BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF MATHEMATICS
ÚSTAV MATEMATIKY

# GROVER'S ALGORITHM IN QUANTUM COMPUTING AND ITS APPLICATIONS

GROVERŮV ALGORITMUS V KVANTOVÉM POČÍTÁNÍ A JEHO APLIKACE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

BSc Joseph Katabira

SUPERVISOR
VEDOUCÍ PRÁCE

doc. Mgr. Jaroslav Hrdina, Ph.D.

BRNO 2021

BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Mechanical Engineering

MASTER'S THESIS

Brno, 2021

BSc Joseph Katabira

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF MECHANICAL ENGINEERING

# Assignment Master's Thesis

| | |
|---|---|
| Institut: | Institute of Mathematics |
| Student: | **BSc Joseph Katabira** |
| Degree programm: | Applied Sciences in Engineering |
| Branch: | Mathematical Engineering |
| Supervisor: | **doc. Mgr. Jaroslav Hrdina, Ph.D.** |
| Academic year: | 2020/21 |

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

## Grover's algorithm in Quantum computing and its applications

**Brief Description:**

Quantum computations in some algorithms use entagulation of quantum states. From a mathematical point of view, these are indecomposable tensors of the respective order.  Entagulation's states are then used as a tool for quantum algorithms as a searchinq or cryptography.

**Master's Thesis goals:**

Learning the basics of quantum computing. The student chooses one of the classical quantum algorithms, programs it in the chosen simulation software, for example python  and will discuss its complexity with respect to classical non–quantum ones.

**Recommended bibliography:**

DE LIMA MARQUEZINO, Franklin, et al.: A Primer on Quantum Computing, SpringerBriefs in Computer Science, 2019.

RUE, Juanjo, XAMBO, Sebastian. Mathematical essentials of quantum computing, Lecture notes UPC, https://web.mat.upc.edu/sebastia.xambo/QC/qc.pdf

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2020/21

In Brno,

L. S.

| | |
|---|---|
| prof. RNDr. Josef Šlapal, CSc.<br>Director of the Institute | doc. Ing. Jaroslav Katolický, Ph.D.<br>FME dean |

**Abstrakt**
Kvantová výpočetní technika je rychle rostoucí obor informatiky, který přenáší principy kvantových jevu do našeho každodenního života. Díky své kvantové podstatě získávají kvantové počítače převahu nad klasickými počítači. V této práci jsme se zaměřili na vysvětlení základů kvantového počítání a jeho implementaci na kvantovém počítači. Zejména se zaměřujeme na popis fungování, konstrukci a implementaci Groverova algoritmu jako jednoho ze základních kvantových algoritmů. Demonstrovali jsme sílu tohoto kvantového algoritmu při prohledávání databáze a porovnávali ho s klasickými nekvantovými algoritmy pomocí implementace prostřednictvím simulačního prostředí QISKit. Pro simulaci jsme použili QASM Simulator a State vector Simulator Aer backends a ukázali, že získané výsledky korelují s dříve diskutovanými teoretickými poznatky. Toto ukazuje, že Groverův algoritmus umožňuje kvadratické zrychlení oproti klasickému nekvantovému vyhledávacímu algoritmu, Použitelnost algoritmu stejně jako ostatních kvantových algoritmů je ale stále omezena několika faktory, mezi které patří vysoké úrovně dekoherence a chyby hradla.

**Summary**
Quantum computing as a new field of computing is a quickly growing field which encapsulates the role of quantum phenomenon in our day to day lives. Because of the quantum characteristics, quantum computers have proved quantum supremacy over the classical computers. In this thesis we focused on discussing basics of quantum computing and in particular we focused on discussing the functioning, construction and implementation of Grover algorithm as a special case of quantum algorithms. We showcased its power as a database search algorithm over the classical non quantum ones through our algorithmic construction implemented through QISKit simulation environment. To simulate our construction, we made use of QASM Simulator and the State vector Simulator Aer backends and the results obtained correlated with the earlier discussed theoretical findings highly proving that Grover's algorithm provides quadratic speed up over the classical non quantum search algorithm which is a much better improvement but as at hand, the applicability of the algorithm as many others is still limited by several factors amongst which includes high decoherence levels and gate errors.

**Klíčová slova**
Qubit, Superpozice, Grover's Algorithm, Complexity, Oracle, Diffusion Operator, Quantum circuit, Quantum Gate, Grover Iterate, Initialisation, Measurement, Algorithm, Search space, Bra-Ket Notation.

**Keywords**
Qubit, Superposition, Grover's Algorithm , Complexity, Oracle , Diffusion Operator, Quantum circuit, Quantum Gate, Grover Iterate, Initialisation, Measurement, Algorithm, Search space,Bra- Ket Notation.

KATABIRA, J.*Groverův algoritmus v kvantovém počítání a jeho aplikace.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2021. 59 s. Vedoucí doc. Mgr. Jaroslav Hrdina, Ph.D.

Declaration I declare that I have written my Master's thesis on the theme "Grover's algorithm in Quantum computing and its applications" independently, under the guidance of my Master's thesis supervisor and using the technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis. As the author of the Master's thesis I furthermore declare that, as regards the creation of this Master's thesis, I have not infringed any copyright.

BSc Joseph Katabira

I would like to thank my supervisor doc. Mgr. Jaroslav Hrdina, Ph.D. for his invaluable support help and advice with the thesis work. I would also like to thank my family, graduate colleagues and friends for giving me physical, mental , emotional and moral support that has helped me through out the process. Lastly, I would like to thank the INTERMATHS consortium for the studentship that allowed me to do my master's studies.

BSc Joseph Katabira

# Contents

# 1. Basics of Quantum Computing

## 1.1. Introduction

In this section we provide an introduction to the subject of quantum computing and gives an explanation of the related concepts necessary for understanding the work presented in the thesis report. The goals of this thesis are;

1. To give a simplified explanation of the basics of quantum computing.
2. To discuss how the Grover's quantum algorithm functions.
3. To discuss the complexity of Grover's algorithm with respect to classical non-quantum ones with aid of a simulation example.

Quantum computing is based on qubits as the fundamental building block and the use of quantum phenomena such as superposition, interference and entanglement to perform quantum computations using quantum gates. By definition, a qubit is a two-state (or two-level) quantum-mechanical system with the two basis states being 0 and 1 usually denoted as $|0\rangle$ and $|1\rangle$ in bra-ket notation. A qubit can be in state $|0\rangle$, $|1\rangle$ or in a linear combination of both states(superposition of both states). As a simplest quantum mechanical system, a qubit can represent;

1. The spin of the electron in which the two levels can be taken as spin up and spin down.
2. The polarization of a single photon in which the two states can be taken to be the vertical polarization and the horizontal polarization.

Furthermore, qubits can be subdivided into single and multiqubit states where single qubit states are described with a single qubit and they constitute the two orthogonal basis states often referred to as the computational basis states. In 2-Dimensions, the computational basis states can be:

1. $|0\rangle$ and $|1\rangle$ or,
2. $|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$ sometimes called the computation (plus–minus) basis.

In some literature, the computation basis $[|+\rangle, |-\rangle]$ is defined as;

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

Multiqubit states are described by a string of single qubit states joined together under the tensor product ($\otimes$) operation and maybe classified into pure quantum state or mixed quantum states where **a pure quantum state** is a state which can be described by a single ket vector where as **a mixed quantum state** is a statistical ensemble of pure states. A mixed state is described by its associated density matrix (or density operator), usually denoted $\rho$. As a special case of pure multiqubit states,we shall discuss about the two qubit states which constitute of two qubits joined together under the tensor product $\otimes$ operation to give four possible orthogonal states(computational basis states in two qubits) that is $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$.

**Superposition**

This is essentially the ability of a quantum system to be in multiple states at the same time. By the superposition principle, the quantum state of the a single qubit can be expressed as a linear combination of the two classical states that is;

$$|\Psi\rangle = a_0 |0\rangle + a_1 |1\rangle ,$$

where

$$a_j \in \mathbb{C}, \ \sum_j |a_j|^2 = 1,$$

and similarly for the two qubits we can express the quantum state as a linear combination of the four classical states that is;

$$|\Psi\rangle = a_{00} |00\rangle + a_{01} |01\rangle + a_{10} |10\rangle + a_{11} |11\rangle ,$$

where

$$a_{ij} \in \mathbb{C}, \ \sum_{ij} |a_{ij}|^2 = 1,$$

etc.

## Entanglement

This is a physical phenomenon that occurs when a pair or group of particles are generated, interact, or share spatial proximity in a way such that the quantum state of each particle of the pair or group cannot be described independently of the state of the others including when the particles are separated by a large distance [38]. In quantum mechanics, a strong relationship exists between quantum particles, such that observing one of two entangled quantum states causes it to behave randomly but tells the observer exactly how the other quantum state would act if observed in a similar manner even if separated by great distances.

The **Bell state** is a direct result of entanglement and it is defined as a maximally entangled quantum state of two qubits that exhibit perfect correlations even at spatial separation which cannot be explained without quantum mechanics. There are four maximally entangled states(Bell states) on two qubits namely;

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \ \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \ \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \quad \text{and} \ \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle).$$

The ideas discussed above can be generalised to N- qubits. We shall conclude this section with a brief history on quantum computing.

In 1979, Paul Benioff at Argonne National Labs, submitted a paper entitled "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines" [4]. In this paper, Benioff demonstrated the theoretical basis for quantum computing and then suggested that such a computer could be built.

In 1980 a mathematician Yuri I. Manin in his book "Computable and Uncomputable" [30] laid out the core idea of quantum computing which was further strengthened in 1981 by Feynman when he gave a lecture entitled "Simulating Physics with Computers" [20].

In this talk, he argued that a classical system could not adequately represent a quantum mechanical system.

David Deutsch, a physicist at Oxford, suggested a more comprehensive framework for quantum computing in his 1985 paper [16] . In this work, he describes in detail what a quantum algorithm would look like and anticipated that one day it would become technologically possible to build quantum computers. He went on to develop an example of an algorithm that would run faster on a quantum computer. He then further generalized this algorithm in collaboration with Richard Jozsa [15].

Umesh Vazirani and his student Ethan Bernstein picked up where Deutsch and Jozsa left off and in 1993, Bernstein and Vazirani published a paper which described an algorithm that showed clear quantum-classical separation even when small errors are allowed. Further in their 1993 paper [6], they described a quantum version of the Fourier transform (QFT) which would serve as a critical component for Peter Shor when he developed his algorithm to factor large numbers.

In 1994, Daniel Simon then a postdoc at the University of Montreal outlined a problem that a quantum computer would clearly solve exponentially faster than a classical one in deriving his algorithm called Simon's algorithm [34]. Though prior to Daniel Simon's work on algorithms, Seth Lloyd, working at Los Alamos, published a paper in Science which described a method of building a working quantum computer [29] . He proposed that a system sending pulses into a unit can represent a quantum state.

In 2001, Isaac Chuang et al. implemented Shor's algorithm on a nuclear magnetic resonance (NMR) system to factor the number 15 as a demonstration [37]. Lov Grover also contributed to the quantum algorithm arsenal by demonstrating that one can achieve some speedup in a search algorithm on a quantum computer. [24]

In 1995, Cirac and Zoller proposed an ion trap as the physical system to perform quantum computation. In this setup, lasers are used to ionize atoms which are then trapped in electric potentials which contributed to the advancement of quantum computers [13].

## 1.2. Mathematical preliminaries

The basic playground for quantum computation is the complex vector space. The $n$–dimensional complex vector space denoted as $\mathbb{C}^n$ consists of vectors

$$a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix},$$

defined by elements from complex field $a_i \in \mathbb{C}$.

### Complex field $\mathbb{C}$

The complex field $\mathbb{C}$ are elements of the form;

$$\psi = a_1 + ia_2,$$

where $a_1, a_2 \in \mathbb{R}$ having complex conjugates of the form $\bar{\psi} = a_1 - ia_2$. For two complex numbers $\psi_1 = a_1 + ia_2$ and $\psi_2 = b_1 + ib_2$ we can define two operations namely;

1. **Addition**

$$\psi_1 + \psi_2 = (a_1 + a_2 i) + (b_1 + b_2 i) = (a_1 + b_1) + (a_2 + b_2)i, \tag{1.1}$$

2. **Multiplication**

$$\psi_1 \cdot \psi_2 = (a_1 + a_2 i) \cdot (b_1 + b_2 i) = (a_1 b_1 - a_2 b_2) + (a_1 b_2 + a_2 b_1)i. \tag{1.2}$$

If $\psi_1 \neq 0$ we can to write

$$\psi_1{}^{-1} = \frac{1}{a_1^2 + a_2^2}(a_1 - a_2 i). \tag{1.3}$$

This defines the inverse element in the complex filed. For any complex number $\psi = a_1 + ia_2$ with complex cojugate $\bar{\psi}$, we can define the norm $|\ |$ as;

$$\psi \cdot \bar{\psi} = a_1^2 + a_2^2 = |\psi|^2. \tag{1.4}$$

and an angle $\theta$ as;

$$\theta = \arctan\left(\frac{a_2}{a_1}\right). \tag{1.5}$$

Generally a complex number $\psi$ can be written in exponential form as

$$\psi = |\psi| e^{i\theta} = |\psi|(\cos\theta + i \sin\theta), \tag{1.6}$$

where $0 \leq \theta \leq 2\pi$.

## Axiomatic definition

**Definition 1.2.1.** A *Vector space* over a field $\mathbb{F}$ is a set $V$ together with two operations:
$+ \colon V \times V \longrightarrow V$, takes any two vectors $a_1, a_2 \in V$ and assigns them to $a_1 + a_2. \in V$,
$\cdot \colon \mathbb{F} \times V \longrightarrow V$ takes any scalar $b_1 \in \mathbb{F}$ and any $a_1 \in V$ and gives $b_1 a_1 \in V$,
such that for any three elements $a_1, a_2$ and $a_3 \in V$, and $b_1, b_2 \in \mathbb{F}$ the following axioms hold;

1. $a_1 + (a_2 + a_3) = (a_1 + a_2) + a_3$.
2. $a_1 + a_2 = a_2 + a_1$.
3. There exists an element $0 \in V$ such that $a_1 + 0 = a_1$, $\forall a_1 \in V$ .
4. $\forall a_1 \in V$ there exists an element $-a_1 \in V$ such that $a_1 + (-a_1) = 0$.
5. $b_1(b_2 a_1) = (b_1 b_2) a_1$.
6. $1a_1 = a_2$, where $1 \in V$.
7. $b_1(a_1 + a_2) = b_1 a_1 + b_1 a_2$.
8. $(b_1 + b_2) a_1 = b_1 a_1 + b_2 a_1$.

**Definition 1.2.2.** A *norm* is real-valued function $\| \cdot \| \colon V \longrightarrow \mathbb{R}$ with the following properties:

1. It is non–negative, that is for every vector $a_1$, one has $\|a_1\| \geq 0$.
2. It is positive on nonzero vectors, that is, $\|a_1\| = 0 \Longleftrightarrow a_1 = 0$.
3. For every vector $a_1$, and every scalar $b$ , one has $\|ba_1\| = |b|\|a_1\|$ .
4. The triangle inequality holds; that is, for every vectors $a_1$ and $a_2$, one has
$$\|a_1 + a_2\| \leq \|a_1\| + \|a_2\|.$$

**Remark.** *A norm induces a distance by the formula* $d(a_1, a_2) = \|a_2 - a_1\|$ *which makes any normed vector space into a* metric space and a *topological vector space.*

**Definition 1.2.3.** An *inner product space is a vector space* $V$ *over the field* $\mathbb{F}$ *together with a map* $\langle \cdot, \cdot \rangle \colon V \times V \to \mathbb{F}$ *satisfying for all vectors* $a_1, a_2, a_3 \in V$ *and all scalars* $b \in \mathbb{F}$ *the following conditions;*

1. *Linearity in the first argument:*
$$\langle ba_1, a_2 \rangle = b\langle a_1, a_2 \rangle.$$
$$\langle a_1 + a_2, a_3 \rangle = \langle a_1, a_3 \rangle + \langle a_2, a_3 \rangle.$$
2. *Conjugate symmetry or Hermitian symmetry:*
$$\langle a_1, a_2 \rangle = \overline{\langle a_2, a_1 \rangle}.$$
3. *Positive definiteness/semi–definiteness:* $\langle a_1, a_1 \rangle \geq 0$ *if* $a_1 \neq 0$ *and* $\langle a_1, a_1 \rangle = 0$ *if and only if* $a_1 = 0$.

The Complex vector space $\mathbb{C}^n$ defined by (1.1) and (2) in Section 1.2 satisfies the Definition 1.2.1 and together with a norm defined by (1.4) satisfies the Definition 1.2.3.

**Remark.** *If condition 1 holds and if* $\langle \cdot, \cdot \rangle$ *is also anti–linear (also called, conjugate linear) in its second argument then* $\langle \cdot, \cdot \rangle$ *is called a sesquilinear form. Conjugate symmetry and linearity in the first variable implies;*
$$\langle a_1, ba_2 \rangle = \overline{\langle ba_2, a_1 \rangle} = \bar{b}\overline{\langle a_2, a_1 \rangle} = \bar{b}\langle a_1, a_2 \rangle.$$
$$\langle a_1, a_2 + a_3 \rangle = \overline{\langle a_2 + a_3, a_1 \rangle} = \overline{\langle a_2, a_1 \rangle} + \overline{\langle a_3, a_1 \rangle} = \langle a_1, a_2 \rangle + \langle a_1, a_3 \rangle.$$
*In the case of* $\mathbb{F} = \mathbb{R}$*, conjugate-symmetry reduces to symmetry, and sesquilinearity reduces to bi–linearity. Hence we get*
$$\langle a_1, a_2 \rangle = \langle a_2, a_1 \rangle.$$
*Inner product spaces are normed vector spaces with the norm defined as;*
$$\|a_1\| = \sqrt{\langle a_1, a_1 \rangle}.$$
*An inner product space is a metric space, with the distance defined by;*
$$d(a_1, a_2) = \|a_2 - a_1\|.$$

**Remark.** *The axioms of the inner product guarantee that the map above forms a norm, which will have the following properties;*

1. Polarization identity*: The inner product can be retrieved from the norm by the polarization identity.*

$$\|a_1 + a_2\|^2 = \|a_1\|^2 + \|a_2\|^2 + 2\,\mathrm{Re}\langle a_1, a_2\rangle,$$

*which is a form of the law of cosines.*

2. Orthogonality*: Two vectors are orthogonal if their inner product is zero. In the case of Euclidean vector spaces, the inner product allows us to define the (non oriented) angle $\theta$ of two nonzero vectors by;*

$$\theta = \arccos\frac{\langle a_1, a_2\rangle}{\|a_1\|\,\|a_2\|},$$

*and $0 \leq \theta \leq \pi$.*

3. Pythagorean theorem*: Whenever $a_1, a_2 \in V$ and $\langle a_1, a_2\rangle = 0$, then*

$$\|a_1\|^2 + \|a_2\|^2 = \|a_1 + a_2\|^2.$$

4. Parseval's identity*: An induction on the Pythagorean theorem yields: if $a_1, \cdots, a_n$ are orthogonal vectors;*

$$\sum_{i=1}^{n} \|a_i\|^2 = \left\|\sum_{i=1}^{n} a_i\right\|^2.$$

5. Parallelogram law*: For $a_1, a_2 \in V$,*

$$\|a_1 + a_2\|^2 + \|a_1 - a_2\|^2 = 2\|a_1\|^2 + 2\|a_2\|^2.$$

*The parallelogram law is a necessary and sufficient condition for the existence of an inner product corresponding to a given norm.*

6. Ptolemy's inequality*: For $a_1, a_2, a_3 \in V$,*

$$\|a_1 - a_2\|\|a_3\| + \|a_2 - a_3\|\|a_1\| \geq \|a_1 - a_3\|\|a_2\|.$$

*Ptolemy's inequality is also a necessary and sufficient condition for the existence of an inner product corresponding to a given norm.*

Suppose that $\langle\cdot,\cdot\rangle$ is an inner product on $V$ (so it is anti–linear in its second argument). The polarization identity shows that the real part of the inner product is;

$$\mathrm{Re}\langle a_1, a_2\rangle = \frac{1}{4}\left(\|a_1 + a_2\|^2 - \|a_1 - a_2\|^2\right).$$

If $V$ is a real vector space then

$$\langle a_1, a_2\rangle = \mathrm{Re}\langle a_1, a_2\rangle = \frac{1}{4}\left(\|a_1 + a_2\|^2 - \|a_1 - a_2\|^2\right)$$

and the imaginary part (also called the complex part) of $\langle\cdot,\cdot\rangle$ is always 0. Assuming that $V$ is a complex vector space. The polarization identity for complex vector spaces shows that,

$$\begin{aligned} 4\langle a_1, a_2\rangle &= \frac{1}{4}\left(\|a_1+a_2\|^2 - \|a_1-a_2\|^2 + i\|a_1+ia_2\|^2 - i\|a_1-ia_2\|^2\right) \\ &= \operatorname{Re}\langle a_1, a_2\rangle + i\operatorname{Re}\langle a_1, ia_2\rangle. \end{aligned}$$

We shall work in the N–dimensional Hermitian-complex space $\mathbb{C}^{2^n}$ which comes endowed with a norm $||.||$ defined as;

$$\sqrt{\psi\cdot\bar{\psi}^\dagger} = ||\psi||, \tag{1.7}$$

where $\psi$ is a complex vector and $\psi^\dagger$ is an complex conjugation and transposition. This norm is induced by the inner product which is defined by

$$\langle\psi|\psi\rangle = \psi\cdot\bar{\psi}^\dagger.$$

The qubits are going to be represented by q-vectors (quantum vectors) of order dimension $2^n$ which can be expressed as;

$$\psi = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} a_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ a_1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_{2^n-1}. \end{bmatrix}$$

in simplified notation, this can be written as;

$$\psi = \sum_{i=0}^{2^n-1} a_i\psi_i$$

where $\psi_i$ is the i-th-vector with the i-th nonzero entry.

### Dirac notation

In Dirac notation as often used in quantum mechanics, a vector $\psi$ defined over $\mathbb{C}^{2^n}$ often called a q-vector $\psi$ and its denoted as $|\Psi\rangle$ and it is called the ket vector which will be expressed as;

$$|\Psi\rangle = \sum_{i=0}^{2^n-1} a_i\,|\Psi_i\rangle\,, a_i \in \mathbb{R}, \tag{1.8}$$

where the q-vector $\psi_i$ denoted as $|\Psi_i\rangle$ is one with the i-th component equal to 1 and others zero. Similarly a bra vector which is the transpose of the ket vector is denoted as $\langle\Psi|$ and this will be expressed as

$$\langle\Psi| = \sum_{j=0}^{2^n-1} a_j\,\langle\Psi_j|\,, a_j \in \mathbb{R}, \tag{1.9}$$

where the q-vector $\psi_j$ denoted as $\langle\Psi_j|$ is one with the j-th component equal to 1 and others are zero. A qubit is described up to a phase factor by a unit vector in $\mathbb{C}^2$. By convention, we will always take the basis of $\mathbb{C}^2$ to be;

$$[|0\rangle, |1\rangle] = \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right]. \tag{1.10}$$

such that the state of qubit can written in the form

$$|\Psi\rangle \equiv a_0 |0\rangle + a_1 |1\rangle,$$

where $a_o$ and $a_1$ are complex numbers satisfying the normalisation requirement that

$$a_0^2 + a_1^2 = 1.$$

From the postulates of quantum mechanics, it holds that upon taking measurement of a vector $|\Psi\rangle$ we will obtain the state $|0\rangle$ with probability $a_0^2$ (sometimes called amplitudes) and the state $|1\rangle$ with probability (amplitude) $a_1^2$, and these probabilities(amplitudes) must sum to one. More generally;

$$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i |\Psi_i\rangle, \sum_{i=0}^{2^n-1} |a_i|^2 = 1, \tag{1.11}$$

where the $a_i$ are the respective probabilities (amplitudes) of each state. [14]

### Bloch sphere

The Bloch sphere is a geometric representation of qubit states as points on the surface of a unit sphere. Quantum operations on single qubits can be neatly described within the Bloch sphere 1.1.

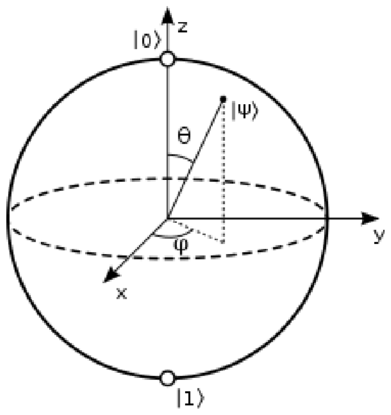


Figure 1.1: Bloch sphere [36] .

[3] In a Bloch sphere, a single qubit state can be written;

$$|\Psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle), \tag{1.12}$$

where $\theta, \phi$ and $\gamma$ are real numbers. The numbers $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$ define a point on a unit three-dimensional sphere. Qubit states with arbitrary values of $\gamma$ are all represented by the same point on the Bloch sphere because the factor of $e^{i\gamma}$ has no observable effects, and we can therefore effectively write:

$$|\Psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle. \tag{1.13}$$

### Tensor product

**Definition 1.2.4.** The Tensor product of two vector spaces $\psi_{\mathbf{1}} = (a_1, a_2, \ldots, a_m)$ and $\psi_{\mathbf{2}} = (b_1, b_2, \ldots, b_n)$ (over the same field) denoted as $\psi_1 \otimes \psi_2$ is itself a vector space, endowed with the operation of bi–linear composition, denoted by $\otimes$ , from ordered pairs in the Cartesian product $\psi_1 \times \psi_2$ to $\psi_1 \otimes \psi_2$ in a way that generalizes the outer product.

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{2^n-1} \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{2^n-1} \end{pmatrix} = \begin{bmatrix} a_1b_1 & a_1b_2 & \ldots & a_1b_n \\ a_2b_1 & a_2b_2 & \ldots & a_2b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_mb_1 & a_mb_2 & \ldots & a_mb_n \end{bmatrix}. \tag{1.14}$$

Or in index notation:

$$(\mathbf{a} \otimes \mathbf{b})_{ij} = a_i b_j.$$

**Lemma 1.2.1.** *The Tensor product of vectors satisfies the following properties:*

1. $(\mathbf{a_1} \otimes \mathbf{a_2})^T = (\mathbf{a_2} \otimes \mathbf{a_1})$.
2. $(\mathbf{a_1} + \mathbf{a_2}) \otimes \mathbf{a_3} = \mathbf{a_1} \otimes \mathbf{a_3} + \mathbf{a_2} \otimes \mathbf{a_3}$.
3. $\mathbf{a_1} \otimes (\mathbf{a_2} + \mathbf{a_3}) = \mathbf{a_1} \otimes \mathbf{a_2} + \mathbf{a_1} \otimes \mathbf{a_3}$ .
4. $b(\mathbf{a_1} \otimes \mathbf{a_2}) = (b\mathbf{a_1}) \otimes \mathbf{a_2} = \mathbf{a_1} \otimes (b\mathbf{a_2})$.
5. $(\mathbf{a_1} \otimes \mathbf{a_2}) \otimes \mathbf{a_3} = \mathbf{a_1} \otimes (\mathbf{a_2} \otimes \mathbf{a_3})$.

In bra-ket notation we shall write:

$$|a\rangle \otimes |b\rangle = \sum_{i=0,j=0}^{2^n-1} a_i b_j \, |ij\rangle \,, \tag{1.15}$$

where

$$|i\rangle \otimes |j\rangle = |ij\rangle \,.$$

This in an N-dimensional vector space can be generalised to

$$|i\rangle_1 \otimes |i\rangle_2 \ldots \otimes |i\rangle_n \equiv |i_1 i_2 \cdots i_n\rangle \,.$$

Thus from above one can see how decomposition of a decomposable q- vector is done. Its worth noting that this observation that lies at the heart of quantum computation allows us to decompose any operation on an entire quantum system into operations on individual components and makes the construction of quantum algorithms much simpler. Further its important to note that non-decomposable q-vectors are said to be entangled and a decomposable q-vectors are also called composite.

### Linear maps

**Definition 1.2.5.** Let $V$ and $W$ be vector spaces over the same field $\mathbb{F}$. A function $f : V \to W$ is said to be a linear map if for any two vectors $a_1, a_2 \in V$ and any scalar $b \in \mathbb{F}$ the following two conditions are satisfied:

1. $f(a_1 + a_2) = f(a_1) + f(a_2)$.
2. $f(ba_1) = bf(a_1)$.

A $\mathbb{C}^{2^n}$–linear map $T : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ (also called an operator) is determined by the $2^n$ images $t_j = \sum_{i=0}^{2^n-1} \psi_i |i\rangle$ and its operation can be defined as;

$$T\left(\sum_{i=0}^{2^n-1} \psi_i |i\rangle\right) = \sum_{i=0}^{2^n-1} T(\psi_i |i\rangle) = \sum_{i=0}^{2^n-1} \psi_i |t_i\rangle . \qquad (1.16)$$

The linear map T, is a (unique) linear bijective map $T|i\rangle = t_i$ and in the sequel, this observation will be the basic method used to prescribe operators. [32]

### Unitary Matrices

**Definition 1.2.6.** A Matrix A is said to be self adjoint if $A = A^\dagger$.

**Definition 1.2.7.** A Matrix A said to be unitary if it satisfies the relation $A^\dagger A = AA^\dagger = I$, where I is an identity matrix and $A^\dagger$ is called the conjugate transpose.

**Definition 1.2.8.** A q-computation is a unitary matrix A of dimension $2^n$.

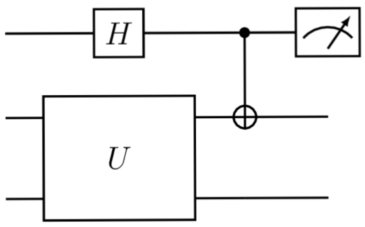
For given a matrix $A_{m,n} = (a_{ij})$ we shall define its transpose as $A^T_{m,n} = (a_{ij})$, its conjugate as $\bar{A} = (\bar{a}_{ij})$ and its conjugate transpose as $A^\dagger = (\bar{a}_{ij})^T$. Since a set of unitary matrices constitutes a group under multiplication, it forms a fundamental work group that will be a used in our construction. Q-computations come with special characteristics of composition and reversibility thus we can say that a composition of two q-computations of order $n$ is a q–computation of order $n$; and reversibility guarantees that the inverse of a q-computation of order $n$ is a q–computation of order $n$. [32]

## 1.3. Circuits in Quantum computing

### Quantum gates

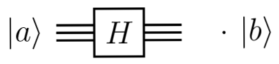
Quantum gates can be represented as matrices describing transformations on qubits. The normalization condition $a_1^2 + a_2^2 = 1$ must hold for all states, so all actions performed on qubits must give rise to unitary vectors . The matrices describing such transformations are unitary matrices. Since unitary matrices always have an inverse, all transformations described by them are reversible. Consequently all quantum gates must be reversible (except the measurement gate which destroys the quantum state).This implies that the circuit must be able to be run in reverse.

A quantum circuit as represented by a diagram in equation 1.17 constitutes of lines in which each line represents the timeline of a qubit read from left to right. A gate acting on a qubit is denoted by the symbol of the gate placed on the qubit it is acting on. When describing quantum gates as discussed in 1.3 below, the corresponding circuit representation can be shown together with the equivalent matrix representation.



(1.17)

The horizontal lines in 1.18 are called q-wires.



(1.18)

where we have the q-input **a** and q-output **b**. A q- computation of order 1 can be expressed in matrix form as;

$$\psi = \begin{bmatrix} a_0 & a_1 \\ -\bar{a}_1 & \bar{a}_0 \end{bmatrix} = \begin{bmatrix} e^{-i\lambda}\cos\frac{\theta}{2} & -e^{i\mu}\sin\frac{\theta}{2} \\ e^{-i\mu}\cos\frac{\theta}{2} & e^{i\lambda}\cos\frac{\theta}{2} \end{bmatrix}. \tag{1.19}$$

with the relation

$$a_0\bar{a}_0 + a_1\bar{a}_1 = 1$$

being satisfied by the matrix and $\theta \in [0, \pi]$. By further notation, if we let

$$\lambda = \frac{\beta+\gamma}{2} \text{ and } \mu = \frac{\gamma-\beta}{2},$$

then we can write;

$$\begin{bmatrix} e^{-i\lambda}\cos\frac{\theta}{2} & -e^{i\mu}\sin\frac{\theta}{2} \\ e^{-i\mu}\cos\frac{\theta}{2} & e^{i\lambda}\cos\frac{\theta}{2} \end{bmatrix} = R_Z(\beta)R_Y(\theta)R_Z(\gamma), \tag{1.20}$$

where,

$$R_Z = \begin{bmatrix} e^{-\frac{i\psi}{2}} & 0 \\ 0 & e^{\frac{i\psi}{2}} \end{bmatrix} \tag{1.21}$$

and

$$R_Y = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}. \tag{1.22}$$

Thus we conclude that a general element of $SU^{(1)}$ (the elements of $U^{(1)}$ with determinant 1) has the form

$$U(\theta, \beta, \gamma) = R_Z(\beta)R_Y(\theta)R_Z(\gamma). \tag{1.23}$$

The geometrical meaning of the statement above is closely related to rotations in Euclidean 3-dimensional space. [32]

**Single qubit gates**

1. **Pauli gates**: The Pauli gates are four namely the X-, Y- and Z-gate.

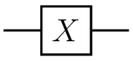
   (a) **The X -gate** : This rotates the qubit around the X-axis of the Bloch sphere. It is also called the bit flip gate and it can be represented as;

   $$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| \tag{1.24}$$

   As an example of its operation, we can see that it flips the qubit $|0\rangle$ to $|1\rangle$ when applied as shown below;

   $$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle .$$

   As a gate it can be represented as;

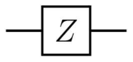
   (b) **The Z -gate** : This rotates the qubit around the Z-axis of the Bloch sphere. It is also called the phase flip gate and it can be represented as;

   $$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \tag{1.25}$$

   As an example of its operation, we can see that it flips phase of the qubit $|1\rangle$ to $-|1\rangle$ when applied as shown below;

   $$Z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -|1\rangle .$$

   As a gate it is represented as;

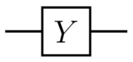
   (c) **The Y -gate** : This rotates the qubit around the Y-axis of the bloch sphere.It is both a bit and phase flip gate and it satisfies the relation $Y = iXZ$. It can be represented as;

   $$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0| \tag{1.26}$$

   As an example of its operation, we can see that it flips phase of the qubit $|1\rangle$ to $-i|0\rangle$ when applied as shown below;

   $$Z|1\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i \\ 0 \end{bmatrix} = -i|0\rangle .$$

   As a gate it can be represented as;

(d) **The I -gate** : This does nothing to the qubit.it is also called the identity gate and It can be represented as;

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| \tag{1.27}$$

As an example of its operation, we can see that it does nothing to the qubit $|0\rangle$ when applied as shown below;

$$I|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle .$$

As a gate, it can be represented as;

$$-\boxed{I}-$$

**NOTE**: The Pauli matrices are self-adjoint and unitary.

2. **The Hadamard -gate** : This is a q- computation of order one that is unitary and self adjoint. Geometrically, The Hadamard gate performs a $\pi$ rotation about the X-axis and a $\frac{\pi}{2}$ rotation about the Y-axis in the Bloch sphere. The gate is used to put the target qubit into a superposition of single qubit states having an equal chance of being measured as $|0\rangle$ or $|1\rangle$. It can be represented as;

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.28}$$

As an example of its operation,we can see that it puts state $|0\rangle$ in the state of equal superposition of $|0\rangle$ and $|1\rangle$ when applied to it and similarly the same to $|1\rangle$ as shown below;

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

As a gate it can be represented as;

$$-\boxed{H}-$$

3. **Phase shift gate** : This is a q- computation of the form ;

$$S_\alpha = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{bmatrix} \tag{1.29}$$

The circuit diagram representation of the phase shift gate is;

$$-\boxed{T}-.$$

The $S_\alpha$ -gate is a phase shift gate related to the Pauli Z-gate via $S_\alpha^4 = Z$ , meaning that performing a $S_\alpha$-gate four times will yield the same result as applying a Z-gate once . The $S_\alpha$-gate corresponds to a rotation of $\frac{\pi}{4}$ around the Z-axis in the Bloch sphere.

**Two qubit gates**

**The controlled U– gate**: This is a q-computation of order 2. A qubit can be added as a control bit to any gate, so that its operation will only be executed on the target qubit if the control bit is a one. To indicate a controlled–U gate, a C is added to the gate's name. Common examples of controlled–U gates are the controlled not (CNOT) and swap (SWAP) gates. A controlled– U is denoted by;

$$C_{12}(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}, \tag{1.30}$$

where $U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$.

1. **CNOT gate**:

   A controlled-not gate is a special case of controlled–U gates which is equal to a CX-gate and can be represented as;

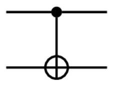
$$C_{12}(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{1.31}$$

   The CNOT gate flips the second qubit (the target qubit) if and only if the first qubit (the control qubit) is $|1\rangle$. Below is a caption of all the possible results obtained from action of CNOT gate.

Table 1.1: Operation of CNOT gate

| Before | | After | |
|---|---|---|---|
| Control | Target | Control | Target |
| $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ |
| $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ |

   The circuit diagram representation of the controlled not gate is;



.

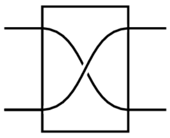
2. **The Swap gate** : This a q-computation of order 2. It is another special case the controlled U gate that swaps the state of the two qubits involved in the operation and it is denoted by;

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.32}$$

Table 1.2: Operation of SWAP gate

| Before | | After | |
|---|---|---|---|
| $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ |
| $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ |
| $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ |

Above is a caption of all the possible results obtained from action of SWAP gate
The circuit diagram representation of the swap gate is;



**Three qubit gates**

1. **The Toffoli gate** This a q-computation of order 3 which is also called double controlled not gate or a Toffoli gate.it can be represented as;

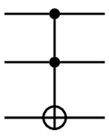
$$K = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.33}$$

The CCNOT gate flips the third qubit (the target qubit) if and only if the first and second qubits (the control qubits) is $|1\rangle$. Below is a caption of all the possible results obtained from action of CCNOT gate.

Table 1.3: Operation of CCNOT gate

| Before | | | After | | |
|---|---|---|---|---|---|
| Control 1 | Control 2 | Target | Control 1 | Control 2 | Target |
| $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ |
| $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ |
| $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ |
| $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ |
| $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ |

The circuit diagram representation of the Toffoli gate is;



2. **The Fredkin gate**: This is a q-computation of order 3 which at times is called CSWAP gate. The Fredkin gate transmits the first bit unchanged and swaps the last two bits if and only if the first bit is $|1\rangle$.it denoted as;

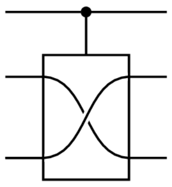
$$K = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.34}$$

Below is a caption of all the possible results obtained from action of CSWAP gate.

Table 1.4: Operation of CSWAP gate

| Before | | | After | | |
|---|---|---|---|---|---|
| Control | target 1 | Target 2 | Control | target 1 | Target 2 |
| $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|0\rangle$ |
| $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ |
| $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ |
| $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|0\rangle$ |
| $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ | $\|0\rangle$ | $\|1\rangle$ |
| $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ | $\|1\rangle$ |

The circuit diagram representation of the toffoli gate is;

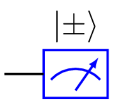


## Q-procedure

A q-procedure is a sequence of actions each of which is either a q-computation or a q-measurement that are applied successively to $|0\cdots0\rangle$ (the default initial state (zero state) of the q-memory) [32].

## Q-measurement

This is the last step of any q-computation and it is non reversible process. In this step, a measurement gate is used to project a qubit's state onto the basis vectors $|0\rangle$ and $|1\rangle$ and once a qubit is measured its quantum state is destroyed. A measurement in a quantum circuit is represent by the circuit element below called the meter [32]



(1.35)

### Quantum register

Like classical computers, quantum computers use quantum registers made up of multiple qubits. When collapsed, quantum registers are bit strings whose length determines the amount of information they can store. In superposition, each qubit in the register is in a superposition of $|0\rangle$ and $|1\rangle$ and consequently a register of n qubits is in a superposition of all $2^n$ possible bit strings that could be represented using n bits. The state space of a size-n quantum register is a linear combination of n basis vectors, each of length $2^n$.

$$\sum_{i=0}^{2^n-1} \psi_i |i\rangle \tag{1.36}$$

Here $i$ is the base-10 integer representation of a length-n number in base-2. As with single qubits, the squared absolute value of the amplitude associated with a given bit string is the probability of observing that bit string upon collapsing the register to a classical state, and the the squares of the absolute values of the amplitudes of all $2^n$ possible bit configurations of an n-bit register sum to unity [32].

$$\sum_{i=0}^{2^n-1} |\psi_i|^2 = 1 \tag{1.37}$$

### Q-computer and Q-algorithm

A q-computer of order n is a system endowed with the following operations [32]:

1. **Q-memory**

   This is a store capable of holding any unit q-vector $a \in \mathbb{C}^n$ which is called the q-memory state.The elementary procedures 2, 3 and 4 below, are applied successively one at a time to the current q-memory state during quantum processing.

2. **One Q-bit rotations**

   This is a q-computation on the i-th q-bit by the U-gate defined as follows:

   $$|\cdots\rangle = |\cdots\rangle |\cdots j_i \cdots\rangle \Longrightarrow |\cdots\rangle U |j_i\rangle |\cdots\rangle$$

3. **controlled negations** $N_{r,s}$

   This q-computation negates the s-th q-bit if (an only if) the r-th q-bit is $|1\rangle$. It is the linear map which is the identity on the basis q-vectors of the form $|\cdots 0_j \cdots\rangle$ and such that

   $$|\cdots 1_j \cdots 0_k \cdots\rangle \Longrightarrow |\cdots 1_j \cdots 1_k \cdots\rangle$$
   $$|\cdots 1_j \cdots 1_k \cdots\rangle \Longrightarrow |\cdots 1_j \cdots 0_k \cdots\rangle$$

   This kind of elementary q-procedures will be called CNOT gates.

4. **Measurement**

   The quantum measurement is as discussed in the section about Q-measurement.

**Q-algorithms**

A q-algorithm is a q-procedure involving elementary q-procedures only. We will say that a q-algorithm is internal if it does not involve measurements. A q-algorithm (internal or not) will be called restricted if it only involves restricted U-gates. As a measure of the complexity of a q-algorithm we take the number of elementary gates it involves. A q-algorithm is polynomial if its complexity is bounded by a polynomial in n [32].

## 1.4. Computational complexity

Computational complexity is an important theme as far as the theory of computation is concerned and it focuses on the classification of problems according to their inherent difficulty in computation both in space and time used. This thus culminates in a what is called performance of a computer algorithm. To measure performance of an algorithm, we determine how the resource requirements of space and time scale as the size of the computational problem being solved gets larger or harder. By definition,we shall call a computational problem a task that can be solved by either a deterministic, probabilistic, non-deterministic or quantum computer via an algorithm. By classifying computational problems we inherently address questions of complexity.

Finally we define time complexity to be the total number of steps the machine makes before it halts and outputs the answer where is complexity in space relates to the amount of memory space required in solving a computational problem as function of its input [1].

### Big-O and small-o notation

The complexity of an algorithm is often expressed using big–O or small–o notation though big–O is more common. They are both mathematical notions that describes the limiting behaviour of a function when the argument tends towards a larger or smaller value respectively that is (positive or negative infinity respectively ).

In computer science, the notions are used to classify algorithms according to how their running time or space requirements grow or reduce as the input size grow or reduces respectively.

Quite often we are usually interested in the worst case scenario of a computational problem and thus the notions above help in giving us the upper or lower bound of the computational time and memory space that can be used to perform a given computation. Below are some of the classical Big-O and small–o notations [1].

Table 1.5: Big–O and Small–o notation

| Big–O | $O(1)$ | $O(\log n)$ | $O(n), O(n^2), O(n^3)$ | $O(2^n), O\left(2^{n^2}\right)$ |
|---|---|---|---|---|
| Small–o | $o(1)$ | $o(\log n)$ | $o(n), o(n^2), o(n^3)$ | $o(2^n), o\left(2^{n^2}\right)$ |
| Function | Constant | Logarithmic | Linear, quadratic, polynomial | Exponential |

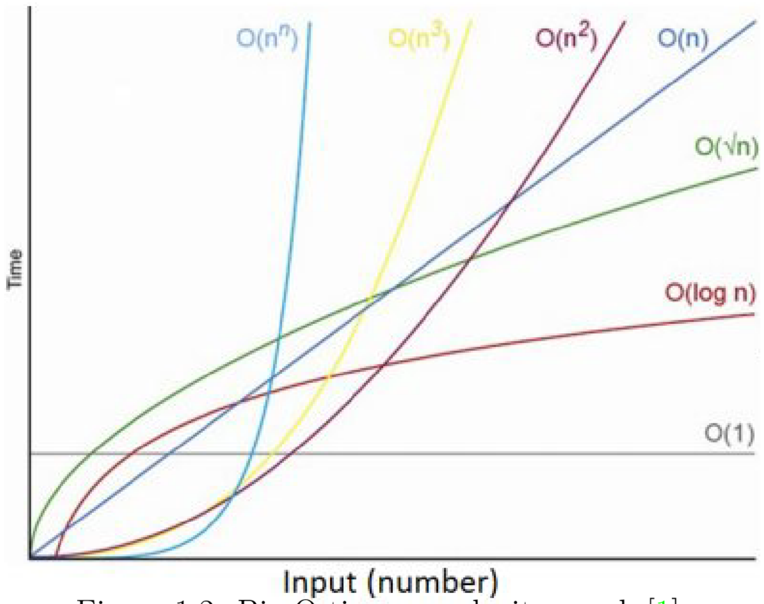


Figure 1.2: Big-O time complexity graph [1].

An algorithm is said to be of polynomial time if its running time is upper bounded by a polynomial expression in the size of the input for the algorithm, that is, $T(n) = O\left(n^k\right)$ for some constant $k$. All the basic arithmetic operations can be done in polynomial time.

On the other hand, an algorithm is said to take super polynomial time if $T(n)$ is not bounded above by any polynomial. For example, an algorithm that runs for $2^n$ steps on an input of size $n$ requires super polynomial time. For our case its those that require exponential time.

### Complexity Classes

The known common complexity classes in computing include [26]:

1. **P** – Polynomial time: Problems that can be solved in polynomial time on a classical computer. This includes Problems of $O(1), O(\log(n)), O(n), O(n\log(n))$, and $O\left(n^2\right)$, as examples.

2. **NP**– Non-deterministic polynomial time: A problem is in **NP** if whenever the answer is "yes," there's a polynomial size witness or proof for the yes-answer which a polynomial-time algorithm can verify.

   A problem **B** is said to be **NP**–Complete if and only if it is in **NP** and all **NP** problems are polynomial-time reducible to **B**.

   A problem **B** is said to be **NP**–hard if it is in **NP** but not all **NP** problems are polynomial-time reducible to **B**.

3. **PSPACE** – Polynomial space: It is the class of decision problems that are solvable by some algorithm whose total space usage can be upper-bounded by a polynomial in terms of size. This class focuses on memory resources as opposed to time.

4. **BPP**–Bounded-error probabilistic polynomial time: BPP is the class of decision problems for which there exists a polynomial-time randomized algorithm that solves

the problem with success probability of at least 2/3. Randomized algorithms give us faster time results than a deterministic algorithm trying to achieve the same goal. Problems that are in BPP either have a deterministic algorithm that can run in polynomial time or have a probabilistic algorithm which will give the wrong answer to a decision problem no worse than 1/3 of the time. Below are the complexity classes that arise with quantum computing.

5. **BQP** - Bounded-error quantum polynomial time: **BQP** is the primary complexity class for quantum computers and it is the quantum analogue of the class BPP for classical computation. A decision problem is in BQP if it can run in polynomial time and yields a correct result with a high probability.

   It is believed that BQP contains problems which are thought to be intractable in the classical regime but are thought to be tractable in bounded-error polynomial time for a quantum computer but still this awaits proof.

   **EQP**- Exact quantum polynomial time:
   This is the set of decision problems solvable by a quantum computer that yields the correct answer with a probability of 1. In other words, this class is the same as BQP except that it must give the correct answer with probability of 1 instead of having some bounded error margin. [1]

   **Note** that a quantum computer does not render all NP problems tractable; only problems that have some structure we can exploit can be handled efficiently by a QC. For example, Shor's algorithm takes advantage of the periodicity of the function which then enables us to solve the equivalent problem of factoring a large number.

   **QMA** - Quantum Merlin-Arthur:
   QMA is the quantum analogue to the non-probabilistic class MA. In a Merlin-Arthur (MA) problem, a prover (Merlin) sends a message to a verifier (Arthur). In the classical complexity class MA, Arthur can verify the message in polynomial time.

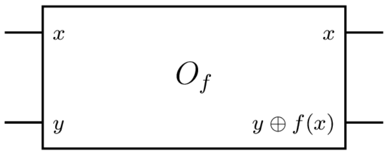
Table 1.6: Table of Classical and Quantum Complexity Classes

| Classical | Quantum |
|---|---|
| P | EQP |
| BPP | BQP |
| NP | QMA |

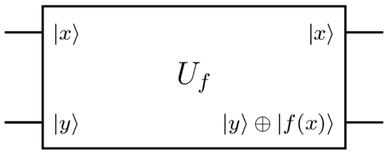
# 2. Grover's algorithm

In this chapter, we shall give a review some literature relevant to our study, a rough overview of some classical quantum algorithms and a broad description of Grover's algorithm. Since Grover's algorithm and other classical quantum algorithms are oracle based algorithms, we shall begin by giving a precise notion of what an oracle is. An oracle or sometimes called a black box function is a system which can be viewed in terms of its inputs and outputs (or transfer characteristics), without any knowledge of its internal workings. We shall classify the oracle types into classical and quantum oracle types.

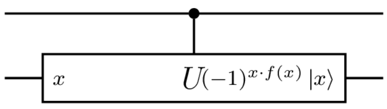
A classical oracle of a function $f(\cdot)$ is a "black box" that when given a value $x$, computes $f(x)$. A special type of a classical oracle is the reversible oracle seen below.



Unlike a reversible classical oracle, a Quantum oracle may be given a superposition of inputs $\sum_x \alpha_x |x, 0\rangle$, and produces a superposition of pairs $\sum_x \alpha_x |x, f(x)\rangle$, as seen below.



When $f$ is binary, we define the phase quantum oracle as a black box that flips the phase of its input state $|\psi\rangle$ if and only if $f(x) = 1$. Below is an illustration of a phase quantum oracle.



Its quite clear to see that a quantum oracle of $f(\cdot)$ is sharply better than its classical counterpart because when fed with a single basis state (no superposition) it easily simulates a classical oracle.

## 2.1. Some literature on Grover's algorithm

This section will show research conducted on the use of Grover's search algorithm.

1. **(1996) A fast quantum mechanical algorithm for database search**

   [23] This is the main paper by Lov K. Grover which described the Quantum search algorithm. In the paper it is stated that in an unsorted database containing N records of which just one satisfies a particular property with the problem of identifying the unique element. Any classical algorithm deterministic or probabilistic would take $O(N)$ steps since on the average it will have to examine a large fraction of the N records where as Quantum mechanical systems could do several operations simultaneously and achieve the same goal in $O(\sqrt{N})$ .

2. **(1996) Strengths and Weaknesses of Quantum Computing**

   [5] This paper addressed the question of whether all of NP can be efficiently solved in quantum polynomial time by a quantum computer proving that relative to an oracle chosen uniformly at random with probability 1 the class NP cannot be solved on a quantum Turing machine in time $O(2^{n/2})$. It was also revealed that relative to a permutation oracle chosen uniformly at random with probability 1 the class $NP \bigcap co - NP$ cannot be solved on a quantum Turing machine in time $O(2^{n/3})$. In this paper it was stated that the former bound was tight since Lov K. Grover's work showed how to accept the class NP relative to any oracle on a quantum computer in time $O(2^{n/2})$.

3. **(1996) Tight bounds on quantum searching**

   [9] This paper provided a tight analysis of Grover's recent algorithm for quantum database searching. It gives a simple closed-form formula for the probability of success after any given number of iterations of the algorithm. Furthermore, the paper analyses the behaviour of the algorithm when the element to be found appears more than once in the table and they provided a new algorithm to find such an element even when the number of solutions is not known ahead of time. Finally a lower bound on the efficiency of any possible quantum database searching algorithm is provided and the paper shows that Grover's algorithm nearly comes within a factor comes within 2.62 percent of being optimal in terms of the number of probes required in the table.

4. **(2004) Quantum query complexity of some graph problems**

   [18] This paper showed the potential of Quantum algorithm to speed up some of the classical graph problems such as:

   (a) Minimal Finding :

       Suppose we are given a function $f$ defined on a domain of size $n$ and we want to find an index $i$ so that $f(i)$ is a minimum in the image of $f$.

   (b) Minimum Spanning Tree:

       In this section undirected graphs with weighted edges are considered. In Minimum Spanning Tree we wish to compute a cycle free edge set of maximal cardinality that has minimum total weight.

   (c) Connectivity:

       A special case of Minimum Spanning Tree when all edge weights are equal is Graph Connectivity. The input is an undirected graph and the output is a spanning tree provided the graph is connected.

5. **(2016) On the advantages of using relative phase Toffolis with an application to multiple control Toffoli optimization**

   [35] In this paper an approach for systematic optimization of quantum circuits via replacing suitable pairs of the multiple control Toffoli gates with their relative phase implementations was reported. This operation preserves the functional correctness. The advantage can be witnessed through the optimized resource counts. Our demonstrated optimizations include a simultaneous optimization of the T count

by a factor of 4/3 in the leading constant the CNOT count by a factor 2 in the leading constant and the number of ancillary qubits by a factor of 2 in the leading constant.

6. **(2017) Complete 3-Qubit Grover search on a programmable quantum computer**

   [21] In this paper results for a complete three-qubit Grover search algorithm using the scalable quantum computing technology of trapped atomic ions with better-than-classical performance were reported. Two methods of state marking were used for the oracles namely:

   (a) a phase-flip method where the oracle is implemented with a circuit consisting of Z and $C^k(Z)(k \leq n-1)$ gates that directly flip the phase(s) of the state(s) to be marked and

   (b) a Boolean method which requires an ancilla qubit that is directly equivalent to the state marking scheme required to perform a classical search.

   Results of deterministic implementation of a Toffoli-4 gate which is used along with Toffoli-3 gates to construct the algorithms were also reported and it was revealed that these gates have process fidelities of 70.5 percent and 89.6 percent respectively.

7. **(2018) An Introduction to Quantum Search Algorithm and Its Implementation**

   [17] This article described Grover's search with an example of its applications and limitations. Also exploration the functionality of quantum circuit oracle circuit that is particular to Grover's was made. This article concluded with the Grover's search advantage over classical search.

8. **(2020) Grover's search algorithm for n qubits with optimal number of iterations**

   [33] In this work, a general scheme for the construction of n-qubit Grover's search algorithm with $1 \leq M \leq N$ target states is presented, along with the procedure to find the optimal number of iterations for a successful search. It is also shown that for given N and M, there is an upper-bound on the success probability of the algorithm.

## 2.2. Grover based quantum algorithms

In this section we will discuss the most common Grover based quantum search algorithm.

### Amplitude amplification

Originally Grover's Algorithm was designed for search of a single item in an unstructured search space. [11] further developed the algorithm by generalizing its core idea of amplitude amplification. In this algorithm knowledge of the exact solution is not required and it provides a possibility for multiple solution search. Further, they introduced amplitude amplification which uses Shor's phase estimation to estimate the success probability of a quantum algorithm. Since there are some polynomial-time heuristic search algorithms, they showed that the combination of classical heuristic and amplitude amplification would still lead to a quadratic speedup in the estimated time a solution is found.

#### Fix–point quantum search

Because in situations where the fraction of valid states within the overall search space is unknown Grover's algorithm and the generalized amplitude amplification are both hard to use. [25] suggested the so called fix-point search in which one only needs a lower bound on this fraction and always amplifies marked states through running the algorithm long enough which improves the success probability of the solution asymptotically but the only cost of this algorithm is that the initial quadratic speedup is lost. However, [39] recently presented a fixed-point search that achieves both quadratic speed up and improved success probability of the solution through adjusting the phases of Grover's reflection operator. It can be used as a subroutine for every amplitude amplification application and eliminates the need to run the algorithm multiple times.

#### Grover adaptive search

Grover Adaptive Search is based on the work of [19] and [22] illustrated its application on combinatorial optimization problems in his work which uses amplitude amplification from [10] to solve the minimum searching problem with Grover's quadratic speedup.

In the work by [12] an illustration on how to implement pure adaptive search with the generalized version of Grover's Search Algorithm was made. The Grover Adaptive Search searches for the optimum value of a function by iteratively applying Grover's Search Algorithm while sampling randomly from all the better solutions and uses them to further optimize the solution. [22] uses this method and it provides a framework for an efficient automated oracle construction in its core. This framework is efficient for constraint polynomial binary optimization and especially for quadratic unconstrained binary optimization which are common to model combinatorial optimization problems.

## 2.3. Classical quantum algorithms

Most of the classical algorithms as described in this section are known as "black box" or "query model" quantum algorithms. They are endowed with an underlying function which is unknown to us but through constructing another function called an oracle which we can query, we are able to determine the relationship of specific inputs with specific outputs. We shall consider and provide more insight as into major classical algorithms that are highly correlated to Grover's algorithm in addition to Grover's algorithm.

#### Deutsch Josza algorithm

The Deutsch Josza algorithm is a generalisation of the Deutsch's algorithm [2]. In the Deutsch–Jozsa problem, we are given an oracle that implements some function $f : (0,1)^n \to (0,1)$ that is a function that takes n-digit binary values as input and produces either a 0 or a 1 as output for each such value with a promise that the function is either constant (0 on all outputs or 1 on all outputs) or balanced (returns 1 for half of the input domain and 0 for the other half). The task then remains to determine if $f$ is constant or balanced by using the oracle. Clearly, for any classical algorithm run on a deterministic computer this can take it $2^{n-1}+1$ evaluations of $f$ in the worst case to prove that $f$ is constant which is slightly over half the set of inputs must be evaluated and their outputs found to be identical. Under Deutsch Josza, the lucky case occurs where the function is balanced and the first two output values that happen to be selected are different. For a randomized algorithm run on a classical computer, a constant $p$ evaluations of the function suffices to produce the correct answer with a high probability (failing with probability $\epsilon \leq 1/2^p$ with $p \geq 1$). However, if we want an answer that is always correct, we always need $p = 2^{n-1}+1$ evaluations. The Deutsch-Jozsa quantum algorithm produces an answer that is always correct with a single evaluation of $f$. Its important to ensure that the oracle computing $f(x)$ from $x$ doesn't decohere $x$ as the Deutsch–Jozsa algorithm wont work in case $x$ is decohered. Also, it must not leave any copy of $x$ lying around at the end of the oracle call. The algorithmic steps of Deutsch Josza algorithm are;

1. We input $n+1$ bit state $|0\rangle^{\otimes n}|1\rangle$ . That is, the first n bits are each in the state $|0\rangle$ and the final bit is $|1\rangle$ .

2. We apply Hadamard transform 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 apply the quantum oracle which maps the state $|x\rangle|y\rangle$ to $|x\rangle|y\oplus f(x)\rangle$. Applying the quantum oracle gives;

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

   where for each $x, f(x)$ is either 0 or 1. Testing these two possibilities, we see the above state is equal to;

$$\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle).$$

   We may ignore $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ and therefore we remain with;

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

4. Applying the Hadamard transform to each qubit we obtain;

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

   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 bit wise product.

5. Finally we perform the measurement and 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). In other words, the final measurement will be $|0\rangle^{\otimes n}$(i.e. all zeros) if $f(x)$ is constant and will yield some other states if $f(x)$ is balanced.

Below is the circuit diagram of deutsch josza algorithm.

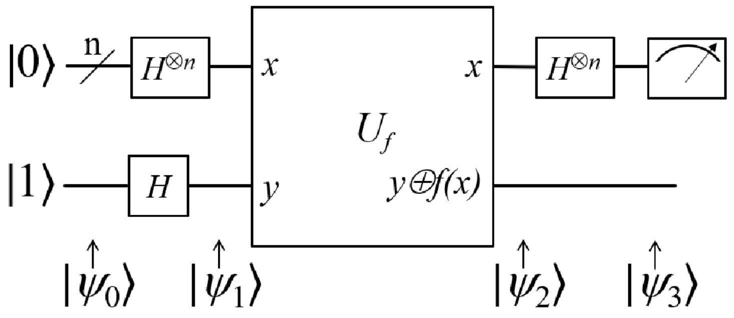


Figure 2.1: Deutsch-Jozsa. Source: Wikipedia
Figure 2.1 shows Deutsch-Jozsa quantum algorithm.

**Deutsch's algorithm**

Deutsch's algorithm was the first algorithm to show the advantage quantum computers have the classical ones. It is a special case of the general Deutsch–Jozsa algorithm. The oracle in this case checks the condition $f(0) = f(1)$ which is equivalent to checking $f(0) \oplus f(1)$ which is again equivalent to a quantum XOR gate implemented as a Controlled NOT gate where if zero, then f is constant, otherwise f is balanced. The procedure is as follows;

1. We begin with an input of two-qubit state $|0\rangle |1\rangle$ and we apply a Hadamard transform to each qubit. This yields
$$\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle).$$

2. Applying the quantum oracle which maps $|x\rangle |y\rangle \, to \, |x\rangle |y \oplus f(x)\rangle$ yields;
$$\begin{cases} \frac{1}{2}(|0\rangle (|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle) + |1\rangle (|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle) \\ = \frac{1}{2}((-1)^{f(0)} |0\rangle (|0\rangle - |1\rangle) + (-1)^{f(1)} |1\rangle (|0\rangle - |1\rangle)) \\ = (-1)^{f(0)} \frac{1}{2} \left(|0\rangle + (-1)^{f(0) \oplus f(1)} |1\rangle\right) (|0\rangle - |1\rangle). \end{cases}$$

Ignoring the last bit and the global phase then yields the state

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

3. Applying a Hadamard transform to this state we get

$$\begin{cases} \frac{1}{2}(|0\rangle + |1\rangle + (-1)^{f(0)\oplus f(1)} |0\rangle - (-1)^{f(0)\oplus f(1)} |1\rangle) \\ = \frac{1}{2}((1 + (-1)^{f(0)\oplus f(1)}) |0\rangle + (1 - (-1)^{f(0)\oplus f(1)}) |1\rangle) \end{cases}$$

   $f(0) \oplus f(1) = 0$ if and only if we measure $|0\rangle$ and $f(0) \oplus f(1) = 1$ if and only if we measure $|1\rangle$ .

So with certainty we know whether f(x) is constant or balanced.

### Bernstein and vazirani algorithm

The Bernstein–Vazirani quantum algorithm was invented by Ethan Bernstein and Umesh Vazirani in 1992 and it solves the Bernstein–Vazirani problem [7]. It's a restricted version of the Deutsch–Jozsa algorithm in such a way that instead of distinguishing between two different classes of functions i.e balanced or constant, it tries to learn about a string, $s$ encoded in a function. The Bernstein Vazirani algorithm was the first algorithm developed that shows a clear separation between quantum and classical computing because even if we allow for some room of error, it still has an advantage over the classical ones in terms of speedup which fails in the Deutsch Josza case. It was designed to prove an oracle separation between complexity classes BQP and BPP. Given an oracle that implements a function $f\colon \{0,1\}^n \to \{0,1\}$ in which $f(x)$ is defined to be the dot product between $x$ and un known string secret string $s \in \{0,1\}^n$ modulo 2 i.e $f(x) = x \cdot s = x_1s_1 + x_2s_2 + \cdots + x_ns_n$ we ought to find $s$.

Classically, the most efficient method to find the secret string is by evaluating the function $n$ times with the input values $x = 2^i$ for all $i \in \{0, 1, ..., n-1\}$ In contrast to the classical solution which needs at least $n$ queries of the function to find $s$, only one query is needed using quantum computing. The quantum algorithm is as follows:

1. Apply a Hadamard transform to the $n$ qubit state $|0\rangle^{\otimes n}$ to get $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$ .

2. Apply the oracle $U_f$ which maps $|b\rangle |x\rangle \to |b \oplus f(x)\rangle |x\rangle$ . This makes a transformation of $|x\rangle$ to $(-1)^{f(x)} |x\rangle$.By applying this oracle to $\frac{|0\rangle - |1\rangle}{\sqrt{2}} |x\rangle$. This transforms the superposition into ;

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

3. Applying the Hadamard transform to each qubit. This gives that for qubits with $s_i = 1$, its state is converted from $|-\rangle$ to $|1\rangle$ and for qubits where $s_i = 0$, its state is converted from $|+\rangle$ to $|0\rangle$.

4. Perform a measurement in the standard basis $(|0\rangle, |1\rangle)$ on the qubits to obtain $s$.

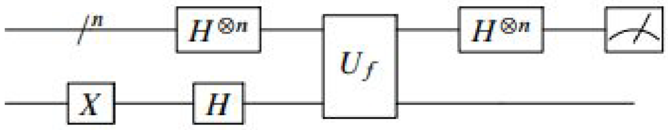


Figure 2.2: Bernstein–Vazirani, Source: Wikipedia

Graphically, the algorithm may be represented by the following diagram.

Mathematically we can summarise the steps as illustrated below;

$$|0\rangle^n \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \xrightarrow{U_f}$$

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

$$\begin{cases} = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)+x\cdot y} \\ = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot s+x\cdot y} \\ = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot(s\oplus y)} \\ = 1 \text{ if } s\oplus y = \vec{0}, 0 \text{ otherwise} \end{cases} = |s\rangle$$

Since $s \oplus y = \vec{0}$ is only true when $s = y$, this means that the only non-zero amplitude is on $|s\rangle$. So, measuring the output of the circuit in the computational basis yields the secret string $s$.

### Simons algorithm

As stated in [34] Simon's problem is a computational problem that can be solved exponentially faster on a quantum computer than on a classical (or traditional) computer. We Assume we are presented with a quantum oracle $U_f$ of a function $f\colon \{0,1\}^n \to \{0,1\}^n$ that is 2-to-1. We are assured that $\forall x\colon f(x \oplus b)$ for some unknown constant b. The problem is to find that b. Below is a description of The quantum algorithm to solve this problem as adapted from [27] and it requires two n-bit registers;

1. (a) Initialize the input register to $H\,|0\rangle$, and the output register to $|0\rangle$.
   (b) Apply the oracle $U_f$ to the combined register.
   (c) Apply the Hadamard transform $H$ to the input register.
   (d) Measure the input register. The result $m_i$ satisfies $b \cdot m_i = 0$ where "$\cdot$" denotes the inner product mod 2.
2. Repeat these steps until acquiring n linearly independent $m_i's$.

Extracting b from them is a straightforward polynomial classical process (Gauss-Jordan elimination) which requires no oracle queries. The average number of repetition (and oracle queries) required to find the linearly independent set of $m_i's$. is $O(n)$. An m with

$a \cdot m_i = 0$ is never measured because of a special feature of the Hadamard transform when $|x\rangle + |x \oplus b\rangle$ are transformed their "odd" $|m\rangle$ elements cancel out. A classical algorithm requires $O(2^{\frac{n}{2}})$ oracle queries on average. Hence in the case of this problem quantum computation is exponentially faster than classical computation.

### Shors algorithm

As stated in [27] Shor's algorithm solves two problem (Factorization and Discrete Logarithm) whose hardness is the core of the security of RSA and DiffieHellman cryptographic protocols respectively. In his paper, [31] Shor showed that both the Factorization and the Discrete Logarithm problems are reducible to finding the period of a function. In the case of factorization, finding the prime factors$p, q$ of $N$ is equivalent to finding the period of $f_{N,a}(x) = ax \bmod N$ (This is true for most $a \in 2, \cdots, N-1$). Further, Shor showed how a period can be found efficiently using a quantum computer. The key algorithm to perform this task is the Quantum Fourier Transform

$$\sum_x f(x) |x\rangle \xrightarrow{QFT} \sum_y \left( \frac{1}{\sqrt{N}} e^{\frac{2\pi ixy}{N}} f(x) \right) |y\rangle .$$

When $|y\rangle$ is measured, the outcome is in the close vicinity of the period of $f(x)$ with high probability. Two other important aspects are answered by Shor that is how the initial distribution $\sum_x f(x) |x\rangle$ can be created efficiently for the given $f_{N,a}(x)$ and how to perform QFT efficiently. The overall complexity of Shor's algorithm is $O \log^3(N)$ (polynomial in the number of bits) while the best known classical algorithm's complexity is $\theta(e^{c(\log(N))\frac{1}{3}(\log(N))\frac{2}{3})})$ (super-polynomial in the number of bits) [27].

## 2.4. Grover's algorithm

As coined by Lov Grover in his paper [23], Grover's Algorithm is a quantum algorithm used to make a search over an unordered set of $N = 2^n$ items to look for a unique element satisfying a specific condition.

**An unstructured search problem** is one where we known nothing(or we have no assumption made) about the structure of the search space and search statement $f$. For example determining $f(x_0)$ gives no idea related to the possible value of $f(x_1)$ for $x_0 \neq x_1$. [8]

**A structured search problem** on the other hand is one where we know something about the search space and search statement $f$. For instance searching an alphabetical list gives information about the structure of the search space and this can be exploited to construct efficient algorithms. [8]

Consider a search space of size $N$, Grover's Algorithm in a perfect environment would run in $O(\sqrt{n})$ time steps and $O(\sqrt{n})$ operations which is a quadratic speed up compared to the fastest search over an unordered data set in a classical algorithm which runs in steps of order $O(n)$.

### Problem set up

Consider an $N$-dimensional search space, supplied by a register with $n = \log_2 N$ qubits. Assume that the search space has exactly M solutions with $1 \leq K \leq N$ and consider the

problem of finding an index of the search space that satisfies the search criterion $f$ which is a function that maps search space entries $x$ to 1 or 0 that is $f(x) = 1$ if and only if $x$ satisfies the search criterion $(x = \omega)$ otherwise $f(x) = 0$.
The simplest case is if $K = 1$ that is there exist exactly one $x$ such that $f(x) = 1$. To explicitly define this problem we have the following to do;

1. Label the search items of the search space with integers $0, 1, 2, \cdots, N - 1$ ,
2. label the unknown marked item by $\omega$,
3. let $f$ be an $n$ bit binary function $f : (0, 1)^n \longleftrightarrow (0, 1)$

$$\begin{cases} f(x) = 1 & \text{for } x = \omega, \\ f(x) = 0 & \text{for } x \neq \omega. \end{cases}$$

We are then provided with access to an oracle(subroutine)which is a unitary operator $U_\omega$ that acts as follows:

$$\begin{cases} U_\omega |x\rangle = -|x\rangle & \text{for } x = \omega, \text{ that is,} f(x) = 1, \\ U_\omega |x\rangle = |x\rangle & \text{for } x \neq \omega, \text{ that is, } f(x) = 0. \end{cases}$$

Alternatively the oracle maybe defined with an ancillary qubit system in which case the oracle $U_\omega$ then acts with a conditioned inversion (NOT gate) conditioned by the value of $f(x)$ on the main system. (like in the quantum circuit depicted below).

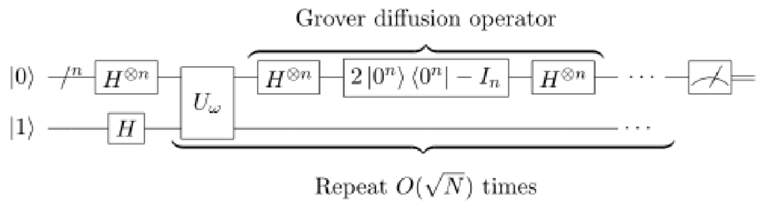

Grover diffusion operator
$|0\rangle$
$n$
$H^{\otimes n}$
$|1\rangle$
$H$
$U_\omega$
$H^{\otimes n}$
$2|0^n\rangle\langle 0^n| - I_n$
$H^{\otimes n}$
Repeat $O(\sqrt{N})$ times

Figure 2.3: Grover"s algorithm , Source : Wikipedia

The oracle then acts as follow;

$$\begin{cases} U_\omega |x\rangle |y\rangle = |x\rangle \neg |y\rangle \ \text{ for } x = \omega \text{ that is , } f(x) = 1, \\ U_\omega |x\rangle |y\rangle = |x\rangle |y\rangle \ \text{ for } x \neq \omega \text{ that is } f(x) = 0, \end{cases}$$

or briefly,

$$U_\omega |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

If the ancillary qubit is prepared in the state

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

then the variants of the oracle are equivalent as illustrated below and this leaves the ancillary system removed from the main system:

$$\begin{aligned} U_\omega\big(\,|x\rangle \otimes |-\rangle\,\big) &= \frac{1}{\sqrt{2}}\left(U_\omega |x\rangle |0\rangle - U_\omega |x\rangle |1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\left(|x\rangle |f(x)\rangle - |x\rangle |1 \oplus f(x)\rangle\right) \\ &= \begin{cases} \frac{1}{\sqrt{2}}\left(|x\rangle |1\rangle - |x\rangle |0\rangle\right) = -|x\rangle \otimes |-\rangle & \text{if } f(x) = 1, \\ \frac{1}{\sqrt{2}}\left(|x\rangle |0\rangle - |x\rangle |1\rangle\right) = |x\rangle \otimes |-\rangle & \text{if } f(x) = 0 \end{cases} \end{aligned}$$

In both setting,the goal remains the same .

**Algorithmic set up**

Suppose $|s\rangle$ denotes the uniform superposition over all states i.e

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle .$$

Then we can define the Grover diffusion operator as:

$$U_s = 2\,|s\rangle\,\langle s| - I$$

The steps of Grover's algorithm are as follows;

1. We initialize the system to the state $|s\rangle$ by application of hadamard transformation on the initial state.
$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle .$$
2. We perform the following "Grover iteration" described below $r(N)$ times.
   (a) we apply the operator $U_\omega$ (i.e the oracle reflection $U_\omega$ to state $|s\rangle$).
   (b) we apply the operator $U_s$ (i.e final reflection $U_s$ that maps $U_\omega$ to $U_\omega U_s$ ) .
3. We then perform the measurement $\sigma$ with the measurement result being the eigenvalue $\lambda_\omega$ having probability close to 1 for $N \gg 1$. From $\lambda_\omega$, $\omega$ may be obtained.

**Description of first iteration**

From our definition of $U_s$ i.e
$$U_s = 2\,|s\rangle\,\langle s| - I,$$
$U_\omega$ can be expressed as;
$$U_\omega = I - 2\,|\omega\rangle\,\langle\omega|$$
To prove that this works, it suffices to check how $U_\omega$ acts on basis states:

$$\begin{aligned} &(I - 2\,|\omega\rangle\,\langle\omega|)\,|\omega\rangle = |\omega\rangle - 2\,|\omega\rangle\,\langle\omega|\,|\omega\rangle = -\,|\omega\rangle = U_\omega\,|\omega\rangle\,, \\ &(I - 2\,|\omega\rangle\,\langle\omega|)\,|x\rangle = |x\rangle - 2\,|\omega\rangle\,\langle\omega|\,|x\rangle = |x\rangle = U_\omega\,|x\rangle \qquad \forall x \neq \omega. \end{aligned}$$

The following computations shows what happens in the first iteration:

$$\langle\omega|\,|s\rangle = \langle s|\,|\omega\rangle = \frac{1}{\sqrt{N}}\langle s|\,|s\rangle = N\frac{1}{\sqrt{N}}\cdot\frac{1}{\sqrt{N}} = 1$$

$$U_\omega |s\rangle = (I - 2|\omega\rangle\langle\omega|)|s\rangle = |s\rangle - 2|\omega\rangle\langle\omega|\,|s\rangle = |s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle$$

$$\begin{aligned}U_s\left(|s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle\right) &= (2|s\rangle\langle s| - I)\left(|s\rangle - \frac{2}{\sqrt{N}}|\omega\rangle\right)\\ &= 2|s\rangle\langle s|\,|s\rangle - |s\rangle - \frac{4}{\sqrt{N}}|s\rangle\langle s|\,|\omega\rangle + \frac{2}{\sqrt{N}}|\omega\rangle\\ &= 2|s\rangle - |s\rangle - \frac{4}{\sqrt{N}}\cdot\frac{1}{\sqrt{N}}|s\rangle + \frac{2}{\sqrt{N}}|\omega\rangle = |s\rangle - \frac{4}{N}|s\rangle + \frac{2}{\sqrt{N}}|\omega\rangle\\ &= \frac{N-4}{N}|s\rangle + \frac{2}{\sqrt{N}}|\omega\rangle\,.\end{aligned}$$

.

It is worth noting that for a special case of $N = 4$ with a single marked state with a single grover iteration We obtain $U_sU_w|s\rangle = |\omega\rangle$.

After application of the operators $U_\omega$ and $U_s$, the square amplitude of the queried element increases from;

$$||\omega\rangle\langle s||^2 = \frac{1}{N}$$

to

$$|\langle\omega|\,U_sU_\omega\,|s\rangle|^2 = \left|\frac{1}{\sqrt{N}}\cdot\frac{N-4}{N} + \frac{2}{\sqrt{N}}\right|^2 = \frac{(3N-4)^2}{N^3} = 9\left(1 - \frac{4}{3N}\right)^2\cdot\frac{1}{N}.$$

**Description of the oracle**

In this description, we will leave the inner workings of the oracle as a black box, but we will explain how the sign is flipped. The oracle in Grover's algorithm is a function f that returns $f(x) = 1$ if $|x\rangle$ is a solution to the search problem and $f(x) = 0$ otherwise. The oracle is a unitary operator operating on two qubits:

$$|x\rangle\,|q\rangle\,, \xrightarrow{U_\omega}, |x\rangle\,|q \oplus f(x)\rangle\,,$$

where $|x\rangle$ is the index qubit and $|q\rangle$ is the oracle qubit. The operation flips the oracle qubit if $f(x) = 1$ and leaves it unchanged otherwise. In Grover's algorithm this is achieved by setting the oracle qubit in the state $(|0\rangle - |1\rangle)/\sqrt{2}$ which is flipped to $(|0\rangle - |1\rangle)/\sqrt{2}$ if $|x\rangle$ is a solution otherwise nothing happens to $(|0\rangle - |1\rangle)/\sqrt{2}$:

$$|x\rangle\,(|0\rangle - |1\rangle)\,/\sqrt{2} \xrightarrow{U_\omega} (-1)^{f(x)}\,|x\rangle\,(|0\rangle - |1\rangle)\,/\sqrt{2}.$$

We regard $|x\rangle$ as flipped thus the oracle qubit is not changed. So by common analogy the oracle qubits are usually not mentioned in the specification of Grover's algorithm and thus the operation of the oracle $U_\omega$ is simply written as

$$|x\rangle \xrightarrow{U_\omega} (-1)^{f(x)}\,|x\rangle.$$

### Algebraic proof

Algebraically when we repeatedly apply $U_sU_\omega$ we can see that by eigenvalue analysis of a matrix, We can write the action of $U_s$ and $U_\omega$ in the space spanned by $|s\rangle, |\omega\rangle$ as:

$$U_s : a|\omega\rangle + b|s\rangle \mapsto [|\omega\rangle, |s\rangle] \begin{bmatrix} -1 & 0 \\ 2/\sqrt{N} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$U_\omega : a|\omega\rangle + b|s\rangle \mapsto [|\omega\rangle, |s\rangle] \begin{bmatrix} -1 & -2/\sqrt{N} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

So in the basis $[|\omega\rangle, |s\rangle]$ (which is neither orthogonal nor a basis of the whole space) the action $U_sU_\omega$ of applying $U_\omega$ followed by $U_s$ is given by the matrix;

$$U_sU_\omega = \begin{bmatrix} -1 & 0 \\ 2/\sqrt{N} & 1 \end{bmatrix} \begin{bmatrix} -1 & -2/\sqrt{N} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2/\sqrt{N} \\ -2/\sqrt{N} & 1-4/N \end{bmatrix}$$

which is in a very convenient Jordan form. If we define;

$$t = \arcsin(1/\sqrt{N}),$$

we have that

$$U_sU_\omega = M \begin{bmatrix} \exp(2it) & 0 \\ 0 & \exp(-2it) \end{bmatrix} M^{-1},$$

where

$$M = \begin{bmatrix} -i & i \\ \exp(it) & \exp(-it) \end{bmatrix}$$

It follows that r-th power of the matrix (corresponding to r iterations) is

$$(U_sU_\omega)^r = M \begin{bmatrix} \exp(2rit) & 0 \\ 0 & \exp(-2rit) \end{bmatrix} M^{-1}$$

From this form, we can use trigonometric identities to compute the probability of observing $\omega$ after r iterations mentioned in the previous section;

$$\left| \begin{bmatrix} |\omega\rangle & |s\rangle \end{bmatrix} (U_sU_\omega)^r \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right|^2 = \sin^2\left((2r+1)t\right).$$

A short calculation now shows that the observation yields the correct answer $\omega$ with error $O(\frac{1}{N})$.

### Geometrical proof

Consider the plane spanned by $|s\rangle$ and $|\omega\rangle$;

$$|s'\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq \omega} |x\rangle.$$

Considering the first iteration acting on the initial ket $|s\rangle$ ,the overlap is;

$$|s'\rangle \langle s| = \sqrt{\frac{N-1}{N}}$$

In geometric terms the angle $\theta/2$ between $|s\rangle$ and $|s'\rangle$ is given by ;

$$\sin\frac{\theta}{2} = \frac{1}{\sqrt{N}}$$

The operator $U_\omega$ is a reflection at the hyper plane orthogonal to $|\omega\rangle$ for vectors in the plane spanned by $|s'\rangle$ and $|\omega\rangle$ i.e. it acts as a reflection across $|s'\rangle$ . The operator $U_s$ is a reflection through $|s\rangle$ therefore the state vector remains in the plane spanned by $|s'\rangle$ and $|\omega\rangle$ after each application of the operators $U_s$ and $U_\omega$ and it is straightforward to check that the operator $U_sU_\omega$ of each Grover iteration step rotates the state vector by an angle of $\theta = 2\arcsin\frac{1}{\sqrt{N}}$. We need to stop when the state vector passes close to $|\omega\rangle$. After this, subsequent iterations rotate the state vector away from $|\omega\rangle$, reducing the probability of obtaining the correct answer. The exact probability of measuring the correct answer is $\sin^2\left(\left(r+\frac{1}{2}\right)\theta\right)$ , where r is the (integer) number of Grover iterations. The earliest time that we get a near-optimal measurement is therefore $r \approx \pi\sqrt{N}/4$.

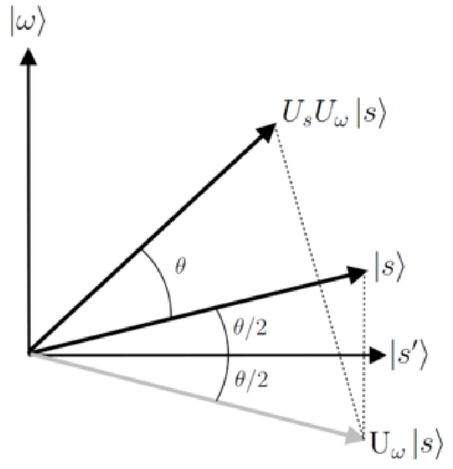


Figure 2.4: Grovers algorithm geometry, Source : Wikipedia

**Extension to space with multiple targets**

If instead of 1 matching entry there are K matching entries with $1 \leq K \leq N$ and $K$ is known. In this case the oracle introduces a reflection in the hyper plane orthogonal to the vector

$$|\beta\rangle = \frac{1}{\sqrt{K}}\sum_{i=1}^{K}|\omega_i\rangle$$

or in other manner

$$\frac{1}{\sqrt{K}}\sum_{f^{-1}(1)}^{K}|x\rangle$$

the equal weighted superposition of the marked computational basis states.The original state

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N-1} |x\rangle$$

can be rewritten as :

$$|s\rangle = \frac{\sqrt{N-K}}{\sqrt{N}} \left( \frac{1}{\sqrt{N-K}} \sum_{x=f^{-1}(0)} |x\rangle \right) + \frac{\sqrt{K}}{\sqrt{N}} \left( \frac{1}{\sqrt{K}} \sum_{x=f^{-1}(1)} |x\rangle \right)$$

or

$$|s\rangle = \frac{\sqrt{N-K}}{\sqrt{N}} \alpha + \frac{\sqrt{K}}{\sqrt{N}} \beta$$

where

$$\alpha = \frac{1}{\sqrt{N-K}} \sum_{x=f^{-1}(0)} |x\rangle \text{ and } \beta = \frac{1}{\sqrt{K}} \sum_{x=f^{-1}(1)} |x\rangle .$$

Following the same algorithm as described in the previous section we obtain the values of $\omega_i$ but the number of iterations is $\frac{\pi}{4}\left(\frac{N}{K}\right)^{1/2}$ instead of $\frac{\pi}{4}N^{1/2}$. For the case when K is unknown, there are several ways to handle the problem For example one could run Grover's algorithm several times with $\frac{\pi}{4}N^{1/2}, \frac{\pi}{4}\left(\frac{N}{2}\right)^{1/2}, \frac{\pi}{4}\left(\frac{N}{4}\right)^{1/2}, \ldots, \frac{\pi}{4}\sqrt{\frac{N}{2^k}}, \ldots$ iterations. For any K, one of the iterations will find a matching entry with a sufficiently high probability. The total number of iterations is at most

$$\pi \frac{N^{1/2}}{4} \left( 1 + \frac{1}{\sqrt{2}} + \frac{1}{2} + \cdots \right) = \pi \frac{\sqrt{N}}{4} \left( 2 + \sqrt{2} \right),$$

which is still $O\left(N^{1/2}\right)$. It can be shown that this can be improved. If the number of marked items is K, where K is unknown, there is an algorithm that finds the solution in $\sqrt{\frac{N}{K}}$ queries. This fact is used in order to solve the collision problem.

# 3. Grover's algorithm for a database of size 8

## 3.1. General Mathematical overview

In this section, we will provide a description in detail of how to perform a single item search in an 8 elements database search using Grover's algorithm that is ($M = 1, N = 2^3 = 8$) as discussed in chapter 2, we will take the following steps.

1. Input:
   (a) n = 3 qubits in the state $|0\rangle$.
   (b) the oracle qubit in state minus that is $\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$.
2. Procedure:
   (a) Apply the Hadamard gates to the 3 input qubits that is ($H^{\otimes 3}$) to the first 3 qubits: $\Rightarrow \frac{1}{\sqrt{8}}\sum_{x=0}^{7}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$.
   (b) Apply the Grover iteration $k \approx \frac{\pi}{4}\sqrt{8}$ times.
   $$[(2|s\rangle\langle s| - I)(I - 2|\omega\rangle\langle\omega|)]^k \frac{1}{\sqrt{8}}\sum_{x=0}^{7}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \approx |\omega\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right].$$
   (c) Measure the first $n$ qubits $\Rightarrow |\omega\rangle$.
3. Output:
   The searched item.

From above its clear that for 3 qubits $k = 2$ which means we need two iterations in order to get the searched element with high probability. In the classical case we would need at least 4 queries to attain the searched element with a probability greater than 1/2.

**Encoding**

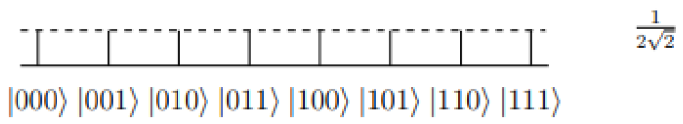
To uniquely encode all 8 entries in the search space, we need 3 qubits and this forms our first register. The second register will contain the oracle qubit required to mark the desired element.

**Initialisation**

We begin with our system initialised in the state $|000\rangle$. We follow by applying the Hadamard transformation to the first register which assign equal probabilities of $\frac{1}{2\sqrt{2}}$ to all states in the system as in equation (3.1).

$$|s\rangle = H^{\otimes 3}|000\rangle = \frac{1}{2\sqrt{2}}\sum_{x=0}^{7}|x\rangle. \tag{3.1}$$

Geometrically this can be represented as;



Supposing that the desired element is the element 4 which is encoded as $|4\rangle = |100\rangle$ in binary encoding, In order to evaluate the state of our system after every computational step we define;

$$|u\rangle = \frac{1}{\sqrt{7}} \sum_{x=0 \wedge x\neq 4}^{7} |x\rangle = \frac{|000\rangle + |001\rangle + |010\rangle + |011\rangle + |101\rangle + |110\rangle + |111\rangle}{\sqrt{7}}.$$

Using $|u\rangle$ we can rewrite $|s\rangle$ as

$$|s\rangle = \frac{\sqrt{7}}{2\sqrt{2}} |u\rangle + \frac{1}{2\sqrt{2}} |100\rangle . \tag{3.2}$$

The circuit component in the figure 3.1 shows the construction to achieve initialisation.

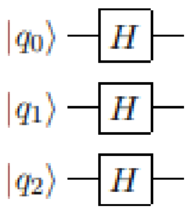


Figure 3.1: Initialisation circuit component

### Grover oracle

In each iteration, we apply the quantum oracle $U_w$ first which performs flip on the desired state followed by the diffusion operator which performs an inversion about the mean. During flip of the desired state, the oracle query will negate the amplitude of the state desired state which is $|100\rangle$, in this case thus we obtain the configuration:

$$\begin{cases} U_w(|100\rangle |-\rangle) = -|100\rangle |-\rangle \\ U_w(|x\rangle |-\rangle) = |x\rangle |-\rangle , \text{ if } x \neq 4 \end{cases}$$

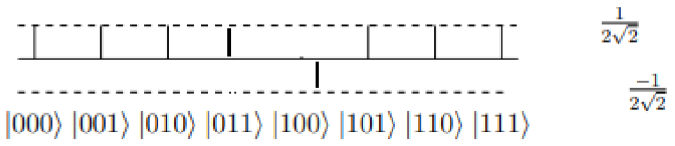
Generally oracle application on $|s\rangle$ yields

$$U_w(|s\rangle |-\rangle) = \frac{|000\rangle + |001\rangle + |010\rangle + |011\rangle + |101\rangle + |110\rangle + |111\rangle - |100\rangle}{2\sqrt{2}} |-\rangle \tag{3.3}$$

Rewriting $U_w(|s\rangle |-\rangle)$ using (3.2) we get

$$U_w(|s\rangle |-\rangle) = \frac{\sqrt{7}}{2\sqrt{2}} |u\rangle - \frac{1}{2\sqrt{2}} |100\rangle$$

Geometrically this is represented as;



The circuit component in the figure 3.2 shows the oracle construction.

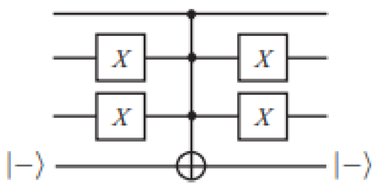


Figure 3.2: Oracle circuit component

Applying Grover's diffusion operator $2|s\rangle\langle s| - I$ increases the probabilities of the states by their difference from the average if the difference is positive and decreases if the difference is negative:

$$\begin{aligned}
&[2|s\rangle\langle s| - I]|x\rangle \\
=&[2|s\rangle\langle s| - I]\left[|s\rangle - \frac{2}{2\sqrt{2}}|100\rangle\right] \\
=&2|s\rangle\langle s||s\rangle - |s\rangle - \frac{2}{\sqrt{2}}|s\rangle\langle s||100\rangle + \frac{1}{\sqrt{2}}|100\rangle
\end{aligned}$$

Since

$$\langle s||s\rangle = 8\frac{1}{2\sqrt{2}}\left[\frac{1}{2\sqrt{2}}\right] = 1.$$

and

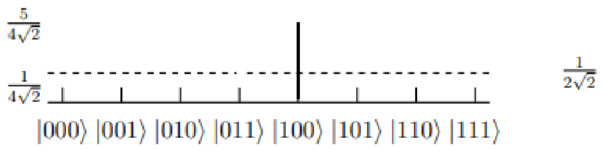
$$\langle s||100\rangle = \langle 100||s\rangle = \frac{1}{2\sqrt{2}}$$

$$\begin{aligned}
&= 2|s\rangle - |s\rangle - \tfrac{2}{\sqrt{2}}\left(\tfrac{1}{2\sqrt{2}}\right)|s\rangle + \tfrac{1}{\sqrt{2}}|100\rangle \\
&= |s\rangle - \tfrac{1}{2}|s\rangle + \tfrac{1}{\sqrt{2}}|100\rangle \\
&= \tfrac{1}{2}|s\rangle + \tfrac{1}{\sqrt{2}}|100\rangle
\end{aligned}$$

Substituting for $|s\rangle$ gives:

$$\begin{aligned}
&= \tfrac{1}{2}\left[\tfrac{1}{2\sqrt{2}}\textstyle\sum_{x=0,x\neq 4}^{7}|x\rangle\right] + \tfrac{1}{\sqrt{2}}|100\rangle \\
&= \tfrac{1}{4\sqrt{2}}\textstyle\sum_{x=0\ \ ,x\neq 4}^{7}|x\rangle + \tfrac{1}{4\sqrt{2}}|100\rangle + \tfrac{1}{\sqrt{2}}|100\rangle \\
&= \tfrac{1}{4\sqrt{2}}\textstyle\sum_{x=0\ \ ,x\neq 4}^{7}|x\rangle + \tfrac{5}{4\sqrt{2}}|100\rangle
\end{aligned}$$

Which geometrically appears as:



The circuit component in the figure 3.3 shows the construction to achieve an inversion about the mean of the amplitudes. It can alternatively be implemented as **HRH** where **H** is the Hadamard transform and **R** a phase shift transform.

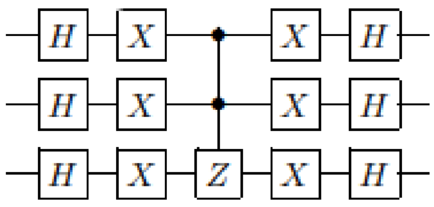


Figure 3.3: Diffusion operator circuit component

This completes the first iteration. We Repeat the Grover iteration process but with the new input state having;

$$|x\rangle = \frac{1}{4\sqrt{2}}|000\rangle + \frac{1}{4\sqrt{2}}|001\rangle + \frac{1}{4\sqrt{2}}|010\rangle + \frac{1}{4\sqrt{2}}|011\rangle + \frac{5}{4\sqrt{2}}|100\rangle + \ldots + \frac{1}{4\sqrt{2}}|111\rangle$$

Applying the oracle we get;

$$|x\rangle = \frac{1}{4\sqrt{2}}|000\rangle + \frac{1}{4\sqrt{2}}|001\rangle + \frac{1}{4\sqrt{2}}|010\rangle + \frac{1}{4\sqrt{2}}|011\rangle - \frac{5}{4\sqrt{2}}|100\rangle + \ldots + \frac{1}{4\sqrt{2}}|111\rangle$$

$$\begin{aligned} &= \tfrac{1}{4\sqrt{2}}\textstyle\sum_{x=0\ ,x\neq 4}^{6}|x\rangle - \tfrac{5}{4\sqrt{2}}|100\rangle \\ &= \tfrac{1}{4\sqrt{2}}\textstyle\sum_{x=0\ ,x\neq 4}^{7}|x\rangle - \tfrac{6}{4\sqrt{2}}|100\rangle \\ &= \tfrac{1}{2}|s\rangle - \tfrac{3}{2\sqrt{2}}|100\rangle \end{aligned}$$

Applying the diffusion transform we get;

$$\begin{aligned} &[2|s\rangle\langle s| - I]\left[\tfrac{1}{2}|s\rangle - \tfrac{3}{2\sqrt{2}}|100\rangle\right] \\ &= 2\left(\tfrac{1}{2}\right)|s\rangle\langle s||s\rangle - \tfrac{1}{2}|s\rangle - 2\left(\tfrac{3}{2\sqrt{2}}\right)|s\rangle\langle s||100\rangle + \tfrac{3}{2\sqrt{2}}|100\rangle \end{aligned}$$

After the oracle query and after applying the diffusion transform:

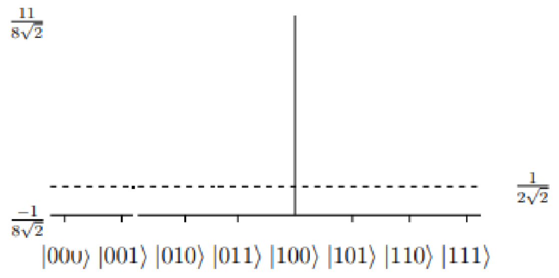
$$\begin{aligned} &= |s\rangle - \tfrac{1}{2}|s\rangle - \tfrac{3}{\sqrt{2}}\left(\tfrac{1}{2\sqrt{2}}\right)|s\rangle + \tfrac{3}{2\sqrt{2}}|100\rangle \\ &= -\tfrac{1}{4}|s\rangle + \tfrac{3}{2\sqrt{2}}|100\rangle \\ &= -\tfrac{1}{4}\left[\tfrac{1}{2\sqrt{2}}\textstyle\sum_{x=0\ ,x\neq 4}^{7}|x\rangle + \tfrac{1}{2\sqrt{2}}|100\rangle\right] + \tfrac{3}{2\sqrt{2}}|100\rangle \\ &= -\tfrac{1}{8\sqrt{2}}\textstyle\sum_{x=0\ ,x\neq 4}^{7}|x\rangle + \tfrac{11}{8\sqrt{2}}|100\rangle \end{aligned}$$

Or in the expanded notation:

$$|x\rangle = -\frac{1}{8\sqrt{2}}|000\rangle - \frac{1}{8\sqrt{2}}|001\rangle - \frac{1}{8\sqrt{2}}|010\rangle - \frac{1}{8\sqrt{2}}|011\rangle + \frac{11}{8\sqrt{2}}|100\rangle - \ldots - \frac{1}{8\sqrt{2}}|111\rangle$$

## Measurement

Finally the qubits are measured using the gate component construction in figure 3.4.
Geometrically we get the solution with probability of success $\frac{11}{8\sqrt{2}}$;



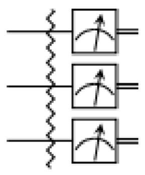


Figure 3.4: Circuit component for measurement

Observing the system, the probability that the correct solution $|100\rangle$ will be measured is $\left|\frac{11}{8\sqrt{2}}\right|^2 = 121/128 \approx 94.5\%$. The probability of finding an incorrect state is $\left|\frac{-\sqrt{7}}{8\sqrt{2}}\right|^2 = 7/128 \approx 5.5\%$. Its worthwhile to mention that as the input size increases the error decreases further.

To implement Grover's algorithm on a physical quantum computer the following complete circuit architecture is used.

## 3.2. Explicit Mathematical overview

Below we make a further more explicit step by step mathematical discussion of what goes on inside each algorithmic step involved in our construction of the Grover circuit as discussed in the previous section section. To lay the ground for our for our main problem of discussion (database of size 8) , we chip in a small discussion using a basic search space (database of size 4). The choice of chronology of discussion is to highlight the computational complexity faced as the size of the search space increases. The corresponding search item for $N = 4$ and $N = 8$ are $|11\rangle$ and $|111\rangle$ respectively.

**Case** $N = 4$

In the search space with $N = 4$ with search element $|11\rangle$ the following mathematical treatment underlies the procedure followed.

#### Initialisation

Applying Hadamard gates to the two input qubits yields an equal superposition as shown below;

$$|\Psi\rangle_1 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle).$$

#### Oracle

Applying the oracle, the amplitude of the marked state is negated;

$$\begin{aligned}|\psi\rangle_2 =& \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle)(|0\rangle - |1\rangle) + |11\rangle (|1\rangle - |0\rangle) \\ =& \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)(|0\rangle - |1\rangle).\end{aligned}$$

#### Diffusion operator

Applying Hadamard gates to the first 2 qubits we get;

$$\begin{aligned}|\Psi_3\rangle =& \frac{1}{4}[(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)] + [(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)] \\ & + [(|0\rangle - |1\rangle)(|0\rangle + |1\rangle)] + [(|1\rangle - |0\rangle)(|1\rangle - |0\rangle)] \\ =& \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle).\end{aligned}$$

Applying an X gate to $|\Psi_3\rangle$ we have;

$$|\Psi_4\rangle = \frac{1}{2}(|11\rangle + |10\rangle + |01\rangle - |00\rangle).$$

Applying Hadamard to the second qubit we get;

$$\begin{aligned}|\Psi_5\rangle =& \frac{1}{2\sqrt{2}}[|1\rangle (|0\rangle - |1\rangle) + |1\rangle (|0\rangle + |1\rangle) \\ & + |0\rangle (|0\rangle - |1\rangle) - |0\rangle (|0\rangle + |1\rangle) \\ =& \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle).\end{aligned}$$

Application of the CNOT gate gives;

$$|\Psi_6\rangle = \frac{1}{\sqrt{2}}(|11\rangle - |01\rangle).$$

Applying a Hadamard gate to the second qubit we get;

$$\begin{aligned}|\Psi_7\rangle &= \frac{1}{2}[|1\rangle(|0\rangle - |1\rangle) - |0\rangle(|0\rangle - |1\rangle)]\\ &= \frac{1}{2}(|10\rangle - |11\rangle - |00\rangle + |01\rangle).\end{aligned}$$

Applying the X gates we get;

$$|\Psi_8\rangle = \frac{1}{2}(|01\rangle - |00\rangle - |11\rangle + |10\rangle).$$

Finally after applying Hadamard gates to both qubits we obtain the searched item as illustrated below;

$$\begin{aligned}|\Psi_9\rangle =&\frac{1}{4}[(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) - (|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\\ &- (|0\rangle - |1\rangle)(|0\rangle - |1\rangle) + (|0\rangle - |1\rangle)(|0\rangle + |1\rangle)]\\ =&\frac{1}{4}(4|11\rangle) = |11\rangle.\end{aligned}$$

The searched item $|11\rangle$ is obtained with probability 1 in one iteration.

**Case** $N = 8$

In the search space with $N = 8$ with search element $|111\rangle$ the following mathematical treatment underlies the procedure followed.

**Initialisation**

Applying Hadamard gates to the three input qubits yields an equal superposition as shown below;

$$\begin{aligned}|\Psi_1\rangle =&\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\\ =&\frac{1}{2\sqrt{2}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)\end{aligned}$$

**Oracle**

Applying the oracle, the amplitude of the marked state is negated;

$$\begin{aligned}|\psi_2\rangle =&\frac{1}{4}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle)(|0\rangle - |1\rangle) + |111\rangle(|1\rangle - |0\rangle)\\ =&\frac{1}{4}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle - |111\rangle)(|0\rangle - |1\rangle)\end{aligned}$$

**Diffusion operator**

Applying Hadamard gates to the first 3 qubits gives;

$$\begin{aligned}|\Psi_3\rangle =& \frac{1}{8}[(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)] + [(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle)] \\ &+[(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) \otimes (|0\rangle + |1\rangle)] + [(|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle)] \\ &+[(|0\rangle - |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)] + [(|0\rangle - |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle)] \\ &+[(|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle) \otimes (|0\rangle + |1\rangle)] + [(|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle)] \\ =& \frac{6}{8}|000\rangle + \frac{2}{8}|001\rangle + \frac{2}{8}|010\rangle - \frac{2}{8}|011\rangle + \frac{2}{8}|100\rangle - \frac{2}{8}|101\rangle - \frac{2}{8}|110\rangle + \frac{2}{8}|111\rangle\end{aligned}$$

After applying an X gate to $|\Psi_3\rangle$ we have;

$$|\Psi_4\rangle = \frac{6}{8}|111\rangle + \frac{2}{8}|110\rangle + \frac{2}{8}|101\rangle - \frac{2}{8}|100\rangle + \frac{2}{8}|011\rangle - \frac{2}{8}|010\rangle - \frac{2}{8}|001\rangle + \frac{2}{8}|000\rangle$$

Applying Hadamard to the second qubit;

$$\begin{aligned}|\Psi_5\rangle =& \frac{1}{8\sqrt{2}}[6|11\rangle(|0\rangle - |1\rangle) + 2|11\rangle(|0\rangle + |1\rangle) + 2|10\rangle(|0\rangle - |1\rangle) \\ &- 2|10\rangle(|0\rangle + |1\rangle) + 2|01\rangle(|0\rangle - |1\rangle) - 2|01\rangle(|0\rangle + |1\rangle) \\ &- 2|00\rangle(|0\rangle - |1\rangle) + 2|00\rangle(|0\rangle + |1\rangle)] \\ =& \frac{1}{8\sqrt{2}}[4|001\rangle - 4|011\rangle - 4|101\rangle - 4|111\rangle + 8|110\rangle]\end{aligned}$$

Application of the CCNOT gate gives;

$$|\Psi_6\rangle = \frac{1}{8\sqrt{2}}[4|001\rangle - 4|011\rangle - 4|101\rangle - 4|110\rangle + 8|111\rangle]$$

Applying a Hadamard gate to the last qubit gives;

$$\begin{aligned}|\Psi_7\rangle =& \frac{1}{16}[4|00\rangle(|0\rangle - |1\rangle) - 4|01\rangle(|0\rangle - |1\rangle) - 4|10\rangle(|0\rangle - |1\rangle) \\ &- 4|11\rangle(|0\rangle + |1\rangle) + 8|11\rangle(|0\rangle - |1\rangle)] \\ =& \frac{1}{4}[|000\rangle - |001\rangle - |010\rangle + |011\rangle - |100\rangle + |101\rangle + |110\rangle - 3|111\rangle]\end{aligned}$$

Applying the X gates we get;

$$|\Psi_8\rangle = \frac{1}{4}[|111\rangle - |110\rangle - |101\rangle + |100\rangle - |011\rangle + |010\rangle + |001\rangle - 3|000\rangle]$$

Applying Hadamard gates to the three qubits yields;

$$|\Psi_9\rangle = \frac{1}{4\sqrt{2}}[|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + 5|111\rangle]$$

Further going through the same process for the second iteration as discussed above, we obtain the state $|111\rangle$ with probability $\left|\frac{11}{8\sqrt{2}}\right|^2 = 121/128 \approx 94.5\%$.

# 4. Construction and Implementation of Grover's Algorithm

In this section, we begin with the discussion of the construction and implementation of our Grover's algorithm using QISKit simulation environment and we conclude with a discussion of results. QISkit is an open-source quantum simulation software developed created by IBM. It consists of the quantum-lab which is written in Python and offers a variety of tools used to create and manipulate quantum programs which then can either be simulated on a local device or run on the IBMQ backend.

We implemented and executed our algorithm using QISKit simulation environment and to simulate our Quantum circuit, we made use of **QASM Simulator** and the **Statevector Simulator** Aer backends. We executed our implementation of the algorithm 2000 times in each run.This was defined by the number of execution shots and the results obtained were stored in the dictionary called results.

## 4.1. Circuit Construction

### N-Toffoli gates

Since the N-Toffoli gate as a main component of our algorithm is not available through the QISKit environment and we had to build it. To construct it, we made use Ancilla bits as illustrated in figure **??**. Ancilla bits are extra bits which are not involved in the logical operation being performed that give circuit constructions "room to move". In addition to making constructions possible in the first place, ancilla bits can allow for simpler or more efficient constructions. The type of Ancilla bits used were borrowed bits where the bits can be in any state beforehand and must be restored to that same state afterwards. The advantage of using Borrowed Bits is that they can be reused and hence reduce the number of Qubits required. The downside of our construction is that we increase the number of qubits and gates gates required thus increasing circuit depth and width.

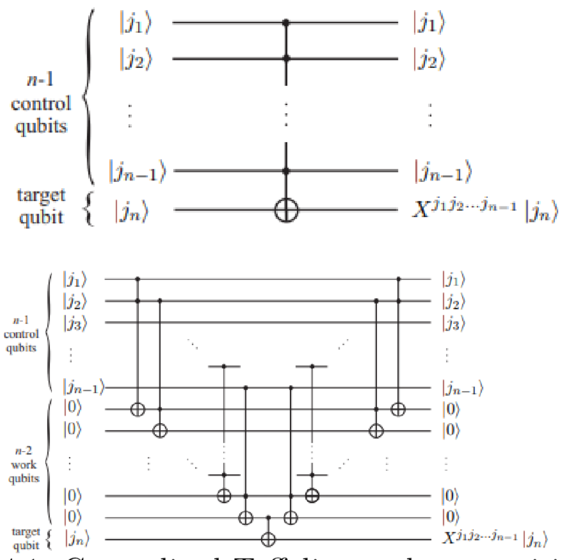


Figure 4.1: Generalized Toffoli gate decomposition [28]

## The implemented stages of the algorithm

### Initialization

The circuit diagram component below was used to put the six input qubit states in the state of equal superposition with amplitudes of $\frac{1}{8}$.

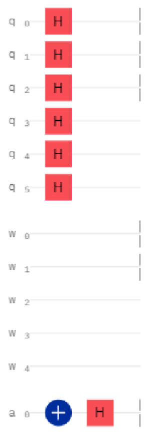


Figure 4.2: Initialisation circuit component for N = 64

### Oracle

Below is the oracle construction that was used to negate the amplitude of the marked state $|111111\rangle$.

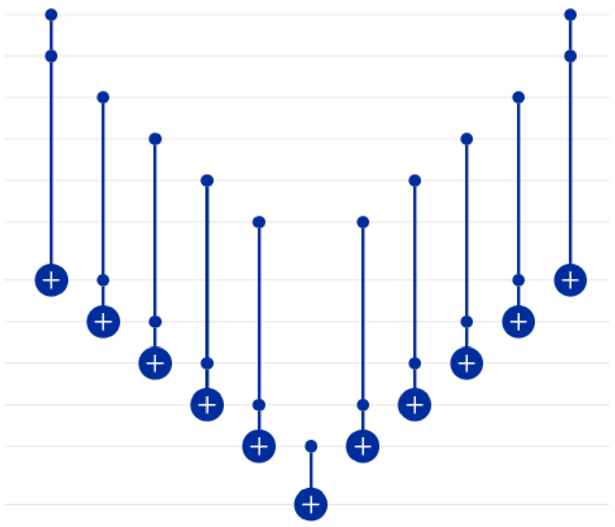


Figure 4.3: Oracle construction for N =64

### Amplification

The construction below describes the amplification stage in which an inversion about the mean of the amplitudes is made. It was implemented using borrowed ancilla bits and Z-gate as shown below.

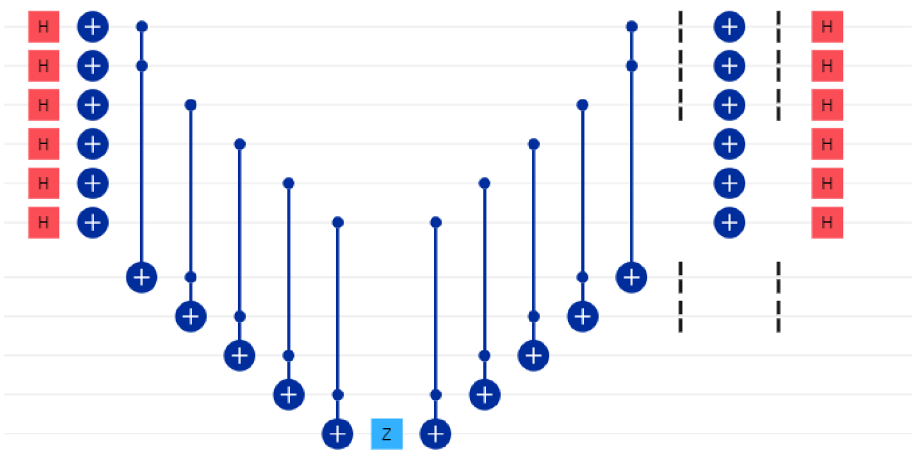


Figure 4.4: Grover diffusion operator construction for N =64

**Measurement**

Finally the qubits were measured and below a thorough discussion of the obtained results is made.

## 4.2. Discussion of Results

The execution was performed for a single item search using search spaces of 8 elements, 16 elements , 32 elements and 64 elements as represented in the figures below. The search items were $|111\rangle$ , $|1111\rangle$ , $|11111\rangle$ and $|111111\rangle$ respectively.A total number of 2000 shots was made for each execution. The execution results correspond to one iteration of Grover's iterate in each search space instance.

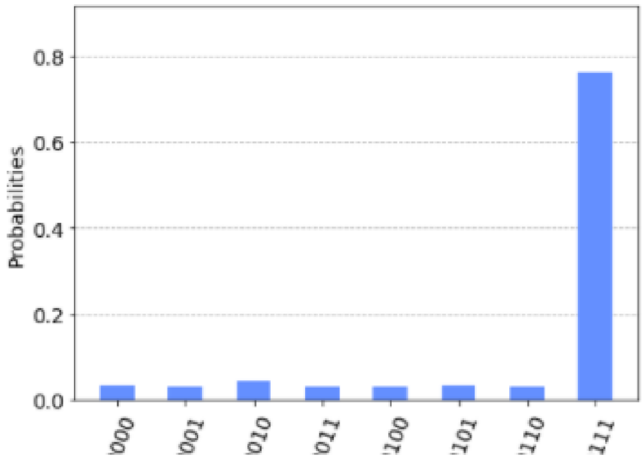


Figure 4.5: QASM simulation results for 8 element search space

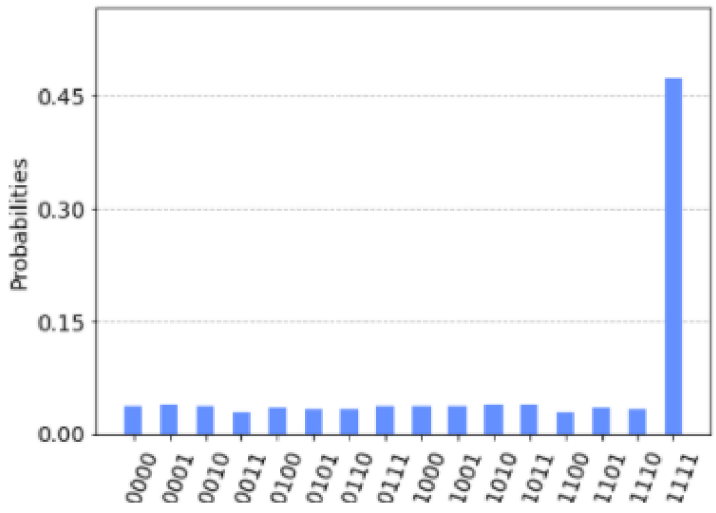


Figure 4.6: QASM simulation results for 16 element search space

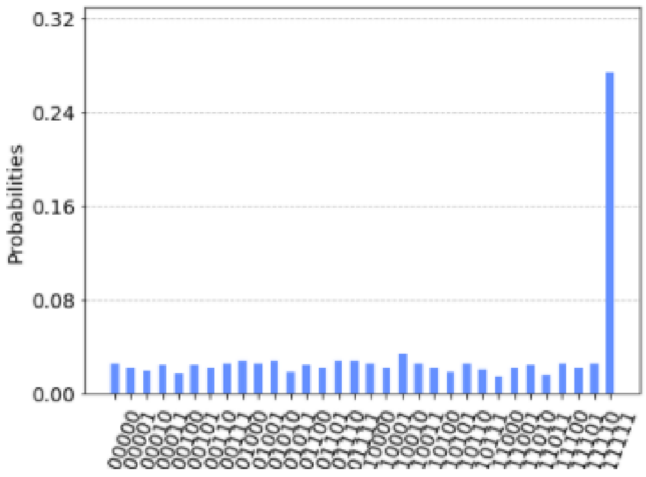


Figure 4.7: QASM simulation results for 32 element search space

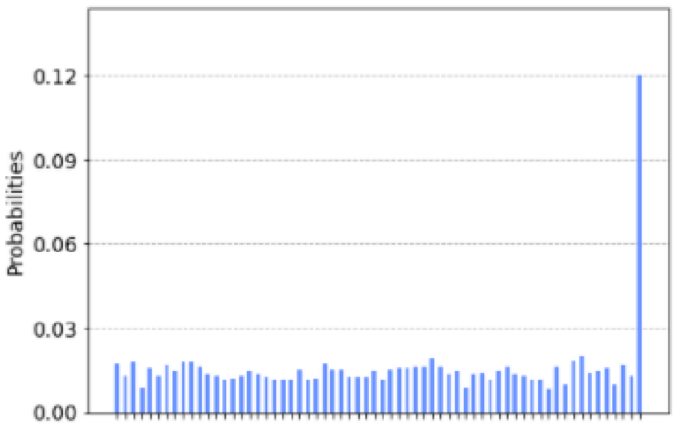


Figure 4.8: QASM simulation results for 64 element search space

From the results above it clear that in just one execution one can get the desired item, or marked item with probability greater than 0.5.

Table 4.1: Execution results

| **SUMMARY OF EXECUTION RESULTS** | | | | |
|---|---|---|---|---|
| Measurement parameters. | N=8 | N=16 | N=32 | N=64 |
| Probability of having state $\|1\rangle$ in a disentangled state system. | 0.8750 | 0.7187 | 0.6172 | 0.5606 |
| Purity of reduced state. | 0.9063 | 0.8770 | 0.9103 | 0.9468 |
| Probability of marked state based on counts. | 0.7840 | 0.4820 | 0.2640 | 0.1490 |
| Amplitude of marked state. | 0.8839 | 0.6875 | 0.5082 | 0.3672 |
| Number of valid counts per 2000 shot. | 1568 | 948 | 528 | 266 |

**Discussion of the differences in execution results**

From the data, the probability of having state $|1\rangle$ for all states in all search spaces kept on reducing because the level of entanglement goes on increasing with the increase in search space size. The choice to of consideration of this parameter is because all the search items in all cases were strings of 1's.

From the data, its observable that the purity level of reduced states increases with increase in the number qubits. This implies that with larger qubit string sizes, the likelihood of getting the searched item in its pure state increases.

It is also observable from the data that the probability of the marked state and its amplitude based on counts reduces with increase in search space size. This spins from increase in sample size and the confusion (noise) resulting from the high error rates of current quantum devices (NISQ) coupled with the data being representative of one iteration as opposed to 3,4,6 and 8 necessary iterations of the Grover iterate that are required for the various search space respectively.

From the data it is expected that for the 8, 16, 32 and 64 search space sizes, in every 1568, 948, 528 and 266 counts made respectively out of the 2000 made the marked state is obtained.

The number of valid counts of the in every 2000 shots made also reduces with increase in qubit length still due to the high error rates of current quantum devices and the data representing one iteration as opposed to 3,4,6 and 8 necessary iterations of the Grover iterate that are required for the various search space respectively.

### Discussion of differences of execution results and theoretically obtained optimal results

Table 4.2: Comparison of execution and theoretical results

| Result type | Search space size | Probability of success |
|---|---|---|
| Theoretical results | 8 | 0.9453125 |
| | 16 | 0.9613190 |
| | 32 | 0.9991823 |
| | 64 | 0.9965857 |
| Simulation results | 8 | 0.875 |
| | 16 | 0.71875 |
| | 32 | 0.6172 |
| | 64 | 0.5606 |

From the data above, it clear that there is high discrepancy between the the theoretical result and the ones achievable via simulation mainly because the data represents one iteration as opposed to 3,4,6 and 8 necessary iterations of the Grover iterate that are required and secondly due high error rates of current quantum devices.

### Computational complexity

Table 4.3: Comparison of computational complexity

| Computational type | Search space size | Number of iterations |
|---|---|---|
| Classical | 8 | 8 |
| | 16 | 16 |
| | 32 | 32 |
| | 64 | 64 |
| Quantum | 8 | 3 |
| | 16 | 4 |
| | 32 | 6 |
| | 64 | 8 |

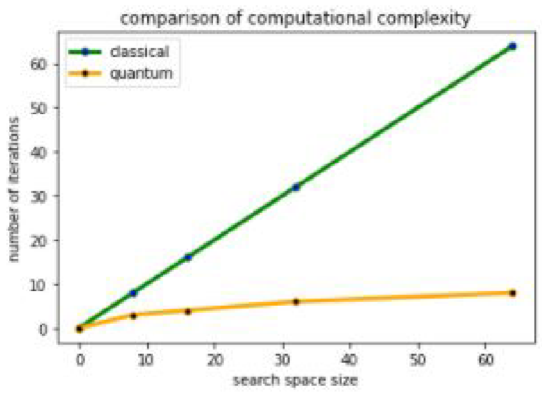


Figure 4.9: Grover computational complexity

From the data and plot above it is clear that quantum search using Grover's provides quadratic speed up over classical search which provides linear search. It is also easily seen that the power of quantum search using Grover's algorithm on small search spaces is not fully exploitable.

# 5. Recommendations and Conclusion

In this section we make discussions about the limitations to effective operation of our algorithm, possible recommendations for better performance, further explanations on the vast discrepancy in results is still made and We conclude the section with recommendations for further possible studies that can be done on the work and on the quantum studies as a whole.

## Limitations

In performing our simulations, we were faced with a number of challenges and among included;

1. The threat that having multiple iterations would increase decoherence and gate error which would reduce the occurrences of measuring the marked state.
2. Secondly the fact that the N-Toffoli gate construction is not available in the provisions made available by the QISKit environment prompted introduction of more qubits inform of work bits which constituted increase in circuit depth which automatically affected the performance since each gate addition comes with its error addition.
3. Performing single iterations efficiently brought out the desired effect of the study but in the long run it brought about vast increase in discrepancy of the results with increase in search space.

A combination of the above issues among others as suggested above constituted a big threat to performance of our simulations and thus limited us to relatively small sample size.

## Suggestions for improvements

To improve the quality of our simulation work in the QISKit environment, the following could have been adopted as solutions;

1. We could have reduced gate error by application of quantum error correction but because it comes with a cost of requiring more qubits and gates that further threatens shrinking our executable problem sizes, it was not adopted.
2. We could also have improved performance by using more efficient oracle and diffuser constructions that don't add more work bits. This would have eliminated redundant gates thus reducing circuit depth and width but it was not the case because of inability to achieve the construction with our intended circuit depth requirement for easy simulatability.
3. Performing the search with the right number of iteration would be the best option as it maps exactly to the desired outcome but the threat of decoherence and gate error with larger circuit sizes threatened the occurrence of the correct solution to our problem thus it was not adopted.

## Suggestions for further studies

Given the scope of our study and results achieved,the following ideas can be taken on for further study;

1. A study on how to make possible gate combinations to reduce gate complexity or gate error which would provide means to improve performance of results.
2. A more analytical study using more measures of central tendency and measures of variation can be done to shade more light on the practical impact of Grover's algorithm in the quantum era.
3. A more performance related study using Grover's algorithm carried out with the required amount of iterations with efficient circuit construction the is susceptible to less decoherence and gate errors can also be done.

These among many possible related studies can be undertaken.

## Conclusions

While our study was more involved in a more rigorous theoretical and mathematical overview of Grover's algorithm and our implementation inclined to that end, it is worth noting that the algorithm doesn't fall short of challenges as pointed out in the above section and finding solutions to the problems would serve to enhance the impact of the algorithm for further future use in quantum applications related to solving NP problems. I hope that the mathematical discussion availed in this thesis report would help create a more intuitive understanding of the Grover's algorithm to those still having questions about its operation.

# Bibliography

[1] Scott Aaronson, G Kuperberg, C Granade, and V Russo. Complexity zoo. *URL http://www. complexityzoo. com*, 2005.

[2] Marco Affronte, Filippo Troiani, Alberto Ghirri, Stefano Carretta, Paolo Santini, Valdis Corradini, Raffael Schuecker, Chris Muryn, Grigore Timco, and Richard E Winpenny. Molecular routes for spin cluster qubits. *Dalton Transactions*, -(23):2810–2817, 2006.

[3] Vladimir M Akulin. Entangled states of composite quantum systems. In *Dynamics of Complex Quantum Systems*, pages 477–527. Springer, 2014.

[4] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.

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

[6] E Bernstein and U Vazirani. Proceedings of the 25th annual acm symposium on theory of computing. *ACM, New York*, 11, 1993.

[7] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997.

[8] Eva Borbely. Grover search algorithm. *arXiv preprint arXiv:0705.4171*, 2007.

[9] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum search. In *Proceedings of the Workshop on Physics of Computation: PhysComp'96*, pages 36–43, 1996.

[10] Gilles Brassard. Searching a quantum phone book. *Science*, 275(5300):627–628, 1997.

[11] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[12] David Bulger, William P Baritompa, and Graham R Wood. Implementing pure adaptive search with grover's quantum algorithm. *Journal of optimization theory and applications*, 116(3):517–529, 2003.

[13] Juan I Cirac and Peter Zoller. Quantum computations with cold trapped ions. *Physical review letters*, 74(20):4091, 1995.

[14] Franklin de Lima Marquezino, Renato Portugal, and Carlile Lavor. *A Primer on Quantum Computing*. Springer, 2019.

[15] D Deutsch. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society A*, 435:563–574, 1991.

[16] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[17] Jose P Dumas, Kapil Soni, and Akhtar Rasool. An introduction to quantum search algorithm and its implementation. In *Data Management, Analytics and Innovation*, pages 19–31. Springer, 2019.

[18] Christoph Dürr, Mark Heiligman, Peter Hoyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006.

[19] Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

[20] Richard P Feynman. Simulating physics with computers. *Int. J. Theor. Phys*, 21(6/7), 1982.

[21] Caroline Figgatt, Dmitri Maslov, KA Landsman, Norbert Matthias Linke, Shantanu Debnath, and C Monroe. Complete 3-qubit grover search on a programmable quantum computer. *Nature communications*, 8(1):1–9, 2017.

[22] Austin Gilliam, Stefan Woerner, and Constantin Gonciulea. Grover adaptive search for constrained polynomial binary optimization. *Quantum*, 5:428, 2021.

[23] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[24] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325, 1997.

[25] Lov K Grover. Fixed-point quantum search. *Physical Review Letters*, 95(15):150501, 2005.

[26] Jack D Hidary. *Quantum Computing: An Applied Approach*. Springer, 2019.

[27] Dan Kenigsberg and Eli Biham. *Grover's Quantum Search Algorithm and Mixed States*. PhD thesis, Computer Science Department, Technion, 2001.

[28] Carlile Lavor, LRU Manssur, and Renato Portugal. Grover's algorithm: quantum database search. *arXiv preprint quant-ph/0301079*, 2003.

[29] Seth Lloyd. A potentially realizable quantum computer. *Science*, 261(5128):1569–1571, 1993.

[30] Yuri Manin. Computable and uncomputable. *Sovetskoye Radio, Moscow*, 128, 1980.

[31] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[32] Juanjo Rué and SEBASTIAN XAMBÓ. Mathematical essentials of quantum computing. In *Preprint. Seminar on Quantum Processing.*

[33] Simanraj Sadana. Grover's search algorithm for $n$ qubits with optimal number of iterations. *arXiv preprint arXiv:2011.04051*, 2020.

[34] Daniel R Simon. On the power of quantum computation. *SIAM journal on computing*, 26(5):1474–1483, 1997.

[35] Akanksha Singhal and Arko Chatterjee. Grover's algorithm. 2018.

[36] Philip Strömberg and Vera Blomkvist Karlsson. 4-qubit grover's algorithm implemented for the ibmqx5 architecture, 2018.

[37] Lieven MK Vandersypen, Matthias Steffen, Gregory Breyta, Costantino S Yannoni, Mark H Sherwood, and Isaac L Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–887, 2001.

[38] Edward TH Wu. Quantum entanglement and hidden variables interpreted by yangton and yington theory. *IOSR Journal of Applied Physics (IOSR-JAP)*, 12(2), 2020.

[39] Theodore J Yoder, Guang Hao Low, and Isaac L Chuang. Fixed-point quantum search with an optimal number of queries. *Physical review letters*, 113(21):210501, 2014.

# 6. List of abbreviations and symbols used

| | |
|---|---|
| $\mathbb{C}$ | Set of Complex numbers. |
| $\mathbb{F}$ | A Field. |
| $\mathbb{R}$ | Set of real numbers. |
| $\mathbb{C}^n$ | Complex vector field in n–dimensions space. |
| $\mathbb{C}^{2^n}$ | Complex vector field in $2^n$–dimensions. |
| $A_{m,n}$ | Matrix of dimension $m \times n$. |
| $A^T_{m,n}$ | Transpose of $A_{m,n}$. |
| $A^\dagger_{m,n}$ | Conjugate transpose of $A_{m,n}$. |
| $\bar{A}_{m,n}$ | Conjugate of matrix $A_{m,n}$. |
| $a_{ij}$ | i–jth component of $A_{m,n}$. |
| $U_\omega$ | Oracle function. |
| $U_s$ | Grover diffusion operator. |
| $\otimes$ | Tensor product operator. |
| $N_{r,s}$ | Controlled Negation of sth bit with rth contol. |
| $U^1$ | Unitary operator of determinant 1. |
| $SU^1$ | General group of Unitary operators of determinant 1. |
| $H^{\otimes n}$ | Hadamard transform on n Qubits . |
| NISQ | Noisy Intermediate-Scale Quantum. |
| QFT | Quantum Fourier Transform. |
| NMR | Nuclear Magnetic Resonance. |
| P | Polynomial time. |
| NP | Non-deterministic polynomial time. |
| PSPACE | Polynomial space. |
| BPP | Bounded-error probabilistic polynomial time. |
| BQP | Bounded-error quantum polynomial time. |
| EQP | Exact quantum polynomial time. |

| | |
|---|---|
| QMA | Quantum Merlin-Arthur. |
| CNOT | Controlled NOT gate. |
| CCNOT | Controlled Controlled NOT gate. |
| CSWAP | Controlled SWAP gate. |
| Q-Computer | Quantum Computer. |
| Q-Computation | Quantum Computation. |
| Q-Measurement | Quantum Measurement. |
| Q-Procedure | Quantum Procedure. |
| Q-Algorithm | Quantum Algorithm. |
| Q-Memory | Quantum Memory. |
| Q-Vector | Quantum Vector. |
| Q-bit rotation | Quantum bit rotation. |

# List of Tables

# List of Figures

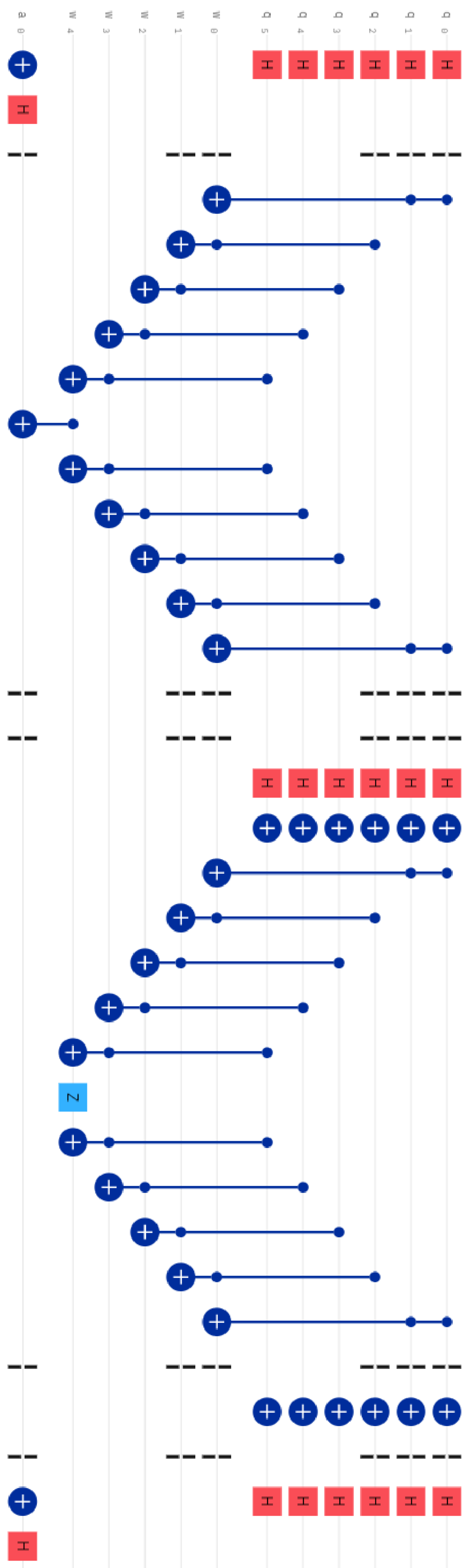
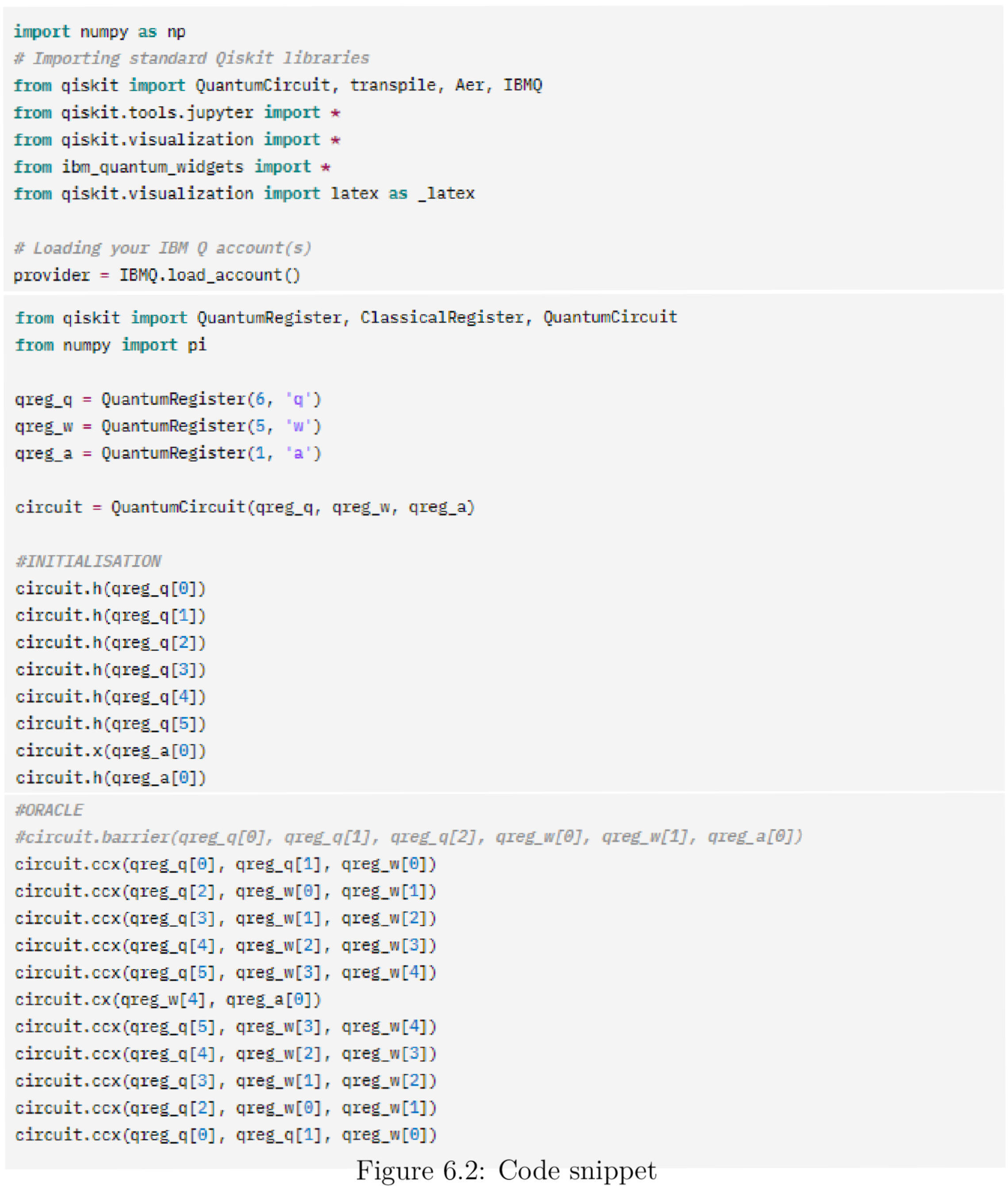


Figure 6.1: Grover circuit for N = 64

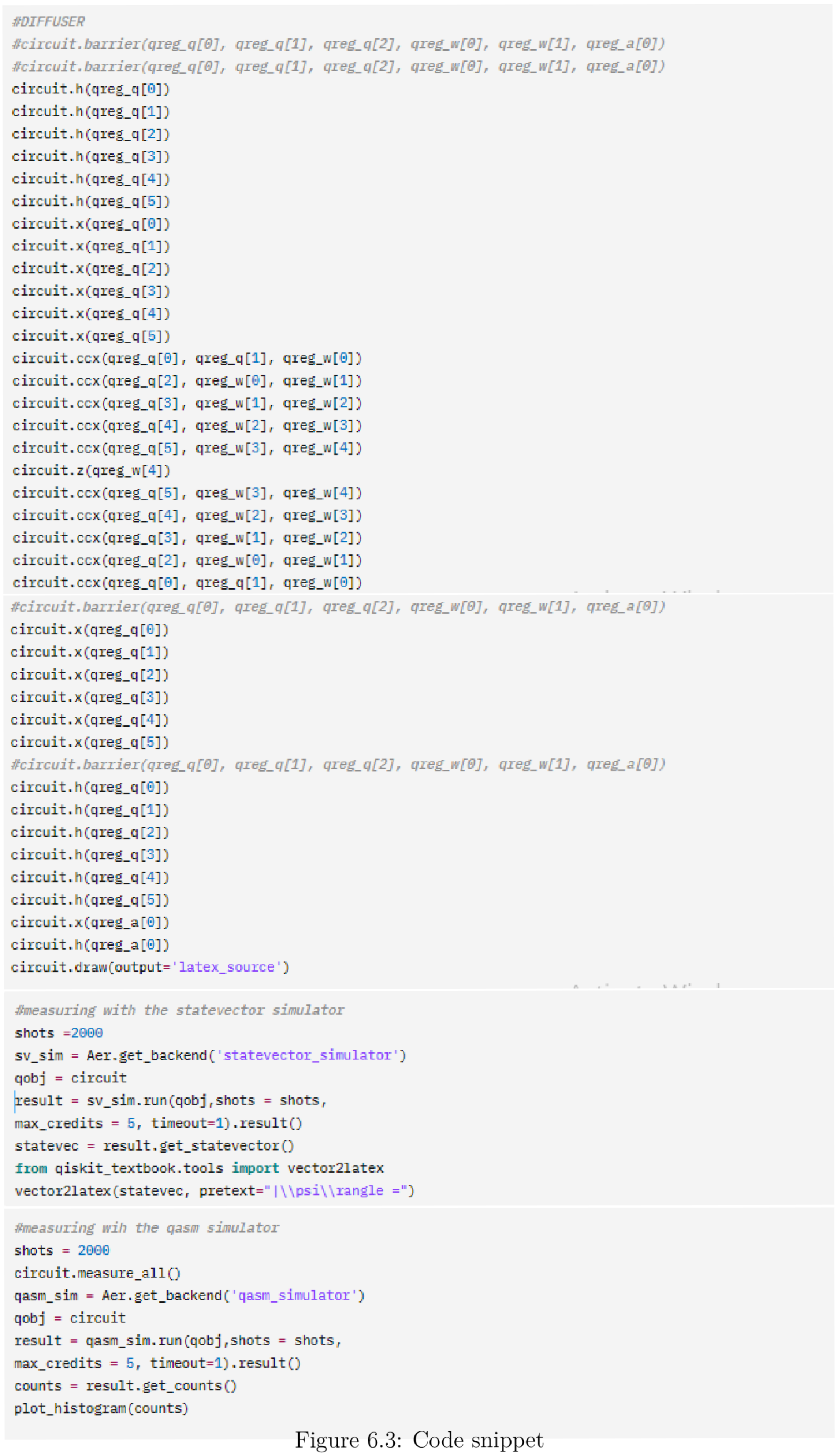


```
import numpy as np
# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, transpile, Aer, IBMQ
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *
from qiskit.visualization import latex as _latex

# Loading your IBM Q account(s)
provider = IBMQ.load_account()
```

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(6, 'q')
qreg_w = QuantumRegister(5, 'w')
qreg_a = QuantumRegister(1, 'a')

circuit = QuantumCircuit(qreg_q, qreg_w, qreg_a)

#INITIALISATION
circuit.h(qreg_q[0])
circuit.h(qreg_q[1])
circuit.h(qreg_q[2])
circuit.h(qreg_q[3])
circuit.h(qreg_q[4])
circuit.h(qreg_q[5])
circuit.x(qreg_a[0])
circuit.h(qreg_a[0])
```

```
#ORACLE
#circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_w[0], qreg_w[1], qreg_a[0])
circuit.ccx(qreg_q[0], qreg_q[1], qreg_w[0])
circuit.ccx(qreg_q[2], qreg_w[0], qreg_w[1])
circuit.ccx(qreg_q[3], qreg_w[1], qreg_w[2])
circuit.ccx(qreg_q[4], qreg_w[2], qreg_w[3])
circuit.ccx(qreg_q[5], qreg_w[3], qreg_w[4])
circuit.cx(qreg_w[4], qreg_a[0])
circuit.ccx(qreg_q[5], qreg_w[3], qreg_w[4])
circuit.ccx(qreg_q[4], qreg_w[2], qreg_w[3])
circuit.ccx(qreg_q[3], qreg_w[1], qreg_w[2])
circuit.ccx(qreg_q[2], qreg_w[0], qreg_w[1])
circuit.ccx(qreg_q[0], qreg_q[1], qreg_w[0])
```

Figure 6.2: Code snippet



```
#DIFFUSER
#circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_w[0], qreg_w[1], qreg_a[0])
#circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_w[0], qreg_w[1], qreg_a[0])
circuit.h(qreg_q[0])
circuit.h(qreg_q[1])
circuit.h(qreg_q[2])
circuit.h(qreg_q[3])
circuit.h(qreg_q[4])
circuit.h(qreg_q[5])
circuit.x(qreg_q[0])
circuit.x(qreg_q[1])
circuit.x(qreg_q[2])
circuit.x(qreg_q[3])
circuit.x(qreg_q[4])
circuit.x(qreg_q[5])
circuit.ccx(qreg_q[0], qreg_q[1], qreg_w[0])
circuit.ccx(qreg_q[2], qreg_w[0], qreg_w[1])
circuit.ccx(qreg_q[3], qreg_w[1], qreg_w[2])
circuit.ccx(qreg_q[4], qreg_w[2], qreg_w[3])
circuit.ccx(qreg_q[5], qreg_w[3], qreg_w[4])
circuit.z(qreg_w[4])
circuit.ccx(qreg_q[5], qreg_w[3], qreg_w[4])
circuit.ccx(qreg_q[4], qreg_w[2], qreg_w[3])
circuit.ccx(qreg_q[3], qreg_w[1], qreg_w[2])
circuit.ccx(qreg_q[2], qreg_w[0], qreg_w[1])
circuit.ccx(qreg_q[0], qreg_q[1], qreg_w[0])
```

```
#circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_w[0], qreg_w[1], qreg_a[0])
circuit.x(qreg_q[0])
circuit.x(qreg_q[1])
circuit.x(qreg_q[2])
circuit.x(qreg_q[3])
circuit.x(qreg_q[4])
circuit.x(qreg_q[5])
#circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_w[0], qreg_w[1], qreg_a[0])
circuit.h(qreg_q[0])
circuit.h(qreg_q[1])
circuit.h(qreg_q[2])
circuit.h(qreg_q[3])
circuit.h(qreg_q[4])
circuit.h(qreg_q[5])
circuit.x(qreg_a[0])
circuit.h(qreg_a[0])
circuit.draw(output='latex_source')
```

```
#measuring with the statevector simulator
shots =2000
sv_sim = Aer.get_backend('statevector_simulator')
qobj = circuit
result = sv_sim.run(qobj,shots = shots,
max_credits = 5, timeout=1).result()
statevec = result.get_statevector()
from qiskit_textbook.tools import vector2latex
vector2latex(statevec, pretext="|\\psi\\rangle =")
```

```
#measuring wih the qasm simulator
shots = 2000
circuit.measure_all()
qasm_sim = Aer.get_backend('qasm_simulator')
qobj = circuit
result = qasm_sim.run(qobj,shots = shots,
max_credits = 5, timeout=1).result()
counts = result.get_counts()
plot_histogram(counts)
```

Figure 6.3: Code snippet