Part III **Michaelmas 2011**

QUANTUM COMPUTATION

Lecture notes

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Contents

For updates to these notes, see http://www.damtp.cam.ac.uk/user/am994/.
Version 1.11 (February 17, 2012).

1 Lower bounds on quantum search

We have seen that Grover's algorithm finds a marked item in an unstructured database of n items using only $O(\sqrt{n})$ queries. This is already a remarkable result. However, it would be more remarkable still if there existed a quantum algorithm which could search such a database significantly more quickly – e.g. using $O(\log n)$ queries. It is a significant understatement to say that such an algorithm would have many applications; indeed, it would revolutionise computer science by allowing any problem whose solution can be *verified* efficiently to be *solved* efficiently, or in terms of complexity classes proving that $\mathbf{NP} \subseteq \mathbf{BQP}$.

Unfortunately, we will now show that such an algorithm cannot exist, and indeed that Grover's algorithm is optimal up to constant factors.

1.1 A note on big-O notation

In order for clarity in our discussion of lower and upper bounds, it will be helpful to formalise some standard notation relating to asymptotic complexity. First, we will say that $f(n) = O(g(n))$ if there exist real $c > 0$ and integer $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \leq c\, g(n)$. Conversely, $f(n) = \Omega(g(n))$ if there exist real $c > 0$ and integer $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \geq c\, g(n)$. Clearly, $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$. Finally, when we say "bounded error" below, we mean with error probability upper bounded by some constant below $1/2$.

1.2 The query complexity model

We will prove a lower bound on the number of queries required to solve the unstructured search problem for a unique marked item in a database of N items (as discussed in Section 8.1 of the previous notes). Recall that in this problem we have a function $f : \{0,1\}^n \to \{0,1\}$ (where $N = 2^n$) such that f takes the value 1 on precisely one input x_0. We are given access to f via a unitary operator U_f defined by

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

and would like to determine x_0 using the smallest possible number of uses of ("queries to") U_f.

Consider an arbitrary quantum algorithm $\mathcal{A}$ which completes this task. Such an algorithm can be expressed as a quantum circuit operating on three registers: an n-qubit input register, a 1-qubit output register and a workspace register of m qubits, where m is arbitrary. A computational basis state of such a system can therefore be written as $|x\rangle|y\rangle|w\rangle$ for $x \in \{0,1\}^n$, $y \in \{0,1\}$, $w \in \{0,1\}^m$. Note that Grover's algorithm does not in fact use any additional workspace (i.e. $m = 0$), but if we want to put a lower bound on more general algorithms we should allow the use of such a workspace.

The operation of a completely general quantum query algorithm can be expressed as follows. Starting in the state $|0\rangle|0\rangle|0\rangle$, apply a sequence of arbitrary but fixed unitary operations $V_1, V_2, \ldots$ which do not depend on f, interspersed with oracle operations U_f (which depend on f and operate on the input and output registers), followed by a final measurement, after which the algorithm outputs x_0. Without loss of generality, we assume that the final step of the algorithm is to measure n qubits of the workspace and to output the measurement result.

The circuit diagram in Figure 1 illustrates such a quantum query algorithm.

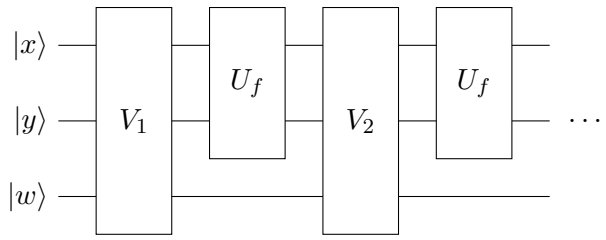


Figure 1: General quantum query algorithm operating on input register $|x\rangle$, output register $|y\rangle$, and workspace $|w\rangle$.

1.3 Quantum search lower bound

In order to prove our lower bound, we will modify the unstructured search problem slightly, by also allowing the possibility for no items to be marked. That is, we will consider the following problem.

Unstructured search (modified)
Input: *a black box for a boolean function* $f : \{0,1\}^n \to \{0,1\}$
Promise: *there is either a unique "marked"* x *such that* $f(x) = 1$*, or* $f(x) = 0$ *for all* x*.*
Problem: *find this special* x*, or output that no such* x *exists.*

The complexity of solving the modified problem is very similar to that of solving the original problem; indeed, it is an easy exercise to show that any (classical or quantum) algorithm for the original problem can be converted into an algorithm for the modified problem at the cost of one additional query.

We will now prove the following lower bound on quantum search.

Theorem 1. *Let* $\mathcal{A}$ *be a quantum algorithm which solves the (modified) unstructured search problem using* T *queries with failure probability* $\epsilon < 1/2$*. Then*

$$T \geq \left(\frac{1}{2} - \epsilon\right)\sqrt{N}.$$

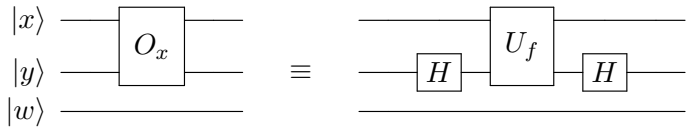
So, when ϵ *is any constant strictly less than* $1/2$*,* $T = \Omega(\sqrt{N})$*.*

The intuition behind the lower bound is as follows. For any x, in order to distinguish the case where the marked element is x from the case where there is no marked element, the quantum algorithm must put significant "weight" on queries to x during the algorithm. If the position of the marked element is picked uniformly at random, the algorithm must put significant weight on all N positions. This is only possible if the number of queries is $\Omega(\sqrt{N})$. The proof will use the following lemma, which we prove afterwards.

Lemma 2. *Let* $|\psi_1\rangle$ *and* $|\psi_2\rangle$ *be quantum states such that* $\||\psi_1\rangle - |\psi_2\rangle\| \leq \epsilon$*. Given a quantum state* $|\psi\rangle$ *promised to be either* $|\psi_1\rangle$ *or* $|\psi_2\rangle$*, no measurement of* $|\psi\rangle$ *in the computational basis followed by arbitrary post-processing can determine which is the case with worst-case error probability lower than* $(1 - \epsilon)/2$*.*

Proof of Theorem 1. Assume the marked element is x, so $f(x) = 1$. Write O_x for the matrix produced by applying H to the output register, then applying U_f to the input and output registers, then applying H to the output register. As we only count the number of queries made to the oracle

U_f in the query complexity model and we are allowed to apply arbitrary fixed unitary operators before and after oracle calls, counting queries to O_x is the same as counting queries to U_f. We illustrate this with the following circuit diagram.



It can readily be verified that

$$O_x|z\rangle|y\rangle|w\rangle = (-1)^{y\delta_{xz}}|z\rangle|y\rangle|w\rangle,$$

where $\delta_{xz} = 1$ if $x = z$, and $\delta_{xz} = 0$ otherwise, so this application of H on the output register diagonalises U_f. Fix a quantum query algorithm that makes T queries to the input and let the state of the system after $t \leq T$ queries, given that $f(x) = 1$, be

$$|\psi_{x,t}\rangle = O_x V_t O_x V_{t-1} \dots O_x V_1 |0\rangle.$$

We will compare the states $\{|\psi_{x,t}\rangle\}$ with the result of applying the V_t operators without the oracle queries, or equivalently when there is no marked item (so U_f and O_x are both equal to the identity matrix). Define the state

$$|\phi_t\rangle = V_t V_{t-1} \dots V_1 |0\rangle.$$

We would like to upper bound the quantity $\||\psi_{x,T}\rangle - |\phi_T\rangle\|$. To do so, we introduce a "deviation" vector $|D_{x,t}\rangle$ which, intuitively, measures the effect of an oracle call at time t:

$$|D_{x,t}\rangle = O_x|\phi_t\rangle - |\phi_t\rangle.$$

In terms of these deviation vectors, we have

$$\begin{aligned}
|\psi_{x,1}\rangle &= |\phi_1\rangle + |D_{x,1}\rangle \\
|\psi_{x,2}\rangle &= |\phi_2\rangle + |D_{x,2}\rangle + O_x V_2 |D_{x,1}\rangle \\
&\vdots \\
|\psi_{x,T}\rangle &= |\phi_T\rangle + |D_{x,T}\rangle + O_x V_T |D_{x,T-1}\rangle + \dots + O_x V_T \dots O_x V_2 |D_{x,1}\rangle.
\end{aligned}$$

Thus, by the triangle inequality,

$$\begin{aligned}
\||\psi_{x,T}\rangle - |\phi_T\rangle\| &= \||D_{x,T}\rangle + O_x V_T |D_{x,T-1}\rangle + \dots + O_x V_T \dots O_x V_2 |D_{x,1}\rangle\| \\
&\leq \sum_{t=1}^{T} \||D_{x,t}\rangle\| = \sum_{t=1}^{T} \|(O_x - I)|\phi_t\rangle\|.
\end{aligned}$$

On the other hand, as we are assuming the algorithm fails with probability at most ϵ in the worst case, by Lemma 2 we have

$$\|V_{T+1}|\psi_{x,T}\rangle - V_{T+1}|\phi_T\rangle\| = \||\psi_{x,T}\rangle - |\phi_T\rangle\| \geq 1 - 2\epsilon,$$

because the algorithm must be able to distinguish between x being the marked item, and there being no marked items. Combining these two inequalities and squaring both sides, we obtain

$$(1 - 2\epsilon)^2 \leq \left(\sum_{t=1}^{T} \|(O_x - I)|\phi_t\rangle\| \right)^2 \leq T \sum_{t=1}^{T} \|(O_x - I)|\phi_t\rangle\|^2,$$

valid for any $\epsilon \leq 1/2$, where the second inequality is Cauchy-Schwarz. This inequality holds for all x. We can therefore take the average of the right-hand side over x to find

$$(1-2\epsilon)^2 \leq \frac{T}{N}\sum_{t=1}^{T}\sum_{x\in\{0,1\}^n} \|(O_x - I)|\phi_t\rangle\|^2 \leq T^2 \max_{|\phi\rangle} \frac{1}{N}\sum_{x\in\{0,1\}^n} \|(O_x - I)|\phi\rangle\|^2.$$

Now observe that $O_x - I$ performs the following map:

$$(O_x - I)|z\rangle|y\rangle|w\rangle = -2y\delta_{xz}|z\rangle|y\rangle|w\rangle.$$

Thus

$$(1-2\epsilon)^2 \leq 4T^2 \max_{|\phi\rangle} \frac{1}{N}\sum_{x\in\{0,1\}^n}\sum_{w} |\langle\phi|x\rangle|1\rangle|w\rangle|^2 = \frac{4T^2}{N}.$$

We therefore have the inequality

$$T \geq \left(\frac{1}{2} - \epsilon\right)\sqrt{N},$$

which completes the proof. □

1.4 Proof of Lemma 2

We still need to prove this technical lemma, which we restate for clarity.

Lemma. *Let $|\psi_1\rangle$ and $|\psi_2\rangle$ be quantum states such that $\||\psi_1\rangle - |\psi_2\rangle\| \leq \epsilon$. Given a quantum state $|\psi\rangle$ promised to be either $|\psi_1\rangle$ or $|\psi_2\rangle$, no measurement of $|\psi\rangle$ in the computational basis followed by arbitrary post-processing can determine which is the case with worst-case error probability lower than $(1-\epsilon)/2$.*

Proof. We will show that the probability distributions p_1, p_2 over the measurement outcomes obtained for $|\psi_1\rangle$, $|\psi_2\rangle$ satisfy

$$\frac{1}{2}\sum_x |p_1(x) - p_2(x)| \leq \epsilon.$$

This will imply the lemma, because no strategy for choosing between p_1 and p_2, given a single sample from an unknown distribution which might be either, can achieve a worst-case error probability smaller than

$$\frac{1}{2} - \frac{1}{4}\sum_x |p_1(x) - p_2(x)|.$$

This is a standard result from probability theory; to see it directly, observe that the average error (over the choice of p_1, p_2) of any randomised or deterministic strategy which infers that the distribution was p_1 with probability $s(x)$, given outcome x, is

$$\begin{aligned}
\frac{1}{2}\left(\sum_x p_1(x)(1-s(x)) + \sum_x p_2(x)s(x)\right) &= \frac{1}{2} - \frac{1}{2}\sum_x s(x)(p_1(x) - p_2(x)) \\
&\geq \frac{1}{2} - \frac{1}{2}\sum_{x, p_1(x)\geq p_2(x)} (p_1(x) - p_2(x)) \\
&= \frac{1}{2} - \frac{1}{4}\sum_x |p_1(x) - p_2(x)|,
\end{aligned}$$

so the worst-case error probability can only be higher. Now, writing $a_x = |\langle x|\psi_1\rangle|$, $b_x = |\langle x|\psi_2\rangle|$, we have

$$\begin{aligned}
\sum_x |p_1(x) - p_2(x)| &= \sum_x |a_x^2 - b_x^2| = \sum_x |a_x - b_x|(a_x + b_x) \\
&\leq \sqrt{\sum_x (a_x - b_x)^2} \sqrt{\sum_x (a_x + b_x)^2} \\
&\leq \sqrt{\sum_x |\langle x|\psi_1\rangle - \langle x|\psi_2\rangle|^2} \left(\sqrt{\sum_x a_x^2} + \sqrt{\sum_x b_x^2} \right) \\
&= 2\||\psi_1\rangle - |\psi_2\rangle\|,
\end{aligned}$$

where the first inequality is Cauchy-Schwarz, the second inequality combines the reverse triangle inequality $||x| - |y|| \leq |x - y|$ and the triangle inequality for the Euclidean norm, and the final equality uses the fact that $|\psi_1\rangle$ and $|\psi_2\rangle$ are unit vectors. □

1.5 Further reading

The lower bound on unstructured quantum search was originally proven by Bennett et al. [3] – notably *before* Grover's quantum search algorithm was found. The lower bound can in fact be improved to show that Grover's algorithm is exactly optimal and its performance cannot be improved by even one query [19]. However, this only holds in the worst-case setting; if the goal is to minimise the *average* number of queries to the database, it turns out that one can do better, but only by a constant factor. There is a significant generalisation of the above lower bound to a general technique for quantum lower bounds known as the adversary method; for an accessible review, see [10].

2 General lower bounds on quantum query complexity

We now turn to a different technique for lower bounding quantum query complexity known as the polynomial method, which has the advantage of both being fairly straightforward to apply and having interesting consequences.

In particular, we will work towards the following result.

Theorem 3. *Let $f : \{0,1\}^n \to \{0,1\}$ be a total boolean function computed with bounded error by a quantum algorithm using T queries to the input. Then there is a deterministic classical algorithm which computes f with certainty using $O(T^6)$ queries to the input.*

A total function is simply one which has no promise on the input. It is important to note that in this theorem we have changed terminology somewhat from our previous query complexity language. Previously, we thought of problems in the quantum query complexity model as follows: given oracle access U_f to a function f, with some promise on f, output some property of f (e.g. whether f is constant or balanced) using the minimum number of queries to U_f. Now we consider the following setting: Given oracle access to a bit-string x via an oracle U_x which performs the map

$$U_x|i\rangle|y\rangle = |i\rangle|y \oplus x_i\rangle,$$

output $g(x)$, for some known function g. This does not change the model, but is merely a change of language. Observe that there are indeed some total functions which require fewer quantum than classical queries to be computed. For example, the OR function on n bits is defined by $\mathrm{OR}_n(x) = 1$ if and only if $x \neq 0^n$. OR_n can be computed with bounded error by Grover's algorithm for unstructured search with an unknown number of good items, using only $O(\sqrt{n})$ quantum queries to the input. Other examples of functions we will consider below are the AND and PARITY functions on n bits defined by $\mathrm{AND}_n = x_1 \wedge x_2 \wedge \cdots \wedge x_n$ (i.e. $\mathrm{AND}_n(x) = 1$ if and only if $x = 1^n$) and $\mathrm{PARITY}_n = x_1 \oplus x_2 \cdots \oplus x_n$ (i.e. the sum of the bits of x modulo 2). These functions are all *symmetric*: they depend only on the Hamming weight $|x|$, i.e. the number of ones in x.

Note that Theorem 3 does not contradict the exponential separations between quantum and classical algorithms which you have seen for period-finding (Shor's algorithm) and Simon's problem. This is because these functions are *partial*, or in other words have a promise on the input. This result thus highlights the importance of promises in quantum algorithms.

2.1 The polynomial method

The polynomial method is (unsurprisingly) based on understanding polynomials. Any function $f : \{0,1\}^n \to \mathbb{R}$ can be written as a polynomial in n variables $x_1, \ldots, x_n$. As each variable x_i only takes values 0 or 1, we never need to raise a variable to a power greater than 1. Such polynomials are said to be *multilinear*. Any function $f : \{0,1\}^n \to \mathbb{R}$ in fact has a unique representation as a multilinear polynomial. The *degree* $\deg(f)$ is the largest number of variables in any term in the polynomial representation of f.

For example, the AND function $f(x_1, x_2) = x_1 \wedge x_2$ has polynomial representation $x_1 x_2$, whereas the PARITY function $f(x_1, x_2) = x_1 \oplus x_2$ has polynomial representation $x_1 + x_2 - 2x_1x_2$. Both of these functions have degree 2.

The first step to proving Theorem 3 is the following important observation.

Lemma 4. *Let $\mathcal{A}$ be a quantum query algorithm which makes T queries to the input. Let $p(x)$ be the probability that $\mathcal{A}$ outputs 1 on input x. Then p is a polynomial of degree at most $2T$ in real variables $x_1, \ldots, x_n$.*

Proof. The proof is by induction. Let the state of the computer just before the $(t+1)$'st query, when the input is x, be

$$\left|\psi_x^t\right\rangle = \sum_{i,y,w} \alpha_{iyw}^t(x)|i\rangle|y\rangle|w\rangle.$$

We will show by induction that, for any i, y, w, t, the function $\alpha_{iyw}^t(x)$ is a (complex-valued) polynomial of degree at most t. First, when $t = 0$, $\alpha_{iyw}^t(x)$ is constant with respect to x. $\mathcal{A}$ consists of alternating between oracle calls and unitary operators which do not depend on the input x. The latter operators cannot increase $\max_{i,y,w} \deg(\alpha_{iyw}^t)$. What about the oracle calls? As discussed in Section 1.3 of these notes, U_x can be diagonalised by the application of Hadamard gates on the output qubit to give a diagonal matrix O_x satisfying

$$O_x|i\rangle|y\rangle|w\rangle = (-1)^{yx_i}|i\rangle|y\rangle|w\rangle = (1 - 2yx_i)|i\rangle|y\rangle|w\rangle.$$

An application of O_x maps $\alpha_{iyw}^t(x) \mapsto (1 - 2yx_i)\alpha_{iyw}^t(x)$, which is a polynomial of degree at most $t + 1$. We can assume without loss of generality that the last step of the algorithm is to measure

the output qubit and return the measurement result. When this qubit is measured after T queries, the probability that the output is 1 is

$$\sum_{i,w} |\alpha_{i1w}^T(x)|^2.$$

For each i and w, the real and imaginary parts of α_{i1w}^T are polynomials of degree at most T. Thus the above sum is a polynomial of degree at most $2T$. □

Lemma 4 allows us to go from lower bounds on the degree of polynomials (which are well-studied) to lower bounds on quantum query complexity. In the case of exact algorithms, $p(x) = f(x)$ for all x, so we immediately see that any quantum algorithm computing a boolean function f exactly using T queries must satisfy $T \geq \frac{1}{2}\deg(f)$. This allows us to prove strong lower bounds easily for certain functions.

Fact 5. $\deg(AND_n) = \deg(OR_n) = \deg(PARITY_n) = n$. *Thus, any quantum algorithm computing any of these functions exactly must make at least $n/2$ queries to the input.*

The lower bound on PARITY is tight, as you have seen that the parity of 2 bits can be computed exactly using only one quantum query using the Deutsch-Jozsa algorithm, which allows PARITY_n to be computed using $\lceil n/2 \rceil$ quantum queries. The bounds on AND and OR can actually be improved to show that any quantum algorithm computing either of these functions exactly must make precisely n queries to the input, and hence can achieve no speed-up at all over classical algorithms.

We would also like to prove lower bounds on quantum algorithms which do not necessarily succeed with certainty. We make the following definition.

Definition 1. Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. The ϵ-approximate degree $\widetilde{\deg}_\epsilon(f)$ is defined as the minimum of $\deg(\tilde{f})$ over all functions $\tilde{f} : \{0,1\}^n \to \mathbb{R}$ such that $|f(x) - \tilde{f}(x)| \leq \epsilon$ for all $x \in \{0,1\}^n$. The *approximate degree* of f is defined as $\widetilde{\deg}(f) = \widetilde{\deg}_{1/3}(f)$.

Corollary 6. *Let $f : \{0,1\}^n \to \{0,1\}$ be a total boolean function computed with error at most $1/3$ on every input by a quantum algorithm using T queries. Then $T \geq \frac{1}{2}\widetilde{\deg}(f)$.*

Proof. By Lemma 4, any quantum algorithm that makes T queries and achieves worst-case failure probability at most $1/3$ gives a polynomial of degree at most $2T$ that approximates f to within $1/3$ on every input. The corollary follows from the definition of $\widetilde{\deg}(f)$. □

Theorem 3 will now follow from a classical *upper* bound in terms of $\widetilde{\deg}(f)$. We will show that, if $\widetilde{\deg}(f) = d$, there is a classical algorithm that makes $O(d^6)$ queries to the input and computes f with certainty.

2.2 Block sensitivity and approximate degree

The bound required to prove Theorem 3 is obtained by chaining together several complexity measures of boolean functions. The first such complexity measure is *block sensitivity*.

Definition 2. The block sensitivity $\text{bs}_x(f)$ of f on x is the maximum number b such that there are disjoint sets $B_1, \ldots, B_b$ for which $f(x) \neq f(x^{B_i})$ for all $1 \leq i \leq b$, where x^B is the bit string defined to be equal to x with the bits in B flipped. The block sensitivity of f is $\text{bs}(f) = \max_x \text{bs}_x(f)$.

It turns out that functions with high block sensitivity have high approximate degree.

Lemma 7. *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. Then $\mathrm{bs}(f) \leq 6\,\widetilde{\deg}(f)^2$.*

Lemma 7 is an easy corollary of the following result.

Lemma 8. *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function such that $f(0^n) = 0$ and $f(x) = 1$ for all x such that $|x| = 1$. Then $\widetilde{\deg}(f) \geq \sqrt{n/6}$.*

The proof of Lemma 8 is based on the notion of *symmetrisation*, which is a way of obtaining a univariate polynomial from a multivariate polynomial. For arbitrary $f : \{0,1\}^n \to \mathbb{R}$, the symmetrisation of f is the function f^{sym} defined by

$$f^{sym}(x) = \frac{1}{n!} \sum_{\sigma \in S_n} f(\sigma(x)),$$

where S_n is the symmetric group on n elements and $\sigma(x)$ is the bit-string obtained by permuting the bits of x according to the permutation σ.

Lemma 9. *For any multilinear polynomial $f : \mathbb{R}^n \to \mathbb{R}$, there exists a univariate polynomial $p : \mathbb{R} \to \mathbb{R}$ such that $f^{sym}(x) = p(|x|)$. Further, $\deg(p) = \deg(f^{sym}) \leq \deg(f)$.*

Proof. Set $d = \deg(f)$. As f is a multilinear polynomial, it can be written as

$$f(x) = \sum_{S \subseteq \{1,\ldots,n\}, |S| \leq d} \alpha_S \prod_{i \in S} x_i$$

for some real coefficients $\{\alpha_S\}$. Thus

$$f^{sym}(x) = \frac{1}{n!} \sum_{S \subseteq \{1,\ldots,n\}, |S| \leq d} \alpha_S \sum_{\sigma \in S_n} \prod_{i \in S} \sigma(x)_i = \sum_{S \subseteq \{1,\ldots,n\}, |S| \leq d} \alpha_S \frac{\binom{|x|}{|S|}}{\binom{n}{|S|}}.$$

The claim follows from observing that

$$\binom{|x|}{|S|} = \frac{|x|(|x|-1)\ldots(|x|-|S|+1)}{|S|!}$$

is a degree $|S|$ polynomial in $|x|$, and that $|S| \leq \deg(f)$. □

We will also need a result from approximation theory, which we state without proof (for details, see [16] or [5]).

Lemma 10. *Fix an integer n and real numbers b_1, b_2, c, and let $p : \mathbb{R} \to \mathbb{R}$ be a univariate polynomial which satisfies these two properties: (a) for every integer $0 \leq i \leq n$, $b_1 \leq p(i) \leq b_2$ for some constants b_1, b_2; and (b) for some real $0 \leq x \leq n$, the derivative of p satisfies $|p'(x)| \geq c$. Then $\deg(p) \geq \sqrt{cn/(c + b_1 - b_2)}$.*

We are now ready to prove Lemma 8.

Proof of Lemma 8. Let $\tilde{f}$ be a function approximating f to within an additive error of $1/3$ on every input, and let p be the univariate polynomial obtained by symmetrising $\tilde{f}$. Then:

1. $\deg(p) \leq \deg(\tilde{f})$ (by Lemma 9);
2. For every integer $0 \leq i \leq n$, $-1/3 \leq p(i) \leq 4/3$ (as f is a boolean function);
3. $p(0) \leq 1/3$ (as $f(0^n) = 0$);
4. $p(1) \geq 2/3$ (as $f(x) = 1$ for all x such that $|x| = 1$).

These last two properties imply that, for some real $0 \leq z \leq 1$, $|p'(z)| \geq 1/3$. So, by Lemma 10, $\deg(p) \geq \sqrt{n/6}$ and hence $\deg(\tilde{f}) \geq \sqrt{n/6}$. □

2.3 Certificate complexity and block sensitivity

The other complexity measure which we will need is *certificate complexity*. Informally, this measures the number of bits of x we need in order to know the value of $f(x)$.

Definition 3. Let S be a subset of $\{1, \ldots, n\}$, and let $C : S \to \{0, 1\}$ be an assignment of values to the variables in S. The size of C is $|S|$. We say that x is consistent with C if $x_i = C(i)$ for all $i \in S$. A b-certificate for f is an assignment C such that $f(x) = b$ whenever x is consistent with C.

The certificate complexity of f on x, $C_x(f)$, is the size of a smallest $f(x)$-certificate which is consistent with x. The certificate complexity of f is $C(f) = \max_x C_x(f)$.

For example, the assignment $x_1 = 1$ is a 1-certificate for the OR function on n bits. However, there is no 0-certificate for this function with size strictly lower than n. Certificate complexity and block sensitivity are related as follows.

Lemma 11. *For any function $f : \{0, 1\}^n \to \{0, 1\}$, $C(f) \leq \mathrm{bs}(f)^2$.*

Proof. For an arbitrary input $x \in \{0, 1\}^n$, let $B_1, \ldots, B_b$ be minimal disjoint sets of variables achieving $b = \mathrm{bs}_x(f)$. We will show that the certificate $C : \bigcup_i B_i \to \{0, 1\}$ which sets each of the variables in $\bigcup_i B_i$ according to x is a small enough $f(x)$-certificate.

Towards a contradiction, assume that C is not a certificate. Then there is an input y such that y is consistent with C, but $f(y) \neq f(x)$. Define B_{b+1} as the set of bits i such that $x_i \neq y_i$. Then B_{b+1} is disjoint from $B_1, \ldots, B_b$, and yet flipping any of the bits of x in B_{b+1} flips $f(x)$, contradicting $b = \mathrm{bs}_x(f)$. Thus C is indeed a $f(x)$-certificate.

We finally show that $|C| \leq \mathrm{bs}(f)^2$, which will follow from showing that $|B_i| \leq \mathrm{bs}(f)$ for all i. Consider the input z_i which is equal to x with the bits in B_i flipped. Now flipping any of z_i's bits in the set B_i must flip the value of f from $f(z_i)$ to $f(x)$, or B_i would not be minimal, so $\mathrm{bs}(f) \geq \mathrm{bs}_{z_i}(f) \geq |B_i|$. □

The final ingredient we will need to prove Theorem 3 is a classical upper bound in terms of certificate complexity and block sensitivity.

Lemma 12. *For any function $f : \{0, 1\}^n \to \{0, 1\}$, there exists a classical algorithm which computes $f(x)$, for any x, using at most $C(f)\,\mathrm{bs}(f)$ queries.*

Proof. We give an explicit algorithm achieving the required number of queries. For a certificate C (or input y), we say that C (or y) is consistent at a particular point of the algorithm's execution if it agrees with the values of all variables queried by the algorithm up to that point. Then the algorithm is as follows.

1. Repeat at most $\mathrm{bs}(f)$ times:
   (a) Attempt to pick a consistent 1-certificate C.
   (b) If there is no such C, stop and return 0. Otherwise, query the variables in C which are unknown.
   (c) If the queried values agree with C then stop and return 1.
2. Pick a consistent input y and return $f(y)$.

It is clear that the algorithm uses at most $C(f)\,\mathrm{bs}(f)$ queries. We now show that it is always correct. If it returns an answer early (in step 1), it is either because there are no consistent 1-certificates, so it returns 0 as required, or if has found a consistent 1-certificate, so it returns 1 as required.

We will show that, if it does not return an answer in step 1, f must be constant on all remaining consistent inputs y. Assume towards a contradiction that $f(y) = 0$, $f(z) = 1$ for some consistent y, z. Let the $\mathrm{bs}(f)$ 1-certificates the algorithm has queried be $C_1, \ldots, C_{\mathrm{bs}(f)}$. As $f(z) = 1$, z contains a consistent 1-certificate $C_{\mathrm{bs}(f)+1}$. For each i, define B_i as the set of variables on which y and C_i disagree. B_i is non-empty as otherwise the algorithm would have returned 1 in step 1.

Now it holds that $f(y^{B_i}) \neq f(y)$, because y^{B_i} agrees with C_i and hence $f(y^{B_i}) = 1$. For any i such that $1 \leq i \leq \mathrm{bs}(f)$, if variable k occurs in B_i, then $x_k = y_k \neq C_i(k)$. If $j > i$, then it cannot hold that $x_k = y_k \neq C_j(k)$, because C_j was consistent with all variables previously queried, including x_k. Thus $k \notin B_j$, implying that the sets B_i are disjoint. This implies that f is sensitive to $\mathrm{bs}(f) + 1$ disjoint sets of variables on y, which is a contradiction and implies that f must be constant on all consistent y in step 2, so the algorithm returns the right answer. □

We are finally ready to combine all these ingredients to prove Theorem 3, which we restate for clarity.

Theorem. *Let $f : \{0,1\}^n \to \{0,1\}$ be a total boolean function computed with bounded error by a quantum algorithm using T queries to the input. Then there is a deterministic classical algorithm which computes f with certainty using $O(T^6)$ queries to the input.*

Proof of Theorem 3. By Lemma 12, for any f there is a classical algorithm which computes f with certainty using at most

$$C(f)\,\mathrm{bs}(f) \leq \mathrm{bs}(f)^3 \leq \widetilde{\deg}(f)^6$$

queries to the input, where the first inequality is Lemma 11 and the second is Lemma 7. By Corollary 6, if there is a bounded-error quantum algorithm for f making T queries, $\widetilde{\deg}(f) = \Omega(T)$. This completes the proof of the theorem. □

The alert reader may have noticed that Theorem 3 can actually be proven without the use of the polynomial method, as Theorem 1 can be used to show that any quantum algorithm that computes a boolean function f with bounded error must make $\Omega(\sqrt{\mathrm{bs}(f)})$ queries to the input. However, there are other problems for which the polynomial method gives better lower bounds than any other known technique.

2.4 Perspective on lower bounds on quantum algorithms

The study of lower bounds can be somewhat dispiriting, as fundamentally it is about not what we can do, but what we cannot. However, it is of importance for the following reasons. First,

lower bounds give us a guide to what we can hope to achieve with quantum computers. Finding new quantum algorithms is a daunting task, and any hints as to what we can hope to achieve are helpful. Second, and perhaps more importantly, as we believe the model of quantum computation encapsulates all computation that can be performed physically, lower bounds are nothing less than fundamental statements about our physical universe.

2.5 Further reading and open problems

There is an excellent survey on complexity measures of boolean functions, including the bounds on quantum query complexity discussed above, by Buhrman and de Wolf [5]. The polynomial method was introduced by Beals et al [2], following prior work by Nisan and Szegedy [16] in the classical literature. Høyer and Špalek have since written a good general survey on quantum query complexity, including the far-reaching generalisation of the lower bound on unstructured search known as the adversary method [10].

There are still many open and accessible problems in the study of quantum lower bounds. In what follows, let $D(f)$ denote the minimum number of queries to the input required for any classical algorithm that computes f exactly. Also let $Q_E(f)$ and $Q_2(f)$ denote the minimum number of queries required for any quantum algorithm that computes f exactly and with bounded error, respectively.

- Theorem 3 states that $D(f) = O(Q_2(f)^6)$. It is conjectured that this bound can be improved to $D(f) = O(Q_2(f)^2)$, which would be tight by Grover's algorithm, but any improvement would be interesting.
- No function f is known such that $D(f) > 2Q_E(f)$. On the other hand, the tightest known relationship between these two complexity measures is cubic: $D(f) = O(Q_E(f)^3)$, by a result of Midrijānis [14].
- It is conjectured that the general lower bound $Q_2(f) = \Omega(\log n)$ holds for any boolean function f which depends on n input variables, for some universal constant C. A lower bound $Q_2(f) = \Omega(\sqrt{\log n})$ is already known, which is based on block sensitivity arguments.

3 Phase estimation

We now discuss an important primitive used in quantum algorithms called *phase estimation*, which provides a different and unifying perspective on some of the quantum algorithms (factoring, search) which you have already seen. Phase estimation is once again based on the quantum Fourier transform (QFT) over $\mathbb{Z}_{2^n}$. Recall that this is the unitary operator "QFT" that satisfies

$$\mathrm{QFT}|x\rangle = \sum_{y=0}^{2^n-1} e^{i\pi xy/2^{n-1}}|y\rangle.$$

Imagine we are given a unitary operator U. U may either be written down as a quantum circuit, or we may be given access to a black box which allows us to apply a controlled-U^j operation for integer values of j. We are also given a state $|\psi\rangle$ which is an eigenvector of U: $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$ for some real ϕ such that $0 \leq \phi < 1$. We would like to determine ϕ 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

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|\psi\rangle$$

by applying Hadamards to each qubit in the first register. We then apply the unitary operator

$$U' = \sum_{x=0}^{2^n-1} |x\rangle\langle x| \otimes U^x.$$

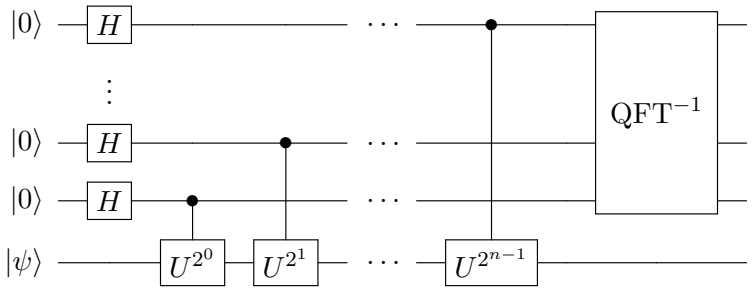
This operator can be described as: if the first register contains x, apply U x times to the second register. We are left with the state

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} e^{2\pi i\phi x}|x\rangle|\psi\rangle;$$

note that the second register is left unchanged by this operation. We now apply the operator QFT^{-1} to the first register and then measure it, receiving outcome x (say). We output the binary fraction

$$0.x_1x_2\ldots x_n = \frac{x_1}{2} + \frac{x_2}{4} + \cdots + \frac{x_n}{2^n}$$

as our guess for ϕ. The following is an explicit circuit for the above algorithm, up to the measurement of the first register.



Why does this algorithm work? When we perform the final measurement, the probability of getting the outcome x is

$$\frac{1}{2^n}\left|\sum_{y=0}^{2^n-1} e^{2\pi i\phi y - i\pi xy/2^{n-1}}\right|^2 = \frac{1}{2^n}\left|\sum_{y=0}^{2^n-1} e^{2\pi iy(\phi - x/2^n)}\right|^2.$$

First imagine that the binary expansion of ϕ is at most n bits long, or in other words $\phi = z/2^n$ for some $0 \le z \le 2^n - 1$. In this case we have

$$\frac{1}{2^n}\left|\sum_{y=0}^{2^n-1} e^{2\pi iy(\phi - x/2^n)}\right|^2 = \frac{1}{2^n}\left|\sum_{y=0}^{2^n-1} e^{\pi iy(z-x)/2^{n-1}}\right|^2 = \delta_{xz}$$

by the unitarity of the QFT, so the measurement outcome is guaranteed to be z, implying that the algorithm outputs ϕ with certainty. If the binary expansion of ϕ is longer than n bits, we now show that we still get the best possible answer with constant probability, and indeed are very likely to get an answer close to ϕ.

Theorem 13. *The probability that the above algorithm outputs the real number with n binary digits which is closest to ϕ is at least $4/\pi^2$. Further, the probability that the algorithm outputs θ such that $|\theta - \phi| \geq \epsilon$ is at most $O(1/(2^n\epsilon))$.*

Proof. If the binary expansion of ϕ has n binary digits or fewer, we are done by the argument above. So, assuming it does not, let $\widetilde{\phi}$ be the closest approximation to ϕ that has n binary digits, and write $\widetilde{\phi} = a/2^n$ for some integer $0 \leq a \leq 2^n - 1$. For any z, define $\delta(z) := \phi - z/2^n$ and note that $0 < |\delta(z)| \leq 1/2^{n+1}$ for all z. For any ϕ, the probability of getting outcome z from the final measurement is

$$\frac{1}{2^{2n}} \left| \sum_{y=0}^{2^n-1} e^{2\pi i y(\phi - z/2^n)} \right|^2 = \frac{1}{2^{2n}} \left| \sum_{y=0}^{2^n-1} e^{2\pi i y \delta(z)} \right|^2 = \frac{1}{2^{2n}} \left| \frac{1 - e^{2^{n+1}\pi i \delta(z)}}{1 - e^{2\pi i \delta(z)}} \right|^2, \tag{1}$$

where we evaluate the sum using the formula for a geometric series. This quantity should be familiar from the proof of correctness of the periodicity determination algorithm, and indeed the first part of this proof is the same as the proof of Theorem 4 in Section 6.2; nevertheless, we repeat it for convenience.

We first lower bound this expression for $z = a$ to prove the first part of the lemma; for conciseness, write $\delta := \delta(a) = \phi - \widetilde{\phi}$. As $|\delta| \leq 1/2^{n+1}$, we have $2^{n+1}\pi\delta \leq \pi$. We now claim that (a) $|1 - e^{2^{n+1}\pi i\delta}| \geq 2^{n+2}\delta$; (b) $|1 - e^{2\pi i\delta}| \leq 2\pi\delta$. Together, these claims imply that

$$\frac{1}{2^{2n}} \left| \frac{1 - e^{2^{n+1}\pi i\delta}}{1 - e^{2\pi i\delta}} \right|^2 \geq \frac{1}{2^{2n}} \left(\frac{2^{n+2}\delta}{2\pi\delta} \right)^2 \geq \frac{4}{\pi^2}.$$

Claims (a) and (b) are proven by geometric arguments. For part (a), set $\alpha = 2^{n+1}\pi\delta$ and observe that $|1 - e^{i\alpha}|$ is the length of a chord between 1 and $e^{i\alpha}$ in the complex plane. As $\alpha \leq \pi$, $|1 - e^{i\alpha}| = 2\sin(\alpha/2) \geq (2/\pi)\alpha$. For part (b), simply observe that the length of the chord between two points is upper bounded by the length of the minor arc between them.

In order to prove the second part of the theorem, we now find an *upper* bound on expression (1). First, it is clear that $|1 - e^{2^{n+1}\pi i\delta(z)}| \leq 2$ always. For the denominator, by the same argument to part (a) above we have $|1 - e^{2\pi i\delta(z)}| \geq 4\delta(z)$ and hence, for all z,

$$\Pr[\text{get outcome } z] \leq \frac{1}{2^{2n}} \left(\frac{2}{4\delta(z)} \right)^2 = \frac{1}{2^{2n+2}\delta(z)^2}.$$

We now sum this expression over all z such that $|\delta(z)| \geq \epsilon$. The sum is symmetric about $\delta(z) = 0$, and as z is an integer, the terms in this sum corresponding to $\delta(z) > 0$ are $\delta_0, \delta_0 + 1/2^n, \ldots$, for some $\delta_0 \geq \epsilon$. The sum will be maximised when $\delta_0 = \epsilon$, when we obtain

$$\begin{aligned} \Pr[\text{get outcome } z \text{ with } |\delta(z)| \geq \epsilon] &\leq \frac{1}{2^{2n+2}} \sum_{k=0}^{\infty} \frac{1}{(\epsilon + k/2^n)^2} \leq \frac{1}{4} \int_0^{\infty} \frac{1}{(2^n\epsilon + k)^2} dk \\ &= \frac{1}{4} \int_{2^n\epsilon}^{\infty} \frac{1}{k^2} dk = O\left(\frac{1}{2^n\epsilon} \right). \end{aligned}$$

□

We observe the following points regarding the behaviour of this algorithm.

- What happens if we do not know an eigenvector of U? If we input an arbitrary state $|\varphi\rangle$ to the phase estimation algorithm, we can write it as a superposition $|\varphi\rangle = \sum_j \alpha_j |\psi_j\rangle$ over eigenvectors $\{|\psi_j\rangle\}$. Therefore, the algorithm will output an estimate of each corresponding eigenvalue ϕ_j with probability $|\alpha_j|^2$. This may or may not allow us to infer anything useful, depending on what we know about U in advance.

- In order to approximate ϕ to n bits of precision, we needed to apply the operator U^{2^m}, for all $0 \leq m \leq n-1$. If we are given U as a black box, this may be prohibitively expensive as we need to use the black box exponentially many times in n. However, if we have an explicit circuit for U, we may be able to find a more efficient way of computing U^{2^m}.

3.1 Application to quantum counting

An elegant application of phase estimation is to a generalisation of the unstructured (Grover) search problem. Imagine we have an oracle $f : \{0,1\}^n \to \{0,1\}$ which takes the value 1 on k inputs, for some unknown k, and again set $N = 2^n$. We would like to estimate k by querying f.

Classically, a natural way to do this is by sampling. Imagine that we query f on q random inputs and get that f is 1 on ℓ of those inputs. Then as our estimate of k we output $\widetilde{k} = \ell N/q$. One can show using properties of the binomial distribution that this achieves

$$|\widetilde{k} - k| = O\left(\sqrt{\frac{k(N-k)}{q}}\right)$$

with high probability. We can achieve improved accuracy by using the phase estimation algorithm. Consider the "Grover iteration" Q_G defined on p. 40 of the previous notes. As Q_G is a rotation through angle 2θ in a 2-dimensional plane, where θ satisfies $\sin\theta = \sqrt{k/N}$, its eigenvalues are $e^{2i\theta}$ and $e^{-2i\theta}$. In order to estimate k, we can apply the phase estimation algorithm to Q_G to estimate either one of these eigenvalues. As it does not matter which we estimate, we can input any state within this 2-dimensional plane to the phase estimation algorithm as a claimed eigenvector of Q_G. In particular, the state $\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle$ will work.

By Theorem 13, if we apply the phase estimation algorithm to Q_G, we can find the closest m-digit number to θ, for any m, with constant probability of success using $O(2^m)$ queries. For small θ, we have $\theta \approx \sqrt{k/N}$, so we learn $\sqrt{k/N}$ up to additive error $O(1/2^m)$ using $O(2^m)$ queries. Setting $2^m = \sqrt{N}/\delta$ for some real $\delta > 0$, we have learnt $\sqrt{k}$ up to additive error $O(\delta)$ using $O(\sqrt{N}/\delta)$ queries; or in other words have learnt k up to additive error $O(\delta\sqrt{k})$ using $O(\sqrt{N}/\delta)$ queries. In order to achieve a similar level of accuracy classically, we would need $\Omega(N/\delta^2)$ queries for small k.

Another application of phase estimation, to the order finding problem, is discussed in the Exercises.

3.2 Further reading

Phase estimation is a useful primitive with many other applications, including implementation of the QFT over $\mathbb{Z}_d$, where d is not a power of 2, and efficiently extracting information from systems of linear equations [9]. See the lecture notes by Dieter van Melkebeek at `http://pages.cs.wisc.edu/~dieter/Courses/2010f-CS880/` for a discussion of these applications.

4 Hamiltonian simulation

One of the earliest – and most important – applications of a quantum computer is likely to be the simulation of quantum mechanical systems. There are quantum systems for which no efficient classical simulation is known, but which we can simulate on a universal quantum computer. What does it mean to "simulate" a physical system? According to the OED, simulation is "the technique of imitating the behaviour of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus". What we will take simulation to mean here is approximating the *dynamics* of a physical system.

According to the laws of quantum mechanics, time evolution of the state $|\psi\rangle$ of a quantum system is governed by Schrödinger's equation:

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle,$$

where $H(t)$ is a Hermitian operator known as the *Hamiltonian* of the system (for convenience, we will henceforth absorb $\hbar$ into $H(t)$). An important special case on which we will focus is the *time-independent* setting where $H(t) = H$ is constant. In this case the solution of this equation is

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle.$$

Given a physical system specified by some Hamiltonian H, we would like to simulate the evolution of the system on an arbitrary initial state for a certain amount of time t. In other words, given H, we would like to implement a unitary operator which approximates

$$U(t) = e^{-iHt}.$$

What does it mean to approximate a unitary? The "gold standard" of approximation is approximation in the operator norm (aka spectral norm)

$$\|A\| := \max_{|\psi\rangle\neq 0} \frac{\|A|\psi\rangle\|}{\||\psi\rangle\|},$$

where $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$ is the usual Euclidean norm of $|\psi\rangle$. Note that this is indeed a norm, and in particular satisfies the triangle inequality $\|A + B\| \leq \|A\| + \|B\|$. We say that $\widetilde{U}$ approximates U to within ϵ if

$$\|\widetilde{U} - U\| \leq \epsilon.$$

This is a natural definition of approximation because it implies that, for any state $|\psi\rangle$, $\widetilde{U}|\psi\rangle$ and $U|\psi\rangle$ are only distance at most ϵ apart.

4.1 Simulation of k-local Hamiltonians

For simplicity, assume that H is a Hamiltonian of n two-level systems (qubits). In order for our quantum simulation of H to be efficient, we need $U = e^{-iHt}$ to be approximable by a quantum circuit containing poly(n) gates. A fairly straightforward counting argument shows that not all Hamiltonians H can be simulated efficiently. However, it turns out that several important physically motivated classes can indeed be simulated. The first of these is *k-local* Hamiltonians.

A Hamiltonian H of n qubits is said to be k-local if it can be written as a sum

$$H = \sum_{j=1}^{m} H_j$$

for some m, where each H_j is a Hermitian matrix which acts non-trivially on at most k qubits. That is, H_j is the tensor product of a matrix H'_j on k qubits, and the identity matrix on the remaining $n-k$ qubits. For example, the operator on 3 qubits

$$H = X \otimes I \otimes I - 2I \otimes Z \otimes Y$$

is 2-local. Many interesting physical systems are k-local for small k (say $k \leq 3$), some of which you may have heard of. Simple examples include the two-dimensional Ising model on a $n \times n$ square lattice,

$$H = J \sum_{i,j=1}^{n} Z^{(i,j)} Z^{(i,j+1)} + Z^{(i,j)} Z^{(i+1,j)}$$

and the Heisenberg model on a line,

$$H = \sum_{i=1}^{n} J_x X^{(i)} X^{(i+1)} + J_y Y^{(i)} Y^{(i+1)} + J_z Z^{(i)} Z^{(i+1)},$$

both of which are used in the study of magnetism (in the above, $M^{(j)}$ denotes a single qubit operator acting on the j'th qubit, and J, J_x, J_y, J_z are constants).

Note that, if H is k-local, we can assume that $m \leq \binom{n}{k} = O(n^k)$. We usually assume that k is constant, in which case m is polynomial in n. We first show that each of the individual H_j operators can be simulated efficiently, which will be immediate from the following theorem.

Theorem 14 (Solovay-Kitaev theorem)**.** *Let U be a unitary operator which acts non-trivially on $k = O(1)$ qubits, and let S be an arbitrary universal set of quantum gates. Then U can be approximated in the operator norm to within ϵ using $O(\log^c(1/\epsilon))$ gates from S, for some $c < 4$.*

Proof. Sadly beyond the scope of this course. For a readable explanation, see Andrew Childs' lecture notes (`http://www.math.uwaterloo.ca/~amchilds/`). □

As each $e^{-iH_j t}$ acts non-trivially on only at most k qubits, it follows from the Solovay-Kitaev theorem that we can approximate each of these operators individually to within ϵ in time $O(\text{polylog}(1/\epsilon))$. In the special case where all of the H_j operators commute, we have

$$e^{-iHt} = e^{-i(\sum_{j=1}^{m} H_j)t} = \prod_{j=1}^{m} e^{-iH_j t}.$$

Thus a natural way to find a unitary operator approximating e^{-iHt} is to take the product of our approximations of $e^{-iH_1 t}, \ldots, e^{-iH_m t}$. Although each of these approximates $e^{-iH_j t}$ to within ϵ, this does not imply that their product approximates e^{-iHt} to within ϵ. However, we now show that the approximation error only scales linearly (similarly to the situation with "deviation" vectors in the quantum search lower bound).

Lemma 15. *Let (U_i), (V_i) be sequences of m unitary operators satisfying $\|U_i - V_i\| \leq \epsilon$ for all $1 \leq i \leq m$. Then $\|U_m \ldots U_1 - V_m \ldots V_1\| \leq m\epsilon$.*

Proof. The proof is by induction on m. The claim trivially holds for $m = 1$. Assuming that it holds for a given m, we have

$$\begin{aligned}
&\|U_{m+1}U_m \dots U_1 - V_{m+1}V_m \dots V_1\| \\
&= \|U_{m+1}U_m \dots U_1 - U_{m+1}V_m \dots V_1 + U_{m+1}V_m \dots V_1 - V_{m+1}V_m \dots V_1\| \\
&\leq \|U_{m+1}U_m \dots U_1 - U_{m+1}V_m \dots V_1\| + \|U_{m+1}V_m \dots V_1 - V_{m+1}V_m \dots V_1\| \\
&= \|U_{m+1}(U_m \dots U_1 - V_m \dots V_1)\| + \|(U_{m+1} - V_{m+1})V_m \dots V_1\| \\
&= \|U_m \dots U_1 - V_m \dots V_1\| + \|U_{m+1} - V_{m+1}\| \\
&\leq (m+1)\epsilon.
\end{aligned}$$

□

Thus, in order to approximate $\prod_{j=1}^{m} e^{-iH_j t}$ to within ϵ, it suffices to approximate each of the H_j to within ϵ/m. We formalise this as the following proposition.

Proposition 16. *Let H be a Hamiltonian which can be written as the sum of m commuting terms H_j, each acting non-trivially on $k = O(1)$ qubits. Then, for any t, there exists a quantum circuit which approximates the operator e^{-iHt} to within ϵ in time $O(m \operatorname{polylog}(m/\epsilon))$.*

4.2 The non-commuting case

Unfortunately, this simulation technique does *not* necessarily work for non-commuting H_j. The reason is that if A and B are non-commuting operators, it need not hold that $e^{-i(A+B)t} = e^{-iAt}e^{-iBt}$. However, we can simulate non-commuting Hamiltonians via an observation known as the Lie-Trotter product formula.

In what follows, the notation $X + O(\epsilon)$, for a matrix X, is used as shorthand for $X + E$, where E is a matrix satisfying $\|E\| \leq C\epsilon$, for some universal constant C (not depending on X or ϵ).

Lemma 17 (Lie-Trotter product formula)**.** *Let A and B be Hermitian matrices such that $\|A\| \leq K$ and $\|B\| \leq K$, for some real $K \leq 1$. Then*

$$e^{-iA}e^{-iB} = e^{-i(A+B)} + O(K^2).$$

Proof. From the Taylor series for e^x, for any matrix A such that $\|A\| = K \leq 1$, we have

$$e^{-iA} = I - iA + \sum_{k=2}^{\infty} \frac{(-iA)^k}{k!} = I - iA + (-iA)^2 \sum_{k=0}^{\infty} \frac{(-iA)^k}{(k+2)!} = I - iA + O(K^2).$$

Hence

$$e^{-iA}e^{-iB} = \left(I - iA + O(K^2)\right)\left(I - iB + O(K^2)\right) = I - iA - iB + O(K^2) = e^{-i(A+B)} + O(K^2).$$

□

Applying this formula multiple times, for any Hermitian matrices $H_1, \dots, H_m$ satisfying $\|H_j\| \leq K \leq 1$ for all j,

$$\begin{aligned}
e^{-iH_1}e^{-iH_2} \dots e^{-iH_m} &= \left(e^{-i(H_1+H_2)} + O(K^2)\right) e^{-iH_3} \dots e^{-iH_m} \\
&= \left(e^{-i(H_1+H_2+H_3)} + O((2K)^2)\right) e^{-iH_4} \dots e^{-iH_m} + O(K^2) \\
&= e^{-i(H_1+\dots+H_m)} + O(K^2) + O((2K)^2) + \dots + O(((m-1)K)^2) \\
&= e^{-i(H_1+\dots+H_m)} + O(m^3K^2).
\end{aligned}$$

Therefore, there is a universal constant C such that if $n \geq Cm^3(Kt)^2/\epsilon$,

$$\left\| e^{-iH_1 t/n} e^{-iH_2 t/n} \dots e^{-iH_m t/n} - e^{-i(H_1+\dots+H_m)t/n} \right\| \leq \epsilon/n.$$

By Lemma 15, for any such n

$$\left\| \left(e^{-iH_1 t/n} e^{-iH_2 t/n} \dots e^{-iH_m t/n} \right)^n - e^{-i(H_1+\dots+H_m)t} \right\| \leq \epsilon.$$

Given this result, we can simulate a k-local Hamiltonian simply by simulating the evolution of each term for time t/n to high enough accuracy and concatenating the individual simulations. We formalise this as the following theorem.

Theorem 18. *Let H be a Hamiltonian which can be written as the sum of m terms H_j, each acting non-trivially on $k = O(1)$ qubits and satisfying $\|H_j\| \leq K$ for some K. Then, for any t, there exists a quantum circuit which approximates the operator e^{-iHt} to within ϵ in time $O(m^3(Kt)^2/\epsilon)$, up to polylogarithmic factors.*

It seems somewhat undesirable that, in order to simulate a Hamiltonian for time t, this algorithm has dependence on t which is $O(t^2)$. In fact, using more complicated simulation techniques, this can be improved to time $O(t^{1+\delta})$ for an arbitrarily small constant $\delta > 0$ [4].

4.3 Further reading

The quantum simulation algorithm discussed here is due to Lloyd [13]; there is an interesting previous discussion by Feynman [7] about why efficient *classical* simulation of quantum systems should not be possible. Can we do better than this quantum simulation algorithm? It has been shown using quantum lower bound techniques that, in an oracular setting, simulating e^{iHt} in time faster than $O(t)$ is not possible (a "no-fast-forwarding theorem" [4]).

5 Quantum error-correction

Modern computer hardware is extremely reliable. Indeed, it can usually be assumed to be error-free for all intents and purposes[1]. However, early quantum computing hardware is likely to be far from reliable. Even worse, efficient quantum algorithms rely on delicate quantum effects (superposition and entanglement) which must be preserved in the presence of errors. Luckily, it turns out that errors can be fought using the notion of quantum error-correcting codes. To understand these codes, it is helpful to first consider a basic classical notion of error correction.

Imagine we have a single bit x which we would like to store in a noisy storage device. A natural model for this noise is that each bit stored in the device gets flipped with probability p, for some $0 \leq p \leq 1$, and is left the same with probability $1-p$. So if we store x in the device and then read it back later, we get the wrong answer with probability p. One way to improve this works as follows. Instead of just storing x, store the string xxx, i.e. repeat x three times. Then read out each of the bits of the (potentially corrupted) string to get $y := y_1 y_2 y_3$, and output 0 if the majority of the bits of y are 0, and 1 otherwise.

What is the probability of failing to output x if this strategy is followed? The wrong answer will be returned if two or more of the bits of y are flipped by noise, which will occur with probability

[1] Software, of course, is another matter.

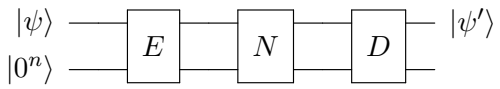
$3p^2(1-p)+p^3 = p^2(3-2p) = O(p^2)$. Thus, if p is small, this strategy has improved our resistance to noise. Indeed, for any p such that $0 < p < 1/2$, we have

$$p^2(3-2p) < p,$$

so the probability of error has been reduced. Another way of looking at this situation is that we have stored a bit in such a way that it is impervious to an error affecting a single bit in the storage device. The map $x \mapsto xxx$ is a very simple example of an *error correcting code* known as the binary repetition code of length 3.

5.1 Quantum errors and error-correction

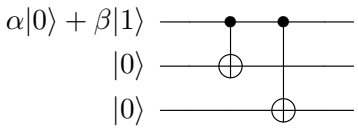
We would like to find a quantum analogue of this notion of error correction. Rather than preserving classical bits x, our quantum error correcting code should preserve a qubit $|\psi\rangle$ under some notion of error. We say here that an error affecting one or more qubits is simply an arbitrary and unknown unitary operator N applied to those qubits[2]. The classical bit-flip error discussed above is an example of this, as it can be seen as simply applying the operator X to a qubit in a computational basis state (recall that $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$). The process of correcting errors in a qubit state $|\psi\rangle$ can be written diagrammatically as



for some unitary encoding operation E, noise operation N, and decoding operation D. In other words, we encode some qubit state $|\psi\rangle$ as a larger state $|E(\psi)\rangle$ using some ancilla qubits (initially in the state $|0^n\rangle$), some noise is applied, and later we decode the noisy encoded state to produce a state $|\psi'\rangle$. The goal is that after this process $|\psi'\rangle \approx |\psi\rangle$ for some set of correctable noise operations N.

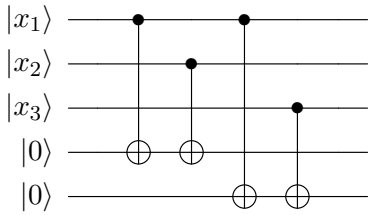
There are two obvious ways in which the classical repetition code could be translated to the quantum regime, both of which unfortunately do not work. First, we could measure $|\psi\rangle$ in the computational basis to obtain a bit 0 or 1, then just encode this with the classical repetition code. This is not suitable for quantum error correction because it does not preserve quantum coherence: if $|\psi\rangle$ is in a superposition of 0 and 1 and will be used as input to a subsequent quantum algorithm, it is necessary to preserve this superposition to see any interesting quantum effects. A second idea is that we could map $|\psi\rangle \mapsto |\psi\rangle|\psi\rangle|\psi\rangle$, by analogy with the classical code. However, this is impossible (for general $|\psi\rangle$) by the no-cloning theorem.

We therefore have to take a different approach, which will be split into two steps. In the first step, we try encoding $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ as $|E(\psi)\rangle = \alpha|000\rangle + \beta|111\rangle$. Note that this is *not* the same as the "cloning" map discussed previously. Indeed, the map $\alpha|0\rangle + \beta|1\rangle \mapsto \alpha|000\rangle + \beta|111\rangle$ can be implemented via the following simple quantum circuit.



[2]This is in fact a simplification of the real situation; in order to describe the full set of possible errors, one needs to introduce the framework of mixed quantum states and completely positive maps. This will be discussed in the Quantum Information Theory course next term.

Our decoding algorithm for this code will be based on the following quantum circuit.



Call the first three qubits the input qubits and the last two the output qubits. Following this circuit, for any basis state input $|x_1x_2x_3\rangle$, the first of the two output qubits contains $x_1 \oplus x_2$, and the second contains $x_1 \oplus x_3$. Each of these quantities is invariant under the operation of flipping all the bits of x. Thus, for any input superposition of the form $\alpha|x_1x_2x_3\rangle + \beta|x_1x_2x_3 \oplus 111\rangle$, the circuit performs the map

$$(\alpha|x_1x_2x_3\rangle + \beta|x_1x_2x_3 \oplus 111\rangle)|0\rangle|0\rangle \mapsto (\alpha|x_1x_2x_3\rangle + \beta|x_1x_2x_3 \oplus 111\rangle)|x_1 \oplus x_2\rangle|x_1 \oplus x_3\rangle.$$

This implies that, if we measure the two output qubits, we learn both $x_1 \oplus x_2$ and $x_1 \oplus x_3$ without disturbing the input quantum state. Now observe that the encoded state of $|\psi\rangle$ is always of this form, even after arbitrary bit-flip errors are applied to $|E(\psi)\rangle$:

$$\begin{aligned} |E(\psi)\rangle &= \alpha|000\rangle + \beta|111\rangle, \\ (X \otimes I \otimes I)|E(\psi)\rangle &= \alpha|100\rangle + \beta|011\rangle, \\ (X \otimes X \otimes X)|E(\psi)\rangle &= \alpha|111\rangle + \beta|000\rangle, \text{ etc.} \end{aligned}$$

The result of measuring the output qubits is known as the *syndrome*. We now consider the different syndromes we get when different noise operators N are applied to $|E(\psi)\rangle$. First, if $N = I$ (so there has been no error applied to $|E(\psi)\rangle$), we always measure 00. On the other hand, if $N = X \otimes I \otimes I$ (i.e. a bit-flip error on the first qubit) we obtain 11 with certainty. We can write all the possible outcomes in a table as follows.

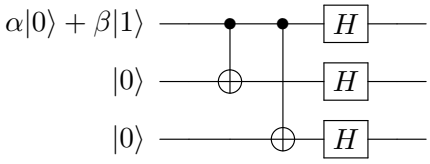
N	Syndrome
$I \otimes I \otimes I$	00
$I \otimes I \otimes X$	01
$I \otimes X \otimes I$	10
$I \otimes X \otimes X$	11
$X \otimes I \otimes I$	11
$X \otimes I \otimes X$	10
$X \otimes X \otimes I$	01
$X \otimes X \otimes X$	00

Observe that the syndromes corresponding to no error, and to bit flips on single qubits (i.e. $I \otimes I \otimes I$, $I \otimes I \otimes X$, $I \otimes X \otimes I$ and $X \otimes I \otimes I$) are all distinct. This means that, if one of these four errors occurs, we can detect it. After we detect a bit-flip error on a given qubit, we can simply apply the same bit-flip operation to that qubit to restore the original encoded state $\alpha|000\rangle + \beta|111\rangle$, which can easily then be mapped to $\alpha|0\rangle + \beta|1\rangle$ by reversing the original encoding circuit. On the other hand, if bit-flip errors occur on more than one qubit, we do not detect them (and indeed this "error correction" process can make matters worse!).

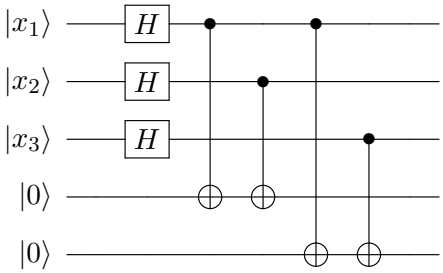
While this code is sufficient to protect against single bit-flip errors, there are other, less classical, errors acting on single qubits which it does not protect against. For example, consider the effect of a Z ("phase") error acting on the first qubit of the encoded state $|E(\psi)\rangle$, which maps it to

$\alpha|000\rangle - \beta|111\rangle$ (recall that $Z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$). It is easy to see that the syndrome measurement still returns 00, so the error correction operation does nothing and the Z error is not corrected.

However, these Z errors can be detected using a different code. Observe that $Z = HXH$, where H is the Hadamard gate. Thus Z acts in the same way as X, up to a change of basis. If we use the same code as before, but perform this change of basis for each qubit, we obtain a code which corrects against Z errors. In other words, we now encode $|\psi\rangle$ as $\alpha|+++\rangle + \beta|---\rangle$. Our new encoding circuit is simply



and our decoding circuit is



The analysis for the previous code goes through without change to show that this code protects against Z errors on an individual qubit. However, it is easy to see that the new code no longer protects against X errors! Can we protect against both errors simultaneously? The answer is yes, by *concatenating* these two codes. We first encode $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ using the code protecting against phase flips, and then encode each of the resulting qubits using the code that protects against bit flips. In other words, we perform the map

$$\begin{aligned}
\alpha|0\rangle + \beta|1\rangle &\mapsto \frac{1}{2\sqrt{2}}(\alpha(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) + \beta(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)(|0\rangle - |1\rangle)) \\
&\mapsto \frac{1}{2\sqrt{2}}(\alpha(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle) \\
&\qquad + \beta(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)).
\end{aligned}$$

The single qubit $|\psi\rangle$ is now encoded using 9 qubits. These qubits can naturally be split into three blocks, each of which encodes one qubit of the state $\alpha|+++\rangle + \beta|---\rangle$. To decode this encoded state, first the decoding circuit for the bit-flip code is applied to each block. Assuming at most one bit-flip error has occurred in each block, the result will be the state $\alpha|+++\rangle + \beta|---\rangle$, perhaps with a Z error applied to one of the qubits. This state can then be mapped back to $\alpha|0\rangle + \beta|1\rangle$ using the decoding algorithm for the phase-flip code.

Example. Imagine a ZX error occurs on the fourth qubit of the encoded state. The input to the decoding algorithm is thus the state

$$\frac{1}{2\sqrt{2}}(\alpha(|000\rangle + |111\rangle)(|100\rangle - |011\rangle)(|000\rangle + |111\rangle) + \beta(|000\rangle - |111\rangle)(|100\rangle + |011\rangle)(|000\rangle - |111\rangle)).$$

We apply the bit-flip decoding algorithm to each of the three blocks of three qubits, and get syndromes of 00, 11, 00 ("no error", "error on first qubit", "no error"). So we perform an X

operation on the fourth qubit to correct this, and then the map $|000\rangle \mapsto |0\rangle$, $|111\rangle \mapsto |1\rangle$ on each block of three qubits. The result is the state

$$\alpha|+-+\rangle + \beta|-+-\rangle.$$

Applying the phase-flip decoding algorithm to this state gives $\alpha|0\rangle + \beta|1\rangle$ as required.

We now have a code that can protect against X or Z errors acting on an arbitrary qubit. It may seem that this is only the beginning of the gargantuan task of protecting against every one of the infinitely many possible errors that can occur. In fact, it turns out that we have already done this! The key observation is that the matrices $\{I, X, Y, Z\}$, where $Y = ZX$, form a basis for the complex vector space of all 2×2 matrices, so an arbitrary error operation N acting on a single qubit can be written as a linear combination of these matrices. By the linearity of quantum mechanics, this implies that our code, which as discussed can correct single qubit X, Z and ZX errors, in fact can correct an arbitrary error N on an individual qubit (up to an unobservable global phase). A nice way of looking at this is that, if the state of the computer is in a superposition of each of these different errors having occurred, measuring the syndrome projects onto only one of them having occurred, which can then be corrected.

5.2 Further reading

The code discussed above is known as Shor's 9 qubit code and was one of the first quantum error-correcting codes discovered. The field of quantum error correction has since developed an extensive literature; a good recent review is [8].

6 Discrete-time quantum walk

We now discuss a quantum generalisation of the classical concept of random walk. Classically, random walks are an important algorithmic tool, and the same has proven to be true for quantum walks. We begin by introducing the most basic variant of quantum walk, walk on the line.

6.1 Quantum walk on the line

Classically, the simple random walk on the line is defined as follows. A particle ("walker") is placed on an infinite line at position 0. At each time step, the walker flips an unbiased coin. If the result is heads, it moves left by 1; otherwise, it moves right by 1. The probability of being found at position x after t steps is exactly

$$\frac{1}{2^t}\binom{t}{\frac{t+x}{2}},$$

where we define $\binom{t}{r} = 0$ for non-integer r. (This is easily derived as follows: there are 2^t different paths of n steps that could be taken; the paths which end up at x are exactly those with $(t+x)/2$ right-moving steps; and there are $\binom{t}{(t+x)/2}$ such paths.) We can see from this that the probability decays quickly for $|x|$ outside the range $O(\sqrt{t})$, and indeed one can calculate the variance of the position p as

$$\mathbb{E}[p^2] = \mathbb{E}\left[\left(\sum_{i=1}^{t} Z_i\right)^2\right] = \sum_{i,j=1}^{t} \mathbb{E}[Z_i Z_j] = \sum_{i=1}^{t} \mathbb{E}[Z_i^2] = t,$$

where the $Z_i \in \{\pm 1\}$ are random variables determining whether the i'th step is to the left or the right. We look for a quantum generalisation of this simple process.

One natural generalisation is as follows. Consider a quantum system with two registers $|x\rangle|c\rangle$, where the first holds an integer position[3] x and the second holds a coin state $c \in \{L, R\}$. As in the classical case, at each step our quantum walk will flip a coin and then decide in which direction to move. These two operations will be unitary: a coin operator C, and a shift operator S. The coin operator acts solely on the coin register, and consists of a Hadamard operation:

$$C|L\rangle = \frac{1}{\sqrt{2}}\left(|L\rangle + |R\rangle\right), \quad C|R\rangle = \frac{1}{\sqrt{2}}\left(|L\rangle - |R\rangle\right).$$

The shift operator acts on both registers, and simply moves the walker in the direction indicated by the coin state:

$$S|x\rangle|L\rangle = |x-1\rangle|L\rangle, \quad S|x\rangle|R\rangle = |x+1\rangle|R\rangle.$$

Then a quantum walk on the line for t steps consists of applying the unitary operator $(S(I \otimes C))^t$ to some initial state. Remarkably, these simple dynamics can lead to some fairly complicated results. Consider the first few steps of a quantum walk with initial state $|0\rangle|L\rangle$ (position 0, facing left). One can calculate that the state evolves as follows.

$$\begin{aligned}
|0\rangle|L\rangle &\mapsto \frac{1}{\sqrt{2}}\left(|-1\rangle|L\rangle + |1\rangle|R\rangle\right) \\
&\mapsto \frac{1}{2}\left(|-2\rangle|L\rangle + |0\rangle|R\rangle + |0\rangle|L\rangle - |2\rangle|R\rangle\right) \\
&\mapsto \frac{1}{2\sqrt{2}}\left(|-3\rangle|L\rangle + |-1\rangle|R\rangle + 2|-1\rangle|L\rangle - |1\rangle|L\rangle + |3\rangle|R\rangle\right) \\
&\mapsto \cdots
\end{aligned}$$

Consider the result of measuring the position register after the third step. Positions -3, 1 and 3 are obtained with probability 1/8 each, and position -1 is obtained with probability 5/8. In other words, the most likely position for the particle to be found in is -1. By contrast, the classical random walk is symmetric about 0 (and in fact is found in position -3 or 3 with probability 1/8 each, and -1 and 1 with probability 3/8 each). The bias of the quantum walk is an effect of interference. If the quantum walk is run for more steps before measuring the position, we obtain the pattern illustrated in Figure 2.

Note that, unlike the classical walk, the quantum walk is not symmetric about 0. Intuitively, this is caused by the Hadamard coin treating the L and R directions differently: only the $|R\rangle$ state gets mapped to a state containing a negative amplitude, leading to destructive interference in this direction. This asymmetry can be removed by changing the initial state of the coin register to $\frac{1}{\sqrt{2}}|0\rangle(|L\rangle + i|R\rangle)$, or alternatively by using a different coin operator. Figure 3 compares the distribution of probabilities obtained for classical and (symmetric) quantum walks.

The quantum walk seems to spread out from the origin faster than the classical walk. Indeed, it can be shown that the variance is $\Omega(t^2)$, noticeably faster than the classical $O(t)$. Unlike the straightforward analysis of the classical random walk, the effect of interference means that it is quite involved to prove this, so we will not do so here; for details, see [15]. This difference between quantum and classical walks will become more pronounced when we consider more general graphs.

[3] The reader might object to a register that stores an arbitrary integer. In this case, consider it to store an integer mod m for some large m.

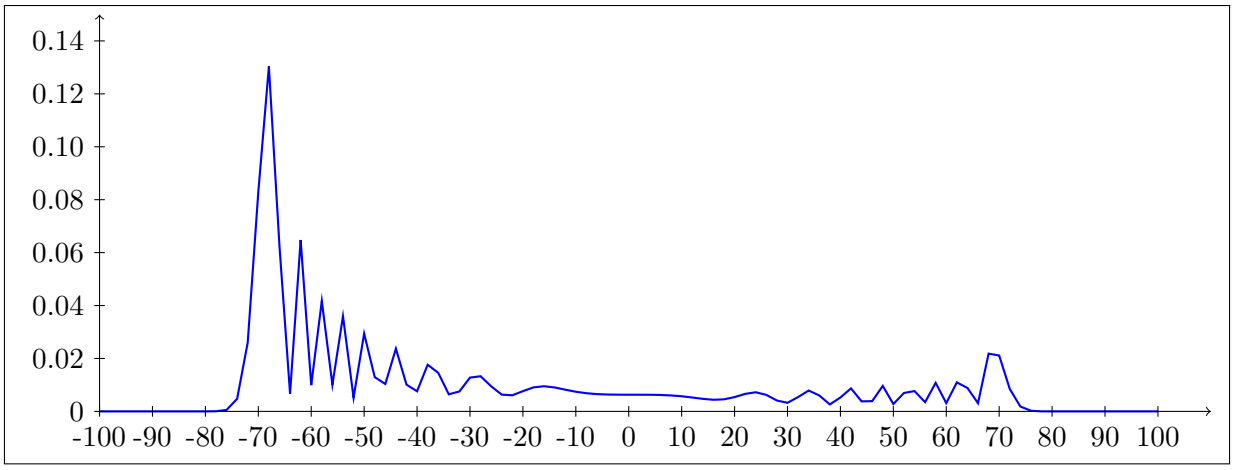


Figure 2: The distribution of positions following a quantum walk on the line for 100 steps using the Hadamard coin. Only even positions are shown as the amplitude at odd positions is 0.

6.2 Quantum walk on general graphs

There is a natural generalisation of the classical random walk on the line to a random walk on an arbitrary graph G with m vertices. The walker is positioned at a vertex of G, and at each time step, it chooses an adjacent vertex to move to, uniformly at random. Here we will consider only *undirected* graphs where the ability to move from A to B implies the ability to move from B to A. Further, we restrict to *regular* graphs for simplicity, i.e. those whose every vertex has degree d. Labelling vertices by integers between 1 and m, the probability of being at vertex j after t steps, given that the walk started at vertex i, can then be written in compact form as $\langle j|M^t|i\rangle$ for some matrix M, where

$$M_{ij} = \begin{cases} \frac{1}{d} & \text{if } i \text{ is connected to } j \\ 0 & \text{otherwise.} \end{cases}$$

Random walks on graphs have been intensively studied in computer science for their many algorithmic applications, as well as for their intrinsic mathematical interest. For example, one of the most efficient (and simplest) algorithms known for the fundamental problem of boolean satisfiability (SAT) is based on a random walk [17].

We now consider how a quantum generalisation of this process can be found. We still have position and coin registers, but now the position register is m-dimensional and the coin register is d-dimensional. Label each vertex with a distinct integer between 1 and m, and for each vertex, label its outgoing edges with distinct integers between 1 and d such that, for each i, the edges labelled with i form a cycle. For each vertex $v \in \{1, \ldots, m\}$, let $N(v, i)$ denote the i'th neighbour of v (i.e. the vertex at the other end of the i'th edge).

Then our quantum walk will once again consist of alternating shift and coin operators S and C, i.e. each step is of the form $(S(I \otimes C))$. The shift operator simply performs the map

$$S|v\rangle|i\rangle = |N(v, i)\rangle|i\rangle,$$

and the coin operator C once again acts only on the coin register. However, as this is now d-dimensional, we have many possible choices for C. One reasonable choice is the so-called *Grover*

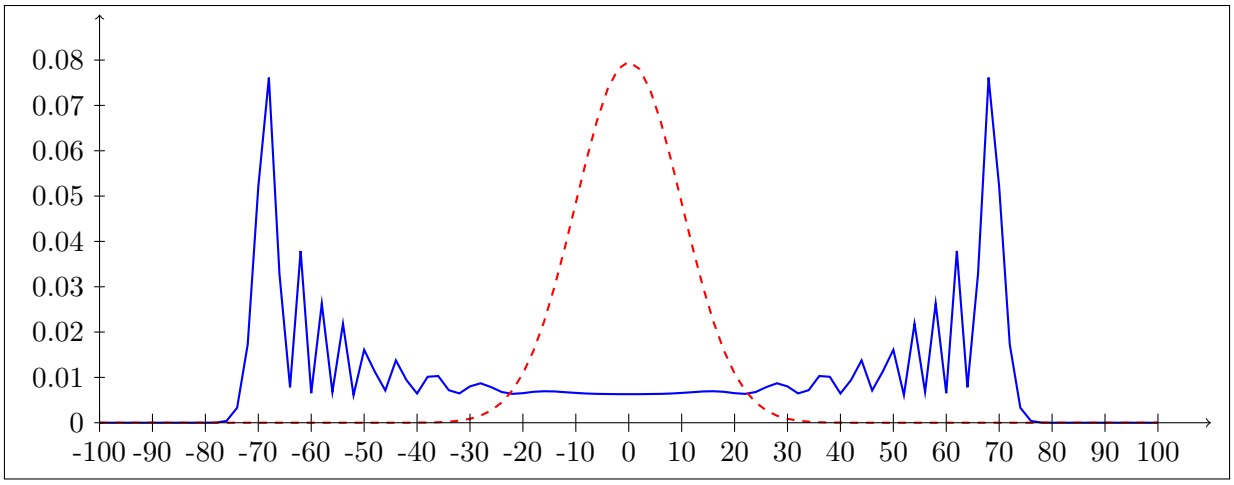


Figure 3: Quantum walk on the line for 100 steps with starting coin state $\frac{1}{\sqrt{2}}|0\rangle(|L\rangle + i|R\rangle)$, compared with classical walk for 100 steps (red dashed line). Only even positions are shown as the amplitude at odd positions is 0.

coin,

$$C = \begin{pmatrix} \frac{2}{d} - 1 & \frac{2}{d} & \cdots & \frac{2}{d} \\ \frac{2}{d} & \frac{2}{d} - 1 & \cdots & \frac{2}{d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{d} & \frac{2}{d} & \cdots & \frac{2}{d} - 1 \end{pmatrix}.$$

This is just the iteration used in Grover's algorithm. This operator is an appealing choice because it is permutation-symmetric (i.e. treats all edges equally), and it is far away from the identity matrix (i.e. has a large mixing effect). If $d = 2$, we would get $C = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, so in this case the coins used earlier for the walk on the line lead to more interesting behaviour.

6.3 Exponentially faster hitting on the hypercube

We now focus on one particularly interesting graph: the n-dimensional hypercube (aka the Cayley graph of the group $\mathbb{Z}_2^n$). This is the graph whose vertices are n-bit strings which are adjacent if they differ in exactly one bit. Each edge $x \leftrightarrow y$ of this graph is naturally labelled by the index of the single bit where x and y differ. We will be interested in the expected time it takes for a random walk on this graph to travel from the "all zeroes" string 0^n to the "all ones" string 1^n, i.e. to traverse the graph from one extremity to the other, which is known as the *hitting time* from 0^n to 1^n.

Classically, this time can be analysed by mapping the walk to a (biased) random walk on the line. Imagine the walker is currently at a vertex with Hamming weight k. The probability of moving to a vertex with Hamming weight $(k-1)$ is k/n, and the probability of moving to a vertex with Hamming weight $(k+1)$ is $1 - k/n$. Observe that, as k increases, the probability of a step leading to the Hamming weight increasing decreases, so intuitively the walker becomes "stuck" in the "middle" of the graph (i.e. near Hamming weight $n/2$).

More rigorously, we have the following proposition.

Proposition 19. *The hitting time from* 0^n *to* 1^n *is at least* $2^n - 1$.

Proof. Let $h(k)$ be the expected number of steps until the walk hits 1^n, given that it starts with Hamming weight k. We have the recurrence

$$h(k) = \frac{k}{n}h(k-1) + \left(1 - \frac{k}{n}\right)h(k+1) + 1,$$

with the boundary case $h(n) = 0$. Writing $\delta(k) = h(k) - h(k+1)$, this can be simplified to

$$\delta(k) = \left(\frac{n}{k+1} - 1\right)\delta(k+1) - \frac{n}{k+1}$$

and the recurrence solved to give

$$\delta(k) = \frac{1}{\binom{n-1}{k}}\sum_{j=0}^{k}\binom{n}{j}, \text{ and hence } h(0) = \sum_{k=0}^{n-1}\delta(n) = \sum_{k=0}^{n-1}\frac{1}{\binom{n-1}{k}}\sum_{j=0}^{k}\binom{n}{j}.$$

Thus $h(0)$ can be lower bounded by rearranging the sum to give

$$h(0) = \sum_{j=0}^{n-1}\binom{n}{j}\sum_{k=j}^{n-1}\frac{1}{\binom{n-1}{k}} \geq \sum_{j=0}^{n-1}\binom{n}{j} = 2^n - 1.$$

□

We thus see that the expected time to reach the 1^n vertex is exponential in n. However, for quantum walks the situation is very different, and we have the following result.

Theorem 20. *If a quantum walk on the hypercube is performed for $T \approx \frac{\pi}{2}n$ steps starting in position 0^n, and the position register is measured, the outcome 1^n is obtained with probability $1 - O(\text{polylog}(n)/n)$.*

Once again, the proof of this result is too technical to include here (for details, see [12]). However, we can give a sketch of the first part of one proof strategy, which is analogous to the classical case, and consists of simplifying to a walk on the line. Define a set of $2n$ states $\{|v_k, L\rangle, |v_k, R\rangle\}$ indexed by an integer $k = 0, \ldots, n$ as follows:

$$|v_k, L\rangle := \frac{1}{\sqrt{k\binom{n}{k}}}\sum_{x,|x|=k}\sum_{i,x_i=1}|x\rangle|i\rangle, \; |v_k, R\rangle := \frac{1}{\sqrt{(n-k)\binom{n}{k}}}\sum_{x,|x|=k}\sum_{i,x_i=0}|x\rangle|i\rangle,$$

with the exception of the special cases $|v_0, L\rangle$ and $|v_n, R\rangle$, which will not be used and are undefined. Now observe that the quantum walk on the hypercube preserves tbe subspace spanned by this set of states:

$$S|v_k, L\rangle = \frac{1}{\sqrt{k\binom{n}{k}}}\sum_{x,|x|=k}\sum_{i,x_i=1}|x \oplus e_i\rangle|i\rangle = \frac{1}{\sqrt{k\binom{n}{k}}}\sum_{x,|x|=k-1}\sum_{i,x_i=0}|x\rangle|i\rangle = |v_{k-1}, R\rangle,$$

and similarly $S|v_k, R\rangle = |v_{k+1}, L\rangle$. In the case of the coin operator, it turns out that

$$\begin{aligned}(I \otimes C)|v_k, L\rangle &= \left(\frac{2k}{n} - 1\right)|v_k, L\rangle + \frac{2\sqrt{k(n-k)}}{n}|v_k, R\rangle \\ \text{and } (I \otimes C)|v_k, R\rangle &= \frac{2\sqrt{k(n-k)}}{n}|v_k, L\rangle + \left(1 - \frac{2k}{n}\right)|v_k, R\rangle.\end{aligned}$$

This behaviour is similar to the quantum walk on the line, with two differences: first, the direction in which the walker is moving flips with each shift, and second, the coin is different at each position (i.e. depends on k). Based on this reduction, it is easy to plot the behaviour of this quantum walk numerically for small n, as shown in Figure 4.

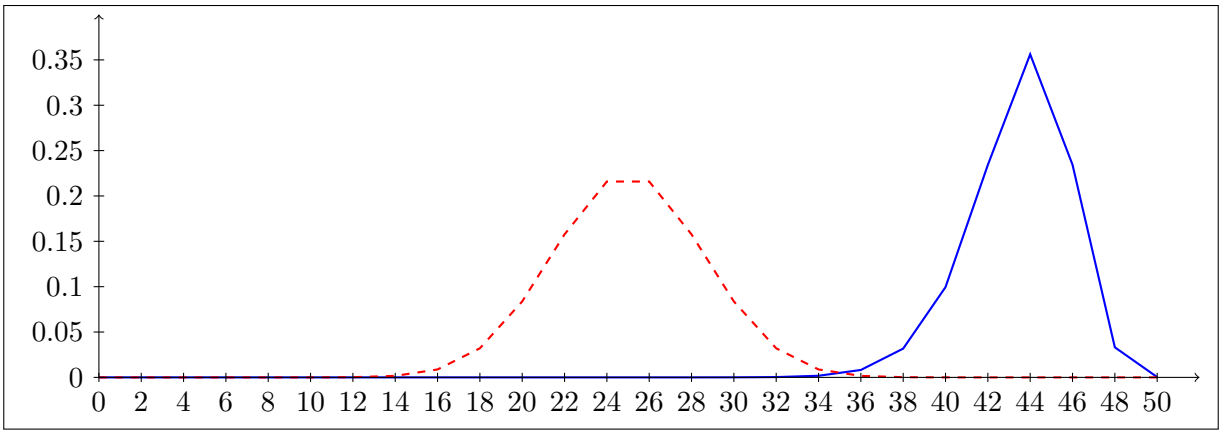


Figure 4: Quantum walk on a 50-dimensional hypercube for 60 steps starting in state $|v_0, R\rangle$, compared with classical walk for 1000 steps (red dashed line). Graph plots probability of being at a point with Hamming weight k. Only even positions are shown as the amplitude at odd positions is 0.

6.4 Further reading

Two good (though now somewhat dated) survey papers on quantum walks are [1, 11]. The result that quantum walks on the hypercube hit exponentially faster is originally due to Julia Kempe [12]; also see [18] for a connection to quantum search. Perhaps surprisingly, this exponentially faster hitting does not imply an exponential quantum speed-up over classical algorithms for search on the hypercube (i.e. the task of finding the 1^n vertex, starting at the 0^n vertex). However, such a speed-up has been proven for a different family of graphs [6].

References

[1] A. Ambainis. Quantum walks and their algorithmic applications. *International Journal of Quantum Information*, 1:507–518, 2003. `quant-ph/0403120`.

[2] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. `quant-ph/9802049`.

[3] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. `quant-ph/9701001`.

[4] D. Berry, G. Ahokas, R. Cleve, and B. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Comm. Math. Phys.*, 270(2):359–371, 2007. `quant-ph/0508139`.

[5] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21–43, 2002.

[6] A. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proc. 35th Annual ACM Symp. Theory of Computing*, pages 59–68, 2003. `quant-ph/0209131`.

[7] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6–7):467–488, 1982.

[8] D. Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation, 2009. `arXiv:0904.2557`.

[9] A. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for solving linear systems of equations. *Phys. Rev. Lett.*, 15(103):150502, 2009. `arXiv:0811.3171`.

[10] P. Høyer and R. Špalek. Lower bounds on quantum query complexity. *Bulletin of the European Association for Theoretical Computer Science*, 87:78–103, 2005. `quant-ph/0509153`.

[11] J. Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 44(4):307–327, 2003. `quant-ph/0303081`.

[12] J. Kempe. Quantum random walks hit exponentially faster. *Probability Theory and Related Fields*, 133(2):215–235, 2005. `quant-ph/0205083`.

[13] S. Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[14] G. Midrijānis. Exact quantum query complexity for total Boolean functions, 2004. `quant-ph/0403168`.

[15] A. Nayak and A. Vishwanath. Quantum walk on the line, 2000. `quant-ph/0010117`.

[16] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.

[17] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proc. 40th Annual Symp. Foundations of Computer Science*, pages 410–414, 1999.

[18] N. Shenvi, J. Kempe, and K. Whaley. Quantum random-walk search algorithm. *Phys. Rev. A.*, 67(5):052307, 2003. `quant-ph/0210064`.

[19] C. Zalka. Grover's quantum searching algorithm is optimal. *Phys. Rev. A.*, 60(4):2746–2751, 1999. `quant-ph/9711070`.