Chapter 2

Quantum Computing Principles

The massive amount of processing power generated by computer manufacturers has not yet been able to quench our thirst for speed and computing capacity. In 1947, American computer engineer Howard Aiken said that just six electronic digital computers would satisfy the computing needs of the United States. Others have made similar errant predictions about the amount of computing power that would support our growing technological needs. Of course, Aiken didn't count on the large amounts of data generated by scientific research, the proliferation of personal computers or the emergence of the Internet, which have only fuelled our need for more, more and more computing power.

Will we ever have the amount of computing power we need or want? If, as Moore's Law states, the number of transistors on a microprocessor continues to double every 18 months, the year 2020 or 2030 will find the circuits on a microprocessor measured on an atomic scale. And the logical next step will be to create quantum computers, which will harness the power of atoms and molecules to perform memory and processing tasks. Quantum computers have the potential to perform certain calculations significantly faster than any silicon-based computer.

Scientists have already built basic quantum computers that can perform certain calculations; but a practical quantum computer is still years away. In this chapter, we explore what a quantum computer is and how it operates.

2.1 Qubit - The Quantum Bit

In quantum computing, a qubit or quantum bit is a unit of quantum information the quantum analogue of the classical bit.

2.1.1 Bits vs. Qubits

A bit is the basic unit of information. It is used to represent information by computers. Regardless of its physical realization, a bit is always understood to be either a 0 or a 1. An analogy to this is a light switch with the off position representing 0 and the on position representing 1.

A qubit is a two-state quantum-mechanical system, such as the polarization of a single photon: here the two states are vertical polarization and horizontal polarization. It has a few similarities to a classical bit, but is overall very different. Like a bit, a qubit can have two possible values normally a 0 or a 1. The difference is that whereas a bit must be either 0 or 1, a qubit can be 0, 1, or a superposition of both.

2.1.2 Superposition

Think of a qubit as an electron in a magnetic field. The electron's spin may be either in alignment with the field, which is known as a spin-up state, or opposite to the field, which is known as a spin-down state. Changing the electron's spin from one state to another is achieved by using a pulse of energy, such as from a laser - let's say that we use 1 unit of laser energy. But what if we only use half a unit of laser energy and completely isolate the particle from all external influences? According to quantum law, the particle then enters a superposition of states, in which it behaves as if it were in both states simultaneously. Each qubit utilized could take a superposition of both 0 and 1.

The principle of quantum superposition states that if a physical system may be in one of many configurations arrangements of particles or fields then the most general state is a combination of all of these possibilities, where the amount in each configuration is specified by a complex number.

2.1.3 Representation

The two states in which a qubit may be measured are known as basis states (or basis vectors). As is the tradition with any sort of quantum states, Dirac, or *bra-ket* notation, is used to represent them. This means that the two computational basis states are conventionally written as $|0\rangle$ and $|1\rangle$ (pronounced "ket 0" and "ket 1"). A pure qubit state is a linear quantum superposition of the basis states. This means that the qubit can be represented as a linear combination of $|0\rangle$ and $|1\rangle$:

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

where α and β are probability amplitudes and can in general both be complex numbers.

The possible states for a single qubit can be visualised using a Bloch sphere as shown in Figure 2.1 [1]. Represented on such a sphere, a classical bit could only be at the "North Pole" or the "South Pole", in the locations where $|0\rangle$ and $|1\rangle$ are, respectively. The rest of the surface of the sphere is inaccessible to a classical bit, but a pure qubit state can be represented by any point on the surface. For example, the pure qubit state $\frac{|0\rangle+i|1\rangle}{\sqrt{2}}$ would lie on the equator of the sphere, on the positive y-axis.

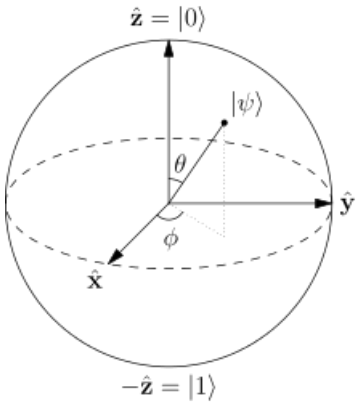

Figure 2.1: Sphere representation for a qubit in the state: $\alpha = \cos\left(\frac{\theta}{2}\right)$ and $\beta = e^{i\phi}\sin\left(\frac{\theta}{2}\right)$

2.2 Quantum States

2.2.1 Entanglement

An important distinguishing feature between a qubit and a classical bit is that multiple qubits can exhibit quantum entanglement. Entanglement is a non-local property that allows a set of qubits to express higher correlation than is possible in classical systems. Take, for example, two entangled qubits in the Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

Imagine that these two entangled qubits are separated, with one each given to Alice and Bob. Alice makes a measurement of her qubit, obtaining $|0\rangle$ or $|1\rangle$.

[1] Source:`http://en.wikipedia.org/wiki/Bloch_sphere`

Because of the qubits' entanglement, Bob must now get exactly the same measurement as Alice; i.e., if she measures a $|0\rangle$, Bob must measure the same, as $|00\rangle$ is the only state where Alice's qubit is a $|0\rangle$.

This is a real phenomenon (Einstein called it **"spooky action at a distance"**), the mechanism of which cannot, as yet, be explained by any theory - it simply must be taken as given. Quantum entanglement allows qubits that are separated by incredible distances to interact with each other instantaneously (not limited to the speed of light). No matter how great the distance between the correlated particles, they will remain entangled as long as they are isolated.

Entanglement also allows multiple states (such as the Bell state mentioned above) to be acted on simultaneously, unlike classical bits that can only have one value at a time. Entanglement is a necessary ingredient of any quantum computation that cannot be done efficiently on a classical computer. Many of the successes of quantum computation and communication, such as quantum teleportation and superdense coding, make use of entanglement, suggesting that entanglement is a resource that is unique to quantum computation.

2.2.2 Registers

A number of entangled qubits taken together is a qubit register. Quantum computers perform calculations by manipulating qubits within a register. An example of a 3-qubit register:

Consider first a classical computer that operates on a three-bit register. The state of the computer at any time is a probability distribution over the $2^3 = 8$ different three-bit strings $000, 001, 010, 011, 100, 101, 110, 111$. If it is a deterministic computer, then it is in exactly one of these states with probability 1. However, if it is a probabilistic computer, then there is a possibility of it being in any one of a number of different states. We can describe this probabilistic state by eight non-negative numbers A,B,C,D,E,F,G,H (where A = probability computer is in state 000, B = probability computer is in state 001, etc.). There is a restriction that these probabilities sum to 1.

The state of a three-qubit quantum computer is similarly described by an eight-dimensional vector (a,b,c,d,e,f,g,h), called a ket. However, instead of the sum of the coefficient magnitudes adding up to one, the sum of the squares of the coefficient magnitudes, $|a|^2 + |b|^2 + ... + |h|^2$, must equal one. Moreover, the coefficients can have complex values. Since the absolute square of these complex-valued coefficients denote probability amplitudes of given states, the phase between any two coefficients (states) represents a meaningful parameter, which presents a fundamental difference between quantum computing and probabilistic classical computing.

Now, an eight-dimensional vector can be specified in many different ways depending on what basis is chosen for the space. The basis of bit strings (e.g., 000, 001, ..., 111) is known as the computational basis. Other possible bases are unit-length, orthogonal vectors, etc. Ket notation is often used to make the choice of basis explicit.

For example, the state (a,b,c,d,e,f,g,h) in the computational basis can be written as: $a|000\rangle + b|001\rangle + c|010\rangle + d|011\rangle + e|100\rangle + f|101\rangle + g|110\rangle + h|111\rangle$ where, e.g., $|010\rangle = (0, 0, 1, 0, 0, 0, 0, 0)$.
Similarly, the computational basis for a single qubit (two dimensions) is $|0\rangle = (1, 0)$ and $|1\rangle = (0, 1)$.

Taken together, quantum superposition and entanglement create an enormously enhanced computing power. Where a 2-bit register in an ordinary computer can store only one of four binary configurations (00, 01, 10, or 11) at any given time, a 2-qubit register in a quantum computer can store all four numbers simultaneously, because each qubit represents two values. If more qubits are added, the increased capacity is expanded exponentially.

2.3 Operators - Quantum Gates

2.3.1 Reversible Logic Gates

Ordinarily, in a classical computer, the logic gates other than the NOT gate are not reversible. Thus, for instance, for an AND gate one cannot recover the two input bits from the output bit; for example, if the output bit is 0, we cannot tell from this whether the input bits are 0,1 or 1,0 or 0,0.

In quantum computing and specifically the quantum circuit model of computation, a quantum gate (or quantum logic gate) is a basic quantum circuit operating on a small number of qubits. They are the building blocks of quantum circuits, like classical logic gates are for conventional digital circuits. Unlike many classical logic gates, **quantum logic gates are reversible**. However, classical computing can be performed using only reversible gates. For example, the reversible Toffoli gate can implement all Boolean functions. This gate has a direct quantum equivalent, showing that **quantum circuits can perform all operations performed by classical circuits**.

2.3.2 Matrix Operator Correspondence

We can treat an n-qubit state as a vector consisting of 2^n complex numbers, each representing the coefficient of a state from the computational basis. Now, a gate operates on such a state and yields another of the same dimension. So, a gate can

be seen as a function that transforms a 2^n dimensional vector to another. Hence, in the vector-matrix representation in n-qubit space, a gate is a square matrix of dimensions 2^n, whose i^{th} column is the vector that results when we apply the gate on the i^{th} element of the computational basis.

For a quantum computer gate, we require a very special kind of reversible function, namely a unitary mapping, that is, a mapping on the state-space that preserves the inner product. So, if H is a gate and $|\psi\rangle$ and $|\phi\rangle$ represent two quantum states in n-qubit space, then $\psi^{'} = H|\psi\rangle$ and $\phi^{'} = H|\phi\rangle$ will also be n-qubit states and will satisfy the property that $\langle\psi^{'}|\phi^{'}\rangle = \langle\psi|\phi\rangle$, where $\langle..|..\rangle$ denotes the inner-product in bra-ket notation.

Hence, quantum logic gates are represented by unitary matrices. Note - a complex square matrix U is unitary if $U^*U = UU^* = I$, where I is the identity matrix and U^* is the conjugate transpose of U. The real analogue of a unitary matrix is an orthogonal matrix.

The most common quantum gates operate on spaces of one, two or three qubits. This means that as matrices, quantum gates can be described by $2X2$ or $4X4$ or $8X8$ unitary matrices.

2.3.3 Commonly used gates

Quantum gates are usually represented as matrices. A gate which acts on k qubits is represented by a 2^kX2^k unitary matrix. The number of qubits in the input and output of the gate have to be equal. The action of the quantum gate is found by multiplying the matrix representing the gate with the vector which represents the quantum state.

- **Hadamard gate**
  The Hadamard gate acts on a single qubit. It maps the basis state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, and represents a rotation of π about the axis $(\hat{x} + \hat{z})/\sqrt{2}$. It is represented by the Hadamard matrix:

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

  Since $HH^* = I$ where I is the identity matrix, H is indeed a unitary matrix.

- **Controlled Gates**
  Controlled gates act on 2 or more qubits, where one or more qubits act as a control for some operation. For example, the controlled NOT gate (or CNOT) acts on 2 qubits, and performs the NOT operation on the second qubit only when the first qubit is $|1\rangle$, and otherwise leaves it unchanged. It is represented by the matrix:

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

More generally if U is a gate that operates on single qubits with matrix representation

$$U = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix},$$

then the controlled-U gate is a gate that operates on two qubits in such a way that the first qubit serves as a control. It maps the basis states as follows:
$|00\rangle \mapsto |00\rangle$
$|01\rangle \mapsto |01\rangle$
$|10\rangle \mapsto |1\rangle U|0\rangle = |1\rangle \left(x_{00}|0\rangle + x_{10}|1\rangle\right)$
$|11\rangle \mapsto |1\rangle U|1\rangle = |1\rangle \left(x_{01}|0\rangle + x_{11}|1\rangle\right)$

The matrix representing the controlled U is:

$$\text{C}(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & x_{00} & x_{01} \\ 0 & 0 & x_{10} & x_{11} \end{bmatrix}$$

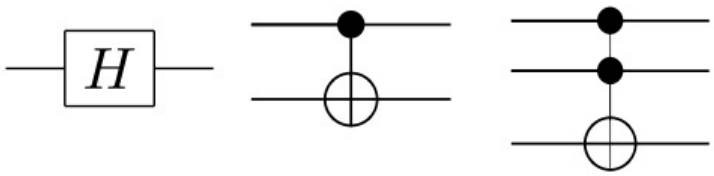

Figure 2.2: Circuit representation of Hadamard, CNOT and Toffoli gates, respectively

- **Toffoli Gate**
  The Toffoli gate, also CCNOT gate, is a 3-bit gate, which is universal for classical computation. The quantum Toffoli gate is the same gate, defined for 3 qubits. If the first two bits are in the state $|1\rangle$, it applies a Pauli-X (bit inversion) on the third bit, else it does nothing. It is an example of a controlled gate. It swaps the states $|110\rangle$ and $|111\rangle$; it is an identity map for the other 6 states in the computational basis for a 3-qubit space. The matrix representation is:

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

It can be also described as the gate which maps $|a, b, c\rangle$ to $|a, b, c \oplus ab\rangle$.

2.3.4 Quantum Fourier Transform

This is a linear transformation on quantum bits, and is the quantum analogue of the discrete Fourier transform. The quantum Fourier transform is a part of many quantum algorithms, notably Shor's algorithm for factoring and computing the discrete logarithm, the quantum phase estimation algorithm for estimating the eigenvalues of a unitary operator, and algorithms for the hidden subgroup problem.

The quantum Fourier transform can be performed efficiently on a quantum computer, with a particular decomposition into a product of simpler unitary matrices. Using a simple decomposition, the discrete Fourier transform can be implemented as a quantum circuit consisting of only $O(n^2)$ Hadamard gates and controlled phase shift gates, where n is the number of qubits. This can be compared with the classical discrete Fourier transform, which takes $O(n2^n)$ gates (where n is the number of bits), which is exponentially more than $O(n^2)$.

The quantum Fourier transform is the classical discrete Fourier transform applied to the vector of amplitudes of a quantum state. The classical (unitary) Fourier transform acts on a vector $(x_0, ..., x_{N-1})$ and maps it to the vector $(y_0, ..., y_{N-1})$ according to the formula:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega^{jk}$$

where $\omega = e^{\frac{2\pi i}{N}}$ is a primitive N^{th} root of unity.

Similarly, the quantum Fourier transform acts on a quantum state $\sum_{i=0}^{N-1} x_i |i\rangle$ and maps it to a quantum state $\sum_{i=0}^{N-1} y_i |i\rangle$ according to the formula:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega^{jk}$$

This can also be expressed as the map

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} |k\rangle$$

Equivalently, the quantum Fourier transform on a n-qubit vector ($N = 2^n$) can be viewed as a unitary matrix acting on quantum state vectors, where the unitary matrix F_N is given by

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

2.4 Measurement in Quantum Mechanics

2.4.1 A Qualitative Overview

One of the most difficult and controversial problems in quantum mechanics is the so-called measurement problem. Opinions on the significance of this problem vary widely. At one extreme the attitude is that there is in fact no problem at all, while at the other extreme the view is that the measurement problem is one of the great unsolved puzzles of quantum mechanics. The issue is that **quantum mechanics only provides probabilities for the different possible outcomes in an experiment it provides no mechanism by which the actual, finally observed result, comes about**. Of course, probabilistic outcomes feature in many areas of classical physics as well, but in that case, probability enters the picture simply because there is insufficient information to make a definite prediction. In principle, that missing information is there to be found, it is just that accessing it may be a practical impossibility. In contrast, there is no missing information for a quantum system, what we see is all that we can get, even in principle.

In Dirac's words - *The intermediate character of the state formed by superposition thus expresses itself through the probability of a particular result for an observation being intermediate between the corresponding probabilities for the original states, not through the result itself being intermediate between the corresponding results for the original states.*

2.4.2 The Quantitative Overview

For an ideal measurement in quantum mechanics, also called a von Neumann measurement, the only possible measurement outcomes are equal to the eigenvalues (say k) of the operator representing the observable. Consider a system prepared in state $|\psi\rangle$. Since the eigenstates of the observable $\hat{O}$ form a complete basis called eigenbasis, the state vector $|\psi\rangle$ can be written in terms of the eigenstates as

$$|\psi\rangle = c_1|1\rangle + c_2|2\rangle + c_3|3\rangle + \cdots$$

where $c_1, c_2, \ldots$ are complex numbers in general. The eigenvalues $O_1, O_2, O_3, ...$ are all possible values of the measurement. The corresponding probabilities are given by

$$\Pr(O_n) = \frac{|\langle n|\psi\rangle|^2}{\langle\psi|\psi\rangle} = \frac{|c_n|^2}{\sum\limits_k |c_k|^2}$$

Usually $|\psi\rangle$ is assumed to be normalized, i.e. $\langle\psi|\psi\rangle = 1$. Therefore, the expression above is reduced to

$$\Pr(O_n) = |\langle n|\psi\rangle|^2 = |c_n|^2.$$

A quantum computer operates by setting the n qubits in a controlled initial state that represents the problem at hand and by manipulating those qubits with a fixed sequence of quantum logic gates. The sequence of gates to be applied is called a quantum algorithm. The calculation ends with measurement of all the states, collapsing each qubit into one of the two pure states, so the outcome can be at most n classical bits of information.

For example, if we prepare a 2-qubit system in the state $|psi\rangle = \frac{1}{\sqrt{(2)}}|00\rangle + \frac{1}{\sqrt{(3)}}|01\rangle + \frac{1}{\sqrt{(6)}}|11\rangle$, then a measurement on the system will yield results corresponding to the state $|00\rangle$ with probability $\frac{1}{2}$, state $|01\rangle$ with probability $\frac{1}{3}$ and state $|11\rangle$ with probability $\frac{1}{6}$.

Partial measurement We can even perform a measurement on just one register. Then, the probability of the state $|0\rangle$ being measured on the register is just a sum of the probabilities of all states wherein this particular register is in the $0\rangle$ state. So, in the above example, a measurement on the first register will yield $|0\rangle$ with probability $= \frac{1}{2} + \frac{1}{3} = \frac{5}{6}$

2.4.3 Collapsing of States

A postulate of quantum mechanics states that the process of measurement formally causes an instantaneous collapse of the quantum state to the eigenstate corresponding to the measured value of the observable. A consequence of this is that the results of a subsequent measurement essentially unrelated to the form of the pre-collapse quantum state (unless the eigenstates of the operators representing the observables coincide). So, in the example mentioned in the previous subsection, if a measurement on the system had yielded the result corresponding to eigenstate $|00\rangle$, then all subsequent measurements would have given the same result too because the system would have collapsed to this state.

The scenario is slightly different in the case of partial measurement. Here, the measured register collapses entirely into a particular state and then, the states that remain in the system must all have this register in the measured state. Also, as expected, the mutual ratio of the probability associated with these states stays conserved. So, in the example where we did a measurement on the first register only, the resultant state would be

$$\sqrt{\frac{\frac{1}{\sqrt{2}}^2}{\frac{1}{\sqrt{2}}^2+\frac{1}{\sqrt{3}}^2}}|00\rangle + \sqrt{\frac{\frac{1}{\sqrt{3}}^2}{\frac{1}{\sqrt{2}}^2+\frac{1}{\sqrt{3}}^2}}|01\rangle = \sqrt{\frac{3}{5}}|00\rangle + \sqrt{\frac{2}{5}}|01\rangle$$

Chapter 4

Some popular Quantum Computing Ideas

A quantum algorithm is a step-by-step procedure such that each of the steps can be performed on a classical computer. Quantum computers can execute algorithms that sometimes dramatically outperform classical computation. The best-known example of this is Shor's discovery of an efficient quantum algorithm for factoring integers, whereas the same problem appears to be intractable on classical computers. Understanding what other computational problems can be solved significantly faster using quantum algorithms is one of the major challenges in the theory of quantum computation. In an attempt to gain an insight in the same, we study a few of the existing quantum algorithms.

The first among them is the Deutsch-Jozsa algorithm used to determine the nature of a function, followed by Shor's algorithm for factoring integers and then Grover's algorithm which efficiently searches for an element in an unsorted database.

4.1 Deutsch-Jozsa Algorithm

4.1.1 Problem Statement

In the Deutsch-Jozsa problem, we are given a black box quantum computer known as an oracle that implements the function $f : \{0,1\}^n \to \{0,1\}$. In layman's terms, it takes n-digit binary values as input and produces either a 0 or a 1 as output for each such value. We are promised that the function is either constant (0 on all inputs or 1 on all inputs) or balanced (returns 1 for half of the input domain and 0 for the other half); the task then is to determine if f is constant or balanced by using the oracle.

4.1.2 Motivation and a Classical Approach

The DeutschJozsa problem[1] is specifically designed to be easy for a quantum algorithm and hard for any deterministic classical algorithm. The motivation is to

show a black box problem that can be solved efficiently by a quantum computer with no error, whereas a deterministic classical computer would need exponentially many queries to the black box to solve the problem.

For a conventional deterministic algorithm where n is number of bits/qubits, $2^{n-1}+1$ evaluations of f will be required in the worst case. To prove that f is constant, just over half the set of inputs must be evaluated and their outputs found to be identical.

4.1.3 The Deutsch Quantum Algorithm

1. The algorithm begins with the $n+1$ qubit state $|0\rangle^{\otimes n}|1\rangle$. That is, the first n qubits are each in the state $|0\rangle$ and the final one is $|1\rangle$.
2. Apply a Hadamard transformation to each bit to obtain the state $\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|0\rangle-|1\rangle)$.
3. We have the function f implemented as quantum oracle. The oracle maps the state $|x\rangle|y\rangle$ to $|x\rangle|y\oplus f(x)\rangle$, where $\oplus$ is addition modulo 2.
4. Applying the quantum oracle gives $\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|f(x)\rangle-|1\oplus f(x)\rangle)$.
5. For each x, $f(x)$ is either 0 or 1. A quick check of these two possibilities yields $\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$.
6. At this point, ignore the last qubit. Apply a Hadamard transformation to each qubit to obtain
   $\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\sum_{y=0}^{2^n-1}(-1)^{x\cdot y}|y\rangle=\frac{1}{2^n}\sum_{y=0}^{2^n-1}\left[\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot y}\right]|y\rangle$
   where $x\cdot y=x_0y_0\oplus x_1y_1\oplus\cdots\oplus x_{n-1}y_{n-1}$ is the sum of the bitwise product.
7. Finally we examine the probability of measuring $|0\rangle^{\otimes n}$, $\left|\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\right|^2$ which evaluates to 1 if $f(x)$ is constant (constructive interference) and 0 if $f(x)$ is balanced (destructive interference).

The DeutschJozsa algorithm provided inspiration for Shor's algorithm and Grover's algorithm, two of the most revolutionary quantum algorithms, which are described now.

4.2 Shor's Algorithm

Shor's algorithm, given in 1994 by mathematician Peter Shor, is an algorithm for integer factorization. On a quantum computer, Shor's algorithm runs in polynomial time. First, we describe the problem of factorization more formally followed

by an overview of some mathematical concepts required to understand the algorithm. The familiar reader can skip these subsections and continue reading from the subsection 'Reduction of the Factorization problem'.

4.2.1 The factorization problem

The factorization problem definition is given below.
Problem Definition: *Given an integer n, factorize n as a product of primes.*

Typically the integer n is very large (a few hundred digits long). Hence the brute force approach of checking whether each number between 2 and $n-1$ is a factor of n which takes exponential time to complete, is not efficient and it can take many years for the computation to finish. In fact, there is no deterministic algorithm known that can factorize n in polynomial-time. This limitation is exploited by the famous Rivest-Shamir-Adleman encryption scheme (RSA).

We will assume (both for simplicity and with a view to RSA cryptanalysis) that $n = pq$ where p and q are large unknown primes. We must determine p and q.

4.2.2 The integers mod n

Let $R = 0, 1, 2, \ldots, n-1$ with addition and multiplication modulo n. For $a, b \in R$ we compute $a + b$ mod n and ab mod n by first computing the sum or product as an ordinary integer, then taking the remainder upon division by n.

These operations are easily performed in polynomial time in the input size $l = log(n)$ using a classical logical circuit of size polynomial in l. For $x \in R$ and $a \geq 0$, the value of x^a mod n can also be determined in polynomial time and space via the square-and-multiply algorithm which is described in brief below.

4.2.3 A fast classical algorithm for modular exponentiation

The method is based on the following observation:

$$x^a = \begin{cases} x\,(x^2)^{\frac{a-1}{2}}, & \text{if } a \text{ is odd} \\ (x^2)^{\frac{a}{2}}, & \text{if } a \text{ is even.} \end{cases} \tag{4.1}$$

Now, due to the modular nature of squaring, the number of digits of x^2 are limited by the length of n. We computed x^a by repeated squaring taking the result modulo n each time before proceeding to the next iteration, which gives rise to the following recursive algorithm for exponentiation.

```
Function exp-by-squaring(x,n)
      if n<0 then return exp-by-squaring(1/x, -n);
      else if n=0 then return 1;
      else if n=1 then return x;
      else if n is even then return exp-by-squaring(x*x, n/2);
      else if n is odd then return x*exp-by-squaring(x*x, (n-1)/2).
```

4.2.4 Reduction of the Factorization problem

Using randomization, factorization can be reduced to finding the order of an element in the multiplicative group (mod n), where order or r is the smallest $r \geq 1$ such that x^r mod n is 1.

Suppose we choose x randomly from $\{2, \ldots, n-1\}$ and find the order r of x with respect to n. Then if r is not odd

$$(x^{\frac{r}{2}} - 1)(x^{\frac{r}{2}} + 1) \equiv 1 \text{ (mod } n)$$

Now consider the $gcd(x^{\frac{r}{2}} - 1, n)$. This fails to be a non-trivial divisor of n only if $x^{\frac{r}{2}} \equiv -1$ (mod n) or r is odd. This procedure, when applied to a random x (mod n), yields a factor of n with probability at least $1 - \frac{1}{2^{k-1}}$, where k is the number of distinct odd prime factors of n. We will accept this statement without proof.
It can be seen that the above probability is at least $\frac{1}{2}$ if $k \geq 1$. If $k = 1$ implying n had only one odd prime factor, it can be easily factored in polynomial time using classical algorithms. (Reference here)

Shor's algorithm finds the factors of n indirectly by first choosing a random x and then finding the order of x with respect to n. Then it finds $gcd(x^{\frac{r}{2}} - 1, n)$ which will be a factor of n with high probability. It continues doing the same until n has been completely factorized. The algorithm requires a quantum computer only for finding the period of x in polynomial time. This part of the algorithm is presented next.

4.2.5 The Algorithm

We present only the quantum part of the algorithm in this section. The complete algorithm is presented at the end of the section. The algorithm uses two quantum registers which hold integers represented in binary and some additional workspace.

1. Find q, such that $q = 2^l$ for some integer l and $n^2 \leq q < 2n^2$. In a quantum gate array we need not even keep the values of n, x and q in memory, as they can be built into the structure of the gate array.

2. Next, the first register is put in the uniform superposition of states representing numbers a (mod q). This leaves the registers in the following state.

$$\frac{1}{q^{\frac{1}{2}}} \sum_{a=0}^{q-1} |a\rangle|0\rangle. \tag{4.2}$$

3. Next x^a mod n is computed using the square-and-multiply algorithm. This can be done reversibly. This leaves our registers in the following state.

$$\frac{1}{q^{\frac{1}{2}}} \sum_{a=0}^{q-1} |a\rangle|x^a(\text{mod n})\rangle. \tag{4.3}$$

4. Then the Fourier transform is performed on the first register, as described in Chapter 2 which maps $|a\rangle$ to

$$\frac{1}{q^{\frac{1}{2}}} \sum_{c=0}^{q-1} exp(2\pi iac/q)|c\rangle. \tag{4.4}$$

This leaves the registers in the following state

$$\frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} exp(2\pi iac/q)|c\rangle|x^a(\text{mod n})\rangle. \tag{4.5}$$

5. Finally we observe the system. We now compute the probability that our machine ends in a particular state $|c, x^k \bmod n\rangle$, where $0 \leq k < r$. Summing up over all possible ways to reach this state, this probability is

$$\left| \frac{1}{q} \sum_{a:x^a \equiv x_k} \exp(2\pi iac/q) \right|^2$$

Since the order is r, this sum is over all a such that $a \equiv k \pmod r$. Therefore, tha above sum can be written as,

$$\left| \frac{1}{q} \sum_{b=0}^{\lfloor (q-k-1)/r \rfloor} \exp(2\pi i(br+k)c/q) \right|^2$$

Since, $\exp(2\pi ikc/q)$ factors out of the sum and has magnitude 1, we drop it. Now, on the remaining part of the expression Shor's algorithm performs an estimation analysis of the above probability expression and derives the following lemma which we present without proof.

Lemma 1. *The probability of seeing a given state $|c, x^k \pmod n\rangle$ is at least $\frac{1}{3r^2}$ if there is a d such that,*

$$\frac{-r}{2} \leq rc - dq \leq \frac{r}{2}. \tag{4.6}$$

Next, Shor proceeds to prove that the probability of obtaining r via the above algorithm is at least $\frac{\delta}{loglogr}$. We will accept the above statement without proof. Hence by repeating the experiment $O(loglogr)$ times, we are assured of a high probability of success.

4.2.6 An example factorization

We show the running of Shor over the factorization of $n = 55$. Since $n^2 \leq q < 2n^2$ and $q = 2^l$, $q = 2^{13} = 8192$. Suppose we choose $x = 13$. The running of the algorithm on this input is described below.

1. We initialize the initial state to be a superposition of states representing a (mod 8192).

$$|\psi\rangle = \frac{1}{\sqrt{8192}}(|0,0\rangle + |1,0\rangle + \ldots + |8191,0\rangle)$$

2. Next the modular exponentiation gate is applied.

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{8192}}(|0,1\rangle + |1,13\rangle + |2,13^2 \text{ mod}55\rangle \ldots + |8191,13^{8191} \text{ mod}55\rangle) \\ &= \frac{1}{\sqrt{8192}}(|0,1\rangle + |1,13\rangle + |2,4\rangle \ldots + |8191,2\rangle) \end{aligned}$$

3. Next we perform the Fourier transform on the first register.

$$|\psi\rangle = \frac{1}{8192}\sum_{a=0}^{8191}\sum_{c=0}^{8191} exp(2\pi iac/8192)|c\rangle|13^a(\text{mod } 55)\rangle.$$

4. Now we observe the registers. Register 2 can be in any of the states with equal probability. Hence all power of x mod 55 are almost equally likely to be observed if $r << q$. Suppose we observe 28 as a power of x mod 55. This occurs 410 times in the series as a varies from 0 to 8191. Then the probability of observing register 1 to be in state c is

$$Pr(c) = \frac{1}{8192}\frac{1}{410}\left\|\sum_{d=0}^{409} exp(2\pi irdc/8192)\right\|$$

   Here $r = 20$. Among the states which can be observed with reasonably high probability is $|4915\rangle$ which is observed with probability of 4.4%.

5. Now $\frac{c}{q} = \frac{4915}{8192}$. Shor's algorithm uses the method of continued fractions to find d/r from c/q. Applying it here would give us r to be a multiple of $r_1 = 5$ and that on trying $r_1, 2r_1, \ldots \lfloor log(n)^{1+\epsilon} \rfloor r_1$ as values for r we are guaranteed to find r with a very high probability. Here, we find that $r = 20$.

6. Now, the algorithm uses the Euclidean algorithm to find the factors of 55.

$$m = 13^{(20/2)} \text{ mod } 55 = 13^{10} \text{ mod } 55 = 34$$

   and the factors of $n = 55$ are,

$$\begin{aligned} gcd(m+1, 55) &= gcd(35, 55) = 5 \\ gcd(m-1, 55) &= gcd(33, 55) = 11 \end{aligned}$$

4.3 Grover's Algorithm

The Grover algorithm, given by Lov Grover in 1996, solves the problem of searching for an element in an unsorted database with N entries in $O(N^{\frac{1}{2}})$ time. Note that with classical computation models this problem cannot be solved in less than linear time ($O(N)$).

4.3.1 The search problem

Assume $N = 2^n$. Suppose that we have a function $f(x)$ from $\{0,1\}^n$ to $0,1$ which is zero on all inputs except for a single (marked) item $x_0 : f(x) = \delta_{x,x_0}$. By querying this function we wish to find the marked item x_0. If we have no information about the particular x_0, then finding this marked item is very difficult. In the worst case it will take $2^n - 1$ queries to find x_0 for a deterministic algorithm. In general, if the search problem has M solutions, then the classical algorithm might take as many as $2^n - M$ steps.

For large N, the Grover algorithm could yield very large performance increases. The key idea is that although finding a solution to the search problem is hard, recognising a solution is easy. We wish to search through a list of N elements, lets index the elements by $x \in 0, N-1$ and call them y_x.

4.3.2 The Oracle

Rather than dealing with the list itself, we focus on the index of the list, x. Given some value of x, we can tell whether y_x solves the search problem. We assume that we can construct some device to tell us if y_x solves the search problem. This device is called an Oracle.

- The Oracle takes as input an index value in a qubit register $|x\rangle$. It also takes a single *Oracle qubit*, $|q\rangle$. The state given to the Oracle is thus $|\psi\rangle = |x\rangle|q\rangle$.
- The Oracle is represented by a unitary operator, O. If x indexes a solution to the search problem, O sets $f(x) = 1$, and $f(x) = 0$ if it doesnt index a solution.
- If $f(x) = 1$, the Oracle flips the state of $|q\rangle$. We write this as $O|x\rangle|q\rangle = |x\rangle X^{f(x)}|q\rangle$. X is just our quantum NOT operator.
- So if $f(x) = 1$, $|q\rangle \mapsto X|q\rangle$, else $|q\rangle \mapsto |q\rangle$.
- We choose to initially program $|q\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then $X|q\rangle = \frac{1}{\sqrt{2}}(-|0\rangle + |1\rangle) = -|q\rangle$. And $O|x\rangle|q\rangle = (-1)^{f(x)}|x\rangle|q\rangle$.
- The Oracle therefore takes $|x\rangle \mapsto (-1)^{f(x)}|x\rangle$. So the term indexing the solution is marked with a $-$ sign.

The Oracle does not find the solution to the problem, it simply recognises the answer when presented with one. The key to quantum search is that we can look at all solutions simultaneously: the Oracle just manipulates the state coefficients using a unitary operator!.

4.3.3 The Grover Iteration

1. Begin with $|x\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$.
2. Apply the Oracle to $|x\rangle$:

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} (-1)^{f(x)} |j\rangle$$

3. Apply the QFT to $|x\rangle$.
4. Reverse the sign of all terms in $|x\rangle$ except for the term $|0\rangle$.
5. Apply the Inverse QFT.
6. Return to step 2 and repeat.

4.3.4 Performance of the algorithm

The point at which we terminate Grovers algorithm and measure the result is critical. This is because the probability associated with the correct state rises to 1 after a certain number of iterations and then oscillates periodically between the two extremes, 0 and 1. It has been shown that the optimum number of iterations is $\sim \frac{\pi}{4}\sqrt{\frac{N}{M}}$, where M is the number of solutions. It has also been shown that this is the best that any quantum search algorithm can do.

4.3.5 An example

Apply Grovers algorithm to $N = 4$ with solution $x = 2$.

- We start with $|x\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)$.
- Apply the Oracle: $|x\rangle \mapsto \frac{1}{2}(|0\rangle + |1\rangle - |2\rangle + |3\rangle)$.
- Apply the QFT: $F|x\rangle = \frac{1}{2}(|0\rangle + |1\rangle - |2\rangle + |3\rangle)$.
- Flips signs of all terms except $|0\rangle$: $F|x\rangle \mapsto \frac{1}{2}(|0\rangle - |1\rangle + |2\rangle - |3\rangle)$.
- Inverse QFT: $|x\rangle = |2\rangle$.
- So when we measure $|x\rangle$, we are guaranteed the right answer.

4.4 The Quantum Minimum Algorithm

We now present a quantum algorithm for finding the minimum value among a given set of numbers. This algorithm is faster than the fastest possible classical algorithm and, as usual, is probabilistic. This algorithm uses the Grover search algorithm repeatedly to find the minimum with a high probability. First, we formally present the problem with notation and then we present the algorithm.

4.4.1 The Problem

Let $T[0..N-1]$ be an unsorted table of N items, each holding a value from an ordered set. The minimum searching problem is to find the index y such that $T[y]$ is minimum. This clearly requires a linear number of probes on a classical probabilistic Turing machine. [16] gave a simple quantum algorithm which solves the problem using $O(N^{1/2})$ probes. The algorithm makes repeated calls to Grover's search algorithm to find the index of a smaller item than the value determined by a particular threshold index. If there are $t \geq 1$ marked entries, Grover's algorithm will return one of them with equal probability after an expected number of $O(\sqrt{N/t})$ iterations. If no entry is marked, it will run forever.

4.4.2 The Algorithm

The algorithm is as follows:

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 2c.
   (a) Initialize the register as a uniform superposition over the N states, i.e., give each state a coefficient of $\frac{1}{\sqrt{N}}$. Mark every item j for which $T[j] < T[y]$. This would be an $O(N)$ operation on a classical computer but here, the entire state which is a superposition of the N basis states, is acted upon at once by a quantum operator.
   (b) Apply the quantum exponential searching algorithm of [15].
   (c) Observe the register: let y' be the outcome. If $T[y'] < T[y]$, then set threshold index y to y'.
3. Return y.

4.4.3 Running Time and Precision

By convention, we assume that stage 2a takes $log(N)$ time steps and that one iteration in Grover's algorithm takes one time step. The expected number of iterations used by Grover to find the index of a marked item among N items where t items

are marked is at most $\frac{9}{2}\sqrt{N/t}$. The expected total time before the y holds the index of the minimum is at most $m_0 = \frac{45}{4}\sqrt{N} + \frac{7}{10}log^2 N$.

The algorithm given above finds the minimum with probability at least $\frac{1}{2}$. This probability can be improved to $1 - \frac{1}{2^c}$ by running the algorithm c times.

4.5 Quantum Walks

A generalization of Grover's search technique, quantum walks[17] have lead to a number of quantum algorithms for problems such as element distinctness (which will be described later). In this section, we present the basics of quantum walks and also an application of quantum walks due to Ambainis , namely, element distinctness. First, we describe random walks which are the classical analogue of quantum walks. Random walks provided the inspiration for quantum walks.

4.5.1 Random Walks

A random walk is a mathematical formulation of a path that consists of a succession of random steps. For example, the path traced by a gas molecule, the path traced by an animal foraging for food are all random walks. Often, random walks are assumed to be Markov chains or Markov processes in discrete time although there can be other types of random walks too. A classical Markov chain is said to be a random walk on an underlying graph if the nodes of the graph are the states in S, and a state s has a non-zero probability to go to state t if and only if the egde (s,t) exists in the graph. A simple random walk on a graph $G(V,E)$ is described by repeated applications of a stochastic matrix P, where $P(u,v) = \frac{1}{(d_u)}$ if (u,v) is an edge in G and d_u is the degree of the vertex u. If G is connected and non-bipartite, the distribution of the random walk $D_t = P^t D_0$ converges to a stationary distribution π which is independent of the initial distribution D_0.

An example: A one-dimensional random walk

The elementary one-dimensional random walk is a walk on the integer line $\mathbb{Z}$ which starts at 0 and at each time step moves +1 or -1 with equal probability.

4.5.2 Terminology used with Random Walks

There are many definitions which capture the rate of convergence to the limiting distribution in a random walk. Some important terms are defined here.

Definition 3. ***Mixing Time:***

$$M_\epsilon = min\{T | \forall t \geq T, D_0 : ||D_t - \pi|| \leq \epsilon\} \tag{4.7}$$

where the distance between two distributions d_1 and d_2 is given by $||d_1 - d_2|| = \sum_i |d_1(i) - d_2(i)|$.

Definition 4. ***Filling Time:***

$$\tau_\epsilon = min\{T | \forall t \geq T, D_0, X \subseteq V : D_t(X) \geq (1-\epsilon)\pi(X)\} \tag{4.8}$$

Definition 5. ***Dispersion Time:***

$$\xi_\epsilon = min\{T | \forall t \geq T, D_0, X \subseteq V : D_t(X) \leq (1+\epsilon)\pi(X)\} \tag{4.9}$$

4.5.3 Quantum Analogue: Quantum Markov Chains or Quantum Walks

Let $G(V, E)$ be a graph, and let $\mathbb{H}_V$ be the Hilbert space spanned by the states $|v\rangle$ where $v \in V$. We denote by n, or $|V|$ the number of vertices in G. Assume that G is d-regular. Let $\mathbb{H}_A$ be the auxiliary Hilbert space of dimension d spanned by the states $|1\rangle$ through $|d\rangle$. Let C be a unitary transformation on $\mathbb{H}_A$. Label each directed edge with a number between 1 and d, such that the edges labelled a form a permutation. Now we can define a shift operator S on $\mathbb{H}_A \otimes \mathbb{H}_V$ such that $S|a, v\rangle = |a, u\rangle$ where u is the ath neighbour of v. Hence, one step of the quantum walk is given by the operator $U = S.(C \otimes I)$. This is called a coined quantum walk.

Example: Consider the cycle graph with n nodes. This graph is 2-regular. The Hilbert space for the walk would be $C^2 \otimes C^n$. We choose C to be the Hadamard transform

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

and the shift S is defined as

$$S|R, i\rangle = |R, i + 1 \text{mod n}\rangle$$
$$S|L, i\rangle = |L, i - 1 \text{mod n}\rangle$$

where R denotes a move to the node on the right of the node indexed i and L denotes a move to the left. The quantum walk is then defined to be the repeated application of the Hadamard transform on the first register followed by an application of the shift operator S.

Having this general idea of quantum walks in mind, we now proceed to examine their applications to algorithmic problems. We first make a remark that the Grover's search algorithm is a special case of a quantum walk. Next we describe the application of quantum walks to the element distinctness problem.

4.5.4 Application to Element-Distinctness Problem

The element distinctness problem is as follows. Given numbers $x_1, x_2 \cdots x_N \in [M]$, are there $i, j \in [N]$, such that $i \neq j$ and $x_i = x_j$? Any classical solution to this problem will need $\Omega(N)$ queries. Ambainis gave a quantum walk algorithm for this problem that gives the answer in $O(N^{2/3})$ queries. The main idea is as follows. We have vertices v_S corresponding to sets $S \subseteq \{1, 2, \cdots, N\}$. Two vertices

v_S and v_T are connected by an edge is S and T differ in one variable. A vertex is marked if S contains i, j such that $x_i = x_j$. At each moment of time, we know x_i for all $i \in S$. This enables us to check if the vertex is marked with no queries. Also, it enables us to move to an adjacent vertex v_T by querying just one variable x_i for $i \notin S$ and $i \in T$.
Then we define a quantum walk on subsets of the type S. Ambainis shows that if $x_1, x_2 \cdots x_N$ are not distinct, this walk finds a set S containing i, j such that $x_i = x_j$ within $O(N^{2/3})$ steps.

With that, we conclude the current chapter on quantum computing literature. Next, we move on to see the applications of quantum computing to more complex algorithmic applications involving intelligence tasks.