6.841/18.405J: Advanced Complexity Theory May 13, 2002

Lecture 23

Lecturer: Madhu Sudan *Scribe: Rebecca Hitchcock*

Algorithms in Quantum Computing

This lecture covers two algorithms that make use of the quantum computational structures that were set up last lecture. We begin with a review of the Quantum Computing Model.

1 Recap: Quantum Computing Model

1.1 Quantum Circuits

A quantum circuit is s circuit with n wires in and n wires out. A k-nary gate in this circuit is a $2^k \times 2^k$ matrix which is a map on vectors of size k. At the output of the circuit, we observe some the state by sampling some subset of the output wires.

A quantum circuit can compute any function that can be computed by a classical circuit. We wish to determine what is a sufficient collection of gates to compute any function using a quantum circuit. There are three such gates:

1. The Hadamard Gate $\mathrm{H}_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} \end{bmatrix}$. This is a unitary gate which means that H_2 times the complex conjugate of H_2^T is I, the identity matrix. This is the gate that gives us the power of quantum computing by introducing randomness. If we input $|0\rangle$ into this circuit we get $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Measuring the qubit will therefore show 0 with probability $\frac{1}{2}$ and 1 with probability $\frac{1}{2}$.
2. A classical negation gate. This gate has two inputs a and b and shows on its output a on the first wire and $a \oplus b$ where $\oplus$ is XOR. The negation is obtained by inputting 1 for a and the input to be negated as b.
3. A classical AND gate. This gate has three inputs, a, b, and c. The output of the first two inputs are unchanged, that is they output a and b respectively. The third output gives $c \oplus (a \wedge b)$.

Using these gates we can simulate and polynomial sized classical circuit with a polynomial sized quantum circuit.

1.2 Quantum Turing Machines

Quantum Turing machines are represented very similarly to classical turing machines. The differences are that the tape has qubits on it instead of regular bits, and the transition function is given by a local unitary matrix. This model leads us to desire a definition of quantum polynomial time. There are two definitions that appear in the literature, BQP and EQP, which are analogous to BPP and ZPP respectively; BQP represents computation with bounded error and EQP represents computation with zero error.

There are two ways to formalize these complexity classes. The first is in the expected way, defining completeness and soundness in the usual ways and allowing a polynomial number of steps on a quantum turing machine.

The second way uses quantum circuits as described above:

Definition 1 **EQP** $= \{L | \exists$ *a family of quantum circuits* $\{Q_n\}_n$*, where* Q_n *takes n inputs, and a classical polynomial time turing machine* M *that on input* 1^n *outputs* Q_n *such that:*

$$w \in L \Leftrightarrow Pr\left[Q_{|w|}(w) = 1\right] = 1$$

$$w \notin L \Leftrightarrow Pr\left[Q_{|w|}(w) = 1\right] = 0 \ \}$$

Definition 2 **BQP** $= \{L | \exists$ *a family of quantum circuits* $\{Q_n\}_n$*, where* Q_n *takes n inputs, and a classical polynomial time turing machine* M *that on input* 1^n *outputs* Q_n *such that:*

$$w \in L \Leftrightarrow Pr\left[Q_{|w|}(w) = 1\right] \geq \frac{2}{3}$$

$$w \notin L \Leftrightarrow Pr\left[Q_{|w|}(w) = 1\right] \leq \frac{1}{3} \ \}$$

This second set of definitions is generally easier to work with. It is important to remember however that the definition requires not only a polynomial sized circuit, but also the ability to build that circuit in classical polynomial time. This enforces a measure of uniformity, which is necessary because as we recall from our study of classical circuits there exist undecidable languages for which there are polynomial sized circuits.

2 Simon's algorithm

There are three main algorithms that exploit the power of quantum computing. These are Simon's Algorithm, Grover's Algorithm, and Shor's algorithm. Grover's algorithm is for NP search and we will not cover it in this lecture. Both Grover's and Simon's algorithm deal with relativized promise problems.

2.1 Simon's Problem

Simon was interested in a problem inspired by cryptography. In this problem we are given an oracle for a function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ and we wish to determine whether or not it is a one-to-one function. He sets up his problem in the form of a promise problem:

Π_{YES}: $\exists s \in \{0,1\}^n / 0^n \ s.t \ \forall x \ f(x+s) = f(x)$
Π_{NO}: f is a one-to-one function

In order to approach this problem classically, you could imagine a world divided into two subsets, x, and x+s. But the space is so large that we are unlikely to see two vectors that map to the same output because there are 2^n vectors. This problem is in Promise-NP, but Simon's algorithm gives a method for solving it more efficiently.

In order to discuss Simon's algorithm we need a little more information on the unitary Hadamard Transform.

2.2 The Hadamard Transform Revisited

Consider a number of Hadamard gates in parallel. If we have six wires coming in, with an input $x = 1010111$ what do we see on the output? Each output wire has equal probability of being one of two choices. The output is the product of the states, so we know the magnitude of the output is $(\frac{1}{\sqrt{2}})^6$. The expression for the general case is as follows:

$$\frac{1}{2^{\frac{n}{2}}} \sum_{y \in \{0,1\}^n} (-1)^{\langle x,y \rangle} |y\rangle$$

Where $\langle x, y\rangle$ represents the inner product modulo two. This operation is very important to Simon's algorithm.

2.3 Simon's Algorithm

The main idea is to compute a non-trivial function on the 2^n inputs. Unlike a trivial function such as the average, we cannot compute this in classical probabilistic polynomial time. Notice here that we are trying to discover properties of the oracle, so we can set the input to whatever we like. Here we set it to $|0^n 0^n\rangle$. Consider the following sequence of transformations(gates):

$$|0^n 0^n\rangle \xrightarrow{H_2^n} \frac{1}{2^{\frac{n}{2}}} \sum_{x \in \{0,1\}^n} |x 0^n\rangle \xrightarrow{f(x)} \frac{1}{2^{\frac{n}{2}}} \sum_{x \in \{0,1\}^n} |x f(x)\rangle \xrightarrow{H_2^n} \frac{1}{2^{\frac{n}{2}}} \sum_{x \in \{0,1\}^n} \frac{1}{2^{\frac{n}{2}}} \sum_{y \in \{0,1\}^n} (-1)^{\langle x,y \rangle} |y f(x)\rangle$$

The first arrow simply gives us the Hadamard transform of the first n qubits. No signs have been introduced at this point because our input was zero, and if we were to go through another Hadamard transform we would get $|0^n 0^n\rangle$ back. The second transformation is possible because of a relativized version of the idea that we can perform and classical computation using a quantum circuit. The details of this are beyond the scope of this lecture. The third arrow is another Hadamard transform on the first n qubits.

Something nontrivial has happened at this point. Consider what happens if we observe the tape in both the yes and no cases:

NO case: In this case you see every possible combination of strings $\langle x, y\rangle$. The state of the system is:

$$\frac{1}{2^n} \sum_{y,z} (\pm 1) |y, z\rangle$$

There are no collisions and we are left with a uniformly distributed random sample.

YES case: In this case we look at the terms of the sum $2^{-n}(1)^{\langle x,y\rangle}|y, f(x)\rangle$ and $2^{-n}(1)^{\langle (x+s),y\rangle}|y, f(x+s)\rangle$ Since $f(x) = f(x+s)$ we know that these terms are equal, with the possible exception of the (± 1). There are two cases. If $\langle y, s\rangle = 1$ then these two terms will have opposite signs and cancel out. In this case, the corresponding $|y, f(x)\rangle$ will not appear in the output. In the other case, $\langle y, s\rangle = 0$, and they will have the same sign and the resulting coefficient will be $\pm 2 \cdot \frac{1}{2^n}$.
Thus in the yes case we see there are 2^{2n-2} possible vectors $\langle y, f(x)\rangle$, and each appears with equal probability.

In order to distinguish these two cases we take $2n$ samples of the circuit, and focus on the y portion. If the set of vectors $y_1 \ldots y_{2n}$ is of rank n, then we know we are in the NO case. Only if we had a 1:1 function could these vectors have full rank. If these vectors are of rank $n-1$, then we know we are in the YES case and using linear algebra we can find a candidate vector for s. The algorithm has zero error and the s will be correct with high probability.

3 Shor's Algorithm

Simon's algorithm gives us exponential speedup over classical algorithms, given the assumption of relativized quantum computations. Shor's algorithm gives us this same speedup on a specific classical problem (factoring), and doesn't rely on this idea. The main idea of the algorithm is inspired by Simon's method.

3.1 Intuitive Idea

Shor's Algorithm is based on a classical result from the 1970's which says for a factoring a number N reduces to finding the order of a number a mod N. The order of a is the place in the sequence $a^0, a^1, a^2, ..., a^r$ where $a^r \equiv 1(\bmod \mathrm{N})$.This reduction is based on factoring $a^r - 1 \equiv 0(\bmod \mathrm{N})$ to $(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0(\bmod \mathrm{N})$. With high probability, r is even and neither of the terms of the product are zero.

Since Simon's Algorithm seems to compute periods, Shor felt that it might be possible to use the ideas in it to find order. We can find an s s.t.

$$\forall x\, f(x) = f(x \cdot s)$$

Where the $\cdot$ can be any group operation. We can phrase the problem of finding the order of a in the same language.

$$f(i) \triangleq a^i$$

$$\forall i\, f(i) = f(i + r)$$

3.2 A New Unitary Operation

We introduce here a new operation which we claim to be unitary for all choices of q. This operation computes the complex Fourier transform and its expression is given below:

$$|j\rangle \longrightarrow \frac{1}{\sqrt{q}} \sum_{k=0}^{q-1} e^{\frac{2\pi i}{q} \cdot j \cdot k} |k\rangle$$

Although this operation is always unitary, it may not always be local. We ignore this problem for now, and return to it briefly at the end of the lecture.

3.3 Shor's Algorithm

Fix a, N, and some large q. Consider the following sequence of transformations (gates). The operation described above is notated as "O" and the input is two zero vectors from $\mathbb{Z}_q$

$$|00\rangle \xrightarrow{O} \frac{1}{\sqrt{q}} \sum_j |j0\rangle \overset{\text{modular exp.}}{\longrightarrow} \frac{1}{\sqrt{q}} \sum_j |jf(j)\rangle \xrightarrow{O} \frac{1}{q} \sum_j \sum_k e^{\frac{2\pi i}{q} \cdot j \cdot k} |kf(j) \longrightarrow \text{Observe State}$$

Note that this is very similar to the progression made in Simon's Algorithm. In the second step, however, we do not need to rely on relativized arguments to see that this can be computed using a quantum circuit. It is simply a matter of converting a classical circuit into a quantum one.

Claim 3 *k is very close to a multiple of $\frac{q}{r}$, thus, $\frac{q}{k}$ gives a good approximation of r.*

Assume that q is a multiple of r. Then we have:

$$\sum_{j_1=0}^{q-1} \sum_{j_2=0}^{r-1} \sum_k e^{\frac{2\pi i}{q}(rj_1 + j_2)k} |kf(j_2)\rangle = \sum_k \sum_{j_2} |kf(j_2)\rangle \cdot e^{\frac{2\pi i}{q} j_2 k} (\sum_{j_1=0}^{\frac{q}{r}-1} e^{\frac{2\pi i}{q} r j_1 k})$$

Consider the last term of the product, with m set equal to $\frac{q}{r}$.

$$\sum_{j_1}^{m-1} (e^{\frac{2\pi i}{m} k})^{j_i}$$

We see some m^{th} root of unity, depending on the value of k. If k is a multiple of m, we get m for the value of the sum and if not we get 0. We find r by observing many samples and then using a gcd calculation to find $\frac{q}{r}$ from a number of multiples of $\frac{q}{r}$.

3.4 Remaining Details

There are two main ideas that we glossed over in this discussion of Shor's Algorithm.

1. **q is not necessarily a multiple of r**. We handle this using analysis such that $[kr]_q$ is a very small contribution. We are then left with a problem that can be solved using an integer program in two variables.

2. **A q-ary Fourier Transform is not always local** If we pick q to be a power of 2, then in this case we can construct a small quantum circuit implementing q-ary FT.