Introduction to Quantum Computing

Lecture notes for the Isogeny-based Cryptography School 2021

Changpeng Shao
School of Mathematics, University of Bristol, UK

August, 2021

Abstract

Quantum computers are designed to use quantum mechanics to outperform any possible standard, "classical" computer based only on classical physics. In this note I will introduce the field of quantum computation and some of the most important ideas in this area.

1 Introduction

Today's computers—both in theory (Turing machines) and practice (PCs, HPCs, laptops, tablets, smartphones, ...)-are based on classical physics. They are limited by locality (operations have only local effects) and by the classical fact that systems can be in only one state at the time. However, modern quantum physics tells us that the world behaves quite differently. A quantum system can be in a superposition of many different states at the same time, and can exhibit interference effects during the course of its evolution. Moreover, spatially separated quantum systems may be entangled with each other and operations may have "non-local" effects because of this.

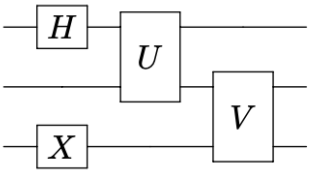
Quantum computation is the field that investigates the computational power and other properties of computers based on quantum-mechanical principles. It combines two of the most important strands of 20th-century science: quantum mechanics (developed by Planck, Einstein, Bohr, Heisenberg, Schrödinger and others in the period 1900-1925) and computer science (whose birth may be dated to Turing's 1936 paper). An important objective is to find quantum algorithms that are significantly faster than any classical algorithm solving the same problem.

Quantum computation started in the early 1980s with suggestions for analog quantum computers by Yuri Manin, Richard Feynman, and Paul Benioff, and reached more digital ground when in 1985 David Deutsch defined the universal quantum Turing machine. The following years saw only sparse activity, notably the development of the first algorithms by Deutsch and Jozsa and by Simon, and the development of quantum complexity theory by Bernstein and Vazirani. However, interest in the field increased tremendously after Peter Shor's very surprising discovery of efficient quantum algorithms for the problems of integer factorization and discrete logarithms in 1994. Since most of current classical cryptography is based on the assumption that these two problems are computationally hard, the ability to actually build and use a quantum computer would allow us to break most current classical cryptographic systems, notably the RSA system. In contrast, a quantum form of cryptography due to Bennett and Brassard is unbreakable even for quantum computers.

2 Preliminaries

Quantum computation takes place within the framework of quantum information theory. We have a system of n qubits, and choose a basis $\{|0\rangle, |1\rangle\}$ for each qubit. By taking tensor products, this gives a basis of the form $\{|x\rangle : x \in \{0,1\}^n\}$ for the whole system, where for conciseness we write $|x\rangle$ for $|x_1, x_2, \cdots, x_n\rangle = |x_1\rangle|x_2\rangle \cdots |x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$. This basis is called the computational basis. Any quantum state is a unit linear combination of $|x\rangle, x \in \{0,1\}^n$. A quantum computation is the application of some unitary operator U to some initial state (usually $|0\rangle^{\otimes n}$, which we often just write as $|0\rangle$), followed by a measurement of k of the qubits in the computational basis, giving some outcome $y \in \{0,1\}^{\otimes k}$. This outcome is then the output of the computation. If the state of a quantum computer is $\sum_{x\in\{0,1\}^n} \alpha_x|x\rangle$ for some coefficients α_x, and we measure all of the qubits, the output is x with probability $|\alpha_x|^2$.

A commonly used framework for running quantum algorithms is the quantum circuit model. A quantum circuit can be drawn as a diagram by associating each qubit with a horizontal "wire", and drawing each gate as a box across the wires corresponding to the qubits on which it acts. This is easiest to illustrate with an example: the circuit

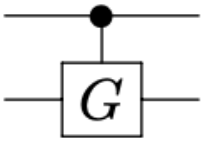


corresponds to the unitary operator $(I_2 \otimes V)(U \otimes I_2)(H \otimes I_2 \otimes X)$ on 3 qubits. Beware that a circuit is read left to right, with the starting input state on the far left, but the corresponding unitary operators act right to left!

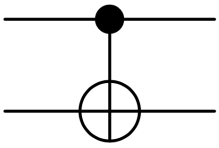
Exercise 1. *Check* $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, U = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}, V = \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{pmatrix}$ *are unitary matrices. If the initial state of the above circuit is* $|0,0,0\rangle$*, compute the final state. Show the results of measureing the first qubit.*

Some special gates turn out to be particularly useful, such as

- Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$.
- Pauli matrices $X = \sigma_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, Y = \sigma_y = \begin{pmatrix}0 & i\\ -i & 0\end{pmatrix}, Z = \sigma_z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$. We can think of the X gate as implementing a NOT operation, i.e., $X|0\rangle = |1\rangle, X|1\rangle = |0\rangle$.
- Controlled-G gates: For any gate G, the corresponding controlled-G gate CG uses an extra qubit to control whether the gate is applied or not. That is, $CG|0\rangle|x\rangle = |0\rangle|x\rangle, CG|1\rangle|x\rangle = |0\rangle G|x\rangle$. In a circuit diagram, this is denoted using a filled circle on the control line:



A particularly useful such gate is controlled-NOT (CNOT), denoted



Exercise 2. *Write out the matrix form of CG gate.*

Any classical computation which maps a bit-string (element of $\{0,1\}^n$) to another bit-string (element of $\{0,1\}^m$) can be broken down into a sequence of logical operations, each of which acts on a small number of bits (e.g. AND, OR and NOT gates). Such a sequence is called a (classical) circuit. As a first step in understanding the power of quantum computers, we would like to show that any classical circuit can be implemented as a quantum circuit, implying that quantum computation is at least as powerful as classical computation.

But there is a difficulty: in quantum mechanics, if we wish the state of our system to remain pure, the evolution that we apply has to be unitary, and hence reversible. Some classical logical operations (such as AND) are not reversible. However, reversible variants of these can be developed using the following trick. If we wish to compute an arbitrary classical operation $f : \{0,1\}^n \to \{0,1\}^m$ reversibly, we attach a so-called "ancilla" register of m bits, each originally set to 0, and modify f to give a new operation $f' : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^m$ which performs the map $f'(x,y) = (x, y \oplus f(x))$, where $\oplus$ is the bitwise XOR, i.e., addition modulo 2.

Now let us take an example. Consider the classical AND gate, by the above construction, for any $x_1, x_2, y \in \{0,1\}$, we have

$$f'(x_1, x_2, y) = (x_1, x_2, y \oplus (x_1 \wedge x_2)).$$

In matrix form, it looks like

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

This is an operation known as the Toffoli gate.

It is known that the Toffoli gate, together with the Hadamard gate, are even sufficient for universal quantum computation. That is, any quantum computation whatsoever can be approximately represented as a circuit of Toffoli and Hadamard gates. Another representative universal set of quantum gates is $\{H, X, CNOT, T\}$, where $T = e^{\pi i \sigma_z/8}$. It turns out that almost any non-trivial

set of gates is universal in this sense; therefore, we generally do not worry about the details of the gate set being used. The time (or gate) complexity of a quantum computation refers to the number of elementary gates used in a universal set.

While time complexity is a practically important measure of the complexity of algorithms, it suffers from the difficulty that it is very hard to prove lower bounds on it, and that technical details can sometimes obscure the key features of an algorithm. One way to sidestep this is to use a model which is less realistic, but cleaner and more mathematically tractable: the model of query complexity.

In this model, we assume we have access to an oracle, or "black box", to which we can pass queries, and which returns answers to our queries. Our goal is to determine some property of the oracle using the minimal number of queries. Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function, we produce a unitary operator O_f which performs the map

$$O_f : |x, y\rangle \mapsto |x, y \oplus f(x)\rangle.$$

When $m = 1$, it would also make sense to consider an oracle U_f which does not use an ancilla, but instead flips the phase of an input state $|x\rangle$ by applying the map

$$U_f : |x\rangle \mapsto (-1)^{f(x)}|x\rangle.$$

This variant is thus sometimes known as the phase oracle.

Exercise 3. *We can implement U_f by O_f in the following way: denote $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Apply O_f to $|x\rangle|-\rangle$, check that the result is $(-1)^{f(x)}|x\rangle|-\rangle$. Note that the ancilla qubit is left unchanged by this operation, which is called the phase kickback trick.*

Also note that the effect of the phase oracle is not observable if we apply it to just one computational basis state $|x\rangle$, but only if we apply it to a superposition:

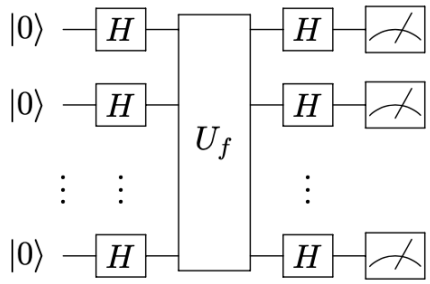
$$U_f : \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle = \sum_{x\in\{0,1\}^n} (-1)^{f(x)}\alpha_x|x\rangle.$$

3 The Deutsch-Jozsa algorithm

The Deutsch-Jozsa algorithm was the first to show a separation between the quantum and classical difficulty of a problem.

The Deutsch-Jozsa problem is defined as follows. Consider a function f that takes as input n-bit strings x and returns 0 or 1. Suppose we are promised that f is either a constant function that takes the same value on all inputs x, or a balanced function that takes each value 0 and 1 on exactly half of the inputs. The goal is to decide whether f is constant or balanced by making as few function evaluations as possible. Classically, it requires $2^{n-1} + 1$ function evaluations in the worst case. Using the Deutsch-Jozsa algorithm, the question can be answered with just one function evaluation.

We now verify that this algorithm can be implemented as an efficient quantum circuit on n qubits. Indeed, the circuit is very simple, where the last step means measurement:



Exercise 4. *Check that* $H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle$*, where* $x\cdot y = \sum x_i y_i$ *is the bitwise product. Hint* $H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle$*.*

In the above circuit, we perform

$$\begin{aligned}
|0\rangle^{\otimes n} &\mapsto \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}|y\rangle \\
&\mapsto \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{f(y)}|y\rangle \\
&\mapsto \frac{1}{2^n}\sum_{z\in\{0,1\}^n}\left(\sum_{y\in\{0,1\}^n}(-1)^{f(y)+y\cdot z}\right)|z\rangle.
\end{aligned}$$

Now let us check the possible outputs:

- If f is constant, say $f(y) = 0$ for all y, then $\sum_{y\in\{0,1\}^n}(-1)^{f(y)+y\cdot z} = 2^n\delta_z^0$, where δ is the Kronecker notation. So the output is $|0\rangle^{\otimes n}$. So if we perform measurement, we always obtain $|0\rangle^{\otimes n}$.
- If f is balanced, then the coefficient of $|0\rangle^{\otimes n}$ is 0. We therefore never obtain $|0\rangle^{\otimes n}$ by measuring the final state.

4 The Simon's algorithm

Simon's algorithm was the first quantum algorithm to show an exponential speed-up versus the best classical algorithm in solving a specific problem. This inspired the quantum algorithms based on the quantum Fourier transform, which is used in the most famous quantum algorithm: Shor's factoring algorithm.

The Simon's Problem is defined as follows: We are given an unknown black box function f which is guaranteed to be either one-to-one (maps exactly one unique output for every input) or two-to-one (maps exactly two inputs to every unique output). The problem is to determine which is the case. In the two-to-one case, there is a hidden bit string s such that $f(x) = f(x\oplus s)$ for all x. So if f is two-to-one, then we need to determine s. Note that if $s = 00...0$, then f is one-to-one.

The quantum circuit for Simon's algorithm is very similar to Deutsch-Jozsa algorithm, except that U_f is changed to O_f.

$$|0\rangle^{\otimes n} \mapsto \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}|y\rangle$$

$$\mapsto \quad \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} |y\rangle|f(y)\rangle$$
$$\mapsto \quad \frac{1}{2^n} \sum_{z\in\{0,1\}^n} \sum_{y\in\{0,1\}^n} (-1)^{y\cdot z}|z\rangle|f(y)\rangle.$$

We can rewrite the quantum state obtained in the second step as

$$\frac{1}{\sqrt{2^n}} \sum_{y\in A} (|y\rangle + |y \oplus s\rangle)|f(y)\rangle,$$

where A is a subset of $\{0,1\}^n$ of size 2^{n-1} if f is two-to-one and $A = \{0,1\}^n$ if f is one-to-one. In the former case, f is one-to-one in A. As a result, the final state is

$$\frac{1}{2^n} \sum_{z\in\{0,1\}^n} \sum_{y\in A} (-1)^{y\cdot z}(1 + (-1)^{s\cdot z})|z\rangle|f(y)\rangle.$$

So if f is two-to-one, we only obtain z such that $s \cdot z = 0$ after measurement the first n qubits. Specifically, if we run the above process $n-1$ times, we obtain $z_1, \ldots, z_{n-1}$ such that $s \cdot z_i = 0$ for all i. This is a system of $n-1$ linear equations in n unknowns, from which we can determine s.

Exercise 5. *Show the probability that $z_1, \ldots, z_{n-1}$ are linear independent is*

$$\prod_{j=1}^{n} (1 - \frac{1}{2^j}),$$

which is grater than $1 - \sum_{j=1}^{n} 1/2^j = 1/2 - 1/2^n$. Hint: Assume that $z_1, \ldots, z_m$ are linear independent, then the next bit string z_{m+1} is linearly independent to them is $(2^n - 2^m)/2^n$.

5 The Quantum Fourier Transform

We now introduce an important unitary transformation which is used in a number of different contexts in quantum information theory: the quantum Fourier transform (QFT) over $\mathbb{Z}_N$. The QFT is the map

$$Q_N : |x\rangle \to \frac{1}{\sqrt{N}} \sum_{y\in\mathbb{Z}_N} \omega^{xy}|y\rangle,$$

where $\omega = e^{2\pi i/N}$. The QFT is exactly the same transformation as the Discrete Fourier Transform (DFT) used for classical computation and signal processing, up to the nonstandard normalisation of $1/\sqrt{N}$.

Exercise 6. *Prove that Q_N is unitary.*

One of the most important applications of the QFT is determining the period of a periodic function. Imagine we are given access to an oracle function $f : \mathbb{Z}_N \to \mathbb{Z}_M$, for some integers N and M, such that:

- f is periodic: there exists r such that r divides N and $f(x + r) = f(x)$ for all $x \in \mathbb{Z}_N$;
- f is one-to-one on each period: for all pairs (x, y) such that $|x - y| < r, f(x) \neq f(y)$.

Our task is to determine r.

The periodicity determination algorithm is presented as below:

1. We start with the state $|0\rangle|0\rangle$, where the first register is dimension N, and the second dimension M.
2. Apply Q_N to the first register.
3. Apply O_f to the two registers.
4. Measure the second register.
5. Apply Q_N to the first register.
6. Measure the first register; let the answer be k.
7. Simplify the fraction k/N as far as possible and return the denominator.

Now let us see more details: The initial sequence of operations which occur during the algorithm is

$$|0\rangle|0\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{x \in \mathbb{Z}_N} |x\rangle|0\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{x \in \mathbb{Z}_N} |x\rangle|f(x)\rangle.$$

When the second register is measured, we receive an answer, say z. By the periodic and one-to-one properties of f, all input values $x \in \mathbb{Z}_N$ for which $f(x) = z$ are of the form $x_0 + jr$ for some x_0 and integer j. The state therefore collapses to something of the form

$$\sqrt{\frac{r}{N}} \sum_{j=0}^{N/r-1} |x_0 + jr\rangle.$$

After we apply the QFT, we get the state

$$\sqrt{\frac{r}{N^2}} \sum_{y \in \mathbb{Z}_N} \omega^{x_0 y} \left(\sum_{j=0}^{N/r-1} \omega^{jry} \right) |y\rangle.$$

Note that if $\omega^{ry} \neq 1$ we have

$$\sum_{j=0}^{N/r-1} \omega^{jry} = \frac{\omega^{ry(N/r)} - 1}{\omega^{ry} - 1} = 0.$$

When $\omega^{ry} = 1$, i.e., N divides ry, the summation equals N/r. Hence we obtain the following state

$$\frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} \omega^{x_0 N l/r} |Nl/r\rangle.$$

When we perform the final measurement, we receive an outcome $k = l_0 N/r$, for some l_0 picked uniformly at random from $0, \ldots, r-1$. We know that

$$\frac{k}{N} = \frac{l_0}{r}.$$

In this equation, we know N and k and would like to determine r. If it happened that l_0 were coprime to r, we could cancel the fraction on the left-hand side and output the denominator. What is the probability that we are lucky in this way? [1]

[1] Fix an positive integer a and pick b uniformly at random from the integers between 0 and a. Then the probability that b is coprime to a is at least $1/\log\log a$.

6 Integer factorisation

The main application of periodicity determination is Shor's quantum algorithm for integer factorisation. Given an n-digit integer N as input, this algorithm outputs a non-trivial factor of N (or that N is prime, if this is the case) with high success probability in time $O(n^3)$. The best classical algorithm known (the general number field sieve) runs in time $e^{O(n^{1/3}log^{2/3}n)}$. In fact, this is a heuristic bound and this algorithm's worst-case runtime has not been rigorously determined; the best proven bound is somewhat higher. Shor's algorithm thus achieves a super-polynomial improvement. This result might appear only of mathematical interest, were it not for the fact that the very widely-used RSA public-key cryptosystem relies on the hardness of integer factorisation. Shor's efficient factorisation algorithm implies that this cryptosystem is insecure against attack by a large quantum computer. Although no large-scale quantum computer yet exists, this result has already had major implications for cryptography, as national security agencies start to move away from the RSA cryptosystem.

Unfortunately, proving correctness of Shor's algorithm requires going through a number of technical details.

- First we need to show that factoring reduces to a periodicity problem – though in this case an approximate periodicity problem. This part uses only classical number theory.
- Then we need to show that periodicity can still be determined even in the setting where the input function is only approximately periodic. This part uses the theory of continued fractions.

The basic skeleton of the quantum factorisation algorithm is given as follows:

1. Choose $1 < a < N$ uniformly at random.
2. Compute $b = \gcd(a, N)$. If $b > 1$ output b and stop. [2]
3. Determine the order r of a modulo N. If r is odd, the algorithm has failed; terminate.
4. Compute $s = \gcd(a^{r/2} - 1, N)$. If $s = 1$, the algorithm has failed; terminate.
5. Output s.

We start by showing that this algorithm does work, assuming that the subroutines used all work correctly. If a is coprime to N, there exists r such that $a^r \equiv 1 \mod N$. If, further, such an r is even, we can factor

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

So N divides the product $(a^{r/2} - 1)(a^{r/2} + 1)$. If neither term in the product is a multiple of N itself, N must divide partly into each of them, so if we computed $\gcd(a^{r/2} \pm 1, N)$, we would obtain a factor of N. Because r is the smallest integer x such that $a^x \equiv 1 \mod N$, we know that $a^{r/2} \pm 1$ are not divisible by N.

It remains to show how to implement step 3. Consider the function $f : \mathbb{Z} \to \mathbb{Z}_N$ defined by

$$f(x) = a^x \mod N.$$

[2] gcd = greatest common divisor.

We can check that f is periodic with period r and one-to-one on each period. However, although this function is periodic on the whole domain $\mathbb{Z}$, we will need to truncate it to a finite size. If we knew what the period was, we could choose this size to make the function periodic again, but of course we do not know this in advance. This will lead to the function becoming no longer exactly periodic, but just approximately periodic. We will not go deep in this step.

7 Quantum phase estimation

We now discuss an important primitive used in quantum algorithms called quantum phase estimation, which provides a different and unifying perspective on the quantum algorithms which you have just seen. Phase estimation is once again based on the QFT over $\mathbb{Z}_N$, where $N = 2^n$.

Imagine we are given a unitary operator U. We are also given a state $|\psi\rangle$ which is an eigenvector of $U : U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$ for some real $\theta \in [0, 1)$. We would like to determine θ to n bits of precision, for some arbitrary n.

To do so, we prepend an n qubit register to $|\psi\rangle$, initially in the state $|0\rangle$, and create the state

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle|\psi\rangle$$

by applying a Hadamard gate to each qubit in the first register. We then apply the control unitary operator

$$\sum_{x=0}^{N-1} |x\rangle\langle x| \otimes U^x$$

to it. This operator can be thought of as performing the map where if the first register contains x, we apply Ux times to the second register. After applying this operator, we are left with the state

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{2\pi i\theta x}|x\rangle|\psi\rangle.$$

We now apply QFT to the first register $|x\rangle$, then we obtain

$$\frac{1}{N} \sum_{y=0}^{N-1} \left(\sum_{x=0}^{N-1} e^{2\pi i(\theta - y/N)x} \right) |y\rangle|\psi\rangle.$$

As a geometric series, the coefficient of $|y\rangle|\psi\rangle$ is close to a constant when $\theta \approx y/N$ and close to 0 otherwise. Indeed, denote $\theta - y/N = \delta_y$, then

$$\frac{1}{N} \left| \sum_{x=0}^{N-1} e^{2\pi i\delta_y x} \right| = \frac{1}{N} \left| \frac{e^{2\pi i\delta_y N} - 1}{e^{2\pi i\delta_y} - 1} \right| = \frac{1}{N} \left| \frac{\sin(\pi\delta_y N)}{\sin(\pi\delta_y)} \right|.$$

If $\delta_y N \leq 1/2$, then the above quantity is lower bounded by

$$\geq \frac{1}{N} \left| \frac{2\delta_y N}{\pi\delta_y} \right| = \frac{2}{\pi}$$

based on the fact $\sin(t) \geq 2t/\pi$ when $|t| \leq \pi/2$.

As an application, we can reconsider the integer factorisation problem. Recall that the main step is finding the order of an integer. Below we use quantum phase estimation to solve this problem. Consider the unitary

$$U|x\rangle = |ax \mod N\rangle.$$

A simple calculation shows that the states defined by

$$|\psi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi isk/r} |k\rangle$$

for integer $0 \leq s \leq r-1$ are eigenstates of U, because $U|\psi_s\rangle = e^{2\pi is/r}|\psi_s\rangle$. Observe that

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\psi_s\rangle.$$

When we apply quantum phase estimation with initial state $|0\rangle|1\rangle$, we will obtain a state close to

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\theta_s\rangle|\psi_s\rangle,$$

where $s/r \approx \theta_s$. From θ_s we can determine r with the method of continued fraction expansion. This step is similar to the step omited in the quantum integer factorisation algorithm.

8 Grover algorithm

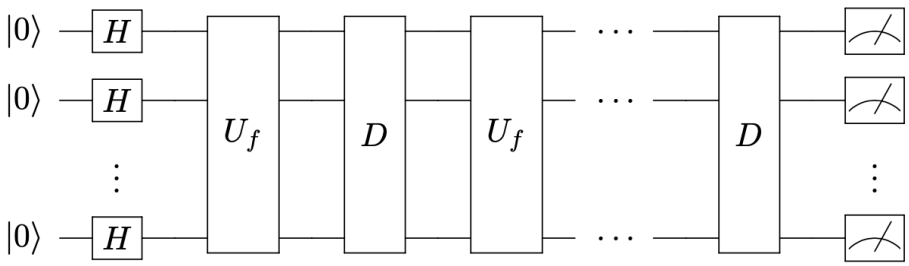
A simple example of a problem that fits into the query complexity model is unstructured search on a set of N elements for a unique marked element. In this problem we are given access to a function $f : \mathbb{Z}_N \to \{0, 1\}$ with the promise that $f(x_0) = 1$ for a unique "marked" element x_0. Our task is to output x_0.

It is intuitively clear that the unstructured search problem should require about N queries to be solved (classically!). In the quantum setting, we will see that the unstructured search problem can be solved with significantly fewer queries.

For simplicity, assume that $N = 2^n$ for some integer n (this is not an essential restriction). In this case, we can associate each element of $\mathbb{Z}_N$ with an n-bit string. Then Grover's algorithm is described as follows:

1. Prepare $|0^n\rangle$
2. Apply $H^{\otimes n}$
3. Repeat the following operations $O(\sqrt{N})$ times:
   (a) Apply U_f
   (b) Apply $D := -H^{\otimes n} U_0 H^{\otimes n}$, where U_0 maps $|0^n\rangle$ to $-|0^n\rangle$ and keeps all other basis states invariant.
4. Measure all the qubits and output the result.

In circuit diagram form, Grover's algorithm looks like this:



It may be far from clear initially why this algorithm works, or indeed whether it does work. We now describe Grover's algorithm in detail. First, we denote

$$|\phi\rangle = H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \mathbb{Z}_N} |x\rangle.$$

It formally equals

$$|\phi\rangle = \frac{1}{\sqrt{N}}|x_0\rangle + \frac{\sqrt{N-1}}{\sqrt{N}}|x_0^\perp\rangle,$$

where

$$|x_0^\perp\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \in \mathbb{Z}_N, x \neq x_0} |x\rangle.$$

We also denote $\sin\theta = 1/\sqrt{N}$ and $\cos\theta = \sqrt{N-1}/\sqrt{N}$. In step 3, we can denote $U_0 = I - 2|0^n\rangle\langle 0^n|$ so that $D = -(I - 2|\phi\rangle\langle\phi|)$. So in step 3, first U_f maps $|\phi\rangle$ to

$$-\sin\theta|x_0\rangle + \cos\theta|x_0^\perp\rangle.$$

Then apply D to obtain

$$\begin{aligned} & \sin\theta(I - 2|\phi\rangle\langle\phi|)|x_0\rangle - \cos\theta(I - 2|\phi\rangle\langle\phi|)|x_0^\perp\rangle \\ = \; & \sin\theta(|x_0\rangle - 2\sin\theta(\sin\theta|x_0\rangle + \cos\theta|x_0^\perp\rangle)) - \cos\theta(|x_0^\perp\rangle - 2\cos\theta(\sin\theta|x_0\rangle + \cos\theta|x_0^\perp\rangle)) \\ = \; & \sin(3\theta)|x_0\rangle + \cos(3\theta)|x_0^\perp\rangle. \end{aligned}$$

As we have seen, DU_f is the product of two reflections in the plane spanned by $\{|x_0\rangle, |x_0^\perp\rangle\}$. So DU_f is a rotation of angle 2θ.

Hence, after T steps of iteration we obtain

$$\sin((2T+1)\theta)|x_0\rangle + \cos((2T+1)\theta)|x_0^\perp\rangle.$$

Since $\sin\theta = 1/\sqrt{N}$, we have $\theta \approx 1/\sqrt{N}$. To make $\sin((2T+1)\theta)$ close to 1, we can choose T so that $(2T+1)\theta \approx \pi/2$. Namely, $T \approx \sqrt{N}\pi/4 - 1/2$.

9 Further readings

You may find the following lecture notes and books useful:

- Lecture Notes on Quantum Algorithms, Andrew Childs, University of Maryland
  `http://www.cs.umd.edu/~amchilds/qa/`
  An excellent resource for more advanced topics on quantum algorithms.

- Quantum Computing: Lecture Notes, Ronald de Wolf, QuSoft, CWI and University of Amsterdam

  `https://export.arxiv.org/abs/1907.09415`

  A comprehensive lecture note for more topics on quantum computing.

- Quantum Computation and Quantum Information, Nielsen and Chuang

  Cambridge University Press, 2001

  The Bible of quantum computing.