Implementation of Shor's Algorithm

Main Seminar Quantum Optics

Chia-Tso, Lai

December 2022

Abstract

With the emergence of quantum computers, many tasks deemed elusive for the classical counterparts to execute are becoming much easier to solve thanks to the superior computation power of quantum algorithms. Quantum algorithms exploit the superposition nature of qubits to simultaneously compute multiple states at once as opposed to being limited to binary states as in the case of classical computers. Since the birth of quantum computing and quantum information science, a variety of quantum algorithms have been developed. Arguably one of the most famous and potent among them is the *Shor's algorithm*. *Shor's algorithm* enables the factorization of a large integer N in polynomial time *O(logN)*, which is a significant leap in computation speed compared to the classical algorithm whose computation time scales up exponentially as the target number increases. In this written report, I will introduce the theoretical background of *Shor's algorithm*, the essential building blocks of the algorithm, and the quantum circuit that implements the algorithm using the *Qiskit* package from *Python*.

1 Introduction

1.1 Prime Factors of an Integer

If two integers A and B have no common factors, namely A and B coprime, then one can claim that $A^p = mB + 1$ for some power p and some multiple m (for example $7^4 = 160 \cdot 15 + 1 \quad 42^3 = 5699 \cdot 13 + 1$). Using this property, one can take a random guess g, which coprimes with the target number N and yield a power p which satisfies the relation:

$$g^p = mN + 1 \Rightarrow g^p - 1 = mN \Rightarrow (g^{p/2} + 1)(g^{p/2} - 1) = mN \tag{1}$$

From the above relation, one can tell $g^{p/2}+1$ and $g^{p/2}-1$ terms are either factors of N or share factors with N. Ideally, in the former case, one successfully factors integer N, while in the latter case, the prime factors of N can easily be found

by taking the greatest common divisors $gcd(g^{p/2}+1, N)$ and $gcd(g^{p/2}-1, N)$ using the classical *Euclid's algorithm*.

There are however two obvious problems with this method of factorizing. Firstly, one scenario can take place where

$$\begin{cases} g^{p/2}+1 &= aN \\ (g^{p/2}-1)a &= m \end{cases} \tag{2}$$

for some integer multiple a.

This of course does not give us any useful information about the factors of N. The other troublesome situation is when the power p is an odd number, giving rise to non-integer values for both terms. These two scenarios arise due to a bad choice of g, which, thanks to the fast computation of *Shor's algorithm*, can be circumvented without significant loss of time by simply making a new choice. Empirically speaking, for a random guess g, 37.5% of the time these problems do not occur. Therefore, one can expect over 99% success rate of factorization within 10 guesses.

1.2 General Procedure

The principle of *Shor's* algorithm hinges upon eq(1), and thus the objective of the algorithm is to find out the smallest power p that serves as a solution to the equation $g^p \mod N = 1$. The general procedure of *Shor's* algorithm can be broken down into four steps:

- Superposition Create a superposition of multiple power states $|p\rangle$ with Hadamard gates
- Modulo Function Take $|p\rangle$ as the input state and generate the value $g^p \mod N = r$ via some unitary transformation to form a product state $|p, r\rangle$
- Measurement Measure the state of the remainder. This would lead to a collapse of the system into a superposition of power states with a specific remainder r
- Periodicity Extract the period from the repeating pattern in power p

Once the period p' of the power states is obtained, one can proceed with the classical post-processing to compute $g^{p'/2}+1$ and $g^{p'/2}-1$ as well as their greatest common divisors with N.

1.3 From Factorization to Period Finding

In this subsection, I will explain why the period of power in a superposition state is the solution to the equation $g^p \mod N = 1$. Namely, how *Shor's* algorithm cleverly turns the problem of factorization into a period finding problem.

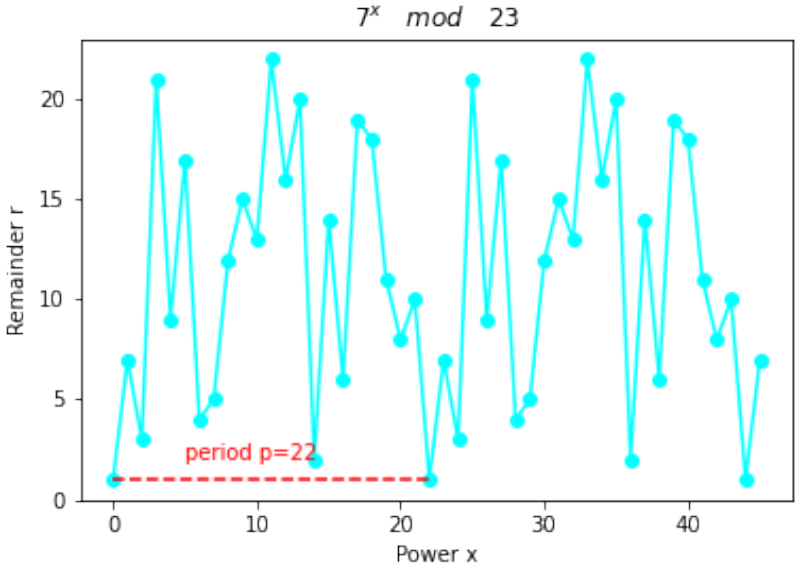


Figure 1: Repeating pattern of $7^x \mod 23$ with the period $p = 22$

Let p be a solution to the desired equation and x be any power that gives rise to a remainder of some integer r after the modulo operation with respect to N:

$$\begin{cases} g^p &= m_1 N + 1 \\ g^x &= m_r N + r \end{cases} \tag{3}$$

If one takes $x+p$ as the power, it can be shown that the same modulo operation would also lead to a remainder r:

$$g^{x+p} = g^x g^p = (m_r N + r)(m_1 N + 1) = MN + r \Rightarrow g^{x+p} \mod N = r \tag{4}$$

Therefore, a shift of p in power does not change the value of the modulo operation. In the same fashion, $g^{x-p}, g^{x+p}, g^{x+2p}$ and so on would have the same remainder r. It's become obvious that the power of interest has repeating property, and the most crucial power p is nothing but the period of these repeating powers such as an example depicted in Figure 1. As a result, if the measurement is done on the collective state of the system mentioned in (1.2) and the remainder state collapses into a constant r, then the consequential superposition state is composed of powers with an equal difference (or period) p, One can now claim that *Shor's* algorithm essentially deals with the problem of period finding.

2 Components of Shor's Algorithm

2.1 Quantum Fourier Transform

To solve the period finding problem, *Quantum Fourier Transform (QFT)*[2] is needed for the construction of *Shor's algorithm*, for it maps the qubits from computational basis to Fourier basis. This is analogous to the mapping of the classical Fourier transform where the conversion takes place between position and frequency spaces. A simple example of QFT is the single-qubit Hadamard gate that brings $|0\rangle$ in the computational basis to $|+\rangle$ in the Hadamard basis. Hence, the Hadamard gate can be seen as the one-qubit QFT.
For the general n-qubit QFT, the definition goes as follows:

$$QFT : |x\rangle \rightarrow \ |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} w_N^{xy} |y\rangle \ , \ w_N = e^{\frac{2\pi i}{N}} \tag{5}$$

Here $|x\rangle$(as well as $|y\rangle$) is the binary number notation of the n-qubit state $|x_1\rangle \otimes |x_2\rangle \ldots \otimes |x_n\rangle$, where $x = 2^{n-1}x_1 + 2^{n-2}x_2 + \ldots + x_n$. In total, there are $N = 2^n$ basis states for any n-qubit state.
The definition of QFT can be further expanded to yield the tensor product of the individual qubit states on the Fourier basis.

$$QFT\,|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} w_N^{xy} |y\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i x y}{2^n}} |y\rangle \tag{6}$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i (\sum_{k=1}^{n} \frac{y_k}{2^k}) x} |y_1 y_2 \ldots y_n\rangle \tag{7}$$

$$= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=1}^{n} e^{\frac{2\pi i x y_k}{2^k}} |y_1 y_2 \ldots y_n\rangle \tag{8}$$

$$= \frac{1}{\sqrt{N}} (\sum_{y_1=0}^{1} e^{\frac{2\pi i x y_1}{2}} |y_1\rangle) \otimes (\sum_{y_2=0}^{1} e^{\frac{2\pi i x y_2}{2^2}} |y_2\rangle) \otimes \ldots \otimes (\sum_{y_n=0}^{1} e^{\frac{2\pi i x y_n}{2^n}}) |y_n\rangle) \tag{9}$$

$$= \frac{1}{\sqrt{N}} \bigotimes_{k=1}^{n} (|0\rangle + e^{\frac{2\pi i x}{2^k}} |1\rangle) \tag{10}$$

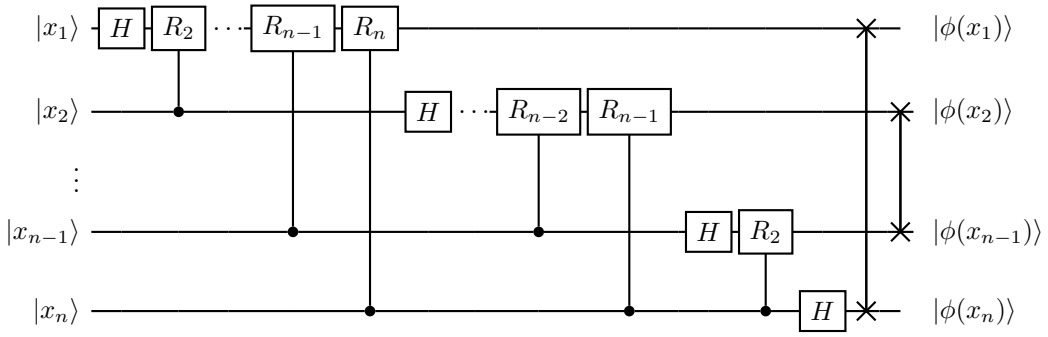
Taking a closer look at the expression of QFT as the tensor product of n qubits, one might notice that each qubit constitutes a quantum state on the x-y plane of the Bloch's sphere with a certain phase. In light of this, a quantum gate of QFT can be constructed with Hadamard gates, which bring qubits into Hadamard basis, and some phase gates R_k which take the following matrix form:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix} \tag{11}$$

To include x in the phases, controlled-R_k gates should be utilized:

$$CR_k \left|x_1 x_2\right\rangle = \left|x_1\right\rangle \otimes e^{\frac{2\pi i x_1}{2^k}} \left|x_2\right\rangle \tag{12}$$

With the necessary tools, a QFT gate of n qubits can then be built up by assembling these component gates as shown in the circuit below.



Note that the order of qubits is reversed after a series of Hadamard and controlled-phase gates, so one simply has to apply SWAP gates in the end to restore the correct order.

2.2 Quantum Phase Estimation

Quantum Phase Estimation(*QPE*)[3] is a quantum algorithm that computes the phase of the eigenvalue of a given unitary operator and its eigenvector. It's not trivial at first sight how this is related to *Shor's* algorithm, but we will see later on that the period of power p is encoded in the phase of a specific unitary operator.

Let us now define the problem of quantum phase estimation more explicitly:

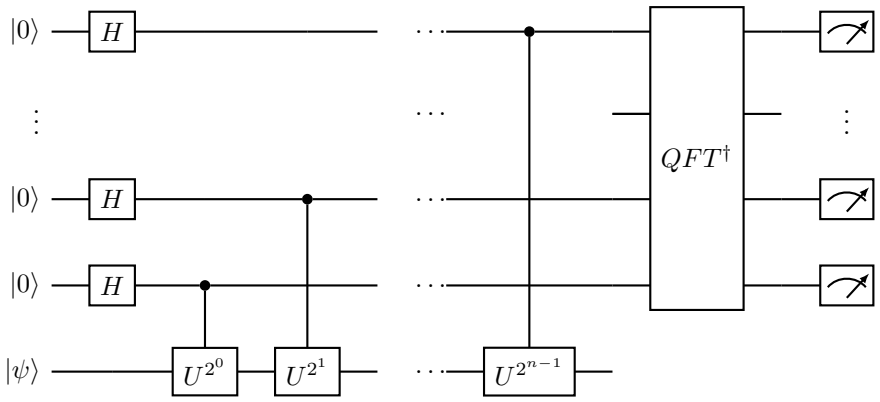
$$U \left|\psi\right\rangle = e^{2\pi i\theta} \left|\psi\right\rangle \tag{13}$$

where $\left|\psi\right\rangle$ is the eigenvector and θ is the phase (namely, the eigenvalue) of the unitary operator U that one intends to find.

The construction of QPE exploits its resemblance to QFT. One takes two different parts of quantum registers: The upper registers representing the superposition of 2^n eigenstates of an n-qubit state, which will be used as controlled qubits, and the lower registers representing the eigenvector $\left|\psi\right\rangle$. Next, a series of controlled U gates of different powers are applied to the circuit. In each application of controlled U gates, the phase of $\left|\psi\right\rangle$ subject to a certain power 2^k is passed down to the controlled qubit, resulting in the individual qubit state $\frac{1}{\sqrt{2}}(\left|0\right\rangle + e^{2\pi i\theta 2^k} \left|1\right\rangle)$. The collective effect of these operations is equivalent to

performing QFT on the statevector $|2^n\theta\rangle$ of the upper registers (In other words, taking x = $2^n\theta$ in eq(8)). Therefore, by measuring the upper registers after an inverse QFT, one can deduce the desired phase of the unitary operator U.

The exact circuit of QPE takes the following form:



2.3 Modular Exponentiation

The most crucial and challenging part of building the structure of *Shor's* algorithm is the unitary operator that performs modular exponentiation. Up until today, There have been many proposals for the circuit design of modular exponentiation to simplify the operation or reduce the number of qubits as well as circuit depth, and it's definitely not a trivial task. Consider the unitary transformation $U|x\rangle = |ax \ mod \ N\rangle$ for some integer a and N. By taking the initial state $|x\rangle = |1\rangle$, one can easily verify that after r rounds of operation, the state returns to the initial state $|1\rangle$.(In fact, r is exactly the period sought after in *Shor's* algorithm)

$$U^r|1\rangle = |a^r \ mod \ N\rangle = |1\rangle \tag{14}$$

Hence, a superposition of one cycle $|u_0\rangle$ would be an eigenstate of U:

$$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left|a^k \ mod \ N\right\rangle \tag{15}$$

Without loss of generality, additional phases can be introduced to each basis state:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi iks}{r}} \left|a^k \ mod \ N\right\rangle \qquad s \in Z \ , \ 0 \leq s \leq r-1 \tag{16}$$

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{2\pi is}{r}} e^{\frac{-2\pi i(k+1)s}{r}} \left|a^{k+1} \ mod \ N\right\rangle = e^{\frac{2\pi is}{r}} |u_s\rangle \tag{17}$$

If one sums up all the possible $|u_s\rangle$, all terms cancel out except for $|1\rangle$ (k=0)

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle \tag{18}$$

$$U|1\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2\pi i s}{r}} |u_s\rangle \tag{19}$$

Eq(17) incorporates the period r in the phase of each basis state. Hence, if QPE is applied to such unitary transformation U and the superposition of its corresponding eigenstates $|1\rangle$, one would expect to yield an output of $\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|2^n \frac{s}{r}\right\rangle$

Some rough ideas for designing the unitary gate discussed here will be introduced in the next section. However, due to its complexity, the detailed composition would be out of the scope of this report.

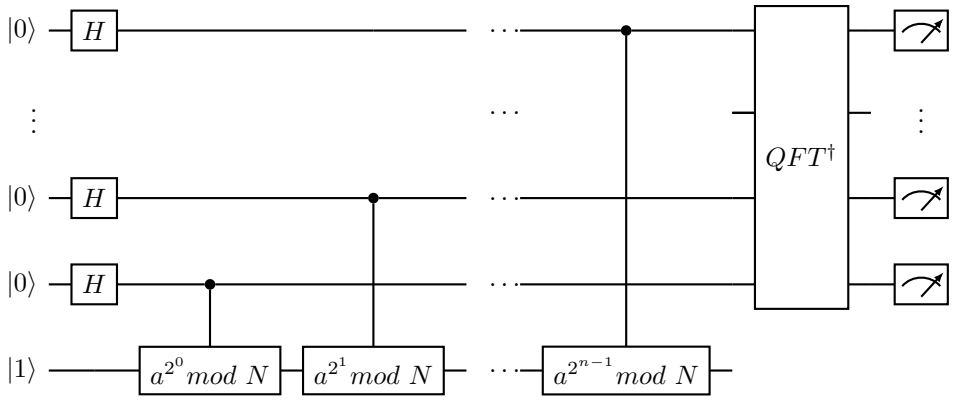
2.4 Complete Layout of Shor's Algorithm

Given all the components introduced in this section, one can assemble them together and eventually construct the circuit for *Shor's* algorithm. From (2.3), one learns that the period r, which serves as the key to factoring the integer N can be derived from executing QPE on a specific unitary operator satisfying the following relation:

$$U^k|1\rangle = \left|a^k \mod N\right\rangle \tag{20}$$

We may simply denote this unitary gate as $U^k \equiv a^k \mod N$. Using the binary expansion, one may decompose this unitary operator as a series of product $U^k = U^{2^0 k_n} U^{2^1 k_{n-1}} \dots U^{2^{n-1} k_1}$, which can be realized as a chain of controlled gates with the controlled qubits being the registers of power k.

Now it's more obvious that *Shor's* algorithm is nothing but QPE of modular exponentiation in disguise. Following the same structure as QPE, the circuit layout of *Shor's* algorithm can be mapped out as:



A short remark on the layout of *Shor's* algorithm is that the upper registers, after acted on by Hadamard gates, carry the superposition state of multiple powers x, meaning the algorithm computes 2^n values of $a^x mod\ N$ with different values x_s simultaneously, which highlights the advantage of quantum computing over its classical counterpart.

3 Implementation of Shor's Algorithm with Qiskit

In this section, I would introduce the implementation of *Shor's* algorithm on the simplest possible case using *Qiskit*: factoring $N = 15$ with the base $a = 7$[4].

3.1 Circuit of QFT†

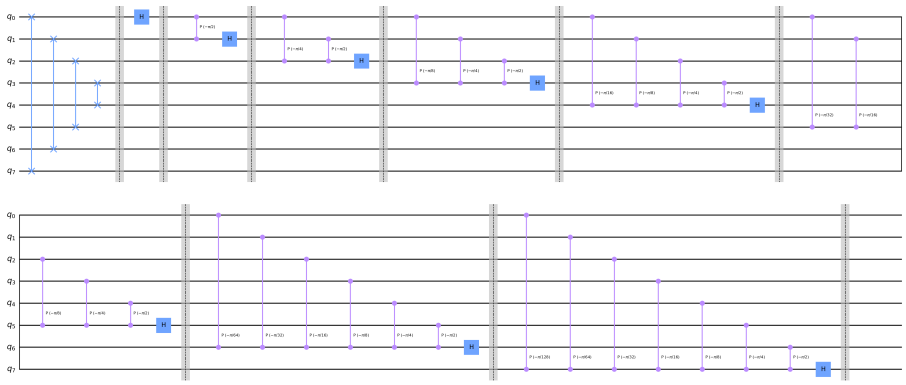


Figure 2: 8-qubit Inverse Fourier Transform

As shown in the previous section, inverse QFT is the final step of *Shor's* algorithm before measurement. In the case of our consideration, we take $2n = 8$ qubits ($2^n = N$) in the upper registers (2^8 powers would be parallelly computed) and therefore an 8-qubit $QFT^\dagger$ (Figure 2) is needed. The construction of $QFT^\dagger$ is rather intuitive, we just need to take the daggers of all the gate operations within the QFT circuit in (2.1) (Namely, flipping the sign of the argument of every phase gate) and reverse the order of the layout. Note that since both SWAP gate and Hadamard gate are Hermitian operators, their inverse operations remain unaltered.

3.2 Circuit of Modular Exponentiation

As shown in (2.3), constructing a unitary gate U that transforms $|x\rangle$ into $|ax \mod N\rangle$ is the major goal. The circuit of modular exponentiation can be broken down into several layers of unitary gates, each of which builds up the unitary gate of the next layer[1]. The first layer is made of a $\phi ADD(a)$

gate that computes the addition of an integer with another given integer a on the Fourier basis. An input $\phi(b)$ subject to this gate will result in the output $\phi(b+a)$. The second layer is the $\phi ADD(a) \mod N$ gate which brings an input $\phi(b)$ to $\phi(b+a \mod N)$, again on the Fourier basis. Up next is the $CMULT(a) \mod N$ gate consisting of a series of controlled-$\phi ADD(2^k a) \mod N$ gate with $k = 0, 1, \ldots, n-1$. This layer conducts the transformation from a state $|b\rangle$ to $|(b+ax) \mod N\rangle$. The final layer of the modular exponentiation is composed of a controlled-$CMULT(a) \mod N$, a controlled-SWAP gate, and a controlled-$CMULT(a^{-1}) \mod N$. With this combination of gates, a unitary operator U_a mapping an input state $|x\rangle$ to an output state $|ax \mod N\rangle$ can be realized.

In theory, a general circuit for modular exponentiation can be built up following such a structure. However, due to the number of qubits and unitary gates needed, such a circuit is not yet executable on currently available quantum computers. Instead, one can simplify the circuit design of our specific case ($N = 15$, $a = 7$) with a little bit of hindsight and then implement it on a quantum simulator. In *Shor's* algorithm, a series of controlled-U^{2^k} gates are included. In this case study, we simply apply the U_a gate 2^k times to produce the U^{2^k} gate. An example circuit carrying out $7^4 \mod 15$ is shown in Figure 3.

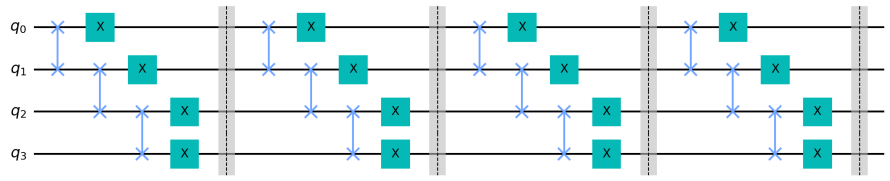


Figure 3: An exemplary circuit that executes $7x \mod 15$ for four rounds. If the input is in state $|1\rangle$, then the circuit computes $7^4 \mod 15$

3.3 Final Implementation: N=15 a=7

With all the required gates, one can now assemble the circuit of *Shor's* algorithm as shown in Figure 4. In the circuit, 8 qubits followed by Hadamard gates are set up in the upper registers to generate 2^8 basis states for the superposition of power, while in the lower registers, 4 qubits are used since $2^4 - 1 = 15$ is the smallest value greater or equal to $N = 15$. The lower register is initialized to be in the $|1\rangle$ state as explained in the last section. Following the initial setup is a chain of controlled-U^{2^k} gates which execute $7^x \mod 15$. Finally, an 8-qubit inverse QFT is applied to the upper registers, which are then measured. The measurement result is shown in Figure 5. The expected outcome is a superposition state of $\left|2^4 \frac{s}{r}\right\rangle$ $(s = 0, 1, \ldots, r-1)$ with equal probability. From the histogram, one can deduce the period $r = 4$. The rest is then carried out

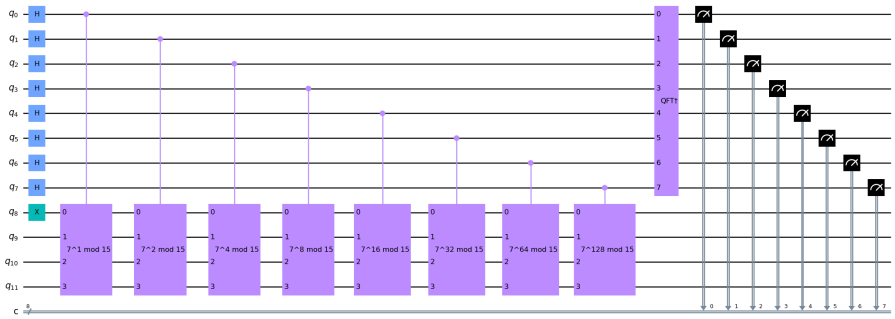


Figure 4: The final layout of *Shor's* algorithm factoring $N = 15$ with the chosen base $a = 7$

on a classical computer to search for the greatest common divisors $gcd(7^{4/2} + 1, 15) = 5$ and $gcd(7^{4/2} - 1, 15) = 3$. Here we have successfully launched the *Shor's* algorithm for the easiest possible case. However, There's still a long way to go to implement a general *Shor's* algorithm for a considerably large N on a scalable quantum computer. Nevertheless, theoretically constructing the *Shor's* algorithm is undoubtedly a revolutionary step in quantum computing and quantum information science.

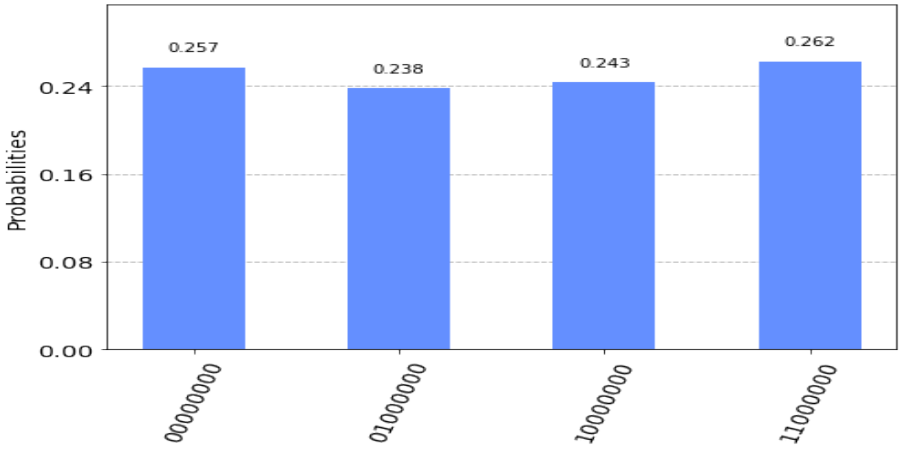


Figure 5: Probability of measuring each state $\left|2^n \frac{s}{r}\right\rangle$ with a quantum simulator. From left to right, the binary value represents 0,64,128 and 192, which corresponds to $\frac{s}{r} = 0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}$

References

[1] S. Beauregard. Circuit for shor's algorithm using 2n+3 qubits. *Quantum Information and Computation*, 3(2):175–185, 2003.

[2] The Qiskit Team. Quantum fourier transform, Nov 2022.

[3] The Qiskit Team. Quantum phase estimation, Nov 2022.

[4] The Qiskit Team. Shor's algorithm, Nov 2022.