Quantum Computing and Shor's Algorithm

Boxian Wang

June 5, 2022

1 Introduction

Quantum computing is a computational model that uses a hypothetical quantum computer capable of performing basic operations on a quantum system. It is a probabilistic model which utilizes the stochastic nature of quantum mechanics. Not much is known about the exact powers and limitations of quantum computers, especially whether they can solve NP-complete problems efficiently. However, the existence of quantum algorithms that offer non-trivial speedups compared to best known classical algorithms provides some evidence that the quantum model is more powerful than the classical probabilistic model.

In this paper we provide one such non-trivial quantum algorithm. No classical algorithm has been found to factor an integer n in polynomial time (the input size being $\log n$), with the best known algorithm taking roughly $2^{(\log n)^{1/3}}$ time [4]. In this paper we present a quantum algorithm by Shor [6] that solves integer factorization in poly$(\log n)$ time .

In section 2 we present the definition of quantum computation and some basic consequences. In section 3 we introduce two tools integral in implementing Shor's algorithm. The first is some number theoretic facts connecting multiplicative order and factorization. The second is discrete Fourier transform and its efficient quantum implementation. In section 4 we present Shor's algorithm of finding the multiplicative order of a number and analyze its success probability. In section 5 we discuss the classification of quantum states as an attempt to differentiate between quantum states that are practically realizable and those that are necessary to implement Shor's algorithm.

2 The Quantum Model

In this section we introduce basic concepts in quantum computation. We mostly follow the fomulation in Arora-Barak [2].

2.1 Basics; Quantum Registers

A basic postulate of quantum mechanics is each quantum system has a **quantum state** that contains all information about the system. This quantum state is a **normalized** vector in the complex Hilbert space, often expressed as $|\phi\rangle$.

The simplest example is a **qubit** system, where the Hilbert space has dimension 2. Let $|0\rangle$ and $|1\rangle$ be an orthonormal basis of the Hilbert space, then the general quantum state of the qubit is $|\phi\rangle = c_0 |0\rangle + c_1 |1\rangle$, where $c_1, c_2 \in \mathbb{C}$ and $|c_0|^2 + |c_1|^2 = 1$.

Thus, if $|0\rangle$ and $|1\rangle$ corresponds to certain observable physical properties (e.g. the spin of a particle), a quantum system can be in a mixture of two states, called a **superposition** state. However, upon observation, the system **collapses** to a single state $|i\rangle$, with probability $|c_i|^2$. Therefore,

a qubit system does not store a definite bit of information, but a probabilistic mixture of both 0 and 1.

As such, the power of quantum computation comes from utilizing the probabilistic nature of a quantum system. Let us first expand a single qubit into a quantum register that can accommodate values of arbitrary lengths. A quantum register of length n is just a quantum state with 2^n basis states, namely $\{|a_{n-1}\dots a_0\rangle, a_i \in \{0,1\}\}$. Note that the actual state of the register is a mixture of the 2^n basis states, and can be represented as a normalized vector in $\mathbb{C}^{2^n}$. (For the mathematically inclined, this is the space resulted by tensoring the qubit space n times.)

Definition 2.1. *A n-bit quantum register is a vector $\mathbf{v} \in \mathbb{C}^{2^n}, |\mathbf{v}| = 1$. If we index $\mathbf{v}$ by $i \in \mathbb{Z}_{2^n}$, the content of the register, then upon measuring the register, the probability of reading $|i\rangle$ is $|v_i|^2$.*

Since all quantum states are normalized, sometimes we use a non-unit length vector to represent a quantum state and normalization is implied.

2.2 Quantum Operations

Having defined a n-bit register as a 2^n dimensional vector, next we need to define valid operations on this register. It is another axiom of quantum mechanics that quantum states can only be transformed via **unitary matrices**. A unitary matrix A is defined over complex matrices such that $AA^* = I$, where A^* is the Hermitian adjoint, i.e. conjugate transpose of A. Unitary matrices have the property that it always sends unit vectors to unit vectors, so applying such a matrix to a quantum state always gives us another well-formed quantum state. Also note that a unitary matrix is by definition invertible; thus quantum operations are all reversible.

A simple but important example of quantum operation is the **Hadamard** operator. It acts on a single qubit via

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

It send $|0\rangle \to |0\rangle + |1\rangle$, $|1\rangle \to |0\rangle - |1\rangle$.

Theoretically, any unitary matrix can be a valid quantum operation. In practice, it is very difficult to implement an operation involving multiple qubits. Thus in defining the computation model we use the following definition instead.

Definition 2.2. *An elementary quantum operation is a unitary matrix acting on three or less qubits.*

For example, if U acts on two qubits, then applying U to a quantum state $|a_n \dots a_0\rangle$ at the location i yields $|a_n \dots a_{i+2}\rangle\, U(|a_{i+1}a_i\rangle)\, |a_{i-1}\dots a_0\rangle$.

It can be shown that our arbitrary choice of three qubits is nonessential.

2.3 Quantum Computing and BQP

We now define quantum computation and its associated complexity class. The idea should be straightforward now: The input is given as one of the basis states of the quantum register. It then goes through a series of quantum operations (gates). The output is then measured and we want it to be correct with high probability.

Definition 2.3. *A language is $T(n)$-quantum computable if*

0. *Given an input x of length n, a (classical) TM computes at most $T(n)$ elementary quantum gates in polynomial time. (Uniformity)*

1. *Input is given in* $|x\rangle |0^m\rangle$. *(The trailing zeroes are scratch spaces.)*

2. *Gates* $U_1, U_2, \ldots$ *are applied to the register sequentially.*

3. *The first bit of the result is measured, for accepting or rejecting. The result should be correct with probability* $> 2/3$.

The class of languages that are polynomial-time-quantum computable is denoted class ***BQP****.*

We ask the reader to believe that this definition is robust, and that adding such features as allowing intermediate measurements of the register does not change the computational power of the model.

However, we will not be so pedantic as to restrict quantum computation to decision problems, as prime factorization requires returning a factor. Also, we may interleave quantum and classical computation in presenting out algorithm. This presents no problem formally, as we shall see that classical computation can be simulated by quantum computation.

2.4 Relation to the Classical Models

There are several similarities to classical models to point out here.

First, note that elementary quantum operations are similar to classical Boolean gates. However, the major difference is that each quantum gate can only have one fan-out while Boolean gates can have many.

In fact, any Boolean circuit can be simulated by a similar-sized quantum circuit. Note that we cannot directly implement AND: $|b_1 b_2\rangle \to |(b_1 \wedge b_2) b_2\rangle$, as the operation is not invertible. However, we can utilize a new bit b_0 that is presumed to be zero to store the result:

$$|b_0 b_1 b_2\rangle \to |(b_0 \oplus (b_1 \wedge b_2)) b_1 b_2\rangle$$

(As an aside, this is reminiscent of the 'static single-assignment form' in compiler optimization, but I digress.) This operation is reversible. NOT and OR gates can be similarly implemented. To deal with multiple fan-outs, we can copy the register onto scratch space in a write-once fashion. For more details, see Arora-Barak [2]. As such, we may implement a Boolean circuit with quantum gates.

Since we can simulate a computation in **P** with polynomially many Boolean gates, we get $\mathbf{P} \subseteq \mathbf{BQP}$.

Second, let us consider the difference between **BQP** and **BPP**. It can easily be proved that $\mathbf{BPP} \subseteq \mathbf{BQP}$, since we can simulate coin tosses by measuring a qubit. It is an open problem whether the inclusion is proper, although there is evidence against it, with Shor's algorithm being a prominent contribution. The power of **BQP** seems to come from its ability to use complex coefficients in transforming and evaluating probabilistic vectors [2].

Theorem 2.4.

$$\boldsymbol{P \subseteq BPP \subseteq BQP}$$

3 Classical and Quantum Tools

In this section we develop some necessary tools in both classical and quantum computation that will help us formulate Shor's algorithm.

3.1 Factoring and Multiplicative Order

We first define multiplicative order.

Definition 3.1. *The order of $x \pmod n$ is the smallest positive integer r s.t. $x^r \equiv 1 \pmod n$.*

We will only use the following basic properties of order from elementary number theory.

Lemma 3.2. *Let r be the order of $x \pmod n$. Then*

1. *$r|\phi(n)$, where ϕ is the Euler totient.*
2. *$x^a \equiv x^b \pmod n$ if and only if $a = b + kr$ for some integer k.*

We can find a factor of n with high probability of success by finding the order for a randomly selected x. The following two lemmas show how this is possible.

Lemma 3.3. *If $y^2 \equiv 1 \pmod n$ but $y \neq \pm 1 \pmod n$, then $1 < gcd(y-1, n) < n$.*

Proof. The assumption says $n|(y-1)(y+1)$ but n does not divide either $y-1$ or $y+1$. This suggests n and $y-1$ cannot be coprime, and n itself is not the gcd. The lemma follows. □

Lemma 3.4. *Given odd n not prime and not a prime power with k distinct prime factors, if we choose $x \in \mathbb{Z}_N^*$ coprime with n uniformly at random, then with probability at least $1 - 2^{-k+1}$ the following is true:*

1. *x has even order r.*
2. *$x^{r/2} \neq -1 \pmod n$.*

Proof. Let $n = \prod p_i^{\alpha_i}$ be its factorization. Then the order of x is equal to the least common multiple of the order of x modulo each of $p_i^{\alpha_i}$. Furthermore, by the Chinese remainder theorem, x modulo n corresponds uniquely to its modulo of each coprime factor, so choosing x at random is equal to choosing all its modulos at random.

Let r_i be the order of x modulo $p_i^{\alpha_i}$. Then our condition fails only if all r_i's contain the same factor of 2. But, since $(\mathbb{Z}/(p_i^{\alpha_i})\mathbb{Z})^*$ is cyclic, the probability is at most $1/2$ of choosing an x having a particular power of 2 in its order. Then the probability of all k orders having the same power of 2 is $2^{-(k-1)}$, so with $1 - 2^{-(k-1)}$ probability the condition above is satisfied. □

Therefore if we can compute the order of x efficiently, we may use the following randomized algorithm to find a factor of n. First, pick x uniformly at random. Then, compute $gcd(x, n)$ using Euclid's algorithm. If they are not coprime, we have found a factor of n. If they are coprime, we then compute the order r. By lemma 3.4, with high probability, r is even and $x^{r/2} \neq -1 \pmod n$. Then by lemma 3.3, $1 < gcd(x^{r/2} - 1, n) < n$. Then we may use Euclid's algorithm to find the gcd, which is a factor of n. Having found a factor of n, we divide n by it and recurse.

If finding one factor takes $\text{poly}(\log n)$ time, since there are at most $\log n$ factors, the factorization takes $\text{poly}(\log n)$ time.

3.2 Quantum Fourier Transformation

Thus we have reduced the problem of factorization to find the multiplicative order of $x \pmod n$. To do so we need the help of discrete Fourier transformation. We will show how this can be efficiently implemented in quantum gates. The following proofs are adapted from Shor's original paper [6].

Definition 3.5. *For $f \in \mathbb{C}^n$, its Fourier transform $\hat{f} \in \mathbb{C}^n$ is defined as $\hat{f}(x) = \frac{1}{\sqrt{n}} \sum_{k=1}^{n} f(k)\omega^{xk}$, where $\omega = \exp(2\pi i/n)$ is the n-th root of unity.*

Remark. *If we define $\chi_x : \chi_x(k) = \frac{1}{\sqrt{n}}\omega^{-xk}$, then the χ_x's form an orthonormal basis, as you can check. Thus $\hat{f}$ is just the coordinate of f under such a basis.*

The discrete Fourier transformation lies at the heart of many modern algorithms, and using fast fourier transformation (FFT), a classical TM is able to do the transformation in $O(n \log n)$ time. The rest of this section is dedicated to presenting a quantum algorithm that only uses $O(\log n)$ gates.

Theorem 3.6. *Let $n = 2^l$. There exists a $O(l^2)$ sized quantum circuit that transforms $f = \sum f(x) |x\rangle$ to $\hat{f} = \sum \hat{f}(x) |x\rangle$ where $\hat{f}(x) = \frac{1}{\sqrt{n}} \sum_{k=1}^{n} f(k)\omega^{xk}$.*

As evident from our remark, $f \to \hat{f}$ is essentially a linear transformation whose matrix we can specify by looking at the transformation applied to basis elements, i.e. $|x\rangle$. Then

$$|x\rangle \to \frac{1}{\sqrt{n}} \sum_{k=1}^{n} \omega^{xc} |c\rangle$$

Hence the matrix has $\frac{1}{\sqrt{n}}\omega^{xc}$ has its (c, x) entry. This is a unitary matrix, and the problem now is to implement it using elementary quantum gates.

Let bits be numbered right to left, so that $|a\rangle = |a_{l-1} \dots a_0\rangle$. Recall the Hadamard operator $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Let H_j be the Hadamard operator acting on bit j. Let $S_{j,k}$ act on bits $j < k$, and

$$S = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \exp\bigl(i\pi/2^{k-j}\bigr) \end{pmatrix}$$

That is, S alter the phase for $|11\rangle$ and leave the rest unchanged.

Now we claim the following sequence of gates, applied from left to right, implements Fourier transform. For each bit l:

- Place $S_{l,k}$ for each $k > l$
- Place H_l

The end result is something like

$$H_{l-1} S_{l-2,l-1} H_{l-2} S_{l-3,l-1} S_{l-3,l-2} R_{l-3} \cdots$$

Note that applying the above actually gives $\frac{1}{\sqrt{n}} \sum_{k=1}^{n} \omega^{xc} |b\rangle$, where b is the bit reversal of c, but we can then easily reverse the bits.

To show that the gates correctly implement Fourier transform, Consider applying it to $|a\rangle = |a_{l-1} \dots a_0\rangle$ and we study the resulting component of $|b\rangle = |b_{l-1} \dots b_0\rangle$. We shall only verify the phase is correct.

Apparently $|b\rangle$ comes from applying H_j at each bit, which splits each bit to the superposition of both $|1\rangle$ and $|0\rangle$. If a_j and b_j are both 1, this also adds π to the phase as it reverses the sign. Furthermore, S adds $\pi/2^{k-j}$ to the phase for each a_j, b_k where $j < k$.

Thus the total phase change is

$$\sum \pi a_j b_j + \sum_{j<k} \frac{\pi}{2^{k-j}} a_j b_k = \sum_{j \le k} \frac{\pi}{2^{k-j}} a_j b_k = \sum_{j \le k} \frac{\pi}{2^{k-j}} a_j c_{l-1-k}$$

Substituting k for $l - 1 - k$ yields

$$\sum_{j+k<l} 2\pi \frac{2^j 2^k}{2^l} a_j c_k$$

But it is okay to let $j + k \ge l$ since it only adds multiples of 2π to the phase, hence

$$\sum_{j,k} 2\pi \frac{2^j 2^k}{2^l} a_j c_k = \frac{2\pi}{2^l} (\sum 2^j a_j)(\sum 2^k c_k) = 2\pi ac/2^l = 2\pi ac/n$$

And that is exactly the phase of ω^{ac} as specified earlier in the matrix element. Thus the $O(l^2)$ gate successfully implements our specified matrix.

4 Shor's Order Finding Algorithm

We have seen that once we are able to find the multiplicative order of $x \pmod n$ in $\text{poly}(\log n)$ time, factorization is $\text{poly}(\log n)$ as well. Now we present the central piece of Shor's algorithm, Shor's order finding algorithm. The following proofs use a combinantion of techniques from Shor [6] and Arora-Barak [2].

Theorem 4.1. *There is a quantum algorithm that finds the order of $x \pmod n$ with high probability in $\text{poly}(\log n)$ time.*

The algorithm is rather simple but the analysis is relatively involved. It comes in three main steps

1. Choose q a power of 2 between n^2 and $2n^2$. Prepare the register first in uniform state $\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle |0\rangle$ with q base states. Then, fill in the zero register with $|x^a \pmod n\rangle$ using fast exponentiation in $O(\log n)$ time.

2. Perform quantum Fourier transform on the first part of the register, mapping $|a\rangle$ to $\frac{1}{\sqrt{q}} \sum \omega^{ac} |c\rangle$.

3. Measure the register (or just the first part) to get $|c\rangle$. Using continuous fraction technique, find (d, r) coprime such that $|\frac{d}{r} - \frac{c}{q}| < \frac{1}{2q}$. Verify if $x^r = 1 \pmod n$.

If the verification fails at the last step, we simply repeat the measurement. We shall see in our analysis that by repeating no more than $\text{poly}(\log n)$ time, we are guaranteed a high probability of success.

Remark. *For details on fast exponentiation and continuous fractions, see Shor [6] and Arora-Barak [2] resp.*

Let us now analyze the probability of success of the algorithm above. The Fourier transform in step 2 leaves the machine in the following state:

$$1/q \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} \omega^{ac} |c\rangle |x^a \pmod n\rangle$$

Now the amplitude of measuring a particular $|c\rangle \left|x^k \pmod n\right\rangle$ state is

$$\left| 1/q \sum_{x^a \equiv x^k} \omega^{ac} \right|^2$$

But if the order of x is r, $a = k + br$ for some b. So the amplitude is

$$\left| 1/q \sum_{b=0}^{(q-k-1)/r} \omega^{(k+br)c} \right|^2$$

We can factor out ω^{kc}:

$$1/q \cdot \left| \sum_{b=0}^{(q-k-1)/r} \omega^{brc} \right|^2$$

We can also replace rc with $\overline{rc}$ where $rc \equiv \overline{rc} \pmod q$ and $|\overline{rc}| \leq q/2$.

$$1/q \cdot \left| \sum_{b=0}^{(q-k-1)/r} \omega^{b\overline{rc}} \right|^2$$

Let $K = \lceil q/r \rceil$. We evaluate the geometric series, letting $\omega^{\overline{rc}} = \alpha$. Then the amplitude is at least

$$1/q \cdot \left| \sum_{b=0}^{K} \alpha^b \right|^2 = 1/q \cdot \left| \frac{1-\alpha^K}{1-\alpha} \right|^2 = 1/q \cdot \left| \frac{\sin(K\theta/2)}{\sin(\theta/2)} \right|^2$$

where $\theta = \overline{rc}/q$ is the argument of α.

Now, if we stipulate that $|\overline{rc}| < r/2$, then $|K\theta/2| = |\lceil q/r \rceil \overline{rc}/q/2| < 1/4$ is small, so $\sin x \sim x$ works reasonably well. (More careful analysis using power series is possible but not consequential.) Thus the amplitude becomes

$$1/q \cdot (\lceil q/r \rceil)^2 = O(1/r^2)$$

To summarize, if $|\overline{rc}| < r/2$, then there is at least $O(1/r^2)$ chance that we measure $|c\rangle$. However, $|\overline{rc}| < r/2$ translates to $-r/2 \leq rc - dq \leq r/2$, which is

$$\left| \frac{c}{q} - \frac{d}{r} \right| < \frac{1}{2q}$$

Since $q > n^2$, there is at most one $r < n$ that satisfies such property. If such r exists, then then continuous fractions technique allows us to find it by finding the best fraction approximation with denominator less than n.

Now the idea is clear: we can expect high probability of success because the probability of hitting a $|c\rangle$ for which the closest approximation returns the correct r as denominator is high.

Let us know calculate the number of states $|c\rangle |x^a \pmod n\rangle$ for which this works. For each d coprime to r, there is clearly a unique c for which $|c/q - d/r| < 1/2q$. Thus this gives $\phi(r)$ choices for c. There are r choices for the exponent of x, so a total of $\phi(r)r$ states. Each of them has $O(1/r^2)$ chance of getting measured, so the overall probability of success is $O(\phi(r)/r)$. Utilizing a fact in analytical number theory, $O(\phi(r)/r) = O(1/\log\log r)$ [3]. Thus it suffices to repeat the measurement $O(\log\log n)$ times. (I omit the finer probabilistic analysis, but note that this is intuitively the expected number of repetition before a success, and adding a constant factor can comfortably boost the success rate past $\frac{2}{3}$.)

5 Classifying Quantum States

There has been skepticism of whether the complex quantum states used in Shor's algorithm are physically realizable. To address this, Aaronson [1] devised a complexity measure on quantum states based on *tree sizes*. A tree represents the 'tensor-addition' construction of a quantum state in a tree-like manner, with internal vertices of either $+$ or $\otimes$ and $|0\rangle$ or $|1\rangle$ as leaves. I only provide a definition that is intuitively clear; for finer details, consult Aaronson's original paper.

Definition 5.1. *A tree representation of a state $|\phi\rangle$ is a tree whose internal vertices are either $+$ or $\otimes$ and whose leaves are either $|0\rangle$ or $|1\rangle$. For $+$ nodes coefficients are also specified. $|\phi\rangle$ must result from evaluating the tree in the obvious way from the bottom up, using addition and tensoring. Note that the bit position of various tree branches must be consistent. Then the tree size of $|\phi\rangle$, $TS(|\phi\rangle)$, is defined as the minimum number of nodes among all tree representation of $|\phi\rangle$. Define the class* TREE *as the set of n-bit quantum states with tree sizes polynomial in n.*

Aaronson argues that the class TREE is a good candidate for separating what can be physically achieved and what might not be possible, at least in theory. In particular, he proved that (randomly chosen) quantum states appearing in quantum error-correcting codes and Shor's algorithm has non-polynomial tree size lower bounds (with high probability), hence might not be actually physically feasible.

To prove these claims, a connection between quantum trees and multilinear polynomials is made.

Definition 5.2. *A polynomial can be represented as a tree with $+, \times$ as internal nodes and variables x_i or complex numbers as leaves. The polynomial is then the evaluation of the tree in the obvious way. A multilinear polynomial is where each variable has power at most 1. Then the tree size of a multilinear polynomial p, $MFS(p)$, is the minimal amount of nodes among all its tree representations.*

And for an n-ary function $f : \{0,1\}^n \to \mathbb{C}$, its tree size is just the minimal tree size of any multilinear polynomial over $\mathbb{C}$ that takes the same value as f.

For $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$, let $f_\phi(x) = \alpha_x$. The following relationship between the tree size of f and $|\phi\rangle$ is proved:

Lemma 5.3. $MFS(f_\phi) \leq 6TS(|\phi\rangle)$

Now that we have introduced some algebra into the picture, a theorem by Raz and its corollaries [5, 1] can help us prove our stated lower bounds. We omit the algebraic details here but refer the reader to Aaronson's original paper.

Finally, for the big theorem on lower bounds of quantum complexity concerning Shor's algorithm, let 'Shor states' be of the form $|a + p\mathbb{Z}\rangle = w^{-1/2} \sum_{i=0}^{w} |a + pi\rangle$, where $a < p < 2^n$, p prime, $w = \lfloor (2^n - a - 1)/p \rfloor$. This is the state of the register after measuring the second half, before the Fourier transform. It is also shown that for lower bound purposes, it suffices to consdier only $a = 0$. However, the following theorem requires a number theoretic conjecture, which we omit here but again refer the reader to Aaronson's paper.

Theorem 5.4. *Given the conjecture, if p were to be selected uniformly at random, then with $\Omega(1)$ probability, $TS(|p\mathbb{Z}\rangle) = n^{\Omega(\log n)}$.*

Thus, at least in this sense, quantum states required in Shor's algorithm are different from simpler and physically realizable states.

References

[1] Aaronson, S. Multilinear formulas and skepticism of quantum computing. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing* (2004), pp. 118–127.

[2] Arora, S., and Barak, B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[3] Hardy, G. H., Wright, E. M., et al. *An introduction to the theory of numbers*. Oxford university press, 1979.

[4] Lenstra, A. K., Lenstra Jr, H. W., Manasse, M. S., and Pollard, J. M. The number field sieve. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing* (1990), pp. 564–572.

[5] Raz, R. Multi-linear formulas for permanent and determinant are of super-polynomial size. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing* (2004), pp. 633–641.

[6] Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review 41*, 2 (1999), 303–332.