Report for implementation of Shor's Algorithm

Xiao Yunxuan, Li Yikai, Sun Xuehui

August 7, 2018

Abstract

For reason that this report is a conclusion for our implementation of Shor's Algorithm. We will firstly introduce the algorithm straightaway, with the corresponding code implementation follows.

1 Traditional Integer Factorization

Having business with Shor's Algorithm, there is a classical algorithm based on *order* in Algebra to find one prime factor of a given integer N. The procedure of such algorithm is as follows :

When the integer N is given, it must belongs to one of the three situations :

(i) N is even

(ii) $N = a^b, \qquad a, b \in \mathcal{N}$

(iii) $N = P_1^{\alpha_1} P_2^{\alpha_2} \cdots P_k^{\alpha_k} \qquad P_i$ is odd, $k > 1$

If N belongs to situation (i) or (ii), it is trivial that we get one prime factor of N. Consequently, we only need to deal with N belongs to situation (iii). So, we first select a random integer x, where $1 < x < N$.

If $gcd(x, N) > 1$, the 'Great Common Divisor' is the prime factor we want. Else, we have to find the order r of integer pair (x, N), which means solving the equation $x^r = 1 \pmod N$ and finding the smallest positive integer r.

As theories guarantee that when integer N follows situation (iii), the probability of getting a even order r is no less than $1 - \frac{1}{2^{k-1}}$.

When the order r is even, we have that $(x^{\frac{r}{2}} + 1)(x^{\frac{r}{2}} - 1) = 0 \pmod N$ with both $x^{\frac{r}{2}} + 1$ and $x^{\frac{r}{2}} - 1$ non-trivial. So that, we can get the prime factor of N just by once more calculation of gcd.

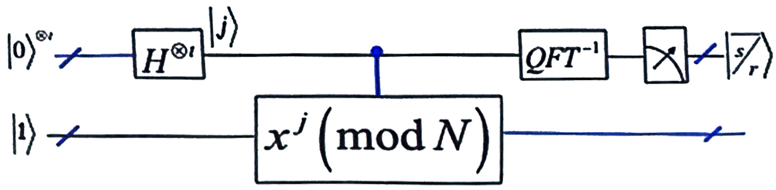
In conclusion Using this classical algorithm to do integer factorization, we have great probability to get the ans with once order-finding and twice gcd calculation. However, to find the order of a pair of large integer, it costs too much for classical computer, which is the key point where we introduce quantum algorithm and perform the great quantum acceleration.

2 QOrderFinding

The quantum algorithm for order finding is just the *Quantum Phase Estimation Algorithm* (Kitaev 1995; Cleve et al 1998) applied to the unitary operator U, which satisfies

$$U |y\rangle = |xy \ mod \ N\rangle$$

We can solve the problem with the following quantu circuit:



In the first step, we encode the phase into a superposition through Hadamard gate and then decode the phase by measuring the Fourier basis in the second step. Just design a special control-U gate U_x, we can find the order following this algorithm. The controlled U_x is designed as follows:

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

By simple computation, u_s is eigenstates of U_x.

$$\begin{aligned} U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^{k+1} \bmod N\rangle \\ &= \exp\left[\frac{2\pi i s}{r}\right] |u_s\rangle . \end{aligned}$$

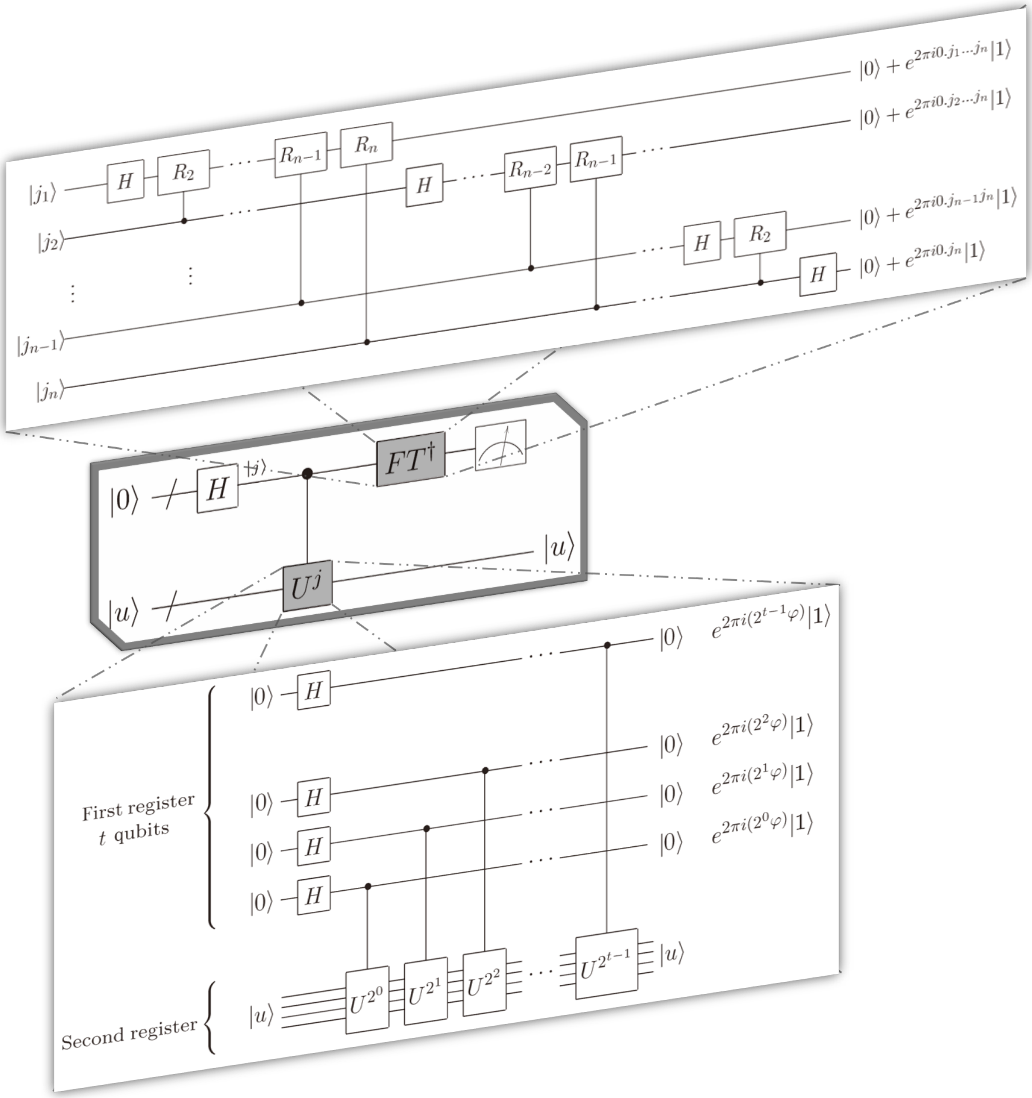
Now we obtain the corresponding eigenvalues $e^{2\pi i s/r}$, which will be later passed into continued fraction expansion to do further calculation. And theory guarantees that the right order r we want must be in one of its approximation, which we can get easily by checking it all over.

Procedure In sum, the over all procedure of the QOrderFinding algorithm can be described as follows:

1.	$\|0\rangle\|1\rangle$	initial state
2.	$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|1\rangle$	create superposition
3.	$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|x^j \bmod N\rangle$	apply $U_{x,N}$
	$\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} \|j\rangle\|u_s\rangle$	
4.	$\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \|\widetilde{s/r}\rangle\|u_s\rangle$	apply inverse Fourier transform to first register
5.	$\rightarrow \widetilde{s/r}$	measure first register
6.	$\rightarrow r$	apply continued fractions algorithm

3 The Quantum Circuits

The schematic of our quantum circuit is as follows :

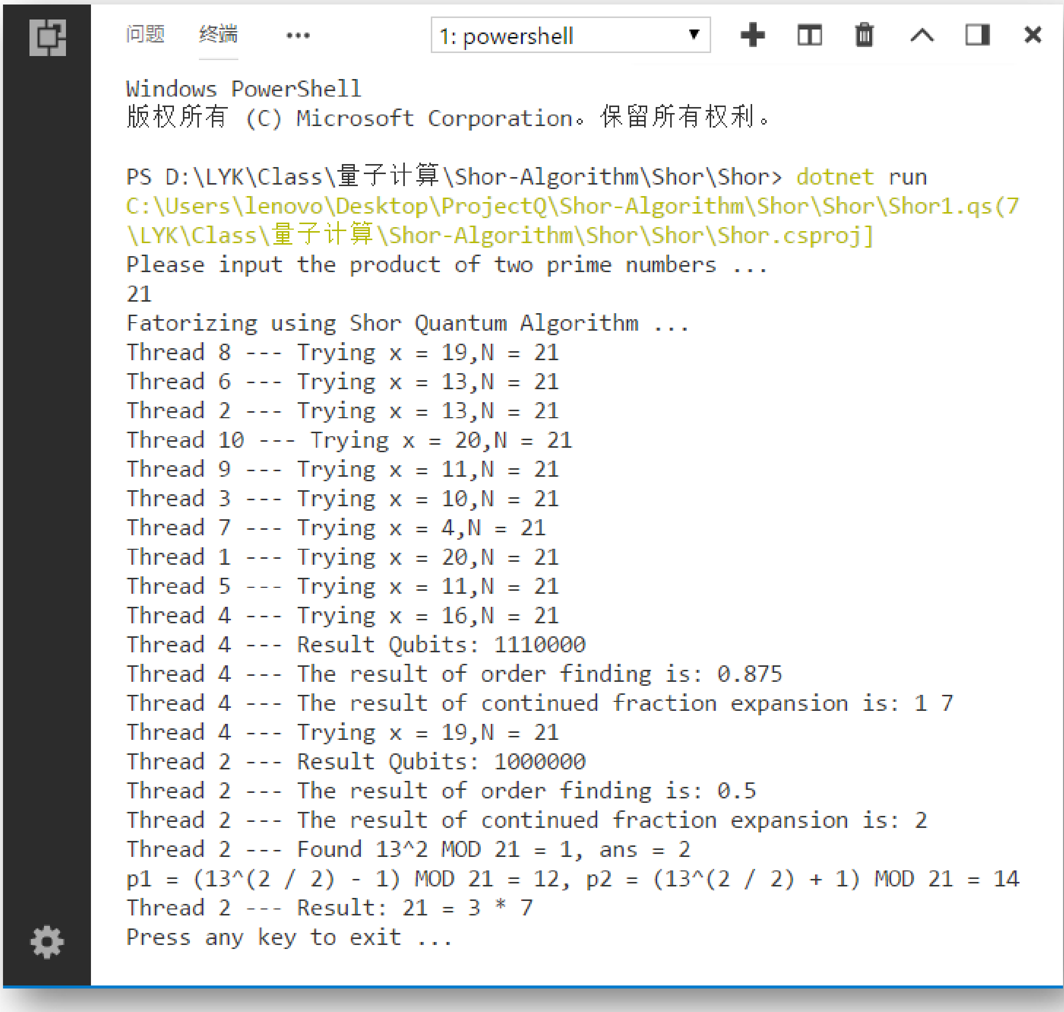


Since we use 5 quantum bits and 7 auxiliary bits, the total number of quantum bits we use, the n in the schematic, is 12.

4 Experiments and Results

In our experiment, we choose N equals to 21 and our program creates 10 threads, taking different x for calculation. For example, in Thread 4, it takes $x = 19$ and fails while in thread 2 it takes $x = 13$ and finally finds that the order $r = 2$, which satisfies $13^2 = 1 \pmod{N}$. And then, the thread which successfully finds the order abort other threads.

The details of the results are as follows : ($p1 = x^{\frac{r}{2}} - 1$ and $p2 = x^{\frac{r}{2}} + 1$)



5 The Advances

The advanced task we tried is: Introduce parallel computation.

For reason that Shor's Algorithm is a probabilistic one, to find the right answer we have to run the program repeatedly, which is a waste of time. So we consider to select x parallel and perform the acceleration by multi-thread programming, which is a optimization based on pipeline and OS.

For more details, please refer to our code implementation.