Department of Information Engineering
MSc in Computer Engineering (a.y. 2024/2025)
University of Pisa

Quantum Computing and Quantum Internet

Luciano Lenzini
Full Professor
Department of Information Engineering
School of Engineering
University of Pisa, Italy
e-mail: lenzini44@gmail.com
http://www.iet.unipi.it/~lenzini/
http://www.originiinternetitalia.it/it/

DISCRETE FOURIER TRANSFORM (DFT)

Discrete Fourier Transform

- One of the most useful ways of solving a problem in mathematics or computer science and engineering is to transform it into some other problem for which a solution is known
- There are a few transformations of this type which appear so often and in so many different contexts that the transformations are studied for their own sake

- A great discovery of quantum computation has been that some such transformations can be computed **much faster** on a quantum computer than on a classical computer, a discovery that has enabled the construction of fast algorithms for quantum computers

Discrete Fourier Transform

- One such transformation is the **Discrete Fourier Transform (DFT)**
- In the usual mathematical notation, the discrete Fourier transform takes as input a vector of complex numbers, $a_0, a_1, \ldots, a_{N-1}$ where the length *N* of the vector is a fixed parameter
- It outputs the transformed data, a vector of complex numbers $\phi_0, \phi_1, \ldots, \phi_{N-1}$ defined by

$$\phi_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi ijk/N}, \qquad k \in \{0, 1, \ldots., N-1\} \qquad [1]$$

DFT

$$\phi_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi ijk/N}, \qquad k \in \{0,1,\ldots.,N-1\}$$

- Calculating just one ϕ_k using Eq. [1] requires adding together *N* terms
- Since there are *N* different ϕ_k's, altogether, this is a total of N^2 terms
- Although this $O\left(N^2\right)$ runtime is efficient in the language of computational complexity, since it is a polynomial in *N*, in practice, it can be quite slow because *N* can be very large

Discrete Fourier Transform

- Fortunately, faster *classical algorithms* for the discrete Fourier transform exist that only take $O(N\log(N))$ steps
- These are called *Fast Fourier Transform (FFT)* algorithms
- The precise workings of these algorithms are beyond the scope of this course, but they are used by computer algebra systems, like *Mathematica* and *SageMath*

DFT

$$\phi_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi i jk/N} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j \left(e^{2\pi i/N}\right)^{jk} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j (\omega)^{jk}$$

- Another way to interpret Eq. [1] is as a matrix-vector multiplication
- To write it more cleanly, let us define $\omega = e^{2\pi i/N}$, then, Eq. [1] becomes

$$\phi_0 = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^0 \quad \rightarrow \quad \phi_0 = \frac{1}{\sqrt{N}} \left(a_0 + a_1 + a_2 + \cdots + a_{N-1}\right),$$

$$\phi_1 = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi i j/N} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j \omega^j \quad \rightarrow \quad \phi_1 = \frac{1}{\sqrt{N}} \left(a_0 + a_1\omega + a_2\omega^2 + \cdots + a_{N-1}\omega^{N-1}\right),$$

$$\phi_2 = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi i j/N} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j \omega^{2j} \quad \rightarrow \quad \phi_2 = \frac{1}{\sqrt{N}} \left(a_0 + a_1\omega^2 + a_2\omega^4 + \cdots + a_{N-1}\omega^{2(N-1)}\right),$$

$$\vdots$$

$$\phi_{N-1} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi i j/N} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j \omega^{(N-1)j} \quad \rightarrow \quad \phi_{N-1} = \frac{1}{\sqrt{N}} \left(a_0 + a_1\omega^{N-1} + a_2\omega^{2(N-1)} + \cdots + a_{N-1}\omega^{(N-1)^2}\right).$$

Discrete Fourier Transform

- The above equations can be written as

$$\begin{aligned}
\phi_0 &= \frac{1}{\sqrt{N}}\left(a_0 + a_1 + a_2 + \cdots + a_{N-1}\right),\\
\phi_1 &= \frac{1}{\sqrt{N}}\left(a_0 + a_1\omega + a_2\omega^2 + \cdots + a_{N-1}\omega^{N-1}\right),\\
\phi_2 &= \frac{1}{\sqrt{N}}\left(a_0 + a_1\omega^2 + a_2\omega^4 + \cdots + a_{N-1}\omega^{2(N-1)}\right),\\
&\vdots\\
\phi_{N-1} &= \frac{1}{\sqrt{N}}\left(a_0 + a_1\omega^{N-1} + a_2\omega^{2(N-1)} + \cdots + a_{N-1}\omega^{(N-1)^2}\right).
\end{aligned}
\rightarrow
\begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N-1} \end{bmatrix}
= \frac{1}{\sqrt{N}}
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\
1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)^2}
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-1} \end{bmatrix}
\quad [2]$$

Quantum Fourier Transform

- The *Quantum Fourier Transform (QFT)* is exactly the same transformation, although the conventional notation for the quantum Fourier transform is somewhat different
- The *Quantum Fourier Transform* on an orthonormal basis $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ is defined to be a linear operator with the following action on the basis states

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle, \qquad j \in \{0,1,\ldots,N-1\} \qquad [3]$$

- Equivalently, the action on an arbitrary state may be written

$$QFT\left(\sum_{j=0}^{N-1} x_j |j\rangle\right) = \sum_{k=0}^{N-1} y_k |k\rangle$$

where the amplitudes y_k are the Discrete Fourier Transform (DFT) of the amplitudes x_j

QFT

$$\phi_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} a_j e^{2\pi ijk/N}, \qquad k \in \{0,1,\dots,N-1\}$$

PROOF

- Since

$$QFT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle$$

$$y_k = DFT(x_0,\dots,x_{N-1})$$

it follows:

$$QFT\sum_{j=0}^{N-1} x_j|j\rangle = \sum_{j=0}^{N-1} x_j QFT|j\rangle = \sum_{j=0}^{N-1} x_j \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle = \sum_{k=0}^{N-1} \underbrace{\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N}}_{y_k} |k\rangle = \sum_{k=0}^{N-1} y_k|k\rangle$$

where

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N} \qquad \square$$

Quantum Fourier Transform

- It is easy to prove that the *QFT* is unitary. Starting from the definition

$$QFT\left| j \right\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}\left| k \right\rangle$$

$$\left\langle h \right| QFT^{\dagger} QFT \left| j \right\rangle = \frac{1}{N}\sum_{l=0}^{N-1} e^{-2\pi ihl/N}\left\langle l \right| \sum_{k=0}^{N-1} e^{2\pi ijk/N}\left| k \right\rangle = \frac{1}{N}\sum_{k,l=0}^{N-1} e^{-2\pi ihl/N} e^{2\pi ijk/N}\left\langle l | k \right\rangle$$

$$= \frac{1}{N}\sum_{k,l=0}^{N-1} e^{-2\pi ihl/N} e^{2\pi ijk/N}\delta_{l,k} = \frac{1}{N}\sum_{k=0}^{N-1} e^{-2\pi ihk/N} e^{2\pi ijk/N} = \frac{1}{N}\sum_{k=0}^{N-1} e^{2\pi ik(h-j)/N}$$

Quantum Fourier Transform

- For $h = j \quad \rightarrow \quad \langle h | QFT^{\dagger} QFT | h \rangle = \frac{1}{N} \sum_{l=0}^{N-1} 1 = 1$

- For $h \neq j \quad \rightarrow \quad \langle h | QFT^{\dagger} QFT | j \rangle = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i k (h-j)/N} = \frac{1}{N} \frac{1 - e^{2\pi i (h-j)}}{1 - e^{2\pi i (h-j)/N}} = 0$

- The reason for the latter equality is that $k(h-j)$ is an integer and therefore

$$e^{2\pi i (h-j)} = 1$$

- As a conclusion

$$QFT^{\dagger} \times QFT = QFT \times QFT^{\dagger} = I$$

Quantum Fourier Transform

- There is a clever way to implement the *QFT* using single-qubit and two-qubit gates, and it only takes $O\left(\log^2(N)\right)$ of them, which is an exponential speedup in circuit complexity over the classical fast Fourier transform algorithms

Quantum Fourier Transform

- To construct this implementation, we express *j* as an *n*-bit binary number

$$j = j_{n-1} j_{n-2} \cdots j_1 j_0$$

$$= j_{n-1} 2^{n-1} + j_{n-2} 2^{n-2} + \cdots + j_1 2 + j_0$$

- Then, $j/N = j/2^n$ can be represented using a binary point, which is like a decimal point, but in base 2:

$$\frac{j}{N} = \frac{j}{2^n} = \frac{j_{n-1} 2^{n-1} + j_{n-2} 2^{n-2} + \cdots + j_1 2 + j_0}{2^n}$$

$$= \frac{j_{n-1}}{2^1} + \frac{j_{n-2}}{2^2} + \cdots + \frac{j_1}{2^{n-1}} + \frac{j_0}{2^n}$$

$$= 0.j_{n-1} j_{n-2} \cdots j_1 j_0 \quad \longrightarrow \quad \boxed{0.j_{n-1} j_{n-2} \cdots j_1 j_0 \equiv \frac{j_{n-1}}{2^1} + \frac{j_{n-2}}{2^2} + \cdots + \frac{j_1}{2^{n-1}} + \frac{j_0}{2^n}}$$

QFT

$$QFT\,|\,j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}\,|k\rangle, \qquad j \in \{0,1,\dots,N-1\}$$

- Similarly, we express *k* as an *n*-bit binary number

$$k = k_{n-1}k_{n-2}\dots k_1k_0$$
$$= k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_1 2 + k_0$$

- Using these, the exponential in Eq [3] is

$$e^{2\pi ijk/N} = e^{2\pi i(j/N)k}$$
$$= e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)(k_{n-1}2^{n-1}+k_{n-2}2^{n-2}+\cdots+k_1 2+k_0)}$$
$$= e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)(k_{n-1}2^{n-1})+2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)(k_{n-2}2^{n-2})+\cdots+2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)(k_1 2)+2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)(k_0)}$$
$$= e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)k_{n-1}2^{n-1}}e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)k_{n-2}2^{n-2}}\dots$$
$$\dots e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)k_1 2}e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1j_0)k_0} \qquad [4]$$

Quantum Fourier Transform

- Let's develop the first term….

$$e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1 j_0)k_{n-1}2^{n-1}} = e^{2\pi i\left(\frac{j_{n-1}}{2^1}+\frac{j_{n-2}}{2^2}+\cdots+\frac{j_1}{2^{n-1}}+\frac{j_0}{2^n}\right)2^{n-1}k_{n-1}}$$

$$= e^{2\pi i\left(j_{n-1}2^{n-2}+j_{n-2}2^{n-3}+\cdots+j_1+j_0/2\right)k_{n-1}}$$

$$\rightarrow \quad = \underbrace{e^{2\pi i j_{n-1}2^{n-2}k_{n-1}}}_{=1} \times \underbrace{e^{2\pi i j_{n-2}2^{n-3}k_{n-1}}}_{=1} \times\cdots\times \underbrace{e^{2\pi i j_1 2^0 k_{n-1}}}_{=1} \times e^{2\pi i(j_0/2)k_{n-1}}$$

$$= e^{2\pi i(0.j_0)k_{n-1}}$$

The exponentials are 1 because they are either $e^0 = 1$ or $e^{2\pi i m} = 1$ for some positive integer *m*

Quantum Fourier Transform

…and continue with the second one

$$e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1 j_0)k_{n-2}2^{n-2}} = e^{2\pi i\left(\frac{j_{n-1}}{2^1}+\frac{j_{n-2}}{2^2}+\cdots+\frac{j_1}{2^{n-1}}+\frac{j_0}{2^n}\right)k_{n-2}2^{n-2}}$$

$$= e^{2\pi i\left(j_{n-1}2^{n-3}+j_{n-2}2^{n-4}+\cdots+j_2+j_1/2+j_0/2^2\right)k_{n-2}}$$

$$= \underbrace{e^{2\pi i j_{n-1}2^{n-3}k_{n-2}}}_{=1} \times \underbrace{e^{2\pi i j_{n-2}2^{n-4}k_{n-2}}}_{=1} \times\cdots\times \underbrace{e^{2\pi i j_2 2^0 k_{n-2}}}_{=1} \times e^{2\pi i(j_1/2)k_{n-2}} \times e^{2\pi i(j_0/2^2)k_{n-2}}$$

$$= e^{2\pi i(0.j_1 j_0)k_{n-2}}$$

QFT

$$k = k_{n-1}k_{n-2}\ldots k_1k_0$$
$$= k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_1 2 + k_0$$

Using the same approach as for the other terms, the exponential [4] can be rewritten in the following manner

$$e^{2\pi ijk/N} = e^{2\pi i(0.j_0)k_{n-1}} e^{2\pi i(0.j_1j_0)k_{n-2}} \cdots e^{2\pi i(0.j_{n-2}\ldots j_1j_0)k_1} e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1j_0)k_0}$$

Plugging this into [3]

$$|j\rangle \xrightarrow{QFT} \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle$$
$$= \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi i(0.j_0)k_{n-1}} \times e^{2\pi i(0.j_1j_0)k_{n-2}} \times \cdots \times e^{2\pi i(0.j_{n-2}\ldots j_1j_0)k_1} \times e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1j_0)k_0}|k\rangle$$

Quantum Fourier Transform

Since we are summing over all *n*-bit binary numbers *k*, each bit $k_{n-1}, k_{n-2}, \ldots, k_1, k_0$ sums through 0 and 1, so this becomes

$$= \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} \cdots \sum_{k_0=0}^{1} e^{2\pi i(0.j_0)k_{n-1}} \times e^{2\pi i(0.j_1 j_0)k_{n-2}} \times \cdots \times e^{2\pi i(0.j_{n-2}\ldots j_1 j_0)k_1} \times e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1 j_0)k_0} \left|k\right\rangle$$

Since $\left|k\right\rangle \triangleq \left|k_{n-1}\ldots k_0\right\rangle$ is shorthand for $\left|k_{n-1}\right\rangle \otimes \ldots \otimes \left|k_0\right\rangle \triangleq \left|k_{n-1}\right\rangle \ldots \left|k_0\right\rangle$, we can move the terms to get

$$= \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} e^{2\pi i(0.j_0)k_{n-1}} \left|k_{n-1}\right\rangle \sum_{k_{n-2}=0}^{1} e^{2\pi i(0.j_1 j_0)k_{n-2}} \left|k_{n-2}\right\rangle \cdots \sum_{k_1=0}^{1} {}^{2\pi i(0.j_{n-2}\ldots j_1 j_0)k_1} \left|k_1\right\rangle \sum_{k_0=0}^{1} e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1 j_0)k_0} \left|k_0\right\rangle$$

Quantum Fourier Transform

- Let's comment on the previous expression before moving on

$$\left| j_{n-1} j_{n-2} \dots j_1 j_0 \right\rangle \xrightarrow{QFT} = \frac{1}{\sqrt{N}} \sum_{k_{n-1}=0}^{1} e^{2\pi i (0.j_0) k_{n-1}} \left| k_{n-1} \right\rangle \sum_{k_{n-2}=0}^{1} e^{2\pi i (0.j_1 j_0) k_{n-2}} \left| k_{n-2} \right\rangle \cdots$$

$$\cdots \sum_{k_1=0}^{1} {}^{2\pi i (0.j_{n-2} \dots j_1 j_0) k_1} \left| k_1 \right\rangle \sum_{k_0=0}^{1} e^{2\pi i (0.j_{n-1} j_{n-2} \dots j_1 j_0) k_0} \left| k_0 \right\rangle$$

- The following relationships connect *k* and *j* in all exponents

$\rightarrow$

$$0.j_0 \leftrightarrow k_{n-1}$$
$$0.j_1 j_0 \leftrightarrow k_{n-2}$$
$$0.j_2 j_1 j_0 \leftrightarrow k_{n-3}$$
$$\vdots$$
$$0.j_{n-1} j_2 j_1 j_0 \leftrightarrow k_0$$

Quantum Fourier Transform

Now we develop the summations taking into account that $e^0 = 1$

$$= \frac{1}{\sqrt{N}}\left(|0\rangle + e^{2\pi i(0.j_0)}|1\rangle\right)\left(|0\rangle + e^{2\pi i(0.j_1 j_0)}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i(0.j_{n-2}\dots j_1 j_0)}|1\rangle\right)\left(|0\rangle + e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1 j_0)}|1\rangle\right)$$

Finally, since $\sqrt{N} = \sqrt{2^n} = \left(\sqrt{2}\right)^n$, we get the product state

$$= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_0)}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_1 j_0)}|1\rangle\right)\cdots$$

$$\times\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-2}\dots j_1 j_0)}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-1}j_{n-2}\dots j_1 j_0)}|1\rangle\right)$$

$$QFT\left|j\right\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}\left|k\right\rangle, \qquad j \in \{0,1,\dots,N-1\}$$

$$0.j_{n-1}j_{n-2}\cdots j_1 j_0 \equiv \frac{j_{n-1}}{2^1} + \frac{j_{n-2}}{2^2} + \cdots + \frac{j_1}{2^{n-1}} + \frac{j_0}{2^n}$$

- As a conclusion, equation [3] can be transformed as follows

$$\left|j_{n-1}j_{n-2}\cdots j_1 j_0\right\rangle \xrightarrow{QFT} \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}\left|k\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_0)}\left|1\right\rangle\right)\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_1 j_0)}\left|1\right\rangle\right)\cdots$$

$$\downarrow$$

$$\boxed{j_{n-1}2^{n-1} + j_{n-2}2^{n-2} + \cdots + j_1 2 + j_0}$$

$$\times\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_{n-2}\cdots j_1 j_0)}\left|1\right\rangle\right)\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_{n-1}j_{n-2}\cdots j_1 j_0)}\left|1\right\rangle\right) \quad [5]$$

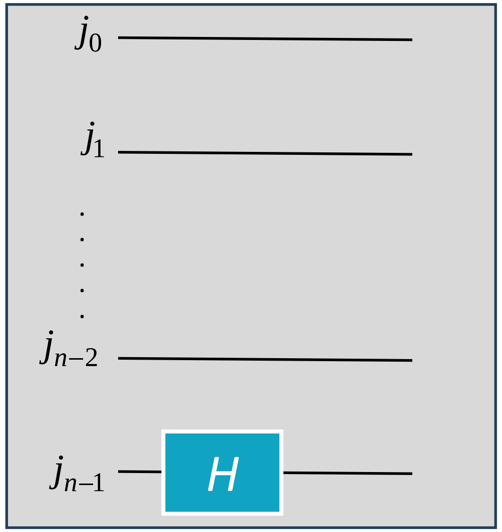
which is another way of stating the definition of the *QFT*, but in binary

- If we can create a quantum circuit that converts $\left|j\right\rangle = \left|j_{n-1}\cdots j_0\right\rangle$ to Eq. [5] we will have a quantum circuit for the *QFT*

Quantum Fourier Transform

$|j\rangle = |j_{n-1} \cdots j_0\rangle$

- Let us now prove that we can create a circuit for the *QFT* using **Hadamard gates** and **controlled rotations**
- Consider the rightmost term of Eq. [5], i.e., $\left(|0\rangle + e^{2\pi i(0.j_{n-1}j_{n-2}\cdots j_1 j_0)}|1\rangle\right)/\sqrt{2}$
- To begin considering it, we apply the Hadamard gate to a qubit $|j_{n-1}\rangle$



$$H|j_{n-1}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{j_{n-1}}|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + \left(e^{i\pi}\right)^{j_{n-1}}|1\rangle\right)$$

$$= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\, j_{n-1}/2}|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-1})}|1\rangle\right)$$

Quantum Fourier Transform

- Let us now define a general 1-qubit *phase rotation gate* represented by the following matrix

$$R_h = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^h} \end{bmatrix}$$

- It acts on basis states

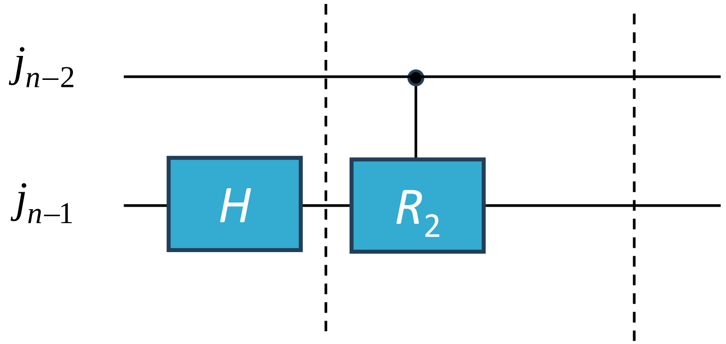
$$R_h|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^h} \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

$$R_h|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^h} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ e^{2\pi i/2^h} \end{bmatrix} = e^{2\pi i/2^h}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = e^{2\pi i/2^h}|1\rangle$$

$$\rightarrow \quad \begin{cases} R_h|0\rangle = |0\rangle \\ R_h|1\rangle = e^{2\pi i/2^h}|1\rangle \end{cases}$$

Quantum Fourier Transform

- After the previous Hadamard matrix, we apply R_2 to qubit *n* – 1, controlled by qubit *n* – 2
- That is, for the state of qubit *n* – 1, the amplitude of $|j_{n-1}\rangle$ is multiplied by $e^{2\pi i/2^2}$ if $j_{n-2} = 1$, and nothing happens otherwise
- That is, the state of qubit $|j_{n-1}\rangle$ becomes



$$|\psi\rangle = |j_{n-2}\rangle \frac{|0\rangle + e^{2\pi i(0.j_{n-1})}|1\rangle}{\sqrt{2}} \qquad cR_2|\psi\rangle$$

Quantum Fourier Transform

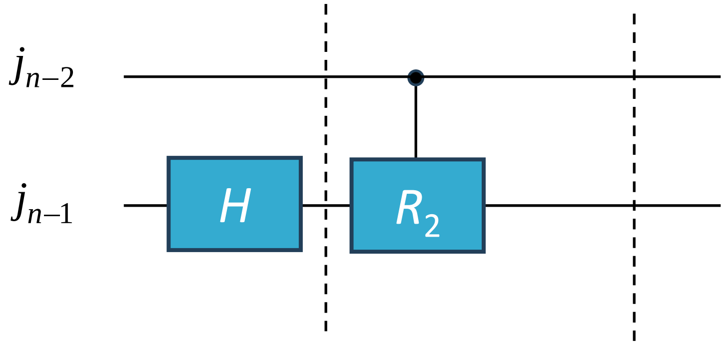
- Thus

$$cR_2\left|\psi\right\rangle = cR_2\left(\left|j_{n-2}\right\rangle\frac{\left|0\right\rangle + e^{2\pi i\left(0.j_{n-1}\right)}\left|1\right\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}cR_2\left(\left|j_{n-2}\right\rangle\left|0\right\rangle\right) + \frac{e^{2\pi i\left(0.j_{n-1}\right)}}{\sqrt{2}}cR_2\left(\left|j_{n-2}\right\rangle\left|1\right\rangle\right)$$

- By keeping into consideration that

$$cR_2\left(\left|j_{n-2}\right\rangle\left|0\right\rangle\right) = \left|j_{n-2}\right\rangle\left|0\right\rangle, \qquad \forall j_{n-2} \in \{0,1\}$$

$$cR_2\left(\left|j_{n-2}\right\rangle\left|1\right\rangle\right) = \left(e^{2\pi i/2^2}\right)^{j_{n-2}}\left|j_{n-2}\right\rangle\left|1\right\rangle$$



$$\left|\psi\right\rangle = \left|j_{n-2}\right\rangle\frac{\left|0\right\rangle + e^{2\pi i\left(0.j_{n-1}\right)}\left|1\right\rangle}{\sqrt{2}} \qquad cR_2\left|\psi\right\rangle$$

Quantum Fourier Transform

- We can continue our development of $cR_2|\psi\rangle$

$$cR_2|\psi\rangle = \frac{1}{\sqrt{2}}\left[|j_{n-2}\rangle|0\rangle + e^{2\pi i(0.j_{n-1})}\left(e^{2\pi i/2^2}\right)^{j_{n-2}}|j_{n-2}\rangle|1\rangle\right]$$

$$=|j_{n-2}\rangle\left[\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-1})}\left(e^{2\pi i/2^2}\right)^{j_{n-2}}|1\rangle\right)\right]$$

$$\left(e^{2\pi i/2^2}\right)^{j_{n-2}} = e^{2\pi i\left(j_{n-2}/2^2\right)} = e^{2\pi i 0.0 j_{n-2}}$$

$$=|j_{n-2}\rangle\left[\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-1})}e^{2\pi i(0.0 j_{n-2})}|1\rangle\right)\right]$$

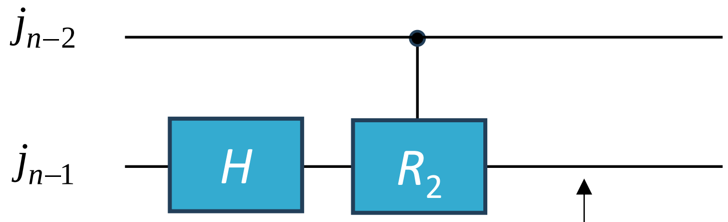
Quantum Fourier Transform

- Still continuing with our development of $cR_2|\psi\rangle$

$$cR_2|\psi\rangle = |j_{n-2}\rangle\left[\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-1})}e^{2\pi i(0.0\,j_{n-2})}|1\rangle\right)\right] = |j_{n-2}\rangle\left[\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_{n-1}}{2^1}\right)}e^{2\pi i\left(\frac{j_{n-2}}{2^2}\right)}|1\rangle\right)\right]$$

$$= |j_{n-2}\rangle\left[\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_{n-1}}{2^1}\right)+2\pi i\left(\frac{j_{n-2}}{2^2}\right)}|1\rangle\right)\right] = |j_{n-2}\rangle\left[\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{j_{n-1}}{2^1}+\frac{j_{n-2}}{2^2}\right)}|1\rangle\right)\right]$$

$$= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-1}j_{n-2})}|1\rangle\right)$$



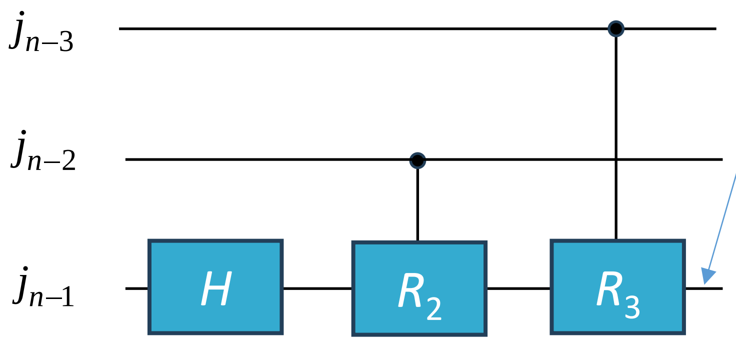
- Thus, the state of qubit *n*-1 would be

$$cR_2|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i(0.j_{n-1}j_{n-2})}|1\rangle\right)$$

Quantum Fourier Transform

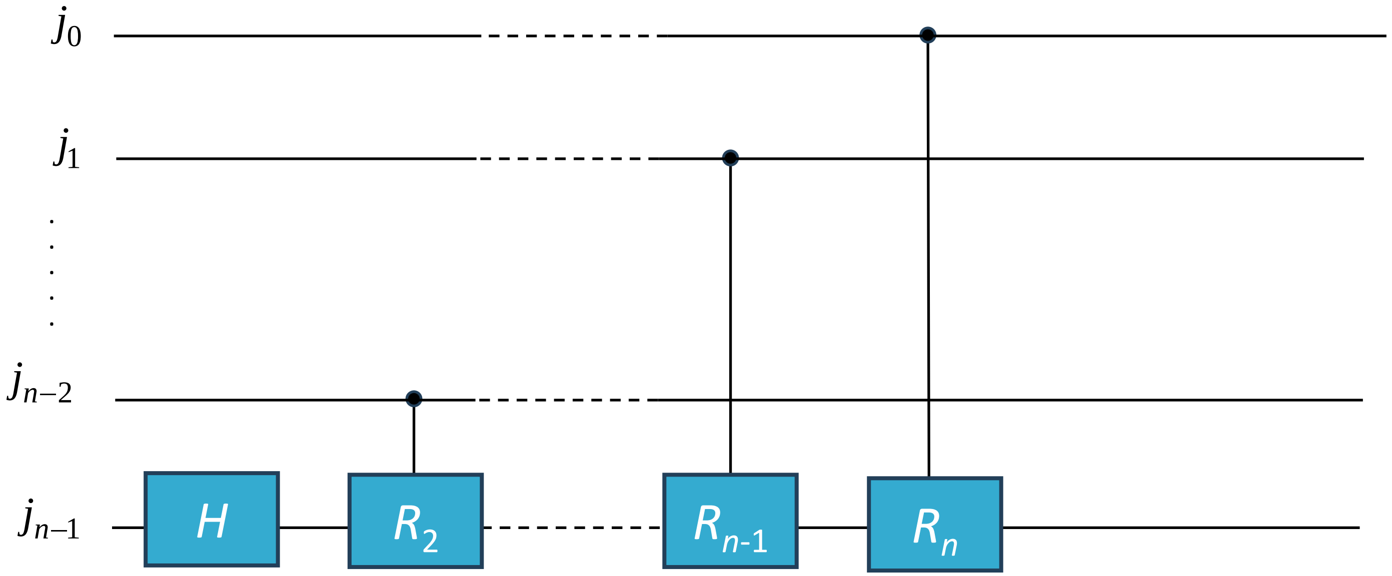
- Similarly, we can apply R_3 to *n* – 1, controlled by qubit *n* – 3
- Then, the state of qubit *n*−1 would be

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(0.j_{n-1}j_{n-2}j_{n-3}\right)}|1\rangle\right)$$



Quantum Fourier Transform

- Continuing this through R_n, controlled by qubit 0, the state of qubit *n* − 1 is

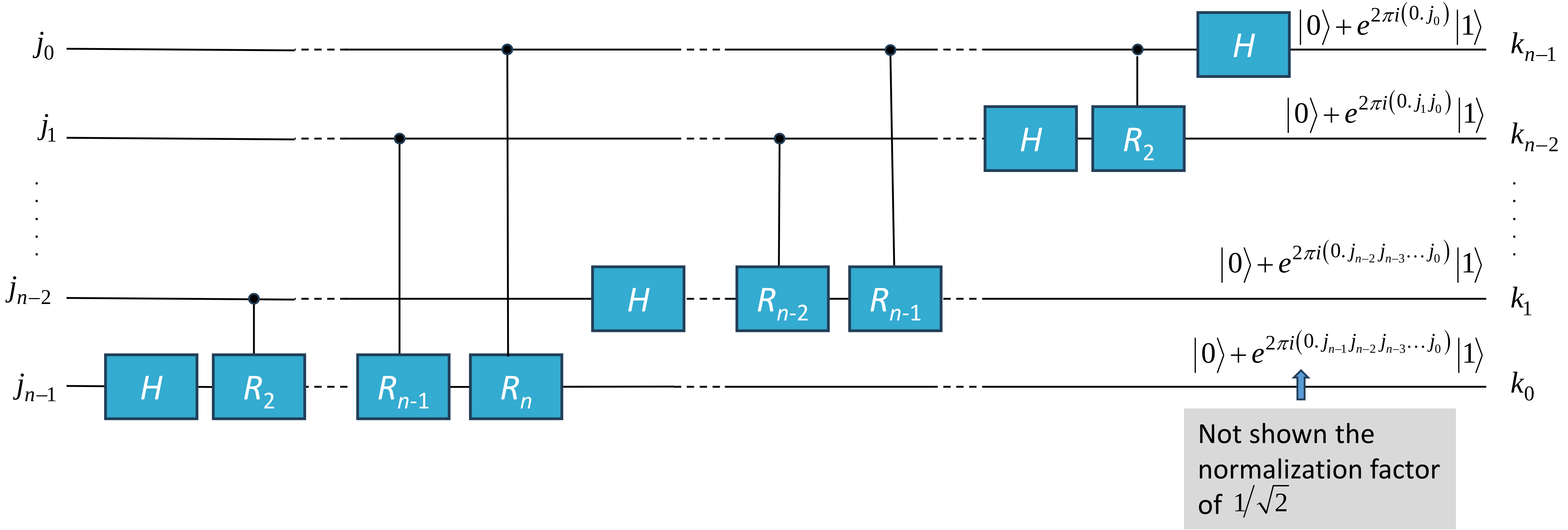


$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(0.j_{n-1}j_{n-2}j_{n-3}\cdots j_0\right)}|1\rangle\right)$$

- This is the rightmost factor of Eq. [5]

Quantum Fourier Transform

- Similarly, we can apply Hadamard and cR_h gates to the other qubits to construct the other factors, resulting in the following quantum circuit



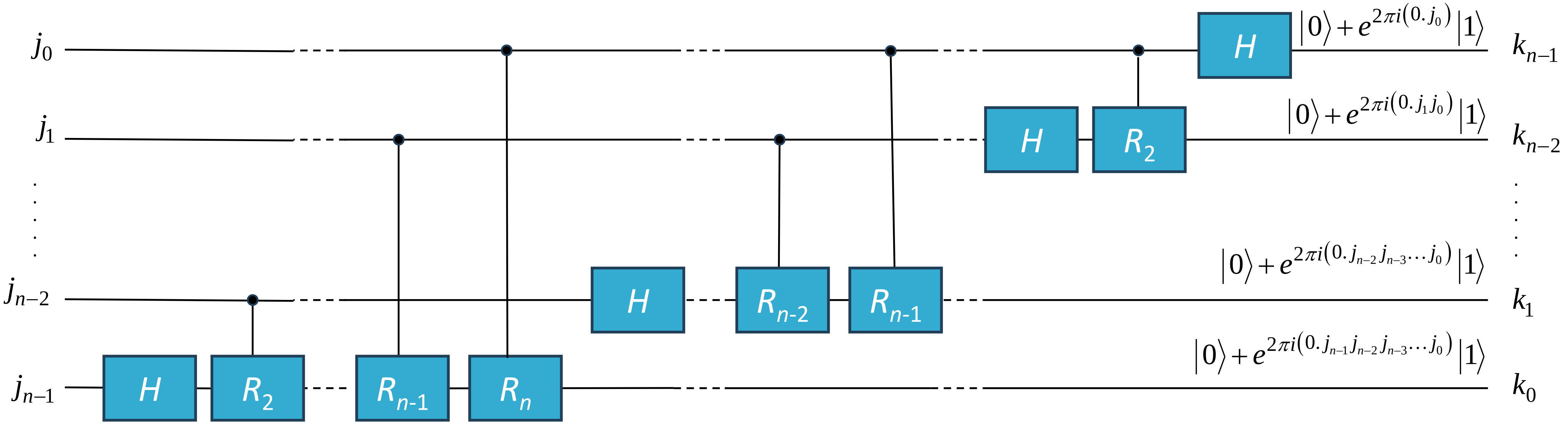
Quantum Fourier Transform

$j = j_{n-1} j_{n-2} \cdots j_1 j_0$

The order of the outputs produced by [5], derived by the *QFT* definition, and by the quantum circuit is reversed

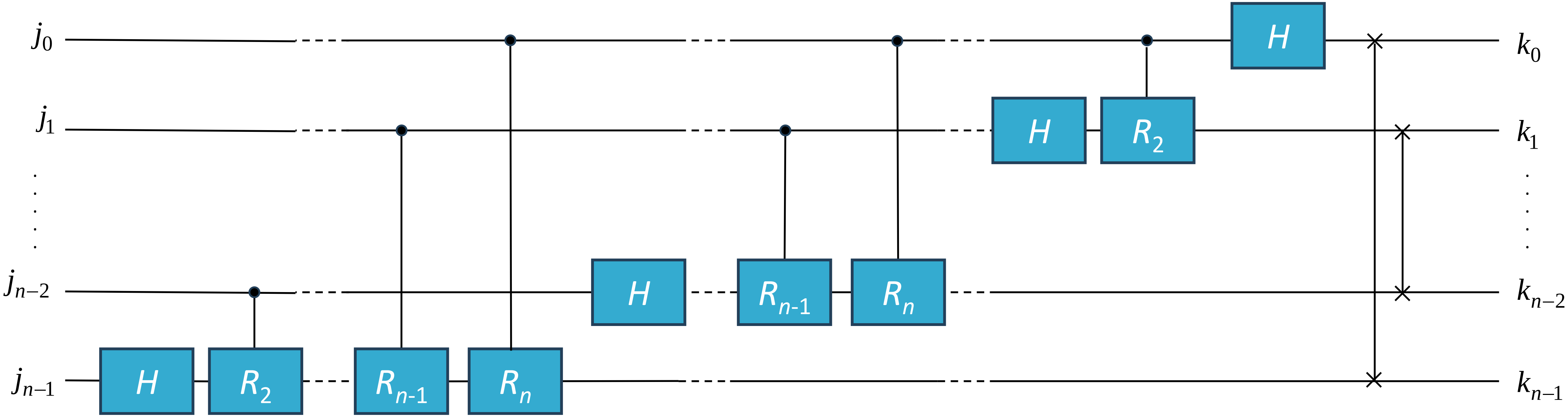
$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i jk/N} |k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (0.j_0)}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (0.j_1 j_0)}|1\rangle\right)\cdots$$

$$\cdots \times \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (0.j_{n-2}\cdots j_1 j_0)}|1\rangle\right)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (0.j_{n-1} j_{n-2}\cdots j_1 j_0)}|1\rangle\right) \quad [5]$$



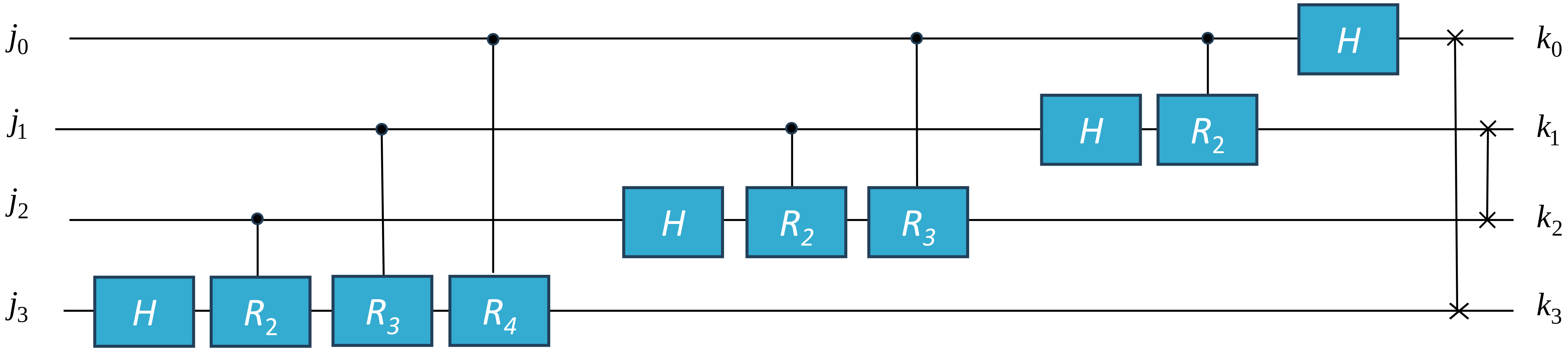
Quantum Fourier Transform

- Therefore, we need to reverse the order, such as by using *SWAP* gates



Quantum Fourier Transform

- For example, with n=4 qubits



Quantum Fourier Transform

- Let us add up the total number of gates in the *QFT* circuit with n qubits, beginning with the Hadamard and controlled-R_h gates

- The bottom row of the circuit uses $\boldsymbol{n}$ gates, the row above it uses $\boldsymbol{n-1}$ gates, and so fourth, until we get to one gate at the top row

- So, the total number of Hadamard and controlled-R_h gates

$$n+(n-1)+(n-2)+\cdots+3+2+1=\frac{n(n+1)}{2}$$

Quantum Fourier Transform

- There are also ***n/2*** swap gates to reverse the order of the outputs

- Altogether, the total number of single-qubit and two-qubit gates is

$$\frac{n(n+1)}{2} + \frac{n}{2} = O(n^2) = O(\log^2 N) \qquad \leftarrow \left(N = 2^n \to n = \log N \right)$$

- This runtime of $O(\log^2 N)$ is an exponential speedup over the classical fast Fourier transform algorithms, which run in $O(N \log N)$ time
- At first glance, this sounds fantastic, as the Fourier transform is a crucial step in many real-world data processing applications

Quantum Fourier Transform

- However, the problem with the *QFT* is that the amplitudes in a quantum computer cannot be directly accessed by measurement
- Thus, there is no way of determining the Fourier transformed amplitudes of the original state

Quantum Fourier Transform

- Worse still, there is in general no way to efficiently prepare the original state to be Fourier-transformed

- Thus, finding uses for the quantum Fourier transform is more subtle than we might have hoped

- In the following we show **two algorithms** based upon a more subtle application of the quantum Fourier transform: **phase estimation** and **factoring**

Inverse Quantum Fourier Transform

- The inverse quantum Fourier transform *(IQFT)* does undo the *QFT*
- Since the *QFT* performs the mapping in Eq. [3]

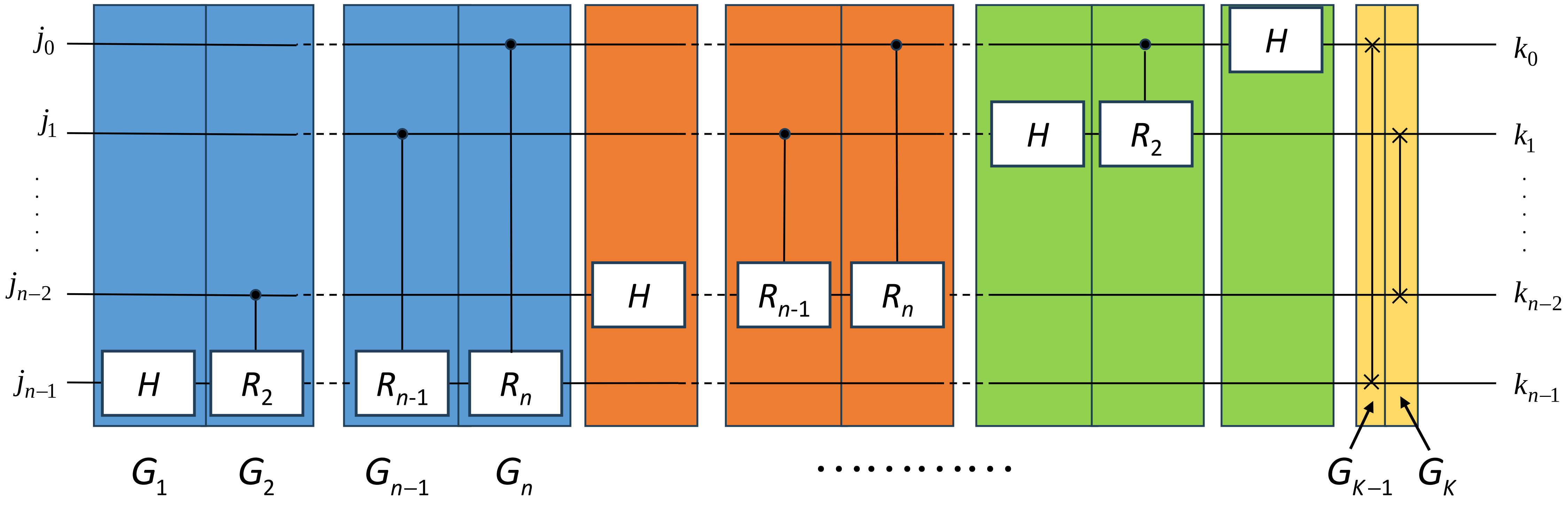
$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle$$

the *IQFT* does the reverse:

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle \rightarrow |j\rangle$$

Inverse Quantum Fourier Transform

- As we showed earlier the *QFT* quantum circuit encompasses a number of unitary gates equal to $K = \frac{n(n+1)}{2} + \frac{n}{2}$



Inverse Quantum Fourier Transform

- Therefore, the Quantum Fourier Transform (QFT) circuit can be succinctly described by the product of *K* unitary operators

$$QFT = G_K G_{K-1} \cdots G_1$$

where G_j, $\forall j \in \{1,2,....,K\}$ are such that $G_j^\dagger \times G_j = G_j \times G_j^\dagger = I$

- Since quantum gates are unitary, the inverses are their conjugate transposes

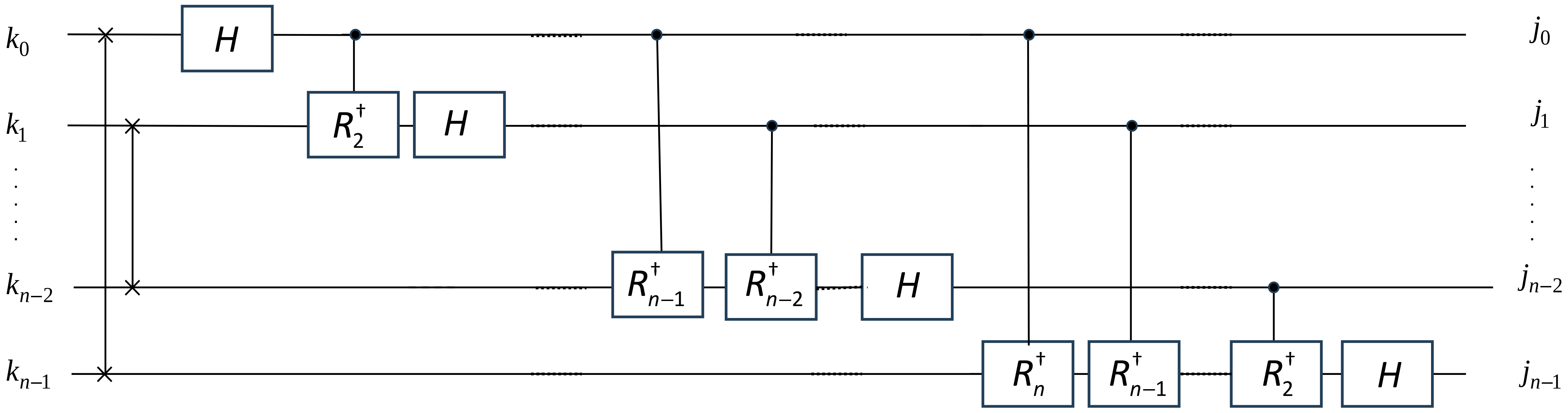
$$G_j^{-1} = G_j^\dagger, \ \forall j \in \{1,2,....,K\}$$

- Thus, the *Inverse QFT* or *IQFT* is given by

$$IQFT = QFT^{-1} = QFT^\dagger = \left(G_K G_{K-1} \cdots G_1\right)^\dagger = G_1^\dagger \times G_2^\dagger \times \cdots \times G_K^\dagger = G_1^{-1} \times G_2^{-1} \times \cdots \times G_K^{-1}$$

Inverse Quantum Fourier Transform

- As a quantum circuit, the *IQFT* can be performed by reversing the order of the gates of the *QFT* and replacing them with their inverses or, equivalently, with their conjugate transposes



Inverse Quantum Fourier Transform

- Note $SWAP^{\dagger} = SWAP$, $H^{\dagger} = H$, and $R_h^{\dagger}$ is a rotation about the *z*-axis of the Bloch sphere by $-2\pi/2^h$ radians

- The *IQFT* has the same gate complexity as the *QFT*, which is $O(n^2)$

- Please note that

$$R_h = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^h} \end{bmatrix} \quad \rightarrow \quad R_h^{\dagger} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-2\pi i/2^h} \end{bmatrix}$$

Phase/Eigenvalue Estimation

Phase/Eigenvalue Estimation

- In the following we will focus on a specific problem:

  Given a unitary matrix U and one of its eigenvectors $|v\rangle$, *find or estimate its eigenvalue.*

- We have previously shown that the eigenvalues of a **unitary matrix** must have the form $e^{i\theta}$ for some **real number** $\boldsymbol{\theta}$
- For this reason, this problem is called **phase estimation**, since finding the eigenvalue is equivalent to finding the phase θ

Classical Solution

- We know that multiplying $|v\rangle$ by *U* will result in $|v\rangle$ multiplied by $e^{i\theta}$, i.e.,

$$U|v\rangle = e^{i\theta}|v\rangle$$

- If $|v\rangle$ is an *N*-dimensional vector and *U* is an *N* × *N* matrix, we can write out this equation as

$$\begin{bmatrix} U_{11} & U_{12} & \cdots & U_{1N} \\ U_{21} & U_{22} & \cdots & U_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ U_{N1} & U_{N2} & \cdots & U_{NN} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = e^{i\theta} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N \\ U_{21}v_1 + U_{22}v_2 + \cdots + U_{2N}v_N \\ \vdots \\ U_{N1}v_1 + U_{N2}v_2 + \cdots + U_{NN}v_N \end{bmatrix} = e^{i\theta} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix}$$

Classical Solution

- We can use any row to find $e^{i\theta}$
- For example, using the first row

$$U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N = e^{i\theta}v_1$$

- Thus, the eigenvalue is

$$e^{i\theta} = \frac{U_{11}v_1 + U_{12}v_2 + \cdots + U_{1N}v_N}{v_1}$$

- This takes N multiplications, $N-1$ additions, and one division, for a total of $2N = O(N)$ elementary arithmetic operations

Quantum Solution

- Say the unitary matrix U is an n-qubit quantum gate, so U is an $N \times N$ matrix, where $N = 2^n$
- We assume that we have ***n*** qubits whose state is the **eigenstate** $|v\rangle$:

$$\underset{n\ qubits}{|v\rangle}$$

- To estimate the phase of its corresponding eigenvalue to ***m*** bits of precision, we also have ***m*** additional qubits, all initially in the $|0\rangle$ state:

$$\underbrace{|0\cdots000\rangle}_{m\ qubits}\ \underset{n\ qubits}{|v\rangle}$$

- So, the total number of qubits in our circuit is *m+n*

Quantum Solution

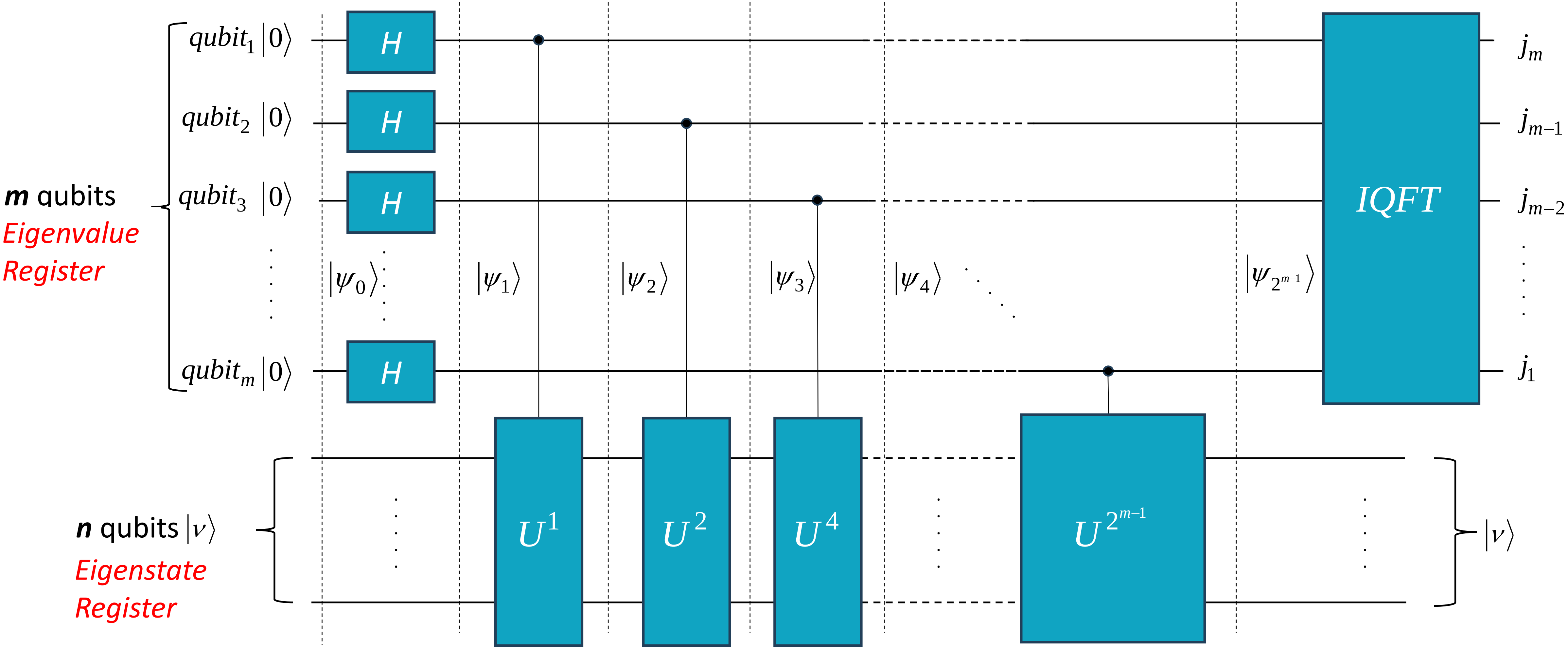
- Let us refer to these groupings as the *"eigenvalue register"* and the *"eigenstate register"* since the *m* qubits will eventually contain an *m* bit approximation of the phase of the eigenvalue, and the *n* qubits are in the eigenstate $|v\rangle$

$$\underbrace{|0\cdots000\rangle}_{m\ qubits} \quad \underbrace{|v\rangle}_{n\ qubits}$$

Eigenvalue Register *Eigenstate Register*

- To estimate the phase of the eigenvalue, we apply the following quantum circuit

Quantum Solution



Quantum Solution

- Before delving into any details, let’s note that

$$\left|\psi_0\right\rangle = \left|00\cdots 0\right\rangle \left|\nu\right\rangle$$

where the rightmost qubit is less significant while the leftmost qubit is the most significant

Quantum Solution

- Let us go through each step of this circuit to see how it works
- First, we apply the Hadamard gate to each qubit of the *eigenvalue register,* and we get

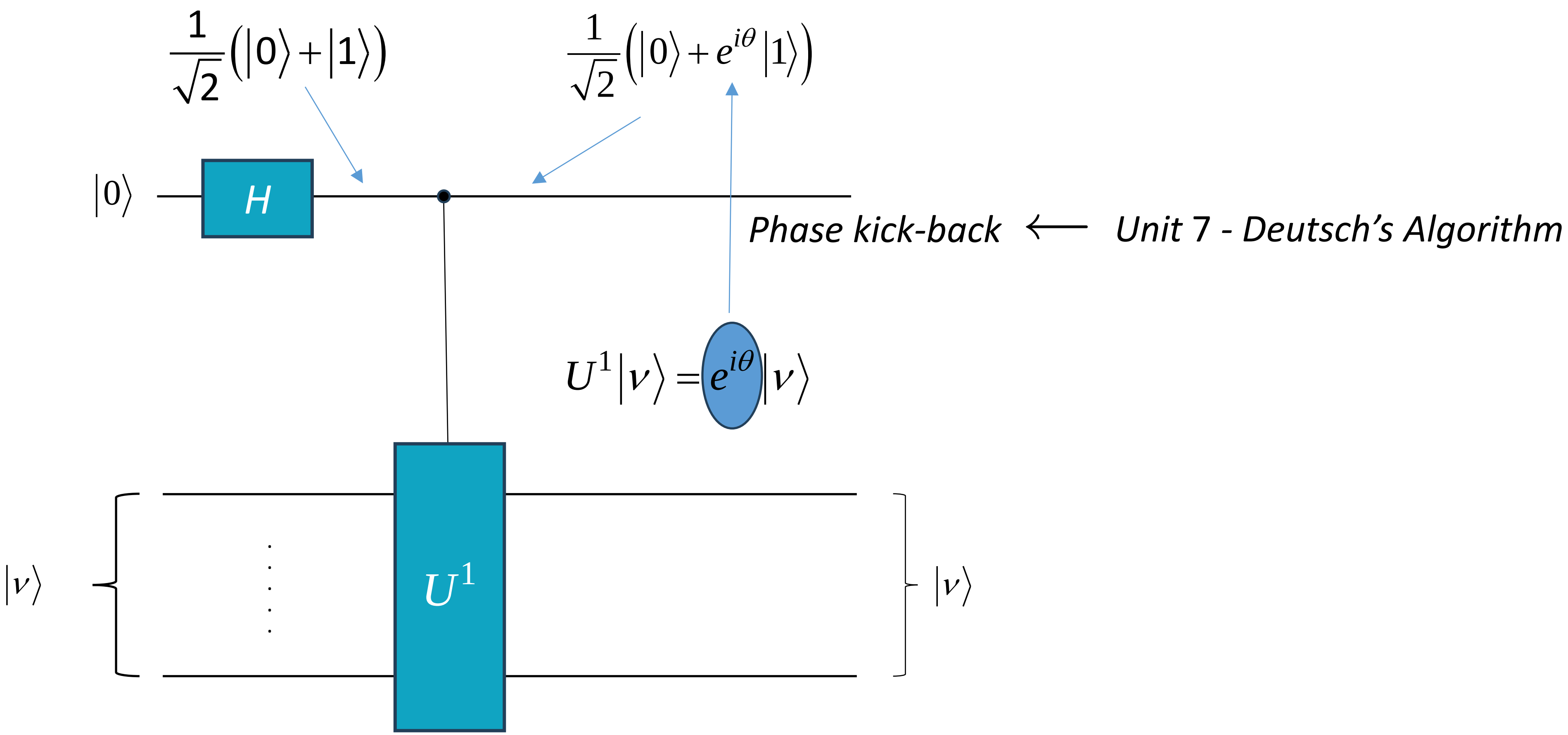
$$|\psi_1\rangle = |+ + \cdots +\rangle |v\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes |v\rangle$$

$qubit_m$ $qubit_1$

$$= \frac{1}{\sqrt{2^m}}\left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + |1\rangle\right) \otimes |v\rangle$$

$qubit_m$ $qubit_1$

Quantum Solution



Quantum Solution

- Next, we apply a controlled-*U* gate, where the rightmost qubit (j_1) of the eigenvalue register is the control, and the eigenstate register is the target
- Since $U|v\rangle = e^{i\theta}|v\rangle$, this causes the state to acquire a phase of $e^{i\theta}$ when the control qubit is $|1\rangle$

control *target*

$$|\psi_2\rangle = cU|\psi_1\rangle = \frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)\otimes\cdots\otimes(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes cU\left(\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\otimes|v\rangle\right)$$

$$= \frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)\otimes\cdots\otimes(|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes\left(cU|0\rangle|v\rangle + cU|1\rangle|v\rangle\right)$$

- Since

$$cU|0\rangle|v\rangle = |0\rangle|v\rangle$$

$$cU|1\rangle|v\rangle = |1\rangle U|v\rangle = e^{i\theta}|1\rangle|v\rangle$$

Quantum Solution

- … it follows

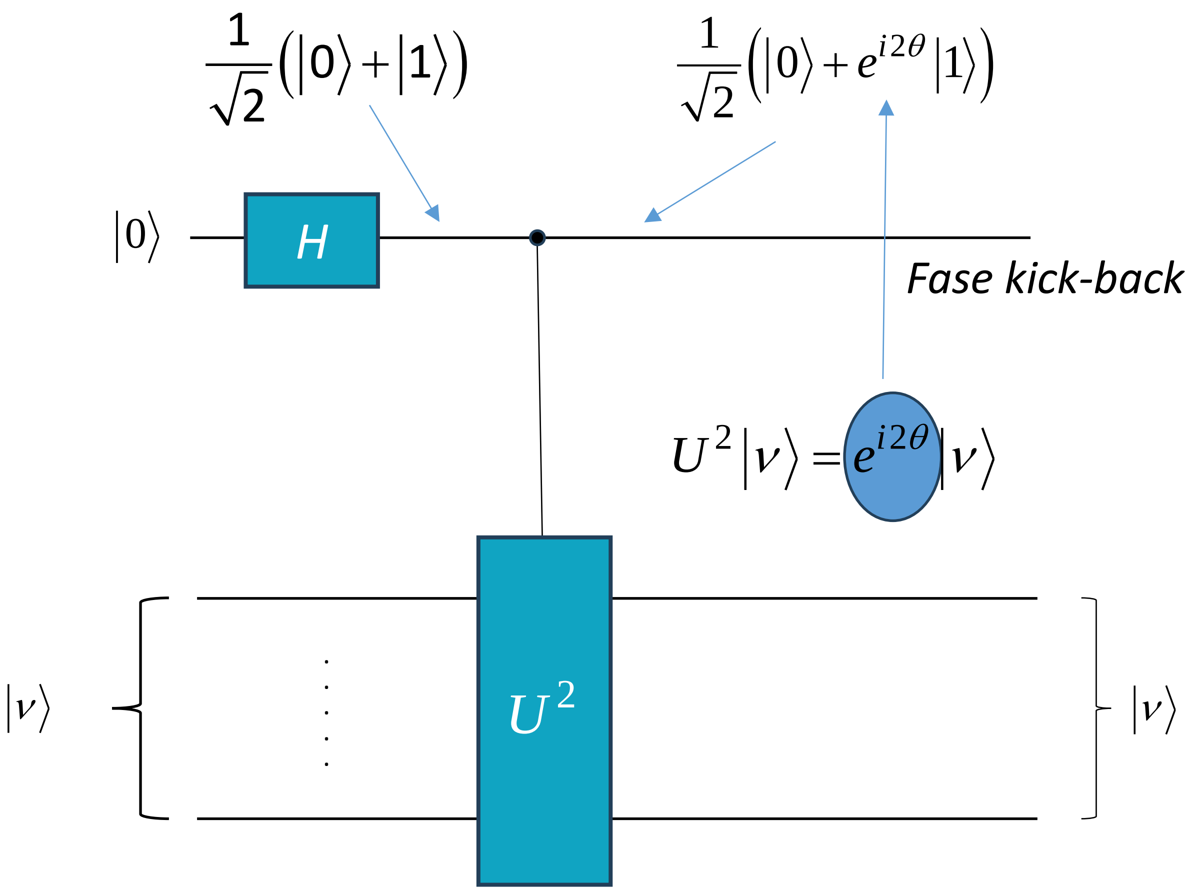
$$= \frac{1}{\sqrt{2^m}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle|v\rangle + e^{i\theta}|1\rangle|v\rangle\right)$$

$$= \frac{1}{\sqrt{2^m}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle + e^{i\theta}|1\rangle\right)|v\rangle$$

$$\rightarrow$$

$$|\psi_2\rangle = cU|\psi_1\rangle = \frac{1}{\sqrt{2^m}}\left(|0\rangle + |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle + e^{i\theta}|1\rangle\right)|v\rangle$$

Quantum Solution



Quantum Solution

- Then, we apply the controlled-U^2 gate, which causes the second-to-rightmost qubit (j_2) of the eigenvalue register to acquire a phase of $e^{2i\theta}$, when the control qubit is $|1\rangle$

$$cU^2|0\rangle|v\rangle = |0\rangle|v\rangle$$

$$cU^2(|1\rangle|v\rangle) = cU \times (cU|1\rangle|v\rangle) = cU|1\rangle(e^{i\theta}|v\rangle) = e^{i\theta}cU|1\rangle|v\rangle = e^{i\theta}(e^{i\theta}|1\rangle|v\rangle) = e^{2i\theta}|1\rangle|v\rangle$$

- Thus

$$|\psi_3\rangle = cU^2|\psi_2\rangle = \frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)\otimes\cdots\otimes(|0\rangle+|1\rangle)\otimes cU^2(|0\rangle+|1\rangle)\otimes(|0\rangle+e^{i\theta}|1\rangle)|v\rangle$$

$$= \frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)\otimes\cdots\otimes(|0\rangle+|1\rangle)\otimes(|0\rangle+e^{2i\theta}|1\rangle)\otimes(|0\rangle+e^{i\theta}|1\rangle)|v\rangle \quad \rightarrow$$

$$|\psi_3\rangle = \frac{1}{\sqrt{2^m}}(|0\rangle+|1\rangle)\otimes\cdots\otimes(|0\rangle+|1\rangle)\otimes(|0\rangle+e^{2i\theta}|1\rangle)\otimes(|0\rangle+e^{i\theta}|1\rangle)|v\rangle$$

Quantum Solution

- Then, we apply the controlled-U^4 gate, which applies a phase of $e^{4i\theta}$

$$|\psi_4\rangle = \frac{1}{\sqrt{2^m}}\left(|0\rangle + |1\rangle\right)\otimes\cdots\otimes\left(|0\rangle + e^{4i\theta}|1\rangle\right)\otimes\left(|0\rangle + e^{2i\theta}|1\rangle\right)\otimes\left(|0\rangle + e^{i\theta}|1\rangle\right)|v\rangle$$

- Continuing with the controlled gates, we eventually apply $\text{controlled}-U^{2^{m-1}}$ where the phase is $e^{2^{m-1}i\theta}$

$$|\psi_{2^{m-1}}\rangle = \frac{1}{\sqrt{2^m}}\left(|0\rangle + e^{2^{m-1}i\theta}|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle + e^{4i\theta}|1\rangle\right)\otimes\left(|0\rangle + e^{2i\theta}|1\rangle\right)\otimes\left(|0\rangle + e^{i\theta}|1\rangle\right)|v\rangle$$

Quantum Solution

- Now, let us change the variables using $\theta = 2\pi j$, so if we can find *j*, we simply multiply it by 2π to find θ
- Substituting, the previous state becomes

$$\left|\psi_{2^{m-1}}\right\rangle = \frac{1}{\sqrt{2^m}}\left(\left|0\right\rangle + e^{2\pi i 2^{m-1} j}\left|1\right\rangle\right)\otimes\cdots\otimes\left(\left|0\right\rangle + e^{2\pi i 2^{2} j}\left|1\right\rangle\right)\otimes\left(\left|0\right\rangle + e^{2\pi i 2 j}\left|1\right\rangle\right)\otimes\left(\left|0\right\rangle + e^{2\pi i 2^{0} j}\left|1\right\rangle\right)\left|\nu\right\rangle \quad [1]$$

- Since

$0 \le \theta < 2\pi$,

it turns out that $0 \le j = \dfrac{\theta}{2\pi} < 1$

Quantum Solution

- We can express *j* as an *m*-bit binary number

$$j = 0.j_1 j_2 \dots j_{m-1} j_m = \frac{j_1}{2} + \frac{j_2}{2^2} + \cdots + \frac{j_m}{2^m} < \sum_{n=1}^{\infty} \frac{1}{2^n} = 1$$

where $j_1 j_2 \dots j_{m-1} j_m \big|_2 \equiv j_1 2^{m-1} + j_2 2^{m-2} + \cdots + j_{m-1} 2 + j_m \big|_{10}$

- We can now prove that $j = 0.j_1 j_2 \dots j_{m-1} j_m$, has the following properties

$$2^{m-1}(0.j_1 j_2 \dots j_{m-1} j_m) = j_1 j_2 \dots j_{m-1}.j_m$$

$$\vdots$$

$$2^2(0.j_1 j_2 j_3 \dots j_{m-1} j_m) = j_1 j_2.j_3 \dots j_{m-1} j_m$$

$$2(0.j_1 j_2 j_3 \dots j_{m-1} j_m) = j_1.j_2 j_3 \dots j_{m-1} j_m$$

[2]

Quantum Solution

- Let's start showing the first property

$$2^{m-1}(0.j_1 j_2 \dots j_{m-1} j_m) = 2^{m-1}\left(\frac{j_1}{2} + \frac{j_2}{2^2} + \cdots + \frac{j_m}{2^m}\right) = \frac{2^{m-1} j_1}{2} + \frac{2^{m-1} j_2}{2^2} + \cdots + \frac{2^{m-1} j_m}{2^m}$$

$$= 2^{m-2} j_1 + 2^{m-3} j_2 + \cdots + j_{m-1} + \tfrac{1}{2} j_m = j_1 j_2 \cdots j_{m-2} j_{m-1}.j_m \qquad \rightarrow$$

$$2^{m-1}\left(0.j_1 j_2 \cdots j_{m-1} j_m\right) = j_1 j_2 \cdots j_{m-1}.j_m \qquad \square$$

Quantum Solution

- By substituting *j* in [1], we obtain

$$\left|\psi_{2^{m-1}}\right\rangle = \frac{1}{\sqrt{2^m}}\left(\left|0\right\rangle + e^{2\pi i 2^{m-1}(0.j_1 j_2 \dots j_{m-1} j_m)}\left|1\right\rangle\right) \otimes \cdots \otimes \left(\left|0\right\rangle + e^{2\pi i 2^{2}(0.j_1 j_2 \dots j_{m-1} j_m)}\left|1\right\rangle\right)$$

$$\otimes\left(\left|0\right\rangle + e^{2\pi i 2(0.j_1 j_2 \dots j_{m-1} j_m)}\left|1\right\rangle\right) \otimes \left(\left|0\right\rangle + e^{2\pi i 2^{0}(0.j_1 j_2 \dots j_{m-1} j_m)}\left|1\right\rangle\right)\left|\nu\right\rangle$$

- By exploiting [2] reported below

$$2^{m-1}(0.j_1 j_2 \dots j_{m-1} j_m) = j_1 j_2 \dots j_{m-1}.j_m$$

$$\vdots$$

$$2^{2}(0.j_1 j_2 j_3 \dots j_{m-1} j_m) = j_1 j_2 . j_3 \dots j_{m-1} j_m$$

$$2(0.j_1 j_2 j_3 \dots j_{m-1} j_m) = j_1 . j_2 j_3 \dots j_{m-1} j_m$$

Quantum Solution

- ..it follows

$$\left|\psi_{2^{m-1}}\right\rangle = \frac{1}{\sqrt{2^m}}\left(\left|0\right\rangle + e^{2\pi i\left(j_1 j_2 \ldots j_{m-1} . j_m\right)}\left|1\right\rangle\right) \otimes \cdots \otimes \left(\left|0\right\rangle + e^{2\pi i\left(j_1 j_2 . j_3 \ldots . j_{m-1} j_m\right)}\left|1\right\rangle\right)$$

$$\otimes\left(\left|0\right\rangle + e^{2\pi i\left(j_1 . j_2 j_3 \ldots j_{m-1} j_m\right)}\left|1\right\rangle\right) \otimes \left(\left|0\right\rangle + e^{2\pi i\left(0. j_1 j_2 \ldots j_{m-1} j_m\right)}\left|1\right\rangle\right)\left|v\right\rangle$$

- As we saw earlier, we can ignore the bits to the left of the binary point because they contribute to multiples of $e^{2\pi i} = 1$, so the state is equivalent to

$$\left|\psi_{2^{m-1}}\right\rangle = \frac{1}{\sqrt{2^m}}\left(\left|0\right\rangle + e^{2\pi i\left(0. j_m\right)}\left|1\right\rangle\right) \otimes \cdots \otimes \left(\left|0\right\rangle + e^{2\pi i\left(0. j_{m-1} j_m\right)}\left|1\right\rangle\right)$$

$$\cdots \otimes \left(\left|0\right\rangle + e^{2\pi i\left(0. j_2 j_3 \ldots j_{m-1} j_m\right)}\left|1\right\rangle\right) \otimes \left(\left|0\right\rangle + e^{2\pi i\left(0. j_1 j_2 \ldots j_{m-1} j_m\right)}\left|1\right\rangle\right)\left|v\right\rangle$$

Quantum Solution

- By comparing the *QFT* obtained earlier

$$QFT\left|j\right\rangle \equiv QFT\left|j_{n-1}\ldots j_0\right\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}\left|k\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_0)}\left|1\right\rangle\right)\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_1j_0)}\left|1\right\rangle\right)\cdots$$

$$\times\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_{n-2}\ldots j_1j_0)}\left|1\right\rangle\right)\frac{1}{\sqrt{2}}\left(\left|0\right\rangle + e^{2\pi i(0.j_{n-1}j_{n-2}\ldots j_1j_0)}\left|1\right\rangle\right)$$

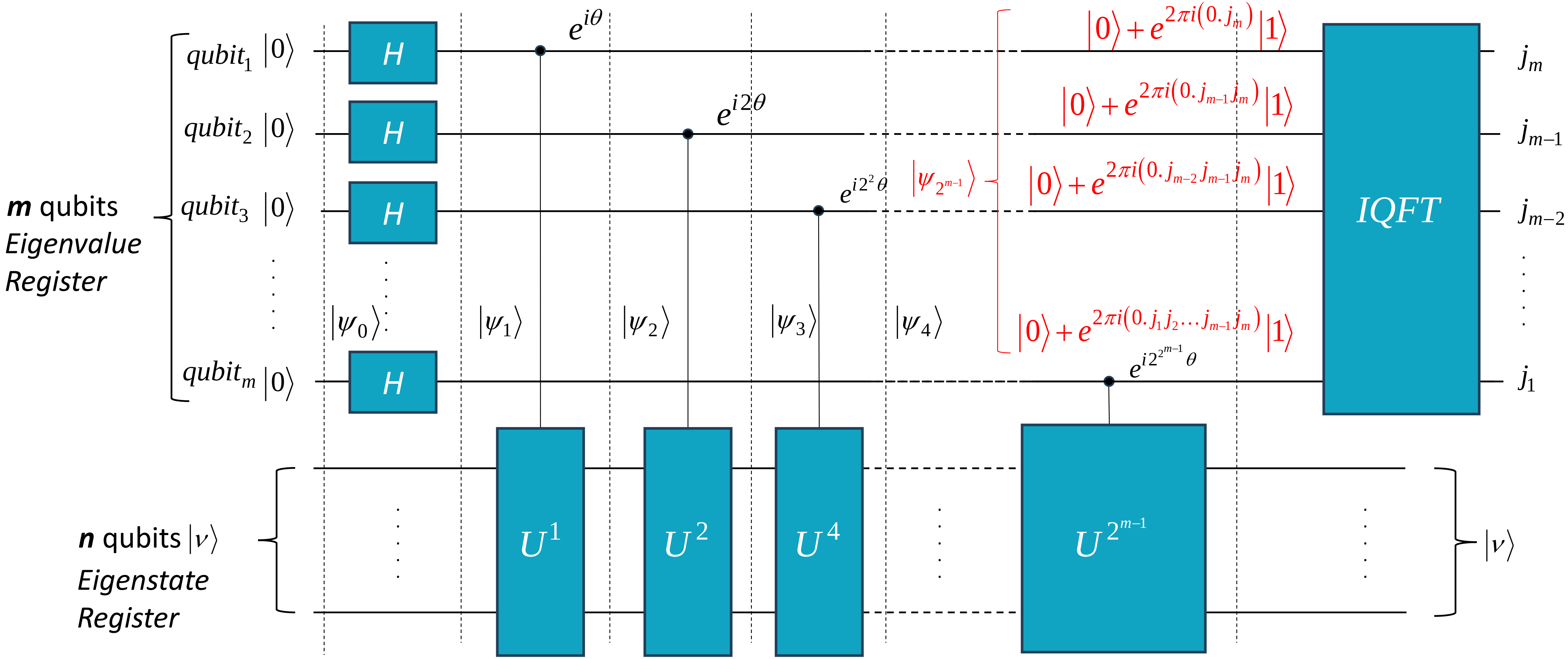
and $\left|\psi_{2^{m-1}}\right\rangle$, it turns out that

$$\left|\psi_{2^{m-1}}\right\rangle = \frac{1}{\sqrt{2^m}}\left(\left|0\right\rangle + e^{2\pi i\left(0.j_m\right)}\left|1\right\rangle\right)\otimes\cdots\otimes\left(\left|0\right\rangle + e^{2\pi i\left(0.j_{m-1}j_m\right)}\left|1\right\rangle\right)$$

$$\cdots\otimes\left(\left|0\right\rangle + e^{2\pi i\left(0.j_2j_3\ldots j_{m-1}j_m\right)}\left|1\right\rangle\right)\otimes\left(\left|0\right\rangle + e^{2\pi i\left(0.j_1j_2\ldots j_{m-1}j_m\right)}\left|1\right\rangle\right)\left|\nu\right\rangle = \left(QFT\left|j_1j_2\ldots j_m\right\rangle\right)\otimes\left|\nu\right\rangle$$

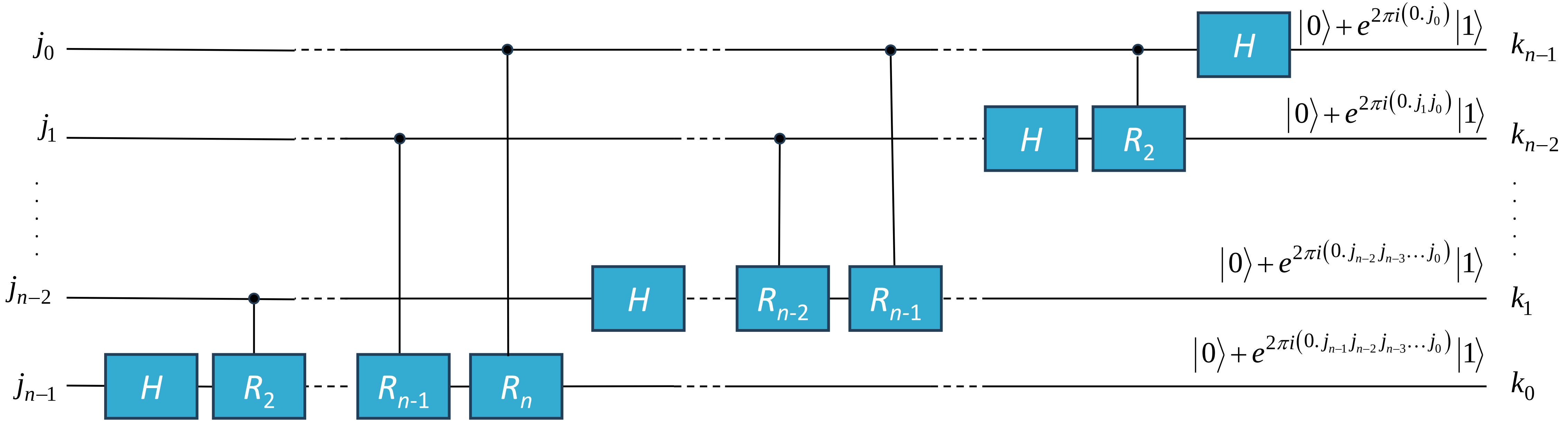
Quantum Solution

$$\left(QFT\left|j_1 j_2 \dots j_m\right\rangle\right)\otimes\left|\nu\right\rangle$$

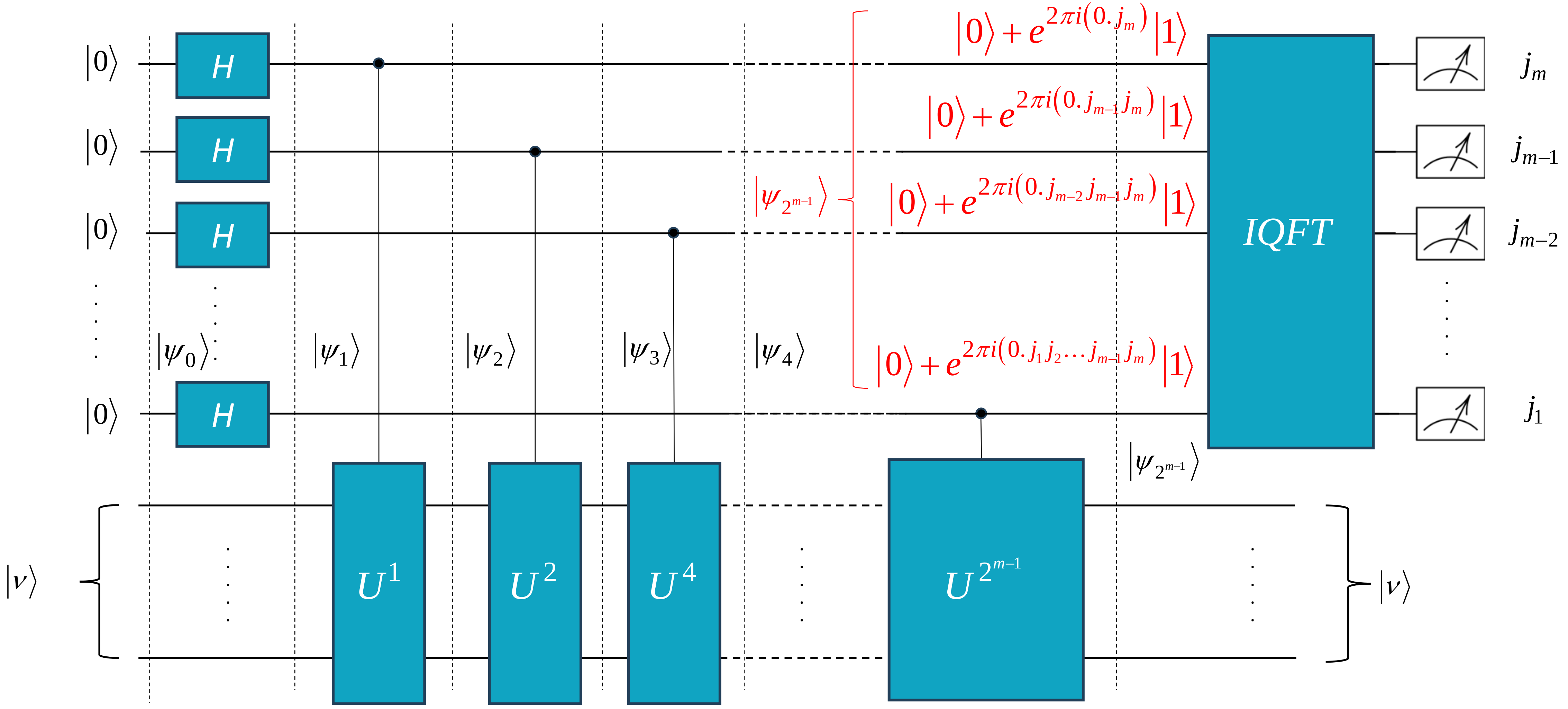


Quantum Fourier Transform

$$j = j_{n-1} j_{n-2} \cdots j_1 j_0$$



Quantum Solution



Quantum Solution

- In other words, $\left|\psi_{2^{m-1}}\right\rangle$, is exactly the *QFT* of $\left|j_1 j_2 \ldots j_m\right\rangle$
- Thus, we can find $\left|j_1 j_2 \ldots j_m\right\rangle$, by taking the *IQFT* of the eigenvalue register, resulting in:

$$\left|j_1 j_2 \ldots j_m\right\rangle\left|\nu\right\rangle$$

- This completes the quantum circuit for phase estimation
- After measuring these qubits and obtaining $j_1, j_2, \ldots, j_m$ we do a little postprocessing by calculating

$$j = 0.j_1 j_2 \ldots j_m$$
$$= \frac{j_1}{2} + \frac{j_2}{2^2} + \cdots + \frac{j_m}{2^m}$$

$\rightarrow$ Then, the phase of the eigenvalue is $\theta = 2\pi j$, and the eigenvalue is $e^{i\theta}$

Quantum Solution

- To estimate the eigenvalue to *m* bits of precision, we need *m* Hadamard gates, *m* controlled-U^p operations, and an *IQFT* on *m* qubits that takes $O(m^2)$ gates
- Altogether, the number of gates is $O(m^2)$
- The classical method takes $O(N)=O(2^n)$ elementary arithmetic operations, so depending on the number of bits of precision *m*, the quantum method can be faster, although it assumes we can create $|v\rangle$ and do controlled-U^p operations

Multiple Eigenstates

- Assume we have two eigenstates of *U*, which we call $|v_1\rangle$ and $|v_2\rangle$, with corresponding eigenvalues $e^{2\pi i\lambda_1}$ and $e^{2\pi i\lambda_2}$
- Say we are using the previous phase estimation algorithm but **prepare the eigenstate register** in the following **superposition** of $|v_1\rangle$ and $|v_2\rangle$

$$\frac{\sqrt{3}}{2}|v_1\rangle+\frac{1}{2}|v_2\rangle$$

- We also have the *m* qubits that each start in the state $|0\rangle$, so the initial state of the phase estimation circuit is

$$|0\ldots000\rangle\left(\frac{\sqrt{3}}{2}|v_1\rangle+\frac{1}{2}|v_2\rangle\right)=\frac{\sqrt{3}}{2}|0\ldots000\rangle|v_1\rangle+\frac{1}{2}|0\ldots000\rangle|v_2\rangle$$

Multiple Eigenstates

- Following the same calculation as the previous section, the final state of the phase estimation circuit

$$\frac{\sqrt{3}}{2}\left|j_1 j_2 \ldots j_m\right\rangle\left|v_1\right\rangle+\frac{1}{2}\left|j_1^{'} j_2^{'} \ldots j_m^{'}\right\rangle\left|v_2\right\rangle \qquad [11]$$

where $0.j_1 j_2 \ldots j_m$ is an *m*-bit approximation of λ_1 and $0.j_1^{'} j_2^{'} \ldots j_m^{'}$ is an *m*-bit approximation of λ_2

- Then, when we measure the qubits at the end of the circuit, we get an approximation of λ_1 with probability 3/4 or an approximation of λ_2 with probability of 1/4

Period of Modular Exponentiation

Modular Arithmetic Break

1. Congruence Relation

Two integers a and b are congruent modulo n if they have the same remainder when divided by n:

$$a \equiv b \pmod{n} \quad \text{if and only if } n \mid (a - b).$$

- Example: $17 \equiv 2 \pmod{5}$, because $17 - 2 = 15$ is divisible by 5.

Modular Arithmetic Break

2. Addition, Subtraction, and Multiplication

These operations are well-defined in modular arithmetic, meaning the results are consistent with standard arithmetic modulo n:

1. **Addition**: $(a + b) \mod n = [(a \mod n) + (b \mod n)] \mod n$.
2. **Subtraction**: $(a - b) \mod n = [(a \mod n) - (b \mod n)] \mod n$.
3. **Multiplication**: $(a \cdot b) \mod n = [(a \mod n) \cdot (b \mod n)] \mod n$.

- Example: $7 + 5 \equiv 12 \equiv 2 \pmod 5$.

Modular Arithmetic Break

3. Division (Modular Inverse)

Division in modular arithmetic is not straightforward. It requires the use of a **modular inverse**:

$$a \cdot a^{-1} \equiv 1 \pmod{n}.$$

- The modular inverse of a modulo n exists if and only if a and n are **coprime** ($\gcd(a, n) = 1$).
- Example: The modular inverse of 3 modulo 7 is 5, since $3 \cdot 5 \equiv 15 \equiv 1 \pmod{7}$.

Modular Arithmetic Break

4. Reduction Property

Any integer can be reduced modulo n without changing its equivalence class:

$$a \equiv a \mod n.$$

Modular Arithmetic Break

5. Exponentiation (Fermat's Little Theorem and Euler's Theorem)

- **Fermat's Little Theorem**: If p is a prime number and a is not divisible by p, then:

$$a^{p-1} \equiv 1 \pmod{p}.$$

- **Euler's Theorem**: If a and n are coprime, then:

$$a^{\phi(n)} \equiv 1 \pmod{n},$$

where $\phi(n)$ is Euler's totient function.

Period of Modular Exponentiation

- *Modular exponentiation* is taking powers of a number **modulo** some other number

Period of Modular Exponentiation

- *Modular exponentiation* is taking powers of a number **modulo** some other number
- For example, consider powers of 2 taken modulo 7

$$
\begin{aligned}
2^0 \bmod 7 &= 1 \bmod 7\\
2^1 \bmod 7 &= 2 \bmod 7\\
2^2 \bmod 7 &= 4 \bmod 7\\
2^3 \bmod 7 &= 8 \bmod 7 = 1 \bmod 7\\
2^4 \bmod 7 &= 16 \bmod 7 = 2 \bmod 7\\
2^5 \bmod 7 &= 32 \bmod 7 = 4 \bmod 7\\
2^6 \bmod 7 &= 64 \bmod 7 = 1 \bmod 7\\
2^7 \bmod 7 &= 128 \bmod 7 = 2 \bmod 7\\
2^8 \bmod 7 &= 256 \bmod 7 = 4 \bmod 7\\
2^9 \bmod 7 &= 512 \bmod 7 = 1 \bmod 7
\end{aligned}
$$

$\longleftarrow$ *Notice the results are 1,2,4,... repeated*

Period of Modular Exponentiation

- The *period* or *order r* of the **modular exponentiation** is *the length of the repeating sequence*, so in this example, $r = 3$

Period of Modular Exponentiation

- Next, let us consider another example: powers of 3 taken modulo 10

$$3^0 \bmod 10 = 1 \bmod 10$$
$$3^1 \bmod 10 = 3 \bmod 10$$
$$3^2 \bmod 10 = 9 \bmod 10$$
$$3^3 \bmod 10 = 27 \bmod 10 = 7 \bmod 10$$
$$3^4 \bmod 10 = 81 \bmod 10 = 1 \bmod 10$$
$$3^5 \bmod 10 = 243 \bmod 10 = 3 \bmod 10$$
$$3^6 \bmod 10 = 729 \bmod 10 = 9 \bmod 10$$
$$3^7 \bmod 10 = 2187 \bmod 10 = 7 \bmod 10$$
$$3^8 \bmod 10 = 6561 \bmod 10 = 1 \bmod 10$$

- Now, the pattern is 1,3,9,7 repeated, and the *period* is $r = 4$

Period of Modular Exponentiation

- In both examples, the repeated sequences started with a 1
- This is always true because $a^0 = 1$ for any positive integer *a*
- Furthermore, the modular exponential $a^x \bmod N$ always follows a repeated pattern **as long as** *a* and *N* are *relatively prime* (i.e., their **greatest common divisor** is 1 (*GCD*(*a*,*N*)=1), so they share no common factors except 1)
- This fact comes from a branch of mathematics called *number theory*

Period of Modular Exponentiation

- Since the repeated sequence always starts with 1, another way to define the period is as the **smallest positive exponent *r* such that**

$$a^r \bmod N = 1 \bmod N$$

- For example, with $2^x \bmod 7$, r = 3 was the smallest positive exponent to yield 1mod 7, so it takes r = 3 terms for the pattern to repeat to 1

- For the second example, r = 4 is the smallest exponent so that

$$3^x \bmod 10 = 3^4 \bmod 10 = 1 \bmod 10$$

- More generally, since the numbers repeat every *r* powers

$$\boxed{a^{x+r} \bmod N = a^x \bmod N}$$

Period of Modular Exponentiation

- The problem is to find the **period** of the **modular exponential**
- We often just call this problem **period finding** or **order finding**
- Note the *period r* must be *less than N,* and so the challenge is to find the period for *large N*

Classical Solution/Repeated Squaring

- Finding a **single modular exponent** is fast using the **repeated squaring method**
-
- For example, say we want to find

$$91^{43} \bmod 131$$

- We do not want to calculate 91^{43}, as this is a very big number
- Instead, we want to calculate it in pieces, taking it modulo 131 as we go

Classical Solution/Repeated Squaring

- To do this, we express the exponent in binary

$$
\begin{aligned}
43\big|_{10} &= 101011\big|_{2} \\
&= 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 \\
&= 1\cdot 32 + 0\cdot 16 + 1\cdot 8 + 0\cdot 4 + 1\cdot 2 + 1\cdot 1
\end{aligned}
$$

- So, we want to calculate

-

$$
\begin{aligned}
91^{43} \bmod 131 &= 91^{1\cdot 32+0\cdot 16+1\cdot 8+0\cdot 4+1\cdot 2+1\cdot 1} \bmod 131 \\
&= 91^{1\cdot 32} 91^{0\cdot 16} 91^{1\cdot 8} 91^{0\cdot 4} 91^{1.2} 91^{1.1} \bmod 131 \\
&= \left(91^{32}\right)^1 \left(91^{16}\right)^0 \left(91^{8}\right)^1 \left(91^{4}\right)^0 \left(91^{2}\right)^1 \left(91^{1}\right)^1 \bmod 131
\end{aligned}
$$

[10]

Classical Solution/Repeated Squaring

- Now, we use the standard number theoretic fact that

$$a \times b \mod N \equiv \left[a \mod N \times b \mod N \right] \mod N \quad [11]$$

- So, we want to calculate

$$91^{43} \mod 131 = \Big[\left(91^{32} \mod 131\right) \times \left(91^{8} \mod 131\right) \times \left(91^{2} \mod 131\right) \times \left(91^{1} \mod 131\right) \Big] \mod 131 \quad [12]$$

Classical Solution/Repeated Squaring

- By exploiting the above result, we square powers of 91 modulo 131, and we can calculate them by starting with 91^1, then squaring it to get 91^2, then squaring it to get 91^4, then squaring it to get 91^8, and so forth:

$$91^1 \bmod 131 = 91 \bmod 131 = 91$$

$$91^2 \bmod 131 = 91^2 \bmod 131 = 8281 \bmod 131 = 28 \bmod 131 = 28$$

$$91^4 \bmod 131 = 28^2 \bmod 131 = 784 \bmod 131 = 129 \bmod 131 = 129$$

$$91^8 \bmod 131 = 129^2 \bmod 131 = 16641 \bmod 131 = 4 \bmod 131 = 4$$

$$91^{16} \bmod 131 = 4^2 \bmod 131 = 16 \bmod 131 = 16$$

$$91^{32} \bmod 131 = 16^2 \bmod 131 = 256 \bmod 131 = 125 \bmod 131 = 125$$

Classical Solution/Repeated Squaring

- By repeatedly squaring, we were able to calculate these using relatively small numbers. Plugging these into Eq. [12] we get

$$
\begin{aligned}
91^{43} \bmod 131 &= \big[(125 \bmod 131)\times(4 \bmod 131)\times(28 \bmod 131) \\
&\quad \times(91 \bmod 131)\big] \bmod 131 \\
&= 125\cdot 4\cdot 28\cdot 91 \bmod 131 \\
&= 1274000 \bmod 131 \\
&= 25 \bmod 131
\end{aligned}
$$

- In this case, multiplying 125·4·28·91 is small enough to be done on an ordinary calculator, but if it were not, it could also be multiplied progressively, e.g.,

$$91^{43} \bmod 131 = (125)^1(16)^0(4)^1(129)^0(28)^1\left(91^1\right)^1 \bmod 131$$

Classical Solution/Repeated Squaring

$$
\begin{aligned}
125\cdot 4\cdot 28\cdot 91 \bmod 131 &= 125\cdot\left(4\cdot\left(28\cdot 91\right)\right) \bmod 131\\
&= 125\cdot\left(4\cdot\left(2548\right)\right) \bmod 131\\
&= 125\cdot\left(4\cdot\left(59\right)\right) \bmod 131\\
&= 125\cdot\left(236\right) \bmod 131\\
&= 125\cdot\left(105\right) \bmod 131\\
&= 13125 \bmod 131\\
&= 25 \bmod 131
\end{aligned}
$$

To go from the second to the third line, we used 2548 mod 131 = 59 mod 131

Altogether, the repeated squaring method allows us to compute modular exponentials (we showed that 9143 mod 131 = 25 mod 131), and we were able to calculate this using relatively small numbers, as opposed to trying to calculate 91^{43} from the start

Computational Complexity

- For the computational complexity of the **repeated square method**, say we are calculating $a^x \bmod N$, where ***x* is an *n*-bit binary number**
- Then, we start with ***a*** and square it ***n* – 1 times**, **modulo *N***
- Once we have these, we may have to multiply them together, which following the **progressive approach** above takes up to $n - 1$ multiplications, modulo N

Computational Complexity

- Together, this is $(n - 1) + (n - 1) = 2(n - 1) = O(n)$ **decimal arithmetic operations** modulo N
- We may be interested in the number of **bit operations**, however, rather than **decimal operations**
- Recall from elementary school that you can multiply two d-digit numbers by multiplying $O(d)^2$ pairs of digits

Computational Complexity

- For example, to multiply 123 and 456

$$\begin{array}{r} 123 \\ \times 456 \\ \hline 738 \\ 6150 \\ +49200 \\ \hline 56088 \end{array}$$

- That is, we multiplied each digit of 123 by 6, then multiplied each digit of 123 by 5, and then multiplied each digit of 123 by 4, doing the carries along the way
- Altogether, we multiplied 9 pairs of numbers
- Then, we added 9 digits together, ignoring the zeros that we padded on the right

- So, the total number of operations on digits is

$$9+9 = d^2 + d^2 = 2d^2 = O(d^2)$$

Computational Complexity

- Similarly, to multiply two ***n*-bit strings**, this method takes $\boldsymbol{O(n^2)}$ multiplications of pairs of bits and additions
- For example, to multiply 101_2 and 110_2,

$$
\begin{array}{r}
101 \\
\times 110 \\
\hline
000 \\
1010 \\
+10100 \\
\hline
11110
\end{array}
$$

- Now, the repeated squares method takes $O(n)$ multiplications/squares, and we just saw that each of these takes $O(n^2)$ gates, *so the total gate complexity of modular exponentiation is $O(n^3)$,* which is still a polynomial and is hence efficient

Computational Complexity

- Although calculating a single modular exponential using the previous repeated squares method is fast, calculating the period finding is slow because, when N is large, we may need to calculate so many individual modular exponentials before a pattern forms
- There is no known efficient algorithm for period finding
- Computer algebra systems often have functions for finding the period of modular exponentials
- Although they are slow for large N, they are fast for small values

Quantum Solution

- A quantum computer can efficiently find the period of $a^x \bmod N$, by utilizing a quantum gate U that performs **modular multiplication**, which multiplies a number y by $a \bmod N$, so it maps

$$U|y\rangle = |ay \bmod N\rangle, \text{ where } 0 \le y \le N-1 \qquad [13]$$

- If N can be written using n bits, then $|y\rangle$ would require n qubits

- Before moving on, we need to take into account some results from the number theory

Quantum Solution

- Since

$$a = a \bmod N, \text{ for } a < N$$

$$b = b \bmod N, \text{ for } b < N$$

then

$$a \times b \bmod N = \left[\left(a \bmod N\right) \times \left(b \bmod N\right)\right] \bmod N$$

- From this fact we get the formula

$$a^x \bmod N = \left(\left(a^{x-1} \bmod N\right) \times a \bmod N\right) \bmod N$$

- Since $a < N$ and $a \bmod N = a$, this reduces to

$$a^x \bmod N = \left(\left(a^{x-1} \bmod N\right) \times a\right) \bmod N$$

[14]

Quantum Solution

$$U|y\rangle = |ay \bmod N\rangle, \text{ where } 0 \le y \le N-1$$

- By repeatedly applying U to $|1\rangle$, we get

$$U^1|1\rangle = \left|a^1 \bmod N\right\rangle = |a \bmod N\rangle \quad \longleftarrow \quad \text{This follows from the definition [13]}$$

- Let's now calculate

$$U^2|1\rangle = U\left(U|1\rangle\right) = U\left(|a \bmod N\rangle\right) = \left|\left((a \bmod N)\times \mathrm{a}\right) \bmod N\right\rangle = \left|a^2 \bmod N\right\rangle$$

$$U^3|1\rangle = U\left(U^2|1\rangle\right) = U\left(\left|a^2 \bmod N\right\rangle\right) = \left|\left(\left(a^2 \bmod N\right)\times \mathrm{a}\right) \bmod N\right\rangle = \left|a^3 \bmod N\right\rangle$$

$$\vdots$$

$$U^r|1\rangle = \left|a^r \bmod N\right\rangle = \left|a^0 \bmod N\right\rangle = |1 \bmod N\rangle$$

- The last term is $a^r \bmod N = 1 \bmod N$ because r is the order of $a^x \bmod N$
- In other words, $a^r \bmod N = a^0 \bmod N$ because the sequence repeats itself

Quantum Solution

$U|y\rangle = |ay \bmod N\rangle, \text{ where } 0 \le y \le N-1$

- Since the repeated sequence always starts with 1, we may define
  $U^0|1\rangle = |1 \bmod N\rangle = |a^0 \bmod N\rangle$
- Thus, by repeatedly applying U to $|1\rangle$, we get ***a*** to some power

$$\left.\begin{aligned} U^0|1\rangle &= |a^0 \bmod N\rangle \\ U^1|1\rangle &= |a^1 \bmod N\rangle \\ U^2|1\rangle &= |a^2 \bmod N\rangle \\ U^3|1\rangle &= |a^3 \bmod N\rangle \\ &\vdots \\ U^r|1\rangle &= |a^0 \bmod N\rangle \end{aligned}\right\} \longleftarrow$$

This is exactly the modular exponential $a^x \bmod N$ because exponentiation is repeated multiplication

Quantum Solution

- Now, consider a superposition of

$$\begin{array}{ccccc} \left|a^0 \bmod N\right\rangle & \left|a^1 \bmod N\right\rangle & \left|a^2 \bmod N\right\rangle & \cdots & \left|a^{r-1} \bmod N\right\rangle \\ \downarrow & \downarrow & \downarrow & & \downarrow \\ e^{-2\pi is(0)/r} & e^{-2\pi is(1)/r} & e^{-2\pi is(2)/r} & \cdots & e^{-2\pi is(r-1)/r} \end{array}$$

with respective coefficients

where s is an integer taking values 0, 1, ..., $r-1$

$$\left|v_s\right\rangle = \frac{1}{\sqrt{r}}\Big(e^{-2\pi is(0)/r}\left|a^0 \bmod N\right\rangle + e^{-2\pi is(1)/r}\left|a^1 \bmod N\right\rangle + \cdots$$

$$\cdots + e^{-2\pi is(r-2)/r}\left|a^{r-2} \bmod N\right\rangle + e^{-2\pi is(r-1)/r}\left|a^{r-1} \bmod N\right\rangle\Big)$$

$$= \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-2\pi is(k)/r}\left|a^k \bmod N\right\rangle$$

Quantum Solution

- Let us show that $|\nu_s\rangle$ is an eigenvector of U with eigenvalue $e^{2\pi is/r}$

$$U|\nu_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-2\pi is(k)/r} U|a^k \bmod N\rangle$$

$$= \frac{1}{\sqrt{r}}\Big(e^{-2\pi is(0)/r}U|a^0 \bmod N\rangle + e^{-2\pi is(1)/r}U|a^1 \bmod N\rangle + \cdots$$

$$\cdots + e^{-2\pi is(r-2)/r}U|a^{r-2} \bmod N\rangle + e^{-2\pi is(r-1)/r}U|a^{r-1} \bmod N\rangle\Big)$$

$$= \frac{1}{\sqrt{r}}\Big(e^{-2\pi is(0)/r}|a^1 \bmod N\rangle + e^{-2\pi is(1)/r}|a^2 \bmod N\rangle + \cdots$$

$$\cdots + e^{-2\pi is(r-2)/r}|a^{r-1} \bmod N\rangle + e^{-2\pi is(r-1)/r}\underbrace{|a^r \bmod N\rangle}_{|a^0 \bmod N\rangle}\Big)$$

Quantum Solution

$$= \frac{1}{\sqrt{r}}\left(e^{-2\pi is(r-1)/r}\left|a^0 \bmod N\right\rangle + e^{-2\pi is(0)/r}\left|a^1 \bmod N\right\rangle + e^{-2\pi is(1)/r}\left|a^2 \bmod N\right\rangle + \cdots\right.$$

$$\left.\cdots + e^{-2\pi is(r-2)/r}\left|a^{r-1} \bmod N\right\rangle\right)$$

- Multiplying by

$$1 = e^0 = e^{2\pi is/r - 2\pi is/r} = e^{2\pi is/r}e^{-2\pi is/r} \qquad \rightarrow$$

$$U\left|\nu_s\right\rangle == e^{2\pi is/r}\frac{1}{\sqrt{r}}\left(e^{-2\pi is(r)/r}\left|a^0 \bmod N\right\rangle + e^{-2\pi is(1)/r}\left|a^1 \bmod N\right\rangle\right.$$

$$\left.+e^{-2\pi is(2)/r}\left|a^2 \bmod N\right\rangle + \cdots + e^{-2\pi is(r-1)/r}\left|a^{r-1} \bmod N\right\rangle\right)$$

Quantum Solution

$$\left|v_s\right\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-2\pi i s(k)/r}\left|a^k \bmod N\right\rangle$$

- Since ***s*** is an integer, the first coefficient can be written as

$$e^{-2\pi i s(r)/r} = e^{-2\pi i s} = 1$$

- Furthermore, since $e^{-2\pi i s(0)/r} = 1$, the previous equation can be written as

$$U\left|\nu_s\right\rangle = e^{2\pi i s/r}\frac{1}{\sqrt{r}}\Big(e^{-2\pi i s(0)/r}\left|a^0 \bmod N\right\rangle + e^{-2\pi i s(1)/r}\left|a^1 \bmod N\right\rangle$$

$$+e^{-2\pi i s(2)/r}\left|a^2 \bmod N\right\rangle + \cdots + e^{-2\pi i s(r-1)/r}\left|a^{r-1} \bmod N\right\rangle\Big)$$

$$= e^{2\pi i s/r}\left(\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{-2\pi i s(k)/r}\left|a^k \bmod N\right\rangle\right) = e^{2\pi i s/r}\left|v_s\right\rangle \qquad \square$$

- Thus, $U\left|\nu_s\right\rangle = e^{2\pi i s/r}\left|\nu_s\right\rangle$, where $s \in \{0, 1, \ldots, r-1\}$

Quantum Solution

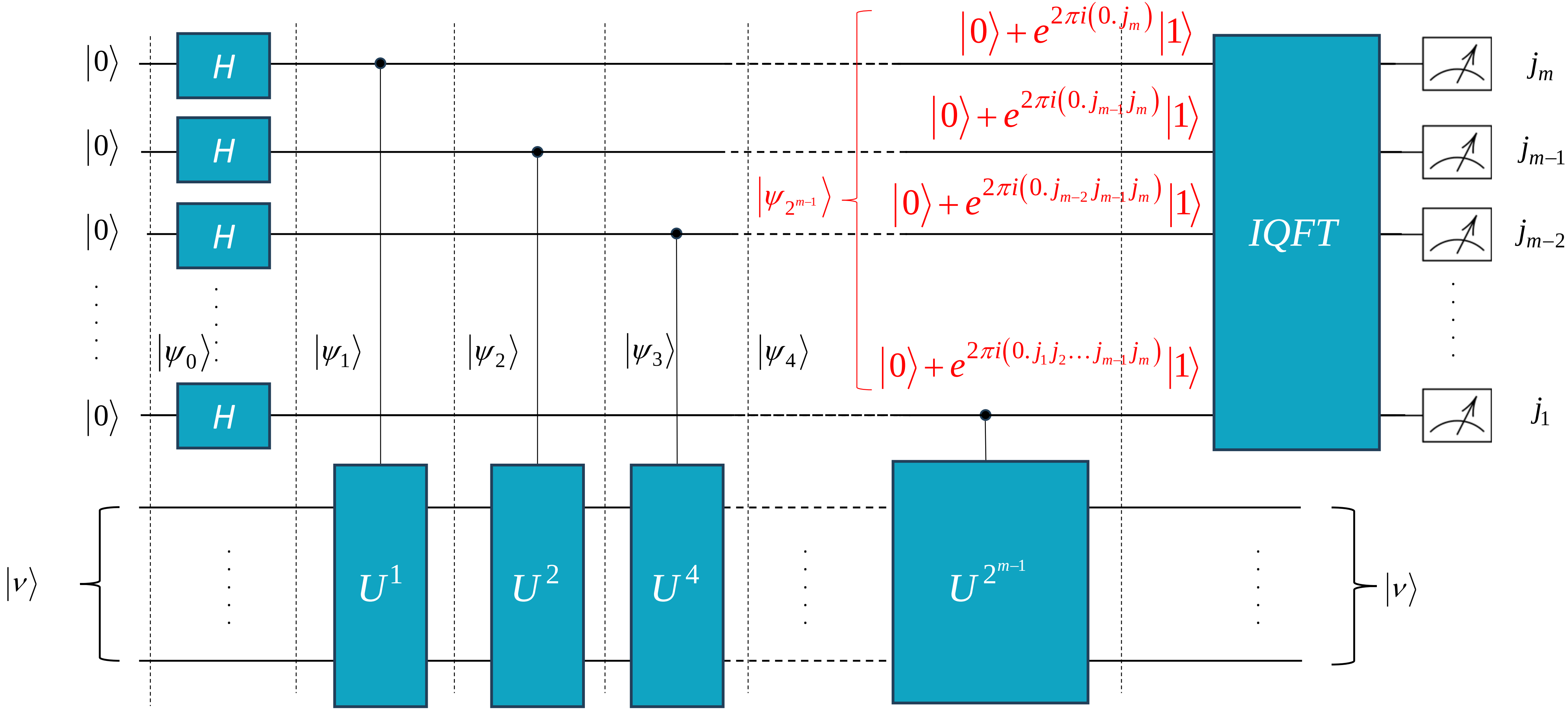
$$U|v_s\rangle = e^{2\pi is/r}|v_s\rangle,$$

- Since $|v_s\rangle$ is an eigenvector of *U*, we can use the phase estimation algorithm to estimate its eigenvalue $e^{2\pi is/r}$

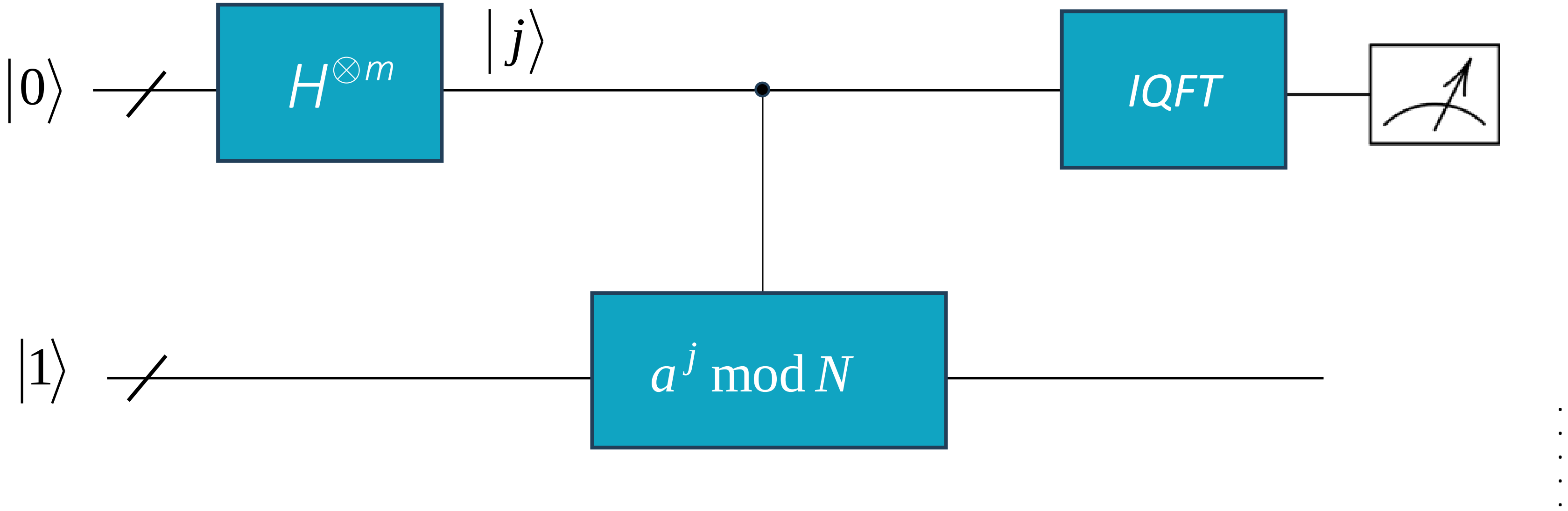
- That is, we can find *s*/*r* for some *s*, which will allow us to find *r*, the period of the modular exponential, hence solving the problem

- To do this, however, we need to work out three more items:
  1. How to construct the **controlled-*U* gates** for the **phase estimation algorithm**
  2. How to construct **the eigenvector** $|v_s\rangle$ **for the phase estimation algorithm**
  3. How to take the result of the phase estimation, which is an *m*-bit estimate for *s/r*, and **find *r***

Quantum Circuit For Phase Estimation



Quantum Circuit For Period Finding



Quantum Solution

- For the *first item*, we need

  controlled-U

  controlled-U^2

  controlled-U^4

  $\vdots$

  controlled-$U^{2^{m-1}}$

- We choose to approximate the eigenvalue to $m=O(n)$ bits

Quantum Solution

- Writing the control qubit as $|z\rangle$ and the target qubits as $|y\rangle$, the operation of cU^{2^j} is **defined** as follows:

$$cU^{2^j}|z\rangle|y\rangle = |z\rangle\left|a^{z2^j} y \text{ mod } N\right\rangle$$

- This way, when *z* = 0, the target remains unchanged as *y*, and when *z* = 1, the target is multiplied by the modular exponential $a^{2^j} \text{ mod } N$

$$cU^{2^j}|0\rangle|y\rangle = |0\rangle|y \text{ mod } N\rangle = |0\rangle|y\rangle$$

$$cU^{2^j}|1\rangle|y\rangle = |1\rangle\left|a^{2^j} y \text{ mod } N\right\rangle$$

Quantum Solution

- As we saw earlier, we have a fast classical method for computing $\left| a^x \mod N \right\rangle$ that takes $O(n^2)$, and we can convert this into a reversible circuit and hence a quantum gate

Quantum Solution

- For the *second item*, we need to prepare an **eigenvector** of U
- A trick is, instead of preparing a single eigenvector of U, we prepare the following equal superposition of them:

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

- We will see very shortly that this superposition is easy to construct
- This superposition will be used in the phase estimation algorithm

Quantum Solution

- Since the eigenvalue of $|v_s\rangle$ is $e^{2\pi is/r}$, the phase estimation will yield an *m*-bit approximation to s/r for one $s = 0,1,...,r-1$, where each value of s has a probability of $1/r$
- Now, let us show that the equal superposition is easy to construct
- Plugging in the definition of $|v_s\rangle$, the equal superposition becomes

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|v_s\rangle = \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{-2\pi is(k)/r}\left|a^k \mod N\right\rangle$$

$$= \frac{1}{r}\sum_{k=0}^{r-1}\underbrace{\sum_{s=0}^{r-1}e^{-2\pi is(k)/r}}_{\substack{r \text{ when } k=0 \\ 0 \text{ otherwise}}}\left|a^k \mod N\right\rangle \qquad [15]$$

Quantum Solution

- Let us show why the term in parenthesis is *r* when *k* = 0 and why it is 0 when $k \neq 0$

- First, when *k* = 0, the term in parenthesis is

$$\sum_{s=0}^{r-1} e^{-2\pi i s(k)/r} = \sum_{s=0}^{r-1} e^{0} = \sum_{s=0}^{r-1} 1 = r$$

- Next, when $k \neq 0,$ let us define $\omega = e^{-2\pi i k/r}$

- Then, the term in parenthesis is

$$\sum_{s=0}^{r-1} e^{-2\pi i s k/r} = \sum_{s=0}^{r-1} \left(e^{-2\pi i k/r}\right)^{s} = \sum_{s=0}^{r-1} \omega^{s} = \frac{1-\omega^{r}}{1-\omega}$$

Quantum Solution

- By exploiting $\omega = e^{-2\pi ik/r}$, we obtain

$$\frac{1-\omega^r}{1-\omega} = \frac{1-e^{-2\pi ik}}{1-e^{-2\pi ik/r}} = \frac{1-1}{1-e^{-2\pi ik/r}} = 0$$

- Thus, we have proved that the term in parenthesis of Eq. [12] is 0 when $k \neq 0$
- Thus, the equal superposition that we were considering is equal to

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left| v_s \right\rangle = \frac{1}{r} r \left| a^0 \text{ mod } N \right\rangle = \left| 1 \text{ mod } N \right\rangle$$

Quantum Solution

- Thus, the equal superposition of the eigenstates $|v_s\rangle$ is precisely equal to

$$|1 \bmod N\rangle$$

which is easily prepared by starting all the qubits as

$$|00\ldots00\rangle$$

and then applying an *X*-gate to the rightmost qubit to yield

$$|00\ldots01\rangle = |1 \bmod N\rangle$$

Quantum Solution

- Then, from [11], if we use the phase estimation algorithm, we get an approximation to the phase of one $|v_s\rangle$, with $s \in \{0,...,r-1\}$, with probability $1/r$
- That is, since $|v_s\rangle$ is an eigenstate of U with eigenvalue $e^{2\pi i s/r}$, the phase estimation yields $0.j_1 j_2 \dots j_m$ which is an m-bit approximation to $j = s/r$

Quantum Solution

- For example, let us implement this in Qiskit to find the order of $3^x \bmod 7$

Probability	Binary Approx. of s/r	Decimal Approx. of s/r
16.7963%	$\|00000\rangle$	0
11.4759%	$\|00101\rangle$	0.1562
11.4760%	$\|01011\rangle$	0.3438
16.7963%	$\|10000\rangle$	0.5
11.4759%	$\|10101\rangle$	0.6562
11.4760%	$\|11011\rangle$	0.8438

- The table shows the likely values for our approximation of s/r

Quantum Solution

- For example, we have an 11.4759% chance of measuring the qubits to be $|00101\rangle$, so 0.00101 is a binary approximation of *s/r*
- Converting 0.00101 to decimal

$$0.00101 = 0\times\frac{1}{2} + 0\times\frac{1}{2^2} + 1\times\frac{1}{2^3} + 0\times\frac{1}{2^4} + 1\times\frac{1}{2^5} = \frac{1}{8} + \frac{1}{32} = 0.15625$$

we get that *s/r* is approximately 0.1562

Probability	Binary Approx. of s/r	Decimal Approx. of s/r
16.7963%	$\vert 00000\rangle$	0
11.4759%	$\vert 00101\rangle$	0.1562
11.4760%	$\vert 01011\rangle$	0.3438
16.7963%	$\vert 10000\rangle$	0.5
11.4759%	$\vert 10101\rangle$	0.6562
11.4760%	$\vert 11011\rangle$	0.8438

Quantum Solution

- Now for the *third item,* how do we take an approximation φ to s/r, like $\varphi = 0.1562$ from above, and find s and r?

- We use a method called *continued fractions*

- A continued fraction has the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_l}}}}$$

for some non-negative integer l

Quantum Solution

- For example, from the table above, consider the number 0.1562

- To express this as a continued fraction, we begin by expressing 0.1562 as 1562/10000, which we express as a mixed number, i.e., a whole number 0 and fractional part 1562/10000:

$$0.1562 = \frac{1562}{10000} = 0 + \frac{1562}{10000}$$

- Next, we invert the fractional part to get

$$0 + \frac{1}{\frac{10000}{1562}}$$

Quantum Solution

- Again, we invert the fractional part and then express it as a mixed number

$$0+\frac{1}{6+\frac{628}{1562}}=0+\frac{1}{6+\frac{1}{\frac{1562}{628}}}=0+\frac{1}{6+\frac{1}{2+\frac{306}{628}}}$$

- Continuing this, we eventually arrive at

$$0.1562=0+\frac{1}{6+\frac{1}{2+\frac{1}{2+\frac{1}{19+\frac{1}{8}}}}}$$

Quantum Solution

- By listing all the whole numbers, plus the very last denominator, we can write the continued fraction as

$$[a_0, a_1, \ldots, a_5] = [0,6,2,2,19,8]$$

- The reason why we care about continued fractions is they allow us to find rational approximations to numbers by truncating the continued fraction
- These are called *convergents*
- For example, for 0.1562, the convergents are:

$$0th\ convergent = [0] = 0$$

$$1st\ convergent = [0,6] = 0 + \frac{1}{6} = \frac{1}{6}$$

Quantum Solution

$$2nd\ convergent = [0,6,2] = 0 + \cfrac{1}{6 + \cfrac{1}{2}} = \frac{2}{13}$$

$$3rd\ convergent = [0,6,2,2] = 0 + \cfrac{1}{6 + \cfrac{1}{2 + \cfrac{1}{2}}} = \frac{5}{32}$$

$$4rd\ convergent = [0,6,2,2,19] = 0 + \cfrac{1}{6 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{19}}}} = \frac{97}{621}$$

Quantum Solution

$$5th\ convergent = [0,6,2,2,19,8] = 0 + \cfrac{1}{6 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{19 + \cfrac{1}{8}}}}} = \frac{781}{5000}$$

- In this example, the 5*th* convergent contains all the terms of the continued fraction, and so the 5*th* convergent is exactly

$$0.1562 = 781/5000 = 1562/10000$$

- The higher the convergent, the better the approximation to 0.1562

Quantum Solution

- Why are we interested in computing *continued fraction representations* and the *convergents* created in the process?
- The reason lies with the following

  Theorem: if a fraction s/r satisfies

$$\left|\frac{s}{r}-\varphi\right| \le \frac{1}{2r^2}$$

  then s/r appears in the list of *convergents* of φ

Quantum Solution

- For our period finding problem, $\varphi = 0.1562$ is a guess for s/r, where r is the period of the modular exponential $a^x \bmod N$, and s is an integer between 0 and $r-1$

- Note r must be less than N

- Then, looking at the *convergents*, the best approximation to s/r such that $r < N = 7$ is $1/6$

Quantum Solution

- Thus, using the *convergents* of continued fractions, we were able to guess that $s = 1$ and $r = 6$

- To check whether our guess is correct, we can calculate $3^r \bmod 7$ and see if we get $1 \bmod 7$:

$$3^6 \bmod 7 = 1 \bmod 7$$

- Thus, with this measurement result, we successfully found the period $r = 6$

- Note it is known that the continued fraction algorithm yields a guess for s and r in $O(n^3)$ steps, if s and r are n-bit numbers

Quantum Solution

- From the previous table of *significant* measurement outcomes of the Qiskit circuit for phase estimation, some other likely estimates for s/r are

$$0, 0.3438, 0.5, 0.6562, \text{ and } 0.8438$$

- For 0, we get $s = 0$ and no guess for r, so if we get this value, we need to run the quantum circuit again in hopes of a better outcome

Quantum Solution

- For the other values, we can use the continued fraction algorithm and get the following guesses for *s* and *r*

Probability	Binary Approx. of s/r	Decimal Approx. of s/r	Guess of s/r	3^r mod 7
16.7963%	$\|00000\rangle$	0	N/A	N/A
11.4759%	$\|00101\rangle$	0.1562	$1/6$	1
11.4760%	$\|01011\rangle$	0.3438	$1/3$	6
16.7963%	$\|10000\rangle$	0.5	$1/2$	2
11.4759%	$\|10101\rangle$	0.6562	$2/3$	6
11.4760%	$\|11011\rangle$	0.8438	$5/6$	1

- For example, for 0.3438, the continued fraction algorithm yields *s* = 1 and *r* = 3

Quantum Solution

- Checking if this guess for the period is correct, we calculate

$$3^r \bmod 7 = 3^3 \bmod 7 = 6 \bmod 7 \neq 1 \bmod 7,$$

  so **3 is not the period**

- Then, we run the quantum circuit again, hoping to get a better guess for *r*

- The number of times we may have to repeat the quantum circuit is small enough that it does not affect the overall runtime of the algorithm

- See Nielsen and Chuang for a proof of this fact

Quantum Solution

- Speaking of the overall runtime, a detailed analysis of the errors shows that we can take $m = O(n)$
- Then, the quantum algorithm takes:
  - one X-gate to prepare the eigenvector register in the state $|00\ldots01\rangle$,
  - m Hadamard gates, and
  - m controlled-U^{power} gates, and
  - an *IQFT* on m qubits.

Quantum Solution

- Each of the m controlled-U^{power} gates takes $O(n^2)$ gates for a total of $O(mn^2) = O(n^3)$ gates

- The *IQFT* takes $O(m^2) = O(n^2)$ gates

- Finally, the continued fraction algorithm takes $O(n^3)$ gates

- Thus, the gate complexity of the quantum period algorithm is $\boldsymbol{O(n^3)}$, which is a polynomial in n, so it is efficient

FACTORING

The Problem

- Say we are given a number N that is the product of two *prime numbers* p and q
- The goal is to factor N, i.e., to find its factors p and q
- Note that RSA cryptography is founded on the belief that factoring is a challenging task for classical computers

Classical Solution

- The best-known classical algorithm for factoring is the *general number field sieve*

- Its workings are beyond the scope of this course, but to factor an *n*-bit number, its runtime is

$$e^{\left(\sqrt[3]{64/9}+o(1)\right)(\ln n)^{1/3}(\ln\ln n)^{2/3}}$$

- This is an example of a *sub-exponential* function

- It grows faster than polynomial, so factoring is not efficient for classical computers, but it is also not exponential because of the natural logarithms

Note on Complexity

- Any algorithm is said to have **sub-exponential time complexity** if for any input size n, it runs in time $2^{o(n)}$.
- Example #1

$$2^{\sqrt{n}}$$

- Example #2

$$e^{\left(\sqrt[3]{64/9}+o(1)\right)\left(\ln n\right)^{1/3}\left(\ln\ln n\right)^{2/3}}$$

Quantum Solution: Shor's Algorithm

- An efficient quantum algorithm for factoring was invented by Peter Shor in 1994

- This means quantum computers, if they can be built at scale, can break RSA cryptography

- Historically, this greatly increased the amount of money for research in quantum computing and is one of the reasons why quantum computing has developed into the field it is today

Shor's Algorithm-1st Step

To factor $N = pq$, Shor's algorithm consists of the following three steps:

- Pick any integer number $1 < a < N$

- See if we were extraordinarily lucky and picked a multiple of p or q by calculating $\mathrm{GCD}(a,N)$

- If the GCD is not 1, then the GCD is a nontrivial common factor of a and N, and so we have found one of the factors of N. Let us call it $p = \mathrm{GCD}(a,N)$

- Then, $q = N/p$, and we are done factoring

- If $\mathrm{GCD}(a,N) = 1$, we continue to the next step

Shor's Algorithm-2nd Step

- Find the *period r* of $a^x \bmod N$

- Note this is believed to be hard for classical computers, but it is efficient for quantum computers using the **period finding algorithm** showed earlier

- Make sure the *period r* is **even**; if it is **odd**, go back to step 1 and pick a different a

- Also, calculate $a^{r/2} \bmod N$ and make sure it does not equal $N - 1$; if it equals $N - 1$, go back to step 1 and pick a different a

Shor’s Algorithm-2nd Step

- It is known that there is at least a 50% chance of picking a “good” *a* that meets both criteria, so we will not have to try too many times

- The proof of this is beyond the scope of this course, but **Theorem 5.3** of **Nielsen and Chuang** has details

Shor's Algorithm-3rd Step

- Since we calculated the period r in the previous step, we know that $a^r = 1 \bmod N$
- Subtracting 1 from both sides, this means

$$a^r - 1 = (1 \bmod N) - 1 = (1 \bmod N) - (1 \bmod N) = 0 \bmod N$$

- This says $a^r - 1$ divided by N has a remainder of 0, so $a^r - 1$ is a multiple of N
- Let us call the multiple k, so

$$a^r - 1 = kN$$

- Also substituting $N = pq$, we get

$$a^r - 1 = kpq$$

Shor's Algorithm-3rd Step

- Now, factoring the left-hand side, we get

$$(a^{r/2} - 1)(a^{r/2} + 1) = kpq$$

- From Step 2, we know that r is even

- So, $a^{r/2}$ is an integer, and $a^{r/2} \pm 1$ are also integers

- Now, for the product of $a^{r/2} - 1$ and $a^{r/2} + 1$ to equal kpq, at least one of the terms $a^{r/2} - 1$ or $a^{r/2} + 1$ must contain p and/or q as a factor

Shor's Algorithm-3rd Step

- That is, for some integers c and d such that $cd = k$, we have three possibilities for $a^{r/2} - 1$ and $a^{r/2} + 1$

1. $\underbrace{\left(a^{r/2}-1\right)}_{c}\underbrace{\left(a^{r/2}+1\right)}_{dpq}=kpq$

2. $\underbrace{\left(a^{r/2}-1\right)}_{cp}\underbrace{\left(a^{r/2}+1\right)}_{dq}=kpq$

3. $\underbrace{\left(a^{r/2}-1\right)}_{cpq}\underbrace{\left(a^{r/2}+1\right)}_{d}=kpq$

Shor's Algorithm-3rd Step

- Let us show that the first and third cases are not possible, i.e., $a^{r/2} - 1$ and $a^{r/2} + 1$ are not multiples of N

- $(a^{r/2} - 1) \bmod N \neq 0 \bmod N$ and $(a^{r/2} + 1) \bmod N \neq 0 \bmod N$, so neither has N as a factor

Proof

- **Case 1** - Let us start with $(a^{r/2} - 1) \bmod N = 0 \bmod N$ and show that this equation is not true

- If we add 1 (which is equivalent to $1 \bmod N$) to both sides, we get

$$a^{r/2} = 1 \bmod N$$

Shor's Algorithm-3rd Step

- We know that r is the period of a^x mod N, however, which means r is the smallest value of x such that $a^x = 1 \bmod N$

- Thus, it cannot be that $a^{r/2} = 1 \bmod N$, otherwise $r/2$ would be a smaller value of x such that $a^x = 1 \bmod N$

- Therefore, the equation $(a^{r/2} - 1) \bmod N = 0 \bmod N$ is incorrect, and it must be that $(a^{r/2} - 1) \bmod N \neq 0 \bmod N$, so $(a^{r/2} - 1)$ does not have N as one of its factors

Shor's Algorithm-3rd Step

- **Case 2** - Next, let us show that $(a^{r/2} + 1) \bmod N = 0 \bmod N$ is not true
- If we subtract 1 from both sides, we get $a^{r/2} \bmod N = -1 \bmod N$
- Recall that the modulus works in a "cyclical" fashion
- For example, with a 12-hour clock, 15 o'clock corresponds to 3 o'clock
- Similarly, −1 o'clock corresponds to 11 o'clock

Shor's Algorithm-3rd Step

- Thus, our modular equation becomes $a^{r/2}$ mod N= N-1 mod N
- This is not true, however, because in Step 2, we made sure that $a^{r/2}$ mod $N \neq$ N-1 mod N
- Thus, $a^{r/2} + 1$ also does not have N as one of its factors
- Thus, only the second case is possible:

$$\underbrace{\left(a^{r/2}-1\right)}_{cp}\underbrace{\left(a^{r/2}+1\right)}_{dq}=kpq$$

Shor's Algorithm-3rd Step

- This means $a^{r/2} - 1$ and $a^{r/2} + 1$ each share a nontrivial factor with $N = pq$, and we can obtain them using the greatest common divisor:

$$p = \mathrm{GCD}(a^{r/2} - 1, N)$$

$$q = \mathrm{GCD}(a^{r/2} + 1, N)$$

- Thus, we have factored N

Example

- As an example, we want to factor $N = 15$
- We begin Shor's algorithm by picking a value for a such that $1 < a < N$
- **Case #1.** Assume we pick $a = 6$
  a. We calculate $\text{GCD}(a,N) = \text{GCD}(6,15) = 3$, which means that 3 is a factor of both a and N
  b. Thus, we have found one of the factors of N that we call $p = 3$
  c. The other factor is $q = N/p = 15/3 = 5$
  d. So, we have factored $N = 15$ into $pq = 3 \cdot 5$, and we are done

Example

- Let us work out what might happen if we did not have such a lucky pick for a
- **Case #2.** Assume we pick $a = 2$
    a. We calculate $\mathrm{GCD}(a,N) = \mathrm{GCD}(2,15) = 1$, so we continue to Step 2
    b. We find the period of $a^x \bmod N = 2^x \bmod 15$
    c. We can use the quantum period finding algorithm to determine $r = 4$
    d. This period is even, and we confirm that

       $a^{r/2} + 1 = 2^2 + 1 = 5 \bmod 15 \neq 0 \bmod 15$

       $a^{r/2} \bmod N = 2^2 \bmod 15 = 4 \bmod 15 \neq N\text{-}1 \bmod N = 14 \bmod 15$
    e. Calculate the factors

       $p = \mathrm{GCD}(a^{r/2} - 1, N) = \mathrm{GCD}(2^2 - 1, 15) = \mathrm{GCD}(3, 15) = 3$

       $q = \mathrm{GCD}(a^{r/2} + 1, N) = \mathrm{GCD}(2^2 + 1, 15) = \mathrm{GCD}(5, 15) = 5$

- Thus, the factors of $N = 15$ are $p = 3$ and $q = 5$

Conclusions

- The bottleneck for Shor's algorithm is Step 2, finding the period of the modular exponential
- It is efficient on a quantum computer, but there is no known polynomial-time algorithm for a classical computer

Conclusions

- Although quantum computers would break RSA cryptography, their creation does not necessarily mean the end of digital privacy
- Already, efforts are underway to choose a new public-key cryptography standard that is resistant to quantum computers
- Post-quantum cryptography refers to such classical cryptographic algorithms that are resistant to attacks from future quantum computers
- Besides this, there is also quantum key distribution protocols, such as BB84 which will be covered later, that are secure from quantum computers
- They require a quantum network, however, to be used