

Shor's Algorithm

Isaac Storch

June 12, 2009

1 Introduction

- Factoring large numbers is an NP-complete problem; the difficulty scales exponentially with size of the number.
- The assumed difficulty of this problem is the basis of many cryptosystems, including the widely used RSA cryptosystem.
- Shor's algorithm would use a quantum computer to solve this problem in polynomial time $O((\log n)^3)$.²
- If quantum computers overcome their decoherence and scalability issues, it would have huge implications for cryptography.

2 Outline of Shor's Algorithm

- Assume we have a perfectly working classical and quantum computer at our disposal and we want to factor some integer n .
- The quantum computer has been programmed to perform Quantum Fourier Transforms and can efficiently produce the order r of an element $x \in \mathbb{Z}/n\mathbb{Z}$.
- The classical computer does everything else, including using the Euclidean algorithm to efficiently calculate the greatest common divisor (gcd) of two integers.³
- Then Shor's factoring algorithm works as follows: (Note, the algorithm simply finds a nontrivial factor of n ; it is not even necessarily prime.)²
 - 1) Choose a random $x \in \mathbb{Z}/n\mathbb{Z}$
 - 2) First things first, calculate $\gcd(x, n)$. If the result is nontrivial, congratulations! You have just found a factor of n all by yourself. Have a cookie.
 - 3) If not, use your quantum computer to find the order r of x (the smallest integer such that $x^r = 1$). If r is odd, the rest of the algorithm does not make sense, so start again at step 1 with a different x .

- 4) If $x^{r/2} + 1 \equiv 0 \pmod{n}$, then start again at step 1 (I explain this below).
- 5) Otherwise, you will find that $\gcd(x^{r/2} \pm 1, n)$ are two nontrivial factors of n .

2.1 Explanation of Steps 4 and 5

- Let $n = \prod_i p_i^{\alpha_i}$ where the p_i are prime numbers. Then

$$(x^{r/2} - 1)(x^{r/2} + 1) = x^r - 1 \equiv 0 \pmod{n} \quad (1)$$

because r is the order of x . This implies

$$\begin{aligned} (x^{r/2} - 1)(x^{r/2} + 1) &= nm \quad m \in \mathbb{Z} \\ &= \prod_i p_i^{\alpha_i} \prod_j q_j^{b_j} \end{aligned} \quad (2)$$

- Now, we know $x^{r/2} - 1 \neq n$ because by definition r is the *smallest* integer that gives us the identity.
- Thus, $x^{r/2} + 1$ contains at least one p_i .
- We want to find a nontrivial factor of n , so if $x^{r/2} + 1 = n$ we should start over.
- Therefore, both $x^{r/2} + 1$ and $x^{r/2} - 1$ each contain some prime factors of n , and if we compute the $\gcd(x^{r/2} \pm 1, n)$ we will get two (not necessarily prime) factors of n .

2.2 An Example

- Suppose we want to factor $n = 21$.
- Pick $x = 2$, then our fancy quantum algorithm will tell us $r = 6$. $2^6 = 32 \equiv 11$ $11 \cdot 2 = 22 \equiv 1$
- Now, $2^3 + 1 = 9$ and $2^3 - 1 = 7$, and we have our prime factors: $\gcd(9, 21) = 3$ and $\gcd(7, 21) = 7$.

3 How to Find the Order using a Quantum Computer

- The problem is, given x and n , find the smallest r such that $x^r \equiv 1 \pmod{n}$.
- First, pick some q that is a power of 2, such that $n^2 \leq q < 2n^2$.

- We will need two quantum registers with $\log_2 q$ qubits each; they will be represented by kets and initialized to $|0\rangle|0\rangle$ (Remember, each “0” here actually represents $\log_2 q$ 0’s)
- Put the first register in a uniform superposition of all possible values.

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0\rangle$$

- This can be achieved by sending each bit in the first register through a Hadamard gate. For example, if we had 3 bits and we wanted to represent a superposition of the integers from 0 to $q - 1 = 2^3 - 1$ (and forgetting about normalization):

$$\begin{aligned} |000\rangle &\rightarrow (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \\ &\equiv |000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle \quad (3) \\ &\equiv |0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle + |5\rangle + |6\rangle + |7\rangle \end{aligned}$$

- Next, place the second register in the state $|x^a(\text{mod } n)\rangle$. How to achieve this using only 1 and 2 qubit universal gates is left as an exercise. Our quantum state now looks like

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|x^a(\text{mod } n)\rangle$$

- Then perform a Quantum Fourier Transform on the first register:

$$|a\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{c'=0}^{q-1} \exp(2\pi i a c' / q) |c'\rangle$$

- So our quantum state is now

$$\frac{1}{q} \sum_{a=0}^{q-1} \sum_{c'=0}^{q-1} \exp(2\pi i a c' / q) |c'\rangle |x^a(\text{mod } n)\rangle$$

- Now we take measurements. The probability of being in the state $|c\rangle|x^k(\text{mod } n)\rangle$, where $0 \leq k < r$, will give us r .

$$\begin{aligned} &\left| \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c'=0}^{q-1} \exp(2\pi i a c' / q) \langle c|c'\rangle \langle x^k(\text{mod } n)|x^a(\text{mod } n)\rangle \right|^2 \\ &= \left| \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c'=0}^{q-1} \exp(2\pi i a c' / q) \delta_{c c'} \delta_{x^k(\text{mod } n), x^a(\text{mod } n)} \right|^2 \quad (4) \\ &= \left| \frac{1}{q} \sum_{a: x^a \equiv x^k} \exp(2\pi i a c / q) \right|^2 \end{aligned}$$

where the sum is over all a , $0 \leq a < q$ such that $x^a \equiv x^k \pmod{n}$.

- This is where we use the fact that r is the order of x .
- Recall $k < r < n < q$ and $a < q$. The first term in the sum is $a = k$. Because $x^r \equiv 1$ we can proceed to find all of the other terms in the series adding multiples of r to a .
- In other words, the sum can be written as a sum over $b \in \mathbb{Z}$ when we write $a = br + k$.

$$\begin{aligned}
& \left| \frac{1}{q} \sum_{b=0}^{(q-k-1)/r} \exp(2\pi i(br+k)c/q) \right|^2 \\
&= \left| \exp(2\pi i kc/q) \frac{1}{q} \sum_{b=0}^{(q-k-1)/r} \exp(2\pi i brc/q) \right|^2 \quad (5) \\
&= \left| \frac{1}{q} \sum_{b=0}^{(q-k-1)/r} \exp(2\pi i brc/q) \right|^2
\end{aligned}$$

- Without loss of generality, we can replace rc with $\{rc\}_q$ such that the argument $2\pi\{rc\}_q/q \in (-\pi, \pi]$, i.e. $-q/2 < \{rc\}_q \leq q/2$

$$\begin{aligned}
& \left| \frac{1}{q} \sum_{b=0}^{(q-k-1)/r} \exp(2\pi i brc/q) \right|^2 \\
&= \left| \frac{1}{q} \sum_{b=0}^{(q-k-1)/r} \exp(2\pi i b\{rc\}_q/q) \right|^2 \quad (6)
\end{aligned}$$

- This is where things get hairy. Shor pulls out a bunch of math and error analysis that I do not understand, so I will hand wave from here on out.
- Suppose $|\{rc\}_q| \ll q$. Then each term in the sum above has about the same phase, and we can just turn the sum into an integral.

$$\left| \frac{1}{q} \int_0^{q-k-1} \exp(2\pi i b\{rc\}_q/q) db \right|^2 \quad (7)$$

If we substitute $u = rb/q$ and suppose $k+1 \ll q$, then we have

$$\left| \frac{1}{r} \int_0^1 \exp(2\pi i \{rc\}_q u/r) du \right|^2 = \left(\frac{\sin(\pi \{rc\}_q/r)}{\pi \{rc\}_q} \right)^2 \quad (8)$$

- The value of this sinc function is maximal when $\{rc\}_q \approx 0$.

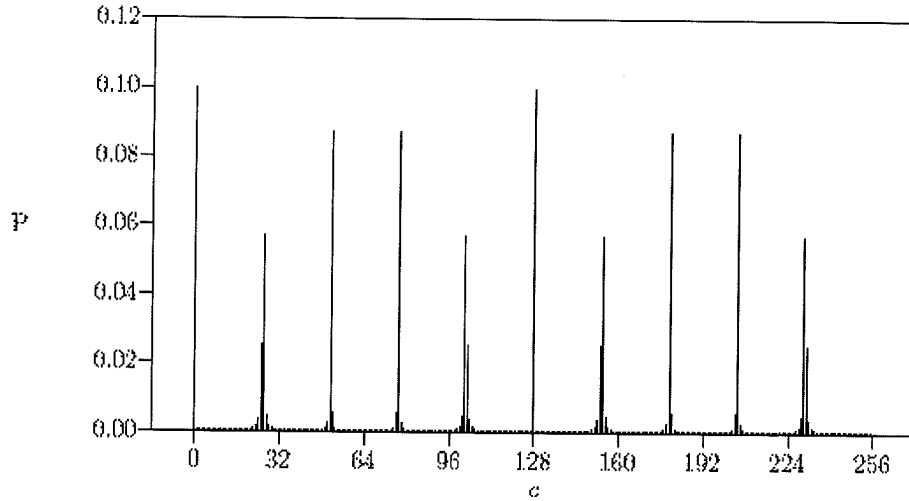


Figure 1: $q = 256$, $n = 33$, $x = 5$, $r = 10^1$

- Recall that $\{rc\}_q$ is just rc modulo q .
- Therefore, if we measure the system to be in state $|c\rangle$ with high probability (note that k has magically disappeared), then

$$\begin{aligned} \{rc\}_q &= rc - dq \approx 0 \\ \Rightarrow c &\approx d \frac{q}{r} \end{aligned} \tag{9}$$

where $d \in \mathbb{Z}$.

- Hence, all we need to do is plot probability versus c and count the number of peaks.

References

- [1] Peter W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," <http://arxiv.org/abs/quant-ph/9508027>, 25 Jan 1996
- [2] http://en.wikipedia.org/wiki/Shor_algorithm
- [3] http://en.wikipedia.org/wiki/Euclidean_algorithm