18.405J/6.841J Advanced Complexity Theory March 22, 2001

Lecture 13

Lecturer: Dan Spielman *Scribe: Abhinav Kumar*

This lecture will be on oracle quantum turing machines.
We wish to compare Quantum complexity classes to ordinary complexity classes.

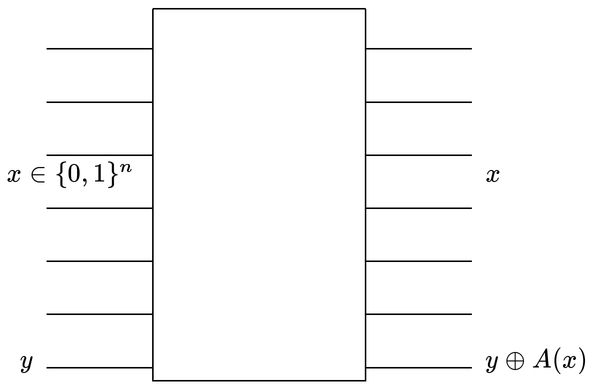
Fact 1 $BQP \subseteq PSPACE$
This is easy to prove, in fact it's on the problem set. A harder proposition is the following.

Fact 2 $BQP \subseteq P^{\#P}$
Another result says

$$\exists A \text{ such that } NP^A \notin BQP^A.$$

So at least in its ability to access an oracle, NP can be more powerful than BQP.

We need to define the notion of accessing an oracle for a quantum computer. The idea is to put quantum gates in the circuit representing the oracle. An oracle is viewed as a function $A : \{0,1\}^* \to \{0,1\}$. We consider the quantum gate Q_{A_n}, as shown in the figure.



Gate Q_{A_n}

To describe the quantum gate Q_A, assume that A tells us what Q_A does on basis states $|x, y>$, $x \in \{0,1\}^n, y \in \{0,1\}$. Q_A sends this to $|x, A(x) \oplus y>$. Extend by linearity to get a unitary transformation.

Then an oracle quantum turing machine just means a quantum circuit which is allowed to incorporate some gates of this form.

Let $L(A) = \{w | \exists x \in A, |x| = |w|\}$. Clearly $L(A) \in NP^A$. We will see that quantum computers can't decide $L(A)$ very well. Putting in one word of a given length makes very slight difference to the functioning of a quantum circuit trying to decide $L(A)$. The rough idea is that if $|A \cap \{0,1\}^n| = 0$ or 1, it doesn't make much of a difference which of these values it is (to a Quantum circuit). The proof idea is similar to the proof of $P^A \subsetneq NP^A$.

Theorem 3 $L(A) \notin BQP^A$

Proof: Suffices to argue about one quantum circuit. The way we elude all Quantum circuits is the same as in the proof of $P^A \subsetneq NP^A$. In fact, if a quantum circuit with T oracle gates accepts $L(A)$ with error probability less than a fixed constant, then $T = \Omega(2^{n/2})$ (this bound is tight - see Grover's algorithm). Define Let the initial state be $|\phi_0>$.

$$|\phi_T> := U_T Q_A ... Q_A U_1 |\phi_0>$$

where U_i's are unitary matrices, T refers to time steps for the circuit. the language A acts as an oracle: if $x \in \{0,1\}^n$, then $A(x) = 0$ if $x \in A$, 1 if $x \notin A$. Note that if $|A \cap \{0,1\}^n| = 0$, then Q_{A_n}, is the identity. We will compare two cases: no words against exactly one word. Define

$$|\theta_i> := U_i ... U_1 |\phi_0>$$

Our goal is to find $x \in \{0,1\}^n$ such that if $A_n = \{x\}$, then $||\phi_T - \theta_T|| > const. \Rightarrow T$ is huge.

Lemma 4 $\exists x \in \{0,1\}^n$ *such that* $||\phi_T - \theta_T|| \leq \sqrt{\frac{2T^2}{2^n}}$

Before we prove the lemma, we state a corollary:
Claim: If $T = o(2^{n/2})$ then this circuit can't distinguish between $|A \cap \{0,1\}^n| = 0$ or 1.

Now our goal is to find x so that it's not queried in a meaningful way. Assume w.lo.g. that there is a distinguished set of wires that are inputs to the Q_A gate, i.e. assume dedicated input wires for Q_A. For a state

$$|\psi> = \sum_{c \text{ basis state}} \alpha_c |c>$$

we define the query magnitude of x in ψ to be $\sum_{c \in S_x} |\alpha_c|^2$, where

$$S_x = \{ \text{ basis states with x on oracle wires } \}$$

Def. The query magnitude of x at time i is $q_{i,x}$ = the query magnitude of x in θ_i.

Def. The query magnitude of x, $q_x = \sum_{i=1}^{T} q_{i,x}$.

Idea: take x of mimimal query magnitude so $q_x < \frac{T}{2^n}$, because $\sum_x q_{i,x} = 1$ for every i.

Claim: $||\theta_T - \phi_T|| \leq \sqrt{T \cdot 2q_x}$ (this $\Rightarrow$ Lemma 4).

Proof of claim:

Let $V_i = U_i Q_A$.

$$\begin{aligned}
|\phi_T> &= (V_T V_{T-1} ... V_1)|\phi_0> \\
|\theta_T> &= (U_T U_{T-1} ... U_1)|\phi_0> \\
\psi_{k,i} &= V_i V_{i-1} ... V_{k+1} U_k ... U_1 |\phi_0> \text{ (called the k'th hybrid at time i} \\
\text{so } \psi_{0,T} &= V_T ... V_1 |\phi_0 \\
\psi_{k,T} &= V_T ... V_{k+1} U_k ... U_1 |\phi_0> \\
\psi_{T,T} &= U_T ... U_1 |\phi_0>
\end{aligned}$$

Compare $\psi_{k,k+1}$ with $\psi_{k+1,k+1}$.
$\theta_k = \psi_{k,k}$, we apply either V_k or U_k. It is easily verified that

$$||\psi_{k,k+1} - \psi_{k+1,k+1}|| \leq 2q_{k,x}$$

$$\Rightarrow ||\psi_{k,k+1} - \psi_{k+1,k+1}||^2 \leq 2q_{k,x} \text{ since } ||\psi_{k,T} - \psi_{k+1,T}||^2 = ||\psi_{k,k+1} - \psi_{k+1,k+1}||^2$$

because unitary transformations preserve norms. Therefore

$$\sum_{k=0}^{T-1} ||\psi_{k,T} - \psi_{k+1,T}||^2 \leq \sum_{k=0}^{T-1} 2q_{k,x} \leq 2q_x$$

By Cauchy-Schwartz,

$$||\theta_T - \phi_T|| = ||\psi_{0,T} - \psi_{T,T}|| \leq \sum_k ||\psi_{k,T} - \psi_{k+1,T}|| \leq \sqrt{2q_x T}$$

Grover's Algorithm

Search in $O(\sqrt{N})$ time. Let H be the Hadamard matrix

$$M = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Also, let us consider the matrix corresponding to the gate NOT,

$$Q_{NOT_2} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

We shall use the matrices H and NOT to get a unitary transformation that flips the sign of one bit, namely $U = H \cdot NOT \cdot H$.

$$U = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Using this machinery, we can construct a gate Q'_A for an oracle A, which takes $|x > to - |x >$ if $x \in A$, and leaves $|x >$ unchanged otherwise (note this is a unitary transformation, since it is unitary on the orthogonal basis vectors, and takes them to orthogonal vectors).

So if $A = 01$, $Q'_A|01 >= -|01 >$. Suppose we apply this transform to a uniform linear combination of states

$$|x_0 >= \frac{1}{2}(|00 > +|01 > +|10 > +|11 >)$$

We get out

$$\frac{1}{2}(|00 > -|01 > +|10 > +|11 >)$$

Now suppose to the result we apply a Hadamard transform to each bit, then apply the matrix

$$\begin{bmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & -1 \end{bmatrix} \text{ (reflection about mean)}$$

to it, and again the Hadamard transform to it. We get out just $|01 >$!

The Hadamard transform acts as a Fourier transform. The general algorithm is based on these lines, the idea is to iterate, applying these two gates (the diffusion transform and Q'_A), when we have more than 2 inputs.

$$RQ'_A RQ'_A ... |x_0 > (\frac{4}{\pi}\sqrt{2^n} \text{ times})$$

the dominant term will be the word we want to select. This takes time $O(\sqrt{N})$ since $N \approx 2^n$.