Shor's Algorithm

Elliott Pryor, Benjamin Bushnell, Shengnan Zhou

18 November, 2019

For our +1 we've decided to walkthrough Shor's algorithm using a classical approach. We will implement Shor's algorithm on a classical computer. This will show that Shor's algorithm works at factoring numbers. The classical implementation will take longer than on a quantum computer, but the logic is the same on any computer. The reason we chose this +1 is because we found it impossible to improve upon Shor's algorithm, and unreasonable compare Shor's algorithm to other existing factoring methods as they all function on a classical computer. Our +1 provides context for how Shor's algorithm functions on real world numbers. In this paper we will walk through an example to provide insight into how Shor's algorithm works. This gives a high level overview of how the classical algorithm will work.

1 Introduction

Shor's algorithm was invented in 1994, and it cleverly uses properties of quantum computing to quickly solve prime factorization of even large numbers. This is exactly what RSA assumes is not possible. The short description of Shor's algorithm is that we can take any random guess g, run the guess through Shor's algorithm, and get back a better guess $g^{p/2} \pm 1$. For any large number N, when we take a random guess g, $mN = g^p + 1$ for some p and m. Break this down, we get $(g^{p/2} + 1)(g^{p/2} - 1) = mN$. Once we have the two better guesses, we can use Euclid's algorithm to find the shared factors, thus breaks the encryption. However, the most important part of this algorithm is to find p, and it takes very very long time without quantum computing. Quantum computing can take in multiple values as superpositions, and find p fast.

For sections **??** and **??** assume we are trying to find:

$$ab = C$$

where a and b are factors of N and $a, b, C \in \mathbf{N}$.

2 Classical Computation

The classical part of this algorithm is really simple. We just have to guess a random number as our factor. First we run Euclid's algorithm on the guess to make sure that the GCD of the guess and N is 1. If the GCD is 1 then we know that our guess and N are co-prime, we then feed our guess into Shor's algorithm to improve our guess. If the GCD of the guess and N is not 1, then we guessed a factor (or multiple of a factor) and we do not need to use the quantum portion.

3 The Quantum Algorithm

The quantum part of this algorithm is what turns the random number that we guessed into the actual factor. In short, it does this by finding the period of some function that we can relate to the factors. Given that that is a gross oversimplification of what happens, we will cover some of the math supporting Shor's Algorithm.

First, we must state a few known relations.

Euler's Theorem, as seen in Equation **??**, provides a relation given two co-prime integers, a, n. In Equation **??**, r represents the order of a in the multiplicative group. r is the Euler-Totient function, but for our case we can consider it some integer.

$$a^r \equiv 1(\text{mod n}) \tag{1}$$

Given $a^x = mN + r$ for $a, x, m, N, r \in \mathbf{Z}$. Then equation **??** holds for some $y, p, k \in \mathbf{Z}$.

$$a^{x+yp} = kN + r \tag{2}$$

We know that the guess given to the quantum portion of this algorithm, g, does not share any factors with the number we are trying to factor, C. Then by Euler's theorem (Theorem **??**) we can express this as $r^p = mC + 1$. We can rearrange this to $(r^{p/2} + 1)(r^{p/2} - 1) = mC$. Then we know that our factors of C are related to $(r^{p/2} + 1)$ and $(r^{p/2} - 1)$. Once we find these numbers, we can use Euler's formula to calculate the factors.

In order to find p we use theorem **??** and some properties of quantum computers. By theorem **??**, we know that $g^{x+yp} = kC + r \text{ mod(C)} = r \ \forall y \in \mathbf{Z}$. So we can raise our guess to integer powers and search for when the remainder repeats. Classically, we cannot do this efficiently. However, with quantum computers we have a superposition of states that we can exploit to efficiently compute this. If we poll a random remainder value from our modulus calculation we get a superposition of the states that result in this value: g^x, g^{x+p}, g^{x+2p}... This has a period of p which we can by taking the Fourier transform of this superposition. The Fourier transform returns the period of the function [**?**]. Assuming that p is even, then we are done! We have now found the factors of C. If p is odd, then $r^{p/2}$ is not an integer, and we have to restart the process with a new guess.

4 The +1 Outline

In the +1 we implement Shor's algorithm on a classical computer. The main parts of this are:

1. Euclid's Algorithm for GCD/finding the guess
2. Period finding

Euclid's algorithm for finding the greatest common divisor (GCD) has been known for centuries. We will implement this algorithm in our final +1 solution.

The second part is where the quantum computer would have significant advantages. We can find the period of the function classically, but it will require more time. Once we find the period we can continue with

Shor's algorithm as normal. The period finding approach that we will use classically will involve checking each value until we detect repetition. This will take time proportional to the period of the function.

We will run this algorithm on several numbers to show that it successfully factors these numbers and walk through a couple of examples showing (in-detail) how the algorithm arrives at its conclusions.

5 Example

To better understand how Shor's algorithm works, let's walk through an example.

Let's say $N = 21$, and we want to find a, b such that $N = a \cdot b$ and a, b are prime numbers. First, we take a random guess $g = 11$. We know that the GCD of 11 and 21 is 1, so we proceed to turn g into a better guess $g^{p/2} \pm 1$. Now we want to find p such that $g^p = m \cdot N + 1$ for some m.

We can find p by calculating the period at which a^{x+yp} (mod N) repeats. On a quantum computer this is very fast; however, we can easily do this by hand as 21 is a very small number (and our power is also small). Below, is 11 raised to the powers 0-19 (mod 21).

$$1, 11, 16, 8, 4, 2, 1, 11, 16, 8, 4, 2, 1, 11, 16, 8, 4, 2, 1, 11$$

We can see that the cycle repeats every 6 iterations. Therefore, we know that $p = 6$.

We find $11^6 = 84360 \cdot 21 + 1$. Now we can have the better guess $g^{p/2} + 1 = a \cdot s$ for some factor s, and $g^{p/2} - 1 = b \cdot t$ for some factor t. To find a, b, we need to use Euclid's algorithm to find the common factors. $g^{p/2} + 1 = 1332$, $gcd(1332, 21) = 3$. And $g^{p/2} - 1 = 1330$, $gcd(1330, 21) = 7$.

Now we found the two prime factors of $N = 21$ are $a = 3$ and $b = 7$.