CS 410/510 Introduction to Quantum Computing

Lecture 16

Portland State U, Spring 2017 — *May 25, 2017*

Lecturer: Fang Song — *Scribe: Enis K. Inan*

VERSION: OCTOBER 9, 2017

Last time we talked about quantum complexity classes. Recall that we used the quantum circuit model (1) as the basis for doing our quantum computations instead of quantum Turing machines (2). Both models are equivalent, it's just easier to use (1) instead of (2). With (1), we have a collection of quantum circuits $Q = \{Q_x \mid x \in \{0,1\}^*\}$, i.e. Q contains one quantum circuit for each string x. We say that Q is *polynomial-time uniform* or *polynomial-time generated* if there exists a polynomial algorithm that outputs a classical description of Q_x given input x. We then proceeded to state some important complexity classes.

Of interest is BQP, which states that a promise problem $A \in$ BQP if there exists a collection of quantum circuits Q such that for all $Q_x \in Q$ where x is a valid input, $|Q_x| = \text{poly}(|x|)$, i.e. the circuit Q_x is polynomial in the size of the input,

- If $x \in A_y$, then $\langle 1|Q_x|1\rangle \geq 2/3$.
- If $x \in A_n$, then $\langle 1|Q_x|1\rangle \leq 1/3$

where the fancy $\langle 1|Q_x|1\rangle$ is understood as follows. When we run Q_x, we do not start with any zeros. When Q_x finishes its computation, we will end up with a 1-qubit mixed state (ignoring ancilla qubits) that we also denote as Q_x since the circuit Q_x produces it. Then $\langle 1|Q_x|1\rangle$ is the probability that Q_x will measure the outcome 1, i.e. output that input x is a YES instance. Thus, Case 1 is saying that if x is a YES instance, then Q_x should indicate it is so with at least $2/3$ probability. Case 2 states that if x is a NO instance, then Q_x should give a false-positive YES result with *at-most* $1/3$ probability. Note that Cases 1 and 2 are known as the *completeness* and *soundness* conditions, respectively.

Example 1: Integer factorization is in BQP.

- **Input:** (N, k), $2 \leq k \leq N$
- **Output:** Whether there exists a proper factorization of N where each factor $a \in \{2, \ldots, k\}$. Using Shor's algorithm, we can answer this question in $O(\text{poly}(\log(N)))$ time.

Note that P $\subseteq$ BQP and BPP $\subseteq$ BQP, but we do not know if BQP $\subseteq$ BPP, i.e. if BQP $=$ BPP. We know the first two statements are true because we have showed in the beginning of class that we can turn *any* classical circuit into a quantum circuit in a polynomial number of steps with a polynomial change in size.

1 BQP $\in$ EXP

We will now prove that BQP $\in$ EXP. Intuitively, this containment is formalizing the notion that a classical computer can simulate a quantum computer in exponential time. The proof idea is to take a quantum circuit Q_x for an input x that outputs 1 with probability p_1 and 0 with probability p_0 and create a classical algorithm which simulates it, i.e., given an input x, it outputs 1 with probability $\widetilde{p_1}$ and 0 with probability $\widetilde{p_0}$ such that $p_1 \approx \widetilde{p_1}$ and $p_0 \approx \widetilde{p_0}$. Remember that a classical algorithm and a classical circuit are the same thing, so what we really want to do is to create a classical circuit that sufficiently approximates our quantum circuit.

Now we proceed to the proof. First, note that *any* quantum circuit can be approximated by a combination of Toffoli and Hadamard gates. As a brief recap, here's what these gates do:

- H acts on a single qubit, and induces the following mappings:

$$\mathrm{H}|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$\mathrm{H}|1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

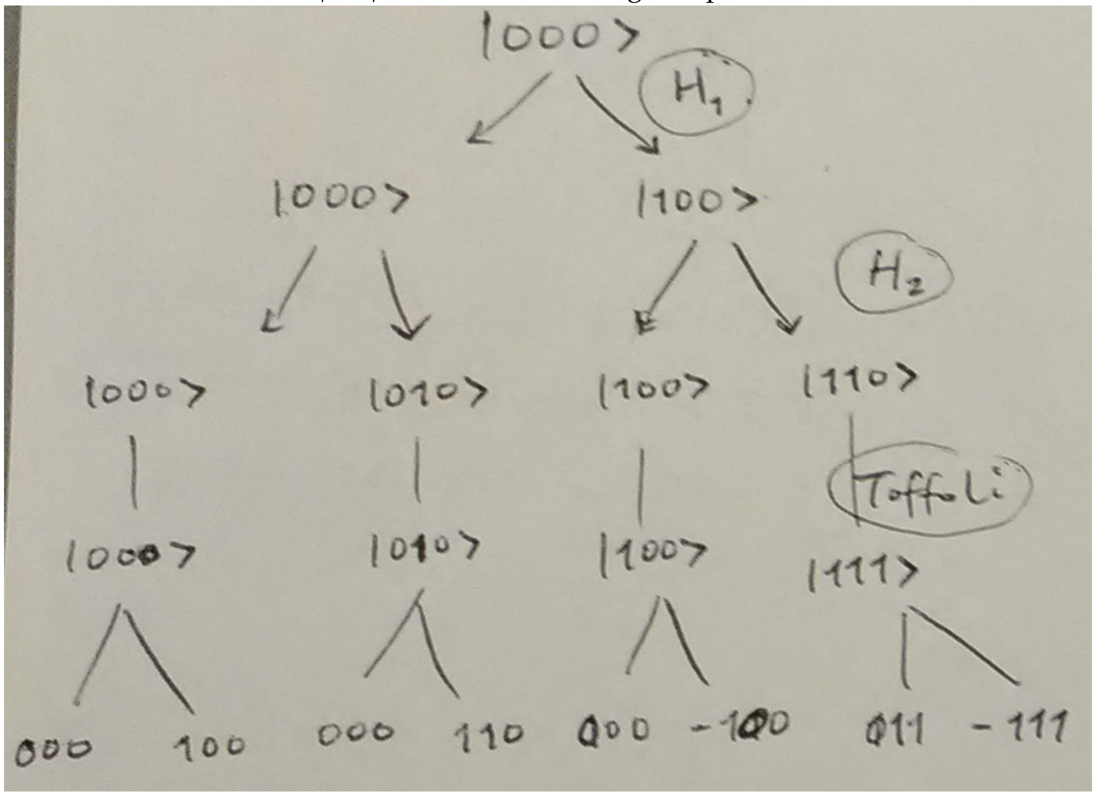
- Tf acts on three qubits. Given a three qubit state $|abc\rangle$, we get the following mapping:

$$\mathrm{Tf}|abc\rangle \to |ab\rangle|(a \wedge b) \oplus c\rangle$$

  where the first two qubits do not change while the last qubit gets flipped if the first two qubits are both 1.

The first thing we do is reconstruct Q_x so that it consists of *only* H and Tf gates that are sequenced one after the other. For example if we want to apply $H^{\otimes 2}$, we break it up into the equivalent sequence $(H \otimes I)(I \otimes H)$. This can be done in polynomial time. From here on out when we reference Q_x, assume that it is the reconstructed version. Now pretend we are a classical computer trying to simulate Q_x starting with an initial input $|\psi_0\rangle$. We go through Q_x one gate at a time. If we encounter an H_x gate, where H_x is applying H to the x^{th} qubit, then we branch off two computation paths: one with the x^{th} qubit set to 0, and the other with the x^{th} qubit set to 1, keeping track of the resulting amplitudes for each. This makes sense upon examining our description of H above, where we see that it turns a classical bit to a superposition – forking off two distinct computational paths *captures* that superposition in the classical world. If we encounter a Tf_{xyz} gate, where Tf_{xyz} is applying Tf to the x^{th}, y^{th} and z^{th} qubits, then we follow our description of Tf above: leave x and y alone but flip z iff $x \wedge y = 1$. With Tf then, we see that we do not need to fork any additional paths as it does not create a new superposition. Thus our simulation of Q_x is really maintaining a *tree* where the nodes represent intermediate stages in the computation. Assume that we have m gates in our circuit. Then each level i of the tree represents the resulting superposition after applying the first i gates, where $0 \leq i \leq m$. The nodes of the tree are the components of the superposition at their corresponding level. Thus for each node, we would like to maintain two pieces of information: the binary representation of that specific component, and its corresponding amplitude. In fact because we are applying only H and Tf gates, every node will have the same amplitude as any other node on that level; this amplitude is $\frac{1}{\sqrt{2^m}}$ where m is the number of nodes on that level. The leaves of our tree will contain the final superposition. Thus, a path from the root to *a* leaf represents *one* possible outcome assuming that we measure all of our qubits. Because we are using Tf and H gates, it is possible to have two *different* computation paths that produce the same final result (i.e. nodes on the same level can have the same superposition component). The exponential time complexity comes from the size of our tree, where we need to compute each superposition component in the internal nodes

Example 2: Say our circuit consists of the gates H_1, H_2, Tf, and H_1 in that order. Assume that we start with an initial state of $|000\rangle$. Then our resulting computation tree will be:



Notice in our tree that we have two possible computation paths that result in 000 as the final outcome. These are 000, 000, 000, 000, 000 and 000, 100, 100, 100, 000.

Assume without any loss of generality that the first qubit in our final superposition contains our desired output, i.e. that $\langle 1|Q_x|1\rangle \approx \widetilde{p_1} = \Pr(\text{First qubit} = 1)$ – the remaining qubits are ancilla qubits. Then, we see that $\widetilde{p_1}$ is just the absolute value squared of the sum of the amplitudes of all the qubit-leaves where the first qubit is 1. Because each path from root to leaf is unique, we can represent our final superposition at the leaves using the individual paths as our orthonormal basis. Let P be the set of computation paths. Then our superposition is expressed as:

$$\sum_{p \in P} \left(\frac{1}{\sqrt{2}}\right)^{|P|} \text{sign}(p)|p\rangle \tag{1}$$

where sign(p) is the signum function that indicates if the final superposition component upon traversing the path p has negative or positive amplitude. Let α_x be the amplitude that represents outcome x where $x \in \{0,1\}^m$, with m being the number of qubits at our leaf nodes. Then we see that our final superposition is also:

$$\sum_{x \in \{0,1\}^m} \alpha_x |x\rangle \tag{2}$$

where

$$\alpha_x = \sum_{p \in P \text{ s.t. } p=x} \left(\frac{1}{\sqrt{2}}\right)^{|P|} \text{sign}(p) \tag{3}$$

where $p = x$ means that the path p leads to the outcome x – remember that multiple paths can have the same outcome, which is why Eqn. 3 is a summation. Thus, we see that we can calculate $\widetilde{p_1}$ as:

$$\widetilde{p_1} = \left| \sum_{x \text{ s.t. } x_1=1} \alpha_x \right|^2 \tag{4}$$

where $x \in \{0,1\}^m$ and $x_1 = 1$ indicates that x's first qubit is 1. Equation 4 is mathematically expressing what we said earlier.

Because new computational paths are only forked when we apply H, our worst-case complexity for the size of our tree is when Q_x has h Hadamard gates. Here $|P| = 2^h$ so that there are 2^h leaves in the tree and hence 2^h possible paths, resulting in an exponential worst-case time. Thus, BQP $\subseteq$ EXP.

In fact if we reuse the space when calculating the amplitudes, it turns out that we only need polynomial space to calculate $\widetilde{p_1}$ so that BQP $\subseteq$ PSPACE. We can adapt the proof even further to show that BQP $\subseteq$ PP.

2 QMA

We will now present the quantum analogue of NP, QMA. We use MA instead of NP because the quantum world is random. Like NP for a classical computer, QMA represents problems that are efficiently verifiable by a quantum computer.

We say that a promise $A \in$ QMA if there exists a collection of quantum circuits Q such that for all $Q_x \in Q$ where x is a valid input, $|Q_x| = \text{poly}(|x|)$,

- *Completeness:* If $x \in A_y$ then there exists a state ρ on poly$(|x|)$ qubits such that

$$\langle 1|Q_x(\rho)|1\rangle \geq \frac{2}{3}$$

  We say that ρ is a "quantum certificate" which verifies that $x \in A_y$.

- *Soundness:* If $x \in A_n$ then for all states ρ on poly$(|x|)$ qubits:

$$\langle 1|Q_x(\rho)|1\rangle \leq \frac{1}{3}$$

We assume that in Q_x, the first qubit yields the answer to our decision problem.

Here are some things we can say about QMA.

1. BQP $\subseteq$ QMA and BPP $\subseteq$ QMA since we do not need a certificate to decide a BQP or BPP instance – we can just directly solve the problem by running x through our circuit Q_x.
2. NP $\subseteq$ MA $\subseteq$ QMA. We know why NP $\subseteq$ MA. MA $\subseteq$ QMA for the same reason as BPP $\subseteq$ BQP – we can create a polynomial-size reversible quantum circuit from any classical circuit, and encode any classical certificate as a corresponding quantum certificate.
3. QMA $\subseteq$ PP

For more information on all the complexity classes out there, see the Complexity Zoo.

3 Interactive Proofs

There is a recurrent pattern in the complexity classes NP, MA and QMA. In each case, we have an all-powerful "prover" (think computing machine) that prepares a certificate c which we pass along to a verifier. c represents a possible "solution" candidate to a problem instance encoded by some input x. A is the general category that these problems fall under, where A_y represents instances that do have a solution and A_n those instances that don't. For example if A is integer factorization, then x is the encoded input (N, k). If x *has* a solution, i.e. if $x \in A_y$, then our verifier will accept any c that *is* a solution – we don't care what it does for those cs that don't correspond to valid solutions. However if our problem does not have a solution, i.e. if $x \in A_n$, then our verifier will reject *all* certificates c – it cannot be fooled by the prover into accepting a solution candidate for an unsolvable problem. Our verifier must makes its decision in polynomial time with respect to the input size, represented by the sizes of x and c. What we have just now (informally) described is the complexity class NP. For MA and QMA, our verifier can make mistakes; the probability that it does so is bounded. In QMA for example, we require our verifier to accept a solution certificate c for $x \in A_y$ with at least $2/3$ probability (1), while ensuring that any certificate c for an $x \in A_n$ is accepted with probability at most $1/3$ (2). The mistake in (1) is rejecting a solution certificate c while in (2), it is accepting any certificate c. In both cases, the mistake probability is bounded above by $1/3$.

We can generalize the interaction between the prover and the verifier by letting the prover send multiple certificates (messages) instead of a single certificate to the verifier, which is still required to make its decision in polynomial time. Here, we say that the prover and the verifier form an *interactive proof system*. The complexity class IP represents those problems A that have an interactive proof system; QIP is similar to IP except that the prover can send quantum messages to the verifier in addition to classical ones. From our preceding discussion, NP and MA are special cases of IP while QMA is a special case of QIP. Several breakthrough results have shown that IP = QIP = PSPACE.

Aside: You might be wondering why we coined the term MA. The answer is from the tale of King Arthur. There, Merlin was an all-powerful wizard while Arthur was an ordinary mortal. In IP, we have an "all-powerful" prover (Merlin) and a "puny" verifier (Arthur) – putting Merlin and Arthur together, we get the name MA.

4 Group Non-membership

We now present the group non-membership (GNM) problem. This is a problem that we know is in QMA but not if it is in MA. Experts think that it probably isn't. Here is our problem statement.

- **Input:** $\langle g_1, \ldots, g_k \rangle = H \leq G$ and $x \in G$
- **Output:** If $x \notin H$, then YES. Otherwise, NO.

where $g_1, \ldots, g_k$ are the generators of some subgroup H of a group G – remember from group theory that we can specify any group G using only its generators. Informally, given a subgroup H and an element x from the subsuming group G, GNM asks if $x \notin H$. We will now prove that GNM $\in$ QMA by constructing our verifier.

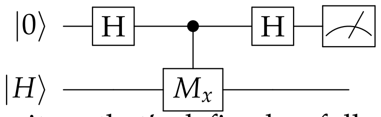
We will represent the subgroup H as a uniform superposition over its group elements, defined as:

$$|H\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle \tag{5}$$

Now consider $|H_x\rangle$ defined below:

$$|H_x\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |h \bullet x\rangle \tag{6}$$

where $H_x = \{h_1 x, h_2 x, \ldots\} = Hx$ is a coset of H in G. If $x \in H$, then we know from group theory that $H_x = H$ implying that $|H_x\rangle = |H\rangle$. Otherwise if $x \notin H_x$, then $H_x \cap H = \emptyset$. In quantum terms, this means that $|H_x\rangle$ is orthogonal to $|H\rangle$, i.e. that $\langle H_x | H \rangle = 0$. This makes sense since here, H_x and H are independent sets and orthogonality captures independence. Now consider the following circuit that will form our verifier:



where M_x is the controlled unitary that's defined as follows:

$$M_x\left(|0\rangle|H\rangle\right) \rightarrow |0\rangle|H\rangle$$
$$M_x\left(|1\rangle|H\rangle\right) \rightarrow |1\rangle|H_x\rangle$$

Let's see what this circuit does starting with the initial state $|0\rangle|H\rangle$. After applying the first Hadamard gate to the first qubit, we get:

$$\frac{1}{\sqrt{2}}(|0\rangle|H\rangle + |1\rangle|H\rangle)$$

Applying M_x, we get:

$$\frac{1}{\sqrt{2}}(|0\rangle|H\rangle + |1\rangle|H_x\rangle)$$

Finally applying the last Hadamard gate to the first qubit, we get:

$$\frac{1}{2}|0\rangle\left(|H\rangle + |H_x\rangle\right) + \frac{1}{2}|1\rangle\left(|H\rangle - |H_x\rangle\right) \tag{7}$$

We now have two cases to consider:

1. $x \in H$ so that $H_x = H$. Here, Eqn. 7 reduces to $|0\rangle|H\rangle$ so that we will measure 0 with certainty.
2. $x \notin H$. Then examining Eqn. 7, we see that there is a 1/2 chance of measuring 0, and a 1/2 chance of measuring 1.

Looking at Cases 1 and 2, we will say that our verifier outputs YES when we measure a 1 on the first qubit, otherwise it outputs NO. Here's why we do this. When $x \in H$ and our prover gives us $|H\rangle$ as a certificate, we know for sure that our circuit will measure 0 from Case 1 so we output NO to correctly reject $|H\rangle$ – the case when the prover gives us a certificate that isn't $|H\rangle$ will be considered shortly.

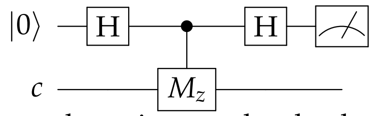
Consider the *completeness* requirement. Here, when $x \notin H$ and we are given $|H\rangle$ as our certificate, we want our verifier to output YES with probability $p \geq 2/3$. Unfortunately from Case 2, $p = 1/2$ so our verifier is not good enough. However, all is not lost. We can tweak our verifier to instead run the circuit at most t times. This is OK since our prover is all powerful, so it can generate as many copies of $|H\rangle$ as we need. This time, our verifier will output YES as soon as it measures a 1. Now, our probability p is the chance that we measure 1 *at least* once within t trials, which we can calculate using a binomial distribution as:

$$p = 1 - \left(\frac{1}{2}\right)^t$$

Setting $t = 2$ will yield $p = 3/4$, which is good enough.

Now we consider the *soundness* requirement, using our tweaked verifier from above. Here we know that $x \in H$. We must show that our verifier will output NO for *any* certificate given to it by the prover with probability *at least* 2/3 (complement of outputting YES with probability at most 1/3). We have two cases to consider here. If the prover gives us $|H\rangle$ as the certificate, then Case 1 from our analysis shows that we will always measure a 0 in all t iterations of running our circuit so that we correctly output NO with certainty. However if the prover gives us a certificate c that isn't $|H\rangle$, we cannot pass c into our verifier as-is because our circuit is only defined for $c = |H\rangle$. But we can construct a new circuit T that does the following. If $c = |H\rangle$, it will run our verifier as described in the completeness proof above. Otherwise, T will output NO. Thus, T becomes our new verifier. The question now is how to test that $c = |H\rangle$. We cannot just check the generators of H because the check is not always guaranteed to complete in polynomial time.

What we can do, however, is use a slightly modified version of the main circuit as a part of our testing procedure. This circuit is below:



where M_z is the same as M_x except that z is a randomly chosen element of H, and we compute the coset cz instead of Hz. From Case 1 above, if $c = |H\rangle$ then we will measure 0 with certainty. Thus when we see a 1, we know that $c \neq |H\rangle$ so that our test can output NO to correctly reject c. To ensure that our test works with the required probability for rejection, we can repeat our testing procedure several times using different, random elements of z – remember that our prover is all-powerful, so it can generate as many copies of c as we need to do this. For more details on the analysis for this case, see these lectures notes.

The above argument establishes that our new verifier exhibits the correct behavior for the soundness case; it is also polynomial in the size of the input. Further, our completeness argument still holds with this new verifier because the testing procedure will always measure 0 when $c = |H\rangle$ so that the probability of outputting YES still depends only on the main circuit.

This completes our proof that GNM $\in$ QMA.