Week 7: Shor's Algorithm

COMS 4281 (Fall 2024)

Admin

1. Midterm, Monday October 21.
    - Two exam rooms: Havemeyer 209 or Hamilton 703.
    - **Plan on arriving at least 10 minutes early**
    - If you require additional exam accommodations and have documentation from student services, please contact me ASAP.
    - Exam skeleton will be released in a day or two.

Upcoming event

Today at 11:45 in Mudd 829: Applied Physics Colloquium by IBM Quantum.

Title: The Future of IBM Quantum: Pioneering the Next Era of Computing

Last time

Phase Estimation Algorithm (which uses DFT as subroutine).

Goal of PEA:

- Ability to run controlled versions of $U, U^2, U^4, \ldots, U^{2^j}, \ldots$
- An **eigenstate** $|\psi\rangle$ where $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$,

estimate θ.

Last time

Public key cryptography

RSA cryptosystem

Factoring problem

Input: Positive integer N.

Output: Prime factorization of N as $p_1^{a_1} p_2^{a_2} \cdots$.

Factoring problem

Input: Positive integer N.

Output: Prime factorization of N as $p_1^{a_1} p_2^{a_2} \cdots$.

The prime factorization of N is unique by the **Fundamental Theorem of Arithmetic**.

To find a factorization of N, it suffices to be able to find *some* nontrivial divisor of N.

Factoring problem

It is widely believed that Factoring is hard for classical computers. The best classical algorithm, known as the **General Number Field Sieve**, takes time roughly

$$\exp\left(O(\log N)^{1/3}\right).$$

This is essentially **exponential** in the number of digits of N.

Shor's algorithm

A quantum algorithm to solve Factoring in $\mathrm{poly}(\log N)$ steps.

Discovered by Peter Shor in 1993. He was inspired by Simon's Algorithm.

Shor's algorithm

A quantum algorithm to solve Factoring in $\mathrm{poly}(\log N)$ steps.

Discovered by Peter Shor in 1993. He was inspired by Simon's Algorithm.

Shor's Algorithm is also a hybrid classical-quantum algorithm.

1. **Classical part**: reduce the factoring problem to **order finding**.
2. **Quantum part**: solve order finding.

Order Finding

Input: given positive integers N, x such that

1. $1 \leq x < N$
2. $\gcd(N, x) = 1$ (i.e. they do not have any nontrivial factors in common)

Order Finding

Input: given positive integers N, x such that

1. $1 \leq x < N$
2. $\gcd(N, x) = 1$ (i.e. they do not have any nontrivial factors in common)

Output: find smallest integer r such that $x^r = 1 \bmod N$ (called the **order** of x mod N).

Quantum algorithm for Order Finding

Quantum Algorithm for Order Finding

Input: Integers N and $1 \leq x < N$ coprime to N.

Order Finding algorithm uses Phase Estimation Algorithm with respect to the **modular multiplication unitary** U_x, defined as

$$U_x |y\rangle = |xy \bmod N\rangle$$

where $0 \leq y < N$.

Modular multiplication unitary

$$U_x |y\rangle = |xy \bmod N\rangle$$

Fact 1: This map is unitary.

Modular multiplication unitary

$$U_x |y\rangle = |xy \bmod N\rangle$$

Fact 1: This map is unitary.

Fact 2: U_x is computable by a quantum circuit with $\mathrm{poly}(n)$ gates.

Modular multiplication unitary

Subtlety: it's easy to implement a quantum circuit for the map

$$V\,|x, y, 0\rangle = |x, y, xy \bmod N\rangle\,.$$

Modular multiplication unitary

Subtlety: it's easy to implement a quantum circuit for the map

$$V |x, y, 0\rangle = |x, y, xy \bmod N\rangle .$$

This is different from U_x. The proof that U_x is unitary uses the fact that $gcd(x, N) = 1$.

Modular multiplication unitary, repeated

$$U_x^{2^j} |y\rangle = |x^{2^j} y \bmod N\rangle$$

Fact: This map is unitary, and is computable by a quantum circuit with $\mathrm{poly}(n, j)$ gates.

Modular multiplication unitary, repeated

$$U_x^{2^j} |y\rangle = |x^{2^j} y \text{ mod } N\rangle$$

Fact: This map is unitary, and is computable by a quantum circuit with $\mathrm{poly}(n, j)$ gates.

This uses fact that one can "shortcut" compute x^{2^j} modulo N without doing 2^j multiplications, by repeatedly squaring, reducing mod N, squaring, reducing mod N, etc...

$$x \to x^2 \to x^2 \text{ mod } N \to (x^2 \text{ mod } N)^2 \to x^4 \text{ mod } N \to \cdots$$

Remember that we're trying to use Phase Estimation Algorithm.

We have efficient way to implement controlled $U_x^{2^j}$ operations.

Remember that we're trying to use Phase Estimation Algorithm.

We have efficient way to implement controlled $U_x^{2^j}$ operations.

We just need an eigenvector of U_x.

Eigenvectors of U_x

Let r denote the **order** of x (i.e. $x^r = 1 \bmod N$). Then for all $0 \leq s < r$, define the state

$$|v_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \omega_r^{-sk} \left|x^k \bmod N\right\rangle$$

Eigenvectors of U_x

Let r denote the **order** of x (i.e. $x^r = 1 \bmod N$). Then for all $0 \leq s < r$, define the state

$$|v_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \omega_r^{-sk} \left|x^k \bmod N\right\rangle$$

Claim: $U_x |v_s\rangle = \exp\left(2\pi i \frac{s}{r}\right) |v_s\rangle$.

Phase Estimation Algorithm

Given:

1. Ability to run controlled versions of $U, U^2, U^4, \ldots, U^{2^j}, \ldots$
2. An **eigenstate** $|\psi\rangle$ where $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$

Using $O(t)$ qubits of ancilla, PEA outputs an estimate $\widetilde{\theta}$ satisfying

$$|\widetilde{\theta} - \theta| \leq 2^{-t}$$

with high probability. The algorithm runs in $\mathrm{poly}(t)$ time.

Quantum Algorithm for Order Finding

Input: Integers N, x where $2^{n-1} \le N < 2^n$ and $1 \le x < N$ coprime to N.

Suppose we run Phase Estimation algorithm with $O(n)$ ancilla qubits, with controlled-$U_x^{2^j}$ operations and an eigenvector $|v_s\rangle$ for some $0 \le s < r$.

Quantum Algorithm for Order Finding

Input: Integers N, x where $2^{n-1} \leq N < 2^n$ and $1 \leq x < N$ coprime to N.

Suppose we run Phase Estimation algorithm with $O(n)$ ancilla qubits, with controlled-$U_x^{2^j}$ operations and an eigenvector $|v_s\rangle$ for some $0 \leq s < r$.

Complexity: $\mathrm{poly}(n)$ (including complexity of controlled-$U_x^{2^j}$).

Output: estimate $\widetilde{\theta}_s$ that is within $2^{-3n} \leq 1/N^3$ of s/r.

Remember that the goal is to recover the integer r, the order of x.

Remember that the goal is to recover the integer r, the order of x.

Issue 1: The algorithm outputs something that looks like

$$\widetilde{\theta} = 0.011110110011\ldots$$

We don't know s, r. How do find the s/r fraction corresponding to this?

Remember that the goal is to recover the integer r, the order of x.

Issue 1: The algorithm outputs something that looks like

$$\widetilde{\theta} = 0.011110110011\ldots$$

We don't know s, r. How do find the s/r fraction corresponding to this?

Issue 2: How do we get our hands on the eigenvector $|v_s\rangle$?

Getting the eigenvectors

We can solve Issue 2 by running Phase Estimation on the superposition

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

Getting the eigenvectors

We can solve Issue 2 by running Phase Estimation on the superposition

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

While it's hard to construct $|v_s\rangle$ individually, this superposition is easy to prepare, because this is equal to the standard basis state $|1\rangle$. **(exercise!)**

After running PEA on input $|1\rangle$, the output will be approximately

$$\approx \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle \otimes |\widetilde{\theta}_s\rangle$$

where $|\widetilde{\theta}_s - s/r| \leq 2^{-3n} \leq 1/N^3$.

After running PEA on input $|1\rangle$, the output will be approximately

$$\approx \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle \otimes |\widetilde{\theta}_s\rangle$$

where $|\widetilde{\theta}_s - s/r| \leq 2^{-3n} \leq 1/N^3$.

Measuring the second register yields $|\widetilde{\theta}_s\rangle$ for a uniformly random $0 \leq s < r$.

Imagine that $\widetilde{\theta}_s$ was actually s/r *exactly*, and furthermore s was coprime to r.

Then we can recover s, r from $\widetilde{\theta}_s$.

Imagine that $\widetilde{\theta}_s$ was actually s/r *exactly*, and furthermore s was coprime to r.

Then we can recover s, r from $\widetilde{\theta}_s$.

Example: Suppose $\widetilde{\theta}_s = 0.358$. Then clearly

$$\widetilde{\theta}_s = \frac{358}{1000} = \frac{179}{500}.$$

Imagine that $\widetilde{\theta}_s$ was actually s/r *exactly*, and furthermore s was coprime to r.

Then we can recover s, r from $\widetilde{\theta}_s$.

Example: Suppose $\widetilde{\theta}_s = 0.358$. Then clearly

$$\widetilde{\theta}_s = \frac{358}{1000} = \frac{179}{500}.$$

This is equal to s/r. But since s/r is already in most reduced form, it must be $s = 179$ and $r = 500$.

However $\widetilde{\theta}_s$ is not s/r exactly.

We solve **Issue 1** by the **Continued Fractions Algorithm**.

Continued fractions algorithm

This is a classical algorithm from number theory.

Let φ be a real number and s/r a fraction such that

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

Then the Continued Fractions Algorithm, given input φ, will output the reduced form of s/r in time $\mathrm{poly}(\log r)$.

We can apply Continued Fractions to our setting because we have

$$\left|\widetilde{\theta}_s - \frac{s}{r}\right| \leq \frac{1}{N^3}$$

which is less than $\frac{1}{2r^2}$. So Continued Fractions outputs s/r in reduced form. If s is coprime to r, then we get s, r exactly.

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U_x (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U_x (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U_x (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.
3. Use Continued Fractions algorithm on $\widetilde{\theta}$ to obtain reduced form a/b of s/r.

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U_x (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.
3. Use Continued Fractions algorithm on $\widetilde{\theta}$ to obtain reduced form a/b of s/r.
4. Check whether $x^b = 1 \bmod N$. If so, then $b = r$. Otherwise, go back to Step 1.

Summary of Order Finding algorithm

1. Run Phase Estimation with the unitary controlled-U_x (and its powers) and input state $|1\rangle$ (which is uniform superposition of eigenvectors).
2. Get random estimate $\widetilde{\theta}$.
3. Use Continued Fractions algorithm on $\widetilde{\theta}$ to obtain reduced form a/b of s/r.
4. Check whether $x^b = 1 \bmod N$. If so, then $b = r$. Otherwise, go back to Step 1.

Each loop succeeds with probability $1/\log\log r$, so we only have to repeat a few times.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.
2. If we can solve Order Finding, we can factor integers.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.
2. If we can solve Order Finding, we can factor integers.
3. By running Phase Estimation with Modular Multiplication unitary a few times, we can get noisy estimates of s/r.

Summary of Factoring via Quantum Computers

1. If we can factor, we can break RSA.
2. If we can solve Order Finding, we can factor integers.
3. By running Phase Estimation with Modular Multiplication unitary a few times, we can get noisy estimates of s/r.
4. We can "decode" these estimates by using Continued Fractions algorithm.

How far away is Shor's factoring algorithm?

Gidney, Ekera 2018: given current methods for error correction, we would need

1. 20 million noisy qubits
2. 8 hours

to factor a 2048-bit RSA integer.

How far away is Shor's factoring algorithm?

Gidney, Ekera 2018: given current methods for error correction, we would need

1. 20 million noisy qubits
2. 8 hours

to factor a 2048-bit RSA integer.

IBM Roadmap: 1 million qubits by 2030.

What will replace RSA?

US National Institute for Standards and Technology (NIST) just concluded a multiyear competition to find **postquantum** cryptosystems to replace RSA. The winners are...

What will replace RSA?

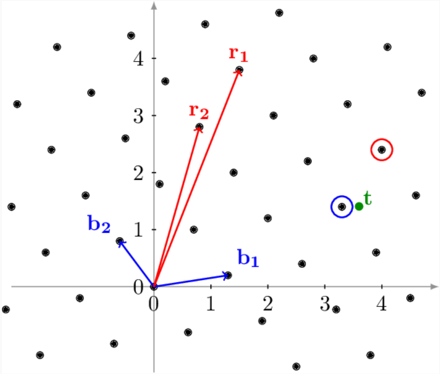
US National Institute for Standards and Technology (NIST) just concluded a multiyear competition to find **postquantum** cryptosystems to replace RSA. The winners are...

1. CRYSTALS-Kyber
2. CRYSTALS-Dilithium
3. FALCON
4. SPHINCS+

Kyber is for encryption, and the last three are for digital signatures.

Post-quantum cryptography from lattices

Kyber, Dilithium, and Falcon are all based on **lattice problems**, where the goal is to find short vectors in a high-dimensional lattice.



It is believed that these problems cannot be quickly solved by quantum computers.

Post-quantum cryptography from lattices

It is an important research problem to find better evidence that lattice problems are hard for quantum computers.

But there's always the possibility that someone can find a fast quantum algorithm for them...

It will take time to build confidence in these new cryptosystems.