Quantum Computation

Lectures by Richard Jozsa

Contents

1 Review of Shor's Algorithm

This result is powered by the **quantum period finding algorithm**, and will lead us to the **hidden subgroup problem** (henceforce HSP).

1.1 Factoring Problem

Given an integer N, with $n = O(\log N)$ digits, we want to find a non-trivial factor in time complexity $O(\text{poly}(n))$.

The important concept here is that of **polynomial time complexity**: any computation has an input, from which we obtain an input *size* n. Then by polynomial time complexity, we mean that the number of steps/gates (either classical or quantum) grows only polynomially with n (*i.e.* is $O(\text{poly}(n))$).

When we refer to **efficient** computation, we are always referring to polynomial time complexity.

The best known *classical* factoring algorithm has complexity $\mathrm{e}^{O\left(n^{\frac{1}{3}}(\log n)^{\frac{2}{3}}\right)}$. However, the best known quantum algorithm (due to Shor) runs in $O(n^3)$, a considerable improvement.

1.2 Quantum Factoring Algorithm Summary

First, we convert factoring into period determination:

Given N, choose $a < N$ with $(a, N) = 1$ and consider $f : \mathbb{Z} \to \mathbb{Z}_N,\ x \mapsto a^x \mod N$. Euler's Theorem tells us that f is periodic, and the period r is the order of a modulo N, *i.e.* the least $m > 1$ such that $a^m \equiv 1 \mod N$ - this exists if and only if a, N are coprime. Through knowledge of r we are able to compute a factor of N.

While the process of determining r is *mathematically* very simple, it is in fact as difficult to compute from a classical perspective as factoring N itself. Instead we use the **Quantum algorithm for periodicity determination**.

The task: Given an oracle/black box for $f : \mathbb{Z}_M \to \mathbb{Z}_N$ with promises:

- f is periodic, with (unknown) period $r \in \mathbb{Z}_M$, *i.e.* f(x+r) = f(x) for all $x \in \mathbb{Z}_M$.
- f is $1-1$ in each period, *i.e.* $f(x_1) \neq f(x_2)$ for any $0 \leq x_1 < x_2 < r$.

We want to find r in time $O(\text{poly}(m)),\ m = \log M$ (with any prescribed success probability $1 - \varepsilon,\ \varepsilon > 0$).

Remark: Queries to the oracle count as 1 step. In the quantum context we assume the oracle is a unitary gate $\mathcal{U}_f$ on $\mathcal{U}_M \otimes \mathcal{U}_N$, where $\mathcal{U}_M$ is the state space with dimension M, basis $\{|i\rangle\}_{i \in \mathbb{Z}_M}$. U_f acts on basis states as

$$U_f \underbrace{|i\rangle}_{\text{input}} \underbrace{|j\rangle}_{\text{output}} = |i\rangle|j + f(i)\rangle, \qquad i \in \mathbb{Z}_M,\ j \in \mathbb{Z}_M$$

The **Query complexity** of an algorithm is the number of times the oracle is queried, which is also required to be $O(\text{poly}(m))$.

To solve the periodicity problem classically, it can be shown that it is both necessary and sufficient to query the oracle $O(\sqrt{N})$ times, so there is no polynomial algorithm. However, there *is* a quantum algorithm.

1.3 Quantum Algorithm for Periodicity Determination

For further details *c.f.* Part II notes pp.60-64.

Write $A = M/r = \#\text{periods}$. We work in the state space $\mathcal{U}_M \otimes \mathcal{U}_N$ with basis $\{|i\rangle|k\rangle : \ i \in \mathbb{Z}_M, \ k \in \mathbb{Z}_N\}$.

Step 1: obtain the state

$$\frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |i\rangle|0\rangle$$

Step 2: apply U_f to obtain

$$\frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |i\rangle|f(i)\rangle$$

Step 3: measure the output register, obtaining result y. By the **Born rule**, the input register collapses to all those i such that $f(i) = y$, *i.e.* $i = x_0, \ x_0 + r, \cdots, \ x_0 + (A-1)r$ where $0 \le x_0 < r$ in the first period has $f(x_0) = y$.

We discard the output reigister to obtain

$$|\text{per}\rangle = \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |x_0 + jr\rangle$$

Note that each $0 \le x_0 < r$ occurs with probability $1/r$.

If we naively measure $|\text{per}\rangle$, the Born rule implies we get $x_0 + jr$ with $j = 0, \cdots, A-1$ chosen uniformly with probability $1/A$, *i.e.* a random element of a random period; this is a uniformly random integer in $\mathbb{Z}_M$. This is useless to us. Instead...

Step 4: apply **Quantum Fourier Transform** (QFT). Recall that Lecture 2

$$\text{QFT}|x\rangle = \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} \omega^{xy} |y\rangle$$

Fact: QFT modulo M is unitary, and can be implemented in $O(m^2)$ time, $m = \log M$. See Part II QIC notes for circuit details of implementation.

Then

$$\begin{aligned} \text{QFT}|\text{per}\rangle &= \frac{1}{\sqrt{MA}} \sum_{j=0}^{A-1} \left(\sum_{j=0}^{M-1} \omega^{(x_0+jr)y} |y\rangle \right) \\ &= \frac{1}{\sqrt{MA}} \sum_{y=0}^{M-1} \omega^{x_0 y} \left[\sum_{j=0}^{A-1} \omega^{jry} \right] |y\rangle \end{aligned}$$

Note that $[\cdots]$ is a geometric series, with ratio $\omega^{ry} = \mathrm{e}^{2\pi i r y/M} = \left(\mathrm{e}^{e\pi i/A}\right)^y$. So the sum equals zero unless y is a multiple of $A = M/r$, in which case it every term in the sum is 1 so the sum equals A. So the non-multiples of A get sifted out by QFT.

Hence, we have

$$\text{QFT}|\text{per}\rangle = \sqrt{\frac{A}{M}}\sum_{k=0}^{r-1}\omega^{x_0 kM/r}|k\frac{M}{r}\rangle$$

Then measuring QFT$|\text{per}\rangle$ we get a value $c = k_0 M/r$, with $0 \leq k_0 \leq r-1$ chosen uniformly at random. Thus we have $k_0/r = c/M$, where the values c, M are known and k_0 has been chosen at random; we want r. Note that if we are fortunate enough to have $(k_0, r) = 1$, then we can (efficiently) cancel c/M down to its lowest terms, and read off r as the denominator. But in general this will not be the case:

Theorem: (Coprimality Theorem) *The number of positive integers $< r$ that are coprime to r grows as $O\left(r/\log\log r\right)$ for large r.*

Hence the above $\mathbb{P}(k_0$ coprime to $r = O\left(1/\log\log r\right)$. So if we do it enough times, we will almost surely be successful:

Probability Lemma: If a single trial has success probability p, then we repeat k times, and for any $0 < 1-\varepsilon < 1$, we have that

$$\begin{aligned} \text{if} \quad & k = -\frac{\log\varepsilon}{p} \\ \text{then} \quad & \mathbb{P}(\geq 1 \text{ success in } k \text{ trials}) > 1-\varepsilon \end{aligned}$$

So after finding c, cancel c/M down to its lowest terms a/b (classically, in polynomial time using Euclid's algorithm). We get r as denominator b if $(k_0, r) = 1$, which happens with probability $O(1/\log\log r)$, otherwise c, M have more common factors, so $b < r$.

We don't know immediately whether that has happened or not, but we can check the b value by making two more queries to the oracle, $f(0)$ and $f(b)$; these are equal iff $b = r$.

So if we repeat this $K = O(\log\log r)$ times, then we will obtain r with any high probability we desire - and this runs in polynomial time.

Origin and utility of QFT here

Write $R = \{0, r, 2r, \ldots, (A-1)r\} \subset \mathbb{Z}_M$, and

$$|R\rangle = \frac{1}{\sqrt{A}}\sum_{k=0}^{A-1}|kr\rangle$$

$$|\text{per}\rangle = |x_0 + R\rangle = \frac{1}{\sqrt{A}}\sum_{k=0}^{A-1}|x_0 + kr\rangle$$

The problem is that the $|x_0 + kr\rangle$ terms are distributed randomly.

For each $x_0 \in \mathbb{Z}_M$, consider the map $k \mapsto k + x_0$ on $\mathbb{Z}_M$; this is the 1-1 reversible map "shift by x_0".

This gives rise to a linear map $U(x_0)$ on $\mathcal{U}_M$, and $U(x_0) : |k\rangle \to |k + x_0\rangle$ is unitary, and $|x_0 + R\rangle = U(x_0)|R\rangle$.

Since $(\mathbb{Z}_M, +)$ is an *abelian* group, these shift operators all commute, *i.e.* $U(x_0)U(x_1) = U(x_0+x_1) = U(x_1)U(x_0)$. So they have an orthonormal basis of common eigenvectors $\{|\chi_k\rangle\}_{k\in\mathbb{Z}_M}$, called the *shift-invariant* states. Note that they are not left entirely unchanged by the $U(x_0)$ operators, but they are shifted only by a constant phase factor, i.e. $U(x_0)|\chi_k\rangle = \omega(x_0,k)|\chi_k\rangle$ for all $x_0, k \in \mathbb{Z}_M$, and $|\omega(x_0,k)| = 1$.

Now consider $|R\rangle$ written in the χ-basis

$$|R\rangle = \sum_{k=0}^{M-1} a_k|\chi_k\rangle$$

where the amplitudes a_k depend only on r, and not on x_0 (obviously). Then $|\text{per}\rangle = U(x_0)|R\rangle = \sum a_k\omega(x_0,k)|\chi_k\rangle$, and measurement in the χ-basis has $\mathbb{P}(k) = |a_k\omega(x_0,k)|^2 = |a_k|^2$, independent of x_0, depending only on r. So we want to measure in this basis, but aren't allowed to do that (computationally) since the basis is too complicated.

So we introduce QFT as the unitary mapping that rotates the χ_k-basis onto the standard basis $|k\rangle$, and follow this up by a standard basis measurement.

But what does this mapping look like, and where does it come from? We need the explicit form of the shift-invariant eigenstates:

$$\begin{aligned}
|\chi_k\rangle &= \frac{1}{\sqrt{M}}\sum_{\ell=0}^{M-1} \mathrm{e}^{-2\pi i k\ell/M}|\ell\rangle \\
\implies \mathcal{U}(x_0)|\chi_k\rangle &= \frac{1}{\sqrt{M}}\sum_{\ell=0}^{M-1} \mathrm{e}^{-2\pi i k\ell/M}|\ell + x_0\rangle \\
&= \frac{1}{\sqrt{M}}\sum_{\tilde{\ell}=0}^{M-1} \mathrm{e}^{-2\pi i k(\tilde{\ell}-x_0)}|\tilde{\ell}\rangle \\
&= \mathrm{e}^{2\pi i k x_0/M}|\chi_k\rangle
\end{aligned}$$

So $\omega(x_0,k) = \mathrm{e}^{2\pi i k x_0/M}$.

The matrix of QFT^{-1} (mapping $|k\rangle$ to $|\chi_k\rangle$) has components of $|\chi_k\rangle$ as the kth column, so $[\text{QFT}^{-1}]_{\ell k} = \frac{1}{\sqrt{M}}\mathrm{e}^{-2\pi i \ell k/M}$. Since QFT is unitary, to find the inverse we need only take the conjugate transpose.

Hence $[QFT]_{k\ell} = \frac{1}{\sqrt{M}}\mathrm{e}^{2\pi i k\ell/M}$, as previously defined.

This notion of QFT in fact occurs very naturally in group theory as the *discrete Fourier transform*. The fact that this QFT is unitary means that all the group theoretic results it relies on slot in perfectly, allowing us to make as much use of this as we want in a way that we are not able classically. Lecture 3

The following algorithm was inspired by a desire to generalise the successful technique of Shor's celebrated algorithm.

2 The Hidden Subgroup Problem (HSP)

Let G be a finite group, of size $|G|$.

We are given an oracle $f : G \to X$ (where X is some set), and a promise that there is a subgroup $K < G$ such that

- f is constant on (left) cosets of K in G
- f is distinct on distinct cosets.

Problem: Determine the "hidden subgroup" K.

For instance, this might entail outputting a set of generators, or we might be happy enough with just sampling uniformly from the elements of K.

$*$ we want to solve this in time $O(\text{poly}\log|G|)$, with any constant probability $1-\varepsilon$.

Examples of problems that can be seen as HSPs:

(a) Periodicity. $f : \mathbb{Z}_M \to X$ periodic, period r, bijective within periods.

Then let $G = \mathbb{Z}_M$, $K = \{0, r, 2r, \dots\} < G$, and the cosets are $x_0 + K = \{x_0, x_0 + r, x_0 + 2r \dots\}$. It is then clear that f is constant/distinct on the cosets in the desired way.

(b) Discrete Logarithms: this was also solved by Shor in his original paper.

Take p prime, and consider $\mathbb{Z}_p^*$ the group of units modulo $p = \{1, 2, \dots, p-1\}$. We say that $g \in \mathbb{Z}_p^*$ is a *generator*, or a *primitive root modulo* p, if the powers of g generate all of $\mathbb{Z}_p^*$.

Fact: generators always exist, *i.e.* $\mathbb{Z}_p^*$ is always cyclic. For instance, $2, 3$ both generate $\mathbb{Z}_5^*$ - though $1, 4$ do not.

So any $x \in \mathbb{Z}_p^*$ can be written as $x = g^y$ for $y \in \mathbb{Z}_{p-1}$. We thus write $y = \log_g x$ for the ***discrete logarithm*** of x, to base g.

The discrete log problem is then: given a generator g and $x \in \mathbb{Z}^*p$, compute $y = \log_g x$. This is very difficult classically, and it underpins public key cryptography.

To express this as a hidden subgroup problem, consider $f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$ by $f(a, b) = g^a x^{-b} = g^{a-yb} \mod p$. Then (*check*) $f(a_1, b_1) = f(a_2, b_2)$ iff $(a_2, b_2) = (a_1, b_1) + \lambda(y, 1)$, $\lambda \in \mathbb{Z}_{p-1}$.

So we let $G = \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1}$, and $K = \{\lambda(y, 1) : \lambda \in \mathbb{Z}_{p-1}\} < G$. Then f is constant/distinct as appropriate on the cosets of K, and the generator $(y, 1)$ of K gives $y = \log_g x$.

(c) Graph Problems.

Consider a graph $A = (V, E)$, $|V| = n$. We stipulate that these are undirected, that there is at most one edge between any pair of vertices, and that the vertices are labelled by $[n] = \{1, 2, \dots, n\}$. We also might be interested in the adjacency matrix M_A, the $n \times n$ matrix given by $[M_A]_{ij} = \mathbb{I}[\{i, j\} \in E]$, which is always symmetric for an undirected graph.

The group that will be of interest to us is $P_n :=$ the permutation group of $[n]$. So $|P_n| = n! \sim \sqrt{2\pi n}(n/e)^n$ has $|P_n| \sim O(n \log n) < O(n^2)$, which is polynomial in the number of vertices. This is what we want for the running time of a graph algorithm.

The subgroup of interest is $\text{Aut}(A)$, the ***automorphism group*** of $A < P_n$ the set of permutations $\pi \in P_n$ such that for all i, j, $\{i, j\} \in E$ iff $\{\pi(i), \pi(j)\} \in E$. What this means is that after permuting the labels of the graph, we are left with the same labelled graph.

An associated HSP (non-abelian G):

Take $G = P_n$, and $X =$ the set of all labelled graphs on n vertices (equivalent to the set of all symmetric $n \times n$ 0/1-matrices).

For any $A \in X$, we consider $f_A : G \to X$, with $f_A(\pi) = \pi(A)$, *i.e.* the graph A with its vertex labels permuted by π. A little bit of thought shows that $K = \text{Aut}(A)$ is the hidden subgroup for this problem; f_A is constant on the automorphisms of A, for instance.

An important application of this is that if we can sample uniformly from K, then we can solve the ***Graph Isomorphism Problem*** (GI), which has received a lot of attention in complexity theory in recent years.

Two labelled graphs A, B each on n vertices are ***isomorphic*** if there is a bijective map (permutation) on the labels $\pi : [n] \to [n]$ such that for all $i, j \in [n]$, $\{i, j\} \in A$ iff $\{\pi(i), \pi(j)\} \in B$. In other words, A,B are the same underlying graph (*i.e.* ignoring their labels they are indistinguishable). We write $A \cong B$.

The GI Problem is then: given graphs A and B, determine whether or not they are isomorphic. This has many useful applications; *e.g.* if you can see some proteins and their structure, you may want to be able to tell which proteins are actually the same.

This can again be expressed as a non-abelian HSP, *c.f.* Sheet 1.

There is no known polynomial time classical algorithm, and in fact there is no known polynomial time quantum algorithm either. The problem is in NP, but is *not* believed to be NP-complete. A problem is NP if it is, roughly speaking, difficult to solve but easy to validate that you have the right answer once you've solved it. A problem is NP-complete if you can rephrase any other NP problem as this one, and then solve it that way - so solving an NP-complete problem solves all NP problems; it is the hardest problem in NP.

We currently do not believe that even quantum algorithms are able to solve NP-complete problems efficiently, so it is in some sense hopeless to try and work on these problems even from a quantum perspective. However, they can do NP-incomplete problems, so factoring and GI *etc*... are good candidates to attempt.

Laslo Babai (2017) found a *quasi*-polynomial time *classical* algorithm for GI; it has runtime $n^{O((\log n)^2)}$. This is slower than polynomial time, but faster than exponential time. We have the following hierarchy:

$$\begin{gathered} \text{poly}(n) < n^{O((\log n)^2)} < \exp \\ 2^{O(\log n)} < 2^{O}((\log n)^3) < 2^{O(n)} \end{gathered}$$

So in terms of exponents, these are linear/polynomial/exponential in $\log n$.