An Introduction to Grover's Algorithm

Austin Wu
austin4705@gmail.com

Euler Circle

July 19, 2023

What is Quantum Computing (Mathematically)

The main idea behind quantum computing is that we can consider **Qubits**, or bits that are a quantum superposition of probabilities. We can store these states as complex vectors and manipulate them (run quantum operations on) by transforming them mathematically with Unitary matrices.
Although the hardware behind these devices is very complicated, we only need worry about the abstracted mathematical picture for quantum algorithms.

Notes about the Notation

Definition (Ket)

A Ket is a complex valued vector $|\psi\rangle$.

Definition (Bra)

We can define a Bra as the Hermitian Conjugate to a Ket.
$\langle\psi| = |\psi\rangle^{\dagger} = |\psi\rangle^{T*}$

Definition (Qubit)

A Qubit is the smallest unit of quantum information. It can be thought of as a $\mathbb{C}^2$ vector $|\psi\rangle$ such that

$$|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \quad a, b \in \mathbb{C}^2, \ |a|^2 + |b|^2 = 1$$

Notes about the Notation 2

Definition (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.

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

Definition (Unitary Matricies)

A Unitary Matrix is a matrix transformation that follows $U^*U = UU^* = UU^{-1} = U^\dagger U = UU^\dagger = I$. This preserves the inner product and also as a result the probability amplitudes.

Notes about the Notation 3

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.

Hadamard Transform

The Hadamard Transform is a type of **Quantum Gate** operation that we can apply on Qubits. It is normally used to mix the position of the Qubits into a sum of all suppositions. Given n Qubits the Hadamard transform applies the transformation

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

onto $N = 2^n$ states.
The transformation itself is also just a Discrete Fourier Transform onto the Finite Abelian Boolean Group. This is also not a state we can make with classical computers, as they canot be in multiple states at once.

Deutch's Problem

Deutch's Problem is perhaps one of the simplest quantum algoritms, hence why I want to show it first. The main idea behind the algorithm is that given a function $f(x) : \mathbb{B}^1 -> \mathbb{B}^1$, we want to determine if it is constant or balanced ($f(0) = f(1)$ or $f(1) \neq f(0)$). We can do this by running the Hadamard Transofrm first, allowing us to make use of **Quantum Parallelism** and making both measurable states part of the current superposition state.

Deutch's Problem 2

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
$U_f = |x\rangle\,|y\rangle \to |x\rangle\,|y \oplus f(x)\rangle$

Deutch's Provlem 3

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.

What is Grover's Algorithm?

Grover's Algorithm is a quantum algorithm that allows you to answer if given $f : \{0, \ldots, N-1\} \to \{0, 1\}$, there is an x such that $f(x) = 1$ and what that x is in $\mathcal{O}(\sqrt{N})$ operations while still only requiring $\mathcal{O}(\log(N))$ Qubits to run.

This is important because we can map just about any problem that relates to searching through N queries into some $f(x)$ that we can input into Grover's Algorithm, allowing us to receive a $\sqrt{N}$ speedup on searching. This also impacts other NP complete problems that rely on searching as well, allowing us to gain a polynomial speedup there.

Grover's Algorithm

The first step of the algorithm is quite standard for quantum algorithms: Applying the **Hadamard Transform** onto the blank state $|0\rangle$.

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

This leaves us with a even sum of the superposition.

Grover's Algorithm 2

The next step in the algorithm is to apply the unitary oracle transformation. An oracle is just a function that we don't know what is inside of it but we can call it in a single operation. For our purposes we need a search function,

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

By setting $|y\rangle = |-\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$, U_w becomes the phase oracle

$$U_w : |x\rangle \otimes |-\rangle \mapsto (-1)^{f_w(x)} |x\rangle \otimes |-\rangle \text{ where } (-1)^{f_w(x)} = \begin{cases} 1 & x \neq w \\ -1 & x = w \end{cases}$$

which eventually simplifies into $U_w = I - 2|w\rangle\langle w|$.

Grover's Algorithm 3

Finally we can run the Grover Diffusion Operator over our quantum states. It can be constructed as followed:

$$U_D = 2\left|d\right\rangle\left\langle d\right| - I$$

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

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

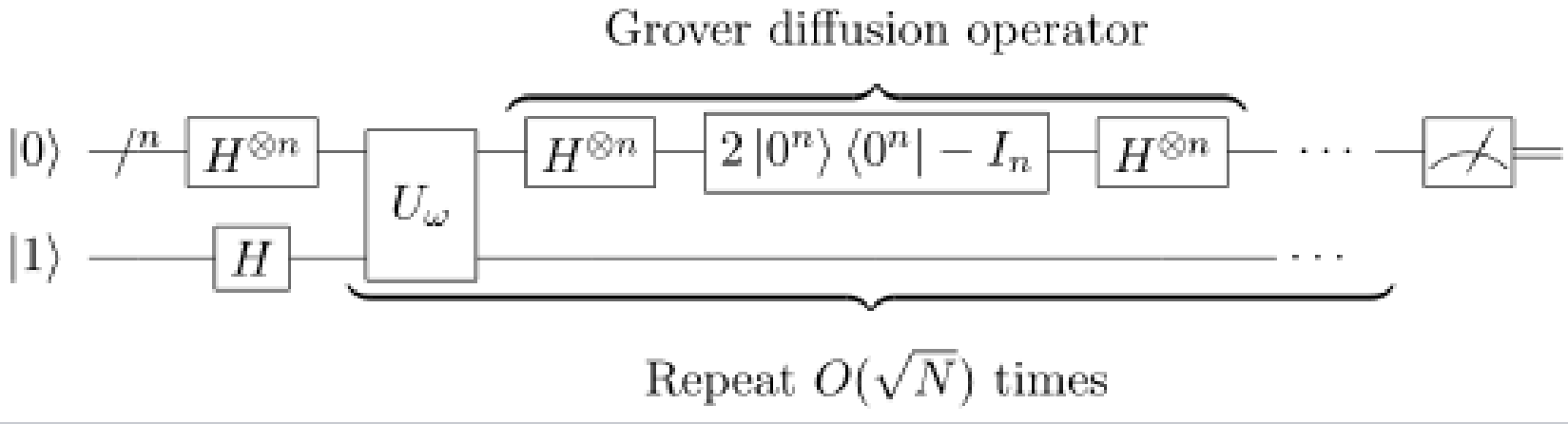
where $H^n = \underbrace{H \otimes H \cdots \otimes H}_{n \text{ times}}$. We then apply this unitary transformation.

Grover's Algorithm 4

This represents a single iteration of Grover's Algorithm. If we look at the output after the first kernel, the value at the state we desire will approximately go from $\frac{1}{\sqrt{N}}$ to $\frac{3}{\sqrt{N}}$. If we run the two unitary matrix operators $U_{\text{Grover}} = U_D U_w$ state over and over, our amplitudes on the matching element should increment as $\frac{1}{\sqrt{N}}, \frac{3}{\sqrt{N}}, \frac{4}{\sqrt{N}}, \cdots, \frac{\sqrt{N}}{\sqrt{N}}$, giving us 100% probability that we when we measure the states (make the output observable), we will return the quantum state of the position of the element we are looking for! Notice how it would take $\mathcal{O}(\sqrt{N})$ iterations of U_{Grover} in order to reach probability of 1, hence why the time complexity is $\mathcal{O}(\sqrt{N})$.

Quantum Gate Description

Here is a drawing of Grover's Algorithm in a Quantum Gate Diagram.



Polynomial Method for Quantum Query Complexity

The polynomial method is a way to show that the lower bounds in the amount of queries to the function $f(x)$ in Grover's Algorithm is $\Omega(\sqrt{N})$. It works as follows

Theorem

Let N be a quantum algorithm that makes t queries to the function $f(x)$. Then there exists $2^n - 1$ complex polynomials each with degree of most t.

The proof is nontrivial so I will not cover it here, but we can use this fact if we have a polynomial of degree Ω then we must make at least $\Omega(N)$ queries to the oracle function in order for the algorithm to work. We can do this specifically for Grover's Algorithm by generating a approximation polynomial over the entire oracle space and then using that to create a complex polynomial which is at least size $\Omega(\sqrt{N})$. This allows us to then show that there is a minimum amount of queries that our quantum system must make, although it still does not prove anything about a maximum.

End of Presentation

Thank you for Listening!
Feel free to ask any questions or message me later at austin4705@gmail.com or @austin4705 on discord.
PS: My paper talks about other stuff besides Grover's, so I suggest you read it!