Final Project

Shor’ s algorithm

Zhen Wang

U6904458

Table of Content

Introduction

Background

The modern cryptosystem originates with the assumption that it is computational infeasible to tackle integer factorization problem. Specifically, it would computational expensive to find the factors given an integer N. The Diffie-Hellman key exchange protocol was invented to distribute the key so that it only allows for the intended user has the authority. The RSA encryption takes advantage of that key distribution system to build the widely used public-key cryptography system. These protocols are replying on the fact it is not possible to solve the integer factorization problem without prior information. Shor’ s algorithm is invented in 1994 by Peter Shor. It groundbreakingly solves the factorization problem in a near polynomial time complexity if given sufficient quantum memory. It made it feasible to break RSA in an ideal situation that leads into a new era, post-quantum cryptography.

Problem outline

Shor’ s algorithm is often known as a camouflaged version of quantum phase algorithm. The goal is for some large nonprime number N, find its factor x and y. Whereas a classic computer has to enumerate every possible candidate to test if it is the solution factor. Basically, the Shor’ s algorithm magically generates a better guess each time that increase the chance that the output number is the solution factor or at least it shares factors with the solution factor(co-factor). The integer factorization problem is firstly turned into an order-finding problem with respect to the modular exponentiation function. And it takes the use of quantum phase estimation and quantum Fourier Transform to find its period. The algorithm uses the period to update our guess iteratively until we find its factor or co-factor.

Notes: Only theoretical aspect will be discussed in this report. For implementation subtleties, it will be in notebook. ipynb file. This project is a tutorial kata, so the code file has all the details in breakdown steps.

Classic Computer Part

Intuition

We want find factor for any given large non-prime number *N*. It is to solve for *N* = x * y. If the algorithm outputs a co-factor, we use the Euclid’ s algorithm to check if we find the factor that shares the factor. Each iterative step by Shor’ s algorithm is supposed to increase our chance in finding the solution factor. Essentially, the key is to update guess.

To factorize on any given number *N*, Shor’ s algorithm starts with any random guess that might share a factor with your target number. It is very unlikely that the initial guess is already the solution factor. We use a trick to transform the initial guess into a pair of guesses that are way more likely to share factors with *N*.

Consider:
The number *A* and *B* do not have any common factors. It implies the equation $A^p = m * B + 1$, which means that there exist some number p and m such that raise *A* to the power p equals to multiply *B* by *p* and plus one. This is essentially rewrite of Euler’ s theorem, which originally goes $a^{\varphi(n)} \equiv 1\ (mod\ n)$. It also means that the greatest common divisor *A* and *B* share is one.

rewrite the equation:

$$A^p = m * B + 1$$
$$A^p - 1 = m * B$$
$$(A^{\frac{p}{2}}-1)\ (A^{\frac{p}{2}}+1) = m * B$$

A is the initial guess and our next guess simply is $(A^{\frac{p}{2}}-1)$ or $(A^{\frac{p}{2}}+1)$.

Assumption

There are some potential issues in the actual implementation. Note that the Shor’ s algorithm is guaranteed to find *p* but it could be non-even number which *p*/2 could be a fraction. Additionally, the new guess could be multiple of *A* or factor of m, which are useless at all. Qsharp are running in a simulation-based quantum on a classic computer that possibly could not have enough memory to hold up more than few qubits. Let’ s assume that none of these issues happened.

According to some online research, it turns out at least 37.5% of time, none of these issues happened that it could turn into a good guess upon a bad initial guess. we can find our goal with 99% chance within 10 guesses.

Quantum Computer Part

Quantum Fourier Transform

For a n-qubit state |x⟩ it equals $\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i\, xy}{2^n}}|y\rangle$

For simplicity, we write N = 2^n. QFT |x⟩ = $\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i\, xy}{N}}|y\rangle$ -------- (1)

We use the notation if y = $\sum_{k=1}^{n} y_k 2^{n-k}$, which essential says y = $2^0 + 2^1 + 2^2 + 2^3$ =15 if y = |1111⟩

Rewrite equation (1):

$$\text{QFT}(|x\rangle) = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i\, x \sum_{k=1}^{N} y_k}{2^k}} |y_1,\ y_2,\ y_3 \cdots y_n\rangle$$

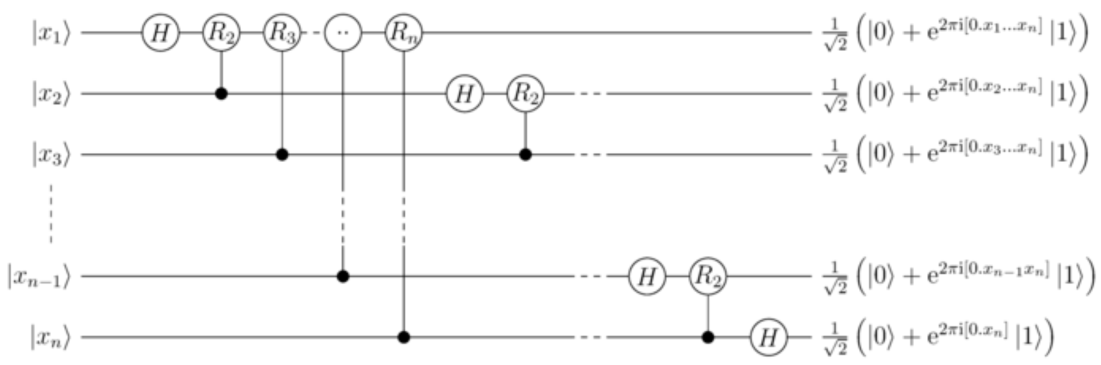
$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=1}^{N} e^{\frac{2\pi i x y_k}{2^k}} |y_1,\ y_2,\ y_3 \cdots y_n\rangle$$

$$= \frac{1}{\sqrt{N}} (|0\rangle + e^{\frac{2\pi\, ix}{2^1}}|1\rangle) \otimes (|0\rangle + e^{\frac{2\pi\, ix}{2^2}}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{\frac{2\pi\, ix}{2^N}}|1\rangle) \text{ -------(2)}$$

Property of (2):

|x⟩ = $|x_1,\ x_2,\ x_3 \cdots x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle \otimes \cdots \otimes |x_n\rangle$ where each term $|x_n\rangle$ corresponds to the term $(|0\rangle + e^{\frac{2\pi\, ix}{2^N}}|1\rangle)$ in (2).

Full circuit:



$|x_1\rangle$
$|x_2\rangle$
$|x_3\rangle$
$|x_{n-1}\rangle$
$|x_n\rangle$
H
R_2
R_3
R_n
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_1\ldots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_2\ldots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_3\ldots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_{n-1}x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_n]}|1\rangle\right)$

Fig 1. QFT circuit

Source Wikipedia

Relevant Operation:

H gate: H(|x⟩)= $\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{2\pi\, ix}{2}}|1\rangle)$

Unitary rotation: $\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}$ it only applies phase $e^{\frac{2\pi i}{2^k}}$ to state |1⟩.

Example of 5 qubits:

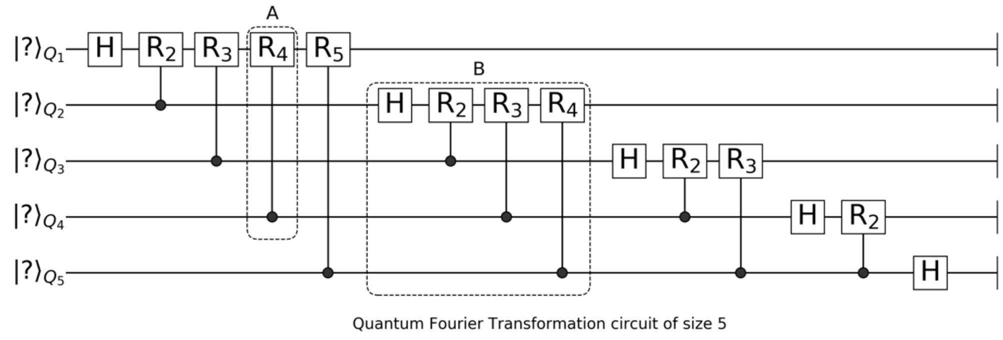


Fig 2. Discrete example of QFT

Source yaoquantum

Reverse ordering of actual workflow:

Workflow w.r.t first qubit:

$|x_1, \ x_2, \ x_3, \ x_4, \ x_5\rangle$

$\rightarrow (|0\rangle \ + \ e^{\frac{2\pi i}{2^1}x1}|1\rangle) \ \otimes \ |x_2, \ x_3, \ x_4, \ x_5\rangle$

$\rightarrow (|0\rangle \ + \ e^{\frac{2\pi i}{2^1}x1} \ e^{\frac{2\pi i}{2^2}x2} \ |1\rangle) \ \otimes \ |x_2, \ x_3, \ x_4, \ x_5\rangle$

$\rightarrow (|0\rangle \ + \ e^{\frac{2\pi i}{2^1}x1} \ e^{\frac{2\pi i}{2^2}x2} \ e^{\frac{2\pi i}{2^3}x3} \ |1\rangle) \ \otimes \ |x_2, \ x_3, \ x_4, \ x_5\rangle$

$\rightarrow (|0\rangle \ + \ e^{\frac{2\pi i}{2^1}x1} \ e^{\frac{2\pi i}{2^2}x2} \ e^{\frac{2\pi i}{2^3}x3} e^{\frac{2\pi i}{2^4}x4} \ |1\rangle) \ \otimes \ |x_2, \ x_3, \ x_4, \ x_5\rangle$

$\rightarrow (|0\rangle \ + \ e^{\frac{2\pi i}{2^1}x1} \ e^{\frac{2\pi i}{2^2}x2} \ e^{\frac{2\pi i}{2^3}x3} e^{\frac{2\pi i}{2^4}x4} \ e^{\frac{2\pi i}{2^5}x5} \ |1\rangle) \ \otimes \ |x_2, \ x_3, \ x_4, \ x_5\rangle$

→ keep expanding the rest qubits ···

Quantum Phase Estimation

Full circuit:

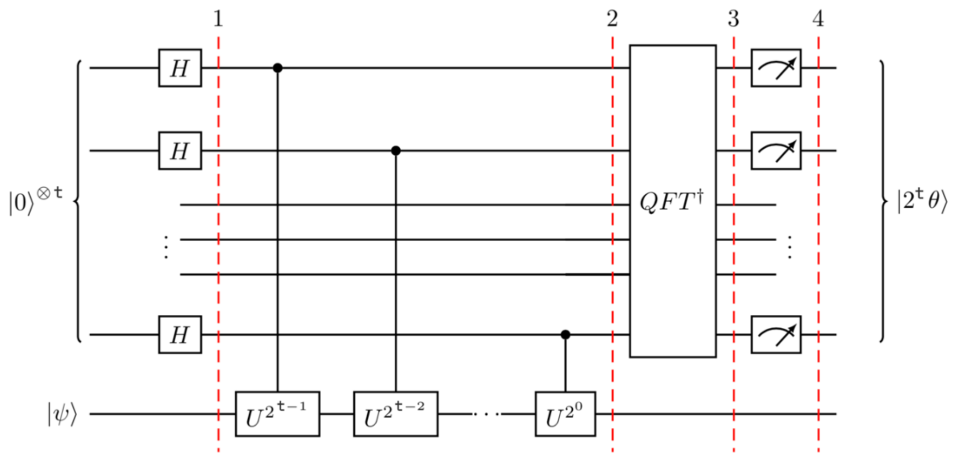


Fig 3. Phase estimation circuit

Source qiskit

Where $|0\rangle^{\otimes t}$ is the n qubits to be measured and $|\psi\rangle$ is the target qubit.
Workflow:
Initial state: $|0\rangle^{\otimes t}\ |\psi\rangle$

Apply n-bits Hadamard gate: $\frac{1}{\sqrt{2}}^{t}\ (|0\rangle \pm |1\rangle)^{\otimes t}\ |\psi\rangle$

Recall that QFE are constructed under the assumption that U is a Unitary operation, and it has the eigenvalue of $e^{2\pi i\theta}$ and corresponding eigenvector $|\psi\rangle$.

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

So, for a unitary operator $U^{2^t}\ |\psi\rangle = U^{2^{t-1}}U\ |\psi\rangle = U^{2^{t-1}}\ e^{2\pi i\theta}\ |\psi\rangle = e^{2\pi i\theta\ 2^t}\ |\psi\rangle$

Eventually, appley all t-controlled operations U^{2^t} it will yield the following:

$$\frac{1}{\sqrt{2}}^{t}\ (|0\rangle + e^{2\pi i 2^{t-1}}|1\rangle) \otimes (|0\rangle + e^{2\pi i 2^{t-2}}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i 2^{0}}|1\rangle)$$

$$= \frac{1}{\sqrt{2}}^{t}\ \sum_{k=0}^{2^t-1} e^{2\pi i\theta k}|k\rangle|\psi\rangle \qquad \text{------- (3)}$$

Last, apply inverse QFT on the result of t-controlled operations U^{2^t}.

Note that complex conjugate transpose of (1) is just by putting a minus sign in front its exponent.

$$QFT^\dagger = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{-\frac{2\pi i\, xy}{N}}|y\rangle \qquad \text{------ (4)}$$

$Apply\ inverse\ of\ (4)\ on\ (3) \rightarrow\ \frac{1}{\sqrt{2}}^{t}\ \sum_{x=0}^{N-1}\sum_{k=0}^{2^t-1} e^{-\frac{2\pi i\, k}{N}(x - 2^n\theta)}\ |x\rangle \otimes |\psi\rangle$ ------- (5)

where t in (3) is equal to n in (4) and $N = 2^n$.

Shor' s workflow

Modify the above QPE circuit such that the unitary operation U takes on the form of modular exponentiation $f\,x = a^x (mod\,N)$ where N is the larger number we are trying to factorize and denote additional qubits for QPE as ω by convention. By doing this, the circuit can keep track of reminder throughout.

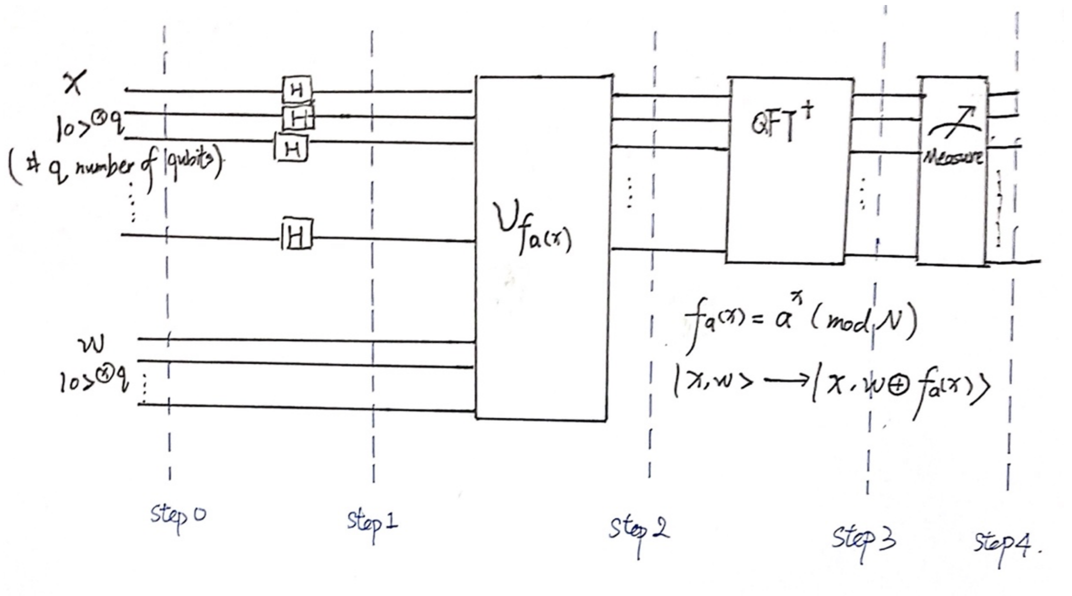


Fig 4. Shor' s circuit.

Source original

Since detail workflow are already mentioned in previous part, we have equation (3) (4) and (5). We can simply the following sub-step derivation process. q means the number of qubits used to encode a large number N. The large number to be factorized is N.

Step0: We need an equal number of additional qubits in ω to record the remainder.

For the x register, we have $|x_1,\ x_2,\ x_3 \cdots\ x_q\rangle$ with all qubits in state $|\,0\,\rangle$.

For the ω register, we have $|\omega_1,\ \omega_2,\ \omega_3 \cdots\ \omega_q\rangle$ with all qubits in state $|\,0\,\rangle$.

For simplicity' s sake, we can just denote $|\,0\,\rangle^{\otimes\, q}\,|\,0\,\rangle^{\otimes\, q}$

Step1: Only apply Hadamard gate to x register, not ω register

$$\frac{1}{\sqrt{2}}^{q}\ (|0\rangle\ \pm\ |1\rangle)^{\otimes\, q}\ |\,\omega\rangle\ =\ \frac{1}{\sqrt{2}}^{q}\ \sum_{q\,=0}^{2^q-1} |x_q\rangle\,|\,\omega\rangle$$

Again, for simplicity, we can rewrite as

$$[\,H^{\otimes\, q}\,|\,0\,\rangle^{\otimes\, q}\,]\ |\,0\,\rangle^{\otimes\, q}\ \text{-----(6)}$$

It means $[\,H^{\otimes\, q}\,|\,0\,\rangle^{\otimes\, q}\,]$ in x register while $|\,0\,\rangle^{\otimes\, q}$ in ω register.

Step2: Apply unitary operation on modular exponentiation

$$\frac{1}{\sqrt{2}}^{q}\ [\,|0\rangle\,|0 \oplus\ a^0(mod\ N)\,\rangle\ + |1\rangle\,|0 \oplus\ a^1(mod\ N)\,\rangle\ +\ |2\rangle\,|0 \oplus\ a^2(mod\ N)\,\rangle + \ldots + |N\rangle\,|0 \oplus\ a^N(mod\ N)\,\rangle\,]$$

since $\oplus$ is addition modulo where 0 $\oplus$ any number wouldn' t make any impact.

$= \frac{1}{\sqrt{2}}^{q} [\,|0\rangle\,|a^0(mod\ N)\,\rangle + |1\rangle\,|a^1(mod\ N)\,\rangle + |2\rangle\,|a^2(mod\ N)\,\rangle + \ldots + |N\rangle\,|\,a^N(mod\ N)\,\rangle\,]$

------- (7)

The modular operation to keep track of remainder make sense as you can always find another number with some interval to has the same remainder in the modular exponentiation function.

Property: For some random number x, it has a whole number m such that

$$a^x = m * N + 1$$

If exist another number x_1 such that

$$a^{x1} = m * N + r$$

Then for some concrete interval p, it has

$$a^{x1+p} = m1 * N + r$$

$$\ldots$$

$$a^{x1+n*p} = m_n * N + r$$

The reminder (r) stays the same.

Where for example $|7\rangle\, just\ means\ |111\rangle$ if q = 3, for q > 3 $|7\rangle\, just\ means\ |0\ldots0111\rangle$ that only the last three digit is 1. Recall that a is the first random number we pick that is coprime to the number we are trying to factorize.

For simplicity, $U_f \frac{1}{\sqrt{2}}^{q} \sum_{q=0}^{2^q-1} |x_q,\ \omega\rangle = \frac{1}{\sqrt{2}}^{q} \sum_{q=0}^{2^q-1} |x_q,\ f(x_q)\rangle$

Step3: Only apply $QFT^{\dagger}$ on x register, not ω register

Before that, we can rewrite the (7) as the periodicity come from (7), if we denote period found with r, rewrite (7) as:

$$\frac{1}{\sqrt{2}}^{q} [\,|0\rangle\,|a^0(mod\ N)\,\rangle + |1\rangle\,|a^1(mod\ N)\,\rangle + \ldots + |r-1\rangle\,|\,a^{r-1}(mod\ N)\,\rangle$$
$$+ \quad |r\rangle\,|a^0(mod\ N)\,\rangle + |r+1\rangle\,|a^1(mod\ N)\,\rangle + \ldots + |2r-1\rangle\,|\,a^{r-1}(mod\ N)\,\rangle$$
$$+ \quad |2r\rangle\,|a^0(mod\ N)\,\rangle + |2r+1\rangle\,|a^1(mod\ N)\,\rangle + \ldots + |3r-1\rangle\,|a^{r-1}(mod\ N)\,\rangle$$
$$\cdots)$$ ------- (8)

Where you can easily see a pattern, one period repeat multiple times with the remainder stays the same for different periods. For each remainder in one period, we can denote as $|0\rangle\,|j_0\,\rangle + |1\rangle\,|j_1\rangle + \ldots + |r-1\rangle\,|0 \oplus j_{r-1}\rangle$, j is the remainder.

For simplicity, $\frac{1}{\sqrt{2}}^{q} \sum_{x=0}^{2^q-1} \sum_{k=0}^{2^q-1} e^{-\frac{2\pi i\,k}{2^q}} |\mathrm{k}, \mathrm{f(x)}\,\rangle$

Now, if we measure the ω register against (8), $M(\omega)$ has the equal probability of being $j_0\ to\ j_{r-1}$. If you think (8) as a N/r by r matrix, if $M(\omega)$ is j_c where c is from the range 0 to r-1, then x register is the c^{th} column of matrix with the form $\frac{1}{\sqrt{2}}^{q} (|c\rangle + |r+c\rangle + .. + |n * r + c\rangle) \otimes |j_c\rangle$ ------ (9).

When apply $QFT^{\dagger}$ to each term in (9), it will only be left with few terms, and all other vanishes. It is often referred as quantum interference.

Step4: Measure the x register, with each term has equal probability of being measured. Shor pointed out that measure result peak near $j\frac{2^q}{r}$. That is for a measurement m, we can have multiple set of answers j and r corresponding m if we solve for $j\frac{N}{r}$=m. For each set of j and r, compute $a^{r/2}(mod\ N)$ and gcd ($a^{r/2}(mod\ N) \pm 1, N$) to check if it contains cofactors.

Flowchart:

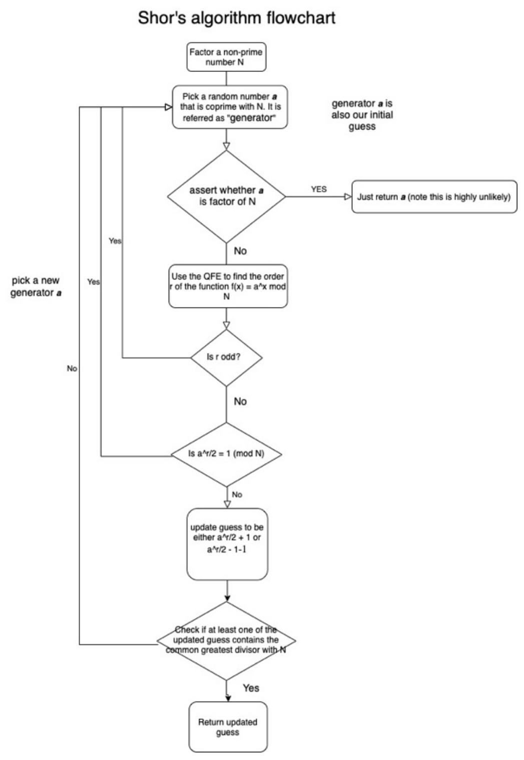

Shor's algorithm flowchart
Factor a non-prime number N
Pick a random number a that is coprime with N. It is referred as "generator"
generator a is also our initial guess
assert whether a is factor of N
YES
Just return a (note this is highly unlikely)
No
pick a new generator a
Yes
Yes
No
Use the QFE to find the order r of the function f(x) = a^x mod N
Is r odd?
No
Is a^r/2 = 1 (mod N)
No
update guess to be either a^r/2 + 1 or a^r/2 - 1-1
Check if at least one of the updated guess contains the common greatest divisor with N
Yes
Return updated guess

Fig 5. Flow chart.

Source original

Conclusion

Complexity Analysis:

The best algorithm ever published on a classic computer to factor integers is only sub-exponential with a n-bit input number in complexity of $O((c)(\ln n^{\frac{1}{3}})(\ln \ln n^{\frac{2}{3}}))$ where c is some constant around $\sqrt[3]{\frac{64}{9}}$. It is known as general number field sieve (GNFS). No algorithm or it is proven the existence of a polynomial time algorithm. The problem of factoring is suspected to be not NP-complete. Surprisingly, Shor algorithm can have a complexity less then $O(n^3)$, which is rather ground-breaking. The main runtime composite comes from modular exponentiation that is dominated by Fourier transform. Specifically, Shor used the repeated squaring on tracking remainder, which consumes many ancillary qubits and gates. The circuit for QFT has $O(\log N)^2 \text{ for preision N being } 2^n$ and squaring is said to has complexity of $O((n\,(\log x))^k)$ where k is the fixed operation on a sub-grouping. A simple way to improve is use ripple-carry adders as reversible gates. The latest successful attempt made is to factor number 21 (Martín-López et al. 2012). The IBM failed to factor number 35 in 2019 due to accumulating errors.

Reflection

Although the basic idea sound straightforward, it gets tons of details that you really need pay attention to. Some subtle including how to ensure the interference generated by QFT can indeed cancel out the answer you don’ t expect. It is essential that structure the program to handle the exception if the update guess is not valid or the case that the period finding is not giving expected answer. The most essential one is to use the strategy of continued fraction expansion to find fraction from the frequency, which also has the rate of failing. In the test section, the implementation can successfully factorize the number less than 100. In some extreme case, it takes 10 mins to run through the phase estimation. It is problematic in finding the period from frequency by calculate fraction depending on the frequency peaks. The behavior is expected since a simulator cannot model too many qubits at once taking consideration of massive ancillary qubits need in tracking remainder.

Although Shor’ s algorithm has been published long time ago, but recently there’ s still been a rich content of paper discovering the potential implication of Shor’ s algorithm. Some interesting academic research is conducted focusing on reducing noisy qubits (Craig et al. 2021), error correction with stabilizer (Robert Vandermolen et al. 2021) and exploring parallelism in quantum control (Mengyu Zhang et al. 2021) and so on. All of those provide a novel view on how the project can be expend. One direction is to explore the possibility of utilizing parallel control over multiple programming language and find out if the complexity would improve by parallel execution.

Reference

1. Shor, P.W. (1999). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, [online] 41(2), pp.303–332. Available at: https://arxiv.org/pdf/quant-ph/9508027.
2. Häner, T., Roetteler, M. and Svore, K.M. (2017). Factoring using 2n+2 qubits with Toffoli based modular multiplication. *arXiv:1611.07995 [quant-ph]*. [online] Available at: https://arxiv.org/abs/1611.07995 [Accessed 20 Oct. 2021].
3. Gidney, C. and Ekerå, M. (2019). How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *arXiv:1905.09749 [quant-ph]*. [online] Available at: https://arxiv.org/abs/1905.09749.
4. Bourdon, P.S. and Williams, H.T. (2006). Sharp probability estimates for Shor's order-finding algorithm. *arXiv:quant-ph/0607148*. [online] Available at: https://arxiv.org/abs/quant-ph/0607148 [Accessed 20 Oct. 2021].
5. Vandermolen, R. and Wright, D. (2021). Quantum Error Correction with Reflexive Stabilizer Codes and Cayley Graphs. *arXiv:2110.08414 [math-ph, physics:quant-ph]*. [online] Available at: https://arxiv.org/abs/2110.08414 [Accessed 20 Oct. 2021].
6. Zhang, M., Xie, L., Zhang, Z., Yu, Q., Xi, G., Zhang, H., Liu, F., Zheng, Y., Zheng, Y. and Zhang, S. (2021). Exploiting Different Levels of Parallelism in the Quantum Control Microarchitecture for Superconducting Qubits. *arXiv:2108.08671 [quant-ph]*. [online] Available at: https://arxiv.org/abs/2108.08671 [Accessed 20 Oct. 2021].
7. algassert.com. (n.d.). *Shor's Quantum Factoring Algorithm*. [online] Available at: https://algassert.com/post/1718.
8. Wu, Z. (2021). *Course Project - Shor's Algorithm*. [online] GitHub. Available at: https://github.com/Michaelvll/myQShor [Accessed 20 Oct. 2021].
9. livebook.manning.com. (n.d.). *Chapter 11 Arithmetic With Quantum Computers · Learn Quantum Computing with Python and Q#: A Hands-on approach LB*. [online] Available at: https://livebook.manning.com/book/learn-quantum-computing-with-python-and-q-sharp/chapter-11/v-8/121 [Accessed 20 Oct. 2021].
10. Matsuzaki, T. (2019). *Programming Quantum Phase Estimation (with Quantum Fourier Transform)*. [online] tsmatz. Available at: https://tsmatz.wordpress.com/2019/04/26/quantum-computing-qsharp-quantum-fourier-transform-and-phase-estimation/ [Accessed 20 Oct. 2021].
11. Wikipedia Contributors (2021). *Shor's algorithm*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Shor%27s_algorithm#cite_note-10 [Accessed 20 Oct. 2021].

12. Wikipedia. (2020). *Exponentiation by squaring*. [online] Available at: https://en.wikipedia.org/wiki/Exponentiation_by_squaring.
13. Wikipedia. (2021). *Quantum phase estimation algorithm*. [online] Available at: https://en.wikipedia.org/wiki/Quantum_phase_estimation_algorithm [Accessed 20 Oct. 2021].
14. www.e-booksdirectory.com. (n.d.). *Quantum Computer Science by David Mermin - Download link*. [online] Available at: https://www.e-booksdirectory.com/details.php?ebook=3828 [Accessed 20 Oct. 2021].