# An Introduction to Quantum Computing and a Collection of Interesting Topics in it

Austin Wu

July 19, 2023

# Contents



**Acknowledgements:** I would like personally thank Sawyer Dobson, Simon Rubenstein-Salzedo, and all the people who reviewed my paper for spending their time helping me grow in this field.

# 1 An Introduction to Quantum Computing


## Abstract

In this paper, we will discuss the main ideas in quantum computing that lead up to Grover's Algorithm. This will start with an introduction to quantum computing, followed by some preliminary algorithms to eventually lead up to Grover's. After that, I will talk about some proofs in the topic and then try and talk about QAOA. My hope with this paper is to give an introduction to the mathematics behind this topic with an understanding of about the level of an introductory Linear Algebra class and almost no needed understanding of quantum mechanics. Ideally you should walk


## 1.1 A Dive into the notation

Before we start talking about quantum computing specifically, it might be best to talk a little bit about the notation for quantum mechanics. Quantum mechanics itself is primarily based around complex vector spaces, yet the way physicists represent them can differ slightly from traditional mathematics.

### Basic Mathematical Definitions

**Definition 1.1.1.** Remember that **Complex Numbers**, or $\mathbb{C}$, are the set of all numbers $a + bi$ where $i = \sqrt{-1}$.

**Definition 1.1.2. The Complex Conjugate** of a complex number $z \in \mathbb{C}$ is $\bar{z}$, where if $z = a + bi$ then $\bar{z} = a - bi$.

**Definition 1.1.3.** The **Absolute Value** of a complex number $z$ is $|z| = a^2 + b^2$.

Here are some properties of complex numbers:

1. $\overline{(\bar{z})} = z$
2. $\overline{z_1 + z_2} = \bar{z_1} + \bar{z_2}$

3. $\bar{z_1 z_2} = \bar{z_1}\bar{z_2}$
4. $|\bar{z}| = |z|$
5. $|z|^2 = z\bar{z}$
6. $z^{-1} = \frac{\bar{z}}{|z|^2}$

## Standard Physics Notation and Mathematical Operations

Bra-Ket notation is a indispensable tool that allows us to cleanly and conveniently denote a systems Quantum state.

**Definition 1.1.4** (Ket)**.** We define a **ket** $|\psi\rangle$ as a column vector in complex vector space $V$.

In other words this is an object in the space $|\psi\rangle \in \mathcal{C}^n$ such that

$$|\psi\rangle = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

where $z_i \in \mathcal{C}$ (the complex plane consisting of all numbers $a + bi$.

The ket part of the notation is primarily there to distinguish the space from the other main part of the notation, the **bra**.

**Definition 1.1.5** (Bra)**.** A **bra** is a row vector in the space

$$\langle\psi| = \begin{bmatrix} z_1 & z_2 \cdots & z_n \end{bmatrix}$$

We can also define the Bra in terms of the Ket, where one is the **Hermitian Conjugate** of the other.

**Definition 1.1.6.** Hermitian Conjugate For any given vector in complex vector space $v \in V$, $v^\dagger = v^{\top *}$ where $\top$ is a transpose and $*$ is the complex conjugate.

It follows then that $\langle\psi| = |\psi\rangle^\dagger$ and vice versa.

**Definition 1.1.7.** Inner Product An innter product is a vector space $V$ over $F$ defied as

$$\langle\cdot,\cdot\rangle : V \times V \to F$$

such that the following axioms are satisfied

- Conjugate Symmetry: $\langle x, y\rangle = \overline{\langle y, x\rangle}$

- Linearity: $\langle ax + by, z\rangle = a\langle x, z\rangle + b\langle y, z\rangle$
- Positive-definiteness: $x \neq 0 \implies \langle x, x\rangle > 0$

Now it just so happens that the inner product is also just an abstraction over the dot project. This makes it easier to visualize.

$$\langle x|y\rangle = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = x_1 \cdot y_1 + x_2 \cdot y_2 + \cdots + x_n \cdot y_n$$

### The Tensor Product

**Definition 1.1.8** (Tensor Prodcut)**.** A Tensor Product is an operation that takes two vector spaces of arbitrary size $V = \{v_1, \cdots v_n\}$, $W = \{w_1 \cdots w_m\}$ and combines them to make a $V \otimes W = \{vw_1 \cdots vw_{nm}\}$ vector space.

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots \\ a_{21}B & a_{22}B & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

## 1.2 What is Quantum Computing?

To me, quantum computing is just the manipulation of quantum data by applying quantum gate operations to it. This will obviously get extended on as we talk about more and more algorithms, but the core idea and what runs on the hardware essentially boils down to this.

### Understanding Quantum States

> "Quantum Computation is a distinctively new way of harnessing nature. It will be the first technology that allows useful tasks to be performed in collaboration between parallel universes." - David Deutsch

To understand Quantum Computing, we must first understand the idea of computing in general. To put it simply, computation is the manipulation of data. The main difference between classical states and quantum states is that quantum systems can employ the power of superposition-the states can all exist at once. As we will see later this is the crucial part behind Quantum Computing, and by utilizing this fact with the realization that we can manipulate the probability that each one of these states exists we can begin to form algorithms with it.

**Definition 1.2.1.** Bit A physical system with two possible distinguishable states.

We can represent a bit as $\mathbb{B} = \{0, 1\}$

**Definition 1.2.2.** Qubit A two-state quantum mechanical system

Mathematically, we can define a qubit as

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. $\alpha$ and $\beta$ both represent the respective probability that when the Qubit is measured the observed state will be 0 or 1 respectively, preventing us from generating a system that doesn't make sense where the probabilities don't correlate with real life.

The crucial difference here between a Bit and a Qubit is that a Qubit is a vector where each state is a probability. This as well see, allows us to have both states be represented at the same time, leading to very strange but powerful phenomenon.

## Quits in Bra-Ket notation

With regards to the notation we can define $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. You can think about it as a zero indexed notation.

We can expand this idea further by writing things such as

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

and further. Essentially, the state of binary bit maps to the index in the vector for which there is a 1.

Now whats really cool about this notation and the tensor is that they are equivalent. In other words, $|1\rangle \otimes |0\rangle = |10\rangle$. This can be thought of as bit concatenation, with their states being preserved under the tensor product. Here the matrix is a one hot encoding of all the states of $\log_2$(number of states) with the binary digit representing which state has 100% probability based off of a 0 indexed scheme.

*Proof.* A ■

**Common Qubits**

Note that we also have Qubits so important we write special symbols for them

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

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

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

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

### Unitary Matrices and Quantum Operations

**Definition 1.2.3.** A Unitary Matrix is a matrix transformation that follows $U^*U = UU^* = UU^{-1} = U^\dagger U = UU^\dagger = I$.

Unitary matrices are a subset of the matrix group that preserves the inner product and in result the probability amplitudes. Applying a unitary matrix operation onto a quantum state can shift the probabilities in certain ways. We will see this in quantum gates as all transformations in quantum computing can be written this way.

## 1.3 Some Physics Behind the Idea

When it comes to quantum computation hardware, it is assumed in the quantum information theorist perspective to be perfect. This means perfect coherence, no error correction needed, and as many qubits as possible. Although this is very far from the real world, mathematicians tend to prefer to abstract the details of the day to day and focus only on the theoretical algorithms.

This being said, here are some of the ways modern quantum architectures work

- Nuclear Magnetic Resonance Quantum Computer

  This is a quantum computer that stores its quantum states through nuclear magnetic resonances.

- Quantum Superconducting

  This is arguably the most common type of quantum computing that exists today. It is where you use superconducting electronics circuits like quantum dots in order to store the states of quantum systems.

- Quantum Phonics

  Quantum Phonics is where you store the quantum superposition in the quantum properties of light.

- Trapped Ion Quantum Computing

  Trapped ion architectures rely on storing the state of Qubits in ionic particles trapped by electromagnetic fields.

- Topological Quantum Computing

  Topological computers are the most mathematical of the bunch. They work by embedding quanutm information inside the topological properties of particles. This is a very recent architecture.

Now the cool thing about quantum information is that none of this matters. All we have to do is abstract the architecture into a idealized form with $n$ qubits an $U$ unitary transformations, allowing theorists to develop algorithms based off of mathematical principles and not having to wait for hardware to catch up.

## 1.4 Quantum Gates

### Classical Gates

**Definition 1.4.1.** Controlled NOT Gate The CNOT gate is a two-qubit gate that can apply a not operation onto the second qubit if the first qubit is $|1\rangle$. It can be written as

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

### Pauli Matrices

The Pauli gates are operations from quantum mechanics we can use that represent rotations in space.

#### X-gate

**Definition 1.4.2.** The Pauli X-gate is the matrix

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

that represents the inversion or NOT gate onto a single Qubit. Similarity this inverts the spin of a Qubit from $|0\rangle$ to $|1\rangle$ and vice versa.

### Y-gate

The Y-gate is interesting as it bascially flips the spin of a particle. It can be described as

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

### Z-gate

The Z-gate is the matrix also flips the spin of a electron.

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

## Hadamard Gates

The Hadamard Gate is utterly crucial for quantum computing. It allows us to generate a base state with all the states in a equal superposition of each other. We can think of it as

**Definition 1.4.3.** Hadamard Gate

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

We can expand this idea to $n$-qubits as following

$$\mathbf{H}^{(n)} = \underbrace{\mathbf{H} \otimes \mathbf{H} \otimes \cdots \otimes \mathbf{H}}_{n \text{ times}} = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x\dot{y}} |y\rangle$$

Where

$$x \cdot y = (x_1 \wedge y_1) \oplus (x_2 \wedge y_2) \oplus \cdots \oplus (x_n \wedge y_n).$$

Now we can apply it to the base to get the super position

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

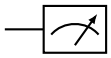
where $(\frac{1}{\sqrt{2}})^2 + (\frac{1}{\sqrt{2}})^2 = 1$. This generalizes to

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{N-1} |x\rangle = \frac{1}{2^{\frac{n}{2}}} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \Bigg\} 2^n \text{ times.}$$

See how this one hot encoded state matrix has equal probability for every state? Thus we have achieved our goal.

## Measurement

Now measurement isn't in itself a gate, but it can be represetned on a diagram by

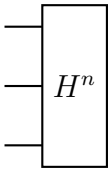


where it is the measure operation. Measurement essentially collapses the state of the Qubit $|x\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ into $|0\rangle$ or $|1\rangle$. This is how we can observe and gather information about or quantum system. In quantum superposition, we cannot know information about the state by the **No-Cloning Theorem** unless we destroy/measure/collapse it. We will see here how we can devise algorithms where we can eventually collapse superpositioned states into $|0\rangle$ or $|1\rangle$ with 100% probability, while some algorithms only slightly change the probabilities to our favor, meaning we must run it many times.

## Quantum Gate Diagrams

Quantum Gate Diagrams are ways we can represent Quantum Circuits graphically. We can represent each line on a sheet as a Qubit state as it time evolves over the $x$ axis. We can then represent a dot as the control bit for a 2-qubit gate. After that we can represent the unitary transformations as boxes along the qubits. Here are some examples:

Hadamard: 


N-Hadamard 

# 2 Preliminary Algorithms in Quantum Computing

Before we talk about Grover's algorithm, it makes sense to talk about simpler to understand ones. The original design behind quantum programs was developed by the Deutch Problem, and so we will start there.

## 2.1 The Quatum Oracle

**Definition 2.1.1.** A oracle, $O$, is an unexposed operation that is used as input to another algorithm.

This is usually defined as $f : \mathbb{B}^n \to \mathbb{B}^m$, or a map from $n$ bits to $m$ bits. We dont have to know what is inside them, only the inputs and outputs. That being said to analyze the total runtime of algorithms we must account for the time complexity of the oracle itself as well (for example in Grover's Algrotihm except that its time complexity is $\mathcal{O}(1)$.

## 2.2 Deutch Problem

Deutch created a though experiment problem that tries to solve

**Given some $f(x)$, $f : \mathbb{B} \to \mathbb{B}$, is $f$ constant or balanced?**

where constant implies $f(0) \neq f(1)$ and balanced implies $f(0) = f(1)$. This is a trivial problem, but it requires two comptuatoins of $f(x)$ in order to solve.

Given some quantum oracle, we can write a two-quibit unitary transformation $U_f$ as $U_f : |y\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$. Now for out main goal we can write the algorithm as a Hadamard of $|01\rangle$ followed by $U_f$ followed by a hadamard followed by a measurement on the first Qubit.

We input the state of both bits, $|01\rangle$. Then we apply the Hadamard

$$H(x) = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) & |0\rangle \\ \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & |1\rangle \end{cases} \implies |01\rangle = |0\rangle \otimes |1\rangle \to \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$$

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

Then we need to apply the Unitary Matrix transformation $U_f = |x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle$

Where now

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

$$\frac{1}{2}(|0f(x)\rangle - |0\rangle |1 \otimes f(x)\rangle + |1f(x)\rangle - |1\rangle |1 \otimes f(x)\rangle)$$

We can then apply another Hadamard transformation on the $|x\rangle$ Qubit only.

$$\rightarrow \frac{1}{2}[((-1)^{f(0)} + (-1)^{f(1)}) |0\rangle + ((-1)^{f(0)} - (-1)^{f(1)}) |1\rangle] \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

From here, we need only try each of the 4 permutations of $f(x), x \in \mathbb{B}$ to prove that if we decompose both bits again the $|y\rangle$ bit in $|x\rangle |y\rangle$ is equal to the answer of our balanced problem. This works as long as we can map our function to a unitary matrix.

This is a really perplexing result. We have just shown that we can get the information of the function by only truing 1 state.

## 2.3 Deutch-Jonza Problem

The Deutch-Jonza Problem is merely a generalization of Deutch's problem. It states that

**On good authority that either $f : \mathbb{B}^n \rightarrow \mathbb{B}$, $f(x) = c$ or $f(x) = 0$ (constant or balanced) for all values, which ones is it?**

Now conventional wisdom states that we should need to try all $2^n$ possibilities, but the Deutch-Jonza problem only requires one computation, meaning that we can gain a exponential speedup over the classical algorithm.

It works the exact same way as before, requiring a hadamard transform in the beginning followed by a unitary oracle transformation followed by another hadmard transformation. The math works out as followed:

# 3 Grover's Algorithm

Grover's Algorithm is one of the most famous algorithms in quantum computing due to its universal applicability. It allows the ability to solve any searching problem given $\mathcal{O}(\sqrt{N})$ quantum operations, yet can be generalized much more with the Quantum Oracle method. For now we will talk about the generalized version and show how this applies to Quantum search later.

**Given an oracle function $f : \{0, \cdots, N-1\} \to \{0, 1\}$ where $N = 2^n$, is there an $x$ where $f(x) = 1$ and what is it?**

Mathematically, this rests on two operations, the oracle unitary operator and the Grover Diffusion Operator.

## 3.1 Algorithm

The Algorithm itself once again starts with a Hadamard transformation on the $|0\rangle$ vector.

$$|s\rangle - H^n |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=1}^{2^n-1} |x\rangle$$

We now have a uniform superposition of all the states. We now want to shift our states so when the function collapses it will be at the $x$ such that the function is satisfied. Lets define our two unitary matrix operators

$$U_f : |x\rangle \otimes |y\rangle \mapsto |x\rangle \otimes |y \oplus f_w(x)\rangle$$

Setting $|y\rangle = |-\rangle$

$$U_f : |x\rangle \otimes |-\rangle (-1)^{f_w(x)} |x\rangle \otimes |-\rangle$$

Now if we define $f_w(x)$ for searching, we essentially want

$$f_w(x) = \begin{cases} 0 & x \neq w \\ 1 & x = w \end{cases}$$

so then

$$(-1)^{f_w(x)} = \begin{cases} 1 & x \neq w \\ -1 & x = w \end{cases}.$$

We can also write it as

$$U_w = I - 2\left|w\right\rangle\left\langle w\right|$$

This completes our oracle unitary transformation case. Now we must define the Grover Diffusion Operator.

**Definition 3.1.1.** Grover Diffusion Operator We can define the operator as $U_D = 2\left\langle s|s\right\rangle - I$ where $|s\rangle$ is our Hadamarded state.

This can also be represnted as

$$U_D = \mathbf{H}(2\left|0\right\rangle\left\langle 0\right| - I)\mathbf{H}$$

and as a $n$-qubit state $|\psi\rangle$ as

$$U_D = \mathbf{H}^n(2\left|0\right\rangle\left\langle 0\right| - I)\mathbf{H}^n.$$

It can also be seen decomposed as

$$U_D = \begin{bmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} - 1 \end{bmatrix}.$$

These two unitary operators get combined to form $U_{grover} = U_D U_w$. Whach what happened when we apply our operator to our base Hadamard state:

Using Grover's Unitary Kernel we have now slightly nudged the probabilities closer towards our desired distribution ($|x\rangle$), where $x$ is the binary representation of our value that satisfies $f(x)$.

Now sine this gate moved the probabilities from $\frac{1}{\sqrt{2^n}}$ to $\frac{3}{\sqrt{2^n}}$, it follows that we must do it $\sqrt{2^n} = \sqrt{N}$ times in order to search through everything and collapse with 100% probability.

Now the proof that this works:

# 4 QAOA

Quantum Approximate Optimization Algorithm, or QAOA for short, is a method of computation for solving combinatorial optimization problems.

The main idea behind this algorithm is to approximate a time-evolution unitary matrix that like Grover's algorithm nudges the probabilities in the right direction. We can do this by Trotter-Suzuki decomposition.

$$U(H,t) = e^{\frac{-iHt}{\hbar}}$$

We can then decompose the unitary matrix

$$e^{A+B} \approx (e^{\frac{A}{n}} e^{\frac{B}{n}})^n$$

and write it as a time-evolution unitary

$$U(H,t,n) = \Pi_{j=1}^{n} \Pi_k e^{\frac{-iH_k t}{n}}$$

leaving us with

$$H = H_1 + H_1 + \cdots + H_N = \sum_k H_k,$$

a much simpler form to deal with. If we can write unitary matrices for each individual Hamiltonian $U(H_1), \cdots, U(H_N)$, then in theory we should be able to approximate our original Hamiltonian.

This is critical in QAQA as we can then define a cost and mixer Hamiltonian for and combanatorial problem and combine them together into a $U(H_i)$, thus leading to a problem where after all the $\sum_k U(H_k)$ gates are applied we reach our desired stat matrix.

# References


[1] Jack Ceroni. Intro to qaoa. *PennyLane Demos*, 2020.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10Th Anniversary Edition*. Cambridge University Press, Cambridge, England, 2010.

[3] Benjamin Schumacher and Michael Westmoreland. *Quantum Processes Systems, and Information*. Cambridge University Press, Cambridge, England, 2010.

[4] Odonnel. Quantum computation and quantum information cmu 18-859bb, fall 2015.

[5] Umesh Vazirani. *CS191 Fall 08*.

[6] Birgitta Whaley. *CS191 Fall 14*.

[7] John Preskill. *CS 219*.

[8] Scott Aaronson. *Introduction to Quantum Information Science*.

[9] Microsoft Research. *Microsoft Quantum Documentaiton*.

[10] IBM Quantum. *Learn quantum computing: a field guide - IBM Quantum*.