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

Lecture 5: Quantum Query Complexity

September 23, 2015

Lecturer: Ryan O'Donnell *Scribe: Linus Hamilton*

1 Quick Summary

Today, we'll introduce the query model, in which you get a black-box function f and have to answer a question about it. We'll show off various query problems where quantum computers give a huge speedup over classical computers. Without the query model, we wouldn't be able to do this, because we have no idea how to prove useful lower bounds for ordinary time complexity.

2 Grover Recap

This is the problem which Grover's algorithm [Gro96] solves:

Problem. (Grover) You get a function $f : \{0,1\}^n \to \{0,1\}$. You are promised that there is a unique input x such that $f(x) = 1$. You have to find x.

A few details need clarifying. How do we "get" the function f? In order to use f in a quantum algorithm, we had better be able to put it in a quantum circuit. To put it in a quantum circuit, we need to make it reversible. So, we will think of f as a big gate which inputs x, and CNOTs $f(x)$ over to some free registers. In bra-ket terms, we will get f in the form of a black box gate O_f which does $|x\rangle \otimes |b\rangle \mapsto |x\rangle \otimes |b \oplus f(x)\rangle$.[1]

In CS, this kind of black-box computation is often called an oracle, after the mysterious question-answering Usenet Oracle. So we will call O_f an oracle gate for f.

When the output of f is only one bit, as it is for Grover's algorithm, it is often convenient to define an alternate oracle gate $O_f^{\pm}$ by $|x\rangle \to (-1)^{f(x)} |x\rangle$. It doesn't matter which one we pick; if you have one of O_f and $O_f^{\pm}$, you can easily emulate the other. After all, they differ by a unitary transformation.

Using $O_f^{\pm}$, we can rewrite Grover's algorithm.

[1] In real life, when computing f we will probably need some ancillas. But we can ignore them. We just have to remember to use the 'uncompute' trick to set them all back to $|0\rangle$, so that they don't interfere with amplitude cancellation later.

1. Start with the state $|000\ldots00\rangle$, with the same number n of registers as the number of input bits to f.
2. Apply a Hadamard gate to each qubit. This is the same as applying $H^{\otimes n}$, the Hadamard gate tensored with itself n times.
3. Apply $O_f^{\pm}$.
4. Apply the Grover flip operator D, which flips all the amplitudes over their average.
5. Repeat steps 3 and 4 for $O(\sqrt{2^n})$ many times. Then measure all the qubits. With $> 1\%$ probability, you will measure the hidden answer x. (The value of 1% doesn't matter – as long as it's a nonzero constant, you can repeat the algorithm $O(1)$ times until it works.)

Grover's algorithm shows the power of quantum computing. A quantum computer can search a black-box list of $N = 2^n$ items in $O(\sqrt{N})$ time. Compare this to the classical setting: with no information about f, you would have to try $O(N)$ inputs on average to find the right answer.

Grover's algorithm gives a free quadratic speedup for any search problem. For example, to solve SAT in $\sqrt{2^n}$ time, just set f to a checker for your SAT instance, and use Grover to check all 2^n assignments to your variables.

(Actually, this has a catch: Grover's algorithm requires that f have a *unique* input returning 1, whereas a SAT instance need not have a unique solution. You will show how to patch this hole in Homework 3.)

Intuitively, quantum computers excel at finding patterns in data. Many quantum algorithms follow the same lines as Grover: input a black-box function f with some pattern, and figure out what the pattern is.

3 The Query Model

Let's invent a model to analyze algorithms like Grover's. In this *query model*, you are given black-box access to some function f. You have to answer some question about f. Instead of measuring the time complexity of your algorithm, we measure the *query complexity*: the number of queries it makes to f.

Why do we use the query model? Shouldn't we, as computer scientists, only care about how much time an algorithm takes? It turns out that the query model has several advantages:

- It's simple to analyze.
- In the query model, we can prove nontrivial lower bounds! Compare this to the rest of complexity theory, where we don't even know whether you can solve SAT in linear time.
- Often, the query complexity of an algorithm is the same as its time complexity. That is to say, often the function f is efficient to implement, and often the non-oracle parts of an algorithm are also efficient.
- All (?) known interesting quantum algorithms fit in the query paradigm. (For example, Shor's factorization algorithm is really a special use-case of a general period-finding query problem. More about that later!)

4 Example Query Problems

In this section, all our black-box functions will input a string of n bits. We will set $N = 2^n$ to be the number of possible inputs to the function.

We will sometimes think of a function as a list of N outputs. For example, Grover's algorithm inputs a list f containing N bits, and finds the location of the unique 1 bit using only $\sqrt{N}$ queries.

4.1 The Hidden Shift Problem

Input: two lists f and g, both containing N distinct elements. It is promised that f and g are rotations of each other; that is, there exists s such that $f(x) = g(x + s \pmod N)$ for all integers $x \in [N]$.

Output: the hidden shift s.

For now, ignore quantum, and say we're in the classical setting. Then the obvious hidden shift finding algorithm has complexity $O(N)$. However, it is possible to solve this problem classically with $O(\sqrt{N})$ queries. Notice that since f has all distinct elements, it suffices to find one pair of matching elements between f and g. One can do this using the baby-step giant-step attack. Look at the first $\sqrt{N}$ elements of f ("the baby steps"), and also look at every $\sqrt{N}^{\text{th}}$ element of g ("the giant steps"). Since every shift can be expressed as a baby step plus a giant step, you're guaranteed to find a match.[2]

Classically, you can't beat $\sqrt{N}$. However, in the quantum case, one can actually solve the problem in $O(\log N)$ queries! The algorithm is as follows... kind of.

Step 1: Prepare the state

$$\frac{1}{\sqrt{2N}} \sum_{x \in [N]} |xf(x)\rangle + |xg(x)\rangle \,.$$

Step 2: Apply a Fourier transformation over $\mathbb{Z}_N$. (We'll explain what that means later in the course.)
Step 3: Do some more stuff. [Ip02]

We may fill in the "more stuff" later in the course. The point of this example is (1) that quantum computers can do some pretty crazy things, and (2) that the first two steps of this algorithm are extremely common in quantum query algorithms. Prepare a superposition over all states, and then apply a Fourier transform. Intuitively, the period-finding qualities of the Fourier transform help extract hidden patterns in f.

[2] Baby-step giant-step is a real attack used to break cryptosystems such as discrete log.

4.2 Simon's Problem

Simon's problem [Sim94] is the same as the hidden shift problem, except that instead of a shift, we have an XOR.

> Input: two lists f and g, both containing N distinct elements. It is promised that f and g are XOR-shifts of each other; that is, there exists s such that $f(x) = g(x \oplus s)$ for all integers $x \in \{0,1\}^n$.
>
> Output: the hidden XOR-shift s.

Again, this problem takes $\sqrt{N}$ queries classically. Use the baby-step giant-step algorithm, where the first $n/2$ bits are the baby steps and the last $n/2$ bits are the giant steps. Also again, this problem takes $O(\log N)$ queries quantumly. The algorithm is practically the same:

> Step 1: Prepare the state
>
> $$\frac{1}{\sqrt{2N}} \sum_{x \in [N]} |xf(x)\rangle + |xg(x)\rangle \,.$$
>
> Step 2: Apply a Fourier transformation over $\mathbb{Z}/n\mathbb{Z}$. (This turns out to be the same as applying $H^{\otimes n}$, i.e. applying a Hadamard gate to every register.)
> Step 3: Do some more stuff.

Sorry about the repeated "do some more stuff." I swear, we will do actual math in section 4.4.

4.3 Period Finding

This is the problem that Shor used, along with some number-theoretic trickery, to factor numbers efficiently on a quantum computer. [Sho99]

> Input: a list f of N distinct elements. Note: here, N is NOT necessarily a power of 2. It is promised that f is periodic with some period dividing N.
>
> Output: the period of f.

Clasically, there's a tricky way to do this in $N^{1/4}$ time. But that's still exponential. Quantumly, this is again possible in $O(\log N)$ time. Can you guess the algorithm? That's right: prepare the state $\frac{1}{\sqrt{N}} \sum_{x \in [N]} |xf(x)\rangle$ and apply a Fourier transform. This time, there is no "do some more stuff": you just measure and you're done. We'll cover this algorithm in more detail soon.

4.4 Deutsch-Josza

We've had enough "do some more stuff." Let's do some math. Here's the Deutsch-Jozsa problem.[DJ92]

> Input: a list f of N bits. The list is either all-zero or balanced. That is, either all of the bits are 0, or exactly half of them are 1.
>
> Output: whether the list is all-zero or balanced.

Clasically, this is easy. By inspecting bits randomly, you can find the answer with error ϵ with just $\log(1/\epsilon)$ queries. On the other hand, if you want a 100% correct answer, you need to examine more than $N/2$ bits.

But quantumly, you can find the answer, 100% of the time, in *one* query. Here's the algorithm.

> Step 0: Initialize n registers to $|000\ldots0\rangle$.
> Step 1: Apply $H^{\otimes n}$, i.e. apply a Hadamard gate to each wire. The resulting state is
>
> $$\frac{1}{\sqrt{N}}\sum_{x\in[N]}|x\rangle\,.$$
>
> Step 2: Apply $O_f^{\pm}$. (This is the oracle gate $|x\rangle \mapsto (-1)^{f(x)}|x\rangle$ discussed in the Grover Recap section.)
> Step 3: Apply $(H^{\otimes n})^{-1}$.
> Step 4: Measure. If the result is all 0, return "all zeroes." Otherwise, return "balanced."

Let's prove that this works.

Proof. Suppose, initially, that the list was all zeroes. Then $O_f^{\pm}$ is the identity transformation. Then the overall transformation in steps 1-3 is $(H^{\otimes n})^{-1}IH^{\otimes n}$. This cancels out to the identity, so steps 1-3 have no overall effect. Therefore, when you measure, you will find the initial state $|000\ldots0\rangle$. So, you will correctly output "all zeroes."

Now suppose, initially, that the list was balanced. Then, after applying $O_f^{\pm}$, you have the state

$$\psi = \frac{1}{\sqrt{N}}\sum_{x\in[N]}\pm|x\rangle$$

where the $\pm$ is + half of the time and - half of the time.

Then, in Step 4, we apply $(H^{\otimes n})^{-1}$. Now, we could do the math and find the amplitude of $|000\ldots0\rangle$ after applying $(H^{\otimes n})^{-1}$. This would be fairly straightforward, and it would give the answer 0 as we want, but instead we can use a clever trick.

The state ψ is orthogonal to the state $\chi = \frac{1}{\sqrt{N}}\sum_{x\in[N]}|x\rangle$, as one can check by performing an inner product. Therefore, after applying the unitary transformation $(H^{\otimes n})^{-1}$, the states $(H^{\otimes n})^{-1}\chi$ and $(H^{\otimes n})^{-1}\psi$ are still orthogonal. But since $\chi = H^{\otimes n}|000\ldots0\rangle$, the state $(H^{\otimes n})^{-1}\chi$ is equal to $|000\ldots0\rangle$. Therefore, the state $(H^{\otimes n})^{-1}\psi$ is orthogonal to $|000\ldots0\rangle$.

Therefore, the amplitude of $|000\ldots0\rangle$ in the final state $(H^{\otimes n})^{-1}\psi$ must be zero. As a consequence, when we measure our qubits, we can never measure the all-zero state. Therefore, we will always correctly say that the list was balanced. □

5 Coming soon...

Later in the course, we'll cover some of these algorithms in more detail, especially the quantum Fourier transform. We'll also show how to prove lower bounds in query complexity, such as proving that you can't beat Grover's $\sqrt{N}$ performance.

References

[DJ92] D. Deutsch and R. Jozsa. Rapid Solution of Problems by Quantum Computation. *Proceedings of the Royal Society of London Series A*, 439:553–558, December 1992.

[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, pages 212–219, New York, NY, USA, 1996. ACM.

[Ip02] Lawrence Ip. Solving Shift Problems and Hidden Coset Problem Using the Fourier Transform. *arvix*, 2002.

[Sho99] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41:303–332, January 1999.

[Sim94] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26:116–123, 1994.