Quantum algorithms: Shor’s integer factoring quantum part

Yipeng Huang

Rutgers University

February 21, 2024
\[ \chi^2 = \mathbf{D} \mathbf{J} \]

| \( f(00) \) | 0 | 0 | 0 | 0 |
| \( f(01) \) | 0 | 0 | 0 | 0 |
| \( f(10) \) | 0 | 0 | 0 | 0 |
| \( f(11) \) | 0 | 0 | 0 | 0 |

const.  \( \uparrow \)  \( \uparrow \)  balanced  \( \times \)  \( \times \)

\[
|+\rangle \otimes |+\rangle \otimes |-\rangle \rangle = \left( \frac{|000\rangle}{\sqrt{2}} \right) \otimes \left( \frac{|010\rangle}{\sqrt{2}} \right) \otimes \left( \frac{|111\rangle}{\sqrt{2}} \right)
\]

\[
= \frac{|000\rangle}{2\sqrt{2}} - \frac{|010\rangle}{\sqrt{2}} - \frac{|111\rangle}{2\sqrt{2}} + \frac{|101\rangle}{\sqrt{2}} - \frac{|110\rangle}{\sqrt{2}} - \frac{|100\rangle}{2\sqrt{2}} + \frac{|111\rangle}{2\sqrt{2}} - \frac{|011\rangle}{2\sqrt{2}}
\]
The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
  Factoring to modular square root
  Modular square root to discrete logarithm
  Discrete logarithm to order finding
  Order finding to period finding

Shor’s algorithm quantum part: period finding using quantum Fourier transform
  Calculate modular exponentiation
  Measurement of target (bottom, ancillary) qubit register

Simon’s algorithm: setting up for quantum Fourier transform
Private key crypto.
AES-256
key distribution.
BB84 Quantum key exchange.

Public key Crypto.

Public key $f(x)$
Private key $f^{-1}(x)$

RSA-2048
Discrete log
Elliptic curves

Post Quantum Crypto.
lattices
McEliece -
Etc
The factoring problem

One way functions for cryptography

1. Multiplying two $b$-bit numbers: on order of $b^2$ time.

2. Best known classical algorithm to factor a $b$-bit number: on order of about $2^{\sqrt[3]{b}}$ time.

   - Makes multiplying large primes a candidate one-way function.
   - It’s an open question of mathematics to prove whether one way functions exist.

Public key cryptography

Numberphile YouTube channel explanation of RSA public key cryptography: https://www.youtube.com/watch?v=M7kEpwltn50
The factoring problem

One way functions for cryptography

1. Multiplying two $b$-bit numbers: on order of $b^2$ time.
2. Best known classical algorithm to factor a $b$-bit number: on order of about $2^{3\sqrt{b}}$ time.

Quantum integer factoring algorithm

- Quantum algorithm to factor a $b$-bit number: $b^3$.
- Peter Shor, 1994.
- Important example of quantum algorithm offering exponential speedup.
Table of contents

The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
  Factoring to modular square root
  Modular square root to discrete logarithm
  Discrete logarithm to order finding
  Order finding to period finding

Shor’s algorithm quantum part: period finding using quantum Fourier transform
  Calculate modular exponentiation
  Measurement of target (bottom, ancillary) qubit register

Simon’s algorithm: setting up for quantum Fourier transform
The classical part: converting factoring to order finding / period finding

General strategy for the classical part

1. Factoring
2. Modular square root
3. Discrete logarithm
4. Order finding
5. Period finding

The fact that a quantum algorithm can support all these primitives leads to additional ways that future quantum computing can be useful / threatening to existing cryptography.
Factoring

\[ N = pq \]

\[ N = 15 = 3 \times 5 \]
Modular square root

Finding the modular square root

\[ s^2 \mod N = 1 \]
\[ s = \sqrt{1} \mod N \]

Trivial roots would be \( s = \pm 1 \).

- Are there other (nontrivial) square roots?
- For \( N = 15 \), \( s = \pm 4, s = \pm 11, s = \pm 14 \) are all nontrivial square roots. (Show this).
- Later in these slides, we will see how nontrivial square roots are useful for factoring.
Discrete log

1. Pick a that is relatively prime with N.

2. Efficient to test if relatively prime by finding GCD using Euclid’s algorithm. For example, a=6 and n=15.

Exercise: list the possible a’s for N = 15.

2, 4, 7, 8, 11, 13, 14
Discrete log

1. Pick $a$ that is relatively prime with $N$.
2. Efficient to test if relatively prime by finding GCD using Euclid’s algorithm. For example, $a = 6$ and $n = 15$.

So now our factoring problem is:

\[ a^r \mod N = 1 \]

\[ a^r \equiv 1 \mod N \]

In fact, this algorithm for finding discrete log even more directly attacks other crypto primitives such as Diffie-Hellman key exchange.
Order finding

Our discrete log problem is equivalent to order finding.

<table>
<thead>
<tr>
<th>$a^1 \mod 15$</th>
<th>$a^2 \mod 15$</th>
<th>$a^3 \mod 15$</th>
<th>$a^4 \mod 15$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=2</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>a=4</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>a=7</td>
<td>7</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>a=8</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>a=11</td>
<td>11</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>a=13</td>
<td>13</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>a=14</td>
<td>14</td>
<td>1</td>
<td>14</td>
</tr>
</tbody>
</table>

Find smallest $r$ such that $a^r \equiv 1 \mod N$
Period finding

In other words, the problem by now can also be phrased as finding the period of a function.

\[ f(x) = f(x + r) \]

Where

\[ f(x) = a^x = a^{x+r} \mod N \]

Find \( r \).
What to do after quantum algorithm gives you $r$

- If $r$ is odd or if $a^{\frac{r}{2}} + 1 \equiv 0 \mod N$, abandon.
- There is separate theorem saying no more than a quarter of trials would have to be tossed.

Exercise: try for $a = 14$.

\[ \frac{2}{r} \]

\[ a = 14 \quad a + 1 = 14 + 1 = 15 \not\equiv 0 \mod N. \]

don't use $a$ as trivial now.
What to do after quantum algorithm gives you $r$

- If $r$ is odd or if $a^\frac{r}{2} + 1 \equiv 0 \mod N$, abandon.
- There is separate theorem saying no more than a quarter of trials would have to be tossed.

**Exercise:** try for $a = 14$.

Otherwise, factors are $\text{GCD}(a^\frac{r}{2} \pm 1, N)$

- $a=2$ $r=4$ $2^2 \pm 1 = 4 \pm 1$
- $a=4$ $r=2$ $4^1 \pm 1 = 4 \pm 1$
- $a=7$ $r=4$ $7^2 \pm 1 = 49 \pm 1$
- $a=8$ $r=4$ $8^2 \pm 1 = 64 \pm 1$
- $a=11$ $r=2$ $11^1 \pm 1 = 11 \pm 1$
- $a=13$ $r=4$ $13^2 \pm 1 = 169 \pm 1$
- $a=14$ $r=2$ $14^2 \pm 1 = 196 \pm 1$ (bad case)

Notice why we discarded 14.
Proof why this works and why factoring is modular square root

\[ a^r \equiv 1 \mod N \]

So now \( a^\frac{r}{2} \) is a nontrivial square root of 1 mod N.

\[ a^r - 1 \equiv 0 \mod N \quad \Rightarrow \quad (a^\frac{r}{2} - 1)(a^\frac{r}{2} + 1) \equiv 0 \mod N \]

The above implies that

\[ \frac{(a^\frac{r}{2} - 1)(a^\frac{r}{2} + 1)}{N} \]

is an integer. So now we have to prove that

1. \( \frac{a^\frac{r}{2} - 1}{N} \) is not an integer, and
2. \( \frac{a^\frac{r}{2} + 1}{N} \) is not an integer.
Proof why this works and why factoring is modular square root

Suppose $\frac{a^r - 1}{N}$ is an integer

that would imply

$$a^r - 1 \equiv 0 \mod N$$

$$a^r \equiv 1 \mod N$$

but we already defined $r$ is the smallest such that $a^r \equiv 1 \mod N$, so there is a contradiction, so $\frac{a^r - 1}{N}$ is not an integer.

Suppose $\frac{a^r + 1}{N}$ is an integer

that would imply

$$a^r + 1 \equiv 0 \mod N$$

but we already eliminated such cases because we know this doesn’t give us a useful result.
Table of contents

The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
   Factoring to modular square root
   Modular square root to discrete logarithm
   Discrete logarithm to order finding
   Order finding to period finding

Shor’s algorithm quantum part: period finding using quantum Fourier transform
   Calculate modular exponentiation
   Measurement of target (bottom, ancillary) qubit register

Simon’s algorithm: setting up for quantum Fourier transform
The quantum part: period finding using quantum Fourier transform

- After picking a value for $a$, use quantum parallelism to calculate modular exponentiation: $a^x \mod N$ for all $0 \leq x \leq 2^n - 1$ simultaneously.
- Use interference to find a global property, such as the period $r$. 
Calculate modular exponentiation

- Image source: Huang and Martonosi, Statistical assertions for validating patterns and finding bugs in quantum programs, 2019.
- A good source on how to build the controlled adder, controlled multiplier, and controlled exponentiation is in Beauregard, Circuit for Shor’s algorithm using \(2n+3\) qubits, 2002.
Calculate modular exponentiation

- State after applying modular exponentiation circuit is

\[
\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle
\]

- Concretely, using our running example of \( N = 15 \), need \( n = 4 \) qubits to encode, and suppose we picked \( a = 2 \), the state would be

\[
\frac{1}{4} \sum_{x=0}^{15} |x\rangle |2^x \mod 15\rangle
\]
Measurement of target (bottom, ancillary) qubit register

- We then measure the target qubit register, collapsing it to a definite value. The state of the upper register would then be limited to:

\[
\frac{1}{\sqrt{A}} \sum_{a=0}^{A-1} |x_0 + ar\rangle
\]

- Concretely, using our running example of \(N = 15\), and suppose we picked \(a = 2\), and suppose measurement results in 2, the upper register would be a uniform superposition of all \(|x\rangle\) such that \(2^x \equiv 2 \mod 15\):

\[
\frac{|1\rangle}{2} + \frac{|5\rangle}{2} + \frac{|9\rangle}{2} + \frac{|13\rangle}{2}
\]

- The key trick now is can we extract the period \(r = 4\) from such a quantum state. We do this using the quantum Fourier transform.
Table of contents

The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
  Factoring to modular square root
  Modular square root to discrete logarithm
  Discrete logarithm to order finding
  Order finding to period finding

Shor’s algorithm quantum part: period finding using quantum Fourier transform
  Calculate modular exponentiation
  Measurement of target (bottom, ancillary) qubit register

Simon’s algorithm: setting up for quantum Fourier transform