Quantum Computation **(CMU 18-859BB, Fall 2015)**

Lecture 11: Quantum Query Lower Bounds Using Polynomials

October 14, 2015

Lecturer: Ryan O'Donnell *Scribe: Connell Donaghy*

1 Some Quantum Algorithms

Let's review some some algorithms we've learned so far, and revisit their query complexity.

Recall Grover's algorithm for *unstructured search*: Given access to a database of size N and some function $f : [N] \to \{0, 1\}$ along with a promise that $f(x) = 1$ for a unique x^*, find x^*. Using a quantum computer, we find $O(\sqrt{N})$ queries sufficient to find x^*. Now, we compare this result with query bounds on a classical computer. Classically, it is clear to see that we need $O(N)$, or more specifically $N - 1$ queries to find this x^*. It's provable that even randomly with some error probability ϵ, we still need $\Theta(n)$ queries to find x^*. So, now that we've seen some cool quantum algorithms, is there a way to show that $O(\sqrt{N})$ queries is the best we can do? It'd be nice if we could show that we need $\Omega(\sqrt{N})$ for the number of queries to find this x^* using a quantum computer.

Remark 1.1. Grover's search and SAT

We've discussed that we can use Grover search and a reduction to solve Circuit-SAT in $O(\sqrt{2^n})$ or $O(1.41^n)$ time (or gates.) Proving a lower bound for Grover's search doesn't necessarily prove a lower bound for Circuit-SAT, as there could be some clever algorithm which takes advantages of gates, and doesn't simply attempt all input possibilities to the circuit.

Next, recall Simon's problem. For this problem, we are given query access to some function $f : [N] \to [M]$, where N can be represented as a binary string $\{0, 1\}^n$, and M can be considered as M different "colors." We are given the promies that $f(x + s) = f(x)$ for some $s \in \{0, 1\}^n$, and ignore the trivial case of $s = 0$. In this problem, we have seen that we can find s using $O(\log N)$ queries quantumly, and $\Theta(\sqrt{N})$ queries randomly via a birthday attack. [Bac06]

Finally, we consider the element distinctness (ED) problem. In this problem, we are given oracle access to some function $f : \{0, 1\}^n \to [M]$, where once again we think of the output as colors. In this problem, we want to output "yes" if all the outputs of f are distinct, and say no if there is some duplicate color, such that $f(x) = f(y) = c$ for some $x \neq y$. Classically, we can sort the input and scan for duplicates in $O(N \log N)$ times. Randomly, it is possible to solve element distinctness in $\Theta(N)$ time. Quantumly, we can solve element distinctness in $\Theta(N^{\frac{2}{3}})$ queries. [BDH$^+$01]

Well, what's the difference between these different quantum speedups? Why does Simon's problem get an exponential speedup, whereas element distinctness and search only seem to

gain polynomial speedups? To answer these questions, let's first develop some notation, then revisit this remark.

2 Some New Terminology

Let's discuss some terminology which will be useful as we proceed through the polynomial method.

First, we classify problems as either **Search Problems** or **Decision Problems**.

- **Search Problems**: A problem such that the output is a string, (i.e. x^* for Grover's algorithm as discussed in section one.)
- **Decision Problems**: A problem such that the output is binary (yes/no), such as the output to the element distinctness problem.

For many problems, it is easy to construct a decision problem from a search problem. Consider Grover's algorithm. The **search version** for Grover's algorithm is: promised some $f(x) = 1$ for at moste one x find and return x. Now, consider this as a **decision problem**. In this version, we could simply return "yes" if x exists with $f(x) = 1$, or return "no" if no such x exists. We can solve this decision version by searching for some x, and then testing its value in $O(\sqrt{N})$ queries.

Next, let's recall some basic complexity theory. We say a randomized algorithm "solves" a decision problem if it outputs the correct answer more than $\frac{2}{3}$ of the time (although it will work for any probability greater than $\frac{1}{2}$ by repeating the algorithm a constant number of times.)

Lastly, let's define a **promise problem**. A promise problem is an algorithm over some subclass of all functions. This means, that we are given some promise that restricts f and disallows some possible functions. Often, given a promise problem, the algorithm will take advantage of the promise to speed up its runtime. If a problem is not a promise problem, then it is a **total problem**. That is to say, the algorithm works for all inputs on any f. Thus, for a decision problem, we need $f : [N] \to \{0, 1\}$, and all possible functions f must be valid inputs to the algorithm. Now, earlier we discussed a promise version of Grover, where we were promised that there was some x^* such that $f(x^*) = 1$. Let's compare this to total decision Grover search, where we must now output yes if such an x^* exists, and output no if $f(x) = 0$ on all inputs. In Homework 3, we solved that even this total decision version of Grover search is solvable in $O(\sqrt{N})$ queries.

Returning to search versus decision problems, let's revisit Simon's problem. We can see that for Simon's problem, we have a search problem, as we try to find s such that $f(x + s) = f(x)$ for all x. Also, we are given a promise that such an s exists, and that it is not 0. So what would the decision problem be? We simply output "yes" if $s = 0$, and "no" otherwise. If $s = 0$, then we know that we have all distinct colors, so the decision version of Simon's problem is actually the Element Distinctness Problem! However, we no longer have a promise in this decision version, as we must produce a valid answer over any input function f.

At the end of section one, we questioned why Simon's search had such a great speedup from a quantum computer, compared to a relatively unexciting polynomial speedup for element distinctness. With our new terminology, we can give an answer, thanks to the following theorem

Theorem 2.1. *For a total decision problem, a quantum algorithm can NOT beat a classical algorithm by more than a polynomial runtime. Let $D(n)$ be the classical query complexity, and $Q(n)$ be the quantum query complexity. We have*

$$D(n) \leq Q(n)^6$$

Thus, our speedups due to a quantum computer are actually fairly limited on total decision problems! [BBC+01]

Using this theorem, we see that because Grover's algorithm and Element Distinctness are both **total decision** problems, then it is impossible to get the type of exponential speedup we get in Simon's problem. Simon's problem is a **promise problem**, so we are able to achieve some nice speedups.

Remark 2.2. Similar to the maximum polynomial speedup for total decision problems, it has also been proven that $D(n) \leq R(n)^3$ and that $R(n) \leq Q(n)^{2.5}$, where $R(n)$ is the runtime of a randomized algorithm. [BBC+01]

3 How to Prove Lower Bounds

This lecture along with next lecture, we're going to go over the **polynomial method** for proving lower bounds. Another method which will be addressed later in the course is the **adversary method**, which is actually more powerful but a little more difficult to understand.

To understand and use the polynomial method, we're going to switch up some notation. Before, we had data as a function

$$f : [N] \to [M]$$

From now on, we'll consider data as a "string" such as

$$w \in [M]^N$$

or

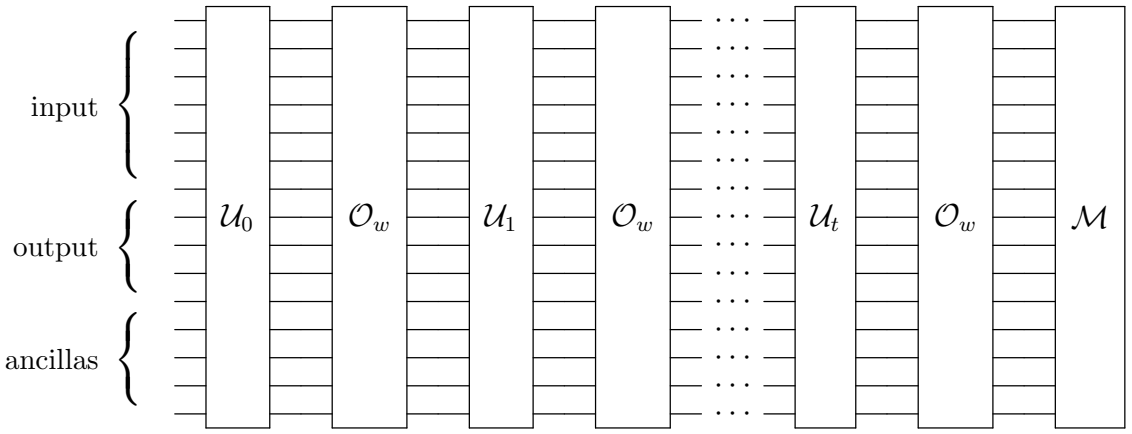
w_1	w_2	$\cdots$	w_{N-1}	w_N

Essentially, our data is now some mystery array full of mystery strings that we can index into via w_i, where $i \in 0, 1, \cdots, N$. Our oracle, previously O_f, will now be denoted as O_w. The classical version of this oracle takes in an i, and outputs w_i. Quantumly, this means if we apply the new oracle to some $|i\rangle$ and preserve reversible computing, we get $|i\rangle \to (-1)^{w_i} |i\rangle$, or $|i\rangle |y\rangle \to |i\rangle |y \oplus w_i\rangle$. Now, our function F, will actually be considered a property of all

of the strings w, instead of a mapping from $[N]$ to $[M]$ as we had previously done. Let's consider Grover's total decision algorithm in this new notation. Each w_i is either 0 or 1. We want to know if there is a $w_i = 1$ for some i. Essentially, we are taking the logical or of all of the $w \in F$, or $OR(w_1, w_2, \cdots, w_n)$. Now, before we go on to prove a lower bound for Grover Search (by proving a lower bound for the logical OR of N bits) we show the following:

Theorem 3.1. *Let $\mathcal{A}$ be a quantum query algorithm making t queries to O_w, with $w \in [M]^N$ Lets denote the following notation for $\widetilde{w_{i,c}} = \begin{cases} 1 & if w_i = C \in [M] \\ 0 & otherwise \end{cases}$. Then, the amplitude of any basis state is a polynomial in $\widetilde{w_{i,c}}$ of degree less than or equal to t.*

Proof. First, let's take a look at what this algorithm $\mathcal{A}$ actually looks like as a quantum circuit.



In this circuit, we have n input bits, m bits where the output will be stored, and a ancilla bits. $\mathcal{U}_i$ denotes the ith unitary transform on the state, and each O_w is an application of our oracle, finally followed by a measurement $\mathcal{M}$. Now, we assume that our input is initially all 0's, and we continue the proof by induction on t. For our base case, we will consider the state of all the qubits after some unitary gate $\mathcal{U}_0$ has been applied. Our basis state originally is

$$|0^n\rangle \otimes |0^m\rangle \otimes |0^a\rangle$$

Now, after we apply the first unitary transformation to this basis state, we get a state which looks like

$$\sum_{j=0}^{w-1} \sum_{b \in \{0,1\}^m} \sum_{z \in \{0,1\}^a} \alpha_{jbz} |j\rangle |b\rangle |z\rangle$$

Now, we can see that since α_{jbz} is always a constant coefficient, that this is a 0 degree polynomial. Thus, our base case is done.

Induction Hypothesis:
After k steps, we have some state $|\psi\rangle$, which looks like

$$\sum_{j=0}^{w-1} \sum_{b\in\{0,1\}^m} \sum_{z\in\{0,1\}^a} P_{j,b,z}(\widetilde{w}) |j\rangle |b\rangle |z\rangle$$

Where each $P_{j,b,z}(\widetilde{w_s})$ is a polynomial of degree at most k.

Inductive Step Now, we want to apply O_w to this state. This application yields:

$$O_w |\psi\rangle = \sum_{j=0}^{w-1} \sum_{b\in\{0,1\}^m} \sum_{z\in\{0,1\}^s} P_{j,b,z}(\widetilde{w_s}) |j\rangle |b \oplus w_j\rangle |z\rangle$$

Which we can rewrite with $b' = b \oplus w_j$ as

$$\sum_{j=0}^{w-1} \sum_{b'\in\{0,1\}^m} \sum_{z\in\{0,1\}^s} \left(\sum_{a\in\{0,1\}^m} \widetilde{w_{j,a}} * P_{j,b'\oplus a,z}(\widetilde{w}) \right) |j\rangle |b'\rangle |z\rangle$$

Now, we can see that this application results in a polynomial of degree at most $k+1$, as $(\sum_{a\in\{0,1\}^m} \widetilde{w_{j,a}} P_{j,b'\oplus a,z}(\widetilde{w}))$ will increment the degree by at most one, as each P is of degree at most k. Thus, by induction, the degree increases by at most once per query made, so for a t query algorithm, the amplitude of any bases state is a polynomial in $\widetilde{w_i, c}$ of degree at most t. □

Corollary 3.2. *Acceptance (YES) of $\mathcal{A}$ is a real-coefficient polynomial P in $\widetilde{w_{i,c}}$ of degree less than or equal to $2t$*

Proof. The probability of measuring some $|j\rangle |b\rangle |z\rangle$ is $P_{j,b,z} P^*_{j,b,z}$. Because $P_{j,b,z}$ and $P^*_{j,b,z}$ are both polynomials of degree at most t from theorem 3.1. Their product must also be real, because we are taking the magnitude of each $P_{j,b,z}$, which is always real. □

Consider a decision problem. In this , $M = 2$ as the two colors we have are "yes" and "no." For $w \in \{0,1\}^n$ then P is just an approximating polynomial of deg $\leq 2t$ in $w_1, w_2, \cdots, w_i, \cdots, w_n$. In this case, we have $\widetilde{w_{i,1}} = w_i$ and $\widetilde{w_{i,0}} = 1 - w_i$. Thus, P is a *multilinear* polynomial of degree wt. We know that each variable is never to a degree of more than 1, because $\widetilde{w_{i,c}}^2 = \widetilde{w_{i,c}}$ Thus, we describe P as

$$P(w_1, \cdots, w_n) = \sum_{S\subseteq\{1,\cdots,N\}} c_s \prod_{i\in S} w_i$$

Corollary 3.3. *If $\mathcal{A}$ solves some decision problem $F : [M] \to \{0,1\}$ with t queries, then there is a degree $2t$ polynomial P in terms of all $\widetilde{w_{i,c}}$ such that $Pr[\texttt{acceptance}] = P(w)$. For an error bounded algorithm, this polynomial has the property that*

$$|P(w) - F(w)| \leq \frac{1}{3}$$

Now, as we proceed, we can show a minimum number of queries for $\mathcal{A}$, by showing a minimum degree of P that satisfies all of its requirements! Let's consider Grover's algorithm as an example.

4 Example: Grover's Total search

Now, we proceed to conclude with an example of Grover decision

We have from earlier that $F = OR$ on N bits, and we want to show that this *requires* $\Omega(\sqrt{N})$ queries to compute this property F. Essentially, we want to show that any approximating polynomial P for OR has $\deg(P) \geq \Omega(\sqrt{N})$.

We begin by defining some function Q, of the form $Q(w_1, w_2, \cdots, w_n)$ to be a symmetrized polynomial of our approximating multivariate polynomial P, and let $\deg P = 2t$. That is to say,

$$Q = \frac{1}{N!} \sum_{\pi \in S_w} P(w_{\pi(1)}, w_{\pi 2}, \cdots, w_{\pi(n)})$$

Since Q is a sum of polynomials of degree $2t$, then $\deg Q \leq \deg P = d$. Since P approximates F, then we have $P(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$ and $P(w_1, w_2, \cdots, w_n) \in [\frac{2}{3}, 1]$ if some $w_i = 1$. Now, we want to show that Q also satisfies this approximation.

Theorem 4.1. *If P approximates F, then Q also does as well.*

Proof. This proof falls into two cases. We can say that the input of P is either all zeroes, or at least one non-zero term.

Case 1 If the input is all zeroes, then we have $P(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$ by our definition of P. Thus, we can compute a range for Q by plugging in, resulting in $Q(0, 0 \cdots 0) = \frac{1}{N!}(N! P(0, 0, \cdots, 0)) = P(0, 0, \cdots, 0)$, so clearly $Q(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$

Case 2 In this case, because our input to P is not all 0s, then each permutation of the inputs cannot be all 0s. Thus, each permutation of these inputs on P can output a different output in $[\frac{2}{3}, 1]$. However, we know that the sum over all permutations must be in $[N!\frac{2}{3}, N!]$ by adding all of the outputs, and dividing this by $N!$ gives us Q, which is still in $[\frac{2}{3}, 1]$

Thus, because $Q(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$ and for all other $Q(w)$ with some $w_i = 1$, $Q(w) \in [\frac{2}{3}, 1]$, Q also approximates F. □

We've shown that Q, like our polynomial P, satisfies F. However, it is interesting to note that Q is a *multivariate* polynomial, which does not change on any permutations of come input. Now, let's observe some interesting properties of Q.

Observation 4.2. *Q is symmetric in $w = (w_1, w_2, \cdots, w_n)$. That is to say, it doesn't change if you permute its arguments. Since $w_i \in \{0, 1\}$, then $Q(w)$ depends only on the numbers of 1s in its input, or its hamming weight (denoted as $|w|$.) This implies, as you'll show in the homework, that Q depends on z, with $z = \sum_{i=1}^{N} w_i$. As a consequence of this, we can argue that $Q = q(z)$ for some univariate polynomial q of z, with $\deg(q) \leq \deg(Q)$. However, we recall that q is defined only on integer values of z. $q(z) = 0$ if $z = 0$, and $q(z) = 1$ $\forall z \geq 1 \in \mathbb{Z}$. Other values of q are undefined, and can take any arbitrary value.*

Claim 4.3.

$$\deg q(z) \geq \Omega(\sqrt{N})$$

Proof. To prove this, we're going to need **Markovs *other* Inequality**, which states

Lemma 4.4 (**Markov's *other* Inequality**). *For some polynomial $P : \mathbb{R} \to R$ of degree d such that P is bounded in some box of height h and length l and univariate of some variable z, then*

$$|P'(z)| \leq d^2 \frac{h}{l}$$

inside the box of height h and length l [Mar90]

For an example, consider a box bounded by $[-1, +1] \times [-1, +1]$. By this other inequality, we have that if $|p(z)| \leq 1$ for all $|z| \leq 1$, then by this inequality we know $|p'(z)| \leq \deg(p)^2$ for all $|z| \leq 1$, because $h = l = 1$.

As we proceed, we'll make a convenient assumption at our conclusion, and then discuss how the assumption was not actually necessary, so our claim holds for all $q(z)$. Let's assume that $q(z) \in [0, 1]$ for all $z \in [0, N]$, even though we only know the restrictions on $q(z)$ for integer inputs z. By assuming this, we can get a height of 1 on our box which has length N. Using Markov's inequality, we can get that $|q(z)| \leq \frac{deg(q)^2}{N} = \frac{4t^2}{N}$ Because q goes from a value at most $\frac{1}{3}$ at $z = 0$ to a value of at least $\frac{2}{3}$ at $z = 1$, we must have by the Mean Value Theorem that $|q'(z)| = \frac{1}{3}$ for some $z \in [0, 1]$ Plugging this $|q'(z)|$ into Markov's other inequality, we get $\frac{1}{3} \leq \frac{4t^2}{N}$, which solves to show $t \geq \Omega(\sqrt{N})$.

Now, we want to show why we can discard our assumption that $q(z) \in [0, 1]$ for all $z \in [0, N]$. Disregarding this assumption, we realize that we know longer have a guarantee of $h = 1$ inside of our box. Now, we'll let the height of our box be $h = \max_{z \in [0,N]} |q(z)|$. If $h = 2$, the above analysis is still valid, with some constant factor going into the $\Omega(\sqrt{N})$. Now, let's consider the case where $h \geq 2$, and our assumption was entirely invalid. In this case, we know we have $|q(z)| = h$ at some $z = u$. Because $q(z)$ is bounded on integers, $q(\lfloor u \rfloor) \leq 1$. By the definition of a floor function, we have $u - \lfloor u \rfloor \leq 1$. Again, we can use the Mean Value Theorem to see that $|q'(z)| \geq \frac{h-1}{u-\lfloor u \rfloor} \geq h - 1 \geq \frac{h}{2}$ for some z with $\lfloor u \rfloor \leq z \leq u$. Now, we have a second relationship between $|q'(z)|$ and the height of our box. We have

$$|q'(z)| \leq \deg(q)^2 \frac{(2h)}{N} \leq 8t^2 \frac{h}{N}$$

And also

$$|q'(z)| \geq \frac{h}{2}$$

Thus, combining these two facts, we get $\frac{h}{2} \leq 8t^2 \frac{h}{N}$. Regardless of the behaviour of $q(z)$ in between integer inputs for z, by this analysis and Theroem 4.1 we have shown that the number of queries t required for taking the OR of N bits is $\Omega(\sqrt{N})$. Consequently, Grover's search algorithm has optimal query complexity for a quantum computer. □

References

[Bac06] Dave Bacon. Simon's algorithm. *CSE 599d Course Notes*, 2006.

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

[BDH+01] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. *Proceedings of 15th IEEE Conference on Computational Complexity*, pages 131–137, 2001.

[Chi13] Andrew Childs. Query complexity and the polynomial method. *Quantum algorithms (CO 781/CS 867/QIC 823, Winter 2013)*, 2013.

[Mar90] A.A. Markov. On a question by d. i. mendeleev. *Zap. Imp. Akad. Nauk SPb.*, 62:1–24, 1890.