Contents

1 Church-Turing Thesis

We begin by asking the question, what can be computed? How do we make a mathematically precise definition of what it means to be computable. Turing created a model to define computability using a Turing Machine. It is an abstract mathematical model, and the claim is that anything that can be computed by a Turing machine is computable, and anything else is not computable. Roughly speaking, it is a tape machine with a pointer to a section of the tape, and we can either read or write onto the tape, or move it or keep it in the same location. Everything that can be computed can be done by this machine.

We have a mathematical system based on functions, known as lambda calculus, which is a mathematical representation of computation. Both Turing machines and lambda calculus can be used to define the class of computable functions, and although they are both different, they generate the same class of functions.

How is this connected to quantum computing? Well we have a definition of computability, but whether this covers all computable operations is unknown, there might be some real world function that is not doable by a Turing machine but is computable.

Theorem 1.1. *The Church-Turing thesis states that every effectively calculable function is a computable function.*

However, we have examples that challenge the Church-Turing Thesis. The first is randomize computation. In λ-calc, or in Turing machines, we don't have any random numbers, whereas in the real world we do have random number generation (for example in sorting algorithms or prime number testing). This is an open question in complexity theory. We don't know whether we can model random number generation mathematically. This raises the extended Church-Turing Thesis, which includes randomness into the computational model. You can think of this as adding another tape to the Turing machine that is full of random numbers. This is a more powerful model than the regular Turing machine, and can cover functions that include random numbers.

The second challenge to the Church-Turing thesis is quantum computation. We don't know anything that is provable that separates quantum computation from classical computation fully. If we look at reality, we don't know how to simulate a quantum computation on a classical computer without using exponential resources, let alone polynomial overheads.

2 Reversible Computation

Reversible computation is a general concept, and it is based on the fact that we want to reverse engineer the effect of a computation, and get the inputs from the outputs. This is important because quantum computation must be reversible, you cannot lose information. The motivation for classical reversible computation is that reversible computations dissipate no heat from the loss of information (there will be some heat generated by the electronics but the actual information loss also generates heat).

Note that we can make every classical computation reversible, so we can turn irreversible computations into reversible ones.

Since quantum computation is less high level than classical computation currently, we think of things such as operations and algorithms as circuits. These circuits map boolean inputs to boolean outputs:

$$f : \{0,1\}^n \to \{0,1\}^m$$

Suppose we want to find the smallest universal gate set, the smallest set of gates that can be used to generate all possible circuits. Classically, the set of gates $\{NOT, AND, OR\}$ is universal, and we also know that $\{NAND\}$ is itself universal.

Suppose we want to convert an AND gate to a reversible computation. The AND gate takes in x and y, and returns $x \wedge y$. This is irreversible, because we cannot get the inputs from the output. We can define the Toffoli gate, which is a 3-qubit input, 3-qubit output gate, which takes in x, y, and z, and returns x, y, and $(x \wedge y) \oplus z$. We can see that if we apply another Toffoli gate, the value in the z bit will be

$$(z \oplus (x \wedge y)) \oplus (x \wedge y) = z$$

For the case of the AND gate, we can use the Toffoli gate, and have $z = 0$, which will take in

$$x, y, 0 \to x, y, 0 \oplus (x \wedge y) = x, y, x \wedge y$$

Thus we can see that we have increased the number of qubits, but have made the AND gate reversible. Similarly, we can implement a reversible NOT:

$$x, 1, 1 \to x, 1, 1 \oplus (x \wedge 1) = x, 1, \bar{x}$$

We also care about a gate known as a FAN-OUT gate, also called a copy gate:

$$\text{FAN-OUT} = \text{TOFFOLI}(x, 1, 0) = x, 1, x$$

If we convert our irreversible gates to reversible gates and substitute them in, we see that the computation has been converted to a reversible computation, just with the addition of ancillary qubits. We see that if we have a irreversible circuit of n inputs, we have n output bits that are the same as the inputs, m output bits, and then some number of ancillary qubits.

We can use a simple trick to clean the workspace, because the ancilla qubits must be intialized to some values. We have n input bits, m output bits, initially set to 0, and then our workspace bits. The circuit keeps the input bits the same, modifies the outputs, and then leaves our workspace bits in unknown states. The trick is to use the FAN-OUT gate to copy each output to a new register of bits. After copying them to the new register, we can run the circuit's inverse on the initial register, which will by definition of a reversible circuit, leave our workspace and the output bits back to their initial states.

We can represent quantum states and circuits via vectors and matrices. Suppose we have a NOT gate, which maps 0 to 1 and 1 to 0. This is a mapping between two objects in a vector space, we can represent 0 via a 2D vector:

$$0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

And we can express the NOT gate as a matrix:

$$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

And we see that if we act this matrix on a 0 vector, we get the 1 vector, and vice-versa. Also note that when we represent an operation, we can think of it as a permutation, and since our basis vectors will only have 1 nonzero element each, the matrix will have only 1 nonzero element per column.

So we have seen the matrix/vector formalism for circuits, but how do we implement randomized computation? When we represent random bits, we have a probability for each measurement, 0 and 1, represented by p_0 and p_1 respectively. We know that $p_0 + p_1 = 1$. We can represent a random bit as a linear combination of the states they represent:

$$p_0 |0\rangle + p_1 |1\rangle = \begin{pmatrix} p_0 \\ p_1 \end{pmatrix}$$

If we apply a NOT gate to this via the matrix:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_0 \end{pmatrix}$$

This random bit represents the state

$$p_1 |0\rangle + p_0 |1\rangle$$

This is also a random bit, but it has flipped probabilities for the measurements. Also note that this allows us to have the entire state known, rather than just sampling the probability distribution, the vector formalism allows us to know everything about the distribution.

Now lets move to the main topic of the course, what happens with quantum computing. The main keyword we use is quantum duality, which stems back to the discovery of wave-particle duality in regards to light. When it comes to computation, we have that interference can change our computation. At a high level, if we have a computation, we can think of it as a computation path. We take the input, do intermediary operations, and end up with an output. When we have randomized computation, we have more than one path, the more randomness, the more the number of branches. The probability of getting an output is the sum of probabilities of the paths that lead to the output you want.

If we bring in the idea of interference, we could have interference between different computation paths. Instead of having just a sum of probabilities, we can have paths that cancel each other out due to interference. Mathematically, this means that each path still has a probability, but the probability distribution is not as simple. In fact, quantum algorithms can be used to exploit the interference to cancel out paths that lead to outputs that you don't want, improving the probability of reaching the correct answer, surpassing the abilities of classical randomized computations.

3 Interference

Suppose we want to do some photonic computation. We have two optical paths, path 0 and path 1. Each path has its own detector, D_0 and D_1 respectively. If this was a classical experiment, the beam splitter splits the photon beam into a 50/50 split of probability for each path. We then see that the probability of being detected at D_0 is still 50/50, since the two beam splitters will keep the two paths equal.

If we treat it as a quantum experiment, we have to treat the beamsplitter's operation in a quantum formalism. We have that the operations for each beam splitter is

$$G_1 = G_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

To find what happens after the first beamsplitter G_1, we can apply it to the basis vectors:

$$G_1 |0\rangle \quad G_1 |1\rangle$$

We can then add on the effect of G_2 after this:

$$G_2 G_1 |0\rangle \quad G_2 G_1 |1\rangle$$

If we compute these two, we see that $G_2 G_1 |0\rangle = |0\rangle$. How can we interpret this? Well classically we said that we would have a 50/50 split, but now when using the quantum formalism we see that we never have a chance of seeing D_1 if we start with a 0. This is an example of interference, as we see that before the action of G_2, we have a chance of being measured as 1, but the second Hadamard gate removes the probability of getting a 1, it interferes and "removes" the path that leads to measuring a 1.

4 Axioms of QM

We will talk about 4 postulates, where we have discretized to finite spaces.

1. State space postulate
2. Evolution postulate
3. Composite system postulate
4. Measurement postulate

4.1 State Space Postulate

Let us begin with the state space postulate. We can think of any quantum state as a unit vector of $\mathbb{C}^d$. A classical bit can only take the states $|0\rangle$ and $|1\rangle$. However, a qubit can take advantage of the fact that $|0\rangle$ and $|1\rangle$ span the space, and we can represent a qubit as an arbitrary vector in the space:

$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$$

Where $\alpha_0, \alpha_1 \in \mathbb{C}^2$ and $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

We can also define the conjugate transpose:

$$\langle\psi| = |\psi\rangle^\dagger = \begin{pmatrix} \alpha_0^* & \alpha_1^* \end{pmatrix}$$

We can then define an inner product:

$$\langle\psi|\psi\rangle = \alpha_0^* \alpha_0 + \alpha_1^* \alpha_1$$

This is the same as the sum of the magnitudes that we had before. For a unit vector, the inner product with itself will be 1.

Using Dirac notation, we can represent $|\psi\rangle$ as $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, which gets rid of a lot of the clunkiness of the vector formalism. We can then write the bra:

$$\langle\psi| = (\alpha_0 |0\rangle + \alpha_1 |1\rangle)^\dagger = \alpha_0^* \langle 0| + \alpha_1^* \langle 1|$$

We can see that this does indeed get us the matrix form that we defined before. We can also write out the inner product using the two statements we just wrote, giving us 4 terms, two of which are 0, because we have an orthonormal basis. We see that we will be left with the exact definition that we had before.

$$\langle\psi|\psi\rangle = \alpha_0^*\alpha_0 + \alpha_1^*\alpha_1$$

We can also define an outer product, which is when we flip the order of the inner product, and we see that we get a matrix:

$$|\psi\rangle\langle\psi|$$

For example:

$$|0\rangle\langle 0| = \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}$$

In fact, it is true that $|0\rangle\langle 0| + |1\rangle\langle 1|$ generates the identity matrix.

Lets apply this identity matrix to some arbitrary vector:

$$I|\psi\rangle = (|0\rangle\langle 0| + |1\rangle\langle 1|)(\alpha_0|0\rangle + \alpha_1|1\rangle)$$

$$= \alpha_0|0\rangle\langle 0|0\rangle + \alpha_0|1\rangle\langle 1|0\rangle + \alpha_1|0\rangle\langle 0|1\rangle + \alpha_1|1\rangle\langle 1|1\rangle$$

$$= \alpha_0|0\rangle + \alpha_1|1\rangle$$

We see that this does indeed encode the identity operation.

How can we visualize a qubit? We have 4 parameters: which we can reduce:

$$|\psi\rangle = \gamma_0 e^{i\phi_0}|0\rangle + \gamma_1 e^{i\phi_1}|1\rangle$$

$$= e^{i\phi_0}\left(\gamma_0|0\rangle + \gamma_1 e^{i(\phi_1-\phi_0)}|1\rangle\right)$$

The outer term is known as a global phase, and is a non-measurable quantity. We also have that $\gamma_0^2 + \gamma_1^2 = 1$, and we can represent the 3 parameters that we have left as defining a unit sphere. Using spherical coordinates, we have that an arbitrary ket can be represented as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$

We now have 2 parameters, θ and ϕ, where θ represents the angle from the vertical axis, and ϕ is the sweep angle. We can also note that $|1\rangle$ is the South pole of the sphere, and $|0\rangle$ is the North pole. The $|+\rangle$ state, the equal superposition, is given by $\theta = \frac{\pi}{2}$ and $\phi = 0$. This lies along the x axis, and the $|-\rangle$ state is given by a rotation of π in the ϕ.

4.2 Evolution Postulate

The second postulate is the evolution postulate. For us, this tells us that two properties will always hold no matter what. The first is linearity. This means that if we have some mapping Φ:

$$\Phi(\alpha_0|0\rangle + \alpha_1|1\rangle) = \Phi(\alpha_0|0\rangle) + \Phi(\alpha_1|1\rangle)$$

The second property is that all mappings preserve the unity of the vectors, they cannot change the magnitude. Suppose we have some mapping M that is represented by a matrix. If we apply it to

any state, $|M|\psi\rangle|^2 = \langle\psi|M^\dagger M|\,|\psi\rangle\rangle = \langle\psi|\psi\rangle = |\,|\psi\rangle\,|^2$. This tells us that $M^\dagger M = I$. This is the derivation for the fact that quantum operators must be unitary.

Let us list some important 1-qubit unitary matrices. We have the identity:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

And the 3 Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

These matrices have some special properties:

$$\mathrm{Tr}(X) = \mathrm{Tr}(Y) = \mathrm{Tr}(Z) = 0$$

$$X^2 = Y^2 = Z^2 = I$$

Another thing to note is that the 4 matrices given, I, X, Y, Z, generate an orthonormal basis for $\mathbb{C}^{2\times 2}$. Another important gate is the Hadamard gate:

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

$$H|0\rangle = |+\rangle \quad H|1\rangle = |-\rangle$$

We can also define the computational basis, which is the basis given by the states $\{|0\rangle, |1\rangle\}$. Note that we can also create a basis using $\{|+\rangle, |-\rangle\}$.

If we are in the computational basis, the X gate is a bit flip, and Z is a phase flip. However, if we're in the Hadamard basis, we see that $X|+\rangle = |+\rangle$, there is no change. Looking at $X|-\rangle$, we see that $X|-\rangle = -|-\rangle$. We see that in the Hadamard basis, X acts as a phase flip, rather than a bit flip.

Looking at the Z gate, in the $\pm$ basis, we see that Z acts as a bit flip. The X and Z gates essentially have flipped their actions when we change their basis.

4.3 Composition Postulate

The third postulate refers to composite systems. When we have a system of two qubits, we rely on the tensor product space:

$$\mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2} \cong \mathbb{C}^{d_1, d_2}$$

We see that when we add more qubits, we add more and more tensor products to add more spaces:

$$\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 = \mathbb{C}^{2^n}$$

This is why quantum systems are hard to simulate classically, the state space grows exponentially.

Lets do an example. Suppose we have two qubits, both in state $|0\rangle$:

$$|0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Note that the order of the qubits matters, even if they are in the same state.

If we do the states $|0\rangle$ and $|1\rangle$:

$$|0\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

To prove that the order matters, we can compute $|1\rangle \otimes |0\rangle$:

$$|1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

We see that the two states have different statevectors.

For any state $|\psi\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$, if $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, then the state is a product state. Otherwise, it is known as an entangled state.

For example, if we think of a two-qubit state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$$

This is known as an EPR pair or a Bell pair.

4.4 Measurement Postulate

So far we have had no interaction between classical computation and quantum computation. We need som classicla method of reading out data, in order for our quantum computers to be usable. Consider a d dimensional quantum state:

$$|\psi\rangle \in \mathbb{C}^d$$

When measuring, we first choose an orthonormal basis for the space:

$$\{|\varphi_i\rangle\}$$

The outputs will be an integer from 1 to d, each one corresponding to a basis vector. Post-measurement, we change the quantum state itself.

We have our initial state $|\psi\rangle$, and we want to decompose it using the measurement basis:

$$|\psi\rangle = \sum \alpha_i |\varphi_i\rangle$$

We then say that the probability of outcome i corresponding to index i in the vector, is $|\alpha_i|^2$, and the post-measurement state when the outcome is i is $|\varphi_i\rangle$. This is a valid distribution because our initial state is a unit vector, and thus $\sum |\alpha_i|^2 = 1$.

Let's do an example of this. Assume we are in the state $|0\rangle \in \mathbb{C}^2$. Suppose we measure the qubit in the computational basis, $\{|0\rangle, |1\rangle\}$, and measure an identical qubit in the Hadamard basis, $\{|+\rangle, |-\rangle\}$.

For the first measurement:

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

We can see that the probability of measuring 0 is $|1|^2 = 1$, and the probability of measuring a 1 is 0. The post-measurement state will be $|0\rangle$.

If we instead measure in the Hadamard basis:

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

Which can be obtained by using $\langle +|0\rangle$ and $\langle -|0\rangle$ for the coefficients. For this reason, the probability of measuring $|+\rangle$ can also be given by $|\langle +|0\rangle|^2$. We see that we have a 50/50 split in probability, we can measure either $|+\rangle$ or $|-\rangle$ evenly. If we measure $|+\rangle$, our post-measurement state will be $|+\rangle$. We see that we start with the same state, but obtained different results by measuring in different bases, something that is not possible in classical computation.

4.4.1 Partial Measurement

Suppose we have a composite system:

$$|\psi\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$$

Suppose we wanted to measure the first qubit, but not the second one. We can decompose our state, but we are only given a basis for the first system, not the second system:

$$|\psi\rangle = \sum_{i=0}^{d_1-1} \alpha_i\,|\varphi_i\rangle \otimes |\phi_i\rangle$$

Where $\{|\varphi\rangle_i\}$ is the orthonormal basis of $\mathbb{C}^{d_1}$, and $|\phi_i\rangle$ is a unit vector, not necessarily a basis vector. Applying the partial measurement, the probability of outcome i is similar to what we see before, $|\alpha_i|^2$, but the difference is the post-measurement state, we are left in the state $|\varphi_i\rangle \otimes |\phi_i\rangle$. This is a product state, meaning that there is no entanglement post-measurement. This intuitively makes sense, as we have converted quantum information to classical information, and entanglement is a strictly quantum correlation.

Consider a two qubit system

$$|\psi\rangle = \left(\sqrt{\frac{1}{10}}\,|00\rangle + \sqrt{\frac{2}{10}}\,|01\rangle\right) + \left(\sqrt{\frac{3}{10}}\,|10\rangle + \sqrt{\frac{6}{10}}\,|11\rangle\right)$$

Suppose we apply a computational basis measurement to the first qubit. The first thing that we want to do is rewrite the state into the correct form:

$$|\psi\rangle = |0\rangle \otimes \left(\sqrt{\frac{1}{10}}\,|0\rangle + \sqrt{\frac{2}{10}}\,|1\rangle\right) + |1\rangle \otimes \left(\sqrt{\frac{3}{10}}\,|0\rangle + \sqrt{\frac{6}{10}}\,|1\rangle\right)$$

We now need to normalize our $|\phi_i\rangle$ vectors:

$$= \sqrt{\frac{3}{10}}\,|0\rangle \otimes \left(\sqrt{\frac{1}{3}}\,|0\rangle + \sqrt{\frac{2}{3}}\,|1\rangle\right) + \sqrt{\frac{7}{10}}\,|1\rangle \otimes \left(\sqrt{\frac{3}{7}}\,|0\rangle + \sqrt{\frac{6}{7}}\,|1\rangle\right)$$

This is the correct state, and thus we know that the probability of getting outcome 0 is $\frac{3}{10}$, with post-measurement state

$$|0\rangle \otimes \left(\sqrt{\frac{1}{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle\right)$$

Similarly, we see that the probability of measuring 1 is $\frac{7}{10}$, and the post-measurement state is

$$|1\rangle \otimes \left(\sqrt{\frac{3}{7}}|0\rangle + \sqrt{\frac{6}{7}}|1\rangle\right)$$

If we apply a full measurement using the computational basis, $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. This is easier to do than the partial measurement, and we can just read off the magnitude of the coefficients for the probability. Note that if we look at the 4 possible outcomes and their probabilities, if we sum the probabilities of the cases where the first qubit is $|0\rangle$, we see that it adds to $\frac{3}{10}$, the same outcome as the partial measurement we did on the first qubit. The same holds for measuring $|1\rangle$.

If we only care about the output distribution, then the two measurements are equivalent, but when we care about the output state, and whether we want to preserve the quantum nature of the state, then the two are different.

Let's do another example:

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

Doing a partial measurement in the computational basis to the first qubit, we can read off the probabilities, $P(|00\rangle) = \frac{1}{2}$, and $P(|11\rangle) = \frac{1}{2}$. Making this harder, we can use the Hadamard basis on the first qubit. We would go about this using the nice property that the EPR/Bell state has:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|{+}{+}\rangle + |{-}{-}\rangle) = \frac{1}{\sqrt{2}}(|\psi_0\psi_0\rangle + |\psi_1\psi_1\rangle)$$

Where $\{|\psi_0\rangle, |\psi_1\rangle\}$ form an orthonormal basis. Thus we see that we get the same results in the Hadamard basis as we did in the computational basis.

5 No-Cloning Theorem

Theorem 5.1. *There does not exist an operator U such that $U|\varphi\rangle \otimes |0\rangle = |\varphi\rangle \otimes |\varphi\rangle$. There does not exist a cloner that can map an unknown qubits state onto another qubit.*

Proof. Assume we have a unitary operator U such that $U|\varphi\rangle \otimes |0\rangle = |\varphi\rangle \otimes |\varphi\rangle$. Essentially, assume we have an operator that works as a cloner, for any state $|\varphi\rangle$.

Suppose $|\varphi\rangle = |0\rangle$. By definition:

$$U|\varphi\rangle \otimes |0\rangle = |0\rangle \otimes |0\rangle$$

And similarly if $|\varphi\rangle = |1\rangle$:

$$U|\varphi\rangle \otimes |0\rangle = |1\rangle \otimes |1\rangle$$

Now suppose we have some superposition of $|0\rangle$ and $|1\rangle$:

$$U(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle)$$

However, we can also compute this using linearity:

$$U(\alpha |0\rangle + \beta |1\rangle) \otimes |0\rangle = \alpha |00\rangle + \beta |11\rangle$$

We computed the same thing using two different routes, so these should be the same thing. We see that in general, these two are not equivalent to each other, leading to a contradiction. This unitary operator U does not exist. □

6 Quantum Protocols

6.1 Bennett's 4 Laws

We have the definition of reduction, $A \geq B$, which means that A can do the job of B. This isn't mathematically rigorous, but it helps. We can say that 1 qubit $\geq$ 1 classical bit. We can also say that 1 qubit $\geq$ 1 ebit, where an ebit is an entangled bit, a shared EPR pair in this context. The third statement is that 1 ebit + 1 qubit $\geq$ 2 classical bits. This is known as super-dense coding. The 4th is that 1 ebit + 2 bits $\geq$ 1 qubit. This is known as quantum teleportation.

Note that we currently are working with noiseless channels, there is no information loss when we transfer data from sender to receiver.

6.2 Super-dense Coding

We have a sender and a receiver. The sender gets some information, generally a boolean string $X^* \in \{0,1\}^*$, and sends it to the receiver over our channel, and the receiver wants to recover the boolean string. The trivial case is to just send over our string, but this is not the best method, because we could have information loss during the transmission. This means that we want to have some encoding of X^*, and then having a decoding that can recover the string. We can compress our information to a lower dimension, so that we have to send as little information as possible.

We assume that X^* is chosen uniformly randomly, and the channel has no noise. We also have a stored resource r, which is something shared by both the sender and receiver, such as a source of randomness. They cannot communicate using r, but they can both access it. We will see that if we share a quantum resource, we can use it to transmit information.

If we shared a classical resource like classical randomness r_c. Our encoding of the string will be some function:

$$\text{Encoding}(X^*, r_c)$$

and we have some function on the other end:

$$\text{Decoding}(\text{Encoding}(X^*, r_c), r_c)$$

Suppose instead we shared a quantum EPR pair:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

with the first qubit staying with the sender and the second qubit staying with the receiver. We can define the encoding:

$$\text{Encoding}(X^*, |\psi_1\rangle_{EPR})$$

And the decoding:

$$\text{Decoding}\left(\text{Encoding}(X^*, |\psi_1\rangle_{EPR}), |\psi_2\rangle_{EPR}\right)$$

We see that the decoder has more information, with the access to the second qubit of the EPR pair.

Let's now look at the protocol. We begin with an EPR pair, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. We begin with some local operation on the sender side, which applies the operation to the first qubit of the EPR pair. We then send this first qubit over to the receiver. The receiver then uses some operation on both qubits, to decode it. We can write this out more explicitly:

$$\frac{1}{\sqrt{2}}(|0\rangle_S \otimes |0\rangle_R + |1\rangle_S \otimes |1\rangle_R)$$

The first step will encode at the sender side. The sender has two classical bits, z and x. To encode, the sender applies $Z^z X^x$ to the sender qubit of the EPR pair. We can define our notation, $Z^{0/1}$ will do the identity if the bit is 0, and will do Z if the bit is 1, Similarly for X. Thus we see that this is two operations, first applying either X or I, and then Z or I. There are 4 possible outcomes:

$$(z, x) = (1, 1) \to ZX \quad (1, 0) \to ZI \quad (0, 1) \to IX \quad (0, 0) \to II$$

If we have the case $(0, 1)$ for example:

$$(0, 1) \to (X \otimes I)\,|\psi\rangle_{EPR} = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)$$

And the case $(1, 1)$:

$$(1, 1) \to (ZX \otimes I)\,|\psi\rangle_{EPR} = \frac{1}{\sqrt{2}}(-|10\rangle + |01\rangle)$$

And the other two cases:

$$(0, 0) \to (I \otimes I)\,|\psi\rangle_{EPR} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$(1, 0) \to (Z \otimes I)\,|\psi\rangle_{EPR} = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

We see that after the encoding, we have 4 possible 2-qubit states. We can see that the 4 different states are all orthogonal, they form a orthonormal basis for the 2-qubit states. These are sometimes known as the Bell basis:

$$\{B_{00}, B_{01}, B_{10}, B_{11}\}$$

And we note that all 4 of them are entangled states.

The second step is to send the S qubit over to the receiver side, so the receiver has both qubits. The receiver now wants to recover which case it is. Now we need to measure using the Bell basis, and that will return 100% correctly which basis state the pair was in. From this, the receiver will know exactly what the values of z and x were.

We see that we have started with sharing an EPR pair, and sending 1 qubit, allowed us to transmit 2 classical bits of information.

6.3 Quantum Teleportation

The goal here is to send quantum information without sending over a qubit.

We once again start with a shared EPR pair:

$$\frac{1}{\sqrt{2}}(|0\rangle_S |0\rangle_R + |1\rangle_S |1\rangle_R)$$

We are also given an arbitrary quantums state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$. In the whole system, we have 3 qubits in play. The whole quantum state can be written as

$$|\Phi\rangle = |\psi\rangle \otimes \text{EPR}$$

And we have that $|\psi\rangle$ is on the sender side, and 1 qubit of the EPR pair on the sender side, with only the second EPR qubit on the receiver side. We can write out the qubit states:

$$= (\alpha |0\rangle + \beta |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$$

$$= \frac{1}{\sqrt{2}}(\alpha |000\rangle + \alpha |011\rangle + \beta |100\rangle + \beta |111\rangle)$$

We need to do something to the two sender qubits that will somehow transfer the state of the arbitrary state to the receiver qubit.

The first step is that the sender performs a Bell-basis measurement on the two sender qubits. This is a partial measurement, and we'd have to write this out from the computational basis into the Bell basis. We can use the properties that

$$|00\rangle = \frac{1}{\sqrt{2}}(|B_{00}\rangle + |B_{10}\rangle)$$

$$|01\rangle = \frac{1}{\sqrt{2}}(|B_{01}\rangle + |B_{11}\rangle)$$

$$|10\rangle = \frac{1}{\sqrt{2}}(|B_{01}\rangle - |B_{11}\rangle)$$

$$|11\rangle = \frac{1}{\sqrt{2}}(|B_{00}\rangle - |B_{10}\rangle)$$

Using there relationships, we can rewrite our state using the Bell basis:

$$|\Phi\rangle = \frac{1}{2}(|B_{00}\rangle \otimes |\psi\rangle |B_{01}\rangle \otimes X |\psi\rangle + |B_{10}\rangle \otimes Z |\psi\rangle + |B_{11}\rangle \otimes XZ |\psi\rangle)$$

When we do the Bell-basis measurement, we will recover the (z, x) information. We then send the two classical x and z bits to the receiver. The receiver then applies the correction $Z^z X^x$ to the receiver side qubit.

For example, if both are 0, we do nothing, and we have the arbitrary state as the third qubit. The same goes for the two states where only 1 bit is 1. For the case where we have $z = x = 1$, we have $ZXXZ |\psi\rangle$. We see that the inner X gates cancel out, and then the Z gates cancel, and we are left with the arbitrary state that we want in the hands of the receiver. We have send 2 classical bits, shared 1 entangled pair, and have transferred a qubit's state.

We can interfact teleportation with the no-cloning theorem, to derive what is known as the monogamy of quantum entanglement.

Suppose we have 3 parties, where A and B share a strong correlation, and B shares a strong correlation with C. If this is true, we can use teleportation protocol we just derived and effectively create 2 copies of the unknown state that we start with. This is impossible via the no-cloning theorem, so thus we cannot share entangled pairs like the EPR pair between more than 2 parties.

7 Zeno Effect, Anti-Zeno Effect

7.1 Zeno Effect

Suppose we have a 1 qubit state, $|\varphi\rangle = \cos\theta\,|0\rangle - \sin\theta\,|1\rangle$. If we assume that θ is very small, so $\sin\theta \approx \theta$. If we measure the state, we expect that with very high probability we expect to get 0, since θ is very small, so $\cos^2\theta \gg \theta^2$, and thus $P(0) \gg P(1)$.

The Zeno effect, or the watchdog effect, is the case where we have a state very close to one of the basis vectors. If we then measure, we have a very high probability that we get that vector. Suppose we have some system drift over time. We can keep our state at that basis vector by repeating that measurement, which will snap it back to the basis vector. If we look at the state, it doesn't move.

7.2 Anti-Zeno Effect

We also have the Anti-Zeno effect, where we use the measurement to rotate our state. Say we want to go from the $|0\rangle$ state to the $|1\rangle$ state, via measurement. This is a strictly quantum method, since measurement in classical systems does not change the system. Suppose we begin with $|0\rangle$, and measure in the Hadamard basis. This will get us a 50/50 split between $|+\rangle$ and $|-\rangle$.

We can now do an anti-Zeno measurement, which is where we measure with a slightly rotated basis, $|0\rangle = \cos\theta\,|\hat{0}\rangle - \sin\theta\,|\hat{1}\rangle$. We now want to measure $|0\rangle$ in this rotated basis. We see that we will get $|\hat{0}\rangle$ with probability $\cos^2\theta$, and will get $|\hat{1}\rangle$ with probability $\sin^2\theta$. We want to maximize the probability of getting $|\hat{0}\rangle$, so we want to pick our rotation wisely. If we keep θ small, then we have a high chance of measuring $|\hat{0}\rangle$. This means that with high probability, we have moved the state from $|0\rangle$ to $|\hat{0}\rangle$. Now repeating this step with a rotated basis from our already rotated basis, we can shift the state closer to $|1\rangle$ once again. We can repeat this process $\frac{\pi}{2\theta}$ times, and we will have moved the state $|0\rangle$ to $|1\rangle$.

However, this is not guaranteed, since we have a small chance of having a failed rotation, where we measure the state $|\hat{1}\rangle$. For each step, we have a $\cos^2\theta$ chance to succeed, and a $\sin^2\theta$ change to fail. We want to find the outcome of succeeding. We see that we have a $(\cos^2\theta)^{\frac{\pi}{2\theta}}$ chance of succeeding every single time (since they are independent events). This isn't an easy to work with probability, so we can use the property that $P_s = 1 - P_f$. The protocol can fail at any one of the steps:

$$P_f = P(f_1 \cup f_2 \cup f_3 \cup \cdots \cup f_n)$$

Now using the inequality that $P(A \cup B) \leq P(A) + P(B)$. Using this:

$$P_f \leq \sum_{i=1}^{\frac{\pi}{2\theta}} P(f_i) = \theta^2 \frac{\pi}{2\theta} = \frac{\pi}{2}\theta$$

Thus the protocol success probability has the relation

$$P_s \geq 1 - \frac{\pi}{2}\theta$$

Looking at the limit, we see that if $\theta \to 0$, then the probability of success goes to 1, but this also means that the number of iterations goes to ∞.

8 Elitzur-Vaidman Bomb

We are given a box, and we want to know whether or not there is a bomb in it. The only action that we can take is either open the box, or not open the box.

If we open the box, we will know for sure whether or not there is a bomb, but if there is a bomb, it will explode.

Classically if we guess while keeping it closed, we have a 50/50 chance of guessing correctly, and if we open the box, we may blow up the bomb. There is no good way to solve this problem.

We first define a **quantum action** as an action that can be done in a superposition, such as a superposition of opening or not opening the box. We set up a qubit $|b\rangle$ in a superposition of $|0\rangle$ (not opening the box) and $|1\rangle$ (open the box):

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

This defines some quantum strategy, and the result will be given in another qubit.

We begin in the state $|b\rangle\,|0\rangle$. If there is no bomb:

$$|1\rangle\,|0\rangle \to |1\rangle\,|\text{no explosion}\rangle$$
$$|0\rangle\,|0\rangle \to |0\rangle\,|\text{no explosion}\rangle$$

We see that $|b\rangle\,|0\rangle \to |b\rangle\,|\text{no explosion}\rangle$.

If there is a bomb, we have a different situation:

$$|0\rangle\,|0\rangle \to |0\rangle\,|\text{no explosion}\rangle$$

$$|1\rangle\,|0\rangle \to |1\rangle\,|\text{explosion}\rangle$$

This means that if we have an arbitrary superposition for $|b\rangle$:

$$|b\rangle\,|0\rangle \to \alpha\,|0\rangle\,|\text{no explosion}\rangle + \beta\,|1\rangle\,|\text{explosion}\rangle$$

We have probability $|\alpha|^2$ of no explosion, and probability $|\beta|^2$ of explosion.

We define our protocol with a rotation operator of ε anti-clockwise:

$$R_\varepsilon = \begin{pmatrix} \cos\varepsilon & -\sin\varepsilon \\ \sin\varepsilon & \cos\varepsilon \end{pmatrix}$$

For the protocol, we initialize $|b\rangle$ to $|0\rangle$. We then rotate this by R_ε, and then test the quantum action $|b\rangle$. We then repeat these last two steps for $\frac{\pi}{2\varepsilon}$ iterations, and then measure $|b\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis.

Let us look at the first case, where this is no bomb. In step 3 of the protocol, we would always have $|b\rangle \to |b\rangle\,|\text{no explosion}\rangle$. Essentially, step 3 does not change anything. At the end of the protocol, we have that $|b\rangle = |1\rangle$, where the protocol boils down to the Anti-Zeno effect. If we then measure, the outcome will be $|1\rangle$, with probability of explosion 0.

For the second case, where there is a bomb, we see that the rotation will map $|b\rangle$ to $\cos\varepsilon\,|0\rangle_+ \sin\varepsilon\,|1\rangle$. Now when we test it in step 3, we have a $\cos^2\varepsilon$ chance of measuring that there is no explosion, and a $\sin^2\varepsilon$ change of having the bomb explode. For the same reasoning as for the Anti-Zeno effect, we have a very small chance of having the bomb explode. We see that in this case, we are left with $|b\rangle = |0\rangle$ after the rotations, which will not explode the bomb (this is because testing the action will either blow up the bomb or collapse you to $|0\rangle$, essentially restricting you with very high probability to $|0\rangle$ if there is a bomb). The probability of the bomb exploding will be given by the same method we used before:

$$P_e \leq \frac{\pi}{2\varepsilon}\varepsilon^2 = \frac{\pi}{2}\varepsilon$$

We see that if $\varepsilon \to 0$, the probability of the bomb exploding goes to 0, but once again the number of iterations goes to ∞.

We see that if you measure $|1\rangle$ at the end of the protocol, there is no bomb in the box, and if you measure $|0\rangle$, there is a very high probability that there is a bomb in the box.

9 Quantum Gates and Circuits

Let us look at some more complicated single qubit gates, elements of $\mathbb{C}^{2\times 2}$. We often see the Pauli matrices, which have eigenvalues ± 1, and are their own inverses, $X^2 = Y^2 = Z^2 = I$.

Since we can think of a qubit as a point on the Bloch Sphere, we can define operations that rotate the point along the sphere.

Let us talk about functions on Hermitian matrices. Hermitian means that $A = A^\dagger$. Given any function $f : \mathbb{R} \to \mathbb{R}$, and a Hermitian matrix A, we want to define $f(A) : \mathbb{C}^{d\times d} \to \mathbb{C}^{d\times d}$. We have two equivalent ways to extend these real functions to complex matrices. The first is to get the spectral decomposition of A, which is guaranteed to be doable since the matrix is Hermitian:

$$A = \sum \lambda_i\,|\psi_i\rangle\,\langle\psi_i|$$

We know that $\lambda_i \in \mathbb{R}$, since the matrix A is Hermitian. We define the function to act on the eigenvalues:

$$f(A) = \sum f(\lambda_i)\,|\psi_i\rangle\,\langle\psi_i|$$

The second definition is that we assume that the function f is series expandable:

$$f(x) = \sum_{i=0}^{\infty} \alpha_i x^i$$

We define the function on the matrix as

$$f(A) = \sum_{i=0}^{\infty} \alpha_i A^i$$

Where A^i is repeated matrix multiplication with itself, i times.

Let us show that these two are the same. Assume we have A with some spectral decomposition:

$$A = \sum \lambda_i |\psi_i\rangle \langle\psi_i|$$

We can write out A^2:

$$A^2 = \left(\sum \lambda_i |\psi_i\rangle \langle\psi_i|\right)\left(\sum \lambda_j |\psi_j\rangle \langle\psi_j|\right) = \sum \lambda_i^2 |\psi_i\rangle \langle\psi_i|$$

Where we used the fact that we are in an orthnormal basis. From this, we can see that

$$A^n = \sum \lambda_i^n |\psi_i\rangle \langle\psi_i|$$

Now using the second definition and inserting what we just found:

$$f(A) = \sum_{n=0}^{\infty} \alpha_n A^n = \sum_{i=0}^{\infty} \alpha_n \sum \lambda_i^n |\psi_i\rangle \langle\psi_i|$$

Now swapping the summations (which we can do because the series converges), we see that we are left with

$$\sum_i f(\lambda_i) |\psi_i\rangle \langle\psi_i|$$

Which is just the definition. We see that the two definitions are equivalent.

9.1 Pauli-Rotation Gates

We can define the gates $R_x(\theta) = e^{-i\frac{\theta}{2}X}$, $R_y(\theta) = e^{-i\frac{\theta}{2}Y}$, and $R_z(\theta) = e^{-i\frac{\theta}{2}Z}$. We have essentially applied the function $f(x) = e^{-i\frac{\theta}{2}x}$ to each of the Pauli matrices.

Let us look at the properties of these gates. We claim that for any Hermitian, $e^{-i\theta H}$ is a unitary matrix.

Proof. Since H is Hermitian, we have that $H = \sum \lambda_i |\psi_i\rangle \langle\psi_i|$, via spectral decomposition. We can then write out the application of the exponential:

$$e^{-i\theta H} = \sum e^{-i\theta\lambda_i} |\psi_i\rangle \langle\psi_i|$$

We see that the coefficients have magnitude 1, so this is unitary, but we can do this out more rigorously. We know that unitary means that $AA^\dagger = I$:

$$A = \sum \lambda_i |\psi_i\rangle \langle\psi_i| \quad A^\dagger = \sum \lambda_i^* |\psi_i\rangle \langle\psi_i|$$

From this, we have that

$$AA^\dagger = \sum \lambda_i \lambda_i^* |\psi_i\rangle \langle\psi_i| = I$$

This tells us that $\lambda_i \lambda_i^* = 1$, so $|\lambda_i|^2 = 1$. The eigenvalues must all be of norm 1. Thus we need the exponential to be of magnitude 1, which it is. Thus we have that $e^{-i\theta H}$ is unitary. □

We have defined $R_x(\theta) = e^{-i\frac{\theta}{2}X}$. What does this look like as a matrix? We know that it must be a 2×2 matrix. According to the first definition, we can use the spectral decomposition of X:

$$R_x(\theta) = e^{-i\frac{\theta}{2}} |+\rangle \langle +| + e^{i\frac{\theta}{2}} |-\rangle \langle -|$$

This is one way to write it out. We can also do this for R_z:

$$R_z(\theta) = e^{-i\frac{\theta}{2}} |0\rangle \langle 0| + e^{i\frac{\theta}{2}} |1\rangle \langle 1| = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

We can also use definition 2 to find the definitions:

$$e^{-i\frac{\theta}{2}X} = \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \left(-i\frac{\theta}{2}X\right)^n$$

We can see that we will have different powers of X, and we can use the property that $X^2 = I$, and then $X^3 = X^5 \cdots = X$. Now grouping these terms, we see that

$$\begin{aligned} e^{-i\frac{\theta}{2}X} &= \left(\sum_{n \text{ even}} \frac{(-1)^n}{n!} \left(-i\frac{\theta}{2}\right)^n\right) I + \left(\sum_{n \text{ odd}} \frac{(-1)^n}{n!} \left(i\frac{\theta}{2}\right)^n\right) X \\ &= \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} X \end{aligned}$$

We can then follow this for the Z matrix, and we find that

$$\begin{aligned} e^{-i\frac{\theta}{2}Z} &= \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Z \\ &= \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \end{aligned}$$

We see that the two definitions lead to the same result.

These are called rotation gates because they are rotations on the Bloch Sphere. We can think of any point on the Bloch sphere as some state

$$|\psi\rangle = \cos\frac{\varphi}{2} |0\rangle + e^{i\phi} \sin\frac{\varphi}{2} |1\rangle$$

Where $\varphi \in (0, \pi)$ and $\phi \in (0, 2\pi)$. An R_z rotation will change the state to

$$R_z(\theta) |\psi\rangle = \cos\frac{\varphi}{2} e^{-i\frac{\theta}{2}} |0\rangle + e^{i\phi} \sin\frac{\varphi}{2} e^{i\frac{\theta}{2}} |1\rangle$$

Rewriting this, we can pull out a phase:

$$= e^{-i\frac{\theta}{2}} \left[\cos\frac{\varphi}{2} |0\rangle + e^{i(\phi+\theta)} \sin\frac{\varphi}{2} |1\rangle\right]$$

We see that we have changed the phase of the $|1\rangle$ term, which corresponds to a geometric rotation along the z axis with angle θ.

We can actually do a rotation along any axis, not just the x, y, and z axes. In fact, any 1-qubit unitary matrix can be considered a rotation along some axis. Suppose we define some axis given by the vector $\vec{n} = (n_x, n_y, n_z)$, where $n_x^2 + n_y^2 + n_z^2 = 1$. We can define the arbitrary rotation:

$$R_{\vec{n}}(\theta) = e^{-i\frac{\theta}{2}(n_x X + n_y Y + n_z Z)} = e^{-i\frac{\theta}{2}\vec{n}\cdot\vec{\sigma}}$$

where $\vec{\sigma} = (X, Y, Z)$. This is doable because $(n_x X + n_y Y + n_z Z)^2 = I$.

9.2 General 1-qubit Unitary Gates

Theorem 9.1. *Any 1-qubit unitary operation can be decomposed as a series of rotations:*

$$G = e^{i\phi} R_z(\theta_3) R_x(\theta_2) R_z(\theta_1)$$

As long as the θs can be arbitrary.

The second version of the theorem is stronger, which is that we can use any two arbitrary rotations along the axes $\vec{n}_1$ and $\vec{n}_2$, so long as they are non-parallel.

The implication of this theorem is that it suffices to perform any rotation any over two non-parallel axes (with any angle), and this allows for the creation of any 1-qubit gate (up to a global phase).

9.3 Universality

We have talked about the CNOT, but let us highlight one of its properties, the fact that it is an entangling gate. Applying CNOT to a product state can result into an entangled state:

$$\text{CNOT} \left|+\right\rangle \left|0\right\rangle = \frac{1}{\sqrt{2}}(\left|00\right\rangle + \left|11\right\rangle)$$

Physically speaking, some other gates are easier to implement than a CNOT, but CNOT is the simplest entangling gate.

Theorem 9.2. *If we have a universal gate set for 1 qubit, and we add entangling gates, this forms a universal set for general quantum computation.*

One tip for dealing with the CNOT is that we can write it out in Dirac notation:

$$\text{CNOT} = \left|0\right\rangle \left\langle 0\right| \otimes I + \left|1\right\rangle \left\langle 1\right| \otimes X$$

In general, if we have a general controlled unitary:

$$\text{C-}U = \left|0\right\rangle \left\langle 0\right| \otimes I + \left|1\right\rangle \left\langle 1\right| \otimes U$$

If we are given a set of gates, how can we determine whether it is universal for quantum computation?

Let us do the example, of just Toffoli gates. This is not universal, because we don't have something that can generate a phase flip, like a Z gate. The Toffoli can only generate bit flips. We cannot implement any unitary U that has imaginary numbers in it.

Looking at the set $\{\text{CNOT}, X, Y, Z\}$, we see that there are angles of arbitrary rotations that we cannot access using just combinations of X, Y, and Z, and thus the gate set is not universal.

The set $\{\text{CNOT}, H, T\}$, where $T = e^{i\frac{\pi}{8}} \begin{pmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{pmatrix}$. This is universal, as H and T will allow for an arbitrary rotation along the Bloch Sphere ($HTHT$ and $THTH$ give rotations by irrational multiples of π over two non-parallel axes), and we have added in the CNOT gate.

9.4 Efficiency of Approximating General Functions

Suppose we have some function $f : \{0,1\}^n \to \{0,1\}$, and we want to find how large our implementation of this will be using our gateset. To do this, we can convert this to a counting problem, thinking about how many functions there are that map from n bits to 1 bit. We can think of how many different truth tables we can generate with n inputs and 1 output bit. We have 2^n inputs, with 2 choices for each input. This gives 2^{2^n} different boolean functions. This is double exponential, this is massive.

We define an efficient circuit as one of polynomial length. This will roughly be an exponential of some polynomial of n. This means that for most classical boolean functions, we don't have an efficient polynomial implementation. We can then carry this argument over to quantum circuits.

9.4.1 Approximation of Unitary Gates

We must first define the definition of approximating a unitary gate. We must first define some metric of distance between the ideal gate and the approximation. The idea of distance in a matrix space is not very intuitive, so this is difficult. If we think of two vectors, we can define a metric/distance between the two, which is the natural metric of Euclidean distance:

$$\| |\psi\rangle - |\phi\rangle \|_2$$

We define this on some arbitrary vector:

$$\| |\psi\rangle \|_2 = \sqrt{\langle\psi|\psi\rangle}$$

We see that the metric is just a special case where we take the norm of the difference of vectors:

$$\| |\psi\rangle - |\phi\rangle \|_2 = \sqrt{(\langle\psi| - \langle\phi|)(|\psi\rangle - |\phi\rangle)} = \sqrt{2 - 2\mathrm{Re}\,\langle\phi|\psi\rangle}$$

Where we have assumed that the two vectors are unit vectors. This can also be written as

$$= \sqrt{2 - 2\| |\phi\rangle \| \, \| |\psi\rangle \| \cos\theta}$$

We can bound this value, we know it must be greater than equal to 0, since its a real value and we have a square root, and we know that the maximal possible case is 2:

$$0 \leq \| |\psi\rangle - |\phi\rangle \|_2 \leq 2$$

If the two vectors are the same, this is 0, and if the two are opposites, we have 2. We see that we can use this as a distance metric for two vectors.

It is also noted that this satisfies the triangle inequality:

$$\| |\psi\rangle - |\phi\rangle \|_2 \leq \| |\psi\rangle - |\varphi\rangle \|_2 + \| |\varphi\rangle - |\phi\rangle \|_2$$

We can now apply this metric to matrices. Suppose we have two unitary matrices U and V:

$$\boxed{E(U,V) = \max \| U|\psi\rangle - V|\psi\rangle \|_2}$$

for all inputs $|\psi\rangle$. Essentially, we find an input that will maximize the distance between the application of the unitaries on the input, and that maximized distance is the distance between the unitaries.

Let us do an example of distance between matrices. Suppose we want to find the distance between the X and I matrices, $E(X, I)$. If we plug in $|\psi\rangle = |-\rangle$, we see that we will get $-|-\rangle$, which is opposite to the initial $|\psi\rangle$. This will get us a norm of 2, which is the maximum possible.

We can also apply the Triangle Inequality to the matrix distance:

$$\boxed{E(U, V) \leq E(U, W) + E(W, V)}$$

Proof. The left side will be

$$\max\|U|\psi\rangle - V|\psi\rangle\|_2$$

We know that the Triangle Inequality holds for the inside of the norm. If we now insert a net nothing inside:

$$\max\|U|\psi\rangle - V|\psi\rangle\|_2 = \max\|U|\psi\rangle - V|\psi\rangle + W|\psi\rangle - W|\psi\rangle\|_2$$

We can now group terms, and we see that we can upper bound the norm:

$$\max\|U|\psi\rangle - V|\psi\rangle\|_2 \leq \max\left[\|U|\psi\rangle - W|\psi\rangle\|_2 + \|W|\psi\rangle - V|\psi\rangle\|_2\right]$$

Now we use the fact that the max of a sum is the sum of the maxes:

$$\max\|U|\psi\rangle - V|\psi\rangle\|_2 \leq \max\|U|\psi\rangle - W|\psi\rangle\|_2 + \max\|W|\psi\rangle - V|\psi\rangle\|_2$$

We see that we have proven the Triangle Inquality, and thus our metric is indeed a distance. □

We also have another very useful property. Suppose we have two circuits, each with two gates, U_2U_1 and V_2V_1. Suppose we now want to find the distance between the two circuits. It is indeed true that

$$\boxed{E(U_2U_1, V_2V_1) \leq E(U_1, V_1) + E(U_2, V_2)}$$

Proof. We create an intermediate unitary between U_2U_1 and V_2V_1, such as U_2V_1. We now use the Triangle Inequality:

$$E(U_2U_1, V_2V_1) \leq E(U_2U_1, U_2V_1) + E(U_2V_1, V_2V_1)$$

Looking at the first term, we see that this is

$$\max\|U_2U_1|\psi\rangle - U_2V_1|\psi\rangle\|_2$$

We now use the property of Euclidean vectors that rotation does not change the magnitude, letting us factor:

$$= \max\|U_2(U_1|\psi\rangle - V_1|\psi\rangle)\|_2$$

We can now remove the U_2, since it is an overall rotation, so we see that we are left with $E(U_1, V_1)$.

Expanding the second term:

$$\max\|U_2V_1|\psi\rangle - V_2V_1|\psi\rangle\|_2$$
$$= \max\|(U_2 - V_2)V_1|\psi\rangle\|_2$$

where we have used the fact that the input vectors are the same, and the first unitary is always the same. Thus, we can think of this as a change of variables, and change this to a maximum varying $V_1|\psi\rangle$, which then leaves us with $E(U_2, V_2)$. Thus we have that

$$E(U_2U_1, V_2V_1) \leq E(U_1, V_1) + E(U_2, V_2)$$

□

Let us now think about how we use this. Suppose we have a general distance between two circuits:

$$E(U_nU_{n-1}\dots U_1, V_nV_{n-1}\dots V_1) \leq \sum_{i=1}^{n} E(U_i, V_i)$$

We see that we have created some upper bound on the distance between two circuits.

9.4.2 Solovay-Kitaev Theorem

The motivation for this theorem is that for some unitary U, we can approximate this using gates from some universal gate set. The question is, do the number of required gates from different universal gate sets different by a lot?

If we have two gate sets, and the first uses p gates for implementing U, and the second uses p' gates, we don't want p and p' to be very different from each other. The Solovay-Kitaev theorem says that the universal gate sets **do not** take very different amounts of gates.

Theorem 9.3. *With any fixed universal set of single-qubit gates that is closed under the inverse (If G is in the set, then so is $G^\dagger$), any single-qubit gate can be approximated with error at most ε, with the number of gates $\leq$ poly$\left(\log\left(\frac{1}{\varepsilon}\right)\right)$.*

We can apply this theorem. Consider using gate set 1, a circuit takes m gates, with error ε. Now suppose we wanted to use gate set 2. For each gate in gate set 1, up to error ε', we have a way to construct the gate in less than poly $\left(\log\frac{1}{\varepsilon'}\right)$. Suppose we want the error to be $\varepsilon' = \frac{\varepsilon}{m}$ (equally distributing the original error by the number of gates), which would mean that we would have a number of gates bounded by

$$m \times \text{poly}\left(\log\frac{m}{\varepsilon}\right)$$

In complexity theory, this overhead is considered small, no matter what gate set we use, this doesn't cause much of a difference.

10 Quantum Algorithms

10.1 Query Model

We want to first talk about a model known as the Query Model. Suppose we have a boolean function $f = \{0,1\}^n \to \{0,1\}$, but we don't have access to the description of the function.

We want to learn about the function only through oracle queries of f, which is when we send the oracle x as input of the function, and the oracle tells us the output of the function, $f(x)$. Essentially, we have the function in a box that we cannot access, but we can ask it what the output for a given input is.

We can generate an adaptive algorithm, which uses the input/output pairs from the oracle to get some output of the algorithm.

Let us look at this query model in the quantum setting. The first step is to construct an oracle that is reversible, since we need that for quantum computation. We have some n bit to 1 bit function, and we want to construct some circuit for U_f, which acts on $n+1$ qubits. We have n qubits of input $|x\rangle$, and one output qubit $|y\rangle$, and after passing this to the oracle, this keeps the inputs the same, and modifies $|y\rangle$ to be $|y \oplus f(x)\rangle$.

We can also extend this reversible oracle to any number of output bits. Suppose the function takes in n bits of input and m bits of output. We can make $|y\rangle$ m qubits, and then return the bitwise XOR, $|y \oplus f(x)\rangle$.

10.2 Deutsch Algorithm

Imagine we have a 1 bit to 1 bit function $f : \{0,1\} \to \{0,1\}$. This function is either a constant function, $f(0) = f(1)$, or a balanced function, $f(0) \neq f(1)$. How many queries do we need to find whether this is a constant function or a balanced function? Classically, we need 2 queries, $f(0)$ and $f(1)$.

If we do this using quantum computation, we will see that we only need to apply this oracle 1 time, and we will perfectly distinguish between constant and balanced functions. This is the first separation found between quantum and classical computation in the query model, although the separation is not that large.

10.2.1 Phase Kickback

This is a method in which we can turn our standard oracle $U_f : |x, y\rangle \to |x, y \oplus f(x)\rangle$, where we have stored the information into the computational basis, into another oracle, known as the phase oracle. We construct this via

$$|x\rangle \to (-1)^{f(x)} |x\rangle$$

We store the information in the phase.

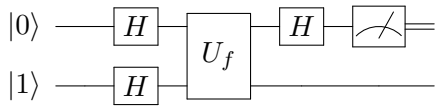
Suppose we choose $|y\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. In this case, acting with the standard oracle gets

$$U_f |x, -\rangle = U_f |x\rangle \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

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

Note that this is indeed consistent, as when we choose $f(x) = 0$, we see that this leaves the state unchanged. If we choose $f(x) = 1$, we see that we are left with $-|x\rangle |-\rangle$, and we have stored the information $(-1)^{f(x)} = -1$ in the phase, and the value of $|y\rangle$ has been left unchanged.

This is the phase oracle, and this property is known as phase kickback. We can use this phase oracle to create our first algorithm. We have the circuit



We can break this down into chunks. The first Hadamard transforms the first qubit's state to

$$|\psi_1\rangle = H |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

The unitary will then leave the state as

$$|\psi_2\rangle = \frac{1}{\sqrt{2}} \left((-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \right)$$

When we have a constant f, $f(0) = f(1)$:

$$|\psi_2\rangle = (-1)^{f(0)} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

The last H gate will then transform this to

$$|\psi_3\rangle \, (-1)^{f(0)} \, |0\rangle$$

When we have a balanced function f, $f(0) \neq f(1)$:

$$|\psi_2\rangle = (-1)^{f(0)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

And the Hadamard will leave us with

$$|\psi_3\rangle = (-1)^{f(0)} \, |1\rangle$$

We see that if the function is balanced, we will get 1, and if it is constant, we will get 0, with 100% certainty.

10.3 Deutsch-Josza Algorithm

This is a generalization of the previous algorithm, where we have some function

$$f : \{0,1\}^n \to \{0,1\}$$

Now we have functions that are neither balanced or constant. Recall that $f(x)$ is constant if $\forall x f(x) = f(0)$, and a function is balanced when the number of 0's as output is the same as the number of 1's.

Thinking classically, the best complexity that we can attain is $\mathcal{O}\left(\frac{2^n}{2} + 1\right)$. This is checking half of the inputs plus another 1 input to determine whether it is balanced or not.

If we have a randomized algorithm, where we sample n inputs $x_1, x_2, \ldots, x_m$, and we check each of the outputs. If those only contain 1 values, we will guess that it is constant, and if contains 2 values, we say that it is balanced. This will always work for the constant case, but will have a chance to be wrong for the balanced case. For example, suppose we sample k items, and they are all outputs of 0, when half of the outputs are 1. We can compute the failure probability when f is balanced:

$$P_f = 2 \times \frac{1}{2^k}$$

Where the 2 comes from sampling from either the outputs that are 0 or the outputs that are 1.

The difference between the Deutsch algorithm and the Deutsch Josza algorithm is that we have n bits of input, rather than just 1. We simply duplicate what we did to the first qubit in the Deutsch algorithm, and extend the unitary to an $n+1$ bit unitary gate.

To run through the state of the circuit, after the first set of H gates:

$$H^{\otimes n} \, |0\rangle^{\otimes n} = |+\rangle^{\otimes n}$$

Now replacing $|+\rangle$ with $\sum_{z\in\{0,1\}} \frac{1}{\sqrt{2}} |z\rangle$:

$$|+\rangle^{\otimes n} = \left(\sum_{x_1\in\{0,1\}} \frac{1}{\sqrt{2}} |x_1\rangle\right) \otimes \cdots \otimes \left(\sum_{x_n\in\{0,1\}} \frac{1}{\sqrt{2}} |x_n\rangle\right)$$

Now using the properties of sums:

$$= \sum_{x_1,x_2,x_3,\ldots x_n} \frac{1}{\sqrt{2^n}} |x_1\rangle |x_2\rangle |x_3\rangle \ldots |x_n\rangle$$

Now we can notice that this is an equal superposition, so this is the same as a length n bitstring:

$$= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

In the second step, we will pass through the phase oracle:

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

The third step is the second set of Hadamard gates:

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} H^{\otimes n} |x\rangle$$

We can use the fact that

$$H|x\rangle = \frac{1}{\sqrt{2}} \sum_{z\in\{0,1\}} (-1)^{z\cdot x} |z\rangle$$

Now using this to apply $H^{\otimes n}$:

$$H^{\otimes n}|x\rangle = (H|x_1\rangle) \otimes (H|x_2\rangle) \otimes \cdots \otimes (H|x_n\rangle)$$

$$= \left(\frac{1}{\sqrt{2}} \sum_{z_1\in\{0,1\}} (-1)^{z_1\cdot x_1} |z_1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}} \sum_{z_2\in\{0,1\}} (-1)^{z_2\cdot x_2} |z_2\rangle\right) \otimes \cdots \otimes \left(\frac{1}{\sqrt{2}} \sum_{z_n\in\{0,1\}} (-1)^{z_n\cdot x_n} |z_n\rangle\right)$$

Once again using the properties of sums:

$$= \sum_{z_1,z_2,\ldots,z_n} \frac{1}{\sqrt{2^n}} \cdot (-1)^{z_1\cdot x_1 + z_2\cdot x_2 + \ldots z_n\cdot x_n} |z_1 z_2 \ldots z_n\rangle$$

Now we once again note that $z = (z_1, z_2, \ldots z_n) \in \{0,1\}^n$, it is an n-bit string, and we finally have the closed form expression for the Hadamard on a bit string.

$$H^{\otimes n}|x\rangle = \sum_z \frac{1}{\sqrt{2^n}} (-1)^{z\cdot x} |z\rangle$$

Now applying this to the state of the system after the unitary:

$$\frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} \sum_z \frac{1}{\sqrt{2^n}} (-1)^{z\cdot x} |z\rangle = \frac{1}{2^n} \sum_z \left(\sum_x (-1)^{f(x)+z\cdot x}\right) |x\rangle$$

Now note that if the function is constant, the phase term on the state will be $\left|\sum_x (-1)^{f(x)}\right| = 2^n$. If it is constant, we only see the all 0 state.

If $f(x)$ is balanced, then $|\sum_x (-1)^{f(x)}| = 0$, since exactly half will have opposite phase to the other half. We will never see the $|00\ldots00\rangle$ in the measurement. This gets us our algorithm.

10.4 Simon's Algorithm

We have seen that in Deutsch's algorithm, we had a classical query complexity of 2, and a quantum complexity of 1, a small but tangible speedup. We then extended this to the Deutsch-Josza algorithm, which had a classical complexity of $\mathcal{O}(2^n)$, and a quantum query complexity of 1. Now we will see something that once again provides a speedup, but one where we can draw parallels to Shor's algorithm, as well as show a nontrivial quantum solution.

We begin with the problem of having a function $f : \{0,1\}^n \to \{0,1\}^n$. We have a promise, where there is some hidden string $s \in \{0,1\}^n$, and $f(x) = f(y)$ iff $x \oplus y \{0^n, s\}$, iff $y = x \oplus s$.

The task is to find s using queries to the oracle for f.

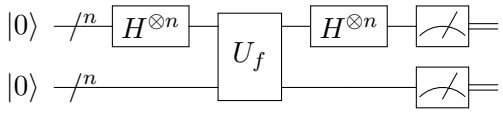
Let us look at classical algorithms for this. Intuitively, the best thing that we can think of is to test a bunch of inputs and hope we get repeated outputs. If we have no collision, we gain no information. If there is a collision, we know that we have x,y, such that $f(x) = f(y)$. Our promise tells us that $y = x \oplus s$. This is enough information to find s:

$$x \oplus y = x \oplus (x \oplus s) = (x \oplus x) \oplus s = 0^n \oplus s = s$$

The question is, how many inputs do we need to test? In the deterministic case, the complexity is $\mathcal{O}(2^n)$, since we need to query $\frac{2^n}{2} + 1$ inputs.

In the randomized case, we pick random inputs, and want to see how many we have to query until we reach a reasonable probability for collision. This is an example of a well-known case in probability, the birthday paradox, which tells us that it will take on the order of $\sqrt{2^n}$ for it to be **likely** that we get the collision. Note that this makes the probability of collision to be more than $\frac{3}{4}$. If we instead do this c times, on the order of $\mathcal{O}(c\sqrt{2^n})$, we would have a failure probability of $\frac{1}{4^c}$. We see that without much overhead, we can make this as small as we want.

We can now move to the quantum case. This algorithm requires some classical post-processing methods. The circuit takes the form



We can analyze what the circuit will do, step by step. The first set of Hadamard gates puts us into the state

The second step passes us into the oracle, which gives us

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$$

We now measure the output bits, and get the bitstring l. We have two cases. If $s = 0^n$, we will have that the input bits will be some input x_l, such that $f(x_l) = l$. In the second case, we have that $s \neq 0^n$, and thus the input bits are in the state $\frac{1}{\sqrt{2}}(|x_l\rangle + |x_l \oplus s\rangle)$.

We now do the Hadamard on the inputs. In the first case, when we apply the Hadamard to the state $|x_l\rangle$, we will have

$$\frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} (-1)^{x \cdot j \bmod 2} |j\rangle$$

This is an equal superposition, because these amplitudes can never cancel, we can measure any j. Since $s = 0^n$, no matter what we measure, say y, $y \oplus s = y \oplus 0^n = 0^n$.

In the second case, where $s \neq 0^n$:

$$\frac{1}{\sqrt{2}}(|x_l\rangle + |x_l \oplus s\rangle) \to \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} \left[(-1)^{x \cdot j \bmod 2} + (-1)^{(x \oplus s) \cdot j \bmod 2}\right] |j\rangle$$

Looking at the amplitudes:

$$(-1)^{x \cdot j \bmod 2} + (-1)^{x \cdot j \bmod 2 + s \cdot j \bmod 2}$$

Where we have used that the dot product is distributive over $\oplus$.

$$= (-1)^{x \cdot j \bmod 2}\left[1 + (-1)^{s \cdot j \bmod 2}\right]$$

The only time the amplitude of $|j\rangle$ is nonzero is when $s \cdot j \bmod 2 = 0$. On the other hand, if $s \cdot j \bmod 2 = 1$, we can compute that we will have 0 amplitude. If we measure some string y, we know that $s \cdot y \bmod 2 = 0$, because all strings that do not satisfy this case have zero amplitude.

Our claim is that any measurement y is such that $s = 0^n$, and y is a uniformly distributed string, or $s \neq 0^n$, and y is a string chosen uniformly randomly from strings such that $s \cdot y \bmod 2 = 0$. Note that this means that no matter what we measure, $s \cdot y \bmod 2 = 0$.

Let us now move on to our classical post-processing. Our quantum part gives us some uniform y such that $s \cdot y \bmod 2 = 0$. This does not uniquely determine a string s. However, we note that this is a linear equation of n variables:

$$s_1 y_1 + s_2 y_2 + \cdots + s_n y_n = 0$$

If our quantum routine returns $y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}, \ldots y^{(n)}$, all of which are linearly independent, then we have a set of n equations $s \cdot y^{(i)} \bmod 2 = 0$. This set of equations can be solved for s. How many times do we need to run the quantum routine? Suppose we run it n times. What is the probability that we have a correct set of equations?

For $y^{(1)}$, the first string we get, the probability of it being "good" is $1 - \frac{1}{2^n}$, where we have noted that we don't want to get 0^n. For the second round, our bad case is that $y^{(2)} = ay^{(1)}$, but since we have bitstrings, $a = 0, 1$. Thus we have 2 bad cases, giving us $P_2 = 1 - \frac{2}{2^n}$.

For $y^{(k)}$, the bad case is that $y^{(k)} = a_1 y^{(1)} + a_2 y^{(2)} + \ldots a_{k-1} y^{(k-1)}$. We see that there are 2^{k-1} bad cases, giving a probability of a good case being $P_k = 1 - \frac{2^{k-1}}{2^n}$. This is the general probability of getting a good new equation for the kth run.

Say we run n rounds. The probability of n good equations is

$$P_{\text{success}} = \prod_{m=1}^{n} \left(1 - \frac{2^{m-1}}{2^n}\right)$$

This is a well-known expression, and is approximately 0.288. This is greater than $\frac{1}{4}$. Just like in the birthday paradox, if we repeated this $4c$ times, we could make the failure probability

$$\left(1 - \frac{1}{4}\right)^{4c} \approx e^{-c}$$

We see that making $4cn$ quantum queries in total, we can obtain s with probability $\geq 1 - e^{-c}$. This is linear in the number of queries. Recall that the classical random method gave a solution of $\mathcal{O}(\sqrt{2^n})$, and quantum gave us a complexity of $\mathcal{O}(n)$.

10.5 Grover's Algorithm

We are given some function $f : \{0,1\}^n \to \{0,1\}$. This is sometimes known as an indicator, it indicates whether the entry has some property. For example, suppose the input represents an image, and the indicator returns whether or not the image has a person in it. Grover's algorithm is essentially searching through an unstructured database. We say unstructured because we don't assume anything about the structure of f, we do it all via queries. The goal is to find an input $x \in \{0,1\}^n$ such that $f(x) = 1$, if it exists. Otherwise, we want to output that there is no such x.

The classical solution for this problem is to query every single input until we either find x or find that x does not exist. However, we can do better than this. Suppose B is the set of solutions, $B = \{x \in \{0,1\}^n | f(x) = 1\}$. If $|B|$ is large, then the problem should be easy, we can pick randomly and hope we get something in B. The strategy is to sample $x \in \{0,1\}^n$, and we know the probability that this will return $f(x) = 1$:

$$P(\text{Solution}) = \frac{|B|}{2^n}$$

If we keep trying uniformly random samples, and checking whether they are solutions, what is the expected number of trials before we can observe an input that is a solution with probability 0.99?

Let I be the indicator for the ith trial. If this is 1, then the ith trial succeeded, and if it is 0, the trial failed. We have

$$I_1, I_2, \ldots, I_m$$

for a run of m trials. We want to calculate the expectation of the sum of the indicator outputs. If this sum is 0, then none of the trials succeeded. If the sum is nonzero, then we succeeded somewhere. Due to the linearity of the expectation, we have that

$$E[I_1 + I_2 + I_3 + \cdots + I_m] = mE[I_1] = m[1p + 0(1-p)] = mP$$

Where P is the success probability we computed earlier. We have that this expectation value must be at least .99:

$$mP \geq .99$$

From this, we can find the number of trials m that we will need, in terms of our solution set, and we find that the classical solution is $\Theta(2^n)$.

Now let us look at the quantum setting. We will find that the algorithm is $\Theta\left(\frac{1}{\sqrt{P}}\right)$, and recall that P can be as small as $\frac{1}{2^n}$. From this we have that $\mathcal{O}(\sqrt{2^n})$. We will later see that we can also derive that the solution is $\Omega(\sqrt{2^n})$, giving us a total $\Theta(\sqrt{2^n})$. We will find that we have a generic quadratic speedup over classical computation. Grover's algorithm is a quantum random walk, also known as a MCMC (Markov Chain Monte Carlo) algorithm. This can be generalized to a lot of different types of quantum algorithms. The limitation is that these speedups are only going to be polynomial when using Grover's algorithm. In order to get superpolynomial speedup, we need to exploit the structure of the problem, which Grover's algorithm ignores.

Let us recap on the setting of the problem. We have an indicator function $f : \{0,1\}^n \to \{0,1\}$. We have an oracle $U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$. Recall that we can also make a phase oracle via $U_f |x\rangle |-\rangle = (-1)^{f(x)} |x\rangle |-\rangle$.

Grover noted that the phase oracle cannot distinguish between different solutions, because it only knows that $f(x) = 1$, not anything about x itself. The whole search space is divided into two sets,

inputs that return 0 and inputs that return 1. The algorithm cannot distinguish any further than that. This allows us to analyze the whole algorithm in a 2 dimensional subspace. We have two sets:

$$A = \{x|f(x) = 0\} \quad B = \{x|f(x) = 1\}$$

We can now define a state

$$|x_A\rangle = \frac{1}{\sqrt{|A|}} \sum_{x \in A} |x\rangle$$

This is the normalized, equal superposition of all inputs that return 0. We can also generate $|x_B\rangle$:

$$|x_B\rangle = \frac{1}{\sqrt{|B|}} \sum_{x \in B} |x\rangle$$

Note that $|x_A\rangle$ and $|x_B\rangle$ are orthogonal to each other, since A and B are disjoint sets. From this, we see that x_A and x_B generate a 2 dimensional space. We can note the effect of the phase oracle on $|x_A\rangle$:

$$U_f |x_A\rangle |-\rangle = |x_A\rangle |-\rangle$$

and on $|x_B\rangle$:

$$U_f |x_B\rangle |-\rangle = (-1) |x_B\rangle |-\rangle$$

Suppose we plot two axes, one being $|x_A\rangle$ and the other being $|x_B\rangle$. Imagine we have some arbitrary state on this plot, $|\psi\rangle$:

$$|\psi\rangle = \cos\theta |x_A\rangle + \sin\theta |x_B\rangle$$

If we apply the phase oracle (dropping the $|-\rangle$ and calling the phase oracle O_f):

$$O_f |\psi\rangle = \cos\theta |x_A\rangle - \sin\theta |x_B\rangle = \cos(-\theta) |x_A\rangle + \sin(-\theta) |x_B\rangle$$

Geometrically, we see that applying the phase oracle reflects the vector with respect to $|x_A\rangle$, it rotates by θ the *opposite* direction. This is the first important tool that we will use.

Suppose we start with a state that is an equal superposition of all states:

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle = \sqrt{\frac{|A|}{2^n}} |x_A\rangle + \sqrt{\frac{|B|}{2^n}} |x_B\rangle = \cos\theta_0 |x_A\rangle + \sin\theta_0 |x_B\rangle$$

Our target for the algorithm is to find $|x_B\rangle$, a superposition of states that are solutions. We begin with the state $|\phi\rangle$ that is very close to $|x_A\rangle$. What we will do is a reflection across $|\phi\rangle$, to get a state closer to $|x_B\rangle$. Suppose we have some state $|\psi\rangle$, and we do a reflection across $|\phi\rangle$ (We have not defined this reflection yet, we are assuming that we can do it) called $R_{|\phi\rangle}$, we will end with a state closer to $|x_B\rangle$. Suppose $|\psi\rangle$ is defined by the angle φ. Then the angle between the reflected state and $|\phi\rangle$ will be $\varphi + 2\theta_0$. This is the second useful tool.

Now let us put the two tools together. Suppose we have some state $|\psi\rangle$. We do the first step, which is to use the phase oracle to reflect across the $|x_A\rangle$ axis. We then apply the $R_{|\phi\rangle}$ operation, and we end with some $|\psi_2\rangle$ state, which has angle $\varphi + 2\theta_0$. Essentially:

$$|\psi_0\rangle \to O_f \to |\psi_1\rangle \to R_{|\phi\rangle} \to |\psi_2\rangle$$

We have moved the state from something that is close to $|x_A\rangle$ to something that is closer to $|x_B\rangle$, by $2\theta_0$. We need to repeat this whole process until we move roughly $\frac{\pi}{2}$. If we start in the state

$\varphi_0 = \theta_0$, and each iteration increases the angle by $2\theta_0$, then the number of iterations that we will need is roughly:

$$(2m+1)\theta_0 \approx \frac{\pi}{2} \to m \approx \frac{\pi}{4\theta_0}$$

If we assume that $|B|$ is very small, then $\sin\theta_0$ is also small. This means that $\theta_0 \approx \sqrt{\frac{|B|}{2^n}}$. This inner portion is what we previously defined as P! This tells us that $\theta_0 \approx \sqrt{P}$. Now inserting this into m, we have that $m \approx \frac{\pi}{4\sqrt{P}}$. Let us walk through the algorithm and count the number of queries. For each step, we have 1 query to the phase oracle, and we will see that in order to do the reflection $R_{|\phi\rangle}$, we need no information about f, and thus we require no addition queries. We can explicitly define this reflection:

$$R_{|\phi\rangle} = |\phi\rangle\langle\phi| - \sum |\phi^\perp\rangle\langle\phi^\perp|$$

The first term keeps everything in the direction of $|\phi\rangle$ the same, and changes the sign of everything in the orthogonal direction. We use a sum in the orthogonal term because this is in a higher dimension. We can rewrite this by using the fact that $|\phi\rangle\langle\phi| + \sum |\phi^\perp\rangle\langle\phi^\perp| = I$. Inserting this, we have that

$$R_{|\phi\rangle} = 2|\phi\rangle\langle\phi| - I$$

Now rewriting this by using how we find $|\phi\rangle$:

$$R_{|\phi\rangle} = 2H^{\otimes n}|0\rangle\langle 0|H^{\otimes n} - I = H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n}$$

Let us cap this off with a general pseudocode look at Grover's Algorithm. We begin by initializing $|\phi_0\rangle$ via $H^{\otimes n}|0\rangle$. We then reflect the state using O_f, and then reflect the state using $R_{|\phi\rangle}$. We then repeat these two steps m times. The final state will have angle $(2m+1)\theta_0$ as the angle between the final state and the $|x_A\rangle$ axis. The complexity will be roughly $\frac{\pi}{4\sqrt{P}}$. This can be compared to the classical complexity of $\frac{1}{P}$.

Now note that in the case where B is empty, then this whole thing breaks down. However, if we assume that B is nonempty, we can run the algorithm and then check our measured output using the oracle.

Grover's algorithm does have some issues. Since we don't know p initially, we might overshoot the solution axis. To solve this, we can use an algorithm known as quantum counting, which tells us what P is, at the cost of $\mathcal{O}(\sqrt{2^n})$. This will let us prevent overshooting.

The other solution is an algorithm known as fixed-point Grover, which we will not cover, but essentially it guarantees that we will always get a rotation to a better state, meaning that overshooting is never a problem.

10.5.1 Lower Bounding Grover's Algorithm

Now we need to figure out what the lower bound for Grover's algorithm is. We guess that it is $\Omega(\sqrt{2^n})$, and it will turn out to be true.

One technique that we use classically for lower bounding is decision trees. Suppose we are given an array $A[n]$ of n elements, and we want to sort the array. We have an operation that we can perform, which is comparing $A[i]$ and $A[j]$. This can be thought of as being a sort of oracle. The question is, how many operations are necessary to sort the array $A[n]$?

Suppose we use mergesort, and we have an $\mathcal{O}(n \log n)$ sorting time. Now we want to prove that $\Omega(n \log n)$ as well. Decision trees are an abstraction of what the algorithm does. We begin with a root, which is the comparison between i_0 and j_0. This node has two children, the child where the operation returned that i_0 was larger, and the child where the operation returned that j_0 is larger. Now at each node, we do another operation, asking a different query. Each node will branch into two children. If we do this for a certain number of queries, the depth of the tree corresponds to the number of operations/queries. Each leaf node will correspond to an output of the algorithm. If our algorithm is correct, we expect the number of leaf nodes to be at least the number of possible correct outputs. In the sorting case, we have $n!$ inputs, so we $2^{n_{\text{operations}}} \geq n!$, which tells us that $n_{\text{operations}} \geq \log(n!) \approx \mathcal{O}(n \log n)$, where we have used the Stirling Approximation.

Suppose the problem is that we have an array $A[n]$, nd we want to find the maximum $A[i^*]$. Our operation is random access to an element in the array. The upper bound is that we query all n elements, $n_{\text{ops}} \leq n$. Intuitively, we think that this cannot be done in $n-1$ queries, but how do we turn this into a mathematical proof? We want to prove that the lower bound is $n_{\text{ops}} \geq n$. We can use something known as the Adversarial Method.

This method assumes that there exists a way to take an algorithm and find a bad input for the algorithm via interaction with the algorithm. Assume there is an algorithm that can find the maximum using $n-1$ operations. If we run this algorithm, and at the first step, we query some i_1, the adversary sets $A[i_1] = 1$. The algorithm will then query some i_2, the adversary sets $A[i_2] = 2$. We repeat this process, until the algorithm queries some i_{n-1}, and the adversary then sets $A[i_{n-1}] = n-1$. We note that the algorithm must either return the maximum of $\{A[i_1], A[i_2], \dots A[i_{n-1}]\}$, or $A[i_n]$. We also note that $i_n \notin \{i_1, i_2, \dots i_{n-1}\}$. If the algorithm chooses to return the maximum of the queried terms, the adversary then picks $A[i_n] = n$, which makes it the correct answer for the problem, making the algorithm wrong. On the other hand, if the algorithm choose i_n, the adversary can choose $A[i_n] = 0$, making it the wrong answer. Thus we have seen that we can construct an input for the problem that makes it impossible to solve the problem in $n-1$ queries.

We have see that to prove the lower bound for something, we have to first model the general algorithm. We then need to identify the hard instance or input.

We now need to model the general quantum algorithm with oracles. We have an oracle $O_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$. Let us first restrict ourselves to quantum circuits. This is a big assumption that we are making without any loss of generality. We can now model any circuit, by having 3 sections of inputs. We have n qubits of input, representing $x \in \{0,1\}^n$. We then have a single qubit representing $y \in \{0,1\}$. We then have some polynomial in n number of workspace qubits. We assume that we begin with the application of some unitary V_1 onto all the qubits. We then apply the oracle to $|x\rangle$ and $|y\rangle$, and then apply some other global unitary V_2. We then repeat this n times, noting that we are applying the oracle n times.

Now what is a hard instance for Grover's algorithm. First let us look at our input cases. the first case is that $\forall x\,, f(x) = 0$. In this case, we have that $O_f = I$.

Another case is that there exists a unique $\tilde{x}$ such that $f(\tilde{x}) = 1$, and$f(x) = 0$ for all $x \neq \tilde{x}$. In this case, all algorithms for this problem need to output $|\psi_{\tilde{x}}\rangle$. The oracle function is no longer trivial, there are 2^n different possible values for $\tilde{x}$, so we have 2^n different oracle functions.

Let us look at the difference between the final state for the case of the all 0 function ($|\psi\rangle$) and the

case with a single input being 1 ($|\psi_{\tilde{x}}\rangle$).

$$\sum_{\tilde{x}} \| |\psi\rangle - |\psi_{\tilde{x}}\rangle \| \geq \Omega(2^n)$$

Suppose have the initial state $|\psi^{(0)}\rangle$ for our algorithms. For each query, we have that in the first case, with $O_f = I$:

$$|\psi^{(i)}\rangle = (I \otimes I)V_i \, |\psi^{(i-1)}\rangle$$

And for the second case:

$$|\psi_{\tilde{x}}^{(i)}\rangle = (O_f \otimes I)V_i \, |\psi_{\tilde{x}}^{(i-1)}\rangle$$

Our goal is to bound $\| |\psi^{(i)}\rangle - |\psi_{\tilde{x}}^{(i)}\rangle \|$ for $\tilde{x}$. We claim that this is $\leq \sum_{i=1}^{T} 2\sqrt{\Pr[i\text{th query is } \tilde{x}]}$. Putting things together, we have

$$c \cdot 2^n \leq \sum_{\tilde{x}} \| |\psi^{(i)}\rangle - |\psi_{\tilde{x}}^{(i)}\rangle \|$$

Now substituting in:

$$c \cdot 2^n \leq \sum_{\tilde{x}} \sum_{i=1}^{T} 2\sqrt{\Pr[i\text{th query is } \tilde{x}]}$$

Now using Cauchy-Schwarz:

$$= 2 \sum_{i=1}^{T} \sum_{\tilde{x}} \sqrt{\Pr[i\text{th query is } \tilde{x}]}$$

This inner sum is $\leq \sqrt{2^n}$, which gives us that

$$= 2 \sum_{i=1}^{T} \sqrt{2^n} = 2T\sqrt{2^n}$$

Now putting the left side back:

$$2T\sqrt{2^n} \geq c \cdot 2^n \to T \geq \Omega(\sqrt{2^n})$$

10.6 Shor's Algorithm

We begin by talking about a subproblem that was initially developed along with Shor's algorithm, but is used widely elsewhere, and so it is it's own algorithm, quantum phase estimation.

10.6.1 Quantum Phase Estimation

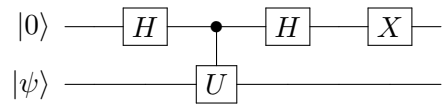
Quantum phase estimation has a fully quantum problem statement. We are given a controlled unitary U, and we are given an eigenstate of the unitary $|\psi\rangle$, with some eigenvalue $e^{i2\pi\theta}$, for some $\theta \in [0, 1]$. This comes from the definition of the eigenvalues of a unitary matrix. The output of the problem is an approximation of θ, $\tilde{\theta} \in [0, 1]$. We represent the real number between 0 and 1 in binary:

$$\tilde{\theta} = 0.\theta_1\theta_2\theta_3 \ldots$$

Where θ_1 carries the weight of $\frac{1}{2}$, θ_2 carries the weight $\frac{1}{4}$, etc.

We hope that $|\tilde{\theta} - \theta| < \varepsilon$, for small ε, we want them to be close to each other.

Let us first look at the case where $\theta = 0.\theta_1$, where θ is either $e^{i2\pi\times 0}$ or $e^{i2\pi\times 1/2}$, giving us an eigenvalue of either 1 or -1. In the case where the eigenvalue is 1, we have that $U|\psi\rangle = |\psi\rangle$, and in the case where the eigenvalue is -1, we have $U|\psi\rangle = -|\psi\rangle$. This is something that we have seen before (on the homework), with a slight modification:



Now we want to generalize this to $\theta = 0.\theta_1\theta_2\ldots$:

Note that we need two new things, a multi-qubit controlled unitary, and a QFT gate, the Quantum Fourier Transform. Let us first generate the multi-qubit controlled unitary. We have our start state:

$$|x\rangle|\psi\rangle$$

Where $|x\rangle \in \{0,1\}^n$, and we can think of $|x\rangle$ as representing a binary representation of an integer. We can think of this as applying the unitary x times:

$$|x\rangle|\psi\rangle \to |x\rangle U^x|\psi\rangle$$

Constructing this is not trivial, and it turns out that the worst case cost of this gate is exponential in n, the number of control bits. This is bad, but this is the worst case. The unitary inside Shor's algorithm is special. We use the unitary

$$U|x\rangle = |ax \bmod p\rangle$$

Applying this multiple times:

$$U^k|x\rangle = |a^k x \bmod p\rangle$$

Now can we compute a^k in poly-log(k) multiplications? Instead of just multiplying a k times, we can do something more clever. Suppose $k = 2^m$. We can then do this in m multiplications, since we can use the powers of a rather than just a. In this case, we can do the multiplication in poly(n) time.

We begin our circuit in the state $|0\rangle^{\otimes n}|\psi\rangle$. We then use the Hadamards:

$$\left(\frac{1}{\sqrt{2^n}\sum_{k\in\{0,1\}^n}}|k\rangle\right)\otimes|\psi\rangle$$

We then apply the unitary:

$$\frac{1}{\sqrt{2^n}}\sum_k |k\rangle\otimes U^k|\psi\rangle$$

Now using the definition of the application of the unitary to our state $|\psi\rangle$:

$$\frac{1}{\sqrt{2^n}}\sum_k e^{i2\pi\theta k}|k\rangle\otimes|\psi\rangle$$

We can say that $\theta = \frac{j}{2^n}$, where j is some integer from 0 to $2^n - 1$. We essentially are saying that we can represent our θ as some integer out of the 2^n possibilities. We can now call this first section of the state $|\phi_j\rangle$:

$$|\phi_j\rangle = \frac{1}{\sqrt{2^n}}\sum_k e^{i2\pi\theta k}|k\rangle$$

This can be written as

$$|\phi_j\rangle = \frac{1}{\sqrt{2^n}} \sum_k e^{i\frac{2\pi}{2^n}jk} |k\rangle$$

Now let $\omega = e^{i\frac{2\pi}{2^n}}$:

$$|\phi_j\rangle = \frac{1}{\sqrt{2^n}} \sum_k \omega^{jk} |k\rangle$$

We now claim that the Quantum Fourier Transform has the action

$$QFT_{2^n} |j\rangle = |\phi_j\rangle$$

And thus

$$QFT_{2^n}^\dagger |\phi_j\rangle = |j\rangle$$

Now we realize that we can use this to go from the state that we have $|\phi_j\rangle$, and get $|j\rangle$. Now using our definition of j, we can find θ from this, since $\theta = \frac{j}{2^n}$. Let us look more at the QFT. We first claim that the set of states $\{|\phi_j\rangle\}_{j=0}^{2^n-1}$ forms an orthonormal basis.

Proof. We have that

$$|\phi_i\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (\omega^{ik})^* \langle k|$$

Now taking the inner product with $|\phi_j\rangle$:

$$\langle \phi_i|\phi_j\rangle = \frac{1}{2^n} \sum_{k=0}^{2^n-1} \omega^{jk}(\omega^{ik})^* \langle k|k\rangle$$

$$= \frac{1}{2^n} \sum_{k=0}^{2^n-1} \omega^{(j-i)k}$$

We can split this into cases. The first case is where $i = j$. We then have that the ω term is 1, and we have that

$$\langle \phi_i|\phi_j\rangle = \frac{1}{2^n} \sum_{k=0}^{2^n-1} 1 = 1$$

Now if $i \neq j$, we have that $\Delta = j - 1$, and $\omega^\Delta \neq 1$:

$$(\omega^\Delta)^{2^n} = \left((e^{i\frac{2\pi}{2^n}})^\Delta\right)^{2^n} = e^{i2\pi\Delta} = 1$$

Now we have that the inner product is

$$\langle \phi_i|\phi_j\rangle = \frac{1}{2^n} \sum_{k=0}^{2^n-1} (\omega^\Delta)^k = \frac{1}{2^n} \frac{(\omega^\Delta)^{2^n} - 1}{\omega^\Delta - 1} = 1$$

Thus we have that this is indeed an orthnonormal basis. □

We see that the space generated by $|\phi\rangle$ is the Fourier space that the transform maps to and from.

We can write out the QFT_{2^n} in Dirac notation using this:

$$QFT_{2^n} = \sum_j |\phi_j\rangle \langle phi_j|$$

We also note that when $n = 1$, $QFT_2 = H$, and $\omega = e^{i\frac{2\pi}{2}} = -1$, so $|\phi_0\rangle = |+\rangle$, and $|\phi_1\rangle = |-\rangle$.

We can write out the matrix form of the QFT:

$$F = \frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{2^n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(2^n-1)} \\ 1 & \omega^3 & \omega^6 & \dots & \omega^{3(2^n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{2^n-1} & \dots & \dots & (\omega^{2^n-1})^{2^n-1} \end{pmatrix}$$

10.6.2 Public-key crypto-systems and RSA

The traditional crypto-system uses private keys, which is where Alice and Bob share a private key, only visible to those two, and then they send messages encrpyted with that private key, so only they will be able to decrypt it. Alice has some message m, and encodes it using the key, $\text{Enc}(m, s)$, and then Bob can decrypt it $\text{Dec}(\text{Enc}(m, s), s) = m$.

On the other hand, public-key or asymmetric crypto-systems are when Alice has a public key as well as a secret key. Bob can send a message to Alice by encoding it using the public key, $\text{Enc}(m, pk)$, and Alice uses the secret key to decrypt it, $\text{Dec}(\text{Enc}(m, pk), sk)$.

RSA is a widely used scheme to do this public-key encryption. The idea is that $pk = n, r$, where $n = pq$, and p and q are large prime numbers. It is hard to go from n to p and q, but easy to go from p and q to n. The secret key is $sk = (p-1)(q-1)$. When we do the encryption:

$$\text{Enc}(m, r) = m^r \bmod n$$

And the decryption:

$$\text{Dec}(m^r, s) = m^{rs} \bmod n = m \bmod n$$

Thus cracking RSA requires a fast method for factoring n, which Shor's algorithm can be shown to do much faster than classical algorithms.

10.6.3 Solving Factorization with Quantum Computing

When we say that we want to factorize a large integer N, the input size of the problem is $\log N$. Most classical algorithms are based on Field Sieve algorithms, which pools together candidates and then removes the worst ones, and the best that these can do is $2^{\sqrt{\log N}}$, which is close to exponential but not quite there. We can do slightly better using a Number Field Sieve, which is of the order $2^{(\log N)^{1/3}}$. We see that this is super-polynomial with respect to the input size, which is why we say its a hard problem classically. Shor's Algorithm on the other hand, will have runtime that is poly-log(N), which is polynomially with respect to the input size, and thus is efficient.

Shor's algorithm does not try to multiply numbers to get N. Instead, we connect the problem of factorizing a number to solving a quadratic congruence relation:

$$x^2 - 1 \cong 0 \pmod{N}$$

Suppose $N = pq$. We know that $x^2 - 1 = (x+1)(x-1)$. In the case where pq fits strictly within one of those terms, then we have no info. However, if p and q are separated in those terms, we can return the greatest common divisor:

$$\gcd(x+1, N) \quad \text{or} \quad \gcd(x-1, N)$$

These will return a nontrivial factor of N. It can be shown that this case occurs often enough for us to leverage in our algorithm.

We want to find something that will give us a solution for x. Shor's Algorithm solves what is known as the order finding problem. We are given a and N, where $2 \leq a < N$, and $\gcd(a, N) = 1$, they are coprime. We want to find the order, the smallest r such that $a^r \cong 1 \pmod{N}$.

We want to leverage the fact that if r is even, we have the solution to the quadratic problem, since $a^{\frac{r}{2}}$ will be a solution of $x^2 \cong 1$. Thus we want a reduction of the problem to the order finding problem.

Here is an outline of the classical part of Shor's Algorithm. We have input N.

We first pick a random $a \in [2, N-1]$. We then compute the gcd of a and N. If it is 1, we have that a and N are coprime. If it is not 1, we are done, we have found a nontrivial factor of N.

We then use the quantum subroutine to find the order r of a and N.

If the order is even, we compute $\gcd(a^{\frac{r}{2}} - 1, N)$ or $\gcd(a^{\frac{r}{2}} + 1, N)$. These are likely to be a nontrivial factor of N, with probability $\geq \frac{1}{\text{poly-log}(N)}$.

10.6.4 Quantum Order Finding

We want to design a quantum method for finding the order r, given a and N such that they are coprime. We want to find the smallest r such that $a^r \cong 1$. We first do some number theory, and begin with the nonnegative integers up to N:

$$\mathbb{Z}_N = \{0, ..N-1\}$$

And then we define the coprimes:

$$\mathbb{Z}_N^* = \{x | x \in \mathbb{Z}_N | \gcd(x, N) = 1\}$$

This actually forms a group, with operation multiplication modulo N. In fact, it can be shown that $a^{\varphi(N)} \cong 1 \pmod{N}$, where $\varphi(N) = |\mathbb{Z}_N^*|$, for all $a \in \mathbb{Z}_N^*$. This takes a while, and in fact classically we don't know any good way of doing this.

We will see that using quantum phase estimation on a specific unitary will recover the order via an eigenvalue. The unitary that we use is $U : M_a |x\rangle = |ax \bmod N\rangle$. We then consider a controlled version, $C-U : C - M_a |k\rangle |\phi\rangle = |k\rangle Ma^k |\phi\rangle$. We know that $M_a^k |x\rangle = |a^k x \bmod N\rangle$. We know that this can be implemented in poly-log(k).

We also need to find the eigenvectors of M_a. Suppose we have the state

$$|\psi_0\rangle = \frac{1}{\sqrt{r}}(|1\rangle + |a^1\rangle + |a^2\rangle + \cdots + |a^{r-1}\rangle)$$

Applying M_a:

$$M_a |\psi_0\rangle = |\psi_0\rangle$$

We see that $|\psi_0\rangle$ is an eigenvector of M_a, with eigenvalue 1. Expanding this, we can construct eigenvectors via:

$$|\psi_1\rangle = \frac{1}{\sqrt{r}}(|1\rangle + \omega_r^{-1} |a\rangle + \omega_r^{-2} |a^2\rangle + \cdots + \omega_r^{-(r-1)} |a^{r-1}\rangle)$$

Where $\omega_r = e^{i\frac{2\pi}{r}}$. When we apply M_a to $|\psi_1\rangle$, we find

$$M_a |\psi_1\rangle = \frac{1}{\sqrt{r}}(|a\rangle + \omega_r^{-1} |a^2\rangle + \omega_r^{-2} |a^3\rangle + \cdots + \omega_r^{-r-1} |1\rangle) = \omega_r |\psi_1\rangle$$

Thus we see that we have an eigenvector of M_a, with eigenvalue $e^{i\frac{2\pi}{r}}$. If we can construct this state, and then use phase estimation, we will output the phase, which is $\frac{1}{r}$. We can generate eigenvector $|\phi_j\rangle$. This has eigenvalue $e^{i\frac{2\pi j}{r}}$. Using phase estimation, we will return $\frac{j}{r}$. Since we know j, we can find r. The issue is now how we generate these eigenvectors.

The observation that helps out here is that we know that

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |\psi_j\rangle$$

We cannot prepare the eigenvectors, but we can generate a superposition of all the eigenvectors. Now applying QPE:

$$QPE\,|1\rangle = \sum_j QPE\,|\psi_j\rangle = \sum_j \frac{1}{\sqrt{r}} |\psi_j\rangle \otimes \left|\frac{j}{r}\right\rangle$$

The output will be an approximation of $\frac{j}{r}$ for a uniformly picked j from $[0, N-1]$. This is the quantum part of quantum order finding. One possible issue is that we get an approximation, not the exact value.

11 Mixed States

Suppose we have an ensemble of states, each with a probability, $\{|\phi_i\rangle, p_i\}$. State $|\psi_i\rangle$ occurs in the measurement with probability p_i. We can instead have one single object to represent an ensemble of quantum states. This is called a density operator, and using only the density operator, we can recover all the physics. Mathematically, a density operator is a positive-semidefinite matrix with trace equal to 1. Recall that positive-semidefinite means that $A = A^\dagger$, which tells us that the eigenvalues of A are real, and $A \geq 0$, which means that either all the eigenvalues are ≥ 0, or equivalently, $\forall |\phi\rangle, \langle\phi|A|\phi\rangle \geq 0$. Also recall that the trace of a matrix is the sum of the eigenvalues of A.

Suppose we have a pure state, an ensemble with only one quantum state, $|\psi\rangle \in \mathbb{C}^d$. How can we connect a vector to our density operator matrix? We just use the outer product of the vector with itself:

$$|\psi\rangle \langle\psi|$$

If we take the trace of this, we get 1. We also need to verify that the outer product is positive semidefinite. We can do this using the first definition, showing that the eigenvalues are all non-negative. We have that $|\psi\rangle$ is an eigenvector with eigenvalue 1, and we have eigenvectors $|\phi\rangle$ that are orthogonal to $|\psi\rangle$, and we have that the eigenvalues are 0. Using the second definition, $\langle\phi|A|\phi\rangle \geq 0$ for all ϕ:

$$\langle\phi|\psi\rangle \langle\psi|\phi\rangle \geq 0 \rightarrow |\langle\psi|\phi\rangle|^2 \geq 0$$

This is always true, by definition. Thus, $|\psi\rangle \langle\psi|$ is the density operator for a pure state.

Now let us move on to a mixed state, $\{|\psi_i\rangle, p_i\}$. We will find that the density operator ρ is given by

$$\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i|$$

Now we need to show that this is a well-defined density operator. We need to verify that the trace is 1.

$$Tr(\rho) = Tr\left(\sum_i p_i |\psi_i\rangle \langle\psi_i|\right)$$

Now leveraging the fact that the trace operation is a linear operation:

$$Tr\left(\sum_i p_i |\psi_i\rangle \langle\psi_i|\right) = \sum_i p_i \, Tr(|\psi_i\rangle \langle\psi_i|) = \sum_i p_i = 1$$

Now we need to prove that $\rho \geq 0$. Using definition 2:

$$\forall |\phi\rangle \quad \langle\phi|\rho|\phi\rangle \geq 0$$

$$\langle\phi|\rho|\phi\rangle = \langle\phi| \sum_i p_i |\psi_i\rangle \langle\psi_i| |\phi\rangle = \sum_i p_i \langle\phi|\psi_i\rangle \langle\psi_i|\phi\rangle = \sum_i p_i |c_i|^2 \geq 0$$

If we have a density operator ρ, and the rank is 1, we are in a pure state. If the rank is greater than 1, we have that $\rho = \sum_i \lambda_i |\psi_i\rangle \langle\psi_i|$, and we have a mixed state. Suppose we have the mixed state $\{(|0\rangle, \frac{1}{2}), (|1\rangle, \frac{1}{2})\}$:

$$\rho = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{1}{2}I$$

Note that density operators are not unique, if we find the density operator of the mixed state $\{(|+\rangle, \frac{1}{2}), (|-\rangle, \frac{1}{2})\}$, we have that $\rho = \frac{1}{2}I$, the same as the previous state. Thus a given density operator can correspond to many different ensembles.

If we look at the Bloch sphere, and we have a point $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$, which corresponds to a point (θ, ϕ) on the sphere. For a 1 qubit state, we can get the density operator via

$$\rho = \frac{1}{2}(I + \alpha_x X + \alpha_y Y + \alpha_z Z)$$

where we have some $(\alpha_x, \alpha_y, \alpha_z) \in \mathbb{R}^3$. Note that, since the Pauli matrices are traceless, the trace of ρ is just the trace of the $\frac{1}{2}I$ matrix, which is 1. If $\alpha_x^2 + \alpha_y^2 + \alpha_z^2 = 1$, we have a pure state. If the value is between 0 and 1, we have a mixed state. If we have that $(\alpha_x, \alpha_y, \alpha_z) = (0, 0, 0)$, this is known as a totally mixed state.

11.1 Composite Systems and Partial Trace

Suppose we have a product state $|\psi_A\rangle \otimes |\psi_B\rangle$. We know that $\rho_A = |\psi_A\rangle \langle\psi_A|$ and $\rho_B = |\psi_B\rangle \langle\psi_B|$. It turns out that $\rho_{A\otimes B} = \rho_A \otimes \rho_B$. This is the simple case, where we have a product state.

Suppose instead we have a separable state. We have the state $|\psi_A^i\rangle \otimes |\psi_B^j\rangle$, with probability P_{AB}^{ij}. We have that $\rho = \sum_{i,j} P_{AB}^{ij} \rho_A^i \otimes \rho_B^j$. We can write this in the form $\rho = \rho_A \otimes \rho_B$ only if $P_{AB}^{ij} = P_A^i \times P_B^j$, i.e, the probabilities are independent:

$$\rho = \sum_i P_A^i \rho_A^i \otimes \sum_j P_B^j \rho_B^j$$

In the product state, we have no correlation between A and B, and in the separable state, we have classical correlation between A and B. Anything that cannot be written as one of these two is considered an entangled state, or quantum correlated state.

We can define a partial trace. Suppose we have a system $M \in \mathbb{C}^{d_A \times d_A} \otimes \mathbb{C}_B^d \times d_B$. The trace over A is define to be

$$Tr_A(M) = \sum_i (\langle i_A| \otimes I_B) M (|i_A\rangle \otimes I_B)$$

This gets an operation on just the B space, we have "traced out" A.

$$Tr_B(M) = \sum_i (I_A \otimes \langle i_B|) M (I_A \otimes |i_B\rangle)$$

Let us do a very simple example. Suppose we have $|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B)$. Let us compute the partial trace over B. We first need to compute

$$I_A \otimes \langle 0_B|\phi\rangle \propto I_A \otimes \langle 0|_B |0\rangle_A \otimes |0\rangle_B + I_A \otimes \langle 0|_B |1\rangle_A \otimes |1\rangle_B = I_A |0\rangle_A \otimes \langle 0|0\rangle_B + I_A |1\rangle_A \otimes \langle 0|1\rangle_B = |0\rangle_A$$

Putting the coefficients back, we have

$$\frac{1}{\sqrt{2}} |0\rangle_A$$

We also need to compute

$$I_A \otimes \langle 1|_B |\phi\rangle = \frac{1}{\sqrt{2}} |1\rangle_A$$

Now we know that

$$Tr_B(|\phi\rangle \langle\phi|) = \frac{1}{\sqrt{2}} |0\rangle_A \frac{1}{\sqrt{2}} \langle 0|_A + \frac{1}{\sqrt{2}} |1\rangle_A \frac{1}{\sqrt{2}} \langle 1|_A = \frac{1}{2} I$$

Likewise, if we go through the calculation for $Tr_A(|\phi\rangle) = \frac{1}{2} I_B$. If we only look at A or only look at B, we see only classical effects, no quantum effects. However, when we put them together, we get entanglement.

11.2 Determining Entanglement

Suppose we are given a bipartite $|\psi\rangle$, $|\psi\rangle \in \mathbb{C}^A \otimes \mathbb{C}^B$. Can we determine whether $|\psi\rangle$ is entangled or not? This problem is difficult to see directly. Let us do this in multiple approaches.

The first approach is by contradiction. We assume that $|\psi\rangle$ is a product state, which means that we can write it as follows:

$$|\psi\rangle = (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle) = \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle$$

Comparing this against a state such as $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, we see that this gets 4 equations with 4 unknowns, and in this case we see that there are no values of our 4 variables that will generate this state. This tells us that our assumption of the state is false, and thus the state is entangled, which confirms what we know about the state.

The first approach is very complicated for larger problems. Let us look at the second approach, using density operators. Suppose $|\psi\rangle$ is a product state, $|\psi\rangle = |\psi\rangle_A \otimes |\psi\rangle_B$. Taking the partial trace of the density operator:

$$Tr_A(|\psi\rangle \langle\psi|) = |\psi\rangle_B \langle\psi|_B$$
$$Tr_B(|\psi\rangle \langle\psi|) = |\psi\rangle_A \langle\psi|_A$$

We note that these are both of rank 1. If we take a state and compute its partial traces, and we get something that is not of rank 1, we know that we do not have a product state. For example, for the EPR state, we have that $Tr_A(EPR) = Tr_B(EPR) = \frac{1}{2}I$, and the rank of this is larger than 1, meaning that they are indeed not product states. It turns out that finding whether a mixed state is entangled or not is an NP-complete problem.

11.3 Schmidt Decomposition

If we are given a bipartite state $|\psi\rangle$, we can always decompose it as follows:

$$|\psi\rangle = \sum_{i=1}^{d} \sqrt{\lambda_i} |U_i\rangle_A \otimes |V_i\rangle_B$$

where $d = \min(d_A, d_B)$, and $\lambda_i \geq 0$, with $\sum_i \lambda_i = 1$. $\{|U_i\rangle_A\}_{i=1}^d$ is an orthonormal basis for A, and likewise $\{|V_i\rangle_B\}_{i=1}^d$ is an orthonormal basis for B.

The Schmidt decomposition gives us a lot of nice properties when writing out a bipartite state.

11.4 Working with Density Operators

Suppose we take a state $|\psi\rangle$ and apply some unitary U to it:

$$|\psi\rangle \to U |\psi\rangle$$

We can look at how the density operator changes:

$$|\psi\rangle \langle\psi| \to U |\psi\rangle \langle\psi| U^\dagger$$

Now how do we deal with measurement? In the vector formalism, we can write out the state

$$|\psi\rangle \sum \alpha_i |\phi_i\rangle$$

and the probability of measure $|\phi_i\rangle$ is $|\alpha_i|^2$, and the post-measurement state is $|\phi_i\rangle$.

Using density operators, we have that

$$\rho = |\psi\rangle \langle\psi| = \sum_{i,j} \alpha_i \alpha^* j \, |\phi_i\rangle \langle\phi_j|$$

If we look at the matrix of ρ, we see that the diagonal elements are just the probabilities, which makes sense, since we know that $Tr(\rho) = 1$, the probabilities should sum up to 1. Mathematically:

$$P_0 = Tr(\rho \, |0\rangle \langle 0|)$$

And more generally:

$$P_k = Tr(\rho \, |\phi_k\rangle \langle\phi_k|)$$

The post-measurement state will be $|\phi_k\rangle \langle\phi_k|$:

$$|\phi_k\rangle \langle\phi_k| = \frac{|\phi_k\rangle \langle\phi_k| \, \rho \, |\phi_k\rangle \langle\phi_k|}{Tr(|\phi_k\rangle \langle\phi_k| \, \rho \, |\phi_k\rangle \langle\phi_k|)}$$

12 Quantum Error Correction

12.1 Classical Error Correction

Let us begin with classical error correction. The motivation for this was hardware issues, back when the error rates in the computer hardware was higher. However, we still have to worry about classical error rate when it comes to communication, where data corruption or packet loss is still an issue.

The first think to start with is an error model. Error correction is only possible in certain error models. For example, if we totally destroy the hardware, there is intuitively no way to recover the data. The key idea behind error correction techniques is that we want to add redundancy, such as more copies of the information.

In the context of communication, we have a sender and a receiver. The sender has some plain message to send over some channel to the receiver. We have that the channel is noisy, and the receiver wants to receive the plain message. If we naively just send over the message, we will not recover the plain text perfectly due to the noise. We instead use an encoder to add redundancy, and then send that encoded message over the channel, and then have the receiver decode that to recover the original string.

Let us use a specific noisy channel, known as a binary symmetry channel. We send over messages consisting of 0s and 1s, and the binary symmetry channel gives a probability p to turn a 0 in the message into a 1. This means that we have a $1 - p$ probability to keep the original bit. The symmetry is there because the probability of flipping a 0 to a 1 is the same as flipping a 1 to a 0.

Suppose our plain message is a single bit, $x \in \{0, 1\}$. The first scheme is to send it directly over, obtaining $\tilde{x}$. The probability of $x = \tilde{x}$ is $1 - p$, by the definition of the channel's noise. The probability of error will be p.

The second scheme is known as repetition code, which is where we map $x \in \{0, 1\}$ to a code $x_1 x_2 x_3$, where $x_1 = x_2 = x_3 = x$. We then send this through the channel. We will receive $\tilde{x_1}\tilde{x_2}\tilde{x_3}$, and we want to decode this into x. A reasonable decoder would be to just take the majority, and consider that to be x. Let's analyze this. We care about the number of flips, not where the flips happen. We

know that the number of flips being 0 occurs with probability $(1-p)^3$. The probability of 1 flipping is $\binom{3}{1}p(1-p)^2$. The probability of 2 flipping is $\binom{3}{2}p^2(1-p)$. The probability of all 3 flipping is p^3. If the number of flips is 0 or 1, the majority decoder will work, and if the number of flips is 2 or 3, it will fail. This means that the probability of error is $\mathcal{O}(p^2)$ (not p^3 because $0 < p < 1$).

Suppose we want to know where the error was. If we look at the input 0 and we at most 1 flip, there are 4 possibilities, $100, 010, 001, 000$, and we can do something similar for 1. Note that these are all disjoint. Is there a way for us to tell which bits have been or have not been flipped? To do this, we can compute the values of $x_1 \oplus x_2$ and $x_2 \oplus x_3$. If we go through the case work, we see that each of the 4 cases for the message leads to a unique pair of values in the two computations:

$$x_1x_2x_3 = 100 \to x_1 \oplus x_2 = 1,\ x_2 \oplus x_3 = 0$$

$$x_1x_2x_3 = 010 \to x_1 \oplus x_2 = 1,\ x_2 \oplus x_3 = 1$$

$$x_1x_2x_3 = 001 \to x_1 \oplus x_2 = 0,\ x_2 \oplus x_3 = 1$$

$$x_1x_2x_3 = 000 \to x_1 \oplus x_2 = 0,\ x_2 \oplus x_3 = 0$$

Thus this allows us to find out where the flips are.

12.2 Quantum Error Correction

We first have to come up with an error model for quantum computation. We have a system (in this case a single qubit), and an environment, and then we have some unitary that is the interaction between the system and the environment:

$$U_{SE} \left|\psi\right\rangle_S \left|0\right\rangle_E$$

Where all we know is $\left|\psi\right\rangle_S$, we have no information on the unitary. However, we know that we end up with something of the form

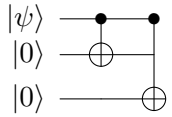
$$\alpha_I I \left|\psi\right\rangle_S \otimes \left|\phi_I\right\rangle_E + \alpha_X X \left|\psi\right\rangle_S \otimes \left|\phi_X\right\rangle_E + \alpha_Z Z \left|\psi\right\rangle_S \otimes \left|\phi_Z\right\rangle_E + \alpha_{XZ} XZ \left|\psi\right\rangle_S \otimes \left|\phi_{XZ}\right\rangle_E$$

The first term is the case where we have no error. The second term is the case where we have bit flip error, the third is where we have a phase error, and the final case is where we have both. Thus our model allows us to only consider X and Z errors.

Let us begin with bit flip errors. We can try the repetition code once more, by mapping

$$\left|0\right\rangle \to \left|000\right\rangle \quad \left|1\right\rangle \to \left|111\right\rangle$$

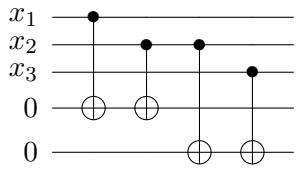
This can be done via two CNOT gates quite easily:



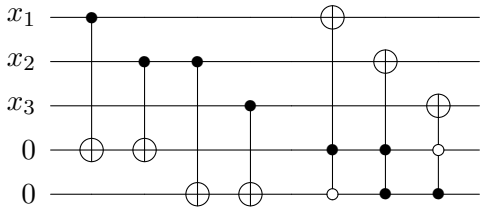
This will map $\left|\psi\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$ to $\alpha\left|0\right\rangle_L + \beta\left|1\right\rangle_L$, where the L denotes "logical" representation, $\left|0\right\rangle_L = \left|000\right\rangle$, and similarly for $\left|1\right\rangle_L$. Now we pass this through the channel, and we will operate one of the following errors (for a single bit flip):

$$I,\ X_1 \otimes I_2 \otimes I_3,\ I_1 \otimes X_2 \otimes I_3,\ I_1 \otimes I_2 \otimes X_3$$

We can once again use the "syndrome" computations, $x_1 \otimes x_2$ and $x_2 \otimes x_3$.



Now we can do the corrections based on the values in the two ancilla, which are the computed syndrome values:



Thus the control chain is that we encode, send over the channel, use error detection, and then error correction. Then we use the decoding.

Now we have to deal with Z errors. Z errors are phase flip errors, $Z|0\rangle = 0$, $Z|1\rangle = -|1\rangle$. Let us look at our encoding. If we state with $|+\rangle$, which gets encoded to $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$, and after the error we are left with $\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)$, which decodes to $|-\rangle$, giving us an error. Thus the current encoding does not work, and we will have to come up with a new one. Let us first recall that a Z is the same as an X in the Hadamard basis, $Z = HXH$. We can apply a Hadamard after encoding on every qubit, and then add a Hadamard to every qubit right before decoding. If there is no Z error, then the Hadamards will cancel, and if there is, the HZH will turn into an X, so we just have an X error. Thus we see that the Z encoder is just the X encoder with additional H gates on every qubit afterwards. Likewise, the Z decoder is just the X decoder with H gates on every qubits prior to the X encoder.

The Z encoder will map

$$|0\rangle \to |000\rangle \to |+++\rangle = |0\rangle_L$$
$$|1\rangle \to |111\rangle \to |---\rangle = |1\rangle_L$$

If the X code is correct, then the Z code is also correct. We can also argue this from first principles, on a test state $|\psi\rangle = |+\rangle$:

$$|\psi\rangle = |+\rangle \to \frac{1}{\sqrt{2}}(|+++\rangle + |---\rangle) \to \frac{1}{\sqrt{2}}(|-++\rangle + |+--\rangle)$$

After the correction, we will see

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

One big issue is that if we protect against X errors, we can't protect against Z errors, and vice-versa. We want something that can protect against both.

12.2.1 9-qubit Shor's Code

This code encodes a single qubit into 9 qubits, and it can protect against any single X or Z error. We have a 3-qubit X code, and a 3-qubit Z code, and we use something known as composition, to merge the two.

We begin with some single qubit original message $|\psi\rangle$. We then can encode the message with the X code, giving $|\psi\rangle_X$. We then apply the Z code to each of the 3 encode qubits, giving 9 qubits that represent $|\psi\rangle_{XZ}$. Likewise, we can also generate $|\psi\rangle_{ZX}$. Note that the last application of a code is the error that we can protect against. $|\psi\rangle_{XZ}$ can protect against a Z error, and $|\psi\rangle_{ZX}$ can protect against an X error. Now the question is whether they can also protect against previous applications of codes, can the $|\psi\rangle_{XZ}$ code also protect against an X error?

Let us first examine $|\psi\rangle_{XZ}$. Suppose we start with some arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. After the X encoding, we have

$$\alpha|000\rangle + \beta|111\rangle$$

And after the Z encoding on this we have

$$\alpha(|+++\rangle)(|+++\rangle)(|+++\rangle) + \beta(|---\rangle)(|---\rangle)(|---\rangle)$$

We know by definition that this can handle a Z error by construction. If we now think about an X error, we see that an X will generate a phase flip on one of the terms, which we have no way of fixing.

The second method involves using the Z code first:

$$\alpha|0\rangle + \beta|1\rangle \to \alpha|+++\rangle + \beta|---\rangle$$

Now applying the X code:

$$\frac{1}{\sqrt{2}}(|000\rangle+|111\rangle)\otimes\frac{1}{\sqrt{2}}(|000\rangle+|111\rangle)\otimes\frac{1}{\sqrt{2}}(|000\rangle+|111\rangle)+\beta\frac{1}{\sqrt{2}}(|000\rangle-|111\rangle)\otimes\frac{1}{\sqrt{2}}(|000\rangle-|111\rangle)\otimes\frac{1}{\sqrt{2}}(|000\rangle-|111\rangle)$$

We know that by definition, this can solve an X error by definition. Can this solve a Z error? We note that if we apply a Z error on one of the 9 qubits, we have that $\frac{1}{\sqrt{2}}(|000\rangle+|111\rangle) \to \frac{1}{\sqrt{2}}(|000\rangle-|111\rangle)$. We note that this goes from the X coded form of $|+\rangle$ to the X coded form of $|-\rangle$. This is exactly the error that the Z code is designed to fix, we apply the X decoder and use the error correction of the Z code. Thus we see that this code works if we apply the Z code first and the X code afterwards.

There exists a 5 qubit QECC for a 1-qubit message, against a single X or Z error, which is the best known.

12.3 Stabilizer Codes

Suppose we have an n qubit system. A stabilizer is a tensor product of Pauli matrices, $I\otimes X\otimes Z\otimes\ldots$, in any combination. If we are given a collection of stabilizers over an n qubit space, we can define a codeword space, as the common $+1$ eigenspace of all the stabilizers. If we have some state in this space, then for any stabilizer, the stabilizer applied to the state returns the state itself:

$$\forall i, S_i|\psi\rangle = |\psi\rangle$$

This condition actually gives us many nice properties about error correction. For example, if we have a 3-qubit X code, we have two stabilizers,

$$S_1 = Z_1 \otimes Z_2 \otimes I_3$$

$$S_2 = I_1 \otimes Z_2 \otimes Z_3$$

When we have an X_1 error, $X_1 |\psi\rangle$, we are no longer in the $+1$ eigenspace of the two stabilizers. We can apply the S_1 stabilizer:

$$S_1 X_1 |\psi\rangle = (Z_1 \otimes Z_2 \otimes I) X_1 |\psi\rangle = (Z_1 X_1 \otimes Z_2 \otimes I)$$

Now we note that Z and X anticommute, so $ZX = -XZ$:

$$= (-X_1 Z_1 \otimes Z_2 \otimes I) |\psi\rangle = -X_1 S_1 |\psi\rangle$$

$$= -X_1 |\psi\rangle$$

Thus we have that it now lies in the -1 eigenspace of the stabilizers. If we do the same thing for S_2, we see that it remained in the $+1$ eigenspace. We can generate a table

	I	X_1	X_2	X_3
S_1	+	-	-	+
S_2	+	+	-	-

We see that this is similar to the syndrome computations that we had before.