CS 880: Quantum Information Processing 09/21/2010

Lecture 8: Quantum Fourier Transform

Instructor: Dieter van Melkebeek Scribe: Balasubramanian Sivan

Last class we established a lower bound on the number of queries needed for quantum search in an unsorted database. Today, we give general techniques for proving quantum lower bounds in the black-box oracle model. We also begin our discussion on the quantum Fourier transform.

1 Generalized adversarial argument

1.1 Oracle as a characteristic sequence

An oracle is a function $f : \{0,1\}^n \to \{0,1\}$ to which we have blackbox access. We view the value of this function f at different inputs as variables, i.e., $f(0^n) = y_0,\ f(0^{n-1}1) = y_1, \ldots,\ f(1^n) = y_{2^n-1}$. This sequence of y_i's is called the characteristic sequence of the oracle f. Given this, the complexity of quantum search, which returns an x such that $f(x) = 1$, is at least the complexity of computing the OR of all these y_i's. We proved yesterday that the query complexity (i.e., number of queries) of any quantum algorithm for computing $\mathrm{OR}(y_0, y_1, \ \ldots, y_{N-1})$ with bounded error is $\Omega(\sqrt{N})$, where $N = 2^n$.

More generally, we ask, given a function $F : \{0,1\}^N \to \{0,1\}$, what is the complexity of computing F. The function F here is to be viewed as a function of the characteristic sequence of the oracle. We give a theorem for general F today, which is a generalization of the adversary argument we gave yesterday for the special case of $F = \mathrm{OR}$.

1.2 Quantum lower bound for general functions

The intuition behind the theorem is that the state vectors for inputs that map to different values under F must start out identical and end up almost orthogonal. The only operations that can induce a difference in the state vectors are the oracle queries. If we can construct a collection of inputs pairs from $F^{-1}(0) \times F^{-1}(1)$ and argue that on average each individual oracle query can only induce a small difference in the state vector, we know that a lot of oracle queries are needed. In order to construct that collection, we make use of the fact that for a given state vector and two inputs x and y, the oracle query can only induce a significant difference if the state vector puts a lot of weight on the positions where x and y differ. Thus, in order to make that difference small on average, we consider pairs of inputs $(x, y) \in F^{-1}(0) \times F^{-1}(1)$ that have small Hamming distance. This leads to the following quantitative statement, in which R denotes the pairs of inputs we consider and which we choose adversarially so as to obtain as large a lower bound on the query complexity as possible.

Theorem 1. *Given $F : \{0,1\}^N \to \{0,1\}$, $X \subseteq F^{-1}(0)$, $Y \subseteq F^{-1}(1)$, and $R \subseteq X \times Y$. Let $d_{left} = \min_x |\{y : (x,y) \in R\}|$ and $d_{right} = \min_y |\{x : (x,y) \in R\}|$. Let d'_{left} and d'_{right} be such that*

- $(\forall x \in X)(\forall i \in \{1,2,\ldots,N\})\left|\{y \in Y \big| (x,y) \in R, \text{ and } x_i \neq y_i\}\right| \leq d'_{left}$, *and,*

- $(\forall y \in Y)(\forall i \in \{1,2,\ldots,N\})\left|\{x \in X \big| (x,y) \in R, \text{ and } x_i \neq y_i\}\right| \leq d'_{right}$.

Then, any constant error quantum algorithm for computing F must make $\Omega\left(\sqrt{\frac{d_{left}d_{right}}{d'_{left}d'_{right}}}\right)$ queries.

The set R in the above theorem can be viewed as a bipartite graph, with X as the left partite set and Y as the right partite set, with and edge between $x \in X$ and $y \in X$ whenever $(x,y) \in R$. With this view, d_{left} is the minimum degree of a vertex in X and d_{right} is the minimum degree of a vertex in Y. The R in the theorem is to be viewed as that subset of $X \times Y$ such that we care about the behavior of the algorithm only in R.

We do not prove the theorem, but show how to apply it for two functions, namely the OR function and the AND of OR's function. Note that the theorem does not place restriction on how we pick our X, Y and R. It holds irrespective of this choice of X, Y and R. But in order to get a high lower bound, we have to be clever in picking these quantities so that d_{left}, d_{right} are high, and d'_{left}, d'_{right} are low.

Example: $F =$ OR. To get a good lower bound, we make the same choice as in last lecture for X, Y and R:

- $X = \{0^N\}$; (note that this is the only choice possible for X)
- $Y = \{y|$ has exactly one 1$\}$;
- $R = X \times Y$.

From this, we compute the relevant quantities for lower bound: $d_{left} = N$, $d_{right} = 1$, $d'_{left} = 1$, and $d'_{right} = 1$. On substituting these values in the formula for lower bound in Theorem 1, we get a $\Omega(\sqrt{N})$ lower bound. ⊠

Example: $F =$ AND of ORs, i.e., F is a Boolean formula in conjunctive normal form with $\sqrt{N}$ clauses, and each clause has $\sqrt{N}$ literals. The first clause is an OR of the first $\sqrt{N}$ variables $y_0, y_1, \ldots, y_{\sqrt{N}-1}$ (called the first block), the second clause has the next $\sqrt{N}$ variables and so on. We make the following choices for X, Y and R.

- $X = \{x|$ x has one block of all zeros and each of the remaining $\sqrt{N}-1$ blocks has exactly one 1$\}$;
- $Y = \{y|$ each of the $\sqrt{N}$ blocks of y has exactly one 1$\}$;
- $R = \{(x,y) \in X \times Y|$ x and y differ in exactly one position$\}$.

From these choices, we can compute: $d_{left} = \sqrt{N}$, $d_{right} = \sqrt{N}$, $d'_{left} = 1$, and $d'_{right} = 1$. On substituting these values in the formula for lower bound in Theorem 1, we get a $\Omega(\sqrt{N})$ lower bound. ⊠

Exercise 1. *Give an algorithm for computing the above F, i.e., AND of OR's, in $O(\sqrt{N}\log N)$ queries. Hint: Use Grover's algorithm. Recall that Grover's algorithm can compute OR, as well as AND.Use it to compute AND of OR's.*

We remark that there exists a stronger version with weights on the edges, including a version where the weights can be negative, which gives tight results up to a constant factor for the bounded-error query complexity of any function F.

2 The polynomial method

In this section we give another general technique, known as the polynomial method, for establishing quantum lower bounds.

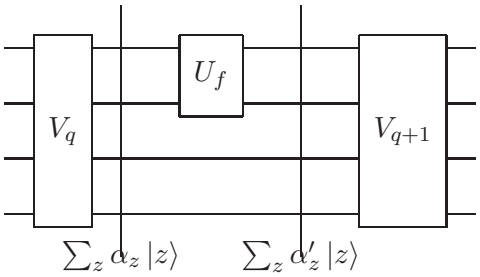
Theorem 2. *Given a quantum query algorithm with q queries, the amplitudes at the end of the algorithm are multivariate polynomials in the y_i's, of degree at most q.*

Proof. We prove this by induction on the number of queries made, q.

q=0: This case is trivial as no y_i is revealed with zero queries, and hence we have a polynomial of degree zero in y_i's.

Inductive step: Recall our model of quantum algorithms:

1. Sequence of unitary operations that do not depend on the oracle,
2. followed by an oracle query



We repeat the above two steps q times. The i-th appplication of unitary operations is denoted by the operator V_i, and the i-th query operation is U_f. Let the state of the system after the operation V_q but before the application of q-th query operation U_f be $\sum_z \alpha_z |z\rangle$, and the state after the q-th query operation be $\sum_z \alpha'_z |z\rangle$. Note that the base states $|z\rangle$ tell us what queries we make. Consider an arbitrary $z = xbu$, (where $|x| = n$). The n bits in x are the query bits, the bit b is the answer bit, and u contains the rest. How are α'_{xbu} and α_{xbu} related? They are related as:

$$\begin{aligned} \alpha'_{xbu} &= \alpha_{xbu} \text{ if } f(x) = 0 \\ &= \alpha_{x\bar{b}u} \text{ if } f(x) = 1 \end{aligned}$$

Note that $f(x)$ is simply y_x, which is one of our variables. Thus we have

$$\alpha'_{xbu} = \alpha_{xbu}(1 - y_x) + \alpha_{x\bar{b}u} y_x$$

i.e., α'_{xbu} is a linear function in y_x and the α's. By inductive hypothesis, we have that α_{xbu} and $\alpha_{x\bar{b}u}$ are multivariate polynomials of degree at most $q-1$. Thus α'_{xbu} is of degree at most q. There is just one more thing left to prove, namely the effect of V_{q+1} on the degree. But V_{q+1} is a linear operation, and hence won't increase the degree. This proves the theorem. □

Now, we use this theorem to establish query lower bounds for computing the OR function. We do this for the bounded error case as well as the exact algorithm case.

Bounded error: Given an ϵ-error algorithm for the OR function with q queries, we have

$$\left| F(y) - \underbrace{\sum_{z\text{: output bit is 1}} \left|\alpha_z^{(final)}\right|^2}_{\text{Probability that the algorithm outputs 1}} \right| \leq \epsilon$$

The quantity subtracted from $F(y)$ in the above equation is a polynomial of degree at most $2q$. This means that there is a multivariate polynomial p of degreee at most $2q$ such that

$$\forall y \in \{0,1\}^N |F(y) - p(y)| \leq \epsilon$$

When F is the OR function, we can show that the minimum degree of such a polynomial is $\Omega(\sqrt{N})$, thus proving the desired lower bound.

Exact algorithm: We can similarly show that $\Omega(N)$ queries are required to compute the OR function exactly because the exact representation of the OR function requires a polynomial of degree N.

Exercise 2. *Prove that N queries are needed for computing the OR function exactly.*

Using the polynomial method, it canbe shown that for any function $F : \{0,1\}^N \to \{0,1\}$, if the optimal deterministic algorithm makes D queries in the worst case, then the quantum algorithm with zero error must make $\Omega(D^{1/2})$ queries and a constant error quantum algorithm must make $\Omega(D^{1/6})$ queries. For symmetric Boolean functions (a function whose value depends only on the number of 1's in the input), it is known that any constant error algorithm is required to make $\Omega(D^{1/2})$ queries, and it is conjectured that the same holds for any function.

Note that the polynomial relationship between the quantum and classical query complexity of functions $F : \{0,1\}^N \to \{0,1\}$ does not contradict the exponential gap in query complexity we have seen for (the decision version of) Simon's problem, for example. This is because the F underlying Simon's problem is not a function but only a partial function, corresponding to promise we are given in Simon's problem that the input oracle is guaranteed to be of one of two special forms.

3 Fourier transform

3.1 Fourier transform over the reals

We briefly discuss Fourier transform in the classical setting in this section. In the classical setting, we define Fourier transform as follows. Let $f : \mathbf{R} \to \mathbf{C}$ such that $\int |f(x)|^2 \, dx < \infty$, i.e., f is square integrable. Then, we write the Fourier transform of f as $\widehat{f}(w)$, and is given by

$$\widehat{f}(w) = \frac{1}{\sqrt{2\pi}} \int f(x) e^{i\omega x} \, dx \tag{1}$$

and we have that,

$$f(x) = \frac{1}{\sqrt{2\pi}} \int \hat{f}(w) e^{-i\omega x} \, dw \tag{2}$$

Equation (2) says that we can write any function f as a superposition of the harmonics, represented by the $e^{i\omega x}$ terms. The coefficients of this superposition are the $\widehat{f}(w)$'s, given by (1). We can think of the Fourier transform operation from f to $\widehat{f}$ as an orthonormal basis transformation, namely a transformation from the standard basis of δ functions, which itself is orthonormal, to another orthonormal basis.

Apart from orthonormality, the most interesting property of Fourier transform is the fact that the Fourier transform of the convolution of two functions, is the product of the Fourier transforms of the individual functions. The convolution $f * g$ of two functions f and g is defined as

$$f * g(x) = \frac{1}{\sqrt{2\pi}} \int f(x-y)g(y)\,dy.$$

We have that

$$\widehat{f * g}(w) = \widehat{f}(w) \cdot \widehat{g}(w).$$

where $\widehat{f}(w) \cdot \widehat{g}(w)$ is a point-wise product. The Fourier transform operation can be generalized to groups other than $\mathbf{R}$. Note that $\mathbf{R}$ is a group under addition.

3.2 Fourier transform over general groups

Let $f : \mathbf{G} \to \mathbf{C}$ be a function from a group $\mathbf{G}$ to complex numbers. Can we come-up with an orthonormal basis such that there exists a transformation from convolutions to products? It turns out that we can do this for any finite group. The situation becomes easier for finite abelian groups, which are groups for which group operation is commutative. For finite abelian groups, the harmonics are the scaled versions of their characters.

Definition 1. *A character of a group is a homomorphism from the group to complex numbers. A homomorphism is a function that preserves the group operation, i.e., if χ is a homomorphism, then $\chi(a \cdot b) = \chi(a) \cdot \chi(b)$.*

Properties: For any two characters, $\chi \neq \chi'$, and any $g \in \mathbf{G}$,

1. $|\chi(g)| = 1$
2. $(\chi, \chi') = 0$, where $(\chi, \chi') = \sum_{g \in G} \overline{\chi(g)}\chi'(g)$

Proof. To prove the first property, recall that, any group element, on sufficient powering gives the unit element of the group, and we know that χ(unit-element of $\mathbf{G}$) = unit-element of $\mathbf{C} = 1$. Now, we use the fact that $\chi(g^n) = (\chi(g))^n$. Thus, some power of $\chi(g)$ must be 1, implying that $|\chi(g)|$=1.

We prove the second property in two steps. In the first step, let χ' be the trivial character which maps all elements to 1. We must now prove that for any χ which does not map every element to 1, $\sum_{g \in G} \overline{\chi(g)} = 0$, which is the same as proving $\sum_{g \in G} \chi(g) = 0$. We know that for every group element a, $\sum_{g \in G} \chi(g) = \sum_{g \in G} \chi(g.a) = \sum_{g \in G} \chi(g)\chi(a) = \chi(a) \sum_{g \in G} \chi(g)$. We pick an a such that $\chi(a) \neq 1$. This means that $\sum_{g \in G} \chi(g) = 0$.

Now consider the case where χ' does not map every element to 1. For any two characters χ and χ', the function $\overline{\chi}\chi'$ is also a character. Observe that $\overline{\chi}\chi'$ cannot map every element to 1, as that would imply that $\chi = \chi'$. Now we use the proof in the first step for our character $\overline{\chi}\chi'$, thus yielding $\sum_{g \in G} \overline{\chi(g)}\chi'(g) = 0$. □

The question now is, whether we have enough of these characters. If the number of characters equals the size of the group, then we have an orthogonal basis, because, the dimension of the space of all functions from $\mathbf{G}$ to $\mathbf{C}$ is $|\mathbf{G}|$. To make the basis orthonormal, we scale the characters by a factor of $|\sqrt{\mathbf{G}}|$, because $(\chi, \chi) = |\mathbf{G}|$.

4 Next time

In the next lecture, we will see that a finite abelian group is isomorphic to a direct sum of finite cyclic groups, and discuss the characters of a cyclic group. We will prove that there are $|\mathbf{G}|$ characters for any finite abelian group by explicitly constructing these characters. We will also discuss some applications of Fourier transform, like phase estimation and eigen-value estimation.