ITIS 6260/8260 Quantum Computing

Lecture 5: Quantum Fourier Transform and Shor's algorithm

Yongge Wang

UNC Charlotte, USA

February 19, 2019

Outline

Quantum computers

- A quantum computer contains n qubits.
- if qubits can only be in non-entangled state, then nothing more powerful could be achieved
- The important thing is that these qubits could be entangled. There could be potentially 2^n states, and we could run a function on all these inputs at the same time
- challenges in building quantum computers: how can we restrict many qubits in a controlled environments so that they will not have too much entanglement with outside world and they could sufficiently entangle with each other in a controlled way?

Shor's algorithm

- Shor's algorithm is a quantum algorithm for factoring a number N in $O((\log N)^3)$ time and $O(\log N)$ space
- The algorithm is probabilistic, and gives the correct answer with high probability, and the probability of failure can be decreased by repeating the algorithm
- Prototype: IBM Q (quantum cloud service) presents backend devices include two processors with 5 superconducting qubits (ibmqx2 and ibmqx4), one 16-qubit processor (ibmqx5) and one 20-qubit processor ($QS1_1$).
- IBM also announced that they have successfully built and tested a 20-qubit and a 50-qubit machine

Period

- Fermat's Little Theorem

$$x^{p-1} \equiv 1 \mod p$$

for all primes p and $x \in \{1, \cdots, p-1\}$.

- Euler's Theorem

$$x^{(p-1)(q-1)} \equiv 1 \mod pq$$

for all primes p, q and $gcd(x, pq) = 1$.

Overview

- Shor's algorithm idea: Miller (1976) showed that factorization could be reduced to finding the order of an element.
- order of x: the least r with $x^r = 1 \bmod N$
- Steps:
  - chooses random x
  - find its order r
  - compute $gcd(x^{r/2} - 1, N)$
  - since $(x^{r/2} - 1)(x^{r/2} + 1) = x^r - 1 = 0 \bmod N$, the process fails only if r is odd or $x^{r/2} = -1$. Thus high probability success.

Shor's Algorithm - Periodicity

- An important result from Number Theory:

$$F(a) = x^a \bmod N$$

  is a periodic function
- Choose N = 15 and x = 7 and we get the following:
  - $7^0 \bmod 15 = 1$
  - $7^1 \bmod 15 = 7$
  - $7^2 \bmod 15 = 4$
  - $7^3 \bmod 15 = 13$
  - $7^4 \bmod 15 = 1$

Shor's Algorithm - Outline

To factor an odd integer N:

- If N is even or in the format of p^i, then use conventional computer to factor it
- Choose an integer n such that $N^2 < 2^n < 2N^2$
- Choose a random x such that $GCD(x, N) = 1$
- Create two quantum registers that are entangled
  - Input register: n qubits
  - Output register: $n/2$ qubits

Shor's Algorithm - Preparing Data

- Use Hardamard transform to put the input register in the uniform superposition of states representing numbers a (mod q).
- Put the output register with all zeros
- This leaves the machine in state

$$\frac{1}{\sqrt{2^n}} \sum_{a=0}^{2^n-1} |a\rangle|0\cdots0\rangle$$

where a the input register of n qubits and $|0\cdots0\rangle$ is the output register

Shor's Algorithm - Computing x^a

- Design a quantum circuit U_f to map $|a\rangle|0\cdots 0\rangle$ to $|a\rangle|f(a)\rangle$ where $f(a) = x^a$
- We do not care about the acutal value of $f(a) = x^a$
- Now assume that we measure the output register $|f(a)\rangle$ and discsard the result
- What is left in the $|a\rangle$ register? It should be an equal superposition over all the possible a's that could have led to the observed value $f(a)$:

$$\frac{1}{\sqrt{L}}(|a\rangle + |a+s\rangle + \cdots + |a+(L-1)s\rangle)$$

  where s is the period
- But how can we get s?
- Any time we have a periodic signal and want to extract the period, we use Quantum Fourier Transform!

2^m-dimensional QFT

- 2^m-dimensional QFT is the $2^m \times 2^m$ matrix F_{2^m} defined by

$$F_{2^m}[i,j] = \omega^{ij}/\sqrt{2^m}$$

where $\omega = e^{2\pi i/2^m}$ is the 2^m-th root of unity. That is,

$$\mathrm{QFT}_{2^m} = \frac{1}{\sqrt{2^m}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{2^m-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(2^m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^m-1} & \omega^{2(2^m-1)} & \cdots & \omega^{(2^m-1)^2} \end{pmatrix}$$

2^m-dimensional QFT

- Examples

$$\mathrm{QFT}_2 = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\mathrm{QFT}_{2^2} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Fast Fourier Transform FFT

- For a $2^m \times 2^m$ matrix F and a quantum state $|\phi\rangle$ of m qubits, it takes 2^{2m} operations to compute $F|\phi\rangle$
- FFT computes $A|x\rangle$ in $O(m2^m)$ steps
- for F_4, if we move even columns (0 and 2) to the left, we got

$$F_4' = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{pmatrix} = \begin{pmatrix} H & AH \\ H & -AH \end{pmatrix}$$

where $A = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ is the phase shift operation

Fast Fourier Transform FFT

- Generally we have

$$F_{2^m} = \frac{1}{\sqrt{2}} \begin{pmatrix} F_{2^{m-1}} & AF_{2^{m-1}} \\ F_{2^{m-1}} & -AF_{2^{m-1}} \end{pmatrix}$$

where

$$A = \begin{pmatrix} 1 & & & \\ & \omega & & \\ & & \ddots & \\ & & & \omega^{2^{m-1}-1} \end{pmatrix}$$

- Thus there is an algorithm to implement $F_{2^m}|\phi\rangle$ in $O(m2^m)$ steps. By quantum circuit speed up, we can implement $F_{2^m}|\phi\rangle$ with a quantum circuit of $O(m^2)$ gates

Fast Fourier Transform FFT quantum circuits

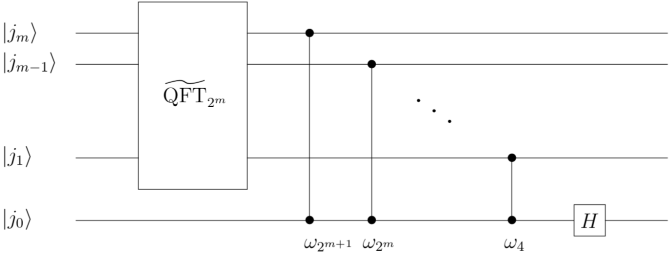
- Let us first define $\widetilde{\mathrm{QFT}}_{2^m}$ which is the same as QFT except with the output qubits in reverse order
- Specifically, if an integer $k \in \{0, \cdots, 2^m - 1\}$ is written in binary notation as $k_{m-1}k_{k-2}\cdots k_0$ then we define

$$\widetilde{\mathrm{QFT}}_{2^m}|j_{m-1}\cdots j_0\rangle = \frac{1}{\sqrt{2^m}}\sum_{k=0}^{2^m-1}\omega_{2^m}^{jk}|k_0k_1\cdots k_{m-1}\rangle$$

(cf. the j-th column of QFT_{2^m})

Fast Fourier Transform FFT quantum circuits

- $\widetilde{\mathrm{QFT}}_2|j\rangle$ is just the Hadamard transform
- For general $m \geq 2$, the following circuit computes $\widetilde{\mathrm{QFT}}_{2^{m+1}}$



Fast Fourier Transform FFT quantum circuits

- We next show that the quantum circuit in the previous slides compute

$$\widetilde{\mathrm{QFT}}_{2^{m+1}}|j_m j_{m-1}\cdots j_0\rangle = \frac{1}{\sqrt{2^{m+1}}}\sum_{k=0}^{2^{m+1}-1}\omega_{2^{m+1}}^{jk}|k_0 k_1\cdots k_m\rangle$$

- Let

$$\begin{array}{l} j' = j_m j_{m-1}\cdots j_1 \\ k' = k_{m-1}k_{m-2}\cdots k_0 \end{array}$$

- So $\widetilde{\mathrm{QFT}}_{2^m}$ maps $|j\rangle$ to

$$\frac{1}{\sqrt{2^m}}\sum_{k'=0}^{2^m-1}\omega_{2^m}^{j'k'}|k'_0 k'_1\cdots k'_{m-1}\rangle|j_0\rangle$$

Fast Fourier Transform FFT quantum circuits

- The controlled phase-shifts then transform this state to

$$\frac{1}{\sqrt{2^m}}\sum_{k'=0}^{2^m-1}\omega_{2^m}^{j'k'}\,\omega_{2^{m+1}}^{j_0k'_0}\,\omega_{2^m}^{j_0k'_1}\cdots\omega_{2^2}^{j_0k'_{m-1}}|k'_0k'_1\cdots k'_{m-1}\rangle|j_0\rangle$$

- By the fact that $\omega_N = \omega_{rN}^r$ for any r, N, we have

$$\frac{1}{\sqrt{2^m}}\sum_{k'=0}^{2^m-1}\omega_{2^{m+1}}^{2j'k'+j_0k'_0+2j_0k'_1+\cdots+2^{m-1}j_0k'_{m-1}}|k'_0k'_1\cdots k'_{m-1}\rangle|j_0\rangle$$
$$=\frac{1}{\sqrt{2^m}}\sum_{k'=0}^{2^m-1}\omega_{2^{m+1}}^{jk'}|k'_0k'_1\cdots k'_{m-1}\rangle|j_0\rangle$$

- Aftr the Hadamard transform, we get

$$\frac{1}{\sqrt{2^{m+1}}}\sum_{k'=0}^{2^m-1}\sum_{k_m=0}^{1}(-1)^{k_mj_0}\,\omega_{2^{m+1}}^{jk'}|k'_0k'_1\cdots k'_{m-1}\rangle|k_m\rangle$$

Fast Fourier Transform FFT quantum circuits

Since

$$(-1)^{k_m j_0} = (-1)^{k_m j} = \omega_{2^{m+1}}^{j(2^m k_m)},$$

the final state is

$$\begin{aligned} &\frac{1}{\sqrt{2^{m+1}}} \sum_{k'=0}^{2^m-1} \sum_{k_m=0}^{1} \omega_{2^{m+1}}^{jk'+j(2^m k_m)} |k'_0 k'_1 \cdots k'_{m-1}\rangle |k_m\rangle \\ &= \frac{1}{\sqrt{2^{m+1}}} \sum_{k=0}^{2^{m+1}-1} \omega_{2^{m+1}}^{jk} |k_0 k_1 \cdots k_m\rangle \end{aligned}$$

Total number of gates:

$$\begin{aligned} g(1) &= 1 \\ g(m+1) &= g(m) + m + 1 \\ g(m) &= \sum_{j=1}^{m} j = \binom{m+1}{2} \end{aligned}$$

Fast Fourier Transform FFT - QFT needed gates

- Specifically, QFT could be done by a sequence of simple quantum R_j and $S_{j,k}$ gates
- R_j is the Hardamard transform on the jth bit
- $S_{j,k}$ operates on the bits in positions j and k with $j < k$

$$S_{j,k} = \begin{array}{c|cccc} & |00\rangle & |01\rangle & |10\rangle & |11\rangle \\ \hline |00\rangle & 1 & 0 & 0 & 0 \\ |01\rangle & 0 & 1 & 0 & 0 \\ |10\rangle & 0 & 0 & 1 & 0 \\ |11\rangle & 0 & 0 & 0 & e^{i\theta_{k-j}} \end{array}$$

where $\theta_{k-j} = \pi/2^{k-j}$.

Fast Fourier Transform FFT - QFT process

- To perform a quantum Fourier transform, we apply the matrices in the order (from left to right)

$$R_{l-1}, S_{l-2,l-1}, R_{l-2}, S_{l-3,l-1}, \cdots, R_1, S_{0,l-1}, S_{0,l-2}, \cdots, S_{0,2}, S_{0,1}, R_0$$

- This will map the input state $|a\rangle$ to the state (cf, the a-th column of F_{2^n}):

$$\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \omega^{aj} |j\rangle$$

How to learn period s

- After measurement, we got a state like $\frac{1}{\sqrt{L}}(|r\rangle + |r+s\rangle + \cdots + |r+(L-1)s\rangle)$
- The QFT maps this state to

$$\frac{1}{\sqrt{2^n L}} \sum_{j=0}^{2^n-1} \sum_{l=0}^{L-1} \omega^{(r+ls)j} |j\rangle$$

- Easy case $s|2^n$ and the general case $s \not| 2^n$

How to learn period s: easy case $s|2^n$

- For $\frac{1}{\sqrt{2^n L}}\sum_{j=0}^{2^n-1}\sum_{l=0}^{L-1}\omega^{(r+ls)j}|j\rangle$, which j can be observed?
- Ignore the global phase ω^r and just look at $\sum_{l=0}^{L-1}\omega^{jsl}$
- If $2^n \not| js$, then $\omega^{js}, \omega^{2js}, \omega^{3js}, \cdots$ point in different directions in the complex plane. Thus the destructive interference will cancel each other out.
- If $2^n|js$ or $j = k2^n/s$ for some k, then $\omega^{js}, \omega^{2js}, \omega^{3js}, \cdots$ point to the same direction and produce constructive interference.
- Repeat this for several times, get a list of j's of multiples of $2^n/s$: $j_1 = k_1 2^n/s, j_2 = k_2 2^n/s, \cdots$
- Use GCD to get $2^n/s$ and then s.

How to learn period s: harder case $s \nmid 2^n$

- For each state $|j\rangle$ we still have $\sum_{l=0}^{L-1} \omega^{jsl}$. Whether we can observe $|j\rangle$, it depends on whether the sum involves constructive or destructive interference
- Whether $j = \lfloor k\frac{2^n}{s} \rceil$? That is, whether j is the nearest integer to some multiple of $\frac{2^n}{s}$.

Harder case $s \not| 2^n$ and $j = \lfloor k\frac{2^n}{s} \rceil$

- Assume that $j = k\frac{2^n}{s} + \varepsilon$ for a small $\varepsilon \leq \frac{1}{10}$
- Ignoring normalization, the final amplitude of basis state j has the form

$$\sum_{l=0}^{L-1} \omega^{\left(k\frac{2^n}{s}+\varepsilon\right)sl} = \sum_{l=0}^{L-1} \omega^{k2^n l}\omega^{\varepsilon sl} = \sum_{l=0}^{L-1} \omega^{\varepsilon sl}$$

since

$$\omega^{k2^n l} = e^{(2\pi i/2^n)(k2^n l)} = e^{2\pi ikl} = 1$$

- This amounts to a rotation around an ε fraction of the unit circle. It will mostly be constructive interference

Harder case $s \not| 2^n$ and $j \neq \lfloor k\frac{2^n}{s} \rceil$

- Assume that j is not the nearest integer to a multiple of $2^n/s$
- In this case, as we cary l, ω^{jsl} will loop all the way around the unit circle one or more times
- We get mostly destructive interference

Harder case $s \not| 2^n$: continued fractions

- We run the algorithm and get integers $j_1 = \lfloor k_1 \frac{2^n}{s} \rceil$, $j_2 = \lfloor k_2 \frac{2^n}{s} \rceil$, etc.
- Whenever an outcome j is observed, we'd like to determine whether it's close to an integer multiple of $2^n/s$, and if so what the multiple is?
- Assume that $j = k\frac{2^n}{s} \pm \varepsilon$ for a small ε, then

$$\left| \frac{j}{2^n} - \frac{k}{s} \right| \leq \frac{\varepsilon}{2^n}$$

- That is, $\frac{j}{2^n}$ is close to a rational number $\frac{k}{s}$ with smaller denominator s (note that $s \leq N$ and $Q \sim N^2$)

Harder case $s \not| 2^n$: continued fractions

- $\left|\frac{j}{2^n} - \frac{k}{s}\right| \leq \frac{\varepsilon}{2^n}$
- Assume that $\frac{j}{2^n} \pm \varepsilon = \frac{25001}{100000}$
- expand the input as a continued fraction:

$$\frac{25001}{100000} = \frac{1}{\frac{100000}{25001}} = \frac{1}{3 + \frac{24997}{25001}} = \frac{1}{3 + \frac{1}{\frac{25001}{24991}}} = \frac{1}{3 + \frac{1}{1 + \frac{4}{24997}}}$$

- $\frac{4}{24997}$ is smaller enough to be discarded
- So $\frac{k}{s} = \frac{1}{3 + \frac{1}{1}} = \frac{1}{4}$?
- What happens if k and s have a non-trivial divisor?

Harder case $s \not| 2^n$: continued fractions

- We run the algorithm multiple times to get $\frac{k_1}{s_1}, \frac{k_2}{s_2}, \cdots$. That is, we goet a list of $s_1, s_2, \cdots$
- With high probability, s is the least common multiple of $s_1, s_2, \cdots$

Generalizing of Shor's algorithm

- No success to generalize it to non-abelian groups
- if one can generalize Shor's algorithm to non-abelian groups, then
  - one can solve Graph Isomorphisms in polynomial time
  - Ragev (2005) showed that one could break lattice-based cryptosystems if one could generalize Shor's algorithm to work for a nonabelian group called the dihedral group.

Shor's Algorithm - An example

To factor an odd integer N (Let's choose 15):

- If N is even or in the format of p^i, then use conventional computer to factor it
- Choose an integer $q = 2^n$ such that $N^2 < q < 2N^2$ let's pick $256 = 2^8$
- Choose a random x such that $GCD(x, N) = 1$ let's pick 7
- Create two quantum registers that are entangled
  - Input register: must contain enough qubits to represent numbers as large as $q - 1$ up to 255, so we need 8 qubits
  - Output register: must contain enough qubits to represent numbers as large as $N - 1$ up to 14, so we need 4 qubits

Shor's Algorithm - Preparing Data

- Put the input register in the uniform superposition of states representing numbers $a \pmod{q}$.
- Put the output register with all zeros
- This leaves the machine in state

$$\frac{1}{\sqrt{256}} \sum_{a=0}^{255} |a\rangle|0000\rangle$$

where a the input register of 8 qubits and $|0000\rangle$ is the output register of 4 qubits

Shor's Algorithm - Modular Arithmetic

- Next we compute x^a (mod N) in the output register (uses only CCN gates). Thus the machine will be in the state

$$\frac{1}{\sqrt{256}} \sum_{a=0}^{255} |a\rangle |x^a \bmod N\rangle$$

- we are using decimal numbers for simplicity

Input Register	7^a mod 15	**Output Register**
$\|0\rangle$	7^0 mod 15	1
$\|1\rangle$	7^1 mod 15	7
$\|2\rangle$	7^2 mod 15	4
$\|3\rangle$	7^3 mod 15	13
$\|4\rangle$	7^4 mod 15	1
$\|5\rangle$	7^5 mod 15	7
$\|6\rangle$	7^6 mod 15	4
$\|7\rangle$	7^7 mod 15	13

Shor's Algorithm - Measuring

- After we measure the output register, it will collapse to one of the following:
$$|1\rangle, |4\rangle, |7\rangle, |13\rangle$$
- As an example, we show the case for $|1\rangle$
- Since the output register collapsed to $|1\rangle$, the input register will partially collapse to:

$$\frac{1}{\sqrt{64}}|0\rangle + \frac{1}{\sqrt{64}}|4\rangle + \frac{1}{\sqrt{64}}|8\rangle + \frac{1}{\sqrt{64}}|12\rangle + \cdots + \frac{1}{\sqrt{64}}|252\rangle$$

The probabilities in this case are $\frac{1}{\sqrt{64}}$ since our register is now in an equal superposition of 64 values $(0, 4, 8, 12, 16, 20, \cdots, 252)$

Shor's Algorithm - QFT

- Let A be the set of all values that 7^a mod 15 yielded 1. In our case $A = \{0, 4, 8, \cdots, 252\}$ and

$$\frac{1}{\sqrt{64}} \sum_{a \in A} |a\rangle |1\rangle$$

- The QFT maps a state $|a\rangle$ to the state

$$\frac{1}{\sqrt{256}} \sum_{c=0}^{255} \omega^{ac} |c\rangle$$

- So the final state of the input register after the QFT is:

$$\frac{1}{\sqrt{64}} \sum_{a \in A} \frac{1}{\sqrt{256}} \sum_{c=0}^{255} \omega^{ac} |c\rangle |1\rangle$$

Shor's Algorithm - QFT

- The QFT will essentially peak the probability amplitudes at integer multiples of q/r , where r is the order of x. In our case r is 4.

$$|0\rangle, |64\rangle, |128\rangle, |192\rangle, \cdots$$

- So we no longer have an equal superposition of states, the probability amplitudes of the above states are now higher than the other states in our register.
- Measure the state of register one, call this value c. Then c has a very high probability of being a multiple of q/r
- With our knowledge of q, and m, there are methods of calculating the the order r

Shor's Algorithm - The Factors

Now that we have the period, the factors of N can be determined by taking the greatest common divisor of N with respect to $x^{(P/2)} + 1$ and $x^{(P/2)} - 1$. The idea here is that this computation will be done on a classical computer.

- We compute
- $GCD(7^{4/2} + 1, 15) = 5$
- $GCD(7^{4/2} - 1, 15) = 3$
- We have successfully factored 15!

The phase estimation problem

- Assuming that we habe a quantum circuit Q acting on n qubits
- Associated with Q is a $2^n \times 2^n$ unitary matrix U
- When n is large, hard to write down U
- Since U is unitary, it has complete, orthonormal collection of eigenvectors

  $$|\phi_1\rangle, \cdots, |\phi_{2^n}\rangle$$

  and associate eigenvalues

  $$e^{2\pi i\theta_1}, \cdots, e^{2\pi i\theta_{2^n}}$$

The phase estimation problem

- **Input**: A quantum circuit Q that performs a unitary operation U, along with a quantum state $|\phi\rangle$

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

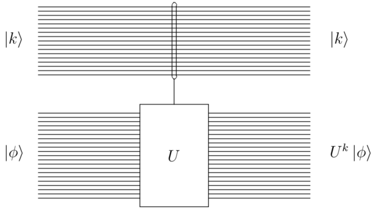
- **Output**: An approximation to $\theta \in [0, 1)$

The phase estimation procedure

- Let $\Lambda_m(U)$ dente the unitary transformation on $m+n$ qubits

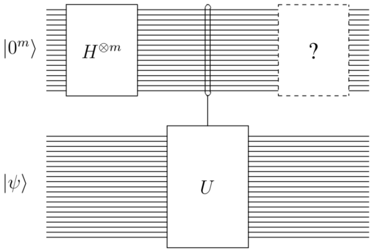
$$\Lambda_m(U)|k\rangle|\phi\rangle = |k\rangle(U^k|\phi\rangle)$$

where $k \in \{0, \cdots, 2^m - 1\}$



- If k is larger, then one may needs exponential time (e.g., 2^m) to implement $\Lambda_m(U)$ from U. But we will assume that $\Lambda_m(U)$ could be efficiently implemented

The phase estimation procedure



- After $H^{\otimes m}$, we get $\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} |k\rangle|\phi\rangle$
- After $\Lambda_m(U)$, we get $\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} |k\rangle(U^k|\phi\rangle)$

The phase estimation procedure

- $|\phi\rangle$ is an eigenvector of U

$$U^k|\phi\rangle = e^{2\pi ik\theta}|\phi\rangle$$

- That is, after $\Lambda_m(U)$, we get

$$\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}|k\rangle(e^{2\pi ik\theta}|\phi\rangle) = \frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}e^{2\pi ik\theta}|k\rangle|\phi\rangle$$

- If we discard the last n qubits, we get

$$\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}e^{2\pi ik\theta}|k\rangle$$

The phase estimation procedure: $\theta = \frac{j}{2^m}$

- If $\theta = \frac{j}{2^m}$ for some $j \leq 2^m - 1$. Then we have

$$\frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} e^{2\pi i \frac{jk}{2^m}} |k\rangle = \frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} \omega^{jk} |k\rangle$$

  where $\omega = e^{2\pi i/2^m}$ is the 2^m-th root of unit
- Let $|\psi_j\rangle = \frac{1}{2^{m/2}} \sum_{k=0}^{2^m-1} \omega^{jk} |k\rangle$
- We know that the first m qubits is in one of the state $|\psi_j\rangle$ and we need to determine which one it is. First we show that $|\psi_0\rangle, \cdots, |\psi_{2^m-1}\rangle$ are orthogonal

The phase estimation procedure: $\theta = \frac{j}{2^m}$

- Note that

$$\langle \psi_j | \psi_{j'} \rangle = \frac{1}{2^m} \sum_{k=0}^{2^m-1} \omega^{k(j-j')} = \frac{1}{2^m} \sum_{k=0}^{2^m-1} \left(\omega^{j-j'} \right)^k$$

- By the fact that $\sum_{k=0}^{2^m-1} x^k = \frac{x^{2^m}-1}{x-1}$ and $\omega^{2^m} = 1$, we have

$$\langle \psi_j | \psi_{j'} \rangle = 1 \text{ iff } j = j' (\text{and } = 0) \text{ otherwise}$$

- Since $|\psi_0\rangle, \cdots, |\psi_{2^m-1}\rangle$ are orthogonal, there is a unitary transformation F with

$$F|j\rangle = |\psi\rangle$$

The phase estimation procedure: $\theta = \frac{j}{2^m}$

- The matrix F can be explicitly described by allowing the vector $|\psi_j\rangle$ to determine the j-th column of F

$$F = \frac{1}{\sqrt{2^m}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{2^m-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(2^m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^m-1} & \omega^{2(2^m-1)} & \cdots & \omega^{(2^m-1)^2} \end{pmatrix}$$

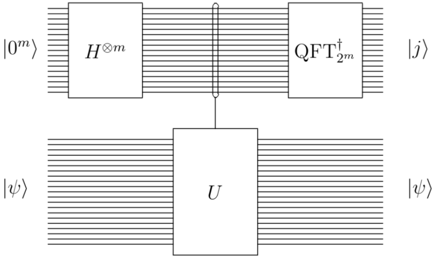
- This is exactly the discrete Fourier transform and we can denote it as QFT_{2^m}

The phase estimation procedure: $\theta = \frac{j}{2^m}$

We can write this as

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

Plug the inverse of QFT_{2^m} into the previous circuit:



Thus, measure the first m qubits and divide it by 2^m to get θ.

The phase estimation procedure: general value of θ

- The state of the first m qubits before the inverse of QFT is

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

- Applying the inverse $\mathrm{QFT}^*_{2^m}$ to the m qubits and we get

$$\frac{1}{2^m} \sum_{k=0}^{2^m-1} \sum_{j=0}^{2^m-1} e^{2\pi i (k\theta - kj/2^m)} |j\rangle = \sum_{j=0}^{2^m-1} \left(\frac{1}{2^m} \sum_{k=0}^{2^m-1} e^{2\pi i k(\theta - j/2^m)} \right) |j\rangle$$

- The probability to get outcome j is

$$p_j = \left| \frac{1}{2^m} \sum_{k=0}^{2^m-1} e^{2\pi i k(\theta - j/2^m)} \right|^2$$

The phase estimation procedure: general value of θ

- Since $\theta \neq j/2^m$, assume that $e^{2\pi ik(\theta - j/2^m)} \neq 1$
- Use the fact that $\sum_{k=0}^{2^m-1} x^k = \frac{x^{2^m}-1}{x-1}$, we get

$$p_j = \frac{1}{2^{2m}} \left| \frac{e^{2\pi i(2^m\theta - j)} - 1}{e^{2\pi i(\theta - j/2^m)} - 1} \right|^2$$

- Our goal will be to show that the probability p_j is large for values of j that satisfy $j/2^m \sim \theta$ and small otherwise

The phase estimation procedure: general value of θ

- Assume that $\theta = \frac{j}{2^m} \pm \varepsilon$
- let

$$a = |e^{2\pi i(2^m\theta - j)} - 1| = |e^{2\pi i\varepsilon 2^m} - 1|$$
$$b = |e^{2\pi i(\theta - j/2^m)} - 1| = |e^{2\pi i\varepsilon} - 1|$$

so

$$p_j = \frac{1}{2^{2m}}\frac{a^2}{b^2}$$

- To get a lower bound for p_j, we need to get a lower bound for a and an upper bound for b

The phase estimation procedure: general value of θ

- Assume that $\theta = \frac{j}{2^m} \pm \varepsilon$
- let

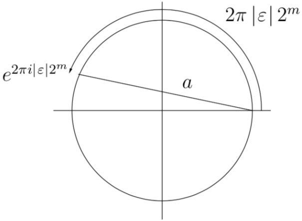
$$a = |e^{2\pi i(2^m\theta - j)} - 1| = |e^{2\pi i \varepsilon 2^m} - 1|$$
$$b = |e^{2\pi i(\theta - j/2^m)} - 1| = |e^{2\pi i \varepsilon} - 1|$$

  so

$$p_j = \frac{1}{2^{2m}} \frac{a^2}{b^2}$$

- To get a lower bound for p_j, we need to get a lower bound for a and an upper bound for b

The phase estimation procedure: general value of θ



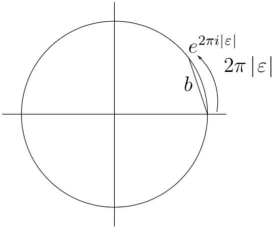
The ratio of the minor arc length to the chord length is at most $\pi/2$, so

$$\frac{2\pi\varepsilon 2^m}{a} \leq \frac{\pi}{2}$$

That is:

$$a \geq 4\varepsilon 2^m$$

The phase estimation procedure: general value of θ



the ratio of arc length to chord length is at least 1, so

$$\frac{2\pi\varepsilon}{b} \geq 1$$

That is:

$$b \leq 2\pi\varepsilon$$

The phase estimation procedure: general value of θ

Putting together, we get

$$p_j \geq \frac{1}{2^{2m}} \frac{16\varepsilon^2 2^{2m}}{2\pi^2\varepsilon^2} = \frac{4}{\pi^2} > 0.4$$

Q&A

Q&A?