A Quantum Algorithm for Integer Factorization

Teng Peng

Supervised by Prof.Ken Tsang

May 14th, 2014

Outline

What is Shor's algorithm?

A quantum algorithm for integer factorization

- Quantum algorithm
  - An algorithm that runs on quantum computers
- Integer factorization
  - Decomposition of a composite number into smaller non-trivial divisors
  - Example
    - Given 24 , find its prime factor
    - $24 = 2^3 \times 3$.
    - Prime factor is 2, 3

Why Shor's algorithm?

Faster

- RSA
  - Cryptosystem
  - Widely used for secure data transmission
  - Principle
    - Easy to multiply prime numbers $\rightarrow$ Encryption
    - Impossible to factor prime numbers $\rightarrow$ Decryption
- Why not classical algorithm?
  - Trial division
    - Find if n can be divided by each number in turn that is less than n
  - Too slow
    - Take 10^{176} years to factoring a 400-digit number.

Why Shor's algorithm?

Embodiment of human ingenuity

- Math
    - Euclidean algorithm 300 BC
    - Chinese remainder theorem 300-500 AD
    - Euler's theorem 1736
    - Fast Fourier transform(Cooley-Tukey) 1965
- Physics
    - Quantum physics 1900-Now
- Computer science
    - Divide & Conquer algorithm 1946

Overview

4 reductions of a complex problem

- Factoring is reduced to finding a nontrivial square root of 1 modulo N
- Computing the order of a random integer modulo N
- Find the period of a periodic superposition
- Found by quantum FFT

The tricks and secret of Shor's algorithm

- Tricks
    - Number theory
        - Classical computer
- Secret
    - Quantum FFT
        - Quantum algorithm

What is QFT?

- Fourier transform
- $\downarrow$ In discrete domain
- Discrete Fourier transform
- $\downarrow$ Plus Divide & Conquer algorithm
- Fast Fourier transform
- $\downarrow$ Modification for quantum computer
- Quantum Fourier transform
  - Quantum implementation of FFT

What is Fourier transform?

Fourier series & Fourier coeffcients

- In 1807, Fourier astounded some of his contemporaries by asserting that an "arbitrary" function could be expressed as a linear combination of sines and cosines.
- Amazingly, it's true.
- Fourier series.
  - $f(x) = \sum_{k=-\infty}^{\infty} c_k sinkx + \sum_{k=-\infty}^{\infty} c'_k coskx$
  - Apply Euler's identity $e^{ix} = cosx + isinx$
  - $f(x) = \sum_{-\infty}^{\infty} c_k e^{ikx}$
  - $c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx} dx$
    - c_k is called the k^{th} Fourier coefficient of $f(x)$

Does it work?

Two vital questions

- Question: Given any reasonable function $f(x)$ on $[-\pi, \pi]$, with Fourier coefficients defined above, is it true that

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}?$$

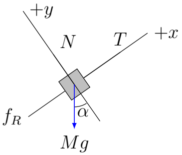
  - Yes

- Question: Are two functions with the same Fourier coefficients necessarily equal?
  - Yes

The idea behind Fourier Transform

Analysis & Synthesis

- Net force & Components force



- Fourier basis

$$\hat{\mathbf{v}}^1 = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \quad \hat{\mathbf{v}}^2 = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ \zeta^2 \\ \zeta^3 \\ \vdots \\ \zeta^{N-1} \end{pmatrix}, \quad \hat{\mathbf{v}}^3 = \frac{1}{\sqrt{N}} \left(\begin{matrix} 1 \\ \zeta^{2\cdot 2} \\ \zeta^{3\cdot 2} \\ \vdots \\ \zeta^{(N-1)\cdot 2} \end{matrix} \right.$$

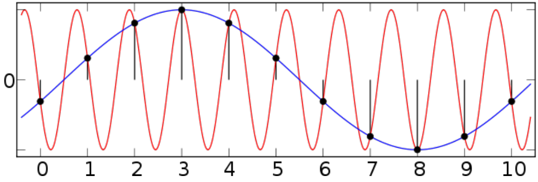
Discrete Fourier transform

What if we do not know F?

- Example: Given audio signals, continuous signals are sampled at discrete time intervals
- Question: Given sample points, how to find Fourier coefficients?

Consequence of sampling

- Aliasing



Discrete Fourier transform

Consequence of Aliasing

- We are allowed to represent $f(x)$ by a finite linear combination, which agrees on the sample points

$$f(x) \sim p(x)$$

$$f(x) = c_0 + c_1 e^{ix} + c_2 e^{2ix} + \ldots + c_{n-1} e^{(n-1)ix} = \sum_{k=0}^{n-1} c_k e^{ikx}$$

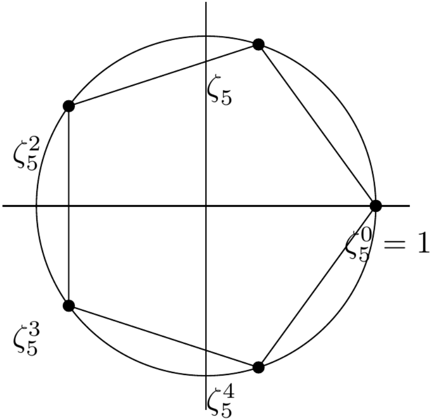
$$\mathbf{f} = c_0 \omega_{\mathbf{0}} + c_1 \omega_{\mathbf{1}} + \ldots + c_{n-1} \omega_{\mathbf{n-1}}$$

$$\omega_{\mathbf{k}} = (e^{ikx_0}, e^{ikx_1}, \ldots, e^{ikx_{n-1}})^T$$

$$\omega_{\mathbf{k}} = (1, \zeta_n^k, \zeta_n^{2k}, \ldots, \zeta_n^{(n-1)k})^T$$

Notation

- Notation $\zeta_m = \sqrt[m]{1}$
  - Fact $\zeta_m = \zeta_n^2$, when $n = 2m$
  - Example $\zeta_4 = \zeta_8^2$



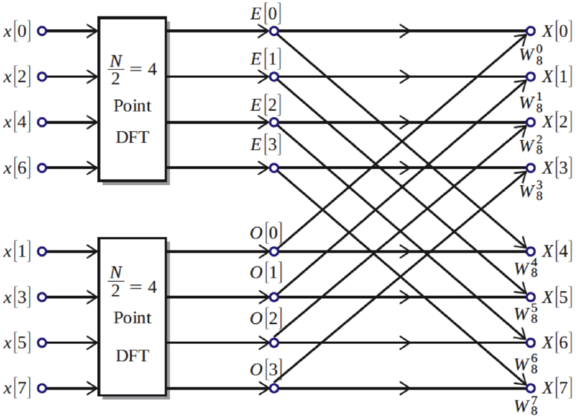
Fast Fourier transform

Mathematical approach

$$
\begin{aligned}
c_k &= \sum_{n=0}^{n-1} \zeta_N^{-nk} f_n \\
&= \sum_{n=0}^{N/2-1} \zeta_N^{2nk} f_{2n} + \sum_{n=0}^{N/2-1} \zeta_N^{k(2n+1)} f_{2n+1} \\
&= \sum_{n=0}^{N/2-1} \zeta_N^{2nk} f_{2n} + \zeta_N^{k} \sum_{n=0}^{N/2-1} \zeta_N^{2nk} f_{2n+1} \\
&= \sum_{n=0}^{N/2-1} \zeta_{N/2}^{nk} f_{2n} + \zeta_N^{k} \sum_{n=0}^{N/2-1} \zeta_{N/2}^{nk} f_{2n+1}
\end{aligned}
$$

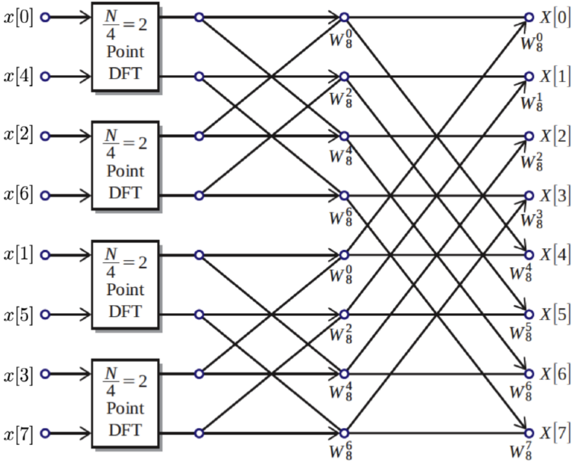
Fast Fourier transform

Visualization



Fast Fourier transform

Divide & Conquer



Qubits, superposition, measurement

Qubits & Superposition

- Ordinary bits
  - Electron
  - Ground state & excited state, 0 & 1
- Quantum bits
  - $|0\rangle$, $|1\rangle$
- Superposition
  - $\alpha|0\rangle + \beta|1\rangle$
- Measurement
  - Goal: determine which state
  - Outcome: 0 or 1
  - Disturbs the system

QFT?

QFT is quantum version of FFT

Why QFT?

- Extremely fast

What's the differences?

- FFT input: 2^m-dimensional complex-valued vector
- QFT input: A superposition of $log 2^m$ qbits
- FFT method: Multiply DFT matrix
- QFT method: Perform quantum operations
- FFT output: 2^m-dimensional complex-valued vector
- QFT output A random m-bit number j, from the probability distribution $Pr[j] = [\beta_j]^2$

QFT?

Why there are differences?

- A short answer: The mysterious principle of quantum world
- A longer answer: The way the data is represented physically
  - Qbits
  - Superposition
  - Measurement

Speed comparison

Big O notation

Definition 7. *Let f and g be two functions defined on some subset of the real number. One writes*

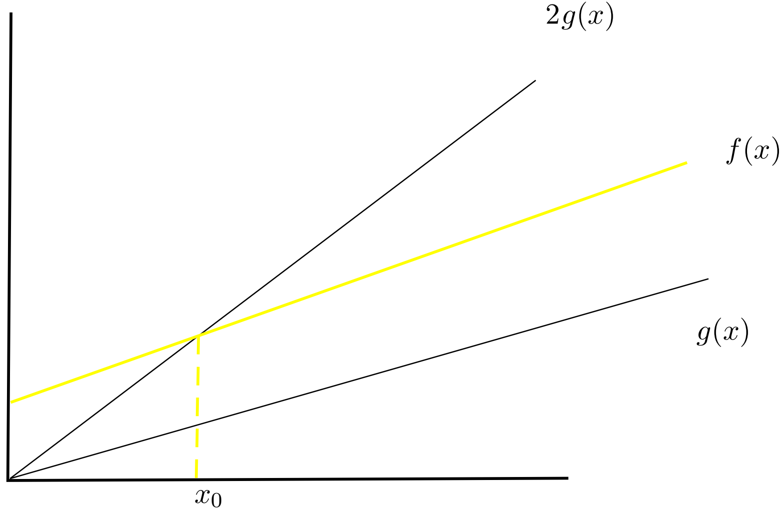
$$f(x) = O(g(x)) \text{ as } x \to \infty$$

if and only if

$$|f(x)| \leqslant M|g(x)| \text{ for all } x \geqslant x_0.$$

Where M is a positive constant, and x_0 is a real number.

Speed comparison



But how does a random number help?

Periodicity

- Input a periodical vector
- Output multiples of period
- Example
  - Input 100-dimensional vector with period 5.
    - 1,3,5,2,4,1,3,5. . . .5,2,4
  - Output 15, 20
    - $GCD(15, 20) = 5$

Shor's algorithm step by step

- Step 1. Choose a random positive integer m. Use Euclidean algorithm to compute common divisor $gcd(m, N)$ of m and N. If greatest common divisor $gcd(m.N) \neq 1$,then we have found a non-trivial factor of N. If, one the other hand, $gcd(m, N) = 1$, then proceed to step 2.
- Step 2(quantum part). Find the unknown period P.
- Step 3. If P is an odd integer, then go to step 1. If P is even, then proceed to Step 4.
- Step 4. $(m^{p/2} - 1)(m^{p/2} + 1) = m^p - 1 = 0 \bmod N$
    - If $m^{p/2} + 1 = 0 \bmod N$, then go to step 1. If $m^{p/2} + 1 \neq 0 \bmod N$, then proceed to step 5.
- Step 5 Use the Euclidean algorithm to compute $d = gcd(m^{p/2} - 1, N)$.

Quantum part in detail

Step 2.0 Initialize registers 1 2

$$|\psi_0\rangle = |0\rangle|1\rangle$$

Step 2.1 Apply QFT to reg_1

$$\begin{aligned}|\psi_0\rangle \to & |\psi_1\rangle \\ & = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} \omega^{0x}|x\rangle|0\rangle \\ & = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle|1\rangle\end{aligned}$$

Step 2.2 Apply linear transformation to the two register

$$|\psi_1\rangle \to |\psi_2\rangle \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle|f(x)\rangle$$

Step 2.3 Apply QFT to reg_1

$$|\psi_2\rangle \to |\psi_3\rangle = \frac{1}{Q} \sum_{x=0}^{Q-1} \sum_{y=0}^{Q-1} \omega^{xy}|y\rangle|f(x)\rangle$$

Step 2.4 Measure reg_1

Output a classical probability distribution on sample space $\{0, 1, 2, \ldots, \}$

$$P(y) = \begin{cases} 0 & \text{if } P \neq \bmod Q \\ \frac{1}{P} & \text{if } P = \bmod Q \end{cases}$$

A Working example

- Given $N = 91(= 7 * 13)$. Choose $Q = 2^{14} = 16384$.
- Step 1. Choose a random positive integer $m = 3$. Since $gcd(91, 3) = 1$, we proceed to step 2 to find the period of the function f given by $f(a) = 3^a \bmod 91$.
  - Unknown to us, f has period 6.
- Step 2. We get period 6 from the quantum part of the Shor's algorithm
- Step 3. Since 6 is an even number, we proceed to Step 4.
- Step 4. Since $3^{P/2} = 3^3 = 27 \neq 0 \bmod 91$, we go to Step 5.
- Step 5. With the Euclidean algorithm, we compute

$$gcd(3^{p/2} - 1, 91) = gcd(3^3 - 1, 91) = gcd(26, 91) = 13$$

- Exit. Output a non-trivial factor of $N = 91$, namely 13.

A Working example

`Step 2.0.` Initialize registers 1 and 2. Thus, the state of the two registers becomes

$$|\psi_0\rangle = |0\rangle|1\rangle$$

`Step 2.1` Apply quantum Fourier transform, which is

$$\frac{1}{\sqrt{16384}} \sum_{x=0}^{16383} \omega^{0x}|x\rangle,$$

to register 1. The ω is a primitive Q-th root of unity,

$$\omega = e^{\frac{2\pi i}{16384}}.$$

Thus the state of the two registers becomes

$$|\psi_1\rangle = \frac{1}{\sqrt{16384}} \sum_{x=0}^{16383} \omega^{0\cdot x}|x\rangle$$

`Step 2.2` Apply the unitary transformation U_f to registers 1 and 2, where

$$U_f|x\rangle = |x\rangle|f(x) - \ell \bmod 91\rangle.$$

Thus, the state of the two registers becomes

$$|\psi_2\rangle = \frac{1}{\sqrt{16384}} \sum_{x=0}^{16383} |x\rangle|3^x \bmod 91\rangle$$

A Working example

`Step 2.3` Apply the Q-point Fourier transform to register 1. Thus, the state of system becomes

$$|\psi_3\rangle = \frac{1}{\sqrt{16384}} \sum_{x=0}^{16383} \sum_{y=0}^{16383} \omega^{xy} |y\rangle |3^x \bmod 91\rangle$$

`step 2.4` Measure register 1. The result of our measurement just happens to turn out to be

$$y = 13453$$

Unknown to us, the probability of obtaining this particular y is

$$0.3189335551 \times 10^{-6}$$

`Step 2.5` For each non-trivial n in succession, we check to see if

$$3^{q_n} = 1 \bmod 91.$$

if this is the case, then we know $q_n = P$, and we immediately exit and proceed to step 3.

In this example, we find $P = 6$. Output $P = 6$.

Q & A