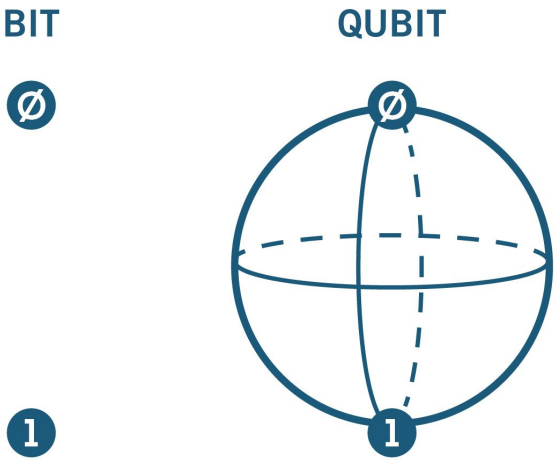
Theoretical Documentation For Quantum Computing

"Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy." - Richard Feynman



Bartu Yaman

Contents

1 Introduction

Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems.Quantum mechanical systems are governed by quantum mechanics, which is a mathematical framework or set of rules for the construction of physical theories.

1.1 Dirac Notation

Assuming that we want to solve the Schrödinger equation numerically and we have the wavefunction Ψ. If we want to represent this wavefunction as a vector:

- $|\Psi\rangle$ is a state vector, pronounced "ket-psi" and can be written as:

$$|\Psi\rangle = \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_n \end{bmatrix}$$

  for the set of coefficients a_i.

- The adjoint of $|\Psi\rangle$ is $\langle\Psi|$, pronounced "bra-psi" and can be written as:

$$\langle\Psi| = \begin{bmatrix} a_1^* & a_2^* & ... & a_n^* \end{bmatrix}$$

1.2 What Does A Quantum Algorithm Look Like?

1. I, the qubit, am in initial state $|0\rangle$.
2. I get moved into a standard superposition. Don't peek!
3. Steps in the algorithm apply zero or more reversible operations to me.
4. I get measured and always become $|0\rangle$ or $|1\rangle$.
5. When these get read out for classical use, they are converted to 0 or 1.

2 What Is A Qubit?

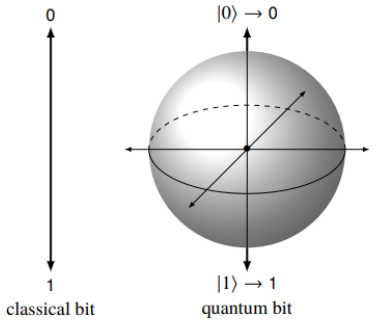


Figure 1: Classical bit and quantum bit.

A quantum bit, or qubit, is the fundamental information unit of quantum computing, implements a two-state quantum mechanical system and is the quantum analog of a classical bit. On the left, we have the classical situation where a bit can only take on the values 0 and 1. More precisely, a bit can be in one of those states and only those. You can look at the bit at any time and, assuming nothing has happened to change the state, it stays in that state.

For the quantum situation on the right, we change the notation slightly. The qubit always becomes the state $|0\rangle$ or $|1\rangle$ when we read information from it by a process called measurement. However, it is possible to move it to an infinite number of other states and change from one of them to another while we are computing with the qubit before measurement. Measurement says "ok, I'm going to peek at the qubit now" and the result is always a 0 or 1 once you do so. We can then read that out as a bit value of 0 or 1, respectively.

Yes, this is weird. This is quantum mechanics and it has amazed, and befuddled, and surprised, and delighted people for close to one hundred years. Quantum computing is based on and takes advantage of this behavior.

2.1 Superposition

At any given time, it is in a superposition state represented by a linear combination of vectors $|0\rangle$ and $|1\rangle$, often called superpositions:

$$a\,|0\rangle + b\,|1\rangle \quad where \quad \|a\|^2 + \|b\|^2 = 1$$

Through measurement, a qubit is forced to collapse irreversibly through projection to either $|0\rangle$ or $|1\rangle$. The probability of its doing either is $\|a\|^2$ and $\|a\|^2$, respectively. a and b are called probability amplitudes. If necessary, we can convert ("read out") $|0\rangle$ and $|1\rangle$ to classical bit values of 0 and 1.

2.2 A Quick Analogy

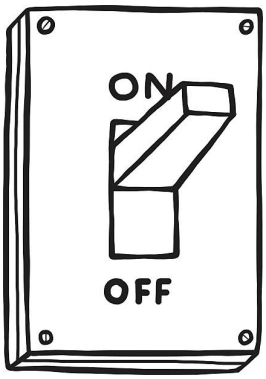


Figure 2: A classical switch

Suppose I am standing in a room with a single overhead light and a switch that turns the light on or off. This is just a normal switch, and so I can't dim the light. It is either fully on or fully off. I can change it at will, but this is the only thing I can do to it. There is a single door to the room and no windows. When the door is closed I cannot see any light. I can stay in the room or I may leave it. The light is always on or off based on the position of the switch.

Now I'm going to do some rewiring. I'm replacing the switch with one that is in another part of the building. I can't see the light at all but, once again, its being on or off is determined solely by the two positions of the switch. If I walk to the room with the light and open the door, I can see whether it is lit or dark. I can walk in and out of the room as many times as I want and the status of the light is still determined by that remote switch being on or off. This is a "classical" light. Now let's imagine a quantum light and switch, which I'll call a "qu-light" and "qu-switch," respectively.

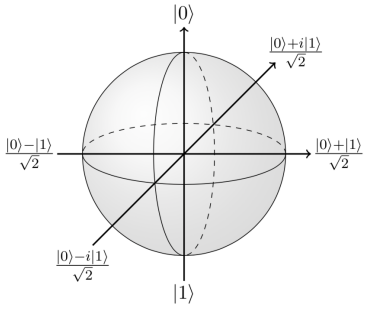


Figure 3: Bloch sphere visualization with superposition states

When I walk into the room with the qu-light it is always on or off, just like before. The qu-switch is unusual in that it is shaped like a sphere with the topmost point (the "north pole") being OFF and the bottommost (the "south pole") being ON. There is a line etched around the middle. The interesting part happens when I cannot see the qu-light, when I am in the other part of the building with the qu-switch. I control the qu-switch by placing my index finger on the qu-switch sphere. If I place my finger on the north pole, the qu-light is definitely off. If I put it on the south, the qu-light is definitely on. You can go into the room and check. You will always get these results. If I move my finger anywhere else on the qu-switch sphere, the qu-light may be on or off when you check. If you do not check, the qu-light is in an indeterminate state. It is not dimmed, it is not on or off, it just exists with some probability of being on or off when seen. This is unusual!

The moment you open the door and see the qu-light, the indeterminacy is removed. It will be on or off. Moreover, if I had my finger on the qu-switch, the finger would be forced to one or other of the poles corresponding to the state of the qu-light when it was seen. The act of observing the qu-light forced it into either the on or off state. I don't have to see the qu-light fixture itself. If I open the door a tiny bit, enough to see if any light is shining or not, that is enough. If I place a video camera in the room with the qu-light and watch it when I try to place my finger on the qu-switch, it behaves just like a normal switch. I will be prevented from touching the qu-switch at anywhere other than the top or bottom. Since I'm making up this example, assume some sort of force field keeps me away from anywhere but the poles! If you or I are not observing the qu-light in any way, does it make a difference where I touch the qu-switch? Will touching it in the northern or southern hemisphere influence whether it will be on or off when I observe the qu-light? Yes. Touching it closer to the north pole or the south pole will make the probability of the qu-light being off or on, respectively, be higher. If I put my finger on the circle between the poles, the equator, the probability of the light being on or off will be exactly 50-50. What I just described is called a two-state quantum

system. When it is not being observed, the qu-light is in a superposition of being on and off.

2.3 Bloch Sphere

Suppose we have $|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$. Since $\|a\|^2 + \|b\|^2 = 1$, we can rewrite our superposition state as:

$$|\Psi\rangle = e^{i\gamma}(cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}sin\frac{\theta}{2}\,|1\rangle)$$

where θ, φ and γ are real numbers. We can ignore the factor of $e^{i\gamma}$ out the front, because it has no observable effects, and for that reason we can effectively write:

$$|\Psi\rangle = (cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}sin\frac{\theta}{2}\,|1\rangle)$$

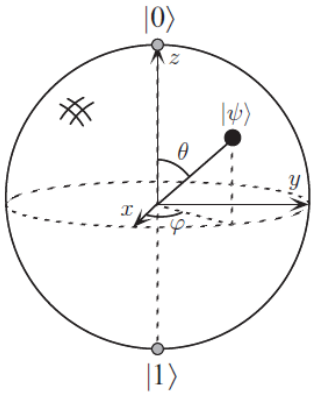


Figure 4: Bloch sphere representation

The numbers θ and φ define a point on the unit three-dimensional sphere, as shown in Figure 4. This sphere is often called the Bloch sphere; it provides a useful means of visualizing the state of a single qubit, and often serves as an excellent testbed for ideas about quantum computation and quantum information. Many of the operations on single qubits which we describe later in this chapter are neatly described within the Bloch sphere picture. However, it must be kept in mind that this intuition is limited because there is no simple generalization of the Bloch sphere known for multiple qubits.

How much information is represented by a qubit? Paradoxically, there are an infinite number of points on the unit sphere, so that in principle one could store an entire text of Shakespeare in the infinite binary expansion of θ. However, this conclusion turns out to be misleading, because of the

behavior of a qubit when observed. Recall that measurement of a qubit will give only either 0 or 1. Furthermore, measurement changes the state of a qubit, collapsing it from its superposition of $|0\rangle$ and $|1\rangle$ to the specific state consistent with the measurement result.

For example, if measurement of $|+\rangle$ gives 0, then the post-measurement state of the qubit will be $|0\rangle$. Why does this type of collapse occur? Nobody knows. This behavior is simply one of the fundamental postulates of quantum mechanics. What is relevant for our purposes is that from a single measurement one obtains only a single bit of information about the state of the qubit, thus resolving the apparent paradox. It turns out that only if infinitely many identically prepared qubits were measured would one be able to determine a and b for a qubit in the state we always used.

But an even more interesting question to ask might be: how much information is represented by a qubit if we do not measure it? This is a trick question, because how can one quantify information if it cannot be measured? Nevertheless, there is something conceptually important here, because when Nature evolves a closed quantum system of qubits, not performing any 'measurements', she apparently does keep track of all the continuous variables describing the state, like a and b. In a sense, in the state of a qubit, Nature conceals a great deal of 'hidden information'. And even more interestingly, we will see shortly that the potential amount of this extra 'information' grows exponentially with the number of qubits. Understanding this hidden quantum information is a question that we grapple with, and which lies at the heart of what makes quantum mechanics a powerful tool for information processing.

Here are two resources to try out the Bloch sphere for yourself: Click here and here.

3 Quantum Gates

Changes occurring to a quantum state can be described using the language of quantum computation. Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a quantum computer is built from a quantum circuit containing wires and elementary

quantum gates to carry around and manipulate the quantum information.

3.1 Single Qubit Gates

3.1.1 The Quantum X Gate

The X gate has the matrix:

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

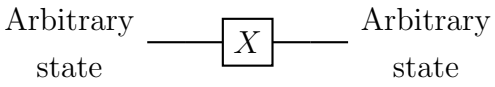
and this is the Pauli X matrix, named after Wolfgang Pauli.

It has the property that:

$$\sigma_x |0\rangle = |1\rangle \quad and \quad \sigma_x |1\rangle = |0\rangle$$

This reverses the probabilities of measuring $|0\rangle$ and $|1\rangle$. The not gate is a "bit flip" and by analogy we also say X is a bit flip.
In terms of the Bloch sphere, the X gate rotates by π around the x axis. So not only are the poles flipped but points in the lower hemisphere move to the upper and vice versa.

$$For \quad |\Psi\rangle = a |0\rangle + b |1\rangle , \quad X |\Psi\rangle = b |0\rangle + a |1\rangle$$



3.1.2 The Quantum Z Gate

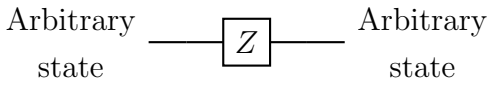
The Z gate has the matrix:

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

and this is the Pauli Z matrix. It rotates qubit states by π around the z axis on the Bloch sphere. The Z gate swaps $|+\rangle$ and $|-\rangle$ as well as $|i\rangle$ and

$|-i\rangle$. It leaves $|0\rangle$ and $|1\rangle$ alone on the Bloch sphere.

$$For \quad |\Psi\rangle = a\,|0\rangle + b\,|1\rangle\,, \quad Z\,|\Psi\rangle = a\,|0\rangle - b\,|1\rangle$$



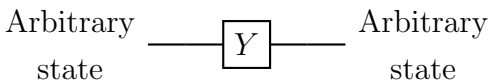
3.1.3 The Quantum Y Gate

The Y gate has the matrix:

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

and this is the Pauli Y matrix. It rotates qubit states by π around the y axis on the Bloch sphere. It swaps $|0\rangle$ and $|1\rangle$ and so is a bit flip. It also swaps $|+\rangle$ and $|-\rangle$ but leaves $|i\rangle$ and $|-i\rangle$ alone.

$$For \quad |\Psi\rangle = a\,|0\rangle + b\,|1\rangle\,, \quad Y\,|\Psi\rangle = -bi\,|0\rangle + ai\,|1\rangle = e^{\frac{3i\pi}{2}}(b\,|0\rangle - a\,|1\rangle)$$



3.1.4 The Quantum H Gate

The H gate or Hadamard gate, has the matrix:

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

By matrix multiplication:

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

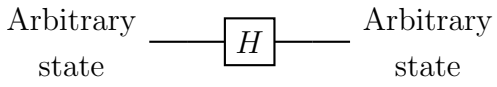
By linearity:

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

$$=$$

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

The Hadamard gate is one of the most frequently used gates in quantum computing. H is often the first gate applied in a circuit. When you read "put the qubit in superposition" it usually means "take the qubit initialized in the $|0\rangle$ state and apply H to it."
The Hadamard matrix is the change of basis matrix from $|0\rangle$, $|1\rangle$ to $|+\rangle$, $|-\rangle$. Since HH= I_2, the H gate is its own inverse.



3.1.5 The Quantum R_φ^x, R_φ^y, R_φ^z Gate

The R_φ^x gate, has the matrix:

$$R_\varphi^x = \begin{bmatrix} cos(\frac{\varphi}{2}) & -sin(\frac{\varphi}{2})i \\ -sin(\frac{\varphi}{2}), & cos(\frac{\varphi}{2}) \end{bmatrix} = cos(\frac{\varphi}{2})I_2 - cos(\frac{\varphi}{2})i\sigma_x$$

which defines an arbitrary rotation around the x axis.

The R_φ^y gate, has the matrix:

$$R_\varphi^y = \begin{bmatrix} cos(\frac{\varphi}{2}) & -sin(\frac{\varphi}{2}) \\ sin(\frac{\varphi}{2}) & cos(\frac{\varphi}{2}) \end{bmatrix} = cos(\frac{\varphi}{2})I_2 - cos(\frac{\varphi}{2})i\sigma_y$$

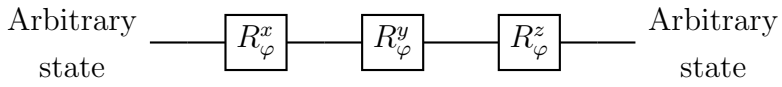
which defines an arbitrary rotation around the y axis.

The R_φ^z gate, has the matrix:

$$R_\varphi^z = \begin{bmatrix} e^{\frac{-i\varphi}{2}} & 0 \\ 0 & e^{\frac{i\varphi}{2}} \end{bmatrix} = cos(\frac{\varphi}{2})I_2 - cos(\frac{\varphi}{2})i\sigma_z$$

which defines an arbitrary rotation around the z axis.

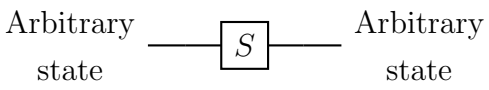
Where I_2 is the 2x2 identity matrix and σ_x, σ_y and σ_z are the Pauli matrices defined earlier.



3.1.6 The Quantum S Gate

The S gate is a shorthand for $R_{\frac{\pi}{2}}^z$. After applying, the phase is adjusted to be greater than or equal to 0 and less than 2π.

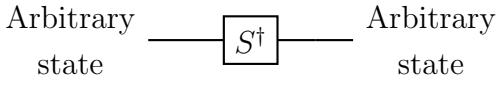
$$S = R_{\frac{\pi}{2}}^z = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$



3.1.7 The Quantum $S^\dagger$ Gate

The $S^\dagger$ gate is a shorthand for $R_{\frac{3\pi}{2}}^z = R_{\frac{-\pi}{2}}^z$. After applying, the phase is adjusted to be greater than or equal to 0 and less than 2π.

$$S^\dagger = R_{\frac{3\pi}{2}}^z = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{3i\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$$

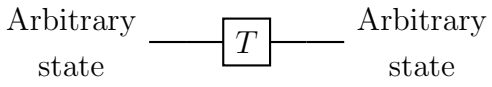


3.1.8 The Quantum T Gate

The T gate is a shorthand for $R_{\frac{\pi}{4}}^z$. After applying, the phase is adjusted to be greater than or equal to 0 and less than 2π.

$$T = R_{\frac{\pi}{4}}^z = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i \end{bmatrix}$$

We can get the S gate by applying the T gate twice.

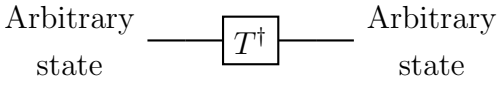


3.1.9 The Quantum $T^\dagger$ Gate

The $T^\dagger$ gate is a shorthand for $R^z_{-\frac{\pi}{4}} = R^z_{7\frac{\pi}{4}}$. After applying, the phase is adjusted to be greater than or equal to 0 and less than 2π.

$$T^\dagger = R^z_{7\frac{\pi}{4}} = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{7\pi}{4}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}i \end{bmatrix}$$

We can get the $S^\dagger$ gate by applying the $T^\dagger$ gate twice.



4 Multiple Qubits

4.1 What Is A Tensor Product?

If the direct sum seems to concatenate two vector spaces, then the tensor product interleaves them. In the first case, if we start with dimensions n and m, we end up with a new vector space of $n + m$ dimensions. For the tensor product, we get nm dimensions.

We can quickly get vector spaces with high dimensions through this multiplicative effect. This means we need to use our algebraic intuition and tools more than our geometric ones.

Let V and W be two finite dimensional vector spaces over F. Define a new vector space $V \otimes W$, pronounced "V" tensor "W" or "the tensor product of V and W," to be the vector space generated by addition and scalar multiplication of all objects $\mathbf{v} \otimes \mathbf{w}$ for each $\mathbf{v}$ in V and $\mathbf{w}$ in W.

If we have:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \quad and \quad B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}$$

then the regular matrix product is:

$$AB = \begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} \end{bmatrix}$$

The matrix tensor product for 2 by 2 matrices is:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B \\ a_{2,1}B & a_{2,2}B \end{bmatrix} = \begin{bmatrix} a_{1,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{1,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \\ a_{2,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{2,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \end{bmatrix}$$

Which yields the result:

$$A \otimes B = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} \end{bmatrix}$$

To show the use case of the tensor product, we can calculate the tensor product of H and σ_y gates:

$$For \qquad H = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \quad and \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

$$H \otimes \sigma_y = \begin{bmatrix} a_{1,1}B & a_{1,2}B \\ a_{2,1}B & a_{2,2}B \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & \frac{\sqrt{2}}{2}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\ \frac{\sqrt{2}}{2}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & -\frac{\sqrt{2}}{2}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \end{bmatrix}$$

Which yields the result:

$$A \otimes B = \begin{bmatrix} 0 & -\frac{\sqrt{2}}{2}i & 0 & -\frac{\sqrt{2}}{2}i \\ \frac{\sqrt{2}}{2}i & 0 & \frac{\sqrt{2}}{2}i & 0 \\ 0 & -\frac{\sqrt{2}}{2}i & 0 & \frac{\sqrt{2}}{2}i \\ \frac{\sqrt{2}}{2}i & 0 & -\frac{\sqrt{2}}{2}, & 0 \end{bmatrix}$$

4.2 What Is Entanglement?

When we have a quantum system with two qubits, we do not consider their collective states in a single C^2 instance. Instead, we use the tensor product of the two copies of C^2 and the tensor products of the quantum state vectors. This gives us a four-dimensional complex vector space where this "4" is 22 rather than the arithmetically equal $2 + 2$.

The tensor product is the machinery that allows up to build quantum systems from two or more other systems. The notation for working with these tensor products starts out as fairly bulky but there are significant simplifications that demonstrate the advantages of bras and kets.

Let $q1$ and $q2$ be two qubits and let $|0\rangle_1$, $|1\rangle_1|0\rangle_2$, $|1\rangle_2$ be the standard orthonormal basis kets for each of their C^2 state spaces.

$$|\Psi\rangle_1 = a_1 |0\rangle_1 + b_1 |0\rangle_1 \quad with \quad |a_1|^2 + |b_1|^2 = 1 \, |\Psi\rangle_2 = a_2 |0\rangle_2 + b_2 |0\rangle_2 \quad with \quad |a_2|^2 + |b_2|^2 = 1$$

Then these four kets:

$$|0\rangle_1 \otimes |0\rangle_2 , \quad |0\rangle_1 \otimes |1\rangle_2 , \quad |1\rangle_1 \otimes |0\rangle_2 , \quad |1\rangle_1 \otimes |1\rangle_2$$

are a basis for the combined state space $C^2 \otimes C^2$ for q_1 and q_2.

To determine the column vector forms of the 2-qubit basis kets in $C^2 \otimes C^2$:

$$From \qquad |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad and \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

With the following operations we obtain the new basis kets:

$$|00\rangle = |0\rangle \otimes |0\rangle \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$|01\rangle = |0\rangle \otimes |1\rangle \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$|10\rangle = |1\rangle \otimes |0\rangle \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$|11\rangle = |1\rangle \otimes |1\rangle \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Entanglement means that the states are so tightly correlated so that when the measurement value of one is known, it uniquely determines the second. You cannot do this with bits. With superposition, entanglement is one of the key differentiators between quantum and classical computing.

Let $|\Psi\rangle$ be a 2-qubit quantum state in $C^2 \otimes C^2$. $|\Psi\rangle$ is entangled if and only if it cannot be written as the tensor products of two 1-qubit kets:

$$|\Psi\rangle_1 \otimes |\Psi\rangle_2 = (a_1 |0\rangle_1 + b_1 |1\rangle_1) \otimes (a_1 |0\rangle_1 + b_2 |1\rangle_1)$$

where:

$$|\Psi\rangle_1 = a_1 |0\rangle_1 + b_1 |1\rangle_1 \qquad |\Psi\rangle_2 = a_2 |0\rangle_2 + b_2 |1\rangle_2$$

For example, suppose that $|Psi\rangle^+$ is not entangled. Then there exists a_1, b_1, a_2 and b_2 in C as above with

$$|\Psi\rangle^+ = 0 |00\rangle + \frac{\sqrt{2}}{2} |01\rangle + \frac{\sqrt{2}}{2} |10\rangle + 0 |11\rangle = a_1 a_2 |00\rangle + a_1 b_2 |01\rangle + b_1 a_2 |10\rangle + b_1 b_2 |11\rangle$$

This gives us four relationships

$$a_1a_2 = 0 \quad a_1b_2 = \frac{\sqrt{2}}{2} \quad b_1a_2 = \frac{\sqrt{2}}{2} \quad b_1b_2 = 0$$

From the first, either a_1 or a_2 is 0. Assume $a_1 = 0$. But then $0 = a_1b_2 = \frac{\sqrt{2}}{2}$. This is a contradiction. So a_2 must be 0. Again, though, $0 = b_1a_2 = \frac{\sqrt{2}}{2}$ and we have another impossibility. This means we cannot write $|\Psi\rangle^+$ as the tensor product of two 1-qubit kets and it is an entangled state.

If a 2-qubit quantum state is not entangled then we can separate it into the tensor product of two 1-qubit states. For this reason, if a quantum state is not entangled then it is separable.

4.2.1 Bell States

There are four Bell states:

$$\left|\Phi^+\right\rangle = \frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle \quad \left|\Phi^-\right\rangle = \frac{\sqrt{2}}{2}|00\rangle - \frac{\sqrt{2}}{2}|11\rangle$$

$$\left|\Psi^+\right\rangle = \frac{\sqrt{2}}{2}|01\rangle + \frac{\sqrt{2}}{2}|10\rangle \quad \left|\Psi^-\right\rangle = \frac{\sqrt{2}}{2}|01\rangle - \frac{\sqrt{2}}{2}|10\rangle$$

4.3 Multiple Qubit Gates

A quantum gate operation that operates on one qubit has a 2 by 2 unitary square matrix in a given basis. For two qubits, the matrix is 4 by 4. For ten, it is 2^{10} by 2^{10}, which is 1024 by 1024.

4.3.1 The Quantum $H^{\otimes n}$ Gate

We start by looking at what it means to apply a Hadamard H to each qubit in a 2-qubit system. Since

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

operating on C^2. Starting with the two qubit states

$$|\Psi\rangle_1 = a_1 |0\rangle_1 + b_1 |1\rangle_1 \quad and \quad |\Psi\rangle_2 = a_2 |0\rangle_2 + b_2 |1\rangle_2$$

Applying H to each qubit means to compute

$$(H |\Psi\rangle_1) \otimes (H |\Psi\rangle_2) = (H \otimes H)(|\Psi\rangle_1 \otimes |\Psi\rangle_2)$$

for some 4 by 4 unitary matrix $H^{\otimes 2}$. Given the definition of H and the technique of creating a matrix tensor product, we can compute

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

This matrix puts both qubits in a 2-qubit system that are each initially initialized to $|0\rangle$ into superposition. The patterns continues. This shows that applying the Hadamard gates to each qubit initialized to $|0\rangle$ creates a balanced superposition involving all the ket basis vectors.

4.3.2 The Quantum SWAP Gate

Given two qubits

$$|\Psi\rangle_1 = a_1 |0\rangle_1 + b_1 |1\rangle_1 \quad and \quad |\Psi\rangle_2 = a_2 |0\rangle_2 + b_2 |1\rangle_2$$

their tensor product is

$$|\Psi\rangle_1 \otimes |\Psi\rangle_2 = a_1 a_2 |00\rangle + a_1 b_2 |01\rangle + b_1 a_2 |10\rangle + b_1 b_2 |11\rangle$$

If we tensor them in the reverse order, we get

$$|\Psi\rangle_2 \otimes |\Psi\rangle_1 = a_2 a_1 |00\rangle + a_2 b_1 |01\rangle + b_2 a_1 |10\rangle + b_2 b_1 |11\rangle$$

The first and the fourth coefficients are the same but the second and third are switched.
The matrix

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

is an example of a 4 by 4 permutation matrix. Therefore:

$$M(|\Psi\rangle_1 \otimes |\Psi\rangle_2) = |\Psi\rangle_2 \otimes |\Psi\rangle_1$$

When used this way, we call the quantum gate with this matrix in the standard ket basis the SWAP gate.

4.3.3 The Quantum CNOT/CX Gate

The CNOT gate is one of the most important gates in quantum computing. It's used to create entangled qubits. It's not the only kind of gate that can do it, but it's simple and very commonly used.

The "C" in CNOT is for "controlled." Unlike the 1-qubit X gate, which unconditionally flips $|0\rangle$ to $|1\rangle$ and vice versa, CNOT has two qubit inputs and two outputs. Remember that quantum gates must be reversible. For this reason we must have the same number of inputs as outputs. We call the qubits q_1 and q_2 and their states $|\Psi\rangle_1$ and $|\Psi\rangle_2$, respectively.

This is the way CNOT works: it takes two inputs, $|\Psi\rangle_1$ and $|\Psi\rangle_2$.

- If $|\Psi\rangle_1$ is $|1\rangle$, then the state of q_1 remains $|\Psi\rangle_1$ but $|\Psi\rangle_2$ becomes $X|\Psi\rangle_2$.
- Otherwise the states of q_1 and q_2 are not changed.

The matrix for CNOT is

$$CX = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

4.3.4 The Quantum CY And CZ Gates

The matrices for CY and CZ are:

$$CY = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{array}\right] \qquad CZ = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{array}\right]$$

4.3.5 The Quantum CR_{Φ}^{Z} Gate

Another useful set of controlled gates are the ones where the action is R_{Φ}^{Z}. The first qubit controls whether the phase change, or rotation around the Bloch sphere z axis, should happen.

$$CR_{\Phi}^{Z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix} \quad where \quad e^{i\phi} = cos(\phi) + sin(\phi)i$$

4.3.6 The Quantum Toffoli CCNOT Gate

The quantum Toffoli CCNOT gate is a double control gate operating on three qubits. If the states of the first two qubits are $|1\rangle$ then it applies X to the third. Otherwise it is ID on the third. In all cases it is ID for the first two qubits.

Its matrix is an 8 by 8 permutation matrix

$$CCX = \left[\begin{array}{cccc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

4.3.7 The Quantum Fredkin CSWAP Gate

The quantum Fredkin CSWAP gate is a control gate operating on three qubits. If the state of the first qubit is $|1\rangle$ then the states of the second and third qubits are swapped, as in SWAP. If it is $|0\rangle$, nothing is changed.

Its matrix is an 8 by 8 permutation matrix.

$$CSWAP = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

5 Quantum Algorithms

5.1 No Cloning Theorem

Given an unknown quantum state, either $|\Phi\rangle$ or $|\Psi\rangle$, use a measurement to guess which one. If $|\Phi\rangle$ and $|\Psi\rangle$ are not orthogonal, then no measurement perfectly distinguishes them, and we always have some constant probability of error. However, if we could make many copies of the unknown state, then we could repeat the optimal measurement many times, and make the probability of error arbitrarily small. The no cloning theorem says that this isn't physically possible. Only sets of mutually orthogonal states can be copied by a single unitary operator.

Suppose there exists a unitary operator U_{cl} that can indeed clone an unknown quantum state $|\Phi\rangle = \alpha|0\rangle + \beta|1\rangle$. Then

$$|\Phi\rangle|0\rangle \xrightarrow{U_{cl}} |\Phi\rangle|\Phi\rangle = (\alpha|0\rangle + \beta|1\rangle)(\alpha|0\rangle + \beta|1\rangle) = \alpha^2|00\rangle + \beta\alpha|10\rangle + \alpha\beta|01\rangle + \beta^2|11\rangle$$

But now if we use U_{cl} to clone the expansion of $|\Phi\rangle$, we arrive at a different state:

$$(\alpha |0\rangle + \beta |1\rangle) |0\rangle \xrightarrow{U_{cl}} \alpha |00\rangle + \beta |11\rangle$$

Here there are no cross terms. Thus we have a contradiction and therefore there cannot exist such a unitary operator U_{cl}.

Note that it is however possible to clone a known state such as $|0\rangle$ and $|1\rangle$.

5.2 Teleportation

Though the name sounds like something out of science fiction, it doesn't involve any dema- terialization and rematerialization, or faster-than-light travel. In fact, it requires two classical bits of information to be transferred via traditional means.

The technique involves three qubits: M, which is my qubit; Y, which is your qubit; and Q, which is the qubit whose state I want to transfer from me to you. We use entanglement as the connection and transference mechanism.

My qubit Q is in some arbitrary quantum state $|\Psi\rangle_Q = a |0\rangle + b |1\rangle$. When we are done, you will know this state but I will no longer have access to it.

We begin by entangling M and Y. There an infinite number of ways of doing this but the usual choice is to use one of the four Bell states. I'll use $|\Phi^+\rangle = \frac{\sqrt{2}}{2} |00\rangle + \frac{\sqrt{2}}{2} |11\rangle$, but you can use any of them with appropriate changes to the algorithm and math below.

To keep track of which qubit belongs to whom, I modify the notation slightly so that

$$\left|\Phi^+\right\rangle_{MY} = \frac{\sqrt{2}}{2} |00\rangle_{MY} + \frac{\sqrt{2}}{2} |11\rangle_{MY}$$

to indicate the first qubit is mine and the second is yours.

We did this entanglement while we were close together, but now you can

get on a plane and go as far away as you wish. You might even take a trip to a space station orbiting the planet. The qubits are entangled and will stay that way in this scenario.

Next, I put Q into the mix to get

$$|\Psi\rangle_Q \otimes \left|\Phi^+\right\rangle_{MY} = (a\,|0\rangle_Q + b\,|1\rangle_Q) \otimes (\frac{\sqrt{2}}{2}\,|00\rangle_{MY} + \frac{\sqrt{2}}{2}\,|11\rangle_{MY})$$

Considering the Bell states introduced earlier and using linearity to rewrite

$$|\Psi\rangle_Q \otimes \left|\Phi^+\right\rangle_{MY} = \frac{1}{2}(\left|\Phi^+\right\rangle_{QM} \otimes (a\,|0\rangle_\gamma + b\,|1\rangle_\gamma) + \left|\Phi^-\right\rangle_{QM} \otimes (a\,|0\rangle_\gamma - b\,|1\rangle_\gamma) + \left|\Psi^+\right\rangle_{QM} \otimes (b\,|0\rangle_\gamma + a\,|1\rangle_\gamma) + \left|\Psi^-\right\rangle_{QM} \otimes (-b\,|0\rangle_\gamma + a\,|1\rangle_\gamma))$$

That was a lot of ket manipulation! Note what happened: the a and the b, which started out in the state of Q, which I own, is now on Y, which you own. Other than entangling qubits we did not do any sort of measurement, we just rewrote the ket and tensor formulas.

Now I measure. I don't do it in the $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ basis, I do it in the $|\Phi^+\rangle$, $|\Phi^-\rangle$, $|\Psi^+\rangle$ and $|\Psi^-\rangle$ basis. After measurement I have exactly one of the expressions

$$\begin{array}{cc} \left|\Phi^+\right\rangle_{QM} \otimes (a\,|0\rangle_\gamma + b\,|1\rangle_\gamma & \left|\Phi^-\right\rangle_{QM} \otimes (a\,|0\rangle_\gamma - b\,|1\rangle_\gamma \\ \left|\Psi^+\right\rangle_{QM} \otimes (b\,|0\rangle_\gamma + a\,|1\rangle_\gamma & \left|\Psi^-\right\rangle_{QM} \otimes (-b\,|0\rangle_\gamma + a\,|1\rangle_\gamma \end{array}$$

The probability of getting any one of them is 0.25 and I know which one I have by looking at Q and M! My measurement did not affect Y other than breaking the entanglement. The original quantum state of Q was destroyed.

Now I call you up or text you or email you or send you a letter through the post and tell you which of the basis vectors I observed. This information is represented using two bits and is sent using a classical communication channel.

- If I saw $\left|\Phi^+\right\rangle_{QM}$ then the quantum state of Q was successfully teleported to Y and you have nothing else to do.

- If I saw $|\Phi^-\rangle_{QM}$ then the quantum state of Y has the sign of b wrong. You apply a Z gate to do the phase flip and Y now has the original state of Q.
- If I saw $|\Psi^+\rangle_{QM}$ then the quantum state of Y has a and b reversed. You apply an X gate to do the bit flip and Y now has the original state of Q.
- If I saw $|\Psi^-\rangle_{QM}$ then the quantum state of Y has a and b reversed with the wrong signs. You apply an X gate and then a Z gate.

5.3 Grover's Algorithm

5.3.1 Quantum Oracles

An oracle is a function which we supply with data and it responds with a 1 for yes and a 0 for no. The oracles we use cannot answer arbitrary questions but are instead built to respond to a specific query. For the algorithms that use them, two things are significant:

- The implementation of the oracle must be as fast and efficient as possible.
- We want to call the oracle the fewest number of times as possible to minimize the complexity of the algorithm.

An oracle is often called a black box, meaning we understand its behavior but not how it does what it does. Its function of answer 1/yes/true or 0/no/false also means it acts like a predicate.
Since all data can ultimately be represented by bits, we express the inputs to the oracle function as strings of 0s and 1s. If we call the function f, which is traditional, we can express it as

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

This is an exceptionally terse way of saying suppose we have a string s of n 0s and 1s, for example $s = 10110010$ for $n = 8$. When we apply f to s we

get back a 1 if the criteria of the oracle is met, 0 otherwise.
For quantum computing, we would call the oracle with a ket.

$$f(|\Psi\rangle) = \begin{cases} |1\rangle\,, & \text{if } |\Psi\rangle = |1000010\rangle = |66\rangle_7 \\ |0\rangle\,, & \text{otherwise} \end{cases}$$

We are primarily interested in the action of the oracle on the standard basis kets.
Here's another oracle:

$$f(x_0x_1x_2x_3x_4x_5x_6x_7) = \begin{cases} |1\rangle\,, & \text{if } (x_0x_1x_2x_3) \times (x_4x_5x_6x_7) = 1111_2 \\ |0\rangle\,, & \text{otherwise} \end{cases}$$

This returns $|1\rangle$ if the product of the two binary numbers encoded in the two halves of the input is 1111_2 (= 15 decimal). The oracle doesn't need to know how to factor, it needs to know how to multiply and test for equality. With many qubits and a lot of time, we could use the oracle to identify the factors of a large integer.

In practice, for quantum computing we need to encapsulate the function of the oracle within a unitary matrix and gate U_f somehow. (The U in U_f stands for "unitary.") One way to do it is to adjust the sign of a ket that meets the oracle's criteria for success.

1. Encode the data we are searching through as standard basis kets $|x\rangle$. For example, if we have 100 data items then use 7 qubits (because $2^7 = 128$) and assign each data element to a basis ket $|x\rangle_7$.
2. Let our oracle f produce $f(|y\rangle) = 1$ for one special encoded piece of data $|y\rangle$, 0 otherwise.
3. Find a unitary (hence reversible) matrix U_f such that

$$U_f\,|x\rangle = \begin{cases} -\,|y\rangle\,, & \text{if } |x\rangle = |y\rangle \text{ (which means } \quad f(|y\rangle) = 1) \\ |x\rangle\,, & \text{otherwise} \end{cases}$$

This means U_f inverts the sign of the ket input that satisfies the condition of the oracle and leaves the rest alone.

To translate all these into an example, suppose we have two qubits and we want the oracle to return $|1\rangle$ if we give it $|01\rangle$, $|0\rangle$ otherwise. For $|\Psi\rangle$ being one of the standard basis kets $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|11\rangle$,

$$f(|\Psi\rangle) = \begin{cases} |1\rangle, & \text{if } |\Psi\rangle = |01\rangle \\ |0\rangle, & \text{if } |\Psi\rangle = |00\rangle, |10\rangle, \text{or } |11\rangle \end{cases}$$

Then U_f should behave as

$$U_f(|\Psi\rangle) = \begin{cases} -|01\rangle, & \text{if } |\Psi\rangle = |01\rangle \\ |\Psi\rangle, & \text{if } |\Psi\rangle = |00\rangle, |10\rangle, \text{or } |11\rangle \end{cases}$$

In standard vector notation, recall we have

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

Then the matrix for U_f

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

does what we need.

Depending on the algorithm and how we use the oracle, we might create a different U_f but it must still reflect what f tells us. While we think of U_f as a matrix, we implement it as a circuit. The trick is to construct it to be extremely computationally efficient.

5.3.2 Amplitude Amplification

Suppose we have three qubits and the standard basis kets for them each correspond to a possible solution to some problem. We want to devise an

algorithm that chooses the best solution among them. I'm purposely not telling you what the problem is or how the kets map to the data and solution. Just assume we want to identify one of them that the algorithm can determine as best.

The first question is how to see that this best ket stands out from another. The general form for a 3-qubit register state is

$$\sum_{j=0}^{7} a_j \left|j\right\rangle_3 = a_0 \left|000\right\rangle + a_1 \left|001\right\rangle + a_2 \left|010\right\rangle + a_3 \left|011\right\rangle + a_4 \left|100\right\rangle + a_5 \left|101\right\rangle + a_6 \left|110\right\rangle + a_7 \left|111\right\rangle$$

with

$$1 = \sum_{j=0}^{7} \left|a_j\right|^2$$

If we initialize each qubit to $\left|0\right\rangle$ and then apply $H^{\otimes 3}$, we get a balanced superposition via a change of basis

$$\left|\Phi\right\rangle = H^{\otimes 3} \left|000\right\rangle = \frac{1}{\sqrt{8}} \sum_{j=0}^{7} \left|j\right\rangle$$

All the coefficients are equal and the square of each absolute value is $\frac{1}{8}$. If we measure the qubits now, we have an equal chance of getting any one of the eight basis kets.

At the qubit level, we have done a change of basis from the computational basis $\{\left|0\right\rangle, \left|1\right\rangle\}$ to $\{\left|+\right\rangle, \left|-\right\rangle\}$ via H.

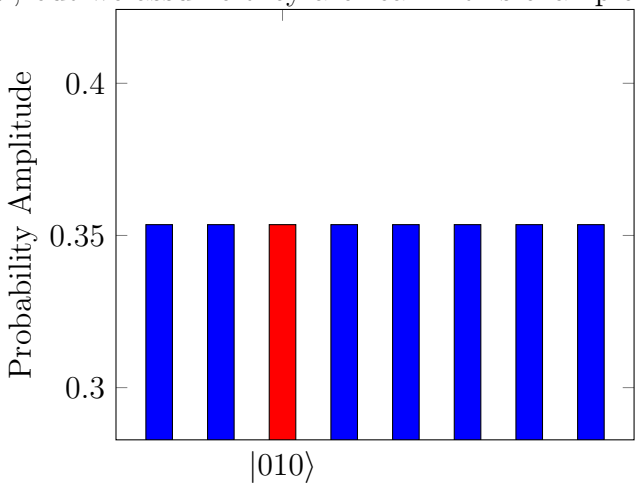
Through a process called amplitude amplification, we manipulate the qubit states so that the basis ket which represents the best solution has the coefficient a_j with the largest probability $\left|a_j\right|^2$. When we measure, that standard basis ket has the highest chance of being observed. We want to make $\left|a_j\right|^2$ as large as possible, ideally equal to 1.

Ultimately, this is the goal of every quantum algorithm: have the results of the final qubit measurements correspond with high probability to the best solution.

In practice, we often create a reusable portion of a circuit, what we call a circuit subroutine, which we call several times. Each time it gets us closer to what we hope is the ideal result by increasing its corresponding probability. We also decrease the probabilities of the "bad" results.

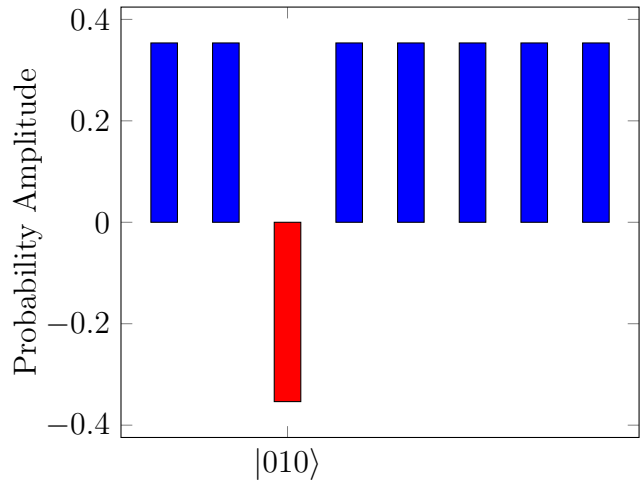
5.3.3 Sign Flipping

In the following graph, the vertical axis shows the probability amplitudes. That is, we map the coefficients a_j on this. Normally these are general values in C, but we assume they are real in this example.



I've highlighted one ket, $|010\rangle$, that I want to modify over several interactions to increase its probability. I've chosen it at random to show the process. The dotted line shows the average of the probability amplitudes.

We know how to negate the sign for a given ket in a balanced superposition. Let's do that now for $|010\rangle$. Create an 8 by 8 matrix $U_{|010\rangle}$, which is the identity matrix, except for the (3, 3) entry, which we change to 1. After applying this, the above changes to



Note how the average amplitude has dropped. The sum of the squares of the absolute values of the amplitudes is still 1. All basis kets have the same probability of being the result when measured.

In this example with three qubits we had one particular computational basis ket whose sign we wanted to flip, the third. In general, it is the oracle f that completely determines which basis ket we sign-flip.

5.3.4 Inversion About The Mean

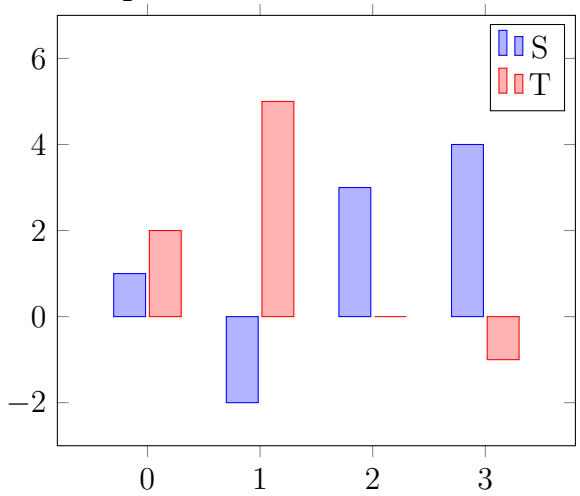
This is the really clever trick.

Let $S = \{s_j\}$ be a finite collection of numbers in R and let m be the mean (average) of the numbers. If we create a new collection T containing the numbers $\{t_j = 2m - s_j\}$, T has the following properties

1. The average of the numbers in T is still m.
2. If $s_j = m$ then $t_j = s_j$.
3. $t_j - m = m - s_j$ and so $|t_j - m| = |s_j - m|$.
4. If $s_j < m$ then $t_j > m$ and if $s_j > m$ then $t_j < m$.

This is called inversion about the mean.

To visualize this, suppose our collection is $S = \{1, -2, 3, 4\}$. The average, or mean, is $\frac{3}{2}$ and so $T = \{2, 5, 0, -1\}$, as seen below.



Note how much higher above the mean is the result in T for the single negative value -2 in S.

After this transformation, the lone negative value stands out significantly once inverted about the mean.

Consider

$$U_\phi = 2 |\phi\rangle \langle\phi| - I_8$$

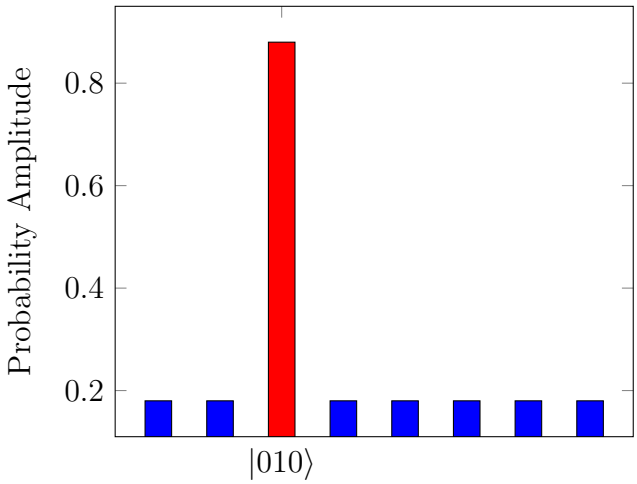
for

$$|\phi\rangle = \sum_{j=0}^{7} \frac{1}{\sqrt{8}} |j\rangle$$

and where I_8 is the 8 by 8 identity matrix. The form should be familiar to you from what we discussed earlier. This is a unitary matrix that accomplishes an "inversion about the mean" where by "mean" we are talking about the balanced superposition ket $|\phi\rangle = H^{\otimes 2} |000\rangle H^{\otimes 2}$.

And now we take our amplitudes in the 3-qubit example from earlier and

invert about the mean.



In general, for n qubits this inversion about the mean is accomplished by

$$U_\phi = 2\left|\phi\right\rangle\left\langle\phi\right| - I_{2^n}$$

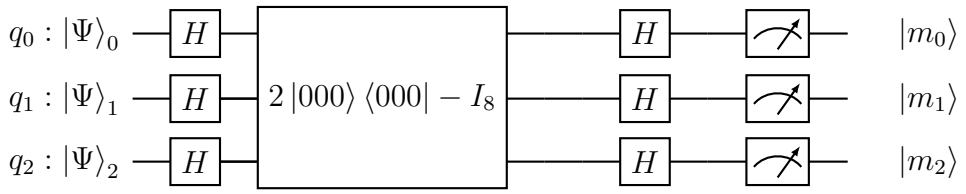
where

$$\left|\phi\right\rangle = H^{\otimes n}\left|0\right\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}\left|j\right\rangle$$

and I_{2^n} is the 2^n by 2^n identity matrix. Using linearity, we can re-express it using gate notation as

$$U_\phi = 2\left|\phi\right\rangle\left\langle\phi\right| - I_{2^n} = H^{\otimes n}(2\left|0\right\rangle^{\otimes n} - ID^{\otimes n})H^{\otimes n}$$

This is called the Grover diffusion operator. The 3 qubit circuit is



This is meant to be employed as a reusable circuit subroutine. The inputs vary based on where in the full circuit it is used

Inversion about the mean is an example of using interference to find the best answer. We manipulate the probability amplitudes to make the "good" result more likely to be seen when we measure while we simultaneously make the "bad" ones highly unlikely. We boost the good one via constructive interference and reduce the bad ones by destructive interference.

We just saw how if we have one standard basis ket in mind, we flip the probability amplitude and then amplify the amplitude for that ket. When we repeat the process enough times, we are likely to measure the right ket with high probability.

5.4 Grover's Search Algorithm

Here we put everything together to describe the famous quantum search algorithm discovered by Lov Kumar Grover, a computer scientist.

Instead of using the magic gate matrix $U_{|010\rangle}$, which flips the sign of the amplitude of the given ket, instead employ U_f, which is related to the oracle f.

In essence, I have an oracle which I can call but cannot see. I create U_f and then by repeating $U_f U_\phi$ enough times, I can find the special element for which f returns 1.

Suppose you say "I'm thinking of a number between 1 and 100" and then you give me an oracle f that can identify the number. I can find the number you are thinking of in approximately $10 = \sqrt{100}$ iterations of $U_f U_\phi$. Clasically, it can require 99 calls to the oracle.

We go from $O(N)$ to $O(\sqrt{N})$ for $N = 100$. This is a quadratic improvement. It doesn't look like much of a gain for small numbers but going from $10000 = 10^4$ to $100 = 10^2$ is significant.

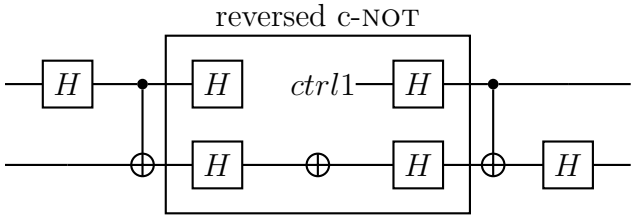
We can even be more precise, it takes approximately $\frac{\pi}{4}\sqrt{N} \approx 0.7856\sqrt{N}$ iterations to maximize the probability of getting the object for which you are searching.

The steps for the Grover search algorithm are:

1. Identify the data you want to search. Let N be the number of items.
2. Find the smallest positive n in Z such that $N \leq 2^n$. You need n qubits for the search algorithm.
3. Figure out a way to map uniquely from the data items to search to basis kets like $|j\rangle_n$ for $j < N$. This means we have an easy to go from the data object to the basis ket and back again.
4. Construct an oracle f and gate/matrix U_f that flips the sign of the object for which you are searching You do not know its basis ket ahead of time or you would otherwise extract it from the database.
5. Run the $U_f U_\phi$ circuit $\sqrt{N}$ times, rounding down.
6. Measure and read off the ket which corresponds to the sought item in the data. Map this back to the item within the data collection.
7. If the answer is not correct, repeat the above. The chance of error is $O(\frac{1}{N})$.

5.4.1 Using The Oracle

Just as there are different circuits that do the same thing more or less efficiently, there are different ways of coding the Grover search circuit. Here is an explicit search example.



After initializing the two qubits $|0\rangle$, we place the entire quantum register in a balanced superposition.

The vertical dashed line allows me to describe the steps in the circuit and

have no effect on computation.

The states after each operation can be shown as

1. $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$

For the circuit section 2, states after each operation can be shown as

1. $\frac{1}{2}(|00\rangle + i\,|01\rangle + |10\rangle + i\,|11\rangle)$
2. $\frac{\sqrt{2}}{4}((1+i)\,|00\rangle + (1-i)\,|01\rangle + (1+i)\,|10\rangle + (1-i)\,|11\rangle)$
3. $\frac{\sqrt{2}}{4}((1+i)\,|00\rangle + (1-i)\,|01\rangle + (1-i)\,|10\rangle + (1+i)\,|11\rangle)$
4. $\frac{1}{2}(|00\rangle + i\,|01\rangle + |10\rangle - i\,|11\rangle)$
5. $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle + |11\rangle)$

For the circuit section 3, states after each operation can be shown as

1. $\frac{1}{2}(|00\rangle + |01\rangle - |10\rangle + i\,|11\rangle)$
2. $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle + |11\rangle)$
3. $\frac{\sqrt{2}}{2}(|01\rangle + |10\rangle)$
4. $\frac{\sqrt{2}}{2}(|01\rangle + |11\rangle)$
5. $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$
6. $\frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)$
7. $-|01\rangle$

For the circuit section 4, states after each operation can be shown as

The states of the qubit measurement after the final step is

$$-|01\rangle$$

With 100% probability, the answer is $-|01\rangle$!

It's rare to get such a high percentage but it always happens with this algorithm in one iteration for two qubits.

5.4.2 Understanding The Oracle

5.5 The Deutsch-Jozsa Algorithm

Before we leave oracles, I want to walk you through one of the other early quantum algorithms that employs them. It also shows us another form in which oracles are expressed in quantum circuits.

Let's begin with an example. Suppose I buy two standard decks of 52 playing cards. In a separate room where you cannot see me, I create a single deck of 52 cards where one of the following is true:

1. All the cards are red or all the cards are black.
2. Half the cards (26) are black and half are red.

The first option is called "constant" and the second is "balanced."

I now go to you and give you the problem of finding out which of the two possibilities is the case for the deck I am holding. You do so by looking at and then discarding the card at the top of the deck.

In the best case, the first card is one color and the second is the other. Therefore the deck is balanced. In the worst case you must examine $27 = 1 + 52/2$ cards. The first 26 cards might be black, say. If the next is black, then all are black. If it is red, the deck is balanced.

Regarding an oracle, we are asking "is the card at the top of the deck black?". It returns 1 if it is, 0 if it is red. As I stated, we must consult the oracle 27 times in the worst case to get the correct answer.

When we first saw the definition of the oracle, it was expressed as a function of f operating on strings of n bits

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

When we translate this to quantum computing, we consider standard basis kets instead of bit strings.

Our new problem is this: for all possible bit strings of length n or standard basis kets for n qubits, the oracle f either always returns 0, always returns 1, or is balanced. How many calls to the oracle do we need to do to determine whether it is constant or balanced?

There are 2^n bir strings of length n and the brute force classical approach could require our looking (that is, calling the oracle) at one more than half of them, which is $2^{n-1} + 1$.

The quantum solution to this, with all the improvements to the original, is called the Deutsch-Jozsa algorithm. It was discovered by physicist David Deutsch and mathematician Richard Jozsa, and was based on earlier work by Deutsch.

With the summation notation, we can write the Hadamard gate as

$$H\left|u\right\rangle = \frac{\sqrt{2}}{2} \sum_{v \text{ in } \{0,\, 1\}} (-1)^{uv} \left|v\right\rangle$$

Please take a moment to verify that this is true and really quite clever. By the way, note how we use the powers of 1 to change the signs of kets. For a physicist, this is a phase change. For a regular person, this is a change of sign.

To show that Hadamard gates transfrom standard kets in a particularly nice way, we can generalize the formula to n qubits as

$$H \otimes^n \left|u\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{v \; in \; \{0,1\}^n} (-1)^{u.v} \left|v\right\rangle$$

Now let's compute a special case which we need in the Deutsch-Jozsa algorithm. For u we take a bit of string of n 0s with a single 1 tackled on at the end. So $\left|u\right\rangle = \left|0\right\rangle^{\otimes n} \otimes \left|1\right\rangle$. In this case, $u \cdot v = v_{n+1}$ because $u_j = 0$ for $1 <= j <= n$ and $u_{n+1} = 1$.

$$H^{\otimes n+1} \left|0...01\right\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{v \; in \; \{0,1\}^n} (-1)^{v_{n+1}} \left|v\right\rangle$$

The last bit of v is controlling the sign. Let's isolate that by rewriting $|v\rangle = |x\rangle \otimes |y\rangle$ where y is now the last bit. Then the formula becomes

$$H^{\otimes n+1} |0...01\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x \ in \ \{0,1\}^n} (|x\rangle \otimes |0\rangle - |x\rangle \otimes |1\rangle) =$$

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

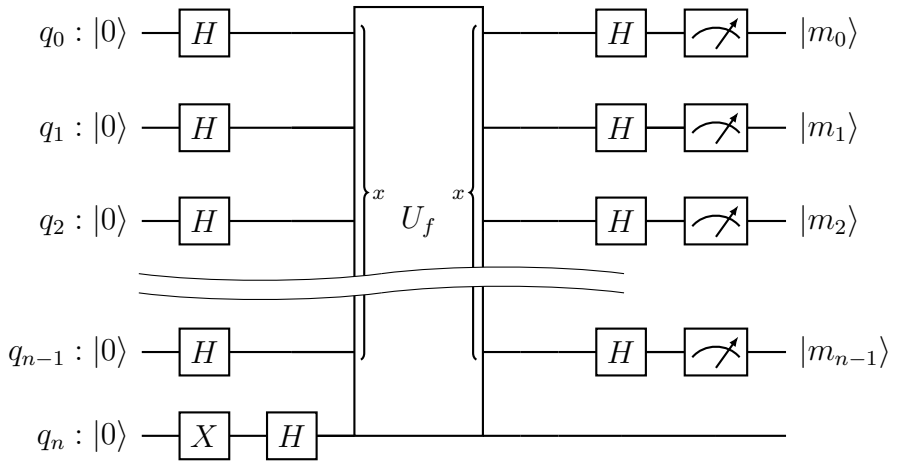
The oracle for Grover search was incorporated into the circuit and modified only the input states. We here look at another way of constructing a circuit with an oracle.

The superposition part of the circuit implements it as

$$H^{\otimes n+1} |0...01\rangle = (\frac{1}{\sqrt{2^n}} \sum_{x \ in \ \{0,1\}^n} |x\rangle) \otimes (\frac{\sqrt{2}}{2} |0\rangle - \frac{\sqrt{2}}{2} |1\rangle) = H^{\otimes n} |0\rangle^{\otimes n} H |1\rangle$$

In the inputs to U_f, we label the collective states for the n "data" qubits q_0 through q_{n-1} as $|x\rangle$. We label the state for the ancilla qubit q_n as $|y\rangle$. All together, the input to U_f is $|x\rangle \otimes |y\rangle$.

The Deutsch-Jozsa circuit is:



The full result of U_f is

$$(\frac{1}{\sqrt{2^n}} \sum_{xin0,1^n} (-1)^{f(x)} |x\rangle) \otimes (\frac{\sqrt{2}}{2}(|0\rangle - |1\rangle))$$

The expression to the right of "$\otimes$" is constant and we no longer need it. Remember that this is the state of the ancilla qubit q_n.

The output of the U_f gate for the n data qubits q_0 to q_{n-1} is

$$(\frac{1}{\sqrt{2^n}} \sum_{xin0,1^n} (-1)^{f(x)} |x\rangle)$$

We have encoded the effect of the oracle into the phase of each $|x\rangle$. We have done this by multiplying $|x\rangle$ by a value of absolute value 1, namely $(-1)^{f(x)}$. This is phase kickback.

We apply the $H^{\otimes n}$ gates to the above expression using the formula

$$H^{\otimes n} |u\rangle = \frac{1}{\sqrt{2^n}} \sum_{vin0,1^n} (-1)^{u.v} |v\rangle$$

This produces

$$\frac{1}{2^n} \sum_{vin0,1^n} (\sum_{xin0,1^n} (-1)^{x.v}) |v\rangle$$

Before we turn to measurement, let me say that I know this section is very intensive with all the arithmetic with the Σ summations. We compress a lot of information about the kets and their amplitudes into formulas and then manipulate and simplify them.

If v is the zero bit string, the sum is 2^n copies of $(-1)^0 = 1$ added together, which is 2^n. If v is not all zeros, half the $(-1)^{x.v}$ are 1 and the other half -1. They cancel each other when added together.

When $|v\rangle = |0\rangle^{\otimes n}$, this expression reduces to

$$\frac{1}{2^n}(\sum_{xin0,1^n}(-1)^{f(x)})\,|0\rangle^{\otimes n}$$

This is the amplitude for $|0\rangle^{\otimes n}$. It is equal to 1 when f is constant and 0 when f is balanced. The first case is constructive interference and the second is destructive interference.

When we run the Deutsch-Jozsa algorithm, if we get $|0\rangle^{\otimes n}$ after measurement, then the oracle is constant. Otherwise it is balanced.

5.6 Shor's Algorithm

5.6.1 Quantum Fourier Transform

The Quantum Fourier Transform (QFT) is widely used in quantum computing, notably in Shor's factorization algorithm.

5.6.1.1 Roots Of Unity

We are all familiar with square roots. For example, $\sqrt{4}$ is equal to either 2 or 2. We can also write $\sqrt{2} = 2^{\frac{1}{2}}$ and say that there are two "2nd-roots of 2." Similarly, 5 is a cube root, or "3rd-root," of 125. In general, we talk about an "Nth-root" for some natural number N. When we consider the complex numbers, we have a rich collection of such Nth-roots for 1.

Let N be in N. An Nth root of unity is a complex number w such that $w^N = 1$. "w" is the lowercase greek letter "omega." There are N Nth roots of unity and 1 is always one of those. If every other Nth root of unity can be expressed as a natural number power of w, then w is a primitive Nth root of unity. If N is prime then every Nth root of unity is primitive except 1.

Putting this "math language" into context, we can say that

- For $N = 1$, there is only one first root of unity, and that is 1 itself.

- For $N = 2$, there are two second roots of unity: 1 and -1. -1 is primitive.
- For $N = 3$, we can see the pattern below.

Remember Euler's formula

$$e^{i\phi} = cos(\phi) + sin(\phi)i$$

and $\left|e^{i\phi}\right| = 1$. If $\phi = 2\pi$, we go all the way around the unit circle and back to 1. If $\phi = \frac{2\pi}{3}$ then we go only one-third of the way. Rotating another $\frac{2\pi}{3}$ radians we get to two-thirds around. The third roots of unity are

$$\begin{aligned} w_0 &= e^{\frac{0x2\pi}{3}i} = 1 \\ w_1 = e^{\frac{1x2\pi}{3}i} &= cos(\frac{2\pi}{3}) + sin(\frac{2\pi}{3})i = -\frac{1}{2} + \frac{\sqrt{3}}{2}i \\ w_2 = e^{\frac{2x2\pi}{3}i} &= cos(\frac{4\pi}{3}) + sin(\frac{4\pi}{3})i = -\frac{1}{2} - \frac{\sqrt{3}}{2}i \end{aligned}$$

and w_1 and w_2 are primitive.
For $N = 4$ we can do something similar, though the situation is much simpler.

$$\begin{aligned} w_0 &= e^{\frac{0x2\pi}{3}i} = 1 \\ w_1 = e^{\frac{1x2\pi}{4}i} &= cos(\frac{\pi}{2}) + sin(\frac{\pi}{2})i = i \\ w_2 = e^{\frac{2x2\pi}{4}i} &= cos(\pi) + sin(\pi)i = -1 \\ w_3 = e^{\frac{3x2\pi}{4}i} &= cos(\frac{3\pi}{2}) + sin(\frac{3\pi}{2})i = -i \end{aligned}$$

Notice something else here: for the first time, we see overlaps in the collection of roots of unity for different N. This happens when the greatest common divisor of the different N is greater than 1.

For $N = 5$ and above we continue in the same way.

$w = e^{\frac{2\pi}{N}i}$ is a primitive Nth root of unity.

Also since $1 = w\bar{w} = ww^{-1}$,

$$\text{If } w_k = e^{\frac{2k\pi}{N}i} \text{ is a root of unity, then } \bar{w_k} = w^{-1} = e^{-\frac{2k\pi}{N}i}$$

Let $w = e^{\frac{2\pi}{N}i}$, then

$$0 = \sum_{k=0}^{N-1} e^{\frac{2k\pi}{N}i} = \sum_{k=0}^{N-1} e^{-\frac{2k\pi}{N}i}$$

with the second equality holding because we are using $\bar{w}$ instead of w as the Nth root of unity. Thought of another way, we are adding together the same set of roots of unity, just in a different order. In these equivalent versions, this is called the summation formula.

For j and k in Z, the Kronecker delta function $\delta_{j,k}$ is defined as

$$\delta_{j,k} = \begin{cases} 0, & \text{if } j \neq k \\ 1, & \text{if } j = k \end{cases}$$

For $0 <= j <= N$, the extended summation formula is

$$\sum_{k=0}^{N-1} e^{\frac{2kj\pi}{N}i} = \sum_{k=0}^{N-1} e^{-\frac{2kj\pi}{N}i} = N\delta_{j,0} = \begin{cases} 0, & \text{if } \delta_{j,0} = 0 \\ N, & \text{if } \delta_{j,0} = 1 \end{cases}$$

As we have seen, a general quantum state $|\phi\rangle$ on n qubits can be written as

$$|\phi\rangle = \sum_{j=0}^{2^n-1} a_j \, |j\rangle_n = \sum_{j=0}^{N-1} a_j \, |j\rangle_n$$

for $N = 2^n$. There are N amplitudes $a_j$corresponding to the N standard basis kets $|j\rangle$.

5.6.1.2 Definition

The quantum Fourier Transform of $|\phi\rangle$ is

$$\mathbf{QFT}_n : |\phi\rangle = \sum_{j=0}^{N-1} a_j |j\rangle_n \rightarrow |\phi\rangle = \sum_{j=0}^{N-1} b_j |j\rangle_n$$

where

$$b_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{\frac{2kj\pi}{N} i}$$

We can simplify this by letting $w = e^{\frac{2\pi}{N} i}$, which is a primitive Nth root of unity.

$$b_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k w^{jk}$$

When we have only 1 qubit, $N = 2$ and $w = -1$. So

$$b_j = \frac{1}{\sqrt{2}}(a_0(-1)^{-0j} + a_1(-1)^{1j}) = \frac{\sqrt{2}}{2}(a_0 + s_1(-1)^j)$$

For $|\Psi\rangle = |0\rangle$, $a_1 = 0$. So $b_0 = \frac{2}{2}$ and $b_1 = \frac{2}{2}$. For $|\Psi\rangle = |1\rangle$, $a_0 = 0$ and $a_1 = 1$. So $b_0 = \frac{2}{2}$ and $b_1 = -\frac{2}{2}$.

All together,

$$\mathbf{QFT}_1 |0\rangle = \frac{\sqrt{2}}{2}(|0\rangle + |1\rangle) = |+\rangle$$
$$\mathbf{QFT}_1 |1\rangle = \frac{\sqrt{2}}{2}(|0\rangle - |1\rangle) = |-\rangle$$

Which is the Hadamard gate, H!

5.6.1.3 Matrix Description

From the definition, we can see that $\mathbf{QFT}_n$ has the following matrix after the necessary algebraic simplifications

$$\mathbf{QFT}_n : \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & ... & 1 \\ 1 & w & w^2 & w^3 & ... & w^{N-1} \\ 1 & w2 & w^4 & w^6 & ... & w^{N-2} \\ 1 & w^3 & w^6 & w^9 & ... & w^{N-3} \\ ... & ... & ... & ... & ... & ... \\ 1 & w^{N-1} & w^{N-2} & w^{N-3} & ... & w \end{bmatrix}$$

5.6.1.4 Recursive Matrix Description

The higher Hadamard matrices $H^{\otimes n}$ are defined recursively in terms of the lower ones by

$$H^{\otimes n+1} = \frac{\sqrt{2}}{2} \begin{bmatrix} H^{\otimes n} & H^{\otimes n} \\ H^{\otimes n} & -H^{\otimes n+1} \end{bmatrix}$$

There is a similar decomposition for $\mathbf{QFT}_n$

$$\mathbf{QFT}_{n+1} = \begin{bmatrix} I_N & \Omega_N \\ I_N & -\Omega_N \end{bmatrix} \begin{bmatrix} \mathbf{QFT}_n & 0 \\ 0 & \mathbf{QFT}_n \end{bmatrix} P_{2^{n+1}}$$

where:

- $N = 2^n$,
- I_N is the N by N identity matrix,
- w is the primitive Nth root of unity $e^{\frac{2\pi i}{N}}$,
- Ω_N is the diagonal matrix

$$\Omega_N = \begin{bmatrix} 1 & 0 & 0 & 0 & ... & 0 \\ 0 & w & 0 & 0 & ... & 0 \\ 0 & 0 & w^2 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & w^{N-1} \end{bmatrix}$$

- $P_{2^{n+1}}$ is a shuffle transform defined by setting its j, k entry via the formula

$$(P_{2^{n+1}})_{j,k} = \begin{cases} 1, & \text{if } 2(j-1) = k-1 \\ 1, & \text{if } 2(j-1-2^n)+1 = k-1 \\ 0, & \text{otherwise.} \end{cases}$$

We do not derive this here but its effect is to move the vector entries with odd numbered indices to the front, followed by the even ones. For example,

$$P_{2^2} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_3 \\ v_2 \\ v_4 \end{bmatrix}$$

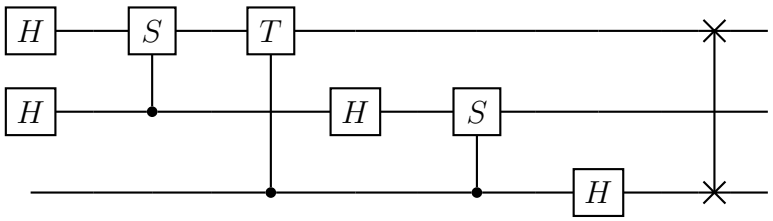
where

$$P_{2^2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Given this recursive decomposition, we can break a QFT down into smaller and smaller gates. This is seen in the relatively simple structure of the circuit

5.6.1.5 The Circuit

The $\mathbf{QFT}_3$ circuit can be constructed entirely from H, S, T and SWAP gates as



5.6.2 Factoring

Now we examine some of the mathematics of factoring and focus in on one famous algorithm. This is Shor's algorithm, named after its discoverer, mathematician Peter Shor. It can factor large integers almost exponentially faster than classical methods given a sufficiently large and powerful quantum computer.

In

$$-60 = (-1)\text{x}2^2\text{x}3\text{x}5$$

the 1 is a unit and 2, 3, and 5 are examples of prime numbers. They are irreducible in the sense that their only factors are 1 and themselves.

While it can be hard to factor large integers, it is not simply the size that makes it difficult.
The number

$$10^{250000000} = 2^{250000000}\text{x}5^{250000000}$$

is bigger than the above prime but is trivially factorable.

When we factor an N in Z greater than 1 we express it as a product

$$N = p_1 p_2 ... p_{n-1} p_n$$

for some integer $n >= 1$ with all the p_j prime numbers. We sort the primes from smallest to largest, which means $p_j <= p_k$ if $j < k$. Some primes may be repeated more than once. If, for example, 11 appears in the factorization five times then we say 11 has multiplicity 5. The notation we use to show that a p_j is a factor of N is $p_j|N$. Read this as "p_j divides N."

5.6.3 Phase Estimation

Be an N by N square matrix with complex entries. The solutions of λ of the equation

$$\det(U - I_N\lambda)$$

are the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_N$ of U. Some of the λ_j may be equal. If a particular eigenvalue λ shows up k times among the N, we say λ has multiplicity k.

Each eigenvalue λ_j corresponds to an eigenvector v_j so that

$$Uv_j = \lambda_j v_j$$

can take each v_j to be a unit vector.

When U is unitary, we can say even more: each λ_j has absolute value 1 and so can be represented as

$$\lambda_j = e^{2i\pi\phi} \text{where} 0 <= \phi_j <= 1.$$

It is the product $2\pi\phi$ that is the full radian measure of the rotation.

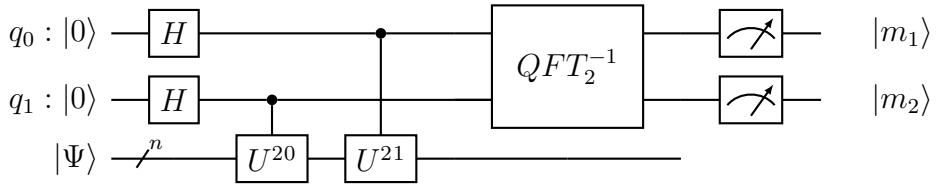
We are looking for a fractional approximation to ϕ. By increasing the number of qubits we use in the circuit, we can get a closer fit to ϕ.

Suppose U, $|\Psi\rangle$ and $e^{2i\pi\phi}$ are as above. By applying U multiple times

$$\begin{aligned} U|\Psi\rangle &= e^{2i\pi\phi|\Psi\rangle} \\ U^2|\Psi\rangle = U(e^{2i\pi\phi|\Psi\rangle}) &= e^{(2+2)i\pi\phi} \\ U^3|\Psi\rangle = U(e^{(2+2)i\pi\phi|\Psi\rangle}) &= e^{(2+2+2)i\pi\phi} \\ &... \\ U^k|\Psi\rangle &= e^{2ki\pi\Psi} \end{aligned}$$

Let m be the number of bits we need to get us as close as we need to the original phase. We use m qubits on the upper portion of the circuit, the "upper register."

When $m = 2$, our phase estimation circuit is



Continuing with our 2-qubit case, the state of the upper register before applying $QFT^{[}-1]_2$ is

$$\frac{1}{2}(|0\rangle + e^{2\pi 2^1 \phi i}|1\rangle) \otimes (|0\rangle + e^{2\pi 2^0 \phi i}|1\rangle) = \frac{1}{2}\sum_{j=0}^{2^2-1} e^{2\pi j \phi i}|j\rangle_2$$

In the 2-qubit case,

$$QFT_2 = \frac{1}{\sqrt{4}}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

because $w = e^{\frac{2\pi i}{2^2}} = i$ is a primitive fourth root of unity. Since this matrix is unitary, its inverse is its conjugate transpose

$$QFT_2^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

In our circuit, $a_k = \frac{1}{2}e^{2\pi\phi k i}$. Putting all this together, the final upper register before measurement is

$$\frac{1}{2^2}\sum_{j=0}^{2^2-1}\sum_{k=0}^{2^2-1} e^{2\pi\phi k i}e^{-\frac{2\pi j k i}{2^2}}|j\rangle_2 = \frac{1}{2^2}\sum_{j=0}^{2^2-1}\sum_{k=0}^{2^2-1} e^{-\frac{2\pi k i}{2^2}(j-2^2\phi)}|j\rangle_2$$

Let's state why we have some of the elements in these messy expressions that we do:

1. ϕ is the phase we want to estimate.
2. The $|j\rangle_2$ are the standard bass kets.
3. The sums involving the $|j\rangle_2$ are from superpositions.
4. The exponential forms involving e are roots of unity or other complex numbers with absolute value 1.
5. The QFT_n^{-1} introduces the minus sign into the exponent while using QFT_n would not have.

The general form of the last expression for m qubits instead of 2 is

$$\frac{1}{2^m}\sum_{j=0}^{2^m-1}\sum_{k=0}^{2^m-1} e^{-\frac{2\pi ki}{2^m}(j-2^m\phi)}\,|j\rangle_2$$

Including the top register and the bottom register (which is Ψ), the full pre-measurement state at the end of the circuit is

$$\frac{1}{2^m}\sum_{j=0}^{2^m-1}\sum_{k=0}^{2^m-1} e^{-\frac{2\pi ki}{2^m}(j-2^m\phi)}\,|j\rangle_2 \otimes |\Psi\rangle$$

The probability amplitudes are

$$\frac{1}{2^m}\sum_{k=0}^{2^m-1} e^{-\frac{2\pi ki}{2^m}(j-2^m\phi)}$$

for the expression to the left of $\otimes|\Psi\rangle$. Now let's get an estimate for ϕ. We do so by finding a good rational approximation between 0 and 1.

Starting with ϕ, multiply it by 2^m and choose the closest c in Z that is within $\frac{1}{2}$ of $2^m\phi$. To be concrete, choose

$$c = |2^m\phi + \frac{1}{2}|$$

Let

$$d = |\frac{2^m\phi - c}{2^m}| = |\phi - \frac{c}{2^m}|$$

Then $0 <= 2^m d <= \frac{1}{2}$. We are looking for $\frac{c}{2^m}$ to be a good approximation to ϕ. In fact, we want the measured state of the upper register to be $|c\rangle$. By increasing m, the number of qubits in each of the registers, we can get a more accurate approximation.

This is the crux of what the algorithm does, find $|c\rangle$ as the measured standard basis ket in the upper register as a good approximation to $2^m\phi$.

Given c and d, we rewrite the upper register quantum state as

$$\frac{1}{2^m}\sum_{k=0}^{2^m-1} e^{-\frac{2\pi ki}{2^m}(j-2^m\phi)}\,|j\rangle_m = \frac{1}{2^m}\sum_{k=0}^{2^m-1} e^{-\frac{2\pi ki}{2^m}(j-c)}e^{2\pi dki}\,|j\rangle_m$$

The probability of getting $|c\rangle$ is the square of the absolute value of the probability amplitude when $j = c$. That is,

$$P(c) = |\frac{1}{2^m}\sum_{k=0}^{2^m-1} e^{-\frac{2\pi ki}{2^m}(c-c)}e^{2\pi dki}\,|j\rangle_m\,|^2 = \frac{1}{2^{2m}}|\sum_{k=0}^{2^m-1} e^{2\pi dki}|^2$$

If $d = 0$ the sum on the right is exactly 1 and the measured result of the upper register is $|c\rangle$ with probability 1. This happens when $\phi = \frac{c}{2^m}$ is a rational number.

All is not lost if $d \neq 0$. In that case, the circuit returns the correct answer with probability $\frac{4}{\pi^2} \approx 0.405$.

This means we must run the circuit multiple times to get the correct answer. If you run the circuit 28 times, the probability of never having gotten the correct answer is less than 10^{-6}.

5.6.4 Order And Period Finding

Consider the function a^k on whole numbers k for a fixed a in N greater than 1. For example, if $a = 3$, then the first 12 values are

$$3^0 = 1,\ 3^1 = 3,\ 3^2 = 9,\ 3^3 = 9$$
$$3^4 = 81,\ 3^5 = 243,\ 3^6 = 729,\ 3^7 = 2187$$
$$3^8 = 6561,\ 3^9 = 19683,\ 3^10 = 59049,\ 3^11 = 177147$$

If we instead use modular arithmetic, 3^k cannot get arbitrarily large. For example, modulo $M = 13$, the values we get are

$$1 = 3^0 \text{mod } 13, 3, 9, 1, 3, 9, 1, 3, 9, 1, 3, 9, 1$$

Working modulo $M = 16$ yields

$$1, 3, 9, 11, 1, 3, 9, 11, 1, 3, 9, 11, 1, 3, 9, 11$$

Finally, for modulo $M = 22$ we get

$$1, 3, 9, 5, 15, 1, 3, 9, 5, 15, 1, 3, 9, 5, 15, 1, 3, 9, 5, 15, 1, 3$$

In each case, the sequence starts repeating. That is, the sequences, and hence the functions, are periodic. If we define $f_x(x) = a^x \mod M$ for a co-prime to M, then the smallest positive integer r such that $f_a(x) = f_a(x + r)$ for all x is called the period of f_a.

For $M = 13$ in the first example, the period is $r = 3$. For $M = 16$, $r = 4$. In the final example with $M = 22$, the period is $r = 5$.

Given such an a as above, we can look at all $a^1, a^2, a^3, ...$ modulo M and ask: what is the smallest r in N, if it exists, such that $a^r \equiv 1 \mod M$? If there is such an r, it is called the order of a mod M. Then $a^{x+r} \equiv a^x \mod M$ by multiplication by a^x. Thus r is also the period of f_a. For this reason, the "period finding problem" is equivalent to the "order finding problem".

In the rest of this section we develop a hybrid quantum-classical algorithm to find the order of such an a, with a few conditions.

This is an important algorithm because one of its applications is integer factorization. With such an r in hand, and assuming it is even,

$$a^r \equiv 1 \bmod M \rightarrow a^r - 1 \equiv 0 \bmod M \rightarrow (a^{\frac{r}{2}} - 1)(s^{\frac{r}{2}} + 1) \equiv 0 \bmod M.$$

Given a good a, a good even r, and Euclid's algorithm, we might be able to find a factorization of M.

Let

$$l_{\text{bits}} = [\log_2 M]$$

be the number of bits we need to represent M. For example, if $M = 7$ then $l_{\text{bits}} = 3$. For $M = 64$, we use $l_{\text{bits}} = 6$.
Now set

$$l_e = 2l_{\text{bits}} + 1 + [\log_2(2 + \frac{1}{2\epsilon})]$$

for some very small ϵ in R.

We use phase estimation with some post-processing using continued fractions to compute the order r of a modulo M. For $0 <= j <= r - 1$, we find the approximations to the phase $\phi_j = \frac{j}{r}$ accurate to $2l_{\text{bits}} + 1$ bits with probability greater or equal to $\frac{1-\epsilon}{r}$. The smaller our ϵ, the larger our l_ϵ is.

Since we are finding periods of functions or orders of numbers modulo M, we need to be able to compute $a^x \bmod M$ in a quantum way.

For a binary string of length l_{bits}, that is, y is in $0, 1^{l_{\text{bits}}}$, define

$$\delta_{j,k} = \begin{cases} |ay \bmod M\rangle, & \text{if } 0 <= y <= M \\ |y\rangle, & \text{if } M <= y <= 2^{l_{\text{bits}}} \end{cases}$$

Remember that a is coprime to M, which means they share no non-trivial factors. This is the same as saying gcd(a, M) = 1 and so there exists b and c in Z so that $ab + Mc = 1$. Looking at this modulo M,

$$1 = ab + Mc = ab \bmod M$$

Thus a is invertible modulo M with $a^{-1} = b$.

When I write an expression like $|j \bmod M\rangle$ it is shorthand for $|j \bmod M\rangle_{l \text{ bits}}$. The $|y\rangle$ are the computational basis vectors in a $2^{l_{\text{bits}}}$ vector space over C. U is a $2^{l_{\text{bits}}}$ by $2^{l_{\text{bits}}}$ square matrix. The lower right $2^{l_{\text{bits}}} - M$ submatrix is the identity matrix. The upper left M by M submatrix is a permutation matrix because a is invertible modulo M. All other matrix entries are zero. Hence all of U is a permutation consisting of 0s and 1s and so is unitary.

If we apply U multiple times, we get the below multiplication and hence exponention by repeated squaring

$$U^2 |y\rangle = UU |y\rangle = U |ay \bmod M\rangle = \left|a^2 y \bmod M\right\rangle$$

As we know, for any non-negative integer z with binary representation $z_{k-1}z_{k-2}...z_1z_0$ with z_0 the least significant bit,

$$a^z = a^{z_{k-1}2^{k-1}} \text{ x } a^{z_{k-2}2^{k-2}} \text{ x ... x } a^{z_1 2^1} \text{ x } a^{z_0 2^0}$$

We can define

$$|z\rangle |y\rangle \rightarrow |z\rangle U^{z_{k-1}2^{k-1}} \text{ x } U^{z_{k-2}2^{k-2}} \text{ x ... x } U^{z_1 2^1} \text{ x } U^{z_0 2^0} = |z\rangle |a^z y \bmod M\rangle$$

In practice we will allow k to be as large as l_ϵ.

With these observations and calculations, we know how to do quantum modular exponentiation. Like many such quantum subroutines, we may need additional qubits for the computations outside the main circuit description. We can do modular exponentiation in $O(l_{\text{bits}^3})$ gates using algorithms similar to the simple classical versions.

We have one last thing to observe about U before we move on to the circuit for order finding and its analysis. With r the order of a modulo M, the kets

$$|w_j\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2kji\pi}{r}} \left|a^k \bmod M\right\rangle$$

are eigenvectors of U for $0 <= j <= r$. If we apply U to each of these expressions:

$$U\,|w_j\rangle = U(\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2kji\pi}{r}} \left|a^k \bmod M\right\rangle) = e^{\frac{2ij\pi}{r}}\,|w_j\rangle$$

Even though we use r explicitly in the above equations, remember that we do not know what it is yet! We have shown only that the eigenvalues corresponding to the eigenvectors $|w_j\rangle$ are $e^{2ji\pi}$ as j goes from 0 to $r-1$. We use phase estimation to get at those eigenvalues and hence r.

To set up our circuit, we need two quantum registers. Since we use phase estimation, the first register needs enough qubits to get us the accuracy we need. This is l_ϵ. The number of qubits in the second register is l_{bits}.
The circuit for the quantum portion of the order finding algorithm was drawn earlier. The part of the circuit that is labeled as U_{PE} is the core of the setup for the phase estimation. Referring back to the general estimation circuit drawn earlier, U_{PE} needs to prepare an eigenvector in the second register and handle the controlled U^{2^j} gates.

Unfortunately, there is nothing that allows us to create

$$|w_j\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2kji\pi}{r}} \left|a^k \bmod M\right\rangle$$

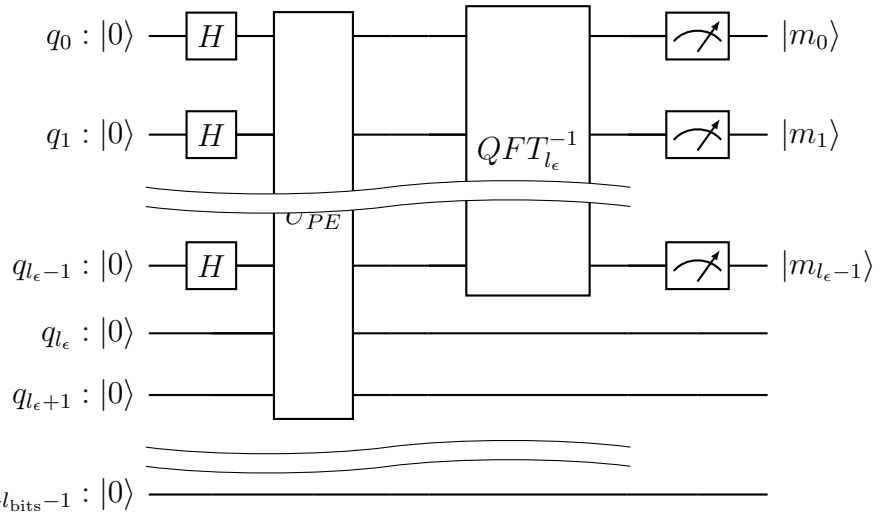
directly. The phases ϕ_j we wish to estimate are $\frac{j}{r}$.
When in a quandary in a quantum algorithm, create superposition! Consider

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

The $\frac{1}{\sqrt{r}}$ out front is the common probability amplitude of all the kets. Squaring this and multiplying with r gives us the required value of 1. Expanding the expression,

$$\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}|w_j\rangle = \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{-\frac{2kji\pi}{r}}\left|a^k \bmod M\right\rangle = |1\rangle_{l_{\text{bits}}}$$

Here is the order finding circuit:



Unlike the phase estimation algorithm, we are using a superposition of eigenvectors instead of a single one as the input to the lower quantum register. We're not done, because what comes out of our circuit after applying the inverse QFT requires more work to get to r. In the next section, we use continued fractions to get us closer to the factorization.

5.6.5 The continued fraction part

We chose l_ϵ so that for $0 <= j <= r-1$ we can find approximations ϕ_j to the phases $\frac{j}{r}$ of U's eigenvalues accurate to $2l_{\text{bits}}+1$ bits with probability $>= \frac{1-\epsilon}{r}$. j satisfies $0 <= j <= r$ where each j is as probable as another.

This is another way of saying that the j we get are uniformly distributed.

The output of the phase estimation algorithm after the measurement is

$$\phi_j = \frac{c_j}{2^\epsilon} \approx \frac{j}{r}$$

We have done a lot of quantum work and we have in hand a rational runmber $\frac{c_j}{2^\epsilon}$ approximation to $\frac{j}{r}$. We want to get to $\frac{j}{r}$ and then r.

If $gcd(c_j, 2^\epsilon) \neq 1$ then we try again because the fraction is not in the reduced form. If c_j and 2^ϵ are coprime, we test if

$$a^{2^\epsilon} \equiv 1 \bmod M$$

If this succeeds, we take $r = 2^\epsilon$ and we are done. c_j and 2^ϵ will be coprime if and only if c_j is odd, and there are many such candidates less than 2^ϵ.

We might just get a bad result with $\frac{c_j}{2^\epsilon}$ being, essentially, garbage. In this case, we can try again or decrease ϵ, thereby increasing l_ϵ.

We now employ this result about approximations and continued fractions:

Let $\frac{a}{b}$ be a rational number in reduced form. Every reduced rational number $\frac{c}{d}$ that satisfies

$$\left|\frac{a}{b} - \frac{c}{d}\right| < \frac{1}{2d^2}$$

is a convergent of $\frac{a}{b}$.

This statement is quite powerful: given two rational numbers, if the second is sufficiently close to the first, then the second is a convergent of the first. Reworking this for our situation gives the following result.

By our choice of l_{bits} and l_ϵ, every reduced rational number $\phi_j = \frac{c_j}{2^\epsilon}$ that satisfies

$$\left|\phi_j - \frac{j}{r}\right| = \left|\frac{c_j}{2^\epsilon} - \frac{j}{r}\right| < \frac{1}{2r^2}$$

is a convergent of $\frac{j}{r}$.

If $\frac{c_j}{2^\epsilon}$ is a good approximation to $\frac{j}{r}$ then it is accurate to $2l_{\text{bits}} + 1$ bits with probability $>= \frac{1-\epsilon}{r}$. What does this mean for some x to be accurate to some y for some number of bits b? Simply that

$$|x - y| <= \frac{1}{2^b} = 2^{-b}$$

Therefore,

$$\frac{1}{2 \text{ x } (2^{l_{\text{bits}}})^2} <= \frac{1}{2M^2} < \frac{1}{2r^2}$$

by our definition of l_{bits} and noting that $r <= M$.

Now we start computing the convergents for the known reduced fraction $\frac{c_j}{2^\epsilon}$. Among these will be $\frac{j}{r}$ in reduced form. We test the denominators of the convergents as candidates for r. When we find one that works, we are done.

The complexity of the overall algorithm is dominated by the quantum modular exponentiation and so is $O(l_{\text{bits}}^3)$

5.6.6 Shor's Algorithm

We now have the tools we need to sketch Shor's algorithm for factoring integers in polynomial time on a sufficiently large quantum computer.

The complete algorithm has both classical and quantum components. Work is done on both kinds of machines to get to the answer. It is the quantum portion that drops us down to polynomial complexity in the number of gates by use of phase estimation, order finding, modular exponentiation, and the Quantum Fourier Transform.

Let M be an odd positive number in Z that is not a power of a prime. It has a reasonable chance of being composite.
The following is the general approach to Shor's algorithm given M as above:

1. Choose a random number a such that 1 ¡ a ¡ M. Keep track of these values since we might need to repeat this step again.

2. Check if $gcd(a, M) = 1$. If not, we have found a factor of M and we are done. This is pretty unlikely but now we know that a and M are coprime: they have no integer factors in common.

3. Now find the non-zero order r of a mod M. This means that $a^r \equiv 1$ mod M. If r is odd, go back to step 1 and try again with a different a.

4. If r is even, we have

   $$a^r \equiv 1 \text{ mod } M \rightarrow a^r - 1 \equiv 0 \text{ mod } M \rightarrow (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \text{ mod } M$$

5. Now look at $gcd(a^{\frac{r}{2}} - 1, M)$ and, if necessary, $gcd(a^{\frac{r}{2}} + 1, M)$. If either of these does not equal 1, we have found a factor of M and we have succeeded.

6. . If both of these greatest common divisors are 1, we repeat all the above from step 1 with another random a. We continue to do this until we find a factor.

6 Physical Implementation

The qubits we make in the lab for research and those we will create for commercial use are physical hardware devices. As such, they are subject to noise from the environment, electronic components, and manufacturing choices. Hardware improvements decrease the disturbances, but software and system ones can too. The long-term goal is to have fully error corrected, fault-tolerant quantum computing devices.

6.1 Physical Qubits

When you build a quantum computer, the fundamental physical implementations of qubits aren't as perfect as logical qubits. Such a qubit, called a "physical qubit," starts to lose its ability to hold onto a state after what is called its "coherence time." We also say that the qubit is "decohering".

It's a goal of quantum computing researchers and engineers to delay the decay of a physical qubit's quantum state as long as possible. Since the decay is inevitable, a goal of fault tolerance and error correction is to handle and fix the effects of the qubits' decoherence throughout the execution of a circuit.

Small errors creep into the state when gates are applied. After too many gates, you either pass the coherence time and the qubit becomes unreliable, or the errors accumulate to such a degree that further use and measurement is too inaccurate for useful computing.

Two simultaneous and connected goals when constructing a quantum computer are to increase coherence time and reduce errors.

You may see or hear the term "short depth circuit." How many gates are in such a circuit? There is no hard and fast number, though expect it to increase over time. A reasonable working definition of a short depth circuit is one you can run and from which you can get useful results before decoherence sets in and errors overwhelm the computation. When you are working with quantum computing hardware, make sure you can see the current operating statistics about coherence times and errors.

There are some requirements for the implementation of qubits to quantum computing. We can discuss them as:

Scalable Physical System

In our phsyical system that we manufacture for quantum computing, we need to build a qubit that has two clearly delineated states, $|0\rangle$ and $|1\rangle$. If these represent energy states, other states may be possible, but we must control the qubit to keep it at either $|0\rangle$ or $|1\rangle$.

The qubit must be able to move into a true superposition of $|0\rangle$ and $|1\rangle$ that obeys the rules with amplitudes and probabilities. As we add more qubits, we must be able to entangle the qubits either directly by physical means or indirectly through a sequence of gates in a circuit.

Initializing Qubits

We must be able to initialize a qubit to a known initial state with very high probability. This is called "high fidelity state preparation."

Since it is common for algorithms to begin with qubits in $|0\rangle$, this is a good choice. If $|1\rangle$ is a better choice for the technology, applying an X gate after initialization gives an equivalent effect.

Long Decoherence

Decoherence causes a qubit to move from a desired quantum state to something else. If too much decoherence happens, the state is random and useless.

The qubit must have a long enough coherence time so that enough quantum gates can be executed to implement an algorithm that does something useful. If you have a long coherence time but your gates take a long time to execute, that may be equivalent to a short coherence time but with fast gates.

Universal Set Of Gates

We looked at how gates can be created from other, more primitive ones. The gates that are native to a particular qubit technology may not look anything like the ones we have seen in this book, but as long as they can be composed into a standard collection, practical algorithms can developed, implemented, and deployed.

Measurement Capability

We must be able to reliably force the qubit into one or the other of two orthonormal basis states, and these are often $|0\rangle$ and $|1\rangle$. The error rate of this operation must be low enough to allow for useful computation: if we get the wrong measurement answer 50% of the time, all the previous work on executing the circuit is lost. This is called "high fidelity readout."

If we move the qubit to $|+\rangle = \frac{\sqrt{2}}{2}|0\rangle + \frac{\sqrt{2}}{2}|1\rangle$, it should measure to either $|0\rangle$ or $|1\rangle$ with 0.5 probability each. The same is true for quantum states with known probability amplitudes before measurement.

6.2 Light And Photons

Light has both wave-like and particle-like characteristics. The fundamental unit of light is called a photon. It has no charge and, according to theory, it has no mass.

If the waves overlap, their amplitudes, positive or negative, add to give a new combined amplitude. If both amplitudes are the same sign, we get constructive interference. If they differ, we get destructive interference. If the new amplitude is zero, we have complete destructive interference.

For a given wave with a repeating period as we have shown, we measure the various points along the horizontal from the beginning of the period to the end in radians and call this the phase ϕ. It takes on values from 0 to 2π. When two waves have the same shape, the same amplitude, and the same frequency but are possibly offset horizontally, we say that the waves are coherent. The offset $\delta\phi$ is the phase difference or phase offset.

6.2.1 Decoherence

There are three important measurements that quantum computing researchers use to measure coherence time T_1, T_2 and its cousin T_2^*. Let's begin with T_1. They are single qubit measurements and so we can use the Bloch sphere to discuss them. Their use goes back to Felix Bloch's work on nuclear magnetic resonance in the 1940s.

6.2.1.1 T_1

T_1 goes under several names, all of them connected to the physics of various quantum processes like relaxation time, thermal relaxation, longitudinal relaxation, spontaneous emission time, amplitude damping and longitudinal coherence time.

It is related to the loss of energy as the quantum state decays from the higher energy $|1\rangle$ state to the $|0\rangle$ from state. This energy is transmitted to, or leaked into, the environment and lost from the qubit. T_1 is measured in seconds or some fraction thereof such as miliseconds.

For the computation of T_1, the qubit is moved via an X gate from $|0\rangle$ to $|1\rangle$. The decay toward the lower energy state $|0\rangle$ is exponential and follows the rule

$$P(|1\rangle) = e^{\frac{-t}{T_1}}$$

for some constant T_1. Informally, the larger the value of T_1, the longer the qubit is staying close to $|1\rangle$.

6.2.1.2 T_2 and T_2^*

If T_1 concerned itself with going from the north pole to the north, T_2 adds in the extra element of what is happening at the equator. As a reminder, when I say "the equator," I mean the intersection of the xy-plane with the Bloch sphere.

6.2.2 Error Correction

For the quantum situation, the No-Cloning Theorem says that we can't copy the state of a qubit, and so traditional repetition is not available.

What we can do is entanglement, and it turns out that this is powerful enough when combined with aspects of traditional error correction to give us quantum error correction, or QEC.

How can we go from $||Psi\rangle = a\,|0\rangle + b\,|1\rangle$ to $a\,|000\rangle + b\,|111\rangle$? As you start thinking about such questions, there are two good starting points: "would applying an H change the situation into something I know how to handle," and "how might a CNOT and entanglement affect things?" How can we fix bit flips?

6.2.2.1 Correcting Bit Flips

In the classical case of a bit, only one thing can go wrong: the value changes from 0 to 1 or vice versa. Of course, noise may cause multiple bits to change, but for one bit there is only one kind of error.

In the quantum case, a bit flip interchanges $|0\rangle$ and $|1\rangle$ so that a general state $a\,|0\rangle + b\,|1\rangle$ becomes $a\,|1\rangle + b\,|0\rangle$ instead. This is what an X does, but when we are thinking about noise and errors, we are saying that a bit flip might happen, not that it definitely does.

If we absolutely knew that an X was applied, we could just do another one and fix the problem. Therefore we need to get more clever.

6.2.2.2 Correcting Sign Flips

A sign flip is a π phase error that switches $a\,|0\rangle + b\,|1\rangle$ and $a\,|0\rangle - b]\,|1\rangle$. A Z gate does this.

Changing between the usual computation basis of $|0\rangle$ and $|1\rangle$ and the Hadamard basis of $|+\rangle$ and $|-\rangle$ has the extremely useful property of interchanging bit flips and sign flips. To fix possible sign flips, we only have to insert some H gates into the above circuit.

6.2.2.3 Error Correcting Codes

Some error correcting codes can be listed as:

- Peter Shor's 9-qubit-code, a.k.a. the Shor code, encodes 1 logical qubit in 9 physical qubits and can correct for arbitrary errors in a single qubit.
- Andrew Steane found a code that does the same with 7 instead of 9 qubits, see Steane code.

- Raymond Laflamme and collaborators found a class of 5-qubit codes that do the same, which also have the property of being fault-tolerant. A 5-qubit code is the smallest possible code that protects a single logical qubit against single-qubit errors.
- A generalisation of the technique used by Steane, to develop the 7-qubit code from the classical [7, 4] Hamming code, led to the construction of an important class of codes called the CSS codes, named for their inventors: A. R. Calderbank, Peter Shor and Andrew Steane. According to the quantum Hamming bound, encoding a single logical qubit and providing for arbitrary error correction in a single qubit requires a minimum of 5 physical qubits.
- A more general class of codes (encompassing the former) are the stabilizer codes discovered by Daniel Gottesman, and by A. R. Calderbank, Eric Rains, Peter Shor, and N. J. A. Sloane; these are also called additive codes.
- Two dimensional Bacon–Shor codes are a family of codes parameterized by integers m and n. There are nm qubits arranged in a square lattice.
- A newer idea is Alexei Kitaev's topological quantum codes and the more general idea of a topological quantum computer.
- Todd Brun, Igor Devetak, and Min-Hsiu Hsieh also constructed the entanglement-assisted stabilizer formalism as an extension of the standard stabilizer formalism that incorporates quantum entanglement shared between a sender and a receiver.

References

[1] IBM. Ibm quantum composer docs, 2021.

[2] IBM. Ibm quantum lab docs, 2021.

[3] et al Johnson, Eric R. *Programming Quantum Computers: Essential Algorithms and Code Samples*. O'Reilly Media, Incorporated, 2019.

[4] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[5] QWorld. Qbronze education material, 2021.

[6] Robert S Sutor. *Dancing with Qubits: How Quantum Computing Works and How It May Change the World*. Packt Publishing, 2019.