearranging gives us the lower bound of $\Omega(M^{1/3})$, reproducing the tight lower bound of [Aar02].

4.3 Reducing Compressed Oracles to Multiplicative Ladder Adversaries

I won't (and can't) reproduce the reduction here, but their main result is that

$$\mathsf{Comp}(\mathscr{P}) \leq 6 \cdot \mathsf{MLADV}(\mathscr{P}). \tag{22}$$

I haven't fully understood the reduction yet, but it seems like a matter of carefully relating the projections $\Pi_{\leq t}^{\text{good}}/\Pi_{\leq t}^{\text{bad}}$ and the projections $\Pi_{\leq t}$. Curiously, the reduction only uses a ladder with two rungs (rung 0 and rung 1), suggesting the compressed oracle method is not even close to using the full power of the multiplicative ladder adversary method. Perhaps there are new lower bound techniques that are a hybrid between the two.

References

[Aar02] Scott Aaronson. Quantum lower bound for the collision problem. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 635–642. ACM, 2002. doi:10.1145/509907.509999.

[JZ25] Stacey Jeffery and Sebastian Zur. The compressed oracle is a worthy (multiplicative) adversary, 2025. URL: https://arxiv.org/abs/2509.07876, arXiv:2509.07876.

[TW26] Ewin Tang and John Wright. Amplitude amplification and estimation require inverses, 2026. URL: https://arxiv.org/abs/2507.23787, arXiv:2507.23787.

[Zha19] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, Lecture Notes in Computer Science, pages 239–268. Springer, 2019. doi:10.1007/978-3-030-26951-7_9.