Some quantum algorithms

Nielsen and Chuang, Chapter 5

We know that a quantum computer can efficiently simulate quantum dynamics

We know that it can efficiently simulate a classical computer

But what else can it do?

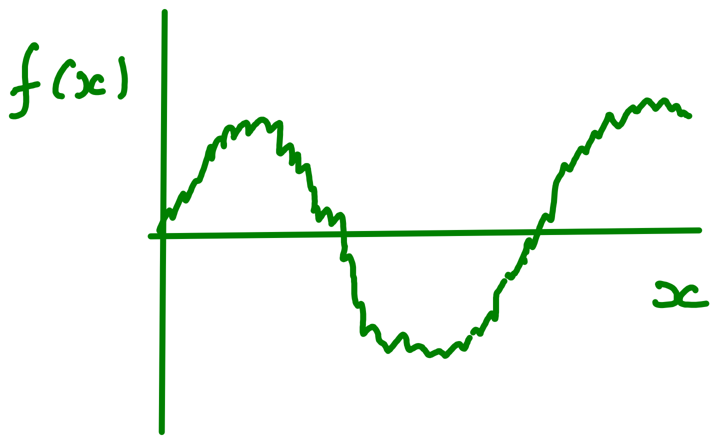
Today we will start to look at some algorithms that are unrelated to physics, all based on the quantum Fourier transform

Quantum Fourier Transform

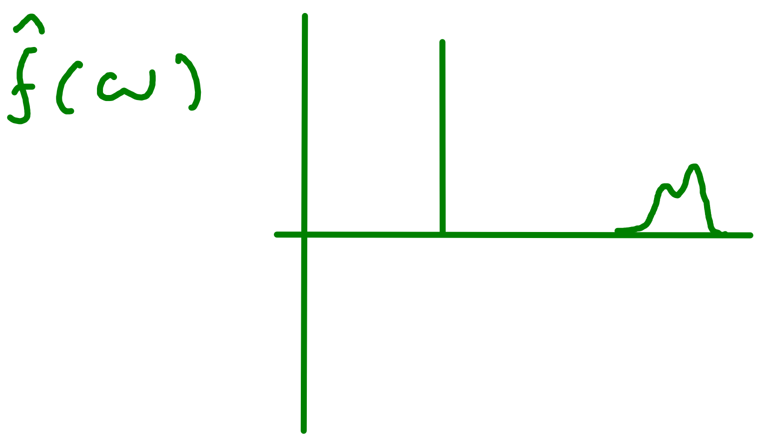
We know about the Fourier transform

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i2\pi x\omega}\, dx$$

This takes a function and outputs its spectrum



$\longrightarrow$



Useful in many applications, such as when a function has some periodicity that must be found and analyzed

A discreet version (DFT) can also be defined, where instead of a function we have a list of values (in a vector)

$$|f\rangle = \sum_{j=0}^{N-1} f_j \, |j\rangle$$

The transform acts on basis states according to

$$|j\rangle \rightarrow |\hat{j}\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i2\pi jk/N} \, |k\rangle$$

And so acts on a general vector as

$$|f\rangle = \sum_{j=0}^{N-1} f_j \, |j\rangle \rightarrow |\hat{f}\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} f_j \, e^{i2\pi jk/N} \, |k\rangle$$

We consider the case that $N = 2^n$ and express the basis

$|0\rangle, |1\rangle, |2\rangle, \ldots |N-1\rangle$

In binary (and so as n qubits)

$|0\ldots00\rangle, |0\ldots01\rangle, |0\ldots10\rangle, \ldots |1\ldots11\rangle$

So for a general Z basis state

$$|j\rangle = |j_1\, j_2\, j_3 \dots j_n\rangle \quad , \quad j = \sum_{l=1}^{n} j_l\, 2^{n-l}$$

Let's also consider the following notation for binary fractions (numbers less than 1 expressed in binary)

$$0.j_1 j_2 \dots j_m = \sum_{l=1}^{m} j_l\, 2^{-l}$$

Now lets see if we can simplify the Fourier transform basis states a bit

$$|\hat{j}\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{i2\pi jk/2^n} |k\rangle = \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} e^{i2\pi j\left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1\, k_2 \dots k_n\rangle$$

Convert k into binary

$$k = \sum_{l=1}^{n} k_l 2^{n-l} \quad \therefore \quad \frac{k}{2^n} = \sum_{l=1}^{n} k_l 2^{-l}$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \dots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{i2\pi j k_l 2^{-l}} |k_l\rangle$$

$$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{K=0}^{1} e^{i2\pi j K 2^{-l}} |K\rangle \right]$$

Using

$$\prod_{x=1}^{n} \sum_{y} f(x,y) = \sum_{y_1 \dots y_n} \prod_{x} f(x,y)$$

$$= \bigotimes_{l=1}^{n} \frac{|0\rangle + e^{i2\pi j 2^{-l}} |1\rangle}{\sqrt{2}}$$

So it turns out to be a product state

$$|\hat{j}\rangle = \bigotimes_{l=1}^{n} \frac{|0\rangle + e^{i2\pi j 2^{-l}}|1\rangle}{\sqrt{2}}$$

Let's also convert j to binary

$$j = \sum_{k=1}^{n} j_k 2^{n-k} \quad \therefore \quad \frac{j}{2^l} = \sum_{k=1}^{n} j_k 2^{n-k-l} = \sum_{k=1}^{n-l} j_k 2^{(n-l)-k} + \sum_{k=n-l+1}^{n} j_k 2^{n-k-l}$$

$$\left(\begin{aligned} -k' &= n-k-l \\ \therefore k' &= k-(n-l) \\ k &= k'+(n-l) \\ &= n-(l-k') \end{aligned} \right) \longrightarrow = \text{Integer} + \sum_{k'=1}^{l} j_{n-(l-k')} 2^{-k'}$$

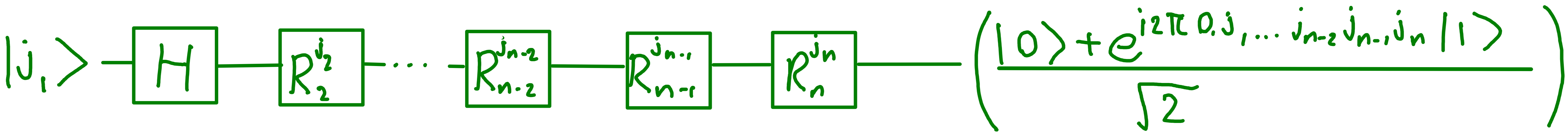
$$= \text{integer} + 0.j_{n-(l-1)} j_{n-(l-2)} \dots j_n$$

$$\therefore e^{i2\pi j 2^{-l}} = e^{i2\pi 0.j_{n-(l-1)} j_{n-(l-2)} \dots j_n}$$

$$|\hat{j}\rangle = \left(\frac{|0\rangle + e^{i2\pi 0.j_n}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + e^{i2\pi 0.j_{n-1}j_n}|1\rangle}{\sqrt{2}}\right) \otimes \dots \otimes \left(\frac{|0\rangle + e^{i2\pi 0.j_1 \dots j_n}|1\rangle}{\sqrt{2}}\right)$$

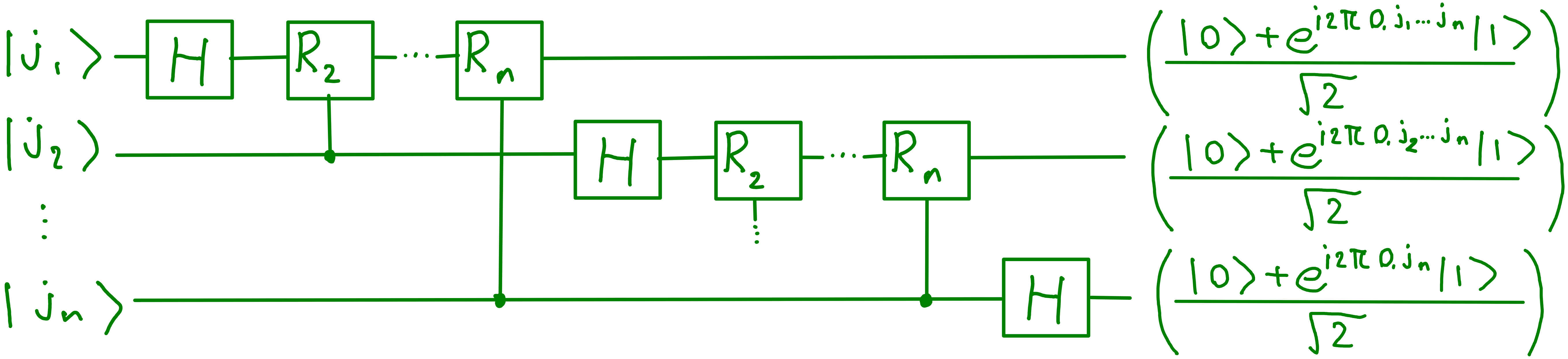
This product represention allows us to see how to perform the DFT on a quantum computer

For the last qubit we could use $R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^k} \end{pmatrix}$



Better to use controlled ops so we can deal with a superposition of different j's

We then find that the circuit



Peforms the FT (and reverses qubit order)

This clearly requires $O(n^2)$ gates

So the DFT (and its inverse) can be implemented on a quantum state efficiently by a quantum computer

The fastest known classical algorithm requires $O(n2^n)$

So can we use quantum computers to do fast DFTs?

Yes and No

'No' because preparing a general state to be transformed is inefficient, even if the transformation itself is efficient

So we cannot use it to do a DFT on any vector that we be interested from a real-world problem

'Yes' because it can be used as a component in larger quantum algorithms that do have efficient read-in and read-out

Phase Estimation

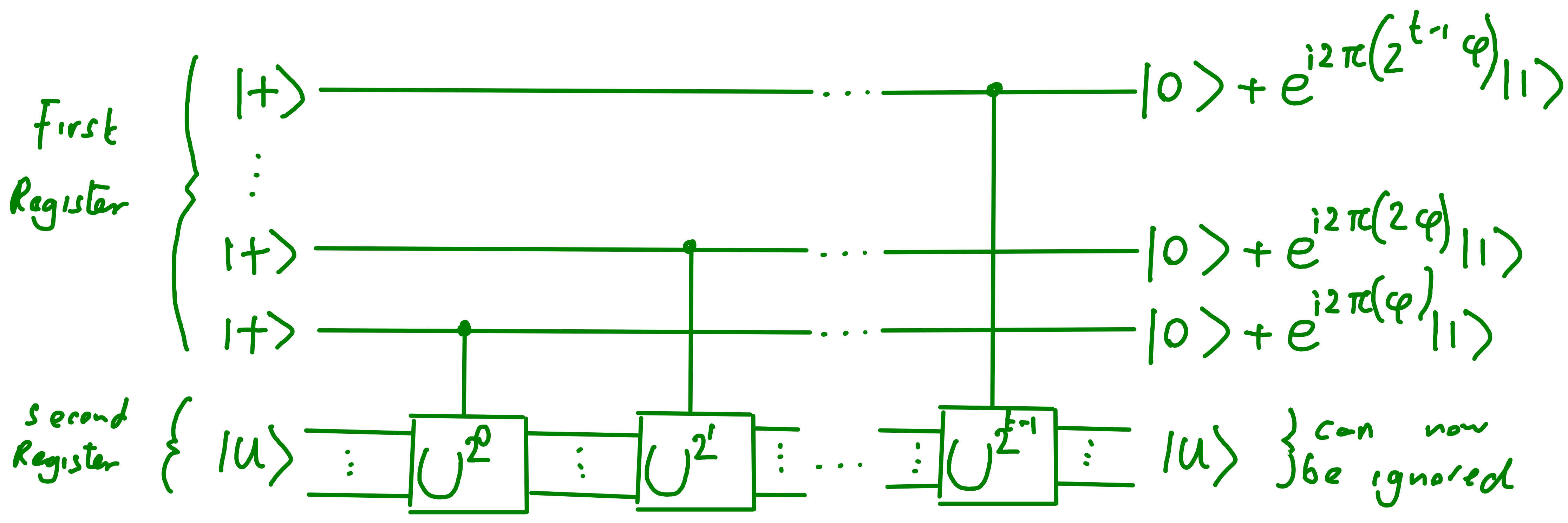
Consider a unitary operation for which we know an eigenstate, and wish to find out the corresponding eigenvalue

$$U|u\rangle = e^{i2\pi\varphi}|u\rangle \qquad \varphi = 0.\varphi_1\varphi_2\ldots\varphi_t$$

t is # bits required to express φ

Assume that we have the ability to prepare the eigenstate and apply a controlled-U

This means we can apply the circuit



Outcome for the first register is

$$\bigotimes_{l=t}^{1} \frac{|0\rangle + e^{i2\pi\varphi 2^{l-1}}|1\rangle}{\sqrt{2}}$$

Let's change our variable a little

$$\varphi = 0.\varphi_1\varphi_2\cdots\varphi_t \quad \therefore \quad \varphi_1\varphi_2\ldots\varphi_t = 2^t\varphi = \phi \quad \therefore \quad \varphi = \phi\, 2^{-t}$$

$$\bigotimes_{l=t}^{1} \frac{|0\rangle + e^{i2\pi\phi\, 2^{l-1-t}}|1\rangle}{\sqrt{2}} = \bigotimes_{l'=1}^{t} \frac{|0\rangle + e^{i2\pi\phi\, 2^{-l'}}|1\rangle}{\sqrt{2}} = |\hat{\phi}\rangle \qquad (l' = 1+t-l)$$

So the outcome is the FT of the state $|\phi\rangle = |\varphi_1\varphi_2\ldots\varphi_t\rangle$

$$\left(\text{recall } |\hat{j}\rangle = \bigotimes_{l=1}^{n} \frac{|0\rangle + e^{i2\pi j 2^{-l}}|1\rangle}{\sqrt{2}}\right)$$

Performing the inverse FT and measuring the state in the Z basis then gives the binary representation of the phase

Final state: $|\phi\rangle \otimes |u\rangle$

Note that this method assumes

the phase can be written using a finite number of bits, t

we know what t is (or at least an upper bound)

In general, this is not the case

However, even if the t we use is too small, it will give a good approximation

To get the phase accurate to n bits with high probability, we need to use

$$t = n + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil, \text{ accurate with probability } 1-\varepsilon$$

Which is efficient

But can phase estimation be used for anything useful?

Order Finding

Consider the positive integers x and N for which $x < N$ and there are no common factors

What is the smallest possible integer r such that

$$x^r = 1 \mod N$$

This is called the order of x modulo N

It is believed that no poly(L) algorithm exists to compute this on a classical computer, where L is the number of bits needed to specify N $L = \lceil \log N \rceil \therefore 2^L \geq N$

To compute it with a quantum computer, consider the operator

$$U = \sum_{j=0}^{2^L-1} |xj \mod N\rangle\langle j|$$

Where we use the convention

$$xj \mod N = j \quad \text{for} \quad N \leq j \leq 2^L - 1$$

The eigenstates of this are

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-i2\pi sk}{r}\right] |x^k \bmod N\rangle$$

With eigenvalues (exercises)

$$e^{i2\pi\varphi(s)} = \exp\left[\frac{i2\pi s}{r}\right] \quad \therefore \quad \varphi(s) = \frac{s}{r}$$

If we can use phase estimation to find these, we can find r

For that we need to efficiently perform the controlled-U's

Efficient methods exist for this (next week)

We also need to prepare eigenvalues of U

This cannot be done efficiently, so is there another option?

Consider the superposition of the first r eigenstates

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = \frac{1}{r}\sum_{s=0}^{r-1}\sum_{k=0}^{r-1}\exp\left[\frac{-i2\pi sk}{r}\right]|x^k \bmod N\rangle$$

$$= \frac{1}{r}\sum_{k=0}^{r-1}\left(\sum_{s=0}^{r-1}\exp\left[\frac{-i2\pi sk}{r}\right]\right)|x^k \bmod N\rangle$$

$\sum_{s=0}^{r-1}\exp\left[\frac{-i2\pi sk}{r}\right] = \delta_k r$, due to the properties of sums of roots of unity (next slide)

$$\therefore \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |x^0\rangle = |1\rangle = |00\ldots01\rangle$$

This can be efficiently prepared

Roots of unity can be written

$$\omega = \exp\left[-\frac{i2\pi}{r}\right] \quad \therefore \quad \omega^r = \omega^0 = 1$$

Summing all powers of roots of unity gives zero

$$\sum_{s=0}^{r-1} \omega^s = 0$$

For example

$r=2$: $\omega = -1$, $\omega^0 + \omega = 1 - 1 = 0$

$r=4$: $\omega = i$, $\omega^0 + \omega + \omega^2 + \omega^3 = 1 + i - 1 - i = 0$

The same is true integer powers of roots of unity

$$\sum_{s=0}^{r-1} \omega^{ks} = 0 \quad \text{for } k \in \{1, \dots, r-1\}$$

z.B. $r=4, k=3$: $\omega^k = -i$, $(\omega^k)^0 + \omega^k + \omega^{2k} + \omega^{3k} = 1 - i - 1 + i = 0$

But things are obviously different if the power is zero (or r)

$$\sum_{s=0}^{r-1} \omega^0 = r$$

Putting it all together

$$\sum_{s=0}^{r-1} \omega^{ks} = \delta\, r$$

If this is used as the input state of the second register and the phase estimation algorithm is applied, the final state is

$$\sum_{s=0}^{r-1} |\varphi(s)\rangle \otimes |u_s\rangle , \qquad \varphi(s) \approx \frac{s}{r}$$

By applying the method $O(r) = O(L)$ times, we can find (approximations of) all the phases $\varphi(s)$, $s = 0, \ldots, r-1$

But since $r = 2^{O(L)}$ this would be inefficient

Fortunately we need only one (randomly chosen) phase

$$\varphi \approx \frac{s}{r}$$

For unknown s and r

These unknowns can be determined by the continued fractions algorithm if the phase is sufficiently accurate

The relevant theorem

If $\left|\frac{s'}{r'} - \varphi\right| \leq \frac{1}{2r'^2}$, for L bit integers s' and r' with no common factors

then the continued fractions algorithm can compute s' and r' from φ in $O(L^3)$ time

Since the phase is accurate to n bits with, we have

$$\left|\frac{s}{r} - \varphi\right| \leq 2^{-n}$$

So for the theorem to apply we require

$$2^{-n} \leq \frac{1}{2r^2} \quad \therefore \quad n \geq 1 + 2\log r \geq 2L+1 \quad \text{since} \quad r \leq N \leq 2^L$$

For a good enough approximation, we need to use

$$t = 2L + 1 + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right) \right\rceil = O(L)$$

bits on the first register, which is efficient

Problem: s and r may have common factors, so the s' and r' output by continued fractions may not be the numbers we want

$\varphi = \frac{s}{r} = \frac{s'}{r'}$ $\quad s' \leq s \,,\; r' \leq r$

But recall the definition of r. It is the smallest integer such that $x^r = 1 \mod N$

We can efficiently (and classically) check if $x^{r'} = 1 \mod N$

If it is, we know that $r' = r$

If not, we can try again until we get it right

This will certainly occur if s is prime, which occurs with probability $O\left(\frac{1}{\log r}\right) = O\left(\frac{1}{\log N}\right)$

So only $O(\log N)$ repetitions are required until r is found

Better methods with only O(1) repetitions also exist

Another problem: Approximation of the phase is bad with probability ε

This probability is efficiently suppressed by using a large enough register

$$t = O\left(\log\left(2+\frac{1}{2\varepsilon}\right)\right) = O(\log(M))$$

where $M = \frac{1}{\varepsilon}$ is the expected # successful runs before an error

So we can find r efficiently with a quantum computer using

modular exponentiation $O(L^3)$ (or less)

Fourier transform $O(L)$

continued fractions $O(L^3)$

repetitions $O(1)$

Total complexity is $\left(O(L^3) + O(L) + O(L^3)\right) O(1) = O(L^3)$