COL7160 : Quantum Computing

Lecture 22: Quantum Lower Bounds

Instructor: Rajendra Kumar **Scribe:** Siddhant Agrawal

1 Introduction and Motivation

So far in this course we have seen quantum algorithms that outperform their best known classical counterparts: Grover's $O(\sqrt{N})$ search [Gro96], Shor's polynomial-time factoring [Sho94], and others. A natural question is: are these algorithms *optimal*? Could a cleverer quantum algorithm solve unstructured search in $O(N^{1/3})$ queries?
This lecture develops tools to answer such questions rigorously, by proving *quantum query complexity lower bounds*: statements of the form "any quantum algorithm for problem φ must make at least T queries to its input." We study two complementary techniques.

1. **The Polynomial Method** [BBC+01]: every T-query quantum algorithm computes, as its acceptance probability, a real polynomial of degree at most $2T$. This gives a lower bound on $Q_E(\varphi)$, the *exact* query complexity—the minimum queries needed to compute φ with zero error.
2. **The Adversary Method** [Amb02]: any algorithm must make the states for YES and NO instances distinguishable. By bounding how fast inner products can decrease per query, one obtains a lower bound on $Q(\varphi)$, the *bounded-error* query complexity (success probability $\geq 2/3$).

For the N-bit OR function, the polynomial method gives $\Omega(N)$ for exact computation, while the adversary method gives the tight $\Omega(\sqrt{N})$ for bounded-error computation, matching Grover's algorithm.

2 The T-Query Quantum Algorithm Model

2.1 Setup

We study Boolean functions $\varphi : \{0,1\}^N \to \{0,1\}$. The inputs $x_0, \ldots, x_{N-1}$ are N bits, and the algorithm's goal is to determine $\varphi(x_0, \ldots, x_{N-1})$.
A T-query quantum algorithm operates on a Hilbert space with three logical parts:

- an *index register* of $\lceil \log_2 N \rceil$ qubits, used to address which input bit to query;
- a *target qubit*, into which the oracle writes the queried bit;
- a *workspace register* of arbitrary size. The query model counts only oracle calls; no restriction is placed on the workspace.

The input $(x_0, \ldots, x_{N-1})$ is not wired into the circuit. The algorithm accesses input bits *only* through the oracle gate U_F described below.

2.2 The Oracle Gate

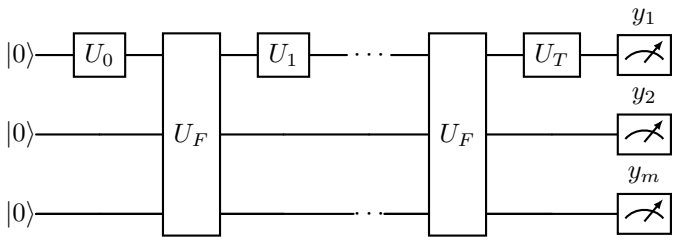
The standard (bit-flip) oracle U_F acts on computational basis states as

$$U_F \left|i\right\rangle \left|b\right\rangle \left|w\right\rangle = \left|i\right\rangle \left|b \oplus x_i\right\rangle \left|w\right\rangle ,$$

where $i \in \{0, \ldots, N-1\}$ is the queried index, $b \in \{0,1\}$ is the target bit, and $|w\rangle$ is the workspace (unchanged). By linearity, U_F acts on a general superposition by updating each basis component independently.

2.3 Circuit Structure

The algorithm interleaves T oracle calls with $(T+1)$ fixed (input-independent) unitaries $U_0, U_1, \ldots, U_T$:



After U_0 acts on $|0\cdots0\rangle$, the state (before any oracle access) has the general form

$$\sum_{i=0}^{N-1} \alpha_i \,|i\rangle\,|b_i\rangle\,|\psi_i\rangle\,,$$

where the amplitudes α_i and workspace states $|\psi_i\rangle$ depend only on U_0, not on the input x. Writing out the leading components,

$$\alpha_{00}\,|0\rangle\,|0\rangle\,|\psi_0\rangle + \alpha_{01}\,|0\rangle\,|1\rangle\,|\psi_0\rangle + \cdots$$

After one application of U_F, the component $\alpha_{00}\,|0\rangle\,|0\rangle\,|\psi_0\rangle$ becomes

$$\alpha_{00}\,|0\rangle\,|0\oplus x_0\rangle\,|\psi_0\rangle = \alpha_{00}\Big[(1-x_0)\,|0\rangle\,|0\rangle\,|\psi_0\rangle + x_0\,|0\rangle\,|1\rangle\,|\psi_0\rangle\Big].$$

We use $x_0 \in \{0,1\}$ to split the two cases: when $x_0 = 0$ the target is unchanged; when $x_0 = 1$ it flips to $|1\rangle$. This is a convenient shorthand for tracking which branch survives—the total norm is preserved.

2.4 Measuring Success

After T queries and the final unitary U_T, the algorithm measures to produce outputs $y_1, \ldots, y_m$. The algorithm *computes* φ with bounded error if

$$\Pr[\text{output} = \varphi(x)] \geq \tfrac{2}{3} \quad \text{for all } x \in \{0,1\}^N.$$

The *quantum query complexity* $Q(\varphi)$ is the minimum T over all such algorithms. For *exact* computation (zero error), we write $Q_E(\varphi)$.

3 The Polynomial Method

3.1 The Main Theorem

The key insight is that a T-query algorithm can only produce a low-degree polynomial as its acceptance probability. The theorem applies to both exact and bounded-error computation; in this section we focus on the exact setting. The adversary method (Section 4) is better suited to the bounded-error setting.

Theorem 1 (Beals et al. [BBC+01])**.** *Let A be a T-query quantum algorithm, and let*

$$p(x_0, \ldots, x_{N-1}) = \Pr[A \text{ accepts} \mid \text{input } x]$$

be its acceptance probability. Then p is a real multilinear polynomial in $x_0, \ldots, x_{N-1}$ of degree at most $2T$. As a consequence:

(a) (Exact computation.) *If A computes φ exactly, then $p(x) = \varphi(x)$ for all x, so $\deg(\varphi) \leq 2T$, giving*

$$Q_E(\varphi) \;\geq\; \tfrac{1}{2}\deg(\varphi).$$

(b) (Bounded-error computation.) *If A computes φ with bounded error, then p satisfies $|p(x) - \varphi(x)| \leq 1/3$ for all $x \in \{0,1\}^N$, so $\widetilde{\deg}(\varphi) \leq 2T$, giving*

$$Q(\varphi) \;\geq\; \tfrac{1}{2}\widetilde{\deg}(\varphi),$$

where $\widetilde{\deg}(\varphi)$ is the minimum degree of any such approximating polynomial.

Remark 2. Over Boolean inputs $x_i \in \{0,1\}$, we have $x_i^k = x_i$ for all $k \geq 1$, so any polynomial can be reduced to a multilinear one without changing its values on $\{0,1\}^N$.

Remark 3. Why degree $\leq 2T$? Each query to x_i introduces one factor of x_i into the amplitude, and probabilities are squared amplitudes—hence the factor of 2. A formal proof proceeds by induction on T, showing amplitudes remain degree-T polynomials throughout.

3.2 Exact Polynomial Degree of OR

We apply Theorem 1(a) to the N-bit OR function,

$$\mathrm{OR}_N(x_0, \ldots, x_{N-1}) = x_0 \vee \cdots \vee x_{N-1} = \begin{cases} 1 & \text{if any } x_i = 1, \\ 0 & \text{if all } x_i = 0. \end{cases}$$

Class discussion question. *Find the degree of any multilinear polynomial exactly computing OR_N. What does this say about $Q_E(\mathrm{OR}_N)$?*

Claim 4. The multilinear polynomial $\mathrm{OR}_N(x_0, \ldots, x_{N-1}) = 1 - (1-x_0)(1-x_1)\cdots(1-x_{N-1})$ exactly computes OR_N on $\{0,1\}^N$ and has degree N.

Proof. Each factor $(1 - x_i)$ equals 1 if $x_i = 0$ and 0 if $x_i = 1$. The product $(1 - x_0)\cdots(1 - x_{N-1})$ equals 1 only when all $x_i = 0$, and 0 otherwise. Subtracting from 1 gives exactly OR_N.
The degree is N: expanding the product yields a leading term $(-1)^N x_0 x_1 \cdots x_{N-1}$, so degree is at least N; multilinearity on N variables makes N the maximum possible degree.
The lower bound—no polynomial of degree less than N can exactly compute OR_N—follows from a symmetrization argument: any such polynomial, when symmetrized, must be a univariate polynomial of degree N agreeing with OR on $\{0, 1, \ldots, N\}$, forcing degree N. □

Corollary 5. $Q_E(\mathrm{OR}_N) \geq N/2$.

Remark 6. This is a lower bound on *exact* computation: any zero-error quantum algorithm for OR_N requires $\Omega(N)$ queries. This is tight—even classically, exact OR requires checking every bit in the worst case.
It says nothing directly about bounded-error computation. There, Grover's $O(\sqrt{N})$-query algorithm is optimal and the correct lower bound is $Q(\mathrm{OR}_N) = \Omega(\sqrt{N})$. The polynomial method can in principle give this via $\widetilde{\deg}(\mathrm{OR}_N) = \Omega(\sqrt{N})$, but proving that approximate degree bound requires a non-trivial Chebyshev argument outside the scope of this lecture. The adversary method (Section 4) gives the $\Omega(\sqrt{N})$ bounded-error lower bound directly.

4 The Adversary Method

4.1 Core Idea

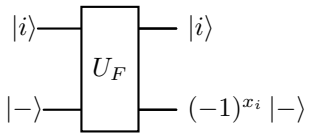
Any quantum algorithm must ultimately *distinguish* YES inputs from NO inputs. Before any queries the algorithm's state is the same regardless of input—it has no information yet. Each oracle query moves the states for different inputs apart. The adversary method formalizes how slowly this separation can occur.

4.2 The Phase Oracle

It is convenient to use the *phase oracle*:

$$U_x |i\rangle = (-1)^{x_i} |i\rangle .$$

This is equivalent to U_F up to a basis change on the target qubit: prepare the target in $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ before each query.



With the phase oracle, U_x is diagonal in the index basis with entries $(-1)^{x_i}$, making the dependence on x explicit.

4.3 Setup: YES and NO Instances

Fix $\varphi : \{0,1\}^N \to \{0,1\}$ and let $\mathcal{Y} = \varphi^{-1}(1), \mathcal{N} = \varphi^{-1}(0)$. Let $|\Psi_y^{(t)}\rangle$ and $|\Psi_z^{(t)}\rangle$ denote the algorithm's state after t queries on YES instance y and NO instance z respectively.

Initially ($t = 0$): the state is $|0 \cdots 0\rangle$ for any input, so $\langle \Psi_y^{(0)} | \Psi_z^{(0)} \rangle = 1$.

Finally ($t = T$): the states must be sufficiently different to allow correct output. Concretely:

- *Exact computation* (Q_E): perfect distinguishability requires $|\langle \Psi_y^{(T)} | \Psi_z^{(T)} \rangle| = 0$.
- *Bounded-error computation* (Q): it suffices to reach $|\langle \Psi_y^{(T)} | \Psi_z^{(T)} \rangle| \leq \varepsilon$ for some constant $\varepsilon < 1$.

In either case the inner product must travel from 1 toward 0. Each query can decrease it by only a bounded amount—this is the engine of the method.

Example 7. Let $\varphi(x_0) = x_0$, $\mathcal{Y} = \{1\}, \mathcal{N} = \{0\}$. Starting in $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$:

$$\begin{aligned} x_0 = 0: &\quad U_0 |+\rangle = |+\rangle, \\ x_0 = 1: &\quad U_1 |+\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle. \end{aligned}$$

After one query, $\langle + | - \rangle = 0$: perfectly distinguishable. One query suffices, consistent with $Q(\varphi) = 1$.

4.4 The Adversary Bound

Theorem 8 (Ambainis [Amb02])**.** *Let $\varphi : \{0,1\}^N \to \{0,1\}$ with $\mathcal{Y} = \varphi^{-1}(1)$ and $\mathcal{N} = \varphi^{-1}(0)$. Suppose there exist non-negative integers m and m' such that:*

(i) *For each $y \in \mathcal{Y}$, there exist at least m strings $z \in \mathcal{N}$ with $\mathrm{dist}(y,z) = 1$.*

(ii) *For each $z \in \mathcal{N}$, there exist at least m' strings $y \in \mathcal{Y}$ with $\mathrm{dist}(y,z) = 1$.*

Here $\mathrm{dist}(y,z) = |\{i : y_i \neq z_i\}|$ is the Hamming distance. Then

$$Q(\varphi) \;\geq\; c\sqrt{m \cdot m'}$$

for an absolute constant $c > 0$.

The full proof is deferred to next Lecture. The intuition: define $R \subseteq \mathcal{Y} \times \mathcal{N}$ as all pairs (y,z) at Hamming distance 1; conditions (i) and (ii) ensure R is large. Track the progress measure

$$P^{(t)} = \sum_{(y,z) \in R} \left| \langle \Psi_y^{(t)} | \Psi_z^{(t)} \rangle \right|,$$

which starts at $|R|$ and must reach near 0 for the algorithm to succeed. The key bound is that each query decreases P by at most $O(|R|/\sqrt{mm'})$, so driving P down by $\Omega(|R|)$ requires $T = \Omega(\sqrt{mm'})$ queries.

Remark 9. The trick is choosing $\mathcal{Y}$ and $\mathcal{N}$ so that YES and NO instances are densely connected by Hamming-1 neighbors—making m and m' as large as possible to get the strongest bound.

4.5 Application: $\Omega(\sqrt{N})$ Lower Bound for Grover's Algorithm

Homework (stated in class). *Use the adversary method to prove a lower bound for Grover's algorithm (the OR problem).*

Solution. Set

$$\mathcal{Y} = \left\{y \in \{0,1\}^N : \mathrm{OR}_N(y) = 1,\ \|y\|_1 = 1\right\}, \qquad \mathcal{N} = \{0^N\}.$$

$\mathcal{Y}$ consists of the N strings with exactly one 1-bit; $\mathcal{N}$ is the single all-zeros string.

Condition (i). Each $y \in \mathcal{Y}$ has $\|y\|_1 = 1$, so flipping its unique 1-bit gives $0^N \in \mathcal{N}$ at distance 1. Thus $m = 1$.

Condition (ii). The single $z = 0^N$ differs from every $y \in \mathcal{Y}$ in exactly one position. Since $|\mathcal{Y}| = N$, we get $m' = N$.

Applying Theorem 8:

$$Q(\mathrm{OR}_N) \ \geq\ c\sqrt{1 \cdot N} = c\sqrt{N}.$$

Hence $Q(\mathrm{OR}_N) = \Omega(\sqrt{N})$. Since Grover's algorithm achieves $O(\sqrt{N})$ queries [Gro96], it is *optimal*.

Remark 10. Weight-1 YES instances are the right choice because they are the only YES instances at Hamming distance exactly 1 from 0^N. Taking all YES instances would introduce pairs at distance > 1, violating the theorem's requirement.

Both bounds for OR_N are tight: $\Omega(N)$ for exact computation and $\Omega(\sqrt{N})$ for bounded-error computation [dW23].

References

[Amb02] Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002.

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

[dW23] Ronald de Wolf. Quantum computing: Lecture notes, 2023.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[Sho94] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.