Lecture 16 – Shor's Factoring Alg.!

[[Great! Today we'll do *the* most famous quantum alg., Shor's efficient quantum alg. for factoring huge #s.]]

[[Actually, today will be 100% classical #-theory algs. All the quantum stuff has been covered in previous lectures!]]

Rec. from last time: Let Q_F be a quantum circuit implementing some $F : \{0, 1, 2, \ldots, N-1\} \to \text{COLORS}$
$(N = 2^n)$ $\qquad$ $\text{COLORS} \subseteq \{0,1\}^m$

which is "L-periodic":

$$F(X) = F(X+L) = F(X+2L) = \cdots \;\&\; F(X) = F(Y) \Rightarrow L \mid Y - X.$$

[[And $X + kL$ is *not* taken mod N, so L need not divide N.]]

Then w/ $\propto n^2 + \text{size}(Q_F)$ quantum gates, can get a "clue" to L, looking like:

- random $0 \le k < L$ picked
- you see $\text{NearestInt}(k \cdot \frac{N}{L})$
- else you see "junk"

[[Actually, "junk" very likely to be another integer near $k \cdot N/L$, still OK...]]

Today: ① [[Shor:]] How to use clue to find L
[[with decent probability, efficiently, using a classical alg.]]

② How this ability helps factor #'s
[[Again, classically & efficiently.]]
[[This was already known in mid-'70s.]]
[[Uses a classically efficient-to-compute F, hence quantumly effic. Q_F.]]

Rem: • Simple algs: $\approx m^3$ quantum gates to factor m-bit integers

• More carefully: $\approx m^2 (\cdot \log m \log\log m \dots)$
[[Best known. Not quasilinear. I'll point out the bottleneck — which is classical!]]

Rec: Naive classical alg.: $\approx \sqrt{2}^m$ steps
Sophisticated " " : $C^{m^{1/3}(\log\dots)}$ steps
[[Still exponential!]]

[[\$100k to factor RSA-1024
Record is < 800 bits. 2048 bits — no way!]]

[[We'll start with ②: How to factor, given ability to find the "L" of an "L-eriodic" F.]]

[[This reduction from factoring to "order-finding" was perhaps folklore in '70s. First(?) published by CMU's Gary Miller in '76. Cf. Long '81, Woll '87.]]

Input: B, a Big number: m bits (e.g. $m = 1000$)

Goal: prime factors of B.

For simplicity today: Assume $B = P \cdot Q$ for primes P, Q.

[[Imagine P, Q both have 512 bits.]]

- this is the case of interest for "RSA crypto"
- intuitively, the "hardest case"

[[If B has >2 prime factors, they're smaller & easier to find. Recall primality testing is doable efficiently, $\approx m^2$ steps.]]

[[General case mainly requires a little more bookkeeping. (And ability to detect perfect powers.)]]

[[OK: how can we use period-finding to help factor B? It's a bit similar to the Miller-Rabin primality test, actually....]]

Goal: Find a "nontrivial" $\underbrace{\text{square-root of } 1}_{R\,:\,R^2=1 \bmod B}$, mod B.

$R \not\equiv \pm 1 \bmod B.$

[[Why does it help?]]

$$\Rightarrow R^2 - 1 \equiv 0 \bmod B$$

$$\Rightarrow (R-1)(R+1) \equiv 0 \bmod B = P \cdot Q$$

$$\therefore\ P, Q \mid (R-1)(R+1),$$

but $PQ \nmid R-1$ or $R+1$

($\because R \not\equiv \pm 1 \bmod B = PQ$)

[[If B is prime, this quadratic can only have 2 roots, ±1, since $\mathbb{Z}_B$ would be a field. But B=P·Q. Turns out to imply there are 2 more square roots of 1 mod B.]]

$\therefore\ P \mid R-1,\ Q \mid R+1$, or vice versa.

Either way, take R+1 (say) & gcd it with B = PQ → gives you either P or Q. ■

[[↑ Classically efficient.]]

How to find R?

Recall (HW3, #5; HW4, #6)

$$\mathbb{Z}_B^* = \{\text{integers } A \in \mathbb{Z}_B \text{ with } \gcd(A,B)=1\}$$
$$= \{ \text{" " " " s.t. } A^{-1} \bmod B \text{ exists}\}$$

① Pick $A \in \mathbb{Z}_B^*$ uniformly at random. How?

Pick $A \in \mathbb{Z}_B$ " " ", compute $\gcd(A,B)$

If it's 1, $A \in \mathbb{Z}_B^*$. :)

If not, it's P or Q → you're done!

[[This is exponentially unlikely, so don't get too excited :).]]

Hypothetically, consider

$$\left.\begin{array}{l} A \bmod B \\ A^2 \bmod B \\ A^3 \bmod B \\ \vdots \\ A^L = 1 \bmod B \end{array}\right\} \text{ all distinct: } A^i \equiv A^j,\ j>i \Rightarrow A^{j-i} \equiv 1 \ (\because A^{-1} \text{ exists})$$

def: L = "order" of A in $\mathbb{Z}_B^*$

[[Least L s.t. $A^L \equiv 1 \bmod B$.]]

Then repeats:

$A^{L+1} \equiv A^1$, $A^{L+2} \equiv A^2$, etc..

[[Remark: L could be enormous, like an $\approx \frac{m}{2}$-bit number. Can't find it naively... Known to be "as hard as factoring. But....]]

$$F: \{0,1,2,\dots,N-1\} \to \mathbb{Z}_B^*, \quad F(x) = A^x \bmod B$$

Need not be B. Will take it to be: • a power of 2 • much bigger than B.

"COLORS", $\subseteq \{0,1\}^m$

• It's "L-periodic"!

• Classically computable

[["Modular exponentiation", HW1, #3]]

in $\approx m^3$ time $\to \tilde{O}(m^2)$ $\Rightarrow$ classical gates.

Can build, make reversible, voila! – get Q_F quantumly implementing F.

$\to$ Can find L with decent prob. (Part ①, coming up.)

[[The efficiency bottleneck for Shor, as it turns out!]]

Given L, hope for 2 lucky things:
(based on random choice of A)

<u>Luck 1:</u> <u>L is even.</u>

$\Rightarrow L/2$ an integer

$\Rightarrow$ "$A^{L/2}$" mod B makes sense. (which you can compute efficiently knowing L, B)

& $(A^{L/2})^2 \equiv A^L \equiv 1 \mod B$

$\therefore A^{L/2}$ a square-root of 1!

<u>Luck 2:</u> $A^{L/2} \not\equiv \pm 1 \mod B$ ("nontriv. sqrt")

<u>Elementary number theory lemma:</u>

$\Pr[\text{Luck 1 \& Luck 2}] \geq \frac{1}{2}$.

<u>Proof:</u> [[Not hard, will be on homework.]]

$\therefore$ can Factor $B = P \cdot Q$ with decent prob. given L! [[Can check your work, too. Now just repeat $O(1)$ times.]]

Part (2): Finding L given "clue":

$$S = \text{NearestInt}\left(k \cdot \frac{N}{L}\right) \quad (\text{with prob } \geq 40\%)$$

for random $0 \leq k < L$.

(Recall that's what the quantum "approximate period-finding alg." based on QFT & Simon's Alg. over $\mathbb{Z}_N$ gave)

Rem: L is m bits.

We'll choose $N = 2^n$ for $n = 10m$.

[Conceptually clearer if you make $n = m^{10}$, which would still lead to "polynomial time"]

Note: • $L \leq 2^m \ll 2^{10m} \sim N$.

• $\frac{N}{L}$ not an integer

[N is an enormous power of 2. Think of L as "smallish" now!]

Get $S \overset{!}{\approx} k \cdot \frac{N}{L}$, $0 \le k < L$ rand. int. [Basically w/prob $\gtrsim 40\%$.]

[My new notation for super-duper-close-to.]

$\frac{S}{N} \overset{!}{\approx} \frac{k}{L}$. Error is $\pm \frac{.5}{N}$.

Alg knows numer. & denom.

Alg. wants to know L.

<u>Key Claim</u>: From $\frac{S}{N}$, classical alg. can efficiently figure out the frac. $\frac{k}{L}$ in lowest terms.

$\Rightarrow$ We're done!

Method ①: Hope random k is prime (or coprime to L). Then $\frac{k}{L}$ <u>is</u> in lowest terms $\Rightarrow$ alg. knows L.

$\Pr[k \text{ prime}] \gtrsim \frac{1}{m}$.

Method ②: Repeat a few times.
With high prob., LCM of denominators is L. (Exercise.)

(By Prime # Theorem. Now can repeat $\approx m$ times in expectation, get L. Efficient!)

[Last step: how to figure out k/L?]

Imagine $N = 10^n$, not 2^n.

Alg gets S, $\frac{S}{N} \overset{!}{\approx} \frac{k}{L}$ to $\pm .5/N$

$\Rightarrow \frac{S}{N} = 0.S$ is frac $\frac{k}{L}$ to n decimal places!

[Only $m \ll \sqrt{n}$ bit number.]

[to n binary digits for $N = 2^n$]

e.g. imagine $L \leq 50$, $N = 1{,}000{,}000$.

Say quantum circuit returns $S = 666{,}667$.

Alg. knows $\frac{k}{L} \overset{!}{\approx} 0.666667$ [to 6 decimal places]

Obviously, $\frac{k}{L} = 2/3$. [Remember, just want it in lowest terms now.]

[Still imagine $N = 1{,}000{,}000$ & $L \le 50$.]

Say $S = 181{,}818$, so $\frac{k}{L} \overset{!}{\approx} 0.181818$.

$$\rightsquigarrow \frac{k}{L} = \frac{2}{11}$$

Say $S = 142{,}857$, so $\frac{k}{L} \overset{!}{\approx} 0.142857$.

$$\rightsquigarrow \frac{k}{L} = \frac{1}{7}.$$

Say $S = 309{,}524$. so $\frac{k}{L} \overset{!}{\approx} 0.309524$

$$\rightsquigarrow \frac{k}{L} = ??$$

[Remark: Go to Maple (or Mathematica, etc.) & type "identify(.309524);" It will give the answer!! How does it do it?]

[[Let's go back to easier cases, try to "discover" the answer.]]

$S = 142857$? Know $S \div N \overset{!}{\approx} \frac{k}{L}$.

Can (efficiently) do integer division

N div S. $\rightsquigarrow$ yields 7

remainder ... 1.

!!!!

[[From algorithm's perspective, this is a preposterously small remainder! Could have been anything in 0 ... 142856, and it was 1?!!

Not a coincidence! Alg. surely now sees that $\frac{S}{N} \overset{!}{\approx} \frac{1}{7}$.]]

[[How about...]] $S = 181818.$ $\overset{!!}{=} S_1$

[[Alg can first try for same miracle.]]

$N \text{ div } S_1 = 1000000 \text{ div } 181818$

$= 5,$ remainder... $90910. =: S_2$

[[Hmm. This is a big remainder, not a preposterously small one.]]

You & I secretly know $\frac{S_1}{N} \overset{!}{\approx} \frac{2}{11} \Rightarrow \frac{N}{S_1} \overset{!}{\approx} 5.5.$ So that remainder S_2 is $\overset{!}{\approx} .5 \cdot S_1.$ That is, $S_1 \div S_2 \approx 2!$

Alg now does S_1 div S_2.

$= 181818 \text{ div. } 90910$

$= 1$ remainder $90908,$

or 2 remainder -2

[[preposterously small!]]

[[So surely sees $S_1/N = \frac{2S_2}{11S_2} = \frac{2}{11}!$]]

Now surely knows: $S_1 \approx 2S_2$, $N \approx 5S_1 + S_2 = 10S_2 + S_2 = 11S_2.$

Alg is... do GCD(N,S) till you hit a preposterously small #!

Back to $N = 1000000$, $S = "S_1" = 309{,}524$

N div S_1 = 3 remainder $71428 = S_2$

S_1 div S_2 = 4 remainder $23812 = S_3$

S_2 div S_3 = 2 remainder 23804

or 3 remainder −8 !!!!

So $S_2 \approx 3S_3$

$S_1 = 4S_2 + S_3 \approx 13S_3$

$N = 3S_1 + S_2 \approx 39S_3 + 3S_3 = 42S_3$

$\therefore \frac{S_1}{N} \approx \boxed{\frac{13}{42}}$:)

[Indeed, it's 0.3095238095.....]

Analysis? Let's recap...

Did $GCD(N, S)$, $S \approx \frac{13}{42}N$ to $\pm \frac{1/2}{N}$.

N		42
	(quotient 3)	
$\approx \frac{13}{42}N$		13
	(quotient 4)	
$\approx \frac{3}{42}N$		3
	(quotient 3)	
$\approx \frac{1}{42}N$		1
≈ 0		0

same steps as GCD(42,13)

Broad steps like doing GCD on L, K, which are m-bit #'s

$\Rightarrow \leq 2m$ steps.

"smallish"

[error analysis?]

$$
\begin{array}{lll}
N & & 42 \\
\approx \frac{13}{42}N \pm \frac{1}{N} & (\text{quotient } 3 =: q_1) & 13 \\
\approx \frac{3}{42}N \pm \frac{3}{N} & (\text{quotient } 4 =: q_2) & 3 \\
\approx \frac{1}{42}N \pm \frac{13}{N} & (\text{quotient } 3 =: q_3) & 1 \\
\approx 0 \pm \frac{42}{N} & & 0
\end{array}
$$

same steps as $\longleftrightarrow$ GCD(42,13)

First error "e_1" at most $\pm \frac{1/2}{N} < \pm \frac{1}{N}$.

Second error e_2 : $< q_1 \cdot e_2 = q_1 \cdot (\pm \frac{1}{N}) = \pm 3/N$

Third error e_3 : $\leq q_2 \cdot e_2 + e_1 = 4 \cdot (\pm \frac{3}{N}) \pm \frac{1}{N} = 13/N$

Fourth error e_4 : $\leq q_3 \cdot e_3 + e_2 = 3 \cdot (\pm \frac{13}{N}) \pm \frac{3}{N} = 42/N$!

Final error is actually $\leq \pm L/N \leq \frac{2^m}{2^n} = \frac{1}{2^{9m}}$, $\because n = 10m$.

our algorithmic threshold for "≈ 0".

Follow chain back $\Rightarrow$ $N = \frac{13}{42} S$ deduction correct up to $\pm \frac{L^2}{N} \leq \frac{2^{2m}}{2^n}$

$\frac{K}{L}$

Could $\frac{K}{L} = \frac{K'}{L'}$ up to $\pm \frac{L^2}{N}$? $\left| \frac{K}{L} - \frac{K'}{L'} \right| \geq \frac{1}{L \cdot L'} \geq \frac{1}{2^{2m}}$.

So no, provided $\frac{1}{2^{2m}} > \frac{2^{2m}}{2^n} \Leftrightarrow 4m < n := 10m.$ ✓ ∎