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

Lecture 1: Introduction to the Quantum Circuit Model

September 9, 2015

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

1 Overview of what is to come

1.1 An incredibly brief history of quantum computation

The idea of quantum computation was pioneered in the 1980s mainly by Feynman [Fey82, Fey86] and Deutsch [Deu85, Deu89], with Albert [Alb83] independently introducing quantum automata and with Benioff [Ben80] analyzing the link between quantum mechanics and reversible classical computation. The initial idea of Feynman was the following: Although it is perfectly possible to use a (normal) computer to simulate the behavior of n-particle systems evolving according to the laws of quantum, it seems be extremely inefficient. In particular, it seems to take an amount of time/space that is exponential in n. This is peculiar because the actual particles can be viewed as simulating *themselves* efficiently. So why not call the particles themselves a "computer"? After all, although we have sophisticated theoretical models of (normal) computation, in the end computers are ultimately physical objects operating according to the laws of physics. If we simply regard the particles following their natural quantum-mechanical behavior as a computer, then this "quantum computer" appears to be performing a certain computation (namely, simulating a quantum system) exponentially more efficiently than we know how to perform it with a normal, "classical" computer. Perhaps we can carefully engineer multi-particle systems in such a way that their natural quantum behavior will do *other* interesting computations exponentially more efficiently than classical computers can.

This is the basic idea behind quantum computers. As it turns out, you *can* get (seemingly) exponential speedups for a (seemingly) small number of natural computational problems by carefully designing a multi-particle quantum system and letting it evolve according to the (100-year old, extremely well-confirmed) laws of quantum mechanics. By far the most spectacular example is Shor's factoring algorithm [Sho97], an algorithm implementable on a quantum computer that can factor any n-digit integer (with high probability) in roughly n^2 time. This is contrast to the fact that the fastest known "classical" algorithm for factoring n-digit integers seems to require roughly $2^{n^{1/3}}$ time, and in fact the presumed computational difficulty of factoring is relied upon in an enormous number of real-world cryptographic applications (e.g., the computations done whenever you type `https://` into your browser).

1.2 Plans for this course and this lecture

Very briefly, in this course we will:

- *Mathematically* formulate the tiny bit of quantum mechanics that is relevant for the field of quantum computation. (We should mention that this course will be heavily slanted towards theoretical computer science and mathematics, and will contain almost no physics.)

- See some quantum algorithms that solve certain computational problems much faster than they are known to be solvable classically.

- Investigate the *limits* of quantum computation. (It is *not* the case that quantum computation automatically provides speedup over classical computation for all problems, or even for a wide class of problems.)

- Study some quantum information theory.

The goal for this first lecture is to give a lightning-fast, as-barebones-as-possible definition of the quantum circuit model of computation. After this lecture, you will theoretically know all you need to know in order to implement and analyze, e.g., Shor's algorithm. (Of course, we will subsequently make a more thorough and leisurely development of quantum computation before actually getting around to sophisticated algorithms.)

90% of the understanding of the quantum circuit model is achieved by reviewing three purely "classical" topics: classical Boolean circuits; reversible classical circuits; and randomized computation. The first and third of these topics should be very familiar to anyone who has studied the basics of theoretical computer science. And the second topic is very cute and elementary. Once we have these three concepts in hand, quantum circuits become practically just a tiny "twist" on randomized computation — what you might get if you tried to invent a model of randomized computation in which "probabilities" can be *negative*…

2 Classical Boolean circuits

Several models of computation/algorithms are studied in the classical theory of computation: Turing Machines, high-level programming languages, and Boolean circuits. It turns out that for the study of quantum computation, the Boolean circuit model is by far the easiest model to generalize (being as it the closest model of the physical reality of computers).

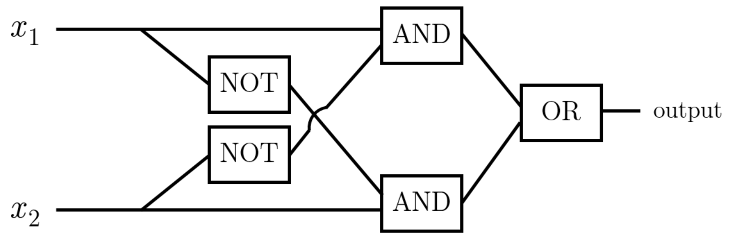
We begin with the following well known fact, stating that any computational task (modeled by a Boolean function) we might want to do is doable with an AND/OR/NOT Boolean circuit.

Proposition 2.1. *Any Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ is computable by a Boolean circuit C using just* AND, OR, *and* NOT *gates. I.e.,* AND, OR, *and* NOT *gates are* universal.

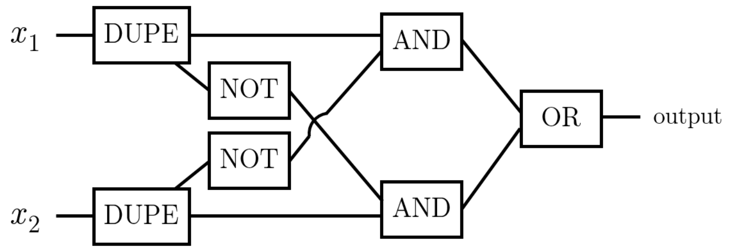
Remark 2.2. The AND and OR gates mentioned in this proposition take 2 input bits and produce 1 output bit. The NOT gate takes 1 input bit and produces 1 output bit.

Remark 2.3. Once we know that every Boolean function is computable by some circuit, we usually become interested in computing it *efficiently*; i.e., with a circuit C of small *size*. The size of the circuit, size(C), is defined to be the number of gates it uses. Circuit size fairly closely corresponds to *running time* in the Turing Machine (sequential algorithm) model. For example, it is known that a circuit of size s can be evaluated in time $O(s \log s)$ by a Turing Machine, and conversely, a Turing Machine operating in time t on length-n inputs can be converted to an n-input circuit of size $O(t \log t)$.

Here is a simple example of a circuit computing the XOR function, $f(x_1, x_2) = x_1 \oplus x_2$:

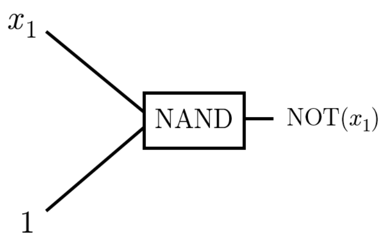


The lines in this diagram are called "wires", and the things inside the rectangles are called "gates". In the diagram we have followed a traditional circuit-theory convention by allowing wires to "branch"; i.e., split into two copies. In reality, some physical mechanism must exist at these branches, and in the future it will be convenient to make this explicit. So we will introduce a new kind of gate called a DUPE (*duplicate*) gate which takes 1 input bit and outputs 2 duplicate copies of that bit. We will then redraw the above diagram as follows:



With this convention, it would be more accurate to say that AND, OR, NOT, and DUPE gates are universal for Boolean circuit computation.

It is also a well known fact that one can get smaller universal gate sets; in fact, one can replace AND/OR/NOT gates with just NAND gates. (Recall that NAND(x_1, x_2) = NOT(AND(x_1, x_2)).) To see this, first note that we can eliminate OR gates using De Morgan's rule: OR(x_1, x_2) = NOT(AND(NOT(x_1), NOT(x_2))). Then we can eliminate AND gates in favor of NAND gates via AND(x_1, x_2) = NOT(NAND(x_1, x_2)). Finally, we need to show that NOT gates can be eliminated using NAND gates. One way to implement NOT(x_1) with a NAND gate is as follows:



On the lower left in this diagram, we have what is called an *ancilla* bit: an input that is "hardwired" to the constant bit 1, for the purposes of assisting the computation. It's actually possible to implement NOT(x_1) using NAND and DUPE without the use of ancillas (specifically, via NAND(DUPE(x_1))). However the above method gives us a good opportunity to introduce the notion of ancillas.

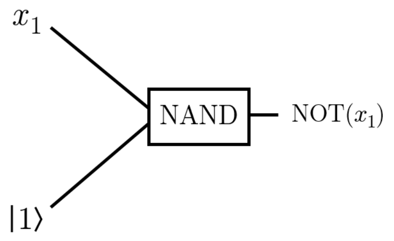
What we have just shown is the following:

Proposition 2.4. *Boolean* NAND *and* DUPE *gates (along with the use of ancillas) are* universal *for computation.*

Remark 2.5. In fact, we have shown something stronger: Not only can every AND/OR/NOT/DUPE circuit C be converted to an equivalent AND/OR/NOT circuit C', this conversion can be done very *efficiently*; there is an efficient algorithm carrying out the conversion, and $\text{size}(C') = O(\text{size}(C))$.

2.1 Bra-ket notation

We take this opportunity to introduce a bit of unusual notation that will play an essential role in the remainder of the course. This is the "bra-ket" notation invented by Paul Dirac. Actually, we will postpone the mathematical definitions to the next lecture; for now we will just introduce it as pure symbolism. We will henceforth enclose bits and bit-strings in asymmetrical brackets called *kets*, writing $|0\rangle$ and $|1\rangle$ instead of 0 and 1. We will also usually eliminate internal brackets when writing strings; e.g., writing $|011\rangle$ instead of $|0\rangle\,|1\rangle\,|1\rangle$. As a small example of this notation, we will redraw the previous diagram as follows:



3 Reversible computation

In actual physical reality, a theoretical bit ($|0\rangle$ or $|1\rangle$) is implemented by a particle or bunch of particles (e.g., high or low voltage on a physical wire). Similarly, a gate is implemented by a physical object (a "switch" or some other gadget) that manipulates the bit-representations. We then would ideally like to think of the circuit as a "closed physical system". Unfortunately, for a typical AND/OR/NOT/DUPE circuit, this is not possible. The reason is that the laws of physics governing microscopic systems (both classical and quantum) are *reversible* with respect to time, but this is not true of most gates we would like to physically implement.

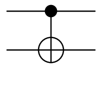
Take for example an AND gate. Suppose its output is $|0\rangle$. Can we infer what its inputs were? The answer is no — they could have been $|00\rangle$, $|01\rangle$, or $|10\rangle$. The AND process is not reversible: information sometimes needs to be deleted; "entropy" is lost. According to the 2nd Law of Thermodynamics, a physical system consisting of a single AND gate cannot be "closed"; its operation must dissipate some energy — typically as escaping heat. On the other hand, a NOT gate *is* theoretically "reversible": its output can be determined from its input; no information is created or destroyed in switching $|0\rangle$ to a $|1\rangle$ or vice versa. Thus, in principle, it is possible to construct a completely closed physical system implementing a NOT gate, without the need for energy dissipation.

These issues were studied in the 1960s and 1970s by Landauer [Lan61] and Bennett [Ben73], among others. They raised the question of whether there are Boolean gates that are both reversible and universal. If so, then by using them it would be possible — at least according to the theoretical laws of physics — to have circuits doing general computation without dissipating any energy. On one hand, as we will see shortly, it *is* possible to find universal reversible gates. On the other hand, it turned out that from a practical point of view, the energy dissipation of standard electronic circuits did not prove to be a major problem (although laptops sometimes *do* get rather hot in your lap). On the other other hand, it turns out to be important for the quantum circuit model that universal reversible computation is possible. So we will now explain how to do it. We begin with a definition:

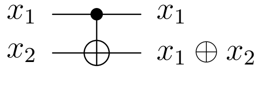
Definition 3.1. A Boolean gate G is said to be *reversible* if it has the same number of inputs as outputs, and its mapping from input strings to output strings is a bijection.

Thus a NOT gate is reversible, whereas most other "standard" gates (e.g., AND, OR, NAND, and DUPE) cannot be reversible since they do not have an equal number of inputs and outputs.

Let's introduce a new, simple, reversible gate, the CNOT (*controlled*-NOT) gate. It has 2 input bits and 2 output bits, and is drawn like this:



Its behavior is as follows:



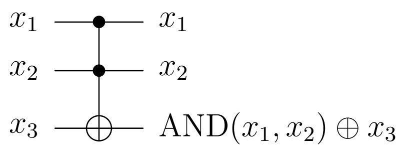
That is, the first input bit x_1 is always passed through directly; the second bit gets NOT applied to it if and only if the "control" bit x_1 is $|1\rangle$. To be even more explicit, CNOT has the following truth table:

CNOT:

input	output
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

You can see that this mapping is indeed a bijection, confirming that CNOT is a reversible gate.

We now describe a small but important generalization, called the CCNOT (*controlled-controlled*-NOT) or *Toffoli gate*. The below diagram indicates how this 3-input, 3-output gate is drawn, as well as its behavior:



In other words, the first two inputs to a CCNOT gate are passed through directly, and the third input is negated if and only if the first two "control" input bits are both $|1\rangle$.[1] Explicitly, we have the following truth table, showing that CCNOT is reversible:

CCNOT:

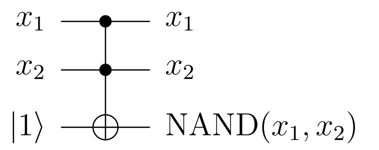
input	output
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$\|111\rangle$
$\|111\rangle$	$\|110\rangle$

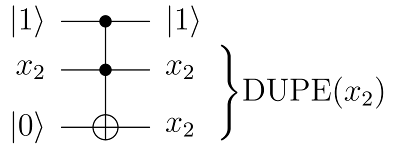
Remark 3.2. The three examples of reversible gates we have seen so far — NOT, CNOT, CCNOT — also have the extra property that *they are their own inverse*; i.e., applying them twice in succession restores the original bits. This is a bit of a coincidence, insofar as it is *not* a property we insist on for reversible gates. We merely insist that reversible gates have *some* inverse gate; the inverse doesn't have to be the gate itself.

The CCNOT gate is extremely handy: as the following two pictures show, we can use it to simulate both NAND gates *and* DUPE gates (assuming, as always, that ancillas are

[1] In general, we use the convention that attaching a dot to a k-input/k-output gate G with a vertical line means creating a "controlled-G" gate. This is the $(k+1)$-input/$(k+1)$-output gate that passes through its first, "control", bit, and which either applies G or doesn't depending on whether the control bit is $|1\rangle$ or $|0\rangle$. Assuming that a NOT gate is drawn as $\oplus$, this explains the picture used for CNOT and CCNOT gates.

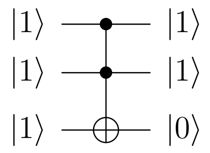
allowed):





Note that, in addition to producing the desired NAND and DUPE outputs, these conversions also produce extra, unneeded bits (namely, x_1 and x_2 in the NAND case, and the top $|1\rangle$ in the DUPE case). This is somewhat inevitable, given that reversible gates are required to have equally many input and output bits. We call such unwanted outputs *garbage*.

As one more very minor note, the above conversions use both $|0\rangle$ ancillas and $|1\rangle$ ancillas. We can also use CCNOT gates to generate $|0\rangle$'s from $|1\rangle$ ancillas as follows:



We have therefore established the following key theorem, which shows that we can do universal computation reversibly.

Theorem 3.3. *The* CCNOT *gate is universal, assuming ancilla inputs (all set to* $|1\rangle$*) and garbage outputs are allowed; any standard* AND/OR/NOT *circuit for a function* $f : \{0,1\}^n \to \{0,1\}^m$ *may be efficiently transformed into a reversible one that looks like Figure 1.*

Remark 3.4. In reversible circuits we will always have $n + \#\text{ancillas} = m + \#\text{garbage}$.

Remark 3.5. "In practice", when doing reversible computing we usually also allow ourselves NOT and CNOT gates. (Given NOT gates, we may assume that all ancillas are fixed to $|0\rangle$ rather than $|1\rangle$, and this is actually a more traditional assumption.)

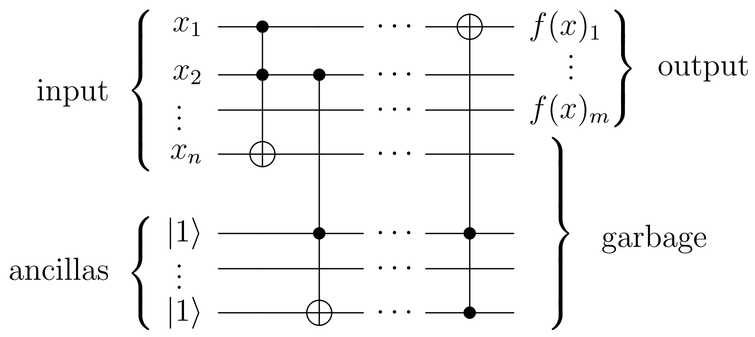


Figure 1: A typical reversible circuit using CCNOT gates. We remark that the output bits need not be the "topmost" m bits on the right; we could designate any of the m bits on the right as the outputs.

With "standard" circuits, the number of wires carrying a bit at any one "time" (vertical slice) may vary. However with reversible circuits, this will always equal the number of inputs+ancillas, as you can see above. Indeed, one sort of stops thinking about wires and instead thinks of each input/ancilla bit being carried in its own *register*, which maintains its "identity" throughout the computation. It's very helpful to think of circuits not just as diagrams but also as "lists of instructions performed on registers", as in the following description, which is completely equivalent to the diagram in Figure 1:

Input is in registers $x_1, x_2, \ldots, x_n$.
"Attach" c ancillas in $x_{n+1}, \ldots, x_{n+c}$, initialized to $|1\rangle$.

- CCNOT(x_1, x_2, x_n)
- CCNOT(x_2, x_{n+1}, x_{n+c})
- $\cdots$
- CCNOT(x_{n+c}, x_{n+1}, x_1)

Output is in registers $x_1, \ldots, x_m$.

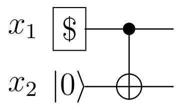
4 Randomized computation

Although we are well used to it now, randomized computation is a bit like the "quantum computation of the '60s and '70s" — a creative new twist on classical computation, seemingly realizable in practice and potentially allowing for big speedups over deterministic computation, but one requiring its investigators to make a big investment in a new area of math (i.e., probability). Indeed, there are some computational tasks which we know how to provably solve efficiently using randomized computation, but which we don't know how to provably solve efficiently using only deterministic computation. (An example: on input "n", generate an n-digit prime number.) Unlike with quantum computation, however, we believe this is mainly due to our lack of skill in proving things, rather than an inherent major advantage of randomized computation. (E.g., we know a deterministic algorithm that we *believe* efficiently generates n-digit prime numbers; we just can't prove its efficiency.)

It is very easy to upgrade the circuit model of computation to a randomized model: we just introduce a single new gate called the COIN gate, drawn like this: $\boxed{\text{COIN}}$— or $\boxed{\$}$—. It has 0 inputs and 1 output; the output is a "fair coin flip", viz., $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$.

Remark 4.1. You might also imagine allowing other kinds of randomized gates; for example, a COIN$_{\frac{1}{3}}$ gate that outputs $|1\rangle$ with probability $\frac{1}{3}$ and $|0\rangle$ with probability $\frac{2}{3}$. It turns out that allowing such gates does not fundamentally change the model of randomized computing; although a fair COIN gate cannot simulate a COIN$_{\frac{1}{3}}$ gate *exactly*, it can simulate it close-to-exactly enough that it doesn't really matter. For this reason we say that the plain COIN gate is (effectively) *universal* for randomized computation. See Homework 1 for more details.

We will now describe how to "analyze" randomized circuits. Although our style of analysis will be very simple and pedantic, it will be great practice for analyzing the closely related quantum circuits. Here is an example randomized circuit:



Alternatively, we could think of this circuit as the following "program":

1. x_1 initialized to $\boxed{\$}$
2. x_2 initialized to $|0\rangle$
3. CNOT(x_1, x_2)

As you can easily see, the output of this circuit is $|00\rangle$ with probability $\frac{1}{2}$ (if the coin flip is $|0\rangle$) and is $|11\rangle$ with probability $\frac{1}{2}$ (if the coin flip is $|1\rangle$). Nevertheless, let's patiently "analyze" it.

The state of x_1 after the coin flip — equivalently, after Line 1 of the program — is

$$\frac{1}{2} \text{ probability of } |0\rangle\,, \quad \frac{1}{2} \text{ probability of } |1\rangle\,. \tag{1}$$

Let us introduce some funny notation for this:

Notation 4.2. *We will write* (1) *as*

$$\frac{1}{2}\cdot|0\rangle + \frac{1}{2}\cdot|1\rangle\,.$$

In the next lecture we will "make mathematical sense" of this notation using linear algebra, but for this lecture you should be perfectly happy just treating it as some "formal notation".

The state of x_2 after Line 2 of the program is

$$1 \text{ probability of } |0\rangle\,, \quad 0 \text{ probability of } |1\rangle\,,$$

which we will write in our new notation as

$$1\cdot|0\rangle + 0\cdot|1\rangle = |0\rangle\,.$$

Here we have used the "usual" laws and notation of arithmetic in the equality. (Again, continue to think of this as shorthand notation.)

Finally, what happens at the end of the circuit, after Line 3? One could say that we have:

$$\begin{aligned}\text{State of } x_1 : \quad & \frac{1}{2}\left|0\right\rangle + \frac{1}{2}\left|1\right\rangle \\ \text{State of } x_2 : \quad & \frac{1}{2}\left|0\right\rangle + \frac{1}{2}\left|1\right\rangle\end{aligned}$$

While in some sense this is true (each "register" *is* equally likely to be $|0\rangle$ or $|1\rangle$), it's grossly misleading. It makes it looks as if the two bits are independent, when in fact they are **correlated**. So the above analysis is true but incomplete; to truly capture the correlations in the system we should say:

$$\text{Joint state of } x_1, x_2 : \quad \frac{1}{2}\left|00\right\rangle + \frac{1}{2}\left|11\right\rangle .$$

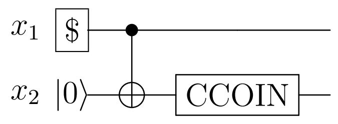
In our analyses of randomized circuits, we will keep track of the joint state of all registers all along. For example, in the circuit we have been analyzing the joint state just *prior* to Line 3 would be

$$\frac{1}{2}\left|00\right\rangle + \frac{1}{2}\left|10\right\rangle .$$

Let's practice some more analysis of "r-bit circuits" (where "r" standards for "randomized"). For the purposes of practice we'll invent a new randomized gate, $-\boxed{\text{CCOIN}}-$ ("controlled-coin"), which has 1 input and 1 output. Its behavior is the following:

$$\text{CCOIN:} \qquad \begin{array}{c|c} \text{input} & \text{output} \\ \hline |0\rangle & |0\rangle \\ |1\rangle & \begin{cases} |0\rangle & \text{with prob. } \frac{1}{2} \\ |1\rangle & \text{with prob. } \frac{1}{2} \end{cases} \end{array}$$

Now let's extend the 2 r-bit circuit we had previously been analyzing, as follows:



Equivalently, we are adding the instruction "4. CCOIN(x_1, x_2)" to our program. Now prior to the CCOIN gate, the joint state of the system is

$$\frac{1}{2}\left|00\right\rangle + \frac{1}{2}\left|11\right\rangle . \tag{2}$$

What is the state after the new CCOIN gate? Here is how you would say it in words:

> Prior to the CCOIN gate, (2) tells us that there is a $\frac{1}{2}$ probability that x_1 is $|0\rangle$ and x_2 is $|0\rangle$. In this case, the CCOIN does not touch x_1, so it stays $|0\rangle$, and the CCOIN gate leaves x_2 as $|0\rangle$ as per its definition. Thus the final state in this case

is still $|00\rangle$. On the other hand, (2) tells us that there is a $\frac{1}{2}$ probability that x_1 is $|1\rangle$ and x_2 is $|1\rangle$. In this case, the CCOIN does not touch x_1, so it stays $|1\rangle$, and the CCOIN gate changes x_2 to $|0\rangle$ with probability $\frac{1}{2}$ and to $|1\rangle$ with probability $\frac{1}{2}$, as per its definition. Thus overall the final state is $|00\rangle$ with probability $\frac{1}{2}$, is $|10\rangle$ with probability $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$, and is $|11\rangle$ with probability $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.

Here is the math symbolism you would write to exactly model those words:

$$\text{the final state is}\quad \frac{1}{2}|00\rangle + \frac{1}{2}\left(\frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle\right) = \frac{1}{2}|00\rangle + \frac{1}{4}|10\rangle + \frac{1}{4}|11\rangle .$$

As you can see, the natural "arithmetic" you would write with this formalism matches up with the actual probabilistic calculations.

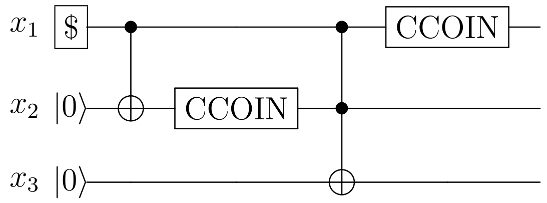
Let's add a few more twists. Suppose that x_3 is a new register that was in the system all along (we forgot to tell you about it), initialized to $|0\rangle$ and never touched. Then we would say that the final state of the system is

$$\frac{1}{2}|000\rangle + \frac{1}{4}|100\rangle + \frac{1}{4}|110\rangle .$$

Suppose we now added a CCNOT(x_1, x_2, x_3) gate to the end of the circuit. What would the new state be? Clearly we just go through the above state and change each of the bit-strings according to CCNOT's operation, leaving the probabilities unchanged:

$$\frac{1}{2}|000\rangle + \frac{1}{4}|100\rangle + \frac{1}{4}|111\rangle . \tag{3}$$

Finally, suppose we now added a CCOIN(x_1) instruction, so that the final circuit looked like this:



Now we can calculate the final state as follows. We start with state (3) and proceed through the "terms" (probabilistic cases) in it. For each one, the last two r-bits in the string will be unchanged, since the final CCOIN gate only operates on x_1. If the first r-bit is $|0\rangle$ then it will stay $|0\rangle$, as per CCOIN's definition. On the other hand, if the first r-bit is $|1\rangle$ then it will become $|0\rangle$ with probability $\frac{1}{2}$ and become $|1\rangle$ with probability $\frac{1}{2}$ (generating two "terms"). Then we simplify. The calculation is:

$$\frac{1}{2}|000\rangle + \frac{1}{4}\left(\frac{1}{2}|000\rangle + \frac{1}{2}|100\rangle\right) + \frac{1}{4}\left(\frac{1}{2}|011\rangle + \frac{1}{2}|111\rangle\right) \tag{4}$$
$$= \frac{5}{8}|000\rangle + \frac{1}{8}|100\rangle + \frac{1}{8}|011\rangle + \frac{1}{8}|111\rangle . \tag{5}$$

And indeed, had you been asked to compute the final joint state of the 3 r-bits in the above circuit, however you analyzed it would ultimately be pretty close to our pedantic style, and you would have indeed computed that there's a $\frac{5}{8}$ chance of ending with $|000\rangle$, a 0 chance of ending with $|001\rangle$, a $\frac{1}{8}$ chance of ending with $|100\rangle$, etc.

Remark 4.3. An obvious yet important takeaway from this kind of analysis is the following: Suppose we have a circuit with n r-bit registers. At any time, the state of the circuit can be written as

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

where the "coefficient" probabilities p_x are *nonnegative* and *summing to* 1.

4.1 On measurement

As a small note, we typically imagine that we provide the inputs to a randomized circuit, and then we observe (or *measure*) the outputs. The probabilistic "state" of the registers at some intermediate time in the circuit's execution reflects only the uncertainty that we, the observers, have about the registers' values. Of course, in reality the registers always have some definite value; it's merely that these variables are "hidden" to us. Analytically, once we observe one or more of the r-bits, the probabilistic state "collapses" to reflect the information we learned.

For example, in the randomized circuit we analyzed in the previous section, the final state (5) is

$$\frac{5}{8}|000\rangle + \frac{1}{8}|100\rangle + \frac{1}{8}|011\rangle + \frac{1}{8}|111\rangle .$$

Suppose for example we measure just the first register, x_1. The probability we observe a $|0\rangle$ is

$$\frac{5}{8} + \frac{1}{8} = \frac{6}{8} = \frac{3}{4}.$$

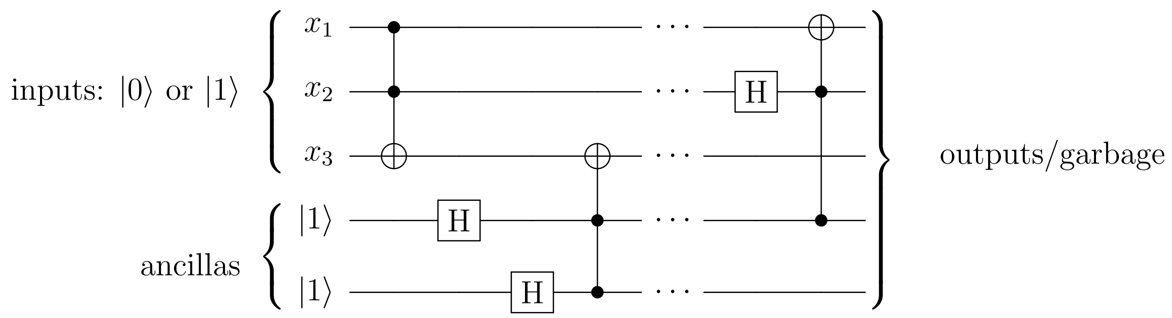
Supposing we do observe a $|0\rangle$, if we wanted to continue the analysis we would use the law of conditional probability to deduce that the state of the system "collapses" to

$$\frac{5/8}{3/4}|000\rangle + \frac{1/8}{3/4}|011\rangle = \frac{5}{6}|000\rangle + \frac{1}{6}|011\rangle .$$

Here, since we observed that the first bit was $|0\rangle$, only the strings consistent with that outcome survive, and the remaining probabilities are renormalized.

5 Quantum computation

Finally we can introduce the (barest essentials) of the quantum circuit model of computation. As mentioned, it is kind of like what you would get if you took randomized computation but found a way to allow the "probabilities" to be negative. It can also arguably be described as classical reversible computation augmented with the *Hadamard gate* $-\boxed{\text{H}}-$. In other words, a typical quantum circuit with 5 **qubit** (*quantum bit*) registers might look like this:



As usual, it is sufficient to just use CCNOT gates in addition to Hadamard gates, but for convenience we also allow NOT and CNOT gates too.

Just as in randomized computation, to analyze such a circuit we need to keep track of the joint state of all 5 qubits as time passes; i.e., as they proceed through the gates. Even though each gate only affects a small subset of all qubits, nevertheless just as in randomized computation we must track the state of all qubits at all times in order to keep track of the "correlations", which are called **entanglement** in the context of qubits.

Further, just as in randomized computation, at each time step the state of the 5 qubits is given by an expression that looks like this:

$$\alpha\,|00000\rangle + \beta\,|00001\rangle + \gamma\,|00010\rangle + \cdots + \omega\,|11111\rangle\,, \tag{6}$$

where the 32 (in general, 2^n) coefficients are **possibly negative** real numbers.[2] Since these numbers may be negative, we don't call them probabilities any more; we call them **amplitudes** (and a state like (6) is called a **superposition** of $|00000\rangle, \ldots, |11111\rangle$). Finally, just as in Remark 4.3, there is a restriction on what amplitudes are possible; this restriction is that the sum of their squares[3] is always 1:

$$\alpha^2 + \beta^2 + \cdots + \omega^2 = 1.$$

Note that this restriction is indeed always satisfied at the input. For example, if the actual input to the above quantum circuit is $|101\rangle$, then the initial state (when ancillas are included) is $|10111\rangle$. Here the amplitude on $|10111\rangle$ is 1, the amplitude on the other 31 strings is 0, and indeed $0^2 + \cdots + 0^2 + 1^2 + 0^2 + \cdots + 0^2 = 1$.

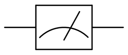
Here are all the rules of how to analyze quantum circuits:

[2]Actually, they can even be *complex numbers*. However if we stick to quantum circuits containing only Hadamard gates and reversible classical gates, then they will always just be real numbers. Further, it is known that these two gates are *universal* for quantum computation, and hence real numbers are universal for quantum computation. I.e., strictly speaking you don't need to worry about complex numbers if you don't want to; though ultimately, it will be more convenient (and physically accurate) to allow them.

[3]Squared magnitudes, if they're complex numbers

- CCNOT gates (and other classical reversible gates like NOT and CNOT) are analyzed exactly as in randomized circuits.
- Hadamard gates $-\boxed{\text{H}}-$ have the following input/output behavior: $|0\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|1\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$.
- Follow all the same arithmetic rules as for analyzing "r-bit (randomized) circuits", except say the word "amplitude" whenever you were going to say "probability".
- At the end, when you *measure* the output bits, the rule is probabilistic: if the final state is as in (6), then you "see"...

$$\begin{aligned} &|00000\rangle \text{ with probability } \alpha^2, \\ &|00001\rangle \text{ with probability } \beta^2, \\ &\qquad\vdots \\ &|11111\rangle \text{ with probability } \omega^2. \end{aligned}$$

  For now we will assume that you always measure all the qubits at the end of a circuit's computation, and nowhere else. The picture for a measurement is 

That's it! We could now, in theory, describe an enormous quantum circuit that carries out Shor's algorithm: it takes as input an n-bit integer, uses roughly n^2 CCNOT and H gates, and has the property that when you measure the final output qubits, they give (with probability at least 99%) the binary encoding of the prime factorization of the input integer (plus some garbage bits). Of course, that would be like if someone described a circuit for computing the maximum flow in a graph immediately after explaining what a NAND gate is. But it's possible.

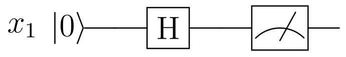
Remark 5.1. As mentioned in a footnote, it is a theorem that CCNOT and Hadamard gates together are universal for quantum computation. But they are not the only gates that are allowed by the physical laws of quantum computation. In the next lecture we will see just what gates nature *does* allow. For now, though, we will mention that any classical reversible gate is okay, and it's convenient to allow NOT and CNOT.

In fact, it is not complete obvious that the reversible gates and the Hadamard gate preserve the key property of quantum states — that the sum of the squares of the (magnitudes of the) amplitudes is 1. You will check this on the homework. In fact, the set of gates that quantum mechanics allows is *exactly* the set of linear transformations of amplitudes which preserve this property.

Remark 5.2. Quantum circuits are at least as powerful/efficient as classical circuits, since we can just not use Hadamard gates, and we're precisely left with reversible classical computation.

Remark 5.3. Quantum circuits are also at least as powerful/efficient as randomized circuits. This will be mostly clear from the following example, which shows how quantum circuits can generate COIN gates. (See the homework for a few clarifying details.)

Let us give an extremely basic example of a quantum circuit:

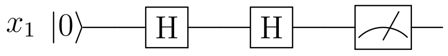


Here the single qubit x_1 is initialized to $|0\rangle$. By definition, after the Hadamard gate, the state of the register is

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle . \tag{7}$$

Finally, when the register is measured at the end, the measurement rule tells us that we observe $|0\rangle$ with probability $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$ and we observe $|1\rangle$ with probability $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$. Thus the outcome is just like a fair coin flip. (This would also have been the case had x_1 been initialized to $|1\rangle$; in that case, we would still see the output $|1\rangle$ with probability $\left(-\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$.)

However an interesting thing happens if, instead of measuring the register after the Hadamard gate is applied, we first apply another Hadamard gate and *then* measure:



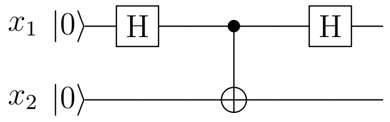
Let us do the analysis here. Just prior to the second Hadamard gate, the state of the register is as in (7). What is the new state after the second Hadamard gate? In words, we say almost exactly the same thing we would say in the case of a probabilistic analysis:

> With ~~probability~~ amplitude $\frac{1}{\sqrt{2}}$, the input is $|0\rangle$, in which case by definition the Hadamard gate outputs $|0\rangle$ with ~~probability~~ amplitude $\frac{1}{\sqrt{2}}$ and outputs $|1\rangle$ with amplitude $\frac{1}{\sqrt{2}}$. On the other hand, with amplitude $\frac{1}{\sqrt{2}}$, the input is $|1\rangle$, in which case by definition the Hadamard gate outputs $|0\rangle$ with amplitude $\frac{1}{\sqrt{2}}$ and outputs $|1\rangle$ with amplitude $-\frac{1}{\sqrt{2}}$. Thus the final state is...

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \left(\frac{1}{2}+\frac{1}{2}\right)|0\rangle + \left(\frac{1}{2}-\frac{1}{2}\right)|1\rangle = |0\rangle \ !$$

Now when we measure the register, we will *always* see $|0\rangle$. It's *kind of* like we "unflipped the coin". The positive/negative cancelation achieved here is precisely both the power and the mystery of quantum computation.

Let us do one more complicated example for practice. We will analyze this circuit:



The initial state is $|00\rangle$.

After the first Hadamard gate, the state is $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle$ (the second qubit is always unchanged).

After the CNOT gate, the state is

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle . \tag{8}$$

(As you can see, when applying classical gates, you really just need to change bit-strings according to the gate's definition; there's no need to do any arithmetic.) This state (8) is a famous entangled state; it is called an *EPR pair* after Einstein, Podolsky, and Rosen.[4] *If* we were to measure the two qubits at this point, we would see $|00\rangle$ with probability $\frac{1}{2}$ and $|11\rangle$ with probability $\frac{1}{2}$.

Finally, after the final Hadamard gate applied to state (8), we get

$$\begin{aligned} &\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle\right) \\ &= \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle . \end{aligned}$$

That's it. If we were to measure now, the rule tells us we would observe each of the four outcomes $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ with probability $\frac{1}{4}$ each.

5.1 On measurement

It is important to stress the following distinction with randomized computation. As mentioned, in the middle of a randomized circuit's computation, the probabilistic state of the registers only represents the uncertainty we (the analyzers) have about the bits' values. However, it's not like the bits have this state in Nature. Rather, the bits are *actually* in some deterministic state; we just don't know what it is.

This is **not** the case in quantum computation. According to the laws of physics, in the middle of a quantum circuit's computation, the superposition state that the n qubits are in is literally the true state they're in in Nature. They are not secretly in one of the basic states; Nature is literally keeping track of the 2^n amplitudes. (This gives you a hint of the potential computational power of quantum mechanics!) In a later lecture we will describe the theory that *proves* that this is the case, as well as the experimental work (including outstanding work from earlier this month) that backs up the theory.

References

[Alb83] David Albert. On quantum-mechanical automata. *Physics Letters A*, 98(5):249–252, 1983.

[Ben73] Charles Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

[4]These three people wrote a paper together doubting that the laws of quantum mechanic could be true, based on the nature of this state. However, EP&R were proven wrong.

[Ben80] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22(5):563–591, 1980.

[Deu85] David Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 400, pages 97–117, 1985.

[Deu89] David Deutsch. Quantum computational networks. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 425, pages 73–90, 1989.

[Fey82] Richard Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982.

[Fey86] Richard Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, 1986.

[Lan61] Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[Sho97] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997.

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

Lecture 2: Quantum Math Basics

September 14, 2015

Lecturer: John Wright *Scribe: Yongshan Ding*

1 Complex Numbers

From last lecture, we have seen some of the essentials of the quantum circuit model of computation, as well as their strong connections with classical randomized model of computation. Today, we will characterize the quantum model in a more formal way. Let's get started with the very basics, complex numbers.

Definition 1.1. A *complex number* $z \in \mathbb{C}$ is a number of the form $a + bi$, where $a, b \in \mathbb{R}$, and i is the imaginary unit, satisfying $i^2 = -1$.

It's always convenient to picture a complex number $z = a + bi$ as a point (a, b) in the two-dimensional *complex plane*, where the horizontal axis is the real part and the vertical axis is the imaginary part:

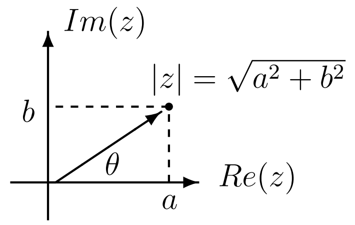


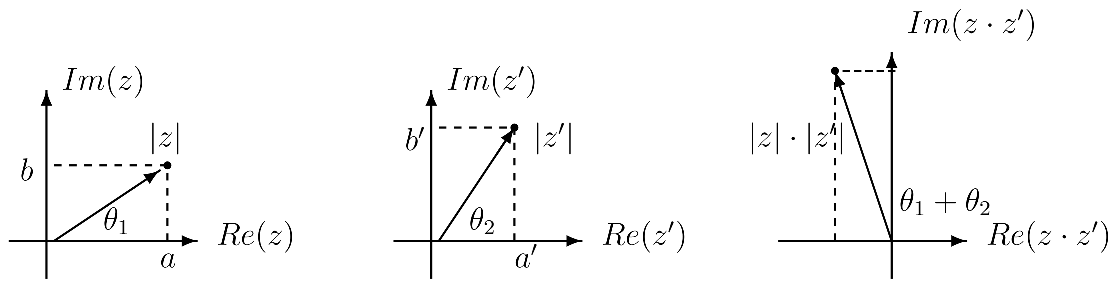
Figure 1: Geometric representation of complex number $z = a + bi$

Another common way of parametrizing a point in the complex plane, instead of using Cartesian coordinates a and b, is to use the *radial coordinate* r (the Euclidean distance of the point from the origin $|z| = \sqrt{a^2 + b^2}$), together with the *angular coordinate* θ (the angle from the real axis). In particular, it can help us visualize geometrically what the *multiplication* operation does:

Observation 1.2. *The product of two complex numbers z_1, z_2 has magnitude $|z_1| \cdot |z_2|$ and angle $\theta_1 + \theta_2$.*

Figure 2 is the geometric visualization of the multiplication of two complex numbers $z = a + bi$ and $z' = a' + b'i$. Another important operation on complex numbers is the complex conjugate:

Definition 1.3. The *complex conjugate* of a complex number $z = a + bi$ is defined as $\bar{z} = a - bi$, also denoted as z^* or $z^\dagger$.



(a) $z = a + bi$ (b) $z' = a' + b'i$ (c) $z \cdot z'$

Figure 2: Geometric representation of $z \cdot z'$

As a convention for complex numbers, we call the product of a complex number with its complex conjugate the *square* of that complex number, which is essentially the square of its magnitude:

$$z \cdot z^\dagger = (a + bi)(a - bi) = a^2 + b^2 = |z|^2$$

Notice that the result is always a real number, which becomes obvious when we realize that $z^\dagger$ is basically a reflection of z about the real axis. Since the sum of their angles in the complex plane is 0, $z \cdot z^\dagger$ always lands on the axis. We can therefore naturally generalize the inner product for complex vectors.

Definition 1.4. The *inner product (or dot product)* of two d-dimensional vectors is defined as $(z_1, \ldots, z_d) \cdot (w_1, \ldots, w_d) = z_1^\dagger w_1 + \cdots + z_d^\dagger w_d$.

The dot product of a vector with itself now becomes:

$$(z_1, \ldots, z_d) \cdot (z_1, \ldots, z_d) = |z_1|^2 + \cdots + |z_d|^2.$$

2 Quantum Bits

Just as a classical bit can have a state of either 0 or 1, the two most common states for a **qubit** *(quantum bit)* are the states $|0\rangle$ and $|1\rangle$. For now, let's just see the notation "$|\ \rangle$" as a way of distinguishing qubits from classical bits. The *actual* difference though is that a qubit can be in *linear combinations* of states, also know as *superpositions*. In other words, we can write a quantum state in a more general form:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where $\alpha, \beta \in \mathbb{C}$, and $|\alpha|^2 + |\beta|^2 = 1$. Two other famous states that we will see very often in this class are:

$$|+\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,,\ \ |-\rangle = \frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle\,.$$

We can also think of $|\psi\rangle$ as a vector in the two-dimensional complex plane spanned by the two basis states $|0\rangle$ and $|1\rangle$. As mentioned last time, often we can view α and β as *real numbers* without losing much. The reason we can sometimes ignore the fact that they are complex numbers is that $\mathbb{C}$ can be easily simulated by $\mathbb{R}^2$. That's in fact exactly what we did when we imagined the two-dimensional complex plane in the previous section. Then, why do we even use complex numbers at all? Well, there are two major reasons: firstly, complex phases are intrinsic to many quantum algorithms, like the Shor's Algorithm for prime factorization. Complex numbers can help us gain some intuitions on those algorithms. Secondly, complex numbers are often just simpler in terms of describing unknown quantum states, and carrying out computations.

We have been talking about qubits for a while, but how are they *implemented*? In fact many different physical systems can accomplish this. Although we won't cover the entire physics behind them, a general idea of how the qubits are realized physically can sometimes help us understand the procedures and algorithms we are dealing with. In particular, they might be represented by two states of an electron orbiting an atom; by two directions of the spin (*intrinsic angular momentum*) of a particle; by two polarizations of a photon. Let's take a spin-$\frac{1}{2}$ particle as an example. If we were to measure its spin along the z-axis, we would observe that it is either *up* (in $+z$ direction) or *down* (in $-z$ direction). In many physics papers, the two states are denoted as $|z+\rangle$ and $|z-\rangle$, or $|\uparrow\rangle$ and $|\downarrow\rangle$. For computational purposes, we can simply regard them as our good old friends $|0\rangle$ and $|1\rangle$.

2.1 Multiple Qubits and the Qudit System

Let's begin the discussion on multiple-qubit system from the simpliest: a *two-qubit system.* Just as classical 2-bit system, we have four possible computational basis states, namely $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. We can thus write a general form of the two-qubit state as follows:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

where the amplitudes satisfy $|\alpha_{00}|^2 + \cdots + |\alpha_{00}|^2 = 1$. For instance, we can have a *uniformly mixed state* where the scalar coefficients are the same: $\alpha_{00} = \alpha_{01} = \alpha_{10} = \alpha_{11} = \frac{1}{2}$. Or more interestingly, we can have the following state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle.$$

Note that this is a system where the two qubit are *correlated*! In particular, they seem to be always in the same state. We will come back to this interesting state very often in the future.

Now, it's time to the extend this to a more general case: *the qudit system.* Specifically, a state in the **d**-dimensional qu**d**it system is a superposition of d basis states. We can write:

$$|\psi\rangle = \alpha_1|1\rangle + \alpha_2|2\rangle + \cdots + \alpha_d|d\rangle,$$

where $|\alpha_1|^2 + \cdots + |\alpha_d|^2 = 1$ as always.

How is a qudit system implemented in physical reality? In fact, particles are allowed to have *spin quantum number* of $0, \frac{1}{2}, 1, \frac{3}{2}, 2,$ etc. For example, a spin-$\frac{1}{2}$ particle, like an electron, is a natural "*qubit*" system, whereas a spin-1 particle, like a photon or a gluon, is a "*qutrit*" system. Although no fundamental particle has been experimentally found to have spin quantum number higher than 1, the two-qubit system we mentioned earlier behaves exactly the same as a qudit system where $d = 4$. In theory, we could construct any qudit system using only qubits.

2.2 Qubits - the Mathematics

As we saw earlier, a quantum state in the qubit system can be represented as a unit (*column*) vector in the $\mathbb{C}^2$ plane, spanned by the following two basis state:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}.$$

With a little bit of algebra, we can write a general state $|\psi\rangle$ as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix}\alpha\\0\end{bmatrix} + \begin{bmatrix}0\\\beta\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix},$$

where $|\alpha|^2 + |\beta|^2 = 1$. Before we look into the properties of quantum states, let's first introduce the style of notations we use: *the Bra-ket notation*, also called *the Dirac notation*.

Notation 2.1. *A quantum state $|\psi\rangle$ is a (column) vector, also known as a* **ket**, *whereas a state $\langle\psi|$ is the (row) vector dual to $|\psi\rangle$, also know as a* **bra**.

To get a bra vector from a ket vector, we need to take the *conjugate transpose*. Thus we can write:

$$\langle\psi| = (|\psi\rangle)^\dagger = \begin{bmatrix}\alpha\\\beta\end{bmatrix}^\dagger = \begin{bmatrix}\alpha^\dagger & \beta^\dagger\end{bmatrix}.$$

Now, suppose we have two quantum states: $|x_1\rangle = \alpha_0\,|0\rangle + \alpha_1\,|1\rangle$ and $|x_2\rangle = \beta_0\,|0\rangle + \beta_1\,|1\rangle$. One possible operation we can do is of course multiplication.

Definition 2.2. The *inner product (or dot product)* of two quantum states $|x_1\rangle$ and $|x_2\rangle$ is defined as $\langle x_1| \cdot |x_2\rangle$, which can be further simplified as $\langle x_1|x_2\rangle$.

For example, the inner product of $|x_1\rangle$ and $|x_2\rangle$ can be carried out:

$$\langle x_1|x_2\rangle = \begin{bmatrix}\alpha_0^\dagger & \alpha_1^\dagger\end{bmatrix} \cdot \begin{bmatrix}\beta_0\\\beta_1\end{bmatrix} = \alpha_0^\dagger\beta_0 + \alpha_1^\dagger\beta_1.$$

What if we take the inner product of $|x_1\rangle$ with itself? Actually, it turns out to be the same as the sum of squares of the amplitudes, which is always 1.

$$\langle x_1|x_1\rangle = \alpha_0^\dagger\alpha_0 + \alpha_1^\dagger\alpha_1 = |\alpha_0|^2 + |\alpha_1|^2 = 1$$

Definition 2.3. The *outer product* of two quantum states $|x_1\rangle$ and $|x_2\rangle$ is defined as $|x_1\rangle \cdot \langle x_2|$, which is often written as $|x_1\rangle\,\langle x_2|$.

This time the result would be in fact a *matrix*, instead of a complex number. Take the outer product of $|x_2\rangle$ and $|x_1\rangle$:

$$|x_2\rangle\langle x_1| = \begin{bmatrix}\beta_0\\ \beta_1\end{bmatrix}\cdot\begin{bmatrix}\alpha_0^\dagger & \alpha_1^\dagger\end{bmatrix} = \begin{bmatrix}\alpha_0^\dagger\beta_0 & \alpha_1^\dagger\beta_0\\ \alpha_0^\dagger\beta_1 & \alpha_1^\dagger\beta_1\end{bmatrix}$$

Observation 2.4. *The relationship between the outer and the inner product:* $tr(|x_2\rangle\langle x_1|) = \langle x_1|x_2\rangle$.

Now let's take a look at some concrete examples:

$$\langle 0|1\rangle = \begin{bmatrix}1 & 0\end{bmatrix}\cdot\begin{bmatrix}0\\ 1\end{bmatrix} = 0,$$

$$\langle +|-\rangle = \begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}\cdot\begin{bmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{bmatrix} = \frac{1}{2}-\frac{1}{2} = 0.$$

It means $|0\rangle$ and $|1\rangle$ are orthogonal to each other, so are $|+\rangle$ and $|-\rangle$. Therefore, we say that they both form an *orthonormal basis* for $\mathbb{C}^2$. For any quantum state $|\psi\rangle$ in $\mathbb{C}^2$, it can be expressed using either basis:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \alpha_+|+\rangle + \alpha_-|-\rangle,$$

where again $|\alpha_0|^2 + |\alpha_1|^2 = |\alpha_+|^2 + |\alpha_-|^2 = 1$.

Let's move on to general qudits, as we always do. A qudit state is a unit vector in $\mathbb{C}^d$:

$$|\psi\rangle = \begin{bmatrix}\alpha_1\\ \alpha_2\\ \vdots\\ \alpha_d\end{bmatrix},$$

such that $\langle\psi|\psi\rangle = 1$ as always. Similarly, we have the d-dimensional bases $\{i\}, i \in [d]$:

$$|1\rangle = \begin{bmatrix}1\\ 0\\ \vdots\\ 0\end{bmatrix}, \ldots, |d\rangle = \begin{bmatrix}0\\ 0\\ \vdots\\ 1\end{bmatrix},$$

2.3 Multiple Qubit System - the Mathematics

Suppose we have two qubits $|x\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ and $|y\rangle = \beta_0|0\rangle + \beta_1|1\rangle$. How can we describe their *joint state*? The first guess might be using multiplication of some sort. So, let's try it, maybe using the notation $\otimes$:

$$\begin{aligned}|x\rangle\otimes|y\rangle &= (\alpha_0|0\rangle + \alpha_1|1\rangle)\otimes(\beta_0|0\rangle + \beta_1|1\rangle)\\ &= \alpha_0\beta_0\underbrace{|0\rangle\otimes|0\rangle}_{|00\rangle} + \alpha_0\beta_1\underbrace{|0\rangle\otimes|1\rangle}_{|01\rangle} + \alpha_1\beta_0\underbrace{|1\rangle\otimes|0\rangle}_{|10\rangle} + \alpha_1\beta_1\underbrace{|1\rangle\otimes|1\rangle}_{|11\rangle}\end{aligned}$$

It seems that if we regard $|0\rangle \otimes |0\rangle = |00\rangle$, etc., as shown above, $|x\rangle \otimes |y\rangle$ looks exactly the same as a linear combination of the four basis states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. However, we do need to check whether the result of $|x\rangle \otimes |y\rangle$ is *actually* a quantum state:

$$\begin{aligned} &|\alpha_0\beta_0|^2 + |\alpha_0\beta_0|^2 + |\alpha_0\beta_0|^2 + |\alpha_0\beta_0|^2 \\ =&(|\alpha_0|^2 + |\alpha_1|^2) \cdot (|\beta_0|^2 + |\beta_1|^2) = 1 \end{aligned}$$

So it is indeed a sensible way of describing joint states!

In general, given kets $|a\rangle = \sum_j \alpha_j |a_j\rangle$ and $|b\rangle = \sum_k \beta_k |b_k\rangle$, we have:

Definition 2.5. The *tensor product* of kets $|a\rangle$ and $|b\rangle$ is

$$|a\rangle \otimes |b\rangle = \sum_j \sum_k \alpha_j \beta_k (|a_j\rangle \otimes |b_k\rangle),$$

where $|a_j\rangle \otimes |b_k\rangle$ can also be written as $|a_j b_k\rangle$ or $|jk\rangle$.

Notice that tensor product is *not commutative*: $|0\rangle \otimes |1\rangle = |01\rangle \neq |10\rangle = |1\rangle \otimes |0\rangle = |00\rangle$. To see what tensor product does more clearly, let's go back to our kets $|x\rangle$ and $|y\rangle$. We can write out the matrix form:

$$|x\rangle \otimes |y\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \otimes \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0 \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \\ \alpha_1 \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{bmatrix}$$

For example, we have the four basis of two-qubit system:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

We have to remember that not all states in the multiple-qubit system are of tensor product form, namely the *product states.* The most famous example would be:

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle.$$

The states that cannot be expressed in terms of a tensor product of two other states are called the *entangled states.* Later in the course, we will see many different interesting properties of the product states and the entangled states.

More generally, in the d dimensional qudit system, we have:

$$\begin{bmatrix}\alpha_0\\ \vdots\\ \alpha_d\end{bmatrix} \otimes \begin{bmatrix}\beta_0\\ \vdots\\ \beta_d\end{bmatrix} = \begin{bmatrix}\alpha_0 \begin{bmatrix}\beta_0\\ \vdots\\ \beta_1\end{bmatrix}\\ \vdots \\ \alpha_1 \begin{bmatrix}\beta_0\\ \vdots\\ \beta_1\end{bmatrix}\end{bmatrix} = \text{a long vector of length } d^2.$$

3 Quantum Computation

How quantum states are allowed to change over time? We have seen some basic quantum circuits in last lecture, but today we will define them in a more formal way. We will start with our favorite gate, the Hadamard gate H. Recall that H maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$. Graphically, it behaves like the following:

$$|0\rangle \;-\boxed{H}-\; \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$
$$|1\rangle \;-\boxed{H}-\; \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

More generally, for any gate U, the circuit will look like this:

$$|\psi\rangle \;-\boxed{U}-\; U|\psi\rangle$$

But what restrictions do we have on gate U? In fact, in order for the circuit to be physically realizable, we have *two restrictions*:

1. U: $\mathbb{C}^d \to \mathbb{C}^d$ has to map from a *quantum state* to another *quantum state.*
2. U has to be *linear* (i.e. $U(|x_1\rangle + |x_2\rangle) = U|x_1\rangle + U|x_2\rangle$). In other words, U is a matrix.

To satisfy restriction 1, we notice that both the input and the output should be quantum states, and thus

$$\langle\psi|\psi\rangle = 1, \;\; (U|\psi\rangle)^\dagger U|\psi\rangle = 1.$$

Combined with restriction 2, we know that for all $\langle\psi|\psi\rangle = 1$, we need,

$$\begin{aligned} &(U|\psi\rangle)^\dagger U|\psi\rangle = 1\\ \Rightarrow &|\psi\rangle^\dagger U^\dagger U|\psi\rangle = 1\\ \Rightarrow &\langle\psi| U^\dagger U|\psi\rangle = 1\\ \Rightarrow &U^\dagger U = I\end{aligned}$$

In other words, $U \in \mathbb{C}^{d\times d}$ is **unitary**!

Observation 3.1. *The angle between two quantum states preserves under any unitary operations.*

Imagine we take two quantum states $|x_1\rangle$ and $|x_2\rangle$. We send them both through a unitary gate U, resulting in two states $U|x_1\rangle$ and $U|x_2\rangle$. The angle between them can thus be expressed in terms of their inner product:

$$(U|x_1\rangle)^\dagger U|x_2\rangle = \langle x_1| U^\dagger U |x_2\rangle = \langle x_1|x_2\rangle .$$

Observation 3.2. *Unitary operations are invertible (reversible) and its inverse is its conjugate transpose.*

Notice that given $U^\dagger U = I$, we have $I = I^\dagger = (U^\dagger U)^\dagger = UU^\dagger$. So it follows that $U^{-1} = U^\dagger$. Therefore, applying U and $U^\dagger$ back to back brings the state to its original input state back again:

$$|\psi\rangle \;-\!\boxed{U}\!-\!\boxed{U^\dagger}\!-\; |\psi\rangle$$

Observation 3.3. *Unitary operations are equivalent to changes of basis.*

Suppose $\{|v_1\rangle, \ldots, |v_d\rangle\}$ is any *orthonormal* basis of $\mathbb{C}^d$, and define the following states:

$$|w_1\rangle = U|v_1\rangle, \ldots, |w_d\rangle = U|v_d\rangle .$$

Since we have shown that unitary operations are rotations that preserve relative angles, the set $\{|w_i\rangle\}$ are also *orthonormal*:

$$\langle w_i|w_j\rangle = \langle v_i|v_j\rangle = \begin{cases} 1 & : i = j \\ 0 & : i \neq j \end{cases}$$

And again this is verified by our favorite example, the *Hadamard* gate, which has the matrix form:

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

We can write:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle ,$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle .$$

We can actually construct the Hadamard matrix H from outer products:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = (|+\rangle\langle 0|) + (|-\rangle\langle 1|).$$

In general, we can express unitary operations in $\mathbb{C}^d$ as follows:

$$U = \sum_{i=1}^{d} |w_i\rangle \langle v_i| \, ,$$

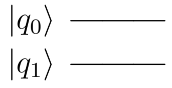
where $\{w_i\}, \{v_i\}$ are the bases of $\mathbb{C}^d$. In fact, the action of U is a change of basis from $\{v_i\}$ to $\{w_i\}$:

$$U |v_j\rangle = \sum_{i=1}^{d} |w_i\rangle \underbrace{\langle v_i| |v_j\rangle}_{\delta_{ij}} = |w_j\rangle \, ,$$

where $\delta_{ij} = \left\{ \begin{array}{ll} 1 & : i = j \\ 0 & : i \neq j \end{array} \right.$. The same rule applies to general states that are in superpositions of basis vectors. We will employ linearity for the states like $|\psi\rangle = \sum_i \alpha_i |v_i\rangle$.

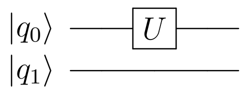
3.1 Mutiple Qubits System

What if we apply unitary operations on multiple qubit systems? What does it mean to apply gates on one of the entangled qubits? Let's again start with the simplest, a two-qubit system. And here is a "*do-nothing*" gate:

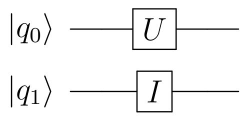


Notice that $|q_0\rangle$ and $|q_1\rangle$ individually would be described as 2×1 column vectors, but when viewing them collectively we have to maintain the joint state of the entire system, which we write as a 4×1 column vector. This is necessary when, e.g., $|q_0\rangle$ and $|q_1\rangle$ are entangled. Now, this "quantum circuit" does essentially nothing to the states, so it's not particularly interesting, but if we wanted to describe how this circuit behaves, what matrix would we use to describe it? The answer is the 4×4 identity matrix I.

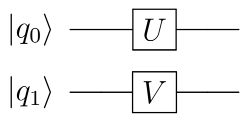
The more interesting case is of course when a unitary is applied to (at least) one of the qubits. For example:



As we repeatedly stressed in Lecture 1, just as in probabilistic computation you have to always maintain the state of all qubits at all times, even though it looks like U is only changing the first qubit. We know that coming in, the state is represented by a height-4 column vector. But U is represented by a 2×2 matrix. And that doesn't type-check (can't multiply 2×2 against 4×1). Well, by the laws of quantum mechanics, the transformation from input to output has to be a big 4×4 (*unitary*) matrix. To clarify things, one might ask oneself – what operation are we applying to the second bit? Well, we're doing nothing, so we're applying an (2×2) I. So we could draw the picture like



So how do U and I get combined into a 4×4 matrix? Or more generally, if we do



how do they get combined?

Well, the short answer is that it's something called $U \otimes V$ (*the Kronecker/tensor product* on matrices), but you can even *tell* what $U \otimes V$ should be. Notice that everything can be spatially separated; perhaps the first qubit coming in is a $|0\rangle$ or a $|1\rangle$ and the second qubit coming in is a $|0\rangle$ or a $|1\rangle$, and they are not in any way entangled, just hanging out in the same room. Clearly after the circuit's over you have $U|q_0\rangle$ and $V|q_1\rangle$ hanging out, not entangled; i.e., we determined that for $|q_0\rangle, |q_1\rangle$ in $\{|0\rangle, |1\rangle\}$, whatever $U \otimes V$ is it must satisfy

$$(U \otimes V)(|q_0\rangle \otimes |q_1\rangle) = (U|q_0\rangle) \otimes (V|q_1\rangle).$$

By linearity, that's enough to exactly tell what $U \otimes V$ must be. This is because for any (possibly entangled) 2-qubit state

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

we have that

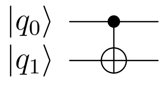
$$(U \otimes V)|\psi\rangle = \alpha_{00}(U \otimes V)|00\rangle + \alpha_{01}(U \otimes V)|01\rangle + \alpha_{10}(U \otimes V)|10\rangle + \alpha_{11}(U \otimes V)|11\rangle,$$

and we have already defined $U \otimes V$ for these four kets.

Of course, this just shows how $U \otimes V$ operates on kets, but we know that underlying $U \otimes V$ must be a *unitary* matrix. From the above, it's easy to derive what this unitary matrix looks like. It's given as follows (notice the similarity to the tensor product of two vectors):

$$U \otimes V = \begin{bmatrix} u_{11}V & u_{12}V \\ u_{21}V & u_{22}V \end{bmatrix} = \begin{bmatrix} u_{11}v_{11} & u_{11}v_{12} & u_{12}v_{11} & u_{12}v_{12} \\ u_{11}v_{11} & u_{11}v_{12} & u_{12}v_{21} & u_{12}v_{22} \\ u_{21}v_{11} & u_{21}v_{12} & u_{22}v_{11} & u_{22}v_{12} \\ u_{21}v_{21} & u_{21}v_{22} & u_{22}v_{21} & u_{22}v_{22} \end{bmatrix}.$$

Of course, not all 2-qubit gates are the result of tensoring two 1-qubit gates together. Perhaps the easiest example of this is the CNOT gate.

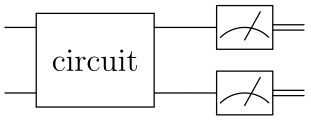


In matrix form, we can write this gate as

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

4 Measurements

We have discussed quantum measurements on a single qubit in last lecture. Briefly, suppose we have a state $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$. Measurements on $|\psi\rangle$ result in 0 or 1, with probability $|\alpha_0|^2$ and $|\alpha_1|^2$, respectively. What happens if we measure a state in a multiple qubit system? Take the 2-qubit system as an example:



Suppose before the measurements were made, we have a joint state in the form of

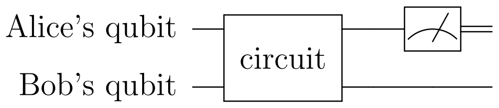
$$|\psi\rangle = \alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \in \mathbb{C}^4,$$

such that $|\alpha_{00}|^2 + \cdots + |\alpha_{11}|^2 = 1$. Then we have the following *measurement rule*:

$$\text{We will observe} \begin{cases} |00\rangle & : \text{ with probability } |\alpha_{00}|^2 \\ |01\rangle & : \text{ with probability } |\alpha_{01}|^2 \\ |10\rangle & : \text{ with probability } |\alpha_{10}|^2 \\ |11\rangle & : \text{ with probability } |\alpha_{11}|^2 \end{cases}.$$

4.1 Partial Measurements

What if we only measure on one of the two qubits? If we measure the second qubit afterwards, how will the outcome of the second measurement be related to the outcome of the first measurement? The circuit for *partial measurements* might look like this:



Say Alice holds on to the first qubit and Bob holds on to the second. If Alice measures the first qubit, her qubit becomes deterministic. We can rewrite the measurement rule as follows:

$$\text{Alice will observe} \begin{cases} |0\rangle & : \text{ with probability } |\alpha_{00}|^2 + |\alpha_{01}|^2 \\ |1\rangle & : \text{ with probability } |\alpha_{10}|^2 + |\alpha_{11}|^2 \end{cases}.$$

Suppose Alice observed $|0\rangle$. The probability rule for Bob's measurements on the second qubit is now the same as the rules of *conditional probability.* In particular, the joint state after Alice measured her qubit becomes:

$$\frac{\alpha_{00} |00\rangle + \alpha_{01} |01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} = |0\rangle \otimes (\frac{\alpha_{00} |0\rangle + \alpha_{01} |01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}).$$

Notice that, the joint state is now "*unentangled*" to a product state. Now, Bob measures. We can then write his measurement rule as:

$$\text{Bob will observe} \begin{cases} |0\rangle & : \text{ with probability } \frac{|\alpha_{00}|^2}{|\alpha_{00}|^2+|\alpha_{01}|^2} \\ |1\rangle & : \text{ with probability } \frac{|\alpha_{01}|^2}{|\alpha_{00}|^2+|\alpha_{01}|^2} \end{cases} .$$

The conditional probabilities is actually the straightforward consequences of the fact that, once Alice made her measurement (say, she observed $|0\rangle$), the probability of observing $|10\rangle, |11\rangle$ becomes zero.

The action of quantum measurements is in fact a lot more subtle and interesting than what we have briefly discussed above. Sometimes a quantum system is "*destroyed*" during a measurement, but in other cases it continues to exist and preserves certain properties. To analyze the effect of quantum measurements, we often employ the notion of "*wave function collapse*", which we will in detail next time.

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

Lecture 3: The Power of Entanglement

September 9, 2015

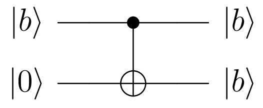
Lecturer: Ryan O'Donnell *Scribe: Joshua Brakensiek*

1 Copying bits

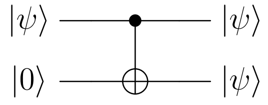
In lecture 2, we demonstrated that every quantum circuit can be thought of as a unitary matrix acting on the space of quantum states as well as described the rules for measurement and partial measurement. In this lecture, we build on these ideas to show how quantum entanglement gives quantum computers an advantage over classical computers in certain settings.

1.1 No-cloning theorem

To start, we describe a way in which quantum computation is *not* superior to classical computation. In classical computation, it is quite easy to copy information, even in a reversible manner.



This leads to a natural question: given a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, is it possible to copy $|\psi\rangle$ so that two *unentangled* qubits each have state $|\psi\rangle$? This would require the two qubits to be in a joint state of $|\psi\rangle \otimes |\psi\rangle$. As a first attempt, it seems that the classical circuit above meets our needs in the quantum setting, too.



To check, before the CNOT gate, the qubits are in the joint state of $(\alpha|0\rangle+\beta|1\rangle)\otimes(|0\rangle) = \alpha|00\rangle + \beta|10\rangle$. After the CNOT, gate the qubit are then in the joint state of $\alpha|00\rangle + \beta|11\rangle$. Unfortunately, this is not equal to our desired state of

$$|\psi\rangle \otimes |\psi\rangle = \alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle = \begin{bmatrix} \alpha^2 \\ \alpha\beta \\ \alpha\beta \\ \beta^2 \end{bmatrix}$$

unless $\alpha = 1$ and $\beta = 0$ or $\alpha = 0$ or $\beta = 1$. That is, the circuit described above fails to copy an arbitrary quantum state into two unentangled copies. It turns out that there does not exist *any* quantum circuit which can take in an arbitrary qubit $|\psi\rangle$ and produce the state $|\psi\rangle \otimes |\psi\rangle$, even when permitted to use ancillas in the input and garbage in the output. This result is call the *No-cloning theorem* and is due to Wootters and Zurek [WZ82] (see also [dW11]).

Theorem 1.1 (No-cloning theorem.). *For all $n \in \mathbb{N}$, there exists no quantum circuit C which upon input $|\psi\rangle \otimes |0^{n-1}\rangle$ outputs $|\psi\rangle \otimes |\psi\rangle \otimes f(|\psi\rangle)$, where $f(\psi)$ (the garbage) is a possibly entangled state of $n-2$ qubits.*

Remark 1.2. We may assume the ancillas are all $|0\rangle$ since we can use a NOT gate to obtain a $|1\rangle$. Additionally, we may assume that there are no measurements in C, since we can defer all measurements until the end of the computation, and we definitely would not want to measure the first two qubits of the output.

Proof. Assume for sake of contradiction that such a C exits. Let U be the unitary matrix representing C. We know that $U(|0^n\rangle) = |00\rangle \otimes f(|0\rangle)$ and $U(|1\rangle \otimes |0^{n-1}\rangle) = |11\rangle \otimes f(|1\rangle)$. Remember that $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. By the linearity of U,

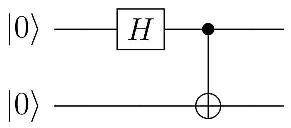
$$U\left(|+\rangle \otimes |0^{n-1}\rangle\right) = \frac{1}{\sqrt{2}}U|0^n\rangle + \frac{1}{\sqrt{2}}U(|1\rangle \otimes |0^{n-1}\rangle) = \frac{1}{\sqrt{2}}|00\rangle \otimes f(|0\rangle) + \frac{1}{\sqrt{2}}|11\rangle \otimes f(|1\rangle).$$

Thus, if we measure the first two qubits, we get the state $|00\rangle$ with probability 1/2 and $|11\rangle$ with probability 1/2. If U were copying $|+\rangle$ correctly, we should see each of $|00\rangle, |10\rangle, |01\rangle, |11\rangle$ with probability 1/4. Hence, U failed to copy $|+\rangle$, so no such unitary U or circuit C exists with the desired properties. □

Although this result may seem unsettling at first, there is an intuitive analogous result in the randomized model of computation: there is no circuit which can take as input a single p-biased random bit (say from a COIN$_p$ gate) and return as output two two independently distributed p-biased bits.

1.2 EPR pairs

Consider the following modification of our classical bit-copying circuit.



The output of this gate is the quantum state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. This entangled state is call an *EPR pair*, named after Einstein, Podolsky, and Rosen[EPR35]. Although Einstein did not believe that EPR pairs existed, but they have been confirmed to exist through experimentation (e.g. [AGR81]). EPR pairs will show up many times throughout the course, including a few times in this lecture.

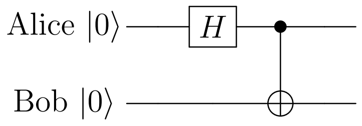
Imagine that we pass an EPR pair $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ through the quantum gate $\mathrm{H}\otimes\mathrm{H}$; that is, we apply a separate Hadamard gate to each qubit. Recall that $\mathrm{H}|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $\mathrm{H}|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Thus, we have that

$$\begin{aligned}
(\mathrm{H}\otimes\mathrm{H})\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) &= \frac{1}{\sqrt{2}}(|+\rangle\otimes|+\rangle) + \frac{1}{\sqrt{2}}(|-\rangle\otimes|-\rangle) \\
&= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\otimes\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \\
&\quad + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\otimes\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\
&= \frac{1}{2\sqrt{2}}|00\rangle + \frac{1}{2\sqrt{2}}|01\rangle + \frac{1}{2\sqrt{2}}|10\rangle + \frac{1}{2\sqrt{2}}|11\rangle \\
&\quad + \frac{1}{2\sqrt{2}}|00\rangle - \frac{1}{2\sqrt{2}}|01\rangle - \frac{1}{2\sqrt{2}}|10\rangle + \frac{1}{2\sqrt{2}}|11\rangle \\
&= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle,
\end{aligned}$$

which is our original EPR pair! In general, the bookkeeping of quantum states can be rather unintuitive.

2 Quantum teleportation

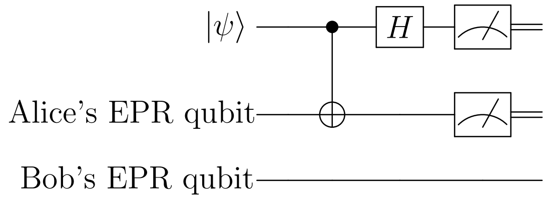
Imagine that two computer scientists, Alice and Bob, each have a qubit initialized to the classical state $|0\rangle$ and that they decide to entangle their qubits into an EPR pair. Thus, their joint state is $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, where the first qubit is Alice's and second is Bob's.



Now, even if Alice's and Bob's qubits are physically separated (say they take their qubits to their own homes), the two qubits will still be an EPR pair as long as neither performs a measurement. Additionally, say that Alice has a qubit $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ at her home, and she would like to *give* $|\psi\rangle$ to Bob. It turns out she can do this without leaving her home.

One plausible idea, is that she determines the values of α_0 and α_1 and tells these values to Bob over a *classical* channel (say over the telephone). There are two problems with this idea. First, Alice cannot learn what α_0 and α_1 are without performing a measurement, which would cause her to lose $|\psi\rangle$. Second, even if she did know what α_0 and α_1 were, since they lie in $\mathbb{C}^2$, she would need infinitely many bits of precision to accurately tell Bob what α_0 and α_1 were.

Instead, Alice can cleverly send ψ and her half of the EPR pair through the following quantum circuit.



At the beginning of the circuit, the three qubits are in the joint state of

$$|\psi\rangle \otimes \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) = \frac{\alpha_0}{\sqrt{2}}|011\rangle + \frac{\alpha_0}{\sqrt{2}}|000\rangle + \frac{\alpha_1}{\sqrt{2}}|100\rangle + \frac{\alpha_1}{\sqrt{2}}|111\rangle.$$

After passing the qubits through the quantum gates, but before the measurement, the state of the circuit ends up being

$$\frac{\alpha_0}{2}|000\rangle + \frac{\alpha_1}{2}|001\rangle + \frac{\alpha_1}{2}|010\rangle + \frac{\alpha_0}{2}|011\rangle + \frac{\alpha_0}{2}|100\rangle - \frac{\alpha_1}{2}|101\rangle - \frac{\alpha_1}{2}|110\rangle + \frac{\alpha_0}{2}|111\rangle.$$

After Alice measures her two qubits. What could the possible states be? These are summarized in the following table. Recall that $|\alpha_0|^2 + |\alpha_1|^2 = 1$

Alice's measurement	Prob. of meas.	Collapsed state
$\|00\rangle$	$\frac{\|\alpha_0\|^2}{4} + \frac{\|\alpha_1\|^2}{4} = \frac{1}{4}$	$\|00\rangle \otimes (\alpha_0\|0\rangle + \alpha_1\|1\rangle)$
$\|01\rangle$	$\frac{\|\alpha_1\|^2}{4} + \frac{\|\alpha_0\|^2}{4} = \frac{1}{4}$	$\|01\rangle \otimes (\alpha_1\|0\rangle + \alpha_0\|1\rangle)$
$\|10\rangle$	$\frac{\|\alpha_0\|^2}{4} + \frac{\|-\alpha_1\|^2}{4} = \frac{1}{4}$	$\|10\rangle \otimes (\alpha_0\|0\rangle - \alpha_1\|1\rangle)$
$\|11\rangle$	$\frac{\|-\alpha_1\|^2}{4} + \frac{\|\alpha_0\|^2}{4} = \frac{1}{4}$	$\|11\rangle \otimes (-\alpha_1\|0\rangle + \alpha_0\|1\rangle)$

Note that for every possible partial measurement Alice makes, the resulting state of Bob's qubit is equal to or very close to Alice's original $|\psi\rangle$. To finish the job, Alice can call up Bob on the telephone and tell him what her partial measurement was. What Bob does next then splits into four cases.

- If Alice says $|00\rangle$, then Bob knows his qubit is in state $\begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix} = |\psi\rangle$.
- If Alice says $|01\rangle$, then Bob applies a NOT gate, $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$, to $\begin{bmatrix}\alpha_1\\ \alpha_0\end{bmatrix}$ to get $|\psi\rangle$.
- If Alice says $|10\rangle$, then Bob applies a $\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$ gate to $\begin{bmatrix}\alpha_0\\ -\alpha_1\end{bmatrix}$ to get $|\psi\rangle$.
- If Alice says $|11\rangle$, then Bob applies a $\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}$, to $\begin{bmatrix}-\alpha_1\\ \alpha_0\end{bmatrix}$ to get $|\psi\rangle$.

Thus, using only an EPR pair and two classical bits, Alice was able to send to Bob her quantum state. This experiment is *not* a violation of the no-cloning theorem since Alice no longer has a copy of the quantum state. Additionally, this is a not a violation of Einstein's special/general relativity since the necessary classical bits could not have been communicated faster than the speed of light. One though could counter this with the following thought experiment. Imagine that Bob measured his qubit *immediately* after Alice measures hers (faster than the time it takes light to travel from Alice's house to Bob's house). Doesn't Bob then learn something about $|\psi\rangle$ faster than the speed of light? It turns out the answer to that question is no. Before Alice sent her qubits through the quantum circuit, if Bob were to measure his EPR-pair, he would see $|0\rangle$ with probability 1/2 and $|1\rangle$ with probability 1/2. After Alice uses her quantum circuit (including the measurement), the probability Bob sees $|0\rangle$ after his measurement is

$$\frac{1}{4}|\alpha_0|^2 + \frac{1}{4}|\alpha_1|^2 + \frac{1}{4}|\alpha_0|^2 + \frac{1}{4}|-\alpha_1|^2 = \frac{1}{2}(|\alpha_0|^2 + |\alpha_1|^2) = \frac{1}{2},$$

which is the exact same probability as before. Thus, until Alice tells Bob her two classical bits, Bob cannot learn anything about $|\psi\rangle$, and thus relativity is not violated in this example.

It turns out this procedure (see [Die82]), called *quantum teleportation*does not merely works in theory but has also been verified to work in practice. This was first verified in 1992 by Bennett, et. al., [BBC$^+$93]. In 2012, Ma, et. al., [MHS$^+$12] performed quantum teleportation at a distance of 143 kilometers. One drawback though to quantum teleportation is that it does not help someone 'believe' in quantum mechanics. That is, in order to interpret the results of these experiments one needs to already accept quantum mechanics. Soon, we discuss another experiment called the CHSH game which *does* give definitive proof that our world is quantum mechanical.

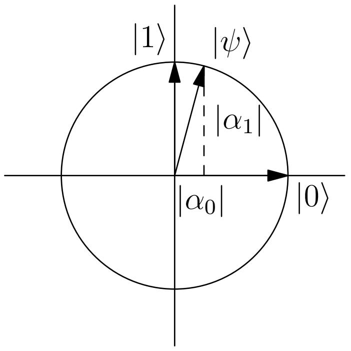
3 Measuring in a different basis

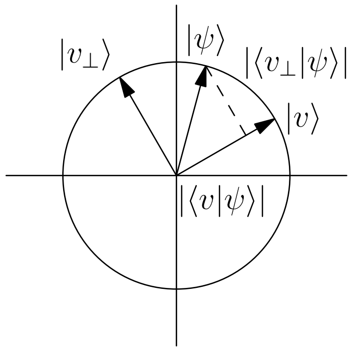
3.1 Measuring in an orthonormal basis

When we write $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, we are expressing $|\psi\rangle$ in terms of the basis $\{|0\rangle, |1\rangle\}$. This basis is called the *standard* or *computational* basis. When we perform a measurement on $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, we see $|0\rangle$ with probability $|\alpha_0|^2$ and $|1\rangle$ with probability $|\alpha_1|^2$. Since $\alpha_0 = \langle 0|\psi\rangle$ and $\alpha_1 = \langle 1|\psi\rangle$, we can rewrite these probabilities as

$$\begin{aligned}|\alpha_0|^2 &= \alpha_0^\dagger\alpha_0 = \langle\psi|0\rangle\langle 0|\psi\rangle\\ |\alpha_1|^2 &= \alpha_1^\dagger\alpha_1 = \langle\psi|1\rangle\langle 1|\psi\rangle.\end{aligned}$$

To visualize this, imagine that $\alpha_0, \alpha_1 \in \mathbb{R}$, thus we can image $|\psi\rangle = \begin{bmatrix}\alpha_0\\ \alpha_1\end{bmatrix}$ as a vector on the unit circle.





If we project $|\psi\rangle$ onto the x (or $|0\rangle$)-axis, the length of the resulting vector is $|\alpha_0|^2$. Furthermore, the distance from $|\psi\rangle$ to the projection is $|\alpha_1|^2$. Since the two resulting segments for a right triangle, we have from the Pythagorean theorem that $|\alpha_0|^2 + |\alpha_1|^2 = 1$, which is a familiar formula.

We can extend this analogy to an arbitrary orthonormal basis $\{|v\rangle, |v^\perp\rangle\}$. If we write $|\psi\rangle = \beta_v|v\rangle + \beta_{v^\perp}|v^\perp\rangle$, then we have that $\beta_v = \langle v|\psi\rangle$ and $\beta_{v^\perp} = \langle v^\perp|\psi\rangle$. Thus, projecting $|\psi\rangle$ onto vectors $|v\rangle$ and $|v^\perp\rangle$ results in a right triangle with legs $|\beta_v|$ and $|\beta_{v^\perp}|$.

Notice that since $\{v, v^\perp\}$ is an orthonormal basis,

$$\begin{aligned}
1 &= \langle\psi|\psi\rangle \\
&= (\beta_v^\dagger\langle v| + \beta_{v^\perp}^\dagger\langle v^\perp|)(\beta_v|v\rangle + \beta_{v^\perp}|v^\perp\rangle) \\
&= \beta_v^\dagger\beta_v\langle v|v\rangle + \beta_v^\dagger\beta_{v^\perp}\langle v|v^\perp\rangle + \beta_{v^\perp}^\dagger\beta_v\langle v^\perp|v\rangle + \beta_{v^\perp}^\dagger\beta_{v^\perp}\langle v^\perp|v^\perp\rangle \\
&= |\beta_v|^2 + |\beta_{v^\perp}|^2.
\end{aligned}$$

Thus, it is quite natural to discuss a probability distribution which says '$|v\rangle$' with probability $|\beta_v|^2$ and says '$|v^\perp\rangle$' with probability $|\beta_{v^\perp}|^2$. This is in fact the definition of measuring in a different basis.

Definition 3.1. Let $\{v, v^\perp\}$ be an orthonormal basis. The process of *measuring* $|\psi\rangle = \beta_v|v\rangle + \beta_{v^\perp}|v^\perp\rangle$ *in the basis* $\{v, v^\perp\}$ is a quantum circuit with a measurement which upon input $|\psi\rangle$ outputs $|0\rangle$ (representing an answer of '$|v\rangle$') with probability $|\beta_v|^2$ and outputs $|1\rangle$ (representing an answer of '$|v^\perp\rangle$') with probability $|\beta_{v^\perp}|$.

It turns out that a simple quantum circuit allows us to measure in the basis $\{v, v^\perp\}$, it consists of a one-qubit gate U and a measurement. Clearly this gate should have the property that $U|v\rangle = |0\rangle$ and $U|v^\perp\rangle = |1\rangle$. As discussed in Lecture 2, we must then have that $U = |0\rangle\langle v| + |1\rangle\langle v^\perp|$. It is easy then to see that if $|\psi\rangle = \beta_v|v\rangle + \beta_{v^\perp}|v\rangle$, then measuring $U|\psi\rangle = \beta_v|0\rangle + \beta_{v^\perp}|1\rangle$ yields $|0\rangle$ with probability $|\beta_v|^2$ and $|1\rangle$ with probability $|\beta_{v^\perp}|^2$, as desired.

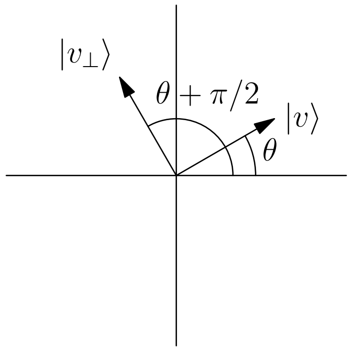
Remark 3.2. It is a standard assumption in quantum computing that all unitary 1-qubit and 2-qubit gates are available to be used at unit cost. This is not necessary since any 1-qubit or 2-qubit gate can be simulated to arbitrary precision with ancillas, CCNOT gates, and Hadamard gates; but this assumption makes quantum circuits easier to reason about.

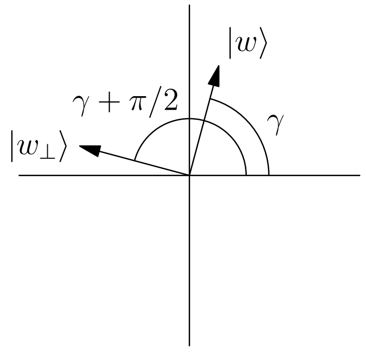
3.2 Example

Often, the basis we which to measure in is a counter-clockwise rotation of the standard basis by an angle of θ. Thus, the change of basis matrix we desire for measuring is the *clockwise* rotation matrix by the angle of θ, which we denote as Rot_θ

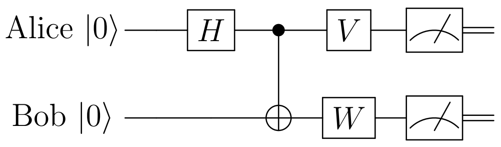
$$\text{Rot}_\theta = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}.$$

Consider a pair of orthonormal bases $\{v, v^\perp\}$ and $\{w, w^\perp\}$ such that $\{v, v^\perp\}$ is a θ radian counter-clockwise rotation of the standard basis, and $\{w, w^\perp\}$ is a γ radian counter-clockwise rotation of the standard basis.





Thus, the unitary matrix V which allows us to measure in the basis $\{v, v^\perp\}$ corresponds to a *clockwise* rotation by θ, $V = \text{Rot}_\theta$. Similarly, the unitary matrix W which allows us to measure in the basis $\{w, w^\perp\}$ is $W = \text{Rot}_\gamma$. Imagine again that Alice and Bob share an EPR pair $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ (Alice owns the first qubit and Bob the second). If Alice applies V to her qubit, and Bob applies W to his qubit.



The resulting state can summarized by the following lemma.

Lemma 3.3. *Let θ and γ be angles and let $\Delta = \theta - \gamma$. Then,*

$$(\text{Rot}_\theta \otimes \text{Rot}_\gamma)\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) = \frac{1}{\sqrt{2}}(\cos\Delta|00\rangle + \sin\Delta|01\rangle - \sin\Delta|10\rangle + \cos\Delta|11\rangle).$$

Proof. This result can be verified by direct computation.

$$\begin{aligned}
(\mathrm{Rot}_\theta \otimes \mathrm{Rot}_\gamma)\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) &= \frac{1}{\sqrt{2}}(\cos\theta|0\rangle - \sin\theta|1\rangle) \otimes (\cos\gamma|0\rangle - \sin\gamma|1\rangle) \\
&\quad + \frac{1}{\sqrt{2}}(\sin\theta|0\rangle + \cos\theta|1\rangle) \otimes (\sin\gamma|0\rangle + \cos\gamma|1\rangle) \\
&= \frac{1}{\sqrt{2}}(\cos\theta\cos\gamma + \sin\theta\sin\gamma)|00\rangle \\
&\quad + \frac{1}{\sqrt{2}}(-\cos\theta\sin\gamma + \sin\theta\cos\gamma)|01\rangle \\
&\quad + \frac{1}{\sqrt{2}}(-\sin\theta\cos\gamma + \cos\theta\sin\gamma)|01\rangle \\
&\quad + \frac{1}{\sqrt{2}}(\sin\theta\sin\gamma + \cos\theta\cos\gamma)|11\rangle \\
&= \frac{1}{\sqrt{2}}(\cos\Delta|00\rangle + \sin\Delta|01\rangle - \sin\Delta|10\rangle + \cos\Delta|11\rangle).
\end{aligned}$$

□

Note that is $\theta = \gamma$, then $\Delta = 0$, and the resulting state is $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, which implies the operation restored the original EPR pair. Thus, when Alice and Bob measure an EPR pair in the same basis, they always get the same result. The lemma easily yields the following corollary about measuring the qubits.

Corollary 3.4. *After applying* $\mathrm{Rot}_\theta \otimes \mathrm{Rot}_\gamma$ *to an EPR pair, perform a measurement. The probability that both qubits in the collapsed state have the same value is* $\cos^2\Delta = \cos^2(\theta-\gamma)$*. Likewise, the probability that both qubits collapse to different states is* $\sin^2\Delta$.

Proof. The probability of measuring $|00\rangle$ is $\frac{1}{2}\cos^2\Delta$. Similarly, the probability of measuring $|11\rangle$ is $\frac{1}{2}\cos^2\Delta$. Therefore, the probability of both qubits collapsing to the same value is $\cos^2\Delta$. This directly implies that the probability of the qubits collapsing to different values is $1 - \cos^2\Delta = \sin^2\Delta$. □

If $\Delta = \pi/2$, then the probability of both qubits collapsing to different values is 1. This makes sense, since the bases Alice and Bob are measuring in are a $\pi/2$ rotation of each other.

4 CHSH Game

To demonstrate the power of entanglement, we discuss how quantum computation can give the players an edge in the following combinatorial game. Such a game was suspected by John Bell [Bel64], and was formulated by Clauser, et. al., [CHSH69].

4.1 Game formulation

Definition 4.1. The *CHSH Game* consists of a team of two players Alice and Bob who are assisted by two referees Ref. 1 and Ref. 2. Alice and Bob are separated sufficiently far away (say 1 light-second) so that they cannot communicate with each other during the game, but Alice is sufficiently close to Ref. 1 and Bob is sufficiently close to Ref. 2. At the start of the game, Ref. 1 and Ref. 2 pick select uniformly random bits $x, y \in \{0, 1\}$, respectively. Ref. 1 tells Alice x and Ref. 2 tells Bob y. Alice is then to respond with a bit $a \in \{0, 1\}$ and Bob is to respond with a bit $b \in \{0, 1\}$. Alice and Bob win if and only if $a \otimes b = x \wedge y$.

Another way to phrase the winning condition, is that Alice and Bob must produce the same bit when $(x, y) \neq (1, 1)$ and must come up with different bits otherwise. Alice and Bob are allowed to agree to a strategy before hand.

4.2 Classical strategy

It is easy to come up with a strategy in which Alice and Bob win with probability 3/4: have both players always respond with 0. It turns out that no classical strategy can do better.

Lemma 4.2. *No classical deterministic or randomized strategy which allows Alice and Bob to win with probability greater than* 3/4.

Proof. First, we prove that any deterministic strategy results in Alice and Bob winning at most 3/4 of the time. In a deterministic strategy, Alice's bit a must be a function of her random bit x. Thus, Alice must choose that either $a(x) = 0$, $a(x) = 1$, $a(x) = x$, or $a(x) = \neg x$. Similarly, Bob must choose between $b(y) = 0$, $b(y) = 1$, $b(y) = y$, and $b(y) = \not{y}$. The winning probability of each pair of strategies is summarized in the following table.

	$a = 0$	$a = 1$	$a = x$	$a = \neg x$
$b = 0$	3/4	1/4	3/4	1/4
$b = 1$	1/4	3/4	1/4	3/4
$b = y$	3/4	1/4	1/4	3/4
$b = \neg y$	1/4	3/4	3/4	1/4

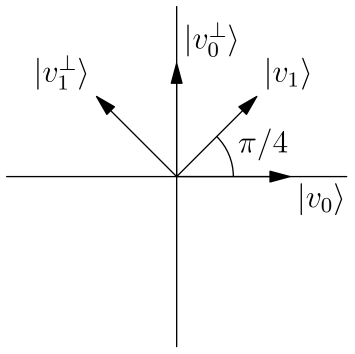
Thus, there is no deterministic strategy which beats 3/4. Since any randomized strategy is a probability distribution of these deterministic strategies, the best a randomized distribution can do is also 3/4. □

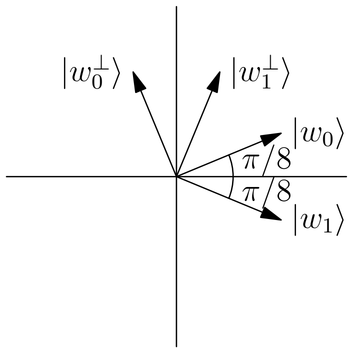
4.3 Quantum strategy

We now show how to beat 3/4 using a quantum strategy. For this quantum strategy to work, we will have Alice and Bob each share a qubit of an EPR pair. Alice and Bob will independently decide which basis to measure their qubit in the EPR pair based on the random bit they receive. By exploiting the correlations of the EPR pair, Alice and Bob will get a win probability significantly greater than 3/4.

Theorem 4.3. *There is a quantum strategy which allows Alice and Bob to win with probability* $\cos^2(\pi/8) \approx .85 > 3/4$*.*

Proof. Have Alice and Bob share an EPR pair $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Alice will measure her qubit in either basis $\{v_0, v_0^\perp\} = \{|0\rangle, |1\rangle\}$ or $\{|v_1\rangle, |v_1^\perp\rangle\} = \{|+\rangle, |-\rangle\}$, which is a $\pi/4$ counterclockwise rotation of the standard basis. Similarly, Bob will measure his qubit in either $\{|w_0\rangle, |w_0^\perp\rangle\}$ which is a $\pi/8$ rotation of the standard basis or $\{|w_1\rangle, |w_1^\perp\rangle\}$ which is a $-\pi/8$ rotation of the standard basis.





Now, when Alice and Bob receive x and y, respectively, Alice will measure in basis $\{|v_x\rangle, |v_x^\perp\rangle\}$ to determine a and Bob will measure in basis $\{|w_x\rangle, |w_x^\perp\rangle\}$ to determine b. Applying Corollary 3.4, we have that the possible scenarios are as follows

x	y	Alice's rotation θ	Bob's rotation γ	$\Delta = \theta - \gamma$	$\mathbf{Pr}$[win]
0	0	0	$\pi/8$	$-\pi/8$	$\cos^2(-\pi/8)$
0	1	0	$-\pi/8$	$\pi/8$	$\cos^2(\pi/8)$
1	0	$\pi/4$	$\pi/8$	$\pi/8$	$\cos^2(\pi/8)$
1	1	$\pi/4$	$-\pi/8$	$3\pi/8$	$\sin^2(3\pi/8)$

We take the sine instead of the cosine when $x = y = 1$ since we want $a \neq b$ in that case. In all four scenarios the probability of winning is equal to $\cos^2(\pi/8)$, so that is the overall win probability, as desired. □

4.4 Experimental verification

Like with quantum teleportation, multiple experiments have been done to verify that the quantum strategy beats the classical one. The first such experiment was done in 1981-82 by Aspect et. al. [AGR81, AGR82, ADR82]. Although the these results very much supported quantum mechanics, there were still "classical" explanations of the results. In 2015, this experiment was redone by Hensen, et. al., [HBD$^+$15] at the University of Delft in a "loophole free" manner to essentially eliminate the possibility that our universe is *not* quantum mechanical. For a more detailed discussion, see Aaronson's article on the topic [Aar15].

References

[Aar15] Scott Aaronson. Bell inequality violation finally done right. *Shtetl-Optimized*, sep 2015. URL: `http://www.scottaaronson.com/blog/?p=2464`.

[ADR82] A. Aspect, J. Dalibard, and G. Roger. Experimental Test of Bell's Inequalities Using Time- Varying Analyzers. *Physical Review Letters*, 49:1804–1807, December 1982. `doi:10.1103/PhysRevLett.49.1804`.

[AGR81] A. Aspect, P. Grangier, and G. Roger. Experimental Tests of Realistic Local Theories via Bell's Theorem. *Physical Review Letters*, 47:460–463, August 1981. `doi:10.1103/PhysRevLett.47.460`.

[AGR82] A. Aspect, P. Grangier, and G. Roger. Experimental Realization of Einstein-Podolsky-Rosen-Bohm Gedankenexperiment: A New Violation of Bell's Inequalities. *Physical Review Letters*, 49:91–94, July 1982. `doi:10.1103/PhysRevLett.49.91`.

[BBC+93] Charles H Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Physical review letters*, 70(13):1895, 1993.

[Bel64] John S Bell. On the einstein-podolsky-rosen paradox. *Physics*, 1(3):195–200, 1964.

[CHSH69] John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23:880–884, Oct 1969. URL: `http://link.aps.org/doi/10.1103/PhysRevLett.23.880`, `doi:10.1103/PhysRevLett.23.880`.

[Die82] D. Dieks. Communication by epr devices. *Physics Letters A*, 92(6):271 – 272, 1982. URL: `http://www.sciencedirect.com/science/article/pii/0375960182900846`, `doi:http://dx.doi.org/10.1016/0375-9601(82)90084-6`.

[dW11] Ronald de Wolf. Quantum computing: Lecture notes, 2011. URL: `http://homepages.cwi.nl/~rdewolf/qcnotes.pdf`.

[EPR35] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935. URL: `http://link.aps.org/doi/10.1103/PhysRev.47.777`, `doi:10.1103/PhysRev.47.777`.

[HBD+15] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenberg, R. F. L. Vermeulen, R. N. Schouten, C. Abellán, W. Amaya, V. Pruneri,

M. W. Mitchell, M. Markham, D. J. Twitchen, D. Elkouss, S. Wehner, T. H. Taminiau, and R. Hanson. Experimental loophole-free violation of a Bell inequality using entangled electron spins separated by 1.3 km. *ArXiv e-prints*, August 2015. `arXiv:1508.05949`.

[MHS$^+$12] Xiao-Song Ma, Thomas Herbst, Thomas Scheidl, Daqing Wang, Sebastian Kropatschek, William Naylor, Bernhard Wittmann, Alexandra Mech, Johannes Kofler, Elena Anisimova, et al. Quantum teleportation over 143 kilometres using active feed-forward. *Nature*, 489(7415):269–273, 2012.

[WZ82] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

Quantum Computation **(CMU 15-859BB, Fall 2015)**

Lecture 4: Grover's Algorithm

September 21, 2015

Lecturer: John Wright *Scribe: Tom Tseng*

1 Introduction

In the previous lecture, we discussed how a quantum state may be transferred via quantum teleportation and how quantum mechanics is, in a sense, more powerful than classical mechanics in the CHSH game. Today, we are going to move on to discussing how we can apply quantum mechanics to computer science. We will see our first example of a quantum algorithm: Grover's algorithm, described in a paper published by Lov Grover in 1996 [Gro96]. Much of the excitement over quantum computation comes from how quantum algorithms can provide improvements over classical algorithms, and Grover's algorithm exhibits a quadratic speedup.

2 Grover's algorithm

2.1 What it solves

Grover's algorithm solves the problem of *unstructured search*.

Definition 2.1. In an *unstructured search* problem, given a set of N elements forming a set $X = \{x_1, x_2, \dots, x_N\}$ and given a boolean function $f : X \to \{0, 1\}$, the goal is to find an element x^* in X such that $f(x^*) = 1$.

Unstructured search is often alternatively formulated as a database search problem in which we are given a database and we want to find an item that meets some specification. For example, given a database of N names, we might want to find where your name is located in the database.

The search is called "unstructured" because we are given no guarantees as to how the database is ordered. If we were given a sorted database, for instance, then we could perform binary search to find an element in logarithmic time. Instead, we have no prior knowledge about the contents of the database. With classical circuits, we cannot do better than performing a linear number of queries to find the target element.

Grover's algorithm leads to the following theorem:

Theorem 2.2. *The unstructured search problem can be solved in $\mathcal{O}(\sqrt{N})$ queries using quantum computation.*

In fact, this is within a constant factor of the best we can do for this problem under the quantum computation model, i.e. the query complexity is $\Theta(\sqrt{N})$ [BBBV97]. In particular, since we cannot solve the unstructured search problem in logarithmic time, this problem cannot be used as a way to solve NP problems in polynomial time; if we did have a logarithmic time algorithm, we could, given n variables arranged in clauses, solve the SAT problem by searching all 2^n possibilities in $\mathcal{O}(\log(2^n)) = \mathcal{O}(n)$ queries after a few manipulations. Nonetheless, solving the search problem in $\Theta(\sqrt{N})$ queries is still significantly better than what we can do in the classical case.

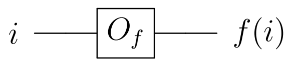
In addition, Grover's algorithm uses only $\mathcal{O}(\sqrt{N}\log N)$ gates. If we think of the number of gates as being roughly running time, then we have running time almost proportional to $\sqrt{N}$.

There is a caveat: the algorithm only gets the correct answer with high probability, say with probability greater than $\frac{2}{3}$. Of course, we can then make the probability of correctness an arbitrarily large constant. We do this by running the algorithm multiple times to get many different outputs. Then we can run each of our inputs through f to see if any match the criterion desired. This strategy only fails if all outputs fail, and since these are all independent outputs, our probability of failure is $\left(\frac{1}{3}\right)^m$.

Note that even with classical algorithms that only need to output the correct answer with two-thirds probability, we still need a linear number of queries. Therefore, this caveat does not provide an unfair advantage to the quantum algorithm.

2.2 The classical case

We need some sort of model for how we are allowed to interact with our database. In a classical algorithm, our medium of interaction is an *oracle* O_f that *implements* a function f. The function f encodes information about the element of interest, and we query the oracle to fetch results from the function. Think of the oracle as a quantum circuit or a big gate. This oracle, given an input of i, outputs $f(i)$ in constant time. We do not care about the details of how it works, and instead treat it as a black box.



As we do with any good algorithm, we want to limit resources used, which in this case are running time and number of queries to the oracle. Of course, in any classical algorithm, the best we can do is $\Theta(N)$ queries and $\Theta(N)$ running time – because the data is unstructured, we cannot help but look at every element in the worst case no matter how cleverly we choose our queries.

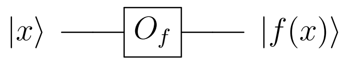
(Note that, in general, number of queries and running time are not the same, but in this case they are both linear. We care more about queries because they are simpler to reason about, plus they represent a lower bound to running time. Running time by itself involves many other factors that we do not know much about. In this course, we will be more concerned about number of queries as a measure of resources used.)

This is the end of the story in the classical situation; we have tight upper and lower bounds, and not much more can be done. Now we shall switch to the quantum setting.

2.3 Modifications in the quantum case

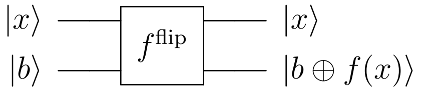
Right off the bat, we're going make some simplifications. First, we are going to assume that the element x^* that satisfies $f(x^*) = 1$ is unique. Second, we are going to assume the size of our database is is a power of two, letting $N = 2^n$ where N is the size of the database. Lastly, we are also going assume that the data is labeled as n-bit boolean strings in $\{0,1\}^n$ rather than being indexed from 1 to N. In turn, our boolean function f that we are studying will map $\{0,1\}^n$ to $\{0,1\}$. These conditions will be more convenient for us, and we do not lose much; the algorithm can be extended to relax the uniqueness assumption, we can always add garbage entries to the database to make the size of the database a power of two, and the naming of the elements is arbitrary.

Our interface with our database is not much different compared to the interface in the classical case. We have an oracle O_f that we give $|x\rangle$, and it will output $|f(x)\rangle$, as shown below:



We immediately see a problem: this oracle gate is not a valid quantum gate. It takes an n-bit input and gives a 1-bit output, and thus is not unitary or even reversible. There are a couple of ways we can fix this.

Our first possible fix is to give the gate an extra bit $|b\rangle$ to use for the output. Call this new gate the f^{flip} gate.

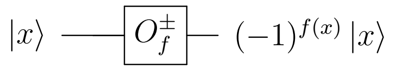


In particular, if we have $|b\rangle$ be an ancillary $|0\rangle$ qubit, it will end up as $|f(x)\rangle$ after passing through the gate. However, this gate is somewhat unwieldy because we have to carry around that extra qubit $|b\rangle$ with us. That brings us to our next fix.

Our second possible fix is to flip the input if and only if $f(x)$ is 1. We will call the gate that does this $O_f^{\pm}$. Given an input $|x\rangle$, $O_f^{\pm}$ will output

$$(-1)^{f(x)} |x\rangle = \begin{cases} |x\rangle & \text{if } f(x) = 0 \\ -|x\rangle & \text{if } f(x) = 1. \end{cases}$$

The diagram of the gate is given below.



This avoids the annoyance of the extra bit that our first fix had.

These two gates are both valid versions of the oracle, and they are equivalent in the sense that one can be constructed from the other. For example, we will prove that $O_f^{\pm}$ may be constructed using f^{flip}.

Proof. Suppose we wanted to construct the $O_f^{\pm}$ gate given access to the f^{flip} gate. Then we choose our ancillary bit that we send along with $|x\rangle$ into f^{flip} to be $|-\rangle$.

Before we hit the f^{flip} gate, we have the state

$$|x\rangle \oplus |-\rangle = |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}(|x0\rangle - |x1\rangle).$$

After we pass the gate, there are two cases. If $f(x) = 0$, then the state is unchanged. Otherwise, if $f(x) = 1$, then we end up with the state

$$|x\rangle \otimes \frac{1}{\sqrt{2}}(|0 \oplus 1\rangle - |1 \oplus 1\rangle) = |x\rangle \otimes \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) = \frac{1}{\sqrt{2}}(|x1\rangle - |x0\rangle)$$

which is the negative of the state we began with. Thus we have

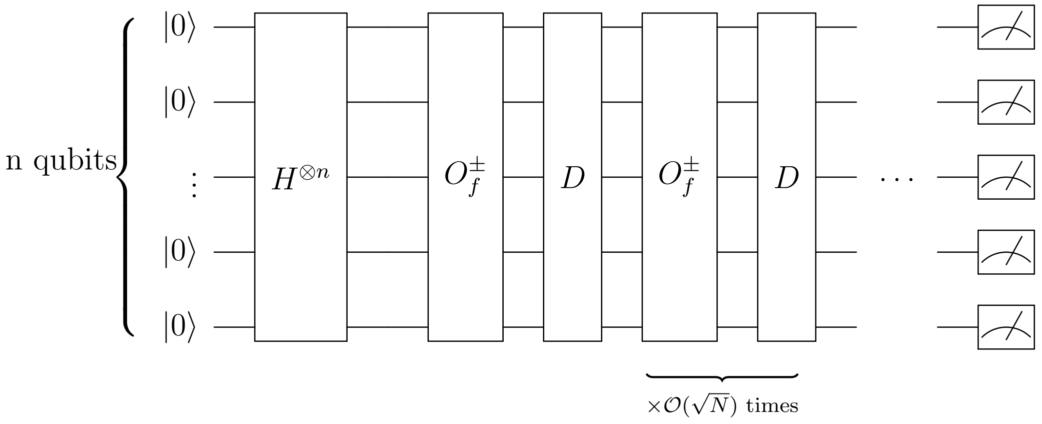
$$|x\rangle \otimes |-\rangle \mapsto (-1)^{f(x)}\left(|x\rangle \otimes |-\rangle\right)$$

as desired to simulate the $O_f^{\pm}$ gate. $\square$

Because the gates are equivalent, we may use either. For the circuits in this lecture, we will use the $O_f^{\pm}$ gate.

2.4 The algorithm

Recall that our goal is to, given the ability to query f using our oracle gate, find x^* such that $f(x^*) = 1$. We shall start by diagramming the entire circuit below and explain the details following that.



We start with n qubits all initialized to $|0\rangle$. Think of these as forming an n-bit string, or an input to f. Let us make our goal concrete by saying that we would like the amplitude of $|x^*\rangle$ to be at least .1 in magnitude at the end of the circuit so that when we measure at the end, we have a $.1^2 = .01$ probability of measuring $|x^*\rangle$.

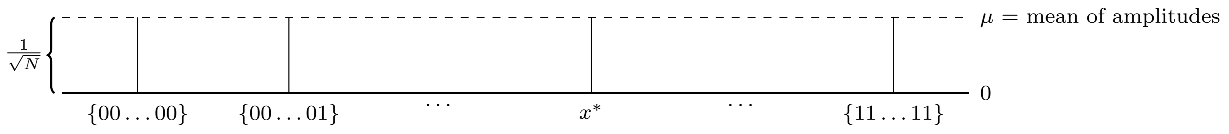
What can we do with our quantum circuit that is not possible classically? We can perhaps exploit superposition. We shall begin by transforming our input into the uniform superposition

$$\sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{N}} |x\rangle .$$

The uniform amplitudes of all the n-bit strings, in a sense, represents our current complete uncertainty as to what x^* is.

We achieve this uniform superposition of all the basis states by sticking a Hadamard gate on every wire, or equivalently a $H^{\otimes n}$ gate on all wires. This superposition trick is common in quantum algorithms, so get used to it.

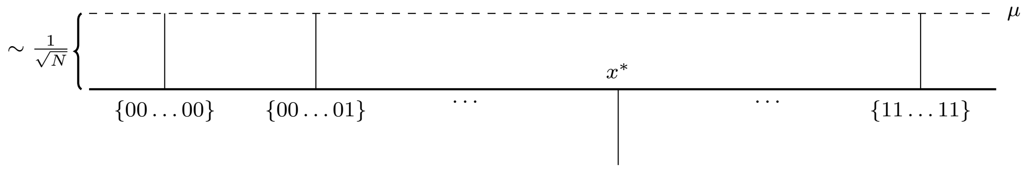
The current state can be visualized with a diagram representing amplitudes in a fashion similar to a bar graph.



Now we shall query this state by adding an oracle $O_f^{\pm}$ gate. That flips the amplitude of the x^* component and leaves everything else unchanged, giving us a state of

$$-\frac{1}{\sqrt{N}}|x^*\rangle + \sum_{\substack{x\in\{0,1\}^n \\ x\neq x^*}} \frac{1}{\sqrt{N}}|x\rangle .$$

Graphically, we now have



What we want is to increase the amplitude of x^* absolutely. This motivates the introduction of a new gate that performs what is called the *Grover diffusion operator*. What does this gate do? First, we have to define a new number

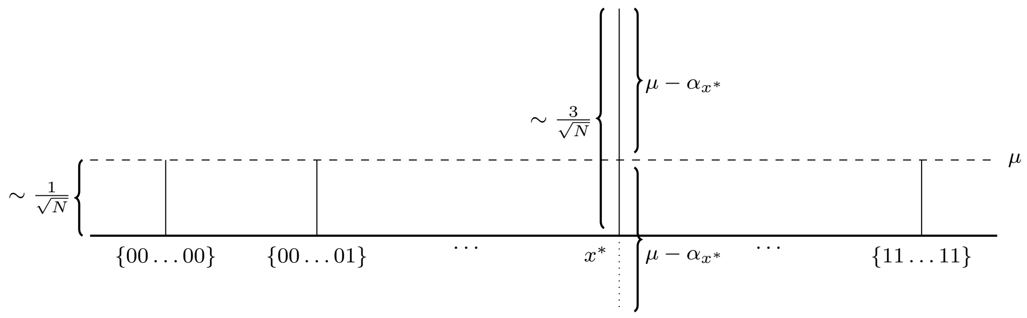
$$\mu = \frac{1}{N}\sum_x \alpha_x = \text{average of } \alpha_x \text{ for } x \in \{0,1\}^n.$$

where for α_x is the amplitude of x for a given x in $\{0,1\}^n$. Our Grover diffusion gate is defined to have the mapping

$$\sum_{x\in\{0,1\}} \alpha_x |x\rangle \mapsto \sum_{x\in\{0,1\}^n} (2\mu - \alpha_x)|x\rangle .$$

Note that this is a valid quantum gate; it is linear since all the operations in the mapping are linear, and it is also unitary. We can show it is unitary by verifying that the operation preserves the norm of the input. In addition, we will later construct the gate out of other unitary gates.

O.K., what does this gate *really* do? It flips amplitudes around the average, μ. To illustrate this, we will apply the gate, labeled D on the original circuit diagram, after the oracle gate. Here is what our amplitudes look like after application:

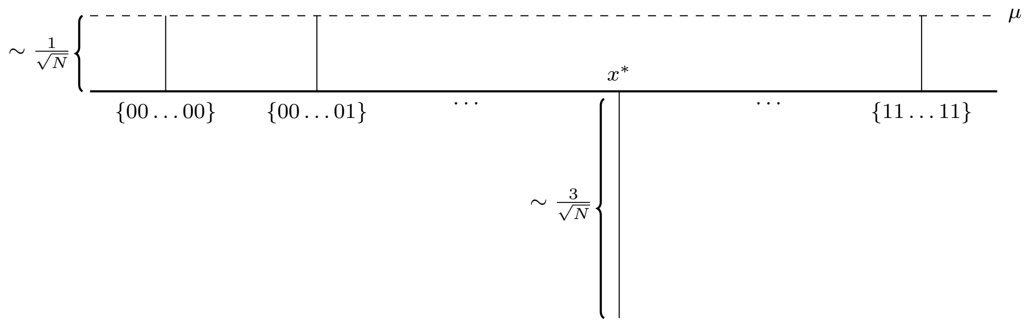


The mean amplitude prior to applying the gate was

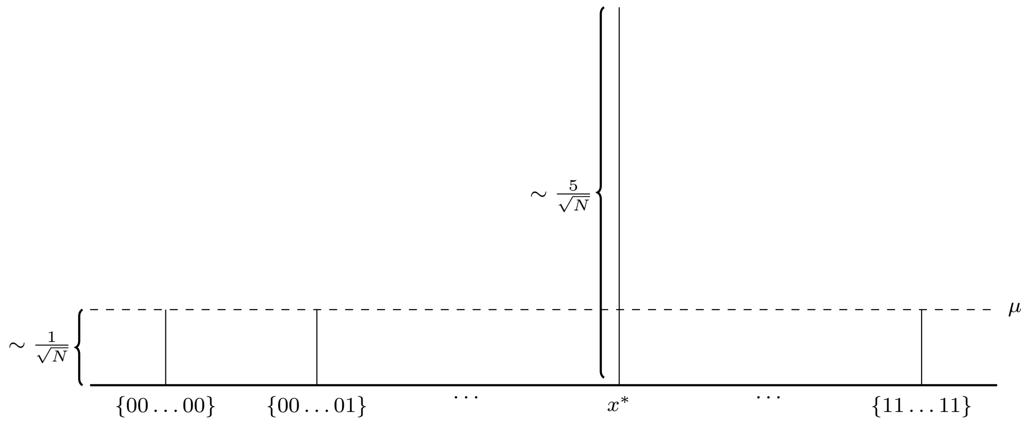
$$\sum_{x\in\{0,1\}^n} \alpha_x = \frac{2^n-1}{\sqrt{N}} - \frac{1}{\sqrt{N}} = \frac{2^n-2}{\sqrt{N}} \approx \frac{N}{\sqrt{N}} = \frac{1}{\sqrt{N}}$$

i.e. $\frac{1}{\sqrt{N}}$ minus an insignificant amount. That means when the "reflection around the mean" occurs with the application of the Grover diffusion operator, almost all the entries $x \in \{0,1\}^n$ stay roughly the same. But – and here is where the magic happens – the amplitude of x^* gets magnified to about $\frac{3}{\sqrt{N}}$ in magnitude!

Still, this is quite far from our goal of making the amplitude of x^* greater than a constant. To make further progress, we can try the most brain-dead thing possible: apply the oracle and the Grover diffusion gate again! After applying $O_f^{\pm}$ for the second time, we have



and after applying the Grover diffusion for the second time, we obtain



The computation to see that this is true is more or less the same. We flip x^* to its negative, and then flip it around the mean while everything else stays roughly unchanged.

The intuitive, high-level picture of what is happening is that α_{x^*} is going up by more than $1/\sqrt{N}$ each time we repeat the application of the pair of gates $O_f^{\pm}$ and D. With that logic, after $\mathcal{O}(\sqrt{N})$ steps, α_{x^*} should exceed the constant .1 as desired.

Of course, α_{x^*} cannot actually increase by $1/\sqrt{N}$ at every step since the amplitude cannot go above 1. The increase of α_{x^*} has to slow down or reverse at some point. For instance, if α_{x^*} is sufficiently large, then when we flip α_{x^*} to negative by using the $O_f^{\pm}$ gate, this large negative value might actually drag the mean down to be negative. With a negative mean, the magnitude of x^* will decrease after using the Grover diffusion gate. In light of that, we must be careful when we are analyzing this algorithm to make sure we never get that kind of negative progress. However, so long as α_{x^*} is not too large, it seems we will always step closer to our goal.

2.5 Analysis

We have constructed the circuit, and now it remains to formally show that the math behind the circuit actually works out.

First, we will introduce some notation. Let $\alpha^{(t)}$ be the amplitude of x^* after the t-th step, where one step consists of applying an $O_f^{\pm}$ gate and then a Grover diffusion gate. Let $\beta^{(t)}$ be the amplitude of any other $x \in \{0,1\}^n$ after the t-th step. Finally, let $\mu^{(t)}$ be the mean of all the amplitudes, which is $-\alpha^{(t)} + \frac{N-1}{N}\beta^{(t)}$, after the oracle gate on the t-th step.

For example, $\alpha^{(0)} = \beta^{(0)} = \frac{1}{\sqrt{N}}$. Also, if we know the values for a given step t, we can calculate $\alpha^{(t+1)} = 2\mu^{(t)} + \alpha^{(t)}$ and so forth.

Now that notation is defined, we can move on to the body of the proof. We first show that α increases by a significant amount if α is not already too large.

Proposition 2.3. *Suppose $\alpha^{(t)} \leq 1/2$ and $N \geq 4$. Then $\alpha^{(t+1)} \geq \alpha^{(t)} + \frac{1}{\sqrt{N}}$.*

Proof. The sum of all the squares of the amplitudes is one, so

$$\begin{aligned} 1 &= (\alpha^{(t)})^2 + (N-1)(\beta^{(t)})^2 \\ &\leq \frac{1}{4} + (N-1)(\beta^{(t)})^2. \end{aligned}$$

With some rearranging, we get a bound on $\beta^{(t)}$ of

$$\beta^{(t)} \geq \sqrt{\frac{3}{4(N-1)}}.$$

Therefore,

$$\begin{aligned}
\mu^{(t)} &= \frac{-\alpha^{(t)} + (N-1)\beta^t}{N} \\
&\geq \frac{-\frac{1}{2} + (N-1)\sqrt{\frac{3}{4(N-1)}}}{N} \\
&= \frac{1}{2}\frac{(N-1)\sqrt{\frac{3}{N-1}} - 1}{N} \\
&= \frac{1}{2}\frac{\sqrt{3N-3} - 1}{N} \\
&\geq \frac{1}{2}\frac{1}{\sqrt{N}} \text{ given } N \geq 4.
\end{aligned}$$

Knowing this bound on $\mu^{(t)}$, we may calculate $\alpha^{(t+1)}$ to be

$$\alpha^{(t+1)} = 2\mu^{(t)} + \alpha^{(t)} \geq \frac{1}{\sqrt{N}} + \alpha^{(t)},$$

and the result is proven. □

The next proposition tells us that the increases of $\alpha^{(t+1)}$ are relatively controlled. This is useful because it will be used to show that $\alpha^{(t)}$ stays below one-half for a number of steps so that the previous proposition may be applied.

Proposition 2.4. *For any t, $\alpha^{(t+1)} \leq \alpha^{(t)} + \frac{2}{\sqrt{N}}$.*

Proof. For any given step t, $\alpha^{(t)}$ is positive, and therefore

$$\mu^{(t)} = \frac{-\alpha^{(t)} + (N-1)\beta^{(t)}}{N} \leq \frac{N-1}{N}\beta^{(t)}.$$

We also know that $(N-1)\left(\beta^{(t)}\right)^2 \leq 1$, and so $\beta^{(t)} \leq \frac{1}{\sqrt{N-1}}$. This gives us

$$\begin{aligned}
\alpha^{(t+1)} &= 2\mu^{(t)} + \alpha^{(t)} \\
&\leq 2\frac{N-1}{N}\beta^{(t)} + \alpha^{(t)} \\
&\leq 2\frac{N-1}{N}\frac{1}{\sqrt{N-1}} + \alpha^{(t)} \\
&= 2\frac{\sqrt{N-1}}{N} + \alpha^{(t)} \\
&\leq \frac{2}{\sqrt{N}} + \alpha^{(t)},
\end{aligned}$$

which is the desired result. □

Now we can show that we get $\alpha > .1$ with $\mathcal{O}(\sqrt{N})$ steps. If $N < 16$, then $\alpha^{(t)} = \frac{1}{\sqrt{N}} \geq .25$, so we are already done. Otherwise, if $N \geq 16$, as long as $t \leq \sqrt{N}/8$ we get

$$\begin{aligned}
\alpha^{(t)} &\leq \alpha^{(0)} + \frac{2}{\sqrt{N}}t \\
&\leq \alpha^{(0)} + \frac{2}{\sqrt{N}}\frac{\sqrt{N}}{8} \\
&= \frac{1}{\sqrt{N}} + \frac{1}{4} \\
&\leq \frac{1}{2}
\end{aligned}$$

by Proposition 2.4. This means that we may apply Proposition 2.3 for $\sqrt{N}/8$ steps to get

$$\alpha^{(\sqrt{N}/8)} \geq \frac{\sqrt{N}}{8}\frac{1}{\sqrt{N}} = \frac{1}{8} > .1,$$

and so we are indeed done with $\mathcal{O}(\sqrt{N})$ steps.

Of course, with just this one pass through the circuit, our probability of measuring x^* is only $\left(\alpha^{(\sqrt{N}/8)}\right)^2 > .01$. However, if we repeat the process a constant number of times, say 110 times, our probability of finding x^* is

$$\begin{aligned}
\mathbf{Pr}[\text{one output out of 110 being } x^*] &= 1 - \mathbf{Pr}[\text{output is not } x^*]^{110} \\
&\geq 1 - .99^{110} \\
&\geq \frac{2}{3}
\end{aligned}$$

which is certainly a respectable probability.

2.6 Gate complexity

We also promised an $\mathcal{O}(\sqrt{N}\log N)$ bound on the number of gates used in the algorithm. To prove this bound, it suffices to construct the Grover diffusion gate. To do so, we will invent yet another gate Z_0 defined by

$$Z_0\,|x\rangle = \begin{cases} |x\rangle & \text{if } |x\rangle = |0^n\rangle \\ -|x\rangle & \text{otherwise} \end{cases}$$

In unitary matrix form, $Z_0 = 2\,|0^n\rangle\,\langle 0^n| - I$ where I is the $n \times n$ identity matrix. We can confirm that this matrix indeed implements Z_0 by trying different inputs. If we give the input $|0^n\rangle$, we get

$$Z_0\,|0^n\rangle = 2\,|0^n\rangle\,\langle 0^n|0^n\rangle - |0^n\rangle = 2\,|0^n\rangle\,1 - |0^n\rangle = |0^n\rangle$$

as desired, and if we give some input $|x\rangle$ that has no 0^n component, then we know that $\langle 0^n|x\rangle = 0$, so we get

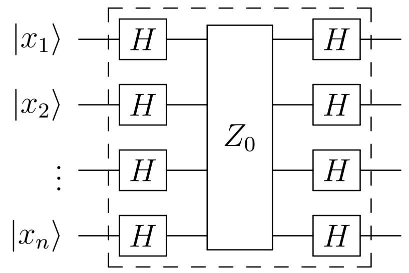
$$Z_0 |x\rangle = 2|0^n\rangle\langle 0^n|x\rangle - |x\rangle = -|x\rangle$$

also as desired.

Now we can construct the Grover diffusion gate, D, with a Z_0 gate and $\mathcal{O}(n) = \mathcal{O}(\log N)$ Hadamard gates. In matrix form, knowing that the Hadamard gate is its own inverse, we can write the diffusion gate as

$$\begin{aligned} D &= H^{\otimes n} Z_0 H^{\otimes n} \\ &= H^{\otimes n}(2|0^n\rangle\langle 0^n| - I)H^{\otimes n} \\ &= 2\left((H|0\rangle)^{\otimes n}\left((H|0\rangle)^{\otimes n}\right)^{\dagger}\right) - H^{\otimes n}H^{\otimes n} \\ &= 2|+^n\rangle\langle +^n| - I. \end{aligned}$$

It is the same as Z_0 but with $|+\rangle$ instead of $|0\rangle$. In diagram form, it is



Let us try giving the matrix an arbitrary input $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$ to make sure it works. We get

$$\begin{aligned} D|\psi\rangle &= (2|+^n\rangle\langle +^n| - I)|\psi\rangle \\ &= 2|+^n\rangle\langle +^n|\psi\rangle - |\psi\rangle. \end{aligned}$$

Notice that we can rewrite $\langle +^n|$ to get

$$\begin{aligned} \langle +^n| &= \left(\frac{\langle 0| + \langle 1|}{\sqrt{2}}\right)^{\otimes n} \\ &= \sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{N}} \langle x|, \end{aligned}$$

and therefore

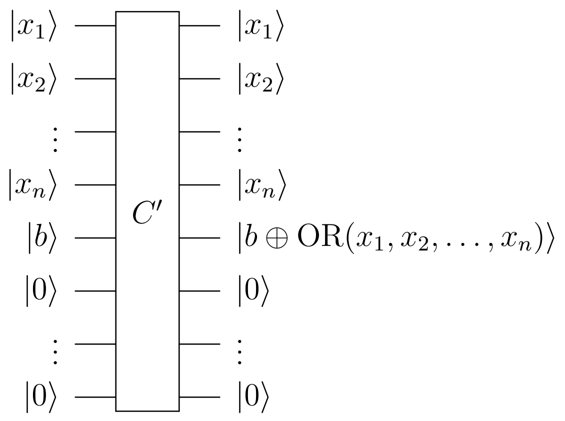
$$\begin{aligned} \langle +^n|\psi\rangle &= \sum_{x\in\{0,1\}^n} \frac{\alpha_x}{\sqrt{N}} \langle x|x\rangle \\ &= \sum_{x\in\{0,1\}^n} \frac{\alpha_x}{\sqrt{N}} \\ &= \mu\sqrt{N}. \end{aligned}$$

That means that we can continue simplifying $D|\psi\rangle$ to get

$$\begin{aligned}
D|\psi\rangle &= 2|+^n\rangle\langle +^n|\psi\rangle - |\psi\rangle \\
&= 2|+^n\rangle \mu\sqrt{N} - |\psi\rangle \\
&= 2\left(\frac{|0\rangle + |1\rangle}{\sqrt{N}}\right)\mu\sqrt{N} - |\psi\rangle \\
&= 2\left(\sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{N}}|x\rangle\right)\mu\sqrt{N} - |\psi\rangle \\
&= 2\left(\sum_{x\in\{0,1\}^n} \mu|x\rangle\right) - |\psi\rangle \\
&= \sum_{x\in\{0,1\}^n} (2\mu - \alpha_x)|x\rangle
\end{aligned}$$

which is precisely what we expect from the diffusion operator.

Lastly, we need to show that we actually can construct the Z_0 gate. What does Z_0 do? It negates $|x\rangle$ if and only if the logical OR of $\{x_1, x_2, \ldots, x_n\}$ is 1, where x_i is the i-th qubit that makes up $|x\rangle$. These kinds of logical operations are easy to compute. In particular, there is a classical circuit that computes the logical OR of n bits with $\mathcal{O}(n) = \mathcal{O}(\log N)$ Toffoli or CNOT gates. This circuit is not reversible, but of course we may use the results of problem 4 on homework 1 to fix that. Given a circuit C that computes the logical OR of n bits, we can construct a circuit C' with $m \in \mathbb{N}$ ancillary bits that does the following:



Now if we send $|x\rangle \otimes |-\rangle \otimes |0^m\rangle$ into C', we get $(-1)^{\mathrm{OR}(x)}(|x\rangle \otimes |-\rangle \otimes |0^m\rangle)$ out, which is exactly what Z_0 does. There are some extra gabrage bits, but those can be ignored.

Ultimately, what this all means is that the Grover diffusion gate can be constructed with $\mathcal{O}(\log N)$ gates. That means that Grover's algorithm takes $\mathcal{O}(\sqrt{N}\log N)$ gates to implement.

3 Conclusion

We now are closely familiar with Grover's algorithm, one of the most famous quantum algorithms at present. It demonstrates a provable polynomial improvement over its classical counterpart by combining the quantum features of superposition and negative amplitudes to solve the unstructured search problem. In this lecture, we showed how to find a unique entry x^* from a database using $\mathcal{O}(\sqrt{N})$ iterations, but, in general, if there are k entries that match the desired criterion, then Grover's algorithm can be extended to find an item in $\mathcal{O}(\sqrt{N/k})$ iterations. Next lecture, we'll discuss the quantum query model and learn about some other algorithms that are based on querying the black-box implementation of a function.

References

[BBBV97] Charlies H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[Gro96] Lov Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM Symposium on Theory of Computing*, pages 212–219, 1996.

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

Lecture 5: Quantum Query Complexity

September 23, 2015

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

1 Quick Summary

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

2 Grover Recap

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

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

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

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

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

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

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

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

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

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

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

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

3 The Query Model

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

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

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

4 Example Query Problems

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

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

4.1 The Hidden Shift Problem

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

Output: the hidden shift s.

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

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

Step 1: Prepare the state

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

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

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

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

4.2 Simon's Problem

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

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

Output: the hidden XOR-shift s.

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

Step 1: Prepare the state

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

Step 2: Apply a Fourier transformation over $\mathbb{Z}/n\mathbb{Z}$. (This turns out to be the same as applying $H^{\otimes n}$, i.e. applying a Hadamard gate to every register.)
Step 3: Do some more stuff.

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

4.3 Period Finding

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

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

Output: the period of f.

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

4.4 Deutsch-Josza

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

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

Output: whether the list is all-zero or balanced.

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

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

Step 0: Initialize n registers to $|000\ldots0\rangle$.
Step 1: Apply $H^{\otimes n}$, i.e. apply a Hadamard gate to each wire. The resulting state is

$$\frac{1}{\sqrt{N}}\sum_{x\in[N]}|x\rangle\,.$$

Step 2: Apply $O_f^{\pm}$. (This is the oracle gate $|x\rangle \mapsto (-1)^{f(x)}|x\rangle$ discussed in the Grover Recap section.)
Step 3: Apply $(H^{\otimes n})^{-1}$.
Step 4: Measure. If the result is all 0, return "all zeroes." Otherwise, return "balanced."

Let's prove that this works.

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

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

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

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

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

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

Therefore, the amplitude of $|000\dots0\rangle$ in the final state $(H^{\otimes n})^{-1}\psi$ must be zero. As a consequence, when we measure our qubits, we can never measure the all-zero state. Therefore, we will always correctly say that the list was balanced. $\square$

5 Coming soon...

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

References

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

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

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

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

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

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

Lecture 6: Boolean Fourier Analysis and Simon's Algorithm

September 28, 2015

Lecturer: John Wright *Scribe: Vikesh Siddhu*

1 Short Summary

Today we will cover the Fourier transform over $\mathbb{Z}_2^n$. Fourier transformation can be viewed as a change of basis transformation. Functions expressed in this basis can make certain patterns in a function more evident. In quantum computing, we frequently perform a Fourier transform. Given its ability to reveal some pattern in the function, it's in some sense at the heart of various quantum algorithms and we will specifically see its role in Deutsch-Josza algorithm [DJ92], Grover's algorithm [Gro96] and Simon's Problem [Sim94].

2 Fourier Transform over $\mathbb{Z}_2^n$

We will show that a Fourier Transform essentially

- Is a change of basis
- Makes it easy to find certain patterns in the function
- In quantum computing parlance it can be computed using $H^{\otimes n}$.

Consider a set of functions $\{\delta_y(x)\}_{y\in\{0,1\}^n}$ with the property that

$$\delta_y(x) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{else} \end{cases}$$

This set of functions constitute a basis for any function $g(x) : \{0,1\}^n \mapsto \mathbb{C}$ (for example $g(x) = (-1)^{f(x)}$ where $f : \{0,1\}^n \mapsto \{0,1\}$), in the sense that we can expand any $g(x)$ as follows

$$g(x) = \sum_{y\in\{0,1\}^n} g(y)\delta_y(x) \tag{1}$$

quite generally the coefficients of $\delta_y(x)$ are called expansion coefficients, in the case at hand they take the value $g(y)$ (the actual value of the function at y). The representation of the function $g(x)$ in equation (1) is called the *standard* representation.

We may represent the function $g(x)$ as a column vector with 2^n rows, such that the $(i,1)^{\text{th}}$ entry $i \in \{0,1\}^n$ is $g(i)$ i.e.

$$g \equiv \begin{bmatrix} g(0^n) \\ g(0^{n-1}1) \\ . \\ . \\ . \\ . \\ g(1^n) \end{bmatrix}$$

In this representation δ_y is the column vector with 0's everywhere except the y^{th} position where we have a 1.

By expressing a function in the standard basis we may not be able to observe certain patterns as efficiently as we would like. For e.g. in the Deutsch-Josza problem, classical algorithms sample the function in the standard basis and we need $\frac{n}{2}+1$ samplings before we can decide whether the function is balanced or all-zero.

2.1 Fourier/Parity basis

2.1.1 Definitions

Let $\sigma, X \in \mathbb{F}_2^n$ then

$$\sigma.X \equiv \sum_{i=1}^{n} \sigma_i X_i \tag{2}$$

$$= \bigoplus_{i:\sigma_i=1} X_i \tag{3}$$

where $\bigoplus$ is additional mod 2. Equation (3) allows us to interpret $\sigma.X$ as the parity of the bits in X where $\sigma_i = 1$ i.e. the parity of a subset of the entries in X. We may also consider the $\pm$ version of the parity function.

$$(-1)^{\sigma.X} = \begin{cases} 1 & \text{if } \sigma.X = 0 \\ -1 & \text{if } \sigma.X = 1 \end{cases} \tag{4}$$

$$= \prod_{i:\sigma_i=1} (-1)^{X_i} \tag{5}$$

$$\equiv \chi_\sigma(X) \tag{6}$$

Where the final equality introduced the $\chi_\sigma(X)$ function which is called the *Fourier characteristic.*

Example 2.1. $\sigma = 0^n \equiv \mathbf{0}$ *then* $\sigma.X = \mathbf{0}.X = 0$ *which gives*

$$\chi_{\mathbf{0}}(X) = (-1)^{\sigma.X} = (-1)^0 = 1$$

We define the Fourier basis as $\{\chi_\sigma\}_{\sigma \in \mathbb{F}_2^n}$ whose elements may be represented as column vectors

$$\chi_\sigma = \begin{bmatrix} \chi_\sigma(0^n) \\ \chi_\sigma(0^{n-1}1) \\ . \\ . \\ . \\ . \\ \chi_\sigma(1^n) \end{bmatrix}$$

2.1.2 Properties

In order to show $\{\chi_\sigma\}_{\sigma \in \mathbb{F}_2^n}$ is a basis we show

(i) χ_σ are orthogonal

(ii) χ_σ are 2^n in number

The second condition follows from the definition of our set of functions. The first can be restated as follows

$$\sum_{X \in \{0,1\}^n} \chi_\sigma(X)\chi_\gamma(X) = \begin{cases} 0 & \text{if } \sigma \neq \gamma \\ 2^n & \text{if } \sigma = \gamma \end{cases}$$

alternatively

$$\frac{1}{2^n} \sum_{X \in \{0,1\}^n} \chi_\sigma(X)\chi_\gamma(X) = \begin{cases} 0 & \text{if } \sigma \neq \gamma \\ 1 & \text{if } \sigma = \gamma \end{cases} \tag{7}$$

We can also write the lhs in equation (7) as $\mathbb{E}_X[\chi_\sigma(X)\chi_\gamma(X)]$ where $X \in$ uniform $\{0,1\}^n$. Finally the condition (i) can be restated as follows.

$$\mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n} [\chi_\sigma(X)\chi_\gamma(X)] = \begin{cases} 0 & \text{if } \sigma \neq \gamma \\ 1 & \text{if } \sigma = \gamma \end{cases} \tag{8}$$

Before we show equation (8) we compute $\mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n} [\chi_\sigma(X)]$,

(i) Case 1: Assume $\sigma \neq \mathbf{0}$ then

$$\begin{aligned}
\mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n} [\chi_\sigma(X)] &= \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n} [(-1)^{\sigma.X}] && (9) \\
&= \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n} [\prod_{i:\sigma_i=1} (-1)^{X_i}] && (10) \\
&= \prod_{i:\sigma_i=1} \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n} [(-1)^{X_i}] && (11) \\
&= \prod_{i:\sigma_i=1} \frac{1}{2}[(-1)^1 + 1^0] && (12) \\
&= 0 && (13)
\end{aligned}$$

The first and second equality follow from the definition, the third follows from the fact the all X_i's are from a *i.i.d* source, the fourth come from taking the expectation over uniform random $X_i \in \{0,1\}$, the final one from basic algebra.

(ii) Case 2: $\sigma = \mathbf{0}$ then it is easy to see

$$\mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[\chi_\sigma(X)] = \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[(-1)^{\mathbf{0}.X}] = \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[(-1)^0] = 1$$

hence

$$\mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[\chi_\sigma(X)] = \begin{cases} 1 & \text{if } \sigma = 0 \\ 0 & \text{else} \end{cases} \tag{14}$$

We now compute $\mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[\chi_\sigma(X)\chi_\gamma(X)]$ which is given by

$$\begin{aligned}
\mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[\chi_\sigma(X)\chi_\gamma(X)] &= \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[\prod_{i:\sigma_i=1}(-1)^{X_i}\prod_{i:\gamma_i=1}(-1)^{X_i}] &(15)\\
&= \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[\prod_{i:\sigma_i\oplus\gamma_i=1}(-1)^{X_i}] &(16)\\
&= \mathop{\mathbb{E}_X}_{X \in \text{uniform } \{0,1\}^n}[\chi_{\sigma\oplus\gamma}(X)] &(17)\\
&= \begin{cases} 1 & \text{if } \sigma \oplus \gamma = 0 \\ 0 & \text{else} \end{cases} &(18)
\end{aligned}$$

Here $\oplus$ refers to the entrywise XOR. The first equality follows from definition, the second can be obtained by a direct calculation or by studying simple examples like $\sigma = 0101\ldots$ and $\gamma = 0011\ldots$ and observing the pattern, the final equality follows from equation (14). $\sigma \oplus \gamma = 0$ implies $\sigma = \gamma$ this gives the desired result of equation (8). Consequently we have verified conditions (i) and (ii)

2.2 Change of Basis view

We can represent any function $g(X) : \{0,1\}^n \mapsto \mathbb{C}$ in the Fourier basis i.e. we may write

$$g(X) = \sum_{\gamma \in \mathbb{F}_n^2} \hat{g}(\gamma)\chi_\gamma(X) \tag{19}$$

where

Claim 2.2.

$$\hat{g}(\sigma) = \mathop{\mathbb{E}_X}_{X \in \textit{uniform } \{0,1\}^n}[\chi_\sigma(X)g(X)] \tag{20}$$

Proof.

$$\begin{aligned}
\mathbb{E}_X[\chi_\sigma g(X)] &= \mathbb{E}_X[\sum_{\gamma\in\mathbb{F}_n^2} \hat{g}(\gamma)\chi_\sigma(X)\chi_\gamma(X)] && (21)\\
&= \sum_{\gamma\in\mathbb{F}_n^2} \hat{g}(\gamma)\mathbb{E}_X[\chi_\sigma(X)\chi_\gamma(X)] && (22)\\
&= \hat{g}(\sigma) && (23)
\end{aligned}$$

where the final equality follows from equation (8). □

Let us see a few examples

Example 2.3. *What is $\hat{g}(\mathbf{0})$?*

$$\hat{g}(\mathbf{0}) = \mathbb{E}_X[\chi_{\mathbf{0}}(X)g(x)] = \mathbb{E}_X[1.g(x)] = \mathbb{E}_X[g(x)] \tag{24}$$

Example 2.4. *Let $g(X) = (-1)^{f(X)}$ where $f(X) = \sigma.X$, then what is the Fourier representation of $g(X)$.*

$$\begin{aligned}
g(X) &= (-1)^{\sigma.X} && (25)\\
&= 1.\chi_\sigma(X) + \sum_{\gamma\neq\sigma} 0.\chi_\gamma(X) && (26)
\end{aligned}$$

which gives

$$\hat{g}(\gamma) = \begin{cases} 1 & \textit{if}\ \gamma = \sigma \\ 0 & \textit{else} \end{cases} \tag{27}$$

If $f(X) \approx \sigma.X$ with probability $1-\epsilon$ where ϵ is a small positive number then

$$\hat{g}(\gamma) = \mathop{\mathbb{E}_X}_{X\in uniform\ \{0,1\}^n}[\chi_\sigma(X)(-1)^{f(x)}] \approx 1-\epsilon$$

2.3 Implementation with Hadamard

In quantum circuits the Fourier transform of a set of qubits is implemented by a Hadamard transformation on the set. Specifically

Claim 2.5. *For $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_X g(X)\,|X\rangle$ where $N \equiv 2^n$*

$$H^{\otimes n}\,|\psi\rangle = \sum_\gamma \hat{g}(\gamma)\,|\gamma\rangle \tag{28}$$

Proof.

$$H^{\otimes n}|\psi\rangle = \frac{1}{\sqrt{N}}\sum_X g(X)H^{\otimes n}|X\rangle \tag{29}$$

What is the coefficient of $|\gamma\rangle = |\gamma_1\gamma_2\dots\gamma_n\rangle$ in equation (29)? We may write

$$H^{\otimes n}|011\dots\rangle = (\frac{|0\rangle+|1\rangle}{\sqrt{2}})\otimes(\frac{|0\rangle-|1\rangle}{\sqrt{2}})\otimes(\frac{|0\rangle-|1\rangle}{\sqrt{2}})\dots$$

The coefficient of $|000\dots0\rangle$ is $\frac{(-1)^0}{2^{n/2}}$, the coefficient of $|010\dots0\rangle$ is $\frac{(-1)^{0+1+0\dots}}{2^{n/2}}$, the coefficient of $|110\dots0\rangle$ is $\frac{(-1)^{0+1+0\dots}}{2^{n/2}}$. In general the coefficient of $|\gamma_1\gamma_2\dots\gamma_n\rangle$ is accumulated as follows, for each $\gamma_i = 0$ we get $(-1)^0$, for each $\gamma_i = 1$ we get a $(-1)^{X_i}$, the general coefficient of $|\gamma_1\gamma_2\dots\gamma_n\rangle$ is a product of the accumulated coefficients which can be written as

$$= \frac{1}{2^{n/2}}(-1)^{\bigoplus_{i:\gamma_i=1}X_i} \tag{30}$$

$$= \frac{1}{2^{n/2}}(-1)^{\gamma.X} \tag{31}$$

continuing the calculation from equation (29) we may write

$$H^{\otimes n}|\psi\rangle = \frac{1}{\sqrt{N}}\sum_X g(X)\sum_\gamma \frac{1}{\sqrt{N}}(-1)^{\gamma.X}|\gamma\rangle \tag{32}$$

$$= \sum_\gamma \frac{1}{N}\sum_X g(X)(-1)^{\gamma.X}|\gamma\rangle \tag{33}$$

$$= \sum_\gamma \mathop{\mathbb{E}_X}_{X\in\text{uniform }\{0,1\}^n}[g(X)\chi_\gamma]|\gamma\rangle \tag{34}$$

$$= \sum_\gamma \hat{g}(\gamma)|\gamma\rangle \tag{35}$$

□

It is worth noting that $H.H = \mathbb{I}$ and hence $H^{\otimes n}H^{\otimes n} = \mathbb{I}_n$. An immediate consequence of this is for $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_X g(X)|X\rangle$ we my write

$$|\psi\rangle = H^{\otimes n}H^{\otimes n}|\psi\rangle \tag{36}$$

$$|\psi\rangle = H^{\otimes n}[H^{\otimes n}\frac{1}{\sqrt{N}}\sum_X g(X)|X\rangle] \tag{37}$$

$$|\psi\rangle = H^{\otimes n}\sum_\gamma \hat{g}(\gamma)|\gamma\rangle \tag{38}$$

The final equality implies that we can invert the Fourier transform by simply acting $H^{\otimes n}$.

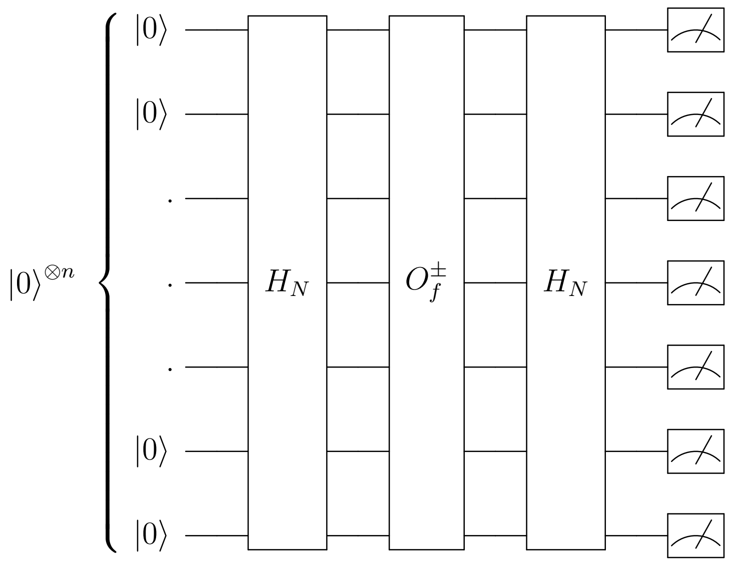
2.4 Use in Quantum Algorithms

2.4.1 Deutsch-Josza algorithm

In this problem we are given a function $f : \{0,1\}^n \mapsto \{0,1\}$ with the promise that

1. Either f is always 0
2. Or f is balanced (0 and 1 with equal probability)

The goal is to find which of the two is true. The Deutsch-Josza algorithm proceeds by applying the following circuit.



Where $H_N \equiv H^{\otimes n}$ where $N = 2^n$, the $O_f^{\pm}$ is the $\pm$ version of the oracle for the function f. After the measurement, if we get $|0\rangle^n$ we say condition (1) is true, else we say condition (2) is true.

One way to see why this procedure is essentially extracting information about the pattern in the function f as revealed by the Fourier transform is as follows. Set

$$g(X) = (-1)^{f(X)}$$

then the above circuit produces the Fourier transform of $g(X)$, observe

$$\begin{aligned}|0\rangle^{\otimes n} &\xrightarrow{H_N} \frac{1}{\sqrt{N}} \sum_{X\in\{0,1\}^n} |X\rangle \xrightarrow{O_f^{\pm}} \frac{1}{\sqrt{N}} \sum_{X\in\{0,1\}^n} (-1)^{f(X)} |X\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{X\in\{0,1\}^n} g(X) |X\rangle \xrightarrow{H_N} \sum_{\gamma} \hat{g}(\gamma) |\gamma\rangle\end{aligned}$$

hence the output is the Fourier transform of the function $g(X)$. We notice that

$$\begin{aligned}\hat{g}(\mathbf{0}) &= \mathbb{E}_X[g(X)] = \mathbb{E}_X[(-1)^{f(X)}] \\ &= \begin{cases} 1 & \text{if condition 1 is true} \\ 0 & \text{if condition 2 is true} \end{cases}\end{aligned}$$

Hence the probability of the outcome $|0\rangle^{\otimes n}$ which is $|\hat{g}(\mathbf{0})|^2$ essentially tells us which of the conditions 1 or 2 is true.

2.4.2 Grover's Algorithm

One key component of the Grover's algorithm is the diffusion operator (D) whose action on a state $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_X g(X) |X\rangle$ is given by

$$D|\psi\rangle = \frac{1}{\sqrt{N}} \sum_X (2\mu - g(X)) |X\rangle \tag{39}$$

where $\mu = \mathbb{E}_X[g(X)]$. The circuit that does the job is easy to see once we do the Fourier analysis,

$$\begin{aligned}g(X) &= \sum_\gamma \hat{g}(\gamma)\chi_\gamma(X) && (40)\\ &= \hat{g}(\mathbf{0})\chi_{\mathbf{0}}(X) + \sum_{\gamma\neq\mathbf{0}} \hat{g}(\gamma)\chi_\gamma(X) && (41)\\ &= \underset{X\in\text{uniform }\{0,1\}^n}{\mathbb{E}_X}[g(X)]\chi_{\mathbf{0}}(X) + \sum_{\gamma\neq\mathbf{0}} \hat{g}(\gamma)\chi_\gamma(X) && (42)\\ & && (43)\end{aligned}$$

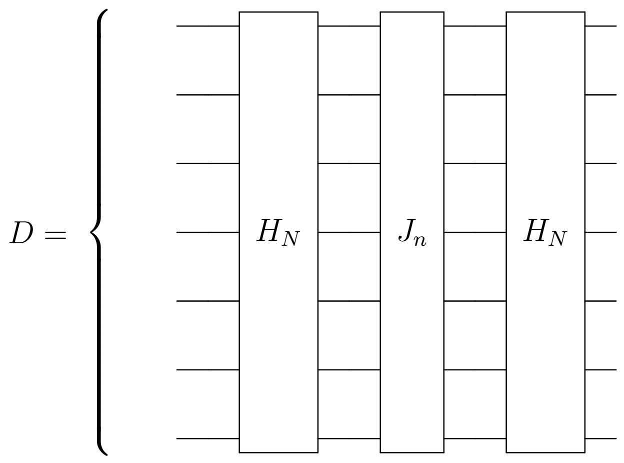
We can define a function $g'(X)$ such that

$$\begin{aligned}g'(X) &\equiv \underset{X\in\text{uniform }\{0,1\}^n}{\mathbb{E}_X}[g(X)]\chi_{\mathbf{0}}(X) - \sum_{\gamma\neq\mathbf{0}} \hat{g}(\gamma)\chi_\gamma(X) && (44)\\ &= \mu\chi_{\mathbf{0}}(X) - (g(X) - \mu\chi_{\mathbf{0}}(X)) && (45)\\ &= 2\mu\chi_{\mathbf{0}}(X) - g(X) && (46)\end{aligned}$$

We wish to replace $g(X)$ with $g'(X)$. Which can be done by taking an input and going to the Fourier basis, then putting a $-$ sign in front of all states that are not $\mathbf{0}$ and then returning from the Fourier basis. If we define

$$J_n|x\rangle = \begin{cases} |x\rangle & \text{if } x = \mathbf{0} \\ -|x\rangle & \text{else} \end{cases} \tag{47}$$

The the circuit that does the job can be written as follows.

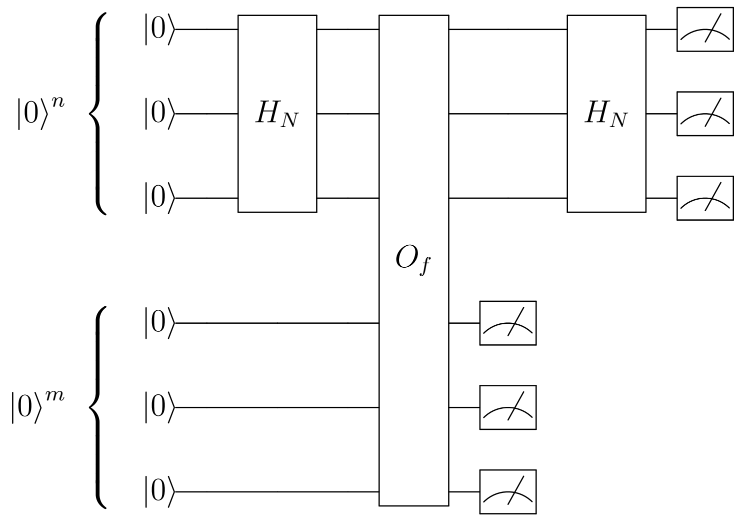


2.4.3 Simon's Problem

Simon's Problem is the precursor to the period finding algorithm which is a precursor to Shor's factorization. It utilizes quantum Fourier transform in a non-trivial way. The problem can be described as follows

- **Given**: $f : \{0,1\}^n \mapsto \{0,1\}^n$
- **Promise**:
  - (i) f is 2 to 1
  - (ii) $f(X) = f(X \otimes s)$ for some $s \in \{0,1\}^n$ and $s \neq \mathbf{0}$
- **Goal**: Find s

From last lecture, we note that classically this would take $O(\sqrt{N})$ queries and the quantum version takes $O(\log N)$ queries. We now proceed to explain the quantum circuit that does the job. Consider the circuit given below, $n+m$ wires come in (only 3 are shown schematically),



Here O_f is the oracle that acts such that

$$sO_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

The state in the circuit changes as follows

$$|0\rangle^n |0\rangle^m \xrightarrow{H_N} \frac{1}{\sqrt{N}} \sum_x |x\rangle |0\rangle^m \xrightarrow{O_f} \frac{1}{\sqrt{N}} \sum_x |x\rangle |f(x)\rangle \tag{48}$$

Once we perform the measurement on the lower m qubits we get some value y. The conditional quantum state of the first n qubits is

$$|\psi\rangle_y = \frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle) \tag{49}$$

where $y = f(x)$. Then by applying the final Hadamard gates we get

$$\begin{aligned} H_N |\psi\rangle_y &= \frac{1}{\sqrt{2N}}(\sum_{\gamma\in\{0,1\}^n} (-1)^{\gamma.x} |\gamma\rangle + \sum_{\gamma\in\{0,1\}^n} (-1)^{\gamma.(x\oplus s)} |\gamma\rangle) &(50)\\ &= \frac{1}{\sqrt{2N}} \sum_{\gamma\in\{0,1\}^n} (-1)^{\gamma.x}[1 + (-1)^{\gamma.s}] |\gamma\rangle &(51)\\ &= \frac{1}{\sqrt{2N}} \sum_{\gamma\in\{0,1\}^n} |\gamma\rangle \begin{cases} 2(-1)^{\gamma.s} & \text{if } \gamma.s = 0 \\ 0 & \text{if } \gamma.s = 1 \end{cases} &(52)\\ &= \sqrt{\frac{2}{N}} \sum_{\gamma\in\{0,1\}^n, \gamma.s=0} (-1)^{\gamma.s} |\gamma\rangle &(53) \end{aligned}$$

The final measurement gives a uniformly random γ_i such that $\gamma_i.s = 0$. By doing a large enough number of runs of the circuit i.e. by getting many different γ_i's we may recover s. But how many γ_i's do we need? $n-1$ linearly independent ones, here's why.

After $n-1$ runs we'll get γ_i's which satisfy

$$\begin{gathered} \gamma_1.s = 0 \\ \gamma_2.s = 0 \\ \dots \\ \gamma_n.s = 0 \end{gathered} \tag{54}$$

We make two claims

(i) If the γ_i's are linearly independent, then this system of linear equations fully determines s

(ii) With constant probability, running the Simon circuit will output $n-1$ linearly independent γ_i's

The above two claims imply, that we have a constant probability of success, which as we have seen, is good enough.

We prove claim (i) by noticing that each linearly independent γ_i cuts down the number of possibilities for s by half. So after $n-1$ equations, there are $2^{n-1}/2^n = 2$ possibilities for s, one of whom is $\mathbf{0}$ which is ruled out assuming that s is non-trivial, hence s is given by the other possibility.

To show (ii), let's first consider the subspace of $\mathbb{F}_2^n$

$$s^\perp := \{\gamma | \gamma . s = 0\}$$

This subspace has dimension $n-1$ since it is defined by a single linear equality, and thus the subspace contains 2^{n-1} distinct vectors.

The easiest way to prove (ii) is to show that γ_2 is linearly independent from γ_1 with high probability, then showing that γ_3 is linearly independent from γ_1,γ_2 with high probability, and so forth. So, let's suppose that $\gamma_1, \dots \gamma_k$ are linearly independent. Then what's the probability that γ_{k+1} is linearly independent from them? i.e.:

$$\begin{aligned}\mathbf{Pr}[\gamma_1, \dots \gamma_{k+1} \text{are linearly independent}] &= 1 - \mathbf{Pr}[\gamma_{k+1} \text{is in the span of} \gamma_1, \dots \gamma_k] \\ &= 1 - \frac{2^k}{2^{n-1}}\end{aligned}$$

The last line follows because there are 2^{n-1} possibilities in $s^\perp$ for γ_{k+1}, and 2^k of them are in the span of $\gamma_1, \dots, \gamma_k$. Now, the probability that $\gamma_1, \dots, \gamma_{n-1}$ are all linearly independent is what we get when we multiply the quantities we just got here for $k = 0, \dots, n-2$ (the $k = 0$ case gives the probability that γ_1 is not $\mathbf{0}$), i.e.

$$\left(1 - \frac{1}{2^{n-1}}\right)\left(1 - \frac{1}{2^{n-2}}\right) \dots \left(1 - \frac{1}{4}\right)\left(1 - \frac{1}{2}\right) = \prod_{i=1}^{n-1} \left(1 - \frac{1}{2^i}\right) \tag{55}$$

To analyze this product, we note that $(1-a)(1-b) \geq (1-(a+b))$ when $a, b \in [0,1]$. Applying this inequality to every term but the $(1-\frac{1}{2})$ term, we get

$$\prod_{i=1}^{n-1} \left(1 - \frac{1}{2^i}\right) \geq \left(1 - \left(\frac{1}{2^{n-1}} + \frac{1}{2^{n-2}} \dots \frac{1}{2^2}\right)\right) . \frac{1}{2} \geq \frac{1}{4} \tag{56}$$

Thus, Simon's algorithm succeeds with probability at least $\frac{1}{4}$, and this probability can be improved via repetition.

References

[DJ92] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 439(1907):553–558, 1992.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

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

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

Lecture 7: Quantum Fourier Transform over Z_N

September 30, 2015

Lecturer: Ryan O'Donnell *Scribe: Chris Jones*

1 Overview

Last time, we talked about two main topics:

- The quantum Fourier transform over $\mathbb{Z}_2^n$
- Solving Simon's problem with the transform over $\mathbb{Z}_2^n$

In this lecture and the next, the theory will be developed again, but over a different group, $\mathbb{Z}_N$. We'll talk about:

- The quantum Fourier transform over $\mathbb{Z}_N$ (today)
- Solving the period-finding problem with the transform over $\mathbb{Z}_N$ (next time)

As a side note, for this lecture and in general, one should not stress too much about maintaining normalized quantum states, that is, states with squared amplitudes that sum to 1. Carrying around (and computing) the normalizing factor can be a big pain, and it's understood that the "true" quantum state has normalized amplitudes. To this end, we agree that the quantum state

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

is in all respects equivalent to the normalized quantum state

$$\left(\sum_{x\in\{0,1\}^n} |\alpha_x|^2\right)^{-1} \left(\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle\right)$$

This tradition is due to physicists who tend to omit normalizing constants for convenience.

2 Defining the Fourier transform over $\mathbb{Z}_N$

2.1 Review of the transform over $\mathbb{Z}_2^n$

For now, we will forget all we know of quantum computing, except for what we learned last lecture. Let's think about the Fourier transform over $\mathbb{Z}_2^n$, and see if we can adopt the idea to $\mathbb{Z}_N$.

Consider an arbitrary function $g : \{0,1\}^n \to \mathbb{C}$. In the previous lecture, we looked at the vector space of functions from $\{0,1\}^n$ to $\mathbb{C}$, and thought of g as an element of that vector space. This vector space is 2^n-dimensional, with a "standard" basis being the indicator functions $\{\delta_y\}_{y\in\{0,1\}^n}$. With respect to this basis, we have the representation

$$g = \begin{bmatrix} g(0^n) \\ g(0^{n-1}1) \\ \vdots \\ g(1^n) \end{bmatrix}$$

We'll more often see vectors of the form

$$h = \frac{1}{\sqrt{N}} \begin{bmatrix} h(0^n) \\ \vdots \\ h(1^n) \end{bmatrix}$$

i.e. with respect to the basis $\{\sqrt{N}\delta_y\}_{y\in\{0,1\}^n}$. This is nicer because the standard dot product of such vectors is an expectation:

$$\langle g, h\rangle = \frac{1}{N}\sum_{x\in\{0,1\}^n} g(x)^* h(x) = \mathop{\mathbf{E}}_{x\sim\{0,1\}^n}[g(x)^* h(x)]$$

In particular, if $g : \{0,1\}^n \to \{-1,+1\}$, g is a unit vector.

Remember: no stress about the constant $\frac{1}{\sqrt{N}}$. It's just there to make more things unit vectors more often.

In this vector space, we found a particular orthonormal basis that made representations of f particularly nice to work with: the basis of *parity functions* $\{\chi_\gamma\}_{\gamma\in\mathbb{Z}_2^n}$, defined by

$$\chi_\gamma(x) = (-1)^{\gamma\cdot x}$$

where $\gamma \cdot x$ is the dot product in $\mathbb{Z}_2^n$. In the vector representation,

$$|\chi_\gamma\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} +1 \\ (-1)^{\gamma\cdot 0^{n-1}1} \\ \vdots \\ (-1)^{\gamma\cdot 1^n} \end{bmatrix}$$

We used the notation $\widehat{g}(\gamma)$ to denote the coefficient of χ_γ in the representation of g. All this notation gave us the final form

$$g(x) = \sum_{\gamma\in\mathbb{Z}_2^n} \widehat{g}(\gamma)\,|\chi_\gamma\rangle$$

If we endowed the domain with the group structure of $\mathbb{Z}_2^n$, or equivalently, the vector space $\mathbb{F}_2^n$, the χ_γ are nice because they are *characters* on $\mathbb{Z}_2^n$.

Definition 2.1. Let $\mathbb{C}^*$ denote the nonzero complex numbers. For a group G with operation $*$, a *character* on G is a function $\chi : G \to \mathbb{C}^*$ satisfying

$$\chi(g * h) = \chi(g) \cdot \chi(h)$$

In mathematical terms, χ is a homomorphism from G into $\mathbb{C}^*$.

We observed that a couple of properties were true of these characters:

- $\chi_{\mathbf{0}} \equiv 1$
- $\underset{x \sim \{0,1\}^n}{\mathbf{E}}[\chi_\gamma(x)] = 0$ for $\gamma \neq \mathbf{0}$.
- $\chi_\alpha(z) = \chi_z(\alpha)$

  This one is new. Though it's true, conceptually, it's generally good to segment the inputs from the index set of the Fourier coefficients.
- $\langle \chi_\gamma | g \rangle = \underset{x \sim \{0,1\}^n}{\mathbf{E}}[\chi_\gamma(x) g(x)] = \widehat{g}(\gamma)$

  This is a statement of a more general fact about orthonormal bases: if $\mathcal{F}$ is any orthonormal basis, the coefficient on $f \in \mathcal{F}$ in the representation of v with respect to $\mathcal{F}$ is given by $\langle f, v \rangle$.

As a consequence of the above properties, we had one further property:

- $\chi_\gamma(x)\chi_\sigma(x) = \chi_{\gamma+\sigma}(x)$

In summary, the Fourier transform over $\mathbb{Z}_2^n$ is defined as the unitary transformation that takes a function into its representation with respect to the χ_γ:

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} g(x) |x\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} g(0,\dots,0) \\ g(0,\dots,0,1) \\ \vdots \\ g(1,\dots,1) \end{bmatrix} \mapsto \begin{bmatrix} \widehat{g}(0,\dots,0) \\ \widehat{g}(0,\dots,0,1) \\ \vdots \\ \widehat{g}(1,\dots,1) \end{bmatrix} = \sum_{\gamma \in \mathbb{Z}_2^n} \widehat{g}(\gamma) |\gamma\rangle$$

Example 2.2. *Let*

$$g_x(y) = \begin{cases} \sqrt{N} & y = x \\ 0 & o.w. \end{cases}$$

then the vector representation of g_x is

$$\frac{1}{\sqrt{N}} \sum_{y \in \{0,1\}^n} g_x(y) |y\rangle = |x\rangle$$

and

$$\widehat{g}(\gamma) = \underset{y \sim \{0,1\}^n}{\mathbf{E}}[g_x(y)(-1)^{\gamma \cdot y}] = \frac{1}{N}\sqrt{N}(-1)^{\gamma \cdot x} = \frac{1}{\sqrt{N}}(-1)^{\gamma \cdot x}$$

$$|x\rangle = \frac{1}{\sqrt{N}} \sum_{\gamma \in \mathbb{Z}_2^n} (-1)^{\gamma \cdot x} |\chi_\gamma\rangle$$

Remark 2.3. Another way to compute the Fourier expansion of $|x\rangle$ is to note that the Fourier expansion of χ_γ is $|\gamma\rangle$, and that the Fourier transform is an involution (since H_N is its own inverse).

Representing a function with respect to this basis revealed patterns specific to the $\mathbb{Z}_2^n$ group structure. The main advantage of quantum computing, though, is that the Fourier transform over $\mathbb{Z}_2^n$ is efficiently computable on a quantum computer. The matrix that implements it, H_N, consists of exactly n gates. In comparison to the classical situation, the fastest known method to compute the Fourier expansion is the fast Walsh-Hadamard transform, which requires $O(N \log N)$ time. See [FA76] for an overview.

All this begs the questions: can we do the same for $\mathbb{Z}_N$? What are the characters on $\mathbb{Z}_N$, and do they form an orthonormal basis? If such a unitary transformation exists, can we implement it efficiently on a quantum computer? As we will now see, the answer is yes.

2.2 An Orthogonal Basis of Characters

Let's look first at the characters on $\mathbb{Z}_N$.

Theorem 2.4. *$\mathbb{Z}_N$ has exactly N characters.*

Proof. Let's try and first deduce what form the characters must have. If χ is a character, for any $x \in \mathbb{Z}_N$ we have

$$\chi(x) = \chi(x+0) = \chi(x)\chi(0)$$

If $\chi(x) \neq 0$ for some x, we can conclude $\chi(0) = 1$. We'll ignore the case where $\chi \equiv 0$, as this is the zero vector in this vector space and won't be helpful to forming an orthonormal basis. So let's conclude $\chi(0) = 1$.

We also know

$$\chi(\overbrace{x + \cdots + x}^{k \text{ times}}) = \chi(x)^k$$

In particular, if we take $k = N$, then $kx = Nx = 0$, modulo N. Combining this with the above, we have, for every $x \in \mathbb{Z}_N$,

$$\chi(x)^N = 1$$

That is, $\chi(x)$ is an N-th root of unity. We also know

$$\chi(k) = \chi(\overbrace{1 + \cdots + 1}^{k \text{ times}}) = \chi(1)^k$$

χ is completely determined by its value on 1! Let $\omega = e^{i\frac{2\pi}{N}}$ be a primitive N-th root of unity. $\chi(1)$ is an N-th root of unity, so write

$$\chi(1) = \omega^\gamma$$

for some $\gamma \in \mathbb{Z}_N$. From all of this we deduce that χ must be given by the formula

$$\chi(x) = \omega^{\gamma \cdot x}$$

where $\gamma \cdot x$ is regular integer multiplication.

For each $\gamma \in \mathbb{Z}_N$, define the function $\chi_\gamma : \mathbb{Z}_N \to \mathbb{C}$ by $\chi_N(x) = \omega^{\gamma \cdot x}$. To complete the theorem we check that every γ creates a distinct character i.e. χ_γ satisfies the homomorphism property:

$$\chi_\gamma(x+y) = \omega^{\gamma(x+y)} = \omega^{\gamma \cdot x + \gamma \cdot y} = \omega^{\gamma \cdot x}\omega^{\gamma \cdot y} = \chi_\gamma(x)\chi_\gamma(y)$$

□

With the set $\{\chi_\gamma\}_{\gamma \in \mathbb{Z}_N}$ in hand, we can check that these do indeed form an orthonormal basis. First we check that the analogous properties from the Fourier basis over $\mathbb{Z}_2^n$ carry over:

- $\chi_0 \equiv 1$

  Proof. $\chi_0(x) = \omega^{0 \cdot x} = 1$ □

- $\mathop{\mathbf{E}}_{x \sim \mathbb{Z}_N}[\chi_\gamma(x)] = 0$ for $\gamma \neq 0$.

  Proof. This is a happy fact about roots of unity. Geometrically, the powers of ω are equally spaced around the unit circle, and will cancel upon summation. Algebraically, when $\gamma \neq 0$,

  $$\frac{1}{N}\sum_{i=0}^{N-1} \omega^{\gamma x} = \frac{1}{N}\frac{\omega^{\gamma N} - 1}{\omega^{\gamma} - 1} = 0$$

  □

- $\chi_\alpha(z) = \chi_z(\alpha)$

  Proof. $\chi_\alpha(z) = \omega^{\alpha \cdot z} = \omega^{z \cdot \alpha} = \chi_z(\alpha)$ □

As before these can be used to deduce one further property,

- $\chi_\sigma(x)\chi_\gamma(x) = \chi_{\sigma+\gamma}(x)$

There's also a new, very important property that we wouldn't have noticed before, when we were only working over $\mathbb{R}$:

- $\chi_\gamma(x)^* = \chi_{-\gamma}(x) = \chi_\gamma(-x)$

Proof.

$$\begin{aligned}\chi_\gamma(x)^* = (\omega^{\gamma\cdot x})^* = \omega^{-\gamma\cdot x} &= \omega^{(-\gamma)\cdot x} = \chi_{-\gamma}(x)\\ &= \omega^{\gamma\cdot(-x)} = \chi_\gamma(-x)\end{aligned}$$

□

From these, the orthonormality of the characters falls right out:

Theorem 2.5. *The functions $\{\chi_\gamma\}_{\gamma\in\mathbb{Z}_N}$ form an orthonormal basis.*

Proof. For $\sigma, \gamma \in \mathbb{Z}_N$,

$$\begin{aligned}\langle\chi_\sigma|\chi_\gamma\rangle &= \mathop{\mathbf{E}}_{x\sim\mathbb{Z}_N}[\chi_\sigma(x)^*\chi_\gamma(x)]\\ &= \mathop{\mathbf{E}}_{x\sim\mathbb{Z}_N}[\chi_{-\sigma}(x)\chi_\gamma(x)]\\ &= \mathop{\mathbf{E}}_{x\sim\mathbb{Z}_N}[\chi_{\gamma-\sigma}(x)]\\ &= \begin{cases} 1 & \gamma-\sigma=0\\ 0 & \text{o.w.}\end{cases}\end{aligned}$$

We have N orthonormal elements in a space of dimension N, implying they form a basis for the space. □

As before, we use the notation $\widehat{g}(\gamma)$ to denote the coefficient on χ_γ in the representation of a function g with respect to this basis. The notation has almost the same form as before:

$$g(x) = \sum_{\gamma\in\mathbb{Z}_N} \widehat{g}(\gamma)\,|\chi_\gamma\rangle$$

The difference is in how we think about the domain of summation: in the first case it was $\mathbb{Z}_2^n$, and now it is $\mathbb{Z}_N$.

Also as before, and as a consequence of orthonormality, we have a method of computing the Fourier coefficients:

$$\widehat{g}(\gamma) = \langle\chi_\gamma|g\rangle = \mathop{\mathbf{E}}_{x\sim\{0,1\}^n}[\chi_\gamma(x)^*g(x)]$$

We call the unitary transformation that takes a function to its Fourier representation the Fourier transform over $\mathbb{Z}_2^n$:

$$\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n} g(x)\,|x\rangle = \frac{1}{\sqrt{N}}\begin{bmatrix} g(0,\dots,0)\\ g(0,\dots,0,1)\\ \vdots\\ g(1,\dots,1)\end{bmatrix} \mapsto \begin{bmatrix} \widehat{g}(0,\dots,0)\\ \widehat{g}(0,\dots,0,1)\\ \vdots\\ \widehat{g}(1,\dots,1)\end{bmatrix} = \sum_{\gamma\in\mathbb{Z}_N}\widehat{g}(\gamma)\,|\gamma\rangle$$

Example 2.6. *Let*

$$g_x(y) = \begin{cases} \sqrt{N} & y = x \\ 0 & o.w. \end{cases}$$

Then the vector representation of g_x is $|x\rangle$, and

$$\widehat{g}_x(\gamma) = \mathop{\mathbf{E}}_{y\sim\{0,1\}^n}[\chi_\gamma(y)^* g_x(y)] = \chi_\gamma(x)^* \qquad\qquad |x\rangle = \frac{1}{\sqrt{N}} \sum_{\gamma\in\mathbb{Z}_N} \chi_\gamma(x)^* |\chi_\gamma\rangle$$

The Fourier transform over $\mathbb{Z}_N$ will help to solve the $\mathbb{Z}_N$-analogue of Simon's problem, the period-finding problem, in the next lecture. From there it is an easy step to Shor's factorization algorithm.

One thing remains for now: efficient implementation of a circuit to compute the transform. This is what the remainder of the lecture will do.

Remark 2.7. The entire analysis above goes through without assuming that N is a power of 2. Though for the most part, we will only concern ourselves with cases where N is a power of 2.

Remark 2.8. Taking $N = 2$ gives an equivalent formulation as taking $n = 1$: the characters are $\mathbf{1}$, and $(-1)^x$.

3 Implementing the Fourier transform over $\mathbb{Z}_N$

The goal of this section is to implement the quantum Fourier transform over $\mathbb{Z}_N$ *efficiently*, that is using only poly(n) 1- or 2-qubit gates. Once we have a circuit computing the Fourier transform over $\mathbb{Z}_N$, it will be a valuable tool for use in quantum algorithms.

A life lesson to take away: if a unitary matrix has easy-to-write-down entries, it can probably be computed using poly(n) gates.

Theorem 3.1. *The quantum Fourier transform over $\mathbb{Z}_N$ can be implemented with $O(n^2)$ 1- and 2-qubit gates.*

Proof. We will build a circuit with exactly $\binom{n+1}{2}$ gates. As usual for a quantum circuit, it suffices to create a circuit that is correct on each classical input $|x\rangle$, $x \in \{0,1\}^n$; by linearity such a circuit is correct on all superpositions.

We want to implement the map

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{\gamma\in\mathbb{Z}_N} \chi_\gamma(x)^* |\gamma\rangle$$

where $\chi_\gamma(x)^* = \omega^{-\gamma\cdot x}$. Consider as example $n = 4$, $N = 2^n = 16$

$$|x\rangle \mapsto \frac{1}{4}\left(|0000\rangle + \omega^{-x}|0001\rangle + \omega^{-2x}|0010\rangle + \omega^{-3x}|0011\rangle + \cdots + \omega^{-15x}|1111\rangle\right)$$

A key realization is that the above output state is actually *unentangled*. That is, there are qubits $|\psi_0\rangle, |\psi_1\rangle, |\psi_2\rangle, |\psi_3\rangle$ such that the above state equals $|\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle \otimes |\psi_4\rangle$. In particular, it is equal to

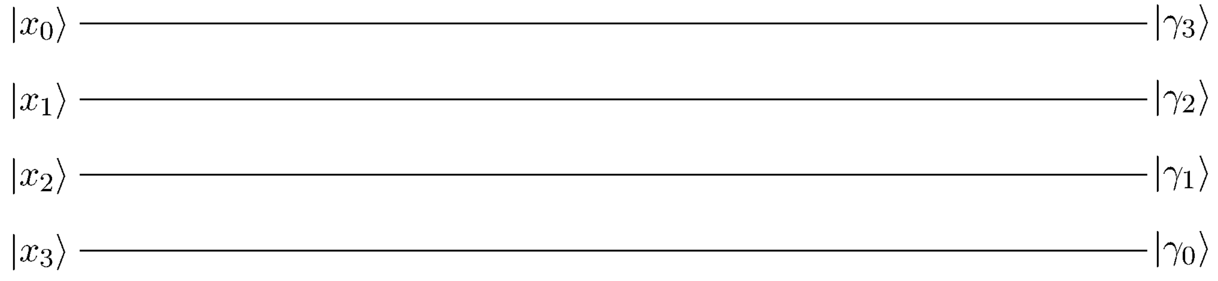
$$\left(\frac{|0\rangle + \omega^{-8x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-4x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-2x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-x}|1\rangle}{\sqrt{2}}\right)$$

In the case for general n, we want to take

$$|x\rangle \mapsto \left(\frac{|0\rangle + \omega^{-2^n x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-2^{n-1}x}|1\rangle}{\sqrt{2}}\right) \otimes \cdots \otimes \left(\frac{|0\rangle + \omega^{-x}|1\rangle}{\sqrt{2}}\right)$$

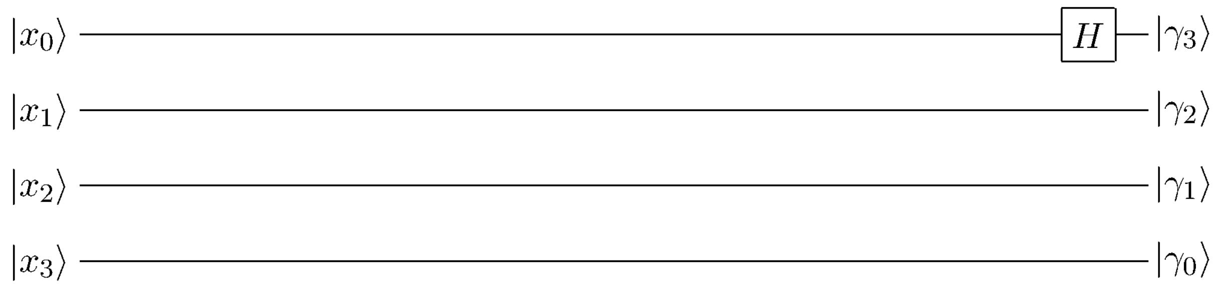
In order to do so, it suffices to perform the transformation wire-by-wire.

We return to our example. Let's write the input $|x\rangle = |x_0 x_1 \ldots x_{n-1}\rangle$ and the output $|\gamma\rangle = |\gamma_{n-1} \ldots \gamma_0\rangle$ and fill in gates as we need them:



As suggested by the notation, it's actually easier to do this transformation if we take the least significant bit of x to the most significant bit of the output. To see this, on the most significant output wire we want $\frac{1}{\sqrt{2}}(|0\rangle + \omega^{-8x}|1\rangle$. At first glance it looks like we need all of x for this. However, since $\omega^N = \omega^{16} = 1$, we only need the least significant bit $|x_0\rangle$. A similar situation occurs for other wires, as we will see. We can finish the circuit by reversing the order of the output bits, for example implemented by $\frac{n}{2}$ XOR swaps.

As we just computed, the most significant output bit will be $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{-x_0}|1\rangle)$. This is exactly the Hadamard gate applied to input $|x_0\rangle$. Here's a circuit that correctly computes the first wire:



What about the second wire? We want to perform

$$|x_1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + \omega^{-4x}|1\rangle)$$

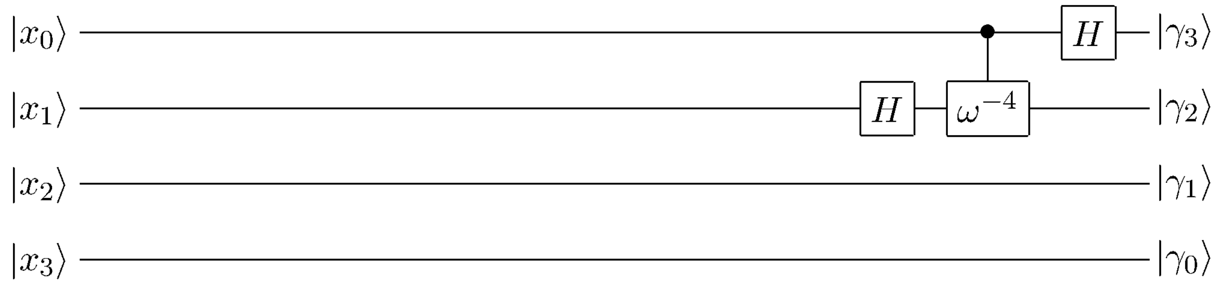
This time, ω^{-4x} depends only on the two lowest bits of x, and is equal to $\omega^{-8x_1-4x_0} = (-1)^{-x_1}\omega^{-4x_0}$. We need:

$$|x_1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{-x_1}\omega^{-4x_0}|1\rangle)$$

First, apply Hadamard to $|x_1\rangle$ to get $\frac{1}{\sqrt{2}}(|0\rangle+(-1)^{-x_0}|1\rangle)$. What we need now is a "controlled ω^{-4} gate": if $x_0 = 1$, multiply the amplitude of $|x_1\rangle$ on $|1\rangle$ by ω^{-4}, else do nothing. Under the assumption that we can implement 1- and 2-qubit gates, use a gate that implements the (unitary) matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \omega^{-4} \end{pmatrix}$$

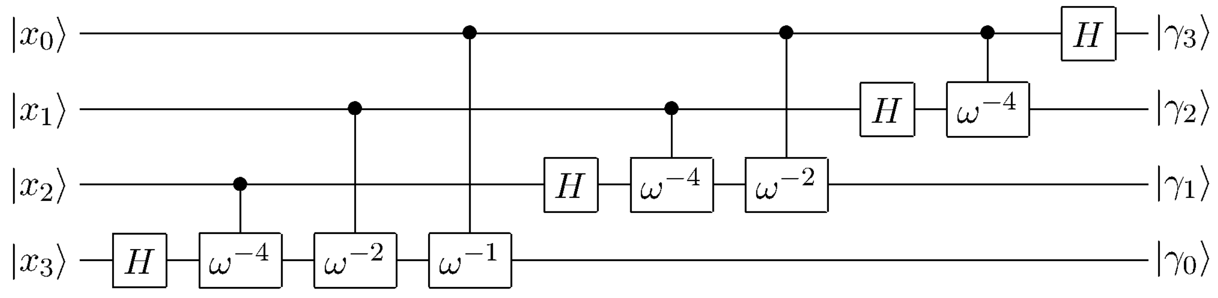
Adding these two gates to the circuit correctly computes the second wire.



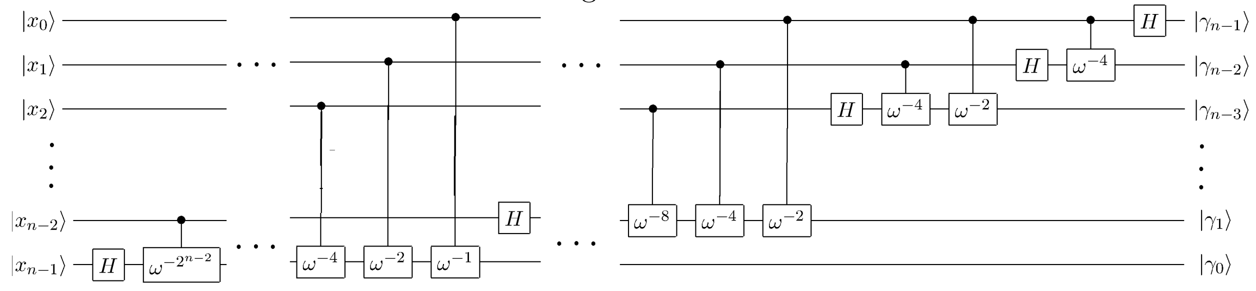
The situation for other wires, and other n, is similar. To finish off the example, we need to compute

$$|x_2\rangle \mapsto \left(\frac{|0\rangle + \omega^{-2x}|1\rangle}{\sqrt{2}}\right) = \left(\frac{|0\rangle + (-1)^{x_2}\omega^{-4x_1-2x_0}|1\rangle}{\sqrt{2}}\right)$$
$$|x_3\rangle \mapsto \left(\frac{|0\rangle + \omega^{-x}|1\rangle}{\sqrt{2}}\right) = \left(\frac{|0\rangle + (-1)^{-x_3}\omega^{-4x_2-2x_1-x_0}|1\rangle}{\sqrt{2}}\right)$$

We can build controlled ω^{-2} and ω^{-1} gates as well. Use the same paradigm: hit each wire with a Hadamard, then use the controlled ω^{-2^k} gates on the lower-order bits. The final circuit looks like this:



To generalize this to any n, transform on $|x_i\rangle$ by first applying a Hadamard gate, then applying a controlled ω^{-2^k} gate controlled by $|x_k\rangle$, for each $k < i$. This works so long as we first transform wire $|x_{n-1}\rangle$, then $|x_{n-2}\rangle$, and so on down to $|x_0\rangle$; no wire depends on wires that are below it. The circuit looks something like:



With the aforementioned reversing of the output wires, this completes the circuit to compute the Fourier transform over $\mathbb{Z}_N$. The total number of gates in this part of the circuit $1 + 2 + \cdots + n = \binom{n+1}{2} = O(n^2)$. Including the $O(n)$ for swapping gives $O(n^2)$ size overall.

□

Remark 3.2. The uncontrolled, 1-qubit version of the controlled ω^{-2^k} is called the "phase shift gate", with matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

This gate does not change the probability of outcome $|0\rangle$ or $|1\rangle$ for this qubit.

Remark 3.3. One can compute the transform to Fourier transform over $\mathbb{Z}_N$ to accuracy ϵ using only $O(n \log \frac{n}{\epsilon})$ gates. The idea is simple enough: phase shifts of very small amounts will change the overall quantum state very little, so if we allow ourselves a little error we can afford to skip a few. A more exact analysis can be found in [HH00].

References

[FA76] B.J. Fino and V.R. Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, 25(11):1142–1146, 1976.

[HH00] L. Hales and S. Hallgren. An improved quantum fourier transform algorithm and applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 515–525, 2000.

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

Lecture 8: Period Finding: Simon's Problem over $\mathbb{Z}_N$

October 5, 2015

Lecturer: John Wright *Scribe: Nicolas Resch*

1 Problem

As mentioned previously, period finding is a rephrasing of Simon's algorithm, but instead of using the Fourier transform over $\mathbb{Z}_2^n$, we will use the Fourier transform over $\mathbb{Z}_N$. Moreover, an efficient quantum algorithm for period finding will "essentially" give us Shor's algorithm [Sho97] for efficient factorization – the details will be presented next lecture.

Definition 1.1 (Period-Finding Problem)**.** Given is some $f : \mathbb{Z}_N \to$ "colors" (i.e. the image of f is some "unstructured" set). Pictorially, f can be imagined as an array:

$$\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \text{R} & \text{G} & \text{B} & \text{Y} & \text{R} & \text{G} & \text{B} & \text{Y} & \cdots \\ \hline \end{array}}_{\text{length } N}$$

As usual, we have oracle access to f, and we denote the oracle by O_f. For this problem, it is most convenient to work with the oracle which behaves as follows:

$$O_f(|x\rangle|b\rangle) = |x\rangle|b \oplus f(x)\rangle,$$

where b is an m-qubit string. We have the promise that f is *periodic*; namely for some $s \in \mathbb{Z}_N \setminus \{0\}$, $f(x) = f(x+s)$ for all $x \in \mathbb{Z}_N$. Otherwise, all of f's values are assumed to be distinct: that is, we never have $f(x) = f(y)$ if x and y don't differ by a multiple of s. The goal of the problem is to find s.

Remark 1.2. Classically, we can actually solve this problem very efficiently. Note that the condition on s implies that s divides N. Assuming $N = 2^n$, then s must lie in the set $\{1, 2, 4, \ldots, N\}$. So we obtain an efficient classical algorithm by simply testing if $s = 1$ is f's period, then if $s = 2$ is f's period, etc. This requires us to test $n = \log N$ values of s, so the query complexity, and run-time, is $O(n)$.

So, why do we care about solving this quantumly? Shor's algorithm [Sho97] for factoring will actually look at a variant where f is "almost-periodic". So we will not necessarily know that s divides N. However, the quantum algorithm we develop today will generalize to account for this case.

2 The Algorithm

Here is the quantum algorithm that we will use to solve this problem:

- Prepare the state $\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}_N}|x\rangle$, i.e. the uniform superposition of all the kets $|x\rangle$.
- Attach the state $|0^m\rangle$, thereby obtaining $\left(\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}_N}|x\rangle\right)\otimes|0^m\rangle = \frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}_N}|x\rangle|0^m\rangle$.
- Query the oracle O_f on the current input. The state of the system will now be

$$\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}_N}|x\rangle|f(x)\rangle. \tag{1}$$

- Measure the color register, i.e. the registers corresponding to the $|0^m \oplus f(x)\rangle$ part of the output of O_f.
- Apply $\mathcal{F}_N$ to the remaining n qubits.
- Measure the remaining n qubits.
- Then, we do a little bit of "classical" computation with the output, which will be made clear later on.

To aid with the analysis of the algorithm, we will use the following notation:

Notation 2.1. *For each color c, define $f_c : \mathbb{Z}_N \to \{0,1\}$ by*

$$f_c(x) = \begin{cases} 1 & \text{if} f(x) = c, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, f_c is the indicator function for the event that $f(x) = c$.

Thus, (1) can be rewritten as

$$\sum_{\text{colors } c}\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}_N}(f_c(x)|x\rangle)\otimes|f(x)\rangle. \tag{2}$$

Indeed, if $f_c(x) = 0$ then the term $|x\rangle|f(x)\rangle$ disappears from the sum, so we only count $|x\rangle|f(x)\rangle$ for the one color c such that $f(x) = c$. Moreover, we comment at that in the summation (2), the number of nonzero terms of the form $(f_c(x)|x\rangle)$ is precisely $\frac{N}{s}$, which is the number of distinct values $x \in \mathbb{Z}_N$ that map to the color c under f.

3 Analysis

The first question that we should ask is: what do we get when we perform the color measurement? That is, with what probability do we measure a fixed color c? Using (2), we see that this probability is precisely

$$\begin{aligned}\sum_{x\in\mathbb{Z}_N}\left(\frac{1}{\sqrt{N}}f_c(x)\right)^2 &= \frac{1}{N}\sum_{x\in\mathbb{Z}_N} f_c(x)^2 = \frac{1}{N}\sum_{x\in\mathbb{Z}_N} f_c(x)\\ &= \mathop{\mathbf{E}}_{x\in_R\mathbb{Z}_N}[f_c(x)] = \mathop{\mathbf{Pr}}_{x\in_R\mathbb{Z}_N}[f(x)=c]\\ &= \text{fraction of } f \text{ outputs which have color } c\\ &= \frac{1}{s}.\end{aligned}$$

In the above computations, we write $x \in_R \mathbb{Z}_N$ to denote the distribution in which x is sampled from $\mathbb{Z}_N$ uniformly at random. We used the fact that $f_c(x)^2 = f_c(x)$ for all x since f_c is $\{0,1\}$-valued. We also observed that $\mathbf{E}_{x\in_R\mathbb{Z}_N}[f_c(x)] = \mathbf{Pr}_{x\in_R\mathbb{Z}_N}[f_c(x)=1]$ since $f_c(x)$ for random x is a $\{0,1\}$-valued random variable, and then the simple fact that $f_c(x) = 1 \iff f(x) = c$ by the definition of f_c to conclude $\mathbf{Pr}_{x\in_R\mathbb{Z}_N}[f_c(x)=1] = \mathbf{Pr}_{x\in_R\mathbb{Z}_N}[f(x)=c]$.

Thus, we obtain a *uniformly random* color as our answer!

Given that we have observed a color c, what does the state (2) collapse to? Well, it's the substate that is consistent with the color c. This is the state

$$\left(\sum_{x\in\mathbb{Z}_N} f_c(x)|x\rangle\right)\otimes|c\rangle \text{ normalized.}$$

The normalizing factor is $\sqrt{\frac{s}{N}}$, so the state is

$$\sqrt{\frac{s}{N}}\left(\sum_{x\in\mathbb{Z}_N} f_c(x)|x\rangle\right)\otimes|c\rangle$$

It will be most convenient for us to rewrite this state as

$$\frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}_N}\sqrt{s}f_c(x)|x\rangle. \tag{3}$$

At this point, we will use the "powerful trick" that always seems to work when we analyze quantum algorithms. This trick is, of course, the quantum Fourier transform. More precisely, we will:

$$\boxed{\text{Apply the Quantum Fourier Tranform over } \mathbb{Z}_N \text{ and measure.}}$$

Let us briefly recall what generally happens when we use this trick. If we have a function $g : G \to \mathbb{C}$ such that $\mathbf{E}_{x\in_R G}[|g(x)|^2] = 1$, where G is either $\mathbb{Z}_2^n$ or $\mathbb{Z}_N$, then the following is a valid quantum state:

$$\frac{1}{\sqrt{N}}\sum_{x\in G} g(x)|x\rangle.$$

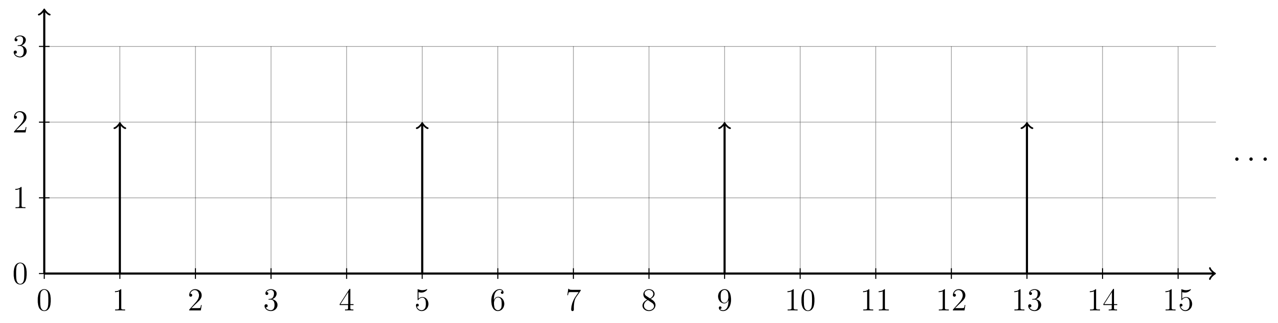


Figure 1: The function $g = 2 \cdot 1_{\{1,5,9,13,\ldots,\}}$

If we apply the Fourier transform, which corresponds to applying H_N if $G = \mathbb{Z}_2^n$ and $\mathcal{F}_N$ if $G = \mathbb{Z}_N$, then the state we obtain is

$$\sum_{\gamma \in G} \hat{g}(\gamma)|\gamma\rangle.$$

Thus, when we measure, we observe $\gamma \in G$ with probability $|\hat{g}(\gamma)|^2$.

Remark 3.1. This procedure is called *spectral sampling*. The set $\{\hat{g}(\gamma)\}_{\gamma \in G}$ is called the *Fourier spectrum* of g. This is *almost* always how exponential speed-ups are obtained in quantum computing.

In our case, recalling (3), we should put $g = \sqrt{s} \cdot f_c$, as then $\frac{1}{\sqrt{N}} \sum_{x \in \mathbb{Z}_N} g(x)|x\rangle$ is the quantum state after measuring the color register.

Remark 3.2. For Simon's algorithm, if we define

$$1_y(x) = \begin{cases} 1 & x = y, \\ 0 & x \neq y \end{cases},$$

we had $g = \sqrt{2}(1_y + 1_{y+s})$, where $f^{-1}(c) = \{y, y+s\}$ for some color c.

Before continuing, let's define the following notation which we will use throughout the analysis:

Notation 3.3. *For $S \subseteq \mathbb{Z}_N$, we define $1_S : \mathbb{Z}_N \to \{0,1\}$ by*

$$1_S(x) = \begin{cases} 1 & \textit{if } x \in S, \\ 0 & \textit{if } x \notin S. \end{cases}$$

Example 3.4. *Say $s = 4$, c is Green, and f is Green on $1, 5, 9, 13, \ldots$. Then $g = \sqrt{s} \cdot f_c = 2 \cdot 1_{\{1,5,9,13,\ldots\}}$. Figure 1 demonstrates what this function looks like.*

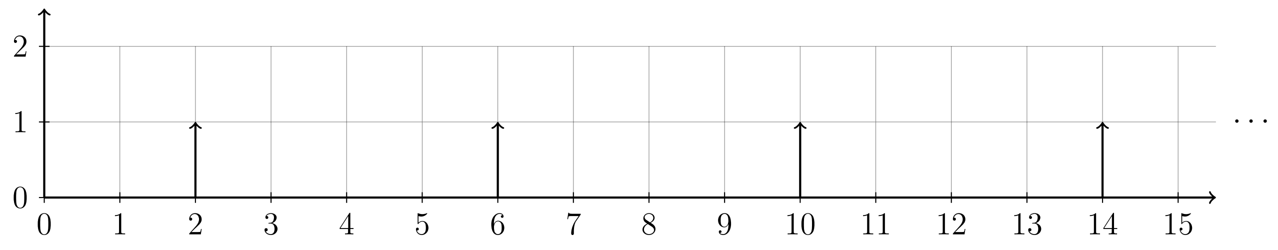


Figure 2: The function $f_{\text{Green}} = 1_{\{2,6,10,\ldots\}}$

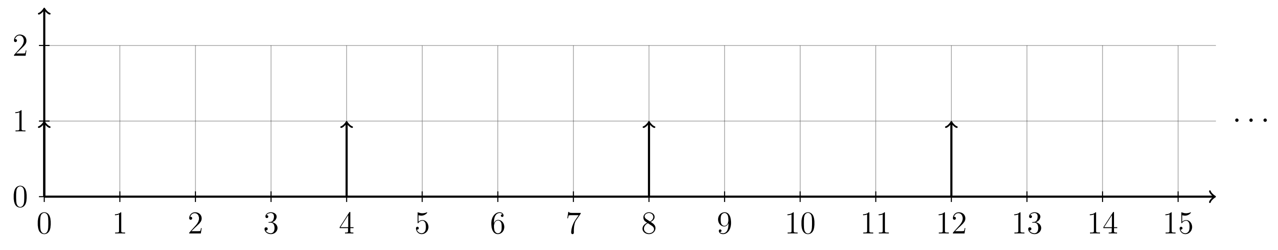


Figure 3: The function $f_{\text{Red}} = 1_{\{0,4,8,\ldots\}}$

Our algorithm outputs $\gamma \in \mathbb{Z}_N$ with probability

$$|\hat{g}(\gamma)|^2 = s \cdot |f_c(\gamma)|^2.$$

So, the question now is: *what are these Fourier coefficients?* We have a periodic spike function. It turns out that the Fourier transform of such a function is also a periodic spike function.

To simplify our analysis, we would like to show that $|f_c(\gamma)|^2$ is independent of c. That way, it will suffice to compute $|f_c(\gamma)|^2$ for one "nice" choice of c. Hence, we will prove the following claim:

Claim 3.5. *Let $g : \mathbb{Z}_N \to \mathbb{C}$ and let $t \in \mathbb{Z}_N$. Define $g^{+t} : \mathbb{Z}_N \to \mathbb{C}$ by*

$$g^{+t}(x) = g(x+t).$$

Then g and g^{+t} have "essentially the same" Fourier coefficients. That is, they differ by a multiplicative factor of magnitude 1, so they have the same magnitude.

Why is this helpful? Well, say $f_{\text{Green}} = 1_{\{2,6,10,\ldots,\}}$ and $f_{\text{Red}} = 1_{\{0,4,8,\ldots,\}}$. Then, these two functions are shifts of each other, as $f_{\text{Green}} = f_{\text{Red}}^{+2}$. It therefore suffices to study f_c for a single value of c, as desired. See figures 2 and 3.

Proof. Let $\omega = e^{\frac{2\pi i}{N}}$, so that $\chi_\gamma(x) = \omega^{\gamma \cdot x}$. We compute:

$$\begin{aligned} \widehat{g^{+t}}(\gamma) &= \mathop{\mathbf{E}}_{x \in_R \mathbb{Z}_N} [g^{+t}(x)\chi_\gamma(x)^*] \\ &= \mathop{\mathbf{E}}_{x \in_R \mathbb{Z}_N} [g(x+t)\chi_\gamma(x)^*] \qquad (*) \end{aligned}$$

At this point, we make the change of variables $y = x + t$. For fixed t, if x is selected from $\mathbb{Z}_N$ uniformly at random, so is y. Thus,

$$\begin{aligned}(*) &= \mathop{\mathbf{E}}_{y \in_R \mathbb{Z}_N}[g(y)\chi_\gamma(y-t)^*] \\ &= \mathop{\mathbf{E}}_{x \in_R \mathbb{Z}_N}[g(y)\chi_\gamma(y)^*\chi_\gamma(-t)^*] \\ &= \chi_\gamma(-t)^* \mathop{\mathbf{E}}_{y \in \mathbb{Z}_N}[g(y)\chi_\gamma(y)^*] \\ &= \omega^{\gamma t}\hat{g}(\gamma).\end{aligned}$$

Recalling that $\omega^{\gamma t}$ is an N-th root of 1, we conclude that it is a complex number of magnitude 1. In the above computations, we used the important fact that $\chi_\gamma(x+y) = \chi_\gamma(x)\chi_\gamma(y)$ for all $x, y \in \mathbb{Z}_N$, as well as the observation that $\chi_\gamma(-t)^*$ does not depend on the randomly selected $y \in_R \mathbb{Z}_N$. □

This immediately yields the following corollary:

Corollary 3.6. $|\widehat{g^{+t}}(\gamma)|^2 = |\omega^{\gamma t}|^2|\hat{g}(\gamma)|^2 = |\hat{g}(\gamma)|^2$.

Thus, the probability of sampling $\gamma \in \mathbb{Z}_N$ is independent of t. In our case, this means that when we do the spectral sampling at the end of the algorithm, it does not matter what color we measured earlier. It is therefore no loss of generality to assume that, if c is the color we sampled, $f(0) = c$, from whence it follows that $f_c = 1_{\{0,s,2s,3s,\ldots\}}$.

What is so special about the set $\{0, s, 2s, \ldots\}$? It's precisely the subgroup of $\mathbb{Z}_N$ generated by the number s!

Remark 3.7. For Simon's algorithm, we would study $1_{\{0,s\}}$, as $\{0, s\}$ is the subgroup of $\mathbb{Z}_2^n$ generated by s.

We are now prepared to analyze the Fourier coefficients of g.

Proposition 3.8. *Let $H = \{0, s, 2s, \ldots\} \subseteq \mathbb{Z}_N$ and let $h = 1_H$. Then*

$$\hat{h}(\gamma) = \begin{cases} \frac{1}{s} & \textit{if } \gamma \in \{0, \frac{N}{s}, \frac{2N}{s}, \frac{3N}{s}, \ldots\}, \\ 0 & \textit{otherwise.} \end{cases}$$

Remark 3.9. Observe that $|\{0, \frac{N}{s}, \frac{2N}{s}, \frac{3N}{s}, \ldots\}| = s$. To see this, recall that $s \cdot \frac{N}{s} = N = 0 \mod N$, so the size of the set is at most s because $\frac{aN}{s} = \frac{(a+s)N}{s}$ for all values of a. But if $r < s$ then $r \cdot \frac{N}{s} \neq 0 \mod N$, so if $a, b < s$ with $a < b$ then $\frac{aN}{s} \neq \frac{bN}{s}$ as otherwise $\frac{(b-a)N}{s} = 0 \mod N$ even though $b - a \leq b < s$. Thus, we conclude that there are precisely s values of γ for which $\hat{h}(\gamma) \neq 0$.

Assuming the proposition, what do we conclude? Well, the state changes as follows. Recalling that the state prior to applying the $\mathcal{F}_N$ gate is (3), we have

$$\frac{1}{\sqrt{N}} \sum_{x \in \mathbb{Z}_N} \sqrt{s} \cdot h(x)|x\rangle \xrightarrow{\mathcal{F}_N} \sum_{\gamma \in \mathbb{Z}_N} \sqrt{s} \cdot \hat{h}(\gamma)|\gamma\rangle$$
$$\xrightarrow{\text{measure}} \gamma \text{ with probability } s \cdot |\hat{h}(\gamma)|^2.$$

Since $\hat{h}(\gamma) = \frac{1}{s}$ iff $\gamma \in H$, we measure $\gamma \notin H$ with probability 0 and $\gamma \in H$ with probability $s \cdot \left|\frac{1}{s}\right|^2 = \frac{1}{s}$. That is, we sample a uniformly random $\gamma \in H$. Recalling that $H = \{0, \frac{N}{s}, \frac{2N}{s}, \ldots\}$, any $\gamma \in H$ satisfies $\gamma s = 0 \mod N$. Thus, we conclude that we sample a uniformly random γ such that $\gamma s = 0 \mod N$. This is just like what happened in Simon's algorithm! There, we sampled a uniformly random $\gamma \in \mathbb{Z}_2^n$ such that $\gamma \cdot s = 0$, so the only difference is that now we consider multiplication modulo N instead of the dot product of two length n strings modulo 2.

Now, we will prove Proposition 3.8.

Proof. We compute

$$\hat{h}(\gamma) = \mathop{\mathbf{E}}_{x \in_R \mathbb{Z}_N}[h(x) \cdot \chi_\gamma(x)^*] = \frac{1}{s} \mathop{\mathbf{E}}_{x \in_R H}[\chi_\gamma(x)^*] = (\dagger).$$

The second equality follows from the fact that $h(x)$ is only ever nonzero on the set H which has size s. We consider two cases:

Case 1. Suppose $\gamma \in \{0, \frac{N}{s}, \frac{2N}{s}, \ldots\}$. Then

$$\chi_\gamma(x)^* = \omega^{-\gamma \cdot x}$$

Now, recall that x is a multiple of s, since s is assumed to be sampled from H. Similarly, if γ is in $\{0, \frac{N}{s}, \frac{2N}{s}, \ldots\}$, γ is a multiple of $\frac{N}{s}$. Thus,

$$\chi_\gamma(x)^* = \omega^{-(\text{multiple of } \frac{N}{s}) \cdot (\text{multiple of } s)} = \omega^{-(\text{multiple of } N)} = 1,$$

using that ω is an N-th root of unity. Thus, $\chi_\gamma(x)^* = 1$ for all $x \in H$, so

$$(\dagger) = \frac{1}{s} \cdot \mathop{\mathbf{E}}_{x \in H}[1] = \frac{1}{s}.$$

Case 2. We could, using some elementary arithmetic, directly compute $\mathbf{E}_{x \in_R H}[\chi_\gamma(x)^*]$ to show that it is 0. However, we'll be a bit more slick. We've already shown that our algorithm outputs $\gamma \in \{0, \frac{N}{s}, \frac{2N}{s}\}$ with probability $s \cdot \left(\frac{1}{s}\right)^2 = \frac{1}{s}$. Since $|\{0, \frac{N}{s}, \frac{2N}{s}, \ldots, \}| = s$, there is no probability left to give to the $\bar{h}(\gamma)$'s! That is,

$$1 = \sum_{\gamma \in \mathbb{Z}_N} s|\hat{h}(\gamma)|^2 = \sum_{\gamma \in H} s|\hat{h}(\gamma)|^2 + \sum_{\gamma \notin H} s|\hat{h}(\gamma)|^2 = 1 + \sum_{\gamma \notin H} s|\hat{h}(\gamma)|^2$$

so

$$\sum_{\gamma \notin H} s|\hat{h}(\gamma)|^2 = 0,$$

implying $\hat{h}(\gamma) = 0$ for all $\gamma \notin H$.

□

4 Summary

Let's reviewed what we've accomplished. We know that f is promised to be of the form

$$\underbrace{\begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \mathrm{R} & \mathrm{G} & \mathrm{B} & \mathrm{Y} & \mathrm{R} & \mathrm{G} & \mathrm{B} & \mathrm{Y} & \cdots \\ \hline \end{array}}_{\text{length } N}$$

That is, f is s-periodic. We have shown that, with one query to O_f and $O(n^2)$ gates along with a couple measurements, we get a uniformly random $\gamma \in \mathbb{Z}_N$ such that $\gamma \cdot s = 0 \mod N$ (or, equivalently, such that $\gamma \in \{0, \frac{N}{s}, \frac{2N}{s}, \ldots\}$).

Well, what should we do? As is usually the case, we'll want to repeat the algorithm some number of times. But how many times show we repeat the algorithm to find s?

First of all, we observe that to find s, it actually suffices to find $\frac{N}{s}$. After that, we can divide through by N and take the reciprocal to compute s.

So, how do we find $\frac{N}{s}$? For the time being, forget that N and s are powers of 2. Let $m = \frac{N}{s}$, so our algorithm samples a random integer multiple of m. Suppose we have two random samples, which we can write am and bm. Note that

$$\gcd(am, bm) = \gcd(a, b)m.$$

So, if $\gcd(a, b) = 1$, by taking the gcd of the two outputs am and bm, we will have found m! Thus, the question becomes the following: If a and b are sampled from $\{0, 1, \ldots, s-1\}$ uniformly at random and independently, what's the probability that $\gcd(a, b) = 1$?

Claim 4.1. *If a and b are sampled as above,* $\mathbf{Pr}[\gcd(a, b) = 1] \geq \Omega(1)$.

Proof. First condition on the event that a and b are not equal to zero. This event occurs with large probability: $\left(\frac{s-1}{s}\right)^2$ which is at least 1/4. Fix a prime p. Observe that

$$\begin{aligned} \mathbf{Pr}[p \text{ divides } a \text{ and } b] &= \mathbf{Pr}[p \text{ divides } a] \cdot \mathbf{Pr}[p \text{ divides } b] && \text{since } a \text{ and } b \text{ are independent} \\ &= \mathbf{Pr}[p \text{ divides } a]^2 && \text{since } a \text{ and } b \text{ are identically distributed} \end{aligned}$$

Note that at most a $1/p$ fraction of the elements in the set $\{1, 2, \ldots, s-1\}$ are multiples of p. Since we are conditioning on the event that $a \neq 0$, we conclude that $\mathbf{Pr}[p \text{ divides } a] \leq 1/p$. Therefore

$$\mathbf{Pr}[p \text{ divides } a \text{ and } b] \leq \frac{1}{p^2}.$$

Thus,

$$\begin{aligned} \mathbf{Pr}[\gcd(a, b) > 1] &= \mathbf{Pr}[a \text{ and } b \text{ share a prime factor}] \\ &\leq \sum_{\text{primes } p} \frac{1}{p^2} && \text{Union Bound} \\ &\leq \sum_{n \geq 2} \frac{1}{n^2} = \frac{\pi^2}{6} - 1 \approx 0.6, \end{aligned}$$

where we have used the famous number-theoretic fact that

$$\sum_{n\geq 1} \frac{1}{n^2} = \zeta(2) = \frac{\pi^2}{6},$$

where ζ denotes the Riemann zeta function. □

Remark 4.2. Computing $\sum_{\text{primes } p} \frac{1}{p^2}$ exactly is an open problem.

References

[Sho97] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997.

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

Lecture 9: Shor's Algorithm

October 7, 2015

Lecturer: Ryan O'Donnell *Scribe: Sidhanth Mohanty*

1 Overview

Let us recall the period finding problem that was set up as a function $f : \mathbb{Z}_N \to$ colors, with the promise that f was periodic. That is, there exists some s for which $f(x+s) = f(x)$ (note that addition is done in $\mathbb{Z}_N$) for all $x \in \mathbb{Z}_N$ and that colors in a block of size s were pairwise distinct.

This setup implies that $s \mid N$, so that greatly narrows down what s could be. This problem is not hard to do classically, but can be done better with a quantum computer. Slight variants of this problem can be solved with a quantum computer too, and we shall explore such a variant in this lecture.

Here is a sketch of the period finding algorithm that was covered during last lecture (see the period finding lecture for a deeper treatment).

- We begin by preparing our favorite quantum state

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

  We then tensor this state with $|0^n\rangle$.

- We pass the state (after tensoring) through an oracle for f and obtain the state

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \, |f(x)\rangle$$

- We then measure the qubits representing $|f(x)\rangle$ and obtain a random color c. This causes the overall state to collapse a superposition of states where $|x\rangle$ is in the preimage of c.

$$\sqrt{\frac{s}{N}} \sum_{k=0}^{\frac{N}{s}-1} |x_0 + ks\rangle \, |c\rangle$$

  The coefficients can be thought of as $f_c(x)\sqrt{\frac{s}{N}}$ where $f_c(x) = 1$ when $f(x) = c$ and 0 otherwise.

- We then apply the Quantum Fourier Transform on this state to obtain a quantum state where the coefficients are $\hat{f}_c(\gamma)\sqrt{\frac{s}{N}}$ where γ is a multiple of $\frac{N}{s}$. From the previous lecture, we know that $\hat{f}_c$ has a period of $\frac{N}{s}$ and hence γ for which $\hat{f}_c(\gamma)$ is nonzero is a multiple of $\frac{N}{s}$.
- Measuring k gives us a random γ in $\left\{0, \frac{N}{s}, \frac{2N}{s}, \cdots, \frac{(S-1)N}{s}\right\}$.
- Take a constant number of samples and take the GCD of all these samples. With high probability, you get $\frac{N}{s}$, from which we can retrieve s.

2 Review of complexity of algorithms involving numbers

In general, an efficient algorithm dealing with numbers must run in time polynomial in n where n is the number of bits used to represent the number (numbers are of order 2^n)

To refresh, let's go over things we can do in polynomial time with integers.

Say P, Q and R are n bit integers.

- $P \cdot Q$ can be computed in polynomial time.
- $\lfloor \frac{P}{Q} \rfloor$ and $P \mod Q$ can be computed in polynomial time.
- P^Q is massive, and writing it out itself would cause the time to go exponential.
- But $P^Q \mod R$ can be done polynomially by computing $p, p^2, p^4, p^8, \ldots, p^{2^n}$ for $2^n \geq Q$.
- The GCD of P and Q can be done polynomially with Euclid's algorithm.
- Now for something more interesting: checking if P is prime. It can be done in $\tilde{O}(n^2)$ using a randomized algorithm (Miller-Rabin) and in $\tilde{O}(n^6)$ using a deterministic algorithm (AKS).
- Now, why not try to factor P? And suddenly we are stuck if we try to approach the problem classically. The best known deterministic algorithm runs in $2^{\tilde{O}(n^{\frac{1}{3}})}$

3 Shor's Algorithm

There are three steps to understanding Shor's algorithm [Sho97].

1. Factoring $\leq$ Order-finding: Factoring reduces to order-finding, which means that if we have an algorithm to solve order-finding efficiently, we can efficiently solve the factoring problem as well by a polynomial time reduction from factoring to order-finding. Note that this reduction can be made classically.

2. Order-finding $\approx$ Period-finding: Vaguely, order-finding is approximately the same problem as period finding for a quantum computer. This will be expanded in more detail this lecture.

3. Identifying simple fractions: This part is necessary in the order-finding algorithm that is crucial for Shor's algorithm and can be done classically as well.

   The second step is the key step in Shor's algorithm.

3.1 What is order finding?

We are given A, M (n-bit numbers) along with a promise that A and M are coprime. The objective is to find the least $s \geq 1$ ($s \leq M$) such that $A^s \equiv 1 \mod M$. s is called the order of A.

Note that s divides $\varphi(M)$, where φ is the Euler Totient function that gives us the number of elements less than M that are coprime with M. As another remark, $\varphi(M)$ is the order of the multiplicative group $\mathbb{Z}_m^*$ and s divides $\varphi(M)$.

3.2 Proof that Factoring $\leq$ Order-finding

In this section, we shall assume that we have an efficient order-finding algorithm.

Say M is a number that we want to factor. The key to solving the factoring problem using order-finding lies in finding a nontrivial square root of $1 \mod M$, that is, a number r with $r^2 \equiv 1 \mod M$ and $r \not\equiv \pm 1 \mod M$. Then we know that $(r+1)(r-1) \equiv 0 \mod M$ and both $r+1$ and $r-1$ are nonzero $\mod M$ and are factors of some multiple of M. (A nontrivial square root may not always exist, for instance, when M is a power of an odd prime, but we'll see how to handle that case)

Computing the GCD of M and $r-1$ would give us a nontrivial factor of M, called c. We can divide out c from M, check if c or $\frac{M}{c}$ are prime and for each of c and $\frac{M}{c}$, if they are not prime, we recursively factor them, and if they are prime, we store them as prime factors and wait until the rest of the terms are factored. We then return the set of all prime factors. (Recall that we can efficiently test primality.)

Note that the number of recursive calls made is logarithmic in M because there are at most $\log M$ prime factors of M and each recursive call increases how many numbers we have not split by 1. Hence, after $\log M - 1$ recursive calls, there are about $\log M$ numbers that we have not split. Splitting further would force the number of prime factors to exceed $\log M$, which is not possible.

Now, one might ask how one would go about finding a nontrivial square root of $1 \mod M$. We take a random $A \in \mathbb{Z}_M^*$, and find its order s.

Perhaps, we get lucky and have s be even, so we could set $r \equiv A^{\frac{s}{2}} \mod M$ (then $r^2 \equiv A^s \mod M \equiv 1 \mod M$). Maybe we could push our luck a bit more and hope $r \not\equiv -1 \mod M$. But turns out, we can actually make these two lucky things happen, thanks to a number theory lemma!

Lemma 3.1. *Suppose M has ≥ 2 distinct odd prime factors. Then if we pick $A \in \mathbb{Z}_M^*$ uniformly at random, the probability that the order s of A is even and that $A^{\frac{s}{2}} \cong 1$ is at least $\frac{1}{2}$.*

Proof. See Lemma 9.2 and Lemma 9.3 of Vazirani's course notes [Vaz04] □

One can pick a uniformly random $A \in \mathbb{Z}_M^*$ by randomly picking elements A from $\mathbb{Z}_M$ and computing $\text{GCD}(M, A)$ until it we find A for which the GCD is 1. And with at least $\frac{1}{2}$ chance, our 'lucky conditions' are satisfied. Repeatedly picking A boosts this probability further. If we cannot find such a number A after picking randomly many times, then it means that M is an odd prime power, in which case, we factorize it by binary searching the k-th root of M where k is guaranteed to be an integer in $[1, \log M]$.

3.3 Quantum algorithm for Order-Finding

By establishing that Factoring $\leq$ Order-Finding, we showed that if we could somehow find the order of $A \in \mathbb{Z}_M^*$, we could then classically factorize M.

Now, we shall see how one actually finds the order. Given n bit integers A and M, let $N = 2^{\text{poly}(n)} >> M$ where $\text{poly}(n)$ is something ridiculously large like n^{10}. Such a number N can still be written in $\text{poly}(n)$ bits.

Define $f : \{0, 1, \ldots, N-1\} \to \mathbb{Z}_M$ to be $f(x) = A^x \mod M$. Notice that $A^0 = A^s = 1$, all powers in between are distinct and then it repeats. So it is almost s-periodic, but not quite, because we do not know if s divides N. But we shouldn't have much trouble modifying period-finding slightly to solve this variant of the original problem.

Just like in period-finding, we start with our favorite state

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$$

And then we tensor this state with $|0^n\rangle$ and pass the overall quantum state through an oracle for f, O_f and end up with the state

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$$

And we measure the second register, collapsing the state to a superposition of states that involve $|x\rangle$ where $f(x)$ is a random element c in the subgroup generated by A. This is where order-finding starts getting different from period finding.

Note that s does not divide N, so we cannot be sure of the exact number of times each color c appears. Instead, we can say that appears only D times where D is either $\lfloor \frac{N}{s} \rfloor$ or $\lceil \frac{N}{s} \rceil$. We will now see how taking a massive N comes in handy.

We apply a Quantum Fourier Transform on our state to obtain the state

$$\sqrt{\frac{1}{ND}} \sum_{\gamma=0}^{N-1} \sum_{j=0}^{D-1} \omega^{\gamma \cdot s \cdot j} |\gamma\rangle |c\rangle$$

In the above state, $\omega = e^{\frac{2\pi i}{N}}$. And sampling γ from this state gives us some fixed γ_0 with probability

$$\mathbf{Pr}[\text{sampling } \gamma_0] = \frac{D}{N} \left| \frac{1}{D} \sum_{j=0}^{D-1} \omega^{\gamma_0 \cdot s \cdot j} \right|^2$$

The reason we separate $\frac{1}{D}$ as $\frac{D}{D^2}$ and move the denominator into the square is that it is nice to think of the sum being squared as an average. We want γ we select by sampling to be of the form $\lfloor \frac{kN}{s} \rceil$ (this is notation for nearest integer) for k uniformly distributed in $\{0, 1, \ldots, s-1\}$. The idea is that if γ is of the given form, then $\frac{\gamma}{N}$ is a real number that is extremely close to the simple fraction $\frac{k}{s}$ where it is known that both k and s are n-bit integers. More formally, given $\frac{\gamma}{N}$ within $\pm\frac{1}{2N}$ of $\frac{k}{s}$, we claim we can find $\frac{k}{s}$.

Now, we call upon another lemma to show how such a γ can be sampled.

Lemma 3.2. *For each γ of the form $\lfloor k\frac{N}{s} \rceil$ with $0 \leq k < s$, there is $\geq \frac{0.4}{s}$ probability of sampling γ.*

Proof. A proof can be found in lemma 9.4 of Vazirani's course notes [Vaz04]. □

We will now show how one can get $\frac{k}{s}$ when they have $\frac{N}{\gamma}$. Continued fractions are a way to approximately describe real numbers in terms of integers. A real number r would look something like

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots + \cfrac{1}{a_M}}}}$$

We will use a method involving continued fractions and go over a rough sketch of this method in lecture to use continued fractions to obtain $\frac{k}{s}$ from $\frac{\gamma}{N}$.

First, let us illustrate with an example how one can obtain the expansion of some number with continued fractions.

Consider the fraction $\frac{42}{23}$. We first split the fraction into its integer part and fractional part and express it as the sum of both.

$$\frac{42}{23} = 1 + \frac{19}{23}$$

We then express the fraction as an inversion of its reciprocal.

$$1 + \frac{19}{23} = 1 + \frac{1}{\frac{23}{19}}$$

Now, split the denominator of the second term into its integer part and fractional part and repeat.

$$1 + \cfrac{1}{1 + \frac{1}{4 + \frac{1}{1 + \frac{1}{3}}}}$$

Now we will see how one could use continued fractions to compute $\frac{k}{s}$. The idea is to use Euclid's algorithm on N and γ and stop when we get some value close to 0 rather than when we get exactly 0, and keep track of quotients of the form $\lfloor \frac{a}{b} \rfloor$ whenever we compute a value of the form $a \mod b$. We will illustrate the method with another example.

If $\frac{k}{s}$ is $\frac{11}{25}$, then $\gamma \approx \frac{11}{25}N$.

Now, we take $N \mod \gamma$ and get approximately $\frac{3}{25}N$ with 2 as the quotient. As the next step, we take $\frac{11}{25}N \mod \frac{3}{25}N$ and get roughly $\frac{2}{25}N$ with 3 as the quotient. Then, we get approximately $\frac{1}{25}N$ as the remainder from $\frac{3}{25}N \mod \frac{2}{25}N$ and get 1 as the quotient. Finally, in the last step, we get the remainder to be approximately 0 and the quotient to be 2 when we take recursively apply Euclidean's algorithm on terms that are approximately $\frac{2}{25}N$ and $\frac{1}{25}N$.

The quotients at any given step in the Euclidean algorithm could be thought of as the integral part, and finding the bigger element modulo the smaller element helps us obtain the fractional part. Using the quotients we obtained, we get the continued fraction approximation for $\frac{\gamma}{N}$ as

$$\cfrac{1}{2 + \frac{1}{3 + \frac{1}{1 + \frac{1}{2}}}} = \frac{11}{25}$$

To wrap up, we will show how we can eliminate possibilities of failure. If k and s have a common factor, then the fraction $\frac{k'}{s'}$ returned by computing the continued fractions approximation of $\frac{\gamma}{N}$ would be one of simplest form, but with $k' \neq k$ and $s' \neq s$. We will treat this possibility by showing that we can always find $\frac{k}{s}$ with k and s coprime by running the algorithm enough times.

We claim that with probability $\frac{1}{\text{poly}(n)}$, k and s are coprime.

Proof. Note that s has at most $\log s$ prime factors. By the prime number theorem, there are at least $\frac{s}{\log s}$ prime numbers less than s. The order of the number of prime numbers less than s that are coprime with s is about the same, because $\log s$ is asymptotically much less than $\frac{s}{\log s}$ so excluding those primes without losing many elements. Thus, when k is picked uniformly at random between 1 and s, there is a $\frac{1}{\log s}$ chance that it is a prime that is coprime to s. s is at most n bits long, and hence the probability that k is a coprime to s is at least $\frac{1}{\mathrm{poly(n)}}$. □

Repeat the algorithm until you get $\frac{k}{s}$ and $\frac{k'}{s}$ in lowest terms with $\mathrm{GCD}(k, k') = 1$.

Once we accomplish this, we can find s, which is the order of element A. And by using the reduction of factoring to order finding that we proved in the previous section, we can efficiently solve the factoring problem!

References

[Sho97] Peter Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5):1484–1509, 1997.

[Vaz04] Umesh Vazirani. Shor's factoring algorithm. *CS 294-2*, Fall 2004. `http://www.cs.berkeley.edu/~vazirani/f04quantum/notes/lec9.pdf`.

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

Lecture 10: Hidden Subgroup Problem

10/12/15

Lecturer: John Wright *Scribe: Lachlan Lancaster*

1 Review and Setting the Stage

We start by reviewing the problems considered in the last lecture, their similarities, and how they can be generalized to a more general problem (The Hidden Subgroup Problem) that is the topic of today's lecture. We then review some ideas of Group Theory before diving in.

1.1 A Single Problem, Leading a Double Life

We recall from last time the Period Finding Problem, which is critical in Shor's Algorithm, the problem can be given as:

Problem:
Given: $f : \mathbb{Z}_N \to$ "`colors`" with the promise that $\exists s \in \mathbb{Z}_N . s \neq 0$ such that $\forall x \in \mathbb{Z}_N$, $f(x) = f(x + s)$ OR all f-vaues are distinct.
Goal: Find s

We will refer to the above statement of the problem as the (A) statement of the problem. We then also have Simon's Problem, which can be stated very similarly to the above and we have also encountered before in the course. This problem can be stated in the following way:

Problem:
Given: $f : \mathbb{Z}_2^n \to$ "`colors`" with the promise that $\exists s \in \mathbb{Z}_2^n . s \neq 0$ such that $\forall x \in \mathbb{Z}_2^n$, $f(x) = f(x + s)$ OR all f-vaues are distinct.
Goal: Find s

We will refer to this definition of Simon's Problem as (1). Obviously the two above problem's look very similar, it's almost as though the scribe just copied the two definitions and simply changed a few details... The answer to this unification is the topic of today's lecture, but first we will go over some simple reminder topics from Group Theory.

1.2 Galois's Soliloquy

We want to restate the statements (A) and (1) of the problems at hand in terms of Group Theory. To do this let's first restrict to the case of the Period-Finding problem and enumerate what could be considered a 'solution' to the problem

$$H := \{0, s, 2s, 3s, \dots\}$$

We see that, according to statement (A), the function f takes the same value on every member of the set H, so in a sense if we know this set, then we have our solution, we can easily find s. We also notice that H is a *subgroup* of the larger group $G_p := (\mathbb{Z}_N, +)$ of the integers modulo N with addition (modulo N) as the group operation. What is we add one to everything in the above set? By the problem statement of the period finding problem, we should have another distinct set of things which are all assigned a different color, and the same thing would happen if we added two to everything in H. So we have:

$$\begin{aligned} 1 + H &:= \{1 + h | h \in H\} \\ 2 + H &:= \{2 + h | h \in H\} \\ &\vdots \\ x + H &:= \{x + h | h \in H\} \end{aligned}$$

We see easily by construction that each of these sets has a the same size and is assigned a different f-value. It is also quite easy to see that, as long as we don't repeat this process to many times and come back to H itself, each of these sets is going to be disjoint. This intuitively makes sense in when we note they have different f-values. Well, in the language of the Group Theory, the sets we generated above are called *cosets* of the subgroup H:

Definition 1.1. A **coset** of a subgroup H of a group G is the set $\{xh | h \in H\}$ for some $x \in G$. This set is usually denoted xH.

Definition 1.2. A group G is said to be **Abelian** or **commutative** if $\forall x, y \in G$ we have that $xy = yx$. That is to say that the group operation is commutative.

Specifically we should note that the above examples are the *left cosets* of H, the *right cosets* being the sets $H + x$, but since the present group G_p is Abelian this doesn't make a difference. We can take for example the case that $N = 2^n$ for some $n \in \mathbb{N}$ (which is usually convneient) and consider the following subgroup and its cosets:

$$\begin{aligned} H &:= \{0, 4, 8, \dots\} \to \text{Green} \\ 1 + H &:= \{1, 5, 9, \dots\} \to \text{Red} \\ 2 + H &:= \{2, 6, 10, \dots\} \to \text{Blue} \\ 3 + H &:= \{3, 7, 11, \dots\} \to \text{Yellow} \end{aligned}$$

We see in the above example that the above exactly have the properties we specified for the above example with $s = 4$, we also note that only *one* of the above sets is a subgroup, obviously it is H.

We now have the machinery we wanted to restate the problems as we stated them above.

Problem:
Given: $\exists H$ which is a subgroup of $\mathbb{Z}_N$ with addition modulo N such that f assigns the same value to every member of H
Goal: Find H

We call the above set theoretic statement of the problem (B). We origianlly considered this to be a *slightly* more general problem because it allows you any subgroup H, but every subroup of $\mathbb{Z}_N$ is cyclic and therefore can be generated in teh sense of statement (A). We then have finally have the following statement of the Simon's Problem, which we will refer to as (2):

Problem:
Given: Let $H = \{0, s\} \subseteq \mathbb{Z}_2^n$ which is a subgroup of $\mathbb{Z}_2^n$ with vector addition modulo 2 (as $s + s = s \oplus s = 0$). There exists an $s \in \mathbb{Z}_2^n$ and equivalent unique subgroup H as definied above such that f assigns a unique color to each coset of H
Goal: Find H

We *do* in this case have that this statement is more general than the statement (1) due to the less trvial structure of the this group. In any case, we can clearly see the similarities between these problems, (B) and (2) look even more similear than (A) and (1)! We're on the edge of a big discovery...

2 Protagonist Revealed: The HSP and Applications

We now introduse the general **Hidden Subgroup Problem** of which both of the above cases can be reduced to.

2.1 Problem Statement

Problem:
Given: Let G be a group and f be a given function such that $f : G \rightarrow$``colors". We are promised that $\exists H \subseteq G$ which is a subroup of G such that f assigns a uniqu color to each coset of H.
Goal: Find H

We note that in some cases the subgroup H may be exponentially large, so the algorithm itself will output a generating set for the subroup H. We are guaranteed that there will always be a generating set for this subgroup of polynomial size as stated (without proof) in lecture. Further similar discussions may be found in [DW71, HHR07].

2.2 Reductions

We care about this because there are a *lot* of problems that can be reduced to this more general case. Here are some examples:

Problem	Group	
Simon's	$\mathbb{Z}_2^n$	Abelian Groups
Factoring/Period Finding	$\mathbb{Z}_N$	
Discrete Log	$\mathbb{Z}_N \times \mathbb{Z}_N$	
Pell's Equation/Princp. Ideal Problem	$\mathbb{R}$	
Graph Isomorphism	S_n (Symmetric Group)	Non-Abelian Groups
Shortest Vector Problem	D_N (Dihedral Group)	

To make things clear we'll give a shor definition of the non-typical groups mentioned above:

Definition 2.1. The **Symmetric group** on N elements, or S_N is the group who's underlying set is the set of all bijections from $[N] = 1, 2, \ldots N-1, N$ to itself. The group operation on this set is the composition of two of these functions, as these functions are bijections, a composition of two of them is clearly a bijection, each one has an inverse, and the identity transformation belongs to this set.

Definition 2.2. The **Dihedral group** D_N is the group who's underlying set ig the set of symmetries of an N-sided regular polygon. Again here the group operation is the composition of two symmetries. Clearly the identitiy transformation is a symmetry, the composition of two symmetries is also a symmetry, and every symmetry has an inverse symmetry.

In returning to trying to solve the problems above we fiind our selves in one of two general cases most of the time for algorithm's to solve these problems on a Quantum Computer:

Case 1: The underlying group G associated with the problem is Abelian, in which case we can solve the problem with polylog$|G|$ number of gates and calls to the provided oracle O_f. This is the case with the first four of the problems listed in the table above. Generally, these cases are not all that interesting.

Case 2: The underlying group G associated with the problem is non-Abelian, in which case we can have an algorithm that has polylog$|G|$ calls to the oracle function O_f, but it may need and exponenital number of gates to run succesfully.

Unfortunately, it seems as though we have generally disjoint sets of interesting problems on one side and tractably solvable problems on the other, but not generally. Additionally (cause we know you're all thinking it) many people believe that quantum computers cannot have algorithms that solve NP-complete problems in polynomial time.

2.3 How to Solve These

So how wuld we go about implenting and algorithm to solve this problem generally? We have the following set-up. Well, we know we have the function of $f : G \rightarrow$"`colors`" (or equivalently $\{0,1\}^n$). It is then convenient to basically "round up" so that the numbers that we are dealing with are nice powers of two, we do this by setting $n = \lceil \log_2 |G| \rceil$ number of bits. This is annoying, but neccessary.

We then have the, supposedly given, oracle function:

$$O_f : |g\rangle |x\rangle \to |g\rangle \otimes |x \oplus f(g)\rangle \tag{1}$$

Which is defined if $g \in G$ and undefined otherwise. Note we *will* hit the case where it is undefined because of the way we have expanded the number of bits for convenience. We then use this set-up in the next section.

3 The Standard Method

We present here the way in which "basically everyone" solves the Hidden Subgroup Problem [Lom04].

3.1 Presentation

Here's the algorithm:

Step 1: We begin by preparing a uniform superposition of $g \in G$ call it $|\psi\rangle$ (or more literally a linear superposition of $x \in 2^n$ where $n = \lceil \log_2 |G| \rceil$). We do this in the usual way by applying the Hadamard gate to each of n input qubits:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{|x\rangle \in [N]} |x\rangle \approx \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle$$

Step 2: We attach the necessary m ancillary bits $|0^m\rangle$

Step 3: We apply the Oracle O_f to get:

$$O_f \left(|\psi\rangle \otimes |0^m\rangle \right) = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |f(g)\rangle$$

Step 4: We measure the second register, that is the bits who represent the ket $|f(g)\rangle$ in the above equation. We get Pr[observe color '$\mathtt{c}$'] = fraction of f-values equal to $\mathtt{c}$. Therefore we see a uniformly random color (as each coset is of the same size). This then causes the state of the system to collapse to collapse to the state that is consistent with seeing that color. We see we go from the generally stated version of the state on the right, to the state on the left after the measurement:

$$\frac{1}{\sqrt{|G|}} \sum_{\mathtt{c} \in \mathtt{colors}} \sum_{g \ \text{if} f(g) = \mathtt{c}} |g\rangle \otimes |\mathtt{c}\rangle \to \left[\frac{1}{\sqrt{|H|}} \sum_{g \ \text{if} f(g) = \mathtt{c}} |g\rangle \right] \otimes |\mathtt{c}\rangle$$

We then note that $f^{-1}(\mathtt{c}) = gH$ for some $g \in G$ and some subgroup H of G. We can then rewrite the above (dropping the tensor with the redundant $|c\rangle$) as:

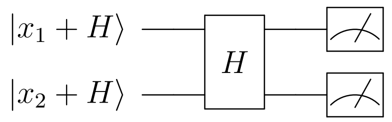
$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} \tag{2}$$

The above equation (2) is by convention called the **coset state** and is usually represented by $|gH\rangle$. So, to summarize, what have we done? We have created an algorithm which calls the oracle gate <u>once</u> and outputs a *uniformly random* coset state $|gH\rangle$. We can see that this state is uniformly random as the $\mathtt{c}$ is uniformly random and we have a bijection between these properties.

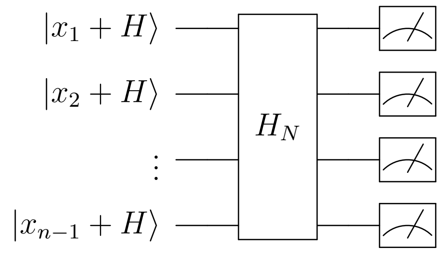
This is the exact smae distribution of picking a uniformly random $g \in G$ and outputting the coset state $|gH\rangle$.

3.2 We've Been Here Before...

We now note that if we look back carefully at our algorithms from the beginning of the lecture, that we've seen this before! In the Period Finding Problem we run the above algorithm twice by gernerating tow states, Hadamard transforming them and the performing some other classical operations. The intermediate quantum steps are shown below:



We saw a similar case for Simon's Problem, this time generating $n-1$ such states

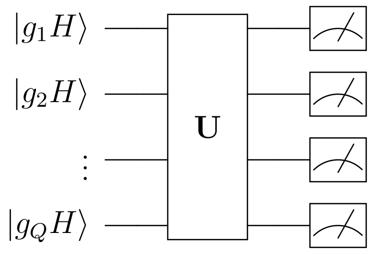


This process is the followed up by more classical computation.

3.3 The EHK Algorithm

We now review an algorithm given by Ettinger, Hoyer, and Knill [EHK04]. They showed by demonstarting and algorithm which did so, that the quantum query complexity of the Hidden Subgroup Problem is $Q = \text{poly}(n)$ where $n = \lceil \log_2 |G| \rceil$ for the associated group G.

This happens by first generateing Q cosets states using the general method outlined above. They then outline a very large and entangled unitary transformation U that must be applied to these states. This unitary transformation can be quite large and take $\mathcal{O}(\log |G|)$ number of gates to run for Abelian groups G. It then has the following picture:



The reader who is interested in the process of creating the (generally very complicated) transformation **U** is encourages to check the paper cited above. We note that the examples we gave above for the Period Finding and Simon's problem did not involve entangled operations, whereas the transformation **U** does.

4 The Final Act: Graph Isomorphism

We now present the algorithm for solving the Graph Isomorphism problem with the Hidden Subgroup Problem standard method. Of course, as noted above, the underlying group here is the Symmetric Group on n elements. We first state the problem for reference:

Problem:

Given: Two graphs $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$ which are represented here as ordered pairs of a set of vertices V_i and a set of edges between these vertices (ordered pairs of vertices) E_i. We call the grpahs C_i instead of G_i in order to avoid confusion with the previous discussion using G to denote groups. We assume here that $V_1 = V_2 = \{1, 2, \ldots, n\}$, that is that the set of vertices have the same size. We also assume for convenience that the sets are connected. We note that is this is not the case then we can separate these graphs into the cases of their disjoint components. We saw in homework 3 that we could decide this connectedness (returning a spanning tree) in $\mathcal{O}(\sqrt{n})$ time, so this is not a huge problem.

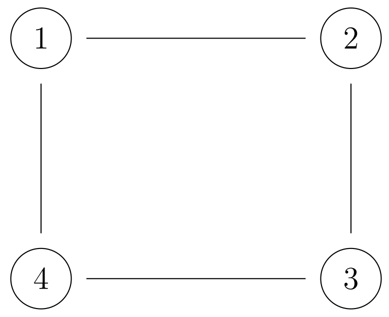
Goal: Output "Yes" if the Graphs C_1 & C_2 are isomorphic, output "No" otherwise.

Let's first give a formal definition of the isomorphism that we mean and the consider an example.

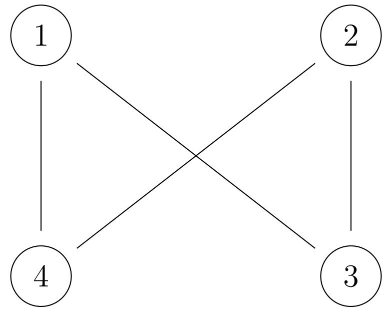
Definition 4.1. Two graphs C_1 and C_2 with sizes $|V_1| = |V_2| = n$ are said to be **isomorphic** if $\exists \pi : [n] \to [n]$ a bijection (permutation of $[n]$ such that π matches the edges of C_1 with the edges of C_2, or equivalently vice versa. That is to say that for $i, j \in V_1$ C_1 and c_2 are isomorphic if $\exists \pi$ such that $(i, j) \in E_1 \leftrightarrow (\pi(i), \pi(j)) \in E_2$.

We then consider the case of the rtwo following graphs who have vertx sets of size four. We will refer to the first graph as C_1 and the second graph as C_2

C_1



C_2



We then note that if we define the permutation π by $\pi : \{1,2,3,4\} \to \{1,3,2,4\}$ that we end up with the same edges between the nodes. So the above two graphs are isomorphic! This somehwat motivates a second definition of isopmorphism:

Definition 4.2. Given a graph $C = (V, E)$ and a permutation $\pi \in S_n$ where $|V| = n$ we may define the new graph $\pi(C)$ by $\pi(C) = (V, \pi(E))$ where $(i, j) \in E \leftrightarrow (\pi(i), \pi(j)) \in \pi(E)$.

We then can say that two graphs C_1 and C_2 are **isomorphic** is there exists a permutation π such that $\pi(C_1) = C_2$.

We will now give some short lemmas and definitions which will be esssential fo rthe main result.

Definition 4.3. The **automorphism group** of a graph C is the set of permutations of the graph nodes which leave it unchanged, denoted $\mathrm{Aut}(C)$. That is:

$$\mathrm{Aut}(C) = \{\pi \in S_n | \pi(C) = C\} \tag{3}$$

If we consider the two graphs above as one disconnected graph (as we will do below) we see that acting with π on C_2 and π^{-1} on C_1 gives us the same composite graph.

Lemma 4.4. *The Automorphism Group on a graph $C = (V, E)$ is a subgroup of the group S_n where $n = |V|$.*

Proof. We first note the it is clear the $\mathrm{Aut}(C) \subseteq S_n$, since all of it's elements are elements of S_n. We then need to show that it (i) contains the identity element, (ii) each of it's elements has an inverse in the set, and (iii) it is closed under the group operation.

(i) It is clear that the identity element is in $\mathrm{Aut}(C)$, as ths identity permutation I just gives back C so $I(C) = C$ trivially.

(ii) Suppose $\pi \in \mathrm{Aut}(C)$, then by definition $\pi(C) = C$. Applying π^{-1} to both sides of this equation we have $\pi^{-1}(\pi(C)) = \pi^{-1}(C)$ this leads simply by definition of the inverse to $C = \pi^{-1}(C)$, showing that $\pi^{-1} \in \mathrm{Aut}(C)$ by definition of the Automorphism group.

(iii) Suppose $\pi, \sigma \in \mathrm{Aut}(C)$, we want to show that their composition is also in this set. This is done easiuly as follows:

$$\begin{aligned}\pi(\sigma(C)) &= \pi(C) \texttt{ as } \sigma \in \texttt{Aut}(C)\\ &= C \texttt{ as } \sigma \in \texttt{Aut}(C)\end{aligned}$$

So that clearly $\pi\pi \in \mathrm{Aut}(C)$ and thus it is closed under the group operation.

We then have shown all the necessary properties to demonstrate that $\mathrm{Aut}(C)$ is a subgroup of S_n. ☐

Lemma 4.5. *Suppose that for a graph C we have a function $f : S_n \to \{$*`graphs on n vertices`*$\}$, defined such that $f(\pi) \to \pi(C)$ which we can treat like our "colors" in this problem. Then this function f 'hides' the Subgroup Aut(C) of C. That is that f assigns a unique value to each coset of* $\mathrm{Aut}(C)$.

Proof. We need to show that f assigns a unique value to each coset of the automorphism group.

Part 1: To do this we state a general coset of the automorphism group as $\sigma\mathrm{Aut}(C)$. We then suppose $\pi_1, \pi_2 \in \sigma\mathrm{Aut}(C)$. Then we may state $\pi_1 = \sigma\omega_1$ and $\pi_2 = \sigma\omega_2$ for $\omega_1, \omega_2 \in \mathrm{Aut}(C)$. We then show that $f(\pi_1) = f(\pi_2)$.

$$\begin{aligned}f(\pi_1) &= \pi_1(C) \texttt{ def of } f\\ &= \sigma\omega_1(C) \texttt{ def of } \pi_1\\ &= \sigma(\omega_1(C))\\ &= \sigma(C) \texttt{ def of } \mathrm{Aut}(C)\\ &= \sigma(\omega_2(C)) \texttt{ def of } \mathrm{Aut}(C)\\ &= \pi_2(C) \texttt{ def of } \pi_2\\ &= f(\pi_2) \texttt{ def of } f\end{aligned}$$

Thus $f(\pi_1) = f(\pi_2)$

Part 2: We then want to show that $f(\pi_1) = f(\pi_2)$ for $\pi_1, \pi_2 \in S_n$ implies that $\pi_1 \in \pi_2\mathrm{Aut}(C)$ or equivalently $\pi_2 \in \pi_1\mathrm{Aut}(C)$, this comes easily from the following:

$$\begin{aligned} f(\pi_1) = f(\pi_2) &\leftrightarrow \pi_1(C) = \pi_2(C) \texttt{ def of } f \\ &\leftrightarrow (\pi_2^{-1}\pi_1)(C) = C \texttt{ def of inverses} \\ &\leftrightarrow \pi_2^{-1}\pi_1 \in \mathrm{Aut}(C) \texttt{ def of } \mathrm{Aut}(C) \\ &\leftrightarrow \pi_1 = \pi_2\pi_2^{-1}\pi_1 \in \pi_2\mathrm{Aut}(C) \end{aligned}$$

This is exactly what we desired to show, so that f does indeed hide the automorphism group of C. □

With these lemmas and definitions in hand we will see that the main result will be wuite easy.

Theorem 4.6. *Graph Isomorphism between two graphs $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$ reduces to HSP on the group S_{2n} where $n = |V_1| = |V_2|$.*

Proof. Given these two graphs we form the composite graph $C_U := C_1 \cap C_2$ defined as the disjoint union of two separate graphs. This is equivalent to taking the two grpahs separately and then simply considering them as a single 'structure.'

We then define the funciton f mapping S_{2n} to the set of graphs on $2n$ vertices in a similar way to that given in Lemma 4.5 by saying $f(\pi) = \pi(C_U)$.

In this case, treating the problem as the HSP problem we see we want to find the automorphism group of C_U. Running the standard method on this problem will give a description of this automorphism group. We may then examine this group and note that if any $\sigma \in (C_U)$ exchanges members of the graphs C_1 and C_2 then we may output "Yes", otherwise, we output "No."

To see why this works we note that if the two graphs are isomorphic then the automorphism group will necessarily have a member that will exchange members of the graphs on the overall graph. Such an example of performing a permutation π that satisfies the defintion of isomorphism to C_1 and then π^{-1} to C_2 so that the overall permutation acts on C_U and leaves the graph unchanged. This clearly cannot be the case if they are not isomorphic.

We additionally note that, since the algorithm should only return a generating set for the the automorphism group, that this doesn't cause us to lose any generality. The generating set should have the same properties as discussed above. □

So we have shown that Graph Isomorphism is simply a special case of the Hidden Subgroup Problem!

CLOSE CURTAIN

References

[DW71] I. M. S. Dey and James Wiegold. Generators for alternating and symmetric groups. *Journal of the Australian Mathematical Society*, 12:63–68, 2 1971.

[EHK04] M. Ettinger, P. Hoyer, and E. Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *eprint arXiv:quant-ph/0401083*, January 2004.

[HHR07] Lorenz Halbeisen, Martin Hamilton, and Pavel Ruzicka. Minimal generating sets of groups, rings, and fields. *Quaestiones Mathematicae*, 30(3):355–363, 2007.

[Lom04] C. Lomont. The Hidden Subgroup Problem - Review and Open Problems. *eprint arXiv:quant-ph/0411037*, November 2004.

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

Lecture 11: Quantum Query Lower Bounds Using Polynomials

October 14, 2015

Lecturer: Ryan O'Donnell *Scribe: Connell Donaghy*

1 Some Quantum Algorithms

Let's review some some algorithms we've learned so far, and revisit their query complexity.

Recall Grover's algorithm for *unstructured search*: Given access to a database of size N and some function $f : [N] \to \{0,1\}$ along with a promise that $f(x) = 1$ for a unique x^*, find x^*. Using a quantum computer, we find $O(\sqrt{N})$ queries sufficient to find x^*. Now, we compare this result with query bounds on a classical computer. Classically, it is clear to see that we need $O(N)$, or more specifically $N-1$ queries to find this x^*. It's provable that even randomly with some error probability ϵ, we still need $\Theta(n)$ queries to find x^*. So, now that we've seen some cool quantum algorithms, is there a way to show that $O(\sqrt{N})$ queries is the best we can do? It'd be nice if we could show that we need $\Omega(\sqrt{N})$ for the number of queries to find this x^* using a quantum computer.

Remark 1.1. Grover's search and SAT

We've discussed that we can use Grover search and a reduction to solve Circuit-SAT in $O(\sqrt{2^n})$ or $O(1.41^n)$ time (or gates.) Proving a lower bound for Grover's search doesn't necessarily prove a lower bound for Circuit-SAT, as there could be some clever algorithm which takes advantages of gates, and doesn't simply attempt all input possibilities to the circuit.

Next, recall Simon's problem. For this problem, we are given query access to some function $f : [N] \to [M]$, where N can be represented as a binary string $\{0,1\}^n$, and M can be considered as M different "colors." We are given the promies that $f(x+s) = f(x)$ for some $s \in \{0,1\}^n$, and ignore the trivial case of $s = 0$. In this problem, we have seen that we can find s using $O(\log N)$ queries quantumly, and $\Theta(\sqrt{N})$ queries randomly via a birthday attack. [Bac06]

Finally, we consider the element distinctness (ED) problem. In this problem, we are given oracle access to some function $f : \{0,1\}^n \to [M]$, where once again we think of the output as colors. In this problem, we want to output "yes" if all the outputs of f are distinct, and say no if there is some duplicate color, such that $f(x) = f(y) = c$ for some $x \neq y$. Classically, we can sort the input and scan for duplicates in $O(N \log N)$ times. Randomly, it is possible to solve element distinctness in $\Theta(N)$ time. Quantumly, we can solve element distinctness in $\Theta(N^{\frac{2}{3}})$ queries. [BDH+01]

Well, what's the difference between these different quantum speedups? Why does Simon's problem get an exponential speedup, whereas element distinctness and search only seem to gain polynomial speedups? To answer these questions, let's first develop some notation, then revisit this remark.

2 Some New Terminology

Let's discuss some terminology which will be useful as we proceed through the polynomial method.

First, we classify problems as either **Search Problems** or **Decision Problems**.

- **Search Problems**: A problem such that the output is a string, (i.e. x^* for Grover's algorithm as discussed in section one.)
- **Decision Problems**: A problem such that the output is binary (yes/no), such as the output to the element distinctness problem.

For many problems, it is easy to construct a decision problem from a search problem. Consider Grover's algorithm. The **search version** for Grover's algorithm is: promised some $f(x) = 1$ for at moste one x find and return x. Now, consider this as a **decision problem**. In this version, we could simply return "yes" if x exists with $f(x) = 1$, or return "no" if no such x exists. We can solve this decision version by searching for some x, and then testing its value in $O(\sqrt{N})$ queries.

Next, let's recall some basic complexity theory. We say a randomized algorithm "solves" a decision problem if it outputs the correct answer more than $\frac{2}{3}$ of the time (although it will work for any probability greater than $\frac{1}{2}$ by repeating the algorithm a constant number of times.)

Lastly, let's define a **promise problem**. A promise problem is an algorithm over some subclass of all functions. This means, that we are given some promise that restricts f and disallows some possible functions. Often, given a promise problem, the algorithm will take advantage of the promise to speed up its runtime. If a problem is not a promise problem, then it is a **total problem**. That is to say, the algorithm works for all inputs on any f. Thus, for a decision problem, we need $f : [N] \to \{0, 1\}$, and all possible functions f must be valid inputs to the algorithm. Now, earlier we discussed a promise version of Grover, where we were promised that there was some x^* such that $f(x^*) = 1$. Let's compare this to total decision Grover search, where we must now output yes if such an x^* exists, and output no if $f(x) = 0$ on all inputs. In Homework 3, we solved that even this total decision version of Grover search is solvable in $O(\sqrt{N})$ queries.

Returning to search versus decision problems, let's revisit Simon's problem. We can see that for Simon's problem, we have a search problem, as we try to find s such that $f(x + s) = f(x)$ for all x. Also, we are given a promise that such an s exists, and that it is not 0. So what would the decision problem be? We simply output "yes" if $s = 0$, and "no" otherwise. If $s = 0$, then we know that we have all distinct colors, so the decision version of Simon's problem is actually the Element Distinctness Problem! However, we no longer have a promise in this decision version, as we must produce a valid answer over any input function f.

At the end of section one, we questioned why Simon's search had such a great speedup from a quantum computer, compared to a relatively unexciting polynomial speedup for element distinctness. With our new terminology, we can give an answer, thanks to the following theorem

Theorem 2.1. *For a total decision problem, a quantum algorithm can NOT beat a classical algorithm by more than a polynomial runtime. Let $D(n)$ be the classical query complexity, and $Q(n)$ be the quantum query complexity. We have*

$$D(n) \leq Q(n)^6$$

Thus, our speedups due to a quantum computer are actually fairly limited on total decision problems! [BBC+ 01]

Using this theorem, we see that because Grover's algorithm and Element Distinctness are both **total decision** problems, then it is impossible to get the type of exponential speedup we get in Simon's problem. Simon's problem is a **promise problem**, so we are able to achieve some nice speedups.

Remark 2.2. Similar to the maximum polynomial speedup for total decision problems, it has also been proven that $D(n) \leq R(n)^3$ and that $R(n) \leq Q(n)^{2.5}$, where $R(n)$ is the runtime of a randomized algorithm. [BBC+01]

3 How to Prove Lower Bounds

This lecture along with next lecture, we're going to go over the **polynomial method** for proving lower bounds. Another method which will be addressed later in the course is the **adversary method**, which is actually more powerful but a little more difficult to understand.

To understand and use the polynomial method, we're going to switch up some notation. Before, we had data as a function

$$f : [N] \to [M]$$

From now on, we'll consider data as a "string" such as

$$w \in [M]^N$$

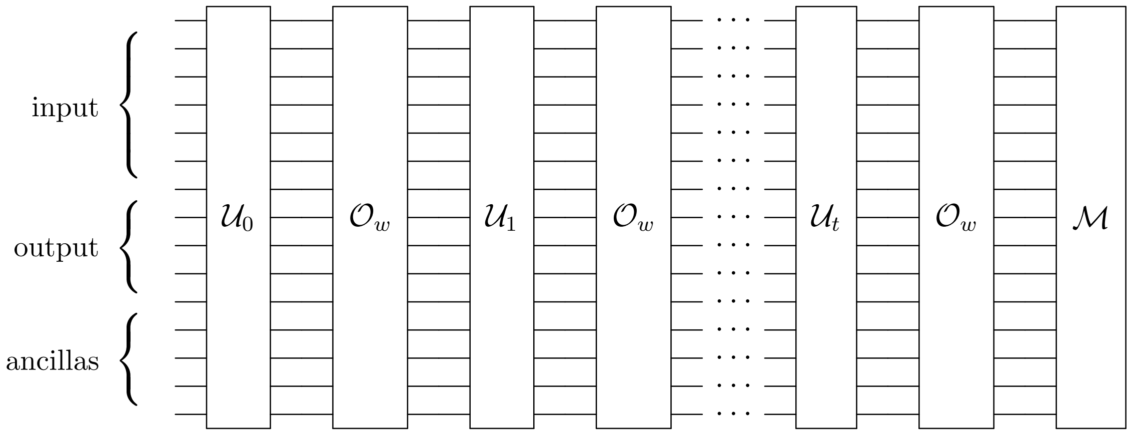
or

w_1	w_2	$\cdots$	w_{N-1}	w_N

Essentially, our data is now some mystery array full of mystery strings that we can index into via w_i, where $i \in 0, 1, \cdots, N$. Our oracle, previously O_f, will now be denoted as O_w. The classical version of this oracle takes in an i, and outputs w_i. Quantumly, this means if we apply the new oracle to some $|i\rangle$ and preserve reversible computing, we get $|i\rangle \to (-1)^{w_i} |i\rangle$, or $|i\rangle |y\rangle \to |i\rangle |y \oplus w_i\rangle$. Now, our function F, will actually be considered a property of all of the strings w, instead of a mapping from $[N]$ to $[M]$ as we had previously done. Let's consider Grover's total decision algorithm in this new notation. Each w_i is either 0 or 1. We want to know if there is a $w_i = 1$ for some i. Essentially, we are taking the logical or of all of the $w \in F$, or $OR(w_1, w_2, \cdots, w_n)$. Now, before we go on to prove a lower bound for Grover Search (by proving a lower bound for the logical OR of N bits) we show the following:

Theorem 3.1. *Let $\mathcal{A}$ be a quantum query algorithm making t queries to O_w, with $w \in [M]^N$ Lets denote the following notation for $\widetilde{w_{i,c}} = \begin{cases} 1 & if w_i = C \in [M] \\ 0 & otherwise \end{cases}$. Then, the amplitude of any basis state is a polynomial in $\widetilde{w_{i,c}}$ of degree less than or equal to t.*

Proof. First, let's take a look at what this algorithm $\mathcal{A}$ actually looks like as a quantum circuit.



In this circuit, we have n input bits, m bits where the output will be stored, and a ancilla bits. $\mathcal{U}_i$ denotes the ith unitary transform on the state, and each O_w is an application of our oracle, finally followed by a measurement $\mathcal{M}$. Now, we assume that our input is initially all 0's, and we continue the proof by induction on t. For our base case, we will consider the state of all the qubits after some unitary gate $\mathcal{U}_0$ has been applied. Our basis state originally is

$$|0^n\rangle \otimes |0^m\rangle \otimes |0^a\rangle$$

Now, after we apply the first unitary transformation to this basis state, we get a state which looks like

$$\sum_{j=0}^{w-1} \sum_{b \in \{0,1\}^m} \sum_{z \in \{0,1\}^a} \alpha_{jbz} |j\rangle |b\rangle |z\rangle$$

Now, we can see that since α_{jbz} is always a constant coefficient, that this is a 0 degree polynomial. Thus, our base case is done.

Induction Hypothesis:
After k steps, we have some state $|\psi\rangle$, which looks like

$$\sum_{j=0}^{w-1} \sum_{b\in\{0,1\}^m} \sum_{z\in\{0,1\}^a} P_{j,b,z}(\widetilde{w}) |j\rangle |b\rangle |z\rangle$$

Where each $P_{j,b,z}(\widetilde{w_s})$ is a polynomial of degree at most k.

Inductive Step Now, we want to apply O_w to this state. This application yields:

$$O_w |\psi\rangle = \sum_{j=0}^{w-1} \sum_{b\in\{0,1\}^m} \sum_{z\in\{0,1\}^s} P_{j,b,z}(\widetilde{w_s}) |j\rangle |b \oplus w_j\rangle |z\rangle$$

Which we can rewrite with $b' = b \oplus w_j$ as

$$\sum_{j=0}^{w-1} \sum_{b'\in\{0,1\}^m} \sum_{z\in\{0,1\}^s} \left(\sum_{a\in\{0,1\}^m} \widetilde{w_{j,a}} * P_{j,b'\oplus a,z}(\widetilde{w}) \right) |j\rangle |b'\rangle |z\rangle$$

Now, we can see that this application results in a polynomial of degree at most $k+1$, as $(\sum_{a\in\{0,1\}^m} \widetilde{w_{j,a}} P_{j,b'\oplus a,z}(\widetilde{w}))$ will increment the degree by at most one, as each P is of degree at most k. Thus, by induction, the degree increases by at most once per query made, so for a t query algorithm, the amplitude of any bases state is a polynomial in $\widetilde{w_i, c}$ of degree at most t. □

Corollary 3.2. *Acceptance (YES) of $\mathcal{A}$ is a real-coefficient polynomial P in $\widetilde{w_{i,c}}$ of degree less than or equal to $2t$*

Proof. The probability of measuring some $|j\rangle |b\rangle |z\rangle$ is $P_{j,b,z}P^*_{j,b,z}$. Because $P_{j,b,z}$ and $P^*_{j,b,z}$ are both polynomials of degree at most t from theorem 3.1. Their product must also be real, because we are taking the magnitude of each $P_{j,b,z}$, which is always real. □

Consider a decision problem. In this , $M = 2$ as the two colors we have are "yes" and "no." For $w \in \{0,1\}^n$ then P is just an approximating polynomial of deg $\leq 2t$ in $w_1, w_2, \cdots, w_i, \cdots, w_n$. In this case, we have $\widetilde{w_{i,1}} = w_i$ and $\widetilde{w_{i,0}} = 1 - w_i$. Thus, P is a *multilinear* polynomial of degree wt. We know that each variable is never to a degree of more than 1, because $\widetilde{w_{i,c}}^2 = \widetilde{w_{i,c}}$ Thus, we describe P as

$$P(w_1, \cdots, w_n) = \sum_{S\subseteq\{1,\cdots,N\}} c_s \prod_{i\in S} w_i$$

Corollary 3.3. *If $\mathcal{A}$ solves some decision problem $F : [M] \to \{0,1\}$ with t queries, then there is a degree $2t$ polynomial P in terms of all $\widetilde{w_{i,c}}$ such that $Pr[\texttt{acceptance}] = P(w)$. For an error bounded algorithm, this polynomial has the property that*

$$|P(w) - F(w)| \leq \frac{1}{3}$$

Now, as we proceed, we can show a minimum number of queries for $\mathcal{A}$, by showing a minimum degree of P that satisfies all of its requirements! Let's consider Grover's algorithm as an example.

4 Example: Grover's Total search

Now, we proceed to conclude with an example of Grover decision

We have from earlier that $F = OR$ on N bits, and we want to show that this *requires* $\Omega(\sqrt{N})$ queries to compute this property F. Essentially, we want to show that any approximating polynomial P for OR has $\deg(P) \geq \Omega(\sqrt{N})$.

We begin by defining some function Q, of the form $Q(w_1, w_2, \cdots, w_n)$ to be a symmetrized polynomial of our approximating multivariate polynomial P, and let $\deg P = 2t$. That is to say,

$$Q = \frac{1}{N!} \sum_{\pi \in S_w} P(w_{\pi(1)}, w_{\pi 2}, \cdots, w_{\pi(n)})$$

Since Q is a sum of polynomials of degree $2t$, then $\deg Q \leq \deg P = d$. Since P approximates F, then we have $P(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$ and $P(w_1, w_2, \cdots, w_n) \in [\frac{2}{3}, 1]$ if some $w_i = 1$. Now, we want to show that Q also satisfies this approximation.

Theorem 4.1. *If P approximates F, then Q also does as well.*

Proof. This proof falls into two cases. We can say that the input of P is either all zeroes, or at least one non-zero term.

Case 1 If the input is all zeroes, then we have $P(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$ by our definition of P. Thus, we can compute a range for Q by plugging in, resulting in $Q(0, 0 \cdots 0) = \frac{1}{N!}(N! P(0, 0, \cdots, 0)) = P(0, 0, \cdots, 0)$, so clearly $Q(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$

Case 2 In this case, because our input to P is not all 0s, then each permutation of the inputs cannot be all 0s. Thus, each permutation of these inputs on P can output a different output in $[\frac{2}{3}, 1]$. However, we know that the sum over all permutations must be in $[N!\frac{2}{3}, N!]$ by adding all of the outputs, and dividing this by $N!$ gives us Q, which is still in $[\frac{2}{3}, 1]$

Thus, because $Q(0, 0, \cdots, 0) \in [0, \frac{1}{3}]$ and for all other $Q(w)$ with some $w_i = 1$, $Q(w) \in [\frac{2}{3}, 1]$, Q also approximates F. □

We've shown that Q, like our polynomial P, satisfies F. However, it is interesting to note that Q is a *multivariate* polynomial, which does not change on any permutations of come input. Now, let's observe some interesting properties of Q.

Observation 4.2. *Q is symmetric in $w = (w_1, w_2, \cdots, w_n)$. That is to say, it doesn't change if you permute its arguments. Since $w_i \in \{0, 1\}$, then $Q(w)$ depends only on the numbers of 1s in its input, or its hamming weight (denoted as $|w|$.) This implies, as you'll show in the homework, that Q depends on z, with $z = \sum_{i=1}^{N} w_i$. As a consequence of this, we can argue that $Q = q(z)$ for some univariate polynomial q of z, with $\deg(q) \leq \deg(Q)$. However, we recall that q is defined only on integer values of z. $q(z) = 0$ if $z = 0$, and $q(z) = 1$ $\forall z \geq 1 \in \mathbb{Z}$. Other values of q are undefined, and can take any arbitrary value.*

Claim 4.3.

$$\deg q(z) \geq \Omega(\sqrt{N})$$

Proof. To prove this, we're going to need **Markovs *other* Inequality**, which states

Lemma 4.4 (**Markov's *other* Inequality**). *For some polynomial $P : \mathbb{R} \to R$ of degree d such that P is bounded in some box of height h and length l and univariate of some variable z, then*

$$|P'(z)| \leq d^2 \frac{h}{l}$$

inside the box of height h and length l [Mar90]

For an example, consider a box bounded by $[-1, +1] \times [-1, +1]$. By this other inequality, we have that if $|p(z)| \leq 1$ for all $|z| \leq 1$, then by this inequality we know $|p'(z)| \leq \deg(p)^2$ for all $|z| \leq 1$, because $h = l = 1$.

As we proceed, we'll make a convenient assumption at our conclusion, and then discuss how the assumption was not actually necessary, so our claim holds for all $q(z)$. Let's assume that $q(z) \in [0, 1]$ for all $z \in [0, N]$, even though we only know the restrictions on $q(z)$ for integer inputs z. By assuming this, we can get a height of 1 on our box which has length N. Using Markov's inequality, we can get that $|q(z)| \leq \frac{deg(q)^2}{N} = \frac{4t^2}{N}$ Because q goes from a value at most $\frac{1}{3}$ at $z = 0$ to a value of at least $\frac{2}{3}$ at $z = 1$, we must have by the Mean Value Theorem that $|q'(z)| = \frac{1}{3}$ for some $z \in [0, 1]$ Plugging this $|q'(z)|$ into Markov's other inequality, we get $\frac{1}{3} \leq \frac{4t^2}{N}$, which solves to show $t \geq \Omega(\sqrt{N})$.

Now, we want to show why we can discard our assumption that $q(z) \in [0, 1]$ for all $z \in [0, N]$. Disregarding this assumption, we realize that we know longer have a guarantee of $h = 1$ inside of our box. Now, we'll let the height of our box be $h = \max_{z \in [0,N]} |q(z)|$. If $h = 2$, the above analysis is still valid, with some constant factor going into the $\Omega(\sqrt{N})$. Now, let's consider the case where $h \geq 2$, and our assumption was entirely invalid. In this case, we know we have $|q(z)| = h$ at some $z = u$. Because $q(z)$ is bounded on integers, $q(\lfloor u \rfloor) \leq 1$. By the definition of a floor function, we have $u - \lfloor u \rfloor \leq 1$. Again, we can use the Mean Value Theorem to see that $|q'(z)| \geq \frac{h-1}{u - \lfloor u \rfloor} \geq h - 1 \geq \frac{h}{2}$ for some z with $\lfloor u \rfloor \leq z \leq u$. Now, we have a second relationship between $|q'(z)|$ and the height of our box. We have

$$|q'(z)| \leq \deg(q)^2 \frac{(2h)}{N} \leq 8t^2 \frac{h}{N}$$

And also

$$|q'(z)| \geq \frac{h}{2}$$

Thus, combining these two facts, we get $\frac{h}{2} \leq 8t^2 \frac{h}{N}$. Regardless of the behaviour of $q(z)$ in between integer inputs for z, by this analysis and Theroem 4.1 we have shown that the number of queries t required for taking the OR of N bits is $\Omega(\sqrt{N})$. Consequently, Grover's search algorithm has optimal query complexity for a quantum computer. □

References

[Bac06] Dave Bacon. Simon's algorithm. *CSE 599d Course Notes*, 2006.

[BBC$^+$01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, July 2001.

[BDH$^+$01] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. *Proceedings of 15th IEEE Conference on Computational Complexity*, pages 131–137, 2001.

[Chi13] Andrew Childs. Query complexity and the polynomial method. *Quantum algorithms (CO 781/CS 867/QIC 823, Winter 2013)*, 2013.

[Mar90] A.A. Markov. On a question by d. i. mendeleev. *Zap. Imp. Akad. Nauk SPb.*, 62:1–24, 1890.

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

Lecture 12: Lower Bounds for Element-Distinctness and Collision

October 19, 2015

Lecturer: John Wright *Scribe: Titouan Rigoudy*

1 Outline

In this lecture, we will:

- Briefly review the Element Distinctness problem.
- Introduce the Collision problem and its applications.
- Show how it reduces to the Element Distinctness problem.
- Prove an $O(N^{1/3})$ upper bound on the quantum query complexity of the Collision problem.
- Use the polynomial method to prove a matching $\Omega(N^{1/3})$ lower bound.

2 Review

2.1 Notation

Last lecture we introduced the polynomial method and used it to prove a quantum query lower bound of $\Omega(\sqrt{N})$ for Grover's total search problem.

Recall that in order to do so, we changed the way we view inputs to our quantum algorithms. Instead of considering that inputs are functions $f : [N] \to [M]$, we now consider them to be strings $w \in [M]^N$, with slots labeled by numbers $i \in [N] : w_i \in [M]$. Pictorially:

$$w : \quad \boxed{w_1}\boxed{w_2}\boxed{\quad \dots \quad}\boxed{w_N}$$

Correspondingly, our oracle for w is now $\mathcal{O}_w$ instead of $\mathcal{O}_f$, and we care about some properties of strings represented by $F : [M]^N \to \{0, 1\}$.

2.2 Element Distinctness

One such property which we mentioned last time is Element Distinctness (hereafter abbreviated ED):

Definition 2.1. Given $w \in [M]^N$, say "yes" if all the values w_i are distinct, "no" if there is a least one duplicate.

Recall that in the previous lecture we defined two classes of quantum problems: *promise problems*, for which the input w is promised to have some property, i.e. w is restricted to some subset of $[M]^N$; and *total problems*, for which the input w may be any string in $[M]^N$. We saw last time that the quantum query complexity of any total problem cannot be exponentially lower than its classical query complexity.

ED is an example of a *total problem*, and as such we cannot hope to find a quantum query algorithm providing a massive speedup over classical query algorithms. In fact, the classical and quantum query complexities are known to be respectively $\Theta(N)$ and $\Theta(N^{\frac{2}{3}})$.

3 The Collision problem

3.1 Definition

The problem is the following:

Definition 3.1. Given $w \in [M]^N$ and $r \geq 2$ such that $r|N$, output:

- ”yes” if all the values in w are distinct, i.e. if w is 1-to-1.
- ”no” if w is r-to-1.

Example 3.2. *Let $N = 4, M = 4$ and $r = 2$. The following input is a ”yes” instance:*

1	2	3	4

Whereas the following input is a ”no” instance:

1	2	2	1

This is clearly a promise problem, as w cannot be any arbitrary string: it must be either 1-to-1 or r-to-1. Also, intuitively, this problem seems easier to solve than ED because of all the repeated elements. One might hope that we may come up with a very efficient quantum query algorithm for solving it.

In the case that $r = 2$, then there is an $O(\sqrt{N})$ randomized query algorithm relying on the birthday paradox. Simply pick $O(\sqrt{N})$ elements from w uniformly at random and a collision will happen with high probability if it is a ”yes” instance. This algorithm extends to a general r, yielding a complexity of $O\left(\sqrt{\frac{N}{r}}\right)$.

What about the quantum query complexity? We know that for total problems, there can only be a polynomial gap between classical and quantum query complexities. Here, however, we are considering a promise problem, so we may hope to do much faster than randomized. On the other hand, this problem is very closely related to ED, which cannot be solved very fast. So what will it be?

It turns out that the best quantum query algorithm for the collision problem has a query complexity of $\Theta\left(\left(\frac{N}{r}\right)^{1/3}\right)$. We will dedicate the end of the lecture to proving this.

Remark 3.3. Henceforth, for simplicity of presentation, we shall only consider the case $r = 2$. Everything we will do can be easily generalized to an arbitrary r.

3.2 Applications

Before we dive in to the math, let us ask why we are interested in solving this problem. It is a nice little problem, and there is no reason why not, but there are some more important applications of this problem. Consider an alternate universe in which the collision problem is solvable using only $O(\log N)$ queries, and let us examine the consequences.

Recall the Graph Isomorphism Problem, one of the most famous NP-intermediate problems:

Definition 3.4. Given $G = ([n], E)$ and $G' = ([n], E')$ two graphs of n nodes each, determine if they are *isomorphic*, i.e. if there exists a permutation $\pi \in S_n$ such that:

$$(i, j) \in E \iff (\pi(i), \pi(j)) \in E'$$

In our alternate universe, it is possible to solve this problem efficiently. First, number all permutations of $S_n = \{\, \pi_1, \dots, \pi_{n!} \,\}$, and write:

$$w = \begin{array}{|c|c|c|c|c|c|}\hline \pi_1(G) & \quad\dots\quad & \pi_{n!}(G) & \pi_1(H) & \quad\dots\quad & \pi_{n!}(H) \\ \hline \end{array}$$

Conceptually w is thus a huge string of length $2n!$, even though in practice one may not need to write it all out, as long as individual elements may be computed efficiently as needed. The "colors" in this case are entire graphs.

If the graphs are isomorphic, then there must be collisions in w. Indeed each permutation of G must appear at least once on the "left" side of w and on the "right" side.

In particular, if the graphs are automorphism-free then w is exactly 2-to-1. Running our efficient algorithm for the collision problem on w thus solves the graph isomorphism problem in this case, with $O(\log(2n!)) = O(n \log n)$ queries.

If the graphs are not automorphism-free, then it is also possible to reduce to the collision problem, but how to do so isn't immediately obvious.

Another interesting application of the collision problem is *collision-resistant hash functions*. Such functions are basic primitives in cryptography, allowing to build cryptographically secure systems. Informally, a hash function $h : \{\, 0, 1 \,\}^n \to \{\, 0, 1 \,\}^m$ is collision-resistant if it is "hard" to find $x, y \in \{\, 0, 1 \,\}^n$ such that $h(x) = h(y)$. In practice, cryptographers pick h at random from a family of hash functions which is proven to resist to known cryptographic attacks with high probability.

However, if we extend our alternate universe to have an efficient solution to the *search problem* version of the collision problem, then there can exist no secure hash functions against quantum computers.

Unfortunately, or fortunately for cryptographers, we do not live in such a universe. There is a known polynomial lower bound for the collision problem.

Interestingly enough, there exists an application of the lower bound to black hole physics. There is a known problem in that field known as the firewall paradox. At a very high level, if a certain number of assumptions related to black hole physics are held to be true, then one can derive a paradox. The problem then becomes that of finding which assumptions are false. A recent paper [HH13] has shown that in order to computationally find those assumptions, it is necessary to solve the collision problem on an intractably large input.

4 Query complexity bounds

4.1 Element Distinctness bounds

In section 2.2 we said that the quantum query complexity of ED is $\Theta(N^{2/3})$. Indeed, there exists a quantum walk-based quantum query algorithm for ED with query complexity $O(N^{2/3})$ [Amb07]. This gives us the upper bound.

Fact 4.1. *The lower bound of for ED follows directly from the collision problem lower bound.*

Proof. By contrapositive. Suppose algorithm $\mathcal{A}$ solves ED with $o(N^{2/3})$ quantum queries. Let $w \in [M]^N$ be either a 1-to-1 string or a 2-to-1 string. The reduction once again follows from the birthday paradox.

Pick a random subset $S \subseteq [N]$ of size $O(\sqrt{N})$. If w is 1-to-1 then $w_{|S}$ is also 1-to-1. Otherwise if w is 2-to-1 then with high probability there is at least one duplicate in $w_{|S}$.

Therefore run $\mathcal{A}$ on $w_{|S}$ and we have solved the collision problem for w using $o\left(\left(N^{1/2}\right)^{2/3}\right) = o(N^{1/3})$ queries. □

4.2 Collision problem upper bound

The upper bound for the collision problem is much easier to prove than the lower bound, so let us tackle it first.

Claim 4.2. *There exists a quantum query algorithm for the collision problem using $O(N^{1/3})$ queries.*

Proof. Let $w \in [M]^N$ be either 1-to-1 or 2-to-1. The algorithm proceeds as follows:

1. Pick a random subset $S \subset [N]$ of size $\Theta(N^{1/3})$.
2. Classically query w on all indices in S, and record all colors in $C = \{\, w_i \mid i \in S \,\}$. If there are any duplicates, say "no" and halt.
3. Implement the following oracle over $N + 1$ qubits:

$$\mathcal{O}_w^C : |i\rangle\,|b\rangle \mapsto \begin{cases} |i\rangle\,|b \oplus 1\rangle & \text{if } w_i \in C \\ |i\rangle\,|b\rangle & \text{if } w_i \notin C \end{cases}$$

   It is possible to implement $\mathcal{O}_w^C$ using two queries to $\mathcal{O}_w$.

   Observe that if w is 2-to-1, then there exists $S' \subset [N] \setminus S$ of size $|S|$ such that $\forall\, i \in S'$, $\mathcal{O}_w^C\,|i\rangle\,|0\rangle = |i\rangle\,|1\rangle$. Conversely, if w is 1-to-1 then $\forall\, i \in [N] \setminus S$, $\mathcal{O}_w^C\,|i\rangle\,|0\rangle = |i\rangle\,|0\rangle$.

4. Run Grover's algorithm on $\mathcal{O}_w^C$ and $[N] \setminus S$. We are looking for one of $|S|$ possible "1" slots out of $N - |S|$, therefore the query complexity is:

$$O\left(\sqrt{\frac{N-|S|}{|S|}}\right) = O\left(\sqrt{\frac{N}{N^{1/3}}}\right) = O(N^{1/3})$$

□

This proves an upper bound of $O(N^{1/3})$ for the collision problem. We will now show that this bound is tight by proving a matching $\Omega(N^{1/3})$ lower bound.

5 A lower bound for the collision problem

5.1 Review: the polynomial method and Grover's algorithm

Recall the polynomial method we introduced in the previous lecture. We want to prove a lower bound on the number of queries needed to compute a given property $F : [M]^N \to \{0, 1\}$.

Let $w \in [M]^N$, and $\forall c \in [M], \forall i \in [N]$, define:

$$\widetilde{w_i^c} = \begin{cases} 1 & \text{if } w_i = c \\ 0 & \text{otherwise} \end{cases}$$

If there exists a t-query quantum algorithm $\mathcal{A}$ to compute F, then there must exist some polynomial P of degree $d = 2t$ in the $\widetilde{w_i^c}$'s such that:

$$\mathbf{Pr}[\mathcal{A} \text{ accepts on } w] = P(w)$$

And therefore, P must approximate F, i.e. for all strings $w \in [M]^N$ the following inequality holds:

$$|P(w) - F(w)| \leq \frac{1}{3}$$

Our goal is then to lower bound d, and we have reduced our quantum query complexity problem to one on approximating polynomials.

Last time we saw how to prove a lower bound for Grover's algorithm using the polynomial method, and the proof was fairly straightforward. This time the proof we will do is fairly involved, so in order to warm up we will review last lecture's proof.

Proof. Let F, the property we are interested in computing be the OR function on N qubits. Suppose we have P, a polynomial of degree d in the $\widetilde{w_i^c}$'s which approximates F.

We want to lower bound d, but P is a complicated multivariate polynomial in NM variables, so the first step is symmetrization. Define Q, another polynomial, as the following:

$$\begin{aligned} Q(w) &= \mathop{\mathbf{E}}_{\pi\sim S_N}\left[P(w_{\pi(1)},\ldots,w_{\pi(N)})\right] \\ &= \frac{1}{N!}\sum_{\pi\in S_N} P(w_{\pi(1)},\ldots,w_{\pi(N)}) \end{aligned}$$

Q is of degree at most d and is symmetric, i.e. $\forall\, w \in [M]^N$, $\forall\, \pi \in S_N$:

$$Q(w_{\pi(1)},\ldots,w_{\pi(N)}) = Q(w_1,\ldots,w_N)$$

Therefore as shown in Homework 4 there exists a univariate polynomial p of degree at most d such that $\forall\, w \in [M]^N$, $Q(w) = p(|w|)$ where $|w|$ is the Hamming weight of w. In fact, the following equality holds $\forall\, k \in [N]$:

$$p(k) = \mathop{\mathbf{E}}_{|w|=k}[P(w)]$$

The next step is to take p and apply approximation theory to it. We know the following:

$$p(0) \le \frac{1}{3} \qquad \forall\, i \ge 1,\ p(i) \ge \frac{2}{3}$$

In other words, p makes a sharp jump between 0 and 1, and thereafter is relatively constant. Using Markov's other inequality, we can then show that p must have large degree. $\square$

5.2 Proof of the lower bound

The proof of the collision problem lower bound has some history. First came Aaronson [Aar02], who proved a lower bound of $\Omega((\frac{N}{r})^{1/5})$. Then Shi [Shi02] improved the exponent from 1/5 to 1/3, but only for M large enough. Finally, Ambainis [Amb05] and Kutin [Kut05] both independently proved the 1/3 lower bound for all M.

We will follow Kutin's proof, as it is a nice self-contained argument.

Suppose we have a t-query algorithm for the collision problem. Then there exists a polynomial P of degree $d = 2t$ such that:

$$\begin{aligned} P(w) &\ge \frac{2}{3} \quad \text{if } w \text{ is 1-to-1} \\ P(w) &\le \frac{1}{3} \quad \text{if } w \text{ is 2-to-1} \\ 0 \le P(w) &\le 1 \quad \forall\, w \in [M]^N \end{aligned}$$

What is perhaps surprising in this proof is that eventually, we will care about and closely examine the behavior of P on all inputs *except* those inside the promise, i.e. when w is neither 1-to-1 nor 2-to-1.

The first step is the same as in Grover's lower bound proof: symmetrization. This time however, we need to be more careful. Define $W_{t,a,b}$ to be the set of all strings w such that:

- w is a-to-1 on a set of t indices.
- w is b-to-1 on the remaining $n-t$ indices.
- Elements are otherwise distinct, i.e. there is no element in both sets.

Example 5.1. *The following string is in $W_{4,2,3}$ as it is 2-to-1 on 4 indices ($\{1,2,3,4\}$), and the remaining $10-4=6$ indices are 3-to-1:*

$$w = \boxed{1}\boxed{1}\boxed{2}\boxed{2}\boxed{3}\boxed{3}\boxed{4}\boxed{3}\boxed{4}\boxed{4}$$

It is also in $W_{6,3,2}$, as we can switch a with b and t with $n-t$.

Example 5.2. *The following string is in $W_{6,3,1}$, as it is 3-to-1 on 6 indices ($\{1,2,4,6,7,8\}$), and 1-to-1 on the remaining $8-6=2$ indices:*

$$w = \boxed{2}\boxed{3}\boxed{1}\boxed{2}\boxed{5}\boxed{1}\boxed{2}\boxed{1}$$

Similarly, it is also in $W_{2,1,3}$.

Example 5.3. *For all t, $W_{t,1,1}$ is the set of all 1-to-1 strings.*

Strings in $W_{t,a,b}$ can therefore be seen as "hybrids" between a-to-1 and b-to-1 strings, leaning more towards the former and less towards the latter as t goes from 0 to N.

Note that $W_{t,a,b}$ is well defined if and only if the following hold:

$$0 \leq t \leq N \quad a|t \quad b|(N-t)$$

Definition 5.4. Such a (t,a,b) triple is called *valid*.

Now define the following function on valid triples:

$$Q(t,a,b) = \mathop{\mathbf{E}}_{w \sim W_{t,a,b}}[P(w)]$$

Theorem 5.5. *Q is a degree d polynomial in (t,a,b).*

Proof. Maybe in a homework. □

We have now symmetrized our polynomial. The second step in the proof is also conceptually the same as for Grover's: apply approximation theory to Q. This time we still have a 3-variable polynomial, but we will deal with it. But first, we need a small fact that will enable us to analyze Q.

Fact 5.6. *Let $q(X)$ be a degree d polynomial in X. Let $a < b$ be integers, and $z \in [a,b]$. If the following inequalities hold:*

$$|q(i)| \leq 2 \quad \forall\, i \in \mathbb{N}, a \leq i \leq b$$
$$|q(z) - q(\lfloor z \rfloor)| \geq \frac{1}{100}$$

Then $d = \Omega\left(\sqrt{(z-a+1)(b-z+1)}\right)$. In particular, $d = \Omega\left(\sqrt{b-a}\right)$.

If additionally $z \approx \frac{a+b}{2}$, then $d = \Omega(b-a)$.

Proof. This is a combination of Markov's other inequality and Bernstein's inequality. □

Now we can finally prove the main result:

Theorem 5.7. *$Q(t,a,b)$ has degree $\Omega(N^{1/3})$.*

Proof. First off, here are some facts about Q we will make use of:

- $\forall\, t \in \{\, 0, 1, 2, \dots, N \,\}$, $W_{t,1,1}$ is the set of 1-to-1 strings, thus $\frac{2}{3} \le Q(t,1,1) \le 1$.
- $\forall\, t \in \{\, 0, 2, 4, \dots, N \,\}$, $W_{t,2,2}$ is the set of 2-to-1 strings, thus $0 \le Q(t,2,2) \le \frac{1}{3}$.
- $\forall\, (t,a,b)$ valid, $0 \le Q(t,a,b) \le 1$.

We have reduced the problem to only polynomials that obey these three facts. Now we have to derive the lower bound on d.

Let $T = 2\left\lfloor \frac{N}{4} \right\rfloor$. Observe that $T \approx \frac{N}{2}$, and T is even.

Now consider the 2-D function $(a,b) \mapsto Q(T,a,b)$. We know that:

$$Q(T,1,1) \ge \frac{2}{3} \qquad Q(T,2,2) \le \frac{1}{3}$$

But what about $Q(T,1,2)$? Either $Q(T,1,2)$ is at least $\frac{1}{2}$, or it is at most $\frac{1}{2}$. Suppose $Q(T,1,2) \le \frac{1}{2}$.

Define $g(x) = Q(T,1,2x+1)$. Observe that:

$$g(0) = Q(T,1,1) \ge \frac{2}{3} \qquad g\left(\frac{1}{2}\right) = Q(T,1,2) \le \frac{1}{2}$$

Let k be the following:

$$k = \min \{\, i \in \mathbb{N}^\star \mid |g(i)| > 2 \,\}$$

By definition, $\forall\, i < k$, $|g(i)| \le 2$. We can therefore apply fact 5.6 and prove that $\deg(g) = \Omega(\sqrt{k})$. Furthermore, we know that $\deg(Q) \ge \deg(g)$ so it follows that $\deg(Q) = \Omega(\sqrt{k})$.

Thus if we could prove that $k = \Omega(N^{2/3})$, then we would be done. The problem is that we have no information about k. Maybe it happens to be tiny, in which case this doesn't prove anything. In order to work around this, we will have to leave 2-D et go back to 3-D.

Let h be the function defined as follows:

$$h(y) = Q(N - (2k+1)y, 1, 2k+1)$$

Observe that the following inequality holds:

$$\left| h\left(\frac{N-T}{2k+1}\right) \right| = |Q(T,1,2k+1)| = |g(k)| > 2$$

In addition, note that $\forall\, i \in \{\, 0, 1, \dots, \frac{N}{2k+1} \,\}$, $(N-(2k+1)i, 1, 2k+1)$ is a valid triple. Indeed, the conditions are met:

$$0 \le N - (2k+1)i \le N \qquad 1 | N - (2k+1)i \qquad 2k+1 | (2k+1)i$$

Thus $h(i)$ is well defined, and $|h(i)| \leq 1$. Finally, observe that we chose $T \approx \frac{N}{2}$ so that $\frac{N-T}{2k+1}$ lies approximately halfway between 0 and $\frac{N}{2k+1}$. From there, we can apply fact 5.6 and derive that:

$$\deg(h) = \Omega\left(\frac{N}{2k+1}\right) = \Omega\left(\frac{N}{k}\right)$$

Which, by virtue of the fact that $\deg(Q) \geq \deg(h)$, proves that $\deg(Q) = \Omega\left(\frac{N}{k}\right)$.

In the end there are two cases:

- Either $k \geq N^{2/3}$, in which case $\deg(Q) = \Omega(\sqrt{k}) = \Omega(N^{1/3})$, and we are done.
- Or $k \leq N^{2/3}$, in which case $\deg(Q) = \Omega\left(\frac{N}{k}\right) = \Omega(N^{1/3})$ and we are also done.

This completes the proof of the lower bound for the collision problem in the case where $Q(T,1,2) \leq \frac{1}{2}$. A similar argument involving $Q(T,2,2)$ instead of $Q(T,1,1)$ can be made in the case where $Q(T,1,2) \geq \frac{1}{2}$. □

References

[Aar02] Scott Aaronson. Quantum lower bound for the collision problem. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 635–642. ACM, 2002.

[Amb05] Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1(1):37–46, 2005.

[Amb07] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

[HH13] Daniel Harlow and Patrick Hayden. Quantum computation vs. firewalls. *Journal of High Energy Physics*, 2013(6):1–56, 2013.

[Kut05] Samuel Kutin. Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1(1):29–36, 2005.

[Shi02] Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 513–519. IEEE, 2002.

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

Lecture 13: Lower Bounds using the Adversary Method

October 21, 2015

Lecturer: Ryan O'Donnell *Scribe: Kumail Jaffer*

1 Introduction

There are a number of known methods to show lower bounds in the quantum query model.

(0) Polynomial Method [BBC+01]

(1) Hybrid Method [BBBV97]

(2) Basic Adversary Method [Amb02]

(3) General Adversary Method [HLS07]

So far in this course we've seen (0).

In this lecture, we will develop a basic version of (2), the already basic adversary method. A generalized form of method (2), method (3), is extremely powerful. Indeed, it can be shown that every lower bound shown using this method has a corresponding upper bound (!) [Rei09]

2 The Super-Basic Adversary Method [Amb02]

To start, let's recall the quantum query model for decision problems. We are given a decision problem in the form of a function $F : \{0,1\}^n \to \{0,1\}$, and an N bit input string w, which we can access via an oracle $O_w^{\pm} : |i\rangle \mapsto (-1)^{w_i} |i\rangle$, perhaps with some promise on the form w can take. Our task is to determine $F(w)$ with high probability (say greater than $2/3$) for arbitrary w in the domain of the problem, using as few queries to $O_w^{\pm}$ as possible.

What does a circuit solving such a problem look like? A T-query circuit for such a problem, using a ancilla bits will look as follows.

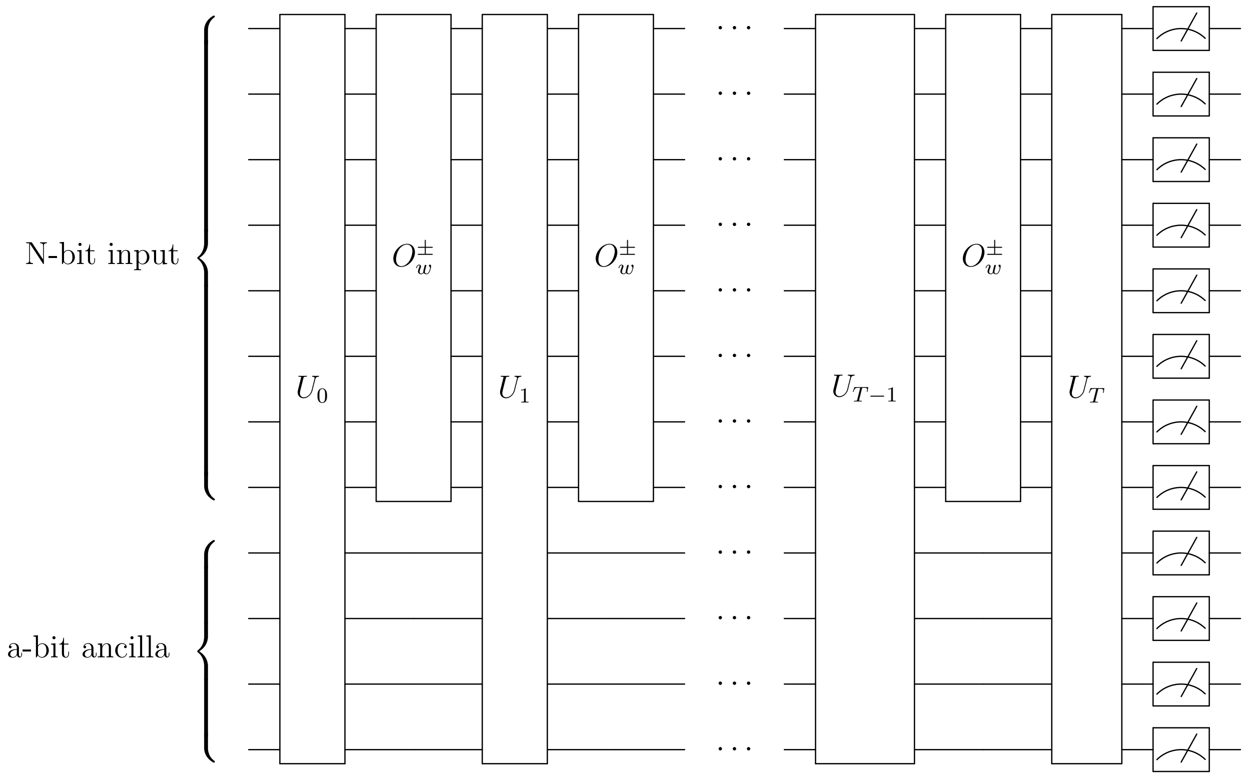


Figure 1: T-query circuit

That is, we call the oracle T times, with some unitary transformations between successive calls, and then measure at the end, and then use the measurement to determine our answer. Since all quantum operations are unitary, we can collapse any number of intermediate quantum gates into one big unitary transformation.

The super high level idea of the adversary method is to show that each call to the oracle can only give us a limited amount of progress toward our goal. More specifically, consider the following progress measure.

Definition 2.1 (Progress Measure)**.** Let $|\psi_w^t\rangle$ be the state of the system after the t'th call to the oracle, and fix some y, z such that $F(y) = 1$, $F(z) = 0$. Then the progress function is $\Phi(t) = \langle\psi_z^t|\psi_y^t\rangle$

Right off the bat we can notice some useful properties

Observation 2.2. Φ *is not affected by the non-oracle unitary transformations.*

This is because unitary transformations preserve inner product by definition. Notice that this does not apply to calls to the oracle, because y and z have different oracles. So it suffices to only examine what happens before and after calls to the oracles.

Observation 2.3. $\Phi(0) = 1$

This is because before application of the first oracle, $|\psi_y^0\rangle$ and $|\psi_z^0\rangle$ are necessarily the same.

Claim 2.4. *If a T-query circuit solves F, $\Phi(T) \leq 0.999$*

Intuitively, $|\psi_y^T\rangle$ and $|\psi_z^T\rangle$ must be sufficiently different in order for us to be able to distinguish them. If their inner product is not sufficiently small, we'll get the same result after measuring them too often to determine our response with high enough probability. More formally, we have the following.

Lemma 2.5. *Let $|\alpha\rangle = |\psi_y^T\rangle, |\beta\rangle = |\psi_z^T\rangle$. Say that C is some circuit solving F. If $|\langle\alpha|\beta\rangle| \geq 1-\epsilon$, then $|\mathbf{Pr}[C(y)=1] - \mathbf{Pr}[C(z)=1]| \leq \sqrt{2\epsilon}$.*

Proof. Assume $|\langle\alpha|\beta\rangle| \geq 1-\epsilon$. Notice that if we multiply α by a scalar with unit norm, the measurement probabilities don't change. So WLOG we can pick some scalar so that $\langle\alpha|\beta\rangle = |\langle\alpha|\beta\rangle| \geq 1-\epsilon$. Now we have

$$\|\alpha-\beta\|_2^2 = \langle\alpha-\beta|\alpha-\beta\rangle = \langle\alpha|\alpha\rangle + \langle\beta|\beta\rangle - 2\mathbf{Re}\,\langle\alpha|\beta\rangle$$

And we know that $\langle\alpha|\beta\rangle = \mathbf{Re}\,\langle\alpha|\beta\rangle \geq 1-\epsilon$, as well as that $\langle\alpha|\alpha\rangle = \langle\beta|\beta\rangle = 1$ since they are valid quantum states. So $\|\alpha-\beta\|_2^2 \leq 2\epsilon$, $\|\alpha-\beta|_2 \leq \sqrt{2\epsilon}$.

Now we think of the outcome of measurement as a probability distribution over strings, where the probability of seeing x when we are in state α is $|\alpha_x|^2$, and similarly for β. We want to show that the total variation distance is at most $\sqrt{2\epsilon}$. This is

$$\begin{aligned}\frac{1}{2}\sum_x \left||\alpha_x|^2 - |\beta_x|^2\right| &= \frac{1}{2}\sum_x \left||\alpha_x| - |\beta_x|\right| \cdot \left||\alpha_x| + |\beta_x|\right| \\ &\leq \frac{1}{2}\sum_x |\alpha_x - \beta_x| \cdot (|\alpha_x| + |\beta_x|) \\ &\leq \frac{1}{2}\sqrt{\sum_x |\alpha_x - \beta_x|^2}\sqrt{\sum_x (|\alpha_x| + |\beta_x|)^2}\end{aligned}$$

The second and third line follow from the triangle inequality, and Cauchy-Schwarz respectively. The left factor is $\|\alpha-\beta\|_2$. Using Jensen's inequality, we know that $(a+b)^2 \leq 2a^2+2b^2$, and since $\sum_x |\alpha_x|^2 = \sum_x |\beta_x|^2 = 1$, the right factor is at most 2. So the whole thing is at most $\|\alpha-\beta\|_2 = \sqrt{2\epsilon}$ and we are done.

□

We can now prove claim 2.4

Proof. $\mathbf{Pr}[C(y)=1] \geq 2/3$ and $\mathbf{Pr}[C(z)=1] < 1/3$, so we can conclude $\sqrt{2\epsilon} \geq 1/3$, which means $\epsilon \geq 1/18$, and $\langle\alpha|\beta\rangle \leq 1-\epsilon \leq 1-1/18 \leq 0.999$. □

Now if we can show that the progress function changes by no more than Δ after each call to the oracle, then necessarily $T \leq 0.001/\Delta$, and we have a lower bound.

Of course, to come up with interesting lower bounds, we have to look at more than just one fixed y and z. Otherwise, since y and z must differ in at least one index (which is fixed since y and z are), all we have to do to differentiate between them is to make one query on the index on which they differ. It turns out that if we look instead at some specially constructed sets $Y \subseteq F^{-1}(1), Z \subseteq F^{-1}(0)$, we can come up with some very useful theorems. Here's a simple one.

Theorem 2.6 (Super-Basic Adversary Method)**.** *Let F be a decision problem, $Y \subseteq F^{-1}(1)$, $Z \subseteq F^{-1}(0)$. For every y we define $Z_y = \{z \in Z : d(y,z) = 1\}$, and for every z, $Y_z = \{y \in Y : d(y,z) = 1\}$. If $\forall y \in Y, |Z_y| \geq m$, and $\forall z \in Z, |Y_z| \geq m'$, then the quantum query complexity of F is $\Omega(\sqrt{mm'})$. Here $d(a,b)$ denotes the hamming distance between strings a and b.*

Before we get into the proof of this, let's apply it to some decision problems we understand in order to get a feel for it.

3 Illustrations

3.1 Decision Grover

In the decision version of the Grover search problem, we are given an n bit string and we are tasked with determining whether it is all 0 or not. That is, we'll return 0 if it is all zero, and 1 if not.

Let's look at $Y = \{x : x \text{ contains exactly one } 1\}$, and $Z = \{00000\ldots\}$. Notice Y has size N.

For all $y \in Y$, the all zero string is hamming distance 1 away, so $\forall y \in Y, |Z_y| = 1$. Similarly, since all the elements in Y are hamming distance 1 away from the all zero string, $\forall z \in Z, |Y_z| = N$.

So by the adversary method, we have a lower bound of $\Omega(\sqrt{N})$.

3.2 Variation on Decision Grover

In this variation of the decision Grover problem, we are again given an n bit string. We are tasked with differentiating between inputs with hamming weight $\geq k$, and those with hamming weight $\leq k-1$. We return 1 in the first case and 0 in the second.

Let's look at $Y = \{x : x \text{ has hamming weight } k\}$ and $Z = \{x : x \text{ has hamming weight } k-1\}$. It is easy to see that $\forall y \in Y, |Z_y| = k$, since for any y we can flip each of its k 1s to get something in Z). Similarly, $\forall z \in Z, |Y_z| = N-k+1$, since we for any z we can flip each of its $N-k+1$ 0s to get something in Y).

Using the adversary method, we have a lower bound of $\Omega(\sqrt{k(N-k+1)})$. If we make the additional assumption that $k \leq N/2$, then this is $\Omega(\sqrt{kN})$. Making this assumption is okay, since if $k > N/2$, then we have the symmetric situation of differentiating between at least $N - k$ zeros and less than $N - k$ zeros, where $N - k \leq N/2$, and so our lower bound still applies.

3.3 Read-once CNF

In this problem, we are given an N bit string and are told to interpret it as a conjunctive normal form expression with $\sqrt{N}$ clauses and $\sqrt{N}$ variables per clause. So each bit is the value of a variable, and each set of $\sqrt{N}$ bits is to be interpreted as a clause. The task is to calculate the result.

Consider $Y = \{x : x$ has exactly one 1 per clause$\}$ and $Z = \{x : x$ has all zeros in one clause, and exactly one 1 in all the others $\}$. $\forall y \in Y, |Z_y| = \sqrt{N}$, since we can take any y and, for each of its clauses, flip the sole 1 to get something in Z. Similarly, $\forall z \in Z, |Y_z| = \sqrt{N}$, since for any z, we can flip each zero in its all zero clause to get something in Y.

Applying the adversary method, we get a lower bound of $\Omega(N)$. This lower bound in particular is very hard to show using the polynomial method, though it is nearly trivial to find with the adversary method.

4 Proof of Super-Basic Adversary Method

Proof. First of all, we need to rewrite our progress measure so that we can make use of multiple ys and zs.

Definition 4.1 (More Useful Progress Measure)**.** Let R be a binary relation defined as $R = \{(y, z) \in Y \times Z : d(y, z) = 1\}$. Then the progress measure $\Phi(t)$ is defined as $\Phi(t) = \sum_{(y,z)\in R} | \langle \psi_y^t | \psi_z^t \rangle |$.

Observation 4.2. *From our prior observations in section 2, we can immediately conclude* $\Phi(0) = |R|$, $\Phi(T) \leq 0.999|R|$, *and that, since unitary transformations don't change our progress measure, we need only concern ourselves with the effect of the oracle.*

Definition 4.3. For each $y \in Y$ let $I_y = \{i : y$ with i'th bit flipped $\in Z\}$.

Observation 4.4. *Notice that* $|I_y| \geq m$*. As a result,* $|R| \geq m|Y|$*.*

With notation out of the way, let's begin in earnest. We will show that

$$\Phi(t-1) - \Phi(t) \leq \frac{2|R|}{\sqrt{mm'}}$$

This will let us conclude that $T \in \Omega(\sqrt{mm'})$.

Fix some t, and consider some particular $(y, z) \in R$. Let $i^* \in [N]$ be the position on which y and z differ. Before the t'th oracle, a circuit solving the problem will be in some quantum state $|\psi_y^{t\prime}\rangle$ for y and $|\psi_z^{t\prime}\rangle$ for z. That looks something like this.

$$|\psi_y^{t\prime}\rangle = \sum_i \alpha_i |i\rangle \otimes |\phi_i\rangle$$
$$|\psi_z^{t\prime}\rangle = \sum_i \beta_i |i\rangle \otimes |\chi_i\rangle$$

Where $|\phi_i\rangle, |\chi_i\rangle$ are unit vectors, $\sum_i |\alpha_i|^2 = 1$ and $\sum_i |\beta_i|^2 = 1$. Their inner product is

$$\langle\psi_y^{t\prime}|\psi_z^{t\prime}\rangle = \sum_i \alpha_i\beta_i \langle\phi_i|\chi_i\rangle$$

Now if we pass these states through the oracle to get $|\psi_y^t\rangle$ and $|\psi_z^t\rangle$, what it does is flip the sign on each α_i or β_i, whenever $y_i = 1$ or $z_i = 1$ respectively. The only index on which the sign will differ is i^*, so our new inner product is exactly

$$\langle\psi_y^t|\psi_z^t\rangle = \sum_{i \neq i^*} \alpha_i\beta_i \langle\phi_i|\chi_i\rangle - \alpha_{i^*}\beta_{i^*} \langle\phi_{i^*}|\chi_{i^*}\rangle$$

The difference between the inner products is then

$$\langle\psi_y^{t\prime}|\psi_z^{t\prime}\rangle - \langle\psi_y^t|\psi_z^t\rangle = 2\alpha_{i^*}\beta_{i^*} \langle\phi_{i^*}|\chi_{i^*}\rangle$$

The quantity we want to bound is

$$\begin{aligned}
\Phi(t-1) - \Phi(t) &= \sum_{(y,z)\in R} |\langle\psi_y^{t\prime}|\psi_z^{t\prime}\rangle| - \sum_{(y,z)\in R} |\langle\psi_y^t|\psi_z^t\rangle| \\
&\leq \sum_{(y,z)\in R} |\langle\psi_y^{t\prime}|\psi_z^{t\prime}\rangle - \langle\psi_y^t|\psi_z^t\rangle| \\
&\leq \sum_{(y,z)\in R} |2\alpha_{i^*_{(y,z)}}\beta_{i^*_{(y,z)}} \langle\phi_{i^*_{(y,z)}}|\chi_{i^*_{(y,z)}}\rangle| \\
&\leq \sum_{(y,z)\in R} 2|\alpha_{i^*_{(y,z)}}||\beta_{i^*_{(y,z)}}| \\
&\leq \sum_{(y,z)\in R} \sqrt{\frac{m}{m'}}|\alpha_{i^*_{(y,z)}}|^2 + \sum_{(y,z)\in R} \sqrt{\frac{m'}{m}}|\beta_{i^*_{(y,z)}}|^2
\end{aligned}$$

The last line follows from Jensen's inequality. Consider just the left summand. This is the same as

$$\begin{aligned}
\sqrt{\frac{m}{m'}} \sum_{(y,z)\in R} |\alpha_{i^*_{(y,z)}}|^2 &= \sqrt{\frac{m}{m'}} \sum_{y\in Y} \sum_{i\in I_y} |\alpha_i|^2 \\
&\leq \sqrt{\frac{m}{m'}} |Y| \\
&\leq \sqrt{\frac{m}{m'}} \frac{|R|}{m} = \frac{|R|}{\sqrt{mm'}}
\end{aligned}$$

The second line comes from the fact that $\sum_i |\alpha_i|^2 = 1$, and the third from Observation 4.4. The right summand is a symmetric situation, and so overall we get

$$\Phi(t-1) - \Phi(t) \leq \frac{2|R|}{\sqrt{mm'}}$$

as promised, concluding the proof. □

5 Generalization

This isn't quite as powerful as you might hope. In particular, it fails to give us a good lower bound when we can't find sets Y and Z which have many strings hamming distance 1 away from each other. As motiviation consider the problem of distinguishing between strings of hamming weight 0 and strings of hamming weight greater than k. In this situation, there are no $y \in F^{-1}(1), z \in F^{-1}(0)$ such that $d(y,z) = 1$.

One natural thing to do is to expand our relation so we're not restricted to just strings hamming distance 1 from each other. This line of reasoning results in the following theorem.

Theorem 5.1 (Basic Adversary Method)**.** *Let F be a decision problem, $Y \subseteq F^{-1}(1), Z \subseteq F^{-1}(0)$, and a binary relation $R \subseteq Y \times Z$. If*

1. *$\forall y \in Y$ there are m distinct $z \in Z$ such that $(y,z) \in R$*
2. *$\forall z \in Z$ there are m' distinct $y \in Y$ such that $(y,z) \in R$*
3. *$\forall y \in Y, i$ there are at most l distinct $z \in Z$ such that $y_i \neq z_i$ and $(y,z) \in R$.*
4. *$\forall z \in Z, i$ there are at most l' distinct $y \in Y$ such that $y_i \neq z_i$ and $(y,z) \in R$.*

then the quantum query complexity of F is $\Omega(\sqrt{\frac{mm'}{ll'}})$.

The proof for this is very similar to the one we just did. In later lectures, we will dive deeper into further generalizations.

References

[Amb02] A. Ambainis. Quantum Lower Bounds by Quantum Arguments. *Journal of Computer and System Sciences*, 64(4):750 – 767, 2002.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, October 1997.

[BBC$^+$01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, July 2001.

[HLS07] P. Hoyer, T. Lee, and R. Spalek. Negative Weights Make Adversaries Stronger. In *Proceedings of the 39th ACM Symposium on the Theory of Computing. ACM*, pages 526–535, November 2007.

[Rei09] B. W. Reichardt. Span Programs and Quantum Query Complexity: The General Adversary Bound is Nearly Tight for Every Boolean Function. In *Foundations of Computer Science, 2009. FOCS '09. 50th Annual IEEE Symposium on*, pages 544–551, October 2009.

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

Lecture 14: Reichardt's Theorem I: Definition of Span Programs

October 26, 2015

Lecturer: Ryan O'Donnell *Scribe: Will Griffin*

1 REVIEW: Super-Basic Adversary Method

In the Super-Basic Adversary Method, we take some decision problem $F : \{0,1\}^N \to \{0,1\}$ and define two sets, $Y \subseteq F^{-1}(1)$ and $Z \subseteq F^{-1}(0)$.
Then we define a relation $R \subseteq Y \times Z, R = \{(y,z) : d(y,z) = 1\}$, so R is the set of pairs of $y \in Y, z \in Z$ such that the distance between y and z is 1, so y is only one bit different from z.
Then let $R(y)$ be the set of all y strings in the pairs in R, $R(z)$ the set of all z strings in R, and define m, m' such that:
$m \leq |R(y)|$
$m' \leq |R(z)|$
Then F requires $\Omega(\sqrt{mm'})$ quantum queries.

Unfortunately, this does not provide a good lower bound for many algorithms.

2 Basic Adversary Method

The idea is the same, but the relation between strings in Y and Z is different. Instead of being restricted to picking Y, Z where the Hamming weights of strings in Y and Z differ by 1, choose an arbitrary relation $R \subseteq Y \times Z$.

Theorem 2.1. *Basic Adversary Method:*
For some decision problem $F : \{0,1\}^N \to \{0,1\}$, let $Y \subseteq F^{-1}(1)$ and $Z \subseteq F^{-1}(0)$, and let $R \subseteq Y \times Z$ be a binary relation. If:

$\forall y \in Y$ there exist at least m distinct $z \in Z$ such that $(y,z) \in R$

$\forall z \in Z$ there exist at least m' distinct $y \in Y$ such that $(y,z) \in R$

$\forall y \in Y$, $i \in \{0,1,2,\ldots,N\}$ there are at most l $z \in Z$ such that $(y,z) \in R$ and $y_i \neq z_i$

$\forall z \in Z$, $i \in \{0,1,2,\ldots,N\}$ there are at most l' $y \in Y$ such that $(y,z) \in R$ and $y_i \neq z_i$

Then F requires $\Omega\left(\sqrt{\frac{mm'}{ll'}}\right)$ quantum queries.

The proof is similar to the proof for the super-basic adversary method, and can be found in [HLS07].

Example: Given oracle access to $w \in \{0,1\}^N$, we want to distinguish between strings with Hamming weight 0 or Hamming weight greater than k. First choose the sets Y and Z: let Z be the string of all 0 bits, and Y the set of strings with Hamming weight k, and let $R = Y \times Z$. Then $m = 1$, $m' = \binom{N}{k}$ and $l = 1$. Since the string in Z is all $0s$, l' is the number of strings in Y that have a 1 bit on coordinate i, which is the number of strings of length $N-1$ with Hamming weight $k-1$, so $l' = \binom{N-1}{k-1}$. Then the lower bound is $\sqrt{\frac{\binom{N}{k}}{\binom{N-1}{k-1}}} = \sqrt{\binom{N}{k}}$.
Another example of a lower bound that can be found with this method is graph connectivity. Given query access to the adjacency matrix of a graph G with n vertices, the basic adversary method can show that checking if G is connected requires $n^{3/2}$ queries. This is also the upper bound, which you may have shown in homework 3.

3 Weighted Adversary Method

The basic adversary method does not always give optimal lower bounds for algorithms. This leads to the next generalization: the Weighted Adversary Method. It is similar, and based on the same idea - define the sets Y and Z in the same way, and use a relation R between the elements, but this time, (y, z) pairs in R are given different weights in a weight matrix. Harder to distinguish pairs are given higher weights to get good results.

Definition 3.1. Let Γ be a weight matrix such that $\Gamma[y,z]$ is the weight of (y,z), and $\Gamma[y,z] \in \mathbb{R} \geq 0$, and $\Gamma[y,z] = 0$ if and only if $F(y) = F(z)$. Γ should be symmetric.

Let $\parallel \Gamma \parallel$ be the maximum of the absolute values of the eigenvalues of Γ. $\parallel \Gamma \parallel$ is the analogue to $\sqrt{mm'}$ in the basic adversary method.

There is also an analog of the query distinguishability, $\sqrt{ll'}$, in the basic adversary.
For $i \in \{0, 1, \ldots n\}$ let Γ_i be the zero-out of Γ except for those $[y,z]$ entries where $y_i \neq z_i$. Then $max\,\{\parallel \Gamma_i \parallel, i \in 0, \ldots, n\}$ is the analog of $\sqrt{ll'}$.

Theorem 3.2. *Quantum Query lower bound for F is $\frac{\|\Gamma\|}{max\{\|\Gamma_i\|, i \in 0,\ldots,n\}}$ [HLS07]*

This proof is more difficult.
Fun Fact: for 2:1 collision problem, this method gives a constant time lower bound.
Other Fun Fact: After the weighted adversary method, there were several different new

adversary methods developed, that appear different. They were all proven to be essentially equivalent, see [SS06]. This suggests that the general adversary method is very robust and powerful.

4 Negative Weights Adversary Method

Exactly the same as the Weighted Adversary method, but allow the weights $\Gamma[y, z]$ to take on negative values. See [HLS07]
Let $ADV^{\pm}(F)$ be the best lower bound achievable using the Negative Weights Adversary Method.
Fun Fact: Given the truth table of F, $ADV^{\pm}(F)$ is computable in polynomial time in the size of the truth table of F using semi-definite programming.

5 Reichardt's Theorem

Theorem 5.1. *Reichardt's Theorem: Quantum Query complexity of F is $ADV^{\pm}(F)$, so the Negative Weights Adversary method gives an upper and lower bound for the quantum query complexity fo F. [Rei09]*

Proof. (Not really. All the details are in next lecture.) $ADV^{\pm}(F)$ is a maximization problem, since it is trying to find the weight matrix that gives the highest lower bound. This can be seen as an SDP (semi-definite program). This allows it to be transformed into the dual of the SDP, which is a minimization problem. See: http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15859
f11/www/notes/lecture12.pdf for notes on SDPs.
Reichardt observed that finding the dual of the SDP is interpretable as finding the lowest complexity Span Program that computes F. This is the easy part. Then Richardt showed that any Span Program for F of complexity T can be converted into a quantum query algorithm for F using $O(T)$ queries. In other words, he showed that a quantum algorithm could evaluate the span program in $\emptyset(T)$ queries (actually it uses $O(T)log(T)$, so it's tight within a logarithmic factor -[Rei09]). This is the hard part of the proof. □

6 Span Programs

Span programs are a linear-algebraic model of computation from before quantum computation was a real topic though you need a different definition for their complexity for quantum computation than for classical.
As Reichardt showed, we can design quantum algorithms by designing span programs, and without loss of generality, these can be optimal.
In these notes, span programs are defined for decision functions $F : \{0,1\}^N \rightarrow \{0,1\}$, but they can be more generally defined [KW93].
Definition: A Span Program P operating on strings $w \in \{0,1\}^N$ consists of a bunch of real

input vectors in $\mathbb{R}^d$ grouped into $2N$ sets, named $w_1 = 0, w_1 = 1, w_2 = 0, w_2 = 1, \ldots, w_N = 0, w_N = 1$, plus a target vector $\tau \in \mathbb{R}^d$.
How P computes:
First it partitions the $2N$ sets of vectors into 2 new sets, *Avail*, and *Unavail*, each of size N based on w: For each $i = 0, 1, \ldots N$, if $w_i = 0$, the set named $w_i = 0$ goes into *Avail* and the set $w_i = 1$ goes into *Unavail*, and if $w_i = 1$, $w_i = 1$ goes into *Avail* and $w_i = 0$ into *Unavail*. Then $P(w) = 1$ if τ is in the span of the vectors in *Avail*. We say P computes F if $P(w) = F(w)$ on all inputs.
Example: Majority for 3 bits:
$d = 3$
P:
$w_1 = 0 : \emptyset$
$w_1 = 1 : \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}$
$w_2 = 0 : \emptyset$
$w_2 = 1 : \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}$
$w_3 = 0 : \emptyset$
$w_3 = 1 : \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}$
$T = \begin{pmatrix}1\\1\\1\end{pmatrix}$
Since any two of the $w_i = 1$ sets form a basis for $\mathbb{R}^3$, but none of them individually contain T in their span, $P(w)$ will compute majority.
Example 2: OR (Search)
$d = 1$
P:
$w_i = 0 : \emptyset \forall i$
$w_i = 1 : \begin{pmatrix}1\end{pmatrix} \forall i$
$\tau = (() \, 1)$

7 Span Program Complexity for Quantum Algorithms

Given a span program P, let I be the set of its input vectors. Then let V be the matrix formed by all the vectors in I:

$$V = \left(\begin{pmatrix} V_{11} \\ \vdots \\ V_{1d} \end{pmatrix} \cdots \begin{pmatrix} V_{|I|1} \\ \vdots \\ V_{|I|d} \end{pmatrix} \right)$$

I is partitioned into $Avail(w)$ and $Unavail(w)$ as described before.
Suppose P computes a function F.
Definition: A *Positive Witness* for some $y \in F^{-1}(1)$ is a linear combination of vectors in $Avail(y)$ that make τ. We can write this as a column vector $\alpha \in \mathbb{R}^{|I|}$ such that $V\alpha = \tau, \alpha_i = 0 \forall i \in Unavail(z)$.
Definition: A *Negative Witness* for some $z \in F^{-1}(0)$ is a row vector $\beta \in \mathbb{R}^d$ such that $\langle \beta, V_i \rangle = 0 \forall i \in Avail(z)$ and $\langle \beta, \tau \rangle = 1$.
The negative witness works because it forces β to be in the nullspace of the available vectors, so no linear combination of them can ever form β, but since $\langle \beta, \tau \rangle = 1$ those vectors would need to form β to have a linear combination of them form τ. Forcing $\beta = 1$ is important for the complexity definition, since otherwise we could make the Positive Witness size arbitrarily small by multiplying τ by a small constant. Forcing this requirement on β ensures that the complexity is independent of constant multiplication of τ.

Definition 7.1. An *Extended Span Program ESP* is some span program P and a witness for every input, so:
$ESP = (P, (\alpha_y)_{y \in F^{-1}(1)}, (\beta_z)_{z \in F^{-1}(0)})$

Definition 7.2. The size of a positive witness α is: $size(\alpha) = |\alpha|^2$

Definition 7.3. The size of a negative witness β is $size(\beta) = \sum_{i=1}^{|I|} |\langle B, V_i \rangle|^2$.

Let $T_1 = max\{size(\alpha_y)\}, y \in F^{-1}(1)$
Let $T_0 = max\{size(\beta_z)\}, z \in F^{-1}(0)$

Definition 7.4. The complexity T of P is $\sqrt{T_0 T_1}$.

Theorem 7.5. *Given an extended span program $(P, (\alpha_y)_{y \in F^{-1}(1)}, (\beta_z)_{z \in F^{-1}(0)})$ for a function F, there is a quantum algorithm for F making $O(T)$ queries.*
Also, the minimum T over all span programs computing F = "dual SDP" = $ADV^{\pm}(F)$, so the best possible span program gives the best possible quantum algorithm. [Rei09]

Example: Majority for 3 bits:

$$V = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

For $w = 111$, the best α is $\begin{pmatrix} 0.5 \\ \vdots \\ 0.5 \end{pmatrix}$, and its size is $6 * 0.5^2 = 1.5$.

For $w = 110$, the last two columns of V are unavailable, so $\alpha = \begin{pmatrix} 1 \\ 0.5 \\ 0.5 \\ 1 \\ 0 \\ 0 \end{pmatrix}$ gives the best value, with $size(\alpha) = 2.5$, and $w = 101$ or 011 gives the same value by permuting the rows in α. Since these are all the positive witnesses, $T_1 = 2.5$.
For $w = 100$, we need a negative witness. Since there are 2 of the 3 standard basis vectors of $\mathbb{R}^3$ in $Avail(w)$, the only choice is the third basis vector, $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, which satisfies $\langle \beta, \tau \rangle = 1$, and has size 2, and the same bound holds for the other strings with Hamming weight 1.
For $w = 000$, any β with $\langle \beta, \tau \rangle$ works, and several choices give a size of 2 (or example, all values $1/\sqrt{3}$, or the β from $w = 100$) and since this matches the best bound bound from the other negative witness, $T_0 = 2$.
Then $T = \sqrt{T_1 T_0} = sqrt5$.
Example: OR function (unstructured search)
For any input string y with $F(y) = 1$, $\alpha_y = \begin{pmatrix} 0 \vdots 1 \vdots 0 \end{pmatrix}$, with the 1 on any coordinate with $y_i = 1$, so $size(\alpha) = 1$, and $T_1 = 1$.
The only possible negative witness is $\beta = 1$, and $\langle \beta, V_i \rangle = 1$ for N vectors in V, since V has N 1 vectors and nothing else, so $T_0 = N$.
Then the quantum query complexity for unstructured search is (SURPRISE!) $\sqrt{N}$.

References

[HLS07] P. Hoyer, T. Lee, and R Spalek. Negative weights make adversaries stronger. *Proceedings of the 39th ACM Symposium on the Theory of Computing*, pages 526–535, November 2007.

[KW93] M. Karchmer and A. Wigderson. On span programs. *Proceedings of the 8th IEEE Structure in Complexity Theory*, pages 102–111, 1993.

[Rei09] Ben W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. *Foundations of Computer Science, 2009*, pages 544–551, October 2009.

[SS06] R. Spalek and M. Szegedy. All quantum adversary methods are equivalent. *Theory of Computing, Volume 2 (2006)*, pages 1–18, January 2006.

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

Lecture 15: Reichardt's Theorem II: Evaluation of Span Programs

October 28, 2015

Lecturer: Ryan O'Donnell *Scribe: Vikesh Siddhu*

1 Short Summary

We have discussed in the past, general methods to lower bound the quantum query complexity. Now we discuss a way to convert the lower bounds given by the general adversary method [HLS07] into an upper bound [Rei09]. In this lecture we will cover the proof that leads to this result. Two key ingredients are Span Programs and how one defines their complexity.

2 Recall Span Programs

Let $w \in \{0,1\}^N$ be a string and $F : \{0,1\}^N \mapsto \{0,1\}$ be a function. A span program computes P for given w. Let $\{|v_i\rangle\}_{i=1}^m$ be vectors in $\mathbb{R}^d$ that are columns of a matrix V and let $|\tau\rangle \in \mathbb{R}^d$ be another vector called *target*. V is split into $2N$ blocks, the $2k^{th}$ and $2k^{th}+1$ block each have vectors corresponding to $w_k = 0$ and $w_k = 1$ resp.

Given w the span program makes available some N blocks, call the these set of vectors in the block avail(w) and the rest unavail(w). For example $w_1 = 0, w_2 = 1, \ldots, w_N = 1$ makes available the blocks $1, 4, \ldots 2N$ and the rest $2, 3, \ldots 2N-1$ become unavailable. Given a span program the function $P(w) = 1$ iff $|\tau\rangle \in \text{span}\{|v_i\rangle : |v_i\rangle \in \text{avail}(w)\}$

Suppose P computes $F : \mathcal{D}^{\subseteq\{0,1\}^N} \mapsto \{0,1\}$, then

- For $y \in F^{-1}(1)$, a *positive witness* is $|\alpha\rangle \in \mathbb{R}^m$ s.t.

$$\alpha_i = 0 \quad \forall i \in \text{unavail}(y) \tag{1}$$
$$V|\alpha\rangle = |\tau\rangle \tag{2}$$

  we define it's *size* to be $||\,|\alpha\rangle\,||^2$.

- For $z \in F^{-1}(0)$, a *negative witness* is $\langle\beta| \in \mathbb{R}^d$ s.t.

$$\langle\beta\,|\,v_i\rangle = 0 \quad \forall i \in \text{avail}(y) \tag{3}$$
$$\langle\beta\,|\,\tau\rangle = 1 \tag{4}$$

  we define it's *size* to be $||\,\langle\beta|\,V||^2$.

An *extended span program* is a span program along with positive and negative witnesses for all possible inputs w. We can define the complexity of an extended span program as follows

- Let $|\alpha_y\rangle$ be a positive witness for $y \in F^{-1}(1)$ then the YES complexity is defined as $T_1 \equiv \max\limits_{y\in F^{-1}(1)} \{\text{size}(|\alpha_y\rangle)\}$.
- Let $|\beta_y\rangle$ be a negative witness for $y \in F^{-1}(0)$ then the NO complexity is defined as $T_0 \equiv \max\limits_{y\in F^{-1}(0)} \{\text{size}(|\beta_y\rangle)\}$
- The overall complexity of the span program is $T = \sqrt{T_0 T_1}$

3 Reichardt's Theorem II

Theorem 3.1. *If a span program P with complexity T computes F, then there exists a quantum query algorithm for F making $O(T)$ queries of the oracle $O_f^{\pm}$.*

Fact 3.2. *The complexity T for the span program P to compute the function F is equal to the adversary lower bound $Adv^{\pm}(F)$*

Example 3.3. *Let $F = OR_N$ i.e. OR on N bits. Define a span program with vectors such that, for $w_i = 1$, the block in V has one vector $|v_i\rangle = [1]$ and the for $w_i = 0$, the block in V a null vector $|v_i\rangle = \phi$. Then*

1. *YES complexity $T_1 = 1$*
2. *NO complexity $T_0 = N$*
3. *The overall complexity is $T = \sqrt{T_0 T_1} = \sqrt{N}$*

We now come to the proof of Theorem 3.1.

Proof. Let $|\tilde{\tau}\rangle = \frac{|\tau\rangle}{c\sqrt{T_1}}$ and define $\tilde{V} \in \mathbb{R}^{d\times m+1}$ as

$$\tilde{V} = \left[\ |\tilde{\tau}\rangle \mid V\ \right] \tag{5}$$

For the Grover case $\tilde{V} = [\frac{1}{c}\ 1\ 1\ \ldots 1]$.

For now the algorithm will work in the $\mathbb{R}^{m+1}$ space and any intermediate state is given by a vector $|s\rangle = \sum_{i=0}^{m} \alpha_i |i\rangle$ where each $\langle i \mid j\rangle = \delta_{ij}$. Define

$$K = \ker(\tilde{V}) = \{|u\rangle \in \mathbb{R}^{m+1} \mid \tilde{V}\,|u\rangle = 0\} \tag{6}$$

For the Grover case K consists of all vectors of mean 0.

Define R_K to be the reflection through K. Then R_K is a unitary operator on reals, i.e. an orthogonal matrix. For the Grover case R_K flips a vector in $\mathbb{R}^{m+1}$ across its mean.

Given $w \in \{0,1\}^N$, let

$$A_w = \text{span}\{|i\rangle\ \ 0 \le i \le m\ i \in \text{avail}(w)\} \tag{7}$$

by definition $|\tilde{\tau}\rangle \in A_w$ and is always available. Let R_{A_w} be the reflection through A_w, which mean we negate all the entries of a vector in $\mathbb{R}^{m+1}$ that are at the unavailable coordinates, so $R_{A_w} = -O_w^{\pm}$.

Let $U = R_{A_w} R_K$, computing R_K is a 0 query step and computing R_{A_w} takes 1 query (well 2 query if you un-compute the garbage).

We now describe a fake algorithm that to give some intuition behind how computes F on w using $O(T)$ queries.

1. Initialize the state $|\psi\rangle = |0\rangle$
2. For $t = 1, 2, \ldots CT$ apply U to $|\psi\rangle$
3. Measure $|\psi\rangle$ in standard basis, output 1 iff you observe $|0\rangle$

The basic idea is that

(i) If w is a YES instance, then U fixes $|0\rangle$

(ii) If w is a NO instance, then $U^{CT}|\psi\rangle$ is far from $|0\rangle$

The **first idea** can also be stated as, if $y \in F^{-1}(1)$ then $|0\rangle$ is 99% in K and A_y, hence U fixes 99% of $|0\rangle$.

Fact 3.4. *The accurate fact is $\exists\ |\eta\rangle$ of length $\leq .01$ s.t. $|0\rangle - |\eta\rangle$ is an eigen vector of U.*

Proof. Let $|\alpha_y\rangle$ be a YES instance, let $|\eta\rangle = \frac{|\alpha_y\rangle}{c\sqrt{T_1}}$, we know $||\ |\alpha_y\rangle\ ||^2 \leq T_1$ which implies, for $c \geq 100$

$$\sqrt{\langle \eta \mid \eta \rangle} \leq \frac{1}{c} \leq .01 \tag{8}$$

$U = R_{A_y} R_K$ where R_K is the reflection through $K = \ker(\tilde{V})$ and R_{A_y} is the reflection through A_w. Notice

1. $(|0\rangle - |\eta\rangle)$ is in the kernel of $\tilde{V}$ so $R_K(|0\rangle - |\eta\rangle) = (|0\rangle - |\eta\rangle)$

$$\begin{aligned} \tilde{V}(|0\rangle - |\eta\rangle) &= |\tilde{\tau}\rangle - \frac{1}{c\sqrt{T_1}}\tilde{V}|\alpha_y\rangle & (9)\\ &= |\tilde{\tau}\rangle - \frac{1}{c\sqrt{T_1}}|\tau\rangle & (10)\\ &= |\tilde{\tau}\rangle - |\tilde{\tau}\rangle & (11)\\ &= 0 & (12) \end{aligned}$$

2. $(|0\rangle - |\eta\rangle)$ is in A_y because by definition $|0\rangle \in A_y$ and $|\eta\rangle \propto |\alpha_y\rangle$ and $|\alpha_y\rangle$ is in A_y, so $R_{A_y}(|0\rangle - |\eta\rangle) = (|0\rangle - |\eta\rangle)$

Hence U fixes $|0\rangle - |\eta\rangle$ □

The **second idea** states, if $z \in F^{-1}(0)$ then $|0\rangle$ is far from states fixed by U.

Fact 3.5. *If $w \in F^{-1}(0)$ then $\exists\ |u\rangle$ s.t. $Proj_{A_w}(|u\rangle) = |0\rangle$ and $||\,|u\rangle\,|| \leq 2cT$ and $|u\rangle \perp K$.*

Proof. Let $\langle\beta_w|$ be a NO witness for w, define

$$\langle u| \equiv c\sqrt{T_1}\,\langle\beta_w|\,\tilde{V} \tag{13}$$

Clearly $\tilde{V}\,|u\rangle \neq 0$, hence $|u\rangle \perp \ker(\tilde{V}) \implies |u\rangle \perp K$. Rewrite $|u\rangle$ as follows

$$\langle u| = c\sqrt{T_1}\{\langle 0|\,\langle\beta_w\,|\,\tilde{\tau}\rangle + \langle\beta_w|\,V\} \tag{14}$$
$$= \langle 0| + c\sqrt{T_1}\,\langle\beta_w|\,V \tag{15}$$

where the second equality follows from eq. (4) which states $\langle\tau\,|\,\beta_w\rangle = 1$ and $|\tilde{\tau}\rangle = \frac{|\tau\rangle}{c\sqrt{T_1}}$. Notice

(a) $\langle\beta_w|\,V$, the second term in the rhs of eq. (15) is a linear combination of unavailable vectors (since $|\beta_w\rangle$ is orthogonal to all available vectors)

(b) $||\,\langle\beta_w|\,V||^2 \leq T_0$ (since *size* of $\langle\beta_w|$ is at most T_0)

Lets switch back to kets $|u\rangle = [\langle u|]^\dagger$ where † is the conjugate-transpose (since everything is real here, it is just the transpose). Using (a) we conclude $\text{Proj}_{A_w}\,|u\rangle = |0\rangle$, using (b) we conclude

$$||\,|u\rangle\,|| \leq \sqrt{1 + c^2 T_0 T_1} \leq 1 + c\sqrt{T_0 T_1} \leq 2cT \tag{16}$$

□

Another key idea is the Kitaev Phase Estimation, which we shall delve into a little later. Before going further we review a few facts about orthogonal matrices from the 19th century. Let $U \in \mathbb{R}^{m\times m}$ then

- U offers a decomposition of $\mathbb{R}^m$ s.t

$$R_m = \underset{\text{1 dim spaces where } U \text{ is } \mathbb{I}}{H_1 \oplus H_2 \ldots \quad \ldots} \quad \underset{\text{1 dim spaces where } U \text{ is } -\mathbb{I}}{H_k \oplus H_{k+1} \ldots \quad \ldots} \quad \underset{\text{2 dim spaces where } U = R(\theta)}{H_r \oplus H_{r+1} \oplus \ldots} \tag{17}$$

  where $R(\theta)$ is a $2-D$ rotation by $\theta \in (-\pi, \pi]$. In other words, there are eigen spaces of U with eigen value $+1$ (the identity spaces), -1 (the reflection space) and $e^{i\theta}$ (the 2-d rotation space)

- Let A, B be subspaces of $\mathbb{R}^m$ and R_A, R_B be reflection through these spaces, construct $U = R_A R_B$. Let H be a 2-d θ rotation subspaces of U, then it is true, that $H \cap A$ and $H \cap B$ are 1 dimensional subspaces of $\mathbb{R}^m$ and the angle between $H \cap A$ and $H \cap B$ is $\theta/2$

  Lemma 3.6. *Suppose $|u\rangle \in H$, $u \perp (H \cap B)$ then $||proj_{A\cap H}(|u\rangle)|| \leq \frac{|\theta|}{2}||\,|u\rangle\,||$*

  Proof. Using Figure (1) we see that $||\text{proj}_{A\cap H}(|u\rangle)|| = \sin\frac{\theta}{2}||\,|u\rangle\,|| \leq \frac{\theta}{2}||\,|u\rangle\,||$ □

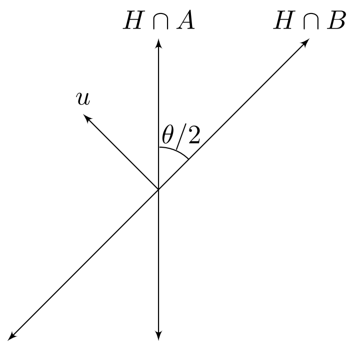


Figure 1: Intersecting subspaces $H \cap A$, $H \cap B$

Corollary 3.7. *Let P_δ be the projection onto all 2-d rotation subspaces of U with angle $\theta \leq \delta$, then*

$$||P_\delta[Proj_A(|u\rangle)]|| \leq \frac{\delta}{2}|| \, |u\rangle \, || \tag{18}$$

Proof. Apply lemma (3.6) subspace by subspace to $\text{Proj}_A(|u\rangle) = |v\rangle$ where it is given that $|u\rangle \perp B$. □

We now make the **second idea** precise. If $w \in F^{-1}(0)$ then $|0\rangle$ is far from states fixed by U. Recall $U = R_{A_w}R_K$ and $w \in F^{-1}(0) \implies \text{Proj}_A(|u\rangle) = |0\rangle$. Since $|0\rangle \perp K$ we use Corollary 3.7 and write

$$||P_\delta \, |0\rangle \, || \leq \frac{\delta}{2}|| \, |u\rangle \, || \leq \delta c T \tag{19}$$

where the final inequality follows from eq. (16). By setting $\delta = \frac{1}{CT}$ and $\frac{c}{C} \leq 100$ we get

$$||P_\delta \, |0\rangle \, || \leq .01 \tag{20}$$

In essence we have shown

- When $w \in F^{-1}(0)$ then $||P_\delta \, |0\rangle \, || \leq .01$
- When $w \in F^{-1}(1)$ then $||P_0 \, |0\rangle \, || \geq .99$, where P_0 is a projection onto the $+1$ eigen space of U.

In order to distinguish whether $w \in F^{-1}(0)$ or $w \in F^{-1}(1)$, we must be able to tell whether $|0\rangle$ is 99% in U's rotation 0 eigen space or $|0\rangle$ is only $\leq 1\%$ in U's rotation $\leq \delta$ subspace. This can be achieved by Kitaev's phase estimation algorithm. □

4 Phase Estimation

Phase Detection is actually a special case of a more general algorithm called **Phase Estimation**, due to Kitaev[Kit97]. Here it the theorem:

Theorem 4.1. *Let U be an unitary operation on $\mathbb{R}^M$, given to a quantum algorithm as a "black box". Let $|\psi\rangle$ be an eigenvector of U, also given (in a sense) as a "black box". Say the eigenvalue of $|\psi\rangle$ is $e^{i\theta}$, where $\theta \in (-\pi, \pi]$ Then with only $O(1/\delta)$ applications of U, it is possible to distinguish the case $\theta = 0$ from $\theta \geq \delta$ with high probability.*

Let's be a bit more precise. Our algorithm (quantum circuit) will work with two registers; an M-dimensional register, and a "workspace" register of dimension $\Theta(1/\delta)$. (You can think of the workspace register as roughly $\log(1/\delta)$ additional qubits.) The circuit is allowed to use U gates on the first register, although it doesn't "know" what U is. (Actually, it will use controlled-U gates; there is a basic quantum circuit theorem, which we skipped, showing that one can construct controlled-U gates from U gates.) It is also assumed that the input to the circuit will be $|\psi\rangle \otimes |0\rangle$, where again, $|\psi\rangle$ is some ("unknown") eigenvector of U with eigenvalue $e^{i\theta}$. Then the Phase Detection circuit has the following properties:

- it contains at most $O(1/\delta)$ controlled-U operations;
- if $\theta = 0$, i.e. $|\psi\rangle$ is fixed by U, then the final state of the circuit will always be exactly $|\psi\rangle \otimes |0\rangle$, the same as the initial state;
- it $\theta \geq \delta$, then the final state will be of the form $|\psi\rangle \otimes |\phi\rangle$, where $|\langle\phi\,|\,0\rangle| \leq 1/4$

Then, in a typical use of Phase Detection, you just measure at the end, and look at the second (workspace) register. If $\theta = 0$ then you will see $|0\rangle$ with probability 1, and if $\theta \geq \delta$ you will see $|0\rangle$ with probability at most 1/4.

Now actually, it's not 100% immediate to finish Reichardt's theorem with Phase Detection, because the summary that we ended suggested applying it with $|\psi\rangle$ equal to this "$|0\rangle$" vector, and $|0\rangle$ wasn't necessarily an eigenvalue of U, even in the YES case (in that case, it was only 99% equal to a 1-eigenvalue of U). Still, we're *almost* finished; I leave it as an exercise for the reader to complete the proof of Reichardt's theorem using the two facts we ended the summary one, plus the Phase Detection algorithm. (Hint: every input to Phase Detection can be written as a linear combination of U's orthogonal eigenvalues; so apply Phase Detection's guarantee to each, and use the fact that the Phase Detection algorithm is, like every quantum algorithm, a unitary linear transformation.)

We now give the proof of the Phase Detection theorem.

Proof. Let D be $8/\delta$ rounded up to the nearest integer power of 2, so $D = O(1/\delta)$. The workspace register will consist of exactly $d = \log 2D$ qubits, thought of as encoding an integer between 0 and $D1$. Now here is the algorithm:

- UniformSuperposition(workspace register) (this is just d little Hadamard gates, one on each workspace wire).

- for $t = 1, \dots, D-1$
  do "controlled-U" on the first register, where the control condition is that the integer in the second register is at least t.

- UniformSuperposition^{-1}(workspace register).

That's it. You see it's indeed $O(1/\delta)$ applications of U. Let us now track the state of the registers throughout the algorithm.

(a) Initial state: $|\psi\rangle \otimes |0\rangle$

(b) After step 1:

$$|\psi\rangle \otimes \left(\frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} |j\rangle \right) = \frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} |\psi\rangle \otimes |j\rangle$$

(c) After step 2:

$$\frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} U^{j+1} |\psi\rangle \otimes |j\rangle = \frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} e^{i\theta(j+1)} |\psi\rangle \otimes |j\rangle \text{ since } |\psi\rangle \text{ is an } e^{i\theta} \text{ eigen vector of } U$$
$$= |\psi\rangle \otimes |\phi\rangle$$

where $|\phi\rangle = \frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} e^{i\theta(j+1)} |j\rangle$ and the final eq

Let us now consider the two cases we need to analyze for the theorem.

- **Case 1**: $\theta = 0$ In this case we simply have $|\phi\rangle = \frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} |j\rangle$ i.e., $|\phi\rangle$ is the uniform superposition in the second register. hus after step 3, it will turn back into $|0\rangle$. Hence the final state is indeed $|\psi\rangle \otimes |0\rangle$

- **Case 2**: $|\theta| \geq \delta$ since UniformSuperposition is a unitary transformation, it (and its inverse) preserve angles. It follows that the exact statement we must show is that $\langle \phi \,|\, \text{uniform superposition} \rangle^2 \leq 1/4$. The unsquared quantity on the left (which we must bound by 1/2) is

$$|\left(\frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} e^{-i\theta(j+1)} \langle j| \right) \left(\frac{1}{\sqrt{D}} \sum_{j=0}^{D-1} |j\rangle \right)| = |\frac{1}{D} \sum_{j=0}^{D-1} e^{-i\theta(j+1)}|$$

  You should be able to see how this will work out; we have a unit complex number with angle $-\theta$ where $|\theta| \geq \delta$. We're averaging it over D rotations of itself, where $D \gg \frac{1}{\delta}$. It should come out close to 0. To be completely precise, the above quantity is exactly (by the formula for the sum of a geometric series)

$$\frac{1}{D} \frac{|1 - e^{-i\theta D}|}{|1 - e^{-i\theta}|}$$

We have $\frac{1}{D} \leq \frac{\delta}{8}$, the numerator above is trivially at most 2, and the denominator is at least $|\theta|/2$ (simple trig), which is at least $\delta/2$. So the above expression is indeed at most $1/2$, as desired

$\square$

References

[HLS07] Peter Hoyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 526–535, New York, NY, USA, 2007. ACM.

[Kit97] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997.

[Rei09] B. W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *Foundations of Computer Science, 2009. FOCS '09. 50th Annual IEEE Symposium*, pages 544–551, October 2009.

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

Lecture 16: Mixed States and Measurement

November 2, 2015

Lecturer: John Wright *Scribe: Zilin Jiang*

1 Mixed States

Today, we will change the topic from quantum algorithm to quantum information theory. To lay the foundation for the qunatum information theory, we first need to generalize the definition of quantum states.

1.1 Motivation

Recall that, in the second lecture, we defined quantum states as vectors. Specifically, a quantum state in the d-dimensional qudit system is a superposition of d basis states. We can write:

$$|\psi\rangle = \alpha_1 |1\rangle + \alpha_2 |2\rangle + \cdots + \alpha_d |d\rangle,$$

where $|\alpha_1|^2 + |\alpha_2|^2 + \cdots + |\alpha_d|^2 = 1$.

However, using vectors to describe the state of a quantum system sometimes is not enough. One motivation to generalize the definition of a quantum state is to model quantum noise. When you implement a quantum system, the qunatum processes involved are naturally "noisy", whatever that means, and they are modeled as devices producing quantum states $|\psi_i\rangle$ with probability p_i. A schematic diagram of such a device is as follows.

$$\text{The devise outputs} \begin{cases} |\psi_1\rangle & \text{with probability } p_1, \\ |\psi_2\rangle & \text{with probability } p_2, \\ & \vdots \end{cases}$$

Roughly speaking, its quantum state is sometimes $|\psi_1\rangle$, sometimes $|\psi_2\rangle$ and so on.

One might be attempted to use a vector, for example $\sum_i p_i |\psi_i\rangle$, to represent the state of such a quantum device. But vectors are just not the correct notions to capture the quantum state of such a device.

1.2 Mixed state represented by matrix

In order to come up with the right notions to describe the physical system of such a devise, let's measure the devise in the basis $|v_1\rangle, |v_2\rangle, \ldots, |v_d\rangle$. We compute

$$\mathbf{Pr}[\text{observe } |v_i\rangle] = \sum_j p_j |\langle v_i|\psi_j\rangle|^2 = \sum_j p_j \langle v_i|\psi_j\rangle\langle\psi_j|v_i\rangle = \langle v_i| \left(\sum_j p_j |\psi_j\rangle \langle\psi_j|\right) |v_i\rangle \quad (1)$$

Definition 1.1. The *mixed state* $\{p_i, |\psi_i\rangle\}$ is represented by the matrix $\rho = \sum_j p_j |\psi_j\rangle \langle\psi_j|$. The state is called a pure state if $p_i = 1$ for some i.

The matrix ρ embodies everything related to the mixed state. In (1), we have seen that the outcome of the measurement in the basis can be expressed in terms of ρ, that is

$$\mathbf{Pr}\left[\text{observe}\,|v_i\rangle\right] = \langle v_i|\,\rho\,|v_i\rangle\,.$$

In the following example, we compute the matrices representing various mixed states.

Example 1.2. *The mixed state*

$$S_1 = \begin{cases} |0\rangle & \textit{with probability } 1 \\ |1\rangle & \textit{with probability } 0 \end{cases}$$

is represented by

$$|0\rangle\langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

The mixed state

$$S_2 = \begin{cases} -|0\rangle & \textit{with probability } 1 \\ |1\rangle & \textit{with probability } 0 \end{cases}$$

is represented by the same matrix

$$(-|0\rangle)(-\langle 0|) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

The mixed state

$$S_3 = \begin{cases} |0\rangle & \textit{with probability } 1/2 \\ |1\rangle & \textit{with probability } 1/2 \end{cases}$$

is represented by

$$\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

Though S_1 and S_2 in Example 1.2 look different, they are not distinguishable from an observer, and so they share the same representing matrix. This reflects the fact that the representing matrix contains only the observable information and does not contain redundant information.

Analogous to the vector formulation of quantum state, we can axiomatize the matrix formulation of mixed state as follows.

1. (Measurement) If ρ represents a mixed state and you measure in the basis $|v_1\rangle, \ldots, |v_d\rangle$, then $\mathbf{Pr}\left[\text{observe}\,|v_i\rangle\right] = \langle v_i|\,\rho\,|v_i\rangle$. We have computed this probability in (1).
2. (Evolution) Suppose a mixed state $S_1 = \{p_i, |\psi_i\rangle\}$ goes though a unitary gate U and transforms to $S_2 = \{p_i, U|\psi_i\rangle\}$. If the matrix representing S_1 is ρ, then the matrix representing S_2 is

$$\sum_i p_i (U|\psi_i\rangle)(U|\psi_i\rangle)^\dagger = \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\left(\sum_i p_i |\psi_i\rangle\langle\psi_i|\right)U^\dagger = U\rho U^\dagger.$$

1.3 Density matrix

The matrix presenting a mixed state has the following property.

Proposition 1.3. *If ρ represents a mixed state, then* $\operatorname{tr}(\rho) = 1$ *and ρ is positive semidefinite (PSD).*

Proof. Note that for a pure state $|\psi\rangle = a_1 |1\rangle + \cdots + a_d |d\rangle$, $\operatorname{tr}(|\psi\rangle\langle\psi|) = |a_1|^2 + \cdots + |a_d|^2 = 1$. Suppose the mixed state is $\{p_i, |\psi_i\rangle\}$. On one hand, we have that $\operatorname{tr}(\rho) = \operatorname{tr}\left(\sum_i p_i |\psi_i\rangle\langle\psi_i|\right) = \sum_i p_i \operatorname{tr}(|\psi_i\rangle\langle\psi_i|) = \sum_i p_i = 1$. On the other hand, for any $|v\rangle$, we have that $\langle v| \rho |v\rangle = \mathbf{Pr}\,[\text{observe } |v\rangle] \geq 0$, hence that ρ is PSD. □

Definition 1.4. We say that a matrix ρ is a *density matrix* if and only if $\operatorname{tr}(\rho) = 1$ and ρ is PSD.

Proposition 1.3 simply says that a matrix representing a mixed state is a density matrix. In fact, the converse is also true.

Proposition 1.5. *If ρ is a density matrix, then it represents a mixed state.*

Proof. By the spectral theorem for Hermitian matrices, we know that $\rho = \sum_{i=1}^d \lambda_i |v_i\rangle\langle v_i|$, where λ_i's are the (real) eigenvalues and $|v_i\rangle$'s from an orthonormal basis. Since ρ is PSD, we know that $\lambda_i \geq 0$ for all i. The assumption $\operatorname{tr}(\rho) = 1$ implies that $\sum_{i=1}^d \lambda_i = 1$. So ρ represents the mixed state $\{p_i, |v_i\rangle : i \in [d]\}$. □

Remark 1.6. Even though a mixed state, say $\{p_i, |\psi_i\rangle\}$, can be rather complicated, for example, there are infinitely many nonzero p_i's, from the proof of Proposition 1.5 we see that $\{p_i, |\psi_i\rangle\}$ is indistinguishable from $\{\lambda_i, |v_i\rangle\}$ as they represent the same density matrix.

We have already seen in Example 1.2 that density matrix is a succinct way to represent a mixed state. One can actually use the density matrices to check whether two mixed states are distinguishable. Here is an example.

Example 1.7. *Suppose mixed state*

$$S_1 = \begin{cases} |0\rangle & \textit{with probability } 3/4 \\ |1\rangle & \textit{with probability } 1/4 \end{cases}$$

and mixed state

$$S_2 = \begin{cases} \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle & \textit{with probability } 1/2 \\ \frac{\sqrt{3}}{2}|0\rangle - \frac{1}{2}|1\rangle & \textit{with probability } 1/2 \end{cases}.$$

Both mixed states are represented by

$$\frac{3}{4}|0\rangle\langle 0| + \frac{1}{4}|0\rangle\langle 0| = \begin{bmatrix} 3/4 & 0 \\ 0 & 1/4 \end{bmatrix}$$
$$= \frac{1}{2}\left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle\right)\left(\frac{\sqrt{3}}{2}\langle 0| + \frac{1}{2}\langle 1|\right) + \frac{1}{2}\left(\frac{\sqrt{3}}{2}|0\rangle - \frac{1}{2}|1\rangle\right)\left(\frac{\sqrt{3}}{2}\langle 0| - \frac{1}{2}\langle 1|\right),$$

and hence indistinguishable.

1.4 Density matrix in quantum algorithms

In the hidden subgroup problem (HSP) for group G, if f "hides" a subgroup $H \leq G$, then the "standard method" outputs uniformly random coset

$$|gH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |gh\rangle .$$

Each coset $|gH\rangle$ is output with probability $\frac{1}{|G|}$. So the density matrix representing the mixed state is

$$\rho_H = \sum_{g \in G} \frac{1}{|G|} |gH\rangle \langle gH| = \frac{1}{|G|} \sum_{g \in G} |gH\rangle \langle gH| .$$

In general, partial measurements would generate mixed states.

2 Measurements

The next step is to generalize the definition of measurements.

2.1 Simple measurements

We first review the "simple measurements" (not standard terminology). Given a pure state $|\psi\rangle$, a "simple measurement" is as follows.

1. Pick orthonormal basis $|v_1\rangle, \ldots, |v_d\rangle$.
2. Receive outcome "i" with probability $|\langle v_i|\psi\rangle|^2$.
3. $|\psi\rangle$ "collapses" to $|v_i\rangle$.

2.2 Most general quantum measurement

The most general quantum measurement can be described using matrices. We first focus on the measurement rules for pure states.

1. Pick $d \times d$ matrices $M_1, \ldots, M_m$ satisfying the *completeness* condition that
$$M_1^\dagger M_1 + \cdots + M_m^\dagger M_m = I. \tag{2}$$
2. Receive outcome "i" with probability
$$|M_i |\psi\rangle|^2 = \langle\psi| M_i^\dagger M_i |\psi\rangle . \tag{3}$$
3. ψ collapses to
$$\frac{M_i |\psi\rangle}{|M_i |\psi\rangle|} = \frac{M_i |\psi\rangle}{\sqrt{\langle\psi| M_i^\dagger M_i |\psi\rangle}} \tag{4}$$

2.3 Sanity check

We first check that the $|M_i |\psi\rangle|^2$'s should sum to 1. Using the completeness condition (2), we have that

$$\sum_i |M_i |\psi\rangle|^2 = \sum_i \langle\psi| M_i^\dagger M_i |\psi\rangle = \langle\psi| \left(\sum_i M_i^\dagger M_i\right) |\psi\rangle = \langle\psi|\psi\rangle = 1.$$

Secondly, we check that the general measurement extends the "simple measurement". In the case of simple measurement, we measure with respect to orthonormal basis $|v_1\rangle, \dots, |v_d\rangle$. Take $M_i = |v_i\rangle\langle v_i|$ for all $i \in [d]$. Note that $M_i^\dagger M_i = |v_i\rangle\langle v_i| |v_i\rangle\langle v_i| = |v_i\rangle\langle v_i|$. According to the definition of the general quantum measurement, we check that

1. Completeness condition:
$$M_1^\dagger M_1 + \cdots + M_d^\dagger M_d = |v_1\rangle\langle v_1| + \cdots + |v_d\rangle\langle v_d| = I.$$
2. Receive "i" with probability
$$\langle\psi| M_i^\dagger M_i |\psi\rangle = \langle\psi|v_i\rangle\langle\psi|v_i\rangle = |\langle v_i|\psi_i\rangle|^2.$$
3. ψ collapses to
$$\frac{M_i |\psi\rangle}{|M_i |\psi\rangle|} = \frac{|v_i\rangle\langle v_i|\psi\rangle}{|\langle v_i|\psi\rangle|} = e^{i\theta} |v_i\rangle.$$

We further define a special kind of general measurement, called projective measurement.

2.4 Projective measurement

Recall that a projection Π is a PSD matrix such that $\Pi^2 = \Pi$ (cf. Problem 6 in Homework 2). Equivalently, Π is a projection provided that $\Pi = \sum_{i=1}^k |v_i\rangle\langle v_i|$, where $|v_i\rangle$'s are orthonormal (but they do not necessarily form a basis).

Definition 2.1. A projective measurement is the case when $M_1 = \Pi_1, \dots, M_m = \Pi_m$, where Π's are projections such that

$$\Pi_1 + \cdots + \Pi_m = I. \tag{5}$$

Equation (5) implies the completeness condition (2). This is simply because $M_i^\dagger M_i = \Pi_i^\dagger \Pi_i = \Pi_i^2 = \Pi_i$. Moreover, we receive outcome "i" with probability $|\Pi |\psi\rangle|^2 = \langle\psi| \Pi_i^\dagger \Pi_i |\psi\rangle = \langle\psi| \Pi_i |\psi\rangle$ and the state collapses to $\Pi_i |\psi\rangle / |\Pi_i |\psi\rangle|$.

One way to think about a projective measurement is to imagine that you pick an orthonormal basis $|v_1\rangle, \dots, |v_d\rangle$ and "bundle" those vectors into projectors: $\Pi_i = \sum_{j \in S_i} |v_j\rangle\langle v_j|$ for all $i \in [m]$, where $S_1, \dots, S_m$ form a partition of $[d]$. Moreover, it is easy to see that the "simple measurement" is a projective measurement.

Example 2.2. *Given state $|\psi\rangle = \frac{1}{\sqrt{3}}(|0\rangle + |1\rangle + |2\rangle)$. If we pick $\Pi_1 = |0\rangle\langle 0|, \Pi_2 = |1\rangle\langle 1| + |2\rangle\langle 2|$ and carry out a projective measurement on $|\psi\rangle$. Since $\Pi_1 |\psi\rangle = \frac{1}{\sqrt{3}} |0\rangle$ and $\Pi_2 |\psi\rangle = \frac{1}{\sqrt{3}}(|1\rangle + |2\rangle)$, we will observe "1" and get $|0\rangle$ with probability $\frac{1}{3}$, and we will observe "2" and get $\frac{1}{\sqrt{2}}(|1\rangle + |2\rangle)$ with probability $\frac{2}{3}$.*

2.5 Measurement rules for mixed states

Now, we derive the measurement rules for mixed states.

Proposition 2.3. *Given mixed state $\{p_i, |\psi_i\rangle\}$ represented by density matrix ρ. If we measure the mixed state with respect to $M_1, \dots, M_m$ satisfying the completeness condition, then*

1. *Receive outcome "i" with probability*
$$\operatorname{tr}\left(M_i^\dagger M_i \rho\right).$$

2. *ρ collapses to*
$$\frac{M_i \rho M_i^\dagger}{\operatorname{tr}\left(M_i^\dagger M_i \rho\right)}.$$

Proof. By the definition of density matrix, we have that $\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i|$. Note that

$$\operatorname{tr}\left(M_i^\dagger M_i \rho\right) = \operatorname{tr}\left(M_i^\dagger M_i \sum_j p_j |\psi_j\rangle\langle\psi_j|\right) = \sum_j p_j \operatorname{tr}\left(M_i^\dagger M_i |\psi_j\rangle\langle\psi_j|\right) = \sum_j \langle\psi_j| M_i^\dagger M_i |\psi_j\rangle.$$

In the last equality, we used the simple fact that $\operatorname{tr}(|a\rangle\langle b|) = \langle b|a\rangle$ Using conditional probability, it is easy to see

$$p_i' := \mathbf{Pr}[\text{receive } i] = \sum_j p_j \, \mathbf{Pr}[\text{receive } i \mid \text{measure } |\psi_j\rangle] = \sum_j p_j \langle\psi_j| M_i^\dagger M_i |\psi_j\rangle = \operatorname{tr}\left(M_i^\dagger M_i \rho\right).$$

Because pure state $|\psi_j\rangle$ collapses to pure state

$$|\psi_j'\rangle = \frac{M_i|\psi_j\rangle}{\sqrt{\langle\psi_j| M_i^\dagger M_i |\psi_j\rangle}} =: \frac{M_i|\psi_j\rangle}{\sqrt{p_{ji}}},$$

with probability p_{ji}, the mixed state $\{p_j, |\psi_j\rangle\}$ collapses to $\left\{p_j p_{ji}/p_i', |\psi_j'\rangle\right\}$ represented by

$$\sum_j \frac{p_j p_{ji}}{p_i'} \left(\frac{M_i|\psi_j\rangle}{\sqrt{p_{ji}}}\right)\left(\frac{M_i|\psi_j\rangle}{\sqrt{p_{ji}}}\right)^\dagger = \frac{1}{p_i'} \sum_j p_j M_i |\psi_j\rangle\langle\psi_j| M_i^\dagger = \frac{M_i \rho M_i^\dagger}{\operatorname{tr}\left(M_i^\dagger M_i \rho\right)}.$$

□

3 Mixed state in entanglement

Suppose Alice and Bob share a pair of qudits with joint state $|\psi\rangle = \sum_{i,j\in[d]} \alpha_{i,j} |i\rangle \otimes |j\rangle$. Sadly, Bob is light-years away from Alice and got forgotten by everyone. The question is

"How do you describe Alice's qudit?"[1].

We claim that from Alice's perspective, her qudit is represented as a mixed state. In other words, if she performs a measurement on her qudit, then her measurement outcomes are consistent with her qudit being a mixed state. Which mixed state?

Suppose prior to Alice's measurement, Bob measures his qudit (in standard basis). He sees $|j\rangle$ with probability $p_j := \sum_i |\alpha_{i,j}|^2$ and Alice's state becomes $|\psi_j\rangle := \frac{1}{\sqrt{p_j}} \sum_i \alpha_{i,j} |i\rangle$. Had Bob measured before Alice, Alice's state would become a mixed state represented by

$$\sum_j p_j |\psi_j\rangle \langle\psi_j| = \sum_j p_j \left(\frac{1}{\sqrt{p_j}} \sum_{i_1} \alpha_{i_1,j} |i_1\rangle\right) \left(\frac{1}{\sqrt{p_j}} \sum_{i_2} \alpha^\dagger_{i_2,j} \langle i_2|\right) = \sum_{i_1,i_2} |i_1\rangle \langle i_2| \sum_j \alpha_{i_1,j} \alpha_{i_2,j}.$$

However, by relativity, Bob's measurement information hasn't propagated to Alice's world since Bob is light-years away. This means that, without loss of generality, we may always assume Bob had measured before Alice.

Well, if you believe in relativity, then you're forced to conclude that Alice's state is given by the mixed state

$$\rho_A = \sum_{i_1,i_2 \in [d]} |i_1\rangle \langle i_2| \sum_{j \in [d]} \alpha_{i_1,j} \alpha^\dagger_{i_2,j}.$$

Of course, you might not believe in relativity (your loss), in which case this argument isn't too convincing. So let's see a quantum mechanical proof of this fact.

Before this, let me describe the standard way that ρ_A is defined. Suppose R and S are $d \times d$ matrices, where we think of R as Alice's matrix and S as Bob's matrix. Then the partial trace tr_B is defined as $\mathrm{tr}_B (R \otimes S) := R \cdot \mathrm{tr}(S)$.

The notation is suggestive: you take the trace of Bob's matrix S and multiply it by R. This shows how to define tr_B for $d^2 \times d^2$ matrices of the form $R \otimes S$. If we further specify that tr_B is linear, then $\mathrm{tr}_B(M)$ is defined for any $d^2 \times d^2$ matrix M. This is because any such M can be expanded as $M = \sum_i R_i \otimes S_i$, where the R_i's and S_i's are $d \times d$.

Now, I claim that ρ_A (as defined above) is equal to $\mathrm{tr}_B (|\psi\rangle \langle\psi|)$. To see this, first note that

$$|\psi\rangle \langle\psi| = \sum_{i_1,j_1,i_2,j_2} \alpha_{i_1,j_1} \alpha^\dagger_{i_2,j_2} |i_1\rangle \langle i_2| \otimes |j_1\rangle \langle j_2|.$$

Thus, by linearity of tr_B and the fact that $\mathrm{tr}(|a\rangle \langle b|) = 1$ if $a = b$ and 0 otherwise,

$$\begin{aligned}
\mathrm{tr}_B (|\psi\rangle \langle\psi|) &= \sum_{i_1,j_1,i_2,j_2} \alpha_{i_1,j_1} \alpha^\dagger_{i_2,j_2} \mathrm{tr}_B (|i_1\rangle \langle i_2| \otimes |j_1\rangle \langle j_2|) \\
&= \sum_{i_1,j_1,i_2,j_2} \alpha_{i_1,j_1} \alpha^\dagger_{i_2,j_2} |i_1\rangle \langle i_2| \cdot \mathrm{tr}(|j_1\rangle \langle j_2|) \\
&= \sum_{i_1,i_2} \sum_j \alpha_{i_1,j} \alpha^\dagger_{i_2,j} |i_1\rangle \langle i_2|,
\end{aligned}$$

[1]The notes below are mostly based on a Piazza post by John Wright.

which, after some trivial manipulations, is exactly ρ_A from above. So any time you see the notation tr_B applied to a multi-qudit state, it just means that you're supposed to compute the mixed state that Alice sees. And the above argument shows how to carry out this process in a simple manner. By the way, if we wanted to figure out which mixed state Bob's qudit is in, then it would be in the state $\mathrm{tr}_A\left(|\psi\rangle\langle\psi|\right)$, where tr_A is defined so that $\mathrm{tr}_A\left(R \otimes S\right) = \mathrm{tr}\left(R\right) \cdot S$.

Okay, so now let's prove that $\rho_A = \mathrm{tr}_B\left(|\psi\rangle\langle\psi|\right)$ is the state of Alice's qudit. This means that if she performs a measurement on her qudit, then her measurement outcome is consistent with her qudit being in the mixed state ρ_A. If Alice measures her qudit in the standard basis $|1\rangle, \ldots, |d\rangle$, then she sees "i" with with probability $\sum_j |\alpha_{i,j}|^2$. This is identical to performing the projective measurement $|1\rangle\langle 1| \otimes I, \ldots, |d\rangle\langle d| \otimes I$ (where I is the $d \times d$ identity matrix) on the whole state $|\psi\rangle$:

$$\mathbf{Pr}\left[\text{Alice observe } |i\rangle\right] = \left|\left(|i\rangle\langle i| \otimes I\right)|\psi\rangle\right|^2 = \left|\sum_{i',j} \alpha_{i',j}(|i\rangle\langle i|i'\rangle) \otimes |j\rangle\right|^2 = \sum_j |\alpha_{i,j}|^2 .$$

Then we claimed that (without proof) if Alice measures her qudit in the basis $|v_1\rangle, \cdots, |v_d\rangle$, then this is identical to performing the projective measurement $|v_1\rangle\langle v_1| \otimes I, \cdots, |v_d\rangle\langle v_d| \otimes I$. Let's now prove this fact. If Alice measures in the basis $|v_1\rangle, \cdots, |v_d\rangle$, this means first applying the unitary $U = \sum_i |i\rangle\langle v_i|$ to her qudit and then measuring in the standard basis. This is equivalent to applying the unitary $U \otimes I$ to the total state $|\psi\rangle$ and then performing the projective measurement $|1\rangle\langle 1| \otimes I, \cdots, |d\rangle\langle d| \otimes I$. So the probability Alice observes outcome $|v_i\rangle$ is the probability she observes outcome $|i\rangle$ when measuring the state $(U \otimes I)|\psi\rangle$. And this probability is equal to

$$\langle\psi|(U^\dagger \otimes I)(|i\rangle\langle i| \otimes I)(U \otimes I)|\psi\rangle = \langle\psi|(U^\dagger|i\rangle)(\langle i|U) \otimes I|\psi\rangle = \langle\psi|(|v_i\rangle\langle v_i| \otimes I)|\psi\rangle ,$$

which is identical to the measurement distribution of the projective measurement $|v_1\rangle\langle v_1| \otimes I, \cdots, |v_d\rangle\langle v_d| \otimes I$, as claimed.

With this fact in hand, we can now prove the main result, which is that Alice's state is given by the mixed state ρ_A. In particular, if she measures in a basis $|v_1\rangle, \cdots, |v_d\rangle$, then she should see outcome $|v_i\rangle$ with probability $\langle v_i|\rho|v_i\rangle$. Let's verify this. By the above fact, the probability that she gets outcome i when measuring is equal to

$$\begin{aligned}
\langle\psi| \left(|v_i\rangle \langle v_i| \otimes I\right) |\psi\rangle &= \left(\sum_{i_1,j_1} \alpha^\dagger_{i_1,j_1} \langle i_1| \otimes \langle j_1|\right) \left(|v_i\rangle \langle v_i| \otimes I\right) \left(\sum_{i_2,j_2} \alpha_{i_2,j_2} |i_2\rangle \otimes |j_2\rangle\right) \\
&= \sum_{i_1,j_1,i_2,j_2} \alpha^\dagger_{i_1,j_1} \alpha_{i_2,j_2} \langle i_1| \otimes \langle j_1| \left(|v_i\rangle \langle v_i| \otimes I\right) |i_2\rangle \otimes |j_2\rangle \\
&= \sum_{i_1,j_1,i_2,j_2} \alpha^\dagger_{i_1,j_1} \alpha_{i_2,j_2} \langle i_1|v_i\rangle\langle v_i|i_2\rangle\langle j_1|j_2\rangle \\
&= \sum_{i_1,i_2,j} \alpha^\dagger_{i_1,j} \alpha_{i_2,j} \langle v_i|i_2\rangle\langle i_1|v_i\rangle \\
&= \langle v_i| \left(\sum_{i_1,i_2,j} \alpha^\dagger_{i_1,j} \alpha_{i_2,j} |i_2\rangle \langle i_1|\right) |v_i\rangle \\
&= \langle v_i| \rho_A |v_i\rangle ,
\end{aligned}$$

exactly as promised. So ρ_A represents Alice's mixed state, and we're done.

One nice thing about the partial trace definition is it allows us to prove the following fact: suppose that Bob applies the unitary U to his qudit. Then Alice's state should remain unchanged. In particular, her mixed state should be the same before and after Bob applies his unitary. To see this, her mixed state after Bob applies his unitary is given by

$$\begin{aligned}
\operatorname{tr}_B\left((I \otimes U) |\psi\rangle \langle\psi| (I \otimes U^\dagger)\right) &= \operatorname{tr}_B\left(\sum_{i_1,j_1,i_2,j_2} \alpha_{i_1,j_1} \alpha^\dagger_{i_2,j_2} \left(|i_1\rangle \langle i_2|\right) \otimes \left(U |j_1\rangle \langle j_2| U^\dagger\right)\right) \\
&= \sum_{i_1,j_1,i_2,j_2} \alpha_{i_1,j_1} \alpha^\dagger_{i_2,j_2} |i_1\rangle \langle i_2| \otimes \operatorname{tr}\left(U |j_1\rangle \langle j_2| U^\dagger\right) \\
&= \sum_{i_1,j_1,i_2,j_2} \alpha_{i_1,j_1} \alpha^\dagger_{i_2,j_2} |i_1\rangle \langle i_2| \otimes \operatorname{tr}\left(|j_1\rangle \langle j_2|\right) \\
&= \operatorname{tr}_B\left(|\psi\rangle \langle\psi|\right) \\
&= \rho_A.
\end{aligned}$$

Here we used the fact that $\operatorname{tr}(ABC) = \operatorname{tr}(BCA)$. So Bob can apply any unitary rotation to his qudit that he wants, but from Alice's perspective her qudit's state never changes. We saw this previously in lecture 3 in the special case of the quantum teleportation circuit.

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

Lecture 17: Discriminating Two Quantum States

November 5, 2015

Lecturer: Ryan O'Donnell *Scribe: Zilin Jiang*

1 Recap and more

1.1 Mixed state

Recall that a *mixed state* $\{p_i, |\psi_i\rangle\}$, where $|\psi_i\rangle \in \mathbb{C}^d$, is represented by *density matrix* $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \in \mathbb{C}^{d\times d}$. Conversely, a density matrix ρ, by definition,

1. is positive semidefinite (PSD), and so it has orthonormal eigenvectors $|v_1\rangle, \dots, |v_d\rangle$ with nonnegative eigenvalues $\lambda_1, \dots, \lambda_d$;
2. and it satisfies $\operatorname{tr}(\rho) = 1$, and so $\lambda_1 + \cdots + \lambda_d = 1$.

Therefore ρ represents mixed state $\{\lambda_i, |v_i\rangle\}$.

We shall use repeatedly use the following properties of traces in this lecture.

Fact 1.1. *Here are some basic properties of the trace operator.*

1. $\operatorname{tr}(A) = \sum_i A_{ii} = \sum \lambda_i$, *where* λ_i*'s are the eigenvalues of* A*;*
2. *and* $\operatorname{tr}(AB) = \sum_{i,j} A_{ij}B_{ji} = \operatorname{tr}(BA)$.

Corollary 1.2. *The following properties follow immediately from the second fact above.*

1. $\operatorname{tr}(|\psi\rangle\langle\phi|) = \operatorname{tr}(\langle\phi|\psi\rangle) = \langle\phi|\psi\rangle$;
2. *the trace operator is invariant under orthonormal transformation: given unitary matrix* U*, we have*
$$\operatorname{tr}\left(UMU^\dagger\right) = \operatorname{tr}\left(U^\dagger UM\right) = \operatorname{tr}(M).$$

1.2 General measurement

A general measurement is defined by matrices $M_1, \dots, M_m$ such that $\sum_i M_i^\dagger M_i = I$. If this general measurement is carried out on a mixed state represented by ρ, in the last lecture we have shown that we will see outcome "j" with probability

$$p_j := \operatorname{tr}\left(M_j^\dagger M_j \rho\right) = \operatorname{tr}\left(M_j \rho M_j^\dagger\right) \tag{1}$$

and the state collapses a mixed stated represented by

$$\frac{M_j \rho M_j^\dagger}{p_j}. \tag{2}$$

If we measure a mixed state represented by ρ in orthonormal basis $|v_1\rangle, \ldots, |v_d\rangle$, or equivalently we carry out a measurement described by $M_i = |v_i\rangle\langle v_i|$, then, from (1) and (2), we see outcome "j" with probability

$$p_j := \langle v_j| \rho |v_j\rangle = \operatorname{tr}(|v_j\rangle\langle v_j| \rho) \tag{3}$$

and the state collapses to the mixed state represented by

$$\frac{1}{p_j} |v_j\rangle\langle v_j| \rho |v_j\rangle\langle v_j| . \tag{4}$$

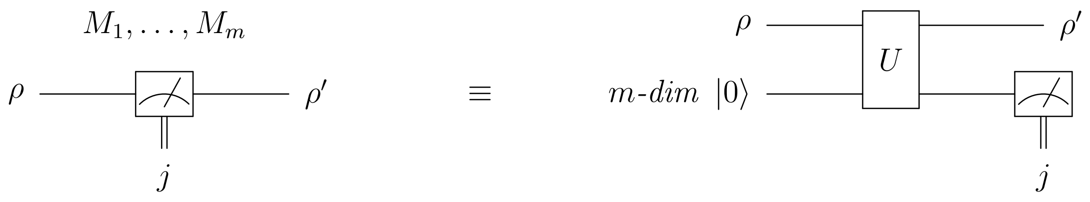
Let $E_i = M_i^\dagger M_i$ in the definition of general measurement. We know that $\{E_i\}$ is a set of PSD matrices such that $\sum_i E_i = I$. Suppose that we are only given E_i. From (1), we will still be able to determine the probability of outcome "j": $p_j = \operatorname{tr}(E_i \rho)$. This makes this set $\{E_i\}$ intersting.

Definition 1.3. A *positive-operator valued measure (POVM)* is a set of PSD matrices $E_1, \ldots, E_m$ such that $E_1 + E_2 + \cdots + E_m = I$.

Specifying E_i's gives probability distribution of the outcome, but implementation of such a measurement still needs M_i. Moreover, from (2), we can see that POVM does not tell what the state collapses to.

Though we have generalized the definitions of quantum state and measurement, we haven't actually added anything new at all. On one hand, one can still reason about mixed states through conditional probability. On the other hand, density matrices are not new due to Naimark's (Neumark's) theorem. The theorem will appear as a homework problem.

Theorem 1.4. *A general measurement described by $M_1, \ldots, M_m$ is equivalent to a quantum circuit with m-dimensional ancilla and a unitary gate U followed by a partial (simple) measurement on the ancilla bits.*



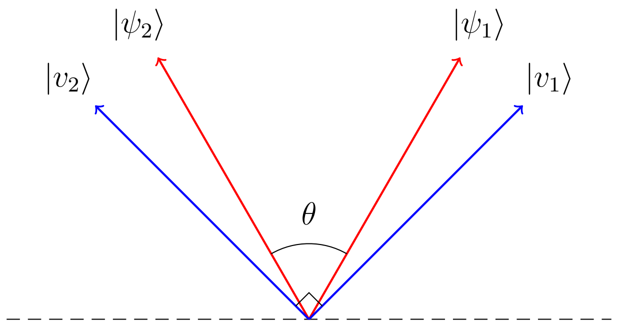
So the point of the generalization should really be thought as a new mathematical way to describe quantum states and measurements nicely.

2 Discriminating two pure states

Given a pure d-dimensional state $|\psi\rangle$ known to be either $|\psi_1\rangle$ or $|\psi_2\rangle$. You must guess which state $|\psi\rangle$ really is. There are two kinds of errors: the probability p_1 we guess wrong when it's $|\psi_1\rangle$ and the probability p_2 we guess wrong when it's $|\psi_2\rangle$. The goal is to find a strategy to minimize $\max(p_1, p_2)$.

Assume with out loss of generality the angle between $|\psi_1\rangle$ and $|\psi_2\rangle$, say θ, is between 0 and $\pi/2$. Otherwise, we replace $|\psi_1\rangle$ by $-|\psi_1\rangle$.

Theorem 2.1. *The best strategy is to do the projective measurement with $\{|v_1\rangle, |v_2\rangle\}$, where $|v_1\rangle, |v_2\rangle$ are in the span of $|\psi_1\rangle$ and $|\psi_2\rangle$ such that $\langle v_1|v_2\rangle = 0$, they are symmetric with respect to the angle bisector of $|\psi_1\rangle$ and $|\psi_2\rangle$, and $|v_i\rangle$ is closer to $|\psi_i\rangle$ for $i = 1, 2$. On outcome "$|v_i\rangle$", we guess $|\psi_i\rangle$.*

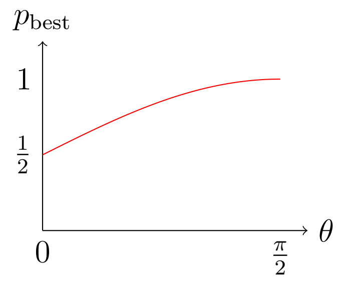


Remark 2.2. The strategy makes sense as measurements in the space perpendicular to $|\psi_1\rangle$ and $|\psi_2\rangle$ does not reveal information. Moreover, the best strategy should be symmetric with respect to $|\psi_1\rangle$ and $|\psi_2\rangle$.

The probability of success using the best strategy is

$$|\langle\psi_1|v_1\rangle|^2 = \cos^2(\angle(\psi_1, v_1)) = \cos^2\left(\frac{\pi/2-\theta}{2}\right) = \frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi}{2} - \theta\right) = \frac{1}{2} + \frac{1}{2}\sin\theta.$$

The best probability distribution of success is plotted below.



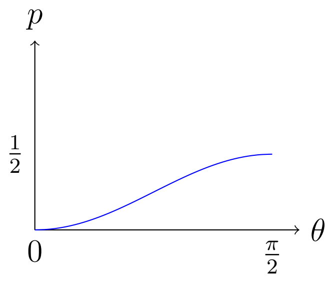
The optimality of the strategy follows from Theorem 3.4 in the next subsection. Over there, we actually prove the average error probability $(p_1 + p_2)/2 \geq \frac{1}{2} - \frac{1}{2}\sin\theta$ and so $\max(p_1, p_2) \geq \frac{1}{2} - \frac{1}{2}\sin\theta$.

2.1 Unambiguous discrimination

Now you can guess which state $|\psi\rangle$ or say "don't know", but you may never be wrong. One simple strategy is to do the projective measurement with $\{|\psi_1\rangle, |\psi_1^\perp\rangle\}$. If the outcome is $|\psi_1\rangle$, we say "don't know", otherwise we are sure that it's $|\psi_2\rangle$. This strategy gives

$$\mathbf{Pr}[\text{"don't known"}] = \begin{cases} 1 & \text{if it is } |\psi_1\rangle \\ \cos^2\theta & \text{if it is } |\psi_2\rangle . \end{cases}$$

A slightly cleverer strategy is to do, with probability 1/2, the projective measurement with $\{|\psi_1\rangle, |\psi_1^\perp\rangle\}$ and do, with probability 1/2, the projective measurement with $\{|\psi_2\rangle, |\psi_2^\perp\rangle\}$. Overall, the probability of "don't know" is $\frac{1}{2} + \frac{1}{2}\cos^2\theta$ and the probability of success is $\frac{1}{2} - \frac{1}{2}\cos^2\theta = \frac{1}{2}\sin^2\theta$. Its probability distribution is plotted below.



Apparently, we can do better in the case when $\theta = \pi/2$. In fact, we are able to decide deterministically $|\psi\rangle$ in this case. This suggests that the slightly cleverer strategy might not be the best strategy.

The strategy above can be thought as a mixture of a projection onto $|\psi_i^\perp\rangle$ for $i = 1, 2$. Set $\Pi_i = I - |\psi_i\rangle\langle\psi_i|$ for $i = 1, 2$. You might want to try the general measurement described by $M_i = \Pi_i$ or POVM with $E_i = M_i^\dagger M_i = \Pi_i$. This is not OK since $E_1 + E_2$ may not equal to I. If $E_1 + E_2 \preceq I$[1], then $E_0 = I - E_1 - E_2$ is PSD and $\{E_0, E_1, E_2\}$ is a POVM. However, in general $E_1 + E_2 \preceq I$ might not be the case.

Here is the way to fix this idea. Say maximum eigenvalue of $\Pi_1 + \Pi_2$ is c. So $\Pi_1 + \Pi_2 \preceq cI$. We will let $E_1 = \Pi_1/c, E_2 = \Pi_2/c, E_0 = I - E_1 - E_2 \succeq 0$.

Claim 2.3. *The maximum eigenvalue, c, of $\Pi_1 + \Pi_2$, is $1 + \cos\theta$.*

Proof. Here's the proof[2]. We have that c is the maximum, over all unit $|v\rangle$, of

$$\langle v| (\Pi_1 + \Pi_2) |v\rangle = \langle v| \Pi_1 |v\rangle + \langle v| \Pi_2 |v\rangle = \sin^2\theta_1 + \sin^2\theta_2,$$

where θ_i is the angle $|v\rangle$ makes with $|\psi_i\rangle$, and we used the most basic geometry. It's clear that this is maximized when $|v\rangle$ is in the same 2-dimensional plane as $|\psi_1\rangle, |\psi_2\rangle$. So we're now maximizing

$$\sin^2\theta_1 + \sin^2\theta_2 = 1 - \frac{1}{2}(\cos(2\theta_1) + \cos(2\theta_2))$$

[1] If $X - Y$ is PSD, we write $X \succeq Y$.

[2] The proof below is based on a Piazza post by Ryan O'Donnell.

subject to $\theta_1 + \theta_2 = \theta$, where I used the double-angle formula $\cos(2\alpha) = 1 - 2\sin^2\alpha$. Using $\cos(\alpha) + \cos(\beta) = 2\cos\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right)$, we have that

$$\frac{1}{2}(\cos(2\theta_1) + \cos(2\theta_2)) = \cos(\theta_1 + \theta_2)\cos(\theta_1 - \theta_2) = \cos\theta\cos(\theta_1 - \theta_2) \geq -\cos\theta.$$

Hence $c \leq 1 + \cos\theta$ and equality occurs when $\theta_1 = \theta/2 - \pi/2$ and $\theta_2 = \theta/2 + \pi/2$. □

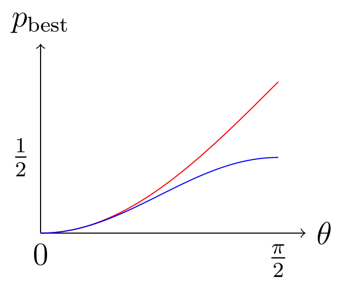
Fact 2.4. *The best strategy to unambiguously discriminate $|\psi_1\rangle$ and $|\psi_2\rangle$ is to carry out the POVM $\{E_0, E_1, E_2\}$. We guess $|\psi_2\rangle$ if the outcome is 1, we guess $|\psi_1\rangle$ if the outcome is 2, and we say "don't know" if the outcome is 0.*

For a proof of the optimality of the strategy, see [Per88, Iva87, Die88]. Here we only compute the best success probability. Say the state is $|\psi_2\rangle$. Then density matrix is $\rho = |\psi_2\rangle\langle\psi_2|$. We compute the probability of each outcome:

$$\begin{aligned}
\mathbf{Pr}[\text{outcome } 1] &= \mathrm{tr}(E_1\rho) \\
&= \mathrm{tr}\left(\frac{\Pi_1}{c}|\psi_2\rangle\langle\psi_2|\right) \\
&= \frac{1}{c}\mathrm{tr}((I - |\psi_1\rangle\langle\psi_1|)|\psi_2\rangle\langle\psi_2|) \\
&= \frac{1}{c}\mathrm{tr}(|\psi_2\rangle\langle\psi_2| - |\psi_1\rangle\langle\psi_1||\psi_2\rangle\langle\psi_2|) \\
&= \frac{1}{c}(\mathrm{tr}(|\psi_2\rangle\langle\psi_2|) - \mathrm{tr}(|\psi_1\rangle\langle\psi_1||\psi_2\rangle\langle\psi_2|)) \\
&= \frac{1}{c}\left(\langle\psi_2|\psi_2\rangle - |\langle\psi_1|\psi_2\rangle|^2\right) \\
&= \frac{1}{c}(1 - \cos^2\theta) \\
&= 1 - \cos\theta; \\
\mathbf{Pr}[\text{outcome } 2] &= \mathrm{tr}(E_2\rho) \\
&= \mathrm{tr}\left(\frac{\Pi_2}{c}|\psi_2\rangle\langle\psi_2|\right) \\
&= 0; \\
\mathbf{Pr}[\text{outcome } 0] &= \cos\theta.
\end{aligned}$$

The probability of success is plotted below (in red) in comparison with the previous strategy (in blue).

In general, if we want to unambiguously discriminate > 2 pure states, it is not clear how to find expression of the best probability is closed form. Using semidefinite programming in general gives good bounds on the best probability.



3 Discriminating two mixed states

Given a mixed state ρ known to be either ρ_1 or ρ_2. You must guess which state ρ really is. Assume it's ρ_1 with probability 1/2 and ρ_2 with probability 1/2. Our goal is to come up with a strategy to minimize the error probability.

Suppose ρ_1 represents ensemble $\{p_i, |\psi_i\rangle\}_{i=1}^d$, where p_i's are the eigenvalues of ρ_1 and $|\psi_i\rangle$'s are the eigenvectors. Similarly, ρ_2 represents ensemble $\{q_i, |\phi_i\rangle\}_{i=1}^d$.

3.1 Easy case

An easy case is when $|\psi_i\rangle = |\phi_i\rangle$ for all $i \in [d]$. Now ρ_1 and ρ_2 are simultaneously diagonalizable. Without loss of generality, we can assume that $|\psi_i\rangle = |\phi_i\rangle = |i\rangle$ for all $i \in [d]$ and

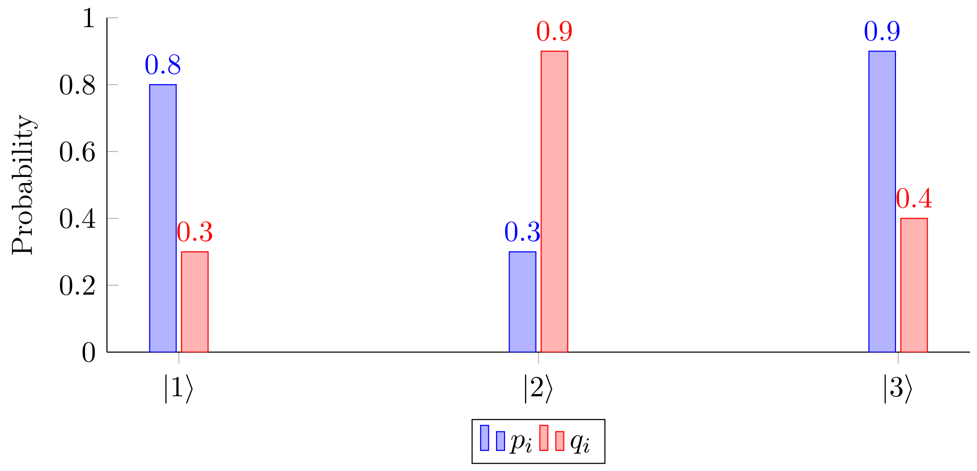
$$\rho_1 = \begin{pmatrix} p_1 & & \\ & \ddots & \\ & & p_d \end{pmatrix} \quad \text{and} \quad \rho_2 = \begin{pmatrix} q_1 & & \\ & \ddots & \\ & & q_d \end{pmatrix}.$$

The optimal strategy is to measure in the standard basis and the problem of discriminating ρ_1 and ρ_2 reduces to a purely classical problem. Now we know that if the state is ρ_1, then with probability p_i you see $|i\rangle$, and if the state is ρ_2, then with probability q_i you see $|i\rangle$.

For example, suppose $d = 3$ and the p_i's and q_i's are displayed in the histogram below. If you see "$|1\rangle$", you should guess ρ_1; if you see "$|2\rangle$", you should guess ρ_2; and if you see "$|3\rangle$", you should guess ρ_1.

Let $A = \{i : p_i \geq q_i\}$. It is optimal to guess ρ_1 if and only if the outcome is in A.

Definition 3.1. The statistical distance / total variation of probability distributions $(p_1, \ldots, p_d)$ and $(q_1, \ldots, q_d)$ is defined by $d_{\text{TV}}(\{p_i\}, \{q_i\}) := \frac{1}{2} \sum_i |p_i - q_i|$



Now we compute the success probability:

$$\begin{aligned}\mathbf{Pr}\left[\text{success}\right] &= \frac{1}{2}\mathbf{Pr}\left[i \in A \mid \rho = \rho_1\right] + \frac{1}{2}\mathbf{Pr}\left[i \notin A \mid \rho = \rho_2\right]\\ &= \frac{1}{2}\sum_{i\in A} p_i + \frac{1}{2}\sum_{i\notin A} q_i\\ &= \frac{1}{2}\sum_i \max\left(p_i, q_i\right)\\ &= \frac{1}{2}\sum_i \left(\frac{p_i+q_i}{2} + \frac{|p_i-q_i|}{2}\right)\\ &= \frac{1}{2} + \frac{1}{2}d_{\mathrm{TV}}(\{p_i\}, \{q_i\}).\end{aligned}$$

3.2 General case

In order to state the result in the general case, we first define the p-norm of a matrix:

Definition 3.2. Suppose matrix A is Hermitian. The p-norm of A is $\|A\|_p := (\sum_i |\lambda_i|^p)^{1/p}$, where λ_i's are the eigenvalues of A. The ∞-norm of A is $\|A\|_\infty := \max_i(\lambda_i)$

In the easy case, the success probability can be rewritten as $\frac{1}{2} + \frac{1}{2}(\frac{1}{2}\|\rho_1 - \rho_2\|_1)$.

Remark 3.3. $\frac{1}{2}\|\rho_1 - \rho_2\|_1$ is called the trace distance between ρ_1 and ρ_2. There are different notations for $\frac{1}{2}\|\rho_1 - \rho_2\|_1$, such as $D, \delta, T, d_{\mathrm{tv}}, d_{\mathrm{TV}}$.

Now we drop the assumption that ρ_1 and ρ_2 are simultaneously diagonalizable, and consider the general case. We have the following result (see [Hol73, Hel69]).

Theorem 3.4 (Holevo–Helstrom)**.** *In general, the best success probability to discriminate two mixed states represented by ρ_1 and ρ_2 is given by $\frac{1}{2} + \frac{1}{2}(\frac{1}{2}\|\rho_1 - \rho_2\|_1)$.*

To prove the theorem, we need the Hölder's inequality for matrices.

Lemma 3.5. *Given two Hermitian matrices A, B. We have that*

$$\operatorname{tr}(AB) \le \|A\|_p \|B\|_q, \forall p, q \in [1, \infty] \ \textit{s.t.}\ \frac{1}{p} + \frac{1}{q} = 1.$$

Proof of Theorem 3.4 assuming Lemma 3.5. Say we have any POVM $\{E_1, E_2\}$ such that E_1, E_2 are PSD and $E_1 + E_2 = I$. Our strategy is to guess ρ_i when the outcome is "i". The success probability can be expressed as

$$\frac{1}{2}\operatorname{tr}(E_1\rho_1)+\frac{1}{2}\operatorname{tr}(E_2\rho_2) = \frac{1}{4}\operatorname{tr}((E_1+E_2)(\rho_1+\rho_2))+\frac{1}{4}\operatorname{tr}((E_1+E_2)(\rho_1+\rho_2)) =: \frac{1}{2}T_1+\frac{1}{2}T_2,$$

the first equality of which uses the linearity of the trace operator. On one hand, because $E_1 + E_2 = I$ and $\operatorname{tr}(\rho_i) = 1$, $T_1 = \frac{1}{2}\operatorname{tr}(\rho_1 + \rho_2) = \frac{1}{2}\operatorname{tr}(\rho_1) + \frac{1}{2}\operatorname{tr}(\rho_2) = 1$. On the other hand, by Lemma 3.5, we have

$$T_2 = \frac{1}{2}\operatorname{tr}((E_1 - E_2)(\rho_1 - \rho_2)) \le \frac{1}{2}\|E_1 - E_2\|_\infty \|\rho_1 - \rho_2\|_1.$$

Since $0 \preceq E_1, E_2 \preceq I$, $\|E_1 - E_2\|_\infty \le 1$ and so $T_1 + T_2 \le \frac{1}{2} + \frac{1}{2}\|\rho_1 - \rho_2\|_1$ gives an upper bound of the best success probability. This bound is achievable. Suppose $\Delta = \rho_1 - \rho_2$ has some eigenvalues $\lambda_1, \ldots, \lambda_m \ge 0$ and $\mu_1, \ldots, \mu_{m'} < 0$. Let P be the eigenspace associated to $\lambda_1, \ldots, \lambda_m$ and Q be the eigenspace associated to $\mu_1, \ldots, \mu_{m'}$. Let E_1 be the projection into P and let E_2 be the projection into Q. Note that $E_1\Delta$ has eigenvalues $\lambda_1, \ldots, \lambda_m$ and $E_2\Delta$ has eigenvalues $\mu_1, \ldots, \mu_{m'}$. We check that $T_2 \le \frac{1}{2}\|\Delta\|_1$ can achieve equality.

$$T_2 = \frac{1}{2}\operatorname{tr}((E_1 - E_2)\Delta) = \frac{1}{2}\operatorname{tr}(E_1\Delta) - \frac{1}{2}\operatorname{tr}(E_2\Delta) = \frac{1}{2}\sum \lambda_i - \frac{1}{2}\sum \mu_i = \frac{1}{2}\|\Delta\|_1.$$

□

Proof of Lemma 3.5. Since A, B are Hermitian, we can write

$$A = \sum_{i=1}^{d} p_i |u_i\rangle \langle u_i|, B = \sum_{j=1}^{d} q_j |v_j\rangle \langle v_j|,$$

where both $\{|u_i\rangle\}$ and $\{|v_i\rangle\}$ are orthonormal bases. Now we rewrite the left hand side of the inequality in terms of p_i, q_j, u_i, v_j and then apply the (usual) Hölder's inequality:

$$\begin{aligned}
\operatorname{tr}(AB) &= \operatorname{tr}\left(\sum_i p_i |u_i\rangle\langle u_i| \sum_j q_j |v_j\rangle\langle v_j|\right) \\
&= \sum_{i,j} p_i q_j \operatorname{tr}(|u_i\rangle\langle u_i|v_j\rangle\langle v_j|) \\
&= \sum_{i,j} p_i q_j |\langle u_i|v_j\rangle|^2 \\
&= \sum_{i,j} \left(p_i |\langle u_i|v_j\rangle|^{2/p}\right)\left(q_j |\langle u_i|v_j\rangle|^{2/p}\right) \\
&\le \left(\sum_{i,j} p_i^p |\langle u_i|v_j\rangle|^2\right)^{1/p} \left(\sum_{i,j} q_i^q |\langle u_i|v_j\rangle|^2\right) && \text{(the Hölder's inequality)} \\
&= \left(\sum_i p_i^p \sum_j |\langle u_i|v_j\rangle|^2\right)^{1/p} \left(\sum_j q_j^q \sum_i |\langle u_i|v_j\rangle|^2\right) \\
&= \left(\sum_i p_i^p\right)^{1/p} \left(\sum_j q_j^q\right)^{1/q} && \text{(the Pythagorean Theorem)} \\
&= \|A\|_p \|B\|_q .
\end{aligned}$$

□

Finally, Theorem 2.1 is a corollary to Theorem 3.4. Consider the case when $\rho_i = |\psi_i\rangle\langle\psi_i|$ for $i = 1, 2$ and $\langle\psi_1|\psi_2\rangle = \langle\psi_2|\psi_1\rangle = \cos\theta$. The best (average) success probability is $\frac{1}{2} + \frac{1}{2}\|\rho_1 - \rho_2\|_1$. Observe that $\rho_1 - \rho_2 = |\psi_1\rangle\langle\psi_1| - |\psi_2\rangle\langle\psi_2|$ has $(d-2)$ zero eigenvalues in the subspace perpendicular to both $|\psi_1\rangle$ and $|\psi_2\rangle$. The other two eigenvalues are associated to eigenvectors of the form $|\psi\rangle = c_1|\psi_1\rangle + c_2|\psi_2\rangle$. If $|\psi\rangle = c_1|\psi_1\rangle + c_2|\psi_2\rangle$ is an eigenvector, then

$$(\rho_1-\rho_2)|\psi\rangle = (|\psi_1\rangle\langle\psi_1| - |\psi_2\rangle\langle\psi_2|)(c_1|\psi_1\rangle + c_2|\psi_2\rangle) = (c_1 - c_2\cos\theta)|\psi_1\rangle + (c_1\cos\theta - c_2)|\psi_2\rangle .$$

So the associated eigenvalue λ satisfies $\lambda c_1 = c_1 - c_2\cos\theta$ and $\lambda c_2 = c_1\cos\theta - c_2$. In other words, the homogeneous system of linear equations

$$(\lambda-1)c_1 + (\cos\theta)c_2 = 0, \quad (-\cos\theta)c_1 + (\lambda+1)c_2 = 0.$$

has a non-trivial solution, and so

$$\det\begin{pmatrix} \lambda-1 & \cos\theta \\ -\cos\theta & \lambda+1 \end{pmatrix} = \lambda^2 - 1 + \cos^2\theta = \lambda^2 - \sin^2\theta = 0.$$

Therefore, the only two nonzero eigenvalues are $\pm\sin\theta$, hence the best success probability is $\frac{1}{2} + \frac{1}{4}\|\rho_1 - \rho_2\|_1 = \frac{1}{2} + \frac{1}{4}(\sin\theta + \sin\theta) = \frac{1}{2} + \frac{1}{2}\sin\theta$.

References

[Die88] Dennis Dieks. Overlap and distinguishability of quantum states. *Physics Letters A*, 126(5):303–306, 1988.

[Hel69] Carl W Helstrom. Quantum detection and estimation theory. *Journal of Statistical Physics*, 1(2):231–252, 1969.

[Hol73] Alexander S Holevo. Statistical decision theory for quantum systems. *Journal of Multivariate Analysis*, 3(4):337–394, 1973.

[Iva87] Igor D Ivanovic. How to differentiate between non-orthogonal states. *Physics Letters A*, 123(6):257–259, 1987.

[Per88] Asher Peres. How to differentiate between non-orthogonal states. *Physics Letters A*, 128(1):19, 1988.

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

Lecture 18: Quantum Information Theory and Holevo's Bound

November 10, 2015

Lecturer: John Wright *Scribe: Nicolas Resch*

1 Question

In today's lecture, we will try to answer the following question:

$$\boxed{\text{How many bits are in a qubit?}}$$

What's the answer? Well, we'll see... In reality, there's more than one way to answer this question. So we'll have to be a little bit more precise if we want to say anything interesting. In this lecture, we will motivate and formally state Holevo's Theorem [Hol73], which does a good job of answering our question.

We begin by considering the following scenario. We have two parties Alice and Bob, and Alice has a string $x \in \{0,1\}^n$ that she would like to transmit to Bob. Classically, we would do the following: Alice would store the string in some shared memory (say, a RAM), and then Bob would read the string from the RAM.

$$\begin{tikzcd} \begin{array}{c}\text{Alice}\\ x \in \{0,1\}^n\end{array} \arrow[r, "\text{store}"] & \boxed{\text{RAM}} \arrow[r, "\text{read}"] & \begin{array}{c}\text{Bob}\\ x \in \{0,1\}^n\end{array} \end{tikzcd}$$

The above scheme works assuming the RAM is big enough, i.e. it can store at least n bits.

Now, since this is a quantum computation course, it is natural to ask if we can do better quantumly, perhaps with respect to the number n? The previous scheme looks as follows in the quantum setting:

$$\begin{tikzcd} \begin{array}{c}\text{Alice}\\ x \in \{0,1\}^n\end{array} \arrow[r, "\text{compute}"] & \boxed{|\psi_x\rangle \in \mathbb{C}^d} \arrow[r, "\text{measure}"] & \begin{array}{c}\text{Bob}\\ x \in \{0,1\}^n\\ \text{(Hopefully)}\end{array} \end{tikzcd}$$

In the above, Alice does some sort of computation to create the state $|\psi_x\rangle \in \mathbb{C}^d$, which depends on her input x. The scheme works only if $d \geq 2^n$. Otherwise there will exist non-orthogonal vectors $|\psi_x\rangle$ and $|\psi_y\rangle$ for $x \neq y$. In the previous lecture, we showed that we are only able to discriminate quantum states with probability 1 if they are orthogonal, so we cannot have this if we want Bob to be guaranteed to recover the correct string x. Since $d \geq 2^n$, we require n qubits.

So, it seems like we've answered our question: we need n qubits to represent n bits. Are we done? Well, we can say a little more.

2 More General Scenario

First of all, it might happen that we don't actually need n bits to represent x, even in the classical setting. As a very simple case, it could very well happen that Bob already knows x, so Alice needs to store 0 bits! To deal with this, we need some way to quantify how much Bob knows about Alice's input x. And to do this, we will need to quantify how much uncertainty there is about Alice's input. We will use the following uncertainty model:

$$\boxed{\text{Alice samples random message } x \in \{0,1\}^n \text{ with probability } p(x).}$$

If all the $p(x)$'s are positive, then we still need n bits to encode all the possible options classically and, just as before, we require n qubits in the quantum setting. But, it is possible that we could do better.

Let's now try to turn our question around slighty. We will now think of d, the dimensionality of the quantum state prepared by Alice, as fixed in advance. Our question will now be: *How much information can Bob learn about x?*

Before continuing, we remark that Alice need not send a vector. As we've seen in recent lectures, quantum states are in the most general case represented by a density matrix, so Alice may send $\sigma_x \in \mathbb{C}^{d\times d}$, a mixed state. Also, Bob can perform the more general quantum measurements that we discussed recently. Hence, the scenario we will analyze is the following:[1]

- Alice samples $X \in \Sigma \subseteq \{0,1\}^n$, where $X = x$ with probability $p(x)$.
- Alice sends $\sigma_X \in \mathbb{C}^{d\times d}$.
- Bob picks POVM's $\{E_y\}_{y\in\Gamma}$, where $\Gamma \subseteq \{0,1\}^n$.
- Bob measures σ_X, and receives output "$Y \in \Gamma$", where $Y = y$ given $X = x$ with probability $\mathrm{tr}(E_y \sigma_x)$.
- Bob tries to infer X from Y.

$$\begin{tikzcd}
\begin{matrix}\text{Alice}\\ X \sim \Sigma\end{matrix} \arrow[r, "\text{compute}"] & \boxed{\sigma_X \in \mathbb{C}^{d\times d}} \arrow[r, "\text{measure}"] & \begin{matrix}\text{Bob}\\ Y \sim \Gamma\end{matrix}
\end{tikzcd}$$

The most natural thing to do is to put $\Gamma = \Sigma$, and if Bob observes "y" he guesses "y". This is essentially without loss of generality, but we will still consider the situation where Σ and Γ may be different.

[1]We now think of Alice's input, and therefore Bob's measurement, as random variables. Hence the capitalization.

3 Analysis

What's Bob's perspective in this game? He sees σ_x with probability $p(x)$, and he's tasked with discriminating between the different σ_x's. This is very reminiscent of the state discrimination problem we saw last time. However, in this case, we can try to come up with σ_x's that are very easy to discriminate from each other (recall that Alice and Bob are working together).

Bob sees the mixed state

$$\begin{cases} \sigma_{x_1} & \text{with prob. } p(x_1), \\ \sigma_{x_2} & \text{with prob. } p(x_2), \\ & \vdots \end{cases}$$
$$\equiv \sum_{x\in\Sigma} p(x)\sigma_x =: \rho_B.$$

Since each σ_x is a probability distribution over pure states, ρ_B is itself a probability distribution over probability distributions of pure states. We do therefore obtain a probability distribution over pure states, as one would expect.

Alice sees

$$\begin{cases} |x_1\rangle & \text{with prob. } p(x_1), \\ |x_2\rangle & \text{with prob. } p(x_2), \\ & \vdots \end{cases}$$
$$\equiv \sum_{x\in\Sigma} p(x)|x\rangle\langle x| =: \rho_A.$$

Note that ρ_A is precisely the diagonal matrix whose diagonal entries are given by $p(x_1), p(x_2), \ldots$.

To determine the joint state, note that the parties see $|x\rangle\langle x| \otimes \sigma_x$ with probability $p(x)$. Hence, the joint mixed system is

$$\rho := \sum_{x\in\Sigma} p(x)|x\rangle\langle x| \otimes \sigma_x.$$

Recalling an earlier Piazza post, one can indeed verify that $\rho_B = \mathrm{tr}_A(\rho)$ and $\rho_B = \mathrm{tr}_B(\rho)$.

Now that we're dealing with random variables and knowledge, we'll turn to:

4 Classical Information Theory

For further reading on this subject, see [CT12]. In classical information theory, we typically have some random variable X distributed according to P on some set Σ. The most basic question one can ask is:

$$\boxed{\text{How much information do you learn from seeing } X\text{?}}$$

Example 4.1. *Suppose that $\Sigma = \{0,1\}^n$.*

- *If P is the uniform distribution, then one gets n bits of info from seeing X.*
- *If P has all its probability on a single string $x_0 \in \{0,1\}^n$, i.e. $X = x_0$ with probability 1 and $X = x$ with probability 0 for all $x \neq x_0$, then we get 0 bits of information from seeing X.*

We can formalize this with the following definition:

Definition 4.2 (Shannon Entropy)**.** The *shannon entropy* of a random variable X distributed on a set Σ is

$$H(X) = \sum_{x \in \Sigma} p(x) \log \frac{1}{p(x)},$$

where $p(x) = \mathbf{Pr}[X = x]$.

Example 4.3. *Let's verify that this definition matches our intuition, at least for the previous two examples.*

- *If X is uniform, i.e. $p(x) = 1/2^n$ for all $x \in \{0,1\}^n$, then*

$$H(X) = \sum_{x \in \{0,1\}^n} \frac{1}{2^n} \log\left(\frac{1}{1/2^n}\right) = 2^n \frac{1}{2^n} \log(2^n) = n.$$

- *If X has all its probability mass on a single string x_0, then*

$$H(X) = 1 \cdot \log(1/1) + (2^n - 1) \cdot 0 \cdot \log(1/0) = 0 + 0 = 0,$$

where we define

$$0 \log(1/0) := \lim_{x \to 0+} x \log(1/x) = 0.$$

We record here a couple important properties of the shannon entropy function:

- $0 \leq H(X) \leq \log |\Sigma|$.
- H is concave.

So, returning to our earlier scenario, the largest amount of information Bob could hope to learn is $H(X)$. How much does he actually learn? We have two correlated random variables X and Y. We want to know how much knowing Y tells us about X.

In general, if we have random variables X and Y supported on the sets Σ and Γ respectively with joint distribution $P(x,y) = \mathbf{Pr}[X = x, Y = y]$, we have

$$H(X,Y) = \sum_{x \in \Sigma, y \in \Gamma} P(x,y) \log \frac{1}{P(x,y)}.$$

Example 4.4. *Let $\Sigma = \Gamma = \{0,1\}^n$.*

- *Suppose X and Y are independent, uniform random variables. Then (X, Y) is a random $2n$-bit string. So $H(X, Y) = 2n$.*
- *Suppose X is a uniform random variable and $Y = X$. Then Y is also a uniform random variable. However, (X, Y) is basically a random n-bit string. So $H(X, Y) = n$.*

In general, we note that if X and Y are independent, then $H(X, Y) = H(X) + H(Y)$. This seems reasonable, as seeing one of the random variables tells us nothing about the other, so seeing half of the pair (X, Y) only decreases tells us the shannon entropy of the random variable that we observe, but the other random variable still has all of its entropy.

Conversely, if X and Y perfectly correlated, then $H(X, Y) = H(X) = H(Y)$. Indeed, seeing half of the pair (X, Y) immediately tells us what the other half of the pair is, so the amount of entropy in the pair is the same as the amount of entropy in the random variables themselves.

We can formalize the notion of "how much does seeing one random variable tell me about the other" as follows:

Definition 4.5 (Mutual Information)**.** The *mutual information* $I(X; Y)$ between two random variables X and Y is

$$I(X; Y) = H(X) + H(Y) - H(X, Y).$$

This is supposed to represent the amount of information you learn about X from knowing what Y is. Since the definition is symmetric in X and Y, it also represents the amount of information you learn about Y from knowing X.

Example 4.6. *Let's return to our earlier examples.*

- *If X and Y are independent then we see that*

  $$I(X; Y) = H(X) + H(Y) - H(X, Y) = H(X, Y) - H(X, Y) = 0.$$

  This intuitively makes sense, as seeing X tells us nothing about Y (and vice versa) since X and Y are independent.
- *If X and Y are perfectly correlated, then*

  $$I(X; Y) = H(X) = H(Y).$$

  In this case, seeing X tells us everything there is to know about Y (and vice versa), so the mutual information between the random variables is a large as possible.

The symmetry of this definition might be a bit surprising. However, the following example might help explain exactly why this should be the case.

Example 4.7. *Suppose $\Sigma = \{0,1\}^n$ and $\Gamma = \{0,1\}^{2n}$. Let Y be uniformly random on the set Γ, and let $X = (Y_1, \ldots, Y_n)$, i.e. it is the first n bits of Y. Then*

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = n + 2n - 2n = n.$$

This makes sense regardless of the way that we look at it.

- *Suppose I know X. Then, I know the first n bits of Y, i.e. I have n bits of information about Y.*
- *Suppose I know Y. Then, I know all of X, and since X is an n-bit string, I have n bits of information about X.*

5 Quantum Information Theory

With this understanding of classical information theory, we can now rephrase our earlier question. Indeed, the mutual information of X and Y is precisely what we are looking for! Recall that we wanted to know how much information Bob can learn about X. Since Alice and Bob see the joint distribution (X,Y), Bob learns $I(X;Y)$ bits of information about X. Note that $I(X;Y)$ depends on the σ_x's for $x \in \Sigma$ and the E_y's for $y \in \Gamma$.

As we are now looking at this scenario quantumly, we will in fact be studying quantum information theory. For further reading on this subject, see [Wat11].

We now provide an important definition:

Definition 5.1 (Accessible Information)**.** The *accessible information* is

$$I_{\text{acc}}(\sigma, p) = \max_{\substack{\text{over all}\\ \text{POVMs}\\ \{E_y\}_{y\in\Gamma}}} I(X;Y).$$

This represents the best Bob can do given Alice's choice of the σ_x's and the distribution p.

The best overall that the parties can do is therefore

$$\max_{\{\sigma_x\}_{x\in\Sigma}} I_{\text{acc}}(\sigma, p),$$

which can be upper bounded by $H(X) \le \log|\Sigma|$. Our new goal is to relate $I_{\text{acc}}(\sigma, p)$ to the amount of "quantum" information in the σ's. Recall that Bob "sees" the mixed state ρ_B. But how much information is in a mixed state? To answer this question, we need to develop the quantum analogue of Shannon's classical coding theory.

So suppose we have the mixed state

$$\begin{cases} |\psi_1\rangle & \text{with prob. } p_1, \\ |\psi_2\rangle & \text{with prob. } p_2, \\ & \vdots \end{cases}$$

One might initially be inclined to define the quantum entropy to be $H(p)$, where p is the distribution over the $|\psi_j\rangle$'s. However, that would be wrong! This is due to the non-uniqueness of the representation of mixed states. For example, the above mixed state could be indistinguishable from the mixed state

$$\begin{cases} |\varphi_1\rangle & \text{with prob. } q_1, \\ |\varphi_2\rangle & \text{with prob. } q_2, \\ \quad\vdots & \end{cases}$$

even if we have $H(p) \neq H(q)$.

Here is the "correct" way to define the quantum analogue of shannon entropy, due to John von Neumann:

Definition 5.2 (Quantum Entropy)**.** Given a mixed state, let ρ be the density matrix, and suppose it has eigenvalues $\alpha_1, \dots, \alpha_d$ with corresponding eigenvectors $|v_1\rangle, \dots, |v_d\rangle$. We define

$$H(\rho) := \sum_{i=1}^{d} \alpha_i \log \frac{1}{\alpha_i} = H(\alpha).$$

This quantity is often referred to as the von Neumann entropy, and is sometimes denoted $S(\rho)$.

Remark 5.3. One can equivalently define

$$H(\rho) = \operatorname{tr}(\rho \log(1/\rho)).$$

While the matrix $\rho \log(1/\rho)$ might look a little scary, it can be simply thought of as the matrix that has the same eigenvectors $|v_1\rangle, \dots, |v_d\rangle$ as ρ, but the corresponding eigenvalues are now $\alpha_1 \log(1/\alpha_1), \dots, \alpha_d \log(1/\alpha_d)$ (with the same convention that $0 \log(1/0) = 0$). So

$$\rho \log(1/\rho) = \sum_{i=1}^{d} \alpha_i \log \frac{1}{\alpha_i} |v_i\rangle\langle v_i|.$$

Example 5.4.

- *Suppose*

$$\rho = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Then $H(\rho) = 0$.

- *Suppose*

$$\rho = \begin{bmatrix} 1/d & 0 & \cdots & 0 \\ 0 & 1/d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/d \end{bmatrix}$$

  Then $H(\rho) = \log d$. As a "cultural note" we remark that this state is often referred to as the "maximally mixed state".

Finally, we can answer the question: *how much quantum information does Bob get?*

Definition 5.5 (Quantum Mutual Information)**.** If ρ is the joint state of two quantum systems A and B then the *quantum mutual information* is

$$I(\rho_A; \rho_B) = H(\rho_A) + H(\rho_B) - H(\rho).$$

Example 5.6. *Suppose $\rho = \frac{1}{d}\sum_{i=1}^{d} |i\rangle\langle i| \otimes |i\rangle\langle i|$. Then we can mechanically compute ρ_A as follows:*

$$\begin{aligned} \rho_A = \mathrm{tr}_B(\rho) &= \mathrm{tr}_B\left(\frac{1}{d}\sum_{i=1}^{d} |i\rangle\langle i| \otimes |i\rangle\langle i|\right) \\ &= \frac{1}{d}\sum_{i=1}^{d} \mathrm{tr}_B(|i\rangle\langle i| \otimes |i\rangle\langle i|) \\ &= \frac{1}{d}\sum_{i=1}^{d} |i\rangle\langle i| \mathrm{tr}_(|i\rangle\langle i|) \\ &= \frac{1}{d}\sum_{i=1}^{d} |i\rangle\langle i|. \end{aligned}$$

At a more conceptual level, we could have immediately determined that $\rho_A = \frac{1}{d}\sum_{i=1}^{d} |i\rangle\langle i|$, the maximally mixed state, as follows. If Bob observes the state $|i\rangle\langle i|$, Alice's system collapses to $|i\rangle\langle i|$. Since Bob observes $|i\rangle\langle i|$ with probability $1/d$ for $i = 1, \ldots, d$, we conclude that Alice's mixed state is precisely given by the ensemble

$$\begin{cases} |1\rangle & \textit{with prob. } 1/d, \\ |2\rangle & \textit{with prob. } 1/d, \\ & \vdots, \end{cases}$$

which is represented by the density matrix $\frac{1}{d}\sum_{i=1}^{d} |i\rangle\langle i|$.

Similarly, we have $\rho_B = \frac{1}{d}\sum_{i=1}^{d} |i\rangle\langle i|$. We thus have

$$H(\rho_A) = H(\rho_B) = \log d.$$

Moreover, observe that

$$H(\rho) = \log d,$$

as ρ is essentially two perfectly correlated copies of the maximally mixed state. Hence,

$$I(\rho_A; \rho_B) = \log d + \log d - \log d = \log d.$$

Example 5.7. *If $\rho = \rho_A \otimes \rho_B$, then $I(\rho_A; \rho_B) = 0$.*

We can now define how much quantum information Bob gets from seeing Alice's state: it is precisely $I(\rho_A; \rho_B)$. This is such an important quantity that it gets its own name.

Definition 5.8 (Holevo Information)**.** The *Holevo information* is

$$\chi(\sigma, p) := I(\rho_A; \rho_B).$$

Next time, we will prove Holevo's Theorem:

Theorem 5.9 (Holevo [Hol73])**.** *$I_{\text{acc}}(\sigma, p) \leq \chi(\sigma, p) \leq \log d$.*

References

[CT12] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[Hol73] Alexander Semenovich Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973.

[Wat11] John Watrous. Theory of quantum information. *University of Waterloo Fall*, 2011.

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

Lecture 19: Quantum Channels and Finishing Holevo's Bound

November 11, 2015

Lecturer: Ryan O'Donnell *Scribe: Connell Donaghy*

1 Quantum Channels

As we move forward in this class, we will be transitioning more from quantum algorithms into quantum information theory. Quantum information theory is more relevant today, as there is more active physics research going on around quantum information theory than quantum computers. To begin talking about quantum information theory, we will first define and show examples of **quantum channels**. Quantum channels are of particular interest as both storing and transmitting quantum information is very non-trivial, due to issues such as noise and degradation [Gri12].

Definition 1.1. A quantum channel is any operation which takes in some density matrix ρ, and outputs a density matrix ρ'

Here's a simple sketch of a quantum channel:

$$\rho \quad \text{——}\boxed{\Phi}\text{——} \quad \rho'$$

Let's discuss a few examples of quantum channels, and we'll discorver we've been working with many of them before!

First, consider a simple unitary gate being applied to some quantum state ρ. This will output a new quantumstate ρ' We see

$$\rho' \to \mathcal{U}\rho\mathcal{U}^\dagger$$

Thus, applying a valid quantum gate is actually a valid quantum channel!

Now, we consider a more randomized quantum channel. Consider taking some input state ρ, and with probability $\frac{1}{2}$ applying a unitary gate $\mathcal{U}$, otherwise just outputting our input state ρ. In this case, we have

$$\rho' \to \frac{1}{2}\left(\mathcal{U}\rho\mathcal{U}^\dagger + \rho\right)$$

Next we consider another randomized example called a **mixed unitary channel**. Thus means we have some series of r unitary gates $\mathcal{U}_i$, and each of them get applied with probability p_i. Thus, our output state for a mixed unitary channel is

$$\rho' \to \sum_{i=1}^{r} p_i \mathcal{U}\rho\mathcal{U}^\dagger$$

Another interesting yet trivial quantum channel is to just ignore our input ρ, and output some other ρ_0 regardless of what our input is. In this case, we have

$$\rho' \to \rho_0$$

Next, we consider an interesting case in which we measure in our standard basis, but actually "forget" our outcome. Consider our simple qubit $|\psi\rangle$ that we first described in lecture 1, which has the form $\alpha|0\rangle + \beta|1\rangle$. This qubit, when considered as a density matrix takes the form

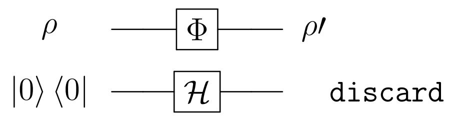
$$|\psi\rangle\langle\psi| = \rho = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}$$

We can see that this state will output $|0\rangle$ with output probability $|\alpha|^2$, and output $|1\rangle$ with probability $|\beta|^2$. Thus, if we consider dephasing this matrix, by measuring and then forgetting the outcome, our channel output will be

$$\rho' \to \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}$$

This channel is commonly referred to as a **complete dephasing channel**.

Finally, as our most important example, we consider adding some register "B", which we initialize to $|0\rangle\langle 0|$. Next, we take our input state ρ along with this register, and apply some giant unitary operation $\mathcal{U}$ on the whole system $\rho \otimes |0\rangle\langle 0|$. Next, we once again measure but forget B, and let ρ' be our output. We can see this as a quantum circuit, by looking at the following:



According to physics, this type of quantum channel is actually the most general quantum channel we can describe. Examples one through five can all be described as this type of quantum channel, as well as any other valid quantum channel which can be dreamt up. Also, we must keep in mind that if we let $\dim\rho = d$ and $\dim\rho' = d'$, and we don't require at $d' = d$. Thus, our output matrix can be different dimensions than our input matrix. This is because we can either measure and forget some qubits from our input, or we can incorporate some qubits from B into ρ'. This is called a quantum channel, or a superoperator.

Definition 1.2. A quantum channel is a set of **completely positive trace preserving** operators, which from here on out we'll abbreviate as CPTP operators. This means that the operators are both linear and completely positive.

Remark 1.3. In the classical (or randomized) sense, we can actually represent a quantum channel as a $d' \times d$ stoachastic matrix. From this intuition, we can see that a quantum channel can be thought of as one step of a Markov Chain.

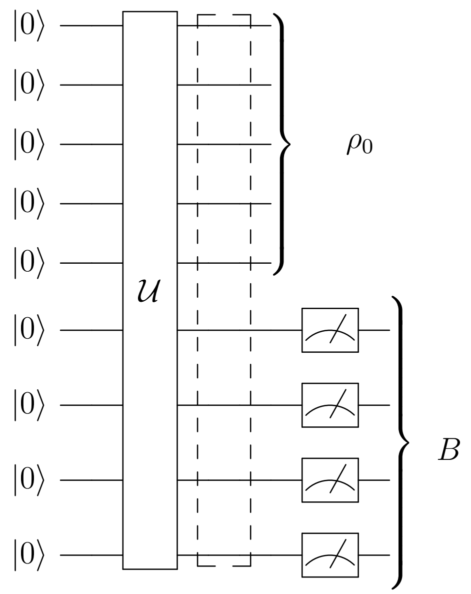
Theorem 1.4. *Every quantum channel has a "Kraus" representation of the channel This representation looks as follows*

$$\rho \;—\boxed{\Phi}—\quad \textstyle\sum_{i=1}^{r} \mathcal{K}_i \rho \mathcal{K}_i^\dagger$$

For this representation, we require that $\mathcal{K}_i \in \mathbb{C}^{d' \times d}$, and that $\sum_{i=1}^{r} \mathcal{K}_i^\dagger \mathcal{K} = I$. [NC11]

Let's touch on some interesting things we can interpret given this definition of a quantum channel. First, consider the case where we output ρ_0, and our input state ρ is 1 dimensional. This case is called state preparation, or something that we've in general taken for granted in class. Now, we can consider any initial state we need as coming from a quantum channel which has no input. In the reverse case, we can have $d' = 1$. In this case, ρ is simply discarded, so this quantum channel would discard a state

Lets consider creating some output state ρ_0 using a quantum circuit. First, we put in some ancillary qubits, and then we discard B bits, after measuring then forgetting them, and then take our output state which is $\rho_0 = \text{tr}_B \left|\psi\right\rangle_{AB}$. We can consider the circuit below to find this definition. We actually call our intermediate pure state $\left|\psi\right\rangle_{AB}$ a pure state. looking at this quantum channel and our mixed output state ρ_0, we note that *every mixed state is a purification of some pure state.*



The boxed area after the application of our giant unitary operator $\mathcal{U}$ is clearly some pure state, as it originates from a unitary transformation on our ancillas. Interestingly, we actually call this boxed state $\left|\psi\right\rangle$ the *purification* of ρ_0.

2 Finishing Holevo's Bound

Now, let's return to our story of Alice and Bob. In this story, Alice has some ensemble of mixed states, which well call $\mathcal{E}$. This ensemble consists of some density matrices σ_x, and each density matrix has some probability $p(x)$ of being sent over to Bob. Now, we define a random variable X, which is in itself a mixed state, which sums ovar all of Alice's possible density matrices with their associated probabilities. Since X is a sum of mixed states, X itself is actually a mixed state, which has the property that the probability $P(X = x) = p(x)$. Alice prepares and sends some density matrix σ_X over to bob. The ultimate question we're looking to answer here is : how much information about X can bob determine from σ_X? To answer this question, let's look into what Bob will do with this σ_X.

Essentially, all Bob can really do with this σ_X is measure it in his favorite way, which produces a classical outcome $Y \in \Gamma$. Bob's mixed state which he measures on, is

$$\rho_B = \sum_{x \in \Sigma} p(x) \sigma_x$$

This makes the overall joint state between Alice and Bob look like the following

$$\rho_{AB} = \sum_{X \in \Sigma} \rho(x) |x\rangle \langle x| \otimes \sigma_x$$

Now, let's get back to our question : *How much classical info can bob get on X?* To look at this, we're going to go back to the previous lecture and what we learned about mutual information, or $I(X, Y)$.

Claim 2.1. *By Holevo's theorem, we have an upper limit on* $I(X, Y) \leq \chi(\mathcal{E})$

Recall our previous definition of $I(X, Y) = H(X) + H(Y) - H(X, Y)$, where each $H(X)$ is the Shannon Entropy, defined as $H(X) = \sum_{x \in \Sigma} p(x) \log(\frac{1}{p(x)})$.

Claim 2.2. $I(X, Y) = H(Y) - H(Y|X)$

Before going into the proof of this we'll first note that we mean $H(Y|X)$ to be defined as

$$H(Y|X) = \sum_{x \in \Sigma} \rho(x) H(Y_x)$$

Where $H(Y_x)$ is the Shannon entropy of Y when we condition that $X = x$.

Proof. It will suffice to show that $H(X, Y) = H(X) + H(Y|X)$, because if we prove this then we can simply show that

$$\begin{aligned} I(X, Y) &= H(X) + H(Y) - H(X, Y) && (1) \\ &= H(X) + H(Y) - H(X) - H(Y|X) && (2) \\ &= H(Y) - H(Y|X) && (3) \end{aligned}$$

To make the proof manageable, we will do this for a classical X and Y, but the result does hold for quantum random variables as well. Keep in mind that Y_x is just Y conditioned on

$X = x$

$$\begin{aligned}
H(X,Y) &= \sum_{x,y} p(x,y)\log(\frac{1}{p(x,y)}) && (4)\\
&= \sum_{x,y} p(x)Pr[Y_x = y]\log(\frac{1}{p(x)Pr[Y_x = y]}) && (5)\\
&= \sum_{x,y} p(x)Pr[Y_x = y](\log(\frac{1}{p(x)}) + \log(\frac{1}{Pr[Y_x = y]})) && (6)\\
&= \sum_{x,y} p(x)Pr[Y_x = y]\log(\frac{1}{p(x)}) + \sum_{x,y} p(x)Pr[Y_x = y]\log(\frac{1}{Pr[Y_x = y]}) && (7)\\
&= \sum_{x} p(x)\log(\frac{1}{p(x)}) + \sum_{x} p(x)H(Y_x) && (8)\\
&= H(X) + H(Y|X) && (9)
\end{aligned}$$

□

Now that we've shown that $H(X,y) = H(X) + H(Y|X)$, let's go on to answer another question, what exactly is $\chi(\mathcal{E})$. (Additionally, remember that our ensemble is $\mathcal{E} = \{p(x), \sigma_x\}$ with $x \in \Sigma$, so this is the same χ as in the lecture 18 scribe notes.

Definition 2.3. Before giving this definition, we define $I(\rho_A;\rho_B)$ to be the quantum mutual information between two density matrices, which is defined whenever you have a mixed state over two registers. Also, we define $H(\rho)$ to be the quantum, or Von Nuemann entropy of some mixed state. With these definition in mind, we define our Holevo information $\chi(\mathcal{E})$ to be

$$\chi(\mathcal{E}) = I(\rho_A;\rho_B) = H(\rho_A) + H(\rho_B) - H(\rho_{AB})$$

Another common definition which is used for the Holevo Information is the following:

Definition 2.4.

$$\chi(\mathcal{E}) = H(\rho_B) - \sum_{x\in\Sigma} \rho(x)H(\sigma_x)$$

Now, lets look at a few useful pieces of information about Von Nuemann entropy and Holevo Information

Remark 2.5. The Holevo Information of an ensemble is always non-negative, because Von Nuemann entropy is concave.

Remark 2.6. $\chi(\mathcal{E}) \leq H(\rho_B)$. This is easy to see from definition 2.4 and the non-negativity of Von Nuemann entropy.

Corollary 2.7. *A quantum channel which sends* $\log_2 d$ *qubits can convey at most an equivalent* $\log_2 d$ *classical bits of information between Alice and Bob [Hol73].*

Although quantum information gains no advantage over classical information in this sense, if Alice and Bob share enough EPR pairs, they actually only need $n/2$ qubits via a technique called *superdense coding* [Aar14].

An interesting theorem which Holevo and others showed in 1997 was that the upper bound of $\chi(\mathcal{E})$ mutual information to be shared between Alice and Bob is acheivable in the case of repeated communication. [SW97]. To connect this theorem concretely back to our discussion, we consider Alice drawing n random variables X, and producing $X^n = (x_1, \cdots, x_n)$. Alice then sends $\sigma_{x_1} \otimes \sigma_{x_2} \cdots \otimes \sigma_{x_n}$ to Bob, who makes n separate measurements to produce a Y^n such that their mutual information takes the following form:

$$I(X^n, Y^n) = \chi(\mathcal{E}) - O_{n\to\infty}(1)$$

3 More comments on Holevo Information

Now, let's consider Alice sending her ensemble through some noisy channel as follows

$$\textit{Alice} \quad — \quad \{p(x), \sigma_x\} \quad ——\boxed{\Phi}—\mathcal{E}' = \{p(x), \Phi(\sigma_x)\}— \quad \textit{Bob}$$

We now define the **Holevo Capacity** of the noise Φ, which is equivalent to the number of bits you can reliably communicate

$$\chi(\Phi) = \max_{\mathcal{E}}(\chi(\mathcal{E}'))$$

Now, lets delve more into the idea (which we won't prove due to time constraints and lack of relevance), that $I(X, Y) = \chi(\mathcal{E})$

Theorem 3.1. *Von Nuemann entropy has the property that it is strongly subadditive. That is to say, if we have some τ_{ABC}, a tripartite quantum state (think of a state over three registers). Now, consider τ_B to be measuring and forgetting registers A and C. Strong subadditivity is to say that*

$$H(\tau_B) + H(\tau_{ABC}) \leq H(\tau_{AB}) + H(\tau_{BC})$$

Since Von Neumann entropy is strongly subadditive, it is also subadditive. Subadditivity is actually a case of strong subadditivity where B is 0 qubits, or it has degree one. This means that $H(\tau_B) = 0$, and that $H(\tau_{ABC}) = H(\tau_{AC})$. In this case, we can define the subadditivity of Von Neumann entropy to be

$$H(\tau_{AC}) \leq H(\tau_A) + H(\tau_C)$$

Remark 3.2. Because $H(\tau_{AC}) \leq H(\tau_A) + H(\tau_C)$, we can clearly see that $I(\tau_A; \tau_C) \geq 0$, which makes sense, as a negative mutual information doesn't make sense intuitively.

Remark 3.3. Discarding a register never increases the mutual information. By combining subadditivity and strong subadditivity, we can deduce that

$$H(\tau_A) + H(\tau_{BC}) - H(\tau_{ABC}) \geq H(\tau_A) + H(\tau_B) - H(\tau_{AB}) \tag{10}$$
$$I(\tau_A; \tau_{BC}) \geq I(\tau_A; \tau_B) \tag{11}$$

From this deduction, we gain that discarding a register can never help you gain information, which also makes intuitive sense.

Corollary 3.4. *Let τ_{AB} be a 2 register state. Say we then pass the B register through some noise, Φ which praudices τ'_{AB}. We say that $I(\tau_A;\tau'_B) \leq I(\tau_A,\tau_B)$.*

By corollary 3.4, we can actually see the holevo bound, This is eessentially bob taking a register, and then measuring it and forgetting it, which produces

$$I(\tau_A,\tau'_B) \leq I(\tau_A,\tau_B) \tag{12}$$
$$I(X,Y) \leq I(\rho_A,\rho_B) \tag{13}$$

Ultimately, since an upper bound of $\chi(\mathcal{E})$ shared information can be both proven and acheived, quantum information theory doesn't get any magical improvments over classical information theory in the same way which quantum algorithms can improve classical algorithms.

References

[Aar14] Scott Aaronson. Lecture 21: Quantum communication complexity. *6.845 Quantum Complexity Theory*, page 1, 2014.

[Gri12] Robert B. Griffiths. Quantum channels, kraus operators, povms. *33-658 Spring 2012*, pages 2–4, 2012.

[Hol73] Alexander S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problems of Information Transmission*, 9:177–183, 1973.

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

[SW97] Benjamin Schumacher and Michael D. Westmoreland. Sending classical information via noisy quantum channels. *Phys. Rev. A*, 56:131–138, Jul 1997.

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

Lecture 20: Pretty Good Measurement

11/16/15

Lecturer: John Wright *Scribe: Lachlan Lancaster*

1 Introduction: Quantum Hypothesis Testing

So we start with a promise problem: given a set of density matrices $\{\sigma_i\}$ and a quantum state ρ we are promised that ρ is in state σ_i with probability p_i. In the general case we have $i \in [m]$ and of course $\sum_{i=1}^{m} p_i = 1$. Our Goal is then to succesfully identify which of the σ_i that our state ρ is actually in, this is known as *Quantum Hypothesis Testing*. The "real life" analog of this is what you can imagine Bob trying to do for the case of the transported information from alice in terms of the Holevo Bound for information transportation.

Getting at the problem we want to at least maximize our probability of getting the state right. This maximization with respect to both the probabilities on each state as well as with respect to any randomness that our approach employs. We then must choose a Quantum POVM (Positive-Operator Valued Measure) $\{E_i\}$ that carries put a measurement and maximizes our probability of getting the state right.

So say we pick a POVM then we know from our previous lectures that:

$$\Pr[\text{observe } \sigma_i] = \operatorname{Tr}(\sigma_i E_i) \tag{1}$$

So that our overall probability of success, (as in telling that we have the right state), is then:

$$\Pr[\text{success}] = \sum_i p_i \operatorname{Tr}(\sigma_i E_i) \tag{2}$$

So this is the quantity that we will seek to maximize in this lecture, we can see that in the case that $m = 2$, this is quite easy.

1.1 Case $m = 2$

In this case we only have two possible outcomes of density matrices. That is, our state ρ can only be in state σ_1 (with probability p_1) or in state σ_2 (with probability p_2). It is easy to see that we must have $p_2 = 1 - p_1$, and in fact that is the main advantage of this case which results in it being an easy problem. In light of this, we will relabel p_1 as simply p. In this case, using equation (2) we have:

$$\Pr[\text{success}] = p\text{Tr}(\sigma_1 E_1) + (1-p)\text{Tr}(\sigma_2 E_2)$$

Where E_1 and E_2 come from the POVM that we have chosen, which is still arbitrary as of now. We then notice that, due to the nature of the POVM, we are abel to make a similar observation that $E_2 = I - E_1$ where I is the identity matrix. We can then again relabel E_1 as simply E and substituting this into the above we have:

$$\begin{aligned}
\Pr[\text{success}] &= p\text{Tr}(\sigma_1 E) + (1-p)\text{Tr}(\sigma_2(I-E)) \\
&= p\text{Tr}(\sigma_1 E) + (1-p)\left(\text{Tr}(\sigma_2) - \text{Tr}(\sigma_2 E)\right) \\
&= (1-p) + p\text{Tr}(\sigma_1 E) - (1-p)\text{Tr}(\sigma_2 E) \\
&= (1-p) + \text{Tr}\left(E(p(\sigma_1+\sigma_2) - \sigma_2)\right)
\end{aligned}$$

Where we note that above we used the linearity of the trace operation as well as the fact that density matrices are required to have trace one. From the result we see that in order to maximize our probability of success we need only maximize the trace of the matrix in the second part of the above equation $E(p(\sigma_1+\sigma_2)-\sigma_2)$. The way to maximize this trace is then to simply choose E so that is projects onto the positive eigenspace of the matrix it is multiplying, that is the matrix $p(\sigma_1+\sigma_2)-\sigma_2$. Thus we have a direct strategy of picking the best POVM for any set of σ_i, so the problem is solved!

We will see that the interesting case is when $m \geq 3$, this problem is **unsolved** in modern quantum information theory. But, as you might have been able to tell from the title of the lecture, we have some pretty ideas of how to do pretty good with this problem.

2 Some Tractable Cases

We wil now begin to tackle the problem of qunatum hypothesis testing in the case that $m \geq 3$ and see some of the problems that arise along the way in not allowing us to do perfectly well. In order to do this well we will first define some notation/terminology.

Definition 2.1. We witl define the *optimal success probability*, denoted P_s^{OPT}, as the best possible probability of success that you can get given a set of possible density matrices $\{\sigma_i\}$ with corresponding probabolities of observing p_i.

Definition 2.2. We then denote the *optimal error probability*, denoted P_e^{OPT}, as the lowest possible probability of failure given the states as above.

One can easily see that the above two definitions are complementary in teh sense that $P_s^{\text{OPT}} + P_e^{\text{OPT}} = 1$. As we will see later,for $m \geq 3$ we **do not** have a way of finding a POVM that achieves these optimal probabilities. However, we **do** no of a measurement/POVM that does pretty well.

2.1 Pure States

To start to look at cases with $m \geq 3$ we will begin with the relatively easy case where each of the σ_i are pure states and are orthonormal to each other. To be precise, we will deal with the case where $\sigma_i = |v_i\rangle\langle v_i|$ where the set $\{|v_i\rangle\}$ is assumed to be an orthonormal basis set for the Hilbert space under consideration.

We then assert that the best measurement you can choose is then the POVM where $E_i = \sigma_i$. Note that this is in fact a POVM due to the nature of the σ_i and their creation from an orthonormal basis so that $\sum_i \sigma_i = I$. So we see that our probability of success is then:

$$\Pr[\text{success}] = \sum_i p_i \mathrm{Tr}\left(\sigma_i^2\right) = \sum_i p_i \mathrm{Tr}\left(\sigma_i\right) = \sum_i p_i = 1$$

Note we used the fact above that $\sigma_i^2 = (|v_i\rangle\langle v_i|)^2 = |v_i\rangle\langle v_i|v_i\rangle\langle v_i| = |v_i\rangle\langle v_i| = \sigma_i$. So that in this specific case we have a way of always being right! Well that's nice, but it is for quite a particular case. Let's generalize a bit.

2.2 Sum of Pure States

We now consider the case where each of the σ_i can be stated as a linear combination of pure states created from an orthonormal basis. For example we could have:

$$\sigma_1 = \frac{|v_1\rangle\langle v_1| + |v_2\rangle\langle v_2|}{2}, \quad \sigma_2 = |v_3\rangle\langle v_3|, \quad \sigma_3 = \frac{|v_1\rangle\langle v_1| + \cdots + |v_4\rangle\langle v_4|}{4}, \ldots$$

Where we note from the above that we require the density matrices to be properly normalized. So what is the best measurement strategy here? Well before coosing $E_i = \sigma_i$ seemed to work, how about we try that here? As you might be able to guess it's not that simple in this case. To see why consider the case where we have:

$$\sigma_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/2 \end{pmatrix}$$

WWhere tha above matrices are stated in terms of the aformentioned orthonormal basis $|v_i\rangle$. We can see that upon inspection we can do much better than simpy picking $E_i = \sigma_i$, as for the second density matrix this would result in a very low probability. So can we do a slight tweak on this strategy to get somthing better? It turns out that we can. Consider the following

$$E_i = \sigma_i^{-1/2} \sigma_i \sigma_i^{-1/2} \tag{3}$$

We will see that this proves useful but first we should clarify what we mean by $\sigma_i^{-1/2}$. As shown by [Pen55] as well as others, there is a general way to give a sort-of-inverse for all matrices, even if they are not square or of full rank. These matrices are called *pseudo-inverses* the most general sense usually referring the the Moore-Penrose pseudoinverse. This is what we refer to when we write σ_i^{-1}, even though the density matrices are not in general invertible. In the context of these density matrices, which are diagonal (as they are sums of pure states created from an orthonormal basis), this simply corresponds to performing the inverse operation only on the diagonal elements. Additionally the square root in this case refers to taking the square root of the eigenvalues of the matrix or equivalently in this case the diagonal elements. For example we then have in terms of the matrix σ_2 above:

$$\sigma_2^{-1/2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & \sqrt{2} \end{pmatrix}$$

So that in this case:

$$\sigma_2^{-1/2}\sigma_2\sigma_2^{-1/2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

So we see the true sense in which this pseudoinverse really acts as an inverse, but only on the part of the marix that it makes sense to try to take an inverse of. Note that this also works if instead of using $\sigma_i^{1/2}$ we use $S^{-1/2}$ where $S = \sigma_1 + \sigma_2$ (the reader is invited to check this for themselves).

In the more general case of more than two σ_i we generalize $S = \sum_i \sigma_i$ and use these as the elements of our POVM. However, this only works when we are dealing with the case that these σ_i are created as the linear combination of an orthonormal basis. In order to move to the most general case we must relax this assumption.

3 Pretty Good Measurement

We see here that the most general case, that is where we are simply given a set of σ_i is not far off from what we had in the previous section. WE try the reasonable first approximation of setting the $E_i = p_i\sigma_i$ where we are now weighting our POVM by the likelihood that we are going to be seeing a certain state. You might see notice that this doesn't work as:

$$S = \sum_i E_i \neq I \tag{4}$$

So that this is not a valid POVM. We see that in fact this can't be the case as:

$$\text{Tr}(S) = \sum_i \text{Tr}(E_i) = \sum_i p_i \text{Tr}(\sigma_i) = \sum_i p_i = 1 \tag{5}$$

But it is quite clear that if $\mathrm{Tr}(S) = 1$ then $S \neq I$. So maybe we can fix *this* to get a POVM that works. Okay, what if we keep S the same but now use (in analogy with the last section) $E_i = S^{-1/2} p_i \sigma_i S^{-1/2}$, we then see that we have:

$$\sum_i E_i = \sum_i S^{-1/2} p_i \sigma_i S^{-1/2} = S^{-1/2} \left(\sum_i p_i \sigma_i \right) S^{-1/2} = S^{-1/2} S S^{-1/2} = I \tag{6}$$

So we again regain the required property of a POVM. Note that we only truly get the identity if we know that the σ_i span the whole space, but we may assume this without loss of generality as we can simply extend the set while assigning probability zero to these additional σ_i. We will see that this POVM is exactly the measurement after which this lecture is named [HW94].

3.1 Example $m = 3$

We'll now go over a specific example where we hae three possible densities, that is $m = 3$. These three matrices are then:

$$\sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

We then assign the probabilities of measurement $p_1 = p_3 = p/2$ and $p_2 = 1 - p$. This then completely specifies the current problem. We note that if we consider the above density matrices we have know σ_1 will always output $|0\rangle$, σ_2 will output $|0\rangle$ with proabability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$, and σ_3 will always output $|1\rangle$. So we see that the best chance we have of getting this right is $P_s^{\mathrm{OPT}} = \max\{p, 1-p\}$ and we can actually give a way of achieving this optimal strategy.

To do this let's try simply measuring in the standard basis (this will work). So suppose that we use this measurement and the resulting state is $|0\rangle$, what should we do? Well it of course depends on p. We see that the probability that the state was σ_1 and we observe $|0\rangle$ is $\Pr(\sigma_1 \& |0\rangle) = p$ and correspondingly $\Pr(\sigma_2 \& |0\rangle) = p - 1$. This is of course onluy possible as $\Pr(\sigma_3 \& |0\rangle) = 0$. We then see that our choice is simply dependent on whether p of $1 - p$ is bigger, then we simply choose the state (σ_1 or σ_2) that is more likely.

So what happens if we implement the "Pretty Good Measurement" in this case? Well, from equation (2) we can find the probability of success in this situation. We first find what our POVM is by fist finding S.

$$S = \sum_i p_i \sigma_i = \frac{p}{2} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{p}{2} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} + (p-1) \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

So that we have:

$$S^{-1/2} = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{pmatrix}$$

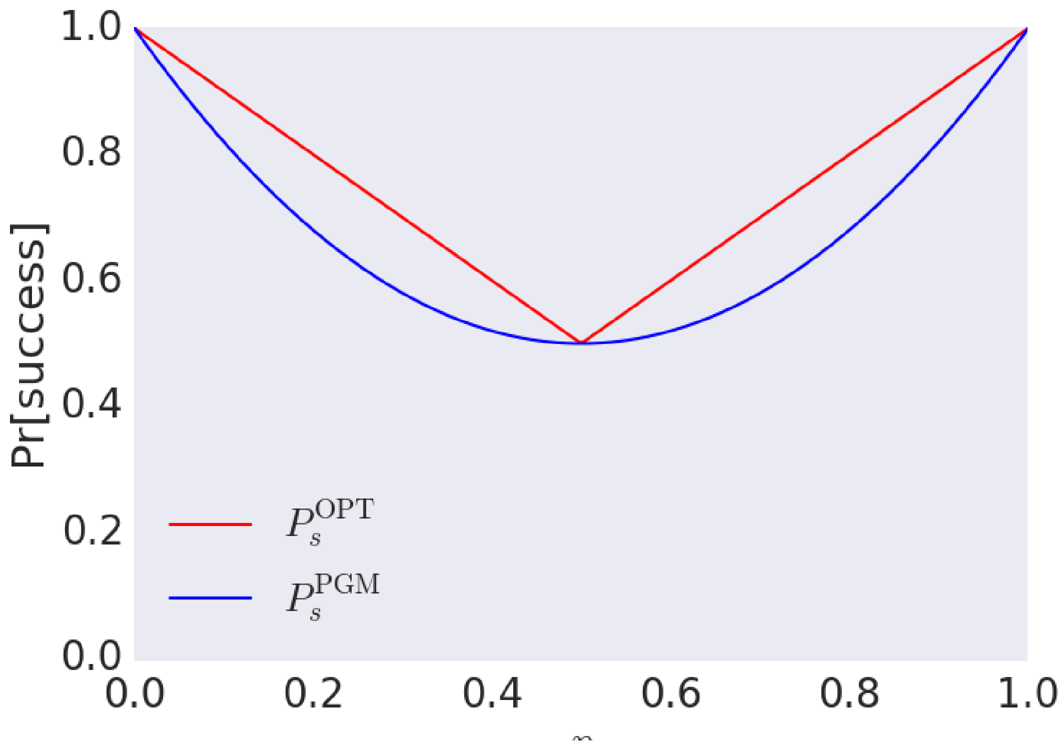
Computing $E_i = S^{-1/2} p_i \sigma_i S^{-1/2}$ we find:

$$E_1 = p\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad E_2 = (1-p)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad E_3 = p\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Finally, using equation (2) we have:

$$P_s^{\mathrm{PGM}} = \frac{p}{2}\mathrm{Tr}(\sigma_1 E_1) + (1-p)\mathrm{Tr}(\sigma_2 E_2) + \frac{p}{2}\mathrm{Tr}(\sigma_3 E_3) = p^2 + (1-p)^2 \tag{7}$$

We can see the comparison of the success probabilites of each model in the graph below. As we can see the Pretty Good Measurement seems to 'smooth' out the success rate in the optimal strategy.



So that we can see very clearly that the Pretty Good Measurement is **not** optimal. So why are we concerned with it? The thing is that while the PGM may not be the best that we can do in this specific case, it is often an optimal strategy. In order to quantify *just how often* we will introduce the following thoerem.

Theorem 3.1. *The error probability of the PGM is bounded above as:*

$$P_e^{\mathrm{PGM}} \leq \sum_{i \neq j} \sqrt{p_i p_j} F(\sigma_i, \sigma_j) \tag{8}$$

This leads us to the question of the mysterious new function that appear in equation (8), this is what we will call the *Fidelity function.*

4 Fidelity Function

We can think of the Fidelity function as a way of measuring "how close" the individual states σ_i are to each other. We have seen similar concepts in past lectures such as the trace distance between two states ρ and σ:

$$d_{\mathrm{Tr}}(\rho, \sigma) = ||\rho - \sigma||_1 \tag{9}$$

Where the subscript 1 above is to indicate that the map being applied is the $\mathcal{L}^1$ norm on the vector consisting of the diagonal elements of $\rho - \sigma$. Note that in this case the more similar the two matrices are, the smaller the trace distance. Though it seems from the placement of the Fidelity function within the above definition, we should have a function that gets larger the more similar the states are. We hence define the Fidelity function as:

Definition 4.1. The Fidelity function for two density matrices ρ and σ is given by:

$$F(\rho, \sigma) = ||\sqrt{\rho}\sqrt{\sigma}||_1 \tag{10}$$

Hence we define the fidelity function as the sum of the square roots of the eigenvalues of the matrix given by $\sqrt{\sigma}\rho\sqrt{\sigma}$.

To make more sense out of what exactly is going on here lets demonstrate this for some diagonal martrices. Say we have $\rho = \mathrm{diag}(a_1, \ldots a_d)$ and $\sigma = \mathrm{diag}(b_1, \ldots b_d)$. Then by $\sqrt{\rho}$ we mean $\sqrt{\rho} = \mathrm{diag}(\sqrt{a_1}, \ldots \sqrt{a_d})$ and by the $\mathcal{L}^1$ norm in this context we mean $||\rho||_1 = \sum_i |a_i|$. We then have that, in this context, $F(\rho, \sigma) = \sum_i \sqrt{a_i}\sqrt{b_i}$. So that we see the larger the fidelity, the more similar the states are, just as we wanted!

Note that since ρ and σ are density matrices we know they have trace 1, so that $F(\rho, \rho) = ||\sqrt{\rho}\sqrt{\rho}||_1 = ||\rho||_1 = 1$. It turns out that this can be made an if and only if statement, so let's go in to some general fidelity function properties:

1. The fidelity function is symmetric in its arguments $F(\rho, \sigma) = F(\sigma, \rho)$.
2. $F(\rho, \sigma) = 1$ if and only of $\rho = \sigma$.

3. If σ is a pure state ($|\psi\rangle\langle\psi|$ for some $|\psi\rangle$) then $\sqrt{\sigma} = \sigma$. In this case $F(\rho, \sigma)$, for some density matrix ρ, is then simply the sum of the square roots of the eigenvalues of the matrix:

$$\sqrt{\sigma}\rho\sqrt{\sigma} = \sigma\rho\sigma = |\psi\rangle\langle\psi|\rho|\psi\rangle\langle\psi| = (\langle\psi|\rho|\psi\rangle)|\psi\rangle\langle\psi| = \chi|\psi\rangle\langle\psi|$$

   Where we have defined $\chi \equiv \langle\psi|\rho|\psi\rangle$, which is simply some scalar. Thus the fidelity is given by $F(\rho, \sigma) = \sqrt{\chi}$, which is quite a simplification.

4. If we have that both of the density matrices (say $\sigma = \langle\psi|\rho|\psi\rangle$ and $\rho = \langle\phi|\rho|\phi\rangle$) are pure states then the Fidelity function is given very easily by a similar analysis to above as:

$$F(\rho, \sigma) = |\langle\phi|\psi\rangle| \tag{11}$$

5. We can actually develop a close realtionship between the trace distance and the fidelity as:

$$1 - F \le d_{\text{Tr}} \le \sqrt{1 - F^2} \tag{12}$$

   Note the quantity $1 - F^2$ id often referred to as the *infidelity.* This then gives that if F is very close to 1, that is $F = 1 - \epsilon$ for small $\epsilon > 0$, then we have a very tight bound on the trace distance as $1 - F = \epsilon \le d_{\text{Tr}} \le \sqrt{1 - F^2} \approx \sqrt{2\epsilon}$.

6. We have a Theorem from Uhlman [Uhl75]:

   Theorem 4.2. *From Uhlman we have the following restatement of the fidelity as:*

$$F(\rho, \sigma) = \max\{|\langle\varphi|\psi\rangle|\} \tag{13}$$

   where the maximum is taken over all possible purifications $|\varphi\rangle$ of ρ and $|\psi\rangle$ of σ.

To conclude we leave off with the general idea that Fidelity is small when your states are "far away" from each other and large when the states are "close" to each other. Returning to the relevance of the Fidelity with respect to the PGM error bound we see that if the fidelity is quite large we have and we have a uniform distribution on our states we have:

$$P_e^{\text{PGM}} \le \frac{1}{2}\sum_{i \ne j}\sqrt{p_i p_j}F(\sigma_i, \sigma_j) \le \frac{1}{2}\sum_{i \ne j}\sqrt{p_i p_j} \le \frac{1}{2}\sum_{i \ne j}\sqrt{m^{-2}} = \frac{m}{2} \tag{14}$$

Where we have assumed above, as the notation indicates earlier in the notes, that there are m possible states. But this is a terrible bound! We don't even know if this is less than one. However, we will see that in some important cases, specifically in the case of the Hidden Subgroup Problem, this will give quite a good bound.

References

[HW94] Paul Hausladen and William K. Wootters. A pretty good measurement for distinguishing quantum states. *Journal of Modern Optics*, 41(12):2385–2390, 1994.

[Pen55] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413, 7 1955.

[Uhl75] A. Uhlmann. The transition probability in the state space of a *-algebra. *Reports of Mathematical Physics*, 9:273–279, 10 1975.

Quantum Computation **(CMU 15-859BB, Fall 2015)**

Lecture 21: HSP via the Pretty Good Measurement

November 18, 2015

Lecturer: John Wright *Scribe: Joshua Brakensiek*

1 The Average-Case Model

Recall from Lecture 20 the following problem. Given the mixed state

$$\sigma = \begin{cases} \sigma_1 & \text{with probability } p_1 \\ \sigma_2 & \text{with probability } p_2 \\ & \vdots \\ \sigma_m & \text{with probability } p_m \end{cases}$$

our goal is to correctly identify which of the m states σ is in as often as possible. To do this, we use a POVM

$$E_1 + \cdots + E_m = I,$$

where $E_1, \dots, E_m$ are positive semi-definite matrices. Upon measuring σ with our POVM, if we yield that measurement $|i\rangle$, then we output "σ_i." For this specific POVM, the success probability is

$$\mathbf{Pr}[\text{success}] = \sum_{i=1}^{m} p_i \cdot \text{tr}(E_i \sigma_i).$$

Since our success probability dependents on the probabilities $p_1, \dots, p_n$, we call this problem the *Average-Case Model*.

We desire to select the optimal POVM $(E_1, \dots, E_m)$ so that our probability of success is maximized. Since it is difficult in general to compute the optimal POVM, we use an accurate, elegant approximation, the Pretty Good Measurement (PGM). To compute analyze the success probability of the PGM, we need to understand the *fidelity* of a pair of mixed states.

Definition 1.1. The *fidelity* of two mixed states ρ and σ is

$$F(\rho, \sigma) = \|\sqrt{\rho}\sqrt{\sigma}\|_1,$$

in which $\sqrt{\rho}$ is the matrix with the same eigenvectors as ρ but with square roots of the eigenvalues of ρ (recall that ρ and σ have nonnegative eigenvalues, so this makes sense) and $\|M\|_1$ is the sum of the absolute values of the eigenvalues of M.

Informally, F is a measure of the similarity between two mixed states. In particular, it has the following two useful properties

- $0 \leq F(\rho, \sigma) \leq 1$, and $F(\rho, \sigma) = 1$ if and only if $\rho = \sigma$.
- $1 - F \leq d_{\text{tr}}(\rho, \sigma) \leq \sqrt{1 - F^2}$, in which $d_{\text{tr}}(\rho, \sigma) = \frac{1}{2}\|\rho - \sigma\|_1$.

Note that the later fact implies that if σ and ρ are 'close' (that is, $F(\sigma, \rho) \sim 1$), then $d_{\text{tr}}(\rho, \sigma) \sim 0$. We then have the following bound of the error probability of the PGM. This result was first proved in a weaker for by [BK02] and was later proved in the following form by [AM14].

Theorem 1.2.

$$\mathbf{Pr}[\textit{PGM errs on } \sigma] \leq \frac{1}{2} \sum_{i \neq j} \sqrt{p_i p_j} F(\sigma_i, \sigma_j).$$

Note that if the σ_i's are pairwise similar, the pairwise fidelities will be quite large, implying a poor bound on the error probability of the PGM. This makes intuitive sense since it is much more difficult to distinguish a collection of objects when they look quite similar. Thus, this theorem is most useful when the pairwise fidelities are small. This theorem implies the following corollary.

Corollary 1.3.

$$\mathbf{Pr}[\textit{PGM errs on } \sigma] \leq m \left(\max_{i \neq j} F(\sigma_i, \sigma_j) \right).$$

Proof. Define $F := \max_{i \neq j} F(\sigma_i, \sigma_j)$. Then, from Theorem 1.2

$$\begin{aligned}
\mathbf{Pr}[\text{PGM errs}] &\leq \sum_{i \neq j} \sqrt{p_i p_j} F \\
&\leq F \sum_{i,j} \sqrt{p_i p_j} \\
&= F \left(\sum_{i=1}^{m} \sqrt{p_i} \right)^2 \\
&\leq F \left(\sum_{i=1}^{m} 1 \right) \left(\sum_{i=1}^{m} p_i \right) \quad \text{(Cauchy-Schwarz inequality)} \\
&= mF.
\end{aligned}$$

□

$$\begin{array}{rc} & \text{Adv. strategies:} \\ & \begin{array}{cccc} \sigma_1 & \sigma_2 & \dots & \sigma_m \\ \downarrow & \downarrow & & \downarrow \end{array} \\ \begin{array}{r} \text{Ph. strategies: } \mathcal{E}_j \rightarrow \\ \mathcal{E}_k \rightarrow \\ \\ \end{array} & \left(\begin{array}{cccc} & \square & & \\ & & & \\ & & & \end{array}\right) \end{array}$$

Figure 1: A matrix representation of the strategies of the Physicist and the Adversary in the 2-player zero-sum game. Each column represents the mixed state the Adversary may pick and each row represents a POVM the Physicist may pick. Note that this matrix cannot exist 'in reality' since there are infinitely many POVMs the Physicist can pick from.

2 The Worst-Case Model

Now, we consider the *Worst-Case Model.* In this case, we are provided with some $\sigma \in \{\sigma_1, \dots, \sigma_m\}$ and we would like to identify the σ_i which σ is with minimal worst-case error probability. For a specific POVM $(E_1, \dots, E_m)$, we defined the probability of success in this model to be

$$\begin{aligned}\mathbf{Pr}[\text{success}] &= \min_i \mathbf{Pr}[\text{guess "}\sigma_i\text{"} : \sigma = \sigma_i] \\ &= \min_i \operatorname{tr}(E_i \sigma_i).\end{aligned}$$

Using Corollary 1.3, we can prove the following. This result is due to [HW12].

Corollary 2.1. *There exists a measurement with worst-case success probability at least*

$$1 - m \cdot \max_{i \neq j} F(\sigma_i, \sigma_j).$$

Proof. The prove this result using a 2-player zero-sum game. This game consists of two players a Physicist and an Adversary.

- The Physicist is given $\sigma \in \{\sigma_1, \dots, \sigma_m\}$. His set of strategies are possible POVMs $\mathcal{E} = (E_1, \dots, E_m)$. Her goal is to *maximize* the worst-case success probability.
- The Adversary possible strategies are to pick $\sigma_i \in \{\sigma_1, \dots, \sigma_m\}$. His goal is to minimize the worst-case success probability.

See Figure 1 for a visualization of these strategies. As is typical in zero-sum game, the Physicist and the Adversary do not have to settle on a single strategy. Instead, they can pick a probability distribution of strategies. Let $\mathcal{D}$ be the Physicists probability distribution of POVMs and let $\mathcal{P}$ be the Adversaries probability distribution of states. Both decide on their strategies simultaneously and independent of the other one. For a particular choice $(\mathcal{D}, \mathcal{P})$ we assign a *score* to be

$$\text{score} = \mathop{\mathbf{E}}_{\mathcal{E}\sim\mathcal{D}} \mathop{\mathbf{E}}_{\sigma_i\sim\mathcal{P}} \mathbf{Pr}[\mathcal{E} \text{ succeeds on } \sigma_i].$$

The inner expected value is equal to the average case success probability of $\mathcal{E}$ on the distribution $\mathcal{P}$.

For the Physicist, this capability of selecting a distribution of $\mathcal{D}$ is redundant.

Fact 2.2. *For any distribution $\mathcal{D}$ of POVMs, there exists a POVM $\widetilde{\mathcal{E}} = (\widetilde{E_1}, \dots, \widetilde{E_m})$ such that for all mixed states σ*

$$\mathop{\mathbf{E}}_{(E_1,\dots,E_m)\sim\mathcal{D}}[\operatorname{tr}(E_i\sigma)] = \operatorname{tr}(\widetilde{E_i}\sigma).$$

That is, $\widetilde{\mathcal{E}}$ is "equivalent" to $\mathcal{D}$.

Proof. For all $j \in \{1, \dots, m\}$, let

$$\widetilde{E_i} = \mathop{\mathbf{E}}_{(E_1,\dots,E_m)\sim\mathcal{D}}[E_i],$$

where the expected value of a random matrix consists of the expected value of each entry. We have that $\widetilde{\mathcal{E}} = (\widetilde{E1i}, \dots, \widetilde{E_m})$ is a POVM because the expected value of a distribution of positive semidefinite matrices is also positive semidefinite and

$$\begin{aligned}\sum_{i=1}^{m} \widetilde{E_i} &= \sum_{i=1}^{m} \mathop{\mathbf{E}}_{(E_1,\dots,E_m)\sim\mathcal{D}}[E_i] \\ &= \mathop{\mathbf{E}}_{(E_1,\dots,E_m)\sim\mathcal{D}}\left[\sum_{i=1}^{m} E_i\right] \\ &= \mathop{\mathbf{E}}_{(E_1,\dots,E_m)\sim\mathcal{D}}[I] = I.\end{aligned}$$

Since trace and matrix multiplication are linear operators, we have that for all σ

$$\begin{aligned}\operatorname{tr}(\widetilde{E_i}\sigma) &= \operatorname{tr}(\mathop{\mathbf{E}}_{(E_1,\dots,E_m)\sim\mathcal{D}}[E_i]\sigma) \\ &= \mathop{\mathbf{E}}_{(E_1,\dots,E_m)\sim\mathcal{D}}[\operatorname{tr}(E_i\sigma)],\end{aligned}$$

as desired. □

If the Adversary's choice of $\mathcal{P}$ is fixed, what POVM should the Physicist chose? She should choose the Pretty Good Measurement! Let $\mathcal{E}$ be the PGM for distribution $\mathcal{P}$ of mixed states. From Corollary 1.3, we have that the score will be

$$\mathbf{Pr}[\mathcal{E} \text{ succeeds on } \mathcal{P}] \geq 1 - m \max_{i\neq j} F(\sigma_i, \sigma_j). \tag{1}$$

Since the choice of $\mathcal{P}$ was fixed, the Adversary can pick the $\mathcal{P}$ which minimized this score. Thus, we have shown that

$$1 - m \max_{i \neq j} F(\sigma_i, \sigma_j) \leq \min_{\mathcal{P}} \max_{\mathcal{E}} \text{score}(\mathcal{E}, \mathcal{P}).$$

By the mini-max theorem for 2-player games [VNM44], we then have that

$$1 - m \max_{i \neq j} F(\sigma_i, \sigma_j) \leq \max_{\mathcal{E}} \min_{\mathcal{P}} \text{score}(\mathcal{E}, \mathcal{P}).$$

Hence, there exists a POVM $\mathcal{E}$ which succeeds with probability at least the RHS of (1) for *any* $\mathcal{P}$. In particular, consider the distributions $\mathcal{P}_i$, $i \in \{1, \ldots, m\}$, which always outputs σ_i. The worst-case success probability of $\mathcal{E}$ is equal to the minimum of the success probabilities of $\mathcal{E}$ on the $\mathcal{P}_i$'s. Thus, the worst-case success probability is bounded from below by the RHS of (1), as desired. □

3 Application of PGM to HSP

We now use the Pretty Good Measurement to devise a polylogarithmic quantum query algorithm for the Hidden Subgroup Problem. Recall the following details of the HSP.

- Let G be a group. You are given quantum query access to $f : G \to [M]$, where $f(g_1) = f(g_2)$ if and only if $g_1 H = g_2 H$, where H is an unspecified subgroup of G.
- We had constructed a quantum circuit which, after a partial measurement, produces the state
$$|gH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |gh\rangle.$$
where $g \in G$ is chosen uniformly at random. In fact, this is equivalent to the mixed state
$$\rho_H = \frac{1}{|G|} \sum_{g \in G} |gH\rangle\langle gH|.$$
- The HSP problem is then, given ρ_H, determine H.

Previously, we had seen that numerous important problems can be reduced to HSP, including factoring, graph isomorphism, and the shortest vector problem. Now, we show that each of these problems has a polylogarithmic quantum query algorithm. This result was originally proved by [EHK04].

Theorem 3.1. *There is a query-efficient algorithm ($q = \text{polylog}\,|G|$ queries) for the HSP on any group G.*

Proof. We can reduce this problem to worst-case discrimination of the states $\{\rho_H\}_{H \leq G}$. If H is our hidden subgroup, we can create the mixed state ρ_H using only one oracle query. We would like to pick a POVM to discriminate among these mixed states using the Pretty Good Measurement. In order to use Corollary 2.1 to bound the worst-case error probability, we need to determine the following two quantities.

- The number of subgroups of G, which we denote by m.
- The maximum fidelity of pairs of mixed states: $\max_{H \neq H'} F(\rho_H, \rho_{H'})$.

We now bound both of these quantities. Proofs of these facts are relegated to the end of the notes.

Fact 3.2.

$$m \leq 2^{\log^2 |G|}.$$

That is, m is *quasipolynomial* in the size of G.

Fact 3.3.

$$\max_{H \neq H'} F(\rho_H, \rho_{H'}) \leq \sqrt{\frac{3}{4}} < .9.$$

Applying Corollary 2.1, we obtain that success probability is at least $1 - 2^{\log^2 |G|} \cdot .9$, which is less than zero when G is not the trivial group! It makes sense though that this attempt failed because we have only queried our oracle once.

The key to achieving a nontrivial success probability is to perform this coset sampling $n \approx \text{polylog}\,|G|$ times in parallel. If H is our hidden subgroup, we can then produce that mixed state $\rho_H \otimes \rho_H \otimes \cdots \otimes \rho_H = \rho_H^{\otimes n}$ using only n quantum queries. Thus, we have turned the problem into worst-case mixed state discrimination of the set $\{\rho_H^{\otimes n}\}_{H \leq G}$. Intuitively, this repetition should help us since if ρ_H and $\rho_{H'}$ have some differences, then $\rho_H^{\otimes n}$ and $\rho_{H'}^{\otimes n}$ should amplify those differences. We formalize this intuition with the following fact.

Fact 3.4. *For all mixed states $\rho, \sigma \in \mathbb{C}^{d \times d}$,*

$$F(\rho^{\otimes n}, \sigma^{\otimes n}) = F(\rho, \sigma)^n.$$

From Facts 3.3 and 3.4 we have that

$$\max_{H \neq H'} F(\rho_H^{\otimes n}, \rho_{H'}^{\otimes n}) < .9^n.$$

Therefore, by Corollary 2.1,

$$\begin{aligned} \mathbf{Pr}[\mathit{success}] &\geq 1 - 2^{\log^2 |G|} \cdot (.9)^n \\ &\geq \frac{1}{2} \text{ when } n = \Theta(\log^2 |G|). \end{aligned}$$

Since our algorithm has the number of queries q equal to n, our algorithm uses polylogarithmically many queries, as desired. □

Remark 3.5. Our use of n tensored copies of the mixed state ρ_H is *not* equivalent to performing the PGM on the single mixed state n times. The correlations the PGM exploited were crucial in obtaining a polylogarithmic number of queries.

Now we provide proofs of the unproven facts.

Proof of Fact 3.2. Let H be an arbitrary subgroup of G and let $\{h_1, \ldots, h_k\}$ be a set of generators of H for which k is minimal. We claim that $k \leq \log_2 |G|$ for all k. To see this, consider the sequence of groups

$$H_0 = \{e\}, H_1 = \text{span}\{h_1\}, H_2 = \text{span}\{h_1, h_2\}, \ldots, H_k = \text{span}\{h_1, \ldots, h_k\} = H.$$

For all $i \in \{0, \ldots, k-1\}$, we have that H_i is a strict subgroup of H_{i+1}. Otherwise, if $H_i = H_{i+1}$, h_{i+1} could be removed from the generators of H, violating minimality of k. Thus, by Lagrange's theorem, $|H_{i+1}|$ is a multiple of $|H_i|$ but must be strictly larger than $|H_i|$. Thus, $|H_{i+1}| \geq 2|H_i|$ for all $i \in \{0, \ldots, k-1\}$. Since $|H_0| = 1$, we have that $|H| = |H_k| \geq 2^k$, implying that $|G| \geq 2^k$, or $\log_2 |G| \geq k$, as desired.

Thus, every subgroup of G can be specified by at most $\log_2 |G|$ generators. In fact, by padding the generating set with copies of the identity element, every subgroup of G can be specified by exactly $\lfloor \log_2 |G| \rfloor$ generators. Since there are $|G|$ choices for each generator, there are at most

$$m \leq |G|^{\lfloor \log_2 |G| \rfloor} \leq 2^{\log^2 |G|}$$

subgroups. □

Remark 3.6. The bound obtained is essentially tight up to constant factors in the exponent. For example, consider the Cartesian product of n copies of the cyclic group $\mathbb{Z}_2$.

Proof of Fact 3.3. Consider any pair $H \neq H'$ of subgroups. To upper bound the fidelity of ρ_H, $\rho_{H'}$, how similar they are, it suffices to lower bound how different they are, their trace distance. Recall that

$$d_{\text{tr}}(\rho_H, \rho_{H'}) \leq \sqrt{1 - F(\rho_H, \rho_{H'})^2}$$

which is equivalent to

$$F(\rho_H, \rho_{H'}) \leq \sqrt{1 - d_{\text{tr}}(\rho_H, \rho_{H'})^2}.$$

Thus, to upper bound the fidelity by $\sqrt{3/4}$, it suffices to lower bound the trace distance by $1/2$. To bound the trace distance, we use a result from Lecture 17 on distinguishing two mixed states in the average-case model. Specifically, consider

$$\rho = \begin{cases} \rho_H & \text{with probability } 1/2 \\ \rho_{H'} & \text{with probability } 1/2 \end{cases}.$$

We showed that the optimal success probability of distinguishing these two states is equal to

$$\frac{1}{2} + \frac{1}{2} d_{\text{tr}}(\rho_H, \rho_{H'}).$$

Thus, to lower bound the trace distance by 1/2, it suffices to give an algorithm which distinguishes the two states with probability at least 3/4.

We now construct a POVM which succeeds with probability at least 3/4. In lecture 20, a key motivation for the PGM is that close-to-optimal POVMs often use a family of PSD matrices quite similar to the mixed states they are trying to distinguish. We strive for something similar (but easier to reason about in our construction). Assume without loss of generality that $|H| \geq |H'|$. Let $\Pi_H = \frac{|G|}{|H|}\rho_H$. We then have our POVM be $(\Pi_H, I - \Pi_H)$, where the first matrix corresponds to measuring 'ρ_H' and the latter matrix corresponds to measuring '$\rho_{H'}$.' We now need to show that we have constructed a valid POVM and that the success probability is at least 3/4.

To establish that $(\Pi_H, I - \Pi_H)$ is a valid POVM, it suffices to show that Π_H is a projection matrix. The simplest way to reason about Π_H is to note that is equals

$$\frac{1}{|H|}\sum_{g\in G}|gH\rangle\langle gH| = \frac{|H|}{|H|}\sum_{C\text{: }H\text{-coset of }G}|C\rangle\langle C|$$

since each H-coset has $|H|$ elements. Note that the set of H-coset states are orthonormal since each element of G appears in exactly one coset. Hence, we have that Π_H is then a projection matrix. Thus, $I - \Pi_H$ is also a projection matrix, so we have a valid POVM.

Next, we bound the success probability of this particular POVM. This success probability equals

$$\begin{aligned}
\mathbf{Pr}[\text{success}] &= \frac{1}{2}\operatorname{tr}(\Pi_H\rho_H) + \frac{1}{2}\operatorname{tr}((I - \Pi_H)\rho_{H'}) \\
&= \frac{1}{2}\operatorname{tr}(\rho_H) \text{ (Π_H is a projection onto the nontrivial eigenvectors of ρ_H)} \\
&\quad + \frac{1}{2}\operatorname{tr}(\rho_{H'}) - \frac{1}{2}\operatorname{tr}(\Pi_H\rho_{H'}) \\
&= 1 - \frac{|G|}{2|H|}\operatorname{tr}(\rho_H\rho_{H'}).
\end{aligned}$$

Now, observe that

$$\begin{aligned}
\frac{|G|}{2|H|}\operatorname{tr}(\rho_H\rho_{H'}) &= \frac{|G|}{2|H|}\operatorname{tr}\left[\left(\frac{1}{|G|}\sum_{g\in G}|gH\rangle\langle gH|\right)\left(\frac{1}{|G|}\sum_{g\in G}|gH'\rangle\langle gH'|\right)\right] \\
&= \frac{1}{2|G||H|}\sum_{g,g'\in G}\operatorname{tr}(|gH\rangle\langle gH|g'H'\rangle\langle g'H'|) \\
&= \frac{1}{2|G||H|}\sum_{g,g'\in G}\operatorname{tr}(\langle gH|g'H'\rangle\langle g'H'|gH\rangle) \\
&= \frac{1}{2|G||H|}\sum_{g,g'\in G}|\langle gH|g'H'\rangle|^2
\end{aligned}$$

Since we defined $|gH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |gh\rangle$, it is easy to see that $\langle gH | g'H' \rangle = \frac{|gH \cap g'H'|}{\sqrt{HH'}}$. Thus, we have that

$$\begin{aligned}
\frac{|G|}{2|H|} \operatorname{tr}(\rho_H \rho_{H'}) &= \frac{1}{2|G||H|^2|H'|} \sum_{g,g' \in G} |gH \cap g'H'|^2 \\
&= \frac{1}{2|G||H|} \sum_{C:H\text{-coset}} \sum_{C':H'\text{-coset}} |C \cap C'|^2 \\
&= \frac{1}{2|G||H|} \sum_{C:H\text{-coset}} \sum_{C':H'\text{-coset}} \sum_{g \in C \cap C', g' \in C \cap C'} 1 \\
&= \frac{1}{2|G||H|} \sum_{g,g' \in G} \mathbf{1}[\exists\ H\text{-coset } C \text{ and } H'\text{-coset } C' \text{ such that } g, g' \in C \cap C'] \\
&= \frac{1}{2|G||H|} \sum_{g \in G} \sum_{g' \in G} \mathbf{1}[g'g^{-1} \in H \cap H'] \\
&= \frac{1}{2|G||H|} \sum_{g \in G} |H \cap H'| \\
&= \frac{|H \cap H'|}{2|H|}.
\end{aligned}$$

Since $H \neq H'$ and $|H| \geq |H'|$, we have that $H \cap H'$ is a strict subgroup of H. Thus, $|H| \geq 2|H \cap H'|$. This implies that

$$\frac{|G|}{2|H|} \operatorname{tr}(\rho_H \rho_{H'}) \leq \frac{1}{4}.$$

Thus, the probability of success is at least $\frac{3}{4}$, yielding that the trace distance is at least $\frac{1}{2}$, so the fidelity is at most $\sqrt{\frac{3}{4}}$, as desired. □

Proof of Fact 3.4. Recall that $F(\rho, \sigma) = \|\sqrt{\rho}\sqrt{\sigma}\|_1$, in which $\sqrt{\rho}$ and $\sqrt{\sigma}$ have the same eigenvectors as ρ and σ, respectively, but with the squareroots of the eigenvalues (mixed states are PSD and thus have nonnegative eigenvalues). Now we seek to show that the tensoring and square root operations are commutative.

Claim 3.7. *For all mixed states ρ, $\sqrt{\rho^{\otimes n}} = \sqrt{\rho}^{\otimes n}$.*

Proof. Since the matrices are Hermitian, it suffices to show that both matrices have identical eigenvalues and eigenvectors. Let $|v_i\rangle, i \in \{1, \ldots, d\}$ be the eigenvectors of ρ and let σ_i be the corresponding eigenvalues. The following table summarizes the eigenvectors and eigenvalues of relevant matrices.

matrix	eigenvectors	eigenvalues
ρ	$\|v_i\rangle$	λ_i
$\rho^{\otimes n}$	$\|v_{i_1}\rangle \otimes \cdots \otimes \|v_{i_n}\rangle$	$\lambda_{i_1} \cdots \lambda_{i_n}$
$\sqrt{\rho}$	$\|v_i\rangle$	$\sqrt{\lambda_i}$
$\sqrt{\rho}^{\otimes n}$	$\|v_{i_1}\rangle \otimes \cdots \otimes \|v_{i_n}\rangle$	$\sqrt{\lambda_{i_1}} \cdots \sqrt{\lambda_{i_n}}$
$\sqrt{\rho^{\otimes n}}$	$\|v_{i_1}\rangle \otimes \cdots \otimes \|v_{i_n}\rangle$	$\sqrt{\lambda_{i_1} \cdots \lambda_{i_n}}$

We can now see that $\sqrt{\rho}^{\otimes n}$ and $\sqrt{\rho^{\otimes n}}$ have identical eigenvectors and eigenvalues. Thus, they are equal, as desired. □

Since we have that the eigenvalues of $\rho^{\otimes n}$ are the products of all n-tuples of eigenvalues of ρ, we have that $\|\rho^{\otimes n}\|_1 = \|\rho\|_1^n$. Applying this fact and the Claim, we have that

$$\begin{aligned}
F(\rho^{\otimes n}, \sigma^{\otimes n}) &= \|\sqrt{\rho^{\otimes n}}\sqrt{\sigma^{\otimes n}}\|_1 \\
&= \|\sqrt{\rho}^{\otimes n}\sqrt{\sigma}^{\otimes n}\|_1 \\
&= \|(\sqrt{\rho}\sqrt{\sigma})^{\otimes n}\|_1 \\
&= \|\sqrt{\rho}\sqrt{\sigma}\|_1^n \\
&= F(\rho, \sigma)^n,
\end{aligned}$$

as desired. □

References

[AM14] Koenraad MR Audenaert and Milán Mosonyi. Upper bounds on the error probabilities and asymptotic error exponents in quantum multiple state discrimination. *Journal of Mathematical Physics*, 55(10):102201, 2014.

[BK02] Howard Barnum and Emanuel Knill. Reversing quantum dynamics with near-optimal quantum and classical fidelity. *Journal of Mathematical Physics*, 43(5):2097–2106, 2002.

[EHK04] Mark Ettinger, Peter Høyer, and Emanuel Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004.

[HW12] Aram W Harrow and Andreas Winter. How many copies are needed for state discrimination? *Information Theory, IEEE Transactions on*, 58(1):1–2, 2012.

[VNM44] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton university press, 1944.

Quantum Computation **(CMU 15-859BB, Fall 2015)**

Lecture 22: Introduction to Quantum Tomography

November 23, 2015

Lecturer: John Wright *Scribe: Tom Tseng*

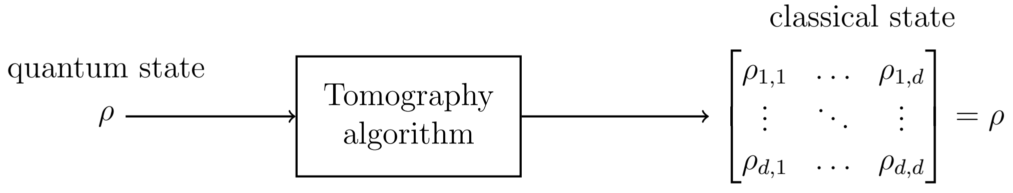
1 Motivation

When conducting research in quantum physics, it is essential that, once we create or are otherwise in possession of a quantum state, we can determine what the state is with high accuracy. For instance, researchers have conducted various experiments that support that the quantum teleportation procedure, which we discussed in lecture 3, works. These experiments involve sending quantum states over long distances. But how does the receiving party verify that the state that they receive is the same as the one that was sent by the other party? This is a problem with significant practical importance and is precisely what tomography aims to solve.

2 The problem

2.1 Definition

Roughly speaking, tomography is the problem where we are given an unknown mixed state $\rho \in \mathbb{C}^{d\times d}$ and where our goal is to "learn" what ρ is. We would like to take the input ρ and put it through some tomography algorithm that will output the classical $d \times d$ state describing ρ.

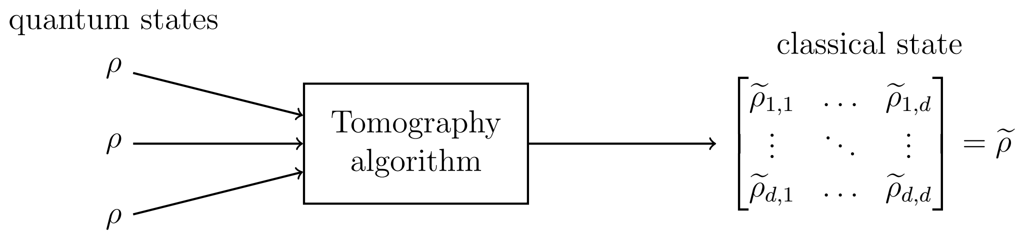

quantum state
ρ
Tomography algorithm
classical state
$\begin{bmatrix}\rho_{1,1} & \cdots & \rho_{1,d} \\ \vdots & \ddots & \vdots \\ \rho_{d,1} & \cdots & \rho_{d,d}\end{bmatrix} = \rho$

This is similar to hypothesis testing, which we have been discussing during the past few lectures. The difference is that we have no prior information as to what ρ is.

Actually, there is a problem with the above description: the goal is impossible to achieve. If we really had an algorithm that could completely learn everything about ρ, then we could replicate ρ, contradicting the no-cloning theorem.

What *can* we say about the state ρ, then? Not much. Suppose we measured ρ with respect to some basis $\{|v_1\rangle, |v_2\rangle, \dots, |v_m\rangle\}$ and observed $|v_i\rangle$. All we learn is that $\langle v_i|\,\rho\,|v_i\rangle$, the probability of observing $|v_i\rangle$, is non-zero, and maybe we can further guess that $\langle v_i|\,\rho\,|v_i\rangle$ is somewhat large. This tells us little useful information about ρ.

Although no perfect tomography algorithm exists, we can make some adjustments to the conditions of the problem to make tomography more tractable. We saw last lecture that when using the Pretty Good Measurement for the hidden subgroup problem, we could generate several copies of the coset state in order to obtain measurement probabilities that were more favorable. That gives our first adjustment: we can stipulate that we are given $n \in \mathbb{N}$ many copies of ρ. In addition, the requirement that we have to output exactly ρ is too restricitive, so our second adjustment will be that we only have to output some $\widetilde{\rho} \in \mathbb{C}^{d\times d}$ that approximates ρ with high probability.


quantum states
ρ
ρ
ρ
Tomography algorithm
classical state
$\begin{bmatrix} \widetilde{\rho}_{1,1} & \cdots & \widetilde{\rho}_{1,d} \\ \vdots & \ddots & \vdots \\ \widetilde{\rho}_{d,1} & \cdots & \widetilde{\rho}_{d,d} \end{bmatrix} = \widetilde{\rho}$

This situation is more sensible. With n copies of ρ, perhaps we can learn enough information to make a fairly precise guess as to what ρ is.

Experiments that test quantum teleportation are verified in this way. In 2012, a group of researchers sent a quantum state between two islands that were 143 miles apart. The result was analyzed by applying quantum tomography to the state, which was sent many times over [MHS+12].

The above discussion motivates the following definition of tomography.

Definition 2.1. In the problem of *quantum state tomography*, the goal is to estimate an unknown d-dimensional mixed state ρ when given $\rho^{\otimes n}$, where $n, d \in \mathbb{N}$. Explicitly, we want a process with the following input and output.

Input: a tensor power $\rho^{\otimes n}$ of the unknown mixed state $\rho \in \mathbb{C}^{d\times d}$, and an error margin $\varepsilon > 0$.

Output: a state $\widetilde{\rho} \in \mathbb{C}^{d\times d}$ such that

$$d_{\mathrm{tr}}(\rho, \widetilde{\rho}) \leq \varepsilon$$

with high probability.

2.2 Optimizing the input

If ρ were given to us in the form $\rho^{\otimes n}$ where n is unimaginably large, we could estimate ρ with great accuracy. However, in reality, preparing so many copies of ρ is expensive. The main question, then, is, given the constraint on the output, *how small can n be*? Here are a couple of results that shed some light on this question.

Fact 2.2. *To solve the tomography problem, it is necessary to have $n \in \Omega\left(d^2/\varepsilon^2\right)$ and sufficient to have $n \in \Theta\left(d^2/\varepsilon^2\right)$.*

Fact 2.3. *If ρ is pure, then it is sufficient to have $n \in \Theta\left(d/\varepsilon^2\right)$.*

If ρ is pure, then it is likely also necessary to have $n \in \Omega\left(d/\varepsilon^2\right)$. However, the best known bound is $n \in \Omega\left(d/\left(\varepsilon^2 \log(d/\varepsilon)\right)\right)$ [HHJ+15].

It makes sense that n need not be as great if ρ is pure, since ρ is then a rank-1 matrix that can be fully described as the outer product of a d-dimensional vector.

At first glance, this looks promising. Quadratic and linear complexity with respect to d seems efficient. However an m-qubit state will have dimension $d = 2^m$. Therefore, in the case of fact 2.2, we would need $n \in \Theta\left(4^m/\varepsilon^2\right)$. This fast exponential growth in the size of the input is the biggest obstacle to quantum tomography at present.

In the remainder of this lecture, we will focus on pure state tomography and prove fact 2.3. The general bound from fact 2.2 is considerably more difficult to prove, so we will not discuss it. For more detail, consult the recent paper written by the instructors of this course [OW15].

3 Pure state tomography

3.1 Relation to hypothesis testing

In pure state tomography, we are given $|\psi\rangle \in \mathbb{C}^d$ in the form $|\psi\rangle^{\otimes n}$, and we want to output $|\widetilde{\psi}\rangle \in \mathbb{C}^d$ such that

$$d_{\text{tr}}\left(|\psi\rangle\langle\psi|, |\widetilde{\psi}\rangle\langle\widetilde{\psi}|\right) \leq \varepsilon.$$

When looking at pure states and their trace distance, we have the following fact.

Fact 3.1. *For $|\psi\rangle, |\widetilde{\psi}\rangle \in \mathbb{C}^d$, the trace distance of $|\psi\rangle$ and $|\widetilde{\psi}\rangle$ satisfies the following equality:*

$$d_{\text{tr}}\left(|\psi\rangle\langle\psi|, |\widetilde{\psi}\rangle\langle\widetilde{\psi}|\right) = \sqrt{1 - \left|\langle\psi|\widetilde{\psi}\rangle\right|^2}.$$

Thus the goal is to make our guess $|\widetilde{\psi}\rangle$ have large inner product with $|\psi\rangle$.

How do we achieve that? This problem resembles worst-case hypothesis testing. In worst-case hypothesis testing, we are given some state $|\psi\rangle^{\otimes n}$ from some set $\{|v\rangle^{\otimes n}\}_{|v\rangle}$, and we try to guess at $|\psi\rangle$. The difference in tomography, of course, is that the set of possibilities is infinitely large and that we have some margin of error.

This problem is also similar to average-case hypothesis testing, which, as we saw last lecture, was related to worst-case hypothesis testing through the minimax theorem. In tomography, we are not given a probability distribution, so it is similar to the situation where we try to guess $|\psi\rangle$ given a uniformly random $|\psi\rangle^{\otimes n} \sim \{|v\rangle^{\otimes n}\}_{\text{unit } |v\rangle}$.

We saw that if we have an optimal algorithm for average-case hypothesis testing, then it is closely related to the optimal algorithm for worst-case hypothesis testing. Thus, to devise the algorithm for tomography, we can get away with just solving the average-case situation on the hardest distribution – the uniform distribution.

3.2 Infinitesimals

The strategy for tomography is that we will merely pretend we are given a probability distribution. With input $|\psi\rangle^{\otimes n}$, we will treat $|\psi\rangle$ as a uniformly random unit vector in $\mathbb{C}^d$.

What does it mean to have a uniformly random unit vector? Since the state is uniformly random, the probability of measuring any particular $|\psi\rangle \in \mathbb{C}^d$ is infinitesimally small. We will write this as

$$\mathbf{Pr}[|\psi\rangle] = d\psi.$$

We need to choose a POVM suitable for tackling this probability distribution. Last lecture, we saw that a good way to deal with average-case hypothesis testing was to run the Pretty Good Measurement, in which, given an ensemble $\{(\sigma_i, p_i)\}_i$ of density matrices and probabilities, we would choose a POVM whose elements resembled the states we knew were in the ensemble. Namely, we took each POVM element E_i to be $p_i\sigma_i$ adjusted by a couple of $S^{1/2}$ factors.

We will reappropriate that technique for our situation. The version of the Pretty Good Measurement for tomography is a POVM with elements resembling

$$E_{|\psi\rangle} \approx d\psi \cdot |\psi\rangle^{\otimes n} \langle\psi|^{\otimes n}.$$

This gives an infinite number of POVM elements, which means that the typical constraint that $\sum_i E_i = I$ needs to be rethought. We instead consider a new constraint that

$$\int_{|\psi\rangle} E_{|\psi\rangle} = I$$

where I is the $n \times n$ identity matrix.

This talk of infinitesimals and integrals might seem non-rigorous, but it is really just informal shorthand that will be notationally convenient for us. A more rigorous analysis could be achieved by instead taking finite-sized POVMs that approximate this infinite scheme and then considering the limit as you make the finite POVMs more and more accurate (and consequently larger and larger in the number of elements).

(A natural question to ask is, “how would we actually implement measurement with a POVM of infinitely many elements?” We would have to take a finite-sized POVM that approximates the infinite POVM. This is not a big sacrifice; we do a similar discretization of randomness in classical algorithms whenever we implement a randomized algorithm that calls for a real number from the uniform distribution over $[0, 1]$.)

3.3 The algorithm

Now we are ready to present the algorithm for pure state tomography that will prove fact 2.3.

Pure State Tomography Algorithm

Input: We are given $|\psi\rangle^{\otimes n}$, where $|\psi\rangle \in \mathbb{C}^d$. We know n and d, which are natural numbers.

Strategy: Measure with the POVM consisting of $\{E_{|v\rangle}\}_{\text{unit vectors } |v\rangle \in \mathbb{C}^d}$, where

$$E_{|v\rangle} = \binom{n+d-1}{d-1} |v\rangle^{\otimes n} \langle v|^{\otimes n} \, dv.$$

Our output: The measurement will give some vector $|\widetilde{\psi}\rangle \in \mathbb{C}^d$, which is what we will output.

Let's do a quick sanity check by making sure our output is sensible. We have

$$\begin{aligned}\mathbf{Pr}[|\widetilde{\psi}\rangle] = \operatorname{tr}\left(E_{|\widetilde{\psi}\rangle} |\psi\rangle^{\otimes n} \langle\psi|^{\otimes n}\right) &= \binom{n+d-1}{d-1} \operatorname{tr}\left(|\widetilde{\psi}\rangle^{\otimes n} \langle\widetilde{\psi}|^{\otimes n} |\psi\rangle^{\otimes n} \langle\psi|^{\otimes n}\right) d\widetilde{\psi} \\ &= \binom{n+d-1}{d-1} \left|\langle\psi|\widetilde{\psi}\rangle\right|^{2n} d\widetilde{\psi} \qquad (1)\end{aligned}$$

where the last equality follows from the cyclic property of the trace operator. Notice that $\left|\langle\psi|\widetilde{\psi}\rangle\right|$ is 1 if $|\widetilde{\psi}\rangle$ is equal to $|\psi\rangle$ and less than 1 otherwise. This signals that the probability of outputting a particular vector is greater if that output vector is close in dot product to the input state, which is what we want.

Before the analyze the accuracy of the algorithm, we need to first verify that $\int_{|\psi\rangle} E_{|\psi\rangle} = I$.

3.4 The integral constraint

The presence of that binomial coefficient in the POVM elements might seem strange, but it is merely a scaling factor that makes the constraint $\int_{|\psi\rangle} E_{|\psi\rangle} = I$ hold. Why does this particular coefficient appear, though?

Notice that the input is guaranteed to be of a certain form, namely, a tensor power with specific dimensions. Naturally, we want to understand the space in which these inputs live.

Definition 3.2. The vector space $\operatorname{Sym}(n,d)$ is $\operatorname{span}\{||v\rangle^{\otimes n}\rangle : |v\rangle \in \mathbb{C}^d\}$.

The notation stands for "symmetric" since vectors of the form $|v\rangle^{\otimes n}$ are, in a way, symmetric as a tensor product. As an abbreviation, we will simply refer to this space as Sym with the n and d parameters being implied.

The binomial coefficient comes from the dimension of this space.

Lemma 3.3. *The dimension of* $\operatorname{Sym}(n,d)$ *is* $\binom{n+d-1}{d-1}$.

We will not prove this lemma. (It might show up as a homework problem.)

We promised that this would make our POVM satisfy the integral constraint, so of course we must now check that

$$\int_{|v\rangle} E_{|v\rangle} = \binom{n+d-1}{d-1} \int_{|v\rangle} |v\rangle^{\otimes n} \langle v|^{\otimes n} \, dv \overset{?}{=} I.$$

holds. (Here, the integral is only over unit vectors, and this will be true of all the integrals we consider in the remainder of this lecture.) In order to show this, we will use the following proposition.

Proposition 3.4. *Let*

$$M = \int_{|v\rangle} |v\rangle^{\otimes n} \langle v|^{\otimes n} \, dv = \mathop{\mathbf{E}}_{\text{unit } |v\rangle} \left[|v\rangle^{\otimes n} \langle v|^{\otimes n} \right]$$

and let

$$\Pi = \binom{n+d-1}{d-1} M.$$

Then Π *projects onto* Sym*.*

Proof. Let P_{Sym} be the projector onto Sym. We will start by showing that $M = CP_{\text{Sym}}$ for some constant C.

To show this, we only need to show that M zeroes out any vector perpendicular to Sym and that M scales vectors in Sym by a constant factor. This is sufficient since it is easy to see that M is linear.

The first part is easy. Because $|v\rangle^{\otimes n} \in \text{Sym}$, we have that for any $|u\rangle \in \text{Sym}^{\perp}$,

$$M |u\rangle = \mathop{\mathbf{E}}_{\text{unit} |v\rangle} \left[|v\rangle^{\otimes n} \underbrace{\langle v|^{\otimes n} |u\rangle}_{0} \right] = 0$$

as desired.

For the second part, take any $|u\rangle \in \text{Sym}$ and write it as $|u\rangle = |w\rangle^{\otimes n}$ for some $|w\rangle$ in $\mathbb{C}^d$. We will illustrate the case where $n = 1$. (The proof for the general case where $n > 1$ is about the same, except more annoying. For that reason, we will skip it.) Since M is linear, we can assume that $|w\rangle$ is a unit vector without loss of generality. We have

$$M |w\rangle = \mathop{\mathbf{E}}_{\text{unit } |v\rangle} \left[|v\rangle \langle v| \, |w\rangle \right]$$

For arbitrary $|v\rangle$, decompose $|v\rangle$ into

$$|v\rangle = \alpha |w\rangle + |w^{\perp}\rangle$$

where $|w^{\perp}\rangle$ is some vector that is perpendicular to $|w\rangle$. Because $|v\rangle$ is random, the coefficient α is random as well. In addition, because $|v\rangle$ is uniformly random, $|v\rangle$ appears with the same probability as $\alpha |w\rangle - |w^\perp\rangle$. We use this to break up the expectation before combining it again with linearity of expectation, getting

$$\begin{aligned}
M|w\rangle &= \mathop{\mathbf{E}}_{\text{unit } |v\rangle} [|v\rangle \langle v| \, |w\rangle] = \mathop{\mathbf{E}}_{\alpha} \left[(\alpha |w\rangle + |w^\perp\rangle)(\alpha^\dagger \langle w| + \langle w^\perp|) \, |w\rangle\right] \\
&= \frac{1}{2} \mathop{\mathbf{E}}_{\alpha} \left[(\alpha |w\rangle + |w^\perp\rangle)(\alpha^\dagger \langle w| + \underbrace{\langle w^\perp|) \, |w\rangle}_{0} \right] + \frac{1}{2} \mathop{\mathbf{E}}_{\alpha} \left[(\alpha |w\rangle - |w^\perp\rangle)(\alpha^\dagger \langle w| - \underbrace{\langle w^\perp|) \, |w\rangle}_{0} \right] \\
&= \frac{1}{2} \mathop{\mathbf{E}}_{\alpha} \left[(\alpha |w\rangle + |w^\perp\rangle)\alpha^\dagger \langle w|w\rangle\right] + \frac{1}{2} \mathop{\mathbf{E}}_{\alpha} \left[(\alpha |w\rangle - |w^\perp\rangle)\alpha^\dagger \langle w|w\rangle\right] \\
&= \frac{1}{2} \mathop{\mathbf{E}}_{\alpha} \left[(\alpha |w\rangle \cancel{+ |w^\perp\rangle})\alpha^\dagger \langle w|w\rangle + (\alpha |w\rangle \cancel{- |w^\perp\rangle})\alpha^\dagger \langle w|w\rangle\right] \\
&= \mathop{\mathbf{E}}_{\alpha} \left[\alpha\alpha^\dagger |w\rangle \underbrace{\langle w|w\rangle}_{1} \right] = \mathop{\mathbf{E}}_{\alpha} \left[|\alpha|^2\right] |w\rangle
\end{aligned}$$

It is not hard to see that $\mathbf{E}_\alpha \left[|\alpha|^2\right]$ is independent of $|w\rangle$, i.e. is a constant, thanks to the symmetry induced by the uniform randomness of $|v\rangle$. Thus $M|w\rangle = C|w\rangle$ where we let C be $\mathbf{E}_\alpha \left[|\alpha|^2\right]$.

All this implies that $M = CP_{\text{Sym}}$ for some constant C. We can find the actual value of the constant by taking the trace of both sides of the equality and performing some manipulations:

$$\begin{aligned}
C\binom{n+d-1}{d-1} &= C \dim \text{Sym} = C \operatorname{tr}(P_{\text{Sym}}) = \operatorname{tr}(CP_{\text{Sym}}) \\
&= \operatorname{tr}(M) = \mathop{\mathbf{E}}_{\text{unit } |v\rangle} \operatorname{tr}\left(|v\rangle^{\otimes n} \langle v|^{\otimes n}\right) = \mathop{\mathbf{E}}_{\text{unit } |v\rangle} \langle v|v\rangle^n = 1
\end{aligned}$$

where the first equality uses lemma 3.3, the penultimate equality uses the cyclic property of the trace operator, and the last equality uses the observation that $|v\rangle$ is a unit vector. Dividing through by $\binom{n+d-1}{d-1}$ gives

$$C = \frac{1}{\binom{n+d-1}{d-1}},$$

and therefore

$$\Pi = \frac{1}{C} M = P_{\text{Sym}}.$$

This proves that Π indeed projects onto Sym. □

This proposition tells us that $\int_{|v\rangle} E_{|v\rangle} = \binom{n+d-1}{d-1} \int_{|v\rangle} |v\rangle^{\otimes n} \langle v|^{\otimes n} dv = \Pi$ projects onto Sym, and consequently it is the identity within Sym. Since Sym is the only subspace our POVM operates upon, this shows $\int_{|v\rangle} E_{|v\rangle} = I$ as desired.

3.5 Algorithm analysis

We have proven that the POVM in our algorithm is valid, and now we will show that our algorithm is accurate.

The previous proposition gives the following corollary.

Corollary 3.5. *For any unit $|\psi\rangle \in \mathbb{C}^d$, the equality*

$$\mathop{\mathbf{E}}_{\text{unit } |v\rangle} \left[|\langle v|\psi\rangle|^{2n} \right] = \frac{1}{\binom{n+d-1}{d-1}}.$$

holds.

Proof. We know that Π projects onto Sym and that $|\psi\rangle^{\otimes n} \in \text{Sym}$. Thus $\Pi |\psi\rangle^{\otimes n} = |\psi\rangle^{\otimes n}$, which gives

$$\operatorname{tr}\left(\Pi |\psi\rangle^{\otimes n} \langle\psi|^{\otimes n}\right) = \operatorname{tr}\left(|\psi\rangle^{\otimes n} \langle\psi|^{\otimes n}\right) = \operatorname{tr}(\langle\psi|\psi\rangle^n) = 1.$$

We also have

$$\operatorname{tr}\left(\Pi |\psi\rangle^{\otimes n} \langle\psi|^{\otimes n}\right) = \operatorname{tr}\left(\langle\psi|^{\otimes n} \Pi |\psi\rangle^{\otimes n}\right) = \langle\psi|^{\otimes n} \Pi |\psi\rangle^{\otimes n}$$
$$= \binom{n+d-1}{d-1} \mathop{\mathbf{E}}_{\text{unit } |v\rangle} \left[\langle\psi|^{\otimes n} |v\rangle^{\otimes n} \langle v|^{\otimes n} |\psi\rangle^{\otimes n} \right] = \binom{n+d-1}{d-1} \mathop{\mathbf{E}}_{\text{unit } |v\rangle} \left[|\langle v|\psi\rangle|^{2n} \right].$$

Putting these two equalities together, we get

$$\binom{n+d-1}{d-1} \mathop{\mathbf{E}}_{\text{unit } |v\rangle} \left[|\langle v|\psi\rangle|^{2n} \right] = 1.$$

Dividing both sides by $\binom{n+d-1}{d-1}$ gives the result. □

The next theorem gives an upper bound on the error of our algorithm.

Theorem 3.6. *Given an input $|\psi\rangle^{\otimes n}$, the pure state tomography algorithm outputs $|\widetilde{\psi}\rangle$ with the property that*

$$\mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle} \left[d_{\text{tr}} \left(|\psi\rangle, |\widetilde{\psi}\rangle \right) \right] \leq \sqrt{\frac{d-1}{n+d}}.$$

Proof. We have

$$\mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle} \left[d_{\text{tr}} \left(|\psi\rangle, |\widetilde{\psi}\rangle \right) \right] = \mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle} \left[\sqrt{1 - \left| \langle\psi|\widetilde{\psi}\rangle \right|^2} \right] \leq \sqrt{1 - \mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle} \left[\left| \langle\psi|\widetilde{\psi}\rangle \right|^2 \right]}.$$

The first equality uses fact 3.1 from earlier in the lecture. The second equality first uses Jensen's inequality on the concavity of the square root operator and then uses linearity of expectation.

Looking only at the inner term, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle} \left[\left| \langle\psi|\widetilde{\psi}\rangle \right|^2 \right] &= \int_{|\widetilde{\psi}\rangle} \left| \langle\psi|\widetilde{\psi}\rangle \right|^2 \mathbf{Pr}\left[|\widetilde{\psi}\rangle \right] \\
&= \int_{|\widetilde{\psi}\rangle} \left| \langle\psi|\widetilde{\psi}\rangle \right|^2 \binom{n+d-1}{d-1} \left| \langle\psi|\widetilde{\psi}\rangle \right|^{2n} d\widetilde{\psi} \\
&= \binom{n+d-1}{d-1} \int_{|\widetilde{\psi}\rangle} \left| \langle\psi|\widetilde{\psi}\rangle \right|^{2(n+1)} d\widetilde{\psi} \\
&= \binom{n+d-1}{d-1} \frac{1}{\binom{n+d}{d-1}} = 1 - \frac{d-1}{n+d}
\end{aligned}$$

where the second line uses equation (1) from early in the lecture and where the last line uses corollary 3.5.

This finally gives

$$\mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle}\left[d_{\mathrm{tr}}\left(|\psi\rangle\,,|\widetilde{\psi}\rangle\right)\right] \le \sqrt{1-\mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle}\left[\left|\langle\psi|\widetilde{\psi}\rangle\right|^2\right]} = \sqrt{1-\left(1-\frac{d-1}{n+d}\right)}$$

which is the desired result. $\square$

In particular, if we choose $n \in \Theta\left(d/\varepsilon^2\right)$ as in fact 2.3, we get

$$\mathop{\mathbf{E}}_{|\widetilde{\psi}\rangle}\left[d_{\mathrm{tr}}\left(|\psi\rangle\,,|\widetilde{\psi}\rangle\right)\right] \le \sqrt{\frac{d-1}{n+d}} \le \sqrt{d/n} \in \Theta(\varepsilon).$$

This proves that the bound given in fact 2.3, $n \in \Theta\left(d/\varepsilon^2\right)$, is indeed sufficient.

References

[HHJ$^+$15] Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu. Sample-optimal tomography of quantum states. 2015.

[MHS$^+$12] Xiao-Song Ma, Thomas Herbst, Thomast Scheidl, Daqing Wang, Sebastian Kropatschek, William Naylor, Alexandra Mech, Bernhard Wittmann, Johannes Kofler, Elena Anisimova, Vadim Makarov, Thomas Jennewein, Rupert Ursin, and Anton Zeilinger. Quantum teleportation using active feed-forward between two canary islands. *Nature*, 489(7415):269–723, 2012.

[OW15] Ryan O'Donnell and John Wright. Efficient quantum tomography. 2015.

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

Lecture 23: Introduction to Quantum Complexity Theory

November 31, 2015

Lecturer: Ryan O'Donnell *Scribe: Will Griffin*

1 REVIEW: CLASSICAL COMPLEXITY THEORY

Usually, when doing complexity theory, we look at decision problems, where the output is 0 or 1. This is because it makes the problems simpler to analyze, and there is not much loss of generality since solutions to decision problems can be efficiently turned into solutions for function problems, where the output is more complex. Using this, we can define a general function to be analyzed:
$f : \{0, 1\}^* \rightarrow$, where the $*$ implies it's domain can be any set of binary strings, and 1 corresponds to yes, 0 to no.
This can be reformulated in terms of formal languages, where for some decision problem $f(x)$, the language L corresponding to f is the set of all binary strings x such that $f(x) = 1$, so: $L = \{x : f(x) = 1\}$.
Examples:

$CONN = \{x : x \text{ encodes a connected graph}\}$

$PRIMES = \{x : x \text{ encodes a prime number}\}$

$SAT = \{x : x \text{ encodes a boolean formula that is satisfiable}\}$

$FACTORING = \{(x, y) : x \text{ has a prime factor between 2 and } y\}$

Of these examples, $CONN, PRIMES$, and SAT are naturally decision problems. For $FACTORING$, however, it seems more natural to return a prime factor of x. However, this is not an issue, since prime factors of x can be found by changing y and using it to binary search the numbers between 2 and x, which takes linear time in the number of bits of x.

2 COMPLEXITY CLASSES

A Complexity Class is a set of Languages with related complexity. First we will define 4 of them, and show how they are related.

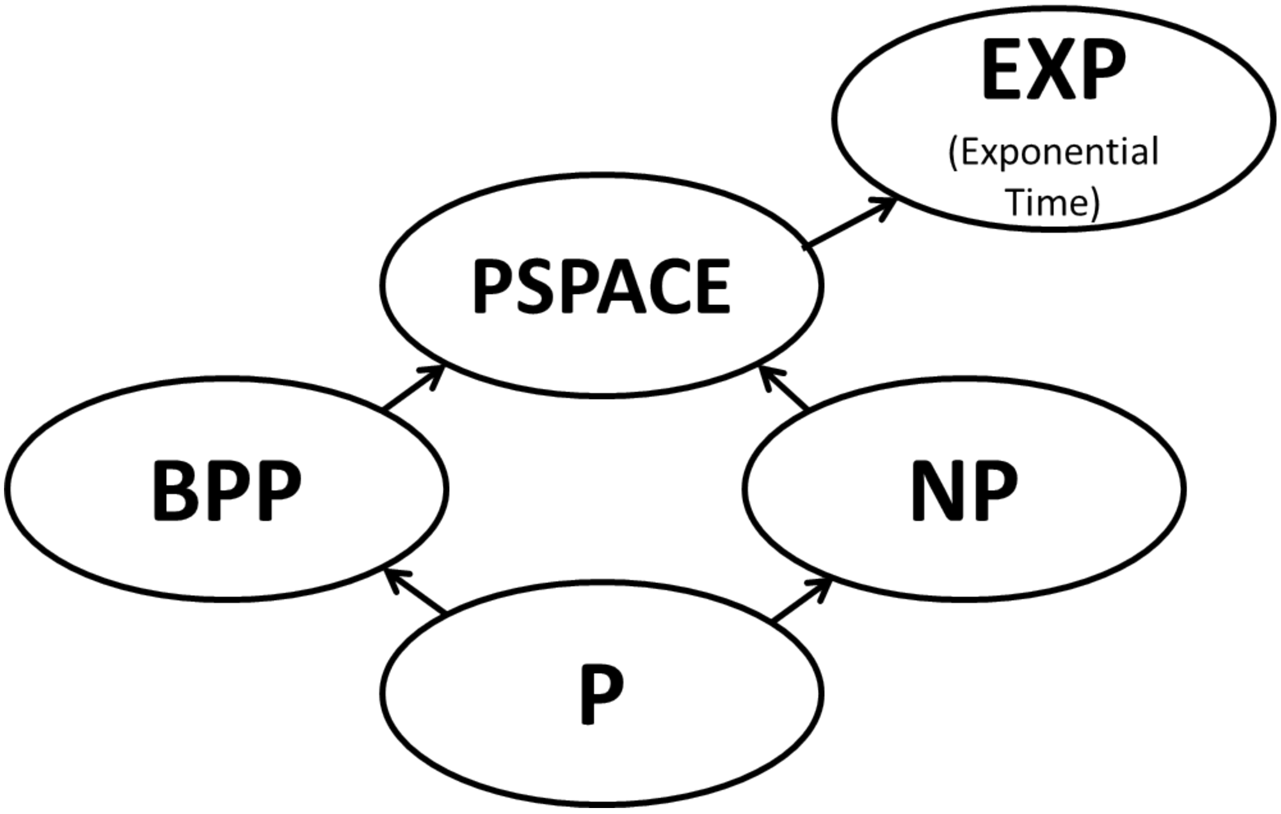
$P = \{$ Languages L : the problem $x \in L$ is decidable in $Poly(n)$ time by a deterministic algorithm $\}$.
In this definition, a deterministic algorithm is a standard Turing machine.
P is considered efficient. $CONN$ (shown in class) and $PRIMES$ are in P [AKS04]. The others are not known to be in P, though most complexity theorists think they are not.

$PSPACE = \{$ Languages L : the problem $x \in L$ is decidable in $Poly(n)$ space deterministically$\}$.
Space can be defined in terms of memory cells or tape spaces in a Turing machine.
All of the examples are in $PSPACE$, as they can all be computed with brute force methods using $Poly(n)$ space.
P is in $PSPACE$ since an algorithm can only use $Poly(n)$ space in $Poly(n)$ time.

BPP (Bounded Error, Probabilistic, Polynomial) $= \{L$: the problem $x \in L$ is decidable in $Poly(n)$ time by a randomized algorithm$\}$
Here, randomized algorithm means a standard Turing machine that has access to a 'coin flipper', which can output 0 or 1 each with probability 1/2.
Decided, for this class, is usually taken to mean that the probability of accepting x is greater than 3/4 is $x \in L$, and is less than 1/4 if x is not in L, so we say an algorithm in BPP decides f if $\forall x, Pr[Alg(x) = f(x) > 3/4$.
The 3/4 is fairly arbitrary, and it can be changed to any θ with the restriction that: $\theta \leq 1 - 2^{-n}$ and $\theta \geq 1/2 + 1/Poly(n)$ without changing the class at all. This can be shown using Chernoff bounds, (it was a problem on homework 1) and is called error reduction, and can be accomplished by running the algorithm multiple times and taking the majority value of the guesses. However, if $\theta < 1/2 + 1/Poly(n)$ it can take exponentially many runs to get to 3/4 probability, and if $\theta > 1 - 2^{-n}$, it can take exponentially many runs to reach that probability. Forcing those constraints on the output is what is meant by 'bounded error'.
BPP is, informally, that class of problems with efficient random algorithms.
$CONN$ and $PRIMES$ are in BPP, is not known whether the other examples are, though most people think they are not.
$P \subseteq BPP$ because a BPP algorithm can just not use its randomness. It is believed that $P = BPP$, but unproven. This is because of excellent pseudo-random algorithms that can effectively simulate truly random computation on a deterministic machine.
$BPP \subseteq PSPACE$. Since any randomized output takes $Poly(n)$ time and space to run, a $PSPACE$ algorithm can try all of them, and exactly compute the probability that the BPP algorithm would get the right answer.

NP (Non-deterministic Polynomial Time) $= \{L : L$ has a one-way, deterministic, $Poly(n)$ time proof system$\}$
The full definition of NP relies on a $Prover$ and a $Verifier$. The $Prover$ is trying to convince the $Verifier$ that some given binary string x of length $Poly(n)$ and language L, that $x \in L$.

The $Verifier$ wants to accept the proof if and only if $x \in L$.
Formally, for some x, L, the $Prover$ sends some string $\pi \in \{0,1\}^{Poly(n)}$ to the $Verifier$. The $Verifier$ computes on (x, π) in deterministic polynomial time and answers $\{0,1\}$ (1 for accept), with the rules:
$\forall x \in L$, $\exists \pi$ such that $Verifier(x, \pi) = 1$
$\forall x \notin L$, $\forall \pi$ $Verifier(x, \pi) = 0$
When $x \in L$, the $Verifier$ and $Prover$ are essentially working together, trying to find some string that shows $x \in L$.
When $x \notin L$, the $Prover$ is essentially working as an adversary, trying to find some string that the $Prover$ will accept even though the $Prover$ shouldn't.
$P \subseteq NP$, since the $Verifier$ can just ignore the proof and compute the question in $Poly(n)$ time.
It is not known if $BPP \subseteq NP$ (but it clearly is if $BPP = P$).
$NP \subseteq PSPACE$. Since each possible proof π can be checked in $Poly(n)$ time and space, a $PSPACE$ algorithm can check them all and accept if any of them accept.
All of the examples are in NP.
$CoNP$ may not be in NP. For example, $UNSAT = \{x : x$ encodes a boolean circuit with no satisfying assignment$\}$ may not be in NP.
The relationship between the different classes shown is in the diagram, which also shows that they are all contained in EXP, the class of problems that take exponential time to solve.



3 QUANTUM COMPLEXITY

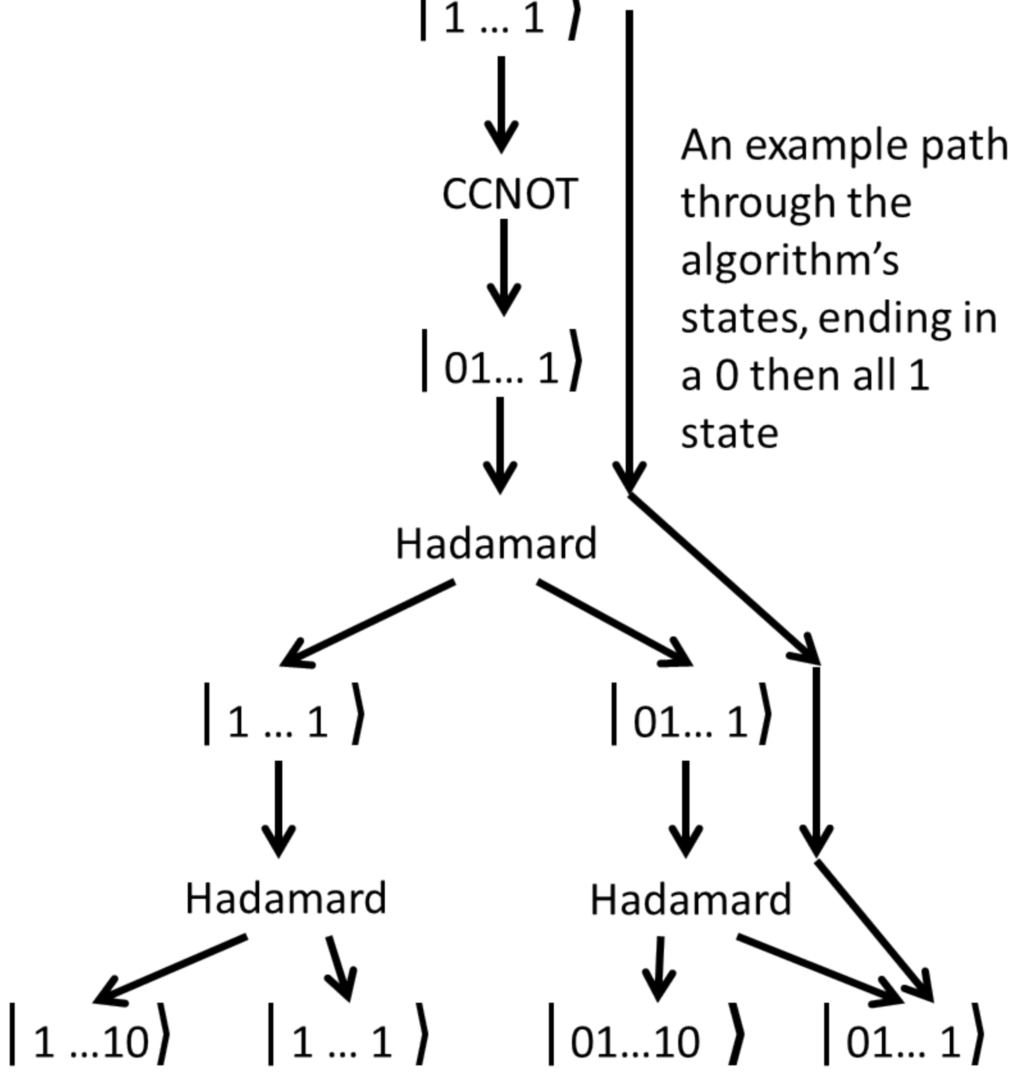
Since we've gone over Classical Complexity, where does Quantum Complexity fit in?
First, our model for quantum algorithms is quantum circuits, not Turing machines. Quantum Turing machines can be defined, but they are confusing and difficult to work with. Each circuit, however, can only take in inputs of one size, so it is not obvious how to define complexity for quantum circuits in a way that allows us to compare them to Turing machines that can take arbitray input sizes.
To overcome this issue, we use a model where a BPP algorithm writes down a quantum circuit with hard coded inputs, and the measurement at the end gives the result.
Then define:
BQP (bounded-error, quantum, polynomial time) $= \{L : \exists$ a BPP algorithm that can write down a quantum circuit with hard-coded inputs that can decide $x \in L\}$ [BV97].
In this definition, decide means that when the q-bits are measured at the end of the quantum circuit, the chance of seeing the wrong answer is less than $1/4$ for any x. This is essentially the same as deciding $x \in L$ for BPP, except that a measurement needs to be done.
Defining circuit complexity as the complexity of a Turing machine that can write down the circuit is robust, and used in classical complexity as well. For example, $P = \{L : \exists$ a P algorithm that can write down a classical circuit that can decide $x \in L\}$ and
$BPP = \{L : \exists$ a P algorithm that can write down a classical randomized circuit that can decide $x \in L\}$.
In addition, the definition is robust to changes in the set of quantum gates used for computation, so using a different set of quantum gates does not change the definition of the class [DN05].
$CONN, PRIMES$, and $FACTORING$ are in BQP. BQP contains P and BPP and $BQP \subseteq PSPACE$. The relationship between BQP and NP is unknown, though it is widely believed that there are problems in NP not in BQP and problems in BQP not in NP.
The evidence for those beliefs is based on oracles - there are oracles relative to which BQP has problems not in NP and vice versa.
Additionally, it is believed that BQP is not a part of the Polynomial Time Hierarchy. [Aar10]

Theorem 3.1. *$BQP \subseteq PSPACE$*

Proof. Suppose a language $L \in BQP$, so there is a $Poly(n)$ time algorithm that, on input x, writes down a Quantum circuit C_x with inputs hardcoded to $|1\rangle$, and when the 1^{st} bit of the output of C_x is measured, it gives $f(x)$ with probability greater than or equal to $1/4$.
C_x starts in the state $y_1, y_2, \ldots y_m = |111\ldots11\rangle$, $m = Poly(n)$.
Then C_x applies a series of $CCNOT$ and $Hadamard$ (H) gates to the bits, and then the 1^{st}

bit is measured.
As drawn below, the state can be explicitly tracked, and a classical machine can track the full state through the operation of each gate. $CCNOT$ gates are easy to track because they are deterministic, but H gates force the classical computer to split each partial state it is tracking into two states, doubling what it must keep track of.
Then, if there are h Hadamard gate, at the end of the computation there will be 2^h branches, each corresponding to a different path through the circuit (even though some of the paths may give the same measurement outcome). Then let $|p\rangle$ be a final state. p can be seen as a path through the 'state tree' drawn, and at each branching off point, there is a certain probability amplitude assigned to p at that point (in this case it's always $1/\sqrt{2}$ since we're using H gates).

Example 'State Tree' for an extremely simple circuit:



Then, at the end of the computation, the final state can be written as a sum over all the paths times their amplitudes, so $|final\rangle = \sum_p amp(p)\,|p\rangle$.
$amp(p) = \left(\frac{1}{\sqrt{2}}\right)^h sign(p)$, where $sign(p)$ is the product of the signs of the amplitude of p. Then $|final\rangle = \left(\frac{1}{\sqrt{2}}\right)^h \sum_p sign(p)\,|p\rangle$. Then the final amplitude of a state $|s\rangle = \left(\frac{1}{\sqrt{2}}\right)^h \sum_{p:|p\rangle=|s\rangle} sign(p)$. Then the probability of measuring the state $|s\rangle$ is the squared magnitude of the amplitude, so $Pr(s) = 2^{-h}\sum_{p,p'} sign(p)sign(p')$, where the sum over p, p' is over $|p\rangle = |s\rangle, |p'\rangle = |s\rangle$.
What we want is the probability that we measure a $|0\rangle$ or $|1\rangle$ for the first bit. But since we know the probability of measuring a state $|s\rangle$, we can just sum over all states $|s\rangle$ with a $|1\rangle$ in the first bit. Then $Pr(1) = \sum_{p,p'} sign(p)sign(p')$, where the sum is over paths p, p' such that $|p\rangle = |p'\rangle$ and the first bit is $|1\rangle$.
Additionally, we can use the formula for $Pr(C_x accept)$ to then get: $Pr(C_x accept) - Pr(C_x reject) = 2^{-h}\sum_{p,p'} sign(p)sign(p')(-1)^{1-state(p)}$, where $state(p)$ is the first bit of $|p\rangle$, and the sum is over $|p\rangle = |p'\rangle$. If $x \in L$, this is greater than 1/2, and if $x \notin L$, this is less than $-1/2$.
This shows how to calculate the output probability classically, but since there are exponentially many paths, it would take an exponential amount of space to store the path information. To make an algorithm to do this in $PSPACE$, the key is to calculate the contribution to the probability from each path, which can be done in $Poly(n)$ time and space since there are $Poly(n)$ gates, and add them up, which allows us to find $Pr(1)$ with a $PSPACE$ algorithm taking exponential time.

□

4 BQP and PP

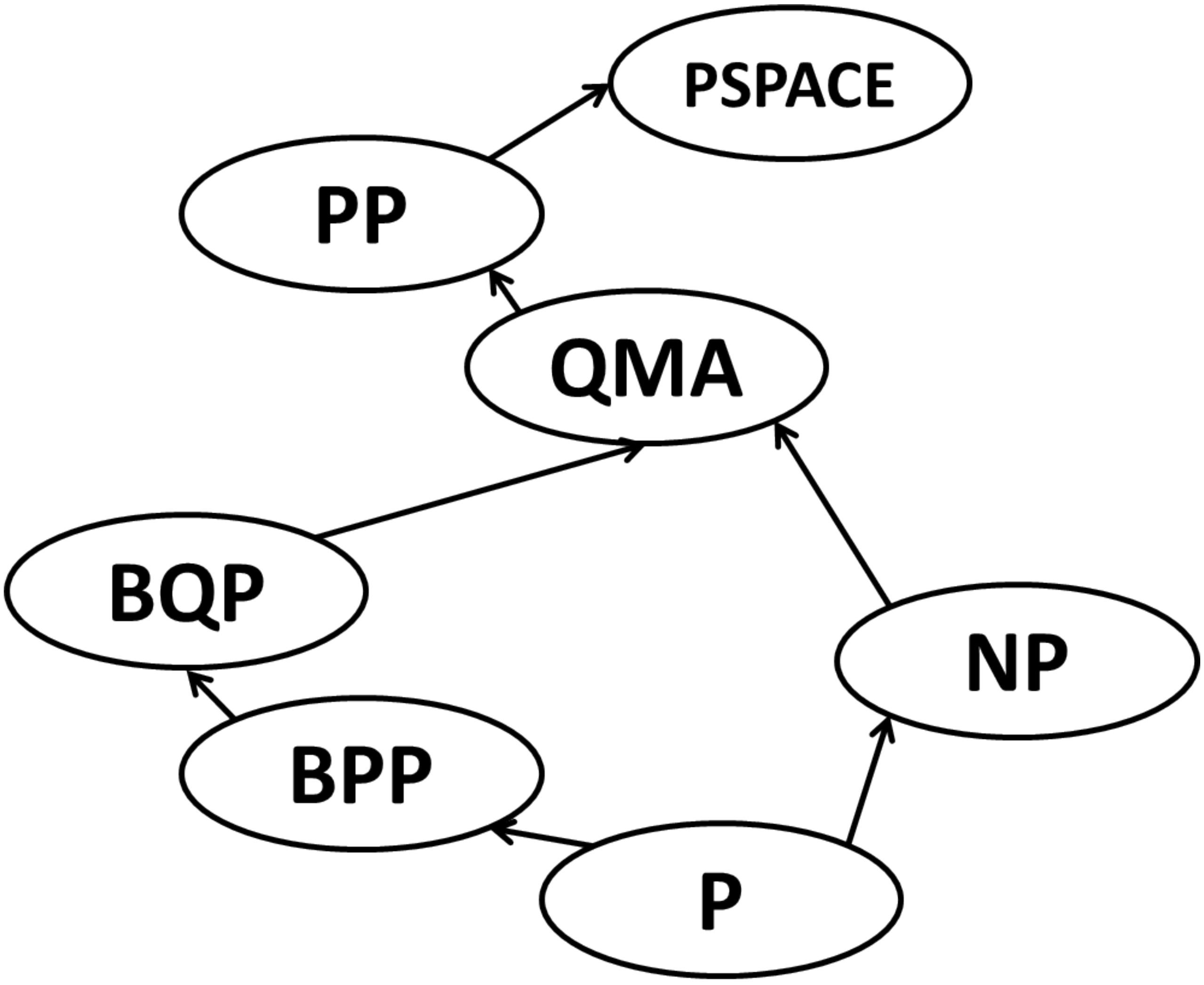
PP (Probabilistic Polynomial) $= \{L : x \in L$ is decidable by a probabilistic polynomial time algorithm$\}$.
Here, decidable means that the probability of the algorithm being wrong is less than 1/2. This is different than BPP and BQP, which force the error to be less than 1/2 by at least a $1/Poly(n)$ amount.
Because of that difference, PP lacks efficient error amplification, so algorithms in PP are not necessarily efficient, since it can take exponentially many runs to get a high success probability. This makes PP less useful than BPP as a complexity class.
$PP \subseteq PSPACE$, this can be seen in the same way as $BPP \subseteq PSPACE$, since a $PSPACE$ algorithm can just iterate through every possible random outcome and find the probability.
PP contains both NP and BQP. It also contains the quantum version of NP, QMA, which is analogous to NP except the $Prover$ sends a quantum states and the $Verifier$ decides to accept with a quantum algorithm.

Theorem 4.1. *$BQP \subseteq PP$ [ADH97]*

Proof. Given some quantum algorithm Q, the PP algorithm Alg first randomly computes the end state of two paths in Q, p and p' independently (by computing the quantum state of the system and randomly picking one choice whenever a Hadamard gate is used).
Then let $|p\rangle, |p'\rangle$ be the end states of the paths, and q be the probability that $|p\rangle \neq |p'\rangle$. If $|p\rangle \neq |p'\rangle$, Alg outputs 1 with probability $P > 1/2$, -1 with probability $P < 1/2$.
If $|p\rangle = |p'\rangle$, it outputs $sign(p)sign(p')(-1)^{1-state(p)}$, where $state(p)$ is the value of the first bit of the final state of path p. Then the expectation value of Alg is:
$E(Alg) = (1-q)2^{-h}\sum_{p,p'} sign(p)sign(p')(-1)^{1-state(p)} + q(2P-1)$,
and $2^{-h}\sum_{p,p'} sign(p)sign(p')(-1)^{1-state(p)}$ is $Pr(accept) - Pr(reject)$ as shown before.
If $x \in L$, then $E(Alg) > 0$.
If $x \notin L$, then $E(Alg) < 0$.
Then this shows that Q can be simulated by a PP algorithm by accepting with probability $P = 1/2 + Alg(x)$.

□

With all the complexity classes we have now defined, the picture of the complexity hierarchy is now:



5 A few more details about PP

Just wanted to add a few things about the complexity class PP, a rather strange class.

Super-minor, boring technical detail.

First, a super-minor and boring technical detail. For PP, we say that the machine "accepts" a string x if it outputs 1 with probability $> 1/2$. We say that it "rejects" a string x if it outputs 1 with probability $< 1/2$. What if, for some x, it outputs 1 with probability exactly 1/2? Does that count as "accept" or "reject"? The answer is: Doesn't really matter; any reasonable convention you decide on gives the same class PP.
To see this, let's say we take the convention that outputting 1 with probability 1/2 counts as "reject". Now suppose we have a randomized algorithm A running in time n^c that "decides"

language L in the sense that $x \in L \Rightarrow \mathbf{Pr}[A(x) = 1] > 1/2$ and $x \notin L \Rightarrow \mathbf{Pr}[A(x) = 1] \leq 1/2$. The first thing to note is that since A runs in time n^c, it flips at most n^c coins, and thus every possible outcome it has occurs with probability that is an integer multiple of 2^{-n^c}. In particular, when $x \in L$ we know that not only is $\mathbf{Pr}[A(x) = 1] > 1/2$, in fact it is $\geq 1/2 + 2^{-n^c}$. Now consider the following algorithm B. On any input x, it first flips n^{c+1} coins. If they all come up 1, then B immediately outputs 0. Otherwise, B simulates A. Note that B is still polynomial time. Furthermore, the effect of that first stage of flipping n^{c+1} coins is to slightly down-shift all of A's probabilities by roughly $2^{-n^{c+1}}$. So now we will indeed have that $x \in L \Rightarrow \mathbf{Pr}[B(x) = 1] > 1/2$ and $x \notin L \Rightarrow \mathbf{Pr}[B(x) = 0] < 1/2$. So you see, with this kind of trick, the exact tie-breaking convention in PP does not matter.

NP is in PP.

Now let me explain why NP is in PP. It's very simple. By the theory of NP-completeness, it is sufficient to show that SAT is in PP. Here is the algorithm. On input F, an m-variable Boolean formula (where $m < |F| = n$)), the PP algorithm guesses a uniformly random assignment $\alpha \in \{0, 1\}^m$ and checks if it satisfies F. If so, the algorithm outputs 1. If not, the algorithm outputs a random coin flip. Now it's clear that if $F \in$ SAT then $\mathbf{Pr}[\text{output } 1] \geq 1/2 + 2^{-m}$ and if $F \notin$ SAT then $\mathbf{Pr}[\text{output } 1] \leq 1/2$. (In fact, it's $= 1/2$ in the second case.) This shows SAT is in PP, by the above super-minor boring discussion.

PP is closed under complement.

As you can see from the original $> 1/2$ vs. $< 1/2$ definition of PP, it is symmetrically defined with respect to yes-instances and no-instances; i.e., it is "closed under complementation"; i.e., if L is in PP then so is L^c. Hence UNSAT is in PP, which means that PP contains not just NP but also coNP (if you remember that class).

A "complete" problem for PP.

An easy-to-show fact in complexity theory: Let MAJ-SAT be the language of all Boolean formulas F for which more than half of all assignments satisfy F. You should be able to easily show that MAJ-SAT is in PP. In fact, it is also very easy to show that MAJ-SAT is "PP-complete". In particular, MAJ-SAT is in P if and only if P = PP.

The weird nature of PP.

On one hand, PP contains both NP and coNP. On the other hand, it is not known to contain the class P^{NP} (if you remember what that is: namely, the languages decidable by polynomial time algorithms with access to a SAT oracle). On the other other hand, it is known that the entire "polynomial hierarchy" (a vast generalization of NP, which is "almost" as large as PSPACE) is contained in P^{PP} (i.e., polynomial-time algorithms with access to an oracle for MAJ-SAT). This is "Toda's Theorem". So yeah, PP is kind of a weird and unnatural class; the basic version is not very powerful; it's not closed under "subroutines"; but when you "close" it under subroutines it becomes super-powerful.

References

[Aar10] S. Aaronson. BQP and the Polynomial Hierarchy. *Proceedings of ACM STOC 2010*, pages 141–150, 2010.

[ADH97] L. Adleman, J. DeMarrais, and M. Huang. Quantum Computability. *SIAM Journal on Computing*, pages 1524–1540, October 1997.

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics 160*, pages 781–793, 2004.

[BV97] E. Bernstein and U. Vazirani. Quantum Complexity Theory. *SIAM Journal on Computing, Volume 26*, pages 1411–1473, October 1997.

[DN05] C. Dawson and M. Nielsen. The Solovay-Kitaev Algorithm. *arXiv:quant-ph/0505030v2*, August 2005.

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

Lecture 24: QMA: Quantum Merlin-Arthur

December 2, 2015

Lecturer: Ryan O'Donnell *Scribe: Sidhanth Mohanty*

1 Introduction

In this lecture, we shall probe into the complexity class QMA, which is the quantum analog of NP.

Recall that NP is the set of all decision problems with a one-way deterministic proof system.

More formally, NP is the set of all decision problems for which we can perform the following setup.

Given a string x and a decision problem L, there is an omniscient prover who sends a string $\pi \in \{0,1\}^n$ to a *polytime deterministic* algorithm called a verifier V, and $V(x, \pi)$ runs in polynomial time and returns 1 if x is a YES-instance and 0 if x is a NO-instance. The verifier must satisfy the properties of soundness and completeness: in particular, if x is a YES-instance of L, then there exists π for which $V(x, \pi) = 1$ and if x is a NO-instance of L, then for any π that the prover sends, $V(x, \pi) = 0$.

From now on, think of the prover as an unscrupulous adversary and for any protocol the prover should not be able to cheat the verifier into producing the wrong answer.

2 Merlin-Arthur: An upgrade to NP

If we take the requirements for a problem to be in NP and change the conditions slightly by allowing the verifier V to be a BPP algorithm, we get a new complexity class, MA. A decision problem L is in MA iff there exists a probabilistic polytime verifier V (called Arthur) such that when x is a YES-instance of L, there is π (which we get from the omniscient prover Merlin) for which $\mathbf{Pr}(V(x, \pi) = 1) \geq \frac{3}{4}$ and when x is a NO-instance of L, for every π, $\mathbf{Pr}(V(x, \pi) = 1) \leq \frac{1}{4}$. The choice of $\frac{3}{4}$ was arbitrary. As long as you pick a constant greater than $\frac{1}{2}$, error reduction is easy.

It turns out that if you mandate that the probability of being correct when x is a YES-instance to 1, MA still stays the same, by a very nontrivial theorem whose proof you can see in [FGM89]. However, if you add the condition that Arthur can never be duped, that is, remove room for error when x is a NO-instance, then the class simply becomes NP.

NP $\subseteq$ MA because for any NP problem, a probabilistic verifier V' could simply run the deterministic algorithm of the NP verifier V, ignore the random bits, and accurately verify proofs.

It is believed that NP = MA, though no proof is known. There is evidence to believe that NP = MA as research suggests that 'good enough' pseudorandom generators exist. For a detailed overview, refer to [BFA03] and [Vad12].

As an interlude, the name Merlin-Arthur for this complexity class is due to Laszlo Babai, introduced in [BM88].

3 Quantum Merlin-Arthur

The next upgrade that results in a new class comes from letting the verifier V be a quantum algorithm and allowing the prover to send qubits instead of bits. In other words, the verifier is in BQP. The class of problems that can be solved under this protocol is called QMA.

Definition: QMA is the set of all decision problems that can be solved with a quantum algorithm V such that for all possible inputs x, if x is a YES-instance, then there is $|\varphi\rangle$ for which $V(x, |\varphi\rangle) = 1$ with probability $\geq \frac{3}{4}$ and if x is a NO-instance, for every $|\varphi\rangle$, the probability that $V(x, |\varphi\rangle) = 1$ is $\leq \frac{1}{4}$.

3.1 Robustness of QMA

The next thing we probe into is how robust QMA is. If we force the acceptance probability to be 1, does QMA remain the same? It is unknown if this same oddity that is a property of MA holds in QMA.

However, we don't want our definition of QMA to hinge on numbers like $\frac{3}{4}$, so we would like an error amplification protocol that amplifies the success probability to any constant less than 1 of our choice. The amplification can be more formally expressed as: say we have a quantum algorithm V and input x for which there exists a state $|\varphi\rangle$ where $\mathbf{Pr}[V(x, |\varphi\rangle) = 1] \geq c$ when x is a YES-instance and $\mathbf{Pr}[V(x, |\varphi\rangle) = 1] \leq s$ when x is a NO-instance with the promise that $c - s \geq \frac{1}{\text{poly}(n)}$, can we boost the success probability to $1 - 2^{-\text{poly}(n)}$?

It would be easy to boost our probability if we could replicate the proof state $|\varphi\rangle$ that Merlin sent several times, run verifier V on $(x, |\varphi\rangle)$ several times for some input x and take the majority, but unfortunately, you cannot clone an unknown quantum state. And using your proof state by making a measurement renders it unusable for a second time.

The solution to this is make Merlin send the state a bunch of times. In the first proof system, the prover was supposed to send $|\varphi\rangle$ but in this new protocol, we require him to send $|\varphi\rangle^{\otimes T}$ where $T \in O(\text{poly}(n))$. And then Arthur runs his algorithm on each one and returns the majority.

But we are not done yet, as there is another subtlety left to address. When the input x is a NO-instance, then Arthur must reject with high probability. What if Merlin, in his efforts to dupe Arthur, sends quantum states that are entangled with eachother? Then the states can be seen as a mixed states (result from lecture 16). Arthur's probability of accepting a mixed state p when it would be correct to reject is the a convex linear combination of the probabilities that Arthur accepts each pure state that comprises the mixed state. There is some pure state for which Arthur has a probability $p_i > p$ of accepting.

And we know that for each pure state $|\pi\rangle$, $\mathbf{Pr}[V(x, |\pi\rangle) = 1] \leq s$. Therefore, $p_i \leq s$ and $p \leq s$ and our boosting works even if Merlin sends entangled states.

3.2 MA $\subseteq$ QMA, NP $\subseteq$ QMA and QMA $\subseteq$ PP

Since NP $\subseteq$ MA, it shall follow from MA $\subseteq$ QMA that NP $\subseteq$ QMA.

To see why MA is contained in QMA, we first recall that any randomized algorithm can be simulated by a quantum computer efficiently. So for any decision problem $L \in$ MA, we first force the proof sent by Merlin to be classical by measuring all the bits, and then we simulate the verifier's algorithm V with a quantum algorithm V' on input x and this classical proof. This means that L is also in QMA and as a result MA $\subseteq$ QMA.

It is also known that QMA $\subseteq$ PP. (Proof is a homework problem)

There is another complexity class between QMA and MA known as QCMA. This is the class of problems that are solvable if the prover can only send classical bits but the verifier has a Quantum algorithm. Not much is known about QCMA.

4 Some problems in QMA

Here are 3 problems of which we will prove membership in QMA.

1. k-Local Hamiltonians Problem [KSV02]
2. Consistency of local density matrices [Liu06]
3. Group Nonmembership [Wat00]

4.1 k-Local Hamiltonians Problem

The k-Local Hamiltonians Problem is the quantum analogue of Max k-SAT, which is the problem of maximizing the nunber of clauses satisfied in a k-CNF formula.

The problem is physically motivated: roughly, it asks 'You have a Hamiltonian, what is the energy of the ground state?'. More formally, the problem involves a n-qubit state $|\psi\rangle$

Let $|\psi\rangle$ be a n-qubit state. A k-Local Hamiltonian is a Hermitian Matrix H_α acting on n qubits and has the property that it is the identity on all except k of the qubits.

The input to the k-Local Hamiltonian problem is m k-Local Hamiltonians, $H_1, H_2, \ldots, H_m$ with the eigenvalues of each H_i being between 0 and 1.

Let $H = H_1 + H_2 + \ldots + H_m$. Define the ground state energy as $\xi = \min_{|\psi\rangle} \langle\psi| H |\psi\rangle$. The decision problem can be framed as $k-\mathrm{LH}_{\alpha,\beta}$ with $\alpha > \beta$, where we say "YES" if $\xi \leq \beta$, "NO" if $\xi \geq \alpha$. In other words, a YES-instance is exactly one where there exists some $|\psi\rangle$ for which $\langle\psi| H |\psi\rangle \leq \beta$ and a NO-instance is one where for every $|\psi\rangle$, $\langle\psi| H |\psi\rangle \geq \alpha$.

Theorem 4.1. $k - \mathrm{LH}_{\alpha,\beta} \in \mathrm{QMA}$ *with* $c = 1 - \frac{\beta}{m}$ *and* $s = 1 - \frac{\alpha}{m}$. *In fact, the problem is in QMA provided* $\alpha - \beta \geq \frac{1}{\mathrm{poly}(n)}$.

Proof. Merlin sends the state $|\psi\rangle$ that minimizes $\langle\psi| H |\psi\rangle$. Pick a uniformly random H_i from $H_1, \ldots, H_m$ and measure $|\psi\rangle$ with the POVM $\{H_i, I - H_i\}$. The probability that the measurement outcome corresponds to H_i for some i is $\sum \langle\psi| \frac{1}{m} H_i |\psi\rangle = \frac{1}{m} \langle\psi| H |\psi\rangle$.

By having Merlin send multiple copies, and measuring each one, we can discern the value of $\langle\psi| H |\psi\rangle$ within a negligible error margin, and check if it is $\leq \beta$ or $\geq \alpha$.

□

As an exercise, one may try showing that 3SAT is a subproblem of $3-\mathrm{LH}_{1,0}$.

Kitaev showed that the problem is QMA-complete for $k \geq 5$ in [KSV02]. This result was improved to $k \geq 3$ in [KR03], and further improved to $k \geq 2$ in [KKR06]. The proof is along the lines of the proof of Cook-Levin theorem, but much harder.

4.2 Consistency of local density matrices

The inputs to this problem are as follows: m k-qubit density matrices $\rho_1, \rho_2, \ldots, \rho_m$ and subsets $S_1, \ldots, S_n \subseteq [n]$ where ρ_i is the identity on all qubits except for the ones in S_i.

You are given a promise that either there is a global n-qubit density matrix σ such that $\mathrm{tr}_{\overline{S_i}}(\sigma) = \rho_i$ for every i or for all n-qubit density matrices σ, there is $i \in [m]$, $\|\mathrm{tr}_{\overline{S_i}} - \rho_i\|_1 \geq \frac{1}{\mathrm{poly}(n)}$.

QMA-completeness: One can reduce this problem from k-Local Hamiltonians problem as illustrated in [Liu06].

To show membership in QMA we sketch a proof.

Theorem 4.2. *Consistency is in QMA.*

Proof. The prover Merlin sends the correct density matrix. The verifier then picks a random subset S_i and checks if $\mathrm{Tr}_{\overline{S_i}}(\sigma)$ is close to ρ_i.

□

4.3 Group Nonmembership

Laszlo Babai conjectured that this problem is in MA. This problem has to do with a whole theory of black box groups due to Laszlo Babai and Endre Szemeredi, introduced and described in [BS84]. For a more modern survey, look at [BB99].

Say you are given a group of size $\leq 2^n$, its elements could either be encoded as permutations (as every group of size n is a subgroup of the symmetric group S_n), finite fields or in black box fashion (a black box specifies the inverse of an element or the product of two elements). And a group G is specified by its generators. A result by Babai and Szemeredi in [BS84] shows that a group can be specified by a set of generators of size at most $(1+\log|G|)^2$.

We shall follow the convention where each group element is encoded by a unique binary string (x is denoted as $\text{enc}(x)$). And we assume we are given an oracle that maps $(\text{enc}(x), \text{enc}(y))$ to $\text{enc}(xy)$ and $\text{enc}(x)$ to $\text{enc}(x^{-1})$. Henceforth, as a measure to not get too cumbersome with notation, $|x\rangle$ shall be used to denote an abstract group element.

4.3.1 An easier problem: Group Membership

Given as input generators of a group $|g_1\rangle, |g_2\rangle, \ldots, |g_n\rangle$ and a target $|x\rangle$, we want to check if $|x\rangle$ is in the group generated by $\{|g_i\rangle\}$.

Theorem 4.3. *Group Membership is in NP.*

At first glance, it may appear that the prover Merlin can simply send a concatenation of generator elements, because it seems like the verifier Arthur can multiply the concatenation and check if it is equal to $|x\rangle$. However, there is an issue with this intuition: it is not at all obvious why the length of the sequence of elements won't be exponential in size. But this has a fix.

Theorem 4.4 (Babai, Szemeredi, Cooperman). *There is a* $\text{poly}(n)$ *time randomized algorithm that multiplies* $T(\approx \text{poly}(n))$ *many* g_i*'s and* g_i^{-1}*'s together and its output is exponentially close to uniformly random on* H*. [Bab91] and [BCF+95]*

Proof. 5-10 pages of Fourier analysis and group theory. □

The theorem is a very strong statement. As a corollary, we know the following.

Corollary 4.5. *Every element of* H *has a chance of being outputted.*

This means that for every element $x \in H$ there is a nonzero chance that it is outputted, which means there is a polylength sequence of g_i's and g_i^{-1}'s whose product is x and the problem is indeed in NP.

4.3.2 Back to group nonmembership

Now, given $g_1, g_2, \ldots, g_n$ as generators for a subgroup H, we want to decide if $x \notin H$.

One natural way to solve this problem in QMA is to have Merlin send the state

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

But before we proceed further to see why sending that state helps us solve the problem in QMA, let us take a step back and see why the verifier cannot prepare the state himself.

While we can sample a uniformly random $h \in H$, running the randomized algorithm on a quantum computer to obtain the uniform superposition would yield something like

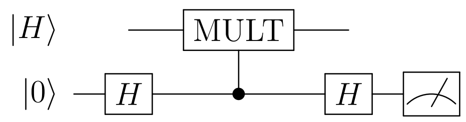
$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle \, |\text{garbage}(h)\rangle$$

And it is not clear how one gets rid of the garbage. Yet, it is the verifier's dream to have

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

Thus, a good prover sends the state $|H\rangle$ to the verifier.

Then the verifier attaches a $|0\rangle$ to the second register and runs it through the following circuit.



Applying the first Hadamard changes the state $|H\rangle\,|0\rangle$ to $|H\rangle \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)$.

And then, the second register undergoes a controlled-MULT with the element $|x\rangle$ and the new state is $\frac{1}{\sqrt{2}}\,|H\rangle\,|0\rangle + \frac{1}{\sqrt{2}}\,|Hx\rangle\,|1\rangle$. If $x \in H$, this state is exactly $|H\rangle \left(\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\right)$ since $Hx = H$. If $x \in H$, the final Hadamard sends this state back to $|H\rangle\,|0\rangle$. Measuring the second register always yields $|0\rangle$.

In the case when $x \notin H$, Hx is disjoint from H. And as a consequence, $|Hx\rangle \perp |H\rangle$. If we Hadamard for a second time and measure the second register, we see $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$ each.

To summarize, when $x \in H$, measurement always gives us $|0\rangle$. Whereas, when $x \notin H$, measurement gives us each possibility with an equal probability. There is a one-sided error of $\frac{1}{2}$ that can be boosted.

Thus, we have checked the completeness case. But it remains to verify that the protocol is sound. That is, the prover cannot cheat the verifier.

(The rest is all taken from Piazza)

What if the prover Merlin sends Arthur some state that is not $|H\rangle$? Let's say Merlin sends Arthur some state given by

$$|M\rangle = \sum_{y \in G} \alpha_y \, |y\rangle$$

Arthur could generate a random element $z_1 \in H$, attach a $|0\rangle$ to $|M\rangle$, apply a Hadamard on $|0\rangle$, perform a controlled MULT on $|M\rangle$ with $|z_1\rangle$, apply another Hadamard and then perform a measurement. If the measurement yields $|1\rangle$, it means that elements in the superposition $|M\rangle$ are not all in the same coset of $|H\rangle$, which means that the state is definitely not $|H\rangle$ and we immediately reject.

If we measure $|0\rangle$, the state becomes

$$\sum_{g \in G} \alpha_g |g\rangle + \alpha_g |gz_1\rangle$$

Now, if we do the same thing with a second uniformly random z_2, the resulting state we get is

$$\sum_{g \in G} \alpha_g |g\rangle + \alpha_g |gz_1\rangle + \alpha_g |gz_2\rangle + \alpha_g |gz_1 z_2\rangle$$

And we keep going with $z_1, z_2, \ldots, z_n$ polynomially many times.

Pretty soon, as long as $|\psi\rangle$ is passing the checks, it will become (exponentially close to) H-invariant, of the form $e^{i\theta} |Hg_0\rangle$.

This leads to a situation where the verifier can assume he has a desirable state and proceed with the protocol.

For a deeper summary of results on QMA, one can refer to [AK07].

References

[AK07] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. In *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*, pages 115–128. IEEE, 2007.

[Bab91] László Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 164–174. ACM, 1991.

[BB99] László Babai and Robert Beals. A polynomial-time theory of black box groups i. *London Mathematical Society Lecture Note Series*, pages 30–64, 1999.

[BCF$^+$95] László Babai, Gene Cooperman, Larry Finkelstein, Eugene Luks, and Ákos Seress. Fast monte carlo algorithms for permutation groups. *Journal of Computer and System Sciences*, 50(2):296–308, 1995.

[BFA03] Harry Buhrman, Lance Fortnow, and Pavan Aduri. Some results on derandomization. *Lecture notes in Computer Science*, 2607:212–222, 2003. `http://people.cs.uchicago.edu/~fortnow/papers/derand.pdf`.

[BM88] László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[BS84] Lászío Babai and Endre Szemerédi. On the complexity of matrix group problems i. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 229–240. IEEE, 1984.

[FGM89] Martin Furer, Oded Goldreich, and Yishay Mansour. On completeness and soundness in interactive proof systems. 1989.

[KKR06] Julia Kempe, Alexei Kitaev, and Oded Regev. The complexity of the local hamiltonian problem. *SIAM Journal on Computing*, 35(5):1070–1097, 2006.

[KR03] Julia Kempe and Oded Regev. 3-local hamiltonian is qma-complete. *arXiv preprint quant-ph/0302079*, 2003.

[KSV02] A Y Kitaev, A H Shen, and M N Vyalyi. *Classical and Quantum Computation.* 2002.

[Liu06] Yi-Kai Liu. Consistency of local density matrices is qma-complete. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 438–449. Springer, 2006.

[Vad12] Salil P Vadhan. *Pseudorandomness.* 2012.

[Wat00] John Watrous. Succinct quantum proofs for properties of finite groups. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 537–546. IEEE, 2000.

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

Lecture 25: QMA(2)

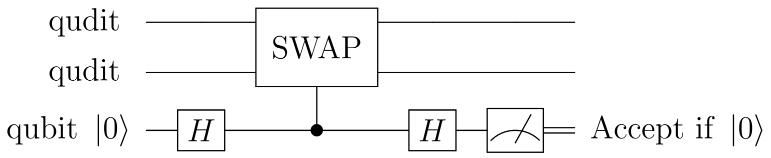
December 7, 2015

Lecturer: Ryan O'Donnell *Scribe: Yongshan Ding*

1 Introduction

So far, we have seen two major quantum complexity classes, namely BQP and QMA. They are the quantum analogue of two famous classical complexity classes, P (or more precisely BPP) and NP, respectively. In today's lecture, we shall extend to a generalization of QMA that only arises in the Quantum setting. In particular, we denote QMA(k) for the case that quantum verifier uses k quantum certificates.

Before we study the new complexity class, recall the following "swap test" circuit from homework 3:



When we measure the last register, we "accept" if the outcome is $|0\rangle$ and "reject" if the outcome is $|1\rangle$. We have shown that this is "similarity test". In particular we have the following:

- $\Pr[\text{Accept } |\psi\rangle \otimes |\varphi\rangle] = \frac{1}{2} + \frac{1}{2}| \langle\psi|\varphi\rangle |^2 = 1 - \frac{1}{2}d_{tr}(|\psi\rangle , |\varphi\rangle)^2$
- $\Pr[\text{Accept } \rho \otimes \rho] = \frac{1}{2} + \frac{1}{2}\sum_{i=1}^{d} p_i^2$, if $\rho = \{p_i \, |\psi_i\rangle\}$.

2 QMA(2): The 2-Prover QMA

Recall from last time, we introduced the QMA class where there is a prover who is trying to prove some instance x is in the language L, regardless of whether it is true or not. Figure. 1 is a simple picture that describes this.

The complexity class we are going to look at today involves multiple provers. Kobayashi et al. [KMY03] first introduced and studied the class QMA with multiple provers who are promised to be unentangled. Intuitively, multiple prover system can often help the verifier to make fewer mistakes. Probably for the same reason, police often wants to cross check among multiple criminals in order to catch lies. For simplicity, let's start with two provers. Similarly, we have the picture for QMA(2) as in Figure. 2:

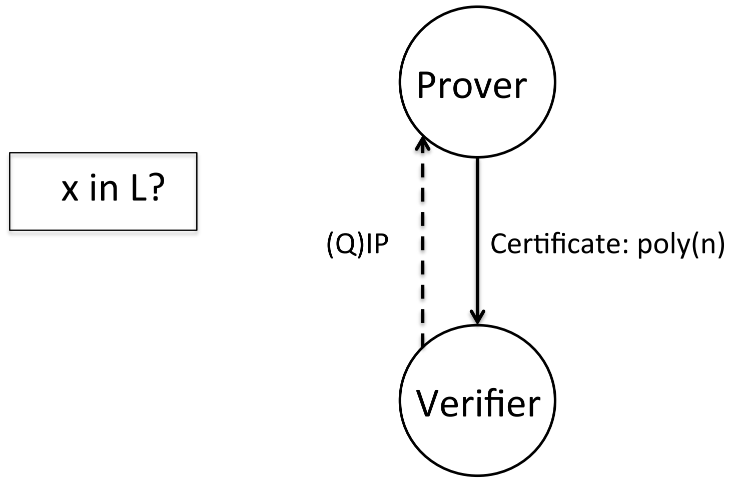


Figure 1: Schematics of QMA

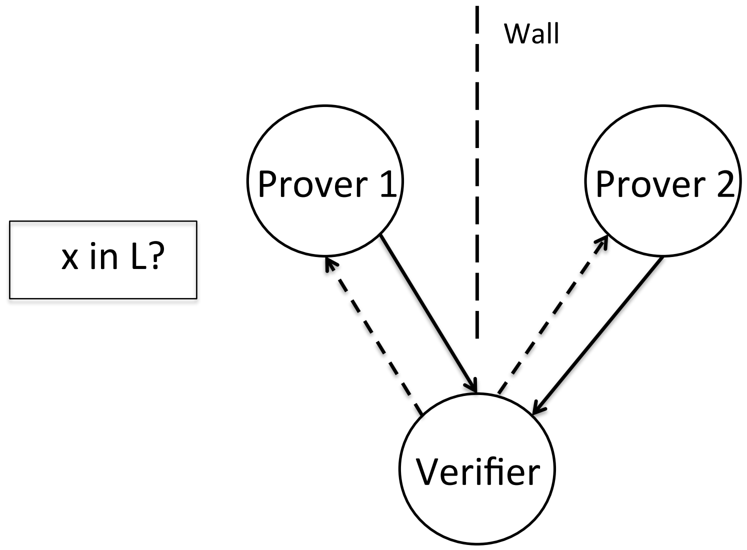


Figure 2: Schematics of QMA(2)

Notice that we have drawn a wall between the two prover so that they cannot communicate with each other. We will explain this assumption more formally very soon. For now, let's first think about the situation in the classical setting. Why don't we have a complexity class that assumes multiple non-interactive provers?

In fact, classically this situation is exactly the same as NP/MA, i.e. the verifier gains no power from the multiple provers. It is pointless to have two provers trying to convince the verifier, because one single prover can give that both sets of information to the verifier and still have the same effect. So the multiple-prover protocol can only help in an interactive setting.

However, in the quantum setting, the "wall" can prevent the provers from being entangled. In other words, assume in the two-prover case, we are granted the assumption that the verifier gets a quantum state $|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, with $|\psi_1\rangle$ from prover 1 and $|\psi_2\rangle$ from prover 2. The fact that we can write the overall state in the tensor form guarantees the states *to be unentangled*.

Let's now look at the quantum setting, and try to derive some properties of the new complexity class with multiple provers.

Definition 2.1. QMA(2) is the set of all languages L such that there exists a ("BQP") verifier V such that

- (Completeness) If $x \in L$, $\exists\ |\psi_1\rangle, |\psi_2\rangle$, each with $l = poly(n)$ qubits long, s.t.

$$\Pr[V(|x\rangle \otimes |\psi_1\rangle \otimes |\psi_2\rangle) \text{ accepts}] \geq 2/3 =: c$$

- (Soundness) If $x \notin L$, $\forall\ |\psi_1\rangle, |\psi_2\rangle$, each with $l = poly(n)$ qubits long, s.t.

$$\Pr[V(|x\rangle \otimes |\psi_1\rangle \otimes |\psi_2\rangle) \text{ accepts}] \leq 1/3 =: s$$

Recall from last time, we said the constants c, s don't matter for QMA because they end up to be the same class. It is not yet clear if this is the case here. In general, we can sometimes put all the parameters together in a compact form: $\text{QMA}_l(k)_{c,s}$, where l is the number of qubits in the certificate that the verifier receives from each prover, k is the number of provers, c is the *completeness* probability and s is the *soundness* probability. It is clear that $\text{QMA}(k) \subseteq \text{QMA}(k')$ for $k \leq k'$, since the verifier can simply interact with the first k provers and ignore the rest.

3 Power of Non-entanglement

It is conjectured that the multi-prover protocol with the non-entanglement promise is more powerful than QMA. In fact, Liu et al [LCV07] proposed a problem that is in QMA(2) but is not known to be in QMA, namely the N-representability problem. Several other works can be found in [BT09], [Bei10], [ABD$^+$09], [CD10]

Theorem 3.1 ([BT09]). *NP $\subseteq$ QMA(2) with* $l = \log(n), c = 1$ *and* $s = 1 - \frac{1}{poly(n)}$.

In other words, using 2 unentangled proofs, each with $O(\log n)$ qubits long, the verifier:

- accpets satisfiable formulas always;
- rejects unsatisfiable formulas with probability $\frac{1}{\text{poly}(n)}$.

Remark 3.2. It is still highly doubtful whether SAT $\in \text{QMA}_{\log}(2)$ with $c = 1, s = 0$, even for $l = o(n)$. Because if it were true, then we can solve SAT quantumly in $2^{o(n)}$ time.

Remark 3.3. It is also doubtful whether NP $\subseteq \text{QMA}_{\log}(2)$ with $c = 2/3, s = 1/3$. For the same logic, we can probably show NEXP $\subseteq \text{QMA}_{\log}(2)$ by extending the length of proofs to exponentially long.

Another invariant of the work is proposed by Aaronson et al. using the 3SAT problem:

Theorem 3.4 ([ABD$^+$09]). *NP $\subseteq QMA_l(k)_{c,s}$ with $l = \log m, k = \tilde{O}(\sqrt{m}), c = 1, s = 0.999$.*

Notice that the soundness probability is now a constant.

Furthermore, Harrow and Montanaro [HM12] have shown that it is possible to apply efficient soundness amplification and prover reduction. And in fact, it is shown by Kobayashi et al. [KMY03] that soundness amplification and prover reduction are equivalent. Intuitively, at least one of the direction sounds plausible. Suppose we used many copes of proofs to reduce error. We let each copy of the proof be sent by each of the k provers. If k can be reduced to 2, then error reduction is also possible.

4 Prover Reduction

Theorem 4.1 ([HM12]). *k number of unentangled provers can be reduced to only two.*

Before we prove this theorem, let's first try to reduce down to one provers. In other words, we want to show QMA(k) $\subseteq$ QMA(1). It is easy to see that the verifier now receives one giant state $|\Psi\rangle$, and the prover can virtually send anything. There is no way we can still guarantee the promise of non-entanglement we had before (i.e. $|\Psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_k\rangle$). Now, let's see how could QMA(2) help us reassure the non-entanglement, thereby showing QMA(k) $\subseteq$ QMA(2).

Clearly, it is equivalent to test non-entanglement of any k-partite state $|\Psi\rangle$ using just two (identical) copies of $|\Psi\rangle$. It is possible to remove the assumption of "identical" copies. We will leave it to the reader.

Now if we are given two copies of $|\Psi\rangle$, the problem is then reduced to testing purity, because if $|\Psi\rangle$ is an entangled states, then discarding $|\psi_2\rangle, \ldots, |\psi_k\rangle$ will give us a mixed state at $|\psi_1\rangle$.

[Insert product test diagram]

Therefore, as shown in the diagram above, we can apply "swap test" to the n pairs of corresponding $|\psi_i\rangle$ in each copy of $|\Psi\rangle$. ■

Combining the results, we thereby obtain that

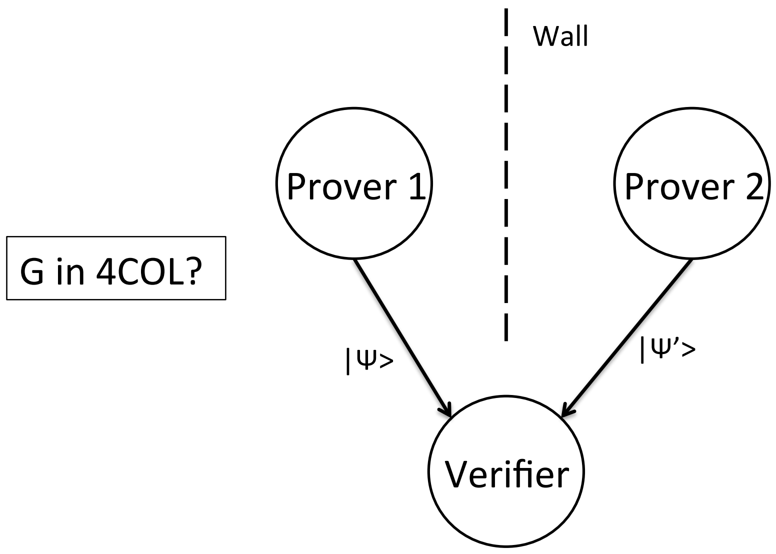


Figure 3: Schematics of QMA(2) for verifying 4COL

- 3SAT $\in$ QMA($\tilde{O}(\sqrt{m})$) where each prover sends $\log(m)$ qubits, with completeness $c = 1$ and constant soundness s.
- 3SAT $\in$ QMA(2) where each prover sends $\tilde{O}(\sqrt{m})$ qubits, with completeness $c = 1$ and constant soundness s.

5 NP $\subseteq$ QMA(2)

In this section, we want to prove the following theorem using 4COL:

Theorem 5.1 ([BT09])**.** *NP $\subseteq$ QMA(2). In particular, 4COL $\in$ QMA$_{\log}(2)$ with $c = 1, s = 1 - \frac{1}{poly(n)}$*

Given input graph $G = (V, E)$, we have the QMA(2) protocol to determine if $G \in$ 4COL: WLOG, let $n = |V|$ be a power of 2. We want to show:

- (Completeness) If $G \in$ 4COL, $\exists\ |\psi\rangle, |\psi'\rangle$ s.t.

$$\Pr[\text{Verifier accepts}] = 1$$

- (Soundness) If $G \notin$ 4COL, $\forall\ |\psi\rangle, |\psi'\rangle$ s.t.

$$\Pr[\text{Verifier rejects}] \geq \frac{1}{\text{poly}(n)}$$

In other words, we want to always accept correct proofs and to catch a lie with non-negligible probability. Let's first consider the easy case ("completeness"), with $G \in$ 4COL. What should the prover send? In this case, trying to help the verifier, the prover might as well send a valid coloring of the graph.

Let $\chi : V \to \{0, 1, 2, 3\}$ be a valid coloring, where each color is labeled as integers $1, \ldots, 3$. Then the two provers will send the state:

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{v \in V} |v\rangle |\chi(v)\rangle$$

Notice that the first "vertex register" $|v\rangle$ contains $\log_2(n)$ qubits, and the "color register" $|\chi(v)\rangle$ contains 2 qubits.

The verifier will thus the following protocol consisting of 3 testing components (by performing one of the following three tests with equal probability):

- Coloring Test
- Equality Test
- Uniformity Test

Let's now look at each test separately.

1. "Coloring":
   - Measure the two proofs: $|\psi\rangle \to |v\rangle |c\rangle$, $|\psi'\rangle \to |v'\rangle |c'\rangle$
   - If $(v, v') \in E$, then accept if $c \neq c'$, and reject otherwise.
   - Else if $v \neq v'$, then accept if $c \neq c'$, and reject otherwise.

   Notice from the second item that if $|\psi\rangle = |\psi'\rangle$ then we are done, since it has $\geq \frac{1}{n^2}$ chance of catching a lie.

2. "Equality":
   - Use "swap test" on $|\psi\rangle, |\psi'\rangle$: This is good for completeness, because if true, the verifier will accept with probability 1.

3. "Uniformity": Recall Fourier Transform $\mathcal{F}_N$ over $\mathbb{Z}_N$, we know that the indicator function can be transformed to a uniform superposition, up to a phase.
   - Apply $\mathcal{F}_N$ to the color register. In the good case, we obtain

   $$\frac{1}{\sqrt{n}} \sum_{v \in V} |v\rangle \left(\frac{1}{2} \sum_{c \in \{0,1,2,3\}} i^{c \cdot \chi(v)} |c\rangle \right)$$

   where i is the forth-root-of-unity.

- Measure the color register: Then reject if we get $|00\rangle$, and accept otherwise.

We have therefore shown that the protocol has completeness $c = 1$: i.e. If $G \in$ 4COL, then there exists a proof that the verifier accepts with probability 1. We will leave the soundness proof of this protocol to the reader. For detailed results, one can refer to [BT09].

References

[ABD$^+$09] Scott Aaronson, Salman Beigi, Andrew Drucker, Bill Fefferman, and Peter Shor. The power of unentanglement. volume 5, pages 1–42, 2009.

[Bei10] Salman Beigi. Tnp vs. qmalog(2). *Quantum Information and Computation*, 54(1 / 2):0141–0151, 2010.

[BT09] Hugue Blier and Alain Tapp. All languages in np have very short quantum proofs. *In ICQNM '09: Proceedings of the 3rd International Conference on Quantum, Nano and Micro Technologies*, pages 34–37, 2009.

[CD10] Jing Chen and Andrew Drucker. Short multi-prover quantum proofs for sat without entangled measurements. 2010.

[HM12] Aram Harrow and Ashley Montanaro. Testing product states,quantum merlin-arthur games and tensor optimization. *J. ACM*, 60(1):1–43, 2012.

[KMY03] Hirotada Kobayashi, Keiji Matsumoto, and Tomoyuki Yamakami. Quantum merlin-arthur proof systems: Are multiple merlins more helpful to arthur? *Algorithms and Computation*, 2906:189–198, 2003.

[LCV07] Yi-Kai Liu, Matthias Christandl, and Frank Verstraete. Quantum computational complexity of the n-representability problem: Qma complete. *Physical Review Letters*, 98(11), 2007.

Quantum Computation **(CMU 15-859BB, Fall 2015)**

Lecture 26: QIP and QMIP

December 9, 2015

Lecturer: John Wright *Scribe: Kumail Jaffer*

1 Introduction

Recall QMA and QMA(2), the quantum analogues of MA and AM. Today we will look at QIP, the quantum analogue of the classical IP, the class of languages with polynomial size "interactive proof systems."

An interactive proof system consists of two algorithms, P, the prover, and V, the verifier. P has access to infinite computational power, while V is limited to computing with the assumptions of BPP. They are allowed to exchange polynomially messages, and at the end the verifier outputs whether or not the input is in the language. More precisely,

Definition 1.1. a language L is in IP if there exists a BPP verifier V such that the following two conditions hold:

1. (completeness) If $x \in L$, there exists a prover P such that the system consisting of P and V accepts x with probability $\geq 2/3$.
2. (soundness) If $x \notin L$, for all provers Q, the system consisting of Q and V accepts x with probability $\leq 1/3$.

This is like MA and AM, except the machines can exchange polynomially many instead of 1 or 2 messages.

The discovery of the fact that IP = PSPACE is one of the great stories of classical complexity [Sha92][LFKN92]. Let's dive into the quantum version of this story, which is being written today.

2 Definition of QIP

Definition 2.1. a language L is in QIP if there exists a BQP verifier V such that the completeness and soundness conditions hold. The prover and the verifier are additionally allowed to exchange quantum messages.

Let's also define another useful notion:

Definition 2.2. QIP(k) is the set of languages with quantum interactive proof protocols of length k.

Using this definition, the following facts are clear

Fact 2.3. $\mathsf{QIP} = \bigcup_{f \text{ is polynomial}} \mathsf{QIP}(f(n))$

Fact 2.4. $\mathsf{QIP}(0) = \mathsf{BQP} \subseteq \mathsf{QIP}(1) = \mathsf{QMA} \subseteq \mathsf{QAM} \subseteq \mathsf{QIP}(2) \subseteq \mathsf{QIP}(3) \subseteq \ldots \mathsf{QIP}$.

Here's another definition:

Definition 2.5. $\mathsf{QIP}(k, p_1, p_2)$ is the set of languages with quantum interactive proof protocols of length k, correctness probability of accepting at least p_1 and soundness probability of accepting at most p_2.

It is not immediately clear that QIP is robust to tweaking the probabilities of the soundness and correctness conditions, but it turns out that it is indeed the case that

Fact 2.6. $\mathsf{QIP}(k, p_1, p_2) = \mathsf{QIP}(k, 1, 2^{-poly})$, *as long as* $p_1 - p_2 \geq poly^{-1}$ *[KW00].*

Finally, Here are two grab bag facts:

Fact 2.7. $\mathsf{IP} \subseteq \mathsf{QIP}$

The proof is very similar to the proof that $\mathsf{MA} \subseteq \mathsf{QMA}$, with the key being to measure the quantum state before doing anything else.

Fact 2.8. *Without loss of generality, we can assume in any interactive proof protocol that the last message sent is from the prover to the verifier.*

Clearly, the verifier accomplishes nothing by sending a message the other way and outputting before receiving a response.

With definitions out of the way, let's look at a problem in QIP to better understand the nature of the class.

3 Quantum State Distinguishability

You are given two mixed states, $\rho_0, \rho_1 \in \mathbb{C}^{d \times d}$ in the form of classical descriptions of quantum circuits that output these states. The task is to output YES if $d_{tr}(\rho_0, \rho_1) \geq .9$, and NO if $d_{tr}(\rho_0, \rho_1) \leq .1$.

Where d_{tr} is the trace distance. Equivalently, we saw in a previous lecture that

Fact 3.1.

$$d_{tr}(\rho_0, \rho_1) = 2 \left\{ \begin{array}{c} \textit{optimal probability of guessing } b, \\ \textit{given a uniformly random } \rho_b \textit{ from } b = \{0, 1\} \end{array} \right\} - 1$$

One natural thing we might want to try is to send the optimal POVM for distinguishing the two states as a certificate. Then the verifier can just measure random states with the POVM many times, and if it gets back the same one it put in sufficiently many times, we can say YES. The trouble with this is that it's unclear how we can encode a POVM efficiently into a message. Even if we just use the Pretty Good Measurement, we don't know an efficient way to encode it. If we could find a nice way to encode it, we'd solve this problem in QMA.

But if we allow ourselves a polynomial length protocol, we have more freedom. In fact, it turns out we don't need to send any information about how the prover solves the problem at all (!). We can use the standard coke-pepsi trick from classical zero knowledge proofs.

Claim 3.2. *The following protocol solves the problem in* QIP*:*

1. *Verifier picks $b \in \{0, 1\}$ uniformly at random, and sends the prover ρ_b without saying which one it is.*
2. *Prover responds with a guess, b', for b*
3. *Verifier accepts if and only if $b = b'$.*

Proof. If $d_{tr}(\rho_0, \rho_1) \geq 0.9$, then by fact 3.1, there's a measurement which gives the prover a probability $\geq (1 + 0.9)/2 = 0.95$ of guessing b correctly. So the correctness condition is satisfied.

On the other hand, if $d_{tr}(\rho_0, \rho_1) \leq 0.1$, then again by fact 3.1, there's no measurement which gives the prover a probability better than $(1 + 0.1)/2 = 0.55$ of guessing correctly. So the soundness condition is satisfied. □

This protocol has the nice property that is "zero-knowledge", that is, the protocol gives the verifier no information about how to solve the problem itself. It only allows it to conclude that the prover knows what it's doing.

Further, it turns out out that the Quantum State Distinguishability problem is complete for QSZK, the class of languages with zero-knowledge quantum interactive proof protocols [Wat02].

Now let's turn to the related problem of quantum circuit distinguishability.

3.1 Quantum Circuit Distinguishability

You are given two classical descriptions of quantum circuits, c_0 and c_1. The task is to determine if they compute "noticeably" different functions.

To make the notion of "noticeably" different more concrete, we use the following definition

Definition 3.3 (Diamond distance)**.**

$$d_\diamond(c_0, c_1) = \max_{\text{pure states } |\chi\rangle} d_{tr}(c_0 |\chi\rangle, c_1 |\chi\rangle)$$

The task, then, is to output YES if $d_\diamond(c_0, c_1) \geq 0.9$, and NO if $d_\diamond(c_0, c_1) \leq 0.1$.

Claim 3.4. *The following protocol solves the problem in* QIP*:*

1. *Prover sends $|\chi\rangle$, a state claimed to cause noticeably different outputs on c_0 and c_1.*
2. *Verifier picks $b \in \{0, 1\}$ uniformly at random, calculates $c_b |\chi\rangle$ and sends it, without saying which one it is.*
3. *Prover responds with a guess, b' for b.*
4. *Verifier accepts if and only if $b = b'$.*

Proof. This clearly works for the same reason that our protocol for Quantum State Distinguishability worked. □

4 QIP collapses to QIP(3) (!!)

Kitaev and Watrous showed in 2000 that every protocol in QIP can be turned into a protocol using only 3 messages [KW00]. In particular, they showed

Theorem 4.1. $\mathsf{QIP}(k, 1, 1 - \epsilon) = \mathsf{QIP}(3, 1, 1 - \frac{\epsilon^2}{4k^2})$

Proof (sketch). Note first that any quantum interactive proof looks something like the following diagram.

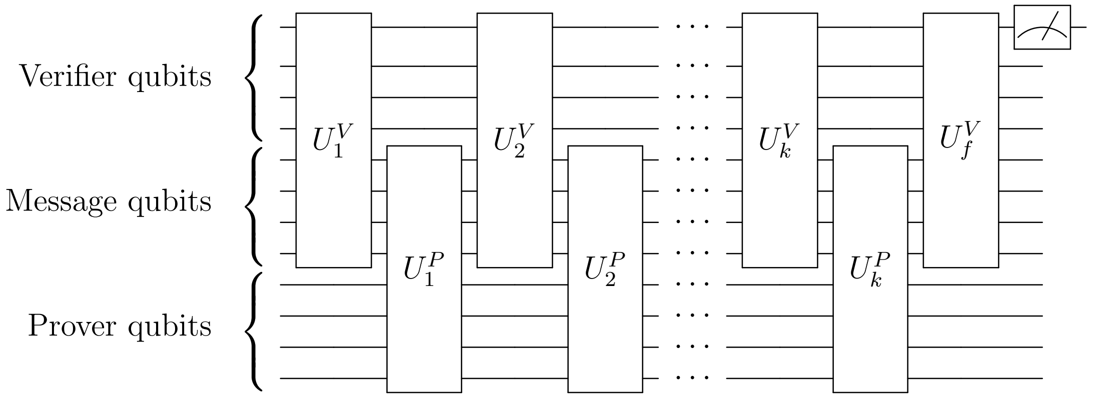


Figure 1: Quantum Interactive Proof System

That is, it consists of multiple rounds of the verifier and the prover applying unitary transformations (U_i^V for the verifier, and U_i^P for the prover) to a set of qubits, where they share some "message qubits", and both the prover and the verifier have their own private qubits that the other cannot interfere with. The protocol ends with the Verifier applying a final unitary transformation, U_f^V, and measuring the first qubit. If it is a 0 it rejects, if it is a 1 it accepts.

Let's call the state before the i'th verifier unitary $|\boldsymbol{V}_i, \boldsymbol{M}_i, \boldsymbol{P}_i\rangle$, where the entries of the tuple are the verifier's qubits, the message qubits, and the prover's qubits, respectively. A transcript is a list of all the states. The following protocol proves the theorem.

1. Prover generates a transcript for the full run of the original length $2k+1$ protocol on the given input, and sends it to the verifier. Verifier preforms U_f^V on $|\boldsymbol{V}_k, \boldsymbol{M}_k\rangle$ and rejects if the first bit of the result is not 1.

2. Verifier prepares entangled state $|b, b'\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle)$, and picks $i \in \{1 \dots k-1\}$ uniformly at random. Verifier applies V_i to $|\boldsymbol{V}_i, \boldsymbol{M}_i\rangle$, controlled swaps $\boldsymbol{V}_i$ and $\boldsymbol{V}_{i+1}$ with control bit b, and sends $\boldsymbol{V}_i, \boldsymbol{V}_{i+1}, b', i$ to the prover.

3. Prover applies U_i^P to $|\boldsymbol{M}_i, \boldsymbol{P}_i\rangle$ and controlled swaps it with $|\boldsymbol{M}_{i+1}, \boldsymbol{P}_{i+1}\rangle$ with control bit b', and sends it back. Verifier receives b' from prover, controlled nots it with b, applies hadamard to b and accepts if and only if result is 0

The high level idea of this protocol is that the verifier picks a random step of the proof to check, and checks it by doing the computation for that step itself, preforming a swap test to confirm that it is what the prover claimed it was. It is easy to show that if the original protocol accepts with certainty, this protocol does as well.

Proving soundness is a bit more difficult. For more details, see [KW00].

□

In fact, it turns out that the above protocol can be somehow formulated as a semidefinite program solvable in exponential time, showing QIP $\subseteq$ NEXP [KW00]. It even turns out to be true that QIP = PSPACE = IP, so quantum gives us no additional power in the domain of interactive proofs [JJUW10].

5 QMIP

One interesting variation on the IP setting that's been widely studied is to allow multiple independent non-communicating provers, resulting in the classical class MIP. Intuitively, this would seem to give us more power because it is the mathematical formalism for the classic policework trick of separating two co-conspirators before interrogating them in an effort to discover the truth. A famous result of Babai, Fortnow, and Lund says that MIP $\subseteq$ NEXP [BFL90].

We can generalize this to the quantum setting where we are again allowed multiple non-communicating provers, except that the verifier is in BQP, and the provers are allowed to prepare an entangled state before beginning the protocol. This is, of course, a highly realistic assumption, since you couldn't prevent two co-conspirators from walking into their interrogation rooms with halves of EPR pairs in their pockets. The resulting class is called QMIP.

Currently, we don't know much about QMIP. It is not immediately clear even that MIP $\subseteq$ QMIP, since the entanglement of the provers prevents us from simply measuring to avoid sneaky quantum behavior. In 2012, however, it was shown that this is the case, as a corollary of the fact that $\mathsf{NEXP} \subseteq \mathsf{QMIP}$ [IV12]. It's also true that $\mathsf{QMIP} = \mathsf{MIP}^*$, where MIP^* is MIP except the provers are allowed to share an entangled state before the protocol (note that the verifier is of BPP and not BQP), so the power of QMIP lies in prover entanglement and not in quantum messaging [RUV13].

The most glaring gap in our knowledge is the lack of an analogue of $\mathsf{MIP} \subseteq \mathsf{NEXP}$. In fact, it's quite a bit worse than that: We don't know any upper bound at all! Not even the class of decidable languages is known to be an upper bound. For all we know, QMIP could be powerful enough to solve the halting problem, or indeed all possible problems.

References

[BFL90] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, volume 1, pages 16–25, October 1990.

[IV12] Tsuyoshi Ito and Thomas Vidick. A multi-prover interactive proof for NEXP sound against entangled provers. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 243–252, Washington, DC, USA, 2012. IEEE Computer Society.

[JJUW10] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP = PSPACE. *Commun. ACM*, 53(12):102–109, December 2010.

[KW00] Alexei Kitaev and John Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 608–617, New York, NY, USA, 2000. ACM.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, October 1992.

[RUV13] Ben W. Reichardt, Falk Unger, and Umesh Vazirani. A classical leash for a quantum system: Command of quantum systems via rigidity of CHSH games. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 321–322, New York, NY, USA, 2013. ACM.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, October 1992.

[Wat02] J. Watrous. Limits on the power of quantum statistical zero-knowledge. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 459–468, 2002.