Quantum algorithms: Shor's integer factoring quantum part

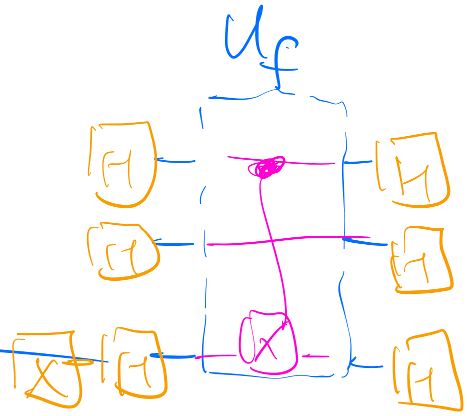
Yipeng Huang

Rutgers University

February 21, 2024

$n = 2$ D-J

	f_0	f_1	f_2	f_3
$f(00)$	0	0	0	0
$f(01)$	0	0	0	0
$f(10)$	0	0	1	1
$f(11)$	0	1	0	1
	const	X	X	balanced



$$|+\rangle \otimes |+\rangle \otimes |-\rangle$$

$$= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$= \frac{|000\rangle}{2\sqrt{2}} - \frac{|001\rangle}{2\sqrt{2}} + \frac{|010\rangle}{2\sqrt{2}} - \frac{|011\rangle}{2\sqrt{2}}$$

$$+ \frac{|100\rangle}{2\sqrt{2}} - \frac{|101\rangle}{2\sqrt{2}} + \frac{|110\rangle}{2\sqrt{2}} - \frac{|111\rangle}{2\sqrt{2}}$$

Table of contents

Private key crypto.

AES-256

Key distribution.

BB84 Quantum key exchange.

Public key Crypto.

Public key — $f(x)$

Private key — $f^{-1}(x)$

$N = p \times q$

p or q

RSA-256
-512
-1024
2048

{ integer factorization
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=M7kEpw1tn50`

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.

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

$f(x) = f(x+r)$ $\quad r?$

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

$s^2 \mod N \equiv 4$

$s^2 - 4 \mod N \equiv 0$

$(s+2)(s-2) \mod N \equiv 0$

$s^2 \mod N \equiv 9$

$s^2 - 9 \mod N \equiv 0$

$(s+3)(s-3) \equiv 0 \mod N$

Finding the modular square root

$$s^2 \mod N = 1$$

$s^2 - 1 \mod N \equiv 0$

$$s = \sqrt{1} \mod N$$

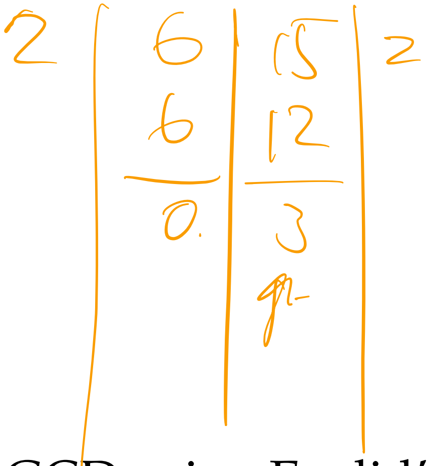
$(s+1)(s-1) \mod N \equiv 0$

Trivial roots would be $s = \pm 1$.

trivial

- 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:

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

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

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

$S = a^{\frac{r}{2}}$

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

Order finding

$a=3$ 3 9 12 6 3

Our discrete log problem is equivalent to order finding.

	$a^1 \mod 15$	$a^2 \mod 15$	$a^3 \mod 15$	$a^4 \mod 15$
a=2	2	4	8	1
a=4	4	1	4	1
a=7	7	4	13	1
a=8	8	4	2	1
a=11	11	1	11	1
a=13	13	4	7	1
a=14	14	1	14	1

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 $a = 14$.

$a = 14 \quad a^{\frac{2}{2}} + 1 = 14 + 1 = 15 \equiv 0 \mod N.$

don't use a as trivial root.

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 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.

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

$$(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \mod N$$

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.

$\Rightarrow$ $(a^r - 1) = kN$

$(a^{\frac{r}{2}}+1)(a^{\frac{r}{2}}-1) = kN$

$\frac{(a^{\frac{r}{2}}+1)(a^{\frac{r}{2}}-1)}{N} = k$

$\frac{(14^{\frac{2}{2}}+1)(13)}{15} = k$

Proof why this works and why factoring is modular square root

Suppose $\frac{a^{\frac{r}{2}}-1}{N}$ is an integer
that would imply

$$a^{\frac{r}{2}} - 1 \equiv 0 \mod N$$

$$a^{\frac{r}{2}} \equiv 1 \mod N$$

but we already defined r is the smallest such that $a^r \equiv 1 \mod N$, so there is a contradiction, so $\frac{a^{\frac{r}{2}}-1}{N}$ is not an integer.

Suppose $\frac{a^{\frac{r}{2}}+1}{N}$ is an integer
that would imply

$$a^{\frac{r}{2}} + 1 \equiv 0 \mod N$$

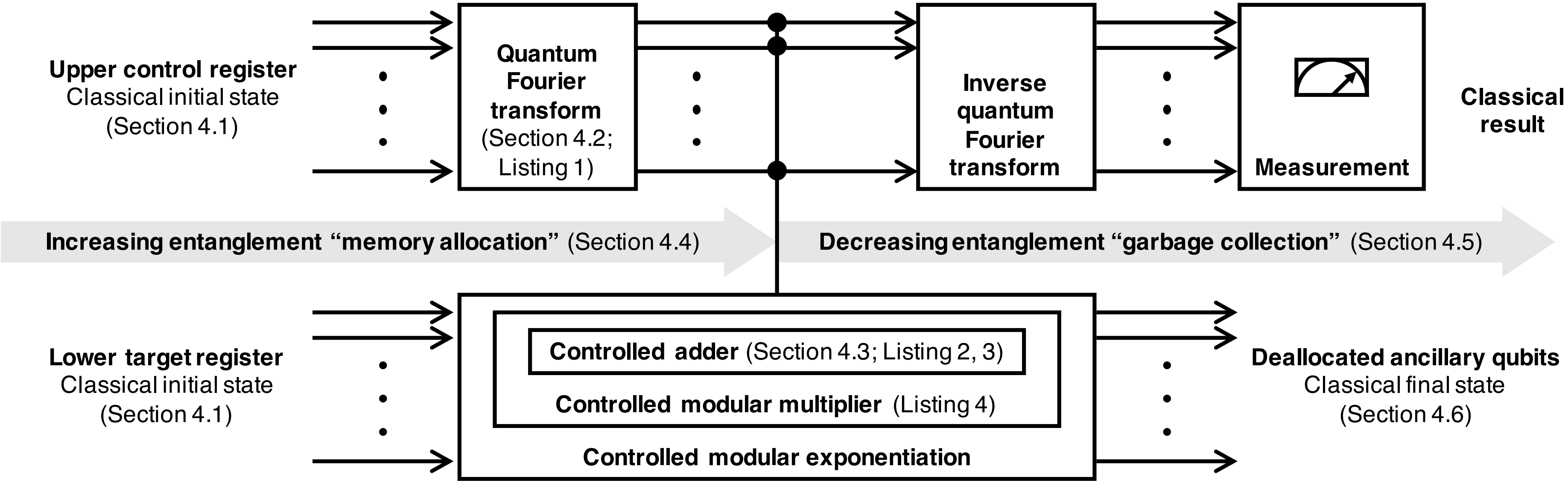
but we already eliminated such cases because we know this doesn't give us a useful result.

Table of contents

The 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