

Quantum Information and Computing

Summer of Science 2020

Mahadevan Subramanian

Roll no. 190260027

Mentor: Thariq Shanavas



Contents

1	Introduction	2
1.1	Turing Machine	2
1.2	The set of computational problems	2
1.3	Reversible circuits	3
1.4	Maxwell's Demon	4
2	Quantum Mechanics	6
2.1	Postulates of Quantum Mechanics	6
2.2	Qubits	6
2.2.1	What is a Qubit	6
2.2.2	Bloch Sphere	7
2.2.3	No-Cloning theorem	8
2.3	Quantum measurement	8
2.3.1	Projective Measurements	8
2.3.2	Heisenberg's uncertainty principle	8
2.3.3	POVM Measurements	9
2.4	Density matrices	9
2.4.1	Partial Trace and Reduced density operator	10
2.4.2	Schmidt Decomposition	10
2.5	EPR and the Bell inequality	11
3	Quantum Circuits	13
3.1	Quantum Gates	13
3.1.1	Controlled gates	14
3.1.2	Measurement in circuits and Quantum Teleportation	16
3.2	Universality in Quantum Computing	17
3.2.1	Two level unitary operators	17
3.2.2	Approximating Unitary operators	18
3.3	Simulation of Quantum systems	19
4	Algorithms for Quantum Computing	21
4.1	The Quantum Fourier Transform	21
4.1.1	Phase estimation	21
4.1.2	Order finding and factoring	22
4.1.3	Period finding	24
4.1.4	Discrete logarithms	24
4.1.5	Hidden subgroup problem	25
4.1.6	Deutsch-Josza Algorithm and Simon's Algorithm	25
4.2	Search algorithms	26
4.2.1	Grover's Search Algorithm	26
4.2.2	Hamiltonian for search algorithm	28
4.2.3	Quantum Counting	29
4.2.4	Searching an unstructured database	30
4.2.5	Optimality of the algorithm and Black box limits	30
4.2.6	Interesting applications of Grover's algorithm	32
	References and other links	34

Chapter 1

Introduction

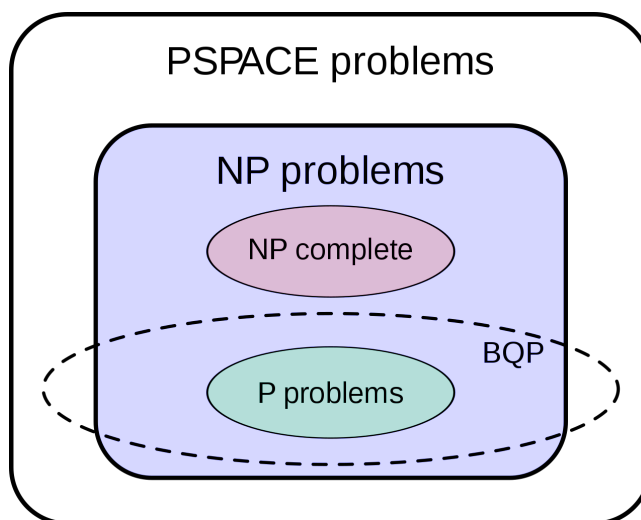
In this chapter I will be discussing some important aspects of computer science and logic which builds up to the quantum computer

1.1 Turing Machine

A Turing machine is an abstract machine, which manipulates symbols on a strip of sufficiently large length according to a table of rules. Given any algorithm a Turing machine can be built to simulate it. According to the strong Church-Turing thesis, a function on the natural numbers can be calculated by an effective method if and only if it is computable by a Turing machine. Essentially it makes claims that all polynomial time computations can be run on a Turing machine since computable generally refers to polynomial time complexity. However it was found out that randomized algorithms can carry out certain efficient computations which cannot be run on a Turing machine clearly violating the strong Church-Turing thesis. This was then modified to being computable on a probabilistic Turing machine. Now the next step was to actually make a computer that could essentially carry out randomized algorithms and here is where the idea for quantum computation was born

1.2 The set of computational problems

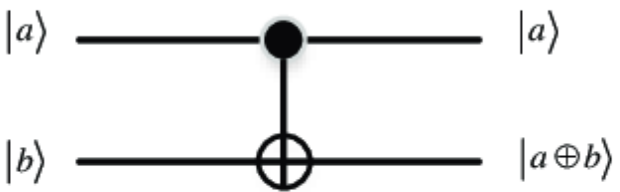
The complete set of computational problems are classified in different sets based on their complexity. PSPACE denotes the set of problems that can be solved using a computer of 'small' size but can do long computations. P denotes the set of problems that can be solved in polynomial time on a deterministic Turing machine an example of which is the shortest path algorithm. NP denotes the set of problems which can be solved in polynomial time by a non deterministic Turing machine essentially requiring a "lucky guess". The set of P lies inside the set NP. NP complete represents a set of NP problems which if solved can essentially extend to the rest of the NP set. PSPACE is believed to include both NP and P within it but it is not properly been proven yet. Factoring is believed to lie in NP but not in NP complete and the interesting thing is that Shor's algorithm can do factoring in polynomial time using a quantum computer. BQP represents problems that can be solved in polynomial time using a quantum computer. We don't yet know the exact extent of this set but it is believed to include P and and also some parts of NP and also problems that lie in PSPACE but not NP.



1.3 Reversible circuits

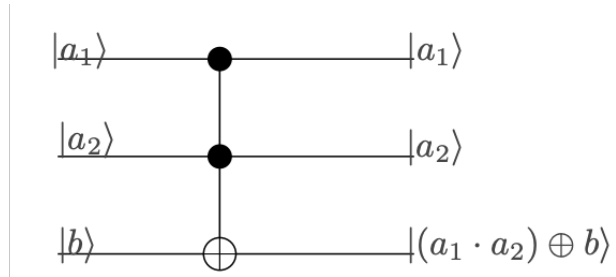
An important aspect of quantum computation is that the logical circuits use different kinds of gates and essentially are reversible circuits which work with reversible logic. Reversible logic simply means that if we know the output we also know the input hence the input and output have a one to one mapping. Right off the bat we know that gates like AND can't be used for this since knowing the output doesn't always tell us enough to know the inputs. In a reversible logic circuit the inputs and outputs are equal in number. One can see that in logic circuit like one of an AND gate essentially has less randomness in the output as to the input. An important thing to understand is that even though this entropy is in terms of information, it still has an energy price to pay and hence there is a considerable amount of energy consumed in all devices like computers and mobile phones, etc. to repay this entropy loss debt and this is the Von Neumann entropy. In an ideal reversible circuit there is no loss of energy in these terms.

An important gate in reversible circuits is the CNOT gate

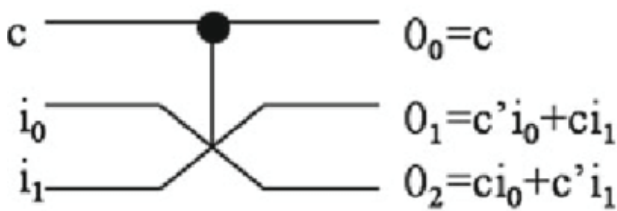
$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$


The diagram shows a CNOT gate with two horizontal lines representing qubits. The top line is labeled $|a\rangle$ on both ends. The bottom line is labeled $|b\rangle$ on the left and $|a \oplus b\rangle$ on the right. A control dot is on the top line, and a target circle with a plus sign is on the bottom line, connected by a vertical line.

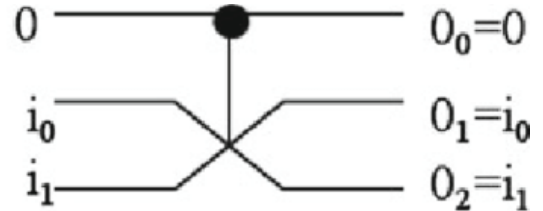
Apart from this is the Toffoli gate and the Fredkin gate described in the graphics below.



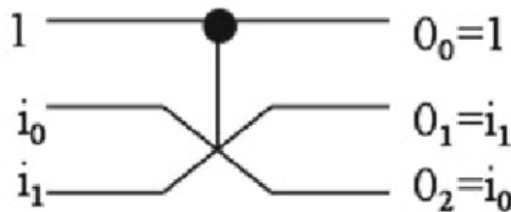
Toffoli gate



(a) Fredkin gate



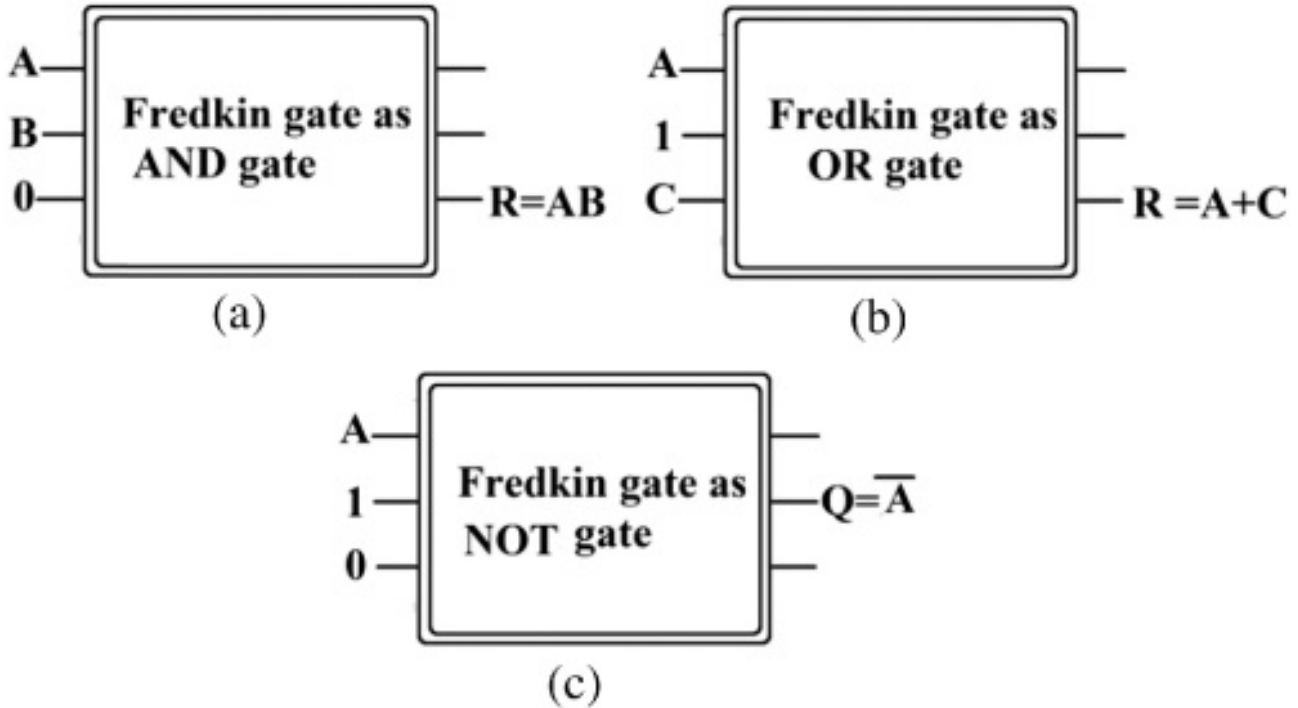
(b) Fredkin gate when $C=0$



(c) Fredkin gate with $c = 1$

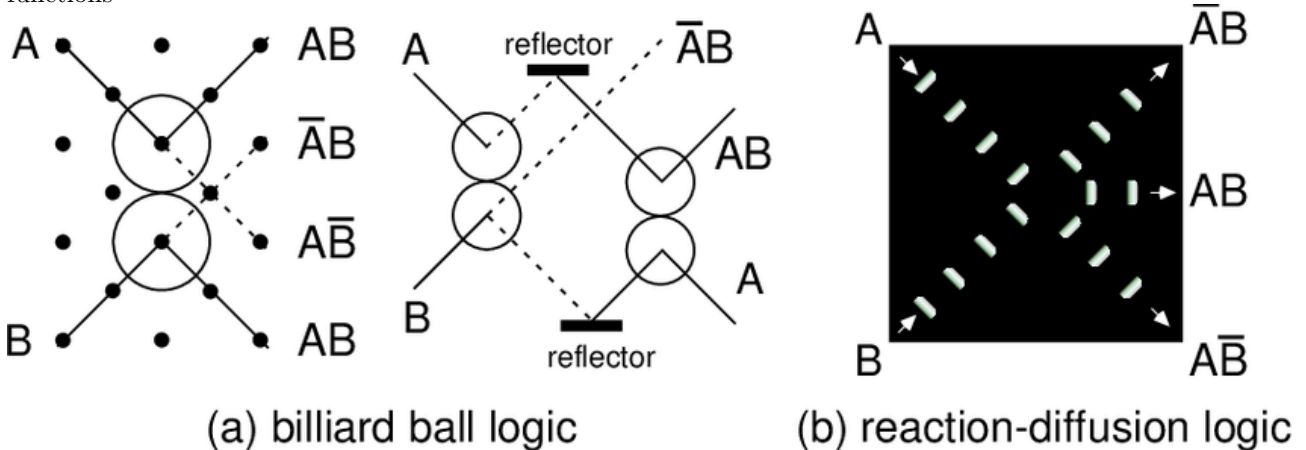
An important thing to note is that both these gates are universal i.e. we can reproduce any boolean function

purely just using only Toffoli gates and Fredkin gates. As seen below we can implement AND and NOT using Toffoli and Fredkin gates under the following setup



It can be proven that using AND gates and NOT gates and FANOUT one can implement all possible boolean functions. (NAND gates are also universal). All Quantum circuits use reversible logic since we define quantum evolution using unitary transforms which are reversible, hence requiring quantum circuits to also have reversible logic. This is of course broken upon measuring a qubit which will be discussed further ahead.

An interesting build of a reversible circuit is the "Billiard ball" model for computing. We essentially start of by considering two billiard balls A and B which for the sake of simplicity are perfectly elastic and everything in the given environment is ideal. Let A represent the boolean variable which is true if the ball A is approaching as shown in the figure and is false if it is not present there. Similarly, let B represent the boolean variable which is true if the ball A is approaching as shown in the figure and is false if it is not present there. One can see that if one were to observe the positions of the balls after some time depending on the values of A and B we will get different scenarios and the presence of a ball at these locations can be represented as the following boolean functions

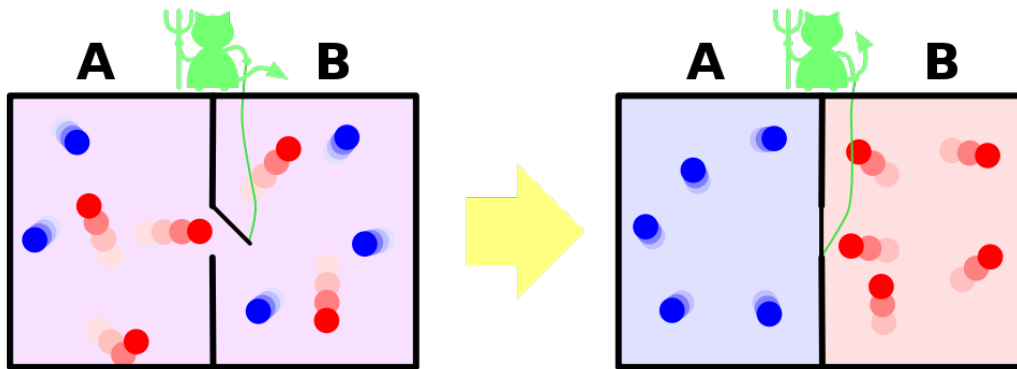


Interestingly there has been no computer built with the billiard ball model since an error in one stage will entirely cause the whole computation to fail and error management wouldn't be that too easy.

1.4 Maxwell's Demon

This is not directly related to computing in the obvious sense but a very interesting thought experiment nonetheless. There was a paradox described by Maxwell in which there is a box filled with gas particles with a middle partition which had a small rolling door. A small demon controls this door and only lets particles with

a high velocity pass through the door from one side. This act of the demon will decrease entropy despite the demon doing little to no work so this violated the second law of thermodynamics.



However an important thing here is that the demon performs measurement over every gas particle as it approaches the door hence he would have to store the information somehow. Since all information storage is finite the demon will eventually reach a point where he would have to erase the information and this actually accounts for the entropy imbalance since erasing information affects entropy by Landauer's principle.

This is a very interesting link between physics and the idea of information since this concept of entropy related to information plays a very important role as discussed in the previous section regarding reversible circuits.

Chapter 2

Quantum Mechanics

In this chapter I will discuss some basic concepts of quantum mechanics which are required for understanding quantum computing. In this report I will be mostly using the bra-ket notation.

2.1 Postulates of Quantum Mechanics

The following are the postulates of Quantum mechanics.

Postulate 1: The state of a quantum mechanical system is completely specified by the wavefunction $|\psi(\mathbf{r}, t)\rangle$.

Postulate 2: To every observable in classical mechanics, there corresponds a linear, Hermitian operator in quantum mechanics. For example, in coordinate space, the momentum operator \hat{P}_x corresponding to momentum p_x in the x direction for a single particle is $-i\hbar\frac{\partial}{\partial x}$.

Postulate 3: In any measurement of the observable associated with operator \hat{A} , the only values that will ever be observed are the eigenvalues a which satisfy $\hat{A}\Psi = a\Psi$. Although measurements must always yield an eigenvalue, the state does not originally have to be in an eigenstate of \hat{A} . An arbitrary state can be expanded in the complete set of eigenvectors of \hat{A} ($\hat{A}\psi_i = a_i\psi_i$) as $\Psi = \sum_i c_i\psi_i$, where the sum can run to infinity in principle. The probability of observing eigenvalue a_i is given by $c_i^*c_i$.

Postulate 4: The average value of the observable corresponding to operator \hat{A} is given by $\langle\hat{A}\rangle = \frac{\langle\psi|\hat{A}|\psi\rangle}{\langle\psi|\psi\rangle}$

Postulate 5: The wavefunction evolves in time according to the time-dependent Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$$

\hat{H} represents the Hamiltonian of the state and $|\psi\rangle$ represents the state of the system.

Postulate 6: The total wavefunction must be anti symmetric with respect to the interchange of all coordinates of one fermion with those of another. Electronic spin must be included in this set of coordinates. The Pauli exclusion principle is a direct result of this anti symmetry principle.

2.2 Qubits

2.2.1 What is a Qubit

A qubit is a form of information which essentially represents the wave function of a two level quantum mechanical system. These two levels are represented as $|0\rangle$ and $|1\rangle$ and the state is written as a linear combination of these two i.e. $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Here $\alpha, \beta \in \mathbb{C}$ with constraint $|\alpha|^2 + |\beta|^2 = 1$ for normalization purposes. Measurement of a qubit causes it to "collapse" onto one of the states $|0\rangle$ or $|1\rangle$ and the probability for it to be measured as $|0\rangle$ is $|\alpha|^2$ and for $|1\rangle$ it is $|\beta|^2$. For representation we represent these kets as two dimensional column vectors.

$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. So essentially $|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$

Qubits are meant to represent information but handling them is not as easy as handling classical bits due to the fact that they lose their information once measured and also they cannot be cloned. However there are many interesting things that can be done with qubits due to the nature of Quantum mechanics and most of their advantages stem from quantum entanglement.

An interesting analogue of qubits are dreams. Say you just had a dream while you slept and on waking up

you remember the details of your dream but you wouldn't be able to communicate the exact details of your dream since it is present by itself in a somewhat vague state. So once you try to describe it to someone you would essentially forget the original information and you dream collapses to the description you had given. Not exactly a perfect analogue but an interesting one nonetheless.

2.2.2 Bloch Sphere

A method used for visualization of qubits is the Bloch sphere representation. Since all qubits are written as $\alpha|0\rangle + \beta|1\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$. Since α and β are complex numbers it is possible to represent each qubit with three real numbers. An important point to understand here is that global phase change over a qubit doesn't affect it. Hence the qubits $\alpha|0\rangle + \beta|1\rangle$ and $e^{i\theta}(\alpha|0\rangle + \beta|1\rangle)$ are the same since there is no way to differentiate between them.

$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $|\alpha|^2 + |\beta|^2 = 1$ can be written as

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle \quad (2.1)$$

Note that since global phase is ignored we can assume α to take on real values. If we take a sphere centered at origin and assume $|0\rangle$ to represent $|+z\rangle$ and $|1\rangle$ to represent $|-z\rangle$ we get that the angle θ from eq 2.1 actually is the polar angle of the vector which connects the origin to the point which actually represents this qubit and the angle ϕ represents the azimuthal angle.

Using this we get $|+x\rangle = \frac{|0\rangle + |1\rangle}{2}$ and $|-x\rangle = \frac{|0\rangle - |1\rangle}{2}$. Also $|+y\rangle = \frac{|0\rangle + i|1\rangle}{2}$ and $|-y\rangle = \frac{|0\rangle - i|1\rangle}{2}$.

We often write $|+x\rangle$ as $|+\rangle$ and $|-x\rangle$ as $|-\rangle$. We can define certain operations over qubits which actually rotate the qubit about a certain axis by a certain angle. These are called single qubit gates.

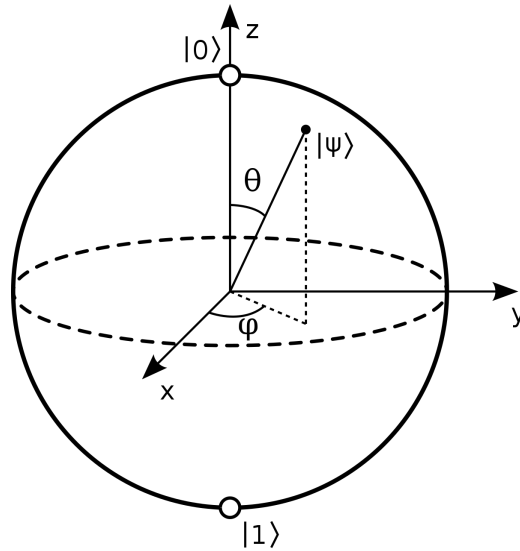


Figure 2.1: Bloch Sphere

Some important ones are the Pauli sigma matrices $\sigma_x, \sigma_y, \sigma_z$. Their eigenvalues are +1 and -1 and their eigenvectors are the corresponding axes like for σ_x the eigenvectors are $|+x\rangle$ with eigenvalue 1 and $|-x\rangle$ with eigenvalue -1. They essentially rotate by π about their respective axes. When represented in the $|0\rangle, |1\rangle$ basis

these are the Pauli sigma matrices: $\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, $\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$.

For performing rotation we define the rotation operators $R_x(\theta) = e^{-i\frac{\theta}{2}\sigma_x}$, $R_y(\theta) = e^{-i\frac{\theta}{2}\sigma_y}$, $R_z(\theta) = e^{-i\frac{\theta}{2}\sigma_z}$ where each of these respectively rotates by an angle of theta about the respective axes. Using this we can prove that rotation about a unit vector \hat{n} by an angle θ can be written as $R_{\hat{n}}(\theta) = e^{-i\frac{\theta}{2}\hat{n}\cdot\sigma}$. Here $\hat{n}\cdot\sigma = n_x\sigma_x + n_y\sigma_y + n_z\sigma_z$. When used in quantum circuits, $\sigma_x, \sigma_y, \sigma_z$ are written as X, Y, Z respectively and are used as gates. Also a very

frequently used gate the Hadamard gate is written as $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and essentially performs a rotation about the vector $(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}})$ by an angle π . Note that I am ignoring any global phase changes since they are irrelevant.

2.2.3 No-Cloning theorem

The no-cloning theorem states that it is impossible to create an identical copy of an arbitrary unknown quantum state. Essentially one cannot simply clone a qubit. The proof of this is as follows:

Let there be two slots A and B in a quantum computer. A is the data slot which has an unknown state $|\psi\rangle$ and B is the target slot where we wish to copy the given state so it is initially in a known state $|s\rangle$.

Let us define a unitary evolution U such that applying U to this given state makes a clone of the data slot to the target slot. Current state is,

$$|\psi\rangle \otimes |s\rangle \quad (2.2)$$

On applying unitary transform U ,

$$U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle$$

Suppose this cloning procedure were to work for two particular pure states $|\psi\rangle$ and $|\phi\rangle$, We have

$$U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle \quad (2.3)$$

$$U(|\phi\rangle \otimes |s\rangle) = |\phi\rangle \otimes |\phi\rangle \quad (2.4)$$

On taking the inner product of these two equations we get

$$\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^2 \quad (2.5)$$

On solving this equation, we get that either $|\psi\rangle = |\phi\rangle$ or $|\psi\rangle$ and $|\phi\rangle$ are orthogonal hence we cannot have a unitary cloning operation which can clone non orthogonal states simultaneously.

2.3 Quantum measurement

Quite possibly one of the most perplexing aspects of quantum mechanics has to be measurement.

2.3.1 Projective Measurements

When measurement is done of a state over a certain basis the wavefunction collapses onto one of the orthonormal states which belong to that basis. We can define there to be certain measurement operator M_m where the index m denotes which state it will collapse to. With this we get

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$$

$$|\psi'\rangle = \frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}$$

Here $|\psi'\rangle$ represents the state after measurement and p_m is the probability for it to collapse onto that state. Since these measurements only give us a final result over an orthonormal basis it is not possible to differentiate between non orthogonal states.

Also these measurement operators must follow the completeness property that $\sum_m M_m^\dagger M_m = I$.

2.3.2 Heisenberg's uncertainty principle

According to the Heisenberg's uncertainty principle, one cannot measure two non commuting operators to full accuracy. Its proof goes like this.

Let there be two Hermitian operators A and B and there is a state $|\psi\rangle$

Let $\langle\psi|AB|\psi\rangle = x + iy$ where $x, y \in \mathbb{R}$. Since A and B are Hermitian we have $\langle\psi|BA|\psi\rangle = x - iy$. Therefore

$$|\langle\psi|[A, B]|\psi\rangle|^2 + |\langle\psi|\{A, B\}|\psi\rangle|^2 = 4|\langle\psi|AB|\psi\rangle|^2 \quad (2.6)$$

By Cauchy-Shwarz inequality we have

$$|\langle\psi|AB|\psi\rangle|^2 \leq \langle\psi|A^2|\psi\rangle \langle\psi|B^2|\psi\rangle \quad (2.7)$$

on combining 2.7 with eq 2.6 we get

$$|\langle\psi|[A, B]|\psi\rangle|^2 \leq 4\langle\psi|A^2|\psi\rangle \langle\psi|B^2|\psi\rangle \quad (2.8)$$

Lets take two observable C and D , Let $A = C - \langle C \rangle$ and $B = D - \langle D \rangle$. On substituting this to eq 2.8 we get

$$\Delta(C)\Delta(D) \geq \frac{|\langle\psi|[C, D]|\psi\rangle|}{2} \quad (2.9)$$

And this is the Heisenberg uncertainty relation Here $\Delta(C)$ represents the standard deviation of the values of C . In its exactness the Heisenberg uncertainty relation states that if we have prepared a large number of identical states $|\psi\rangle$, and we were to measure the observables C and D over this system we would get multiple values for them however their standard deviations will follow this inequality. Hence we mustn't think of these as accuracy limitations but as uncertainty in the value we wish to observe.

2.3.3 POVM Measurements

Projective measurements follow in a way that once the state is measured it remains the same hence doing the same measurement will yield the exact same state with 100% probability. However there are forms of measurement which cannot be repeated to give the same state. For example if we were to measure the position of a photon using a silvered screen we end up destroying the photon.

POVM stands for positive operator valued measure. We define $E_m = M_m^\dagger M_m$ where M_m are the measurement operators. So we have $\sum_m E_m = I$ by the completeness property. We call E_m as the POVM elements. We also have $p(m) = \langle\psi| E_m |\psi\rangle$ Projective measurements are essentially a case of POVM where we have $M_m = \sqrt{E_m}$. An interesting thing to note is that we can use POVM elements to try to differentiate between non orthogonal states. Say we are given one of the two qubits $|\psi_1\rangle = |0\rangle$ and $|\psi_2\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$. Using just projective measurements we would not be able to conclude with complete certainty which state we have been given however say we define the following POVM elements

$$\begin{aligned} E_1 &= \frac{\sqrt{2}}{1 + \sqrt{2}} |1\rangle \langle 1| \\ E_2 &= \frac{\sqrt{2}}{1 + \sqrt{2}} (|0\rangle - |1\rangle)(\langle 0| - \langle 1|) \\ E_3 &= I - E_1 - E_2 \end{aligned}$$

Clearly this follows completeness property. One can see that if we get E_1 we can be sure that we received $|\psi_2\rangle$ and if we get E_2 we can be sure that we received $|\psi_1\rangle$. However we cant make any conclusion if we obtain E_3 . However we can safely conclude that our assumptions wont be wrong. This can be extended to any number of qubits say m by using $m + 1$ POVM elements by smartly choosing each E_i according to what makes each $|\psi_i\rangle$ different from the rest.

2.4 Density matrices

Density matrices are an important extension of a quantum state. An important property of a density matrix is that it stores the complete information of the state. We represent it by ρ . Suppose we have an ensemble of states $\{p_i, |\psi_i\rangle\}$ where p_i is the probability of getting $|\psi_i\rangle$, then we represent density matrix as

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

When describing evolution of density matrix for some unitary evolution U over $|\psi\rangle$ we get the following (ρ' represents evolved density matrix)

$$\rho' = \sum_i p_i U |\psi_i\rangle \langle \psi_i| U^\dagger = U \rho U^\dagger$$

An important concept in density matrices is the distinction between pure states and mixed states. For a pure state we know its exact $|\psi\rangle$ hence the density matrix is simply $\rho = |\psi\rangle \langle \psi|$. Otherwise, ρ is in a mixed state, it is said to be a mixture of the different pure states in the ensemble for ρ . An obvious fact is that $\text{tr}(\rho) = 1$ since the wavefunction has to be normalised. For a pure state we get $\text{tr}(\rho^2) = 1$ but for a mixed state $\text{tr}(\rho^2) < 1$.

For the case where the state is prepared with some density matrix ρ_i with probability p_i we can denote the density matrix as $\sum_i p_i \rho_i$. To prove this lets say that ρ_i arises from some ensemble $\{p_{ij}, |\psi_{ij}\rangle\}$ hence probability for state $|\psi_{ij}\rangle$ is $p_i p_{ij}$. So we get

$$\rho = \sum_{ij} p_i p_{ij} |\psi_{ij}\rangle \langle \psi_{ij}| = \sum_i p_i \rho_i$$

An interesting property of the density operator is that different states can generate the same density operator. Suppose we have two sets $|\psi_i\rangle$ and $|\tilde{\psi}_j\rangle$. Let $|\tilde{\psi}_i\rangle = \sum_j u_{ij} |\tilde{\phi}_j\rangle$ for some unitary u_{ij} . Then we have

$$\sum_i |\tilde{\psi}_i\rangle \langle \tilde{\psi}_i| = \sum_{ijk} u_{ij} u_{ik}^* |\tilde{\phi}_j\rangle \langle \tilde{\phi}_k| \quad (2.10)$$

$$\sum_i |\tilde{\psi}_i\rangle \langle \tilde{\psi}_i| = \sum_{jk} \left(\sum_i u_{ki}^\dagger u_{ij} \right) |\tilde{\phi}_j\rangle \langle \tilde{\phi}_j| \quad (2.11)$$

$$\sum_i |\tilde{\psi}_i\rangle \langle \tilde{\psi}_i| = \sum_{jk} \delta_{kj} |\tilde{\phi}_j\rangle \langle \tilde{\phi}_j| \quad (2.12)$$

$$\sum_i |\tilde{\psi}_i\rangle \langle \tilde{\psi}_i| = \sum_j |\tilde{\phi}_j\rangle \langle \tilde{\phi}_j| \quad (2.13)$$

We can clearly see that these two construct the same density operator as per the relation assumed. For the converse if we start with eq 2.13 we can retrace back to get the above assumed relation.

Another important thing is the representation of mixed states on the Bloch sphere. For pure states the state is represented by a point on the Bloch sphere. Interestingly for mixed states, the state is represented by a point inside the sphere. Let there be a state $|\psi\rangle$ then we know that $\rho = \frac{I + \vec{r} \cdot \sigma}{2}$. This can be proved very easily since ρ is hermitian.

One can also see that for a mixed state $|\vec{r}| < 1$. This vector \vec{r} is called the Bloch vector of ρ .

2.4.1 Partial Trace and Reduced density operator

Suppose we have a density operator ρ^{AB} which describes two systems A and B . The reduced density operator for system A is defined as the following

$$\rho^A = \text{tr}_B(\rho^{AB})$$

tr_B is just the trace over the system B . Partial trace is defined as this

$$\text{tr}_B(|a_1\rangle \langle a_2| \otimes |b_1\rangle \langle b_2|) = |a_1\rangle \langle a_2| \text{tr}(|b_1\rangle \langle b_2|)$$

Since its a composite system we write ρ^{AB} as a tensor product. The reduced density operator is a very useful tool. Lets assume we have the following state $|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$. This happens to be one of the bell pairs. So we have

$$\rho = \frac{|00\rangle \langle 00| + |11\rangle \langle 00| + |00\rangle \langle 11| + |11\rangle \langle 11|}{2} \quad (2.14)$$

For reduced density operator for the first qubit we have

$$\rho^1 = \text{tr}_2(\rho) \quad (2.15)$$

$$\rho^1 = \frac{|0\rangle \langle 0| + |1\rangle \langle 1|}{2} = \frac{I}{2} \quad (2.16)$$

So we can see that the Bloch vector for ρ^1 is actually zero hence it must be a mixed state which is pretty weird since we happen to know $|\psi\rangle$ completely. Interestingly we get the same result for all the bell pairs.

An interesting property of the partial trace is that it is a unique function for creating a map from operators in AB to operators in A . Also reduced density operator explains why quantum teleportation makes sense but more on that later.

2.4.2 Schmidt Decomposition

According to the Schmidt decomposition theorem, suppose $|\psi\rangle$ is a pure state of a composite system, AB . Then there exist orthonormal states $|i_A\rangle$ for system A , and orthonormal states $|i_B\rangle$ of system B such that

$$|\psi\rangle = \sum_i \lambda_i |i_A\rangle |i_B\rangle$$

Where λ_i are non-negative real numbers satisfying $\sum_i \lambda_i^2 = 1$ known as Schmidt coefficients. This is a very strong result since we don't need A and B to even be of the same dimension. Also as a result of this we get $\rho^A = \sum_i \lambda_i^2 |i_A\rangle \langle i_A|$ and $\rho^B = \sum_i \lambda_i^2 |i_B\rangle \langle i_B|$ so essentially we get that ρ^A and ρ^B have the same eigen values. The following is the proof for this theorem.

We know that for an operator O_A in A and some suitable ρ^A

$$\langle \psi | (O_A \otimes I) | \psi \rangle = \text{tr}(O_A \rho^A) \quad (2.17)$$

This is a consequence of the decomposition so our aim is to prove this. There exists some orthonormal basis say $|i_A\rangle$ such that

$$\rho^A |i_A\rangle = p_i |i_A\rangle \quad (2.18)$$

Lets now define the state in this basis along with some arbitrary basis in B $|\phi_i\rangle$

$$|\psi\rangle = \sum_{ij} c_{ij} |i_A\rangle |\phi_i\rangle = \sum_i |i_A\rangle \left(\sum_j c_{ij} |\phi_j\rangle \right) \quad (2.19)$$

if we set $\sum_j c_{ij} |\phi_j\rangle = c_i |i_B\rangle$ for some unit vector $|i_B\rangle$ and a positive c_i . Now we have

$$|\psi\rangle = \sum_i c_i |i_A\rangle |i_B\rangle \quad (2.20)$$

We know that

$$\text{tr}(O_A \rho_1) = \sum_i \langle i_A | O_A \rho_1 | i_A \rangle = \sum_i p_i \langle i_A | O_A | i_A \rangle \quad (2.21)$$

$$\langle \psi | (O_A \otimes I) \psi \rangle = \sum_j c_i^* c_j \langle i_A | O_A | j_A \rangle \langle i_B | j_B \rangle \quad (2.22)$$

$\langle i_B | j_B \rangle$ disappears for $i \neq j$ so we actually get RHS in eq 2.22 equal to RHS in eq 2.21. So we get $|c_i|^2 = p_i$ and if we take c_i as positive then we get $c_i = \sqrt{p_i}$. Now if we repeat this process by starting with some operator O_B in B we get that their density matrices have the same eigen values hence proving our theorem. Interestingly we didn't need to assume that A and B are of the same dimension in all of this. So we get

$$\langle \psi | (I \otimes O_B) \psi \rangle = \text{tr}(O_B \rho^B) \quad (2.23)$$

When it comes down to it we can break down a two system state into the form stated in the theorem and that actually gives rise to some very interesting symmetries. For example in the state $|\psi\rangle = \frac{|00\rangle + |01\rangle + |11\rangle}{\sqrt{3}}$ using Schmidt decomposition we get that $\text{tr}((\rho^A)^2)$ and $\text{tr}((\rho^B)^2)$ both come out to be as $7/9$. The symmetry here is not very obvious but essentially involves breaking it down in a way that the density operators for the two systems have the same eigen values. Also this cannot be extended to three systems since cross terms cannot be avoided there.

Another important thing which can be done with this is purification. It essentially works this way. We are given a density matrix ρ^A of a system A . We can now introduce another system R and define a pure system $|AR\rangle$ in both these systems such that $\rho^A = \text{tr}_R(|AR\rangle \langle AR|)$. This is known as purification and allows us to associate a pure state to a mixed state. If we define $|AR\rangle$ like this

$$|AR\rangle = \sum_i \sqrt{p_i} |i_A\rangle |i_R\rangle$$

Using this we get

$$\text{tr}_R(|AR\rangle \langle AR|) = \sum_{ij} \sqrt{p_i p_j} |i_A\rangle \langle j_A| \text{tr}(|i_R\rangle \langle j_R|) \quad (2.24)$$

$$\text{tr}_R(|AR\rangle \langle AR|) = \sum_{ij} \sqrt{p_i p_j} |i_A\rangle \langle j_A| \delta_{ij} \quad (2.25)$$

$$\text{tr}_R(|AR\rangle \langle AR|) = \sum_i p_i |i_A\rangle \langle i_A| = \rho^A \quad (2.26)$$

One must note that using some $U_R |R\rangle$ instead of $|R\rangle$ where U_R is a unitary evolution in R also gives a purification

2.5 EPR and the Bell inequality

This would quite possibly be one of the most debated topics in Quantum mechanics. In their 1935 paper Einstein, Polensky and Rosen offered an argument to prove that the current Quantum theory is incomplete. Their idea was later refined by David Bohm and today is mainly presented in this form. Suppose we prepare a two qubit state like this, $|\psi\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$. If we measure the first qubit and we get $|1\rangle$ we reach the conclusion that the second qubit would have to become $|0\rangle$ and vice-versa however this violates the idea of local realism which had been assumed to be true.

Later in Bell's 1971 paper Bell assumed a local deterministic world along with the hidden variables theory and reached an inequality which could be checked experimentally. He concluded in this paper that it is not possible for a hidden variable theory to support local determinism. I wont go over the exact math in that paper however later his inequality gave rise to the CSHS inequality.

The experimental setup is something like this, two people Alice and Bob (better get used to these names as they come up quite often) are given one of the qubits from the previously mentioned $|\psi\rangle$. They have a choice to measure along different vectors in the way that Alice can either measure along \vec{q} and \vec{r} and get values Q or R . Similarly Bob can measure along \vec{s} and \vec{t} and get values S or T . Note that all of Q, R, S, T take either +1 or -1 as a value.

According to Bell's inequality we get

$$E(QS) + E(RS) + E(RT) - E(QT) \leq 2$$

But here's the issue, nature doesn't follow this inequality. Various experiments have been performed and it turns out that this inequality isn't completely true and the actual bound comes out to be $2\sqrt{2}$. This is mainly because the math involved in deriving this was actually done without considering any quantum mechanics and the commutation relations between observables. This was refined in the Tsirelson's inequality which essentially sets the higher bound as $2\sqrt{2}$.

It goes like this, we take $Q = \vec{q} \cdot \vec{\sigma}$, $R = \vec{r} \cdot \vec{\sigma}$, $S = \vec{s} \cdot \vec{\sigma}$, $T = \vec{t} \cdot \vec{\sigma}$. We can prove the following equation

$$(Q \otimes S + R \otimes S + R \otimes T - Q \otimes T)^2 = 4I + [Q, R] \otimes [S, T] \quad (2.27)$$

from here on taking average value on both sides and using the inequality from eq. 2.8 we get the following equation

$$\langle Q \otimes S \rangle + \langle R \otimes S \rangle + \langle R \otimes T \rangle - \langle Q \otimes T \rangle \leq 2\sqrt{2} \quad (2.28)$$

This is the Tsirelson's inequality.

Chapter 3

Quantum Circuits

In this chapter I will be discussing the various quantum gates and building quantum circuits

3.1 Quantum Gates

All quantum gates are essentially a unitary evolution which acts on a state. For single qubits an important thing to observe about these are that they are all unitary operations and can be described as a global phase multiplication along with rotations about the x, y, z axes.

Let there be some unitary evolution U acting on a single qubit state. We can prove that all unitary evolutions can be defined in the following way

$$U = e^{i\alpha} R_x(\beta) R_y(\gamma) R_z(\delta) \quad (3.1)$$

Here R_x, R_y, R_z are the rotation operators as defined in the section 2.2.2 in the previous chapter. The phase term is irrelevant when multiplied to a single qubit but ahead we will see that it can be used over a multiple qubit system to actually change it. The following are some important quantum gates which we will be seeing quite often

On a side note the T gate which is called the $\pi/8$ gate actually causes a rotation of $\pi/4$ and affects the global














Operator	Gate(s)	Matrix
Pauli-X (X)	 	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Figure 3.1: Quantum Gates

phase by $\pi/8$ hence the name is actually quite deceiving hence we will be referring to it as the T gate only.

This formulation of eq 3.1 can also be extended like this

$$U = e^{i\alpha} R_{\hat{n}}(\beta) R_{\hat{m}}(\gamma) R_{\hat{n}}(\delta) \quad (3.2)$$

Here \hat{n} and \hat{m} are two non parallel unit vectors about which the rotations are occurring and $\alpha, \beta, \gamma, \delta$ are chosen appropriately for the unitary operation.

Another interesting way of writing the unitary operator is $U = AXBXC$ where $ABC = I$. This can be easily verified by putting $A = R_z(\beta)R_y(\gamma/2)$, $B = R_y(-\gamma/2)R_z(-(\delta + \beta)/2)$ and $C = R_z((\delta - \beta)/2)$ and using eq. 3.2 with $\hat{n} = \hat{z}$ and $\hat{m} = \hat{y}$. Clearly $ABC = I$ and using the identity $XR_y(\theta)X = R_y(-\theta)$ and $XR_z(\theta)X = R_z(-\theta)$ we get $U = AXBXC$.

Some other identities using single qubits are

$$HXH = Z, HYH = -Y, HZH = X$$

3.1.1 Controlled gates

Controlled gates which is controlled by n qubits ($|x_1, x_2, \dots, x_n\rangle$) and causes a unitary evolution U over k qubits (state $|\psi\rangle$) is described by the following gate $C^n(U)$ in this equation

$$C^n(U) |x_1, x_2, \dots, x_n\rangle |\psi\rangle = |x_1, x_2, \dots, x_n\rangle U^{x_1 x_2 \dots x_n} |\psi\rangle \quad (3.3)$$

Put simply it applies the operation U over $|\psi\rangle$ when all of x_1, x_2, \dots, x_n are 1.

One can see that the controlled not ($CNOT$) described in the figure above is a controlled gate. Using this gate entangles the two qubits it acts on. We can also describe a controlled unitary operator using one qubit for control using the $U = AXBXC$ method where $ABC = I$ in the following manner. The Toffoli gate is a not

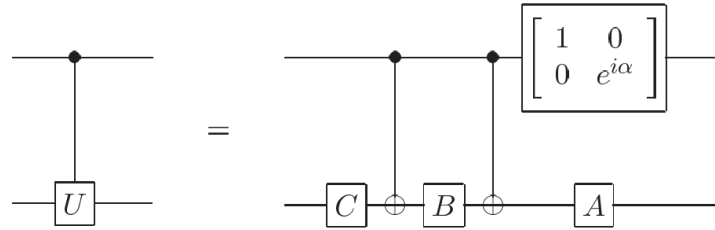


Figure 3.2: A $C^1(U)$ gate

controlled by two qubits. It can be constructed using $CNOT, H, T, T^\dagger$ as shown in fig. 3.3

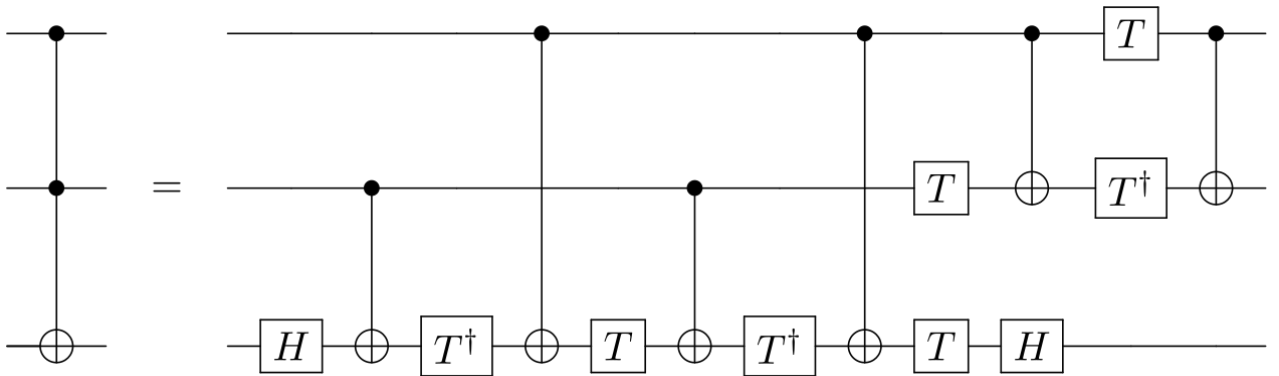


Figure 3.3: Toffoli gate construction

If we are to make a $C^2(U)$ gate we can construct it using $CNOT$ and V gates where $V^2 = U$ and V is unitary.

The circuit would be something like this. If we wish to execute a Toffoli gate we can put $V = \frac{(1 - i)(I + iX)}{2}$. For constructing a fredkin gate using toffoli gates is quite simple. It can even be constructed using one toffoli and two CNOTs as shown here.

However an interesting thing about the fredkin gate is that it can be constructed completely with only five two

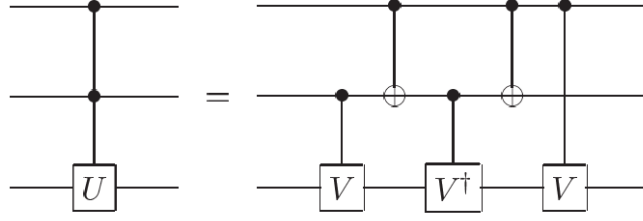


Figure 3.4: A $C^2(U)$ gate

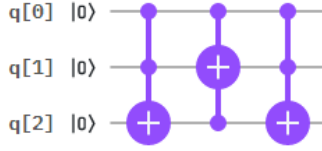


Figure 3.5: Fredkin using toffoli

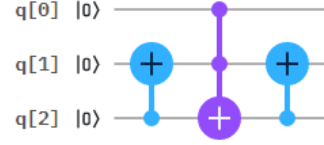


Figure 3.6: Fredkin using CNOT and toffoli

qubit gates. In fig. 3.6 replacing the toffoli by a construction like that of fig. 3.4 we get fig. 3.7 where $V^2 = X$ and V is unitary.

The two $CNOT$ s in the right end commute and so we can combine the last $CNOT$ with the controlled V^\dagger

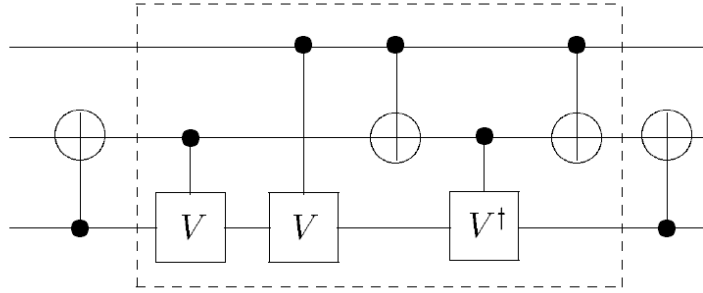


Figure 3.7: Fredkin with 5 two qubit gates

and similarly combine the first $CNOT$ and the controlled V . This yields us five two qubit gates.

Say we want to construct a $C^5(U)$ gate we can do so in a simple method using ancillary or work qubits along with some toffoli gates

This can easily be extended to $C^n(U)$ however the issue is that there is a lot of work qubits involved. Say we want to execute $C^n(Z)$. We can actually solve this quite easily using the method described in fig. 3.4 for executing this. So we can see that if we use $C^{n-1}(X)$ with it acting on the n^{th} qubit controlled by the first $(n-1)$ qubits, followed by controlled \sqrt{Z}^\dagger which acts on the target qubit controlled by n^{th} qubit. This is followed by another $C^{n-1}(X)$ acting on the n^{th} controlled by the first $(n-1)$ bits and after this we have a controlled \sqrt{Z} gate which acts on the target qubit controlled by the n^{th} qubit. This is followed by a $C^{n-1}(\sqrt{Z})$ gate which acts on the target qubit and is controlled by the first $n-1$ qubits. We can now break this gate down recursively following the previous procedure and using controlled $\sqrt[n]{Z}$ gates and go on till we only have gates which can be controlled by a single qubit.

Note that even though we are using multiple control not gates they too can be broken down in a similar process so we can construct the whole thing just using toffolis and CNOTs and two qubit gates. However the circuit complexity has gone in $O(n^2)$ in comparison to the work qubit method.

Another thing about controlled gates is that when a certain control qubit has a white dot on it, it implies that the NOT of that qubit is being used in control. Refer fig. 3.10.

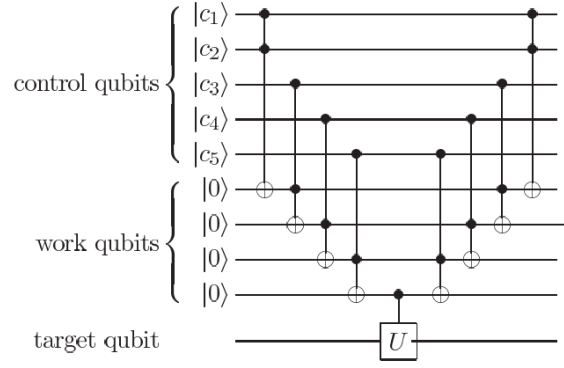


Figure 3.8: $C^5(U)$ with work qubits

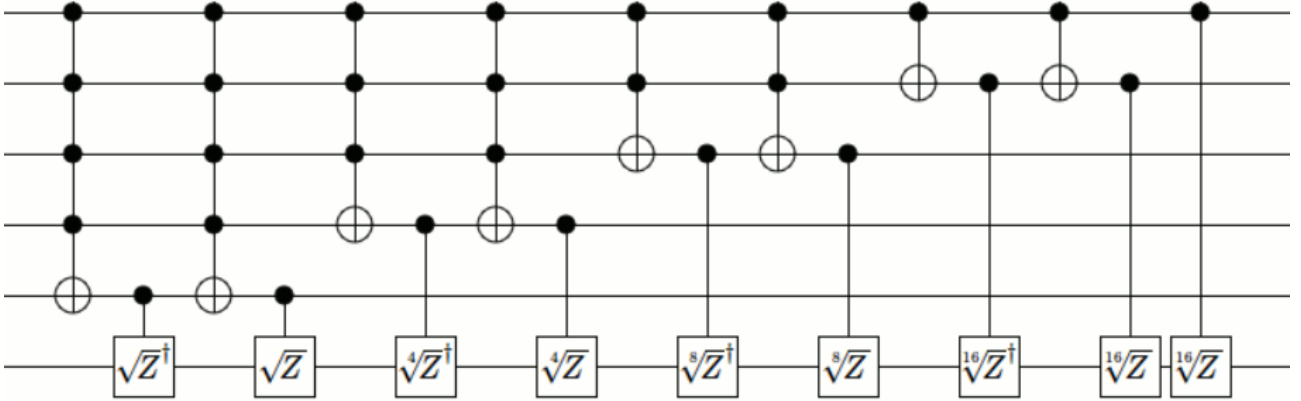


Figure 3.9: Iteratively executing a $C^5(Z)$

3.1.2 Measurement in circuits and Quantum Teleportation

Measurement is represented in circuits like this.

Measurement always has only two possibilities $|0\rangle$ and $|1\rangle$ however one can measure on a different basis by simply multiplying the appropriate operator to the state before measurement. For example if we wish to measure in the $|+\rangle$ and $|-\rangle$ basis then one has to apply the Hadamard gate before measurement.

An important equality in measurement is that if we have $C^1(U)$ and we apply measurement before the gate on the controlling qubit it is the same as measuring the qubit after applying the gate.

An important application of quantum computing arises with using measurement and that is quantum teleportation. In Quantum teleportation we essentially transport an unknown state from one qubit to another.

So we have the original state $|\psi\rangle$ and two other qubits initialized at $|B_{00}\rangle$ which is the EPR pair $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ and they follow the given circuit (fig. 3.11)

To think of the teleportation in terms of the receiver and sender we can think of it this way. Alice and Bob have an EPR pair with them with each having one of the qubits. Alice now makes the qubit with state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ interact with her qubit from the EPR pair and then measures both the qubits and based on her results Bob applies the appropriate transformations over his qubit and will then obtain the state $|\psi\rangle$ on his qubit.

Initially the state of the system is $|\psi_0\rangle = |\psi\rangle |\beta_{00}\rangle$. We can see that after the $CNOT$ and H gates the state of the system is $|\psi_1\rangle = \frac{1}{2}[\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|01\rangle + |10\rangle)]$ this can be rearranged into the following

$$|\psi_1\rangle = \frac{1}{2} [|00\rangle (\alpha|0\rangle + \beta|1\rangle) + |01\rangle (\alpha|0\rangle + \beta|1\rangle) + |10\rangle (\alpha|0\rangle - \beta|1\rangle) + |11\rangle (\alpha|1\rangle - \beta|0\rangle)]$$

Hence on measuring the first two qubits and applying appropriate transformation to the third qubit will make it into the original state $|\psi\rangle$. Note that since Bob requires the result of Alice's measurement to obtain the original state and that is possible only via a classical channel of communication there is no violation of relativity

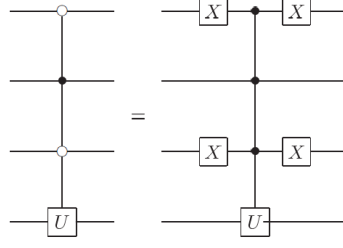


Figure 3.10: Controlled gates with white dots

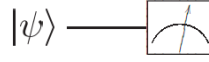


Figure 3.11: Measurement in Quantum circuits

occurring.

Now if we were to calculate the density operator of the state $|\psi_1\rangle$ we see that its equal to the following

$$\rho = \frac{1}{4} [|00\rangle \langle 00| (\alpha |0\rangle + \beta |1\rangle)(\alpha^* \langle 0| + \beta^* \langle 1|) + |01\rangle \langle 01| (\alpha |1\rangle + \beta |0\rangle)(\alpha^* \langle 1| + \beta^* \langle 0|) + |10\rangle \langle 10| (\alpha |0\rangle - \beta |1\rangle)(\alpha^* \langle 0| - \beta^* \langle 1|) + |10\rangle \langle 10| (\alpha |1\rangle - \beta |0\rangle)(\alpha^* \langle 1| - \beta^* \langle 0|)]$$

The reduced density operator for Bob's state can be calculated using partial trace.

$$\rho_B = \frac{2(|\alpha|^2 + |\beta|^2) |0\rangle \langle 0| + 2(|\alpha|^2 + |\beta|^2) |1\rangle \langle 1|}{4} = \frac{I}{2}$$

Clearly this contains no information about $|\psi\rangle$ hence without the measurement and communication of the result Bob has no information regarding $|\psi\rangle$ at all hence this doesn't violate relativity at all.

3.2 Universality in Quantum Computing

We know that using *NAND* gates alone one can implement all boolean functions. Similarly we can also implement any arbitrary unitary evolution using just CNOT, Hadamard, Phase and $\pi/8$ gates upto an arbitrary level of accuracy. It is important to understand that one cannot represent every unitary evolution since they exist over a continuous set of variables hence could require an infinite number of these gates to execute.

3.2.1 Two level unitary operators

One can perform all unitary evolutions using just CNOT gates and single qubit gates. To understand this we must first see that we can represent every unitary evolution as a product of some number of two level unitary matrices.

A two level unitary matrix is a which affects only two or fewer of the components of the vector it acts on. A d dimensional unitary matrix can be written as a product of at most $d(d-1)/2$ two level unitary matrices. The steps in finding this unitary decomposition is to keep making the diagonal elements of U equal to 1 and make the rest of the row and column zero and at each stage we will get a two level unitary evolution to multiply. repeat till we don't end up with a two level unitary matrix.

Now if we want to execute a two level unitary operation we can do so using single qubit gates and *CNOT* gates. So first of all let the two level unitary operation be some U . This acts only on two binary sequences say $|g_1\rangle$ and $|g_m\rangle$ and these are connected by the gray code $|g_2\rangle, |g_3\rangle, \dots, |g_{m-1}\rangle$.

A gray code is a sequence where each term differs from the last by just a single bit. Lets say the $|g_{m-1}\rangle$ and $|g_m\rangle$ differ in their j^{th} bit, then we have to construct a controlled operation such that it acts the unitary \tilde{U} on the j^{th} qubit and is controlled by the rest of the cubits. Here \tilde{U} is the sub matrix of U which makes it differ from an identity matrix. However this must be applied after we interchange the $|g_1\rangle$ state with $|g_{m-1}\rangle$

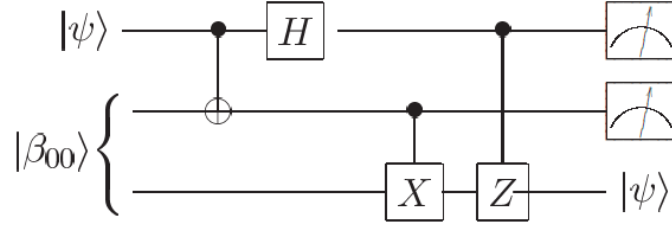


Figure 3.12: Quantum Teleportation

Here is an example. Suppose we wish to execute the unitary U which is the following.

$$U = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{bmatrix}$$

Clearly $\tilde{U} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$ and the two states U affects is $|000\rangle$ and $|111\rangle$. The gray code sequence between them is $|000\rangle \rightarrow |001\rangle \rightarrow |011\rangle \rightarrow |111\rangle$. Here the last two terms differ in their first qubit and we must exchange the state $|000\rangle$ with $|011\rangle$. We can see that the following circuit in fig. 3.13 does this.

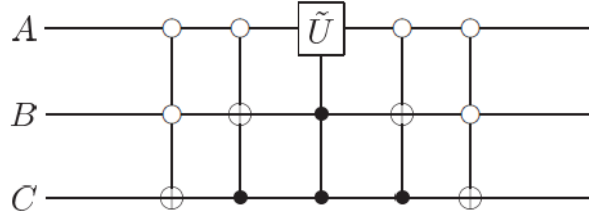


Figure 3.13: Executing the circuit for the operation U

Its trivial to see that this circuit can be broken into *CNOTs* and single qubit gates using methods described previously. For a two level unitary evolution we can see that it takes complexity of $O(n^2)$ on breaking down into *CNOTs* and single qubit gates. If we were to extend this to an arbitrary unitary evolution we can see that it will become $O(n^2 2^{2n})$ since an arbitrary unitary evolution can be represented as a product of $O((2^n)^2)$ since the operator is 2^n dimensional and the number of two level matrices required is in n^2 complexity.

3.2.2 Approximating Unitary operators

Now here's the thing. Notice how we so freely used any arbitrary single qubit gate? Well we often just cannot come up with any arbitrary unitary operation over a single qubit so easily. We will now define a quantity $E(U, V)$ where U and V are unitary operators of the same state space.

$$E(U, V) = \max ||(U - V) |\psi\rangle|| \quad (3.4)$$

The max represents the maximum over all possible $|\psi\rangle$ in the state space. Suppose a quantum system starts in the state $|\psi\rangle$, and we perform either the unitary operation U , or the unitary operation V . Following this, we perform a measurement. Let M be a POVM element associated with the measurement, and let P_U (or P_V) be the probability of obtaining the corresponding measurement outcome if the operation U (or V) was performed. We then have

$$|P_U - P_V| = |\langle \psi | U^\dagger M U | \psi \rangle - \langle \psi | V^\dagger M V | \psi \rangle| \quad (3.5)$$

If we substitute $|\Delta\rangle = (U - V) |\psi\rangle$ we get the following from Cauchy-Schwarz inequality

$$|P_U - P_V| = |\langle \psi | U^\dagger M |\Delta\rangle + \langle \Delta | M V | \psi \rangle| \quad (3.6)$$

$$|P_U - P_V| \leq ||\Delta\rangle|| + ||\Delta\rangle|| \quad (3.7)$$

$$|P_U - P_V| \leq 2E(U, V) \quad (3.8)$$

Hence if $E(U, V)$ is small, the measurement results of U and V will also be very close. Note that if we use sequenced gates, the error value will add up linearly and so wouldn't actually cause big problems

$$E(U_2U_1 - V_2V_1) = ||(U_2U_1 - V_2V_1)|\psi\rangle|| \quad (3.9)$$

$$E(U_2U_1 - V_2V_1) = ||(U_2U_1 - V_2U_1)|\psi\rangle + (V_2U_1 - V_2V_1)|\psi\rangle|| \quad (3.10)$$

$$E(U_2U_1 - V_2V_1) = ||(U_2U_1 - V_2U_1)|\psi\rangle + (V_2U_1 - V_2V_1)|\psi\rangle|| \quad (3.11)$$

Using triangle inequality we get

$$E(U_2U_1 - V_2V_1) \leq ||(U_2 - V_2)U_1|\psi\rangle|| + ||U_1 - V_1)V_2|\psi\rangle|| \quad (3.12)$$

$$E(U_2U_1 - V_2V_1) \leq E(U_2, V_2) + E(U_1, V_1) \quad (3.13)$$

This can be extended to any number of operations using induction.

Consider the operations T and HTH . T gives a rotation about Z axis by an angle of $\pi/4$. HTH gives a rotation about X axis by an angle of $\pi/4$. Combining these both we get

$$\exp(-i\frac{\pi}{8}Z)\exp(-i\frac{\pi}{8}X) = \left[\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)Z\right]\left[\cos\left(\frac{\pi}{8}\right)I - i\sin\left(\frac{\pi}{8}\right)X\right] \quad (3.14)$$

$$\exp(-i\frac{\pi}{8}Z)\exp(-i\frac{\pi}{8}X) = \cos^2\left(\frac{\pi}{8}\right)I - i\left[\cos\left(\frac{\pi}{8}\right)(X + Z) + \sin\left(\frac{\pi}{8}\right)Y\right]\sin\left(\frac{\pi}{8}\right) \quad (3.15)$$

As one can see this obtains us a rotation by an θ such that $\cos\left(\frac{\theta}{2}\right) = \cos^2\left(\frac{\pi}{8}\right)$ about the vector $\vec{n} = \left(\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8}\right)$. We can see that this θ is an irrational multiple of 2π . Hence suppose we wish to make $R_{\vec{n}}(\alpha)$ for some arbitrary α we can replicate to an arbitrary accuracy using repeated iteration $R_{\vec{n}}(\theta)$

Lets say we wish to have it to some accuracy δ and N is an integer larger than $2\pi/\delta$. Lets define a sequence $\theta_k = (k\theta) \bmod 2\pi$. From pigeonhole principle we can see that there exists distinct $j, k \in 1, 2, \dots, N$ such that $|\theta_j - \theta_k| \leq 2\pi/N < \delta$ hence we can use $\theta_{l(j-k)}$ to attain any value in $[0, 2\pi)$ with accuracy of δ .

Now we can see that $HR_{\vec{n}}(\theta)H = R_{\vec{m}}(\theta)$ where $\vec{m} = \left(\cos\frac{\pi}{8}, -\sin\frac{\pi}{8}, \cos\frac{\pi}{8}\right)$. Now if we recall eq. 3.2, we can see that all unitary operators can be represented as the following subject to a certain level of accuracy.

$$U = e^{i\alpha} R_{\vec{n}}(\theta)^{n_1} H R_{\vec{n}}(\theta)^{n_2} H R_{\vec{n}}(\theta)^{n_3} \quad (3.16)$$

Now here is the important question. How efficient is this approach? Say we have m gate circuit we wish to replicate to closeness of ϵ (we define this closeness as $E(U, V) < \epsilon$). This will give us a $\Omega(m2^{(m/\epsilon)})$ which is not really good however with the θ_k approach we can see that the range of angles gets filled up fairly uniformly so we can consider a complexity of some $\Theta(m^2/\epsilon)$. This gives us a fairly good approach however it can get even better. According to the Solovay-Kitaev theorem one can achieve a complexity of $O(m \log(m/\epsilon))$.

3.3 Simulation of Quantum systems

One of the biggest motivations for a Quantum computer was for simulating a Quantum system which would prove to be extremely difficult for a classical computer. Classical computers can simulate quantum systems however they are done so very inefficiently. The equation of concern here is the Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} |\psi\rangle = \hat{H} |\psi\rangle$$

We will rewrite in the following form since we are dealing with real particles. Here $\langle x|\psi\rangle = \psi(x)$

$$i\hbar \frac{\partial}{\partial t} \psi(x) = \left[-\frac{1}{2m} \frac{\partial^2}{\partial x^2} + V(x) \right] \psi(x)$$

The difficulty arises in the number of equations which need to be solved. A n qubit system requires 2^n equations to be solved hence making it exponentially difficult for classical computers. There do exist some approximation techniques which make it a little more feasible on classical computers but there are still many quantum systems which do not have any such approximations. Also a density matrix describing an n qubit system would require

$4^n - 1$ independent real numbers.

Now for a time independent Hamiltonian we can write the following equation

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$$

Since H is usually pretty difficult to remove the exponent of, we make approximations usually to the second or third order of Δt . However most real world systems have multiple interaction terms in their Hamiltonian expressions hence are of form $H = \sum_{k=1}^L H_k$. Unless $[H_j, H_k] = 0$ for all j, k we will have the following equation $e^{-iHt} \neq e^{-iH_1t}e^{-iH_2t} \dots e^{-iH_n t}$. Now this does make things harder for us since the commutation terms will appear alongside these normal terms and they need not always be zero. There is a pretty smooth workaround however. Notice that $\lim_{n \rightarrow \infty} (e^{iAt/n} e^{iBt/n})^n = e^{i(A+B)t}$ since the commutation terms drop out since they have a n in their denominator hence we can simulate the system to a fair level of accuracy for a sufficiently large n . The algorithms functions like this

Inputs: The initial state $|\psi_0\rangle$ along with the Hamiltonian $H = \sum_{k=1}^L H_k$, the error range δ and time t_f for which we have to find $|\psi(t_f)\rangle$

Outputs: The state $|\psi(t_f)\rangle$ such that $|\langle \psi(t_f) | \tilde{\psi} \rangle|^2 < 1 - \delta$

Procedure:

(1) Initialize state as $|\tilde{\psi}_0\rangle = |\psi_0\rangle$

(2) Iterative update $|\tilde{\psi}_{j+1}\rangle = U_{\Delta t} |\tilde{\psi}_j\rangle$

(3) $j = j + 1$; goto (2) while $j\Delta t < t_f$ (4) $|\psi(t_f)\rangle = |\tilde{\psi}_j\rangle$ final result **Runtime:** This typically functions

in $O(\text{poly}(1/\delta))$ operations.

Here $U_{\Delta t}$ is a unitary which describes the approximate evolution over a time of Δt

An interesting kind of Hamiltonian are Hamiltonians of form $H = H_1 \otimes H_2 \otimes H_3$ which is acting on a three qubit system such that H_1 acts on the first qubit and so on. Lets pick the example $H = Z_1 \otimes Z_2 \otimes Z_3$. The circuit for this would look something like this Here's the interesting part, we can execute Hamiltonians which

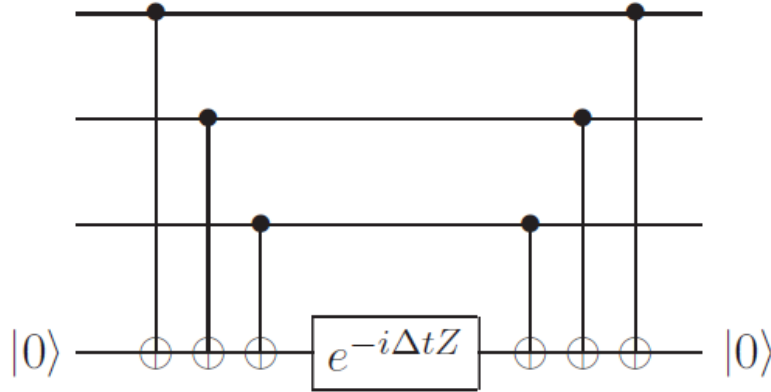


Figure 3.14: Circuit for $H = Z_1 \otimes Z_2 \otimes Z_3$

cause a rotation of π on the Bloch sphere using this circuit along with basis changing operators. For example if we want X_1 instead of Z_1 we can simply put Hadamard on the first bit at the start and at the end and it would work. We can extend it to rotations of arbitrary angles by using a different operator on the ancilla bit. So this way we can execute Hamiltonians of form $H = \sum_{k=1}^L H_k + Z^{\otimes n}$ or something similar however these usually aren't found in nature.

Not all Hamiltonians can be executed efficiently however for example to process of a system achieving equilibrium has no known efficient way of execution also the process by itself is not very well understood. The equilibrium state's density matrix is defined as the Gibb's state and is written as $\rho_{eq} = e^{-\beta H}/Z$ where the partition function $Z = \text{tr}(e^{-\beta H})$. Note that ρ_{eq} is not provably the actual density matrix of the equilibrium state but rather an educated guess for it.

Chapter 4

Algorithms for Quantum Computing

In this chapter I will discuss the algorithms and functionality behind certain algorithms for Quantum Computers.

4.1 The Quantum Fourier Transform

The normal Fourier transform essentially decomposes a signal into it's constituent frequencies. The Quantum Fourier transform in the same idea acts on a $N = 2^n$ dimensional vector $(x_0, x_1, \dots, x_{N-1})$ in \mathbb{C}^N and maps it to a vector $(y_0, y_1, \dots, y_{N-1})$ in \mathbb{C}^N according the following formula

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^N x_j e^{2\pi i j k / N}$$

Now an important thing to note is that this happens to be a unitary transform which means that we can execute it in a quantum circuit thanks to universality. Now we can represent each vector in this space with n qubits. Let each state be represented as $|j_1, j_2, \dots, j_n\rangle$. Let $0.j_1j_2\dots j_n$ represent the binary representation of the sum of fractions $j_1/2 + \dots + j_n/2^n$. We can see that the Fourier transform over this state can be written as

$$U_{QFT} |j_1, j_2, \dots, j_n\rangle = \frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1}j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1j_2\dots j_n} |1\rangle)}{2^{n/2}} \quad (4.1)$$

Now if we define a rotation matrix $R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{bmatrix}$ we can create an efficient circuit for executing the Quantum Fourier transform like this Using this we can do lots of things which classical computers would have

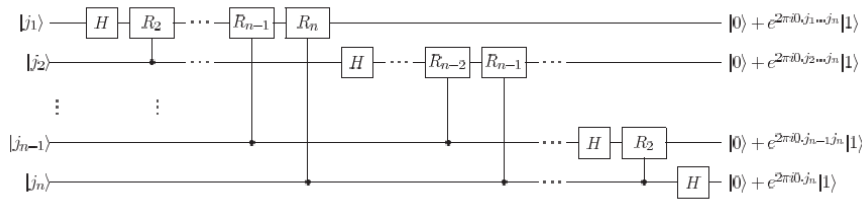


Figure 4.1: Circuit for QFT. Note that this does not include the swap gates for reversing order of output qubits and normalization

taken exponential time for since classical computers would require $\Theta(2^{2n})$ complexity in number of operations θn^2 as seen from fig 4.1 and that is an exponential improvement. Here are some interesting things which can be done using the Fourier transform.

4.1.1 Phase estimation

Suppose we have unitary operator U with an eigen vector $|u\rangle$ with an eigenvalue $e^{2\pi i \phi}$ where ϕ is unknown and we wish to estimate its value. We use two registers, one which has t qubits and the other which is initialized in state $|u\rangle$. Now let's say that the binary representation of ϕ is $0.\phi_1\phi_2\dots\phi_t$ accurate to t places after the radix point.

We will now execute the following circuit described in fig. 4.2. The state of the first register will now be

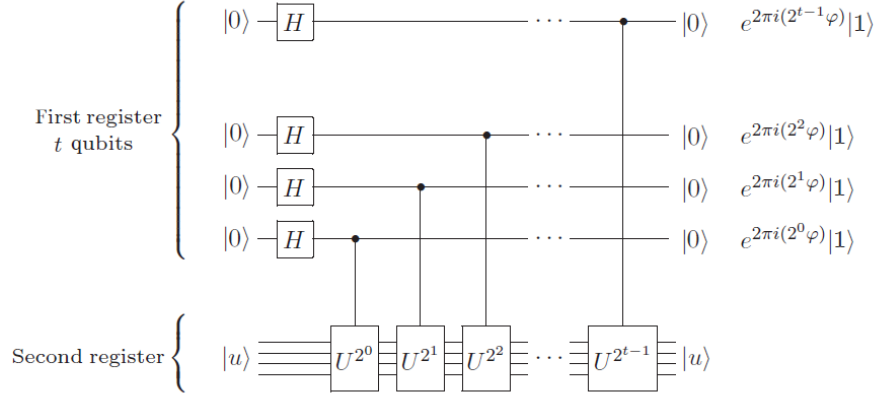


Figure 4.2: Circuit for phase estimation. Note that we perform inverse Fourier transform on the first register and then a measurement on the first register

$$\frac{(|0\rangle + e^{2\pi i 0 \cdot \phi_t} |1\rangle)(|0\rangle + e^{2\pi i 0 \cdot \phi_{n-1} \phi_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0 \cdot \phi_1 \phi_2 \dots \phi_n} |1\rangle)}{2^{n/2}}$$

Clearly doing an inverse Fourier transform over this will give us the state $|\phi_1 \phi_2 \dots \phi_t\rangle$ hence on measurement we will be able to estimate this phase. This will soon prove to be very useful.

4.1.2 Order finding and factoring

Using the phase estimation algorithm we can also use it for order finding and factoring and also order finding and factoring happen to be equivalent problems. For two integers x and N which are co prime we define the order of x modulo N as the least positive integer r such that $x^r = 1 \pmod{N}$. The order finding problem essentially aims to find this r and it is considered to be a hard problem for classical computers since there is no known efficient algorithm for it. Note that r would never exceed N which is trivial to prove. We will now define a unitary which acts like this (note x is coprime to N)

$$U |y\rangle = |xy \pmod{N}\rangle$$

We can see that the eigen states of U would be written like this

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^k \pmod{N}\rangle$$

We can see that the eigen values are $\exp\left[\frac{2\pi i s}{r}\right]$ for s being an integer from 0 to $r-1$. We will be executing controlled U^{2^j} so we would need to know an efficient way to execute that but apart from that we would need the states $|u_s\rangle$ but we would need to know r for that but we obviously don't know that however there is a useful property of these states

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

And using this we can see that the following holds for a t qubit register in the state $|z\rangle$

$$|z\rangle U^{z_1 2^{t-1}} U^{z_2 2^{t-2}} \dots U^{z_t 2^0} |1\rangle = |z\rangle |x^z \pmod{N}\rangle$$

On a side note we can also construct a unitary which does addition over modulo N and initialize our state to zero and get a similar construct. Now to actually obtain the order from this requires use of the continued fractions algorithm. We will only obtain the phase estimate after performing inverse Fourier transform which would be $\phi \approx s/r$. The aim is to find a s' and r' such that $s'/r' = s/r$ and we can verify whether r' is the order by simply calculating $x^{r'} \pmod{N}$. If s and r are co-prime then we will get $r' = r$ so that is not an issue and also for most possible pairs of s and r however in the case that they aren't co-prime, the best approach is to run the algorithm twice so we will have two fractions s'_1/r'_1 and s'_2/r'_2 , if we have $s'_1 \neq s'_2$ we can get r by taking the LCM of r'_1, r'_2 and the probability of this being successful is fairly high and can be proved to be greater than $1/4$.

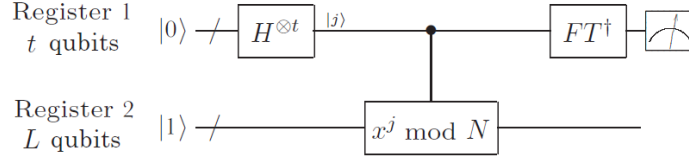


Figure 4.3: Circuit for order finding

The order finding algorithm proceeds in the following manner:

Input: Black box for executing $U_{x,N}$ which performs the following operation $U_{x,N} |j\rangle |k\rangle = |j\rangle |x^j k \bmod N\rangle$ for x co-prime to L bit integer N . Also t qubits initialized to $|0\rangle$ where $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ and an L qubit register set to the state $|1\rangle$.

Output: The smallest positive integer r such that $x^r \bmod N = 1$.

Procedure:

1. Initialize state to $|0\rangle |1\rangle$
2. Create the superposition state $\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |1\rangle$ using Hadamard.
3. Apply the black box and the state becomes $\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |x^j \bmod N\rangle \approx \frac{1}{\sqrt{r} 2^t} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j / r} |j\rangle |u_s\rangle$
4. Applying inverse Fourier transform we get the state $\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle |u_s\rangle$
5. Measure the first register to obtain $\widetilde{s/r}$
6. Apply continued fractions algorithm to obtain r

Now to prove that the problem of factoring actually reduces to order finding, we use two lemmas. If we have $x^2 = 1 \pmod{N}$ but $x \not\equiv \pm 1 \pmod{N}$ then at least one of $\gcd(x-1, N)$ and $\gcd(x+1, N)$ is a non trivial factor of N and this can be computed in $O(L^3)$ operations.

The second lemma is that suppose we have the prime factorization of an odd composite number $N = p_1^{\alpha_1} \dots p_m^{\alpha_m}$. Let x be randomly chosen with requirement $1 \leq x \leq N-1$ and x is co prime to N . Let r be the order of x modulo N then the probability of r being even and $x^{r/2} \not\equiv -1 \pmod{N} \geq 1 - \frac{1}{2^m}$

Using these two theorems we can obtain an algorithm which returns a non trivial factor of N with high probability and this happens to be Shor's Factoring algorithm. The steps of this algorithm are as follows (note this doesn't outline what exactly is happening in the circuit for this):

Inputs: A composite number N

Output: A non trivial factor of N

Procedure:

1. If N is even just return 2 else continue.
2. Determine using a classical algorithm whether $N = a^b$ for integers $a \geq 1$ and $b \geq 2$. The classical algorithm finds the value of $y = \log N$ and then $x = y/b$ for $b \leq L$ and then finds the closest integers to 2^x u_1 and u_2 . If either of u_1^b or u_2^b are equal to N then the N has such a representation for integer a, b according to the previous conditions and return a else continue.
3. Randomly chose x from 1 to $N-1$ and if $\gcd(x, N) > 1$ then return $\gcd(x, N)$ else continue.
4. Do the order finding subroutine to find order r of x modulo N .
5. If r is even and $x^{r/2} \equiv -1 \pmod{N}$ then compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$, and test to see if one of these is a non-trivial factor, returning that factor if so. Otherwise, the algorithm fails.

This runs in $O(L^3)$ complexity and has a success probability of $O(1)$. If we try factoring 91 by this algorithm, we can see that the first two steps would be skipped and then if we choose $x = 4$ then we can see that $r = 6$ and $4^{r/2} = 64 \bmod 91 \not\equiv -1 \bmod 91$ and $\gcd(64 - 1, 91)$ happens to give 7 so the algorithm succeeds. Though this may not seem very efficient if extrapolated to very large numbers it performs much better than classical algorithms since order finding has no known efficient algorithm for classical computers. This is the famous factoring algorithm which is soon to break RSA encryption according to most websites but that is mostly to be many years later considering the largest number that has been factorized on an actual quantum computer using this algorithm is 21.

4.1.3 Period finding

Another way of using the quantum Fourier transform is for finding the period of a periodic function which means given a function $f(x+r) = f(x)$ for $0 < r < 2^L$ we have to find the value of r . The function will be given as a black box which executes this unitary $U |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$. Using the following algorithm we can find the period of by just using U once. **Inputs:** The black box which performs U along with a state to store the function evaluation initialized to $|0\rangle$ and a $t = O(L + \log(1/\epsilon))$ qubit register initialized to $|0\rangle$.

Output: The least integer $r > 0$ such that $f(x+r) = f(x)$.

Procedure:

1. Initialize the state to $|0\rangle |0\rangle$.
2. Create the superposition state $\frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle |0\rangle$.
3. Apply the unitary to get this state $\frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle |f(x)\rangle \approx \frac{1}{\sqrt{r 2^t}} \sum_{l=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i l x / r} |j\rangle |\hat{f}(l)\rangle$.
4. Apply inverse Fourier transform to get $\frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} |\widetilde{l/r}\rangle |\hat{f}(l)\rangle$
5. Measure the register to get $\widetilde{l/r}$.
6. Apply the continued fractions algorithm to get the value of r .

As one can see this bears a lot of similarities to the normal order finding algorithm and interestingly the order finding algorithm is actually just an example of the period finding algorithm since it finds the period of the function $f(x) = a^x \mod N$. Matter of fact $|\hat{f}(l)\rangle = \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{-2\pi i l x / r}$ which is the Fourier transform of $f(x)$ and we can actually see that it also happens to be the eigen state of U with an eigen value of l/r so we are just doing a phase estimation of that.

Now if we define another state $|\tilde{f}(l)\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{-2\pi i l x / N} |f(x)\rangle$ and for the function $f(x+r) = f(x)$ we define a unitary $U_y |f(x)\rangle = |f(x+y)\rangle$. We can see that $|\tilde{f}(l)\rangle$ happens to be eigen states of U_y due to the shift invariance property of the Fourier transform and the eigen value would be $e^{2\pi i l y / N}$. Now if we are given a $|f(x_0)\rangle$ along with the black box for executing U_y we can use it for period finding with some modifications.

4.1.4 Discrete logarithms

The discrete logarithm problem is that if given a and $b = a^s$, we have to determine s . So first lets start with a periodic function slightly more complex than the one in the previous subsection. Say we have the function $f(x_1, x_2) = a^{s x_1 + x_2} \mod N$ and let say r is the smallest positive integer $a^r \mod N = 1$. This function's period is a tuple $(l, -ls)$. Say we have a unitary $U |x_1\rangle |x_2\rangle |y\rangle = |x_1\rangle |x_2\rangle |y \oplus f(x_1, x_2)\rangle$ we can actually find its period in a similar manner to the period finding algorithm described in the previous subsection.

So we just start with the state $|0\rangle |0\rangle |0\rangle$ over three registers where the first two have $t = O(\lceil \log r \rceil + \log(1/\epsilon))$ qubits and the third one stores the function and then create the superposition states for the first two register. We then apply the unitary on the state and that can be decomposed in a similar way as shown in step 3 of the period finding algorithms procedure to

$$\begin{aligned}
& \frac{1}{2^t} \sum_{x_1=0}^{2^t-1} \sum_{x_2=0}^{2^t-1} |x_1\rangle |x_2\rangle |f(x_1, x_2)\rangle \\
& \approx \frac{1}{2^t \sqrt{r}} \sum_{l_2=0}^{r-1} \sum_{x_1=0}^{2^t-1} \sum_{x_2=0}^{2^t-1} e^{2\pi i (s l_2 x_1 + l_2 x_2)} |x_1\rangle |x_2\rangle |\hat{f}(s l_2, l_1)\rangle \\
& = \frac{1}{2^t \sqrt{r}} \sum_{l_2=0}^{r-1} \left[\sum_{x_1=0}^{2^t-1} e^{2\pi i (s l_2 x_1)} \right] \left[\sum_{x_2=0}^{2^t-1} e^{2\pi i (l_2 x_2)} \right] |x_1\rangle |x_2\rangle |\hat{f}(s l_2, l_1)\rangle \tag{4.2}
\end{aligned}$$

We can see that applying the inverse Fourier transform and measuring the first two registers will give us $s l_2 / r$ and l_2 / r from which we can deduce s by using continued fractions algorithm.

Note that $|\hat{f}(l_1, l_2)\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i l_2 j / r} |f(0, j)\rangle$ which is the Fourier transform of $f(x_1, x_2)$. Note that l_1, l_2 must satisfy $\sum_{k=0}^{r-1} e^{2\pi i k (l_1 / s - l_2) / r} = r$ else the amplitude of $f(l_1, l_2)$ would go to zero.

Obtaining the value of s from sl_2/r and l_2/r can be done in the following manner. First if we consider the case where s and $r/\gcd(l_2, r)$ are co prime we can simply see that the two fractions we will get by continued fractions algorithm say $s'l_2'/r_1$ and l_2'/r_2 we can see that s can be obtained since the numerator of first fraction will just be an integer multiple of the second and also their denominators will be the same. The problem arises when s is not co prime with $r/\gcd(l_2, r)$ in which situation we will see that r_1' will be a factor of r_2' so we can run the algorithm twice since probability that both the times the probability of obtaining the same values becomes very small so we can then take the lcm of the denominators of the different fractions and we have a good probability of obtaining r from that is high since it would just require their numerators to be co prime and then from there we can obtain s by multiplying know fractions by the value of r and then seeing what integer multiple one is of the other.

4.1.5 Hidden subgroup problem

The hidden subgroup problem describes a set of problems of this type: let f be a function from a finitely generated group G to a finite set X such that f is constant on the cosets of a subgroup K , and distinct on each coset. Given a quantum black box for performing the unitary transform $U|g\rangle|h\rangle = |g\rangle|h \oplus (g)\rangle$, for $g \in G, h \in X$, and \oplus is an appropriately chosen binary operation on X , find a generating set for K .

Order finding, period finding, discrete logarithms are all instances of the hidden subgroup problem. For the period finding algorithm G is the set of positive integers, and $K = \{r, 2r, 3r, \dots\}$ for $r \in G$ with the function $f(x+r) = f(x)$ and X is any finite set. Also one can notice the similarities between all these algorithms so far and they can be generalized for some finite abelian group G (in fact they can also be done for finitely generated abelian groups).

The first step is applying a Fourier transform (a generalization of the Hadamard operation) to create a superposition state and over this we apply the black box for $f(x)$ to get the state $\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle |f(g)\rangle$. We now express $f(g)$ in it's Fourier basis of $|\hat{f}(l)\rangle$ so the state can be rewritten as

$$\frac{1}{|G|} \sum_{g \in G} \sum_{l=0}^{|G|-1} e^{2\pi i l g / |G|} |g\rangle |\hat{f}(l)\rangle$$

Now this looks very familiar doesn't it? For the $|\hat{f}(l)\rangle$ it will have zero amplitude if l doesn't satisfy $\sum_{h \in K} e^{2\pi i l h / |G|} = |K|$ so now if we can find the l values we can find a generating set for the hidden subgroup K however this is not exactly an easy task since when doing our regular period finding or such tasks we arrange the fraction $l/|G|$ to be such that l and $|G|$ do not have common factors but for some arbitrary G this may not at all be necessary. There is a way however, since abelian groups are isomorphic to a product of cyclic groups of prime power order, that is $G = \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \dots \times \mathbb{Z}_{p_m}$ where \mathbb{Z}_{p_i} is a group over integers $\{0, 1, \dots, p_i - 1\}$ with addition modulo over p_i so we can rewrite our phase as

$$e^{2\pi i l g / |G|} = \prod_{i=1}^M e^{2\pi i l_i' g_i / p_i}$$

And we can easily find l_i' from phase estimation and then solve the hidden subgroup problem. However finding the prime power order groups is a hard problem that is at least as hard as factoring but since we are working with a quantum computer, it can also be used for finding these prime power order groups in efficient time. It is possible to do so since each abelian group has a unique representation of this form and for an abelian group which is the set of integers this really would just boil down to factoring

4.1.6 Deutsch-Josza Algorithm and Simon's Algorithm

The Deutsch-Josza Algorithm is a fairly famous example of a hidden subgroup problem solving algorithm. The problem it solves goes like this, we are given a function $f(x)$ which takes in a n bit string and outputs either 0 or 1. Now we are given the following as a fact that this function is either a constant function or is a balanced function i.e. it gives the outputs of 0 and 1 an equal number of times. Our goal is to find out whether it is balanced or constant.

On a classical computer this would take $2^{n-1} + 1$ iterations to conclude however a quantum computer can do much better, it would take only one call of the oracle. Let the oracle be $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. We will have two registers one of n qubits and the other of only one qubit. The first register is initialized at $|0\rangle$ and the other is at $|1\rangle$. We now apply Hadamard to both registers so we get this state $\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle)$. We now apply the oracle on this state to get $\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|f(x)\rangle - |1 \oplus f(x)\rangle) = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle)$

We can now just ignore the second register and apply Hadamard on the first register again and we will get

$$\frac{1}{2^n} \sum_{y=0}^{2^n-1} \left[\sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y} \right] |y\rangle$$

We now perform a measurement on this register but now here's the thing, the probability of measuring the state $|0\rangle$ is 1 if $f(x)$ is constant but 0 if $f(x)$ is balanced so depending on whether we get the state $|0\rangle$ on measurement or not, we will find out the function.

Simon's algorithm is another such algorithm which solves the problem where we are given a blackbox for a function which is either a one-one function or is a two-one function which does mapping according to some hidden bit string s such that if $f(x_1) = f(x_2)$ then $x_1 \oplus x_2 = s$. Let the domain of this function be over bit strings of n bits then we would have to do $2^{n-1} + 1$ oracle calls on a classical computer but with a quantum computer we can do far better.

Now we have our usual unitary $U|x\rangle|y\rangle = |x\rangle|f(x) \oplus y\rangle$ where $|x\rangle$ and $|y\rangle$ are of n qubit registers. The first step is to initialize the whole state to $|0\rangle|0\rangle$ and then apply Hadamard on the first register and then apply the black box for the unitary to get $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle$. Now we measure the second register and in the case we have the

two one function then we have the state $\frac{1}{\sqrt{2}}(|x\rangle + |y\rangle)$ when we measure to get $|f(x)\rangle$ for the second register and

$y = x \oplus s$. Now we will apply Hadamard again on the first register to get $\frac{1}{\sqrt{2^{n+1}}} \sum_{z \in \{0,1\}^n} [(-1)^{x \cdot z} + (-1)^{y \cdot z}] |z\rangle$.

On measuring this state, the possible values of z satisfy $x \cdot z = y \cdot z \pmod{2}$ which reduces to $s \cdot z = 0$ and if we have n distinct values for z obtained we can find out s by Gauss elimination so this can be run at about n oracle calls which is a potential exponential speed up.

4.2 Search algorithms

Now in a classical computer the complexity of searching a list is $O(N)$ quite obviously but there is an algorithm which can do it on a Quantum computer in $O(\sqrt{N})$. Now this may not look as impressive as the exponential speedups which we obtained for Fourier transform and all but it is still quite useful and important.

4.2.1 Grover's Search Algorithm

Now say we have an indexed list of $N = 2^n$ elements and we have a function $f(x)$ over this list where $f(x) = 0$ if the element is not the solution and $f(x) = 1$ if it is solution. We are having some $\leq M \leq N$ solutions. We have an oracle which executes the following $O|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. So if we have the state $|x\rangle|-\rangle$ where $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ then applying O gives us $(-1)^{f(x)}|x\rangle|-\rangle$ so we can just ignore the second qubit and see that it makes $|x\rangle$ go to $(-1)^{f(x)}|x\rangle$.

An important point to address here is the construction of the oracle itself. If we can construct the oracle then that means we know the solutions, so then what are we searching for? Now knowing the solution and searching for it are two very different things as the goal here is to find the known solutions from the indexed list of elements. The key point is, an oracle which recognizes the solutions to a problem can be constructed without actual knowledge of the solutions and this will be our main principle in utilising this search algorithm.

For describing our circuit we will first define a Grover operator G which does the following (shown in fig 4.4) The phase happens to have the unitary $2|0\rangle\langle 0| - 1$ and on applying the Hadamard on both sides $H^{\otimes N}(2|0\rangle\langle 0| -$

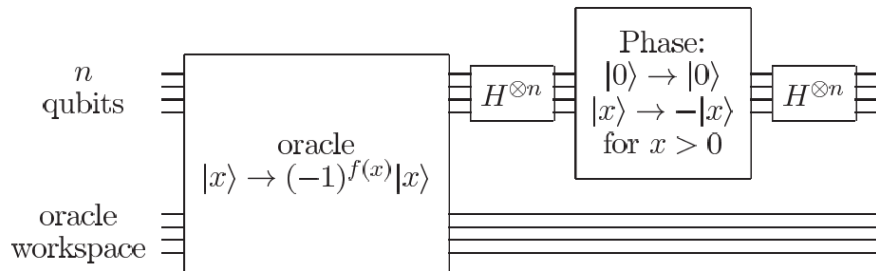


Figure 4.4: Grover iteration

1) $H^{\otimes N} = 2|\psi\rangle\langle\psi| - 1$ where $|\psi\rangle$ is the superposition state. From this we can see

$$G = (2|\psi\rangle\langle\psi| - 1)O$$

An interesting thing to note is that the unitary $2|\psi\rangle\langle\psi| - 1$ will produce an inversion about mean in the sense that only the component of the state that is parallel to the superposition has its sign unchanged. This brings us to the geometric visualization of the Grover's algorithm.

So we will now define $|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{x|f(x)=0} |x\rangle$ and $|\beta\rangle = \frac{1}{\sqrt{M}} \sum_{x|f(x)=1} |x\rangle$. So we write

$$|\psi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle \quad (4.3)$$

Now let $|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |\alpha\rangle + \sin\left(\frac{\theta}{2}\right) |\beta\rangle$, we can observe the following to hold

$$G^k |\psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right) |\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right) |\beta\rangle \quad (4.4)$$

This shows that G just happens to be a rotation in $|\alpha\rangle, |\beta\rangle$ space. The core idea of the search algorithm is to rotate $|\psi\rangle$ till it approaches $|\beta\rangle$ so essentially it would make the state into being composed of the solutions. The fig 4.5 describes the rotation idea. Now the very important question is that how many times must we apply the

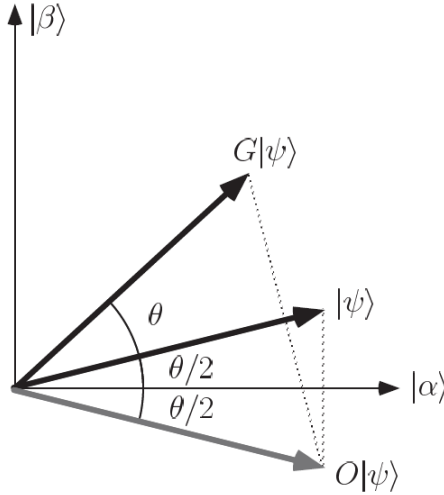


Figure 4.5: Grover rotation

operation to obtain a close estimate of $|\beta\rangle$. We can see that doing these many iterations will give us the needed state.

$$R = \text{Round}\left(\frac{\arccos\sqrt{M/N}}{\theta}\right)$$

Here Round is a function which rounds up to the nearest integer and rounds halves down. An important thing to see here is that the error will be very small if we have $M \ll N$. Now at the moment if we assume that $M \leq N/2$ then since $\theta/2 \geq \sin(\theta/2)$ we obtain the upper bound

$$R \leq \left\lceil \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rceil \quad (4.5)$$

Now this makes things a lot more convenient as we just need the number of solutions and not the actual values of them and this is where we obtain the term for complexity of circuit as $O(\sqrt{N/M})$.

The algorithm proceeds like this, we start with the state $|0\rangle^{\otimes n} |0\rangle$. We now apply Hadamard on the first register of n qubits and HX on the second register of 1 qubit. We now have the state $\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle)$.

Now we apply the Grover iteration for $R \approx \left\lceil \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rceil$ and we will obtain an approximate state to $|\beta\rangle$ in the

first register and we would have to approximately measure M times to know all the solutions. Now if we have $M \geq N/2$ then we simply add another qubit which would essentially double our N so it just goes back to $M \leq N/2$ case. Funnily according to formula of R we mentioned previously, when M increases from $N/2$ to N it would increase the number of iterations. However introducing another qubit would require us to change the oracle. The new oracle can actually be shown to be this circuit in fig. 4.6 however it requires two ancillary qubits. Given that this algorithm speeds up the search process it has a ton of applications given the fact that

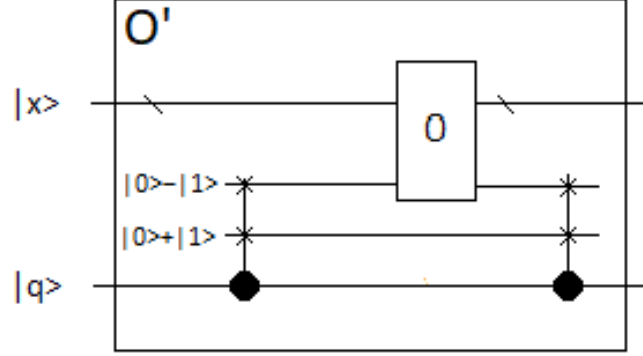


Figure 4.6: New oracle with the extra qubit

a lot of algorithms utilise a search. While the speedup is only polynomial and not exponential it is still very useful especially for speeding up an NP-complete problem. One such problem is the Hamiltonian cycle problem. This is a problem which examines a graph and tells whether it has a Hamiltonian cycle or not where a Hamiltonian cycle is a cycle through the graph where every vertex is visited once. It is widely believed to be NP-complete but it has not been proven yet. The classical algorithm goes like this:

- (1) Find every possible ordering of the vertices (v_1, \dots, v_n) of the graph while allowing repetitions as they ease the analysis.
- (2) If a certain permutation works then there is a Hamiltonian cycle else continue searching.

Since there are $n^n = 2^{(n \log n)}$ orderings possible we just need a search among these elements. Let $m = \lceil \log n \rceil$ then we have to do a search using nm qubits and have some oracle which does $O|v_1, \dots, v_n\rangle = |v_1, \dots, v_n\rangle$ if it isn't a Hamiltonian cycle and $O|v_1, \dots, v_n\rangle = -|v_1, \dots, v_n\rangle$ if it is and as we know this method will require complexity of $O(2^{mn/2})$ which is a speedup over the classical approach.

4.2.2 Hamiltonian for search algorithm

Now this algorithm looks real good and all but how did we reach here? For explaining the approach we will start with some state $|\psi\rangle$ and find a Hamiltonian which will evolve this state to $|x\rangle$ where this state is the solution (for now lets just assume one solution) in some time Δt . Once we find this Hamiltonian we simply construct a circuit for simulating it.

Now if we begin making guesses for the Hamiltonians we can see that the terms would be constructed with $|\psi\rangle$ and $|x\rangle$. Now we write two Hamiltonians

$$H = |x\rangle \langle x| + |\psi\rangle \langle \psi| \quad (4.6)$$

$$H = |x\rangle \langle \psi| + |\psi\rangle \langle x| \quad (4.7)$$

Now the interesting thing is that both of these can be used to describe the search Hamiltonian but for now we will focus on the one in eq. 4.6. Now we just have to run this simulation till time t for which $\exp(-iHt)|\psi\rangle = |x\rangle$. Now if we take $|x\rangle$ and $|y\rangle$ to be the orthonormal basis of this space then we can write $|\psi\rangle = \alpha|x\rangle + \beta|y\rangle$ where $\alpha^2 + \beta^2 = 1$. When we write H in this basis we get $H = I + \alpha(\beta X + \alpha Z)$. We can see that

$$\exp(-iHt)|\psi\rangle = \exp(it)[\cos(\alpha t)|\psi\rangle - i\sin(\alpha t)|x\rangle]$$

Now we can simply ignore the phase factor and note that for $t = \pi/2\alpha$ we will get the result $|x\rangle$ on measurement with unity probability. Now this unfortunately depends on the quantity α which is dependent on $|\psi\rangle$ and $|x\rangle$ but if we choose $|\psi\rangle$ to be the uniform superposition state then we know that α will be the same for all choices of $|x\rangle$ and Voila! This is pretty much how the search algorithm works!

Alright so that's all good but what about the circuit? Well we just need to follow the method described in section 3.3 and we can see the following circuits can be used The complexity of the circuit will depend on the accuracy we are aiming which will decide how small the time steps we will choose. We can see that if we wish

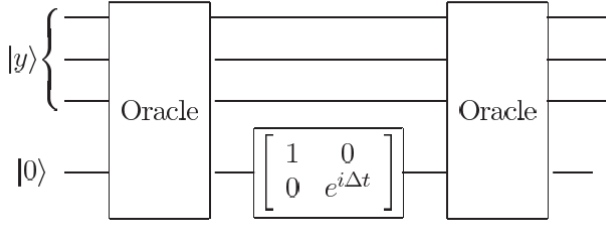


Figure 4.7: Circuit for executing $\exp(-i|x\rangle\langle x|\Delta t)$

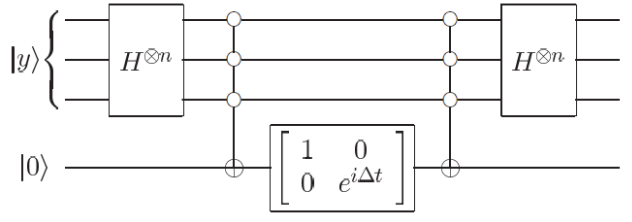


Figure 4.8: Circuit for executing $\exp(-i|\psi\rangle\langle\psi|\Delta t)$

to have an accuracy of $O(\Delta t^r)$ the cumulative error will come out to be $O(\Delta t^r \sqrt{N})$ and since we want the error to be $O(1)$ we choose $\Delta t = \Theta(N^{-1/2(r-1)})$ so then we have the number of steps to be $t/\Delta t$ and the complexity of the whole circuit will be $O(\sqrt{N} \times N^{1/2(r-1)})$ which gives us $O(N^{r/2(r-1)})$ and we can see that the limit goes to $O(\sqrt{N})$ for r tending to infinity.

A thing to note is that here in the analysis of Δt we have used very general methods and we can use a specific choice to actually do much better. Let us define a $U(\Delta t) = \exp(-i|\psi\rangle\langle\psi|\Delta t) \exp(-i|x\rangle\langle x|\Delta t)$ on doing a simple calculation we get the following for $U(\Delta t)$ (note $|x\rangle = \hat{z}$ in this basis)

$$U(\Delta t) = \left(\cos^2\left(\frac{\Delta t}{2}\right) - \sin^2\left(\frac{\Delta t}{2}\right) \vec{\psi} \cdot \hat{z} \right) I - 2i \sin\left(\frac{\Delta t}{2}\right) \left(\cos\left(\frac{\Delta t}{2}\right) \frac{\vec{\psi} + \hat{z}}{2} + \sin\left(\frac{\Delta t}{2}\right) \frac{\vec{\psi} \times \hat{z}}{2} \right) \quad (4.8)$$

Here $\vec{\psi} = (2\alpha\beta, 0, (\alpha^2 - \beta^2))$ which is just the Bloch sphere representation. Now we can see that $U(\Delta t)$ describes a rotation about some axis in the Bloch sphere. On more observation we can see that it rotates about the axis $\vec{r} = \cos\left(\frac{\Delta t}{2}\right) \frac{\vec{\psi} + \hat{z}}{2} + \sin\left(\frac{\Delta t}{2}\right) \frac{\vec{\psi} \times \hat{z}}{2}$ by an angle θ such that $\cos\left(\frac{\theta}{2}\right) = \cos^2\left(\frac{\Delta t}{2}\right) - \sin^2\left(\frac{\Delta t}{2}\right) \vec{\psi} \cdot \hat{z}$

We can also see that $\vec{\psi} \cdot \vec{r} = \hat{z} \cdot \vec{r}$ so a rotation about the axis \vec{r} will make $\vec{\psi}$ go to \hat{z} and the angle for rotation is θ for which $\cos\left(\frac{\theta}{2}\right) = 1 - \frac{2}{N} \sin^2\left(\frac{\Delta t}{2}\right)$.

Now initially we took Δt to be small but if we put it as π we maximize the rotation angle and that way we get $\theta \approx 4/\sqrt{N}$ and the number of oracle calls would be $O(\sqrt{N})$ and in fact if we put $\Delta t = \pi$ we actually get the Grover iteration with $\exp(-i|\psi\rangle\langle\psi|\Delta t) = I - 2|\psi\rangle\langle\psi|$ and $\exp(-i|x\rangle\langle x|\Delta t) = I - 2|x\rangle\langle x|$.

In the case of multiple solutions we can use the very same approach by just changing $|x\rangle$ to the superposition of the solutions and we would approach the same Grover iteration idea.

4.2.3 Quantum Counting

An important thing to note about Grover's algorithm is that it requires us to know the number of solutions but what about the case where we do not know this? Here we use phase estimation for this. So we want to estimate the eigen values of the Grover iteration for this.

Firstly let's take the appended oracle approach so we can be sure about $M \leq N/2$. From Eq. 4.4 we can see that since G is just a rotation operator, its eigen values are $e^{i\theta}$ and $e^{i(2\pi-\theta)}$ and $\sin^2(\theta/2) = M/2N$. Here $t = m + \lceil \log(2 + 1/2\epsilon) \rceil$ where we want to estimate θ to an accuracy of m bits so $|\Delta\theta| \leq 2^{-m}$ and a probability of $1 - \epsilon$. Now we will get some error for M which will be $|\Delta M|$

$$\frac{|\Delta M|}{2N} = \left| \sin^2\left(\frac{\theta + \Delta\theta}{2}\right) - \sin^2\left(\frac{\theta}{2}\right) \right| \quad (4.9)$$

$$\frac{|\Delta M|}{2N} < \left(2 \sin\left(\frac{\theta}{2}\right) + \frac{\Delta\theta}{2} \right) \frac{|\Delta\theta|}{2} \quad (4.10)$$

Substituting $\sin^2(\theta/2) = M/2N$ and $|\Delta\theta| \leq 2^{-m}$ we have

$$\frac{|\Delta M|}{2N} < \left(\sqrt{2MN} + \frac{N}{2^{m+1}} \right) 2^{-m} \quad (4.11)$$

We can in $\Theta(\sqrt{N})$ oracle calls get our accuracy to be $O(\sqrt{M})$ and this is a clear improvement over the classical approach which would require $O(N)$ oracle calls. Also when we run this along with using the Grover's algorithm if we take $m = \lceil n/2 \rceil + 1$ we get an angular error for R as $\pi/4(1 + |\Delta\theta|/\theta) = \pi/4 \times 3/2 = 3\pi/8$ and combining with success probability of the phase estimation for $\epsilon = 1/6$ we get a success probability of $5/6 \times \cos^2 3\pi/8 \approx 0.12$ which can be improved by a few repetitions of the combined search procedure.

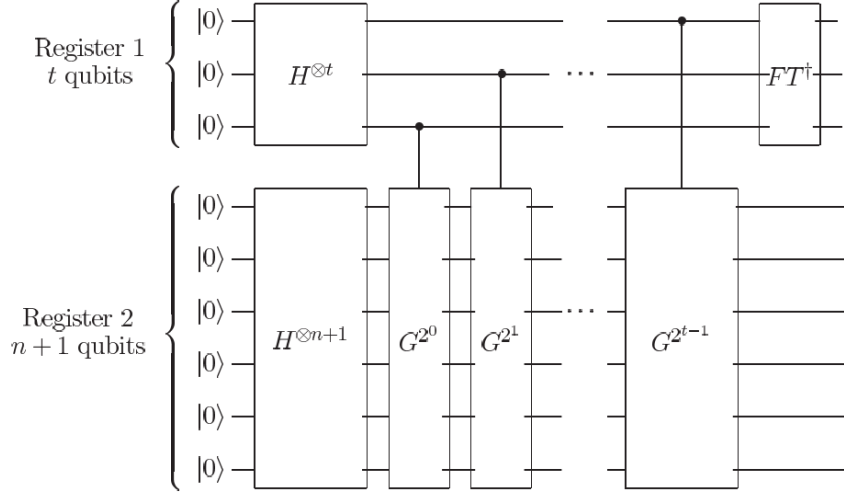


Figure 4.9: Phase estimation for quantum counting

4.2.4 Searching an unstructured database

So far we have conducted searches on structured databases but now we will demonstrate that we can get a speedup even on an unstructured database. We define our problem as follows, we have a database of $N = 2^n$ elements labeled as d_1, d_2, \dots, d_N each of which are l bit strings and we wish to find a specific string s .

When we are running this on a classical computer we consider it to have two components, a CPU where data manipulation is done and it has a small temporary memory and then there is a large memory which stores the database. The CPU can access data from the memory and store it and manipulate it. The method used here is that the elements are given an n bit index for each of them where $n = \lceil \log N \rceil$ and we start with 0 and in each iteration we take one element from the large memory to the CPU and check if it is the required string. If it is then we stop else we continue to the next index and repeat. This is very clearly the most efficient algorithm for a classical computer.

For a Quantum computer we can use a similar architecture of a CPU and a memory. This CPU has 4 registers, (1) an n qubit register initialized to $|0\rangle$, (2) an l qubit register initialized to $|s\rangle$ which remains in that for the rest of the computation, (3) Another l qubit register initialized to $|0\rangle$ and (4) a 1 qubit register initialized to $|-\rangle$.

The load operation will take the state $|x\rangle |s\rangle |d\rangle |-\rangle$ to $|x\rangle |s\rangle |d \oplus d_x\rangle |-\rangle$ so just we go from $|x\rangle |s\rangle |0\rangle |-\rangle$ to $|x\rangle |s\rangle |d_x\rangle |-\rangle$ and our oracle takes $|x\rangle |s\rangle |d_x\rangle |-\rangle$ to $-|x\rangle |s\rangle |d_x\rangle |-\rangle$ if $d_x = s$ else it does nothing. The addressing scheme will be a quantum addressing scheme also. An important thing to note is that we can load all the data as a superposition and will do the search in $O(\sqrt{N})$ operations but the amount of resources needed for storing the data is pretty much the same as that of a classical computer but these resources are far more cheaper for a classical computer which makes this not all that economically viable also the Grover's algorithm has better uses as speeding up NP complete problems like the Hamiltonian cycle problem.

4.2.5 Optimality of the algorithm and Black box limits

Now we know that the search algorithm requires $O(\sqrt{N})$ queries but it turns out that we cannot actually do any better than that. Suppose we start with some $|\psi\rangle$ and for the sake of simplicity we stick to one solution $|x\rangle$ and our oracles $O_x = I - 2|x\rangle\langle x|$ and we use the oracle exactly k times and perform U_1, U_2, \dots, U_k between each of these oracles. We now define

$$|\psi_k^x\rangle = U_k O_x U_{k-1} O_x \dots U_1 O_x |\psi\rangle \quad (4.12)$$

$$|\psi_k\rangle = U_k U_{k-1} \dots U_1 |\psi\rangle \quad (4.13)$$

Our goal is to bound the quantity $D_k = \sum_x ||\psi_k^x\rangle - |\psi_k\rangle||^2$ which is essentially the deviation after k steps without using the oracles. If D_k is small then the states would be very hard to distinguish in between so this gives us a clue of how much oracles are required to be used.

We now prove inductively $D_k \leq 4k^2$

$$D_{k+1} = \sum_x ||O_x(|\psi_k^x\rangle - |\psi_k\rangle) + (O_x - I)|\psi_k\rangle||^2 \quad (4.14)$$

We can write $(O_x - I) |\psi_k\rangle = -2 |x\rangle \langle x | \psi_k \rangle |x\rangle$

$$D_{k+1} \leq \sum_x (||(|\psi_k^x\rangle - |\psi_k\rangle)||^2 + 4||(|\psi_k^x\rangle - |\psi_k\rangle)|| | \langle x | \psi_k \rangle | + 4 | \langle x | \psi_k \rangle |^2) \quad (4.15)$$

Applying Cauchy Shwarz inequality on the second term on RHS and noting $\sum_x | \langle x | \psi_k \rangle |^2 = 1$ gives us

$$D_{k+1} \leq D_k + 4 \left(\sum_x ||(|\psi_k^x\rangle - |\psi_k\rangle)||^2 \right)^{\frac{1}{2}} \left(\sum_x | \langle x | \psi_k \rangle |^2 \right)^{\frac{1}{2}} + 4 \quad (4.16)$$

$$D_{k+1} \leq D_k + 4\sqrt{D_k} + 4 \quad (4.17)$$

Clearly if we assume $D_k \leq 4k^2$ then we get from eq. 4.17 that $D_{k+1} \leq 4(k+1)^2$. Now lets assume that $| \langle x | \psi_k^x \rangle |^2 \geq 1/2$ so we get our search result at least with 0.5 probability. We now define $E_k = \sum_x || |\psi_k^x\rangle - |x\rangle ||^2$ and $F_k = \sum_x || |x\rangle - |\psi_k\rangle ||^2$. We can see that $E_k \leq (2 - \sqrt{2})N$ since $|| |\psi_k^x\rangle - |x\rangle ||^2 = 2 - 2| \langle x | \psi_k^x \rangle |$ and also $F_k \geq 2N - 2\sqrt{N}$ which we can obtain using the Cauchy Shwarz inequality. We know

$$D_k = \sum_x ||(|\psi_k^x\rangle - |x\rangle) + (|x\rangle - |\psi_k\rangle)||^2 \quad (4.18)$$

$$D_k \geq \sum_x || |\psi_k^x\rangle - |x\rangle ||^2 + \sum_x || |x\rangle - |\psi_k\rangle ||^2 - 2 \sum_x || |\psi_k^x\rangle - |x\rangle || || |x\rangle - |\psi_k\rangle || \quad (4.19)$$

From Cauchy Shwarz inequality we know $\sum_x || |\psi_k^x\rangle - |x\rangle || || |x\rangle - |\psi_k\rangle || \leq \sqrt{E_k F_k}$ so we can write eq 4.19 as

$$D_k \geq E_k + F_k - 2\sqrt{E_k F_k} = (\sqrt{E_k} - \sqrt{F_k})^2 \quad (4.20)$$

Now if we use the inequalities of E_k and F_k we get that $D_k \geq cN$ when N is sufficiently large and c is any constant lower than $(\sqrt{2} - \sqrt{2 - \sqrt{2}})^2 \approx 0.42$ and since we established $D_k \leq 4k^2$ we get that $k \geq \sqrt{cN/4}$ so we can see that we get the complexity of $O(\sqrt{N})$ however we cannot get a greater improvement clearly.

While this means that we found the best possible quantum search algorithm, it also means that we cannot get a speedup like that of $O(\log N)$ which would be a dream result. Many researchers believe that NP - complete problems involve unstructured searches so a speedy search algorithm can help solve this but this would be pretty bad for Quantum computing since then it would mean that BPQ does not contain NP complete however that is not necessary since factoring which is a little more difficult than NP but not NP complete (a class called NPI) has an efficient implementation here so it might just be a matter of time till we find some quantum approach for NP complete.

Now if we were to define the decision problem for whether a solution exists or not i.e. given $f(x)$ of the oracle does there exist x such that $f(x) = 1$. $F(X) = X_0 \vee X_1 \vee \dots \vee X_{n-1}$ where $X_k = f(k)$. This decision problem can be shown to be equivalent to the search problem for the given oracle itself. Now if we try to generalize this $F(X)$ for operations other than *OR* to even go over for *AND*, *PARITY*, *MAJORITY* we obtain some interesting limits for quantum query complexity.

Let the deterministic query complexity be $D(F)$ and the equivalent for quantum computers we can take to be $Q_E(F)$. The quantity we are concerned with is $Q_2(F)$ which is the query complexity for an accuracy of $2/3$. This $2/3$ is just arbitrarily chosen since our only concern is to have it larger than $1/2$ so we can make it tend to 1 after running it multiple times. Note that $Q_2(F) \leq Q_E(F) \leq D(F) \leq N$. Now since these are boolean functions, we have $X_k^2 = X_k$ so using this we can define the minimum degree polynomial of $F(X)$ as $p(X)$

$$p(X) = \sum_{Y \in \{0,1\}^N} F(Y) \prod_{k=0}^{N-1} [1 - (Y_k - X_k)^2] \quad (4.21)$$

Quite trivially this will be a unique representation otherwise it wouldn't be the minimum degree polynomial. We will denote the minimum degree of $F(X)$ as $\deg(F)$. The degree of *OR*, *AND*, *PARITY* are actually N and in fact most functions have degree of order N and it has also been proven that

$$D(F) \leq 2\deg(F)^4 \quad (4.22)$$

and also

$$D(F) \leq 216\deg(F)^6 \quad (4.23)$$

Now let us take the situation where we have performed T oracle queries in the quantum computer so our state is

now $\sum_{k=0}^{2^n-1} c_k |k\rangle$. We will prove that c_k are polynomials of at most degree T . Let us start with the state $|\psi_0\rangle = \sum_{ij} (a_{i0j} |i\rangle |0\rangle + a_{i1j} |i\rangle |1\rangle) |j\rangle$ where the first label is of n qubit register followed by a 1 qubit register and then a $m - n - 1$ qubit register. After the oracle query we will obtain $|\psi_1\rangle = \sum_{ij} (a_{i0j} |i\rangle |X_i\rangle + a_{i1j} |i\rangle |X_i \oplus 1\rangle) |j\rangle$ which we can write as

$$|\psi_1\rangle = \sum_{ij} [((1 - X_i)a_{i0j} + X_i a_{i1j}) |i0\rangle + ((1 - X_i)a_{i1j} + X_i a_{i0j}) |i1\rangle] |j\rangle \quad (4.24)$$

An important point to note is that the in between unitary operations do not change the degree and also here we can see in eq. 4.24 that the coefficients are of degree 1 and on extending this we can see that the coefficients after T queries will be of order less than or equal to T . The probability of any of the states would be $|c_k|^2$ which would be of degree less than or equal to $2T$ and so the total probability of obtaining a 1 from the oracle would also be a polynomial of degree at most $2T$ since it is just the sum over the subsets of these probabilities. This probability polynomial would equal $F(X)$ in the case where we are certain of having executed $F(X)$ so we would get $\deg(F) \leq 2T$ so we get $Q_E(F) \geq \deg(F)/2$ and if we have that the probability polynomial approximates $F(X)$ it still would have the same limits on degree so we also get $Q_2(F) \geq \deg(F)/2$ and from eq. 4.22 and 4.23 we get

$$Q_E(F) \geq \left[\frac{D(F)}{32} \right]^{1/4}, \quad Q_2(F) \geq \left[\frac{D(F)}{13824} \right]^{1/6}$$

So as we can see that without modifying the black box itself we cannot achieve anything faster than a polynomial speedup for this kind of problems.

4.2.6 Interesting applications of Grover's algorithm

Apart from also being used for speeding up NP complete problems, we can also develop an algorithm for finding the smallest number from a set of N numbers which are indexed as $T[i]$ for $i \in 0, 1..N - 1$. The procedure for it goes something like this when we assume a success probability of $1/2$:

1. Choose threshold index $0 \leq y \leq N - 1$ uniformly at random.
2. Repeat the following and interrupt it when the total running time is more than $22.5\sqrt{N} + 1.4 \log^2 N$. Then go to stage 2(c).
 - (a) Initialize the memory as $\sum_j \frac{1}{N} |j\rangle |y\rangle$. Mark every item j for which $T[j] < T[y]$.
 - (b) Apply the quantum exponential searching algorithm of [2].
 - (c) Observe the first register: let y' be the outcome. If $T[y'] < T[y]$, then set threshold index y to y' .
3. Return y .

For the step 2(a) it would in convention take $\log N$ steps and the search algorithm we use has complexity of $O(\sqrt{N})$ as we have already established so for a large N the complexity is $O(\sqrt{N})$ also the algorithm would on average require $O(\log(N)\sqrt{N})$ accesses to the database. It must be noted that the algorithm can run infinitely long so we will call that case the infinite algorithm. There are some lemmas of this we can see:

Lemma 1: *Let $p(t, r)$ be the probability that the index of the element of rank r will ever be chosen when the infinite algorithm searches among t elements. Then $p(t, r) = 1/r$ if $r \leq t$, and $p(t, r) = 0$ otherwise*

For $r > t$ it is trivial. We can prove this using induction for $r \leq t$. Lets assume $p(k, r) = 1/r$ for $k \in [r, t]$ if we mark $t + 1$ elements then we have

$$p(t + 1, r) = \frac{1}{t + 1} + \sum_{k=r+1}^{t+1} \frac{1}{t + 1} p(k - 1, r) = \frac{1}{r}$$

Lemma 2: *The expected total time used by the infinite algorithm before y holds the index of the minimum is at most $m_0 = \frac{45}{4}\sqrt{N} + \frac{7}{10}\log^2 N$*

The expected number of iterations used by the exponential searching algorithm is at most $\frac{9}{2}\sqrt{N}/t$ for finding the index of marked item. So the expected number of minimum steps would be

$$\begin{aligned} \sum_{r=2}^N p(N, r) \frac{9}{2} \sqrt{\frac{N}{r-1}} &= \frac{9}{2} \sqrt{N} \sum_{r=1}^{N-1} \frac{1}{r+1} \frac{1}{\sqrt{r}} \\ \sum_{r=2}^N p(N, r) \frac{9}{2} \sqrt{\frac{N}{r-1}} &\leq \frac{9}{2} \sqrt{N} \left(\frac{1}{2} + \sum_{r=2}^{N-1} r^{-3/2} \right) \leq \frac{9}{2} \sqrt{N} \left(\frac{1}{2} + \int_{r=2}^{N-1} r^{-3/2} \right) \leq \frac{45}{4} \sqrt{N} \end{aligned}$$

The expected number of steps from stage 2(a) before $T[y]$ hold the minimum is at most

$$\sum_{r=2}^N p(N, r) \log N = (H_N - 1) \log N \leq \ln N \log N \leq 0.7 \log^2 N$$

Here H_N is the N th harmonic number. After at most $2m_0$ iterations $T[y]$ will hold the value with a probability of $1/2$ so here we can see why this only requires $O(\sqrt{N})$ complexity. Refer no. 12 in references for more about this algorithm.

References and other links

1. *Quantum Computation and Quantum Information* by Isaac Chuang and Michael Nielsen.
2. *Qiskit Textbook* of IBM available [here](#).
3. *Quantum Information Science 1, Part 1* edX course available [here](#).
4. *Notes on Bloch Sphere* available [here](#).
5. *Notes on Schmidt Decomposition* available [here](#).
6. *The original EPR paper* available [here](#) and the [wikipedia article](#).
7. *On the Einstein Podolsky Rosen Paradox* by J.S. Bell available [here](#).
8. *Tsirelson's Bound* wikipedia article available [here](#).
9. *Using Quantum Gates instead of ancilla bits* article available [here](#).
10. *Elementary gates for quantum computation* by Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John Smolin, and Harald Weinfurter available [here](#).
11. *The Hidden Subgroup Problem and Eigenvalue Estimation on a Quantum Computer* by Michele Mosca, Artur Ekert available [here](#).
12. *Quantum Algorithms Revisited* by R. Cleve¹, A. Ekert, C. Macchiavello and M. Mosca available [here](#).
13. *A quantum algorithm for finding the minimum* by Christoph Dürr and Peter Høyer available [here](#).
14. *Quantum Counting* by Gilles Brassard, Peter Høyer, and Alain Tapp available [here](#).
15. [Here is the repository](#) which has my solutions to the exercises 3 and 4 of the IBM Quantum Challenge.