Shor Algorithm

Ming Xu

Shanghai Key Lab of Trustworthy Computing
East China Normal University

December 23, 2022

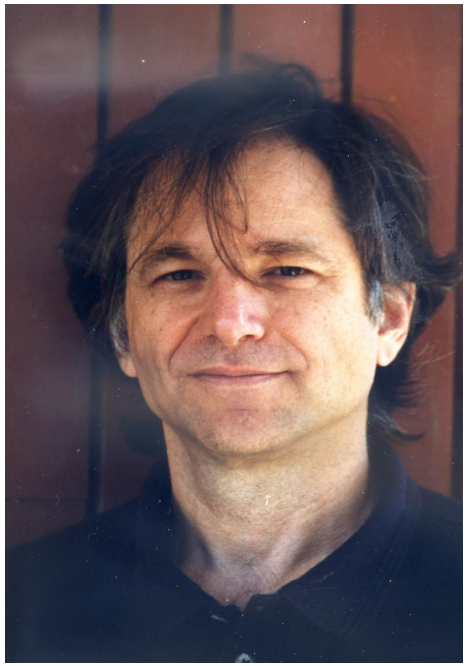
RSA Encryption

RSA Encryption

RSA inventors

R. L. Rivest

A. Shamir

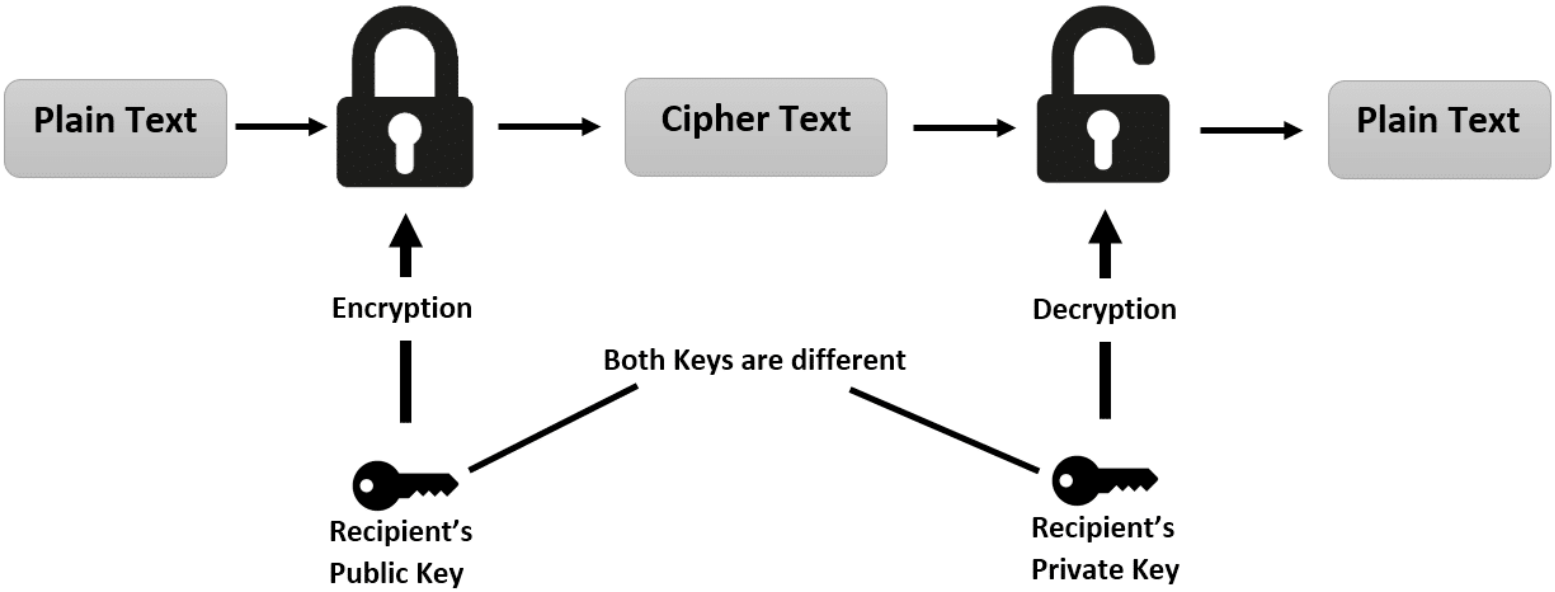


L. Adleman

RSA Encryption

Public & Private key

- **Asymmetric encryption**: different keys for encrypting and decrypting
- **Public key**: is used to encrypt the message
- **Private key**: is used to decrypt the message



RSA Encryption

RSA relies on **factors** and **large prime numbers**.

- Choose two prime numbers 31 and 37.
- What is the product 31×37?
  $\rightarrow$ Easy to solve: 1147.
- What are all the factors of 1147?
  $\rightarrow$ Much hard to solve!

Similarly ...

- What are the factors of $414,863$?
- What are the factors of $2,450,400,991$?

RSA Encryption

Steps of the whole process

1. **Choose** two very large prime numbers p and q.
2. **Let** $N = p \cdot q$.
3. **Let** $\varphi(N) = (p-1) \cdot (q-1)$.
4. **Choose** $e \in \mathbb{N}$ satisfying e and $\varphi(N)$ are co-prime.
5. **Get** $d \in [k, k+\varphi(N))$ for some k, which is the unique solution of

$$e \cdot x \equiv 1 \mod \varphi(N).$$

6. **Publish** N and e, which pair is your public key.
7. **Keep** N and d secret, which pair is your private key.

RSA Encryption

Encryption

Encrypt the plain text with digital m $(0 \leq m < N)$:

$$c = \text{Encrypt}(m) = m^e \mod N.$$

Decryption

Decrypt c to plain text m:

$$m = \text{Decrypt}(c) = c^d \mod N.$$

RSA Encryption

A simple example

- Choose two prime numbers $p = 2$ and $q = 7$.
- $N = 2 \cdot 7 = 14$.
- $\varphi(N) = (2-1) \cdot (7-1) = 6$.
- Choose $e = 5$.
- Get the unique solution of $5x \equiv 1 \mod 6$, which is 11.
- Public key is $(5, 14)$.
- Private key is $(11, 14)$.

RSA Encryption

A simple example

Encryption

The public key is $(5, 14)$. Let's send a one-letter secret message B. Instead of B, let's take the number 2.

$$\text{Encrypted value} = 2^5 \bmod 14 = 4$$

Thus 4 is the encrypted message corresponding to letter D.

Decryption

After getting the secret message D, which can be translated to number 4.

$$\text{Decrypted value} = 4^{11} \bmod 14 = 2$$

RSA Encryption

Proof of Correctness

Before we start the proof, some preparations are given here.

Euler function:

$$\varphi(N) = \|\{i \in \mathbb{N} \mid i < N \text{ and } \gcd(i, N) = 1\}\|.$$

Question: How do we compute the Euler function?

RSA Encryption

Proof of Correctness

Before we start the proof, some preparations are given here.

Euler function:

$$\varphi(N) = \|\{i \in \mathbb{N} \mid i < N \text{ and } \gcd(i, N) = 1\}\|.$$

Question: How do we compute the Euler function?

For N, whose prime factorization is $N = p_1^{\varepsilon_1} p_2^{\varepsilon_2} \dots p_k^{\varepsilon_k}$, the Euler function is

$$\varphi(N) = N \cdot (1 - \tfrac{1}{p_1}) \cdot (1 - \tfrac{1}{p_2}) \cdot \dots \cdot (1 - \tfrac{1}{p_k}).$$

RSA Encryption

Proof of Correctness

Before we start the proof, some preparations are given here.

Euler Theorem

If x and N are co-prime, then $x^{\varphi(N)} \equiv 1 \mod N$.

Fermat Little Theorem

If p is prime and $x < p$, then $x^{p-1} \equiv 1 \mod p$.

RSA Encryption

Proof of Correctness

Recall the process of RSA Encryption.

1. Choose two very large prime numbers p and q.
2. Let $N = p \cdot q$.
3. Let $\varphi(N) = (p-1) \cdot (q-1)$. **Euler function!**
4. Choose $e \in \mathbb{N}$ satisfying e and $\varphi(N)$ are co-prime.
5. Get $d \in (0, \varphi(N))$ which is the unique solution of $e \cdot x \equiv 1 \mod \varphi(N)$.
6. Publish N and e, which are your public key.
7. Keep N and d secret, which are your private key.

As $e \cdot d \equiv 1 \mod \varphi(N)$, we have $e \cdot d = 1 + k \cdot \varphi(N)$ for some k.

RSA Encryption

Proof of Correctness

Recall the encrypting and decrypting process.

$$m = \text{Decrypt}(c) = c^d \mod N.$$

Thus to show them correct, it suffices to show

$$m = m^{e \cdot d} \mod N$$

which is equal to

$$m \equiv m^{e \cdot d} \mod N.$$

RSA Encryption

Proof of Correctness

Next, we will prove $m^{e \cdot d} \equiv m \mod N$.
There are three cases, depending on the number of prime common factors of m and N, to be discussed.

1. m and N are co-prime.

$$m^{\varphi(N)} \equiv 1 \mod N \qquad \text{Euler Theorem}$$
$$m^{k \cdot \varphi(N)} = \left(m^{\varphi(N)}\right)^k \equiv 1 \mod N$$
$$m^{e \cdot d} = m^{k \cdot \varphi(N)+1} \equiv (1 \cdot m) \mod N = m$$

RSA Encryption

Proof of Correctness

2. $\gcd(m, N) = p$. In this case we have $m = p^{\ell} \cdot m_1$ for some $\ell \geq 1$ and m_1 satisfying $\gcd(m_1, N) = 1$.

$$\begin{aligned} m^{\varphi(N)} &= p^{\ell \cdot \varphi(N)} \cdot m_1^{\varphi(N)} \\ &= p^{(q-1)\cdot(p-1)\cdot\ell} \cdot m_1^{(q-1)\cdot(p-1)} \\ &\equiv 1^{(p-1)\cdot\ell} 1^{(p-1)} \mod q \qquad \text{Fermat Little Theorem} \\ &= 1. \end{aligned}$$

We can express $m^{\varphi(N)}$ as $1 + s \cdot q$ for some s, and $m^{k\cdot\varphi(N)}$ as $1 + t \cdot q$ for some t. Then we have

$$\begin{aligned} m^{e\cdot d} = m^{k\cdot\varphi(N)+1} = (1 + t \cdot q) \cdot m &= m + t \cdot q \cdot p^{\ell} \cdot m_1 \\ &\equiv m \mod N. \end{aligned}$$

The case is symmetric when $\gcd(m, N) = q$.

RSA Encryption

Proof of Correctness

3. $\gcd(m, N) = p \cdot q = N$. This case is impossible as $m < N$.

Overall, we can conclude that

$$m^{e \cdot d} \equiv m \mod N.$$

Therefore the RSA protocol is correct.

Motivation

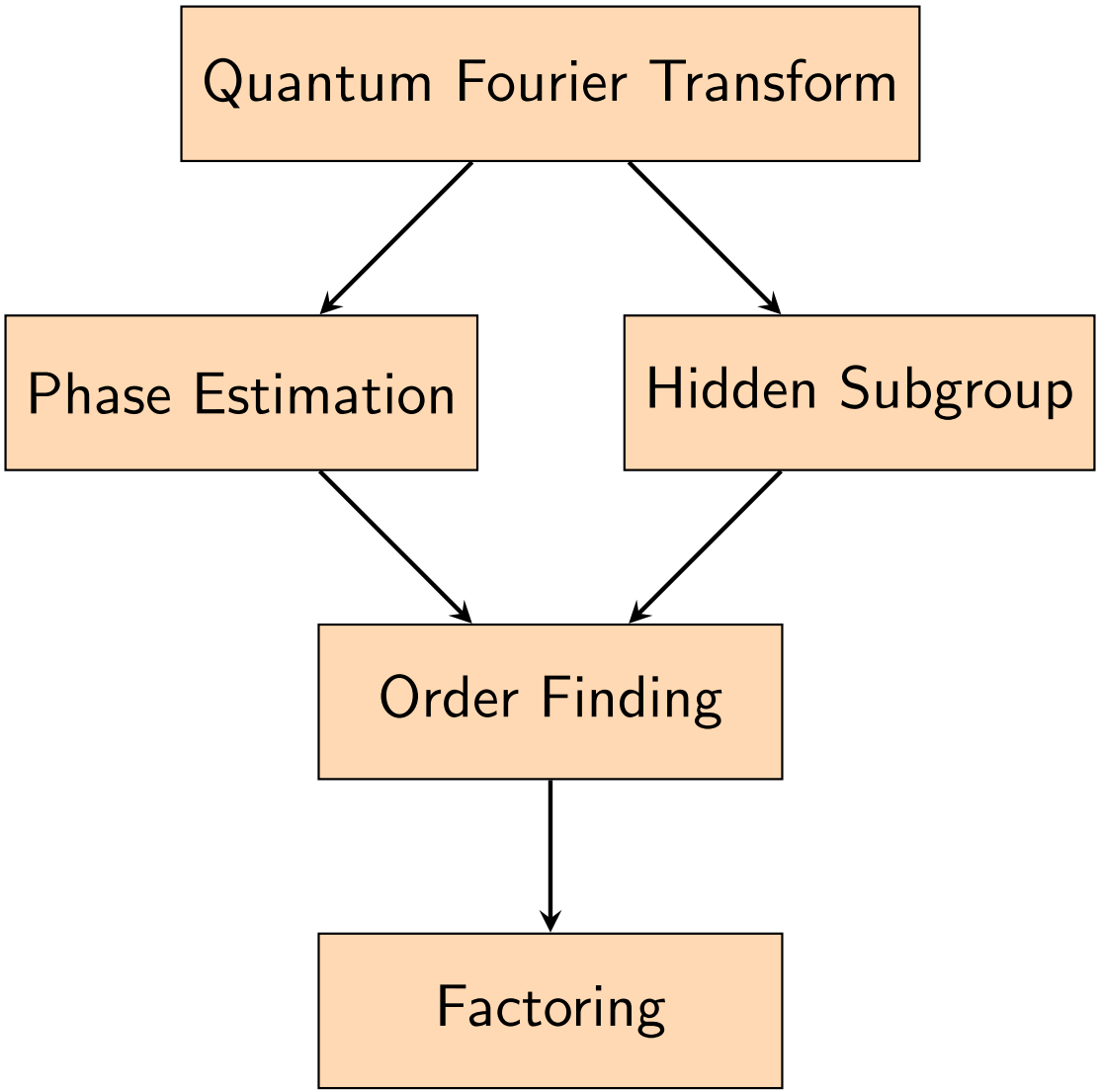
How do we break the RSA protocol? The point is factoring the integer N.

- In classical world, we can recognize factors from 2 to $\lfloor\sqrt{N}\rfloor$.

  For example, for integer 1147, we enumerate all possible factors from 2 to $\lfloor\sqrt{1147}\rfloor = 34$, and test them one by one as a subroutine.

- More precise, the asymptotic distribution of prime numbers tells there are nearly $N/\ln(N)$ prime numbers in $[2, N)$ for sufficiently large N.
- It seems to be a polynomial-time algorithm, which is actually an exponential one! As the input N is encoded in binary, the size of which is $n = \lceil\log_2(N)\rceil$, the time $\sqrt{N}/\ln(\sqrt{N}) = 2^{n/2+1}/n$ of subroutines is exponential to input size n.
- In quantum world, **Shor algorithm is polynomial-time w.r.t. n.**

Outline of Shor Algorithm



Quantum Fourier Transform

Fourier Transform

Definition

One of the most useful ways of solving a problem in mathematics or computer science is to transform it into some other problem for which a solution is known.

- **(Discrete) Fourier Transform**: Transform between vectors of complex numbers, $(x_0, \ldots, x_{N-1}) \to (y_0, \ldots, y_{N-1})$ defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi \imath jk/N},$$

or equivalently $\mathbf{y} = FT_N(\mathbf{x})$.

Fourier Transform

Properties

1. FT_N is a linear operator, i.e. $FT_N(\mathbf{x}_1+\mathbf{x}_2)=FT_N(\mathbf{x}_1)+FT_N(\mathbf{x}_2)$;
2. FT_N is inversible, i.e. there is an operator, namely FT_N^{-1}, such that $FT_N^{-1}(FT_N(\mathbf{x}))=\mathbf{x}$.

To see them, we resort to the matrix representation of FT_N:
Define $\omega_N=\mathrm{e}^{2\pi\imath/N}$ as the Nth root of unit, then

$$FT_N=\frac{1}{\sqrt{N}}\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & \omega_N & \omega_N^2 & \cdots & \omega_N^{N-1}\\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)}\\ 1 & \omega_N^3 & \omega_N^6 & \cdots & \omega_N^{3(N-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2}\end{bmatrix},$$ which is Vandermonde.

Fourier Transform

Properties

More explicitly, the inverse FT_N^{-1} of Fourier transform can be written as

$$x_j \equiv \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k \omega_N^{-jk},$$

which is validated by

$$\begin{aligned}
\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k \omega_N^{-jk} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} x_l \omega_N^{kl} \right) \omega_N^{-jk} \\
&= \frac{1}{N} \sum_{l=0}^{N-1} x_l \left(\sum_{k=0}^{N-1} \omega_N^{k(l-j)} \right) \\
&= \frac{1}{N} x_j \left(\sum_{k=0}^{N-1} \omega_N^{k(j-j)} \right) + \frac{1}{N} \sum_{l \neq j} x_l \left(\sum_{k=0}^{N-1} \omega_N^{k(l-j)} \right) \\
&= \frac{1}{N} x_j \cdot N + \frac{1}{N} \sum_{l \neq j} x_l \cdot 0 = x_j.
\end{aligned}$$

Fourier Transform

Complexity

$\mathbf{y} = FT_N(\mathbf{x})$ is plainly in $\mathscr{O}(N^2)$ by the matrix-vector multiplication over an N-dimensional vector space.

Fourier Transform

Complexity

$\mathbf{y} = FT_N(\mathbf{x})$ is plainly in $\mathscr{O}(N^2)$ by the matrix-vector multiplication over an N-dimensional vector space.

Can we speed up the procedure?
Fast Fourier transform is the one in $\mathscr{O}(N \log N)$.

Fast Fourier Transform

Divide and Conquer

Suppose $N = 2^n$. Split $\mathbf{x} = (x_0, \ldots, x_{N-1})$ into $\mathbf{x}^{\text{even}} \uplus \mathbf{x}^{\text{odd}}$, where $\mathbf{x}^{\text{even}} = (x_0, x_2, \ldots, x_{N-2})$ and $\mathbf{x}^{\text{odd}} = (x_1, x_3, \ldots, x_{N-1})$.
Define $\mathbf{y}^{\text{even}} = FT_{N/2}\mathbf{x}^{\text{even}}$ and $\mathbf{y}^{\text{odd}} = FT_{N/2}\mathbf{x}^{\text{odd}}$.
The Fourier transform can be rearranged as

$$
\begin{aligned}
y_k &\equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk} \\
&= \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{N/2}} \sum_{2|j} x_j \omega_N^{jk} \right] + \frac{\omega_N^k}{\sqrt{2}} \left[\frac{1}{\sqrt{N/2}} \sum_{2\nmid j} x_j \omega_N^{jk} \right] \\
&= \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{N/2}} \sum_{j=0}^{N/2-1} x_{2j} \omega_{N/2}^{j\hat{k}} \right] + \frac{\omega_N^k}{\sqrt{2}} \left[\frac{1}{\sqrt{N/2}} \sum_{j=0}^{N/2-1} x_{2j+1} \omega_{N/2}^{j\hat{k}} \right] \\
&= \frac{1}{\sqrt{2}} \mathbf{y}_{\hat{k}}^{\text{even}} + \frac{\omega_N^k}{\sqrt{2}} \mathbf{y}_{\hat{k}}^{\text{odd}},
\end{aligned}
$$

where $\omega_{N/2} = \omega_N^2$ and $\hat{k} = k \mod N/2$.

Fast Fourier Transform

Complexity Analysis

Let $f(N)$ be the computational cost of FT_N, which is captured by the master equation

$$f(N) = 2f(N/2) + 3N.$$

Fast Fourier Transform

Complexity Analysis

Let $f(N)$ be the computational cost of FT_N, which is captured by the master equation

$$f(N) = 2f(N/2) + 3N.$$

Change the function $f(N)$ to $g(n)$ with $n = \log_2 N$, we get the linear recurrence

$$g(n) = 2g(n-1) + 3 \cdot 2^n.$$

It is not hard to see that the solution $g(n)$ is in $\mathscr{O}(n \cdot 2^n)$, entailing $f(N) \in \mathscr{O}(N \log N)$.

Fast Fourier Transform

Complexity Analysis

Let $f(N)$ be the computational cost of FT_N, which is captured by the master equation

$$f(N) = 2f(N/2) + 3N.$$

Change the function $f(N)$ to $g(n)$ with $n = \log_2 N$, we get the linear recurrence

$$g(n) = 2g(n-1) + 3 \cdot 2^n.$$

It is not hard to see that the solution $g(n)$ is in $\mathscr{O}(n \cdot 2^n)$, entailing $f(N) \in \mathscr{O}(N \log N)$.

Can we further speed up the procedure?
Quantum Fourier transform is the one in time polynomial in $\log N$.

Quantum Fourier Transform

Definition

- **(Discrete) Fourier Transform**: Transform between vectors of complex numbers, $(x_0, \ldots, x_{N-1}) \to (y_0, \ldots, y_{N-1})$ defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi \imath jk/N}$$

- **Quantum Fourier Transform**: Transform on an orthonormal basis $|0\rangle, \ldots, |N-1\rangle$ is defined to be a linear operator with the following action on the basis states,

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

Relation:

$$\sum_{j=0}^{N-1} x_j |j\rangle \to \sum_{j=0}^{N-1} x_j \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi \imath jk/N} |k\rangle = \sum_{k=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi \imath jk/N} |k\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$$

QUIZ

Give a direct proof that the linear transformation defined by

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

is unitary.

Quantum Fourier Transform

Implementation on n-bit quantum computer

Next, we will implement Quantum Fourier Transform on an n-bit quantum computer with $n = \log_2 N$.

- The basis $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$ is the computational basis for an n qubit quantum computer.
- The state $|j\rangle$ can be written as binary representation $j = j_1 j_2 \cdots j_n$. Or equivalently, $j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0$ in decimal. For example, for $n = 3$, $|6\rangle_{\text{dec}} = |110\rangle_{\text{bin}}$.
- It is also convenient to adopt the binary number $0.j_l j_{l+1} \ldots j_m$ to represent the decimal number $j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$.

Quantum Fourier Transform

Implementation on n-bit quantum computer

$$
\begin{aligned}
|j\rangle \to & \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} \mathrm{e}^{2\pi \imath jk/N} |k\rangle \\
= & \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \mathrm{e}^{2\pi \imath j(\sum_{\ell=1}^{n} k_\ell 2^{-\ell})} |k_1 k_2 \ldots k_n\rangle \\
= & \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \left(\bigotimes_{\ell=1}^{n} \mathrm{e}^{2\pi \imath jk_\ell 2^{-\ell}} |k_\ell\rangle \right) \\
= & \frac{1}{2^{n/2}} \bigotimes_{\ell=1}^{n} \sum_{k_\ell=0}^{1} \mathrm{e}^{2\pi \imath jk_\ell 2^{-\ell}} |k_\ell\rangle \\
= & \frac{1}{2^{n/2}} \bigotimes_{\ell=1}^{n} (|0\rangle + \mathrm{e}^{2\pi \imath j 2^{-\ell}} |1\rangle) \\
= & \frac{1}{2^{n/2}} (|0\rangle + \mathrm{e}^{2\pi \imath 0.j_n} |1\rangle)(|0\rangle + \mathrm{e}^{2\pi \imath 0.j_{n-1}j_n} |1\rangle) \cdots (|0\rangle + \mathrm{e}^{2\pi \imath 0.j_1 j_2 \ldots j_n} |1\rangle)
\end{aligned}
$$

Quantum Fourier Transform

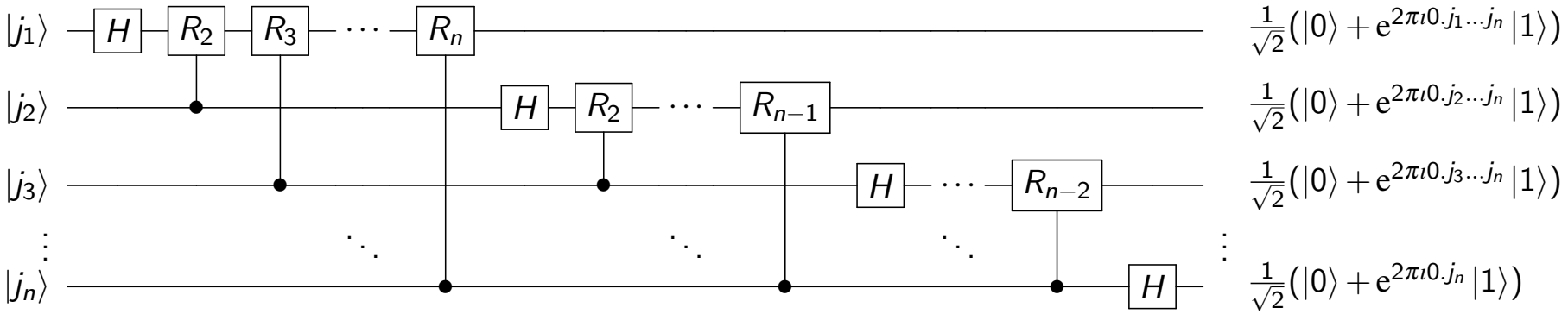


Figure: Quantum circuit for QFT algorithm

The gate R_k denotes the unitary transformation

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi\imath/2^k} \end{bmatrix}$$

After the swap operations, the state of the qubits is

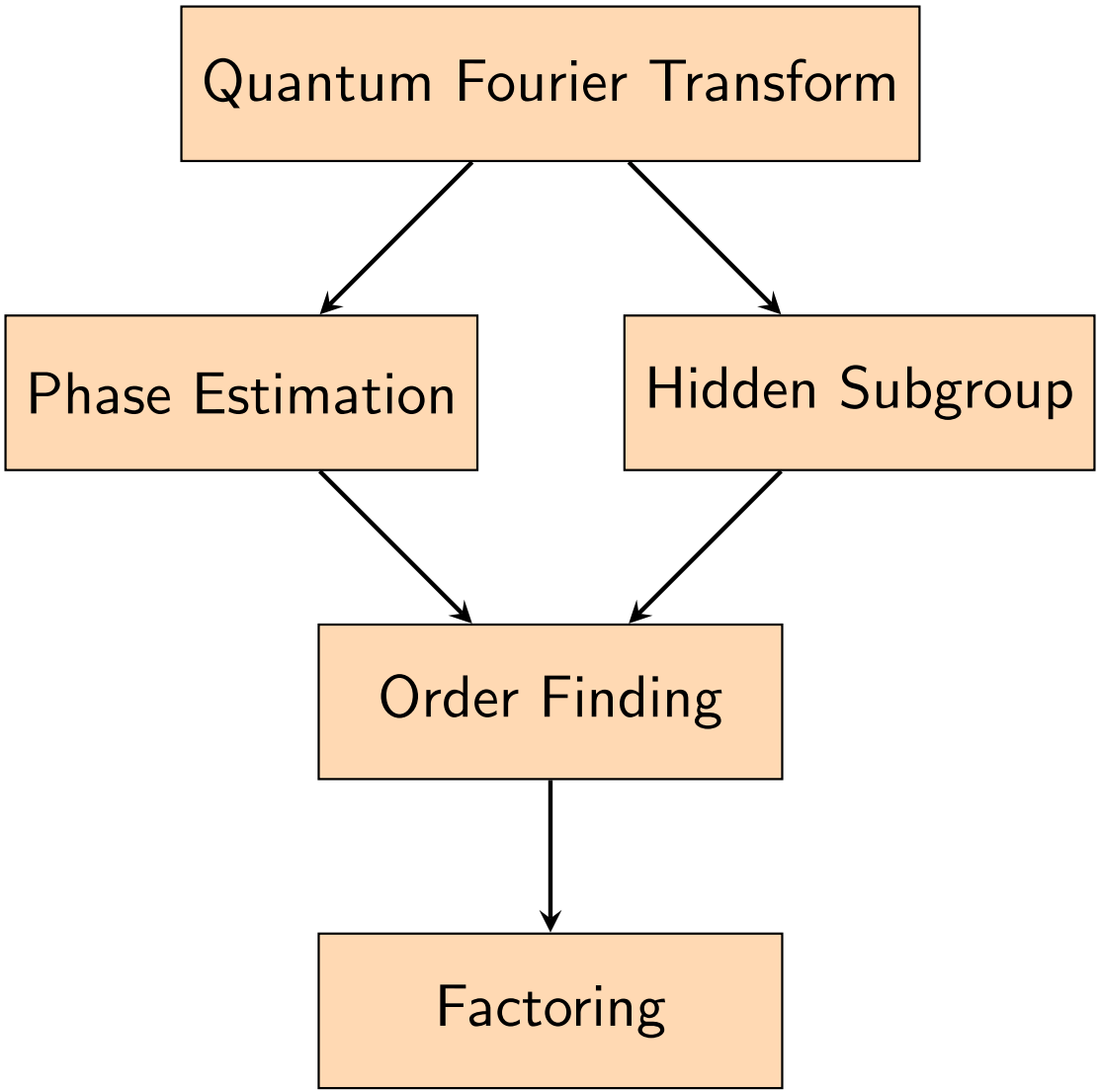
$$\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi\imath 0.j_n}|1\rangle)(|0\rangle + e^{2\pi\imath 0.j_{n-1}j_n}|1\rangle)\cdots(|0\rangle + e^{2\pi\imath 0.j_1 j_2 \ldots j_n}|1\rangle)$$

Quantum Fourier Transform

Complexity Analysis

- For the n-th qubit, there are a Hadamard gate and $n-1$ conditional rotations. Thus, for a total of n bits, there are $n+(n-1)+\cdots+1 = n(n+1)/2$ gates.
- For the swap operation, at most $n/2$ swaps are required, and each swap can be accomplished using three controlled-NOT gates.
- Overall, this circuit provides an algorithm in $\mathcal{O}(n^2)$ for performing quantum Fourier Transform.
- The best classical algorithms for computing the discrete Fourier transform on 2^n elements are algorithms such as the *Fast Fourier Transform (FFT)*, which compute the discrete Fourier transform using $\mathcal{O}(n2^n)$ gates.
- An exponential speed-up is achieved!

Outline of Shor Algorithm



Phase Estimation

Phase Estimation

Suppose that a unitary operator U performing on a quantum state $|u\rangle$,

$$|u\rangle \quad \boxed{U} \quad U|u\rangle$$

if $|u\rangle$ is exactly U's eigenvector, then

$$U|u\rangle = e^{2\pi\iota\varphi}|u\rangle,$$

where $e^{2\pi\iota\varphi}$ is the corresponding eigenvalue, i.e.,

$$|u\rangle \quad \boxed{U} \quad e^{2\pi\iota\varphi}|u\rangle$$

Goal of Phase Estimation: estimate φ.
Once φ is known, we completely know the eigenvalue $e^{2\pi\iota\varphi}$ and partially determine $U = \sum_u e^{2\pi\iota\varphi_u}|u\rangle\langle u|$.

Phase Estimation

Two Stages

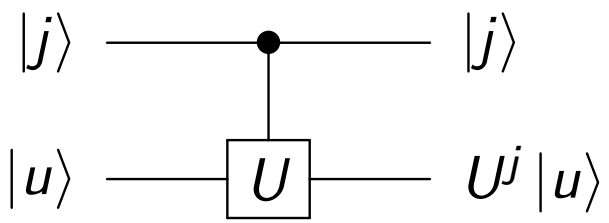
The whole phase estimation process are composed of two stages:

- **First stage**: creating a super-position with geometric series as amplitudes.
- **Second stage**: recovering the phase using inverse Fourier transform.

Phase Estimation

First Stage

A controlled-U operation



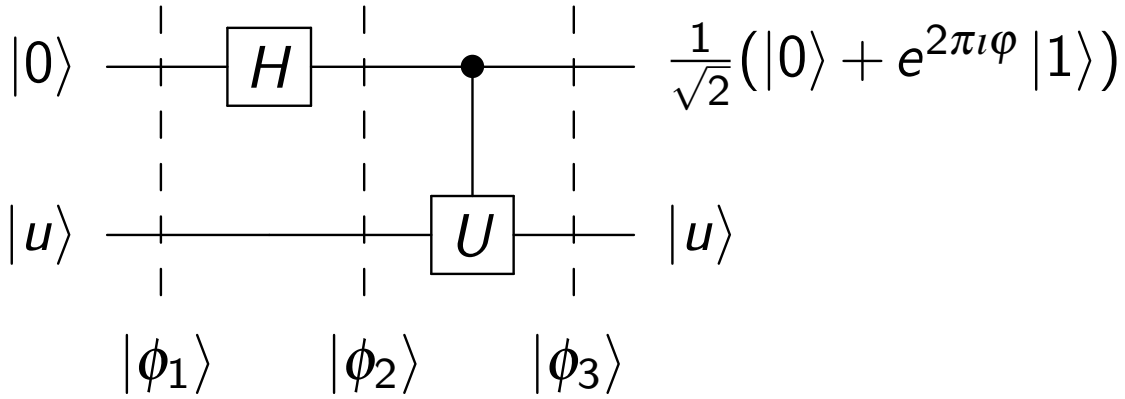
For an arbitrary state $|j\rangle$ with $j \in \{0,1\}$, performing controlled-U is

$$|j\rangle|u\rangle \rightarrow |j\rangle U^j|u\rangle$$

Phase Estimation

First Stage

Consider the following circuit



$$
\begin{aligned}
|\phi_1\rangle &= |0\rangle|u\rangle \\
|\phi_2\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |u\rangle \\
|\phi_3\rangle &= \frac{1}{\sqrt{2}}(|0\rangle|u\rangle + |1\rangle U|u\rangle) \\
&= \frac{1}{\sqrt{2}}(|0\rangle|u\rangle + e^{2\pi\imath\varphi}|1\rangle|u\rangle) \\
&= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi\imath\varphi}|1\rangle)|u\rangle
\end{aligned}
$$

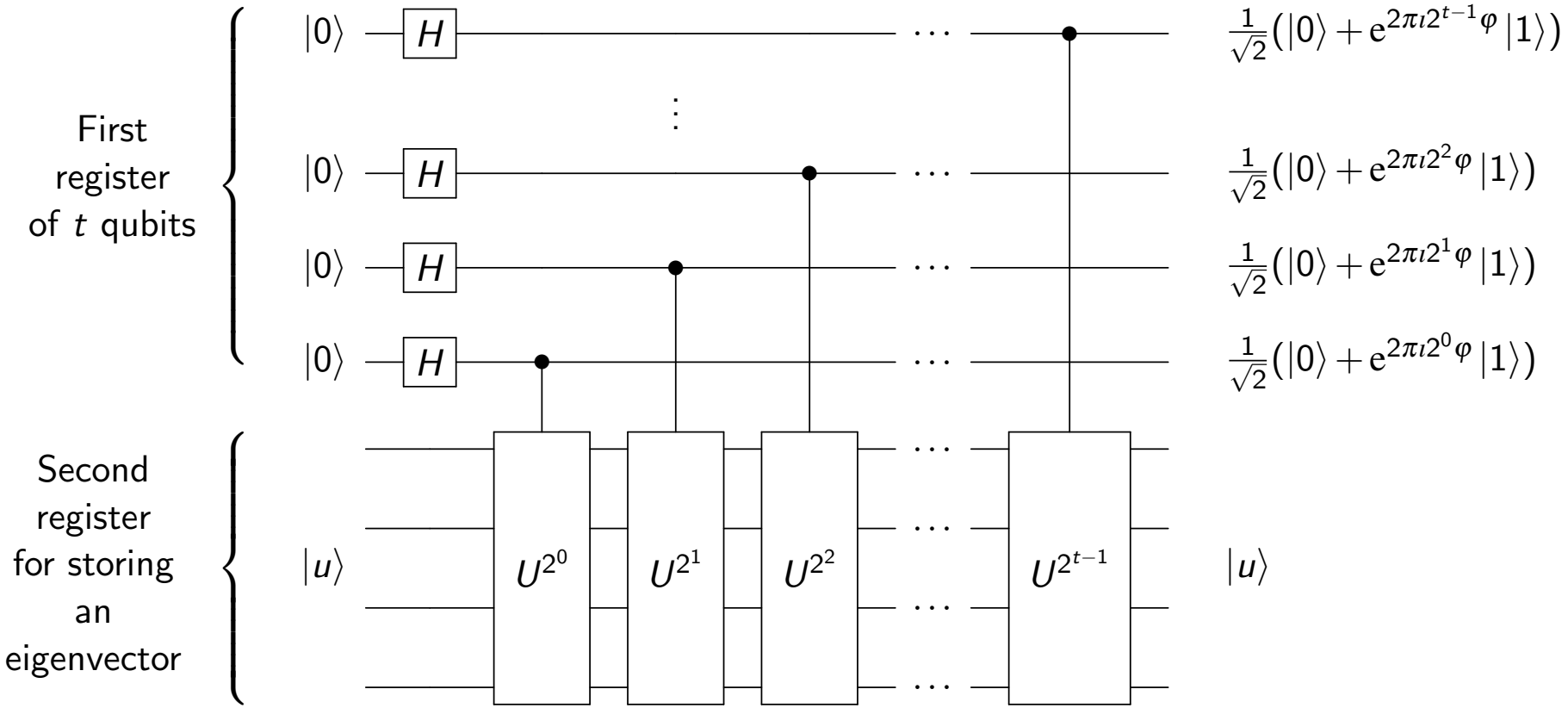
Phase Estimation

Question

Additional insight into the above circuit may be obtained by showing that the effect of the sequence of controlled-U operations is to take the state $|j\rangle |u\rangle$ to $|j\rangle U^j |u\rangle$. (Note that this does not depend on $|u\rangle$ being an eigenstate of U.)

Phase Estimation

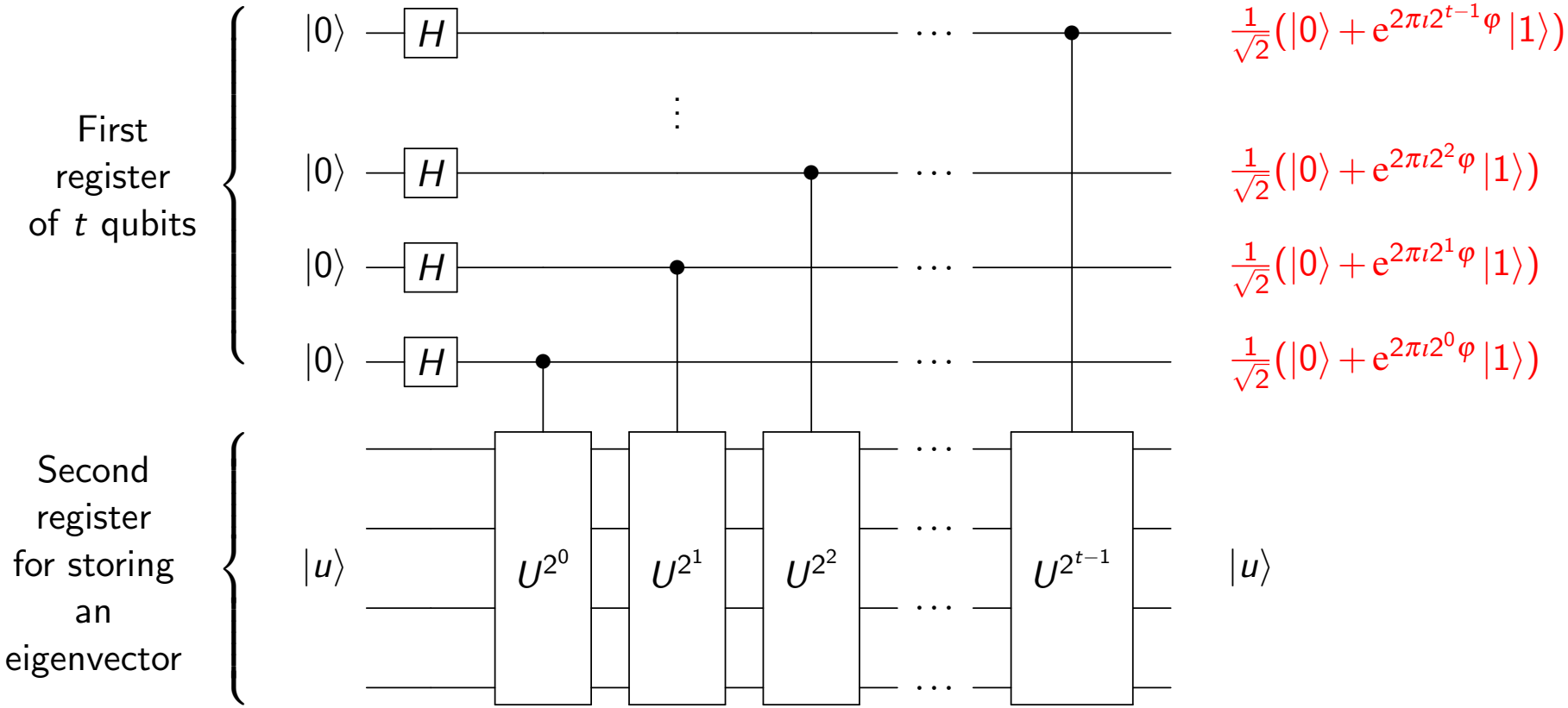
First Stage



- **First register**: contains t qubits initially in the state $|0\rangle$.
- **Second register**: contains as many qubits as necessary to store $|u\rangle$.

Phase Estimation

First Stage



$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi\imath 2^{t-1}\varphi}|1\rangle)(|0\rangle + e^{2\pi\imath 2^{t-2}\varphi}|1\rangle)\cdots(|0\rangle + e^{2\pi\imath 2^{0}\varphi}|1\rangle)$$

$$= \frac{1}{2^{t/2}}\sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi k}|k\rangle$$

Phase Estimation

Second Stage

Note that we want to estimate φ in the phase estimation ...

In first stage, we have already gotten $\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi k} |k\rangle$.

Question: how can we get φ from $\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi k} |k\rangle$?

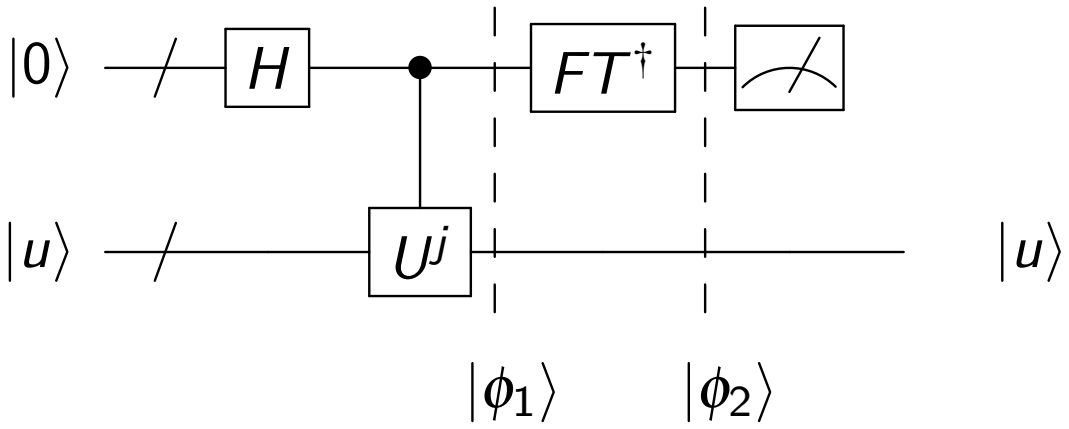
Recall the inverse Fourier transform:

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \mathrm{e}^{2\pi\imath jk/N} |k\rangle \xrightarrow{FT^\dagger} |j\rangle$$

Phase Estimation

Second Stage

We apply inverse Fourier transform to the output of first stage.



$$|\phi_1\rangle = \frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi\cdot 2^t k/2^t} |k\rangle |u\rangle ,$$

$$|\phi_2\rangle = |\widetilde{\varphi}\rangle |u\rangle ,$$

where $\widetilde{\varphi}$ denotes an integer as a good estimator for $\varphi \cdot 2^t$ when measured.

Phase Estimation

Suppose φ can be expressed exactly in t bits, as $\varphi = 0.\varphi_1 \ldots \varphi_t$ in binary.

Then the output of the first stage

$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi\imath 2^{t-1}\varphi}|1\rangle)(|0\rangle + e^{2\pi\imath 2^{t-2}\varphi}|1\rangle)\ldots(|0\rangle + e^{2\pi\imath 2^{0}\varphi}|1\rangle)$$

can be written as

$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi\imath 0.\varphi_t}|1\rangle)(|0\rangle + e^{2\pi\imath 0.\varphi_{t-1}\varphi_t}|1\rangle)\ldots(|0\rangle + e^{2\pi\imath 0.\varphi_1\cdots\varphi_t}|1\rangle)$$

After the second stage, we get $|\varphi_1 \cdots \varphi_t\rangle$ in the first register.

Question: what if φ cannot be expressed exactly in t bits?

Performance and Requirements

- Ideal case
  - φ **can** be written with a t-bit binary expansion and we can get the exact value φ.
  - inverse Fourier transform of phase estimation:

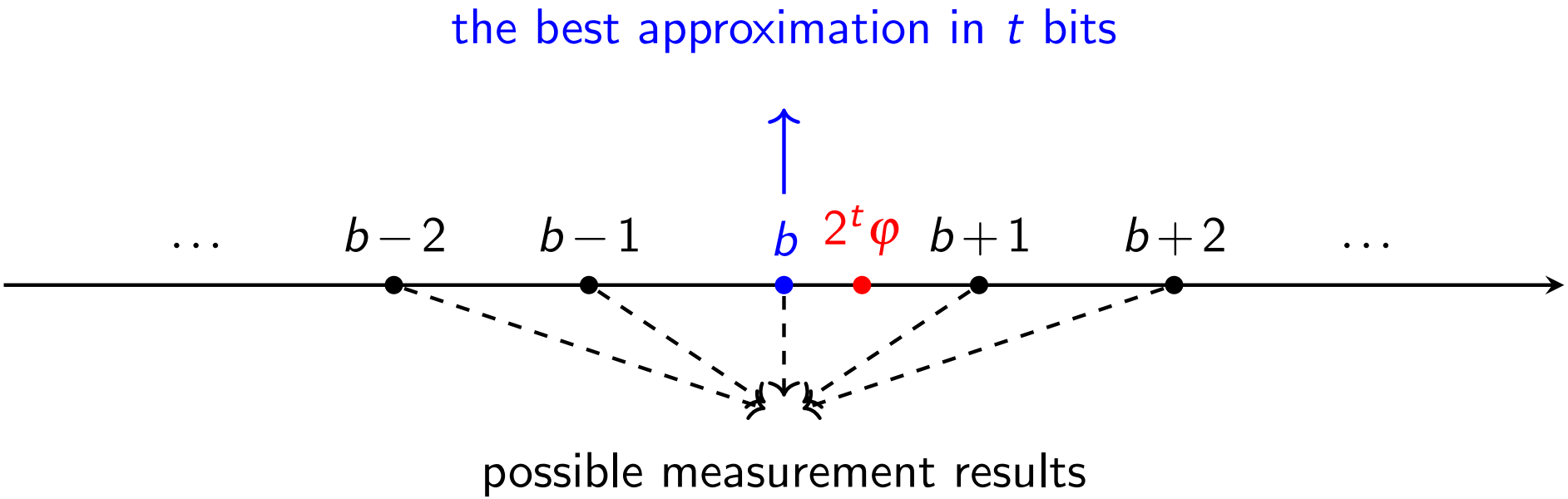
$$\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi k} |k\rangle \rightarrow \left|2^t \cdot \varphi\right\rangle$$

- Real case
  - φ **cannot** be written with a t-bit binary expansion but we can get the best t-bit approximation to φ.
  - inverse Fourier transform of phase estimation:

$$\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi k} |k\rangle \rightarrow \frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi k} \sum_{j=0}^{2^t-1} \frac{1}{2^{t/2}} e^{-2\pi\imath jk/2^t} |j\rangle$$

$$= \frac{1}{2^t} \sum_{k,j=0}^{2^t-1} e^{2\pi\imath(\varphi - j/2^t)k} |j\rangle$$

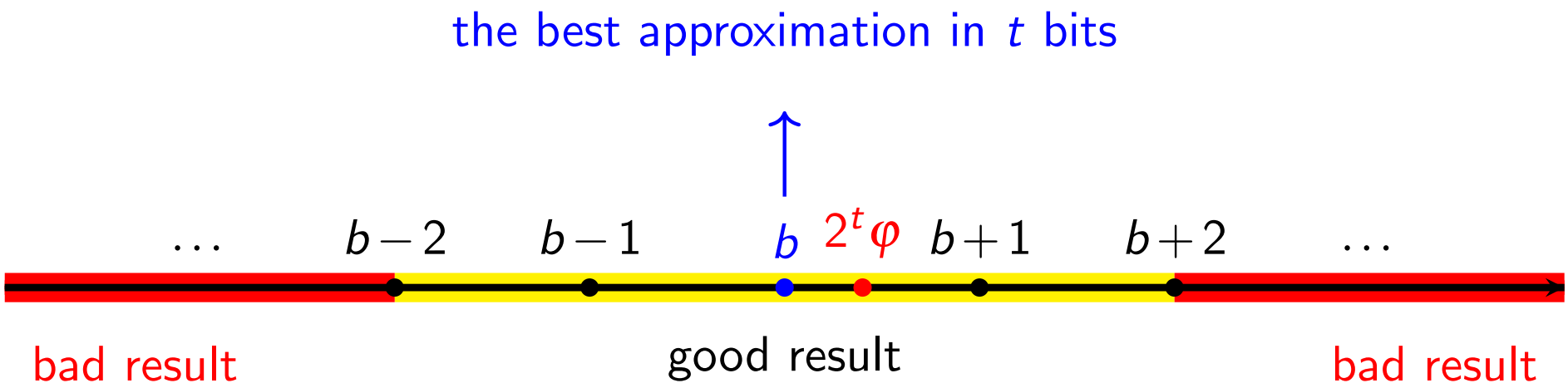
Performance and Requirements



- b is the integer in the range 0 to 2^t-1 such that $b/2^t = 0.b_1\cdots b_t$ is the best t-bit approximation to φ which is not greater than φ. Thus $0 \le 2^t\varphi - b < 1$, i.e. $\delta = \varphi - b/2^t \in [0, 2^{-t})$.
- The possible measurement result is $|(b+\ell) \bmod 2^t\rangle$ as can be expressed in t bits expansion and $\ell \in (-2^{t-1}, 2^{t-1}]$.

Question: why $\ell \in (-2^{t-1}, 2^{t-1}]$?

Performance and Requirements



Given an error bound e, we get

- a good result $b \pm \ell$ satisfying $\ell \leq e$,
- a bad result $b \pm \ell$ satisfying $\ell > e$.

Our goal: Find the lower bound of the probability of getting the good results, i.e., the upper bound of the probability of bad results!

Performance and Requirements

Recall the inverse Fourier transform in real case:

$$\frac{1}{2^{t/2}}\sum_{k=0}^{2^t-1} e^{2\pi\imath\varphi k}\,|k\rangle \to \frac{1}{2^t}\sum_{k,j=0}^{2^t-1} e^{2\pi\imath(\varphi-j/2^t)k}\,|j\rangle$$

The probability of taking state $|j\rangle$ is

$$\Pr(\{|j\rangle\}) = \left|\frac{1}{2^t}\sum_{k=0}^{2^t-1} e^{2\pi\imath(\varphi-j/2^t)k}\right|^2 = \left|\frac{1}{2^t}\frac{1-e^{2\pi\imath(2^t\varphi-j)}}{1-e^{2\pi\imath(\varphi-j/2^t)}}\right|^2.$$

Performance and Requirements

We define

$$\alpha_\ell \equiv \frac{1}{2^t}\frac{1-\mathrm{e}^{2\pi\imath(2^t\varphi-(b+\ell))}}{1-\mathrm{e}^{2\pi\imath(\varphi-(b+\ell)/2^t)}} = \frac{1}{2^t}\frac{1-\mathrm{e}^{2\pi\imath(2^t\delta-\ell)}}{1-\mathrm{e}^{2\pi\imath(\delta-\ell/2^t)}},$$

thus $\Pr(\{|(b+\ell) \bmod 2^t\rangle\}) = |\alpha_\ell|^2$.

The probability $\Pr(\{\text{bad}\})$ of bad results are

$$\sum_{-2^{t-1}<\ell\leq(-e-1)} |\alpha_\ell|^2 + \sum_{e+1\leq\ell\leq 2^{t-1}} |\alpha_\ell|^2.$$

Performance and Requirements

$$\alpha_\ell = \frac{1}{2^t}\frac{1-e^{2\pi\iota(2^t\delta-\ell)}}{1-e^{2\pi\iota(\delta-\ell/2^t)}}$$

$$\Downarrow \quad |\exp(\iota\theta)| = 1$$

$$\Rightarrow \quad |\alpha_\ell| \leq \frac{1}{2^t}\frac{2}{|1-e^{2\pi\iota(\delta-\ell/2^t)}|}$$

$$\Downarrow \quad |1-\exp(\iota\theta)| \geq 2|\theta|/\pi \text{ for } \theta \in [-\pi,\pi]$$

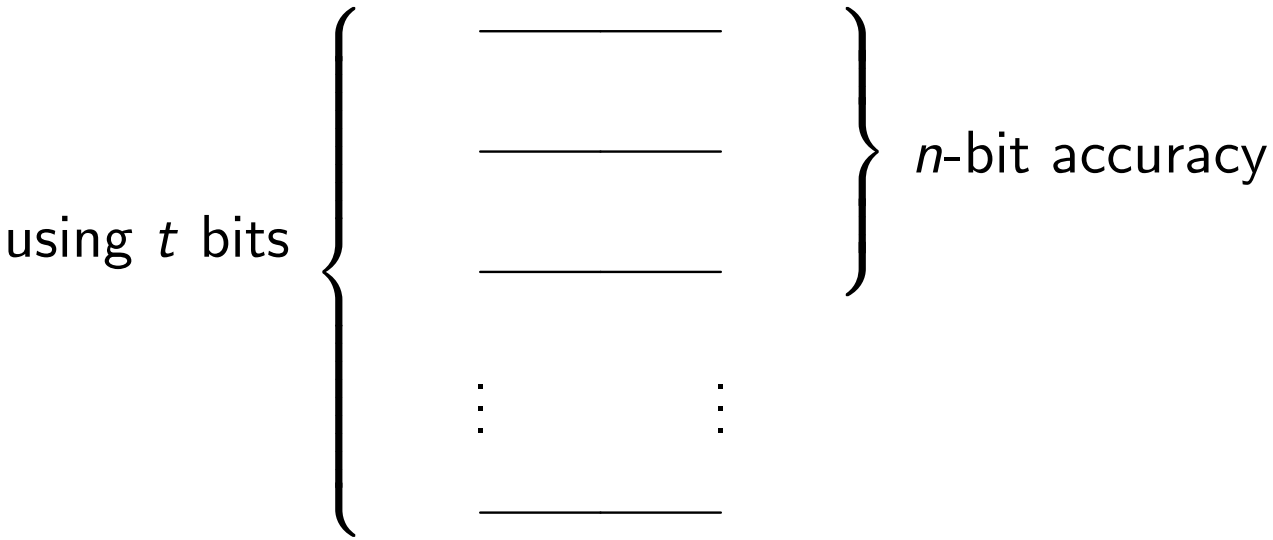
$$\Rightarrow \quad |\alpha_\ell| \leq \frac{1}{2^{t+1}|\delta-\ell/2^t|}$$

Performance and Requirements

$$
\begin{aligned}
\Pr(\{\text{bad}\}) &\leq \frac{1}{4}\left[\sum_{\ell=-2^{t-1}+1}^{-(e+1)} \frac{1}{(\ell-2^t\delta)^2} + \sum_{\ell=e+1}^{2^{t-1}} \frac{1}{(\ell-2^t\delta)^2}\right] \\
&\leq \frac{1}{4}\left[\sum_{\ell=-2^{t-1}+1}^{-(e+1)} \frac{1}{\ell^2} + \sum_{\ell=e+1}^{2^{t-1}} \frac{1}{(\ell-1)^2}\right] \\
&= \frac{1}{4}\left[\sum_{\ell=e+1}^{2^{t-1}-1} \frac{1}{\ell^2} + \sum_{\ell=e}^{2^{t-1}-1} \frac{1}{\ell^2}\right] \\
&\leq \frac{1}{4}\left[2\cdot \sum_{\ell=e}^{2^{t-1}-1} \frac{1}{\ell^2}\right] \\
&\leq \frac{1}{2}\cdot \int_{\ell=e-1}^{2^{t-1}-1} \frac{\mathrm{d}\ell}{\ell^2} \\
&\leq \frac{1}{2}\cdot \int_{\ell=e-1}^{+\infty} \frac{\mathrm{d}\ell}{\ell^2} \leq \frac{1}{2(e-1)}
\end{aligned}
$$

Performance and Requirements

Question: How many qubits do we need to get 2^{-n} accuracy with a probability of at least $1-\varepsilon$, i.e., the probability of bad results is $\leq \varepsilon$?



We set the error bound as $e = 2^{t-n} - 1$.
As $\Pr(\{\text{bad}\}) \leq \frac{1}{2(e-1)}$, $\frac{1}{2(e-1)} \leq \varepsilon$ implies $\Pr(\{\text{bad}\}) \leq \varepsilon$. So we choose

$$t \geq n + \lceil \log(2 + \tfrac{1}{2\varepsilon}) \rceil.$$

Algorithm: Quantum Phase Estimation

Input:
1. a register of $t = n + \lceil \log(2 + \frac{1}{2\varepsilon}) \rceil$ qubits;
2. a register for storing an eigenstate $|u\rangle$ of U with eigenvalue $e^{2\pi\imath\varphi_u}$;
3. a series of black boxes which perform controlled-U^{2^j} operations for integer j.

Output: an n-bit approximation $\widetilde{\varphi_u}$ (i.e. error within ± 1) to $\varphi_u \cdot 2^t$.

1: $|0\rangle^{\otimes t} |u\rangle$ ▷ initial state

2: $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle |u\rangle$ ▷ create superposition

3: $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} |k\rangle U^k |u\rangle$ ▷ apply black box

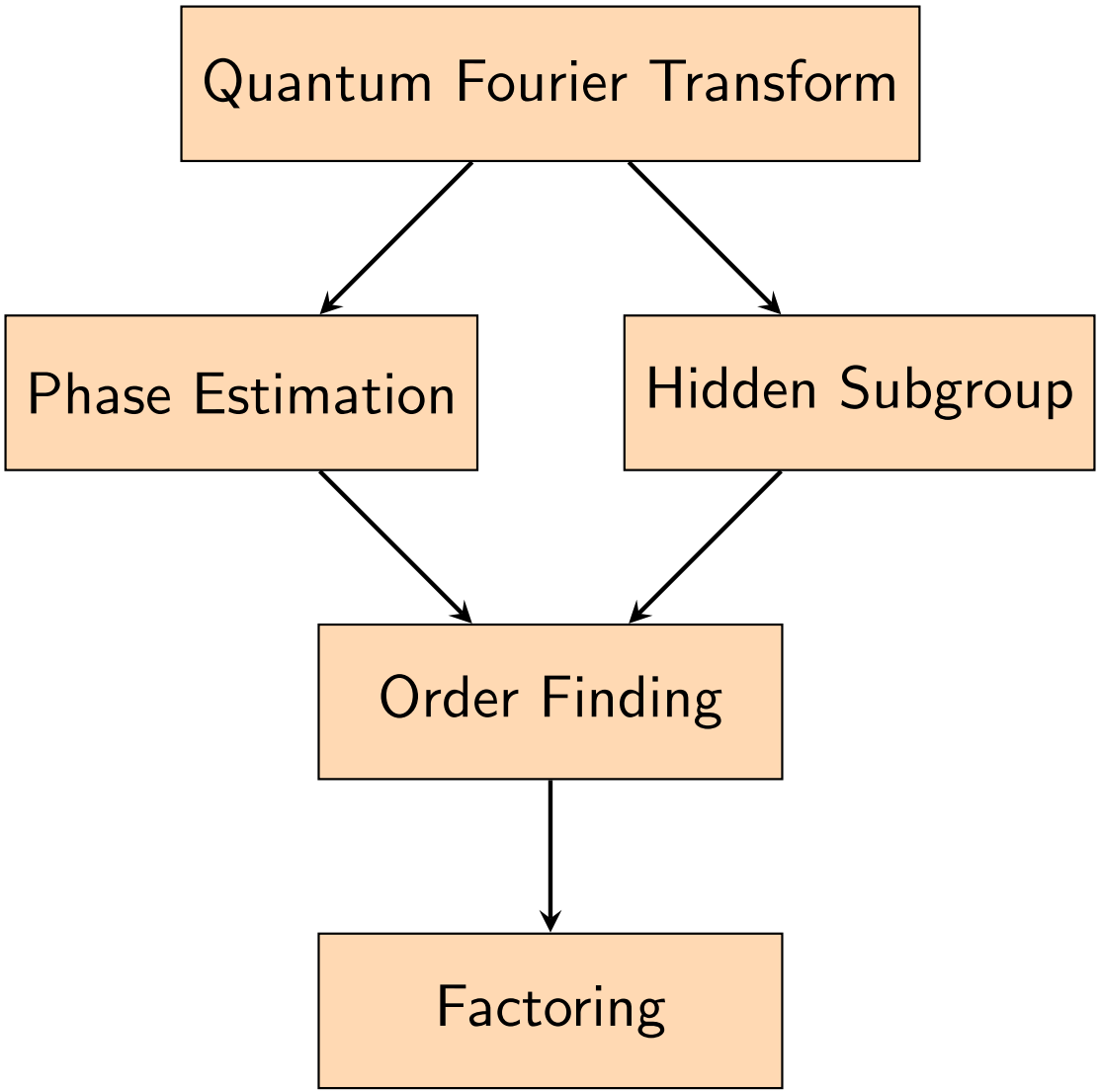
$= \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} e^{2\pi\imath k\varphi_u} |k\rangle |u\rangle$ ▷ result of black box

4: $\rightarrow \left|\widetilde{\varphi_u}\right\rangle |u\rangle$ ▷ apply inverse Fourier transform

5: $\rightarrow \widetilde{\varphi_u}$ ▷ measure first register

Complexity: $O(t^2)$ operations and one call to controlled-U^{2^j} black box. Succeed with probability at least $1 - \varepsilon$.

Outline of Shor Algorithm



Order Finding

Order Finding

Order

For positive integers x and N, $x < N$, with no common factor, the order of x modulo N is the least positive integer r such that

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

For example, the order of $x = 5$ modulo $N = 33$ is 10.

$$5^1 \bmod 33 = 5 \qquad 5^6 \bmod 33 = 16$$
$$5^2 \bmod 33 = 25 \qquad 5^7 \bmod 33 = 14$$
$$5^3 \bmod 33 = 26 \qquad 5^8 \bmod 33 = 4$$
$$5^4 \bmod 33 = 31 \qquad 5^9 \bmod 33 = 20$$
$$5^5 \bmod 33 = 23 \qquad 5^{10} \bmod 33 = 1$$

Order Finding

Order

For positive integers x and N, $x < N$, with no common factor, the order of x modulo N is the least positive integer r such that

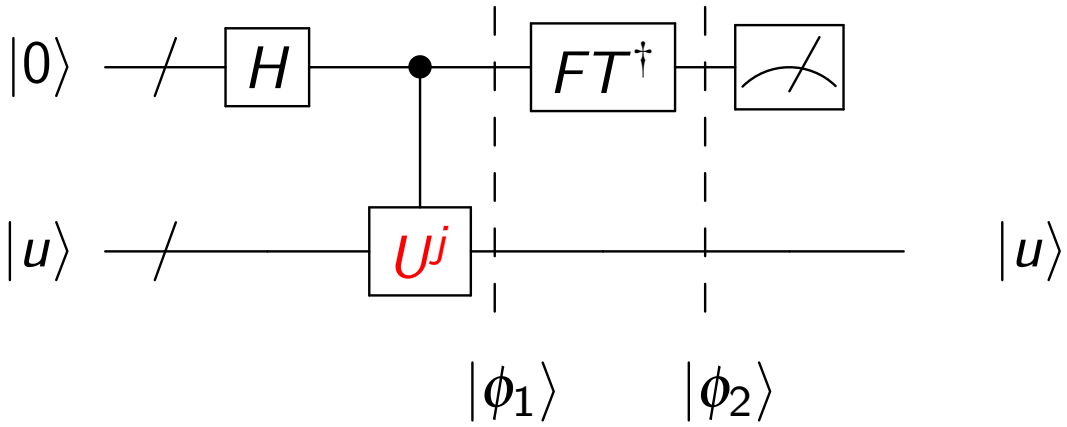
$$x^r \equiv 1 \mod N$$

Question: given x and N, how to find the order of x modulo N in an efficient way? Expected to be polynomial-time w.r.t. $n = \log(N)$.

We will use phase estimation to solve the problem!

Order Finding

Recall the phase estimation



The point is how to design the unitary operator.

Here we design the unitary operator as

$$U|y\rangle \equiv |xy \bmod N\rangle$$

with $y \in \{0,1\}^n$ where $n = \lceil \log_2(N) \rceil$.

Order Finding

There are two important requirements for us to be able to use the phase estimation procedure:

- implement a controlled-U^{2^j} operation for any integer j.
  - modular exponentiation
- prepare an eigenstate $|u_s\rangle$ with a nontrivial eigenvalue.

Order Finding

Modular Exponentiation

Question: how can we compute the controlled-U^{2^j} operations?

That is, we wish to compute the transformation:

$$\begin{aligned}|z\rangle|y\rangle \to |z\rangle U^z|y\rangle &= |z\rangle U^{z_t 2^{t-1}} \cdots U^{z_1 2^0}|y\rangle \\ &= |z\rangle \left| x^{z_t 2^{t-1}} \times \cdots \times x^{z_1 2^0} \bmod N \right\rangle \\ &= |z\rangle |x^z y \bmod N\rangle,\end{aligned}$$

where $z = z_t \cdots z_1$ in binary.

- The first stage: use modular multiplication to compute individual $x^{2^j} \bmod N$, like $y_1 = x^2 \bmod N$, $y_2 = y_1^2 \bmod N = x^4 \bmod N$, $\ldots$, $y_j = y_{j-1}^2 \bmod N = x^{2^j} \bmod N$.
- The second stage:

$$x^z \bmod N = [(x^{z_t 2^{t-1}} \bmod N) \cdots (x^{z_1 2^0} \bmod N)] \bmod N.$$

The multiplication on n-bit numbers costs $\mathcal{O}(n^2)$ bit operations, and totally costs $\mathcal{O}(n^3)$.

Order Finding

Eigenstate Preparation

Define the following state

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi\imath sk}{r}\right] \left|x^k \bmod N\right\rangle,$$

for $0 \leq s < r$, we could find the above state is the eigenstates of U, since

$$\begin{aligned} U|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi\imath sk}{r}\right] \left|x^{k+1} \bmod N\right\rangle \\ &= \exp\left[\frac{2\pi\imath s}{r}\right] |u_s\rangle. \end{aligned}$$

We could obtain the corresponding eigenvalues $\exp(2\pi\imath s/r)$ and the phases s/r (a fraction) using the phase estimation procedure, from which we could further obtain the order r (the denominator of that fraction) with a little bit more work.

Order Finding

Eigenstate Preparation

Recall the eigenstate $|u_s\rangle$:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi\imath sk}{r}\right] \left|x^k \bmod N\right\rangle.$$

Conflict here!

To prepare above state, we must know r at first. However the problem is to find r!

Order Finding

Eigenstate Preparation

Recall the eigenstate $|u_s\rangle$:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi\imath sk}{r}\right] \left|x^k \bmod N\right\rangle.$$

Conflict here!

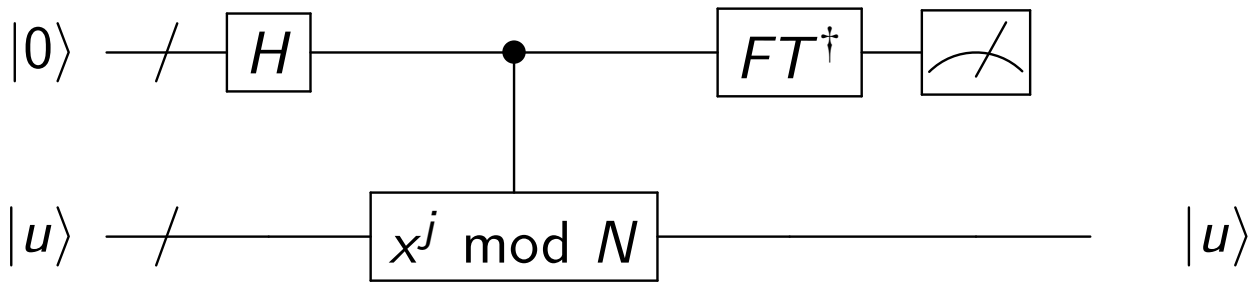
To prepare above state, we must know r at first. However the problem is to find r!

There is a trick:

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

We can get every eigenstate $|u_s\rangle$ with the probability of $\frac{1}{r}$.

Order Finding



- Register 1 use $t = 2n+1+\lceil \log(2+\frac{1}{2\varepsilon}) \rceil$ bits. (Recall n is the number of bits needed to specify N)
- Register 2 is prepared in $|1\rangle$.
- It follows that for each s in range 0 through $r-1$, we will obtain an estimate of the phase $\varphi = s/r$ accurate to $2n+1$ bits, with error probability at most ε/r.

Order Finding

The Continued Fraction Expansion

Question: how to get r from s/r?

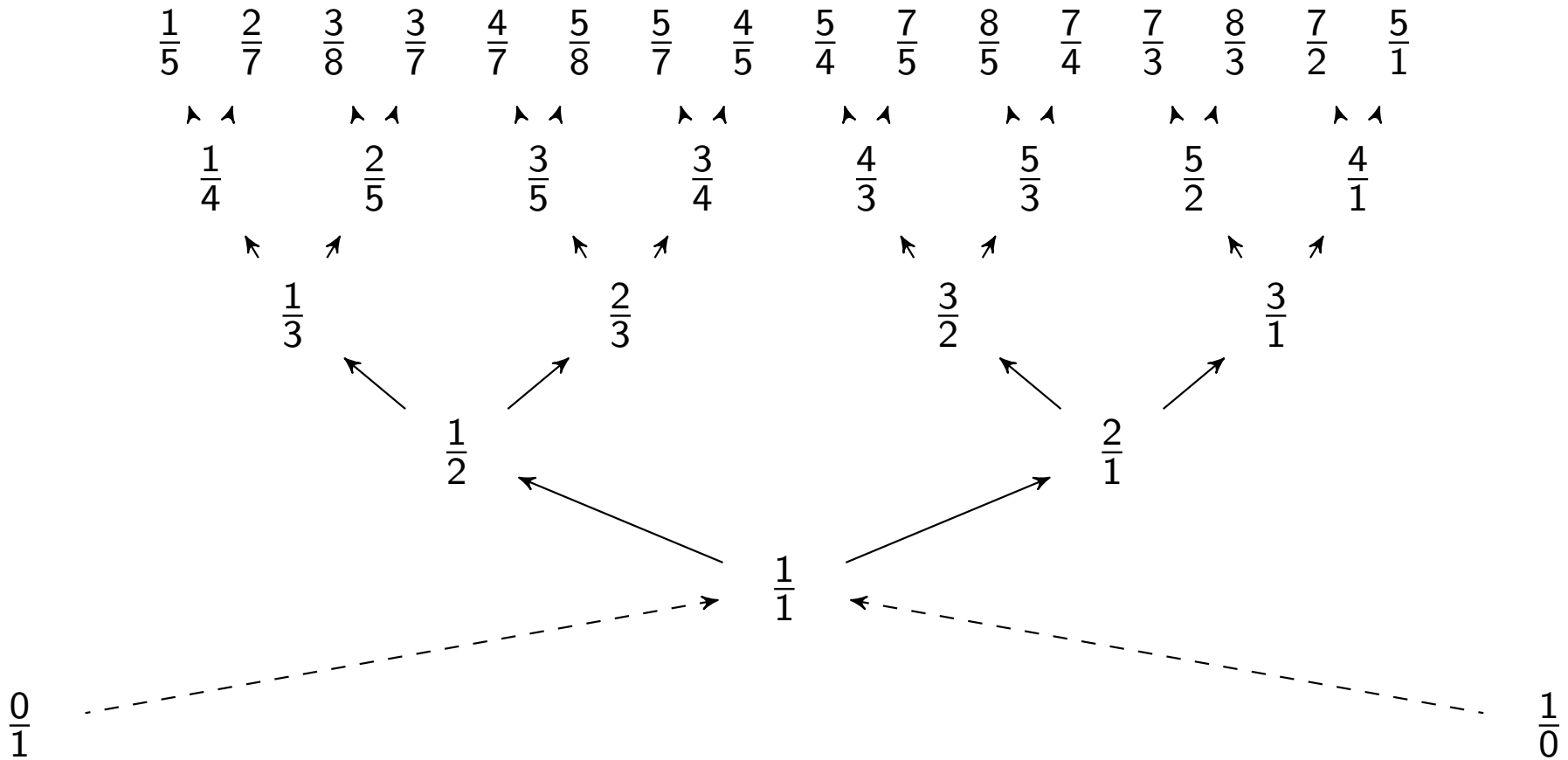


Figure: Bottom levels of the Stern–Brocot tree

Order Finding

Summary

Input:
1. A black box $U_{x,N}$ which performs the transformation $|j\rangle|k\rangle \to |j\rangle\left|x^j k \bmod N\right\rangle$, for x co-prime to the n-bit number N;
2. $t = 2n+1+1\lceil\log(2+\frac{1}{2\varepsilon})\rceil$ qubits initialized to $|0\rangle$;
3. n qubits initialized to the state $|1\rangle$.

Output: The least integer $r > 0$ such that $x^r \equiv 1 \mod N$.

1: $|0\rangle^{\otimes t}|1\rangle$ ▷ initial state

2: $\to \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle|1\rangle$ ▷ create superposition

3: $\to \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle\left|x^j \bmod N\right\rangle$ ▷ apply $U_{x,N}$

$= \frac{1}{\sqrt{r2^t}}\sum_{s=0}^{r-1}\sum_{j=0}^{2^t-1} e^{2\pi\imath sj/r}|j\rangle|u_s\rangle$

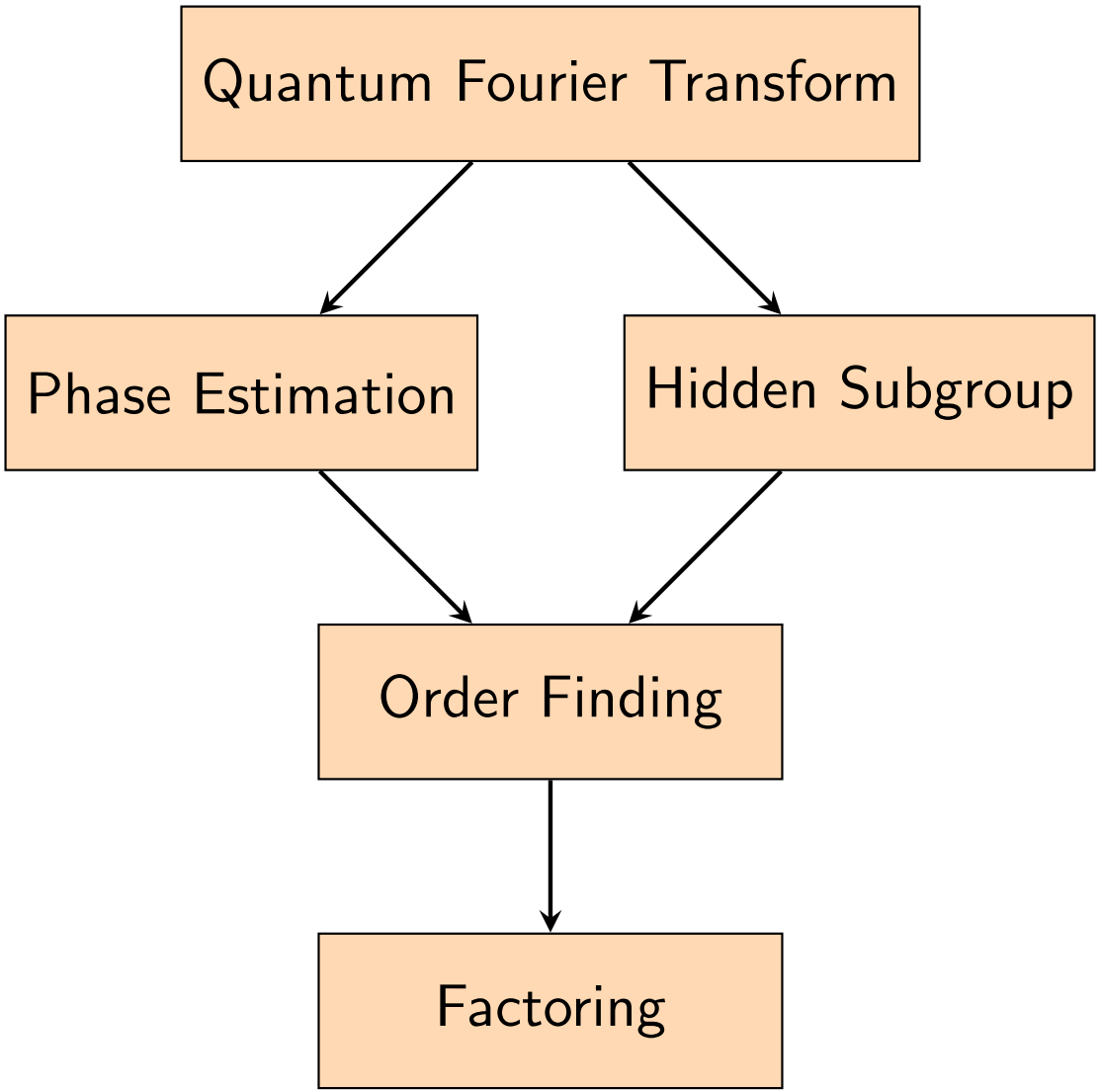
4: $\to \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\left|\widetilde{s/r}\right\rangle|u_s\rangle$ ▷ apply inverse Fourier transform

5: $\to \widetilde{s/r}$ ▷ measure first register

6: r ▷ apply continued fraction method

Complexity: $O(n^3)$ operations. Succeeds with probability $O(1)$.

Outline of Shor Algorithm



Factoring

Factoring

Recall: RSA relies on factors and large prime numbers.

An exciting result! Factoring $\Rightarrow$ Order finding

How?

- Randomly choose x in the range 3 to $N-1$.
- Find order r of x module N:

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

- If r is even and $x^{r/2} \not\equiv -1 \mod N$, then each of $\gcd(x^{r/2}-1, N)$ and $\gcd(x^{r/2}+1, N)$ is a non-trivial factor of N.
- If r is odd, repeat the above procedure until we find the answer.

Factoring

Example

Continued to the last example, 10 is the order of 5 modulo 33. We have $5^{10}-1=(5^5-1)\times(5^5+1)\equiv 22\times 24 \bmod 33$, where $\gcd(22,33)=11$ and $\gcd(24,33)=3$ are non-trivial factors of 33.

Recall

$$5^1 \bmod 33 = 5 \qquad 5^6 \bmod 33 = 16$$
$$5^2 \bmod 33 = 25 \qquad 5^7 \bmod 33 = 14$$
$$5^3 \bmod 33 = 26 \qquad 5^8 \bmod 33 = 4$$
$$5^4 \bmod 33 = 31 \qquad 5^9 \bmod 33 = 20$$
$$5^5 \bmod 33 = 23 \qquad 5^{10} \bmod 33 = 1$$

Factoring

What is the probability of success?

Theorem

Suppose $N = p_1^{\alpha_1} \dots p_m^{\alpha_m}$ *is the prime factorization of an odd composite positive integer. Let* x *be an integer chosen uniformly at random, subject to the requirements that* $1 \leq x \leq N-1$ *and* x *is co-prime to* N*. Let* r *be the order of* x *modulo* N*. Then*

$$\Pr(\{r \text{ is even and } x^{r/2} \not\equiv -1 \ mod \ N\}) \geq 1 - \frac{1}{2^m}.$$

Factoring

Summary

Input: 1. A composite number N.

Output: A non-trivial factor of N.

1: If N is even, return the factor 2.

2: Determine whether $N = a^b$ for some integers $a, b \geq 2$, and if so return the factor a.

3: Randomly choose x in the range 3 to $N-1$. If $\gcd(x, N) > 1$ then return the factor $\gcd(x, N)$.

4: Use the order-finding subroutine to find the order r of x modular N.

5: If r is even and $x^{r/2} \not\equiv -1 \bmod N$ then compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$, and test to see if one of these is a non-trivial factor, returning that factor if so. Otherwise, the algorithm fails.

Complexity: $\mathcal{O}((\log N)^3)$ operations. Succeeds with probability $\mathcal{O}(1)$.

Factoring

Homework

Suppose N is n bits long. The aim of this exercise is to find an efficient classical algorithm to determine whether $N = a^b$ for some integers $a, b \geq 2$.

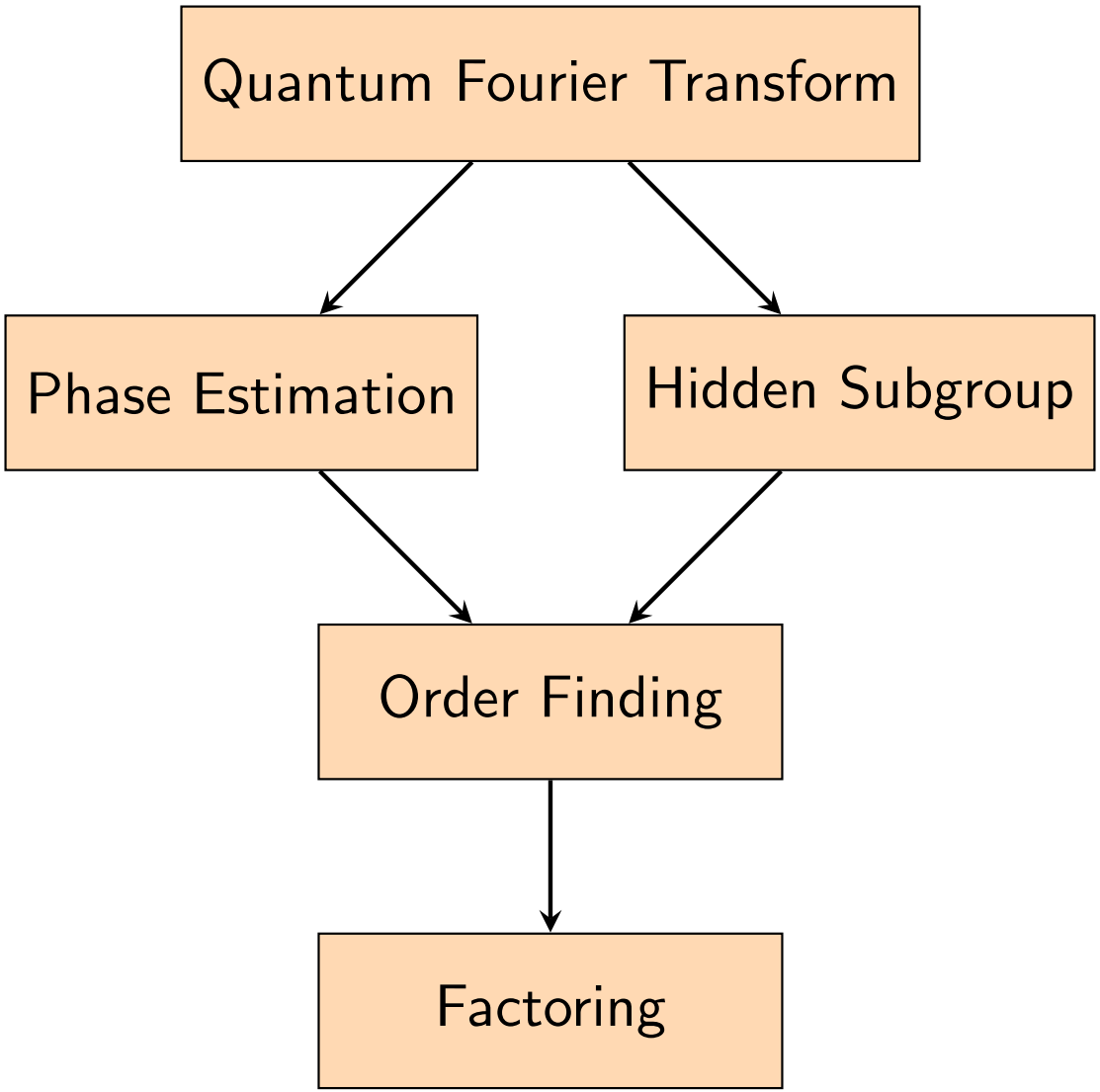
Factoring

Homework

Suppose N is n bits long. The aim of this exercise is to find an efficient classical algorithm to determine whether $N = a^b$ for some integers $a, b \geq 2$.

1. Show that b, if it exists, satisfies $b \leq n$.
2. Show that it takes at most $\mathcal{O}(n^2)$ operations to compute $y = \log_2 N$, $x = y/b$ for a fixed b, and the two integers u_1 and u_2 nearest to 2^x.
3. Show that it takes at most $\mathcal{O}(n^2)$ operations to compute u_1^b and u_2^b (use repeated squaring) and check to see if either is equal to N.
4. Combine the previous results to give an $\mathcal{O}(n^3)$ operation algorithm to determine whether $N = a^b$ for some integers a and b.

Outline of Shor Algorithm



Hidden Subgroup Problem

Group and Subgroup

A group $(G, \circ)$ is a set G of elements equipped with a binary operation $\circ : G \times G \to G$, called group operation on G, satisfying:

- $(g_1 \circ g_2) \circ g_3 = g_1 \circ g_2 \circ g_3 = g_1 \circ (g_2 \circ g_3)$ is associative for all $g_1, g_2, g_3 \in G$,
- there is an identity element $e \in G$ such that $g \circ e = g = e \circ g$ holds for all $g \in G$,
- for each $g \in G$, there is an inverse element $g^{-1} \in G$ such that $g \circ g^{-1} = e = g^{-1} \circ g$.

Usually, $g_1 g_2$ is used to be short for $g_1 \circ g_2$.

A subgroup $(H, \circ)$ is a subset $H \subseteq G$ equipped with the same group operation $\circ$ on H.

Abelian Group and Generator

In the following, we only consider the Abelian (commutative) group $(G, \circ)$ that meets $g_1 \circ g_2 = g_2 \circ g_1$ for all $g_1, g_2 \in G$.

Let T be a subset of G, and $\langle T \rangle$ denote the subgroup of G generated by elements of T, i.e. adding composite $g_1 g_2$ and inverse g^{-1}.

Then T is called the generator of subgroup $\langle T \rangle$. In other words, it suffices to determine a group by its generator.

Partition Induced by a Subgroup

Given a subgroup H of G, $H \neq G$, how can we classify the remaining elements?

A standard way is employing the coset, which is of the form $gH = \{gh \,|\, h \in H\}$ for some $g \notin H$. It is not hard to see

- $|H| = |gH|$ holds for any $g \notin H$,
- two cosets $g_1 H$ and $g_2 H$ are either identical or disjoint.

Then we obtain the partition

$$G = \biguplus_{g \in G} gH,$$

i.e. classfying all elemenets $G \setminus H$ into equally-sized cosets gH.

Hidden Subgroup Problem

Given a group G and a function $f : G \to S$ where S is some finite set.

Suppose f has the property that there exists a subgroup $H \subseteq G$ such that f is constant within H and each coset gH, and distinct on them: $f(g_1) = f(g_2)$ iff $g_1 H = g_2 H$.

Goal: find the hidden subgroup H.

Reduction from the Order Finding Problem

Given positive integers x and N, $x < N$, with no common factor, we are particularly interested in the sequence of positive integers

$$x^0 \mod N,\ x^1 \mod N,\ x^2 \mod N,\ \ldots,\ x^{\varphi(x)} \mod N,\ \ldots$$

where $\varphi(x)$ is the Euler function of x.

Thanks to Euler theorem $x^{\varphi(x)} \equiv 1 \mod N$, it suffices to focus on the fragment

$$\mathit{Frag} = \{x^0 \mod N,\ x^1 \mod N,\ x^2 \mod N,\ \ldots,\ x^{\varphi(x)-1} \mod N\}.$$

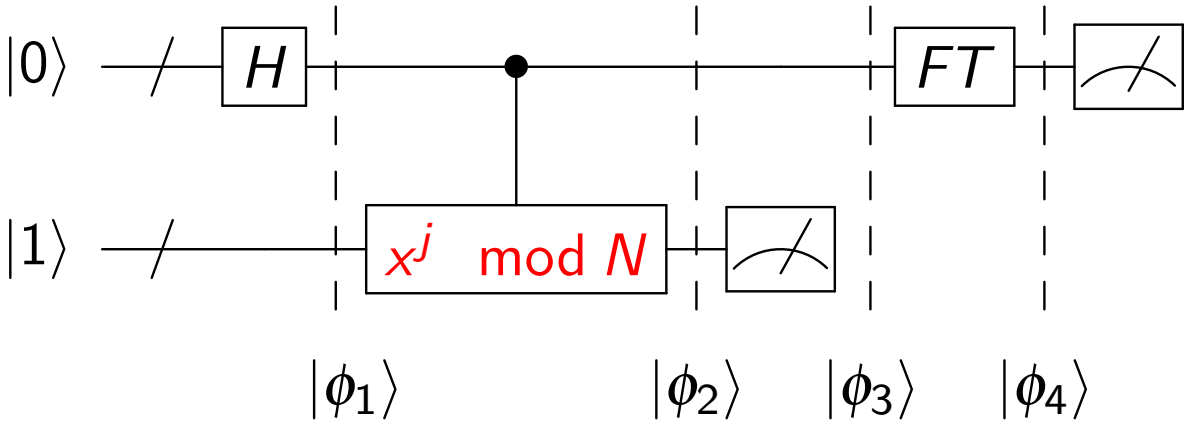
Reduction from the Order Finding Problem

Group: We take the group $G = \{0, 1, 2, \ldots, \varphi(x) - 1\}$ equipped with operation $g_1 \circ g_2 = g_1 + g_2 \mod \varphi(x)$.

Hidden Subgroup: Under the function $f(a) = x^a \mod N$, we could get $f(G) = Frag$. There is a hidden subgroup $H = \{0, r, 2r, \ldots, \varphi(x) - r\}$, such that $f(H) = \{1\}$ and $f(aH) = \{f(a)\}$.

Once H is computed, the order finding problem is solved as the generator r of $H = \langle r \rangle$.

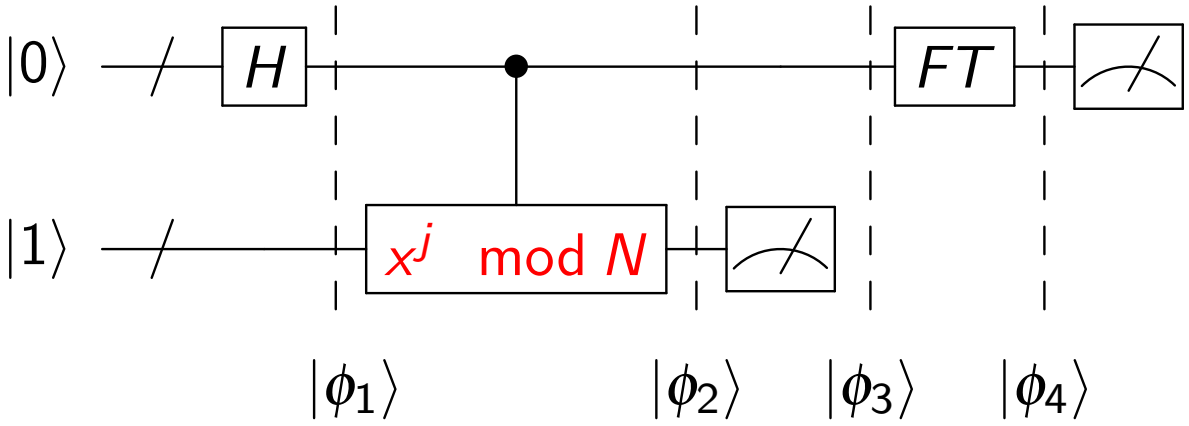
Order Finding



Register 1 has t qubits, $2^t \geq N \geq \varphi(N)$, to give information about r.
Range $[0, 2^t - 1]$ covers the group G.

Register 2 has n qubits to store the integers less than $N \leq 2^n$.

Order Finding



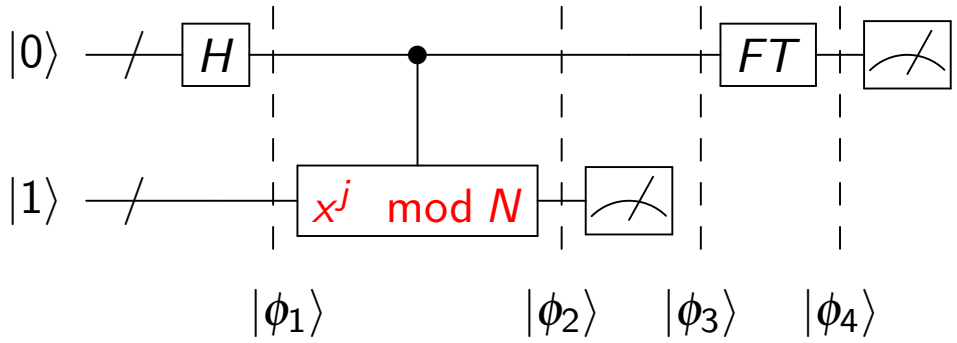
The intermediate states would be

$$|\phi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |1\rangle ,$$

$$|\phi_2\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle \left| x^j \mod N \right\rangle$$

$$= \frac{1}{\sqrt{2^t}} \sum_{i=0}^{r-1} \left[\sum_{j \mod r = i} |j\rangle \right] \left| x^i \mod N \right\rangle .$$

Order Finding



After measuring Register 2, the outcome would be one of

$$x^0 \mod N,\ x^1 \mod N,\ \ldots,\ x^{r-1} \mod N.$$

If it is $x^j \mod N$, Register 1 would be the superposition over a coset

$$|\phi_3\rangle = \frac{1}{\sqrt{m}} \left[\sum_{j \mod r = i} |j\rangle \right],$$

where $m = \lceil \frac{2^t}{r} \rceil$ or $\lfloor \frac{2^t}{r} \rfloor$, depending on $i \leq 2^t - r\lfloor \frac{2^t}{r} \rfloor$ or not.

A Shortcut Treatment

At present, we get the superposition $\frac{1}{\sqrt{m}}[|i\rangle + |i+r\rangle + |i+2r\rangle + \cdots]$.

Repeatedly measuring it, we would get a series of outcomes

$$c_0 = i + m_0 r,\ c_1 = i + m_1 r,\ c_2 = i + m_2 r,\ \ldots$$

The order r seems to be $\gcd(\{c_k - c_0\})$, thus completes the procedure?!

A Shortcut Treatment

At present, we get the superposition $\frac{1}{\sqrt{m}}[|i\rangle + |i+r\rangle + |i+2r\rangle + \cdots]$.

Repeatedly measuring it, we would get a series of outcomes

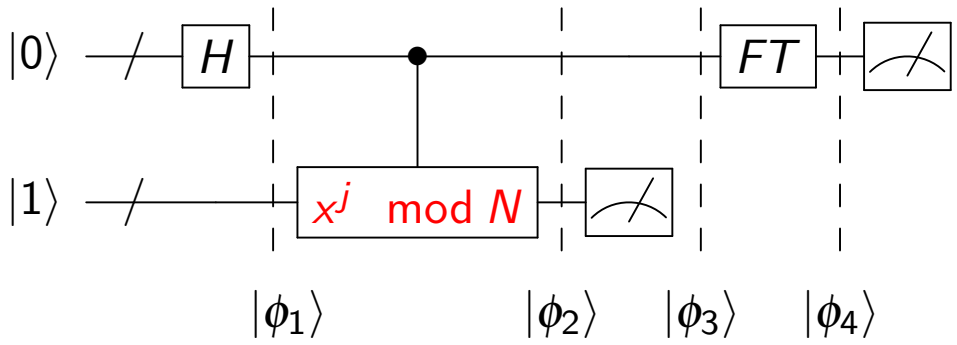
$$c_0 = i + m_0 r,\ c_1 = i + m_1 r,\ c_2 = i + m_2 r,\ \ldots$$

The order r seems to be $\gcd(\{c_k - c_0\})$, thus completes the procedure?!

Actually, the outcomes $c_k = i + m_k r$ depends on the measure on Register 2. We cannot fix the outcome i of that measure.

To overcome it, we should device a procedure always telling something about r, no matter what the outcome i of the measure on Register 2.

Ideal Case: $r \mid 2^t$

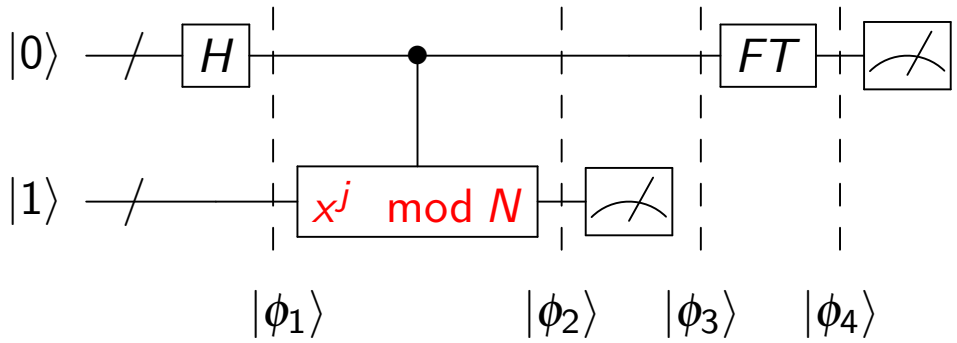


Then, by $|\phi_3\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |i + jr\rangle$ and $2^t = mr$, we have

$$
\begin{aligned}
|\phi_4\rangle &= \frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^t}} \sum_{j=0}^{m-1} \sum_{k=0}^{2^t-1} \mathrm{e}^{2\pi\imath(i+jr)k/2^t} |k\rangle \\
&= \frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^t}} \sum_{m|k} \sum_{j=0}^{m-1} \mathrm{e}^{2\pi\imath(i+jr)k/2^t} |k\rangle .
\end{aligned}
$$

The last equation comes from the fact: If $m \nmid k$, rk is not a multiple of 2^t, and then the sum $\sum_{j=0}^{m-1} \mathrm{e}^{2\pi\imath(i+jr)k/2^t}$ can be cancelled.

Ideal Case: $r \mid 2^t$

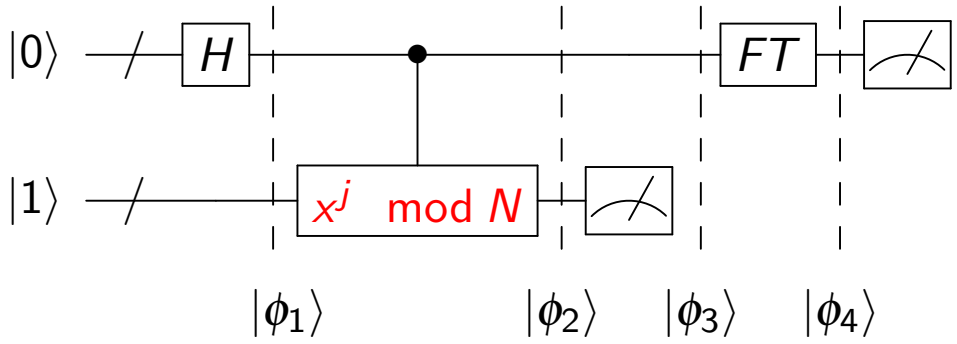


After measuring

$$|\phi_4\rangle = \frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^t}} \sum_{m|k} \sum_{j=0}^{m-1} \mathrm{e}^{2\pi\imath(i+jr)k/2^t} |k\rangle,$$

we would get a multiple k of m with probability 1. By GCDing a series of such multiples k, we could get m as well as $r = 2^t/m$.

Real Case: $r \nmid 2^t$



Anyway, we have

$$|\phi_4\rangle = \frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} \sum_{j=0}^{m-1} \mathrm{e}^{2\pi\imath(i+jr)k/2^t} |k\rangle .$$

Define

$$\alpha_k = \frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^t}} \sum_{j=0}^{m-1} \mathrm{e}^{2\pi\imath(i+jr)k/2^t},$$

thus the probability of taking state $|k\rangle$ is $|\alpha_k|^2$.

Real Case: $r \nmid 2^t$

Note that α_k is a continuous function in variable k, and it's of period $2^t/r$.

During each half-period, saying $[0, 2^t/2r)$, the states $|k\rangle$ with $k < 2^{t-n}$ are good; the others are bad.

We have the upper bound

$$|\alpha_k| = \frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^t}} \left| \frac{1 - e^{2\pi \imath mrk/2^t}}{1 - e^{2\pi \imath rk/2^t}} \right| \leq \frac{1}{\sqrt{m}} \frac{1}{\sqrt{2^t}} \frac{2}{4rk/2^t} = \frac{\sqrt{2^t}}{\sqrt{m}} \frac{1}{2rk}.$$

Real Case: $r \nmid 2^t$

Finally, we have

$$\begin{aligned}\Pr(\{\text{bad}\}) &\leq \sum_{k=2^{t-n}}^{\lceil 2^t/2r \rceil} |\alpha_k|^2 \leq \sum_{k=2^{t-n}}^{\lceil 2^t/2r \rceil} \frac{2^t}{m} \frac{1}{4r^2k^2} \mathrm{d}\, k \\ &\leq \int_{2^{t-n}-1}^{\lceil 2^t/2r \rceil} \frac{2^t}{m} \frac{1}{4r^2k^2} \mathrm{d}\, k \leq \int_{2^{t-n}-1}^{+\infty} \frac{2^t}{m} \frac{1}{4r^2k^2} \mathrm{d}\, k \\ &= \frac{2^t}{m} \frac{1}{4r^2(2^{t-n}-1)} \leq \frac{1}{4r(2^{t-n}-2)},\end{aligned}$$

where the last inequality comes from the fact:

$$\frac{2^t}{mr(2^{t-n}-1)} \leq \frac{2^t}{(2^t-2^n)(2^{t-n}-1)} = \frac{2^{n-t}}{(1-2^{n-t})(1-2^{n-t})} \leq \frac{2^{n-t}}{1-2\cdot 2^{n-t}}.$$

To ensure $\Pr(\{\text{bad}\}) \leq \varepsilon/2r$ during one of $2r$ half-periods,
$t = n + \lceil \log(2 + \frac{1}{2\varepsilon}) \rceil$ in Register 1 suffices!