

Lecture 1

*Lecturer: Scott Aaronson**Scribe: Andy Lutomirski*

1 Introduction

1.1 Course structure

The course website is at <http://stellar.mit.edu/S/course/6/fa08/6.896/>.

Prof. Aaronson wants people to participate and show up. There will also be projects which could contain some original research, not necessarily earth-shattering. A literature survey would be fine as well.

There is no textbook. The syllabus recommends a few books on QC. Prof. Aaronson will try to put them on reserve in the library. There are also some great resources online, as mentioned in the syllabus.

The prerequisites are that you should know something about quantum computing *or* about computational complexity theory. The first few lectures may be boring to some, but Prof. Aaronson guarantees that no other lectures will be boring to anyone.

The central object that we will study is BQP (Bounded error Polynomial Time), a class of problems.

We will also study what is possible in QC in the so-called *black-box model*, the bestiary of quantum complexity classes (beyond BQP), and other miscellaneous topics.

1.2 Computability and Complexity Theory

What is complexity theory? It's the field that contains the $P \stackrel{?}{=} NP$ question.

Computability theory was established by Gödel, Turing, and others, who came up with a formal notion of computability, computability by a Turing machine. They showed that many problems are computable but that certain problems are not computable.

The question of complexity followed naturally. The theory of NP-completeness established the central question today: $P \stackrel{?}{=} NP$.

What is an algorithm? It's a set of instructions for doing some computation. Since the 1930s, we've modeled it as instructions for a Turing machine.

The Church-Turing thesis states that—well, people disagree about what it should mean or even what Church and Turing themselves thought it meant. But one version is that everything that's computable in the physical world is computable by a Turing machine. And in particular, all programming languages are essentially equivalent—they can all simulate each other.

The Extended Church-Turing Thesis says that the time it takes to compute something on any one machine is polynomial in the time it takes on any other machine. This includes Turing machines, register machines, etc. Even before quantum computing, there were indications that the Extended Church-Turing Thesis might be on shakier ground than the original Church-Turing Thesis. For example, randomized algorithms are sometimes faster than deterministic algorithms, leading to a class called BPP (Bounded-Error Probabilistic Polynomial-Time). Obviously $P \subseteq BPP$. (Today we believe that $BPP = P$, but we can't prove it.)

NP (Nondeterministic Polynomial-Time) is the class of problems where, if the answer is “yes,” then there exists a proof that’s verifiable in polynomial time (though it might be very hard to find). An alternative definition is that a problem is in NP iff it’s solvable in polynomial time by a Turing machine that can take branches, evaluate all paths simultaneously, and accept iff any of the paths would accept.

Our best understanding of physics corresponds to the class BQP (Bounded-Error Quantum Polynomial-Time). We’re still trying to figure out exactly how it relates to the classical complexity classes.

2 Quantum mechanics

2.1 What is quantum mechanics?

Quantum mechanics is a theory in which several possibilities seem to happen at once. For example, a single photon given a choice between two slits, shows light and dark fringes, which seems to violate classical probability, in which one would assume that, when a second slit is open, the probability that the photon hits somewhere could only increase.

One question is what effect this has on computers and complexity: can we use quantum effects to compute things faster?

There are all kinds of interesting questions about how different concepts in complexity change with quantum mechanics. You can stick the word quantum in front of all kinds of things: communication channels, proofs, advice, pseudorandomness, etc.

We would like to organize our thoughts on these subjects.

2.2 Quantum mechanics as an extension of classical statistics

Many people today are under the bizarre misapprehension that quantum mechanics is hard. Prof. Aaronson blames the physicists. But, once you take the physics out, quantum mechanics is very easy – it’s what you would inevitably get if you tried to generalize classical probability theory by allowing things that behaved like probabilities but could have minus signs.

The “state” of a classical computer could be written as a vector of all possible classical states, in which one element is 1 and the rest are 0. Operations are matrix multiplication. The downside of this representation is that it’s exponentially inefficient, but the upside is that any operation can be seen as just a matrix multiplication.

One example of an elementary operation is controlled-NOT (CNOT)—it maps $00 \rightarrow 00$, $01 \rightarrow 01$, $10 \rightarrow 11$, and $11 \rightarrow 10$. In matrix language, this is

$$\text{CNOT} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

In classical statistics, we change the vector to be a vector of probabilities—each element is nonnegative, and they all sum to 1. The matrices are now stochastic matrices (i.e. they conserve probability), and they can turn a deterministic state into a probabilistic one. (A stochastic matrix is one that is nonnegative, real, and for which each column sums to 1.)

Now we can ask: what happens if we decided to preserve the L_2 norm instead of the L_1 norm? We can also use complex numbers while we’re at it. The picture is now a unit circle in which the

states (axes) might be named $|0\rangle$ and $|1\rangle$. (These are kets, a bizarre yet convenient notation due to Dirac.) At the end of the day, though, we want probabilities for the various possible outcomes ($|0\rangle$ or $|1\rangle$), and those probabilities should sum to one. The obvious choice is to take the squares of the amplitudes as our probabilities.

There are two things we can do to a computer: look at the result or advance the computation. Looking at the computation uses the probabilities above. Advancing the computation multiplies the vector of amplitudes by some norm-preserving matrix—i.e. a unitary matrix. Some examples are rotations and reflections.

As an example, suppose we apply the 45° rotation $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ twice. Classically, we'd expect that the first one randomizes the state and then the second one randomizes it again. But, in fact, it turns into a standard NOT gate. Another way to think of this is that all possible outcomes happen, but that $|0\rangle \rightarrow |0\rangle \rightarrow |1\rangle$ and $|0\rangle \rightarrow |0\rangle \rightarrow |1\rangle$ occur with opposite-signed amplitudes and cancel each other out. This is also what happens in the double-slit experiment.

Conservation of probability requires unitarity (i.e. the columns of the matrix should be orthogonal and have unit norm, or equivalently $A^{-1} = A^T$). This implies invertibility and thus all unitary evolutions are reversible. But looking at a quantum computer is *not* reversible—it collapses the wavefunction (state) into whatever you saw in the measurement, so the same measurement done twice in a row gives the same result.

Lecture 2

*Lecturer: Scott Aaronson**Scribe: Alex Cornejo*

Quick Recap

The central object of study in our class is BQP, which stands for “**B**ounded error, **Q**uantum, **P**olynomial time”. Informally this complexity class represents problems which are *efficiently* solvable with *high* probability by a quantum computer, a precise definition will be given in later lectures.

For our purposes quantum mechanics is just a generalization of probability theory, the table below describes some of the concepts used in classical probability theory and their equivalent in quantum mechanics.

Probability	Quantum Mechanics
\mathbb{R}^+	\mathbb{C}
L_1 preserved	L_2 preserved
Stochastic Matrices	Unitary Matrices

The quantum state of a physical system can be described by a vector in Hilbert Space, for which we will use the bra-ket notation. Given a basis, any vector can be represented by a linear combination of the basis elements, therefore in general we express a quantum state as,

$$\alpha_1 |1\rangle + \dots + \alpha_N |N\rangle$$

Where $|1\rangle, \dots, |N\rangle$ are the basis vectors and $\alpha_1, \dots, \alpha_N$ are complex numbers. Mostly we will work with normalized states, which means that $\sum_{i=1}^N |\alpha_i|^2 = 1$. When dealing with normalized states $|\alpha_i|^2$ represents the probability that the result of a measurement is $|i\rangle$.

Quantum mechanics can be seen as having only two principles, unitary evolution (multiplying the quantum state by a unitary matrix) and measurement in a standard basis, which results in a collapse of the state to whatever outcome you get. A nice analogy to this is baking a souffle, where if you open the oven to see how it's doing you have collapsed it and have to start over.

Consider the following matrix which represents a 45° counter clock wise rotation in the plane

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

Suppose you are in the state $|0\rangle$ and you apply this operation twice in succession and measure, you would get $|1\rangle$. However if you apply it once, then measure, then apply it again you would get $|0\rangle$ half of the time and $|1\rangle$ half of the time.

This is analog to interference in the double slit experiment.

Why measurement alters the state?

There are three schools of thought that provide different answers to this question,

Niels Bohr (Copenhagen interpretation)

In essence this interpretation amounts to a sophisticated way of saying not to ask the question. Science relies on the notion of measurements and observations, so in essence asking in quantum mechanics “what is a measurement?” is equivalent to asking the axioms of euclidean geometry “what is a point?”.

Hugh Everett (Many worlds interpretation)

There is only one process in quantum mechanics, unitary evolution. What we perceive as measurements is just unitary evolution applied to the measuring equipment and the observer. Essentially the universe splits every time a measurement is performed, and one copy sees $|0\rangle$ while the other sees $|1\rangle$.

David Bohm (Non-local hidden variables)

The third answer says that both of the previous answers are unacceptable, so quantum mechanics is somehow incomplete in the sense that there is an additional aspect that we’re missing. Non-local hidden variables is one proposal to fill that gap, but there are others.

For our purposes it doesn’t matter which of these answers is correct and we are free to pick whatever interpretation is more convenient, since at least for the experiments we can currently conceive it leads to the same outcome.

Why does nature use complex numbers?

One possible explanation is that complex numbers are required if we want our unitary transformations to be continuous. As an example consider the face-flip operation represented by the following matrix:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

This matrix maps the state $\alpha|0\rangle + \beta|1\rangle$ to $\alpha|0\rangle - \beta|1\rangle$. But what is in between these two states? To have a continuous unitary transformation between these two states requires the use of complex numbers.

Why the L_2 -norm?

As it turns out only with the L_1 -norm and the L_2 -norm you can get nontrivial norm-preserving linear transformations.

Entanglement

So far we have considered only a system with a single qubit, now we will start talking about two qubits.

$$\alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle$$

To apply an operation to the first qubit only you can tensor product the transformation with the identity on the second qubit, for example:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

If we measure the qubit the probability of measuring $|0\rangle$ is $|\alpha|^2 + |\beta|^2$. Also, assuming we measured zero, the state of the second qubit collapses to

$$\frac{\alpha |0\rangle + \beta |1\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}}$$

Separable states

An important class of multi-qubit states are those which can be factorized into a sequence of tensored single-qubit states. Such states are called separable, for example:

$$(\alpha |0\rangle + \beta |1\rangle)(\gamma |0\rangle + \delta |1\rangle) = \alpha\gamma |00\rangle + \alpha\delta |01\rangle + \beta\gamma |10\rangle + \beta\delta |11\rangle$$

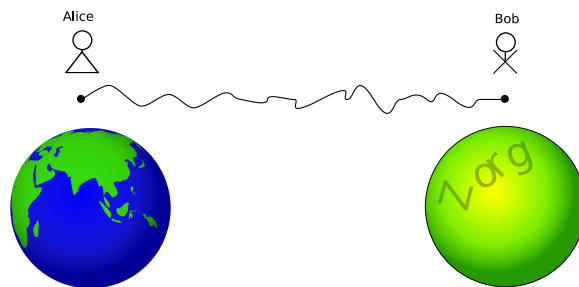
However it is clear that not all states are separable, for example $|00\rangle + |11\rangle$ is not separable. We say such states are entangled.

How is entanglement different from correlation?

EPR paper

In 1935, Einstein, Podolsky and Rosen wrote a famous paper where they brought to widespread attention the tension between quantum mechanics and relativity. One thing that relativity says is that you can't send information faster than light.

However suppose that you have an entangled state like $|00\rangle$ and $|11\rangle$, and suppose one qubit is on Earth and the other is in planet Zorg. If Alice (on earth) measures her qubit and gets a $|0\rangle$, then when Bob (on Zorg) measures his qubit he will also get a $|0\rangle$.



However in this experiment Alice didn't pick what to send. What the EPR paper was trying to ask is if quantum mechanics somehow contradicts relativity. This sets the stage for Bell's theorem.

Bell's inequality

In 1964 Bell played the role of a quantum complexity theorist. He said, let's compare entanglement against classical correlation as resources to perform some task.

The task is the following, Alice and Bob are given random bits a and b respectively, and they will output bits x and y respectively such that

$$a \oplus b = x \wedge y$$

Even with correlated random bits, the best strategy allows them to win at most 75% of the time, for example if they both pick 0. Bell devised a strategy that allows them to win 85% of the time if they share entangled qubits. The strategy is described below:

Alice: If $x = 0$ then measure and output, else apply the matrix $\begin{bmatrix} \cos \frac{\pi}{8} & -\sin \frac{\pi}{8} \\ \sin \frac{\pi}{8} & \cos \frac{\pi}{8} \end{bmatrix}$ measure and output.

Bob: If $y = 0$ then measure and output, else apply the matrix $\begin{bmatrix} \cos \frac{\pi}{8} & \sin \frac{\pi}{8} \\ -\sin \frac{\pi}{8} & \cos \frac{\pi}{8} \end{bmatrix}$ measure and output.

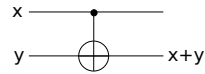
The combined probability of success counting the four possible combinations of a and b is $\cos^2 \frac{\pi}{8} \approx 0.85$.

Notice that this doesn't mean that signals are getting transmitted faster than light, after all you need to bring Alice and Bob's outputs together to even tell if they won or not. However this is a communication task that is possible in a classical universe.

1 Quantum Circuits

Quantum circuits are a nice way to visualize operations with qubits, here we define a couple of useful operations and their notation in quantum circuits. The *Controlled Not* operation $|x, y\rangle \rightarrow |x, x \otimes y\rangle$ is represented by the following matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

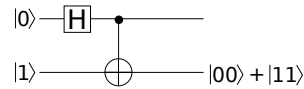


Another popular operation in quantum circuits is the Hadamard operation which intuitively switches between the $|0\rangle, |1\rangle$ basis and the $|0\rangle + |1\rangle, |0\rangle - |1\rangle$ basis, and is represented by the following matrix

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad \boxed{\text{H}}$$

In a quantum circuit we can apply a sequence of unitary operations or “gates”, where each gate can act on the first qubit only (like the Hadamard) or on the second qubit only, or on both qubits together (like the CNOT).

Using the operations described and starting from an unentangled state we create an entangled state with the following circuit,



2 Quantum copying machine

We mentioned before that measuring collapses the state, but what if we could take our quantum state and duplicate it? How would this look like?

$$\alpha |0\rangle + \beta |1\rangle \rightarrow (\alpha |0\rangle + \beta |1\rangle)(\alpha |0\rangle + \beta |1\rangle) = \alpha^2 |00\rangle + \alpha\beta |01\rangle + \alpha\beta |10\rangle + \beta^2 |11\rangle$$

Therefore we just need to find some unitary transformation to perform the above behavior, however the operation required is nonlinear. This is what its called the No-Cloning Theorem.

Lecture 3

*Lecturer: Scott Aaronson**Scribe: Alex Arkhipov*

1. LAST TIME

1.1. **Entanglement.** A state is entangled if it cannot be decomposed into the product of one-qubit states.

1.2. **Bell's Inequality.** Sharing parts of a quantum state lets two players win a game with higher chance than achievable in the classical world (though it still won't let them send messages to each other faster than the speed of light). This shows that quantum physics is a non-local theory in which certain results cannot be explained by particles agreeing in advance.

1.3. **No-Cloning Theorem.** If we could make lots of copies of a quantum state and measure each of them, we could get lots of information about it. Unfortunately, we can't copy unknown quantum states, as a simple argument shows this would violate linearity.

2. MAKING NATURE WORK FOR US

Last time we looked at quantum states with 2 qubits. Today, we jump up to n qubits.

An n qubit state is composed of 2^n state vectors, each with a complex amplitude.

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

This means that to work with 1000 particles, Nature must keep track of 2^{1000} complex numbers! That's more than there are atoms in the universe. As long as Nature is going through this exponential bookkeeping, why not put it to work to solve problems for us?

This is the idea of quantum computing, which was famously explored from two different angles by Richard Feynman and David Deutsch in the 1980's.

2.1. **Feynman.** In his 1982 lecture¹, Simulating Physics with Computers, Richard Feynman noted that having a computer do physics at a molecular level is hard because the work in simulating a quantum system grows exponentially in the number of particles. What if we level the field by simulating quantum physics on a computer made of quantum components? Even more, can we turn the tables and use such a quantum computer to efficiently solve problems that are classically hard?

2.2. **Deutsch.** David Deutsch was out to experimentally test the Many-Worlds Interpretation of quantum physics. Unlike the Bohr view, in which observers are classical and you can't even talk about them being in superposition, The Many-Worlds allows for a person to be in a superposition of possible worlds. Bohr's view raises questions about where the dividing line between quantum and classical line is: Atoms and molecules can be in superpositions, and the double-slit experiment has been done with Buckyballs, but where along the scale to human size is this superposability lost?

It doesn't seem practical to put our brains in superposition, so Deutsch suggested the next best thing: Make an artificially intelligent machine, put it in superposition, then ask it about its experiences.

¹Simulating Physics with Computers <http://www.cs.berkeley.edu/~christos/classics/Feynman.pdf>

3. COMPLEXITY

Deutsch and Feynman had ideas for quantum computing, but as other 80's physicists, they missed the central issue of computation complexity: Is there some function that takes super-polynomial time to compute on a classical computer, but only polynomial time on a quantum computer?

The difficulty is that even if we have an exponential basis vector $\sum \alpha_x |x\rangle$, measuring it just gives a single random state $|x\rangle$ with probability α_x^2 . So if the naive way doesn't work, how do we take advantage of this exponential vector? We exploit interference, choreographing the possible paths so that wrong answers cancel destructively, while correct ones reinforce by adding constructively. This is an intricate process that exploits the structure of the problem, so it's not clear we can do this for interesting problems.

4. COMPUTATION

So what exactly can a quantum computer do? This question was answered in rigorous terms by Bernstein and Vazirani in a 70 page paper² in 1993, who defined a quantum Turing machine which could have the tape head and symbols and superpositions.

Independently, Andy Yao defined quantum circuits, which are computationally equivalent. This is what people use today because they're simpler to think about.

So, in a quantum circuit, we start with N qubits, some of which contain the input to our problem, and the rest of which have been initialized to 0 and will serve as space for work. We apply some unitary transformation to the qubits, then measure them to get the answer.

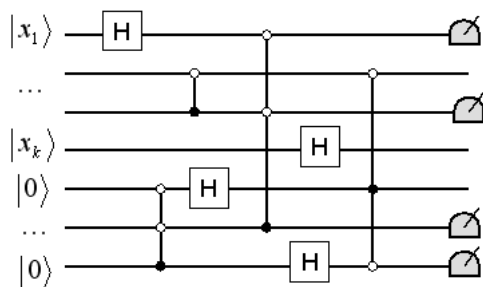


FIGURE 4.1. A quantum circuit

But which unitary transformation can we apply efficiently?

Maybe all of them? Let's return to classical logic circuits and look at a simple counting argument given by Claude Shannon.

How many Boolean functions are there on n bits, $f : \{0,1\}^n \rightarrow \{0,1\}$? The answer, 2^{2^n} is doubly exponential. And how many polynomially sized Boolean circuits are there? Only $2^{\text{poly}(n)}$. So, only a miniscule fraction of Boolean functions are computable by a circuit with a polynomial number of gates.

We want to only use gates that are *local*, meaning that they act on only 1 or 2 bits. We take a set of these gates as a basis, for example, $\{OR, AND, NOT\}$ or just $\{AND, NOT\}$, or even the single gate $\{NAND\}$. Each of these sets is universal, meaning that we can make any Boolean functions using some number of them.

The situation for quantum computers is similar. Physics are local, and interaction requires contact, so we want to work with 1 or 2, or perhaps 3, qubits at once. So, we want to make big unitary transformations as the product of gates which act on a small number of qubits and leave the rest unchanged. Is there some set of gates few-qubit gates that lets compose large transformations,

²Quantum Complexity Theory http://puhep1.princeton.edu/~mcdonald/examples/QM/bernstein_siamjc_26_1411_97.pdf

or at least approximate them? Deutsch, and Bernstein-Vazirani answered *yes* to this question, but the proof is hairy, so we won't go through it.

How many local gates do you need to simulate an arbitrary unitary transformation? Polynomial is definitely not enough, by a degrees-of-freedom counting argument: Each 2^n by 2^n matrix has about 2^{4n} degrees of freedom, while a 2-qubit unitary matrix has 4. So, we need an exponential number of gates. There are other arguments to show that you still need an exponential number to approximate.

For an upper bound, The Solovay-Kitaev Theorem, whose proof is also hairy, says that exponentially many gates are also enough.

Theorem 4.1. *We can approximate an n -qubit unitary transformation to within ε error in L_2 using $O(2^n(n + \text{polylog} \frac{1}{\varepsilon}))$ Hadamard and Toffoli gates.*

The gates they refer are an example of a universal set of gates. Let's look at some examples of these.

CNOT and $\pi/8$.

- CNOT:

$$|x, y\rangle \rightarrow |x, x \oplus y\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



FIGURE 4.2. A CNOT gate

- Rotation by $\frac{\pi}{8}$: $\begin{bmatrix} \cos \frac{\pi}{8} & \sin \frac{\pi}{8} \\ -\sin \frac{\pi}{8} & \cos \frac{\pi}{8} \end{bmatrix}$
- Complex scaling: $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$. The preceding two gates are enough to approximate all real unitary matrices. With this, they can approximate complex ones as well.

CCNOT (Toffoli) and Hadamard.

- CCNOT (Toffoli): Perform a CNOT only if the first bit is 1, thus XOR'ing the AND of the first two bits onto the second.

$$|x, y, z\rangle \rightarrow |x, y, xy \oplus z\rangle$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



FIGURE 4.3. A CCNOT gate

- Hadamard: This gate switches back and forth from the standard $(|0\rangle, |1\rangle)$ basis and the Hadamard basis $\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}, \frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$

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

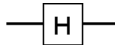


FIGURE 4.4. Hadamard gate

- Complex scaling: $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$

Note that not every combination of small gates is universal. For example, with CNOT and Hadamard, one can only express a special subset of circuits, and this subset can be simulated efficiently by classical computers. This is the Gottesman-Knill Theorem.

5. MEASUREMENT

So we compose some polynomial number of these gates to make a circuit. We put through our inputs and some zeroes, and at the end perform a measurement. Since we're solving a decision problem, we'll just measure one qubit, which will tell us to accept or reject.

Could we gain more power by measuring things in the middle? After all, measurement can change the course of the computation.

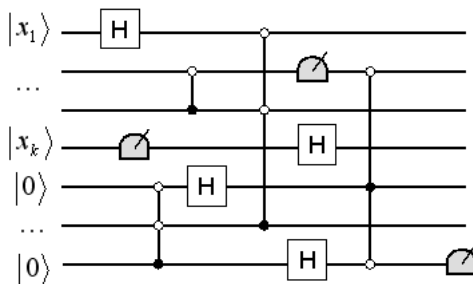


FIGURE 5.1. Do intermediate measurements help?

For example, applying two Hadamards takes $|0\rangle$ to $|0\rangle$, since it's the identity. But, if we put a measurement between them, the measurement collapse to either 0 or 1, making the state into $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ or $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ after the second Hadamard, so we're equally likely to get a 0 or a 1.

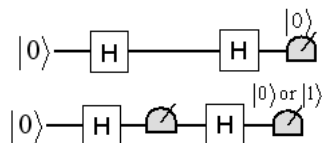


FIGURE 5.2. An intermediate measurement can change the result.

However, it turns out that measuring a qubit can be simulated by applying a CNOT from that qubit to a second qubit. We can call this the second qubit “measuring” the first one, since if we only

look at the first qubit, this is indistinguishable from it actually being measured. So all intermediate measurements can be faked with unitary operations, and we can save actual measurements for the end. This is the Principle of Deferred Measurement.

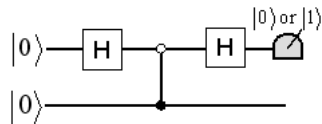


FIGURE 5.3. A CNOT with another qubit acts the same as a measurement

If we trace the evolution of the state through this circuit, we get:

$$|00\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} |0\rangle \rightarrow \frac{|00\rangle + |11\rangle}{\sqrt{2}} \rightarrow \frac{|00\rangle + |01\rangle + |01\rangle - |11\rangle}{2}$$

Measuring the first qubit gives us equal chance of 0 or 1. Note how the effect of CNOT on the first qubit is the same as when we measured it, with the second qubit holding the result of the measurement.

The Principle of Deferred Measurement says something fundamental about the nature of measurement. This is why it's so hard to build a quantum computer - stray particles keep trying to CNOT your qubits.

6. DEFINITION OF BQP

We finally have all the tools to define BQP, the set of problems that can be solved in polynomial time with bounded error by a quantum computer. But first, let's recall the definitions of some classical complexity classes. For simplicity, we look at decision problems.

Definition. **P** is the class of languages $L \subset \{0,1\}^*$ for which there exists a Turing machine M and a polynomial q so that for inputs $x \in \{0,1\}^n$, M terminate in at most $q(n)$ steps and accepts if and only if $x \in L$.

Definition. **PSPACE** is defined like **P**, except we're limited by $q(n)$ space, rather than time.

Definition. **EXP** is defined like **P**, but we're limited by $2^{q(n)}$ time steps.

Definition. **BPP** is the class of languages $L \subset \{0,1\}^*$ for which there exists a Turing machine probabilistic M and a polynomial q so that for inputs $x \in \{0,1\}^n$, M terminate in at most $q(n)$ steps and

- If $x \in L$, then M accepts with probability $> 2/3$
- If $x \notin L$, then M accepts with probability $< 1/3$

The constants $1/3$ and $2/3$ aren't important; we can amplify to make the success probability as high as we want by running the program a bunch of times and taking the majority result. We can pull the randomness out of the definition of BPP by making M take an addition argument r , which can take values in $\{0,1\}^n$.

- If $x \in L$, then $M(x,r)$ accepts for at least $2/3$ the possible values of r
- If $x \notin L$, then $M(x,r)$ accepts for at most $1/3$ the possible values of r .

We can't pull randomness the same way out of BQP as we did for BPP. This is a key difference; randomness is intrinsic to quantum algorithms.

We have some inclusions:

- $P \subset BPP$. Why? Just don't use randomness.
- $BPP \subset PSPACE$. Why? Just run the problem for each possible r and keep a count of how many accepted.

- $PSPACE \subset EXP$. Why? Polynomial space can hold go through exponentially many configurations without looping.

We're ready to define BQP.

Definition. BQP is the class of languages $L \subset \{0, 1\}^*$ for which there exists a *uniform* family of polynomial-size quantum circuits $\{C_n\}$ over some basis of universal gates and a polynomial q so that for all n and inputs $x \in \{0, 1\}^n$.

- If $x \in L$, then $C_n \left(|x\rangle |0\rangle^{\otimes q(n)} \right)$ accepts with probability $> 2/3$
- If $x \notin L$, then $C_n \left(|x\rangle |0\rangle^{\otimes q(n)} \right)$ accepts with probability $< 1/3$

Since circuits have to pre-specify the input size, so we need a circuit for each input size n . By uniform, we mean that there is a classically efficient algorithm to produce C_n given n .

Next time, we'll look at the properties of BQP and figure out where it lies within classical complexity classes.

Lecture 4

Lecturer: Scott Aaronson

Scribe: Beni Yoshida

1 Review of the last lecture

1.1 BQP

BQP is a class of languages $L \subseteq (0,1)^*$, decidable with bounded error probability (say $1/3$) by a uniform family of polynomial-size quantum circuit over some universal family of gate. In today's lecture, we will see where this BQP sits in inclusion diagram of complexity classes.

1.2 Solovay-Kitaev Theorem

With a finite set of gates, we can approximate any n -qubit unitary within L_2 accuracy ϵ using $2^n(n + \text{polylog}(1/\epsilon))$ gates (For example, Hadamard and Toffoli gates). In fact, with CNOT-gate and arbitrary 1-qubit gates, we can apply any n -qubit unitary exactly.

2 Basic properties of BQP

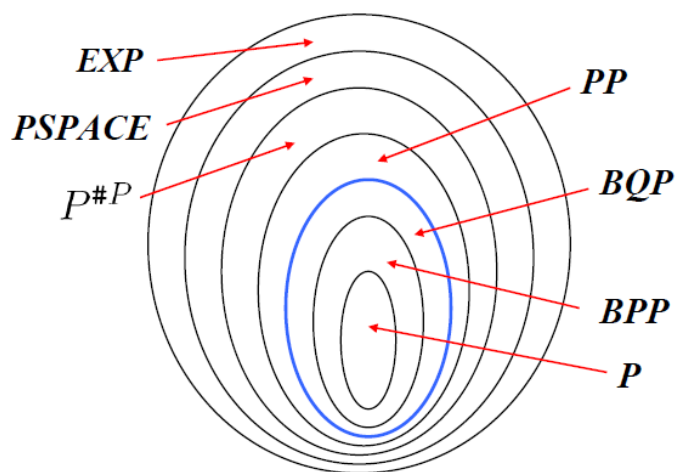


Figure 1: Inclusion diagram of complexity classes.

2.1 $P \subseteq BQP$

We can easily see that quantum circuit can simulate classical circuit.

2.2 $BPP \subseteq BQP$

Quantum computer can solve anything classical probabilistic computer can solve, since quantum property gives us randomness. For example, applying Hadamard gate to $|0\rangle$ gives us a random source of $|0\rangle$ and $|1\rangle$.

2.3 $BPP \subseteq EXP$

Since quantum state is written as $|\psi\rangle = \sum \alpha_x |x\rangle$, we can simulate the whole evolution of all the state vectors with classical computer, within exponential time at most.

2.4 $BQP \subseteq PSPACE$

In terms of computational complexity, the schrodinger picture ($\sum \alpha_x |x\rangle$) and Heisenberg's density matrix (ρ) both lead to an exponential-space simulation since we need to calculate whole evolution of state vectors. On the other hand, the Feynman's path integral, summing up all the histories, leads to a polynomial-space simulation. By writing each final amplitude as a sum of contributions from all the possible paths, we can calculate the sum in $PSPACE$.

For example, the calculation of $H \otimes H|0\rangle$ can be viewed as follows in Feynmann's path integral. We calculate amplitude for each path separately which needs polynomial space only.

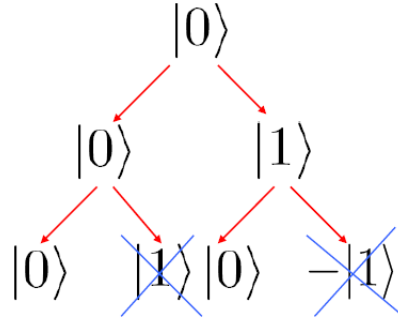


Figure 2: Path integral interpretation of $H \otimes H|0\rangle$. We calculate amplitudes for each four path.

2.5 $BQP \subseteq P^{\#P} \subseteq PSPACE$

$\#P$ is the class of counting problems. To get a class of decision problem, we consider P with $\#P$ oracle, or $P^{\#P}$. Since we can do counting in polynomial space, $P^P \subseteq PSPACE$. Also, $\#P$ can follow all the possible paths non-deterministically in Feynmann's path integral. We can determine that $BQP \subseteq P^{\#P}$.

2.6 $BQP \subseteq PP$

PP stands for probabilistic polynomial time. It is defined as the class of languages L for which there exists a polynomial-time randomized Turing machine M such that for all inputs x :

- if $x \in L$, then $M(x)$ accepts w.p $\geq 1/2$
- if $x \notin L$, then $M(x)$ accepts w.p $< 1/2$.

Note that there is no probability gap, so $1/2$ appears instead of $1/3$ and $2/3$. This class is physical not realistic for we cannot know whether the probability is $1/2$ or $1/2 - 1/2^{|x|}$ without running algorithm exponential time. However, in terms of complexity theory, we can prove that $BQP \subseteq PP$.

PP is the decision version of $\#P$, which means we cannot count the number of accepting paths in the nondeterministic Turing machine, but we can ask whether the number of accepting paths is greater than or less than the number of rejecting paths.

For PP , the threshold is $1/2$, but for BQP , the threshold is $1/3$. However, we can set the threshold which is less than $1/2$ as we like for PP .

At first, we nondeterministically guess x, i, j . Then if $\alpha_{x,i}\alpha_{x,j}^* > 0$, create a number of accepting paths proportional to $|\alpha_{x,i}\alpha_{x,j}^*|$. If $\alpha_{x,i}\alpha_{x,j}^* < 0$, create a number of accepting paths proportional to $|\alpha_{x,i}\alpha_{x,j}^*|$. If $\alpha_{x,i}\alpha_{x,j}^* = 0$, we have the accepting and rejecting paths perfectly balanced each other. Therefore, we know that $BQP \subseteq PP$.

Note that once we get the ability to set the threshold as any number we like, we can determine the exact number by binary search. This fact implies that $P^{\#P} = P^{PP}$.

3 Inclusion diagram

3.1 $BPP \neq BQP$

Can we prove that quantum computer exceeds classical computer? The answer is no since it would imply $P \neq PSPACE$, which is a great challenge as proving $P \neq NP$.

3.2 Where is NP ?

At first, we still don't know where NP sits in the diagram and how NP relates to BQP . We conjecture that $NP \not\subseteq BQP$, which means that quantum computer cannot solve NP complete problems in polynomial time. However, we have no idea as for whether $BQP \not\subseteq NP$ or not.

Another interesting question is that if $P = NP$, then $P = BQP$. Also, if $P = PP$, then $P = BQP$.

4 Structural properties of BQP

4.1 $BQP^{BQP} = BQP$?

In classical computer science, we assume that when we write some algorithm, then we can use it as a subroutine in other algorithm. Does the same thing exist in quantum computer?

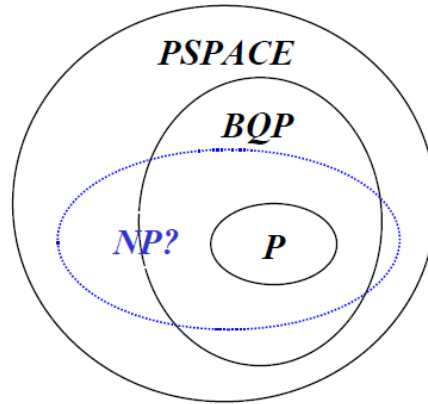


Figure 3: *NP* and *BQP*.

For initial input $|0\rangle$, we get $|work(0)\rangle|output(0)\rangle$. For initial input $|1\rangle$, we get $|work(1)\rangle|output(1)\rangle$. Here, $|work(i)\rangle$ represents subroutine and $|output(i)\rangle$ represent the answer to measure. Usually, we throw away unnecessary qubits other than outputs when we proceed to further calculations.

However, if our input state is $|0\rangle+|1\rangle$, then we will have $|work(0)\rangle|output(0)\rangle+|work(1)\rangle|output(1)\rangle$. This state is an entangled state over work space and output space. Naturally, the states in subroutine space(or work space) affect the result of further operation on output space.

4.2 Uncomputing

This smart trick was introduced by Charlie Bennett. At first, we run the subroutine (unitary operation) and get the answer. Then we apply CNOT-gate to the answer and keep it in some safe location that won't be touched again. Then we run the entire subroutine backwards to erase everything but the answer.

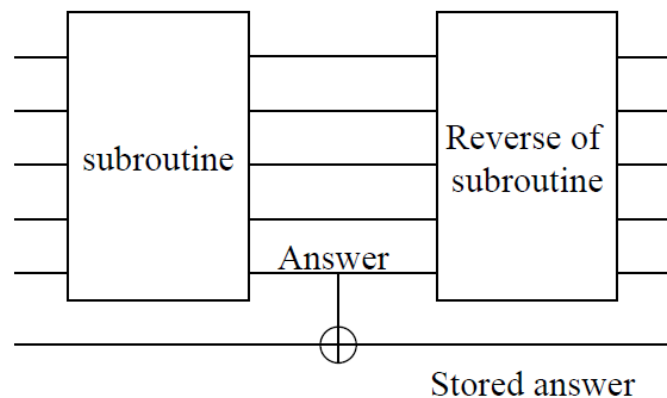


Figure 4: The idea of uncomputing.

The uncomputing step will partly erase the unnecessary residues from the subroutine space, but not completely erase them. However, we can deal with it by amplifying the subroutine part so that error becomes exponentially small. For example, by taking majority vote after many parallel subroutines, we can decrease the error exponentially. The process of taking majority vote can be done in polynomial time, so this amplification process can cope with the error from this uncomputation.

In summary, we know that $BQP^{BQP} = BQP$, in other words, BQP with a BQP oracle is no more powerful than ordinary BQP .

Lecture 5

*Lecturer: Scott Aaronson**Scribe: Andy Drucker*

Last time we looked at what's known about quantum computation as it relates to classical complexity classes. Today we talk somewhat more 'concretely' about the prospects for quantum computing.

1 Recap

1.1 BQP and the Rest

We've informally shown the following complexity class inclusions: $P \subseteq BPP \subseteq BQP \subseteq PP \subseteq P^{\#P} \subseteq PSPACE \subseteq EXP$. We know $P \neq EXP$, so at least one of the inclusions in the chain must be strict (and we suspect that all except the first one are strict), but we can't even show $P \neq PSPACE$. This shows that separating BPP from BQP is at least as hard as the older, notorious open problem of P vs $PSPACE$.

1.2 Operations on Subsystems, Tensor Products

We've identified quantum transformations with unitary matrices acting on states given as vectors of complex amplitudes. As in classical circuit complexity, it is natural to restrict ourselves to applying 'local' transformations on our data. In the quantum case, we realize this as follows. Suppose we conceptualize our quantum state as having two 'registers' each taking multiple possible values (the partition we use may change from gate to gate, as in the circuit setting, depending which parts of the system we are operating upon). We write a quantum state as

$$|\psi\rangle = \sum_{i \leq m, j \leq n} \alpha_{i,j} |i\rangle |j\rangle,$$

where the first register takes on values identified in a manner of our choosing with $\{1, \dots, m\}$ and the second takes values in $\{1, \dots, n\}$.

A unitary transformation T on the system is considered 'local' to the first register if there exists some m -by- m matrix $U = (u_{i,k})_{i,k \leq m}$ such that T 's action on basis vectors can be summarized as

$$|i\rangle |j\rangle \rightarrow \left(\sum_{k \leq m} u_{k,i} |k\rangle \right) |j\rangle,$$

for all $i \leq m, j \leq n$. We note that for T to be unitary, U must itself be unitary.

It can be verified that if we write the amplitudes of a state vector $|\psi\rangle$ in a column in the order $\alpha^T = (\alpha_{1,1}, \alpha_{2,1}, \dots, \alpha_{m,1}, \alpha_{1,2}, \dots, \alpha_{m,2}, \dots, \alpha_{1,n}, \dots, \alpha_{m,n})$, then the resulting state after the transformation T is given by $(U \otimes I_n) \cdot \alpha$. Here the tensor product of an m -by- m matrix $A = (a_{i,j})$, with an n -by- n matrix B , is an mn -by- mn matrix, defined in block form by

$$(A \otimes B) = (a_{i,k} B)_{i,k \leq m}.$$

Similarly, if T is local to the 2nd register, summarized by an n -by- n unitary V , the action of T on the global state α is to produce $(I_m \otimes V)\alpha$.

This representation allows us to reason about quantum computations with tools from linear algebra. For instance, it allows us to show that operations applied to separate subsystems commute; see Pset 1.

2 Prospects for Quantum Algorithmics

2.1 Subroutines

In classical programming, we're used to the idea of designing a small algorithm to solve a simple problem, then calling this algorithm repeatedly to help solve a bigger problem. In our analyses we like to rely only on the fact that this 'subroutine' gives the correct answer, and think of it otherwise as a 'black box'. This is called *modularity*. The fact that this idea works correctly, without an unacceptable increase in running time, is expressed in the oracle result $P^P = P$.

It is less obvious that a similar idea works in the world of quantum circuits, but it does. A trick called 'uncomputing', described in the previous lecture, allows us to call subroutines as we'd like. This yields the complexity result $BQP^{BQP} = BQP$, which gives us confidence that BQP is a 'robust' class of languages, and a world in which we can design algorithms in modular fashion.

2.2 Controlling Error

In the early days of quantum computing (the '80s, say), skeptics said quantum computation was 'just' another form of analog computation (a machine manipulating real-valued quantities), and subject to the same limitation: perturbative noise that could erase information and throw off a computation by successively accumulating errors.

This criticism ignored or glossed over an important insight (explained in Bernstein-Vazirani '93): unlike other types of transformations used in classical analog computers, quantum operations *cannot* amplify error except by directly introducing more of it! To see what is meant, suppose the 'true' input to a unitary U in a computation is the state $|\psi\rangle$, but noise has intruded so that we have instead some $|\psi'\rangle$ at an l_2 distance at most ϵ away from $|\psi\rangle$. Then, using linearity and unitarity, $\|U|\psi'\rangle - U|\psi\rangle\|_2 = \|U(|\psi'\rangle - |\psi\rangle)\|_2 = \||\psi'\rangle - |\psi\rangle\|_2 \leq \epsilon$. Thus if all unitaries in the computation perform as expected, an ϵ error in the initial prepared state will yield an output state ϵ away from correct. This implies that the acceptance probability will be close to what it should've been.

Similarly, suppose that a unitary operation performs not exactly to our specifications, but is off by a 'small' amount. Let's model this as follows: we want to apply U , but instead apply $(U + \epsilon V)$, where V is a matrix with induced norm at most 1, i.e. $\|Vx\|_2 \leq \|x\|_2$ for all x . In this case, inputting some value $|\psi\rangle$ yields $(U + \epsilon V)|\psi\rangle = U|\psi\rangle + \epsilon V|\psi\rangle$, a state whose l_2 distance from our desired state is at most $\|\epsilon V|\psi\rangle\|_2 \leq \epsilon\|V\|_2\| |\psi\rangle \| = \epsilon$.

Applying this idea repeatedly to an entire computation, Bernstein and Vazirani observe that the total error is (in an appropriate sense) at most the 'sum' of the errors in all preparation and gate components. Thus, if engineers can produce circuit elements with error $\frac{1}{t}$, we expect to be able to faithfully execute computations on quantum circuits with size on the order of t .

This idea was improved upon substantially. In a sequence of papers, work by Aharonov, Ben-Or, Knill, Laflange, Zurek and others culminated in the 1996 'Threshold Theorem', which showed how

to use ‘hierarchical coding’ ideas to build quantum circuits for arbitrary *BQP* computations, which would remain reliable even if each gate was subjected to a sufficiently small (but *constant*) rate of error. This result was analogous to (but more complicated than) earlier work by Von Neumann on fault-tolerant classical computation.

There are certain requirements for the Threshold Theorem to be valid. These are: a constant supply of ‘fresh’ qubits; ‘parallel processing’; and an extremely low error rate (on the order of 10^5 to .03, depending on your model). Razborov and others have shown limits to fault-tolerant quantum computation, showing that for particular circuit models there exist absolute constants of allowed error beyond which computational power appears to decrease. In Razborov’s model, at an error rate .5 (since improved by others), quantum circuits provably degenerate to the computational strength of bounded-depth quantum circuits, the class *BQNC*.

However, (a) the true computational power of circuits facing so much error may be much, much lower, and (b) *BQNC* already can perform most if not all of the quantum algorithms we now know of.

It was asked how the ideas of the Threshold Theorem can seem to suppress error in a computation, when we know unitaries are reversible and cannot erase error. This apparent inconsistency is resolved as follows: the Theorem’s constructions do not suppress ‘error’ in an absolute sense by returning very nearly the state that would result from an error-free circuit; it is only the ‘logical structure’ of the computation that is preserved. In particular, the final acceptance probability is near what we’d expect, although the distribution on registers other than the final measured one may be wildly different from the error-free case.

3 Introducing Quantum Query Complexity

3.1 The Query Model

We have already discussed how proving complexity lower bounds on quantum computation seems very difficult. Just as with classical complexity, then, we turn to simplified, idealized models of computation in the hopes of proving something and gaining intuitions. We try to model, in the quantum world, the classical idea of a ‘bounded-query’ algorithm, one which has access to a very large (binary) input and tries to compute some predicate of the input without looking at too many input bits. We think of this input as a function $f(x) : \{0, 1\}^n \rightarrow \{0, 1\}$.

In this model, our algorithm maintains a quantum state over some unrestricted number of qubits. We allow the algorithm to apply *arbitrary* unitary transformations to its own state, as long as these are defined without reference to the values of f . We also allow the algorithm to perform a unitary update whose action on a basis state $|x\rangle$ is specified with reference to a single output value of f , possibly depending on x .

These ‘query’ unitaries are frequently restricted to be one of two types. The first type (a ‘controlled-not query’) maps basis state $|x, w\rangle$ to $|x, w \oplus f(x)\rangle$. Here x is the main register and w an auxiliary or ‘ancilla’ qubit.

The second type of query unitary frequently considered (the ‘textitphase query’) maps $|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$. Note how in either case we take care to make the transformation reversible, and we check that the action is unitary.

It turns out that each of the two types above can simulate the other type in a single query (as long as ancillas are allowed). Thus we have a reasonably robust notion of a quantum query. We

are interested in the *number* of quantum queries required for various computations.

3.2 Example: The Deutsch-Jozsa Algorithm

Now we look at our first example of a quantum query algorithm, one which gives us a taste of how quantum parallelism can lead to faster algorithms than in the classical world. Suppose we are given a function $f : \{0, 1\} \rightarrow \{0, 1\}$, and wish to compute $f(0) \oplus f(1)$. In the classical world, of course, it would take us two queries to f to determine this value. We now show how to find it in only *one* quantum query.

Use a single-qubit register, initialized to $|0\rangle$. Apply a Hadamard operation, yielding $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. (We don't charge for this, since the update rule doesn't refer to f .)

Next apply a phase query to our superposition, yielding (by linearity) the state

$$|\psi'\rangle = \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}.$$

Note that if $f(0) = f(1)$ (equivalent to $f(0) \oplus f(1) = 0$), we have $|\psi'\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ (at least, modulo an irrelevant phase shift ± 1), while if $f(0) \neq f(1)$ we have $|\psi'\rangle = (\pm 1) \cdot \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

Now apply another Hadamard. In the first case we get $\pm|0\rangle$, in the second case $\pm|1\rangle$. Thus a final, standard measurement to the register yields 0 with certainty if $f(0) \oplus f(1) = 0$, and 1 with certainty otherwise. This is the Deutsch-Jozsa algorithm, which yields a factor-2 speedup in computing the XOR of n bits. In future lectures, we will see quantum query algorithms for other problems which achieve much more dramatic speedups over classical query algorithms.

Lecture 6

*Lecturer: Scott Aaronson**Scribe: Megumi Ando***Office Hours:**

- Yinmeng – TBA (See website: <http://stellar.mit.edu/S/course/6/fa08/6.896/>)
- Scott – Mondays, 1:00 pm - 3:00 pm

Last Time:

- Quantum Error-Correction
- Quantum Query Model
- Deutsch-Jozsa Algorithm (Computes $x \oplus y$ in one query.)

Today:

- Bernstein-Vazirani Algorithm
- Simon's Algorithm
- Shor's Algorithm¹
- Hidden Subgroup Framework²

1 Recap of Last Lecture

1.1 Error Propagation

Recall from the previous lecture that error propagates differently in quantum computing compared with in classical computing. In particular, the (1990's) Threshold Theorem states that quantum computing is robust to small errors since it is a linear theory. This implies that small errors are not magnified over the course of the computation and can be corrected using sophisticated hierarchical error-correcting codes. This is unlike classical computing where noise amplification was a major consideration in designing digital systems.

1.2 The Query Model

In quantum complexity theory, we are interested in determining which problems are efficiently solvable by quantum algorithms and which ones are not. This is a difficult problem to answer since we don't know to quantify the number of required computational steps. We don't even know how to prove that the problems we care about do not require an exponential number of steps.

¹We only introduced this.

²We didn't get to this.

The Query Model allows us to make some analysis of the computational complexity of quantum algorithms. In this model, the complexity of an algorithm is measured as the number of queries we make to some query box. So, what do we mean by a query? In classical computing, the idea is fairly self-explanatory. For example, to find the majority of three input bits x_1 , x_2 , and x_3 , we may query for the values for x_1 and x_3 . If we find that both bits are one, then we will need a total of two queries.

In quantum computing, the queries (like all computational steps) must be reversible and, therefore, unitary. In the previous lecture, we gave two equivalent models of quantum queries. In the first model, the function $f(\cdot)$ that we want to query becomes xor-ed to some answer bit b . In the second model, $f(\cdot)$ gets written to a phase; we flip the amplitude if the answer is one and, otherwise, we don't. Written out, the two definitions of queries are:

$$|x\rangle|b\rangle \longrightarrow |x\rangle|b \oplus f(x)\rangle \quad (1)$$

$$|x\rangle \longrightarrow (-1)^{f(x) \cdot b} |x\rangle \quad (2)$$

Last time, we talked about how Equation (1) can simulate Equation (2). We asserted that Equation (2) can also simulate Equation (1).

1.3 Deutsch-Jozsa Algorithm

The Deutsch-Jozsa algorithm is our first example where quantum computing is more efficient than classical computing. The algorithm computes the exclusive-or of two bits in one query rather than two. (In classical computing, we need to query both bits to find the exclusive-or.) We can extend this algorithm to exclusive-or n -bits using only $n/2$ queries. So, this is a factor of two speed-up.

The algorithm is the following. We query in superposition and then apply a Hadamard.

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \longrightarrow \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}$$

If $f(0)$ is equal to $f(1)$, then the Hadamard is going to map above state to $\pm|0\rangle$. Otherwise, if $f(0) \neq f(1)$, then the Hadamard is going to map to $\pm|1\rangle$.

$$\longrightarrow \pm|0\rangle, \text{ if } f(0) = f(1)$$

$$\longrightarrow \pm|1\rangle, \text{ if } f(0) \neq f(1)$$

2 Bernstein-Vazirani Algorithm [93]

The Deutsch-Jozsa algorithm can be generalized to the problem of finding a hidden linear structure. Suppose that there is a boolean function, $f : \{0, 1\}^n \longrightarrow \{0, 1\}$, which maps n -bit strings to a single bit. In addition, we have query access to the function. Like in the Deutsch-Jozsa algorithm, we can query any n -bit string x for $f(x)$; and we can also query in superposition.

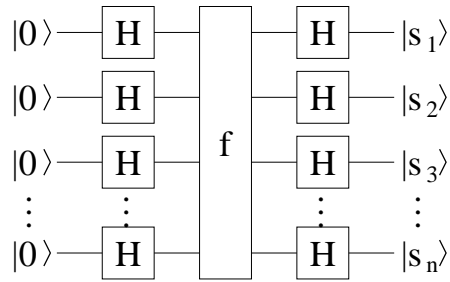
Let us consider the case where $f(x)$ is of the form $s \cdot x \bmod 2$, where s is a fixed and secret string. (Here “ \cdot ” denotes the inner product, i.e. $s \cdot x = s_1x_1 + \dots + s_nx_n \bmod 2$.) In other words, we are promised that there exists a “secret string” s such that $f(x) = s \cdot x \bmod 2$ for all x . The problem is to find s .

First, let us examine the query complexity in the classical world. How many queries do we need to solve this problem classically? n queries are sufficient, because we can query the basis strings. For example, for $n = 5$, we can query the basis strings to find s_1, \dots, s_5 as follows:

$$\begin{aligned} f(10000) &= s_1 \\ f(01000) &= s_2 \\ f(00100) &= s_3 \\ f(00010) &= s_4 \\ f(00001) &= s_5 \end{aligned}$$

In addition, n is also the lower bound. We can prove that we need at least n queries using a basic information theoretic argument. There are 2^n possibilities for s , and each query can only cut the space in half.

In contrast, the Bernstein-Vazarani algorithm solves the problem using only one query. The quantum circuit diagram of the Bernstein-Vazarani algorithm is given below:



Mathematically, this is equivalent to:

$$\begin{aligned} |0\rangle^{\otimes n} &\longrightarrow \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \\ &\longrightarrow \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x} |x\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s_1 x_1} \dots (-1)^{s_n x_n} |x\rangle \end{aligned}$$

After querying the box in superposition, the the secret string s written in the Hadamard basis. So in order to read s , we convert back to standard basis. This is what is accomplished by applying a second Hadamard in the end. Notice that although we needed a linear number of gates, we only have to make one query. This is an n -to-1 speed-up.

3 Simon's Algorithm [93]

Using the Bernstein-Vazarani algorithm, we have demonstrated that we need only one query in the quantum world where we needed at n in the classical world. While this is an interesting result, what we really wish for is an exponential-to-polynomial speed-up. What we want to ask is, "Can we use the quantum model to attack the Beast that is Exponentially?" This questions leads us to Simon's algorithm.³

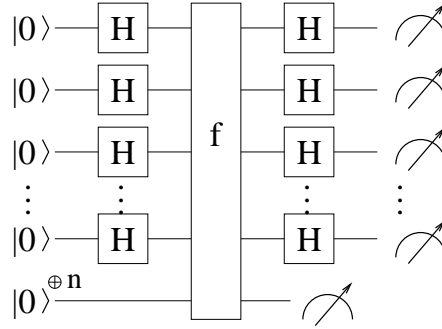
As in the Bernstein-Vazarani algorithm, we are again given a query box that computes some function $f(x)$. In this case, the function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ maps n -bits to n -bits. As before, there is a promise associated with the function $f(x)$. This time, the promise is that there exists a secret string s , where s is not the zero-vector, such that $f(x) = f(y)$ if and only if $x = y \oplus s$. The problem is to find s .

To illustrate what we mean, the secret string s is 110 in the following example:

$$\begin{aligned} f(000) &= 5, & f(100) &= 17 \\ f(001) &= 4, & f(101) &= 42 \\ f(010) &= 17, & f(110) &= 5 \\ f(011) &= 42, & f(111) &= 4 \end{aligned}$$

First, let us examine the query complexity in the classical world. How many queries do we need to solve this problem classically? In classical world, we need on the order of $2^{n/2}$ bits. (To prove that we need $O(2^{n/2})$ queries, pick s randomly and use Yao's Minmax Principle.) By the Birthday Paradox, even with randomness, we still need $\sqrt{2^{n/2}}$ bits.

In quantum computing, we can use Simon's algorithm do this with linear number of circuits, where each circuit is given by:



We apply a Hadamard to the first register, and then query both registers. Then, we measure the second register. After we make the measurement, what is left in the first register is:

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle \rightarrow \frac{|x\rangle + |y\rangle}{\sqrt{2}}, \text{ where } x \oplus y = s.$$

³Simon's original motivation was to try to prove that the quantum model does not provide any exponential speed-up. He failed in his attempt, and what he came up with in the end was a counter-example, which became Simon's algorithm. Shor read this paper and used the general idea to factor integers. So, as we will see later on, Shor's algorithm is Simon's algorithm plus some number theory.

To extract information about s , we apply the Hadamard again and see what's left over from the mess.

$$\begin{aligned} \frac{H^{\oplus n}|x\rangle + H^{\oplus n}|y\rangle}{\sqrt{2}} &\longrightarrow \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle + \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{y \cdot z} |z\rangle \right) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{z \in \{0,1\}^n} [(-1)^{x \cdot z} + (-1)^{y \cdot z}] |z\rangle \end{aligned}$$

We can't directly extract s from here, but we can extract some useful information about s . $(-1)^{x \cdot z} = (-1)^{y \cdot z}$ or else the amplitudes cancel out. This leads us to a linear equation for s .

$$\begin{aligned} x \cdot z &= y \cdot z \\ x \cdot z &= (x \oplus s) \cdot z \\ x \cdot z &= x \cdot z \oplus s \cdot z \\ s \cdot z &= 0 \end{aligned}$$

We didn't learn s , but we did learn that s satisfies the random linear equation $s \cdot z = 0$. We can solve for s using Gaussian Elimination on n linearly independent equations. We only need to run Simon's circuit $O(n)$ times to get n linearly independent equations, so Simon's algorithm runs in $O(n)$ queries. This is an exponential-to-polynomial speed-up!

4 $BPP^A \neq BQP^A$

At the end of the day, we didn't prove that $BPP \neq BQP$. However, we did make some progress. Namely, that via some relativization method, we can prove that there exists some A , such that $BPP^A \neq BQP^A$.

5 Shor's Algorithm [94]

Where Simon's algorithm worked in Z_2^n , Shor's algorithm works in Z_N . So, $f : [N] \rightarrow [N]$. The promise is that there exists some r , such that $f(x) = f(x + r) = f(x + 2r) = \dots$. We'll see more on Shor's algorithm and on Hidden Subgroup next lecture!

Lecture 7

*Lecturer: Scott Aaronson**Scribe: Elena Grigorescu*

1 Short review and plan for this lecture

In previous lectures we started building up some intuition into the way quantum algorithms work, and we have seen examples (Bernstein-Vazirani, Simon) where, excitingly, quantum computation could be performed with less resources (queries/time) than in the classical setting. Simon's algorithm lead to an oracle separation result between BPP and BQP, namely that there exists a language A , such that $BPP^A \neq BQP^A$. While these types of statements are deemed fundamentally interesting by the complexity theory community, they are not going to attract the attention of the NSA. That is why some turn to looking into ways of breaking the RSA cryptographic system instead! If we had a quantum computer, Shor's factoring algorithm would give us a method to efficiently steal people's credit card numbers. In today's lecture we will ready ourselves for this eventual opportunity and broadly discuss the main features of Shor's factoring algorithm. Subsequently, we will place the core ideas of both Simon's and Shor's algorithms into a more general framework, namely the Hidden Subgroup Problem(HSP).

2 General overview of Shor's factoring algorithm

Imagine that you would like to decompose a really large number N into its really large prime factors, in very little time. While we do not know how to do that classically, Peter Shor discovered that the task is possible in the quantum world.

Theorem 1 (Shor '94) $\text{FACTORING} \in BQP$.

A fundamental misconception in solving the factoring question is that quantum computers can try in parallel all possible integers. Unfortunately, that is not the case, and we need to delve more into the specific structure of this type of problems. Shor's algorithm has two modular components: a classical part and a quantum subroutine. The classical part draws from Miller's insight from the '70s that factoring reduces to finding the period of a function, which is then achieved using Quantum Fourier Transforms.

To begin with, let $N = p \cdot q$, $G = \{x \bmod N \mid \gcd(N, x) = 1\}$ and denote by $\text{ord}(x)$ the order of x in G , i.e. the smallest integer r s.t. $x^r = 1 \bmod N$. G is a group under the operation of multiplication and contains $\phi(N) = (p-1)(q-1)$ elements.

We state without a proof the main lemma of the reduction.

Lemma 2 *With constant probability, a uniformly random element x of G has the property that $\text{ord}(x) = 2r$, for some integer $r \geq 1$, and both $\gcd(N, x^r + 1)$ and $\gcd(N, x^r - 1)$ are nontrivial factors of N .*

We now proceed with the order finding subroutine, and rephrase it first as a period finding problem. For comparison, recall that Simon's functions had the property that $f(x) = f(y)$ iff $x \oplus s = y$ for some hidden s to be computed. Similarly, in Shor's case, let $f(r) = x^r \bmod N$, which implies $f(r_1) = f(r_2)$ iff $r_1 = r_2 + s$, where s is the function's period and thus it satisfies $x^s = 1$. The above Lemma 2 states that if we knew the order of some element x (of even order) we could reveal some factors of N .

Period finding quantum algorithm

1. Let $Q = 2^q \approx N^2$. First perform our favorite steps in a quantum algorithm (initialization, quantum superposition) which result into the following state

$$\frac{1}{\sqrt{Q}} \sum_{r=1}^{Q-1} |r\rangle |x^r \bmod N\rangle.$$

Note that computing $x^r \bmod N$ can be done efficiently by repeated squaring.

2. Measure the second register and obtain a global state

$$\frac{1}{\sqrt{l}} \sum_{i=0}^l |r_0 + i s\rangle |f(r_0)\rangle,$$

where $l = \lfloor \frac{Q-r_0-1}{s} \rfloor$. If we now made the mistake of measuring the first register we would end up with an irrelevant random state. Instead, Shor performs the following trick:

3. Apply a Quantum Fourier Transform to the input register. A QFT is a unitary transformation that maps a state $|r\rangle$ into state $\frac{1}{\sqrt{Q}} \sum_{i=0}^{Q-1} \omega^{r i} |i\rangle$, where $\omega^Q = 1$, i.e. $\omega = e^{2\pi j/Q}$. This operation leads to a state

$$\frac{1}{\sqrt{Q}} \frac{1}{\sqrt{l}} \sum_{i=1}^l \sum_{r_1=0}^{Q-1} \omega^{(r_0+is) r_1} |r_1\rangle |f(r_0)\rangle.$$

Fortunately, QFTs can be implemented quantumly by circuits of size $O(\log^2 N)$ using Hadamard gates and controlled phase shift gates, which we will not detail in this lecture.

4. Measure now the first register and observe state $|r_1\rangle |f(r_0)\rangle$, with probability (ignoring the normalization factors) essentially

$$\left| \sum_{i=1}^l \omega^{(r_0+is) r_1} \right|^2 = \left| \omega^{r_0 r_1} \sum_{i=1}^l (\omega^{s r_1})^i \right|^2.$$

This brings us to a pleasant state of affairs, since most of the states have very low amplitude and are most probably not being observed. Indeed, analytic considerations show that the quantity $\left| \sum_{i=1}^l (\omega^{s r_1})^i \right|^2$ is either very large, when $\omega^{r_1 s} \approx 1$, or very small otherwise. The

intuition is that, if the complex vector ω^{sr_1} forms a large angle with the real axis, then summing up over its periodic rotations cancels out the amplitudes, while if that angle is very small the amplitudes add up. In conclusion, if one can observe state $|r_1\rangle|f(r_0)\rangle$ it must be the case that $\omega^{r_1 s} \approx 1 = \omega^Q$, which means that one can estimate a multiple of the period s by $\frac{Q}{r_1}$. Sampling a couple of more times and taking the gcd of the multiples obtained reveal the value of s .

3 The Hidden Subgroup Problem

Simon's and Shor's algorithms are prominent illustrations of a general framework, the Hidden Subgroup Problem, where one is given a black box computing a very structured function and wants to determine its 'generalized period'. More formally, let G be a group, H a subgroup of G , and consider the oracle function $f : G \rightarrow \Omega$ (Ω could be any set) such that $f(x) = f(y)$ iff $\exists h \in H$ s.t. $x = hy$ for some $h \in H$ (in other words, x and y belong to the same left coset of H). The question is now of finding H (i.e. a set of generators for H) using as few queries as possible to f . Let's now state Simon's problem as a HSP. Indeed, there we had $G = \mathbb{Z}_2^n$ and $H = \{0^n, s\}$ since $f(x) = f(x \oplus s)$. Similarly, in Shor's example $G = \mathbb{Z}_{\phi(N)}$, and $H = \{0, s \bmod N, 2s \bmod N, \dots\}$ since $f(x) = f(x + i s)$. It turns out that computing H can be done efficiently quantumly for more general groups, namely all finite abelian groups.

Theorem 3 (Shor, Kitaev) $\text{HSP} \in \text{BQP}$ for any finite abelian group.

For non-abelian groups the question has been a huge challenge for more than a decade.

A curious student: Is HSP NP-complete?

Scott: We do not know but that would be extremely surprising, since we have a theorem that states that if SAT is reducible to HSP then the Polynomial Hierarchy collapses, which is not believed to be true. Recall that $\text{PH} = P \cup \text{NP} \cup \text{NP}^{\text{NP}} \cup \text{NP}^{\text{NP}^{\text{NP}}} \cup \dots$ and the k th level is defined as $\text{NP}^{\text{NP} \dots \text{NP}}$ with k NP oracles. For constant k , the k th level of the PH can therefore be described by problems of the form $\exists x_1 \forall x_2 \exists x_3 \dots \exists x_k \phi(x_1, \dots, x_k)$.

A curious student: If PH collapses can we conclude that $P = \text{PSPACE}$?

Scott: We do not know that either. Indeed a complete problem for PSPACE looks like $\exists x_1 \forall x_2 \exists x_3 \dots \exists x_k \phi(x_1, \dots, x_k)$, but here $k = \text{poly}(n)$. We do know however that $\text{HSP} \in \text{NP} \cap \text{coAM}$ and that $\text{HSP} \in \text{Statistical Zero Knowledge (SZK)}$. Also *Approximate Shortest Vector* reduces to HSP over the dihedral group.

Let's prove some nice fact about HSP. We have seen that FACTORING is reducible to HSP over $\mathbb{Z}_{\phi(N)}$. We also can show that

Theorem 4 GI (Graph Isomorphism) $\leq_T \text{HSP over } S_n$ (the Symmetric group on n elements).

Proof: (Sketch) Let C_1 and C_2 be the two graphs given as input. We can assume that each C_i is connected. Let $C = C_1 \cup C_2$ be the disjoint union of the 2 graphs. Label the vertices of C_1 with

distinct integers $1 \dots n_1$ and label the vertices of C_2 with distinct integers $n_1 + 1, \dots, n_1 + n_2$. Let \mathcal{G} be the set of graphs in $n = n_1 + n_2$ vertices.

Let $G = S_n$ and $H = \text{Aut}(C) = \{\pi \in G \mid \pi(C) \text{ is isomorphic to } C\}$, where $\pi(C)$ is the graph obtained from C by permuting its vertices according to π . Clearly H is a subgroup of G . Define a function $f : G \rightarrow \mathcal{G}$ by $f(\pi) = \pi(C)$. Notice that if $\pi = \tau\rho$ where $\rho \in \text{Aut}(C)$ then $f(\pi) = (\tau\rho)(C) = \tau(C) = f(\tau)$, and thus f is constant on cosets of H . Suppose that we know how to compute H . The main observation is that if $C_1 \not\cong C_2$ then, since C_1 and C_2 are each connected, the permutations that occur in H are only those that act independently on the C_i 's.

□

Coming back to the question of efficiently solving HSP for non-abelian groups we state the following result for which we will sketch a proof in the next lecture.

Theorem 5 (Ettinger, Hoyer, Knill) *HSP over any finite group can be solved with a polynomial number of queries.*

Lecture 8

Lecturer: Scott Aaronson

Scribe: Jing Chen

1 Hidden Subgroup Problem

Last time we talked about Shor's factoring algorithm without going through all the details. Before we continue, first let us say something about the quantum Fourier transform (QFT) used in Shor's algorithm. The circuit of a n -bit QFT is defined recursively, that is, a $(n - 1)$ -bit QFT followed by a sequence of controlled phase rotation R_k and a Hadamard, as shown in Figure 1, where

$$R_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\pi i 2^k / 2^n} \end{bmatrix}.$$

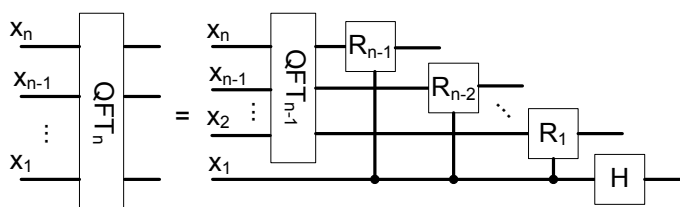


Figure 1: quantum Fourier transform

Shor's algorithm actually solved a particular instance of what we call the *Hidden Subgroup Problem (HSP)*. In this problem, we are given a finite group G which has a hidden subgroup H . Our goal is to find H , or equivalently, to find a set of generators for H . In order to do so, we are given oracle access to a function $f : G \rightarrow Z$, where f is constant on the cosets of H . (Given an element $g \in G$, a coset of H corresponding to g is the set Hg .) Shor's algorithm solves the HSP in the case where G is a cyclic group, e.g., Z_N . It was known since 1970's that if we can find the period of a periodic function, then we are able to factor integers. While finding such a period is the same thing as solving the HSP over a cyclic group where the hidden subgroup is decided by the period.

Let N be the integer that we want to factor. In Shor's algorithm, we prepare a quantum state $\frac{1}{\sqrt{N}} \sum_r |r\rangle$, query the function $f(r) = x^r \bmod N$ in superposition for some x chosen randomly, and get $\frac{1}{\sqrt{N}} \sum_r |r\rangle |x^r \bmod N\rangle$. Then we measure the second register (the $|x^r \bmod N\rangle$ part), and what is left in the first register is a superposition over all possible values of r which could allow us to get the value we observe in the second register. These r 's differ by multiples of the period P of f (the least value P such that $x^P \equiv 1 \bmod N$), and thus the superposition left in the first register can be written as $|r\rangle + |r + P\rangle + |r + 2P\rangle + \dots$.

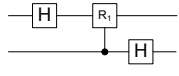
To find out P , we use the quantum Fourier transform, which is the central part of Shor's algorithm. (Notice that given a periodic function, the Fourier transform will map it to its period.) Let n be the number of qubits, and $N = 2^n$ the number of states—that is, the dimension of our

system. The N -dimensional QFT is the unitary matrix

$$QFT(N) = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)^2} \end{bmatrix},$$

where $\omega = e^{2\pi i/N}$. It is easy to check that $QFT(N)$ is indeed a unitary operation, so in principle quantum mechanism allows us to apply this operation. Shor showed that this operation can be implemented (at least approximately) using polynomially many quantum gates —polynomial in n , not in N . The circuit is what we have given in Figure 1, defined recursively, using n^2 quantum gates.

To gain some intuition about QFT , let us see what happens when $n = 2$. First of all, when $n = 1$,

we have a QFT on 1 qubit, which is the Hadamard. So the circuit of QFT_2 is , where

$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}$. The translation procedure for each possible state is as below (unnormalized):

$$\begin{aligned} |00\rangle &\xrightarrow{H} |00\rangle + |10\rangle \xrightarrow{R_1} |00\rangle + |10\rangle \xrightarrow{H} |00\rangle + |01\rangle + |10\rangle + |11\rangle, \\ |01\rangle &\rightarrow |01\rangle + |11\rangle \rightarrow |01\rangle + i|11\rangle \rightarrow |00\rangle - |01\rangle + i|10\rangle - i|11\rangle, \\ |10\rangle &\rightarrow |00\rangle - |10\rangle \rightarrow |00\rangle - |10\rangle \rightarrow |00\rangle + |01\rangle - |10\rangle - |11\rangle, \\ |11\rangle &\rightarrow |01\rangle - |11\rangle \rightarrow |01\rangle - i|11\rangle \rightarrow |00\rangle - |01\rangle - i|10\rangle + i|11\rangle. \end{aligned}$$

The corresponding unitary matrix is (after some reordering) $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$, which is what

we want, i.e., $QFT(4)$. By induction we can prove that the circuit in Figure 1 gives QFT on n qubits.

2 Ettinger-Hoyer-Knill Theorem

Shor's algorithm is an example of this general hidden subgroup paradigm, which includes (not all, but) a huge number of quantum algorithms we know about today. As we have seen before, Simon's algorithm solves in quantum polynomial time a special case of the HSP where $G = Z_2^n$. If we can solve the HSP for general non-Abelian groups, in particular, if we can solve for the symmetric group S_n , then we can solve in quantum polynomial time the graph isomorphism problem.

We do not know how to solve HSP for arbitrary groups in quantum polynomial time, but we do know the following result given by Ettinger, Hoyer and Knill:

Theorem 1 *The hidden subgroup problem can always be solved with $\text{poly}(n)$ queries to f .*

Proof: (*sketch*) To solve the HSP for a given group G , use our favorite procedure: First go into a superposition over all elements in G ($\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle$), then query f in this superposition and get $\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle |f(x)\rangle$. Now measure the $|f(x)\rangle$ register, and what's left in the first register is a superposition $|C\rangle$ over a coset of H , i.e., $|C\rangle = \sum_{h \in H} |hy\rangle$ for some $y \in G$. Repeat this procedure K times, and we get a bunch of superpositions over cosets of H with different values of y , denoted as $|C_1\rangle, \dots, |C_K\rangle$. We claim that if K is large enough, say $\log^2 |G|$, which is just polynomial in the number of qubits, then there exists some measurement (no matter polynomial or not) that can tell us the subgroup.

To prove the above claim, first notice that G can have at most $|G|^{\log |G|}$ different subgroups. This is because each subgroup can have at most $\log |G|$ generators (the size of the subgroup doubles after adding each generator).

Now we need a crucial concept: the *inner product* between two quantum states. It is a way to measure how close the two states are to each other. Let $|\psi\rangle = \alpha_1|1\rangle + \dots + \alpha_N|N\rangle$ and $|\phi\rangle = \beta_1|1\rangle + \dots + \beta_N|N\rangle$ be two quantum states, the inner product between them is denoted by $\langle\psi|\phi\rangle = \alpha_1^*\beta_1 + \dots + \alpha_N^*\beta_N$. Notice that if two quantum states are identical, their inner product is 1; and if they are perfectly distinguishable, their inner product is 0, that is, they are orthogonal to each other.

Consider the coset states $|C_1\rangle \dots |C_K\rangle$ we get when we vary the subgroup H . Let $|\psi_H\rangle = |C_1\rangle \otimes \dots \otimes |C_K\rangle$, and consider $\langle\psi_H|\psi_{H'}\rangle$ for two subgroups $H \neq H'$. Because there exists an element x such that $x \in H \setminus H'$, and because $\forall y \in H \cap H', yx \in H \setminus H'$, we have that $|H \cap H'| \leq \frac{|H|}{2}$. Therefore $|\langle H|H'\rangle| \leq \frac{1}{\sqrt{2}}$, where $|H\rangle$ and $|H'\rangle$ are two quantum states over all elements in H and H' respectively. (That means if two subgroups are almost the same, then they are actually the same. While if they are different from each other, then they are very different —by a constant factor of all of the places.) Moreover, if $\langle\psi|\phi\rangle \leq \varepsilon$, then $(\langle\psi|\otimes\langle\psi|)(|\phi\rangle\otimes|\phi\rangle) = \langle\psi|\phi\rangle \cdot \langle\psi|\phi\rangle \leq \varepsilon^2$. Therefore $\langle\psi_H|\psi_{H'}\rangle \leq (\frac{1}{\sqrt{2}})^K$, since the inner product of two cosets of H and H' can only get smaller than $\langle H|H'\rangle$. As there are at most $|G|^{\log |G|}$ distinct subgroups H_1, H_2, \dots , there are at most this much $|\psi\rangle$'s, and $\langle\psi_{H_i}|\psi_{H_j}\rangle \leq (\frac{1}{\sqrt{2}})^K$ for any $i \neq j$. Choose K such that $(\frac{1}{\sqrt{2}})^K \ll |G|^{-2\log |G|}$. Using the Gram-Schmidt process, we can make all $|\psi\rangle$'s exactly orthogonal, while introducing a total error at most $|G|^{2\log |G|}(\frac{1}{\sqrt{2}})^K \ll 1$. Then there exists some unitary operation U which, when applied to our $|\psi_H\rangle$ will rotate it to a particular state such that the measurement will tell H with exponentially small error probability.

□

Remark. Notice that the above procedure may take exponential time, but when talking about query complexity, we do not care about how much time is needed for computation that does not involve queries to f . This is the distinction between query complexity and computation complexity. Thus it is possible that solving HSP requires exponential computation time. But recall that even assuming computation is free, solving HSP in the classical world may still need exponentially many queries to f , as we have met when discussing Simon's algorithm.

3 Grover's Algorithm

An important question about quantum computation is: can we design polynomial time quantum algorithm to solve NP-complete problems? Along this line we will talk about the other main

quantum algorithm that we know, Grover's algorithm.

Given oracle access in superposition to a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, the problem is to find some $x \in \{0, 1\}^n$ such that $f(x) = 1$, providing that such an x indeed exists. For simplicity, we assume that there is exactly one x for which $f(x) = 1$.

Another way to think about it is to search a database with N items for a “marked item”. In classical world, any deterministic algorithm will require N queries to the database in the worst case, and any randomized algorithm will require $N/2$ queries in expectation. If we can query f in superposition, things gets more interesting. Say we can take a superposition over all items, $\sum_x \alpha_x |x\rangle$, make a query to f in this superposition and get $\sum_x \alpha_x (-1)^{f(x)} |x\rangle$, or equivalently, $\sum_x \alpha_x |x\rangle |f(x)\rangle$. Then can we find the marked item using only n^2 ($n = \log N$) queries? That is, what is the quantum query complexity of searching a database? If this can be done polynomially, and further, if the algorithm can be implemented in polynomial time, then quantum computer can solve NP-complete problems in polynomial time, and $NP \subseteq BQP$. However, a straight-forward method is not going to work. That is, if we make the above query to f and measure the second register, then most of the time we will get an x such that $f(x) = 0$. To extract the good solution, we need to explore the structure of f .

Theorem 2 (Grover) *We can search a database of N items in $O(\sqrt{N})$ queries in quantum computation.*

Remark 1. This result is tight, and we will prove this point later. Actually it is proved to be tight before the algorithm was discovered.

Remark 2. Compared with Simon's and Shor's algorithm, Grover's algorithm gives only a quadratic speedup rather than an exponential one. But it works for a much wider range of problems —any combinatorial searching problem.

The algorithm starts as every quantum algorithm: go into a superposition of all possible solutions, $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$, then query f in this superposition and get $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle$.

Now it comes the magical part: Grover Diffusion Operator. Basically what we want to do is to apply some unitary operation that takes all amplitudes and “inverts them about the average”. Let the amplitude vector be $[\alpha_1 \cdots \alpha_N]^T$ ($N = 2^n$), and $S = \frac{\alpha_1 + \cdots + \alpha_N}{N}$ the average, we want to get the vector $[\alpha_1 - 2(\alpha_1 - S), \cdots, \alpha_N - 2(\alpha_N - S)]^T$. That is, we want to “flip” every amplitude around the average, as shown in Figure 2.

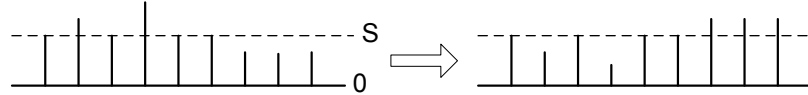


Figure 2: Invert About Average

The corresponding unitary operation is
$$\begin{bmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \frac{2}{N} & \cdots \\ \vdots & \frac{2}{N} & \ddots & \frac{2}{N} \\ \frac{2}{N} & \cdots & \frac{2}{N} & \frac{2}{N} - 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}, \text{ where}$$

the elements in the matrix's diagonal are all $\frac{2}{N} - 1$, and the other elements are all $\frac{2}{N}$. It is easy to verify that this operation is indeed unitary.

The circuit of Grover's algorithm is shown in Figure 3, where f stands for a query to the oracle f , and D stands for a Grover diffusion operator. The basic f, D operation is repeated \sqrt{N} times, and then measure.

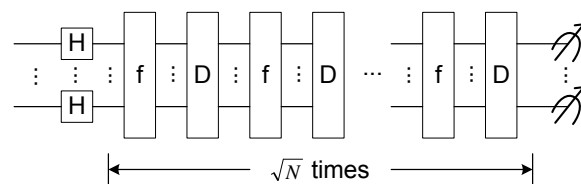


Figure 3: Grover's Algorithm

The circuit for the diffusion operation is shown in Figure 4, where U_0 is the unitary operation that maps $|x\rangle$ to $(-1)^x|x\rangle$, where $x = 0, 1$. Essentially, in the diffusion operation we first switch from the standard basis to the Fourier basis. Then in the Fourier basis, we negate all the Fourier coefficients except for the first one, which corresponds to the average. Finally we return to the standard basis.

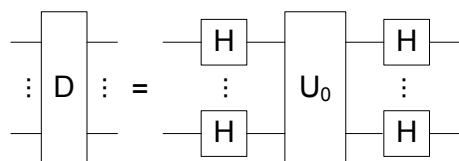


Figure 4: Grover Diffusion Operator

We will analyze this algorithm next time.

Lecture 9

*Lecturer: Scott Aaronson**Scribe: David Gosset*

In this class we discuss Grover's search algorithm as well as the BBBV proof that it is optimal.

1 Grover's Algorithm

1.1 Setup

Given N items $\{x_1, x_2, \dots, x_N\}$ we wish to find an index i such that $x_i = 1$. We are able to query the values x_i in quantum superposition (as usual). Grover's algorithm solves this problem using $O(\sqrt{N})$ quantum queries.

1.2 The Algorithm

Grover's algorithm can be described by the following circuit. In figure 1 the query operator is the

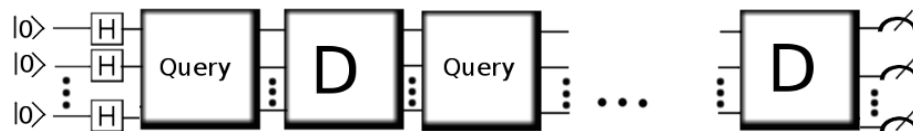
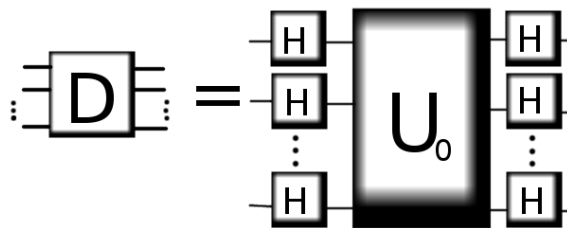


Figure 1: Circuit for Grover Search

standard phase query which transforms $\sum_i \alpha_i |i\rangle \rightarrow \sum_i \alpha_i (-1)^{x_i} |i\rangle$. The operator labeled D is called the Grover Diffusion Operator, which is given by the circuit in figure 2. We could also write



$$U_0 = 2|0\rangle\langle 0| - 1$$

Figure 2: Grover Diffusion Operator

the Grover Diffusion operator as a N by N matrix in the computational basis as follows:

$$\begin{pmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \frac{2}{N} & \dots & \frac{2}{N} \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \frac{2}{N} & \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N} - 1 \end{pmatrix}$$

Note that this is a unitary transformation. Let's consider the action of this operator on a general quantum state $\sum_{i=1}^N \alpha_i |i\rangle$. From figure 2, the operator has the effect of first switching from the computational basis to the Fourier basis (effected by the Hadamard transforms), applying a phase of (-1) to the zero Fourier mode, and then switching back to the computational basis. This operator has the net effect of inverting the amplitudes of the quantum state about their mean value, as illustrated in the following picture. Grover's algorithm, which is sometimes called

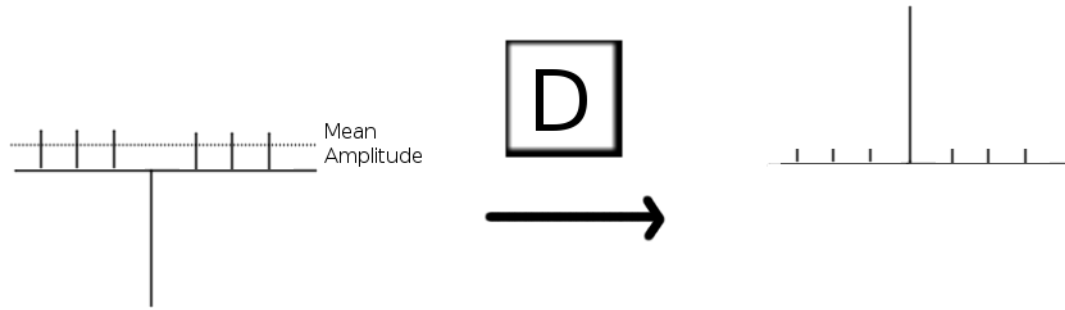


Figure 3: The Grover Diffusion Operator

'amplitude amplification' simply consists of alternating the query and diffusion operators. For the first few steps of the algorithm the amplitude of the solution is increased by approximately $\frac{2}{\sqrt{N}}$ (this is not the case when the amplitude of the solution is large, i.e when we are nearing the end of the algorithm. A more detailed analysis follows.). The reason we can get the answer with constant probability in $O(\sqrt{N})$ steps is because in quantum mechanics we're dealing with the L_2 norm as opposed to the L_1 norm for classical probability. Now let's analyse the algorithm in more detail. Write a_t for the amplitude of the marked item after time t (i.e after t iterations of Query+Diffusion).

We have $a_0 = \frac{1}{\sqrt{N}}$. The state vector after time t can be written as:

$$\begin{pmatrix} \sqrt{\frac{1-(a_t)^2}{N-1}} \\ \sqrt{\frac{1-(a_t)^2}{N-1}} \\ \cdot \\ \cdot \\ \cdot \\ a_t \\ \cdot \\ \cdot \\ \cdot \\ \sqrt{\frac{1-(a_t)^2}{N-1}} \end{pmatrix}$$

We can then apply the query operator followed by inversion about average to obtain an expression for a_{t+1} in terms of a_t :

$$a_{t+1} = \left(1 - \frac{2}{N}\right) a_t + \frac{2}{N} \sqrt{(1 - a_t^2)(N - 1)} \quad (1)$$

We see from the above that when $a_t \ll 1$, " $\frac{da}{dt}$ " = $a_{t+1} - a_t \approx 2 \frac{(1-a_t^2)\sqrt{N-1}}{N}$. If you solve this equation exactly for a_t , given $a_0 = \frac{1}{\sqrt{N}}$, you will obtain that indeed the number of iterations required to get the solution with constant probability is $O(\sqrt{N})$. It is also the case that if you do too many iterations, the amplitude a_t will decrease again. The exact expression(see [1]) is $a_t = \sin\left(\frac{(2t+1)\theta}{2}\right)$ where $\sin\left(\frac{\theta}{2}\right) = \sqrt{\frac{1}{N}}$.

There is also a geometric interpretation of Grover's algorithm. The state of the system at all times during the algorithm is in the subspace of the Hilbert space spanned by the states $|x_i\rangle$ and $\frac{1}{\sqrt{N-1}} \sum_{j \neq i} |j\rangle$, where i is the index of the solution. Each time the Query/Diffusion operators are applied, the state vector is rotated by the angle θ in this subspace.

2 Optimality of Grover's Algorithm (Bennett-Bernstein-Brassard-Vazirani[2])

This is the "Hybrid" argument from [2]. Let's assume we have some quantum algorithm which consists of a sequence of unitaries and phase queries: $U_1, Q_1, U_2, Q_2, \dots, Q_T, U_T$. At first we imagine doing a trial run of the algorithm, where the oracle has $x_j = 0 \forall j \in \{1, \dots, N\}$. Define

$$\alpha_{i,t} = \text{Total amplitude with which the algorithm queries } x_i \text{ at step } t. \quad (2)$$

So if the state of the system at time t is $\sum_{i,z} \alpha_{i,z,t} |i, z\rangle$ (z is an additional register) then $\alpha_{i,t} = \sqrt{\sum_z |\alpha_{i,z,t}|^2}$. Then define the query magnitude of i to be:

$$m_i = \sum_{t=1}^T |\alpha_{i,t}|^2 \quad (3)$$

$$= \text{"Query magnitude of i"} \quad (4)$$

We have that $\sum_{i=1}^N m_i = \sum_{i=1}^N \sum_{t=1}^T \sum_z |\alpha_{i,z,t}|^2 = \sum_{t=1}^T 1 = T$. This means that there must exist a $\tilde{i} \in \{1, \dots, N\}$ such that $m_{\tilde{i}} \leq \frac{T}{N}$. So:

$$\sum_{t=1}^T |\alpha_{\tilde{i},t}|^2 \leq \frac{T}{N} \quad (5)$$

$$\Rightarrow \sum_{t=1}^T |\alpha_{\tilde{i},t}| \leq \sqrt{\sum_{t=1}^T |\alpha_{\tilde{i},t}|^2} \sqrt{T} \quad (\text{Cauchy-Schwarz inequality}) \quad (6)$$

$$\leq \frac{T}{\sqrt{N}} \quad (\text{From 2 lines above}) \quad (7)$$

Now suppose that we modify the first oracle so that item \tilde{i} is marked when it is first used in the algorithm (at $t=1$), but then use an oracle with no marked item for the rest of the queries throughout the algorithm. Then the state at time $t=1$ is modified by replacing $\alpha_{i,z,1} \rightarrow -\alpha_{i,z,1}$. We can think of this modified state as the old state (in the case the oracle had no marked item) plus an error term. The state after the rest of the algorithm after the first step still only differs by a small error term.

If we then change the oracles used so that $x_{\tilde{i}} = 1$ at $t = 1, 2$ but $x_i = 0$ for $t = 3, 4, \dots, T$, we also obtain another error term which adds to the one above. We can then continue this process until the oracle has $x_{\tilde{i}} = 1$ for all $t \in \{1, \dots, T\}$. The total amount by which the amplitude in state \tilde{i} of the final state of the algorithm can change during this process (changing from no marked item to one marked item for all t) is $\frac{cT}{\sqrt{N}}$ where c is a constant. So assuming that the probability of detecting a marked item is zero when there is no marked item, it cannot be greater than $\frac{cT^2}{N}$ when there is a marked item.

If $N = 2^n$ then any quantum algorithm requires $\Omega\left(2^{\frac{n}{2}}\right)$ queries to find the marked item with constant probability.

2.1 Oracle Separation $NP^A \not\subseteq BQP^A$

This result implies that there is an oracle A for which $NP^A \not\subseteq BQP^A$. The oracle A computes a boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Then any quantum algorithm requires $\Omega\left(2^{\frac{n}{2}}\right)$ queries to determine if there exists a value x for which $f(x) = 1$. Standard diagonalisation techniques can be used to make this rigorous.

3 Query Complexity

Definition 1 Given a boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, define the quantum query complexity $Q(f)$ to be the minimum number of queries required by a quantum computer to compute f with error probability $\leq \frac{1}{3}$ on all inputs.

What we know about query complexity so far:

$$Q(OR_n) = O(\sqrt{n}) \quad [\text{Grover}] \quad (8)$$

$$Q(OR_n) = \Omega(\sqrt{n}) \quad [\text{BBBV}] \quad (9)$$

$$Q(PARITY) \leq \frac{n}{2} \quad [\text{Deutsch-Josza}] \quad (10)$$

$$Q(PARITY) = \Omega(\sqrt{n}) \quad [\text{BBBV}] \quad (11)$$

$$(12)$$

In fact, $Q(PARITY) \geq \frac{n}{2}$, which we will see next time using the polynomial method.

References

- [1] *Quantum computation and quantum information*. Cambridge University Press, New York, NY, USA, 2000.
- [2] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing, 1997.

Lecture 10

Lecturer: Scott Aaronson

Scribe: Sam McVeety

*“Science: We seek out ignorance and try to demolish it.”***Last Time: Grover’s algorithm and its optimality**

- $D(f)$ = deterministic query complexity of Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$
- $Q(f)$ = quantum query complexity (with bounded error)
- Trivially, $Q(f) \leq D(f)$ for all f .
- $D(OR_n) = n$
- $Q(OR_n) = O(\sqrt{n})$ [Grover]
- $Q(OR_n) = \Omega(\sqrt{n})$ [BBBV]
- $Q(PARITY_n) \leq \frac{n}{2}$ [Deutsch-Jozsa]

Can we show $Q(PARITY_n) \geq \frac{n}{2}$? BBBV gives us \sqrt{n} as a lower bound, if we consider the case with only one bit flipped. It turns out that this bound is tight, and we will develop a new method to prove this.

Today, we will continue our study of query complexity, where we are completely ignoring computation time in our analysis. Recall that this allows for an adaptive algorithm which chooses its queries based on the results of previous queries.

1 The Polynomial Method

The so-called polynomial method takes questions about quantum lower bounds and reduces them to questions about the degrees of real polynomials. This approach will perhaps be somewhat more palatable to computer scientists who are still at odds with quantum theory. (Though the application is new, the method is old and derives from a book by Minsky, et al.)

Lemma 1 (*Beals, Buhrman, Cleve, Mosca, de Wolf 1998*) *Let Q be a quantum algorithm that outputs a boolean value. We are interested in the likelihood that it accepts. If Q makes T queries to input bits x_1, \dots, x_n , then Q ’s acceptance probability can be represented by a polynomial $p(x_1, \dots, x_n)$ of degree at most $2T$.*

In other words, if you can prove a lower bound on the degree of a real polynomial that behaves properly (high values for accepted inputs, low values for rejections) then T has to be at least the degree of the polynomial divided by 2, and you get a lower bound on the number of quantum queries.

Proof: Claim: every amplitude can be written as a polynomial of degree at most T after T queries.

By induction on T : Initially all amplitudes $\alpha_{i,2}$ are degree-0 polynomials over x_1, \dots, x_n - in other words, constants. A query increases degree by 1:

$$\sum \alpha_{i,2} |i, 2\rangle \rightarrow \sum \alpha_{i,2} (1 - 2x_i) |i, 2\rangle$$

A unitary operation doesn't change the degree because it is a linear transformation, and we get the two from squaring all coefficients to get a probability. For simplicity, we can assume only real numbers, as shown on the problem set.

$$p = \sum_{|i,2\rangle \text{ accepting}} ||\alpha_{i,2}||^2$$

□

Definition 1 $\widetilde{\deg}(f)$ is the minimum degree of a real polynomial p such that

$$|p(X) - f(X)| \leq \frac{1}{3} \quad \forall X \in \{0, 1\}^n$$

Notice that:

$$\begin{aligned} \widetilde{\deg}(f) &\leq 2Q(f) \\ Q(f) &\geq \frac{\widetilde{\deg}(f)}{2} \end{aligned}$$

Now we will attempt to find such polynomials.

Remark 1 We can assume that p is multilinear without loss of generality. In other words, every term is a product of some subset of the variables to the first power, because the inputs are all 0 or 1, so raising to higher powers is meaningless.

This task still seems complicated, so we'll learn a new trick that reduces n -dimensional objects to 1-dimensional objects.

Lemma 2 (Minsky-Papert 1968) Let $q(k) = E_{|X|=k}[p(X)]$ (Expected value over all X with Hamming weight k). Then $q(k)$ is itself a polynomial in k , and $\deg(q) \leq \deg(p)$. One polynomial is over the input bits, the other is over the Hamming weight of the input bits. This process is known as symmetrization.

Proof:

This is a multilinear polynomial, so we can write it as a sum of linear monomials.

$$p(X) = \sum_{S \subseteq \{1, \dots, n\}} \alpha_S \prod_{i \in S} x_i, \quad \alpha_S \in \mathbb{R}$$

Use linearity of expectation:

$$E_{|X|=k}[p(X)] = \sum_{|S| \leq \deg(p)} \alpha_s E_{|X|=k} \left[\prod_{i \in S} x_i \right]$$

This now becomes a combinatorics problem, where we are looking for the probability that all the bits in S will be set to 1. Write out the terms and cancel.

$$\begin{aligned} &= \sum_{|S| \leq \deg(p)} \alpha_s \frac{\binom{n-|S|}{k-|S|}}{\binom{n}{k}} \\ &= \sum_{|S| \leq \deg(p)} \alpha_s \frac{(n-|S|)!}{n!} k(k-1)(k-2) \cdots (k-|S|+1) \end{aligned}$$

(Oh look, a polynomial of degree at most the degree of p .)

What can we say about $p(0, \dots, 0) = q(0)$? $0 \leq q(0) \leq \frac{1}{3}$. Also, $\frac{2}{3} \leq q(1) \leq 1$ and $0 \leq q(2) \leq \frac{1}{3}$. If we draw this polynomial (approximately) we can see that its degree is at least n . (Choose your justification: because it reverse direction n times; if you subtract $1/2$ you can count the zeros, etc.) So:

$$\begin{aligned} n \leq \deg(q) &\leq \deg(p) = \widetilde{\deg}(\text{PARITY}_n) \\ Q(\text{PARITY}_n) &\geq \frac{\widetilde{\deg}(\text{PARITY}_n)}{2} = \frac{n}{2} \end{aligned}$$

Thus, our algorithm is optimal, and parity is in a sense maximally hard for a quantum computer. \square

A natural next question is whether this technique can be applied to Grover's algorithm. It turns out that this is true, giving us a completely different proof of the optimality of Grover's algorithm.

Notice that we can't use our polynomial-degree inference method here, because the OR function starts out low and goes high and never goes back. This indicates that the case is more subtle here, even though the polynomial doesn't appear to be low degree (staying flat for a while.)

We turn to a result of Markov, from a conversation with Mendeleyev (of periodic table fame) regarding the maximum absolute value of the derivative of a polynomial in a given range.

Lemma 3 (*A.A. Markov 1889*)

$$\max_{0 \leq x \leq n} |p'(x)| \leq \deg(p)^2 \left(\frac{\max_{0 \leq x \leq n} p(x) - \min_{0 \leq x \leq n} p(x)}{n} \right)$$

Proof: We won't prove this here, but you can, using Chebyshev polynomials. \square

In other words

$$\deg(p) \geq \sqrt{\frac{n \cdot \text{MaxDeriv}}{\text{MaxHeight}}}$$

Notice that $q'(x) \geq \frac{1}{3}$ for some $x \in [0, 1]$. There is a subtlety here, because we only know the behavior of the polynomial at integer points. (It could do something wacky!) But, it can't go too extreme without making the max derivative larger.

“You have to get back by your curfew, which is the next integer.”

$$\geq \sqrt{\frac{n \cdot \max\{\frac{1}{3}, 2h - 1\}}{h}} = \Omega(\sqrt{n})$$

2 Limiting Possible Speedups

We’ve seen quadratic speedups, but it is natural to ask if there any total boolean function that gives us an exponential speedup by using a quantum algorithm. Factoring isn’t a candidate here, because the speedup is conjectured. Period finding isn’t either, because it requires that the input be periodic, which entails a promise about the input.

It turns out that the following result holds:

Proposition 4 (*BBCMW*): $D(f) = O(Q(f)^6)$ for all f .

Conjecture 1 $D(f) = O(Q(f)^2)$. *Scott’s intuition.*

Indeed, if you’re going to get an exponential speedup, you need some sort of promise about this input.

Definition 2 *Given some boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, a certificate for input $X \in \{0, 1\}^n$ is some subset of input bits that proves the value of $f(X)$. For the OR function, most inputs have certificate size 1. The input of all zeros has certificate size 0.*

Let $c_x(f)$ be the minimum size (number of bits) of a certificate for X , and let $C(f) = \max_x c_x(f)$ (Check: for OR_n , this is n .)

Observe that $C(f) \leq D(f)$, because we have to uncover a certificate before terminating, otherwise there are still plausible inputs which are accepted or rejected. However, equality does not hold due to a special function:

“OR of ANDs”: Represented by a tree of where the output bit (the root) is the OR of \sqrt{n} nodes at the next level, which are in turn the AND of \sqrt{n} leaves of each node, which is each an input bit. Equivalently, we have a $\sqrt{n} \times \sqrt{n}$ matrix with a bit in each position. Here, the question is whether there is an all-ones row. Note that $D(f) = n$, as there is no good deterministic algorithm to check for such a row. However, $C(f) = \sqrt{n}$, because a 0 in each row, or an all-ones row, are certificates.

Theorem 5 (*Folklore*) $D(f) \leq C(f)^2$ for all f . *Observe that this is sort of like P versus NP for query complexity, showing that this world is very different than the conjectured world of computational complexity.*

Proof: Pick a 1-certificate A_1 and query it. If $f(X)$ is forced to 1, then halt. Pick another 1-certificate A_2 that is consistent with the first and query it. (If we are ever blocked because the next certificate does not exist, then we halt and output zero.) Iterate in the obvious way.

Each iteration makes $\leq C(f)$ queries. We will prove a bound on the number of iterations in the following lemma. \square

Lemma 6 *Every 0-certificate intersects every 1-certificate in at least one place. If you had disjoint certificates, then you simultaneously prove true and false by fixing both of them.*

So, every 1-certificate fixes another bit of a zero-certificate. So our zero certificates shrink by at least 1 each time, so the number of iterations is at most $C(f)$.

Lecture 11

Lecturer: Scott Aaronson

Scribe: Jessica Yuan

In this class, we'll finish the proof for $D(f) = O(Q(f)^6)$ for all total Boolean functions f .

1 Block Sensitivity

We already saw the first step in the proof: $D(f) \leq C(f)^2$ for all f , where $C(f)$ is the certificate complexity. We will now do the second step, which involves *block sensitivity*.

Definition 1 Given input $x \in \{0, 1\}^n$ and a subset $S \subseteq \{1, \dots, n\}$ of bits, let $X^{(S)}$ denote X with the bits in S flipped. We call S a sensitive block on input X if $f(X) \neq f(X^{(S)})$.

$bs_x(f)$, the block sensitivity of f on input X , is the maximum number of disjoint sensitive blocks that can be found on input X .

We define the block sensitivity of f to be $bs(f) = \max_x bs_x(f)$.

Claim 1 $bs(f) \leq C(f)$ for all f .

This is because to prove the value of $f(X)$, a certificate must intersect each sensitive block. Otherwise, flipping the bits in a sensitive block would change $f(X)$ without affecting the certificate's value.

Lemma 2 (Nisan) $C(f) \leq bs(f)^2$ for all f .

Proof: Given an input X , consider a maximal set of $k = bs_x(f)$ disjoint sensitive blocks S_1, \dots, S_k for X . Note that $k \leq bs(f)$. Assume without loss of generality that each S_i is *minimal* (i.e., no sub-block of S_i is also sensitive). If S_i was not minimal, we could always use a sensitive sub-block of S_i instead, without affecting the maximality of the set S_1, \dots, S_k .

Claim 3 The union of bits $S_1 \cup \dots \cup S_k$ is a certificate, proving the value of $f(X)$.

To show this, suppose we wanted to flip some bits in X to change the value of $f(X)$. The bits we flipped would have to comprise a sensitive block S_{k+1} . But if S_{k+1} didn't intersect any of S_1, \dots, S_k , it would contradict the assumption that S_1, \dots, S_k was a maximal set of disjoint sensitive blocks.

So the bits must intersect $S_1 \cup \dots \cup S_k$. If we monitor $S_1 \cup \dots \cup S_k$ as a certificate, then we are forcing the value of $f(X)$, since it cannot be changed without the certificate also changing.

Now we put an upper-bound on the size of $S_1 \cup \dots \cup S_k$. As stated before, $k \leq bs(f)$. Now we will show that each S_i has at most $bs(f)$ bits, making

$$|S_1 \cup \dots \cup S_k| \leq bs(f)^2 \quad (1)$$

Consider $X^{(S_i)}$. We know that $f(X^{(S_i)}) \neq f(X)$. By the assumption that S_i is minimal, flipping any bit in S_i changes the value of $f(X^{(S_i)})$ back to $f(X)$; otherwise, there'd be a smaller

sensitive block for X . So each bit in S_i is itself a sensitive block for $X^{(S_i)}$. Hence, $|S_i| \leq bs(f)$. So $|S_i \cup \dots \cup S_k| \leq bs(f)^2$.

Since $S_i \cup \dots \cup S_k$ is a certificate, we can conclude that $C(f) \leq bs(f)^2$ for all f .

□

2 Block Sensitivity and Approximate Degree

The third and final step is to relate $bs(f)$ to the approximate degree $\widetilde{deg}(f)$. We've already proven that $\widetilde{deg}(OR_n) = \Omega(\sqrt{n})$. We will now claim a generalization of that result.

Lemma 4 $\widetilde{deg}(f) = \Omega(\sqrt{bs(f)})$ for all Boolean functions f .

Proof: Consider an input X of f with $k = bs(f)$ disjoint sensitive blocks S_1, \dots, S_k . Assume without loss of generality that $f(X) = 0$. Now define a new Boolean function with k inputs: $f'(y_1, \dots, y_k)$ equals $f(X$ with S_i flipped iff $y_i = 1$). Suppose there were a polynomial p that approximated f . Let $p'(y_1, \dots, y_k)$ equal $p(X$ with S_i flipped iff $y_i = 1$). Symmetrize p' to get a univariate polynomial

$$q(k) = EX_{|Y|=k}[p'(Y)]. \quad (2)$$

$0 \leq q(0) \leq \frac{1}{3}$ and $\frac{2}{3} \leq q(1) \leq 1$. We don't know what $q(2), \dots, q(k)$ are, but we know that as averages of 0's and 1's, they must be bounded between 0 and 1. By Markov's inequality, $deg(q) = \Omega(\sqrt{k})$, which means that $\widetilde{deg}(f) = \Omega(\sqrt{bs(f)})$. □

3 Putting It All Together

Thus, we can conclude that

$$D(f) \leq C(f)^2 \leq bs(f)^4 = O(\widetilde{deg}(f)^8) = O(Q(f)^8) \quad (3)$$

Note that we had claimed that $D(f) = O(Q(f)^6)$. Earlier, we proved that $D(f) \leq C(f)^2$, but it turns out that this bound can be tightened to $D(f) \leq C(f)bs(f)$.

Proof: To show that this is true, recall the algorithm that repeatedly picks a 1-certificate consistent with everything queried so far, and then queries all the bits in it. We proved that this algorithm cannot run for more than $C(f)$ steps, but actually, it cannot run for more than $bs(f) + 1$ steps.

For the purposes of contradiction, suppose it did; suppose we've picked and queried more than $bs(f) + 1$ certificates. Because the algorithm hasn't stopped yet, there must still be both a 0-input and a 1-input compatible with our queries. Let X be such a 0-input. For this X , we can find more than $bs(f)$ disjoint sensitive blocks, a contradiction. This is because in each iteration, we queried a potential 1-certificate, meaning that had a subset of those bits been flipped, f would have been forced to be 1. That subset is a sensitive block. Since there are $bs(f) + 1$ certificates and they're disjoint, we have more than $bs(f)$ disjoint sensitive blocks. □

Thus,

$$D(f) \leq C(f)bs(f) \leq bs(f)^3 = O(\widetilde{deg}(f)^6) = O(Q(f)^6) \quad (4)$$

This is still the best relation known. Another open problem is whether or not there is a similar polynomial relation between $D(f)$ and $Q(f)$ for *almost*-total Boolean functions, where the function is defined for only $(1 - \epsilon)$ fraction of the inputs.

4 Interesting Examples of Boolean Functions

This is a discussion of particular Boolean functions that people are interested in and have done research on.

Recall the previously discussed OR/AND tree problem, a.k.a. “Is there an all-ones row?” The classical query complexity for this problem is n .

An upper-bound for the quantum query complexity is $O(n^{\frac{3}{4}})$, which comes from applying Grover’s algorithm separately to each AND subtree.

Applying Grover’s algorithm recursively, we can get $O(\sqrt{n} \log n)$. (The $\log n$ comes from the need to use amplification to reduce the error.) A more careful recursive application of Grover’s algorithm can even reduce that to $O(\sqrt{n})$.

The OR/AND tree problem can also be thought of as a game, where Player 1 picks a subtree and Player 2 picks a leaf within that subtree. Player 1 wins if a 1 is reached, Player 2 wins if a 0 is reached, and the question is who wins?

To find a lower-bound on the quantum query complexity, we can apply the BBBV lower-bound to a single subtree and get a bound of $\Omega(n^{1/4})$. Raising that lower-bound to $\Omega(\sqrt{n})$ had been an open problem for several years. The difficulty comes from the OR/AND tree not being a symmetric Boolean function, so symmetrization doesn’t yield anything. Proving that the approximate degree is $\Omega(\sqrt{n})$ remains an open problem, but we *can* prove it is $\Omega(n^{\frac{1}{3}})$.

4.1 Ambainis’s Quantum Adversary Method

Andris Ambainis introduced a completely new quantum lower bound method that has since had an enormous influence on the field. We’re not going to go into the details of the proof, but we’ll show what the theorem is and how it’s applied.

Theorem 5 (Ambainis) *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function, X be some set of 0-inputs, and Y be some set of 1-inputs. Let $R \subset X \times Y$ be a relation such that*

- 1. For every $x \in X$, there are at least m $y \in Y$ such that $(x, y) \in R$.*
- 2. For every $y \in Y$, there are at least m' $x \in X$ such that $(x, y) \in R$.*
- 3. For every $x \in X$ and $i \in [n]$, there are at most s $y \in Y$ such that $(x, y) \in R$ and $x_i \neq y_i$.*
- 4. For every $y \in Y$ and $i \in [n]$, there are at most s' $x \in X$ such that $(x, y) \in R$ and $x_i \neq y_i$.*

Then any quantum algorithm to compute f needs $\Omega(\sqrt{\frac{mm'}{ss'}})$ queries.

The basic proof idea is to consider a superposition of inputs. If the algorithm succeeds at distinguishing 0-inputs from 1-inputs, then by the end, this pure state must evolve into a mixed state; certain off-diagonal entries in the density matrix must be 0. The theorem bounds how long this takes to happen.

The conditions for the theorem basically state that every 0-input has many 1-inputs as neighbors, but not too many 1-inputs as neighbors that differ from it at any specific bit. The same goes for 1-inputs with 0-inputs as neighbors.

4.2 Application of the Quantum Adversary Method

Let’s apply the theorem to the OR problem. The set of 0-inputs X contains only the all-0 input. The set of 1-inputs Y will consist of all inputs with Hamming weight 1. In general, we want the 1-inputs to be close to the 0-inputs to maximize the lower bound generated by the theorem.

Every 0-input has $m = n$ neighbors. Every 1-input has $m' = 1$ neighbor. Every 0-input has $s = 1$ neighbor that differs from it at a specific bit. Every 1-input has $s' = 1$ neighbor that differs from it at a specific bit. Plugging these values in, we get $\Omega(\sqrt{n})$, confirming Grover's algorithm's tightness.

Now let's apply the theorem to the OR/AND tree problem, modeled as a $\sqrt{n} \times \sqrt{n}$ matrix where we need to determine if there's a row of all 1's. A useful trick is often to use inputs that are right at the threshold between being 0-inputs or 1-inputs.

Let's say that each row in a matrix has either zero or one 0 in it; that is to say, each row is either all 1's or almost all 1's. A 1-input would have a single row of all 1's and all other rows have 1's for all but one bit. A 0-input would have each row have one zero. If a 0-input matrix x and a 1-input matrix y differ at just a single bit, we'll make them neighbors, i.e. $(x, y) \in R$.

Every 0-input has $m = \sqrt{n}$ neighbors, by changing one of the \sqrt{n} 0 bits into a 1 bit. Every 1-input has $m' = \sqrt{n}$ neighbors, by changing one of the \sqrt{n} 1 bits in the all 1's row into a 0 bit. Every 0-input has at most $s = 1$ neighbor that differs from it at a specific bit. Every 1-input has at most $s' = 1$ neighbor that differs from it at a specific bit. Plugging those values in, we get that an algorithm that solves the OR/AND problem must have quantum query complexity $\Omega(\sqrt{n})$.

Next time, we'll talk about the collision problem.

Lecture 12

*Lecturer: Scott Aaronson**Scribe: Brendan Juba*

1 A second non-symmetric example

1.1 Limitations of speed-ups without structure

In the preceding lectures, we examined the relationship between deterministic query complexity and quantum query complexity. In particular, we showed

Theorem 1 $\forall \text{Boolean } f D(f) = O(Q(f))^6$

so we won't ever obtain a superpolynomial improvement in the query complexity of *total* functions, and to obtain a superpolynomial speed-up, we can't treat the problem as a black-box like Grover's algorithm does. We need to exploit some kind of structure, possibly in the form of a promise on the function, as Shor's algorithm does. Moreover, for symmetric Boolean functions – OR, MAJORITY, PARITY, and the like – the largest separation possible is only quadratic, which is achieved by the OR function, as demonstrated by Grover's algorithm and our various lower bounds. For functions such as MAJORITY, which switches from 0 to 1 around the middle Hamming weight, the advantage only diminishes, and any quantum algorithm can be shown to require $\Omega(n)$ queries, just as classical algorithms do.

We don't have such a clear picture of the state of affairs for non-symmetric Boolean functions, but a few concrete examples have been worked out. A natural next question is to consider what happens when our simple functions are composed. For example, we saw last time that for the two level OR-AND tree (with \sqrt{n} branches at each level), the quantum query complexity is $\Theta(\sqrt{n})$, with the upper bound provided by a recursive application of Grover's algorithm, and the lower bound obtained via Ambainis' adversary method (stated without proof)—the quantum extension of the BBBV hybrid argument and variants, where we argue about what a quantum algorithm can do step-by-step. By contrast, we still don't know how to obtain the lower bound using the polynomial method, where we reduce questions about quantum algorithms to questions about low-degree polynomials, (the “pure math” approach) which is elegant *when* it works. Thus, these two methods seem to have complementary strengths and weaknesses.

1.2 The AND-OR tree, or: the power of randomization in black-box query complexity

The second example which we understand is also an AND-OR tree, but it is deeper, consisting of $\log n$ levels with two branches at each node (see Figure 1). We think of this as modeling a $\log n$ -round game of pure strategy between two players in which the players have two options at each round (we could think of it more generally as a constant number of moves), and the winner is determined by a black-box evaluation function—the natural computational question in such a game is, “is there a move I can make such that for every move you can make, I can force a win?” This is precisely the problem of game tree evaluation. This black-box assumption allows us to begin

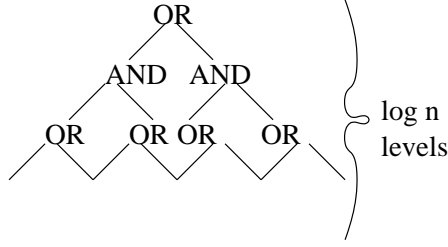


Figure 1: $\log n$ -depth AND-OR tree

to address the question of what kind of a speed-up we can hope to obtain by using a quantum algorithm for playing games. The kind of question we know how to address is again about the number of queries we require—how many of the leaves of the tree (labeled with bits) we need to examine to determine its value.

The best classical algorithm turns out to be very simple:

Algorithm *EVAL – TREE*: For the tree rooted at vertex v ,

1. Let u be the left or right child, chosen uniformly at random; run *EVAL – TREE*(u).
2. If v is an AND and *EVAL – TREE*(u) = 0 or v is an OR and *EVAL – TREE*(u) = 1, return 0 or 1, respectively.
3. Otherwise, if w is the other child, return *EVAL – TREE*(w).

The randomization is very important, since otherwise we might “get unlucky” at each level and need to evaluate every branch of the tree. It is actually an easy exercise to analyze the running time of this algorithm—what is more difficult to see is that this algorithm is actually *optimal*:

Theorem 2 (Saks-Wigderson ’86) $R(\text{AND} - \text{OR}) = \Theta(n^{\log \frac{1+\sqrt{33}}{4}})$. (where $\log \frac{1+\sqrt{33}}{4} \approx .754$)

This function is conjectured to exhibit the largest separation possible between classical query complexity and randomized query complexity (without promises). It is certainly, at least, the largest known separation. Of course, the caveat about promises is essential here too—if we are promised that the input has Hamming weight either at least $2/3n$ or at most $1/3n$ and we need to decide which, then deterministic algorithms need more than $n/3$ queries, but randomized algorithms only need one query.

In any case, the situation with randomized algorithms is similar to the one we faced with quantum algorithms. In the following, let R_ϵ denote the query complexity of a randomized algorithm that is allowed to err with probability ϵ . It is easy to see that for “Las Vegas” algorithms (i.e., $\epsilon = 0$, like our pruning algorithm), $R_0(f) \geq C(f)$, since the algorithm must see a certificate before it halts—otherwise, there’s both a 0-input and a 1-input that are consistent with the bits queried so far, so we would err on some input. Since we also saw $D(f) \leq C(f)^2$, we find that $D(f) \leq R_0(f)^2$ for all Boolean total functions f , but we don’t know whether or not this is tight—the $\log n$ -level AND-OR tree obtains the largest *known* speed-up. Likewise, for “Monte-Carlo” algorithms ($\epsilon > 0$), observe that if f has block sensitivity k , then we have to examine each of our disjoint blocks with

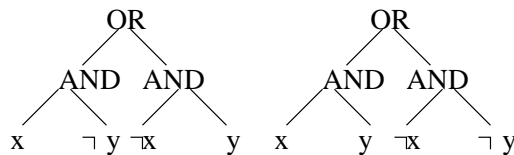


Figure 2: Two-level AND-OR trees computing $x \oplus y$ (left) and $\neg(x \oplus y)$ (right).

probability at least $1 - 2\epsilon$, since again otherwise there would be a 1-input and a 0-input that only differ in some unexamined block, and then even if we randomly output 0 or 1, we are still incorrect with probability greater than $(1/2)2\epsilon = \epsilon$. Since we showed $D(f) \leq \text{bs}(f)^3$, $D(f) = O(R_\epsilon(f)^3)$. Once again, the AND-OR tree obtains the largest known gap, and we don't know if this cubic gap is exhibited by any function. These facts, along with the application to game playing, are why this AND-OR tree is considered to be an extremely interesting example (the “fruit fly” of query complexity).

1.3 The quantum query complexity of the AND-OR tree

We now turn examining the quantum query complexity of this problem. We can get an easy quantum lower bound by a reduction from the parity problem. Observe that the parity of two bits x and y and its negation are computed by two-level binary AND-OR trees (see Figure 2). By recursively substituting these trees for the variables x and y , it is easy to see that we obtain the parity of n bits in $2 \log n$ levels—that is, in a tree with $n' = O(n^2)$ leaves. Given an instance of the parity problem, it is not hard to see how, given the path to a leaf, we could recursively decide which bit x_i of the input we should place at that leaf and whether or not that bit should be negated. Since we saw using the polynomial method that the parity of n bits required $n/2$ queries to compute, this AND-OR tree requires $\Omega(\sqrt{n'})$ queries to evaluate. (Since a more efficient query algorithm for evaluating the AND-OR tree would yield an impossibly query efficient algorithm for computing the parity of n bits.) Alternatively, it is possible to use Ambainis' adversary method to obtain the $\Omega(\sqrt{n})$ -lower bound as well.

We don't know how to obtain a better lower bound. It is also not easy to see how we can obtain an algorithm that is more efficient than our $O(n^{.753})$ -query classical algorithm—it isn't clear how to obtain an upper bound by applying, e.g., Grover's algorithm recursively to this problem since we only have subtrees of size two and moreover there is a recursive build-up of error at the $\omega(1)$ internal nodes. Despite this, in 2006, Farhi, Goldstone, and Gutmann found a $O(\sqrt{n})$ -query “Quantum walk” algorithm – sort of a sophisticated variant of Grover's algorithm – for evaluation of these AND-OR trees using intuitions from scattering theory and particle physics. (Contrary to our experience earlier in the course, this is an example where knowledge of physics turned out to be useful in the design of an algorithm.) Thus in fact, our easy $\Omega(\sqrt{n})$ lower bound turns out to be tight. Interestingly, this is an example of a function where the quantum case is simpler than the classical (randomized) case— $\Theta(\sqrt{n})$ versus $\Theta(n^{.754})$, where the classical lower bound was a highly nontrivial result.

2 The Collision Problem

We will conclude our unit on quantum query complexity with an interesting non-boolean problem, “the collision problem.” It is a problem that exhibits more structure than Grover’s “needle-in-a-haystack” problem, but less structure than the period finding problem, “interpolating” between the two (in some sense). Informally, rather than looking for a “needle in a haystack,” we are merely looking for two identical pieces of hay:

The Collision Problem. Given oracle access to a function $f : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ with n even (i.e., the oracle maps $|x\rangle|b\rangle$ to $|x\rangle|b \oplus f(x)\rangle$) promised that either f is one-to-one or two-to-one, decide which.

Variant: Promised that f is two-to-one, find a collision.

Clearly, the decision problem is easier than this search variant, since an algorithm for the search problem would provide a witness that a function is not one-to-one, i.e., that the function must be two-to-one given the promise (and of course, it would fail to find a collision in a one-to-one function).

Key difference: One way that this problem has “more structure” than Grover’s problem is that any one-to-one function must have distance at least $n/2$ from any two-to-one function. Thus, any fixed one-to-one function can be trivially distinguished from any fixed two-to-one function by querying a random location. (They agree with probability at most $1/2$.)

Obviously, the deterministic query complexity of this problem is (exactly) $n/2 + 1$, since if we are unlucky, we might see up to $n/2$ distinct elements when we query a two-to-one function. It is also not hard to see that the randomized query complexity is $\Theta(\sqrt{n})$ by the so-called “Birthday Paradox.” Similar to our analysis of Simon’s algorithm, suppose we choose a two-to-one function f uniformly at random. Then, for any fixed pair of distinct locations, x_i and x_j , the probability that $f(x_i) = f(x_j)$ is $\frac{1}{n-1}$, so we know that by a union bound, the total probability of seeing a collision in any k queries is $\binom{k}{2} \frac{1}{n-1}$. Thus, in particular, for $k(n) = o(\sqrt{n})$, the probability of seeing a collision is $o(1)$ and we can’t obtain a sufficiently small error probability. By contrast, for $k(n) = \Omega(\sqrt{n})$ queries to random locations in any two-to-one function, it is easy to see (using, e.g., Chebyshev’s inequality) that the expected number of collisions is $\Omega(1)$, so we can find a collision with constant probability, solving the search variant of the problem.

2.1 Motivation

This problem is interesting for a few reasons. First of all, graph isomorphism reduces to this problem. Fix a graph G , and consider the map $\sigma \mapsto \sigma(G)$ (for $\sigma \in S_n$). Notice that, for the graph $G \cup H$ (assuming G and H are rigid, i.e., only have a trivial automorphism), this map is two-to-one if G and H are isomorphic and it is one-to-one otherwise. So, one might wonder if there could be a $O(\log n)$ -query algorithm for the collision problem, since that would lead to a polynomial-time algorithm for the graph isomorphism problem—in particular, if we could find a random collision, we could remove the restriction on rigidity (we could use approximate counting). In any case, note that since the map here is on a domain of size $(2n)!$, we would need a $\text{poly} \log(2n!) \sim \text{poly}(n)$ -query algorithm for this application. That is, we wonder whether or not there is an efficient algorithm for graph isomorphism when we ignore the group structure.

There's also an application to breaking cryptographic (collision-resistant) hash functions, used in digital signatures on the internet, for example—the hash function is applied to some secret message (credit-card number, etc.) to create a “signature” of that message. Often, the security of a protocol depends on the assumption that it is computationally intractable to find a collision in the hash function – to find two messages m and m' that map to the same hash value – e.g., this might happen if the purpose of the hash value was to commit to a value without revealing it. (It may also be that finding a collision helps us find a stronger break in a hash function.)

There is a classical attack based on the “Birthday Paradox,” – the “birthday attack” – which proceeds by hashing random messages and storing their results until two messages are found that hash to the same value. For $N = 2^n$, it follows from the Birthday Paradox that this attack finds a collision in $O(\sqrt{N})$ evaluations with constant probability—a quadratic improvement over what one might naively expect. Again, if there were a poly log N -query quantum algorithm for this problem or, more generally, for finding collisions in any k -to-one function – we can assume that the collisions are evenly distributed, since this minimizes the total number of collision pairs, and any unbalanced distribution generally only makes an algorithm's task easier (this is not a proof, but in what follows, our upper bounds will work in the non-uniform case, and our lower bounds will apply in the uniform case, so we will have covered this, in any case) – that would yield a poly(n) time quantum algorithm for breaking any cryptographic hash function.

2.2 Algorithms for the collision problem

We now consider quantum algorithms for this problem. We have talked at length about how this problem has “more structure” than Grover's problem; how can we use the additional structure exhibited by this problem to find a better algorithm? We start by showing a different (easy) \sqrt{N} -query algorithm for finding collisions: we query $f(1)$, and we use Grover search to find the other index i such that $f(i) = f(1)$. We could also have used Grover's algorithm on all possible collision pairs (x_i, x_j) .

It is possible to combine this Grover search algorithm with our “Birthday” algorithm to obtain a $O(n^{1/3})$ -query algorithm as follows:

Algorithm. (Brassard-Høyer-Tapp 1997)

1. Make $n^{1/3}$ classical queries at random: $f(x_1), \dots, f(x_{n^{1/3}})$
2. Enter superposition over the remaining $n^{2/3}$ positions.
3. Apply Grover search to find an element in the initial list.

Grover's algorithm takes $O(n^{1/3})$ queries to find such an element in the final step, so this algorithm uses $O(n^{1/3})$ queries overall. (Clearly, the $n^{1/3}$ was chosen to optimize the trade-off we obtain.)

We know, by the correctness of Grover's algorithm, that this algorithm will work when there's a collision between the elements sampled in phase 1 and the elements in the superposition in phase 2. Thus, to see that this algorithm works, we only need to examine the probability of a collision. We observe that there are $n^{1/3}n^{2/3} = n$ pairs of phase-1 and phase-2 elements. A uniformly chosen pair of distinct elements would have probability $\frac{1}{n-1}$ of being a collision pair, but this doesn't quite apply here since our n pairs are not uniformly chosen—there are correlations. Nevertheless, the probability of a collision between a fixed pair in the two lists is still $\Omega(1/n)$ so we can apply an

analysis similar to the standard analysis of the Birthday Paradox to find that a collision exists with constant probability.

Is this optimal? It is challenging to find a super-constant lower bound for this problem: for example, it is hard to find a hybrid argument, since it is hard to interpolate between any one-to-one function and any two-to-one function (since they differ in so many places). Likewise, if we tried to apply our block sensitivity lower bound, we find that the block sensitivity of this problem is 2 (there are at most two disjoint ways of changing a one-to-one function to a two-to-one function), so our block sensitivity lower bound yields a mere $\Omega(\sqrt{2})$ -query lower bound.

There is another, more illuminating way of framing the difficulty: a quantum algorithm can “almost” find a collision pair in just a constant number of queries! Suppose we prepare a superposition $\frac{1}{\sqrt{n}} \sum_{x=1}^n |x\rangle|f(x)\rangle$ and measure the second register; this yields a superposition $\frac{|x\rangle+|y\rangle}{2}$ for a random pair x and y such that $f(x) = f(y)$ in the first register. Thus, if we could only measure this state twice, we could find a collision pair. (By contrast, it isn’t clear how measuring twice would allow one to solve the Grover problem in a constant number of queries.)

Next time, we’ll see a $\Omega(n^{1/5})$ -query lower bound (A. 2002) which was improved in subsequent weeks to $\Omega(n^{1/4})$ and then $\Omega(n^{1/3})$ by Yao and Shi, with some restrictions—only when the range of the function was much larger than n . These restrictions were later removed by Kutin and Ambainis, so the $O(n^{1/3})$ algorithm again turns out to be optimal.

2.3 The collision problem and “hidden variable” interpretations of quantum mechanics

An additional motivation for this problem (described in A. 2002) was studying the computational power of “hidden variable theory” interpretations of quantum mechanics. This school of thought says that a quantum superposition is at any time “really” in only one basis state. This basis state is called a “hidden variable” (although ironically, it is the one state that is not “hidden,” but rather is directly experienced). So, like in many-worlds interpretations, there is a quantum state with amplitudes for basis states for the many possibilities of states that the world could be in, but in contrast to the many-worlds interpretation, most of these states are just some guiding field in the background, and the world is actually in one distinguished state.

What significance does this have to quantum computing? One might think, “absolutely nothing,” since all of these interpretations make the same experimental predictions, and thus lead to the same computational model. But, if we could see a complete history of these true states, we could solve the collisions problem – and hence the graph isomorphism problem – in a constant number of queries. We could prepare a superposition over a collision pair as described earlier, and apply transformations (e.g., Hadamards) to “juggle” the true state between the basis states of the superposition. Thus, given a lower bound for the collision problem, we see that the additional information in this history provably gives additional computational power.

Lecture 13

*Lecturer: Scott Aaronson**Scribe: Kai-Min Chung*

Last time we finished quantum query complexity of boolean functions. Recall that the quantum query complexity of recursive AND-OR tree is $\Theta(\sqrt{n})$. We then studied the collision problem:

Collision Problem. Given oracle access to a function $f : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, decide whether f is 1-to-1 or 2-to-1, promised that one of those is the case.

Today, we'll prove the query lower bound for this problem, and discuss two applications of the lower bound. There are two motivations to study this question.

1 Motivations

Scott's original motivation is to understand the computation power of hidden variable theory. Imagine that nature knows not only the quantum state, but also the "real state" about what outcome would be if you perform a measurement, even if you do not look at it. So at any time, there is not only a superposition of particles, but also the "real place" where the particles are. If we believe that, then the real position of particles must follow some trajectory overtime. The trajectory is necessarily guided by the quantum state, because it has to agree with the prediction of the quantum mechanics. But in addition to that, it has to be a real particle trajectory. This is the kind of things you see in Bohm mechanics, a famous reformulation of quantum mechanics in 1950s to involve hidden variables.

The question Scott asked was: suppose you can see the entire trajectory of the hidden variable, what would that give you the power to do? One thing we can do with this ability is to solve the collision problem in a single query, and $O(\log n)$ computational steps. If f is 2-to-1, it is easy to prepare a superposition of a collision pairs $(1/\sqrt{2})(|x\rangle + |y\rangle)$, where $f(x) = f(y)$ (by going to a superposition of all input x 's, computing $f(x)$ in the superposition, and measuring the f register.) Now, if only we can range things so that we can see both x and y then we would find the collision and solve the collision problem. If we can see the entire trajectory of hidden variables, that is exactly the thing we can do. We can arrange so that the hidden variable starts being x , and varies to y and vice versa.

Thus, a lower bound on the quantum query complexity of collision problem proves a separation in the block box world between the ordinary quantum computing, and the quantum computing in the hidden variable theories.

However, a person on the street doesn't necessary care about this interpretation of quantum mechanics. On the other hand, people care more about graph isomorphism, and breaking cryptographic hash functions. These are the "real application" that you could do if you have a quantum algorithm for solving the collision problem. We are interested in rule out this possibility by lower bounds on the collision problem. This would imply, for example, to break the cryptographic hash functions, we have to exploit the structure of the hash functions somehow. Likewise, to solve graph isomorphism, we have to exploit the graph structure.

2 The Lower Bound

We have seen on Tuesday that there is a $O(n^{1/3})$ -query quantum algorithm for solving the collision problem. It turns out to be optimal. Today, we will show a lower bound $\Omega(n^{1/4})$.

We have mentioned that the difficult for proving the lower bound is that sort of all the approaches we saw before are based on the hardness of finding a single needle in a haystack. The underlying principle is that a quantum algorithm cannot monitor the change of many disjoint places. For example, the block sensitivity argument says that we need $\sqrt{(\# \text{ of blocks})}$ queries to detect the change of any single block. However, in this problem, the block sensitivity is only 2.

We use polynomial method to prove the lower bound. For every $x, h \in \{1, \dots, n\}$, define variables $\Delta_{x,h} = 1$ if $f(x) = h$, and $\Delta_{x,h} = 0$ otherwise. The acceptance probability of a quantum algorithm can be expressed as a polynomial over $\Delta_{x,h}$'s.

Lemma 1 *Let Q be a quantum algorithm makes T queries to f . Then Q 's acceptance probability is a degree $2T$ multi-linear polynomial $p(f)$ over variables $\Delta_{x,h}$'s. Furthermore, every monomial of $p(f)$ is of the form $\Delta_{x_1,h_1} \cdot \Delta_{x_2,h_2} \cdots \Delta_{x_d,h_d}$ with distinct x_i 's.*

Proof: (sketch) This is essentially the same as what we have seen before. Every amplitude can be written as a polynomial over $\Delta_{x,h}$. A unitary operation does not change the degree since it is linear. A query can be expressed as

$$\sum_{x,z} \alpha_{x,z} |x, z\rangle \mapsto \sum_{x,z} \alpha_{x,z} |x, z \oplus f(x)\rangle = \sum_{x,z} \left(\sum_h \alpha_{x,z \oplus h} \cdot \Delta_{x,h} \right) |x, z\rangle,$$

so increase the degree by at most 1. The factor 2 comes from squaring the amplitude to get the probability. Since every variable $\Delta_{x,h}$ is either 0 or 1, $\Delta_{x,h}^2 = \Delta_{x,h}$, and we can assume $p(f)$ is multi-linear. Furthermore, observe that for every x , there is only one h such that $\Delta_{x,h} = 1$, and other $\Delta_{x,h}$'s are 0, we can assume every monomial does not contain two variables with the same subscript x . \square

The next step is to reduce the number of variables. This time, we define $q(k) = \mathbb{E}_{f: k\text{-to-1}}[p(f)]$, where the notation means expected value over uniformly random k -to-1 function $f: \{1, \dots, n\} \rightarrow \{1, \dots, n\}$. We hope $q(k)$ to be a polynomial with small degree in k . However, a technicality is that n may not divisible by k , so there is no perfect k -to-1 functions. This is the main difficulty to apply the polynomial method. For now, we first make the unrealistic assumption that n is divisible by k for every k . We will come back to this technicality later.

Lemma 2 *Let $q(k) = \mathbb{E}_{f: k\text{-to-1}}[p(f)]$, where $p(f)$ is defined as above. Assume that n is divisible by k for every k , then $q(k)$ is a polynomial in k , and $\deg(q) \leq \deg(p) \leq 2T$.*

Proof: Let $p(f) = \sum_I \alpha_I I(f)$, where each $I(f) = \Delta_{x_1,h_1} \cdot \Delta_{x_2,h_2} \cdots \Delta_{x_d,h_d}$ is a monomial of f with distinct x_i 's. By linearity of expectation,

$$q(k) = \sum_I \alpha_I \mathbb{E}_{f: k\text{-to-1}}[I(f)].$$

Fix a $I(f) = \Delta_{x_1,h_1} \cdot \Delta_{x_2,h_2} \cdots \Delta_{x_d,h_d}$ with $d \leq \deg(p(f)) \leq 2T$, let us compute $\mathbb{E}_{f: k\text{-to-1}}[I(f)]$.

Observe that $I(f) = 1$ if $f(x_1) = h_1, \dots, f(x_d) = h_d$, and $I(f) = 0$ otherwise. $\mathbb{E}_{f : k\text{-to-1}}[I(f)]$ is the fraction of k -to-1 functions that satisfy the d constraints $f(x_i) = h_i$. The number of k -to-1 functions is

$$\binom{n}{n/k} \cdot \frac{n!}{(k!)^{n/k}}.$$

(We first decide the range of a function, and then decide how to map the domain to the range.) To compute the number of functions satisfying $f(x_i) = h_i$, we need the following variables. Let s be the number of distinct h values in $I(f)$. Let i_1, i_2, \dots, i_s be the number of each h in $I(f)$ (so $i_1 + \dots + i_s = d$). The number of functions satisfying $f(x_i) = h_i$ is

$$\binom{n-s}{(n/k)-s} \cdot \frac{(n-d)!}{(k!)^{n/k-s} \cdot (k-i_1)! \cdot \dots \cdot (k-i_s)!}.$$

(Again, the first factor is to pick the range of a function. Every h_j appears in $I(f)$ must in the range. The second factor is to decide the map from $n-d$ unfixed domain to the range.) Staring at the expression for a while, we have

$$\begin{aligned} f : k\text{-to-1} \mathbb{E}[I(f)] &= \frac{\binom{n-s}{(n/k)-s} \cdot (n-d)!}{(k!)^{n/k-s} \cdot (k-i_1)! \cdot \dots \cdot (k-i_s)!} \cdot \frac{(k!)^{n/k}}{\binom{n}{n/k} \cdot n!} \\ &= \frac{(n-s)! \cdot (n-d)!}{((n/k)-s)!(n-(n/k))!(k!)^{(n/k)-s} (k-i_1)! \cdot \dots \cdot (k-i_s)!} \cdot \frac{(n/k)!(n-(n/k))!(k!)^{n/k}}{n! \cdot n!} \\ &= r(n, d, s) \cdot \frac{(n/k)!(k!)^s}{((n/k)-s)!(k-i_1)! \cdot \dots \cdot (k-i_s)!} \quad \text{for some function } r \\ &= r(n, d, s) \cdot (n/k)((n/k)-1) \cdot \dots \cdot ((n/k)-s+1) \cdot \left(\prod_{j=1}^s k(k-1) \cdot \dots \cdot (k-i_j+1) \right) \\ &= r(n, d, s) \cdot (n)(n-k) \cdot \dots \cdot (n-(s-1)k) \cdot \left(\prod_{j=1}^s (k-1) \cdot \dots \cdot (k-i_j+1) \right) \end{aligned}$$

is a polynomial over k of degree $(s-1) + (i_1-1) + \dots + (i_s-1) = i_1 + \dots + i_s - 1 = d-1$. Therefore,

$$q(k) = \sum_I \alpha_I \mathbb{E}_{f : k\text{-to-1}}[I(f)]$$

is a polynomial over k of degree at most $2T$. □

Now, suppose $p(f)$ is the acceptance probability of a quantum algorithm solving the collision problem, then

$$0 \leq q(1) = \mathbb{E}_{f : 1\text{-to-1}}[p(f)] \leq 1/3, \text{ and } 2/3 \leq q(2) = \mathbb{E}_{f : 2\text{-to-1}}[p(f)] \leq 1.$$

For $k \geq 2$, since $p(f)$ represents a probability, we have

$$0 \leq q(k) = \mathbb{E}_{f : k\text{-to-1}}[p(f)] \leq 1.$$

We can then apply Markov inequality as in Lecture 10 to lower bound the degree of $q(k)$, which gives query lower bound on Q . However, recall the technicality that n may not be divisible by k . The price to resolve this technicality is that the above argument is only valid for small k . (Intuitively, the divisibility problem produces some error, which will hurt us when k is too large.) Originally, Scott's paper got $\Omega(n^{1/5})$. This was subsequently improved to $\Omega(n^{1/4})$. Yaoyun Shi gave a sophisticated way to resolve it, and achieved tight lower bound $\Omega(n^{1/4})$. We do not go through the detail in this lecture, but only intuitively argue that the above is valid up to $k = \Omega(n^{1/2})$, and so we can get a lower bound $\Omega(n^{1/4})$. The following argument should be able to turn into a rigorous proof.

When n is not divisible by k , there is no perfect k -to-1 functions, but there are almost k -to-1 functions. Intuitively, the difference is only on the left-over $(n \bmod k) < k$ inputs. Recall the intuition from the Grover lower bound that a quantum algorithm needs \sqrt{n}/s queries to notice a change of f on s inputs. Therefore, when $k = O(n^{1/2})$, a quantum algorithm with $O(n^{1/4})$ query can not notice this difference.

Theorem 3 *Any quantum algorithm for the collision problem needs to make $\Omega(n^{1/4})$ queries. I.e., $Q(\text{COLLISION}) = \Omega(n^{1/4})$.*

Proof: Let Q be a quantum algorithm for the collision problem making T queries to f , and the corresponding $p(f)$ and $q(k)$ be defined as above. From the above argument, we have $\deg(q) \leq 2T$, $0 \leq q(1) \leq 1/3$, $2/3 \leq q(2) \leq 1$, and $0 \leq q(k) \leq 1$ for $k = 3, 4, \dots, \Omega(n^{1/2})$. Now, we are in the same situation as Lecture 10. We have $q'(k) \geq 1/3$ for some $k \in [1, 2]$, and the Markov inequality says

$$\deg(q) \geq \sqrt{\frac{\text{Length} \cdot \text{MaxDeriv}}{\text{Height}}} \geq \sqrt{\frac{\Omega(n^{1/2}) \cdot 1/3}{O(1)}} = \Omega(n^{1/4}).$$

Therefore, the number of query $T = \Omega(n^{1/4})$. □

We next discuss some applications of this lower bound of collision problem.

3 Implication to Element Distinctness Problem

The first application is to the query lower bound of element distinctness problem. We start by the problem definition.

Element Distinctness Problem. Given oracle access to x_1, x_2, \dots, x_n , decide whether there exists $i \neq j$ such that $x_i = x_j$.

Compare to the collision problem, this problem has much less structure to exploit. There might be only one collision pair, instead of many pairs. What is the query complexity of the element distinctness problem? Let us consider the upper bound first. A good thing to try is to apply Grover's algorithm, and see what we get. If we apply Grover in a naive way, there are n^2 pairs, and we need $\sqrt{n^2} = n$ queries to find collision pairs. Can we use Grover in a smarter way to do better?

The idea is a bit tricky when you see it first time. Let $U = \{x_1, \dots, x_n\}$. Consider the following algorithm.

1. Randomly pick \sqrt{n} elements $S \subset U$, and query those elements classically.

2. If we find collision in S , output the collision.
3. Use Grover's algorithm to find collisions between S and $U - S$.

What is the query complexity of the algorithm? The first step makes \sqrt{n} queries. The third step uses Grover to search $x_j \in U - S$ such that $x_j = x_i$ for some $x_i \in S$. This uses $O(\sqrt{n})$ queries. In total, the algorithm uses $O(\sqrt{n})$ queries.

What is the success probability of this algorithm? Suppose for the worst case, there is only one collision pair $x_i = x_j$. For the algorithm to be able to find the collision, either x_i or x_j needs to be picked in S in the first step. This happens with probability $\Omega(1/\sqrt{n})$. When this happens, the algorithm can find the collision with constant probability in Step 2 or 3. Therefore, the overall success probability is $\Omega(1/\sqrt{n})$.

Now, suppose we invoke the above algorithm $O(\sqrt{n})$ times, we can see a collision with high probability. Doing so naively requires $O(\sqrt{n}) \cdot O(\sqrt{n}) = O(n)$ queries. However, we can use another Grover on top of the $O(\sqrt{n})$ invocation of the algorithm to find a success invocation. Searching over $O(\sqrt{n})$ items only requires $O(\sqrt{\sqrt{n}})$ queries. Overall, the two layers Grover only requires $O(n^{1/4}) \cdot O(\sqrt{n}) = O(n^{3/4})$.

On the other hand, what is the lower bound? $\Omega(\sqrt{n})$ lower bound is easy to observe. Suppose there is only one collision $x_i = x_j$, and we know x_i at beginning, then the task reduce to find x_j . This is exactly the Grover's problem. Therefore, Grover's lower bound gives $\Omega(\sqrt{n})$ lower bound for the element distinctness problem. Can we do better?

By applying the collision lower bound, we can improve the lower bound to $\Omega(n^{2/3})$. Consider the contrapositive, if we can solve the element distinctness problem in $o(n^{2/3})$ queries, can we solve the collision problem in $o(n^{1/3})$ queries? The answer is yes: to solve the collision problem, we randomly pick $O(\sqrt{n})$ elements, and run the element distinctness algorithm on those $O(\sqrt{n})$ elements. If f is 2-to-1, by birthday paradox, we can see a collision with constant probability, and so it reduces to the element distinctness problem on $O(\sqrt{n})$ elements. Thus, the above reduction solves the collision problem in $o(n^{1/3})$ queries, a contradiction.

Now, we have $O(n^{3/4})$ upper bound, and $\Omega(n^{2/3})$ lower bound. People are always interested in closing the gap. It turns out that the lower bound is tight. Ambainis gave a sophisticated $O(n^{2/3})$ -query algorithm for the element distinctness problem based on quantum walks.

4 Oracle Separation to $\mathbf{SZK} \not\subseteq \mathbf{BQP}$

Another application of the collision lower bound is an oracle relative to which $\mathbf{SZK} \not\subseteq \mathbf{BQP}$. What is \mathbf{SZK} ? It stands for statistical zero knowledge, which means a protocol where a verifier interact with a prover, and the result of this interaction is that the verifier become convinced that some statement is true, but he does not learn anything else. In particular, the verifier does not gain the ability to convince anyone else that the statement is true. It sounds paradoxical at beginning, but there is a canonical example to illustrate the point. Let us consider the graph non-isomorphism problem.

Graph Non-isomorphism Problem. Given two graphs G and H , decide whether G and H are not isomorphic. (The answer is yes if G and H are not isomorphic.)

Let us introduce two characters. The verifier Arthur is a skeptical polynomial time king, and the prover Merlin is an omniscient wizard, but not trustworthy. Merlin wants to convince Arthur

that G and H are not isomorphic, but does not want Arthur to learn anything else. Arthur has to test Merlin to discover whether it is the case. How should we design the protocol?

The idea is what called the Coke-Pepsi test. Someone claims that Coke and Pepsi are different but does not able to pin point the difference. How can he convince you? One way to do it is to perform a blind test to see if he can tell them apart or not. Apply this idea, we can do the following.

- Arthur randomly picks one graph, permutes the vertices, send the result to Merlin, and ask, “which graph did I start with?”
- Merlin answers Arthur’s challenge.

If G and H are not isomorphic, then any permutations of G are different to any permutations of H . Since Merlin is an omniscient wizard, he can always answer the challenge. On the other hand, suppose G and H are isomorphic, then the set of permutation of G are exactly the same as the set of permutation of H . Thus, Merlin has no way of knowing the answer, and can only guess correctly with probability $1/2$. We can run this protocol several times to make the probability as lower as we want.

Can Arthur learn anything from the protocol? Let us argue this intuitively. First of all, Arthur has the graphs at beginning, so he does not learn anything new from the graphs. Does he learn anything from Merlin? Merlin tells him which graph he started with. But Arthur already knew that. So, intuitively, Arthur has not learnt anything new, and yet, he is convinced that G and H are not isomorphic if it is the case. This is what we mean by zero knowledge proof.

The way to formalize the concept of not learning anything is that, Arthur should be able to simulate the entire interaction with Merlin, without even bringing Merlin into the picture at all. Note that this is the case for our example, Arthur just need to answer the challenge he produced by himself, who apparently knows the answer.

We will not give the formal definition of **SZK**. Basically, **SZK** is the class of problem for which yes answer can be proved by a protocol like this one. That is, there is a protocol for which a polynomial time Arthur interacts with computationally unbounded Merlin that satisfies three properties:

- Completeness: Arthur should accept when the answer is yes.
- Soundness: Arthur should reject with high probability whenever the answer is no.
- Zero Knowledge: Arthur should not learn anything beyond the answer.

We saw before that Grover lower bound gives an oracle relative to which $\mathbf{NP} \not\subseteq \mathbf{BQP}$. We claim that the collision lower bound gives an oracle relative to which $\mathbf{SZK} \not\subseteq \mathbf{BQP}$. The oracle encode the collision problem. The collision lower bound says that there is no efficient **BQP** algorithm to distinguish 1-to-1 functions from 2-to-1 functions. We need to show that this problem has a **SZK** protocol. The question is, can Merlin convince Arthur f is an 1-to-1 function without giving any other information? The idea is as follows.

- Arthur randomly picks an x , computes $f(x)$, send it to Merlin, and ask, “what is the inverse x ?”
- Merlin answers Arthur’s challenge.

If f is 1-to-1, then Merlin can always answer the challenge. If f is 2-to-1, then Merlin can only succeed with probability $1/2$. Can Arthur learn anything? Merlin tells Arthur the inverse x of $f(x)$, which Arthur already knew it if he follows the protocol. In this case, Arthur learns nothing. However, Arthur may not be honest, and sends Merlin some y that he is interested in. He can then learn the inverse of y from Merlin. Intuitively, this seems not zero knowledge. Indeed, the protocol is zero knowledge only when Arthur is honest. However, there is a powerful theorem says that there is a way to modify the protocol so that it is zero knowledge even if Arthur is dishonest. Therefore, the collision problem has statistical zero knowledge protocol. By standard diagonalization trick, we can get an oracle relative to which $\mathbf{SZK} \not\subseteq \mathbf{BQP}$.

Like \mathbf{BQP} vs. \mathbf{NP} , we do not know the relation between \mathbf{SZK} and \mathbf{NP} . But under some hypothesis that most people believed in, \mathbf{SZK} is contained in $\mathbf{NP} \cap \mathbf{coNP}$. Thus, we do not believe \mathbf{NP} -complete problems have statistical zero knowledge proof protocols. On the other hand, they have computational zero knowledge proof protocols, assuming cryptographic assumption. If Merlin could encode his proof to Arthur cryptographically, and then selectively decode parts of the answer, then there is a computational zero knowledge protocols for \mathbf{NP} -complete problems, due to Goldreich, Micali, Wigderson.

As an interesting side note, we mentioned that there is a whole industry of taking things in classical computation, putting a letter Q in from of them. This is no exception. There is a whole literature now about quantum statistical zero knowledge. Recall that honest verifier and dishonest verifier are equivalent for statistical zero knowledge (and also for computational zero knowledge.) Interestingly, in the quantum world, \mathbf{NP} -complete problems have honest verifier quantum statistical zero knowledge protocol, but apparently not with dishonest verifier.

Lecture 13

*Lecturer: Scott Aaronson**Scribe: Alex Arkhipov*

1. LAST TIME

Last class, we finished the section on collision lower bounds and constructed an oracle relative to which SKZ is not contained in BQP.

Here's something extra from last class. It's tantalizingly easy to get a state $\frac{|x\rangle+|y\rangle}{\sqrt{2}}$, where x and y are a collision pair ($f(x) = f(y)$). If only we could measure twice, we might find out both elements of the collision pair! You can do this in Dynamical Quantum Polynomial Time (DQP), which extends BQP by letting you see the whole trajectory of a quantum state, in effect letting you measure multiple times. We know that DQP contains SZK, the class of statistical zero-knowledge proofs. There exists an oracle relative to which NP is not in DQP.

But can we use the power of DQP to speed up black-box search? Yes, but not by much. We can improve Grover's Algorithm by measuring after each application of the Grover Diffusion Operator. This process finds almost surely finds the marked item in $O(N^{1/3})$ applications rather than the $O(N^{1/2})$ of Grover. This has been proven optimal.

Now we return to quantum complexity land.

2. PLACING BQP

So, where exactly does BQP lie within the classical complexity classes? There's a lot we don't know, such as whether BQP belongs inside NP. There's an even bigger question: How does BQP relate to the polynomial hierarchy (PH)? Whether there's an oracle relative to which BQP is not in PH remains an open problem. We've found oracle problems for which quantum algorithms have exponentially outpaced classical ones, such as Simon's problem and Shor's period-finding problem, but almost all of these lie in NP.

Here's a candidate for an oracle problem in BQP, but above the polynomial hierarchy.

Fourier Checking: Given oracles for two Boolean functions $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$ and $g : \{-1, 1\}^n \rightarrow \{-1, 1\}$, decide whether

- Both f and g are uniformly random.
- f is random, and $g = \text{sgn}(\hat{f})$, where $\hat{f} = \sum_y (-1)^{x \cdot y} f(y)$ is the Fourier Transform of f .

given that one of these is the case.

This problem is in BQP, since it can be solved with a quantum circuit:

In the first case where f and g are random, the final measurements give a random value.

If f and g satisfy the second promise, one can show that starting from a uniform distribution and applying f , then Hadamard gates to each qubit, then g , gives a nearly uniform distribution. So, the final set of Hadamard gates, will give a nearly all-zero state in this case.

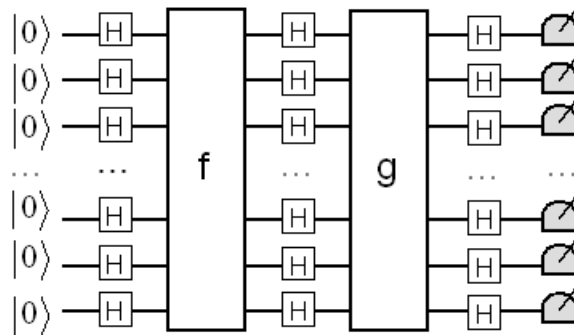


Figure 2.1. A quantum circuit for Fourier Checking

We conjecture that Fourier Checking is not in PH. It definitely does not seem to be in NP, since the relationship between f and g is a global property of the functions, so it would seem to be hard to demonstrate it in a certificate of polynomial size.

3. MA AND AM

We don't know how BQP compares to NP, so let's make a quantum version of NP.

As a first step, we make a probabilistic version of NP, which we'll call MA, for Merlin-Arthur.

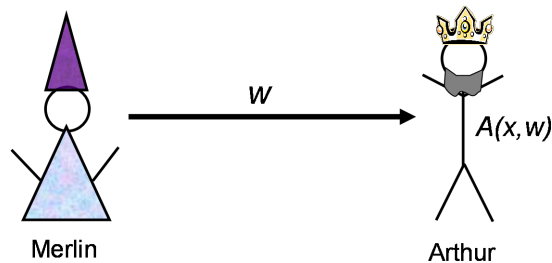


Figure 3.1. The Merlin-Arthur Protocol

The imaginative naming is from Babai. Merlin, an omniscient wizard with unbounded computation resources, wants to prove something to the king Arthur, who only has polynomial time to check. Merlin sends Arthur a certificate to try to convince Arthur to accept. Arthur can't be too gullible, since Merlin is devious and will try to get him to accept false statements as well as true ones. Formally,

Definition. MA is the set of languages $L \subset \{0, 1\}^*$ for which there exist a Turing Machine A in probabilistic polynomial time such that for all inputs x

- If $x \in L$, then there exists a polynomial-sized certificate w so that $A(x, w)$ accepts with probability at least $2/3$.
- If $x \notin L$, then for any polynomial-sized certificate w so that $A(x, w)$ accepts with probability at most $1/3$.

The probabilities are being taken over some random bits A has access to.

It makes not difference if we change the $2/3$ in the definition to a 1 ; this is a nontrivial theorem. However, we can't make it so that Arthur is never duped: If the $1/3$ in the definition is changed to a 0 , we get NP.

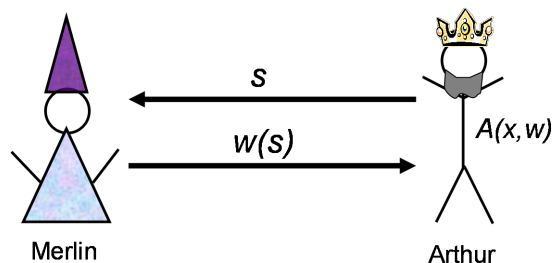


Figure 3.2. The Arthur-Merlin Protocol

Merlin-Arthur has a cousin, called Arthur-Merlin (AM), in which first Arthur sends a challenge string s to Merlin, then Merlin responds to the challenge. Since it gives an extra turn for Arthur to act, AM contains MA. It's also known that $SKZ \subseteq AM$.

Specifically, this means that the Graph Non-Isomorphism (GNI) problem is in AM. We can solve GNI with this AM protocol: Arthur sends Merlin challenge graphs, each a permutation of one of two graphs he's testing, and if Merlin can reliably say which graph was which, Arthur should be convinced that the graphs are not isomorphic. We don't know if $GNI \in MA$.

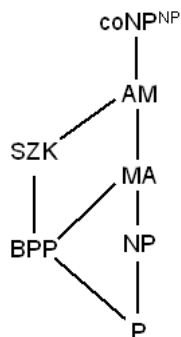


Figure 3.3. An inclusion diagram containing MA and AM

We conjecture that we can derandomize the random algorithms, which will collapse $P=BPP$ and $NP=MA=AM$. Strong circuit bounds would suffice to do this.

4. QMA

To make a quantum analogue of MA, there are two parts we can “quantize”: the verifier algorithm and the certificate. Making both quantum gives the class called Quantum Merlin-Arthur (QMA).

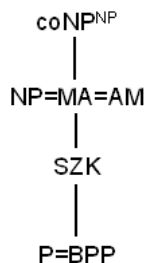


Figure 3.4. What the tree would look like if derandomization succeeds.

Definition. **QMA** is the set of languages $L \subset \{0,1\}^*$ for which there exist a polynomial-time quantum circuit A such that for all inputs x

- If $x \in L$, then there exists a witness state $|\varphi\rangle$ so that $A(x, |\varphi\rangle)$ accepts with probability at least $2/3$.
- If $x \notin L$, then there for any witness state $|\varphi\rangle$, $A(x, |\varphi\rangle)$ accepts with probability at most $1/3$.

If Arthur has the power of a quantum circuit, but the witness must be classical, we get the oxymoronically named Quantum Classical Merlin Arthur (QCMA).

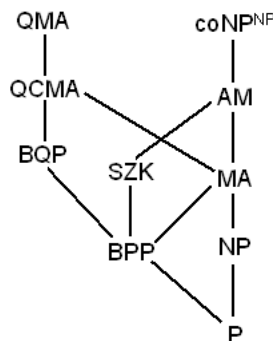


Figure 4.1. Placing BQP, QMA, and QCMA among classical complexity classes

5. AMPLIFYING QMA

Unlike for MA, in QMA we don't know if we can make the accept rate $2/3$ into a 1. One thing we'd like to check is that our choice of constants $1/3$ and $2/3$ is not crucial for QMA; we don't want the definition to hang on arbitrary numbers. We'll check that Arthur can amplify to any probability he wants.

If Arthur could just copy the witness state $|\varphi\rangle$ a bunch of times, Arthur could amplify by running the verifier algorithm once on each witness, and take the majority result. But unknown states can't be cloned, so instead we'll have Merlin send the requisite copies of $|\varphi\rangle$. But can we trust Merlin not to do something tricky?

What if Merlin secretly cheats by sending a bunch of different states $|\varphi_1\rangle, |\varphi_2\rangle, \dots, |\varphi_n\rangle$? Then, each of them has its own probability p_i of Arthur accepting, and Merlin can

do at least as well at making Arthur accept if he just send copies of the $|\varphi_i\rangle$ with the largest p_i .

What if Merlin keeps some qubits that are entangled with the witness states he sends? Then, from Arthur's view, Merlin has sent mixed states, so Arthur's probability of accepting is a weighted mean of the probability of accepting each component pure state. So, again, Merlin would be at least as well off sending the pure part with highest acceptance probability.

Now what if Merlin sends witness states that are entangled with each other? By convexity, some pure state $|\varphi\rangle$ maximizes the probability that Arthur accepts. Even after Arthur has performed an experiment on a register, Merlin can hope for nothing better than for the next register Arthur acts on to contain $|\varphi\rangle$. But then, why not just put the unentangled state $|\varphi\rangle$ there? So, entangling registers doesn't win Merlin anything.

6. GROUP NON-MEMBERSHIP

So, what can we actually do with QMA? One seemingly hard problem in QMA is Group Non-Membership.

We are given a black box group G . What this means is that each group element x has an arbitrary and distinct code string S_x , and we have oracles that implement the multiplication and inverse operations for G on these code strings. For notation, we'll just refer to the elements as themselves, rather than their code strings.

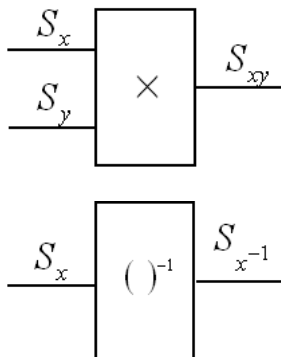


Figure 6.1. Black boxes for some group

Group Membership Problem: Given a subgroup H of G (as a list of generators), and an $x \in G$, decide if $x \in H$.

Note that G and H may have exponential size, but we can always give a polynomial set of generators for G and H .

Is this problem in NP ? Well, we can certainly show that $x \in H$ by giving a product of generators and their inverses to make x , which the verifier can check. But this might be exponential in length. For example, if H is the cyclic group of order 2^n , given by generator $\{1\}$, and $x = 2^{n-1}$, then reaching x takes 2^{n-1} operations. We can remedy this by using square-and-multiply. If we can save already-formed group elements and reuse them, any element can be formed in polynomial time.

What about Group Non-Membership, checking if $x \notin H$? This is trickier.

Here's an idea for to solve this in QMA. Have Merlin send Arthur the state

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

Here's a question: Could Arthur have just made the state himself? It's known that Arthur can pick a random element of H by himself: If Arthur randomly takes two elements from the set of generators, multiplies them with the oracle, and adds the result to the set, then with a polynomial number of repetitions, he'll get an element almost uniformly at random.

So then why can't Arthur make $|\varphi\rangle$ himself? Well, a probability distribution is not the same thing as quantum state. If Arthur runs the random algorithm on a quantum computer, he'll get

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle |garbage_h\rangle$$

where the garbage can from the process of computing $|h\rangle$. There's no obvious way to erase this workspace.

Next class, we'll prove that this verifier state $|\varphi\rangle$ can convince Arthur whether $x \notin H$.

Lecture 15

*Lecturer: Scott Aaronson**Scribe: Jessica Yuan*

Today we continue to work with the class QMA.

1 Last Time

We defined QMA, the class of languages $L \subseteq \{0,1\}^*$ for which \exists a polytime Quantum Turing machine Q , polynomial p such that $\forall x \in \{0,1\}^n$:

$x \in L \rightarrow \exists p(n)$ -qubit witness $|\phi\rangle$ such that $Q(x, |\phi\rangle)$ accepts with probability $\geq \frac{2}{3}$

$x \notin L \rightarrow \forall |\phi\rangle, Q(x, |\phi\rangle)$ accepts with probability $\leq \frac{1}{3}$

The current known complexity inclusion relations are:

$$P \subseteq NP \subseteq MA \subseteq QCMA \subseteq QMA \quad (1)$$

$$P \subseteq BPP \subseteq BQP \subseteq QCMA \quad (2)$$

$$BPP \subseteq SZK \subseteq AM \quad (3)$$

$$BPP \subseteq MA \subseteq AM \quad (4)$$

2 Watrous' QMA protocol for Group Non-Membership

We started on Watrous' QMA protocol for Group Non-Membership. Given a black-group G , a subgroup $H \subseteq G$ (specified by its generators), and an element $x \in G$, the Group Non-Membership problem asks, is $x \in H$?

Being given a quantum superposition is quite different from being given a classical probability distribution. The QCMA vs. QMA question gets at just how different they are.

When we last left him, Arthur was given a superposition $|H\rangle$ by Merlin

$$|H\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle \quad (5)$$

where $|H\rangle$ is not necessarily something Arthur could prepare in polynomial time himself, even if Merlin gave him a classical description of $|H\rangle$. Babai and Szemeredi showed how to sample a random element from the subgroup H in polynomial time, but that's *not the same* as being able to prepare a superposition over the elements.

Given $|H\rangle$, what can Arthur do? First of all, he doesn't actually know whether Merlin gave him $|H\rangle$ or whether he cheated and gave him some other state. But let's assume for the time being that Merlin gave him $|H\rangle$. How could he use that state to test whether $x \in H$?

Arthur computes $|Hx\rangle$, where

$$|Hx\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |hx\rangle \quad (6)$$

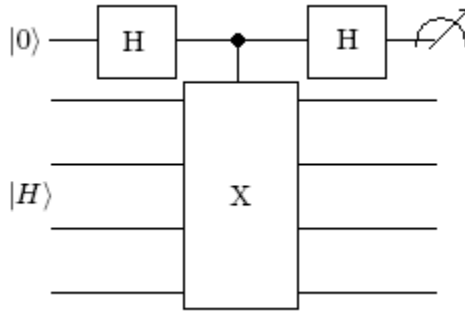


Figure 1: Circuit Diagram

After the above operation, the state becomes $|0\rangle|H\rangle + |1\rangle|Hx\rangle$. Arthur then Hadamards the first qubit and measures.

If $x \in H$, $|H\rangle = |Hx\rangle$ and $\langle H|Hx\rangle = 1$. So the state is $|0\rangle|H\rangle + |1\rangle|H\rangle = (|0\rangle + |1\rangle)|H\rangle$ and Hadamarding the first qubit always yields $|0\rangle$, because the $|0\rangle$ and $|1\rangle$ branches interfere with each other.

If $x \notin H$, Hx is disjoint from H , so $|H\rangle$ and $|Hx\rangle$ are orthogonal, and $\langle H|Hx\rangle = 0$. When Arthur prepares $|0\rangle|H\rangle + |1\rangle|Hx\rangle$, it's as if he's measured the first qubit. So when he Hadamards the first qubit and measures, he gets $|1\rangle$ with probability $\frac{1}{2}$.

However, one issue remains unaddressed: how can Arthur check that Merlin actually gave him $|H\rangle$ and not some other state? This is slightly tricky. Here's what Arthur can do. First, generate a (nearly) random element y of H (recall that Babai and Szemeredi showed that he can do this in polynomial time without help from Merlin). Then, letting $|H'\rangle$ be whatever state Merlin sent, Arthur prepares

$$|0\rangle|H'\rangle + |1\rangle|H'y\rangle \quad (7)$$

Arthur then Hadamards the first qubit and measures. Alternatively, Arthur could prepare a nearly random element y of G , then prepare

$$|0\rangle|H'\rangle + |1\rangle|H'y\rangle \quad (8)$$

then Hadamard the first qubit and measure. One can show that if these tests give the expected results, then $|H'\rangle$ is either equal to $|H\rangle$ or else it's some other state that works just as well for Arthur's intended purpose. (To amplify the error probability, as usual, we can have Merlin send Arthur lots of copies of $|H\rangle$. Then Arthur can choose a different test to run on each copy.)

3 Upper bounds on QMA

It's easy to establish $\text{QMA} \subseteq \text{NEXP}$ as an upper bound: Arthur can simulate all the witness states Merlin could send to him.

Is QMA in EXP? It is, and we can show that.

We can think of what Arthur does as just performing a *measurement* on the witness state $|\phi\rangle$: a measurement that tells him whether to accept or reject. Arthur accepts iff the first qubit of

$$U(|\phi\rangle|0\dots 0\rangle) \quad (9)$$

is measured to be $|1\rangle$, for some unitary U . But what does this mean, really?

$$|\phi\rangle = \alpha_1|1\rangle + \dots + \alpha_N|N\rangle \quad (10)$$

$$U|\phi\rangle =$$

$$\begin{pmatrix} U_1 & 0\dots 0 \\ U_2 & 0\dots 0 \\ \dots & \dots \\ U_N & 0\dots 0 \\ U_{N+1} & 0\dots 0 \\ \dots & \dots \\ U_{2N} & 0\dots 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_N \\ 0 \\ \dots \\ 0 \end{pmatrix}$$

Arthur's acceptance probability thus equals

$$\sum_{i=1}^M |\langle U_i|\phi\rangle|^2 \quad (11)$$

$$= \sum_{i=1}^M \langle \phi|U_i\rangle \langle U_i|\phi\rangle \quad (12)$$

$$= \langle \phi| \left(\sum_{i=1}^M |U_i\rangle \langle U_i| \right) |\phi\rangle \quad (13)$$

where $i = 1, \dots, M$ are the indices where ϕ_i are states that Arthur accepts.

Equivalently, there's a positive semidefinite matrix A such that this probability equals $\langle \phi|A|\phi\rangle$. This value is then just the largest eigenvalue of A . To maximize this value, we want ϕ to be the principle eigenvector, the eigenvector corresponding to this largest eigenvalue.

Thus, the problem reduces to this: given an exponentially-large matrix A for which we can compute each entry in exponential time, is $\lambda_{\max}(A) \geq \frac{2}{3}$ or is $\lambda_{\max}(A) \leq \frac{1}{3}$, promised that one of these is the case? We can solve this problem in EXP because it's just an exponential-size linear algebra problem.

Can we do better than that? We can. Let's solve it in PP.

Note that $\text{Tr}(A) = \sum_i \lambda_i$. Thus, if we're dealing with an n -qubit witness state,

$$(\lambda_{\max})^d \leq \text{Tr}(A^d) = \sum_i \lambda_i^d \leq 2^n (\lambda_{\max})^d \quad (14)$$

So $\lambda_{\max} \leq \frac{1}{3}$ implies $\text{Tr}(A^d) \leq 2^n \left(\frac{1}{3}\right)^d$.

$\lambda_{\max} \geq \frac{2}{3}$ implies $\text{Tr}(A^d) \geq \left(\frac{2}{3}\right)^d$

Hence, by computing $\text{Tr}(A^d)$, we can estimate λ_{\max} . But $\text{Tr}(A^d)$ is just a degree- d polynomial in the entries of A . It can be estimated in #P. Deciding whether it's above or below some threshold is a PP problem.

4 QMA-complete problems

Arguably, the single most famous discovery in theoretical computer science was NP-completeness: the fact that so many different optimization problems not only seem difficult, but capture the entire difficulty of the class NP. So it's natural to wonder: is there a theory of quantum NP-completeness, analogous to the classical theory? It turns out there is, and it was created by Alexei Kitaev in 1999.

First, let's review classical NP-completeness. A problem is NP-complete if it's in NP and every other NP problem can be reduced to it. It's clear, almost by definition, that NP-complete problems exist. For example, "given a polytime Turing machine M , is there an input that causes M to accept?"

So the great discovery was not that NP-complete problems exist, but that many natural problems are NP-complete. The one that started the ball rolling was 3SAT, the problem of whether a boolean function of the AND of OR clauses of three variables each can be satisfied. We can prove 3SAT is NP-complete by taking Circuit-SAT (the problem of whether a boolean circuit has a satisfying assignment) and creating a 3CNF clause to check for each AND/OR/NOT gate to see whether it computed correctly.

What's a problem that's QMA-complete essentially by definition? It will have to be a promise problem:

Given a polytime Quantum Turing machine Q , is $\max_{|\phi\rangle} \Pr[Q(|\phi\rangle) \text{ accepts}] \geq \frac{2}{3}$ or $\leq \frac{1}{3}$, promised that one is the case?

A more "natural" QMA-complete problem is the problem of k -local Hamiltonians:

Given 2-outcome measurements E_1, \dots, E_M , each of which acts on at most k qubits (out of n), does there exist an n -qubit state $|\phi\rangle$ such that $\sum_{i=1}^M \Pr[E_i(|\phi\rangle) \text{ accepts}] \geq b$, promised that this sum is either $\geq b$ or $\leq a$ where b and a differ by $b - a = \Omega(\frac{1}{\text{poly}(n)})$?

The Cook-Levin Theorem proved that the boolean satisfiability problem SAT is NP-complete by simulating a Turing machine and checking to see if the simulation was run properly. Can we do this to prove QMA-completeness of the local Hamiltonians problem? It turns out we can't, and we'll see why next time.

Lecture 16

Lecturer: Scott Aaronson

Scribe: Jing Chen

1 Recap

Last time we introduced the complexity class QMA (quantum Merlin-Arthur), which is a quantum version for NP. In particular, we have seen Watrous's Group Non-Membership (GNM) protocol which enables a quantum Merlin to prove to a quantum Arthur that some element x of a group G is not a member of H , a subgroup of G . Notice that this is a problem that people do not know how to solve in classical world. To understand the limit of QMA , we have proved that $QMA \subseteq PP$ (and also $QMA \subseteq PSPACE$).

We have also talked about QMA -complete problems, which are the quantum version of NP-complete problem. In particular we have introduced the Local Hamiltonians problem, which is the quantum version of the SAT problem. We also have a quantum version of the Cook-Levin theorem, due to Kitaev, saying that the Local Hamiltonians problem is QMA complete.

We will prove Kitaev's theorem in this lecture.

2 Local Hamiltonians is QMA-complete

Definition 1 (Local Hamiltonians Problem) *Given m measurements E_1, \dots, E_m each of which acts on at most k (out of n) qubits where k is a constant, the Local Hamiltonians problem is to decide which of the following two statements is true, promised that one is true:*

1. \exists an n -qubit state $|\varphi\rangle$ such that $\sum_{i=1}^m \Pr[E_i \text{ accepts } |\varphi\rangle] \geq b$; or
2. \forall n -qubit state $|\varphi\rangle$, $\sum_{i=1}^m \Pr[E_i \text{ accepts } |\varphi\rangle] \leq a$.

(Here $b - a > \frac{1}{p(n)}$ for some polynomial $p(\cdot)$.)

Theorem 1 (Kitaev) *The Local Hamiltonians problem is QMA-complete (as a promised problem).*

To prove this theorem, first of all, it is easy to see that the Local Hamiltonians problem is indeed in QMA , because to prove that there exists a state making all measurements accept, Merlin can simply send $|\psi\rangle = \max_{|\varphi\rangle} \sum_{i=1}^m \Pr[E_i \text{ accepts } |\varphi\rangle]$ as a witness.

Now we want to prove that every QMA problem can be reduced to the Local Hamiltonians problem.

The first trial would be to start from the proof that 3SAT is NP-complete —the famous Cook-Levin Theorem we talked about last time— and just put the word “quantum” in front of everything. So what happens if we do this? Let's say Arthur has asked Merlin to send him a computation tableau $|\psi_1\rangle, \dots, |\psi_T\rangle$, and want to check it's valid by making a bunch of *local* measurements — that is, measurements on $O(1)$ qubits each. The trouble is, while Arthur can indeed check that

$|\psi_{t+1}\rangle = U_t|\psi_t\rangle$ —that is, $|\psi_{t+1}\rangle$ is the state that results from applying the t -th local gate to $|\psi_t\rangle$ —, the measurement that checks this point itself is not local. Arthur has to do a *swap test* between $|\psi_{t+1}\rangle$ and $U_t|\psi_t\rangle$ (like on the pset), but that involves all the qubits of $|\psi_t\rangle$ and all the qubits of ψ_{t+1} , and is not local any more.

Therefore instead, Arthur will ask Merlin to send him the witness state

$$\frac{1}{\sqrt{T}} \sum_{t=1}^T |t\rangle |\psi_t\rangle.$$

(To allow Arthur to do repeated test, the state sent is actually $(\frac{1}{\sqrt{T}} \sum_{t=1}^T |t\rangle |\psi_t\rangle)^{\otimes \text{poly}(n)}$, but this is not the critical point.) Further more, in this state, the time t is encoded in unary, that is, $|1\rangle = |100000\rangle$, $|2\rangle = |110000\rangle$, $|3\rangle = |111000\rangle$, etc. Now we can check such a state is correct using a set of measurements on at most 5 qubits each—3 adjacent qubits from the clock register, and at most 2 qubits from the computation register. Each measurement does the following: pick a random t , and measure the $(t-1)$ -th and $(t+1)$ -th qubits of the clock register. See if we get $|1\rangle$ and $|0\rangle$ respectively. If we do, that means we are now at the t -th time step, and the state left in the t -th qubit of the clock register and in the computation register is $\frac{|0\rangle|\psi_t\rangle + |1\rangle|\psi_{t+1}\rangle}{\sqrt{2}}$. (Notice that if the clock register is “bad”, we can detect it with $1/\text{poly}(n)$ probability. The error probability can be reduced by repeating polynomial times.) Now apply U_t^{-1} to the two relevant qubits in the computation register (U_t is local and applies only on 2 qubits), conditioned on the t -th qubit is $|1\rangle$. The state becomes $|0\rangle|\psi_t\rangle + |1\rangle U_t^{-1}|\psi_{t+1}\rangle = |0\rangle|\psi_t\rangle + |1\rangle|\psi_t\rangle$ (unnormalized). Finally, apply a Hadamard to the t -th qubit in the clock register, measure it, and accept if we get $|0\rangle$, reject otherwise. Notice that the final Hadamard translates the state into $|0\rangle|\psi_t\rangle + |1\rangle|\psi_t\rangle + |0\rangle|\psi_t\rangle - |1\rangle|\psi_t\rangle = |0\rangle|\psi_t\rangle$, therefore the final measurement will always get $|0\rangle$ if the computation history is correct and we will accept with probability 1. The key fact that Kitaev proved is that if the history is *far* from correct, we’ll detect that with probability at least $1/\text{poly}(n)$.

It is worth mention that people actually showed that a bunch of measurements acting on 2 qubits each is enough. Notice that 2SAT is in P. Do these results contradict with each other? No, because what we do in the local measurement above is actually sufficient to solve the max- k SAT problem, which is already NP-complete.

Another interesting issue is that there are many variants of the Cook-Levin Theorem in the classical world. One of them is the PCP Theorem, saying that, given a 3SAT formula $\psi(x_1, \dots, x_n)$, deciding whether (1) ψ is satisfiable, or (2) at most 9/10 of ψ ’s clauses can be satisfied, is NP-complete. People still don’t know whether we can have the quantum version of the PCP Theorem, says that the approximation of the Local Hamiltonians problem is already QMA-complete. Scott conjectures that this statement is true.

3 QMA vs QCMA

Last time we also talked about *QCMA*, where the proof sent by Merlin is classical, while Arthur can do a quantum check. As mentioned before, the problem $QMA \stackrel{?}{=} QCMA$ is still a major open problem in quantum computation. Actually, people still don’t know whether there exists an oracle A which separates the two class, that is, $QMA^A \neq QCMA^A$.

In a recent paper, Aaronson and Kuperberg managed to give a “quantum oracle separation” between *QMA* and *QCMA*. Just like an oracle is some Boolean function $A : \{0, 1\}^n \rightarrow \{0, 1\}$ that

an algorithm can call as a black box, a quantum oracle is some unitary transformation U that an algorithm can apply as a black box. As it turns out, sometimes it's extremely useful to let the oracles themselves be quantum.

Just like the oracle that separates P from NP , we expect that the quantum oracle U separating QMA from $QCMA$ will encode a hard unitary search problem. The n -qubit unitary U is defined such that either

- (i) $U = I$, that is, the identity matrix; or
- (ii) there exists a secret “marked state” $|\varphi\rangle$ such that $U|\varphi\rangle = -|\varphi\rangle$, and $U|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle$ orthogonal to $|\varphi\rangle$.

As expected, the YES case that Merlin is going to prove is Case (ii).

Using a QMA protocol, Merlin simply send $|\varphi\rangle$ to Arthur. To verify, Arthur performs a controlled query to U and get the state $|0\rangle|\varphi\rangle + |1\rangle U|\varphi\rangle$. Arthur will get $|0\rangle|\varphi\rangle - |1\rangle|\varphi\rangle$ if the statement is true, while $|0\rangle|\varphi\rangle + |1\rangle|\varphi\rangle$ if false (i.e., $U = I$). Arthur then performs a Hadamard on the first register and measures it, accepts if getting $|1\rangle$, rejects otherwise. Therefore this problem is in QMA .

Is it in $QCMA$? In other words, are there $\text{poly}(n)$ classical bits that Merlin can send to Arthur, that will enable Arthur to find $|\varphi\rangle$ with only $\text{poly}(n)$ queries to U ? Aaronson and Kuperberg prove the answer is no. The intuition is, Merlin's advice divides the set of all n -qubits quantum states into at most $2^{\text{poly}(n)}$ “advice regions”. But remember, the space of n -qubit states is *doubly* exponentially large (in the sense that there are 2^{2^n} states, every pair of which has large inner product). Hence at least one of those advice regions must be extremely large. And using a generalization of the BBBV hybrid argument, they prove that searching any large enough advice region for a random marked state requires exponentially many queries (in particular, $\Omega\left(\sqrt{\frac{2^n}{m+1}}\right)$, where m is the number of advice bits).

So how about a *classical* oracle separation between QMA and $QCMA$? Aaronson and Kuperberg showed that the Group Non-Membership problem cannot lead to such a separation. In particular, they gave a GNM protocol that uses a polynomial-size classical proof and $\text{poly}(n)$ quantum queries to the group oracle (though also an exponential amount of postprocessing). The idea is: Merlin can just *tell* Arthur what the black-box group is. He can say, “it's the symmetric group,” or “it's a semidirect product of the Monster group with a twisted Chevalley group of Ree type.” It's a consequence of the Classification of Finite Simple Groups that Merlin can say what the group is using a polynomial number of bits. Then Merlin also gives Arthur a *mapping* from that explicit group into the black-box group. Then Arthur just has to check two things: that the mapping is homomorphism, and that it's 1-to-1. Checking that the mapping is a homomorphism can be done classically, and checking that it's 1-to-1 can be done by solving the Nonabelian Hidden Subgroup Problem. By the result of Ettinger, Hoyer, and Knill, that requires only a polynomial number of quantum queries. Once Arthur has an embedding from the explicit group into the black-box group, he can then solve the Group Non-Membership problem by looking only at the explicit group, without making any further queries to the black-box group.

4 QMA(k)

The last topic today is $QMA(k)$, quantum Merlin Arthur with multiple proofs. The scenario is that there are k Merlins, and each of them can send to Arthur a quantum proof. This is not interesting in the classical world, because we can just concatenate the k proofs into one, and thus $MA(k) = MA$. But in the quantum case, suppose Arthur knows that the k proofs are unentangled with each other. Can Arthur use that promise to his advantage? In a paper of Aaronson, Beigi, Drucker, Fefferman, and Shor, they give evidence that indeed Arthur can. For example, if a 3SAT formula of size n is satisfiable, then this satisfiability can be proved to Arthur using $O(\sqrt{n}\text{polylog}(n))$ unentangled quantum proofs with $\log(n)$ qubits each.

Next time we will talk about quantum interactive proofs.

Lecture 17

*Lecturer: Scott Aaronson**Scribe: Bhaskar Mookerji (mookerji@mit.edu)*

Last time, on *America's Most Wanted Complexity Classes*:

1. QMA vs. QCMA; QMA(2).
2. IP: Class of languages $L \subseteq \{0, 1\}^*$ for which there exists an interaction protocol between BPP verifier and an omnipotent prover s.t. $\forall x$:
 - (a) $x \in L \implies \exists$ a prover strategy that causes verifier to accept w.p. $> 2/3$
 - (b) $x \notin L \implies \forall$ prover strategies, verifier accepts w.p. $\leq 1/3$.
3. **Theorem 1 (LFKN, Shamir)** $\text{IP} = \text{PSPACE}$.

Everything we've been discussing so far involved one-shot proof systems where Merlin (or Merlins) send some quantum states to Arthur, and then Arthur verifies those states. But one can also study quantum interactive proof systems (QIP), where Arthur and Merlin send quantum messages back and forth.

1 Classical Interactive Proofs

First of all, what do we know about classical interactive proof systems (IP)? Let IP be the class of problems for which a 'yes' answer can be verified (with constant error) by an interactive protocol in which a polynomial-time Arthur exchanges messages with an omniscient Merlin. Here 'omniscient' means that Merlin can do an unlimited amount of computation, but is unaware of the questions Arthur will ask in the future and subsequently commits to his answers without knowledge of Arthur's responses. The intuition behind the power of interactive proofs is not surprising. From our day-to-day experience, we know that reading a proof is usually more difficult than simply asking its author for details. In complexity theory, we have strong evidence that this process is more powerful than static provers. We reached this conclusion when we looked at AM and MA, where proofs in the former are not accessible in the latter ($\text{AM} \subseteq \text{IP}$), and the complexity of the graph isomorphism problem ($\text{GNI} \in \text{IP}$).

A famous result of Lund, Fortnow, Karloff, and Nisan (finished off by Shamir) says that this class is incredibly big: $\text{IP} = \text{PSPACE}$, meaning that the optimal strategy for the prover can be computed in polynomial space. For example, if a super-intelligent alien came to earth, it could convince us that White has the win in chess. The theorem suggests that there is a protocol by which the alien *could* convince us that White has to win in chess. We'd do that by transforming chess into a different game involving polynomials over finite fields. In the new game, the best strategy for one of the two players is to move randomly. If in this randomization scenario, the alien wins, we should be convinced that the alien could win against *any* player.

Before moving on to QIP, we will very briefly consider the LFKN simpler result that $\text{coNP} \subseteq \text{IP}$, i.e., one can prove through an interactive protocol that a Boolean formula is 'unsatisfiable.' Note that this is surprising because \exists an oracle A s.t. $\text{coNP}^A \not\subseteq \text{IP}^A$. This means that the proof of

$\text{coNP} \subseteq \text{IP}$ must be a non-relativizing proof. This is one of the few examples we have of a proof that exploits enough about the structure of computation that they would actually fail in the real world where there is such an oracle.

Theorem 2 (LFKN) $\text{coNP} \subseteq \text{IP}$.

Proof: [Sketch] The idea here is that we have some Boolean formula $\phi(x_1, \dots, x_n)$ that Merlin wants to convince Arthur it is non-satisfiable. Here we ‘arithmetize’ the expression by replacing all the Boolean variables with finite field elements $x_1, x_2, x_3 \in \mathbb{F}_p$, and all the Boolean operations with arithmetic operations over \mathbb{F}_p . For example, a 3-bit OR arithmetizes to

$$x_1 \vee x_2 \vee x_3 = 1 - (1 - x_1)(1 - x_2)(1 - x_3), \quad (1)$$

which is a polynomial over the field of \mathbb{F}_3 .

As as a result, our goal here is to convince Arthur that

$$\sum_{\substack{x \in \{0,1\}^* \\ x_1, \dots, x_n}} p(x) = 0. \quad (2)$$

The omnipotent Merlin can easily verify this statement is true for some Boolean string x and tell Arthur the result. However, Arthur isn’t so gullible. He requires convincing. Instead, Merlin performs the sum over the last $n - 1$ variables such that

$$q_1(x_1) = \sum_{x_2, \dots, x_n} p(x_1, x_2, \dots, x_n). \quad (3)$$

Merlin sends Arthur all the coefficients, and Arthur can check for himself that $q_1(0) + q_1(1) = 0$.

Arthur must verify that the Merlin has determined the above sum correctly. Arthur evaluates $q_1(x_1)$ at some random $x_1 = r_1 \in \mathbb{F}_p$. From this point on, Merlin must use some fixed value of x_1 that Arthur has picked, and then returns

$$q_2(x_2) = \sum_{x_3, \dots, x_n} p(r_1; x_2, \dots, x_n) \quad (4)$$

for which Arthur can verify that $q_2(0) + q_2(1) = q_1(r_1)$. The process iterates, and Arthur picks another $r_2 \in \mathbb{F}_k$ and Merlin returns

$$q_3(x_3) = \sum_{x_4, \dots, x_n} p(r_1, r_2; x_3, \dots, x_n). \quad (5)$$

Arthur and Merlin continue until $q_n(x_n)$ and all the values have been fixed. Arthur can check that $p(r_1, \dots, r_n)$ is the required value.

However, if Merlin is lying, Arthur can catch him with constant probability. To show this we use the following fact that *a d -polynomial has at most d roots* (the Fundamental theorem of algebra). If we have two d -degree polynomials that are not equal, they can only be equal on at most d inputs. This means that the polynomials q and p can only agree on a polynomial number of elements. Therefore, if Arthur picks a random r_1 , the verified polynomial will almost certainly disagree if Merlin is lying. \square

2 Quantum Interactive Proofs

Just as you'd expect, one can also define QIP: Quantum Interactive Proofs. Here the prover and verifier can exchange quantum messages, and the prover is limited only by the laws of quantum physics. The protocol is shown in Figure 1 below.

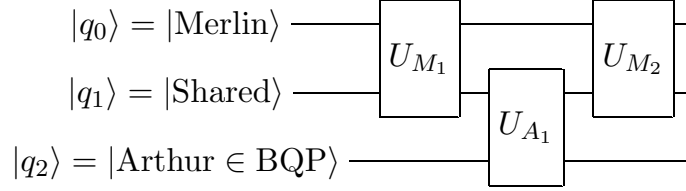


Figure 1: Merlin-Arthur QIP protocol using a polynomial number of gates. To simplify the illustration, each set of private and shared qubits used by Merlin and Arthur are tensored together a polynomial number of times. For example, $|q\rangle = \otimes_i^{p(n)} |i\rangle$.

QIP is defined as the class of languages $L \subseteq \{0, 1\}^*$ for which there exists an interaction protocol between BQP verifier (Arthur) and an omnipotent prover (Merlin) s.t. $\forall x$:

1. $x \in L \implies \exists M_1 \text{ and } M_2 \text{ causing Arthur to accept w.p. } > 2/3$
2. $x \notin L \implies \forall M_1 \text{ and } M_2 \text{ Arthur accepts w.p. } \leq 1/3$.

Certainly $\text{IP} \subseteq \text{QIP}$; that is, quantum interactive proof systems can simulate classical ones. Thus $\text{PSPACE} \subseteq \text{QIP}$. However, it turns out that something new and extremely interesting happens in the case of quantum interactive protocols.

Theorem 3 (Kitaev, Watrous00) *Any QIP protocol can be made three-round. In other words, all QIP rounds are given by $\text{QIP}(1) = \text{QMA}$, $\text{QAM} \subseteq \text{QIP}(2)$, and $\text{QIP}(3) = \text{QIP}$.*

Proof: [Sketch] To illustrate, let's just show how to do PSPACE with three rounds ($\text{PSPACE} \subseteq \text{QIP}(3)$). Assume without loss of generality that Arthur's messages to Merlin are all just uniform random bits. Then Merlin can send Arthur a (claimed) superposition over all possible conversations that they could have had:

$$\frac{1}{\sqrt{2^T}} \sum_{a_1, \dots, a_T} |a_1\rangle |a_2\rangle \cdots |a_T\rangle |m_1\rangle |m_2\rangle \cdots |m_T\rangle \quad (6)$$

For reasons we'll see later, Merlin also keeps a copy of the $|a_T\rangle$ registers for himself. Arthur can now check, in superposition, whether or not the conversation would have caused him to accept. The trouble is, what if Merlin cheated by picking $|a_T\rangle$'s that weren't truly random—and were instead concentrated on the tiny fraction where he can get away with lying?

Arthur needs to verify that the $|a_T\rangle$'s are random. To do so, he first picks a random time step t , and sends Merlin the $|m_u\rangle$ for all $u > t$. Using his copy of the $|a_T\rangle$ registers, Merlin then uncomputes those $|m_u\rangle$. Finally, Merlin sends Arthur his $|a_T\rangle$ registers. Arthur is now able to measure the $|a_u\rangle$ registers with $u > t$ in the Hadamard basis, and check whether the messages

supposedly from him were really uniformly random. If Merlin was honest, Arthur will now accept with probability 1. The nontrivial thing you have to prove is that if Merlin cheated, Arthur will detect it with $1/\text{poly}(n)$ probability. Furthermore, he can amplify that probability by running the protocol polynomially many times in parallel. \square

It's important to note that Merlin will be unable to suddenly swap out his qubits with some other qubits and expect the entanglement he shares with Arthur to remain intact. This fact, known as the 'monogamy of entanglement,' is related to the differences between correlation and entanglement. Here, consider three classical bits x , y , and z , and the correlation $x \sim y \sim z$. By transitivity, if $x \sim y$ and $y \sim z$, $x \sim z$. However, if two variables x and y are entangled, it is not possible to entangle a third variable z with x . This can be shown by taking the partial trace over a sample three-way entangled GHZ state,

$$|xyz\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \quad (7)$$

which results in x and z having statistically-independent distributions.

The remaining question here is the upper bound to QIP. Kitaev and Watrous also showed that $\text{QIP} \subseteq \text{EXP}$. They did this by expressing the problem of finding the best possible strategy for the prover as an exponentially-large, semi-definite programming (SDP) problem. SDP is known to be solvable in polynomial time. Let's see how they did this: SDP is basically the problem of finding positive definite matrices that satisfy a set of linear constraints. A general quantum state (i.e. a mixed state) is just a Hermitian positive definite matrix with trace 1. The trace 1 and Hermitian are linear constraints. So, the problem of finding N -dimensional quantum mixed states that satisfy a bunch of linear constraints is an SDP. The question now becomes, how can we formulate the problem of finding the optimal strategy for Merlin in a quantum interactive protocol, as a problem of finding mixed states that satisfy linear constraints? Imagine the circuit depicted in Figure 1 that relates mixed states ρ and σ through the QIP unitary protocol. The problem is to find states ρ, σ such that ρ is a valid initial state, σ is a final state that accepts with maximum probability, and $\text{Tr}_M(U\rho U^{-1}) = \text{Tr}_M(W^{-1}\sigma W)$. Here we're using the fact that if $\text{Tr}_M(U\rho U^{-1}) = \text{Tr}_M(W^{-1}\sigma W)$, then there must exist a unitary transformation on Merlin's registers only that transforms $U\rho U^{-1}$ to $W^{-1}\sigma W$.

To this day, we don't know exactly where QIP sits between PSPACE and EXP.

2.1 Multi-prover QIP

Finally, many quantum computing people lately have been interested in multi-prover quantum interactive proof systems. In the classical world, putting two people in separate rooms to interrogate them often lets you learn more than if the people could talk to each other. Let MIP be the class of problems for which a 'yes' answer can be efficiently verified with the help of two or more non-communicating provers. Babai, Fortnow, and Lund showed that $\text{MIP} = \text{NEXP}$, whereas IP only equals PSPACE. In the quantum world, though, we don't know whether QMIP contains NEXP. What do you think the difficulty is? The provers could be entangled with each other! And indeed, Cleve, Hoyer, Toner, Watrous 2004 gave examples of protocols that are sound when the provers don't share entanglement, but become unsound when they did.

Nor, embarrassing as it is to admit, do we know any upper bound whatsoever on QMIP—the reason being that we don't know *a priori* how much entanglement the provers need in their strategy.

Doherty, Liang, Toner, Wehner 2008 show that if a finite amount of entanglement suffices, then all QMIP languages are at least recursive. (On the other hand, we still don't know if there are situations where a literally infinite amount of entanglement is needed to play optimally!)

Just as BQP is contained in PP, so BQPSPACE is contained in PSPACE. But Ladner proved that PSPACE = PSPACE, using the same ideas as in Savitch's Theorem. Hence BQPSPACE = PSPACE.

3 The Future

Quantum computing with closed time-like curves.

Lecture 18

*Lecturer: Scott Aaronson**Scribe: Joshua Horowitz*

On Election Day, November 4, 2008, the people voted and their decision was clear: Prof. Aaronson would talk about quantum computing with closed time-like curves. But before we move into strange physics with even stranger complexity-theoretic implications, we need to fill in a more basic gap in our discussion so far – quantum space-complexity classes.

1 BQPSPACE

Recall that PSPACE is the class of all decision problems which can be solved by some classical Turing machine, whose tape-length is bounded by some polynomial of the input-length. To make BQPSPACE, we quantize this in the most straightforward way possible. That is, we let BQPSPACE be the class of all decision problems which can be solved with a bounded rate of error (the B) by some quantum Turing machine (the Q), whose tape-length is bounded by some polynomial of the input-length (the PSPACE). As we've seen before, allowing quantum computers to simulate classical ones gives the relationship $\text{PSPACE} \subseteq \text{BQPSPACE}$.

If we believe that BQP is larger than P, we might suspect by analogy that BQPSPACE is larger than PSPACE. But it turns out that the analogy isn't such a great one, since space and time seem to work in very different ways. As an example of the failure of the time-space analogy, take PSPACE vs. NPSPACE. We certainly believe that nondeterminism gives classical machines exponential time speed-ups for some problems in NP, making $P \neq NP$ at least in our minds. But Savitch's theorem demonstrates that the situation is different in the space-domain: $\text{PSPACE} = \text{NPSPACE}$. That is, nondeterministic polyspace algorithms can be simulated in deterministic polyspace. It turns out even more than this is true. Ladner proved in 1989 that $\text{PSPACE} = \text{PPSPACE}$, establishing PSPACE as a truly solid rock of a class.

This is the result we need to identify BQPSPACE. Using the same technique we used to prove $\text{BQP} \subseteq \text{PP}$, we can prove that $\text{BQPSPACE} \subseteq \text{PPSPACE}$, and then using $\text{PPSPACE} = \text{PSPACE} \subseteq \text{BQPSPACE}$ we have $\text{BQPSPACE} = \text{PSPACE}$.

2 Talking to Physicists

Some physicists have been known to react to the stark difference between time- and space-complexity with confusion and incredulity. "How can $\text{PTIME} \neq \text{PSPACE}$," they ask, "if time and space are equivalent, as suggested by Einstein's theory of relativity?"

The basic answer to this question is that time and space are in fact not perfectly equivalent. Einstein's theory relates them in a non-trivial way, but distinctions remain. At the very least, they correspond to opposite signs in the metric tensor. And, for whatever reason, though objects can move about space at will, movement through time is constrained to the "forward" direction. Thus, our inability to travel back in time is in a sense the reason why (we believe) $P \neq \text{PSPACE}$!

3 Time Travel

But are we really so sure that time travel is impossible? Special relativity says that it takes infinite energy to accelerate faster than the speed of light and into the past. But general relativity extends our concept of space-time, allowing it to take the form of whatever sort of topologically crazy 4D manifold we want (more or less). Is it possible that such a manifold could have loops of space-time which extended forwards in time but ended up bringing us back to where we started? That is, can the universe have *closed time-like curves* (CTCs)?

In an amusing coincidence for a class on complexity theory, one of the first people to investigate this question was Kurt Gödel, who in 1949 constructed a solution to the general-relativistic field equations which included CTCs, and gave it to Einstein as a birthday gift. Like all good birthday gifts, this left Einstein somewhat troubled. The solution was fairly exotic, however, involving such strange constructions as infinitely long massive cylinders.

The problem came up again in the 1980s, when Carl Sagan asked his physicist friend Kip Thorne if there was a scientific basis for the time travel in his novel-in-progress *Contact*. Though Thorne initially dismissed the possibility, he later began to look into whether wormholes could be used for time travel. The problem turned out to be surprisingly difficult and subtle. The conclusion reached by Thorne and his collaborators was that such wormholes were theoretically possible, though they would require the presence of negative energy densities. Given that such negative energy densities have been known to arise in quantum situations such as the Casimir effect, Thorne's work effectively linked the question of wormhole time-travel to that of quantum gravity (the “P vs. NP” of theoretical physics).

Of course, it is not the place of computer scientists to say whether something actually exists or not. They need only ask “what if?”. So we will suppose that we have computers capable of sending information back in time, and see where that takes us.

4 Computing with CTCs: The Naïve Proposal

The first scheme many people think of for speeding up computations with CTCs is very simple:

1. Perform your computation however you like.
2. Send the answer back in time to whenever you wanted it.

At first glance, this sounds pretty good. It makes every computation constant-time! Conceivably, even negative-time...

But there are problems with this scheme, which is fortunate for anyone hoping time-travel computation would be anything other than completely trivial.

- It encourages “massive deficit spending”: Life may be easy for you, now that you’ve received the answer to your NP-complete question in no time at all, but the accounting of time as $O(0)$ ignores the possible millennia of computation your computer still has to do to get that answer. Keeping a computer running that long could be extraordinarily costly or even impossible if the run-time exceeds the lifetime of the universe itself.
- Complexity theory is all about knowing exactly how much of each computational resource you need in order to solve a problem. However, in the above treatment, we are completely

ignoring the CTC as a computational resource. If we accounted for its length, we would soon discover just how blithely we were constructing exponentially long wormholes in our naïve attempt to defeat complexity itself.

- There must be a fundamental flaw in the way we are modeling time travel, since, the way we’ve been talking, there’s nothing to prevent “grandfather” paradoxes¹. As a computational example of this, suppose we had a computer which took its input, NOTed it, and then sent that back in time to serve as its own input. If the machine sent a 0, it would receive a 0 and thus send a 1, and visa versa, so the machine’s existence creates a physical contradiction.

5 Deutsch’s Solution and P_{CTC}

Of the problems mentioned above, the most pressing one to solve is problem the last. How can the universe allow time travel at all if this creates time-loop paradoxes?

In 1991, David Deutsch proposed a resolution to this problem. He claimed that time-loop paradoxes such as the NOTing computer above (or the grandfather paradox itself) can only occur in classical universes. In a quantum universe, such as our own, the universe can always find a way to maintain consistency (according to Deutsch).

As an example of this sort of resolution, take the grandfather paradox, and represent the two possibilities as a vector \vec{v} which can be either

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \leftarrow \text{you kill your grandfather} \quad \text{or} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \leftarrow \text{you do not kill your grandfather} .$$

Your murderous time-travelling adventure puts a constraint on \vec{v} :

$$\vec{v} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \vec{v}.$$

Clearly, neither the two possibilities mentioned above satisfies this fixed-point equation; the two possibilities flip into each other. This is just a restatement of the grandfather paradox. But this restatement presents an interesting idea: what if we could have $\vec{v} = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$? This “superposition” of the two states is a fixed point of the matrix map above. As suggested by the normalization $\frac{1}{2}$ out in front, we are interpreting it as a vector of probabilities. (Truly quantum states will come later; for now we will use a semi-classical or probabilistic approach.)

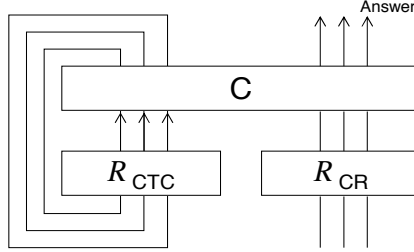
Does turning every time-loop constraint into a matrix and looking for general fixed-point vectors always work? That is, if some computation in a CTC establishes the constraint $\vec{v} = M\vec{v}$, is there always a solution for \vec{v} ? This certainly isn’t true for any matrix M , but matrices which come from time-loop constraints will satisfy additional properties, since they have to take probability vectors to probability vectors. A necessary and sufficient pair of conditions which guarantees that M does this is that M ’s entries are non-negative and that each of its columns sums to one. A matrix with these properties is said to be a *stochastic matrix*.

Fortunately for us, it is well-known that every stochastic matrix has at least one probability vector as a fixed point. This is exactly what we need to ensure that the universe can find a

¹The story is that, by traveling back in time and killing your own grandfather, you prevent your own birth, thus preventing your going back in time to kill your grandfather, thus enabling your own birth, etc.

probability distribution over the classical states which is preserved by the time loop! So in our model, our computer's design will determine for Nature a stochastic matrix M , and Nature will in turn provide the computer with some fixed-point probability distribution which it can sample by observation. (As good computer scientists, we will assume that Nature chooses this distribution as an adversary.)

Let's make this rigorous: We will have some classical computer C . It will have two input registers, one for the CTC-looping bits, called R_{CTC} , and one for the standard "causality-respecting" bits, called R_{CR} :



Given a set of inputs to R_{CR} , C determines a map from R_{CTC} to R_{CTC} . This map can be represented by a stochastic matrix, which we know must have some fixed-point probability distribution. Each such probability distribution gives a probability distribution for values of C 's answer bits.

In the case of decision problems, C has exactly one answer bit. We say that our CTC computer accepts (resp. rejects) some input if every non-zero-probability value of R_{CTC} in every fixed-point probability distribution causes C to give an answer of 1 (resp. accepts). Since we are permitting no uncertainty in the actual output of the computer, these concepts of accepting/rejecting are appropriate for defining the complexity class P_{CTC} of polynomial-time computers with closed time-like curves. The only other subtlety is that we should make sure we only use a polynomially large number of CTC bits.

5.1 An Example: Search

To illustrate the power of CTCs, let us use them to solve an NP-complete problem: search. That is, given a polytime computable function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, we want to find an $x \in \{0, 1\}^n$ such that $f(x) = 1$, or to know that no such x exists. One approach to solving this with CTCs is to find a map for which fixed points correspond to x s with $f(x) = 1$. The simplest candidate is

$$C(x) = \begin{cases} x & f(x) = 1 \\ x + 1 \pmod{2^n} & f(x) = 0 \end{cases}$$

and it turns out that this works perfectly! If there is a x with $f(x) = 1$, the fixed-point distributions of C will be non-zero only on such solutions (since probability flows towards these solutions under the action of the matrix M). If there is no such x , then C acts as a 2^n -cycle on the values of R_{CTC} , and the only fixed-point distribution is the uniform distribution over all possibilities. Thus, by taking the value of R_{CTC} and testing the value of f at that value, we can determine conclusively whether that value is a solution or that there is no solution at all. (Note that, though the value of the R_{CTC} bits is in general probabilistic or undetermined, the final answer is always exactly what we want.)

Therefore, the search problem is in P_{CTC} . Since the search problem is NP-complete, we have $NP \subseteq P_{CTC}$. Assuming $P \neq NP$, this gives us a separation between P and P_{CTC} .

6 A New Sort of Paradox?

We know that by allowing probabilistic superpositions of states, we have successfully averted the problem of time-loop paradoxes (fixed-point-less CTC constraints). But in its stead, we have produced a new sort of problem. CTCs allow us to make programs which can find the answers to problems “out of thin air”, without doing any of the work to find the solution. This is much like the time-travel paradox of going back in time to 1904 and giving Einstein his own papers, which he looks over with great interest and proceeds to furtively publish a year later. Who actually wrote the papers?

Of course, this is not a true paradox of mathematical or physical contradiction (a *falsidical paradox*), but one of a derived result contradicting natural intuition (a *veridical paradox*). It seems that if we are to allow time travel, we must abandon some of our intuitions about causality and information. “Plays get written, plexiglass gets invented, and NP-complete problems get solved,” without anyone having to put in the work.

7 P_{CTC} Vs. PSPACE

We would like to understand exactly what is possible in polynomial time using closed time-like curves. That is, we would like tight bounds on P_{CTC} . We already have a lower bound of $NP \subseteq P_{CTC}$.

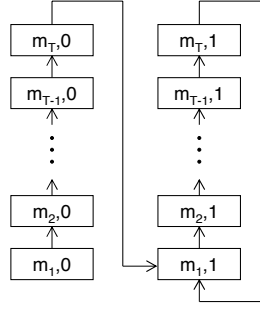
There is also good upper bound: PSPACE. Simulating a P_{CTC} with a PSPACE machine is fairly straightforward: We can visualize the action of our computation on R_{CTC} as a directed graph in every vertex has out-degree 1. Such a graph will have the structure of a collection of disjoint cycles, possibly with trees of edges flowing into the cycles. Thus, starting with any vertex, repeated application of the computation will eventually bring you to a vertex in one of the cycles, which will have non-zero probability on a fixed-point distribution. Since it’s no fun to detect whether a vertex is in a cycle or not, we will just play it safe and iterate the computation on the arbitrarily chosen point 2^n times: we must end up in a cycle, or there would be no cycle, which is impossible.

The more surprising fact is that we can improve P_{CTC} ’s lower bound from NP to PSPACE, thus proving that $P_{CTC} = PSPACE$. Apparently CTCs allow us to interchange time and space, like the physicists suspected!

To do this, we must find a way to perform every PSPACE computation in polytime with CTCs. The basic idea is to store the PSPACE machine’s tape in the the CTC register and have the P_{CTC} machine perform a single step of the PSPACE machine’s computation. But we have to be careful to define our computation so that we can read the final result of the simulated PSPACE computation off of the fixed-point distribution.

We do this by adding an auxiliary bit to the CTC register which represents the outcome of the computation. Usually this auxiliary bit remains untouched, but we redirect every halting state to the initial state with the appropriate auxiliary bit (1 if the halting state was accepting, 0 if it was rejecting).

The directed graph of this operation is shown below (in a situation where the machine accepts):



As this diagram shows, the only cycle in the graph will be one going from the initial state all the way to a halting state and then back to the start, with every state having the auxiliary bit determined by whether that halting state was accepting or rejecting. Thus, by giving us a fixed-point distribution, Nature is doing the entire computation for us and telling us how it went.

Notice exactly how our scheme exploits the relationship between PSPACE and P_{CTC} . Since our tapes have polynomial length, a single computation step can be done in polynomial time. In contrast, we cannot use this technique to put EXPSPACE inside P_{CTC} , which is good, since $\text{P}_{\text{CTC}} \subseteq \text{PSPACE}$ and EXPSPACE is certainly not a subclass of PSPACE .

8 Quantum Computation and Time Travel: BQP_{CTC}

Though our model of time-travelling computers was inspired by quantum physics, we haven't actually given our computers the quantum capabilities which we've been studying all semester. What would that look like?

When we had classical computers with time-loops, the time-loop constraint was of the form $\vec{v}_{\text{CTC}} = M\vec{v}_{\text{CTC}}$ for some stochastic matrix M . What sort of time-loop constraint could a quantum computer make? A good first guess would be $|\psi_{\text{CTC}}\rangle = U|\psi_{\text{CTC}}\rangle$ for some unitary U . But this isn't quite right. Though our computation as a whole, from [input bits + CTC bits] to [output bits + CTC bits] can be represented by a unitary U , restricting this to a map from [CTC bits] to [CTC bits] does not give a unitary. In fact, the possibility of entanglement between the output bits and the CTC bits means that it doesn't make any sense to think of the state of the CTC bits as a pure quantum state: $|\psi_{\text{CTC}}\rangle$ doesn't exist.

The right way to represent a substate of a quantum state is with a *density matrix* $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$. How are we allowed to transform density matrices into each other? Well, every unitary map on a pure-state supersystem restricts to a transformation on the mixed-state subsystem. There is an exact mathematical characterization of the form such transformations can take:

$$\rho \mapsto \sum_i E_i \rho E_i^\dagger \text{ where } \sum_i E_i^\dagger E_i = I.$$

This characterization is exact in the sense that the restriction of any unitary to a mixed-state subsystem gives such a transformation, and every such transformation comes from the restriction of a unitary on a pure-state supersystem. We call these transformations *superoperators* (the analogy to remember is "Pure state : Mixed state :: Unitary : Superoperator").

So, in general, we can expect our quantum computer to create a constraint of the form $\rho_{\text{CTC}} = S(\rho_{\text{CTC}})$ for some superoperator S . Fortunately for the coherence of space-time, such an equation is always solvable! That is, every superoperator has a fixed point. So if we do the same sort of thing we did above to define P_{CTC} but on quantum computers instead, and require only a bounded rate of error rather than perfection, we'll get BQP_{CTC} .

Now to bound this complexity class. Quite trivially, $\text{P}_{\text{CTC}} \subseteq \text{BQP}_{\text{CTC}}$, and $\text{P}_{\text{CTC}} = \text{PSPACE}$, so we can bound BQP_{CTC} below by PSPACE . We expect that adding quantum capabilities will extend our abilities, though, so what's the upper bound?

Well, it turns out that our expectation is wrong: BQP_{CTC} is also bounded above by PSPACE . Once again, we are face-to-face with the solidity of the rock that is PSPACE ! The fact that $\text{BQP}_{\text{CTC}} = \text{PSPACE}$ was proven by Aaronson & Watrous in 2008. The title of their paper, "Closed Timelike Curves Make Quantum and Classical Computing Equivalent", raises an interesting question: why are we spending so much time trying to get quantum computers to work if they would be rendered totally obsolete by CTC computers?

Lecture 18

*Lecturer: Scott Aaronson**Scribe: Sam McVeety*

1 Last Time: Quantum Interactive Proofs

1.1 $\text{IP} = \text{PSPACE} \subseteq \text{QIP} = \text{QIP}(3) \subseteq \text{EXP}$

The first result is interesting, because the fact the PSPACE is contained in IP is slightly counter-intuitive. One consequence of this is that game theory questions (i.e. white has the win in chess) can be proved through a message exchange protocol. We also saw that $\text{QIP} = \text{QIP}(3)$, or in other words, that any quantum protocol can be reduced to three rounds.

1.2 $\text{MIP} = \text{NEXP}$

If we allow for multiple provers, the situation becomes more interesting. Effectively, Alice and Bob are in separate rooms, and we try to interrogate them separately. This protocol gives us more power in the classical case, and possibly even more power in the quantum case.

1.3 $?? \subseteq \text{QMIP} \subseteq ??$

Nothing is known about the relationship of QMIP to other complexity classes, because of the arbitrary amount of entanglement that is allowed. Answering this question requires a better understanding of entanglement than we currently have, which is a reason that this question is very studied currently. There is no way within the laws of physics to require that entanglement is not shared.

Classically, Alice and Bob can agree on their strategy in advance. So, does entanglement actually give us anything more? Ostensibly, the answer is yes. What we can be sure of, is that entanglement breaks some MIP protocols which work classically. CHTW [2] gives examples of such protocols.

1.4 Graph Two-Coloring

Given provers Alice and Bob who claim to know a two-coloring of a graph, your strategy is as follows: flip a coin, with $1/2$ probability ask Alice and Bob how to color a specific vertex. Otherwise, ask them about adjacent vertices.

We can use a convexity argument to show why they can't cheat with perfect reliability in the classical world, on a graph containing an odd cycle. Given any probabilistic strategy for cheating, there will always be a deterministic strategy that does as well as the probabilistic one. Since there is no actual coloring that works, there is always a probability that they will be caught with non-zero probability. Since at least one vertex will trip them up, they will be caught with probability $\Omega(n^{-1})$.

On the other hand, in a quantum world, if Alice and Bob share the entangled state

$$\frac{|00\rangle + |11\rangle}{2}$$

then there is a strategy such that you catch them with probability $O(n^{-2})$ [2].

Here's what you do. We want each of the vertices on the cycle to correspond to quantum state (point on the unit circle) such that they are spaced $2\pi/n$ apart. We have Alice and Bob each measure their qubit in the basis that corresponds to the vertex that is queried. Accordingly, we want adjacent vertices to be nearly orthogonal, so we place adjacent vertices $\pi/2 + 2\pi/n$ radians apart. Given this ordering, Alice and Bob will either measure their qubit in the same basis, or in nearly orthogonal bases. When we say, "nearly orthogonal," this means something like $\cos^2 2\pi/n$ off, which corresponds to a likelihood is something like $1 - n^{-2}$, that Alice and Bob will measure different states.

This bound is tight for provers that are only allowed to share the state given above, but there exist strategies that allow the provers to cheat perfectly by sharing specific entangled states.

Remark 1 *QMIP with finite entanglement is upper bounded by the set of computable functions.*

Remark 2 *If $P = NP$, then $EXP = NEXP$. This follows from a simple padding argument.*

Quantum parallel games: write out best prover strategy as semi-definite program, using parallel repetition theorem.

2 Quantum computing with Postselection

Postselection refers to the process of conditioning the experiment on getting the outcome that you are looking for, and discarding the outcome otherwise. Today, we will investigate the computational power that this additional theoretic resource gives us. This has a very obvious tie-in with the many worlds interpretation of quantum mechanics.

Definition 1 *The many worlds interpretation concludes that the world splits at every quantum branching point, and we continue in a superposition of all of these states.*

For example, if you really want the answer to a problem, you can perform an experiment where you postselect on receiving a certain outcome, and otherwise shoot yourself. Accordingly, in universes where you are still alive, you get the answer that you were looking for.

Less morbidly, you could make a firm commitment, if you get the measurement you want, that you'll have lots of children, and many descendants. Otherwise, you will sterilize the human race. This maximizes the probability that someone will see the answer to your computational problem.

This gives rise to the class **PostBQP**.

Definition 2 *This is the class of $L \subseteq \{0,1\}^*$ such that \exists a polytime quantum algorithm Q such that \forall inputs x ,*

- $Q(x)$ gets selected with probability > 0
- If $x \in L$ conditioned on being selected, $Q(x)$ accepts with probability $\geq 2/3$

- If $x \notin L$, conditioned on being selected, $Q(x)$ accepts with probability $\leq 1/3$

Overall, this algorithm consists of two measurements, one to decide whether to proceed or throw away the computation, and the second, to see if the computation ultimately accepts or rejects. Such experiments are very common in physics. For example, we could postselect on the condition that a photon didn't go where it was supposed to go.

A natural question to this is whether the presence of additional postselection measurements helps us. It turns out that it does not, by the Principle of Deferred Measurement. Without loss of generality, we can assume there is only one measurement, and simulate the rest with controlled-NOT gates.

Remark 3 *It should be immediately clear that $BQP \subseteq \text{PostBQP}$.*

2.1 $\text{NP} \subseteq \text{PostBQP}$?

We devise an algorithm for solving any problem in NP with a PostBQP algorithm. First, go into superposition over all inputs

$$\frac{1}{2^{n/2}} \sum |x\rangle |f(x)\rangle$$

and postselect on finding a valid certificate, then accept or reject accordingly. However, this has the flaw that it does not handle the case of there being no solutions. We fix this by adding a dummy state with extremely low amplitude (say, 2^{-20n} .) If we measure and postselect on getting a 1, and get the dummy solution, then there is almost certainly no real solution.

Can we find an upper bound on PostBQP? PSPACE immediately comes to mind, but it turns out that we can do better.

Theorem 1 (Adleman, DeMoorai, Huang) $\text{PostBQP} \subseteq \text{PP}$

Proof: Do a Feynman path integral, and sum over all contributes to the final amplitude. Restrict to the states that you postselect, and make all of the others cancel. \square

Theorem 2 $\text{PP} \subseteq \text{PostBQP}$

Proof:

PP basically means that you can compute the majority function on an exponentially long string, hence we can model an arbitrary problem in PP in the following way.

- $f : \{0, 1\}^n \rightarrow \{0, 1\}$
- $s = |\{x : f(x) = 1\}|$
- Problem: Decide if $s \geq 2^{n-1}$, assume $s > 0$.

Prepare a 1-qubit state:

$$|\psi\rangle = \frac{(2^n - s)|0\rangle - s|1\rangle}{\sqrt{(2^n - s)^2 + s^2}}$$

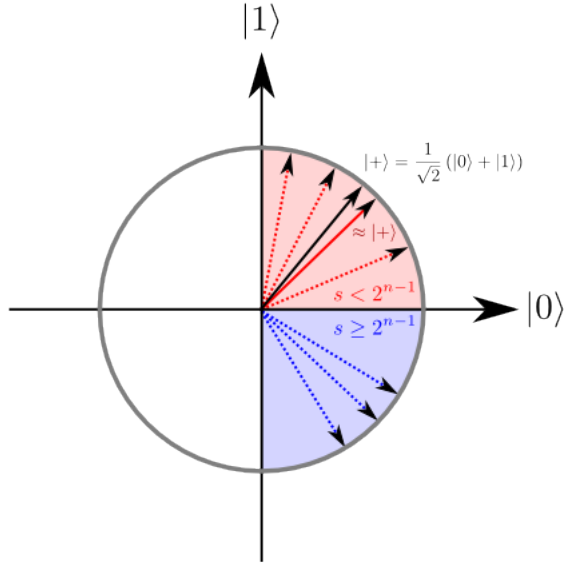


Figure 1: Different values of s

This means that, by applying a Hadamard conditioned on the first qubit, we can also prepare the state

$$\frac{\alpha |0\rangle |\psi\rangle + \beta |1\rangle H |\psi\rangle}{\sqrt{\alpha^2 + \beta^2}}$$

Note that

$$H |\psi\rangle = \frac{\frac{1}{\sqrt{2}} 2^n |0\rangle + \frac{1}{\sqrt{2}} (2^n - 2s) |1\rangle}{\sqrt{(2^n - s)^2 + s^2}}$$

Accordingly, we postselect on the second qubit being 1:

$$|\psi_{\alpha,\beta}\rangle = \alpha s |0\rangle + \beta \frac{2^n - 2s}{\sqrt{2}} |1\rangle$$

If s is appropriately large, the second term is negative, otherwise it is positive. We can find out which by varying $\frac{\alpha}{\beta}$ in a certain way. We do need to be clever here, because we have to postselect based on the outcome of a measurement, we can't just postselect on some non-measurable condition like negative amplitude. Pick β and α from the set:

$$\beta/\alpha = \{2^{-n}, \dots, 1/2, 1, 2, \dots, 2^n\}$$

The vectors corresponding to $\frac{\alpha}{\beta}$ and different values of s are shown on the unit circle below in Figure 2.1.

2.1.1 First Case:

Assume α, β positive. If $s < 2^{n-1}$, then both amplitudes will be positive. So we keep varying alpha, beta, and measure in the Hadamard basis. This gets us very close to $|+\rangle$, which we can detect with non-trivial probability.

2.1.2 Second Case:

If $s \geq 2^{n-1}$, then we're in the fourth quadrant. For some $\frac{\alpha}{\beta}$, we are close to $|-\rangle$, so we can discern the relative value of s .

□

3 Classical Results from Quantum Lower Bounds

It is a bit strange that $\text{PostBQP} = \text{PP}$, in that the latter class has been part of classical complexity theory for many years, while the former has only recently emerged. The fact that they are equal gives rise to the hope that we may be able to use results in quantum complexity theory to better inform our understanding of classical complexity theory. This is not altogether surprising, though, given the efficacy of probabilistic methods in proving classical results that ultimately have little to do with probability.

“Quantum generosity - giving back to the classical world.”

Theorem 3 (Beigel-Reingold-Spielman, 1991 [1]) *PP is closed under intersection:*

$$L_1, L_2 \in \text{PP} \Rightarrow L_1 \cap L_2 \in \text{PP}$$

This is a non-trivial result, showing that the AND of two large majorities can itself be modeled as the AND of a single large majority. Put another way, this takes two PP computations and models it as a single computation - how to do this is not at all obvious, and naive approaches fail accordingly.

However, the equality of PostBQP and PP gives rise to a very simple quantum theoretic proof of the same result.

Theorem 4 *PostBQP is closed under intersection.*

Proof: Given two computations with two postselection criterion, postselect on them both being true, and then run the computations, accepting if both computations accept. □

Remark 4 *It might seem that could lead to answer whether $\text{PP} = \text{P}^{\text{PP}}$, using PostBQP . However, we have to be careful when reasoning about P with a PostBQP oracle. Well, we can't really chain the way that we want to, because P doesn't let us discard bad outcomes, which could create bad chains.*

Next time: time travel. Chosen by democracy!

References

- [1] Richard Beigel, Nick Reingold, and Daniel Spielman. Pp is closed under intersection. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 1–9, New York, NY, USA, 1991. ACM.
- [2] R. Cleve, P. Hyer, B. Toner, and J. Watrous. Consequences and limits of nonlocal strategies. In *Proceedings of 19th IEEE Conference on Computational Complexity*, pages 236–249, 2004.

Lecture 20

*Lecturer: Scott Aaronson**Scribe: Andy Drucker*

1 Last Time

In the previous lecture, we saw the result of Aaronson and Watrous that showed that in the presence of closed timelike curves (a model due to David Deutsch of ‘non-paradox-inducing’ time travel), classical and quantum computation are polynomially equivalent, as are polynomial time and space: $P_{CTC} = BQP_{CTC} = PSPACE_{CTC} = BQPSPACE_{CTC}$.

In the process of proving these results we also discussed superoperators, the most general form of operation on quantum systems, and described how fixed-points of these mappings are guaranteed to exist and can be found in polynomial space.

2 The Information Content of Quantum Systems

Complexity Theory does itself a disservice by inventing boring names for its cool ideas. Today’s lecture is about $P/poly$ and $BQP/poly$, but it’s ‘really’ about understanding the nature of quantum mechanics (QM).

There are three major ‘interpretations’ of the theory of QM: Many-Worlds, Bohmian, and Copenhagen/Bayesian. Empirically they make the same predictions, but they make different descriptions of the underlying state of the universe. In particular, they can be seen as having different estimations of the ‘information content’ of quantum systems.

A quantum state on n qubits is describable by a 2^n -dimensional complex unit vector. Is there ‘really’ an exponential amount of information in such a system? The Copenhagen interpretation suggests that this is so only to the extent that we consider a probability distribution on n -bit strings to contain exponential information: it takes that much to describe it fully, but we only learn n bits in our observation or measurement. This contrasts with the Many-Worlds perspective, in which all complex amplitudes of the quantum state vector are really ‘present’ as simultaneously existing alternative states of affairs.

Complexity theory can in effect stage ‘battles’ between these interpretations, raising questions about the extent to which we can extract and use information in quantum states. One such issue to explore is the extent to which quantum ‘advice’ helps us solve computational problems. To understand how this works, we need first to understand the classical theory of computing with advice.

2.1 Classical Advice

What does it mean to ‘compute with advice’? Suppose there is a trustworthy and all-knowing Advisor, who wants to help a Student solve problems. If the Advisor could see which computational problem the student was working on, s/he could simply state the answer. However, things get more interesting when we suppose that the Advisor is a busy and somewhat hard-to-reach person, who

tends to give out the same advice to all students. Let's say the Advisor gives out advice to a student trying to solve a decision problem ' $x \in L$?' that only depends on L and the length $|x|$... what kind of complexity class does this notion give rise to?

Definition 1 ($P/k(n)$) Define $P/k(n)$, ' P with $k(n)$ bits of advice', as the class of languages $L \subseteq \{0,1\}^*$ such that there exists a polynomial time Turing machine $M(x,y)$, and a collection $\{a_n\}_{n \in \mathbf{N}}$ of binary 'advice' strings, with $|a_n| = k(n)$, such that for all x , $M(x, a_{|x|}) = 1$ iff $x \in L$.

Now if $k(n) = 2^n$ and we're allowed random-access to the advice strings, we can encode any decision problem directly into the advice: $P/2^n = ALL$. Thus it's natural to restrict ourselves to polynomial advice:

Definition 2 ($P/poly$) Define $P/poly$, ' P with polynomial advice', as the class of languages $L \subseteq \{0,1\}^*$ such that there exists a polynomial time Turing machine $M(x,y)$, and a collection $\{a_n\}_{n \in \mathbf{N}}$ of binary 'advice' strings, with $|a_n| = O(poly(n))$, such that for all x , $M(x, a_{|x|}) = 1$ iff $x \in L$.

This class can readily be seen to coincide with the class of languages which have polynomial-sized *circuits*, whose designs may vary in arbitrary ways according to the input length. This arbitrary variation leads us to refer to this class and ones like it as 'nonuniform'. We could also think of these models as exploring the power of 'precomputing', where an exponential amount of computation goes into deriving which algorithm to use for a given length n . (Strictly speaking, though, $P/poly$ contains languages not computable using any uniform precomputing procedure.)

It is easy to see that $P/poly$ is a bigger class than P ; indeed, even $P/1$ is uncountable, hence is not even contained in the class of *recursive* languages! Nevertheless, we do know of some limits on the power of $P/poly$. For one thing, a simple counting argument shows that $P/poly$ does not contain the class of all languages, ALL . In fact we know $P/poly$ does not contain $ESPACE$, the class of problems computable in space $2^{O(n)}$ (and we can say things a bit stronger than this).

We also suspect other limits on $P/poly$. For instance, in the early '80s Karp and Lipton proved that if $NP \subset P/poly$ then the Polynomial Hierarchy collapses to its second level, i.e. $PH = NP^{NP}$. This is considered unlikely.

2.2 Quantum Advice

Now it's time to add Q's to everything... by analogy with $P/poly$, we make the following definition:

Definition 3 ($BQP/poly$). Define $BQP/poly$, ' BQP with polynomial advice', as the class of languages $L \subseteq \{0,1\}^*$ such that there exists a uniform family of polynomial sized quantum circuits $C_n(|x\rangle, |\psi\rangle)$, and a collection $\{a_n\}_{n \in \mathbf{N}}$ of binary 'advice' strings, with $|a_n| = O(poly(n))$, such that for every n and all x of length n , $C_n(|x\rangle, |a_n\rangle) = 1$ iff $x \in L$.

Similarly to before, we have $BQP/poly \neq BQP$. But the fun really starts when we allow our quantum algorithms to receive quantum advice:

Definition 4 ($BQP/qpoly$) Define $BQP/qpoly$, ' BQP with polynomial quantum advice', as the class of languages $L \subseteq \{0,1\}^*$ such that there exists a uniform family of polynomial sized quantum circuits $C_n(|x\rangle, |\psi\rangle)$, and a collection $\{|\psi_n\rangle\}_{n \in \mathbf{N}}$ of quantum 'advice' states on $O(poly(n))$ qubits, such that for every n and all binary strings x of length n , $C_n(|x\rangle, |\psi_n\rangle) = 1$ with probability $\geq 2/3$ if $x \in L$ while $C_n(|x\rangle, |\psi_n\rangle) = 0$ with probability $\geq 2/3$ if $x \notin L$.

So how big is $BQP/qpoly$? ‘High-information content’ interpretations of QM should at least suggest that it ought to be quite big, maybe even *ALL*. Notice that there are way more quantum states on n qubits than strings of length n , so the counting arguments that show obvious limits on $P/poly$ (and $BQP/poly$) no longer work here.

We could indeed try to encode an arbitrary boolean function f_n at each length n , say by preparing advice string $\frac{1}{2^{n/2}} \sum_x |x\rangle |f_n(x)\rangle$. The problem is how to extract this information. Measuring in the standard basis just tells us $(x, f(x))$ for some *random* x , not the one we’re actually interested in! (If we ‘postselect’ on getting the x we were interested in, however, we *can* compute any function: that is, $PostBQP/qpoly = ALL$.)

The *Group Membership* problem is in $BQP/qpoly$: if we consider a group G_n and subgroup $H_n \leq G_n$ as being fixed for each n , and we want to test if an input $x \in G_n$ is in H_n , Watrous’ algorithm allows us to do this given advice state $\frac{1}{\sqrt{|H_n|}} \sum_{y \in H_n} |y\rangle$. (Note that this works for any efficiently computable group operation. If we were using permutation representations of group elements, the problem would actually be in P .)

What other evidence do we have that *quantum* advice might be particularly useful for BQP machines? Aaronson has given a *quantum oracle* relative to which $BQP/qpoly$ properly contains $BQP/poly$. It’s a start...

Now let’s try to get a reasonable *upper*-bound on the power of quantum advice. Surely we can do better than *ALL*...

Theorem 5 $BQP/qpoly \subseteq PostBQP/poly$.

(And we know $PostBQP/poly = PP/poly \neq ALL$ by a counting argument.)

Proof (Sketch): Say $L \in BQP/qpoly$, computed by a family C_n of quantum circuits on advice $\{|\psi_n\rangle\}$.

Our first step is to amplify the success probability of C_n from $2/3$ to $1 - \frac{1}{2^n}$. This can be done by running the algorithm kn times, $k = O(1)$. We need a fresh copy of the advice for each run, so redefine the advice for length n as $|\varphi\rangle = |\psi_n\rangle^{\otimes kn}$.

In a full proof, we would now develop the ‘Almost as Good as New’ Lemma, which states: If the outcome of a computation’s final measurement is some value b with probability $(1 - \epsilon)$ when using advice $|\varphi\rangle$, then using $|\varphi\rangle$ in the computation leaves the advice in a new state $|\varphi'\rangle$ that is $\sqrt{\epsilon}$ -close to $|\varphi\rangle$ in trace distance.

Now the proof takes a seemingly crazy turn. Let I , the ‘maximally mixed state’, be uniform over all $p(n)$ bitstrings, where $p(n)$ is the number of qubits in $|\varphi\rangle$. We ask, does I work as advice in place of the $|\varphi\rangle$ we were previously using? Does it work on all inputs x of length n to help us compute $L(x)$ with success probability $2/3$? Probably not. Then, there exists an x_1 such that $C_n(|x_1\rangle, I)$ succeeds with probability less than $2/3$.

Let’s consider the state ρ_1 that is the residual state of the advice register after we ran C_n on (x_1, I) , *postselecting* on the event that we succeeded in outputting $L(x_1)$. We ask again: is ρ_1 good advice for C_n to use on every input? If not, some x_2 exists such that $C_n(|x_2\rangle, \rho_1)$ succeeds with probability less than $2/3$. Let ρ_2 be the residual state of the advice register after we ran C_n on (x_2, ρ_1) , postselecting on the event that we succeeded in outputting $L(x_2)$. And continue in this fashion for some $t(n)$ stages, t a polynomial. (A technical note: it can be shown that, since we started from the maximally mixed state I for which ‘anything is possible’, the events we postselect upon at each stage have nonzero probability, so this process can in fact be continued.)

If at any stage we cannot find an x_i to move forward, we must be holding a ρ_{i-1} that works as advice for every input, and we can use it to run the quantum circuit C_n on the input x we're actually interested in, succeeding with high probability. So, what we need to show is that the process *must* halt in polynomially many steps.

The maximally mixed state has a key property we exploit: it is a uniform superposition over basis states, not just over the basis of binary strings $\{|x\rangle : x \in \{0,1\}^{p(n)}\}$, but over *any* orthonormal basis (it is 'rotationally invariant'). In particular, it's uniform with respect to a basis that contains $|\varphi\rangle$, our 'good' advice state. Thus since advice $|\varphi\rangle$ yields the right answer on each x with probability at least $(1 - \frac{1}{2^n})$, I yields the right answer with probability at least $\frac{(1 - \frac{1}{2^n})}{2^{p(n)}}$. Similarly, since $|\varphi\rangle$ can be reused on each of $x_1, \dots, x_{t(n)}$ to give the right advice with probability $1 - o(1)$ (by the 'Almost as Good as New' Lemma), the probability that I succeeds on each of these inputs in turn is at least $\frac{(1-o(1))}{2^{p(n)}}$.

But we've designed this sequence of inputs so that the probability that I can be reused on each of them, succeeding at each stage, is less than $(\frac{2}{3})^{t(n)}$. To avoid a contradiction, the process can only continue for $t(n) \leq O(p(n))$ steps. Thus *there exist* $x_1, \dots, x_{O(p(n))}$ such that, if ρ is the residual state after we start with I and postselect on succeeding on each x_i in turn, ρ is a good advice string for *every* $x \in \{0,1\}^n$.

So, we just give this sequence of (classical!) strings to our *PostBQP* algorithm, along with the correct outcomes $b_1, \dots, b_{O(p(n))}$ for each of them. The algorithm prepares I (easily done), runs it on the advice input-sequence, and postselects on getting outcomes $b_1, \dots, b_{O(p(n))}$. The leftover state ρ necessarily has success probability $2/3$ when used as advice in C_n to determine if $x \in L$, for any desired $x \in \{0,1\}^n$. This completes the proof sketch.

■

Lecture 21

Lecturer: Scott Aaronson

Scribe: Jia Zheng (Colin)

Last time we introduced the advice “operator” and quantum complexity classes with quantum advice. We proved that $\mathbf{BQP}/qpoly$ is contained in $\mathbf{PostBQP}/poly$, by showing that the maximally mixed state need only be refined (iteratively) for polynomially many steps, so outcomes of the polynomially many inputs can be encoded in the advice and post-selected.

Let us consider $\mathbf{QMA}/qpoly$: how much more power does quantum advice bring? Scott’s paper upper bounds it by $\mathbf{PSPACE}/poly$. The kernel of the proof is to show the chain of inclusions $\mathbf{QMA}/qpoly \subseteq \mathbf{BQPSPACE}/qpoly \subseteq \mathbf{PostBQPSPACE}/qpoly = \mathbf{PSPACE}/qpoly$ (by Savitch’s hierarchy theorem). The first inclusion is non-trivial, as the $qpoly$ “operator” does not necessarily commute.

What about $\mathbf{PostBQP}/qpoly$? It is easy to see $\mathbf{PostBQP}/qpoly = \mathbf{PostBQP}/rpoly = \mathbf{ALL}$: for any boolean function f , we can encode it in the advice $\frac{1}{2^{n/2}} \sum_x |x\rangle |f(x)\rangle$, measure it in standard basis to get $(x, f(x))$ for some random x , and post-select on getting the x we are interested in.

Today we move on to quantum communication complexity.

Quantum state learning

Given a distribution \mathcal{D} over measurements, $E_1, \dots, E_m \in \mathcal{D}$, some n -qubit state $|\phi\rangle$ and $P_i = \Pr[E_i \text{ accepts } |\phi\rangle]$, we can ask the following: How many measurement samples are enough to learn $|\phi\rangle$? Or how many classical bits are needed to describe $|\phi\rangle$?

Theorem. $O(n)$ measurement samples suffice to learn a n -qubit state $|\phi\rangle$.

Still, just like problems in \mathbf{QMA} , finding the $|\phi\rangle$ consistent with all E_i is hard.

Holevo’s theorem

Consider the scenario where Alice holds an n -bit string x , how many qubits must Alice transfer, in order for Bob to output x ?

Theorem. (Holevo, 1973) n qubits can represent no more than n classical bits.

This is a surprising result, contrasting the many scenarios where quantum computing/information is inherently more powerful than classical.

Holevo’s theorem assumes that Alice and Bob do not share entangled qubits. When they do share EPR pairs, still it can be shown that at least $n/2$ qubits are needed (the $n/2$ technique is called *superdense coding*, described below).

Superdense coding

With a shared EPR pair between Alice and Bob, a single qubit may convey 2 bits of information. Alice simply sends her part of the EPR pair altered according to the 2 bits,

such that the EPR pair becomes one of:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}, \frac{|10\rangle + |01\rangle}{\sqrt{2}}, \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \frac{|10\rangle - |01\rangle}{\sqrt{2}}$$

each corresponding to one of the possibilities of 2 classical bits. Since they are orthogonal pure states, Bob can recover x by measuring simply in this basis. With n shared EPR pairs n qubits can convey $2n$ bits of information. It can be shown that the factor is 2 is tight.

Quantum random access codes

Suppose Alice holds an n -bit string x , and Bob holds some integer i . How many qubits must Alice transfer, so that Bob can find x_i with high probability? (The two-way communication version is less interesting as Bob can send i with $\log n$ bits and Alice sends back x_i .) Quantumly, one can have a factor-of-2 saving with bounded error, without requiring entanglement (contrast this with Holevo's theorem and superdense coding). The qubits sent by Alice are called *quantum random access codes* as they let Bob retrieve x_i for any i , but information about x_j for $j \neq i$ are lost due to measurement.

It is not known whether better than constant-factor saving can be achieved quantumly, but Scott conjectures (i.e. this is a total function separating quantum and classical communication complexity).

The idea is very simple, due to Ambainis, Nayak, Ta-Shma and Vazirani (1999). Let Alice send the state $|\phi_x\rangle$ illustrated in the figure:

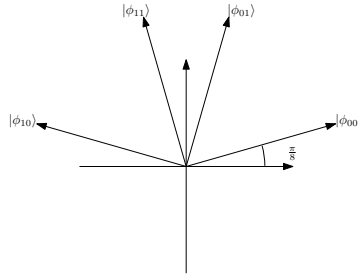


Figure 1: Quantum random access code of 2 classical bits

Bob can learn x_1 by measuring in the standard basis, or learn x_2 in the standard basis rotated $\frac{\pi}{4}$ counterclockwise. In either case probability of the desired outcome is $\cos^2 \frac{\pi}{8} \approx 0.85$. Classically it can be shown that strictly more than $n/2$ bits are needed to have bounded error.

ANTV 1999 has also proved a lower bound $\Omega(n/\log n)$ on any bounded error quantum communication protocol, with the help of Holevo's theorem.

Proof. (Sketch) Suppose a communication protocol below $\Theta(n/\log n)$ exists, with error bound $1/3$. Run it $c \log n$ times for some constant c (amplitude amplification), so that error is bounded below $1/n^c$. In this new protocol Alice sends less than $\Theta(n)$ qubits. Now, the “Almost as Good as New” lemma says that if measurement succeeds with probability at least $1 - \epsilon$, then the state is “damaged” (in terms of trace distance) by at most $\sqrt{\epsilon}$. Plugging $1/n^c$ into this lemma, it can be shown that Bob can find x_i for all i , which contradicts Holevo's theorem! \square

Remark. Alternatively, we can show the $\Omega(n/\log n)$ bound using last Thursday's argument: $D^1(f) \in O(mQ^1(f) \log Q^1(f))$. (The notations D^1, Q^1 are defined below.) Here $m = 1$, so $Q^1(f)$ below $\Theta(n/\log n)$ implies $D^1(f)$ below $\Theta(n)$ (which is, of course, false).

Communication complexity

Let $f : \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}$. Suppose Alice holds some $x \in \{0, 1\}^n$, Bob holds some $y \in \{0, 1\}^m$ and wants to compute $f(xy)$ with Alice's help.

- The deterministic one-way communication complexity $D^1(f)$ is the minimum number of bits Alice has to send to Bob. $D^1(f)$ is equal to the number of distinct rows in the communication matrix of f .
- The randomized one-way communication complexity $R^1(f)$ is the shortest string sampled from a distribution, that Alice has to send to Bob, so that Bob can find $f(x, y)$ with bounded error.
- The quantum one-way communication complexity $Q^1(f)$ is the minimum number of qubits Alice has to send to Bob, so that Bob can find $f(x, y)$ with bounded error by a measurement. Alice may send a mixed state as in the randomized case. Since each mixed state can be represented by a pure state with qubits doubled, pure states are good enough for asymptotic bounds on Q^1 .

Two-way deterministic, randomized, and quantum communication complexity are defined by allowing Bob to send back to Alice, and there is no constraint on number of rounds of communication. In terms of how much we know today, communication complexity is often seen to be between query complexity (where most is known) and computational complexity (where least is known).

Let us consider a simple example, “equality of two n -bit strings”:

$$\text{EQ}(x, y) = \begin{cases} 1 & (x = y) \\ 0 & (x \neq y) \end{cases}$$

Like many other functions, the deterministic communication complexity is n . $R^1(\text{EQ})$ however is exponentially smaller in this case. The idea is fingerprinting. Let $A = \{p \text{ prime}, p \leq n^2\}$, for a randomly chosen p from A the probability that $x = y \pmod p$ is $\frac{|\{p \in A, p | (x-y)\}|}{|A|}$. There are at most n prime factors of $x - y$ but $|A| \in \Theta(n^2 / \ln^2 n)$ by the prime number theorem. Thus from $y \pmod p$ and p , Bob can decide whether $x = y$ with bounded error, i.e. $R^1(\text{EQ}) = O(\log n)$.

How about $Q^1(\text{EQ})$? We don't know much more other than $Q^1(\text{EQ})$ is $\Omega(\log \log n)$. The central question is, are there functions where R^1 and Q^1 are asymptotically separated? Exponentially separated? How about in the two-way communication setting? We will answer these questions in the next lecture.

Lecture 22

*Lecturer: Scott Aaronson**Scribe: Jia Zheng (Colin)*

(Plan of the remaining classes: next Tuesday on classical simulation of quantum circuits, next Thursday Quantum Open House for topics you'd like to hear more about, next next Tuesday project presentation.)

Last time we talked about several results in the setting of one-way communication, which is perhaps the simplest form. Of course we also care about two-way communication, and will discuss problems in both settings today. Basically Alice and Bob each hold some string and want to collectively accomplish some job; we are concerned with how much information they must communicate, rather than the computational feasibility.

We talked about Holevo's theorem, which shows the perhaps surprising result that qubits cannot encode asymptotically more classical bits. We have also seen random access coding, where Bob is only interested in a single bit of Alice's string. But even when Bob only needs to find a single bit of his choice unknown to Alice, asymptotic separation—if it exists at all—is no more than a logarithmic additive factor. Today we'll see some separation results.

Notation. $D^1(f)$, $R^1(f)$, $Q^1(f)$ are the one-way deterministic, bounded-error randomized, and bounded-error quantum (respectively) communication complexities. $D(f)$, $R(f)$, $Q(f)$ are the two-way deterministic, bounded-error randomized, and bounded-error quantum (respectively) communication complexities.

Remark. $D^1(f)$ is simply the number of distinct rows in the communication matrix. It can be shown that the definition of R^1 is not changed if zero-error is required. There can be exponential separation between D^1 and R^1 , e.g. for “equality of two n -bit strings”,

$$\text{EQ}(x, y) = \begin{cases} 1 & (x = y) \\ 0 & (x \neq y) \end{cases}$$

Clearly $D^1(\text{EQ}) = n$, $R^1(\text{EQ}) \in \Theta(\log n)$ as we have seen last lecture.

Remark. While looking at asymptotic separation, we can safely confine our attention to pure states (for the qubits to transfer), as any mixed state can be represented by a pure state with twice as many qubits. Mixed states sometimes *do* save a factor of 2, e.g. for EQ. As a side note, we can show $Q^1(\text{EQ}) \in \Omega(\log n)$ by a counting argument. Let $|\psi_x\rangle$ denote the qubits Alice sends to Bob. For every $x \neq x'$ we want $|\psi_x\rangle$ and $|\psi_{x'}\rangle$ to be sufficiently different. How many k -qubit states are there where no two of them have small inner product? The number is in the order of 2^{2^k} . The intuition is to see states as 2^k -bit strings and consider error correcting codes. So with k asymptotically below $\log n$, k qubits cannot help distinguish $2^{2^{\log n}}$ different values of x .

We have seen exponential separation between R^1 and D^1 . Is there such separation between Q^1 and R^1 ? Today we know the answer to be affirmative for partial functions, as partial functions provide much leeway in the definition. But for total functions we are still clueless.

Exponential separation between Q^1, R^1 for partial f

Gavinsky et al. found some partial f where $R^1(f) \in \Theta(\sqrt{n})$ but $Q^1(f) \in O(\log n)$, based on prior work of Bar-Yossef, Jayram and Kerenidis.

Suppose Alice knows the gender x_1, \dots, x_n of n people where half are male and half female, Bob knows a perfect matching in which either (i) boys are matched to boys, girls to girls, or (ii) every boy is matched to a girl. How much must Alice send, to help Bob decide which case (orientation)?

Remark. If Bob can communicate to Alice he can simply send a pair in the matching using $\log n$ bits to hear back if they are gay.

If Alice sends genders of a random set of people, how large must the set be to contain at least some pair in Bob's matching? This is the birthday paradox, so $R^1(f) \in O(\sqrt{n} \log n)$. The $\log n$ factor used to store the index can in fact be eliminated, giving $R^1(f) \in O(\sqrt{n})$.

Let Alice simply send the equal superposition of all x_i (superposition of basis states $|i\rangle$ phase-shifted by x_i):

$$\frac{1}{\sqrt{n}} \sum_{i=1}^n (-1)^{x_i} |i\rangle$$

Let \mathcal{M} denote Bob's matching. Bob makes up some function satisfying $g(i) = g(j)$ iff $(i, j) \in \mathcal{M}$, and computes

$$\frac{1}{\sqrt{n}} \sum_{i=1}^n (-1)^{x_i} |i\rangle |g(i)\rangle$$

Measuring the first qubit in a basis containing $\frac{|i\rangle+|j\rangle}{\sqrt{2}}, \frac{|i\rangle-|j\rangle}{\sqrt{2}}$ for any $(i, j) \in \mathcal{M}$ will collapse the state to

$$\frac{(-1)^{x_i} |i\rangle + (-1)^{x_j} |j\rangle}{\sqrt{2}}$$

for some $(i, j) \in \mathcal{M}$. Then $x_i \oplus x_j$ (i.e. gay or straight) can be readily found by measuring in Hadamard basis. We can see that this $O(\log n)$ protocol is zero-error.

The hard part of the separation proof is, as usual, to show the lower bound $\Omega(\sqrt{n})$ for $R^1(f)$. This is also from Gavinsky et al.

No strong separation when Bob's input is small

It turns out (Aaronson 2004) that $D^1(f) \in O(mQ^1(f) \log Q^1(f))$ for all partial and total f , where m is the length of Bob's input. The proof technique is the same as in the argument that $\mathbf{BQP}/qpoly \subseteq \mathbf{PostBQP}/poly$ where we start with a maximally mixed state. In this setting Alice has to specify each input bit of Bob, that is why there is a factor of m . This result implies that m must be large for any function to provide strong separation between Q^1 and D^1 . For large m , there can indeed be exponential separation as we saw above.

The Inner Product Problem

This is a problem that seems specifically designed to maximize communication complexity. Alice and Bob hold vectors x and y ; they want to find the inner product mod 2.

Unsurprisingly, the classical and randomized communication complexities are both $\Omega(n)$. As it turns out, there is an incredibly slick proof that $Q^1(f)$ and $Q(f)$ are also $\Omega(n)$.

Proof. Suppose there is a quantum protocol requiring less than n qubits. Then Bob can run the protocol on superposition of all possible inputs:

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

Given this state, Hadamard it would give Bob the state x , just like the Bernstein-Vazirani problem. This means Bob can learn Alice's input using less than n qubits, contradicting Holevo's theorem. You can prove Holevo's theorem even in the setting of two-way communication, so $Q(f)$ cannot be below $\Theta(n)$. (If Alice and Bob share entangled qubits, we can save a factor of 2 but no more, so the bounds remain). \square

Exponential separation between Q, R for partial f

Raz 1999 defined a problem where, albeit highly contrived, qubits exponentially outperform classical bits. Alice holds some $\vec{v} \in \mathbb{C}^n$ and bases of two orthogonal subspaces S_1, S_2 of \mathbb{C}^n , Bob holds a unitary matrix U , how much must they communicate to find whether $U\vec{v} \in S_1$ or $U\vec{v} \in S_2$ (provided that one of them holds)? Raz proved that no classical protocol is below $\Omega(n^{1/4}/\log n)$. The quantum protocol is ridiculously simple (and you guessed it): Alice sends \vec{v} , Bob sends back $U\vec{v}$!

The Disjointness Problem

To this day we do not know of any total function that asymptotically separates Q^1 and R^1 (Scott conjectures such functions to exist), but we do have a candidate where Q and R are only quadratically related (although we know of no function that exponentially separates them).

The asymptotic separation between Q, R on the Disjointness Problem (or more aptly, the non-disjointness problem) was found by Buhrman-Cleve-Wigderson 1998. Consider the situation where Alice and Bob each have a set of available dates (as n -bit strings x and y); how much must they communicate to find a commonly available date (an i with $x_i = y_i = 1$)? That $D(\text{DISJ}) \in \Omega(n)$ is obvious, $R(\text{DISJ}) \in \Omega(n)$ is proved by Razborov, as well as Kalyasundaram-Schnitger.

In the quantum case, what happens if we just plug in Grover's algorithm?

1. Alice prepares the state $\frac{1}{\sqrt{n}} \sum_{i=1}^n |i\rangle |x_i\rangle$ and sends to Bob
2. Bob applies phase shift and sends back $\frac{1}{\sqrt{n}} \sum_{i=1}^n (-1)^{x_i y_i} |i\rangle |x_i\rangle$
3. Alice applies Grover's operator on $\frac{1}{\sqrt{n}} \sum_{i=1}^n (-1)^{x_i y_i} |i\rangle$ and sends to Bob
4. ...

This "distributed Grover's algorithm" shows that $Q(\text{DISJ}) \in O(\sqrt{n} \log n)$, where the $\log n$ factor is needed to represent an index. In fact Razborov 2002 proved that $Q(\text{DISJ}) =$

$\Omega(\sqrt{n})$, the proof ultimately invokes the polynomial method on multiple variables and applies symmetrization.

Aaronson-Ambainis 2003 showed that the bound is tight by producing a $O(\sqrt{n})$ communication protocol.

This result in Aaronson-Ambainis 2003 comes from the study (in the same paper) of an unrelated problem: using Grover's algorithm "in practice", in a model where input bits are spatially separated (like Turing machine rather than random access machine) but can have any physically realistic layout (i.e. not restricted to the 1-d "tape" layout). Thus to query an input bit the quantum robot (like head of a Turing machine) must move to that input bit. If input is a 1-d matrix, Grover's algorithm is no faster than simply walking through the input bits. Likewise if the database is 2-d, as moving to the queried bit requires $O(\sqrt{n})$ steps (along both axes), i.e. $O(n)$ steps for $O(\sqrt{n})$ queries. If the database is a 3-d matrix, one can show that $O(n^{5/6})$ time is needed. On dimension high enough it has been shown that $O(\sqrt{n})$ steps suffice for quantum random walks. Aaronson-Ambainis shows how to archive $O(\sqrt{n})$ time in 3-d. The basic idea is to apply Grover recursively:

Suppose the database is 2-d. Divide it into \sqrt{n} equal subsquares. Each subsquare can be searched in $O(\sqrt{n})$ steps classically, so if we apply Grover to query which of the \sqrt{n} subsquares contains a 1-entry, $O(\sqrt[4]{n}\sqrt{n}) \in O(n^{3/4})$ time is enough. Now let us recursively do this: use the $O(n^{3/4})$ algorithm on each subsquare. This idea will eventually lead to $O(\sqrt{n} \log^2 n)$ running time. (There are more technical concerns, such as taking care of increased error probability due to repeated Grover). On a 3-d database this approach takes $O(\sqrt{n})$ steps.

Aaronson-Ambainis uses this idea to define a $O(\sqrt{n})$ communication protocol, where x and y are treated as 3-d matrices. Rather than communicate the indices as done in the $O(\sqrt{n} \log n)$ protocol, Alice and Bob communicate their moves in the matrices, which is no more than $O(\sqrt{n})$.

Brief summary

For one-way communication: (i) we do not know of any total function separating Q^1, R^1 asymptotically; (ii) we do know partial functions that separate Q^1, R^1 even exponentially.

For two-way communication: (i) we do know total functions that separate Q, R asymptotically (quadratically), but not exponentially; (ii) we do know partial functions that separate Q, R exponentially.

Lecture 23

Lecturer: Scott Aaronson

Scribe: Bhaskar Mookerji (mookerji@mit.edu)

Last time,

1. **2-way communication:** exponential separation for promise problem (Raz), quadratic separation for total function, unknown if there's an exponential separation for total function.
2. **1-way communication:** exponential separation for promise problem, unknown if there's any asymptotic separation for total function. Problem that I conjecture gives a quadratic separation: group membership.

In our last class, we considered some examples of separations in quantum one-way and two-way communication. We don't know any total function for which we can prove that quantum one-way communication gives you any asymptotic advantage over classical one-way communication. We do have a candidate for such a problem: group membership. Suppose that there is some finite group G that is known to both Alice and Bob. Alice has a subgroup $H \leq G$ and Bob has some element $x \in G$. Alice can send a message M_H to Bob that encodes information about H . To specify any subgroup, she requires $\log^2 N$ bits, where N is the order of the group, but only $\log N$ qubits. The open problem here is to prove that any classical communication protocol only requires $\log^2 N$ bits. Note that if G is Abelian, then there is no separation between classical and quantum communication in this problem. If such a separation were to exist, then G would have to be badly non-Abelian.

1 Classical Simulations of Quantum Circuits and Stabilizers Circuits

It turns out that quantum circuits that can be efficiently simulated classically: not quite a unified field, but rather a collection of interesting ideas that people have had. Two reasons I like this area:

- Unlike almost everything else in this course, it actually has some practical applications today! Physicists and chemists care about these sorts of things, particularly density functional theory.
- Illustrates the subtlety of quantum speedups.

It seems that almost all sets of gate we choose are universal, however, this is not necessarily true. We'll start off by considering stabilizer circuits and the Gottesmann-Knill theorem.

Theorem 1 (Gottesmann-Knill) *Any circuit composed of CNOT, Hadamard and phase gates can be efficiently simulated on a classical computer even though such circuits can generate huge amounts of entanglement, and can be used for superdense coding, quantum teleportation, the GHZ paradox, quantum error-correcting codes, etc.*

The idea here is that we can specify a quantum state $|\psi\rangle$ by listing a bunch of unitary matrices that stabilize $|\psi\rangle$, i.e., such that $U|\psi\rangle = |\psi\rangle$. Note that if U and V stabilize $|\psi\rangle$, then UV and U^{-1} also stabilize $|\psi\rangle$. Thus, the set of stabilizers of $|\psi\rangle$ forms a group, called the stabilizer group. We're going to specify a state by giving a set of generators for its stabilizer group. If we start with some initial computational basis state $|00\rangle$, then the actions of the stabilizers gates are:

$$\begin{aligned} |00\rangle &\rightarrow |00\rangle + |10\rangle \\ &\rightarrow |00\rangle + |11\rangle \\ &\rightarrow |00\rangle + i|11\rangle \end{aligned} \tag{1}$$

for a Hadamard, CNOT, and phase gate, respectively. You'll notice that we can only produce equal superpositions over some affine subspace using these gates.

If a state can be generated (starting from $|00\dots 00\rangle$) by CNOT, Hadamard, and phase gates, it turns out that its stabilizer group consists of particularly simple unitaries: namely, tensor products of Pauli matrices:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2}$$

The squares of these matrices resolve the identity

$$X^2 = Y^2 = Z^2 = I, \tag{3}$$

and their products are closed under cyclic permutation

$$\begin{aligned} XY &= iZ = -YX \\ YZ &= iX = -ZY \\ ZX &= iY = -XZ. \end{aligned} \tag{4}$$

They behave exactly as quaternions, in case you've seen those.

As an example, let's look at the two-state computational basis. Their stabilizers are given by:

Gate	Stabilizer Set
$ 0\rangle$	$\{I, Z\}$
$ 1\rangle$	$\{I, -Z\}$
$\frac{1}{\sqrt{2}}(0\rangle \pm 1\rangle)$	$\{I, \pm X\}$
$\frac{1}{\sqrt{2}}(0\rangle \pm i 1\rangle)$	$\{I, \pm Y\}$

The two-qubit initial state is stabilized by $\{ZZ, IZ, ZI, II\}$, so its stabilizers representation is $\{ZI, IZ\}$. In our classical simulation of a quantum circuit, this list of generators is going to be our succinct representation of the quantum state. How many bits does that representation require, for an n -qubit state? Well, each Pauli matrix takes 2 bits, so each generator requires $2n$ bits actually $2n + 1$, since there might be a minus sign in front. And it turns out we always have exactly n generators (yielding a stabilizer group of order $2n$). So, $n(2n + 1)$ bits total.

The key question, of course, is how to update this representation when a CNOT, Hadamard, or phase gate is applied. Rather than prove rigorously what happens, I'll just illustrate the rules through examples. Let's apply these gates to an initial tensor product state $|00\rangle$ and see what happens,

Gate	Stabilizer Generators
$ 00\rangle$	$\{IZ, ZI\}$
$\frac{1}{\sqrt{2}}(0\rangle + 1\rangle) 0\rangle$	$\{XI, IZ\}$
$\frac{1}{\sqrt{2}}(0\rangle + i 1\rangle) 0\rangle$	$\{YI, \pm IZ\}$
$\frac{1}{\sqrt{2}}(0\rangle - i 1\rangle) 0\rangle$	$\{-XI, \pm IZ\}$

In general, after Hadamarding the i th qubit, we go through the i th column of the matrix swapping X 's with Z 's, and Y 's with Y 's. For the phase gate, go through the i th column mapping X to Y to X to Y . For the CNOT gate, $XI \rightarrow XX$, $IX \rightarrow IX$, $ZI \rightarrow ZI$, and $IZ \rightarrow ZZ$. From these rules, we can show that $YI \rightarrow YX$.

One thing I haven't told you is how to handle measurement. Given a stabilizer state, every measurement of a qubit in the standard basis returns the outcome $|1\rangle$ with probability either 0, 1, or $1/2$. Our task is to figure out which. Well, if the i th row contains an X or Y , then the probability is going to be $1/2$. Why? Because if the i th qubit were either definitely $|0\rangle$ or definitely $|1\rangle$, it couldn't possibly be stabilized by X or Z . If the i th row contains only I 's and Z 's, then the probability is going to be either 0 or 1. Why? Because in that case, one can prove that either $IIZII$ or $IIZII$ (i.e., a Z at the i th qubit and I 's everywhere else) must be in the stabilizer group. If $IIZII$ stabilizes the state, then the measurement outcome has to be $|0\rangle$, while if $IIZII$ stabilizes it, then the measurement outcome has to be $|1\rangle$. So we just have to figure out which. But this is a linear algebra problem, which is solvable in polynomial time using Gaussian elimination! Total running time: $O(n)$ per CNOT, Hadamard, phase gate, $O(n^3)$ to simulate a measurement.

In 2004, I decided to implement the Gottesman-Knill simulation algorithm for a course project, but I didn't feel like implementing Gaussian elimination. So Gottesman and I developed a better algorithm, which uses only $O(n^2)$ time to simulate a measurement. We also proved that the problem of simulating a stabilizer circuit is complete for the class $\oplus L$, a subclass of P (under L -reductions). To put it another way, stabilizer circuits have exactly the same computational power as circuits with CNOT gates only. So they're presumably not even universal for classical computation.

2 Match Gates

For a given $n \times n$ matrix A , we can define the *permanent* and the *determinant*,

$$\text{per}(A) = \sum_{\sigma} \prod_{i=1}^n a_{i\sigma(i)} \longleftrightarrow \det(A) = \sum_{\sigma} (-1)^{\text{sign}(\sigma)} \prod_{i=1}^n a_{i\sigma(i)}. \quad (5)$$

Theorem 2 (Valiant) *Permanent is $\#P$ -complete.*

Since $\text{BQP} \subseteq \text{P}^{\#P}$, this means in particular that the problem of simulating a quantum computer can always be reduced to computing the Permanent of some matrix. Well, but that doesn't help much, since the Permanent is $\#P$ -complete! As we know, determinant is computable in classical polynomial time (using Gaussian elimination).

So, if only there were some interesting subclass of quantum computations that reduced to the determinant instead of the permanent! Indeed there is such a subclass—Valiant called them matchcircuits, and wrote a paper developing the theory of them purely mathematically. Then something funny happened: Terhal and DiVincenzo wrote a follow-up paper pointing out that what Valiant had essentially done is rediscovered fermions.

So far in this course, we haven't had to say anything about particles, but it turns out that particles come in two types: bosons and fermions. Bosons are generally force particles like photons; fermions are generally matter particles like quarks. Mathematically, the crucial difference between them has to do with how you add up amplitudes. Let's say we have 2 identical particles that don't interact with each other, and we just want to know the amplitude for them evolving into some new configuration. How can we calculate that configuration? Well, we can take the amplitude for this particle going here multiplied by the amplitude for that particle going there. What else do we need to consider? Right: the cross term. So we get $ab + cd$. Seems obvious! And indeed, that's exactly what happens with bosons.

But there's another possibility, which is realized by fermions: that we have to take $ab - cd$. (You might wonder, why not $cd - ab$? Well, we can only determine the amplitude up to a sign, but amplitudes are only ever determined up to a sign anyhow!) In general, suppose we have n identical, non-interacting particles, and let a_{ij} be the amplitude for particle i going to position j . Form the matrix $A = (a_{ij})$. Then if the particles are bosons, the amplitude for this whole process is $\text{per}(A)$, while if they're fermions, the amplitude for this whole process is $\det(A)$.

3 The Future

Open problems!

Lecture 24

Lecturer: Scott Aaronson

Scribe: Brendan Juba

1 Quantum circuits that can be efficiently simulated

Last time, we saw the Gottesman-Knill Theorem, which says that any circuit composed of CNOT, Hadamard, and phase gates can be simulated in classical polynomial time. We also began our discussion of Valiant’s Matchgates. In this lecture, we finish our discussion of Valiant’s Matchgates, and also describe Vidal’s efficient classical simulation of quantum computation with limited entanglement.

1.1 Valiant’s Matchgates

1.1.1 Bosons vs. Fermions

Recall that last time, we saw that there were two fundamentally different kinds of particles: “bosons,” which were force particles like photons; and “fermions,” which were matter particles like quarks. In a system of identical non-interacting particles, we calculate the amplitudes for states of the system in future configurations in fundamentally different ways for these two kinds of particles. Observe, in Figure 1, the two particles could enter the same configuration by either taking paths a and b or by taking the paths labeled c and d :

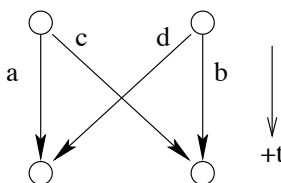


Figure 1: Two identical particles can enter identical configurations by either taking paths a and b , or by “switching places” and taking paths c and d .

The difference between bosons and fermions is that for bosons, the amplitude for this final configuration is given by $ab + cd$, i.e., the amplitudes for these two pairs of paths *add*, whereas for fermions, the amplitude is given by $ab - cd$, i.e., the paths interfere with one another. (Generally, one looks at the sign of the permutation of the particles among the various positions in the configuration corresponding to each term.)

One might wonder how we know whether the amplitude should be $ab - cd$ or $cd - ab$. The simple answer is that we *don’t* know, and moreover it *doesn’t matter*—a global phase shift is undetectable by an observer and either one of these will assign the same probabilities to observations. We only know that the universe does it in some *consistent* way.

It’s worth remarking that, if the two final positions are the same position, then the amplitudes ab and cd are the same, where we find that bosons still add, giving an amplitude of $2ab$ (they end

up on top of each other—for example, a laser consists of many photons “on top of each other”), whereas the two terms for fermions *cancel each other out*, i.e., they have amplitude $ab - cd = 0$. This is interpreted as saying that fermions, in contrast to bosons, “take up space” and is known to physicists as the “Pauli exclusion principle” or “Fermi pressure.” The amplitudes for a fermion are spread out, somewhat (like a Gaussian distribution), so the cancellation of the amplitudes has the effect of keeping fermions from getting too close to each other. It turns out that this is what prevents neutron stars from collapsing.

The n -particle generalization of this is as follows: suppose we have n identical, non-interacting particles, and for $i, j \in \{1, \dots, n\}$ a_{ij} denotes the amplitude for a single particle going from position i in some configuration to position j in some other configuration of the n particles (i.e., a_{ij} is calculated imagining that no other particles are present). Now, for the matrix A such that a_{ij} is the (i, j) entry, the amplitude for the prescribed final configuration is given by $\text{per}(A)$ if the particles are bosons, whereas it is given by $\det(A)$ if the particles are fermions. To repeat, it was crucial that we assumed that the particles were non-interacting and identical (no interference occurs for distinct particles—two states that differ in any way do not interfere in quantum mechanics). We know that, although $\det(A)$ and $\text{per}(A)$ look superficially similar, they are extremely different computationally: the former can be computed in classical polynomial time, whereas the latter is $\#P$ -complete.

Another remark is in order: although calculating the amplitudes for a configuration of bosons reduces to computing the Permanent – a $\#P$ -complete problem – this doesn’t necessarily imply that we can use a system of bosons to solve $\#P$ -complete problems, only that the configuration can be computed with a $\#P$ -oracle—something we already knew since we saw $\text{BQP} \subseteq P^{\#P}$, and we believe that BQP faithfully models what can be computed using quantum mechanics. We don’t expect for this containment to be an equality. We expect that the instances of permanents arising from bosons are of a special form (that can be simulated in BQP), and thus we don’t expect that such systems of non-interacting identical particles can be set up for arbitrary, hard instances.

1.1.2 Matchgates

The link between all of this talk of particle physics and quantum circuits is roughly that a system of identical non-interacting fermions can be simulated in classical polynomial time: suppose we have a quantum circuit in which every gate is a “matchgate,” i.e., has a two-qubit unitary of the form

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & a & c & 0 \\ 0 & d & b & 0 \\ 0 & 0 & 0 & ab - cd \end{bmatrix}$$

for some a, b, c , and d , and suppose our initial state is, for example $|01100010\rangle$, or more generally, any n -bit string. Suppose we want to compute the amplitude for some outcome after applying our circuit to this initial state, e.g., $|10010100\rangle$.

Our first observation is that this is trivial if the strings have different Hamming weights—the amplitude will be zero in this case, since our gates preserve the Hamming weight. The second observation is that if the strings have the same Hamming weight, we can reinterpret the $|1\rangle$ bits as fermionic particles and the $|0\rangle$ bits as “vacuum,” and then calculate the final amplitude as follows: for each individual “fermion” in the input string, and each individual “fermion” in the output string,

we calculate the amplitude for the input fermion ending up in that position in the output string, i.e., the amplitude for a transition between two strings of Hamming weight 1, such as $|01000000\rangle$ and $|00000010\rangle$ in our example. This computation can be done in classical polynomial time since it only involves keeping track of n amplitudes at each time step, and if there are k particles in each configuration, we repeat this k^2 times in total. We then form the $k \times k$ matrix of these amplitudes, and calculate its determinant to obtain the amplitude for this final configuration, which we know can also be done in classical polynomial time. This is correct since we see that in mapping $|11\rangle$ to itself, the gate introduces a factor of $ab - cd$ to the amplitude, exactly as it should for a fermion, and the preservation of the Hamming weight corresponds to particle being neither created nor destroyed. (This matrix of transition amplitudes is apparently also known as the “Jordan-Wigner transformation.”)

More generally (though we won’t get into too much detail here) we can use “Pfaffians” to calculate the probability of measuring some qubit to be a $|1\rangle$ after the circuit is applied. Using an algorithm that is closely related to the algorithm for counting the number of perfect matchings in a planar graph, it turns out that we can also calculate this probability in classical polynomial time. In fact, still more general kinds of circuits can be simulated in classical polynomial time in this way. For example, we can include terms that correspond to fermions being created and annihilating each other in a vacuum, and use any gate of the form

$$\begin{bmatrix} x & 0 & 0 & y \\ 0 & a & c & 0 \\ 0 & d & b & 0 \\ z & 0 & 0 & w \end{bmatrix}$$

such that $ab - cd = xw - yz$, provided that gates act on adjacent qubits on a line only. Strictly speaking, we don’t even need to require that these matrices are unitary for the simulation to be efficient, only for the correspondence with particle physics to hold.

1.2 Quantum computation with limited entanglement

There is one other class of quantum circuits that can be efficiently simulated on a classical computer: circuits with limited entanglement. For example, pure states which are always unentangled can easily be simulated classically, since we only need to write $2n$ amplitudes (rather than 2^n) on each time step. More generally, we consider the following way of measuring the entanglement of a quantum state:

Definition 1 (Schmidt rank) *The Schmidt rank χ of a bipartite state (on two subsystems) $|\psi\rangle$ is given by the minimum number such that we can write*

$$|\psi\rangle = \sum_{i=1}^{\chi} \lambda_i |a_i\rangle \otimes |b_i\rangle$$

It turns out that the Schmidt rank can be computed efficiently by diagonalizing the density matrix on one of the two sides, and finding how many different eigenvalues it has. In general, the Schmidt rank may be as high as $2^{n/2}$; it turns out that in some cases, when it is small (polynomially bounded) we can efficiently simulate the circuit.

Given an n -qubit state, let χ_{\max} denote the maximum of χ over all bipartitions of the n qubits. Then we have

Theorem 1 (Vidal) *Suppose a quantum computation involves nearest-neighbor interactions only among qubits on a line, and that χ_{\max} is polynomially bounded at every step. Then the computation can be efficiently simulated on a classical computer.*

The simulation is essentially carried out via dynamic programming – for each qubit, we store a $\chi \times \chi$ matrix encoding how each qubit interacts with the other qubits (in terms that may be familiar from general relativity, we use contraction of tensors to obtain a more compact representation). We then show that this compact representation can be locally updated for nearest-neighbor operations, and that we can obtain the probabilities from them. Thus, we only need to store $O(n\chi^2)$ amplitudes on each time step to simulate these circuits.

This algorithm is not meant for simulating physical systems—this is for simulating quantum circuits, i.e., applying nearest-neighbor unitaries. Vidal also later developed a related algorithm for simulating physical systems with nearest-neighbor Hamiltonians or for estimating the ground states where there is limited entanglement, and then was able to use this algorithm to solve some problems in condensed-matter physics where the states of interest had dimensions too high to deal with using existing techniques. It turns out that cases with low entanglement encompass most cases that physicists care about, since creating entanglement is hard—this is, after all, what makes building quantum computers so hard in the first place.

2 Grab bag

2.1 Counterfactual Computing [Josza-Mitchison]

This is best described as a cute observation about quantum algorithms. It turns out that it’s easiest to explain using the example of Grover’s algorithm.

$$\overbrace{[|\dots\rangle |i\rangle |\dots\rangle]}^N$$

marked

Suppose we’re searching a list of N items using Grover’s algorithm. If the i th item is marked, then we find it with high probability in $O(\sqrt{N})$ queries. Notice, on the other hand, that if *no* item is marked, then we also learn *that* with high probability in $O(\sqrt{N})$ queries by the algorithm’s failure.

At this point, we ask the slippery question, “what is the probability that the i th location was queried?” Strictly speaking, this question does *not* make sense, but we could imagine that we measured the query register at each time step, and use the probability of outcome $|i\rangle$ as this probability. Then, since in the absence of any marked item, we query using the uniform superposition at every step, the probability of querying location i at each step is $1/N$. Since there are $O(\sqrt{N})$ steps, the “total probability” with which we query location i is $\sim 1/\sqrt{N}$. Thus, one could say that we learn that item i is not marked, but almost certainly without querying it.

2.1.1 Vaidman’s Bomb

We could cast the whole problem more dramatically as follows: suppose the i th position being unmarked corresponds to there being a bomb in the i th location, where the bomb explodes if that position is queried. Imagining ourselves in the role of the Quantum Bomb Squad, we would like to know whether or not there is a bomb in the i th location without querying it and exploding the

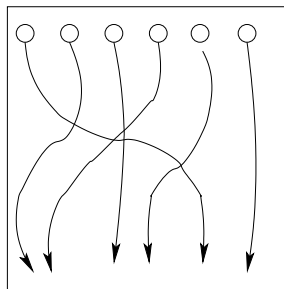


Figure 2: Nonabelian anyons being “passed” across each other in a simulated 2-dimensional surface

bomb. Provided that one can enter a coherent superposition of querying the bomb’s location and not querying it, the problem is solved: for some large N , we query each location with amplitude $1/\sqrt{N}$. For most of the locations we do nothing. If we query the i th location and there is a bomb there, then that branch of the wave function is killed. If there isn’t, we can switch the phase of the amplitude of the i th position (from $1/\sqrt{N}$ to $-1/\sqrt{N}$) and use Grover amplification to increase the amplitude of querying that location in the future, and make another query. After repeating this process $O(\sqrt{N})$ times, if there is no bomb, we learn that with constant probability (since in this case we actually simulate Grover’s algorithm finding the marked i th item), but if there is a bomb, again, the “probability that we query it” is only $\sim 1/\sqrt{N}$ since we die before we would perform the Grover amplification step (and hence, this corresponds to running Grover’s algorithm with no marked item). By taking N sufficiently large, we can make the “probability” of dying arbitrarily low. (Needless to say, this is unlikely to actually work in practice due to the difficulty of entering a coherent superposition, which is made particularly difficult by the possibility of an explosion, which would tend to be highly decoherent.)

Thus, counterfactual computing is the study of in what settings you can learn about something without having to interact with it. (In the case of the bomb, though, the reason that we learn anything is that *if it’s safe*, then we look in the i th position.) Simple classical analogues of this also exist: suppose we have two processes, P_1 and P_2 , where P_1 is computing some boolean function f and P_2 is idle. Suppose that if $f(x) = 1$, then P_1 kills P_2 . Then, if we come back and check that P_2 is still running after P_1 would have completed, then we could learn that P_1 computed $f(x) \neq 1$ from P_2 , even though P_2 never computed f and (in this case) P_1 never interacted with P_2 . In any case, Roger Penrose (apparently) applied counterfactual computation to propose a scheme where, if one was Orthodox Jewish, one could switch on a light on the Sabbath without having to flip the switch.

2.2 Topological Quantum Computation

This is an architectural proposal for implementing a quantum computer that is distinguished by involving “a huge amount of interesting math.” (This is what attracted the attention of Alexei Kitaev and Mike Freedman.) A *nonabelian anyon* is a kind of particle (cf. bosons and fermions) that provably can’t exist in three dimensions, but can exist in two dimensions (although it isn’t clear that they have ever been observed). By analogy, we saw that when bosons switch places, nothing happens, whereas when fermions switch places, “the universe picks up a minus sign;” we could also imagine particles that, when they switch places, some arbitrary group action or some

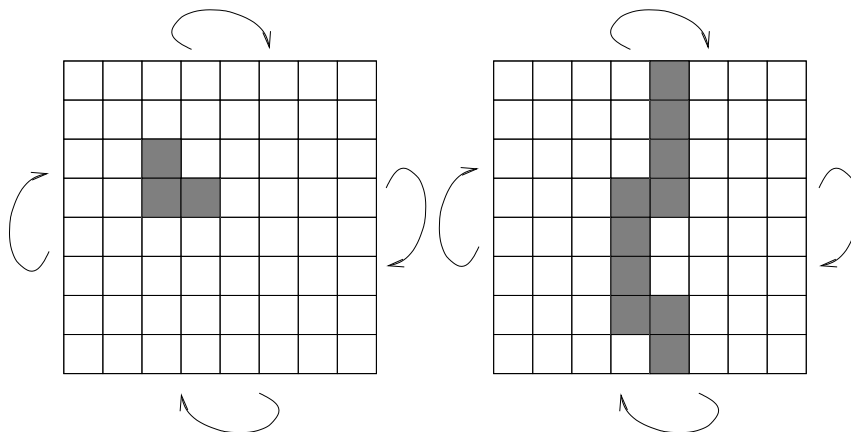


Figure 3: Shaded regions represent sets of errors that are topologically trivial (left) and nontrivial (right) on a torus. Computation might fail in the second case, but not in the first.

arbitrary unitary is applied to the universe—some more general kind of symmetry. No fundamental particles that we have observed in nature are anything like these nonabelian anyons, but supposing we confine things to some two-dimensional lattice, we could hope to build composite particles (“quasiparticles”) that simulate the behavior of these nonabelian anyons. If we could do this, then just by creating n nonabelian anyonic quasiparticles and swapping them past each other in various ways, we could perform a universal quantum computation (see Figure 2). There are lots of interesting connections to knot theory here.

Of course, all of our various proposals which can perform universal quantum computations are theoretically equivalent to one another. The upshot of this proposal is that fault-tolerance comes for free. This is in contrast to our quantum circuit architecture, where we relied on the Threshold Theorem—a complicated hierarchical error-correction scheme operating on all of the qubits in parallel (constantly measuring them and correcting errors) that guaranteed that a constant threshold on the error rate was sufficient to allow arbitrarily long quantum computations. Actually implementing the scheme described in the Threshold Theorem has been extremely difficult. In the topological architecture, fault-tolerance arises from the intrinsic physics of the system: we don’t even measure the actual paths—we only measure topological invariants, and thus we care about the global topology. For example, in a related scheme for topological error-correction, The only way for our computation to be incorrect would be for qubits along a topologically nontrivial path to be corrupted (see Figure 3). As an aside, it would be much easier to make these schemes work if space had four dimensions, rather than three.

2.3 Proposals for more powerful “realistic” models of computation

We’ve been talking for the whole course about what quantum mechanics is, and how it changes the structure of computational complexity theory. A natural thing to wonder is, if we were surprised once by the laws of physics, what further surprises could there be? Could there be another model of computation beyond quantum computing?

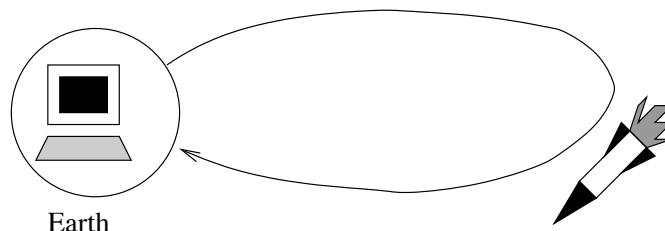


Figure 4: A “relativistic computer” – an observer sets a computation running and travels at relativistic speeds away from Earth and back, to read off the result of the computation.

2.3.1 Quantum Field Theory Computation

We’ve already discussed some such speculative ideas in this course, such as quantum computers with closed timelike curves and quantum computers with post-selection. In terms of physics, people have wondered (for example) if quantum field theories could yield another, more powerful model of computation. Quantum field theories combine quantum mechanics with special relativity (and hopefully, one day, will include general relativity as well). We could ask, would building a computer with these principles in mind yield a larger complexity class than BQP?

This question has been very hard to address since these quantum field theories are not even mathematically well-defined. No one can prove that they even exist! (Proving that they make any kind of mathematical sense is related to one of the Clay problems.) These theories are, to some extent, “hacks” that work up to a certain energy scale, but beyond it they give nonsensical answers. We can’t study computational complexity in this kind of a world.

There is one special class, called *topological quantum field theories*, which we can work with concretely (and for which results exist). In these theories, we have two space dimensions, one time dimension, and no matter or energy—just points and cuts that can move around on a $(2 + 1)$ -dimensional manifold (so the only degrees of freedom are topological). This is the simplest interesting class of quantum field theories—they are studied because they *can* be given rigorous mathematical definitions. These were rigorously studied by Witten in the ’80s.

These topological quantum field theories are directly related to a knot invariant called the *Jones polynomial*, a function of knots which was invented for independent reasons. In 2000, Freedman, Kitaev, Larsen, and Wang defined a corresponding computational problem of simulating topological quantum field theories, and showed that it is BQP-complete, so these topological quantum field theories give computational power equivalent to that granted by quantum mechanics. Since simulating these topological quantum field theories was equivalent to estimating the Jones polynomial for a knot, their simulation also implied an efficient quantum algorithm for estimating the Jones polynomial. Their paper was *extremely* difficult to understand, though. Recently, Aharonov, Jones, and Landau also gave a direct BQP algorithm for estimating the Jones polynomial (which is also BQP-complete, essentially by Witten’s work).

2.3.2 Relativistic Computation

We don’t think we can use a quantum computer to solve NP-complete problems; what about a classical relativistic computer? One proposal is immediate: we set a classical computer to work on some very hard problem, board a space ship, and travel away from the Earth and back at close to

the speed of light (see Figure 4). Then, as in the twins paradox, billions of years have elapsed in the Earth’s reference frame, (your friends are long dead, the sun is a red giant, etc.) but you can read off the answer from your computer. Alternatively, you could set your computer working, and move *extremely* close to the event horizon of a black hole to slow down the rate of time in your reference frame (from the perspective of an external observer) to the same effect.

Assuming you’re willing to pay the price for this, there are additional problems with these proposals: in contrast to classical computer science, where energy is not an issue (we know, e.g., by Landauer’s work that the dissipation of energy is not required by classical computation), approaching relativistic speeds consumes a lot of energy. In particular, to obtain an *exponential* speed-up, since the our time dialation at velocity v is given by the $\frac{1}{\sqrt{1-v^2}}$ factor, an exponential speed-up requires that v is exponentially close to the speed of light—but then, the amount of energy we consume similarly involves a $\frac{1}{\sqrt{1-v^2}}$ factor, so it requires an exponential amount of energy, and thus an exponential amount of fuel is necessary.

Thus, to study relativistic computation, we’d need to consider the energy requirements. But, too much energy in a bounded volume creates a black hole, so the amount of energy and physical space required are *also* related. Thus, in this case, and exponential time speed-up requires an exponential amount of physical space—and thus, since the speed at which the fuel in the far part of a fuel tank is limited by the speed of light, we’d still incur an exponential time overhead.

Now, on the other hand, people have studied circuits with extremely limited computational models, for example threshold circuits in which only a limited number of the gates can be on, which corresponds in a reasonable way to a energy limitation (this was studied in a recent Complexity paper where they prove some interesting things) but this was all done within polynomial factors.

It’s fair to say that we would like a quantum theory of gravity to tell us whether or not it is possible, for example, to pack an large number of bits into a small region of space. We won’t be able to do any kind of experiment in the foreseeable future involving both tiny particles and entire solar systems and galaxies to test any proposed theory of quantum gravity, e.g., string theory, but these questions about the limits of computation – can you build a closed timelike curve? can you build a stable wormhole? or, can you spawn off a “baby universe” to do an unlimited amount of computation for you? – are things that physicists cannot answer because they do not have a quantum theory of space and time.

2.4 Open problem: Public-key cryptography secure against quantum polynomial time adversaries

One topic of particular contemporary interest is the construction of new schemes for public-key cryptography that could be secure against quantum adversaries. (By contrast, *private-key* cryptosystems are not known to be broken by quantum polynomial time adversaries, in general.) One of the reasons for all of the excitement about Shor’s algorithm was that the security of cryptosystems based on the RSA function depend crucially on the presumed intractibility of factoring. In fact, most of the currently known constructions for public-key cryptosystems are based on problems in abelian groups – RSA, elliptic curves, Diffie-Hellman, Buchman-Williams, and El Gamal all are – and are therefore not secure against a quantum polynomial time adversary. The one family of known exceptions to this are based on a proposal by Ajtai and Dwork (and later improved by Regev) based on the problem of finding the shortest vector in an integer lattice—essentially, based on instances of the hidden subgroup problem for the dihedral group: a nonabelian group. Thus,

they are not *known* to be secure against a quantum adversary; they are merely not known to be broken. There are also proposals based on braid groups which are not known to be broken by a quantum adversary, but at the same time are not even known to be secure against even a classical adversary under any standard hardness assumptions. An alternative but related subject is quantum key distribution based on the uncertainty principle—by sending polarized photons over a fiber-optic cable, you can obtain a cryptosystem from this quantum mechanical assumption. This is much easier to do than quantum computing since it doesn't involve entanglement, and it has been demonstrated experimentally, but it requires a quantum communication network for coherently sending photons over long distances, which is a hard engineering problem and does not yet exist (and yet, devices for quantum key distribution are already on the market).

The approximate shortest vector problems used by Ajtai-Dwork are *not* NP-complete, but a variant of these problems are, which naturally leads to the long-standing open question of whether or not cryptography can be based on the assumption that $P \neq NP$ —whether or not we can base a cryptosystem on a NP-complete problem. This question is as old as the modern, complexity-theoretic approach to cryptography, and if we believe that NP-complete problems cannot be solved efficiently on a quantum computer, would solve our problem, but the current sentiment is that it seems unlikely. In the first place, the kind of hardness that we need for cryptographic constructions is not just that infinitely many hard instances exist (as suffices for $P \neq NP$ —this would correspond to there *existing* hard-to-decode messages rather than *most* messages being hard to decode) but that there is some NP-complete problem such that the average case is *provably* as hard as the worst case, then we need a way to efficiently sample from this hard distribution to construct a one-way function (which is essential for any kind of cryptography), and then beyond even *that* we actually need a trapdoor one-way function (one that can be efficiently inverted given an additional secret key) for public-key cryptography. In the first place, the worst-case to average-case reduction seems unlikely to exist, and it has been ruled out for nonadaptive reductions by Bogdanov and Trevisan (unless the polynomial-time hierarchy collapses to the third level), building on a similar classic result by Feigenbaum and Fortnow ruling out nonadaptive reductions to instances chosen uniformly at random, itself extending a beautiful argument by Brassard that one-way permutations cannot be based on such reductions unless $NP=co-NP$. Stronger negative results were recently obtained for the specific problem of basing the constructions of one-way functions on NP-hardness by Akavia, Goldreich, Goldwasser, and Moshkovitz, and also by Pass.

2.5 Open problems: a partial list

We didn't get to discuss any more topics in lecture. The following is a list of suggested open problems and topics for research in quantum complexity theory:

- What is the power of quantum computing with separable mixed states? Closely related: the one-clean-qubit model (Ambainis-Schulman-Vazirani: no gate-by-gate simulation).
- Are there gate sets that yield intermediate power between quantum and classical?
- What is the interplay between noise rates and computational complexity? Related: does $BQP = BPP^{BQNC}$?
- Can the nonabelian hidden subgroup problem be solved in quantum polynomial time? Is there an efficient quantum algorithm for graph isomorphism?

- BQP vs. the polynomial hierarchy
- What is the need for structure in quantum speed-ups: P vs. BQP relative to a random oracle. Is there a function with permutation symmetry yielding a superpolynomial speed-up?
- Quantum algorithms for approximating #P-complete problems and its dual: #P-complete problems arising from quantum computation.