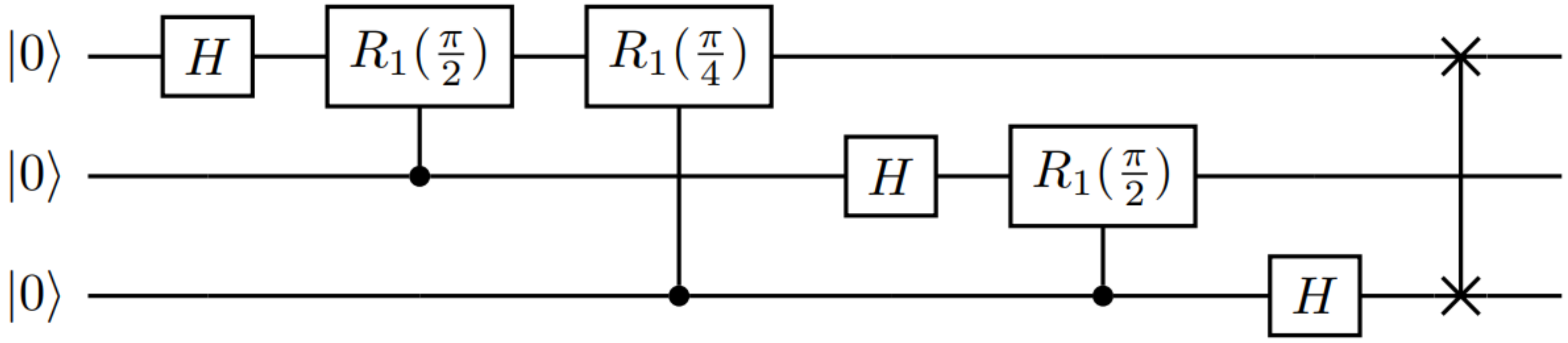
Building up to Shor's algorithm



Mariia Mykhailova
Principal Software Engineer
Microsoft Quantum Systems

Lecture outline

- Eigenvalues and eigenphases
- Phase estimation
- Quantum Fourier transform (QFT)
- Application: quantum counting
- Order finding and integer factorization (Shor’s algorithm)

Linear algebra review: eigenvectors and eigenvalues

Eigenvectors and eigenvalues

Eigenvector $|v\rangle$ of a linear operator A with **eigenvalue** λ:

$$A|v\rangle = \lambda|v\rangle$$

λ is a complex number

The vector $|v\rangle$ changes by a scalar factor when operator A is applied to it

Eigenspace corresponding to eigenvalue λ is a set of eigenvectors which correspond to eigenvalue λ

An eigenspace containing more than one eigenvector (other than the ones different by a scalar factor) is called *degenerate*

Example: Stretch/compression

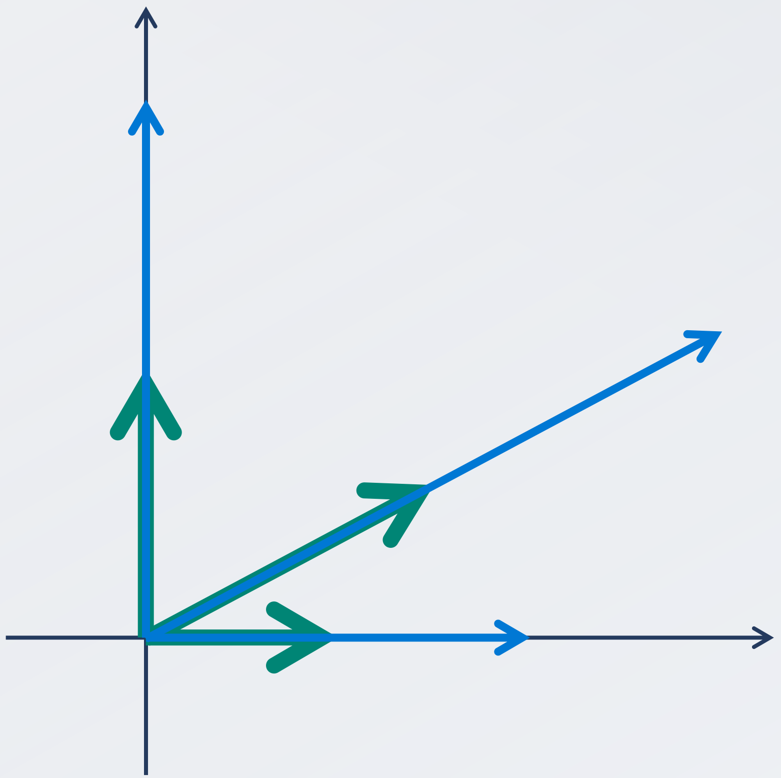
Linear transformation that multiplies all vectors by α:

$$A|v\rangle = \alpha|v\rangle$$

Described by a matrix $\begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}$

Eigenvalue $\lambda = \alpha$

Eigenvectors $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ (degenerate eigenspace)



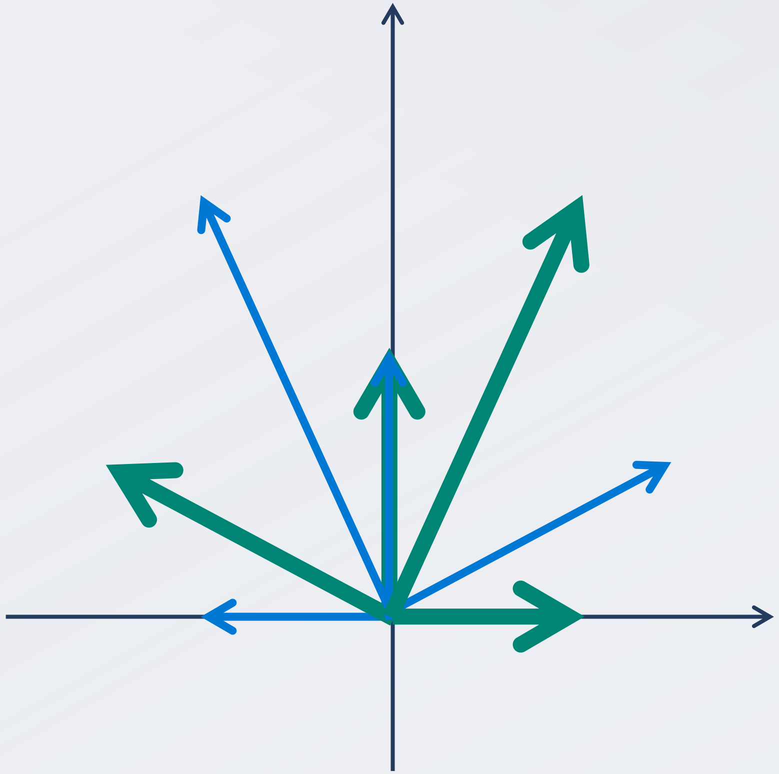
Example: Reflection about the vertical axis

Leaves the y component of the vector unchanged and multiplies the x component by -1

Described by a matrix $\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$

Two eigenvalues:

- $\lambda = 1$ with eigenvector $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- $\lambda = -1$ with eigenvector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$



Eigenvalues and eigenphases of quantum operations

Eigenvalues of unitary matrices

Unitary matrix: its inverse equals its adjoint

$$U^{-1} = U^{\dagger}$$

Unitary matrices preserve the absolute value (“length”) of vector

$$\left|U|v\rangle\right|^2 = \langle v|U^{\dagger}U|v\rangle = \langle v|U^{-1}U|v\rangle = \langle v|v\rangle = \left||v\rangle\right|^2$$
$$\left|U|v\rangle\right| = |\lambda| \cdot \left||v\rangle\right| = \left||v\rangle\right|$$
$$|\lambda| = 1$$

This means that any eigenvalues have form $e^{i\theta}$ for some real θ

$\theta \in [0; 2\pi)$ is called the eigenphase of the unitary

Eigenvalues of Hermitian matrices

Hermitian (self-adjoint) matrix is a unitary matrix for which its inverse equals itself:

$$U^{-1} = U^{\dagger} = U$$

$$U|v\rangle = \lambda|v\rangle$$
$$U^2|v\rangle = U(\lambda|v\rangle) = \lambda U|v\rangle = \lambda^2|v\rangle$$
$$U^2|v\rangle = U^{-1}U|v\rangle = |v\rangle$$
$$\lambda^2 = 1$$

Eigenvalues are ± 1

with corresponding eigenphases 0 and π

Examples: Basic single-qubit gates

Pauli Z gate (self-adjoint): $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Eigenvalues: 1 with vector $|0\rangle$ and -1 with vector $|1\rangle$

Ket-bra representation: $+1|0\rangle\langle 0| - 1|1\rangle\langle 1|$

Pauli X gate (self-adjoint): $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Eigenvalues: 1 with vector $|+\rangle$ and -1 with vector $|-\rangle$

Ket-bra representation: $+1|+\rangle\langle +| -1\ |-\rangle\langle -|$

S gate (not self-adjoint!): $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$

Eigenvalues: 1 with vector $|0\rangle$ and i with vector $|1\rangle$

Ket-bra representation: $+1|0\rangle\langle 0| + i|1\rangle\langle 1|$

Eigenvectors $|v_i\rangle$ that form an orthonormal basis and their eigenvalues λ_i allow to build the ket-bra representation of a gate:

$$A = \sum_i \lambda_i |v_i\rangle\langle v_i|$$

Example: Pauli Y gate

Extra material
(not covered in lecture)

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Find eigenvalues:

$$\begin{vmatrix} -\lambda & -i \\ i & -\lambda \end{vmatrix} = \lambda^2 - 1 = 0, \qquad \lambda = \pm 1$$

Find eigenvectors:

$$\lambda = 1: \quad \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}; \quad \begin{cases} -ix_1 = x_0 \\ ix_0 = x_1 \end{cases}; \quad \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ i \end{pmatrix}$$

$\lambda = -1$: similarly, $\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ -i \end{pmatrix}$

Eigenvalues:

1 with eigenvector $|i\rangle$ and -1 with eigenvector $|-i\rangle$

Examples: Multi-qubit gates

CNOT gate (self-adjoint)

Eigenvalues: 1 with vectors $|00\rangle$, $|01\rangle$ and $\frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$ and -1 with vector $\frac{1}{\sqrt{2}}(|10\rangle - |11\rangle)$

Ket-bra representation:

$$+1(|00\rangle\langle 00| + |01\rangle\langle 01| + \frac{1}{2}(|10\rangle + |11\rangle)(\langle 10| + \langle 11|))$$
$$-1 \cdot \frac{1}{2}(|10\rangle - |11\rangle)(\langle 10| - \langle 11|)$$

Phase oracles

Eigenvalues: 1 with eigenvectors that correspond to $f(x) = 0$
and -1 with eigenvectors that correspond to $f(x) = 1$

Quantum phase estimation problem

Phase estimation: problem statement

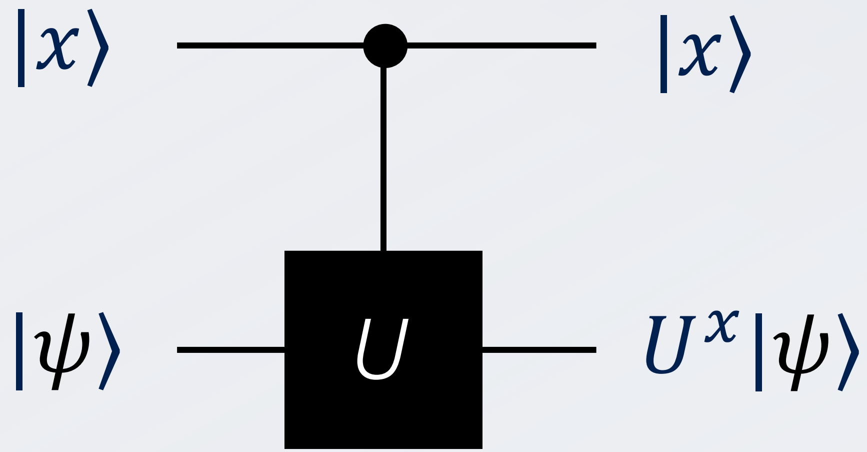
Problem:

given a unitary operator U with eigenvector $|\psi\rangle$ and eigenvalue $\lambda_\psi = e^{2\pi i\theta}$, find θ $(0 \le \theta < 1)$

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$$

Inputs:

- A black box operation that prepares $|\psi\rangle$
- A black box operation that performs controlled-U



Output:

θ with n bits of precision

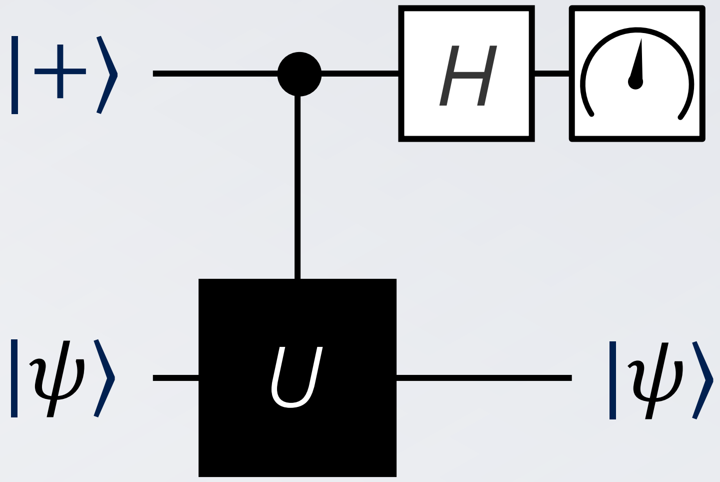
Single-bit variant:

θ has exactly 1 bit of precision (eigenvalue $\lambda_\psi = \pm 1$)

Single-bit phase estimation using phase kickback

Start with $|+\rangle \otimes |\psi\rangle$:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle|\psi\rangle)$$



Apply Controlled U gate:

$$\frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle U|\psi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle\lambda_\psi|\psi\rangle) =$$

$$= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\theta}|1\rangle) \otimes |\psi\rangle = \begin{cases} |+\rangle, \lambda_\psi = 1 \quad (\theta = 0) \\ |-\rangle, \lambda_\psi = -1 (\theta = 1) \end{cases}$$

Apply H gate:

$$|\theta_\psi\rangle \otimes |\psi\rangle, \text{where } \theta_\psi = \begin{cases} 0, \lambda_\psi = 1 \\ 1, \lambda_\psi = -1 \end{cases}$$

Measurement result is exactly the eigenphase!

Multi-bit phase estimation: iterative algorithm

Same circuit, interpreted differently

Start with $|+\rangle \otimes |\psi\rangle$

Apply Controlled U gate:

$$\frac{1}{\sqrt{2}}(|0\rangle + \lambda_\psi |1\rangle)$$

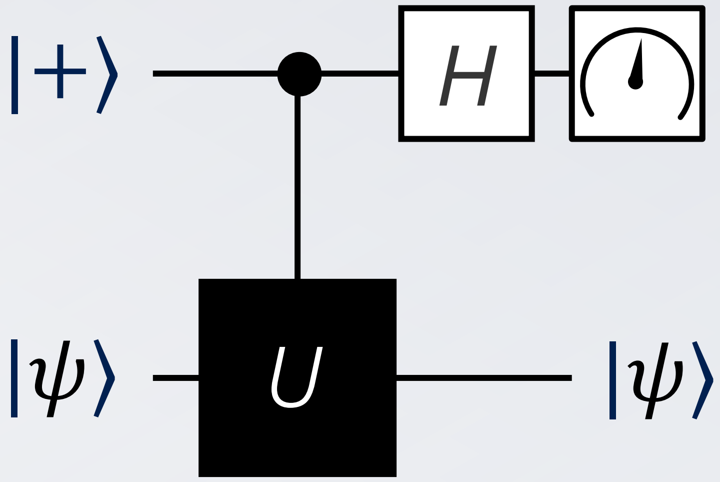
Apply H gate:

$$\frac{1}{2}(1 + \lambda_\psi)|0\rangle + \frac{1}{2}(1 - \lambda_\psi)|1\rangle$$

The probability to measure 0 is now:

$$P(meas = 0) = \frac{1}{4}\left|1 + \lambda_\psi\right|^2 = \cos^2 \pi\theta$$

Run the circuit multiple times to estimate the value



Eigenphase is estimated based on the frequency of measurement outcomes

Multi-bit phase estimation: adaptive algorithms

Learn the phase one binary digit at a time

Adjust later circuits based on information learned earlier

First step of two-bit phase estimation $\theta = 0.\theta_1\theta_2$

Start with $|+\rangle \otimes |\psi\rangle$

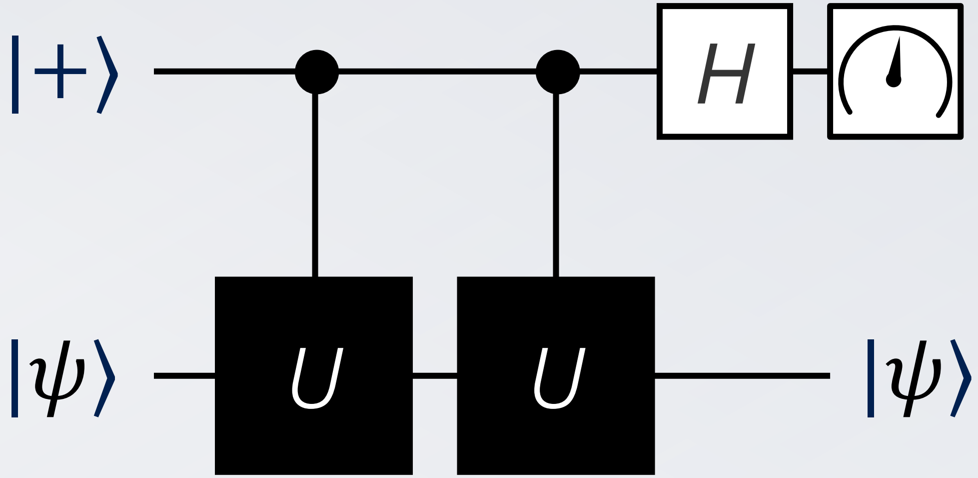
Apply Controlled U gate twice:

$$\frac{1}{\sqrt{2}}(|0\rangle + \lambda_\psi^2|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{4\pi i\theta}|1\rangle) =$$

$$= \frac{1}{\sqrt{2}}(|0\rangle + e^{\pi i(2\theta_1+\theta_2)}|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{\pi i\theta_2}|1\rangle) =$$

$$= \begin{cases} |+\rangle, \theta_2 = 0 \\ |-\rangle, \theta_2 = 1 \end{cases}$$

Apply H gate and measure to learn the value of θ_2



Each digit of the eigenphase is estimated separately based on the frequency of measurement outcomes

Multi-bit phase estimation: adaptive algorithms

Second step of two-bit phase estimation $\theta = 0.\theta_1\theta_2$

Start with $|+\rangle \otimes |\psi\rangle$

Apply Controlled U gate once:

$$\frac{1}{\sqrt{2}}(|0\rangle + \lambda_\psi|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.5\theta_1 + 0.25\theta_2)}|1\rangle)$$

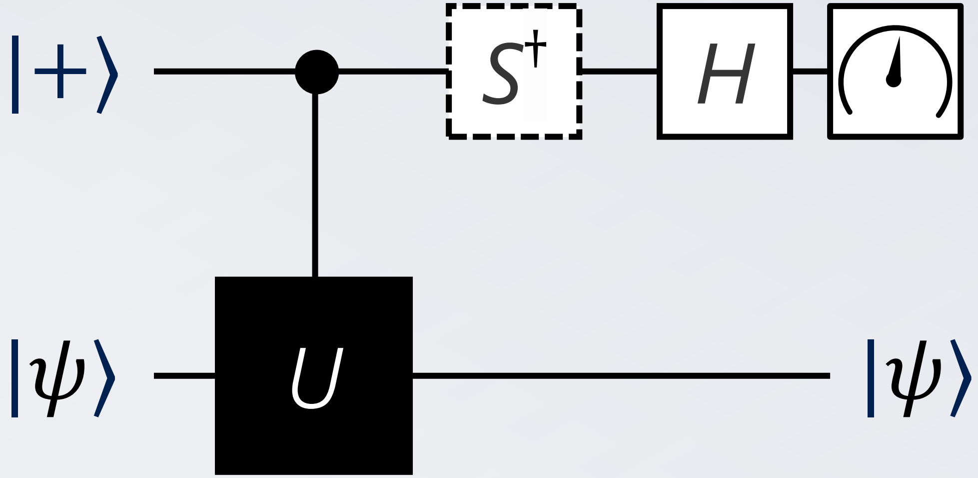
Now, adjust the state based on what we know about θ_2

(apply a classically conditioned $S^\dagger$ gate to cancel the term $e^{0.5\pi i} = i$ if $\theta_2 = 1$, or do nothing if $\theta_2 = 0$)

We end up in a state

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{\pi i\theta_1}|1\rangle) = \begin{cases} |+\rangle, \theta_1 = 0 \\ |-\rangle, \theta_1 = 1 \end{cases}$$

Apply H gate and measure to learn the value of θ_1



Each digit of the eigenphase is estimated separately based on the frequency of measurement outcomes

Quantum Fourier transform and quantum phase estimation algorithm

Discrete Fourier Transform: Definition

Input:

a vector of N complex numbers $x_0, \dots, x_{N-1}$.

Output:

a vector of N complex numbers $y_0, \dots, y_{N-1}$:

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

Why this transform?

Allows to convert between signal and frequency domains

- Signal spectral analysis (frequency information in a signal)
- Image and sound processing, etc.

Quantum Fourier Transform: Definition

Input: an n-qubit state $\sum_{j=0}^{2^n-1} x_j|j\rangle$.

Output: an n-qubit state $\sum_{k=0}^{2^n-1} y_k|k\rangle$, where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j:

$$y_k = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} x_j e^{\frac{2\pi i}{2^n} jk}$$

Notation:

- $N = 2^n$ - the number of amplitudes transformed
- $\omega_N = e^{\frac{2\pi i}{N}}$ - the N-th root of 1 (e.g., for $n = 1$ $N = 2$, $\omega_2 = e^{\pi i} = -1$)

QFT: Small example ($n = 1$)

Input: a 1-qubit state $x_0|0\rangle + x_1|1\rangle$.
Output: a 1-qubit state $y_0|0\rangle + y_1|1\rangle$, where y_k are:

$N = 2,\ \omega_2 = e^{\pi i} = -1$

$$y_k = \frac{1}{\sqrt{2}} \sum_{j=0}^{1} x_j \omega_2^{jk} = \frac{1}{\sqrt{2}}\left(x_0(-1)^0 + x_1(-1)^k\right) = \frac{1}{\sqrt{2}}\left(x_0 + (-1)^k x_1\right)$$

$$y_0 = \frac{1}{\sqrt{2}}(x_0 + x_1), y_1 = \frac{1}{\sqrt{2}}(x_0 - x_1)$$

Effect on basis states:

$$|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), |1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$QFT_1 = H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

That's exactly the Hadamard gate!

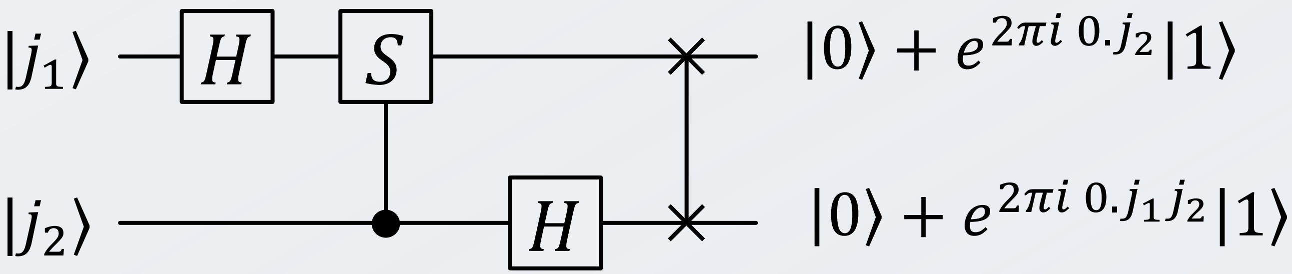
QFT: Slightly larger example ($n = 2$)

Input/output: 2-qubit states

$N = 4,\ \omega_4 = e^{\frac{\pi i}{2}} = i$

$$y_k = \frac{1}{2}\sum_{j=0}^{3} x_j \omega_4^{jk} \qquad |j\rangle \to \frac{1}{2}\sum_{k=0}^{3} \omega_4^{jk} |k\rangle$$

$$QFT_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$



Quick Review: Binary notation

Extra material
(not covered in lecture)

Consider an n-bit integer j.

j can be written as its binary representation:

$$j = j_1 j_2 \dots j_n = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n$$

j_1 is the most significant bit, j_n - the least significant (big-endian notation)

We can represent binary fractions in a similar manner:

$$0.j_k j_{k+1} \dots j_m = j_k \frac{1}{2} + j_{k+1} \frac{1}{2^2} + \cdots + j_m \frac{1}{2^{m-k+1}}$$

$$0.j_1 j_2 \dots j_n = \frac{j}{2^n}$$

QFT: Product representation

Consider an n-qubit basis state $|j\rangle = |j_1 j_2 \dots j_n\rangle$. Its QFT can be represented as a tensor product:

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

$$= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, 0.j_n}|1\rangle) \otimes$$
$$\otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, 0.j_{n-1} j_n}|1\rangle) \otimes$$
$$\dots$$
$$\otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, 0.j_2 \dots j_{n-1} j_n}|1\rangle) \otimes$$
$$\otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, 0.j_1 \dots j_{n-1} j_n}|1\rangle)$$

$$= \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, \frac{j \cdot 2^{n-1}}{2^n}}|1\rangle) \otimes$$
$$\otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, \frac{j \cdot 2^{n-2}}{2^n}}|1\rangle) \otimes$$
$$\dots$$
$$\otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, \frac{j \cdot 2}{2^n}}|1\rangle) \otimes$$
$$\otimes \frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \, \frac{j}{2^n}}|1\rangle)$$

QFT: Product representation for n=2 (example)

Extra material
(not covered in lecture)

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i\frac{jk}{2^n}}|k\rangle =$$

$$= \frac{1}{2}|00\rangle +$$
$$+ \frac{1}{2}e^{2\pi i\,\frac{j\cdot 1}{4}}|01\rangle +$$
$$+ \frac{1}{2}e^{2\pi i\,\frac{j\cdot 2}{4}}|10\rangle +$$
$$+ \frac{1}{2}e^{2\pi i\,\frac{j\cdot 3}{4}}|11\rangle +$$

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\;0.j_2}|1\rangle)\otimes$$
$$\otimes\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\;0.j_1 j_2}|1\rangle\right) =$$

$$= \frac{1}{2}|00\rangle +$$
$$+ \frac{1}{2}e^{2\pi i\;0.j_1 j_2}|01\rangle +$$
$$+ \frac{1}{2}e^{2\pi i\;0.j_2}|10\rangle +$$
$$+ \frac{1}{2}e^{2\pi i(0.j_2+0.j_1 j_2)}|11\rangle$$

QFT: Implement each term of product representation

The first term $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\, 0.j_n}|1\rangle)$**:**
apply the H gate to $|j_n\rangle$

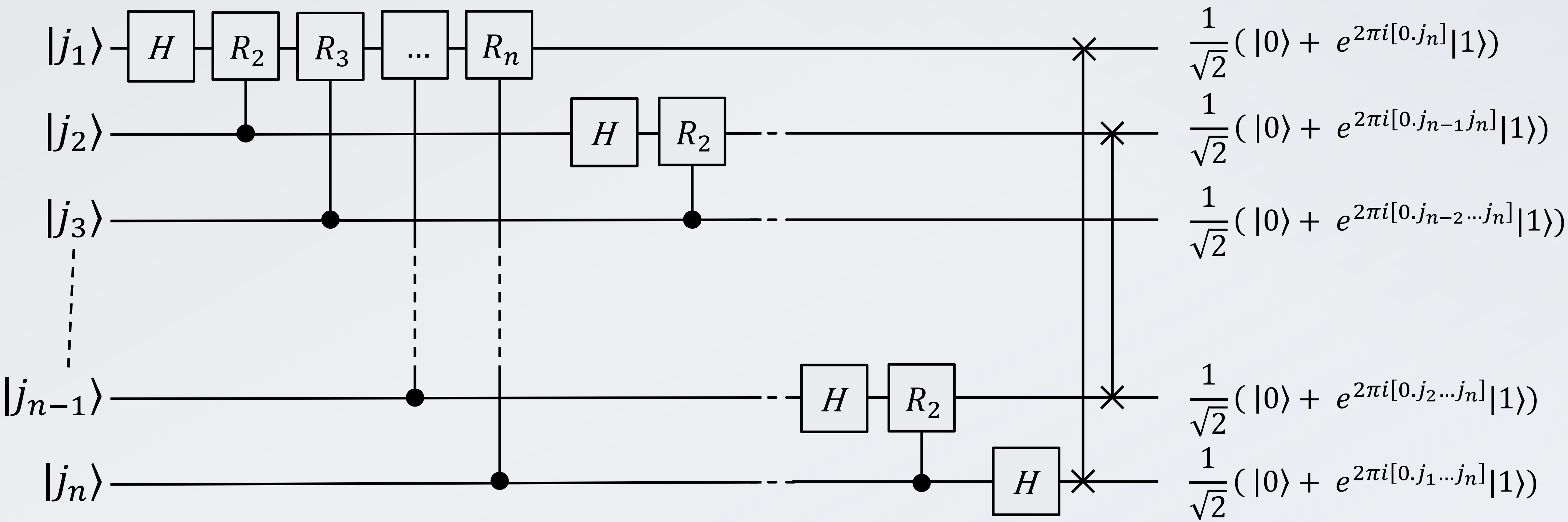
The second term $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\, 0.j_{n-1}j_n}|1\rangle)$**:**

- Get $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\, 0.j_{n-1}}|1\rangle)$ by applying H gate to $|j_{n-1}\rangle$
- Add extra $e^{2\pi i\, 0.0 j_n}$ phase to $|1\rangle$ state using a controlled S gate with $|j_n\rangle$ as control and $|j_{n-1}\rangle$ as target

And so on; the last term $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\, 0.j_1 \dots j_{n-1}j_n}|1\rangle)$**:**

- Get $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i\, 0.j_1}|1\rangle)$ by applying the H gate to $|j_1\rangle$
- Add extra phases using controlled rotation gates (with decreasing rotation angles) with each of the other qubits as controls

QFT: Full circuit



Remember to reverse the order of qubits after the rotations!

Importance of quantum Fourier transform

QFT circuit for n qubits requires $O(n^2)$ gates

The best classical implementation of DFT is fast Fourier transform which requires $O(n2^n)$ operations

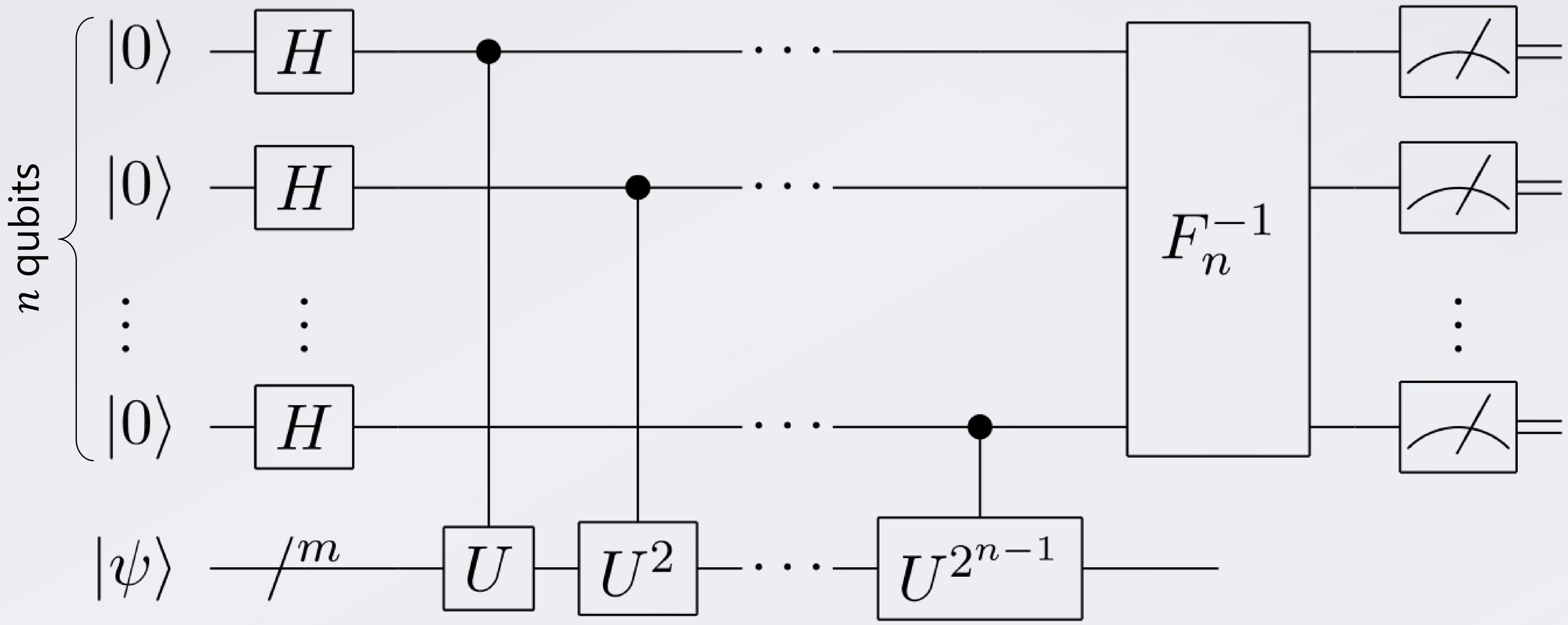
Exponential speedup?

- It is hard to encode the inputs $x_0, \dots, x_{N-1}$ into a quantum state
- And it is hard to read out the results of the transformation $y_0, \dots, y_{N-1}$ from the quantum state that is the result of QFT
- QFT is *not* used to get a speedup for computing DFT

QFT is an important building block for other algorithms:

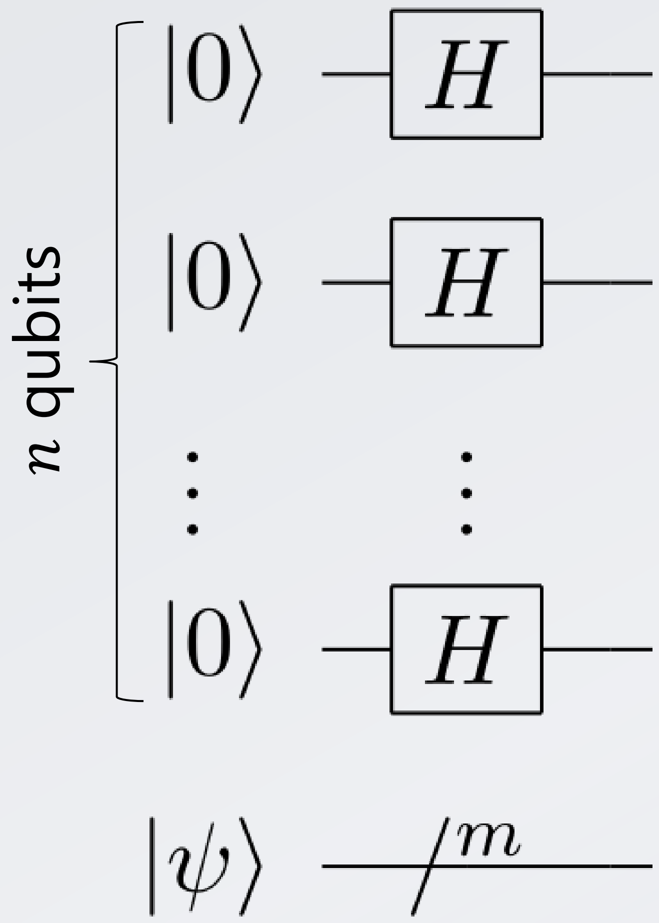
- Quantum phase estimation and all its applications (including counting)
- Order-finding and factoring (Shor's algorithm)
- Hidden subgroup problem, discrete logarithm problem, etc.

Quantum phase estimation algorithm



Phase estimation circuit

Extra material
(not covered in lecture)



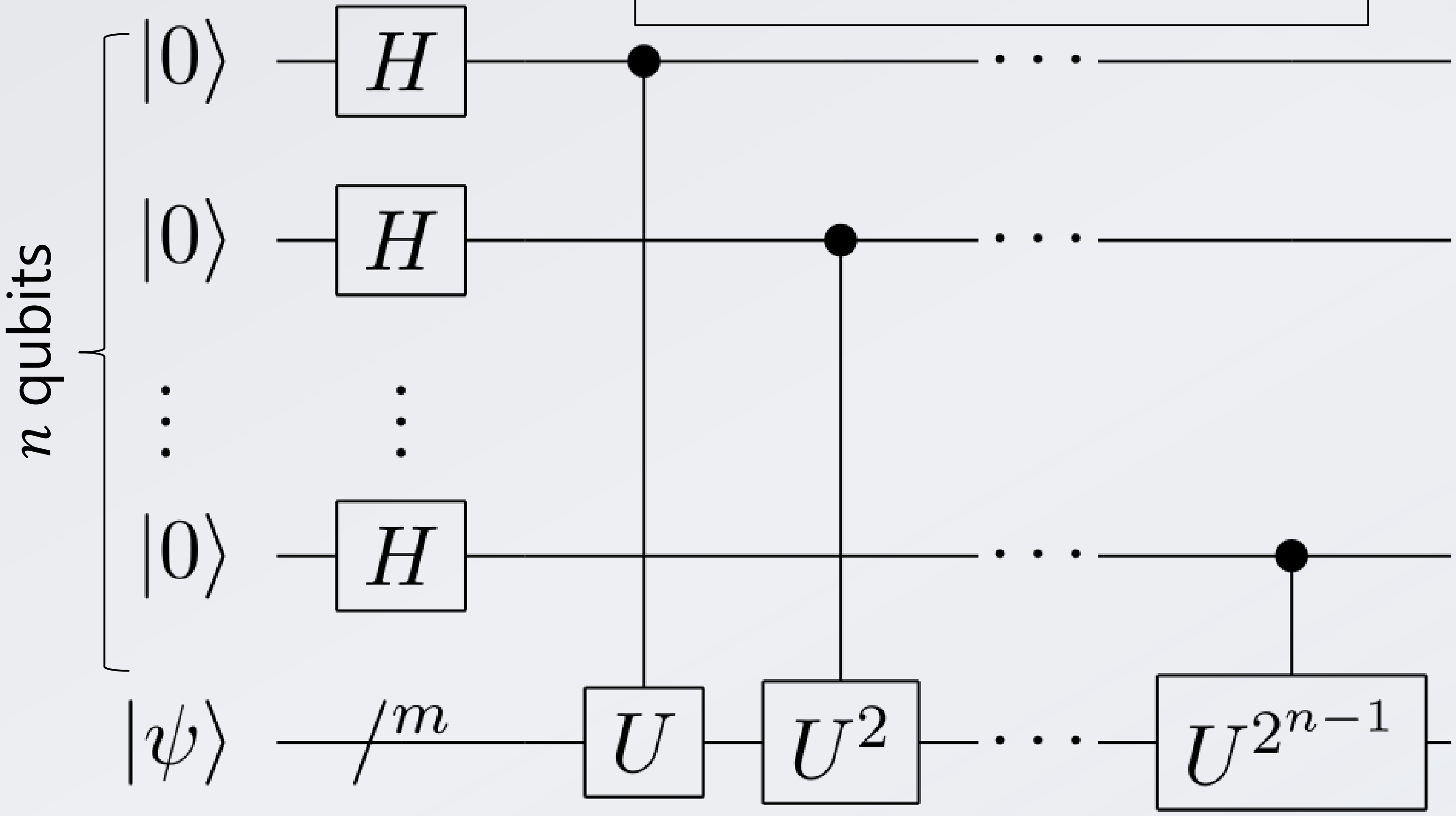
$$|00\ldots0\rangle \otimes |\psi\rangle \to$$

$$\to (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \otimes |\psi\rangle$$
$$= (|0\rangle + |1\rangle + \cdots + |2^n - 1\rangle) \otimes |\psi\rangle$$

Phase estimation circuit

Extra material
(not covered in lecture)

$$|x\rangle|\psi\rangle \to |x\rangle U^x|\psi\rangle$$



$$\to e^{2\pi i\theta\cdot 0}|0\rangle + e^{2\pi i\theta\cdot 1}|1\rangle + e^{2\pi i\theta\cdot 2}|2\rangle + \cdots + e^{2\pi i\theta\cdot(2^n-1)}|2^n-1\rangle$$

$$= \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i\theta\cdot k}|k\rangle$$

Inverse quantum Fourier transform

Extra material
(not covered in lecture)

Represent θ as a binary fraction (since $0 \leq \theta < 1$):

$$\theta = 0.\theta_1\theta_2 \dots \theta_n = \frac{1}{2^n}\theta_1\theta_2 \dots \theta_n$$

Then

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i\theta\cdot k}|k\rangle = \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{\frac{2\pi i}{2^n}\theta_1\theta_2\dots\theta_n\cdot k}|k\rangle =$$

$$= QFT|\theta_1\theta_2 \dots \theta_n\rangle$$

We can use QFT^{-1} to recover the binary notation of θ $\theta_1\theta_2 \dots \theta_n$!

Steps of quantum phase estimation

Extra material
(not covered in lecture)

1. $|0\rangle|\psi\rangle$ — Initialize state

2. $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle|\psi\rangle$ — Apply H gates

3. $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k\rangle U^k|\psi\rangle$
   $= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i k\varphi}|k\rangle|\psi\rangle$ — Apply controlled U ladder

4. $\rightarrow |\tilde{\varphi}\rangle|\psi\rangle$ — Apply inverse QFT

5. $\rightarrow \tilde{\varphi}$ — Measure first register

Obtain φ with n bits of precision

Practical aspects of using phase estimation

What if we don't know eigenvector or how to prepare it?

Phase estimation will work on arbitrary superpositions: applying controlled unitary will produce a superposition of pairs "eigenphase-eigenvector", and measurement will pick a random one

Quantum phase estimation is a probabilistic algorithm

If the phase does not have an exact binary representation, it will return a random phase, not always the binary fraction closest to the phase

Different solutions for different scenarios

- Quantum phase estimation takes the fewest runs, but requires n extra qubits to store n bits of binary representation of the phase (and the deepest circuit)
- Adaptive and iterative phase estimation use smaller circuits (good for NISQ machines), with only 1 extra qubit to extract phase information, but require more runs
- Adaptive phase estimation can use different classical approaches for analyzing the data from experiments

Microsoft

Section done

Application: Quantum counting

Counting problem

Search problem:

given a quantum oracle for function $f: \{0,1\}^n \to \{0,1\}$, find an x such that $f(x) = 1$ or determine that there is no such x.

Can be solved using Grover's algorithm in $O(\sqrt{N})$ queries

Optimal algorithm implementation requires knowing the number of solutions M

How to find M? That's the counting problem

Classical solution: $O(N)$ queries (need to check each input and count how many of them are solutions)

Quantum counting algorithm

Consider the Grover iteration unitary: it represents a rotation by $2\theta \approx 2\sqrt{\frac{M}{N}}$

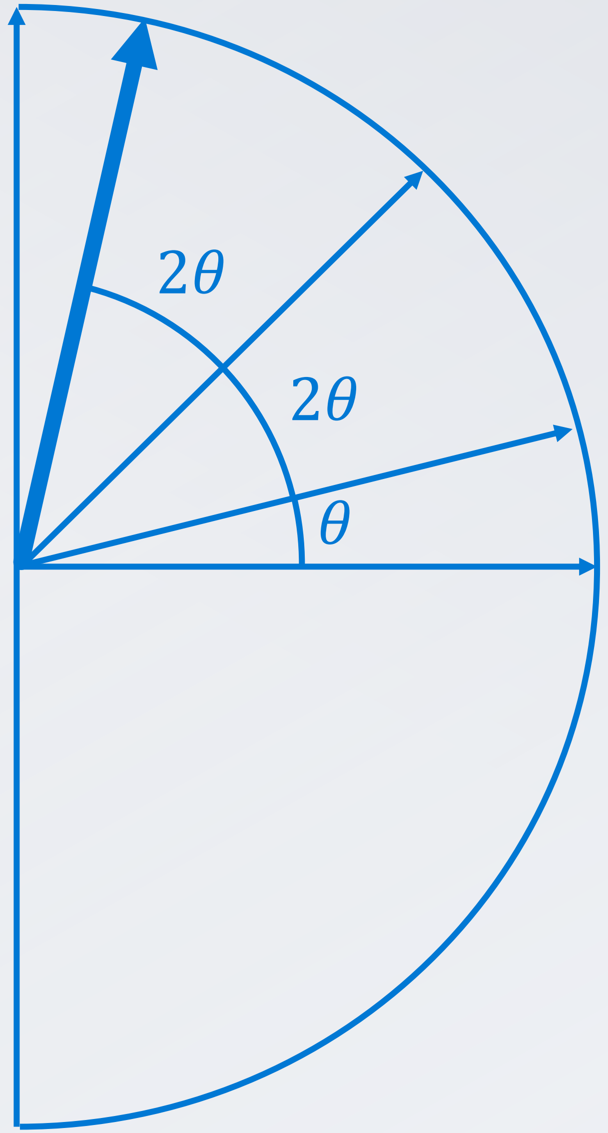
Counterclockwise rotation by 2θ on a 2D plane has the matrix

(in $\{|\psi_{\text{bad}}\rangle, |\psi_{\text{good}}\rangle\}$ basis)

$$\begin{pmatrix} \cos 2\theta & -\sin 2\theta \\ \sin 2\theta & \cos 2\theta \end{pmatrix}$$

This matrix has eigenvalues $e^{i2\theta}$ and $e^{i(2\pi-2\theta)}$

Can use phase estimation to estimate θ and deduce M from it



Microsoft

Factoring problem and RSA

RSA: key generation

Bob wants to receive encoded messages from public

Bob chooses two large primes, $\boldsymbol{p}$ and $\boldsymbol{q}$

Bob calculates $\boldsymbol{N = pq}$ and a number $\mathbf{c}$ that is co-prime to $M = (p-1)(q-1)$ (i.e., $\gcd(c, M) = 1$)

Bob can compute the multiplicative inverse $\boldsymbol{d}$ of $c\ (mod\ M)$ (because he knows p, q):

$$c\,d \equiv 1 (\mathrm{mod}\, M)$$

Public key is $(\boldsymbol{N}, \boldsymbol{c})$

Private (secret) key is $(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{M}, \boldsymbol{d})$

Public info   Bob's secrets   Alice's secret

RSA: encryption

Alice wants to send an encrypted message to Bob

She encodes her message, say, as ASCII codes of characters

The codes can be grouped into larger numbers $\boldsymbol{a_i < N}$

Alice uses Bob's public key $(\boldsymbol{N}, \boldsymbol{c})$ to calculate

$$b \equiv a^c \pmod{N}$$

She sends message $\boldsymbol{b}$ to Bob through a public channel

Public info Bob's secrets Alice's secret

RSA: decryption

Bob receives $\boldsymbol{b}$ and uses $\boldsymbol{d}$ (his secret) to compute

$$b^{d} \pmod{N}$$

Fact: for any integer $\boldsymbol{a}$ $\boldsymbol{a}^{cd} \equiv \boldsymbol{a}\ (\boldsymbol{mod}\ \boldsymbol{N})$

Therefore, if $\boldsymbol{b} \equiv \boldsymbol{a}^{c}\ (\boldsymbol{mod}\ \boldsymbol{N})$ then

$$b^{d} \equiv a \pmod{N}$$

Bob now has the decrypted message $\boldsymbol{a}$!

If Eve can find $(\boldsymbol{p}, \boldsymbol{q})$ from $\boldsymbol{N}$, she can find $\boldsymbol{d}$ and use it to decrypt the message!

Public info Bob's secrets Alice's secret

RSA Factoring Challenge

Extra material
(not covered in lecture)

- The security of the RSA protocol relies on the fact that integer factorization (given N, find (p, q)) is hard
- RSA Factoring Challenge was a series of challenges on factoring increasingly large numbers
- Ran from 1991 to 2007 to encourage research

RSA-2048 challenge: factor the following number:

```
25195908475657893494027183240048398571429282126204032
02777713783604366202070759555626401852588078440691829
06412495150821892985591491761845028084891200728449926
87392807287776735971418347270261896375014971824691165
07761337985909570009733045974880842840179742910064245
86918171951187461215151726546322822168699875491824224
33637259085141865462043576798423387184774447920739934
23658482382428119816381501067481045166037730605620161
96762561338441436038339044149526344321901146575444541
78424020924616515723350778707749817125772467962926386
35637328991215483143816789988504044536402352738195137
8636564391212010397122822120720357
```

Order finding

Order finding

For positive N and $a < N$ which are coprime (i.e., $\gcd(a, N) = 1$), the **order** of a modulo N, denoted as $ord_N(a)$, is the least positive integer r such that

$$a^r \equiv 1 (\text{mod } N)$$

Problem:

Given a and N, find the order r

Example:

$N = 15, a = 2$:

powers of a are 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, …

modulo 15: 1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, …

$$ord_{15}(2) = 4$$

- Finding the first number of sequence that equals 1 is hard
- But the *period* of the sequence is a global property, so there can be a clever quantum trick to detect it!

Relation to integer factoring

For given N and a, if $r = ord_N(a)$, numbers

$$\gcd(N, a^{r/2} - 1) \text{ and } \gcd(N, a^{r/2} + 1)$$

might be factors of N.

- There are several other considerations (such as under which conditions this is true), but that's the idea behind the algorithm!
- The only quantum part of the algorithm is order finding, the rest is classical number theory

Quantum oracle for modular multiplication by a

Unitary operation U that acts as follows (for each basis state $y \in \{0,1\}^n$):

$$U|y\rangle \equiv |a \cdot y \pmod N\rangle$$

- a and N are fixed for the problem, so you can build the circuit
- $n = \log_2 N$ is the number of bits in binary notation of N
- U acts non-trivially only for $0 \leq y \leq N - 1$
- Otherwise, $U|y\rangle = |y\rangle$
- U is a unitary because a and N are coprime, so there are no collisions, $a \rightarrow a \cdot y \pmod N$ is a permutation

A special vector for the oracle

$$|\psi_1\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i \left(\frac{1}{r}\right) j} |a^j \bmod N\rangle$$

Let's look at the effect of U on that vector:

$$U|\psi_1\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i \left(\frac{1}{r}\right) j} |a^{j+1} \bmod N\rangle =$$

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

$$= e^{2\pi i \left(\frac{1}{r}\right)} \frac{1}{\sqrt{r}} \left(\sum_{j=1}^{r-1} e^{-2\pi i \left(\frac{1}{r}\right) j} |a^j \bmod N\rangle + e^{-2\pi i \left(\frac{1}{r}\right) r} |a^r \bmod N\rangle \right) =$$

$$= e^{2\pi i \left(\frac{1}{r}\right)} |\psi_1\rangle$$

$$a^r \bmod N = 1$$
$$= a^0 \bmod N$$

Can we apply phase estimation to find r?

1. We need an efficient method of performing controlled-U^{2^j} operation for any integer j.
   Use modular exponentiation – reversible arithmetic

2. We need an efficient method to prepare the eigenvector $|\psi_1\rangle$.
   It looks complicated and requires knowing r – and that's what we're trying to find?

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

Preparing $|\psi_k\rangle$

We can define multiple states $|\psi_k\rangle$:

$$
\left.\begin{aligned}
|\psi_1\rangle &= \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1} e^{-2\pi i\left(\frac{1}{r}\right)j}\,|a^j \bmod N\rangle \\
&\ \ \vdots \\
|\psi_k\rangle &= \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1} e^{-2\pi i\left(\frac{k}{r}\right)j}\,|a^j \bmod N\rangle \\
&\ \ \vdots \\
|\psi_r\rangle &= \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1} e^{-2\pi i\left(\frac{r}{r}\right)j}\,|a^j \bmod N\rangle
\end{aligned}\right\}
$$

Any of these states can be used to estimate k/r (with classical math).

Then $r = k\left(\frac{k}{r}\right)^{-1}$.

Superposition of $|\psi_k\rangle$

For $1 \leq k \leq r$, $|\psi_k\rangle$ are eigenstates of U:

$$U|\psi_k\rangle = e^{2\pi i k/r}|\psi_k\rangle$$

It turns out that

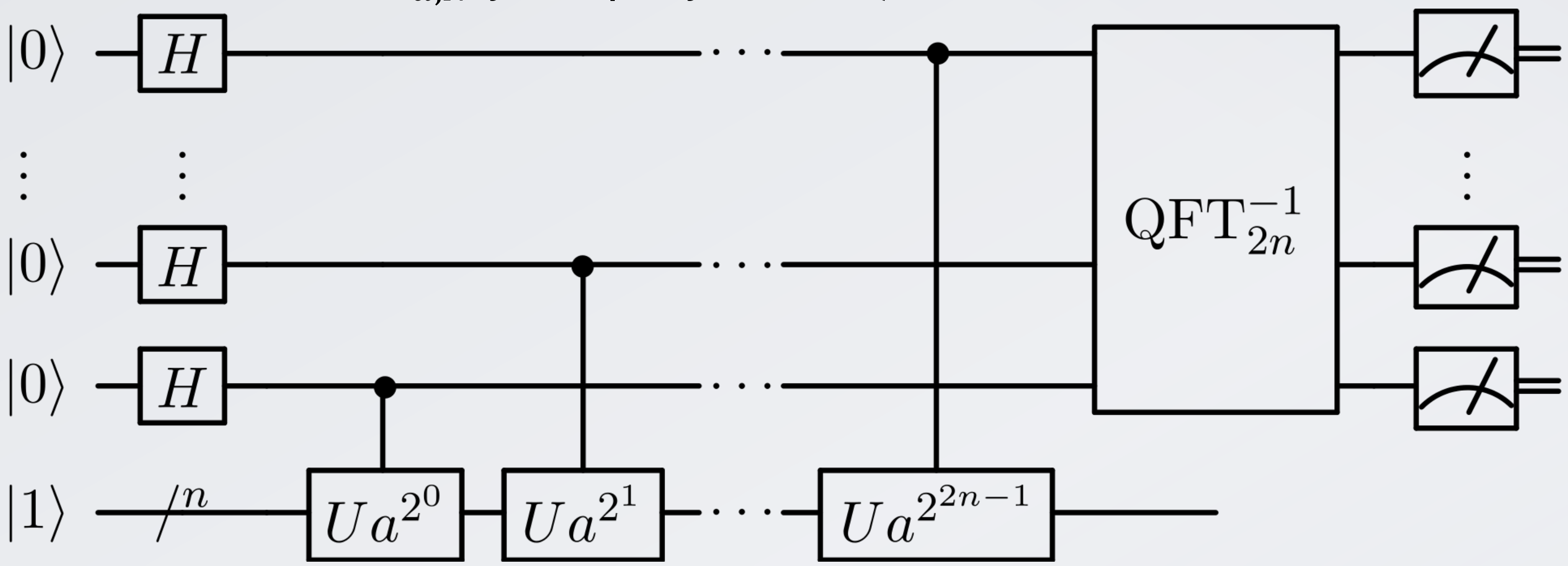
$$\frac{1}{\sqrt{r}}\sum_{k=1}^{r}|\psi_k\rangle = \frac{1}{r}\sum_{k=1}^{r}\sum_{j=0}^{r-1} e^{-2\pi i\left(\frac{k}{r}\right)j}|a^j \bmod N\rangle = |1\rangle$$

We do not need to prepare $|\psi_1\rangle$; instead, prepare their superposition $|1\rangle$

Phase estimation will estimate the phase of each $|\psi_k\rangle$ in the superposition, and then measurement will pick a random one

Quantum algorithm for order finding

$$U_{a,N}|y\rangle = |a \cdot y \; mod \; N\rangle$$



Measure the output.

Apply classical post-processing to get r

Requires $O(n^2 \; log \, n \; log \; log \, n)$ gates for success probability $\Omega\left(\frac{1}{log \, n}\right)$

Repeat $O(log \, n)$ times for constant success probability, taking the smallest r such that $a^r \equiv 1 \; mod \; N$

Integer factoring (Shor’s algorithm)

Integer factoring

Input: a positive integer N ($n = log_2 N$ is the number of bits in binary notation of N)

Output: the prime factorization of N

$$N = p_1^{a_1} \dots p_k^{a_k}$$

Best known classical algorithm:

$$2^{O(n^{1/3} (log\, n)^{2/3})}$$

Shor's algorithm:

$$O(n^3)$$

Integer factoring algorithm

1. Pick a random number $1 < a < N$.
2. If $\gcd(a, N) > 1$, return $\gcd(a, N)$ and $N/\gcd(a, N)$.
   Computing greatest common divisor is fast (polynomial in n)
3. Find the order of $a \bmod N$ $r = ord_N(a)$
4. If r is even, compute $x = a^{r/2} - 1 (\bmod N)$ and $\gcd(x, N)$.
5. If $\gcd(x, N) > 1$, return $\gcd(x, N)$ and $N/\gcd(x, N)$.
6. If r is odd or $\gcd(x, N) = 1$, go back to Step 1.

Special cases:

N is even (return 2 and $N/2$)

N is a prime number or a power of a prime – check using primality testing (“fast” - polynomial)