Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- 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.
- One such transformation is the discrete Fourier transform. The discrete Fourier transform takes as input a vector of complex numbers, $x_0, \ldots, x_{N-1}$ where the length N of the vector is a fixed parameter. It outputs the transformed data, a vector of complex numbers $y_0, \ldots, y_{N-1}$, defined by

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

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- The quantum Fourier transform on an orthonormal basis |0>, . . . , |N − 1> is defined to be a linear operator with the following action on the basis states,

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle \,. \qquad \sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle$$

- Where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j.

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- It is not obvious from the definition, but this transformation is a unitary transformation, and thus can be implemented as the dynamics for a quantum computer.

- Exercise: Give a direct proof that the linear transformation defined previously is unitary.

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- In the following, we take $N = 2^n$, where n is some integer, and the basis $|0\rangle, \ldots, |2^n-1\rangle$ is the computational basis for an n-qubit quantum computer.
- It is helpful to write the state $|j\rangle$ using the binary representation $j = j_1 j_2 \ldots j_n$.
- It is also convenient to adopt the notation $0.j_l j_{l+1} \ldots j_m$ to represent the binary fraction $j_l/2 + j_{l+1}/4 + \cdots + j_m/2^{m-l+1}$.

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- With a little algebra the quantum Fourier transform can be given the following useful *product representation:*

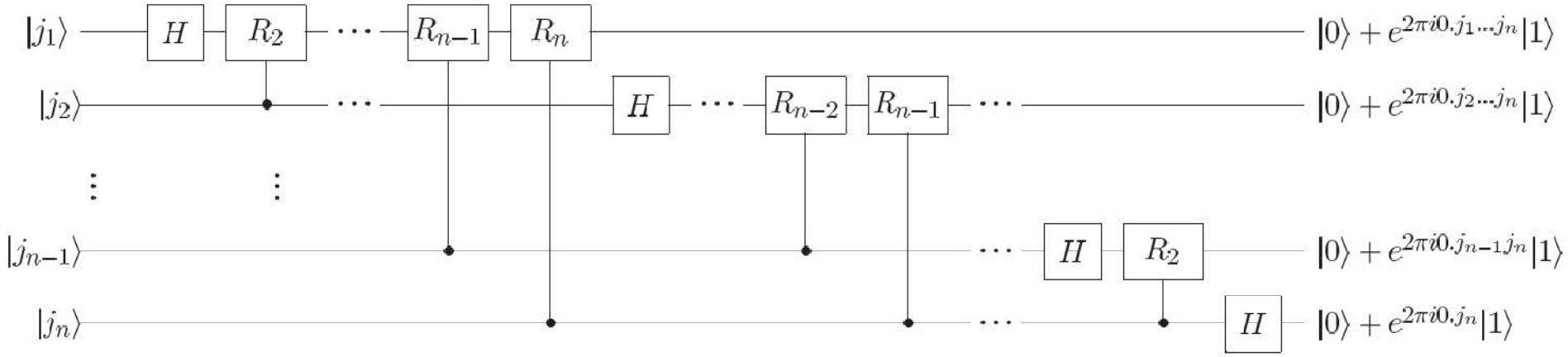
$$|j_1, \ldots, j_n\rangle \rightarrow \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)}{2^{n/2}}.$$

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- The equivalence of the product representation in the previous slide follows from some elementary algebra:

$$
\begin{aligned}
|j\rangle \rightarrow & \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle \\
= & \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \ldots k_n\rangle \\
= & \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
= & \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right] \\
= & \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right] \\
= & \frac{\left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle\right)}{2^{n/2}}
\end{aligned}
$$

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform



- Efficient circuit for the quantum Fourier transform. The gate R_k denotes the unitary transformation

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

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- We start by doing a Hadamard gate and $n - 1$ conditional rotations on the first qubit – a total of n gates. This is followed by a Hadamard gate and $n-2$ conditional rotations on the second qubit, for a total of $n + (n-1)$ gates. Continuing in this way, we see that $n+(n-1)+\cdots+1 = n(n+1)/2$ gates are required, plus the gates involved in the swaps.

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- At most $n/2$ swaps are required, and each swap can be accomplished using three controlled- gates. Therefore, this circuit provides a $\Theta(n^2)$ algorithm for performing the quantum Fourier transform.
- In contrast, 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 $\Theta(n2^n)$ gates. That is, it requires exponentially more operations to compute the Fourier transform on a classical computer than it does to implement the quantum Fourier transform on a quantum computer.

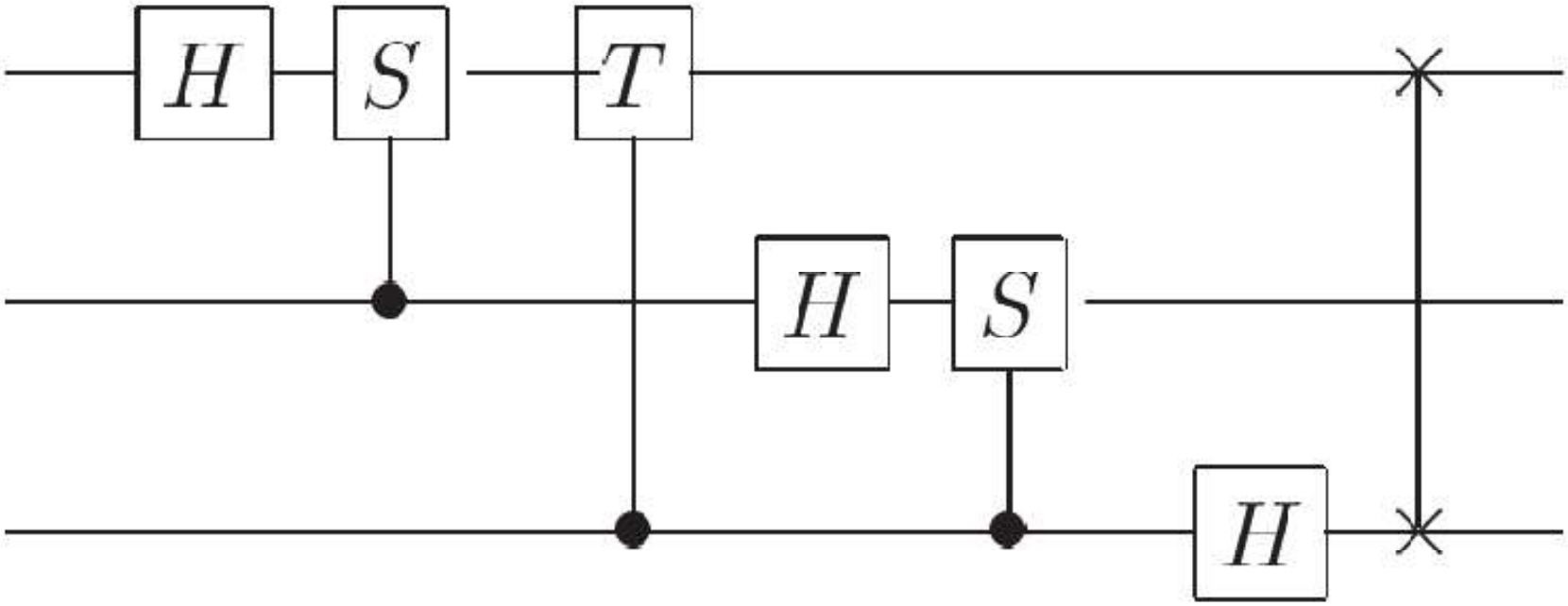
Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform

- Example: Consider the quantum Fourier transform on 3 qubits. Fourier transform in this instance may be written out explicitly, using

$$\omega = e^{2\pi i/8} = \sqrt{i}, \text{ as}$$

$$\frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega^1 & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix}$$

Quantum Computation- Quantum Algorithms-The Quantum Fourier Transform



- Recall that *S* and *T* are the phase and $\pi/8$ gates

Quantum Computation- Quantum Algorithms-Phase Estimation

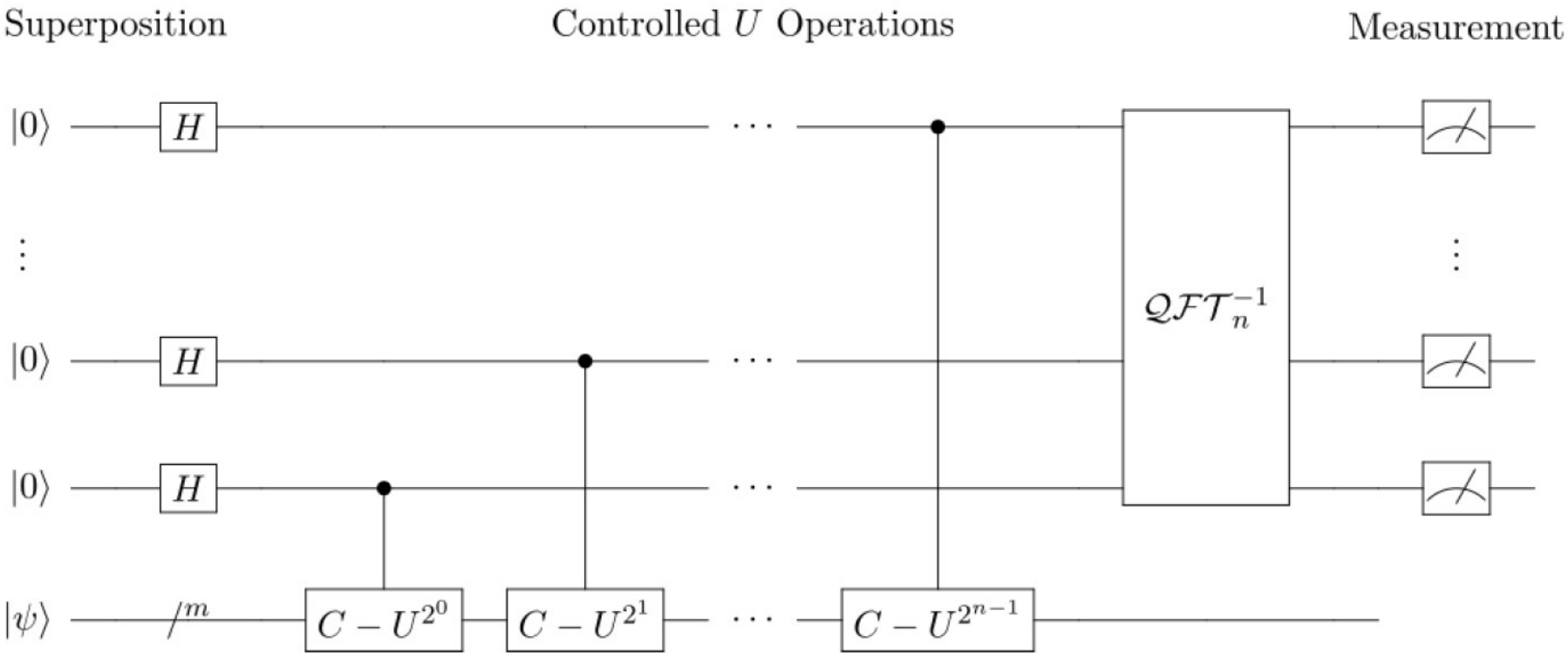
- Problem:
- Let *U* be a unitary operator that operates on *m* qubits with an eigenvector $|\psi\rangle$ such that

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle, 0 \leq \theta < 1.$$

- We would like to find the eigenvalue $e^{2\pi i\theta}$

- which in this case is equivalent to estimating the phase θ, to a finite level of precision.

Quantum Computation- Quantum Algorithms-Phase Estimation



Quantum phase estimation circuit

Quantum Computation- Quantum Algorithms-Phase Estimation

- How it works?
- The input consists of two registers(namely, two parts): the upper *n* qubits comprise the *first register*, and the lower *m* qubits are the *second register*.
- The initial state of the system is

$$|0\rangle^{\otimes n}|\psi\rangle.$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- After applying *n-bit* Hadamard gate operation on the first register, the state of the first register can be described as:

$$\frac{1}{2^{\frac{n}{2}}}(|0\rangle + |1\rangle)^{\otimes n}.$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- **Apply controlled unitary operations**
- Let ***U*** be a unitary operator with eigenvector $|\psi\rangle$ such that

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

- Thus,

$$U^{2^j}|\psi\rangle = U^{2^j-1}U|\psi\rangle = U^{2^j-1}e^{2\pi i\theta}|\psi\rangle = \cdots = e^{2\pi i 2^j\theta}|\psi\rangle.$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- ***C-U*** is a *controlled-U* gate which applies the unitary operator ***U*** on the second register only if its corresponding control bit (from the first register) is $|1\rangle$.
- After applying all the n controlled operations $C-U^{2^j}$ with $0 \leq j \leq n-1$

- as seen in the circuit, and kicking back phases to the control bits in the first register, the state of the **first register** can be described as

$$\frac{1}{2^{\frac{n}{2}}}\underbrace{\left(|0\rangle + e^{2\pi i 2^{n-1}\theta}|1\rangle\right)}_{1^{st}\ qubit} \otimes \cdots \otimes \underbrace{\left(|0\rangle + e^{2\pi i 2^{1}\theta}|1\rangle\right)}_{n-1^{th}\ qubit} \otimes \underbrace{\left(|0\rangle + e^{2\pi i 2^{0}\theta}|1\rangle\right)}_{n^{th}\ qubit} = \frac{1}{2^{\frac{n}{2}}}\sum_{k=0}^{2^n-1} e^{2\pi i\theta k}|k\rangle,$$

- where $|k\rangle$ denotes the binary representation of ***k***.

Quantum Computation- Quantum Algorithms-Phase Estimation

- **Apply inverse Quantum Fourier transform**
- Applying inverse Quantum Fourier transform on

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

- yields

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1} e^{2\pi i\theta k} e^{\frac{-2\pi ikx}{2^n}}|x\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1} e^{-\frac{2\pi ik}{2^n}(x-2^n\theta)}|x\rangle.$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- The state of both registers together is

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1} e^{-\frac{2\pi i k}{2^n}(x-2^n\theta)}|x\rangle \otimes |\psi\rangle.$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- **Phase approximation representation**
- We can approximate the value of $\theta \in [0,1]$ by rounding $2^n\theta$ to the nearest integer. This means that $2^n\theta = a + 2^n\delta$, where a is the nearest integer to $2^n\theta$ and the difference $2^n\delta$ satisfies

$$0 \leqslant |2^n\delta| \leqslant \frac{1}{2}$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- We can now write the state of the first and second register in the following way:

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{k=0}^{2^n-1} e^{-\frac{2\pi i k}{2^n}(x-a)} e^{2\pi i \delta k}|x\rangle \otimes |\psi\rangle.$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- **Measurement**
- Performing a measurement in the computational basis on the first register yields the result $|a\rangle$ with probability

$$\Pr(a) = \left| \left\langle a \underbrace{\left| \frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{\frac{-2\pi i k}{2^n}(x-a)} e^{2\pi i \delta k} \right|}_{\text{State of the first register}} x \right\rangle \right|^2 = \frac{1}{2^{2n}} \left| \sum_{k=0}^{2^n-1} e^{2\pi i \delta k} \right|^2 = \begin{cases} 1 & \delta = 0 \\ \frac{1}{2^{2n}} \left| \frac{1-e^{2\pi i 2^n \delta}}{1-e^{2\pi i \delta}} \right|^2 & \delta \neq 0 \end{cases}$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- For δ=0 the approximation is precise, thus $a=2^n$ and $Pr(a)=1$.
- In this case, we always measure the accurate value of the phase.
- The state of the system after the measurement is

$$|2^n\theta\rangle \otimes |\psi\rangle$$

Quantum Computation- Quantum Algorithms-Phase Estimation

- For $\delta \neq 0$ since $|\delta| \ll \frac{1}{2^{n+1}}$ the algorithm yields the correct result with probability

$$\Pr(a) \geqslant \frac{4}{\pi^2} \approx 0.405$$

- Let's prove it →

Quantum Computation- Quantum Algorithms-Phase Estimation

$$\Pr(a) = \frac{1}{2^{2n}} \left| \frac{1 - e^{2\pi i 2^n \delta}}{1 - e^{2\pi i \delta}} \right|^2 \qquad \text{for } \delta \neq 0$$

$$= \frac{1}{2^{2n}} \left| \frac{2\sin(\pi 2^n \delta)}{2\sin(\pi\delta)} \right|^2 \qquad |1 - e^{2ix}|^2 = 4|\sin(x)|^2$$

$$= \frac{1}{2^{2n}} \frac{|\sin(\pi 2^n \delta)|^2}{|\sin(\pi\delta)|^2}$$

$$\geqslant \frac{1}{2^{2n}} \frac{|\sin(\pi 2^n \delta)|^2}{|\pi\delta|^2} \qquad |\sin(\pi\delta)| \leqslant |\pi\delta| \text{ for } |\delta| \leqslant \frac{1}{2^{n+1}}$$

$$\geqslant \frac{1}{2^{2n}} \frac{|2 \cdot 2^n \delta|^2}{|\pi\delta|^2} \qquad |2 \cdot 2^n \delta| \leqslant |\sin(\pi 2^n \delta)| \text{ for } |\delta| \leqslant \frac{1}{2^{n+1}}$$

$$= \frac{4}{\pi^2}$$

Quantum Computation- Quantum Algorithms-Application: order-finding

- For positive integers *x and N,x < N,with no common factors, the order of x modulo N* is defined to be the least positive integer, *r, such that* $x^r = 1 (\mathrm{mod}\ N)$.
- The order-finding problem is to determine the order for some specified *x and N.*
- Order-finding is believed to be a hard problem on a classical computer, in the sense that no algorithm is known to solve the problem using resources polynomial in the *O(L) bits needed to specify the* problem, where *L* is the number of bits needed to specify *N.*

$$L \equiv \lceil \log(N) \rceil$$

Quantum Computation- Quantum Algorithms-Application: order-finding

- In this section we explain how phase estimation may be used to obtain an efficient quantum algorithm for order-finding.
- The quantum algorithm for order-finding is just the phase estimation algorithm applied to the unitary operator

$$U|y\rangle \equiv |xy(\mathrm{mod}\ N)\rangle$$

- with $y \in \{0, 1\}^L$

Quantum Computation- Quantum Algorithms-Application: order-finding

- (Note that here and below, when $N \leq y \leq 2^L - 1$, *we use the* convention that *xy(mod N) is just y again.*
- That is, *U only acts non-trivially when* $0 \leq y \leq N - 1$*.) A simple calculation shows that the states defined by*

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

Quantum Computation- Quantum Algorithms-Application: order-finding

- for integer $0 \leq s \leq r-1$ *are eigenstates of U, since*

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^{k+1} \bmod N\rangle$$

$$= \exp\left[\frac{2\pi i s}{r}\right] |u_s\rangle \, .$$

- Using the phase estimation procedure allows us to obtain, with high accuracy, the corresponding eigenvalues $\exp(2\pi is/r)$, *from which we can obtain the order r with a little* bit more work.

Quantum Computation- Quantum Algorithms-Application: order-finding

- Two important requirements:
- The first requirement is satisfied by using a procedure known as *modular exponentiation, with which we* can implement the entire sequence of controlled-$U^{2^{\wedge}j}$ *operations applied by the phase* estimation procedure using $O(L^3)$ *gates (Please see Nielsen & Chuang)*

Quantum Computation- Quantum Algorithms-Application: order-finding

- The second requirement is a little tricker: preparing $|u_s\rangle$ requires that we know r, so this is out of the question. Fortunately, there is a clever observation which allows us to circumvent the problem of preparing $|u_s\rangle$, which is that

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

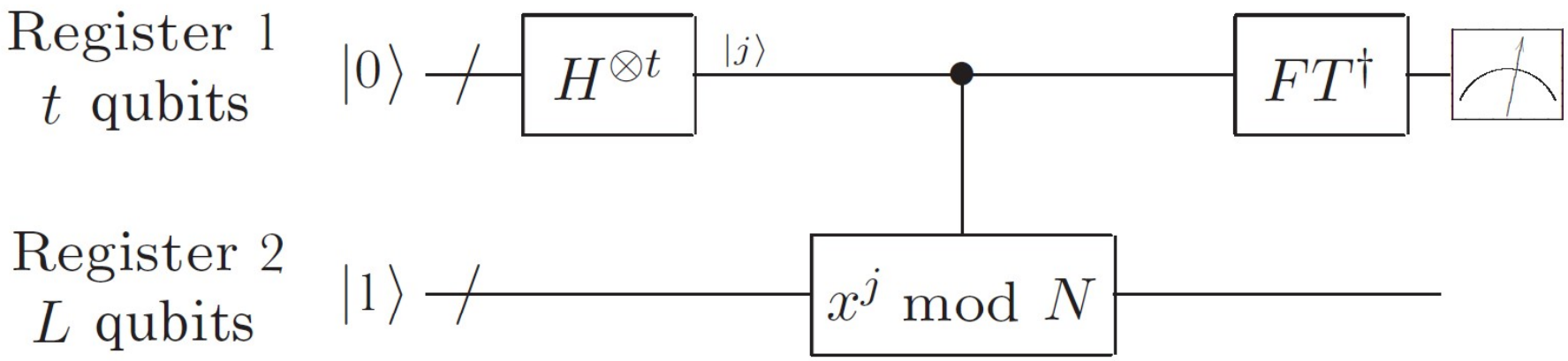
Quantum Computation- Quantum Algorithms-Application: order-finding

- In performing the phase estimation procedure, if we use **t** qubits in the first register, where

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

- and prepare the second register in the state |1> – which is trivial to construct – it follows that for each s in the range 0 through r − 1, we will obtain an estimate of the phase φ ≈ s/r accurate to 2L + 1 bits, with probability at least (1−ϵ)/r.

Quantum Computation- Quantum Algorithms-Application: order-finding



Quantum circuit for the order-finding algorithm. The second register is shown as being initialized to the |1> state.

Quantum Computation- Quantum Algorithms-Application: order-finding

- **The continued fraction expansion**
  - The reduction of order-finding to phase estimation is completed by describing how to obtain the desired answer, r, from the result of the phase estimation algorithm, $\varphi \approx s/r$.
  - We only know φ to $2L + 1$ bits, but we also know a priori that it is a rational number– the ratio of two bounded integers – and if we could compute the nearest such fraction to φ we might obtain r.

Quantum Computation- Quantum Algorithms-Application: order-finding

- Remarkably, there is an algorithm which accomplishes this task efficiently, known as the *continued fractions algorithm.*
- The reason this algorithm satisfies our needs is the following theorem:
- *Theorem: Suppose s/r is a rational number such that*

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

- Then *s/r is a convergent of the continued fraction for ϕ, and thus can be* computed in $O(L^3)$ *operations using the continued fractions algorithm.*

Quantum Computation- Quantum Algorithms-Application: order-finding

- Since φ is an approximation of s/r accurate to 2L + 1 bits, it follows that $|s/r - \varphi| \leq 2^{-2L-1} \leq 1/2r^2$, since $r \leq N \leq 2^L$. Thus, the theorem applies.
- Summarizing, given φ the continued fractions algorithm efficiently produces numbers s and r. The number r is our candidate for the order. We can check to see whether it is the order by calculating x^r mod N, and seeing if the result is 1. If so, then r is the order of x modulo N, and we are done!

Quantum Computation- Quantum Algorithms-Application: order-finding

- Procedure:

1.	$\|0\rangle\|1\rangle$	initial state
2.	$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|1\rangle$	create superposition
3.	$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|x^j \bmod N\rangle$	apply $U_{x,N}$
	$\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} \|j\rangle\|u_s\rangle$	
4.	$\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \|\widetilde{s/r}\rangle\|u_s\rangle$	apply inverse Fourier transform to first register
5.	$\rightarrow \widetilde{s/r}$	measure first register
6.	$\rightarrow r$	apply continued fractions algorithm

Quantum Computation- Quantum Algorithms-Application: Factoring

- Given a positive composite integer *N, what prime numbers when multiplied together* equal it? This *factoring problem turns out to be equivalent to the order-finding problem* we just studied.

Quantum Computation- Quantum Algorithms-Application: Factoring

- The reduction of factoring to order-finding proceeds in two basic steps.
  - The first step is to show that we can compute a factor of N if we can find a non-trivial solution $x \neq \pm 1 (\text{mod } N)$ to the equation $x^2 = 1 (\text{mod } N)$.
  - The second step is to show that a randomly chosen y co-prime to N is quite likely to have an order r which is even, and such that $y^{r/2} \neq \pm 1 (\text{mod } N)$, and thus $x \equiv y^{r/2} (\text{mod } N)$ is a non-trivial solution to $x^2 = 1 (\text{mod } N)$.
- These two steps are embodied in the following theorems→

Quantum Computation- Quantum Algorithms-Application: Factoring

- **Theorem 1:**
- Suppose N is an L bit composite number, and x is a non-trivial solution to the equation $x^2 = 1 (\text{mod } N)$ in the range $1 \leq x \leq N$, that is, neither $x = 1 (\text{mod } N)$ nor $x = N - 1 = -1 (\text{mod } N)$. Then at least one of $\gcd(x - 1, N)$ and $\gcd(x + 1, N)$ is a non-trivial factor of N that can be computed using $O(L^3)$ operations.

Quantum Computation- Quantum Algorithms-Application: Factoring

- **Theorem 2:**
- Suppose $N = p^{\alpha 1}_1 \ldots p^{\alpha m}_m$ *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*

$$p(r \text{ is even and } x^{r/2} \neq -1 (\text{mod } N)) \geq 1 - \frac{1}{2^m}$$

Quantum Computation- Quantum Algorithms-Application: Factoring

- Algorithm: Reduction of factoring to order-finding
  - Inputs: A composite number N
  - Outputs: A non-trivial factor of N.
  - Runtime: $O((\log N)^3)$ operations. Succeeds with probability O(1).

Quantum Computation- Quantum Algorithms-Application: Factoring

- Procedure:

- 1. If N is even, return the factor 2.
- 2. Determine whether $N = a^b$ for integers $a \geq 1$ and $b \geq 2$, and if so return the factor a (uses the classical algorithm which you can find in Nielsen & Chuang).
- 3. Randomly choose x in the range 1 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 modulo N.
- 5. If r is even and $x^{r/2} \neq -1 (\text{mod } N)$ 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.

Quantum Computation- Quantum Algorithms-Application: Factoring

- Exercise:
- Suppose we wish to factor N = 91. Confirm that steps 1 and 2 are passed. For step 3, suppose we choose x = 4, which is co-prime to 91.