A Brief Introduction to Quantum Algorithms

An Exploration of Grover's and Shor's Algorithms

Christopher Kang

June 2020

Abstract

Quantum computing is a fascinating new paradigm of computing that opens many questions into the complexity classes of problems (aka how quickly they can be solved). This paper explores quantum search (Grover's algorithm) and the quantum factoring algorithm (Shor's algorithm). The bounds produced are often surprising and counterintuitive, reflecting the quirky properties of quantum computing.

1 Introduction to Quantum Computing

1.1 Representing Information

Quantum computers, like their classical counterparts, operate on a series of bits. Quantum bits, or qubits, are represented by a state vector

Definition 1.1 (State Vector)**.** *A **state vector** is the representation for a quantum state. It is typically represented as a column vector where each row's value represents the amplitude of the corresponding bit value:*

$$\begin{bmatrix} c_{00..0} \\ c_{00..1} \\ \vdots \\ c_{11..1} \end{bmatrix}$$

A state vector is valid if and only if:

1. $c_k \in \mathbb{C}$
2. $\sum_k |c_k|^2 = 1$

These c_k, or **amplitudes**, are important because they define the probability of the state collapsing to that value when measured. Specifically, $\mathbb{P}(k) = |c_k|^2$ (more on this later).

This seems extraordinarily suspicious (or 'sketchy') - how could information in a quantum state simultaneously encode not just real numbers, but complex numbers with infinite precision? This is a major difference that lends itself to a surprising result.

Fact 1.1. *An unknown quantum state cannot have its amplitudes identified with perfect precision in finitely many trials.*

As a notational convenience, we typically represent state vectors as kets:

Definition 1.2 (Ket Vector)**.** *A **ket** represents a column vector.*

$$|x\rangle = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Kets operate like normal column vectors; for instance, we can still perform outer products of two forms:

Definition 1.3 (Kronecker Product)**.** *The **Kronecker product** is a generalization of the outer product and applies for vectors and matrices. Denoted $\otimes$, it is represented in ket form as:*

$$|A\rangle \otimes |B\rangle = |A\rangle |B\rangle$$

Where, if $|A\rangle$ is $m \times 1$ and $|B\rangle$ is $n \times 1$ then $|A\rangle |B\rangle$ is $mn \times 1$ in the form $A_1B_1, A_1B_2, ...A_2B_1, A_2B_2, ...$

Definition 1.4 (Outer Product)**.** *An **outer product** is notated in Braket notation as:*

$$|A\rangle \langle B| = |A\rangle \otimes \langle B|$$

Additionally, we can still apply inner products. Consider a row vector, or bra:

Definition 1.5 (Bra Vector)**.** *A **bra** represents the conjugate transpose (†) of its corresponding ket vector:*

$$\langle x| = |x\rangle^\dagger = \begin{bmatrix} x_1^\dagger & x_2^\dagger & \dots & x_n^\dagger \end{bmatrix}$$

Thus, our inner product is notated:

Definition 1.6 (Inner Product)**.** *An **inner product** is notated in Braket notation as:*

$$\langle A|B\rangle = A^\dagger B$$

1.2 Quirks of quantum computing

There are two notable aspects of quantum computing that are notable mathematically.

First, quantum computers are inherently probabilistic, unlike classical computers:

Definition 1.7 (Superposition)**.** *A state vector in **superposition** has amplitudes other than 1. As an example of a 1 qubit system:*

$$|A\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}$$

Because the state can be represented as $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$; namely, it exists as both $|0\rangle$ and $|1\rangle$ simultaneously.

However, when we measure the state, the system will 'collapse':

Definition 1.8 (Measurement)**.** ***Measurement** of quantum states is probabilistic. The measurement operator can be represented as a projector; say we'd like to measure the state k. Then, the operator P_k is written as:*

$$P_k = |k\rangle \langle k|$$

Say we have a state $|\psi\rangle$; then, the output state after measurement is:

$$\frac{P_k |\psi\rangle}{\sqrt{\langle \psi | P_k^\dagger P_k | \psi \rangle}}$$

The probability of yielding the $|k\rangle$ state is $\langle \psi | P_k^\dagger P_k | \psi \rangle$, or the magnitude squared of the amplitude of $|k\rangle$.

Measurement is also important for quantum systems because it can affect **entangled** systems:

Definition 1.9 (Entanglement)**.** ***Entanglement*** *means that the value of one qubit can affect the value of others. This emerges as a consequence of superposition and the limits on valid state vectors, and can be seen with the following state vector:*

$$\begin{bmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

In this case, measurement collapses the state into $|00\rangle$ *with probability* $1/2$ *or* $|11\rangle$ *with probability* $1/2$ *- this means that if we only measured on one of the qubits, we'd be able to discern the value of the other without measurement.*

1.3 Acting Upon Information

Now that we can represent information, we must be able to manipulate it. All quantum operations or 'gates' are square unitary matrices with dimension $2^n \times 2^n$:

Definition 1.10 (Unitary)**.** ***Unitary*** *matrices* U *satisfy:*

$$UU^\dagger = U^\dagger U = \mathbb{1}$$

Where $U^\dagger$ *is the complex conjugate transpose of* U*.*

This paper is primarily concerned with the algorithms, not implementation. So, we take the following fact to hold:

Fact 1.2 (Implementability of Unitaries)**.** *We assume that we have a set of gates that enable an arbitrary unitary* U *to be approximated with matrix norm error* $\leq \epsilon$*.*

1.4 Comparing Algorithms

Now that we can produce algorithms, we are interested in comparing them asymptotically. There are three primary notations used:

Definition 1.11 (Big O Notation)**.** ***Big O notation*** *produces a class of functions that are asymptotic upper bounds for a desired function. Suppose some algorithm has cost* $f(n)$*. Then,*

$$g(n) \in O(f(n)) \implies \exists c > 0, \exists n_0 \ni n > n_0 \implies f(n) \leq cg(n)$$

Definition 1.12 (Omega Notation)**.** ***Omega notation*** *produces a class of functions that are asymptotic lower bounds for a desired function. Suppose some algorithm has cost* $f(n)$*. Then,*

$$g(n) \in \Omega(f(n)) \implies \exists c > 0, \exists n_0 \ni n > n_0 \implies g(n) \leq cf(n)$$

Definition 1.13 (Theta Notation)**.** ***Theta notation*** *produces a class of functions that scale to the same degree of a desired function. Suppose some algorithm has cost* $f(n)$*. Then,*

$$g(n) \in \Theta(f(n)) \implies g(n) \in O(f(n)) \wedge g(n) \in \Omega(f(n))$$

As an example, Big O notation gives asymptotic upper bounds - basically, what is the max scaling that a function requires. So, for example, a function in $O(n^2)$ will have roughly quadruple the cost given double the input size. Omega says the minimum scaling that a function requires - trivially, all functions are $\Omega(1)$, because a function at minimum requires constant time. Last, Theta notation is a tight bound - basically, what function best could be the lower and upper bound?

2 Search

For our first problem, we discuss searching. Searching is well-studied for classical computers, and Grover's search was one of the first quantum algorithms. Grover's is particularly interesting because of its algebraic and geometric interpretations, which are discussed in this section.

2.1 Classical Premise

Search from a computational perspective can be described as follows:

Problem 2.1. *Suppose we have a list of bitstrings indexed from 0 to $2^n - 1$ where n is the number of bits, with $N = 2^n$ as the size of the bitspace. Let's call the desired string $m \in [0, 2^n - 1]$. How many times does the list need to be accessed to find the string m?*

Theorem 2.1. *All classical search algorithms must operate worst-case in $\Omega(N)$.*

Proof. Let's say we have an algorithm A. No matter the order A accesses the list, there is always the potential that m is the last string accessed. Thus, $A \in \Omega(N)$. □

This result is unsurprising and intuitive - in essence, we must look at elements of the list one by one to identify the string m.

2.2 Quantum Algorithm

2.2.1 Premise and Algebraic Proof

Grover's algorithm [6], or quantum search, takes the same premise:

Problem 2.2. *Suppose we have a list of bitstrings indexed from 0 to $N = 2^n - 1$. Let's call the desired string m. We are given the quantum marking oracle $U_\omega : |x\rangle \mapsto (-1)^{f(x)} |x\rangle$ where*

$$f(x) = \begin{cases} 1 & x = m \\ 0 & x \neq m \end{cases}$$

How can we identify m with the fewest calls to U_ω?

Definition 2.1 (Oracles)**.** *An **oracle** is a black-box function that is provided externally. While I do not explicitly discuss the implementability of oracles, all oracles used in the paper are valid quantum operations.*

But yields an extroardinarily surprising result:

Theorem 2.2. *Grover's algorithm can identify m in $\Theta(\sqrt{N})$ with high probability, even if the explicit behavior of U_ω is unknown.*

Proof. Our general algorithms proceeds as follows:

Algorithm 1: Quantum Search (Grover's Algorithm)

Result: m known
set $|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle$;
Apply diffusion operator $W = -(\mathbb{1} - 2 |\psi_0\rangle \langle\psi_0|)(1 - 2 |m\rangle \langle m|)$ in $O(\sqrt{N})$ times;
Measure the resulting state $|\psi\rangle$

We begin by creating an equal superposition over all the bitstrings:

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_j |j\rangle$$

Recognize that we can describe $|\psi_0\rangle$ as the combination between $|m\rangle$ and the perpendicular vector $|\perp\rangle$:

$$|\psi_0\rangle = \sin\theta |m\rangle + \cos\theta |\perp\rangle \tag{1}$$

We may decompose $|\psi_0\rangle$ in this way because $|m\rangle \perp |\perp\rangle$ which span the vector space by choosing appropriate $|\perp\rangle$. (Intuitively, imagine the vector from the origin to a point on the unit circle. Then, that vector can be expressed as the sum of two perpendicular vectors).

Now, we create a special operator: Grover's diffusion operator W. The operator is defined as follows:

$$W = -(\mathbb{1} - 2\,|\psi_0\rangle\,\langle\psi_0|)(\mathbb{1} - 2\,|m\rangle\,\langle m|) \tag{2}$$

For more geometric intuition, see Section 2.2.2. Additionally, recall that W must be unitary for a quantum computer to implement the operation. I claim:

Fact 2.1. *W is implementable with $O(1)$ U_ω calls, even when m is not explicitly known.*

Remark. *Again, this is taken to be true, as I will limit discussions on implementation.*

So, let's model the application of W on $|\psi_0\rangle$. Recognize that:

$$\begin{aligned}(\mathbb{1} - 2\,|m\rangle\,\langle m|)\,|\psi_0\rangle &= |\psi_0\rangle - 2\,|m\rangle\,\langle m|\,(\sin\theta\,|m\rangle + \cos\theta\,|\perp\rangle)\\ &= |\psi_0\rangle - 2\sin\theta\,|m\rangle\end{aligned}$$

Because $\langle m|\perp\rangle = 0$. (Additionally, recall that $|m\rangle\langle m|$ is an outer product). Applying the second operator:

$$\begin{aligned}&= -(\mathbb{1} - 2\,|\psi_0\rangle\,\langle\psi_0|)(|\psi_0\rangle - 2\sin\theta\,|m\rangle)\\ &= -(|\psi_0\rangle - 2\sin\theta\,|m\rangle - 2\,|\psi_0\rangle\,\langle\psi_0|\,|\psi_0\rangle + 4\sin\theta\,|\psi_0\rangle\,\langle\psi_0|\,|m\rangle)\\ &= -(-\,|\psi_0\rangle - 2\sin\theta\,|m\rangle + 4\sin^2\theta\,|\psi_0\rangle)\\ &= (1 - 4\sin^2\theta)\,|\psi_0\rangle + 2\sin\theta\,|m\rangle\end{aligned}$$

Because we are solely interested in the effects on the $|m\rangle$, we'll only analyze that amplitude:

$$\begin{aligned}&= \sin\theta(1 - 4\sin^2\theta) + 2\sin\theta\\ &= \sin\theta\cos^2\theta - 3\sin^3\theta + 2\sin^3\theta + 2\sin\cos^2\theta\\ &= 3\sin\cos^2\theta - \sin^3\theta\\ &= \sin 3\theta\end{aligned}$$

Thus, we were able to manipulate the amplitude of $|m\rangle$, potentially increasing the probability of measuring m. More generally, we can describe the behavior of W below:

Lemma 2.1 (Action of Grover's Diffusion Operator)**.** *Grover's Diffusion Operator W can be modeled as such:*

$$W^p(\sin\theta\,|m\rangle + \cos\theta\,|\perp\rangle) = \sin(2p+1)\theta\,|m\rangle + \cos(2p+1)\theta\,|\perp\rangle \tag{3}$$

Proof. Let's prove this by induction. We've already demonstrated the base case for $p = 1$.

Now, let's perform the inductive step. Assume that this holds for $p = k$; we'd like to show this holds for $k + 1$. Then, we'll need to evaluate

$$W(\sin(2p+1)\theta\,|m\rangle + \cos(2p+1)\theta\,|\perp\rangle)$$

Applying the first component of Grover's diffusion operator:

$$\begin{aligned}&= (1 - 2\,|m\rangle\,\langle m|)(\sin(2p+1)\theta\,|m\rangle + \cos(2p+1)\theta\,|\perp\rangle)\\ &= \sin(2p+1)\theta\,|m\rangle + \cos(2p+1)\theta\,|\perp\rangle - 2\sin(2p+1)\theta\,|m\rangle\\ &= -\sin(2p+1)\theta\,|m\rangle + \cos(2p+1)\theta\,|\perp\rangle\end{aligned}$$

Now, applying the second part of the operator, we have:

$$
\begin{aligned}
&= -(\mathbb{1} - 2\,|\psi_0\rangle\,\langle\psi_0|)(-\sin(2p+1)\theta\,|m\rangle + \cos(2p+1)\theta\,|\perp\rangle) \\
&= \sin(2p+1)\theta\,|m\rangle - \cos(2p+1)\theta\,|\perp\rangle - 2\sin(2p+1)\theta\,|\psi_0\rangle\,\langle\psi_0|\,|m\rangle + 2\cos(2p+1)\theta\,|\psi_0\rangle\,\langle\psi_0|\,|\perp\rangle)
\end{aligned}
$$

Again, we only care about the amplitude of the $|m\rangle$ vector, as the quantum state is normalized so the $|\perp\rangle$ vector will have a corresponding amplitude. Thus:

$$
\begin{aligned}
&= \sin(2p+1)\theta - 2\sin(2p+1)\theta\sin^2\theta + 2\cos(2p+1)\theta\cos^2\theta \\
&= \sin(2p+1)\theta(\cos^2\theta - \sin^2\theta) + 2\cos(2p+1)\theta\sin\theta\cos\theta \\
&= \sin(2p+1)\theta\cos 2\theta + \cos(2p+1)\theta\sin 2\theta \\
&= \sin(2p+3)\theta \\
&= \sin(2(p+1)+1)\theta
\end{aligned}
$$

As desired. Thus, by induction, we've demonstrated that the desired relation holds. □

Thus, we can tune the probability of returning m by manipulating the number of times we apply W. Recall that $\sin^{-1}(x) \approx x$, so θ is about $\sin^{-1}(\frac{1}{\sqrt{2^n}}) \approx \frac{1}{\sqrt{2^n}}$. Then, the probability of measuring $|m\rangle$ is:

$$\mathbb{P}(m) = \sin^2((2p+3)\theta) = \sin^2\left(\frac{2p+3}{\sqrt{2^n}}\right) \tag{4}$$

Because $\sin^2\frac{\pi}{2} = 1$, if we can bring the fraction to $\frac{\pi}{2}$, we can maximize the probability of reading the correct value. Thus, choose appropriate p:

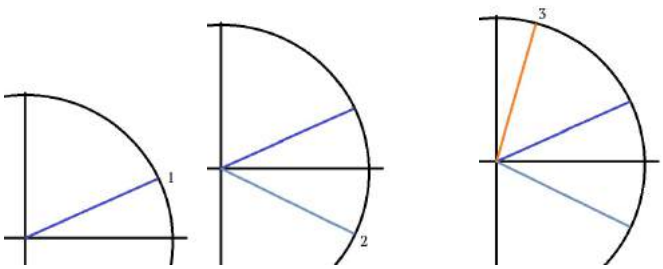
$$
\begin{aligned}
\frac{2p+1}{\sqrt{2^n}} &= \frac{\pi}{2} \\
2p+1 &= \pi\sqrt{2^{n-2}} \\
p &= \frac{\pi\sqrt{2^{n-2}} - 1}{2} \in O(\sqrt{N})
\end{aligned} \tag{5}
$$

Because p integer, choose integer p sufficiently close, thereby allowing us to maximize the probability. Furthermore, because $N = 2^n$, and each W requires $O(1)$ calls to U_ω, Grover's algorithm scales in $\Theta(\sqrt{N})$, as desired. □

So, we can identify a single marked state with high probability in $\Theta(\sqrt{N})$. In essence, we are able to overcome the limitation of classical algorithms where they need to check every entry individually! While the bound isn't necessarily superb, it's fascinating that we can go below $\Omega(N)$.

2.2.2 Geometric Interpretation

The math above seems like magic, to some degree. The geometric interpretation is more intuitive. In essence, we should treat W as two reflections: one about the $|m\rangle$ axis and the other about the $|\psi_0\rangle$ axis.



1. We begin with a geometric interpretation of the starting state defined by Equation (1), where the horizontal axis is $|\perp\rangle$, the vertical axis is $|m\rangle$, and the blue state is $\sin\theta\,|m\rangle + \cos\theta\,|\perp\rangle$.

2. Recall the definition of the diffusion operator Equation (2), and split it into it's components. $(\mathbb{1} - 2\,|m\rangle\langle m|)$ is essentially a reflection across the $|\perp\rangle$ axis - recognize that the unitary *only* negates the amplitude of the $|m\rangle$ component.

3. Finally, we need to apply $-(\mathbb{1} - 2\,|\psi_0\rangle\langle\psi_0|)$. This, in essence, is a reflection across $|\psi_0\rangle$, our starting state.

Applied over many trials, this process essentially amplifies the amplitude of $|m\rangle$. Grover's can be generalized for a process known as **Amplitude Amplification**, which is used in many emerging quantum algorithms [1].

2.3 Grover's Algorithm for Multiply Marked States

We can also extend this technique to work over multiple marked states:

Problem 2.3. *We continue with the same premise of the original search algorithm, but instead have l marked states out of 2^n potential states (where n is the number of qubits). How many times does our oracle U_ω need to be called?*

Theorem 2.3 (Grover's with Multiple Marked States)**.** *Given N possible strings, l of which are marked, we can return a marked state m_k with high probability. Furthermore, each of the m_k has an equal probability of being selected.*

Proof. The proof is briefly sketched: our decomposition of $\sin\theta\,|m\rangle + \cos\theta\,|\perp\rangle$ holds; simply replace $|m\rangle$ with $\sum|m_k\rangle$. Our resulting state will be $\sin\theta'\,|m\rangle + \cos\theta'\,|\perp\rangle$ where $\theta' \approx \frac{\pi}{2}$. Thus, we will measure $\sin\theta'\,|m\rangle$ with high probability, and each of the $|m_k\rangle$ have equal probability of being selected.

□

2.4 Extension: Finding Minimums

Immediately, Grover's seems asymptotically surprising but irrelevant - in what scenario are we given some marking oracle without explicit knowledge of m? This would require an *outsider* to know the value of m.

I present a problem that demonstrates the value and is explored by Durr and Hoyer [3]:

Problem 2.4. *Consider we have a database with values $x_0, x_1 ..., x_{2^n-1}$ on n qubits. Suppose we have some $O_l\,|i\rangle\,|0\rangle \mapsto |i\rangle\,|x_i\rangle$. Is there a quantum algorithm that can identify the minimum value asymptotically quicker than in a classical system?*

Theorem 2.4. *A quantum algorithm can identify the minimum of the database with high probability given only $O(\sqrt{N}\log N)$ calls to O_l, whereas a classical algorithm requires $\Omega(N)$.*

Proof. We'll leverage the following algorithm:

Algorithm 2: Quantum Min Finding

Result: $\min_k x_k$ known
Randomly measure from the database x_k;
Set this value to m;
repeat
  Create the marking oracle U_m that marks all states less than m;
  Apply Grover's with U_m and measure the state, which returns i where $x_i \leq m$;
  Call $O_l\,|i\rangle\,|0\rangle$ to find x_i;
  Set m to x_i
until $\log N$ *trials have passed*;

Explicitly, we define U_m as:

$$U_m |i\rangle \mapsto \begin{cases} -|i\rangle & x_i \leq m \\ |i\rangle & x_i > m \end{cases}$$

Where x_i is the ith element of the database. Additionally, by Theorem 2.3, when we apply Grover's algorithm we sample from a uniform distribution of all database items less than m. So, for example, if we have both $|0\rangle, |1\rangle$ as "marked" states, then Grover's will return $0, 1$ each with a probability of $\frac{1}{2}$.

Remark. *Grover's can actually fail sometimes; let's call this probability p. To address this, if the given $x_i > m$, repeat the calculation with the same m. Call n the number of trials until a valid $x_i \leq m$ given; then:*

$$\mathbb{E}[n] = \frac{1}{1-p}$$

Thus, we'll cost $\frac{1}{1-p}$ calls to O_l instead of 1; additionally, p does not grow asymptotically as N grows (Theorem 2.3), so making Grover's that operates without fault is still in $\Omega(\sqrt{N})$.

Lemma 2.2. *To ensure that we identify the minimum with high probability ($\geq \frac{1}{2}$), we'll need to run quantum search in $O(\log N)$ time.*

Remark. *The following proof actually differs from Durr and Hoyer's, but yields the same asymptotic scaling*

Proof. Let's model the action of Grover's. Create the auxiliary function $p(m, t)$, or the probability of obtaining the lowest value by starting at the mth smallest value and running for t trials. Recognize that this function has a recursive relationship:

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

This holds because Grover's will return every element lower than the threshold with equal probability, so there is a $\frac{1}{m}$ chance of selecting m, a $\frac{1}{m}$ chance of selecting $m-1$, etc.

Now, let's evaluate some of the basic values for p. Recognize that:

$$p(1, n) = 1 \; p(x, 0) = 0, x \neq 1 \; p(k, 1) = \frac{1}{k}$$

These follow because

1. $p(1, n) = 1$ - if we have obtained the lowest element, the probability of obtaining the lowest element is 1, regardless of the number of trials remaining
2. $p(x, 0) = 0, x \neq 1$ - if all trials have been exhausted, but we are not at the lowest element, there is no chance of obtaining the lowest element
3. $p(k, 1) = \frac{1}{k}$ - if there is only 1 trial remaining, the only chance of success is obtaining the lowest element directly

Now, recall the following inequality:

Fact 2.2 (Reciprocal AM-HM Inequality)**.** *Recall by the AM-GM-HM inequality:*

$$\frac{n}{\frac{1}{k_1} + \frac{1}{k_2} + .. + \frac{1}{k_n}} \leq \sqrt[n]{k_1 k_2 ... k_n} \leq \frac{\sum_{j=1}^{n} k_j}{n}$$

Thus, taking the reciprocal:

$$\frac{\sum_{j=1}^{n} \frac{1}{k_j}}{n} \geq \frac{n}{\sum_{j=1}^{n} k_j}$$

So, let's do a simple execution of $p(m, 2)$, leveraging the theorem:

$$\begin{aligned} p(m,2) &= \frac{1}{m}\sum_{k=1}^{m} p(k,1) \\ &= \frac{\sum_{k=1}^{m} \frac{1}{k}}{m} \\ &\geq \frac{m}{\sum_{k=1}^{m} k} \\ &= \frac{m}{(m)(m+1)/2} \\ &= \frac{2}{m+1} \end{aligned}$$

We can generalize a closed form for the lower bound of $p(m, t)$:

Lemma 2.3. *$p(m, t)$ has the following lower bound*

$$p(m,t) \geq \frac{2^{t-1}}{m-1+2^{t-1}}$$

Proof. Let's prove this by induction. Our base case holds, as when $t = 1$

$$p(m,1) = \frac{1}{m} \geq \frac{1}{m}$$

Now, our inductive hypothesis is that $p(m,t) \geq \frac{2^{t-1}}{m-1+2^{t-1}}$ for $t = r$. Let's show this holds for $t = r + 1$. Recognize that:

$$\begin{aligned} p(m,r+1) &= \frac{1}{m}\sum_{k=1}^{m} p(k,r) \\ &\geq 2^{r-1}\frac{\sum_{k=1}^{m} \frac{1}{k-1+2^{r-1}}}{m} \end{aligned}$$

By the AM-GM-HM Inequality:

$$\begin{aligned} &\geq 2^{r-1}\frac{m}{\sum_{k=1}^{m} k-1+2^{r-1}} \\ &= 2^{r-1}\frac{2m}{2^r m - 2m + m(m+1)} \\ &= \frac{2^r}{2^r - 2 + m + 1} \\ &= \frac{2^r}{m-1+2^r} \\ &= \frac{2^{r+1-1}}{m-1+2^{r+1-1}} \end{aligned}$$

We have demonstrated that the inductive step holds. Thus, by induction, the lower bound holds. □

Remark. *This bound was somewhat troublesome to identify; I initally began by applying an integral bound, because $\sum_{k=1}^{n} \frac{1}{k} \geq \int_1^{n+1} \frac{1}{x}\,dx$. Even though $p(m,t)$ would be expressed as iterated integrals, there is actually a closed form. Unfortunately, this bound is not tight enough. I suspect this occurred because the approximation was not strong enough; a potential second avenue for exploration would be the following approximation [5]:*

$$H_n \approx \ln n + \gamma$$

Where H_n is the partial sum of the first n terms of the Harmonic series and γ is the Euler-Mascheroni constant. But, I digress.

Now that we have a lower bound on $p(m, t)$, we can prove an asymptotic bound. So, let's suppose we have some m and we'd like to identify the satisfactory t_0 where the calculation is likely to succeed. Thus, we'd like:

$$p(m, t_0) \geq \frac{1}{2}$$

So,

$$\begin{aligned}
\frac{2^{t_0-1}}{m-1+2^{t_0-1}} &= \frac{1}{2} \\
2^{t_0} &= m - 1 + 2^{t_0-1} \\
2^{t_0-1} &= m - 1 \\
t_0 - 1 &= \log_2(m-1) \\
t_0 &= \log_2(m-1) + 1
\end{aligned}$$

Thus, the number of trials we need to take $t_0 \in \Theta(\log N)$, as desired. □

Because Grover's runs in $O(\sqrt{N})$ and we'll need to repeat it $\log N$ times, our overall algorithm runs in $O(\sqrt{N} \log N)$. In contrast, a classical algorithm must check every value of the database, otherwise it could omit the smallest value. Thus, a classical algorithm would operate in $\Omega(N)$. So, our quantum algorithm runs asymptotically faster, as desired. □

2.5 Review

This section has explored a fundamental yet elegant quantum algorithm. The most fascinating aspect is the geometric connections - in essence, Grover's algorithm reflects vectors to maximize the desired probability. There is **no analog** in classical computing, reflecting how quantum computing creates an entirely new class of algorithms.

Additionally, concerns into implementing the marking oracle can be allayed with our exploration into the quantum minimum finding Section 2.4. This work demonstrated that implementing the marking oracle is feasible, even when the marked state is not explicitly known.

3 Factoring

In this section, we extend our analysis to a problem that experiences a greater quantum speedup: factoring. This section explores how problems can be reduced and made quantum-solvable. Furthermore, we continue introducing techniques that are uniquely quantum, including Quantum Phase Estimation and the Quantum Fourier Transform.

3.1 Classical Premise

Let's begin with the description of the problem of factoring:

Problem 3.1. *Given some number n-bit number N, what are factors x, y where $xy = N$ and $x, y \neq 1, N$? (i.e. What are nontrivial factors of N?)*

Fact 3.1. *Computer scientists suspect that factoring is in sub-exponential time $O((1+\epsilon)^x)$ and not polynomial time (or $O(n^k)$). This means that adding additional bits exponentially increases the cost of factoring.*

Remark. *Note that we specifically care about the length of the input n, not the size of the number N. This differs from the previous case with quantum searching, because asymptotic analysis focuses on the size of input. Grover's algorithm is given an N-long database; factoring algorithms are given n-bit long numbers.*

This difficulty in factoring underlies many modern encryption system - if we some large $N = p_1 p_2$ where p_1, p_2 prime, it will take exponentially long to identify p_1, p_2.

3.2 Quantum Algorithm

3.2.1 Order-Finding

We will not immediately tackle factoring; instead, we discuss how factoring can be reduced to a related problem, order-finding. For details on the reduction, see Section 3.2.2.

The problem of order-finding is as follows:

Problem 3.2 (Order-finding on a quantum computer)**.** *Suppose we have some $x < N$ with $\gcd(x, N) = 1$ (x, N coprime). Then, what is the smallest $r > 0$ satisfying:*

$$x^r \equiv 1 \bmod N$$

Theorem 3.1. *A quantum algorithm can perform order-finding in $O(n^3)$ with high probability, where n is the number of bits required to represent N.*

Proof. We will use this algorithm to identify the order [2, p. 232].

Algorithm 3: Quantum Order-Finding

Result: least possible $r > 0$ known
Initialize $|\psi_0\rangle = |0\rangle^{\otimes t} |1\rangle^{\otimes n}$, with t sufficiently large;
Create superposition $\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle |1\rangle^{\otimes n}$;
Apply modular exponentiation oracle E_x, yielding $\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle |x^k \bmod N\rangle$;
Apply inverse QFT, yielding $\frac{1}{r} |\widetilde{s/r}\rangle |u_s\rangle$;
Measure, yielding $\widetilde{s/r}$;
Split into a continued fraction

Remark. *t scales linearly with the size of the input, and is only relevant to the accuracy of $\widetilde{s/r}$.*

We'll proceed step-by-step, verifying that the states hold. Begin by creating the uniform superposition:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle |1\rangle$$

Now, analyze quantum modular exponentiation:

Fact 3.2. *Consider the following modular exponentiation oracle E:*

$$E_p |x\rangle |1\rangle = |x\rangle |p^x \bmod N\rangle$$

This oracle is implementable on a quantum computer with cost $O(n^3)$.

We apply this once to the state, incurring $O(n^3)$ operations:

$$E_x |\psi\rangle = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle |x^k \bmod N\rangle$$

Lemma 3.1. *Suppose we have the modular multiplication unitary M:*

$$M_x |k\rangle = |kx \bmod N\rangle$$

The eigenstates $|u_s\rangle$ of M are:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} |x^k \bmod N\rangle$$

Proof. Recognize that:

$$\begin{aligned} M_x |u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} M_x |x^k \bmod N\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} |x^{k+1} \bmod N\rangle \\ &= \frac{e^{2\pi is/r}}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi is(k+1)/r} |x^{k+1} \bmod N\rangle \end{aligned}$$

Recall that we have $1 \equiv x^r \bmod N$; so, $|x^r \bmod N\rangle = |1\rangle = |x^0 \bmod N\rangle$. Thus

$$\begin{aligned} &= \frac{e^{2\pi is/r}}{\sqrt{r}} \Big[\sum_{k=1}^{r-1} e^{-2\pi isk/r} |x^k \bmod N\rangle + e^{-2\pi isr/r} |x^0 \bmod N\rangle \Big] \\ &= e^{2\pi is/r} \frac{1}{\sqrt{r}} \Big[\sum_{k=0}^{r-1} e^{-2\pi isk/r} |x^k \bmod N\rangle \Big] \\ &= e^{2\pi is/r} |u_s\rangle \end{aligned}$$

As desired. So, $|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} |x^k \bmod N\rangle$ truly is an eigenvector of M. □

Now that we have a description of the eigenstates of M, we can actually leverage them to describe $|x^l \bmod N\rangle$. Recognize that:

$$M_x^l |1\rangle = |x^l \bmod N\rangle$$

So, if we can decompose $|1\rangle$ into a sum of eigenstates, we can apply M_x and identify $|x^l \bmod N\rangle$. We can, in fact, finding the following equality:

Lemma 3.2 (Decomposition of Modular Exponentiation Into Eigenvectors)**.** *Note that we can actually decompose the following state:*

$$|x^l \bmod N\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{2\pi isl/r} |u_s\rangle$$

Proof. Let's analyze the following quantity:

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1} e^{2\pi isl/r}\,|u_s\rangle = \frac{1}{r}\sum_{s=0}^{r-1} e^{2\pi isl/r}\sum_{k=0}^{r-1} e^{-2\pi isk/r}\,|x^k \bmod N\rangle$$
$$= \frac{1}{r}\sum_{s=0}^{r-1}\sum_{k=0}^{r-1}\exp\left[\frac{2\pi is}{r}(l-k)\right]|x^k \bmod N\rangle$$

To continue with our analysis, we should fix a specific k an analyze the resulting state (we'll implicitly interchange the summation operators, which is valid because these are both finite sequences):

$$\frac{1}{r}\sum_{s=0}^{r-1}\exp\left[\frac{2\pi is}{r}(l-k)\right]|x^k \bmod N\rangle = \frac{1}{r}\sum_{s=0}^{r-1}\exp\left[\frac{2\pi i}{r}(l-k)\right]^s|x^k \bmod N\rangle$$

Let's consider two cases - $l = k$ and $l \neq k$.
If $l = k$, then

$$\frac{1}{r}\sum_{s=0}^{r-1}|x^k \bmod N\rangle = |x^k \bmod N\rangle$$

If $l \neq k$, then this is a typical geometric series. Recalling the formula, we see:

$$\frac{1}{r}\sum_{s=0}^{r-1}\exp\left[\frac{2\pi i}{r}(l-k)\right]^s|x^k \bmod N\rangle = \frac{1}{r}\frac{1-\exp\left[\frac{2\pi i}{r}(l-k)\right]^r}{1-\exp\left[\frac{2\pi i}{r}(l-k)\right]}|x^k \bmod N\rangle$$
$$= \frac{1}{r}\frac{1-1}{1-\exp\left[\frac{2\pi i}{r}(l-k)\right]}|x^k \bmod N\rangle$$
$$= 0$$

So, if $l \neq k$, the term actually vanishes! This is stunning, and allows for the dramatic simplification:

$$\frac{1}{r}\sum_{s=0}^{r-1}\sum_{k=0}^{r-1}\exp\left[\frac{2\pi is}{r}(l-k)\right]|x^k \bmod N\rangle = \sum_{k=l}\sum_{s=0}^{r-1}|x^k \bmod N\rangle$$
$$= |x^l \bmod N\rangle$$

As desired. □

The above lemmas have allowed us to decompose $|x^l \bmod N\rangle$ into a series of more malleable eigenvectors $|u_s\rangle$ which only have a coefficient change when M is applied. In fact, we'd like to convert the entire state over to $|u_s\rangle$:

Lemma 3.3.

$$\frac{1}{\sqrt{2^t}}\sum_{k=0}^{2^t-1}|k\rangle\,|x^k \bmod N\rangle = \frac{1}{\sqrt{r2^t}}\sum_{s=0}^{r-1}\sum_{k=0}^{2^t-1} e^{2\pi isk/r}\,|k\rangle\,|u_s\rangle$$

Proof. Given Lemma 3.2, we can substitute $|x^k \bmod N\rangle$:

$$\frac{1}{\sqrt{2^t}}\sum_{k=0}^{2^t-1}|k\rangle\,|x^k \bmod N\rangle = \frac{1}{\sqrt{2^t}}\sum_{k=0}^{2^t-1}|k\rangle\,\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}e^{2\pi isk/r}\,|u_s\rangle$$
$$= \frac{1}{\sqrt{r2^t}}\sum_{s=0}^{r-1}\sum_{k=0}^{2^t-1}e^{2\pi isk/r}\,|k\rangle\,|u_s\rangle$$

As desired. □

At this point, we have expressed our modularly exponentiated state in terms of eigenvectors, each which has a coefficient. Unfortunately, this coefficient vanishes when we measure the state; so, we'll leverage a technique known as Quantum Phase Estimation:

Definition 3.1 (Quantum Phase Estimation)**.** ***Quantum Phase Estimation (QPE)*** *allows the eigenvalues of unitaries to be determined efficiently. Consider a unitary U with eigenvector $|v\rangle$. Suppose*

$$U\,|v\rangle = e^{2\pi i\theta}\,|v\rangle$$

Then, there is an algorithm to estimate θ given $O(t^2)$ operations with probability exceeding $1-\epsilon$ where $t = n + \lceil \log(2+\frac{1}{2\epsilon})\rceil$ with n the number of bits of precision [2, p. 225].

Interestingly, the algorithm actually creates a *binary decimal representation* of θ! This is completely unexpected from a discrete system. The key component of QPE that yields this result is the inverse of the Quantum Fourier Transform. First, we define the Quantum Fourier Transform:

Definition 3.2 (Quantum Fourier Transform)**.** *The* ***Quantum Fourier Transform (QFT)*** *is a unitary operation that implements the discrete Fourier Transform. Consider a space of vectors from $|0\rangle ... |N-1\rangle$ where $N = 2^n$. Then,*

$$QFT\,|j\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{n-1}e^{2\pi ijk/N}\,|k\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{n-1}\omega_N^{jk}\,|k\rangle$$

Where ω_n is the nth root of unity.

Remark. *QFT is straightforward, but notable because it performs exponentially faster than on a classical computer. While QFT scales in $\Theta(n^2)$ operations, classical algorithms can implement the Fast Fourier Transform in $\Theta(n2^n)$ gates [2, p. 220]. This advantage allows QPE to be much quicker on a quantum computer.*

So, we define the inverse QFT or QFT^{-1} so that $QFT^{-1}QFT = \mathbb{1}$. Now, applying the inverse QFT on the first register:

$$QFT^{-1}\frac{1}{\sqrt{r2^t}}\sum_{s=0}^{r-1}\sum_{j=0}^{2^t-1}e^{2\pi isj/r}\,|j\rangle\,|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\left(QFT^{-1}\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^t-1}e^{2\pi isj/r}\,|j\rangle\right)|u_s\rangle$$
$$= \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|\widetilde{s/r}\rangle\,|u_s\rangle$$

Where $\widetilde{s/r}$ is the binary decimal representation that is close to s/r. Thus, when we measure the register, we yield a decimal approximation of s/r with t digits of precision. Let's say the true value of $s/r = \varphi$; how can we reconstruct φ? (And, by proxy, r).

Problem 3.3. *Given an n bit binary decimal representation of a rational $\varphi = \frac{a}{b}$, how do we identify a, b?*

Theorem 3.2 (Continued Fraction Algorithm)**.** *We can identify some* $\frac{c}{d}$ *where*

$$\left|\frac{c}{d} - \frac{a}{b}\right| \leq \frac{1}{2d^2}$$

Given $O(n^3)$ *operations.*

Remark. *This theorem is also omitted, as it lies outside of quantum computing. This is actually a classical processing technique.*

Theorem 3.2 is a handy result that allows us to measure our original state, yielding $|\widetilde{s/r}\rangle$. With that, we can identify some s', r' where $\frac{s'}{r'}$ close. So, return r'. □

Remark. *Even though order-finding can be performed classically, it is believed to be 'hard,' exceeding* $O(n^k)$ *where* n *is the number of bits used to express* N*.*

Remark. *We have actually implicitly applied QPE in Grover's by finding the phase introduced by the modular exponentiation unitary.*

For keen readers, they may recognize that the r' might be incorrect; this could emerge because $s = 0$ or has common factors with r. However, this chance actually vanishes if we repeat the algorithm $O(\log N)$ times:

Fact 3.3. *While a single iteration of the oracle finding algorithm might fail because* $\gcd(s, r) \neq 1$ *(as the state is in superposition), repeating the computation* $O(\log N)$ *times returns the order with high probability [2, p. 231].*

3.2.2 Reducing Factoring to Order-Finding

While order-finding is neat, the problem does not seem immediately applicable. However, if we can reduce factoring to order-finding with minimal cost, we can utilize the speedup in order-finding to accelerate factoring. In fact, we can reduce the problem [2, p. 233]. Consider:

Fact 3.4 (Factoring is order-finding)**.** *Suppose* N *is an odd composite number that cannot be expressed as* a^b *where* $a \geq 1$ *and* $b \geq 2$*. Then, there exists at least 4* x *such that* $x^2 \equiv 1 \bmod N$ *by the Chinese Remainder Theorem.*

If we can identify such x*, then either* $\gcd(x - 1, N)$ *or* $\gcd(x + 1, N)$ *are nontrivial factors of* N*, and they can be computed with* $O(n^3)$ *operations.*

Essentially, if we can identify the modular square roots of N, we could identify factors. Additionally,

Fact 3.5. *Suppose* N *has a prime factorization* $p_1^{k_1} p_2^{k_2} ... p_n^{k_n}$ *where* p_l *prime and* $k_1 \neq 0$*. Let* x *be chosen randomly from* $1 \leq x \leq N - 1$ *and* x *co-prime to* N*. Let* r *be the order of* $x \bmod N$*. Then,*

$$\mathbb{P}(r \text{ even} \wedge x^{r/2} \not\equiv -1 \bmod N) \geq 1 - \frac{1}{2^m}$$

Remark. *These proofs are omitted as they are more pertinent to modular arithmetic than quantum algorithms. They are found here [2, p. 663].*

In essence, we can decompose factoring into an order-finding problem. We simply need the quantum computer to identify even orders, and our classical computer can check if the potential factors provided actually work.

3.2.3 Proof

Now that we can 1) reduce factoring to order-finding and 2) solve order-finding quickly, we can produce an algorithm:

Theorem 3.3. *A quantum algorithm can find factors with high probability given $O(n^3)$ operations.*

Proof. Let's leverage the following algorithm [2, pp. 233, 234]:

Algorithm 4: Quantum Factoring (Shor's Algorithm)

Result: x, y known with probability in $O(1)$
If N even, return 2;
If $N = a^b$ for $a \geq 1$, $b \geq 2$, return a;
Randomly choose x from 1 to $N - 1$. If $\gcd(x, N) > 1$, return x;
Using the order-finding quantum algorithm, identify r where $x^r \equiv 1 \bmod N$;
If r even and $x^{r/2} \neq -1 \bmod N$, then compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$ and see if either are non-trivial factors;
If none are factors, repeat;

Now, let's evaluate each step individually for its asymptotic cost -

1. We can check if a number is even in constant time; check the last bit. $O(1)$.
2. We take it it as a fact that $N = a^b$ for $a \geq 1$ and $b \geq 2$ can be evaluated in $O(n^3)$ time (again, this classical algorithm is outside the scope of our quantum analysis).
3. Order-finding operates in $O(n^3)$ by Theorem 3.1.

Thus, our overall algorithm is $O(n^3)$, as desired. □

Remark. *Even though the order-finding subroutine is not deterministic, there is emerging work in using classical processing to identify r with reduced cost on a quantum computer.*

3.2.4 Review

In the above section, we've produced a loose explanation of how a number could be factored with a quantum computer. We first reduced the factoring problem to order-finding, then leveraged QPE and the inverse QFT to quickly find the order.

Shor's algorithm demonstrates how speedups in seemingly unrelated problems can translate into dramatic speedups in other environments. Because QPE / QFT are both implementable on quantum computers with asympotically less cost than classical computers, we were able to scale those benefits to order finding, and thus factoring.

4 Conclusion

In this paper, we've performed brief forays into two notable quantum algorithms: Grover's search and Shor's algorithm. We've detailed fundamental results and compared asymptotic costs to their classical counterparts.

Grover's search yielded surprising bounds that require a quantum computer to reference a database in sublinear time (vs linearly on a classical computer). We also discussed how implementation of the marking oracle is feasible, even when the marked bitstring is unknown. This culminated to a discussion in min-finding.

Shor's algorithm posed an exponential speedup over classical algorithms by reducing the problem of factoring to order-finding. We utilized tools for phase estimation, a uniquely quantum phenomenon, to perform order-finding faster than classical computers. Finally, the role of eigenstates and eigenvalues of unitaries was detailed through our exploration of QPE.

I hope these results and exposition serve to demonstrate the quirky and fascinating potential of quantum algorithms. For further reading, I recommend looking at the Quantum Approximate Optimization Algorithm [4] and chemical simulations via a technique called Trotterization [7]. I personally find quantum computation for chemistry fascinating, as it could catalyze leaps in understanding and modeling our world.

References

[1] Gilles Brassard et al. *Quantum Amplitude Amplification and Estimation*. 2000. arXiv: quant-ph/0005055 [quant-ph].

[2] Isaac Chuang and Michael A Nielsen. *Quantum computation and quantum information*. 2002.

[3] Christoph Durr and Peter Hoyer. *A Quantum Algorithm for Finding the Minimum*. 1996. arXiv: quant-ph/9607014 [quant-ph].

[4] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[5] Anatolii Grinshpan. *THE PARTIAL SUMS OF THE HARMONIC SERIES*. URL: https://www.math.drexel.edu/~tolya/123_harmonic.pdf.

[6] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph].

[7] James D. Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. "Simulation of electronic structure Hamiltonians using quantum computers". In: *Molecular Physics* 109.5 (Mar. 2011), pp. 735–750. ISSN: 1362-3028. DOI: 10.1080/00268976.2011.552441. URL: http://dx.doi.org/10.1080/00268976.2011.552441.