※ Computation 3: Quantum Fourier Transform and Shor's Algorithm

14.1 RSA and Number Theory

"We know more than we did before. Let's use that."

- Cypher

We've seen an example of private key cryptography when talking about quantum money, but here we will be interested in **public key cryptography**. In such a scheme, there exists what is called a public key, which anyone can easily know and uses to encrypt a message. On the other hand, each receiving party will have their own private key, which is required to efficiently decrypt a message.

In this section, we will take a look at the RSA protocol, which is one of the most commonly used public key cryptosystems today.

The following is the description of the RSA protocol.

1. Choose two prime numbers p and q.

2. Multiply them to form M.

3. Compute the "Euler function" of M, $\phi(M)$.

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26$$

4. Choose an encryption key e such that
   - $1 < e < \phi(M)$.
   - $\gcd(e, M) = 1$ and $\gcd(e, (p-1)(q-1)) = 1$.

5. Choose d such that
$$de(\text{mod } \phi(N)) = 1. \tag{117}$$

- Public keys are e and M.
- Private keys are p, q, d.

If you know p, q, e, you can use Euclid's algorithm to efficiently compute d.

To encrypt a message, we would use the following protocol. Let m be the plaintext message. The cypher text c is computed by
$$c = m^e \mod M. \tag{118}$$

To decrypt the message, we simply perform

$$m = c^d \mod M. \tag{119}$$

If an attacker intercepted the cypher text c, it would be practically impossible to decode it unless they knew what d was.

Question 142. Suppose the public keys are $e = 5$ and $M = 26$. If we encrypt the message "B" which we will represent using the integer 2, what is the cypher text?

Question 143. Show that $d = 17$ is a valid private key to decrypt the message.

The computer science community believes that finding the plaintext m from the cyphertext c requires solving an exponentially hard problem (namely, factoring an unknown number M). Knowing whether this is true or not has large implications for the security of information, leading to lots of interest in the complexity of the factoring problem.

As usual, let's take a look at how difficult it would be to factor numbers classically. Note that when we analyze the hardness of factoring, we are interested in the number of digits used to represent M, which is $w = O(\log M)$. So an efficient algorithm would mean an algorithm that runs in polynomial time with respect to $w = O(\log M)$.

The trivial algorithm would simply try every number $j \in [1, \sqrt{M}]$ and check if j divides M. This would require $O(2^w)$ iterations.

Other known classical algorithms include

- Quadratic Field Sieve: $O\left(2^{c\cdot\sqrt{w}}\right)$
- Number Field Sieve: $O\left(2^{m^{1/3}}\right)$

Though still exponential, these improvements were enough to require RSA schemes to go up from 512- to 768-bit encryption schemes to the sizes we see today (1024 to 4048 bits).

14.2 Shor's Algorithm

In 1994, Peter Shor developed an algorithm to solve factoring on a quantum computer using just poly($\log M$) gates. Note that this is not even the number of queries, it is the exact circuit size.

Going beyond RSA, another popular public-key cryptosystem is called Diffie-Hellman, which requires solving the discrete log problem (also believed to be difficult for classical computers). Shor's algorithm for factoring is also able to solve discrete log.

Let's first see Shor's factoring algorithm, and we'll dive into the components of it after.

Algorithm 1 qFactor(M)

```
Pick $x$ at random from $\{2, \ldots, M-1\}$
if $\gcd(x, M) \neq 1$ then
    $\gcd(x, M)$ is a non-trivial factor of $M$ so we are done!
else
    Find the smallest $s$ so that $x^s = 1 \mod M$. Call this variable $r$.
    if $r$ is odd then
        Start over..
    else if $x^{r/2} = \pm 1 \mod M$ then
        Start over..
    else
        $x^{r/2} \mod M$ is a non-trivial square root of 1 mod $M$
    end if
end if
Postprocessing: Compute $\gcd(x^{r/2} - 1), M$ and $\gcd(x^{r/2} + 1), M$ to find the factors of
$M$.
```

A key subroutine in Shor's algorithm is that of period finding. In a way, this algorithm shows that factoring an integer can be solved if you know how to find the period of a function quickly.

The function we are searching for the period for is the following:

$$f_x(s) := x^s \mod M \tag{120}$$

where $\gcd(x, M) = 1$ and $M = p \cdot q$.

Before going into how to find the period, let's show that knowing the period does indeed let us know what the factors of M are.

First, we need to confirm that $f_x(s)$ defined above is indeed periodic.

Lemma 14.1. Let x, M be integers such that $\gcd(x, M) = 1$. Then

$$x^s \mod M \tag{121}$$

is periodic in x.

Sketch. If we tried computing the powers of x mod M, since the function can have only M distinct outcomes, if we take more than M powers we will eventually get a repeat.

Let a and b be two powers where the equation evaluates to the same integer. We can express this mathematically:

Since $\gcd(x, M) = 1$, x and M do not share any prime factors. Thus for the equality to hold, M must evenly divide $x^{b-a} - 1$:

$$\begin{aligned} x^{b-a} - 1 &= k'M \qquad (122) \\ x^{b-a} &= 1 + k'M \qquad (123) \\ \Rightarrow x^{b-a} &= 1 \mod M. \qquad (124) \end{aligned}$$

□

Furthermore, we can show that the period of the function is the smallest positive integer s such that $x^s = 1 \mod M$.

In practice, finding a non-trivial square root of 1 mod M is sufficient for factoring.

Lemma 14.2. Given a composite number N and an integer x such that

$$x^2 = 1 \mod N \tag{125}$$

and

$$x \neq \pm 1 \mod N, \tag{126}$$

we can factor N.

Proof. By our assumption, we have that

$$x^2 - 1 = kN \tag{127}$$

$$\Rightarrow (x+1)(x-1) = kN. \tag{128}$$

Furthermore by our second assumption that $x \neq \pm 1 \mod N$, neither $x-1$ nor $x+1$ is a multiple of N. The product is some multiple of N, so what we conclude is that each of $(x-1)$ and $(x+1)$ have some of N's prime factors. To find one of these, we simply compute

$$\gcd(x+1, N) \tag{129}$$

$$\gcd(x-1, N). \tag{130}$$

□

Example 14.3. Let $N = 15$. A number that satisfies the first condition is $x = 11$:

$$11^2 = 121 = 1 \mod 15. \tag{131}$$

We can also easily verify that $11 \neq \pm 1 \mod 15$. Now we compute the gcd of the pair of integers around 11 with 15 to recover the factors:

$$\gcd(12, 15) = 3 \tag{132}$$

$$\gcd(10, 15) = 5. \tag{133}$$

We have successfully recovered the factors 3 and 5.

Note that for an application like RSA, the number N we are trying to factor will always be a composite of two primes, meaning that the gcd will be sufficient for finding these numbers.

Question 144. Find a non-trivial square root of 1 mod 20.

14.3 Quantum Fourier Transform

Before studying the quantum period finding algorithm, let's take a quick look at one of the key pieces required. We begin by reviewing how to transition between the bit string and integer representation. To compute the value of the bit string 11001, we multiply it as follows:

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 1 = 25. \tag{134}$$

More generally, if we have a string $x = x_1 \cdots x_n$ where x_i is the i-th bit of x, we can write this as an integer using the sum

$$x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \cdots + x_n \cdot 2^0 = \sum_{k=1}^{n} x_k \cdot 2^{n-k}. \tag{135}$$

Throughout this section, we will frequently take complex numbers to the power of integers, and it will be useful to have a way to toggle between the integer representation of the power and the bit string representation.

$$\begin{aligned}
\omega^x &= \omega^{x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \cdots + x_n 2^0} && (136)\\
&= \omega^{x_1 \cdot 2^{n-1}} \cdot \omega^{x_2 \cdot 2^{n-2}} \cdot \cdots \cdot \omega^{x_n \cdot 2^0} && (137)\\
&= \prod_{k=1}^{n} \omega^{x_k \cdot 2^{n-k}} && (138)
\end{aligned}$$

One important reason we will be interested in complex numbers is because they are a great tool for analyzing periodic functions. In particular, the N-th roots of unity give us a way to keep track of the "period" in the way a clock would.

Question 145. Let $n = 3$. Let ω be the first 2^3-th root of unity. Compare the values ω^x and ω^{x+1} for $x = 101$.

Definition 14.4 (Quantum Fourier Transform)**.** Let $N = 2^n$. For $x \in \{0, 1, \ldots, N-1\}$:

- Standard basis state: A length N column vector with a 1 in the k-th location

$$|k\rangle = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}. \tag{139}$$

- Fourier basis state ($\omega = e^{2\pi i/N}$ is the first N-th root of unity):

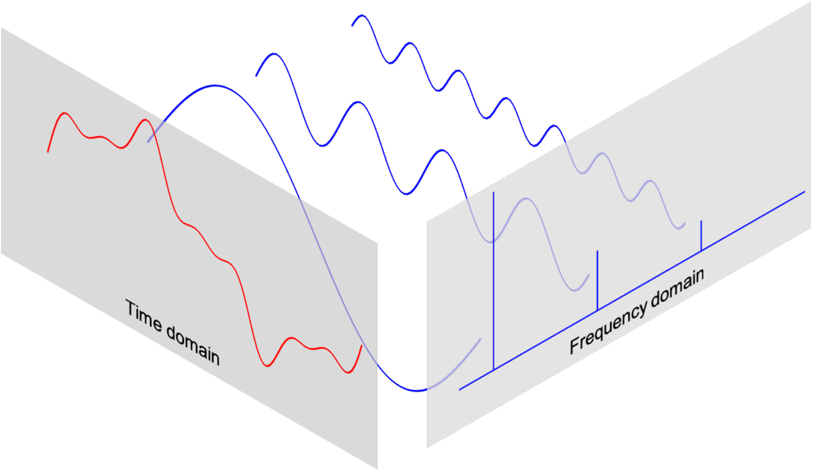
$$|\tilde{k}\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} \omega^0 \\ \omega^k \\ \omega^{2k} \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}. \tag{140}$$

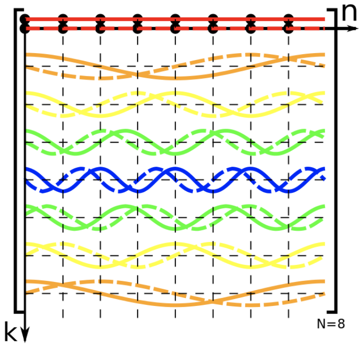
The n-qubit **Quantum Fourier Transform** or QFT_N is the transformation that performs

$$QFT_N |x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle \tag{141}$$

We can model the action of QFT_N by the matrix

$$QFT_N := \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}. \tag{142}$$





(a) Taken from Python Numerical Methods

Let’s get familiar with the Fourier basis by working through an explicit example.

Question 146. Write down the 1-qubit QFT matrix. Use the matrix to find all of the 1-qubit Fourier basis states.

Question 147. Write down the 2-qubit QFT matrix. Use the matrix to find the first 2-qubit Fourier basis state, $|\tilde{1}\rangle$.

14.4 Properties of the Fourier Transform

The power of the QFT is in its ability to *represent* periodic sequences or functions. This may mean that it is also effective at *finding* periodic structure in sequences as well! Before exploring this thought, let's look at some properties of the Fourier Transform.

Let me introduce some more notation we will use through the remainder of this section. We've learned and practiced the mapping from a standard basis state to a Fourier basis state. What happens if we apply the Fourier transform to a superposition of standard basis states?

Question 148. Consider the two qubit state

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle + \alpha_3 |3\rangle \,. \tag{143}$$

What is the result of applying the QFT to this state?

More generally, if we have an n qubit state

$$|\psi\rangle = \sum_{x=0}^{N-1} \alpha_x |x\rangle \tag{144}$$

and apply QFT_N to it, we get the following:

$$\begin{aligned} QFT_N |\psi\rangle &= QFT_N \left(\sum_{x=0}^{N-1} \alpha_x |x\rangle \right) && (145) \\ &= \sum_{x=0}^{N-1} \alpha_x QFT_N |x\rangle && (146) \\ &= \sum_{x=0}^{N-1} \alpha_x \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x \cdot y} |y\rangle && (147) \\ &= \sum_{y=0}^{N-1} \underbrace{\sum_{x=0}^{N-1} \frac{1}{\sqrt{N}} \alpha_x \omega^{x \cdot y}}_{\widetilde{\alpha_y}} |y\rangle && (148) \end{aligned}$$

We will call the amplitudes $\widetilde{\alpha_y}$ the Fourier amplitudes. Using the vector notation, this means the QFT is doing the following transformation:

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-2} \\ \alpha_{N-1} \end{bmatrix} \xrightarrow{QFT_N} \begin{bmatrix} \widetilde{\alpha_0} \\ \widetilde{\alpha_1} \\ \vdots \\ \widetilde{\alpha_{N-2}} \\ \widetilde{\alpha_{N-1}} \end{bmatrix} \tag{149}$$

Question 149. Consider the state $|\psi\rangle$ from the previous question. What is $\widetilde{\alpha_3}$?

14.4.1 The Fourier transform converts between translation and phase

Let

$$|\psi\rangle = \sum_{x=0}^{N-1} \alpha_x |x\rangle \tag{150}$$

and

$$|\psi_{+j}\rangle = \sum_{x=0}^{N-1} \alpha_x |x + j \bmod N\rangle = \sum_{x=0}^{N-1} \alpha_x |x + j\rangle . \tag{151}$$

The $+j$ state is simply the state we get by shifting the amplitudes cyclically by j positions. The last inequality is just a notational difference, where we will abbreviate the mod N in this notation.

Question 150. Suppose we have the following two qubit state:

$$|\psi\rangle = \frac{1}{2}|0\rangle + \frac{i}{2}|1\rangle + \frac{\sqrt{3}}{4}|2\rangle + \frac{\sqrt{5}}{4}|3\rangle . \tag{152}$$

What is $|\psi_{+1}\rangle$?

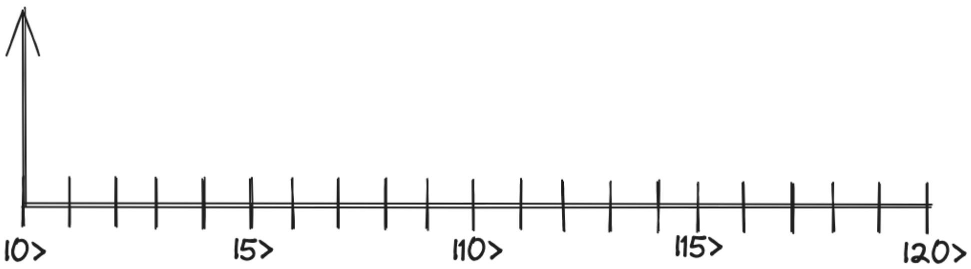
Question 151. Apply the QFT to $|\psi_{+j}\rangle$.

14.4.2 Fourier Transform of Factors

Suppose r is an integer that divides N (N/r is an integer). Define the state

$$|\phi_r\rangle = \sqrt{\frac{r}{N}} \sum_{k=0}^{\frac{N}{r}-1} |kr\rangle \,. \tag{153}$$

Question 152. Let $N = 21$ and $r = 3$. What are the amplitudes that appear in this vector? Draw them in the figure below.



Proposition 14.5. Let r be an integer that divides N. Then,

$$QFT_N\,|\phi_r\rangle = |\phi_{N/r}\rangle \,. \tag{154}$$

Question 153. Continuing the example from Question 42, what is $|\phi_{N/r}\rangle$?

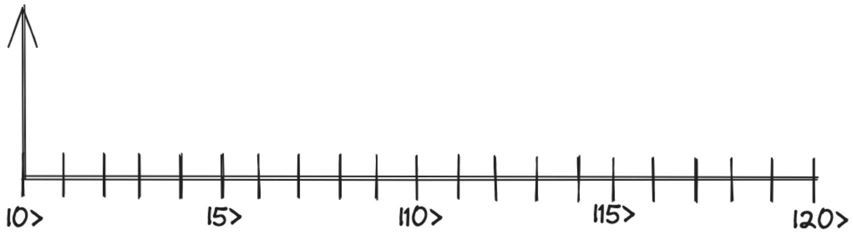
Question 154. Still continuing the example from Question 42, what is $QFT_N\,|\phi_r\rangle$?

14.4.3 Periodic Superpositions with a Shift

The crux of Shor's algorithm is the solution to the following problem: Suppose we have a periodic superposition but with a shift:

$$\sqrt{\frac{r}{N}} \sum_{k=0}^{\frac{N}{r}-1} |kr + l\rangle \,. \tag{155}$$

Can we find an algorithm to find r?



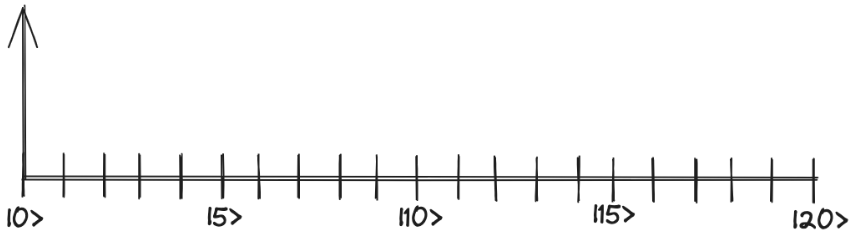
To further complicate the problem, every time we generate a new version of the state, the shift constant l will be different. This causes issues later, because when we use the QFT for factoring, the offset does not allow us to cleanly divide N in the way that it does in Proposition 2.2.

Let's formalize the problem. We consider starting with some state

$$|\phi_r\rangle = \frac{1}{\sqrt{s}} \sum_{k=0}^{s-1} |kr + l\rangle \tag{156}$$

where we have defined $0 \le l < r$ and $s = \left\lfloor \frac{N}{r} \right\rfloor$. There are two challenges we will have to resolve to correctly figure out what r is.

1. N will generally be a power of 2, meaning that r does not divide N most of the time.
2. The shift l causes the QFT to not work as cleanly.



Proposition 14.6. Consider applying an n-qubit QFT to some state

$$|\phi_r\rangle = \frac{1}{\sqrt{s}} \sum_{k=0}^{s-1} |kr + l\rangle\,. \tag{157}$$

We can express this state as

$$QFT_N\,|\phi_r\rangle = \sum_{a=0}^{N-1} \widetilde{\alpha_a}\,|a\rangle\,. \tag{158}$$

If we measure this output state, then with high probability we measure a value a such that

1. $\left|a - k\frac{N}{r}\right| \leq \frac{1}{2}$ for some k.
2. $\gcd(k, r) = 1$.

The first property can be rewritten as

$$\left|\frac{a}{N} - \frac{k}{r}\right| \leq \frac{1}{2N}. \tag{159}$$

From our measurement result, we know what a is, and N depends on the system size. What the above equation is saying is that $\frac{a}{N}$ is a *very* close approximation of $\frac{k}{r}$. We can use this information to find k and r, independent of what l is!

14.5 Period Finding

Let's see a period finding algorithm

Suppose we have some function $f : \{0, \ldots, N-1\} \to \{0, \ldots, M-1\}$ or alternatively in the bitstring notation, $f : \{0,1\}^n \to \{0,1\}^m$. Here, we've defined the integers such that $N = 2^n$ and $M \leq 2^m$. This function will also have the property that for some integer r,

$$f(x) = f(y) \Leftrightarrow x - y \text{ is an integer multiple of } r \leq M < \sqrt{N}. \tag{160}$$

Given quantum access to this function via a unitary U_f, the goal is to find r.

Example.

This problem is interesting because there is no known efficient classical algorithm! The best we can do is a brute force search for "collisions" $f(x) = f(y)$, which on average will require $\sim 2^{n/2}$ time.

Question 155. Consider the function $f : \{0,1\}^4 \to \{0,1\}^4$, where

$$f(s) = 3^s \mod 16. \tag{161}$$

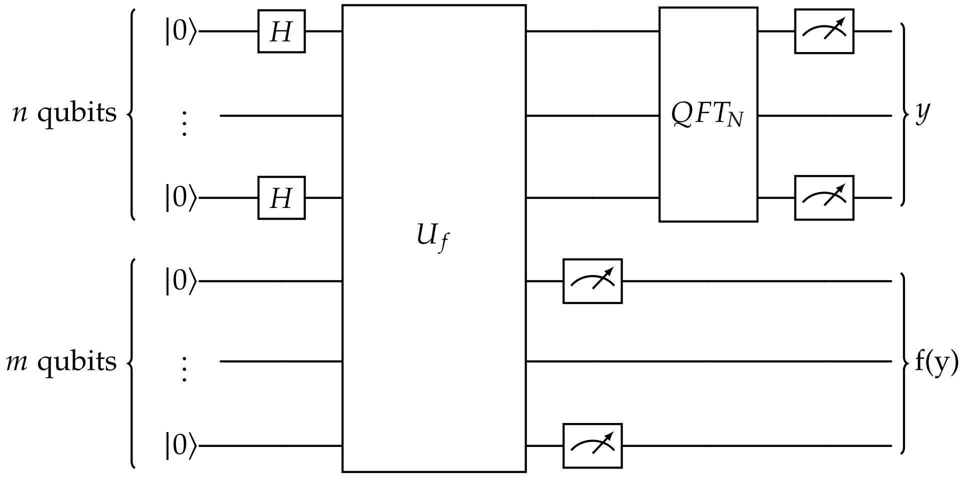
What is the period of f?

I'll begin by just writing out the algorithm, and we will go over the analysis after seeing it. The algorithm will use two registers, the first using n qubits (to store numbers mod N), and the second register will have $\lceil \log M \rceil$ qubits to store numbers mod M. We will also have quantum access to the function f via a unitary U_f which has the action

$$U_f |x\rangle |0\rangle = |x\rangle |f(x)\rangle . \tag{162}$$

Algorithm 2 Period Finding

```
1: Start with $|0\dots0\rangle |0\dots0\rangle$
2: Apply $H^{\otimes n}$ on the first register to get $\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle |0\rangle$
3: Apply $U_f$
4: Measure the 2nd register.
5: Ignore the 2nd register and apply $QFT_N$(mod $N$) to the 1st register.
6: Measure the 1st register to get value $a$.
7: Postprocessing: Use $a$ and $N$ to find $k$ and $r$.
```



Let's go through each step starting at 3 to see what the state is. At step 3, we will have prepared the state

$$\frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle |f(y)\rangle \tag{163}$$

Now recall that the function $f(y)$ is periodic in r, meaning that for every $f(y)$, there will be r inputs that map to it.

Question 156. Consider the function we used in Question 45. If we had unitary access to this function and measured $|9\rangle$ in the second register, what is the state of the algorithm?

More generally, measuring the second register will allow us to collapse into a superposition of inputs that all map to the same state. More concretely, suppose we measured the value w in the second register. Then the first register will be an even superposition of all $|jr + l\rangle$ such that $f(jr + l) = w$, and $0 \leq j < \lfloor \frac{N}{r} \rfloor$, $0 \leq l \leq r - 1$. We can write this in ket notation too:

$$\frac{1}{\sqrt{s}} \sum_{j=0}^{s-1} |jr + l\rangle |w\rangle \qquad s := \left\lfloor \frac{N}{r} \right\rfloor. \tag{164}$$

We can safely ignore the second register for the remaining analysis, because the two registers are now in a product state.

By Proposition 2.3, we know that applying a QFT to this state and measuring will allow us with high probability to observe a state a such that

$$\left| a - k\frac{N}{r} \right| \leq \frac{1}{2} \qquad \gcd(k, r) = 1. \tag{165}$$

The above equation can be combined with our assumption that $r < \sqrt{N}$ to get

$$\left| \frac{a}{N} - \frac{k}{r} \right| \leq \frac{1}{2N} \leq \frac{1}{2r^2}. \tag{166}$$

We will be using **continued fractions** to find k and r.

We will assume that $\gcd(k, r) = 1$. To analyze the final step, let's look at how continued fractions works. Let $\gamma = \frac{a}{N}$. We know that with high probability,

$$\left| \gamma - \frac{k}{r} \right| \leq \frac{1}{2r^2}. \tag{167}$$

The continued fractions representation of a real number is a sequence of integers $a_0, a_1, \ldots, a_n$ as:

$$\gamma \approx a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots \frac{1}{a_n}}}} \tag{168}$$

We can clip this sequence at some a_k to get an approximation of γ. We will define the P_k and Q_k as the numerator and denominator we get respectively for the approximation of γ.

Question 157. Let $\gamma = 7.27$. Find the continued fractions representation of γ. What is P_3 and Q_3?

Once the continued fractions representation is found, we get a series of approximations to γ:

$$\frac{P_0}{Q_0}, \frac{P_1}{Q_1}, \frac{P_2}{Q_2}, \cdots, \frac{P_n}{Q_n} \tag{169}$$

such that

$$Q_0 < Q_1 < Q_2 < \cdots < Q_n. \tag{170}$$

Each of the Q_j's are candidates for r, and with access to the function f we can directly test the n candidates to see if we have the correct period. We know this will happen by the theorem stated below.

Theorem 14.7 (Proven in Nielsen and Chuang)**.** If $\left|\gamma - \frac{k}{r}\right| \leq \frac{1}{r^2}$, then $k = P_j$ and $r = Q_j$ for some j.

Our algorithm will try each one and take the smallest Q_j such that

$$x^{Q_j} = 1 \mod N. \tag{171}$$

We end the section with two important facts:

- If γ is a rational number, eventually for some n, $\frac{P_n}{Q_n} = \gamma$.
- $\frac{P_j}{Q_j}$ is the best approximation to γ by *any* ratio of integers whose denominator is $\leq Q_j$.