Lecture 4: Shor's algorithm

It's time to apply the phase estimation subroutine to a bona fide, interesting and important* computational problem: factoring!

(*At least if you want to break classical public key cryptosystems.)

Recall factoring is not known to be in BPP. Best known classical algorithm (number field sieve) has run-time $\exp(O(\sqrt[3]{\log n}\,\sqrt[3]{\log\log n}^{\,2}))$ (where $n = \log N$ = # digits in N), which is subexponential but superpolynomial. Shor's algorithm puts Factoring in BQP – a superpolynomial speedup!

Shor [1994] originally solved order-finding (& hence factoring) via period finding – see Exercise sheet.

Phase estimation approach we use here is due to Kitaev. Equivalent to original approach, but gives a different, unifying perspective on QFT-based algorithms. But well worth understanding both approaches!

1. Order-Finding

We need some basic number theory first...

Modular arithmetic reminder:

Sometimes called "clock arithmetic"; arithmetic of 24h clock is modulus 24 arithmetic: $15h + 11h = 2h \pmod{24}$

$a \pmod{N}$ = integer remainder left over from dividing a by N $(a, N \in \mathbb{Z})$

$a \equiv b \pmod{N}$ means N divides $a-b$

$\mathbb{Z}_N = \{0, 1, \dots, N-1\}$ with multiplication mod N

$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : a, N \text{ coprime}\}$ ← i.e. $\gcd(a,N) = 1$

$\mathbb{Z}_N^*$ forms a group. In particular, $\forall a \in \mathbb{Z}_N, \exists b \in \mathbb{Z}_N$ s.t. $ab \equiv 1 \pmod{N}$. (We write $b \equiv a^{-1}$.)

For $a_1 \equiv b_1 \pmod{N}$, $a_2 \equiv b_2 \pmod{N}$:

$$a_1 + a_2 \equiv b_1 + b_2 \pmod{N}$$

$$a_1 a_2 \equiv b_1 b_2 \pmod{N}$$

$$a_1^k \equiv b_1^k \pmod{N}, \quad k \in \mathbb{N}_+$$

Def (Order)

Order of $a \in \mathbb{Z}_N$, a, N coprime is minimum $r > 0$ s.t. $a^r \equiv 1 \pmod{N}$.

Order of any element is well-defined, i.e. always exists some such r (Euler's Thm).

Problem (Order-finding)

Input: $N \in \mathbb{N}$, $a \in \mathbb{Z}_N$

Output: order of a

Not known to be in P (or BPP). Naive approach of checking $a^r \pmod{N}$ for increasing values of r takes worst case time $O(2^{\log N})$.

Algorithm

Let $U_a|x\rangle = |ax \pmod{N}\rangle$, $a \in \mathbb{Z}_N^*$.

Note that $0 < x < N$, so $|x\rangle$ contains $O(\log N)$ qubits

Exercise: Prove that U_a is unitary.

The quantum order-finding algorithm is "just" phase estimation applied to U_a, followed by some (poly-time) classical processing of the outcomes.

What are eigvals/vects of U_a?

Recall $|a^r\rangle = |1\rangle$ (def. order r)
and note $U_a |a^k\rangle = |a^{k+1}\rangle \longrightarrow$ guess:

$$|\psi_0\rangle = \frac{1}{\sqrt{r}} \left(|1\rangle + |a\rangle + |a^2\rangle + \cdots + |a^{r-1}\rangle \right)$$

$$U_a |\psi_0\rangle = \frac{1}{\sqrt{r}} \left(|a\rangle + |a^2\rangle + |a^3\rangle + \cdots + |1\rangle \right)$$
$$= |\psi_0\rangle$$

$\rightarrow$ eigval. 1

More generally, add phases:

$$|\psi_k\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega_r^{-jk} |a^j\rangle \qquad \text{recall } \omega_r = e^{2\pi i/r}$$

$$U_a |\psi_k\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega_r^{-jk} |a^{j+1}\rangle$$
$$= \omega_r^k \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega_r^{-jk} |a^j\rangle = \omega_r^k |\psi_k\rangle$$

$\rightarrow$ eigvals. $\omega_r^k = e^{2\pi i (k/r)}$

Estimate phase $\tilde{\theta} = \frac{k}{r} \rightarrow$ order $r = \frac{\tilde{\theta}}{k}$.

However, not clear if we can actually implement this.

Recall two issues we need to address in applying phase-estimation:

1. How to implement $cU_a^{2^n}$ efficiently?

2. How to construct an eigenvector $|\psi_k\rangle$?

Implementing $cU_a^{2^n}$ efficiently

For black-box U, there's nothing better than applying cU 2^n times. But U_a is not a black-box! We know what goes on inside the box: modular multiplication.

Exercise: Use exponentiation by squaring & properties of modular arithmetic to show U_a can be implemented in time $O(\text{poly}(n))$.

Constructing eigenvectors of U_a

Can't construct any $|\psi_k\rangle$ efficiently without knowing order r we're trying to find.

But recall running Phase Estimation on superposition of eigenvectors $|\psi_k\rangle$ gives us an estimate of a randomly chosen eigenvalue.

Consider superposition

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} |\psi_k\rangle = \frac{1}{r}\sum_{k=0}^{r-1}\sum_{j=0}^{r-1} \omega_r^{-jk} |a^j\rangle$$

$$= \frac{1}{r}\sum_{j=0}^{r-1} \underbrace{\left(\sum_{k=0}^{r-1} \omega_r^{-jk}\right)}_{=0 \text{ unless } j=0} |a^j\rangle$$

$$= |a^0\rangle = |1\rangle.$$

State $|1\rangle$ ($= |00\cdots01\rangle$ in binary) is easy to prepare!

Run PE for U_a on $|1\rangle$ to precision ε

$\longrightarrow$ With probability $\geq 1-\delta$ gives $\tilde{\theta}$ s.t. $\left|\tilde{\theta} - \frac{k}{r}\right| \leq \varepsilon$, chosen uniformly at random over $k \in \{0, \ldots, r-1\}$.

Recall runtime is $O(\log \frac{1}{\varepsilon\delta})$.

Issue: we don't know k, so how can we infer r from $\tilde{\theta}$?

We need some notions from the elegant theory of ...

Continued fractions

Any $x \in \mathbb{R}$ can be written as a "continued fraction":

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}$$

Can simplify notation & write this as $x = [a_0; a_1, a_2, a_3, \ldots]$.

"Regular" continued fraction has $a_i \in \mathbb{N}_+$.

Terminates after a finite number of levels iff x is rational.
(Cf. decimal expansions, which don't necessarily terminate even for rationals.)

Note: canonical continued fraction is unique for x irrational; rational x have exactly two: $x = [a_0; a_1, \ldots, a_n] = [a_0; a_1, \ldots, a_n - 1, 1]$.
(Easy to show.)

Truncating the expansion at level k gives increasingly good rational approximations, called "convergents" of x:

$$\begin{array}{cccc} a_0, & \dfrac{1}{a_1}, & \cfrac{1}{a_1 + \cfrac{1}{a_2}}, & \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3}}} \quad \cdots \\ \| & \| & \| & \| \\ [a_0] & [a_0; a_1] & [a_0; a_1, a_2] & [a_0; a_1, a_2, a_3] \end{array}$$

Lemma

Let $\frac{p_n}{q_n} = [a_0; a_1, \dots, a_n]$ (not necessarily regular).

(i) p_n, q_n satisfy recurrence relation:

$$p_n = a_n p_{n-1} + p_{n-2} \ (n \geq 0), \quad p_{-1} = 1, p_{-2} = 0$$
$$q_n = a_n q_{n-1} + q_{n-2} \ (n \geq 1), \quad q_0 = 1, q_{-1} = 0$$

(ii) $p_n q_{n-1} - q_n p_{n-1} = \pm 1$

(iii) $p_n > p_{n-1}, \quad q_n > q_{n-1}, \quad p_n > 2p_{n-1}, \quad q_n > 2q_{n-1}$.

(iv) $\frac{p_n}{q_n}$ is in lowest terms

i.e. $\gcd(p_n, q_n) = 1$

Proof:

(i) Base case $n = 0$, $n = 1$ holds (just calculate).
Assume recurrence holds up to $n-1$.

$$\frac{p_n}{q_n} = [a_0; a_1, \dots, a_n] = [a_0; a_1, \dots, a_{n-1} + \tfrac{1}{a_n}]$$

$$= \frac{(a_{n-1} + \frac{1}{a_n}) p_{n-1} + p_{n-2}}{(a_{n-1} + \frac{1}{a_n}) q_{n-1} + q_{n-2}} \quad \text{by induction hypothesis}$$

$$= \frac{(a_{n-1} p_{n-1} + p_{n-2}) + p_{n-1}/a_n}{(a_{n-1} q_{n-1} + q_{n-2}) + q_{n-1}/a_n}$$

$$= \frac{p_n + p_{n-1}/a_n}{q_n + q_{n-1}/a_n} \quad \text{by induction hypothesis} \quad = \frac{a_n p_n + p_{n-1}}{a_n q_n + q_{n-1}}$$

i.e. can take p_n, q_n satisfying recurrence.

(ii) Easy proof by induction, making use of (i).

(iii) Immediate from (i).

(iv) If d divides p_n & q_n, then by (ii) it also divides ± 1.

Theorem

If $\left| x - \frac{p}{q} \right| < \frac{1}{2q^2}$ $\quad x \in \mathbb{R}, \quad p, q \in \mathbb{N}$ then p/q is a convergent of x. Moreover, there is no other rational $\frac{p'}{q'} \neq \frac{p}{q}$ with $q' \leq q$ & $\left| \tilde{\theta} - p'/q' \right| < \frac{1}{2q^2}$.

Proof:

Let $p/q = [a_0; a_1, \ldots, a_n]$ be the continued fraction expansion of p/q, with convergents p_i/q_i (so $p_n = p$, $q_n = q$).

Write $x = \dfrac{\lambda p_n + p_{n-1}}{\lambda q_n + q_{n-1}} = [a_0; a_1, \ldots, a_n, \lambda]$.

(Not a regular continued fraction, as $\lambda \notin \mathbb{N}$ in general.)

We have

$$\frac{1}{2q_n^2} \leq \left| x - \frac{p}{q} \right| = \left| \frac{\lambda p_n + p_{n-1}}{\lambda q_n + q_{n-1}} - \frac{p_n}{q_n} \right| .$$

Rearranging,

$$\lambda = \left| 2(p_n q_{n-1} - q_n p_{n-1}) - \frac{q_{n-1}}{q_n} \right|$$
$$> |2-1| = 1 \qquad \text{Lemma (ii) \& (iii)}$$

Thus $\lambda = [b_0; b_1, b_2, \dots]$ with $b_0 \geq 1$.

Hence $x = [a_0; a_1, \dots, a_n, b_0, b_1, \dots]$, and $p/q = [a_0; a_1, \dots, a_n]$ is a convergent.

Now assume $p'/q' \neq p/q$ with $q' \leq q$. Then

$$\begin{aligned} \left|\tilde{\theta} - p'/q'\right| &= \left| p'/q' - p/q + p/q - \tilde{\theta} \right| \\ &\geqslant \left| p'/q' - p/q \right| - \left| \tilde{\theta} - p/q \right| \\ &\geqslant \frac{|pq' - qp'|}{q'q} - \frac{1}{2q^2} \\ &\geqslant \frac{1}{q^2} - \frac{1}{2q^2} \qquad \text{using } p'/q' \neq p/q \text{ \& } q' \leq q \\ &= \frac{1}{2q^2} \end{aligned}$$

contradicting $\left|\tilde{\theta} - p'/q'\right| < \frac{1}{2q^2}$. $\square$

Order-Finding classical post-processing

Recall phase estimation yields $\tilde{\theta}$ s.t. $\left|\tilde{\theta} - \frac{k}{r}\right| \leq \varepsilon$ with probability $1-\delta$, for $k \in \{0, \ldots, r-1\}$ chosen uniformly at random.

Want to choose ε s.t. can uniquely identify closest $\frac{k}{r}$ to $\tilde{\theta}$.
Moreover, we want an efficient algorithm to find this $\frac{k}{r}$.

We don't know order r, but we do know $r < N$ (see Def. order).

Choose $\varepsilon = \frac{1}{2N^2} \leq \frac{1}{2r^2}$.

By continued fractions Theorem, $\frac{k}{r}$ is unique convergent p_n/q_n of $\tilde{\theta}$ with $q_n \leq r$ satisfying $\left|\tilde{\theta} - p_n/q_n\right| < \frac{1}{2q_n^2}$.

$\longrightarrow$ Compute successive p_n, q_n for $\tilde{\theta}$ (can be done efficiently by Lemma (i)) until find convergent satisfying conditions $= k/r$.

If k, r coprime $\Rightarrow$ fraction $k/r = p_n/q_n$ is in lowest terms $\longrightarrow$ read off $r = q_n$.

However if $\gcd(k,r) > 1$, fraction k/r is not in lowest terms $\longrightarrow$ convergent p_n/q_n will give reduced fraction $\longrightarrow$ $q_n \neq r$.

What is the probability of getting a "good" k coprime to r?

We quote the following number-theoretic resut on the distribution of coprimes without proof:

Theorem

$$|\{k : k, r \text{ coprime}, k < r\}| = O\left(\frac{r}{\log\log r}\right).$$

$\longrightarrow$ For $k \in \{0, \dots, r-1\}$ chosen uniformly at random

$$\Pr(k, r \text{ coprime}) = \Omega\left(\frac{1}{\log\log r}\right).$$

Algorithm (Order-Finding)

1. repeat $O(\log\log N)$ times:
2. $\quad \theta \leftarrow$ phase estimation to $\varepsilon = \frac{1}{2N^2}$
3. $\quad$ while $|\tilde{\theta} - p_n/q_n| \not< \varepsilon$:
4. $\quad\quad n \leftarrow n+1$
5. $\quad\quad$ compute n^{th} convergent p_n/q_n
6. $\quad r \leftarrow q_n$
7. $\quad$ if $a^r \equiv 1 \bmod N$:
8. $\quad\quad$ return r

Overall runtime is dominated by multipcation in classical post-processing (s ecifically, computing convergents), not quantum part (phase estimation)!

$\longrightarrow$ Total run-time $= O((\log N)^3 \log\log N)$
($= O(n^3 \log n)$ where $n = \#$ bits in input)

Exercise: Prove carefully that above algorithm solves order-finding with probability $\geq 1-\delta$ in time $O(n^3 \log n \log \frac{1}{\delta})$.

Note: using fast multiplication algorithms and better number theory theorems, run-time can be reduced to:

$$O(n^2 (\log n)^2 \log\log n \log \tfrac{1}{\delta}).$$

2. Factoring

Problem (Factoring)

Input: $N \in \mathbb{N}$

Output: Prime factorization $N = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$.

Solve this by (classical probabilistic poly-time) reduction to order-finding — no further quantum part at all.

It suffices to have an algorithm that returns a factor (not neccessarily prime).

If d is a factor of N, can compute N/d in poly-time to give decomposition $N = d \cdot N/d$ then call algorithm recursively on d & N/d.

Algorithm (Factoring)

```
if N is even:
    return 2
if N = p^k:
    return p
repeat log 1/δ times:
    choose a ∈ {2, ..., N-1} uniformly at random
    d ← gcd(a, N)
    if d ≥ 2:
        return d
    else:
        r ← order a^r ≡ 1 (mod N) using
            quantum order-finding alg.
        if r is even:
            d ← gcd(a^{r/2} - 1, N)
            if d ≥ 2:
                return d
```

Analysis

Lines 3-4:

Finding k if $N = p^k$ can be done efficiently: $p \geq 2$, so $k \leq \log N$; for all $k \leq \log N$, compute $\sqrt[k]{N}$ and check if result is integer.

Lines 6-9:

We got lucky with our random choice of a, yielding a factor right away.

Line 10:

From here on, guaranteed $\gcd(a, N) = 1$
$\longrightarrow$ order r of a in N well-defined.

Lines 11-15:

This is the core of the algorithm.

If our random choice of a passes tests in lines 12 & 14, algorithm returns
$d = \gcd(a^{r/2} - 1, N)$.

Note following trivial but useful observation:
$d = \gcd(x, N)$ is always a factor of N.
However, could be trivial factor 1 or N if x, N coprime or N divides x.

Need to show any returned d is necessarily a non-trivial factor. I.e. need to rule out possibilities (i) $d = 1$ and (ii) $d = \gcd(a^{r/2}-1, N) = N$ (i.e. N divides $a^{r/2}-1$).

(i) Line 14 explicitly tests $d \geq 2 \Rightarrow d \neq 1$.

(ii) Line 12 explicitly tests for r even.

If N divides $a^{r/2}-1 \Rightarrow a^{r/2} = 1 \pmod N$, but this is not possible since order r is <u>minimal</u> such integer (Def. order).

$\Rightarrow$ N does not divide $a^{r/2}-1$.

$\longrightarrow$ If algorithm returns d in line 15, d guaranteed to be a non-trivial factor

If choice of a fails, i.e. either r happens to be even, or happens that $d = \gcd(a^{r/2}-1, N) = 1$ (i.e. $a^{r/2}-1$, N coprime), then algorithm just tries again with a new random choice of a.

We need to know the probability of this happening, so we know how many attempts are required to achieve success probability $\geq 1-\delta$.

We quote the following Thm.:

Theorem

Let N be an odd number that is not a prime power. Let a be chosen uniformly at random over all $a \leq N-1$ coprime to N (i.e. $\gcd(a,N)=1$). Then

$$\Pr\left(\text{order } r \text{ is even } \& \ a^{r/2}+1 \neq 0 \pmod N\right) \geq \frac{1}{2}.$$

(See [Nielsen & Chuang, Appendix 4.3], [Preskill lecture notes] or [Ekert & Jozsa, Rev. Mod. Phys. 68, p.733, 1996] for proof.)

$\longrightarrow$ $\log \frac{1}{\delta}$ attempts suffice to succeed with probability $\geq 1-\delta$.

(Cf. analysis of Simon's Algorithm.)