Quantum algorithms (CO 781/CS 867/QIC 823, Winter 2011)
Andrew Childs, University of Waterloo

LECTURE 1: Quantum circuits

This is a course on quantum algorithms. It is intended for graduate students who have already taken an introductory course on quantum information. Such an introductory course typically covers only the early breakthroughs in quantum algorithms, namely Shor's factoring algorithm (1994) and Grover's searching algorithm (1996). The purpose of this course is to show that there is more to quantum computing than Shor and Grover by exploring some of the many quantum algorithms that have been developed since then.

The course will consist of three main parts.

- In the first part, we will discuss algorithms that generalize the main idea of Shor's algorithm. These algorithms make use of the quantum Fourier transform, and typically achieve an exponential (or at least superpolynomial) speedup over classical computers. In particular, we will explore a group-theoretic problem called the *hidden subgroup problem*. We will see how a solution of this problem for abelian groups leads to several applications, and we will also discuss what is known about the nonabelian case.
- In the second part, we will explore the concept of *quantum walk*, a quantum generalization of random walk. This concept leads to a powerful framework for solving search problems, generalizing Grover's search algorithm. It can also be used to solve other problems in query complexity, such as evaluating Boolean formulas. Most of these applications involve a polynomial speedup over classical computers (although there are also some indications that quantum walk may be useful for more dramatic speedups).
- In the third (short) part, we will discuss lower bounds on quantum query complexity, demonstrating limitations on the power of quantum algorithms. We will cover the two main quantum lower bound techniques, the adversary method and the polynomial method.

We will also discuss a few other topics along the way, including the simulation of quantum dynamics and (time permitting) approximate computation of the Jones polynomial.

In this lecture, we will briefly review some background material on quantum computation. If you plan to take this course, most of this material should be familiar to you (except perhaps the details of the Solovay-Kitaev theorem).

Quantum data

A quantum computer is a device that uses a quantum mechanical representation of information to perform calculations. Information is stored in quantum bits, the states of which can be represented as ℓ_2-normalized vectors in a complex vector space. For example, we can write the state of n qubits as

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} a_x|x\rangle \tag{1}$$

where the $a_x \in \mathbb{C}$ satisfy $\sum_x |a_x|^2 = 1$. We refer to the basis of states $|x\rangle$ as the *computational basis*.

It will often be useful to think of quantum states as storing data in a more abstract form. For example, given a group G, we could write $|g\rangle$ for a basis state corresponding to the group element

$g \in G$, and

$$|\phi\rangle = \sum_{g \in G} b_g |g\rangle \tag{2}$$

for an arbitrary superposition over the group. We assume that there is some canonical way of efficiently representing group elements using bit strings; it is usually unnecessary to make this representation explicit.

If a quantum computer stores the state $|\psi\rangle$ *and* the state $|\phi\rangle$, its overall state is given by the tensor product of those two states. This may be denoted $|\psi\rangle \otimes |\phi\rangle = |\psi\rangle|\phi\rangle = |\psi, \phi\rangle$.

Quantum circuits

The allowed operations on (pure) quantum states are those that map normalized states to normalized states, namely *unitary operators* U, satisfying $UU^\dagger = U^\dagger U = I$. (You probably know that there are more general quantum operations, but for the most part we will not need to use them in this course.)

To have a sensible notion of *efficient* computation, we require that the unitary operators appearing in a quantum computation are realized by *quantum circuits*. We are given a set of gates, each of which acts on one or two qubits at a time (meaning that it is a tensor product of a one- or two-qubit operator with the identity operator on the remaining qubits). A quantum computation begins in the $|0\rangle$ state, applies a sequence of one- and two-qubit gates chosen from the set of allowed gates, and finally reports an outcome obtained by measuring in the computational basis.

Universal gate sets

In principle, any unitary operator on n qubits can be implemented using only 1- and 2-qubit gates. Thus we say that the set of all 1- and 2-qubit gates is *(exactly) universal*. Of course, some unitary operators may take many more 1- and 2-qubit gates to realize than others, and indeed, a counting argument shows that most unitary operators on n qubits can only be realized using an exponentially large circuit of 1- and 2-qubit gates.

In general, we are content to give circuits that give good approximations of our desired unitary transformations. We say that a circuit with gates $U_1, U_2, \ldots, U_t$ approximates U with precision ϵ if

$$\|U - U_t \ldots U_2 U_1\| \leq \epsilon. \tag{3}$$

Here $\|\cdot\|$ denotes some appropriate matrix norm, which should have the property that if $\|U - V\|$ is small, then U should be hard to distinguish from V no matter what quantum state they act on. A natural choice (which will be suitable for our purposes) is the spectral norm

$$\|A\| := \max_{|\psi\rangle} \frac{\|A|\psi\rangle\|}{\||\psi\rangle\|}, \tag{4}$$

(where $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$ denotes the vector 2-norm of $|\psi\rangle$), i.e., the largest singular value of A. Then we call a set of elementary gates *universal* if any unitary operator on a fixed number of qubits can be approximated to any desired precision ϵ using elementary gates.

It turns out that there are finite sets of gates that are universal: for example, the set $\{H, T, C\}$

with

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad T := \begin{pmatrix} e^{i\pi/8} & 0 \\ 0 & e^{-i\pi/8} \end{pmatrix} \qquad C := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{5}$$

There are situations in which we say a set of gates is *effectively* universal, even though it cannot actually approximate any unitary operator on n qubits. For example, the set $\{H, T^2, \text{Tof}\}$, where

$$\text{Tof} := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

is universal, but only if we allow the use of ancilla qubits (qubits that start and end in the $|0\rangle$ state). Similarly, the basis $\{H, \text{Tof}\}$ is universal in the sense that, with ancillas, it can approximate any *orthogonal* matrix. It clearly cannot approximate complex unitary matrices, since the entries of H and Tof are real; but the effect of arbitrary unitary transformations can be simulated using orthogonal ones by simulating the real and imaginary parts separately.

Equivalence between different universal gate sets

Are some universal gate sets better than others? Classically, this is not an issue: the set of possible operations is discrete, so any gate acting on a constant number of bits can be simulated exactly using a constant number of gates from any given universal gate set. But we might imagine that some quantum gates are much more powerful than others. For example, given two rotations about strange axes by strange angles, it may not be obvious how to implement a Hadamard gate, and we might worry that implementing such a gate to high precision could take a very large number of elementary operations, scaling badly with the required precision.

Fortunately, it turns out that this is not the case: a unitary operator that can be realized efficiently with one set of 1- and 2-qubit gates can also be realized efficiently with another such set. In particular, we have the following.

Theorem (Solovay-Kitaev)**.** *Fix two universal gate sets that are closed under inverses. Then any t-gate circuit using one gate set can be implemented to precision ϵ using a circuit of $t \cdot \text{poly}(\log \frac{t}{\epsilon})$ gates from other set (indeed, there is a classical algorithm for finding this circuit in time $t \cdot \text{poly}(\log \frac{t}{\epsilon})$).*

Thus, not only are the two gate sets equivalent under polynomial-time reduction, but the running time of an algorithm using one gate set is the same as that using the other gate set up to logarithmic factors. This means that even polynomial quantum speedups are robust with respect to the choice of gate set.

To establish this, we first note the basic fact that errors in the approximation of one quantum circuit by another accumulate linearly.

Lemma. *Let U_i, V_i be unitary matrices satisfying $\|U_i - V_i\| \leq \epsilon$ for all $i \in \{1, 2, \ldots, t\}$. Then $\|U_t \ldots U_2 U_1 - V_t \ldots V_2 V_1\| \leq t\epsilon$.*

Proof. We use induction on t. For $t = 1$ the lemma is trivial. Now suppose the lemma holds for a particular value of t. Then by the triangle inequality and the fact that the norm is unitarily invariant ($\|UAV\| = \|A\|$ for any unitary matrices U, V),

$$\begin{aligned}
&\|U_{t+1}U_t \dots U_1 - V_{t+1}V_t \dots V_1\| \\
&\quad = \|U_{t+1}U_t \dots U_1 - U_{t+1}V_t \dots V_1 + U_{t+1}V_t \dots V_1 - V_{t+1}V_t \dots V_1\| && (7)\\
&\quad \leq \|U_{t+1}U_t \dots U_1 - U_{t+1}V_t \dots V_1\| + \|U_{t+1}V_t \dots V_1 - V_{t+1}V_t \dots V_1\| && (8)\\
&\quad = \|U_{t+1}(U_t \dots U_1 - V_t \dots V_1)\| + \|(U_{t+1} - V_{t+1})V_t \dots V_1\| && (9)\\
&\quad = \|U_t \dots U_1 - V_t \dots V_1\| + \|U_{t+1} - V_{t+1}\| && (10)\\
&\quad \leq (t+1)\epsilon, && (11)
\end{aligned}$$

so the lemma follows by induction. □

Thus, in order to simulate a t-gate quantum circuit with total error at most ϵ, it suffices to simulate each individual gate with error at most ϵ/t.

To simulate an arbitrary individual gate, the strategy is to first construct a very fine net covering a very small ball around the identity using the *group commutator*,

$$[\![U, V]\!] := UVU^{-1}V^{-1}. \tag{12}$$

To approximate general unitaries, we will effectively translate them close to the identity.

Note that it suffices to consider unitary gates with determinant 1 (i.e., elements of SU(2)) since a global phase is irrelevant. Let

$$S_\epsilon := \{U \in \mathrm{SU}(2) \colon \|I - U\| \leq \epsilon\} \tag{13}$$

denote the ϵ-ball around the identity. Given sets $\Gamma, S \subseteq \mathrm{SU}(2)$, we say that Γ is an *ϵ-net* for S if for any $A \in S$, there is a $U \in \Gamma$ such that $\|A - U\| \leq \epsilon$. The following result (to be proved later on) indicates how the group commutator helps us to make a fine net around the identity.

Lemma. *If Γ is an ϵ^2-net for S_ϵ, then $[\![\Gamma, \Gamma]\!] := \{[\![U, V]\!] \colon U, V \in \Gamma\}$ is an $O(\epsilon^3)$-net for S_{ϵ^2}.*

To make an arbitrarily fine net, we apply this idea recursively. But first it is helpful to derive a consequence of the lemma that is more suitable for recursion. We would like to maintain the quadratic relationship between the size of the ball and the quality of the net. If we aim for a $k\epsilon^3$-net (for some constant k), we would like it to apply to arbitrary points in $S_{k\epsilon^{3/2}}$, whereas the lemma only lets us approximate points in S_{ϵ^2}. To handle an arbitrary $A \in S_{k\epsilon^{3/2}}$, we first let W be the closest gate in Γ to A. For sufficiently small ϵ we have $k\epsilon^{3/2} < \epsilon$, so $S_{k\epsilon^{3/2}} \subset S_\epsilon$, and therefore $A \in S_\epsilon$. Since Γ is an ϵ^2-net for S_ϵ, we have $\|A - W\| \leq \epsilon^2$, i.e., $\|AW^\dagger - I\| \leq \epsilon^2$, so $AW^\dagger \in S_{\epsilon^2}$. Then can apply the lemma to find $U, V \in \Gamma$ such that $\|AW^\dagger - [\![U, V]\!]\| = \|A - [\![U, V]\!]W\| \leq k^2\epsilon^3$. In other words, if Γ is an ϵ^2-net for S_ϵ, then $[\![\Gamma, \Gamma]\!]\Gamma := \{[\![U, V]\!]W \colon U, V, W \in \Gamma\}$ is a $k^2\epsilon^3$-net for $S_{k\epsilon^{3/2}}$.

Now suppose that Γ_0 is an ϵ_0^2-net for S_{ϵ_0}, and let $\Gamma_i := [\![\Gamma_{i-1}, \Gamma_{i-1}]\!]\Gamma_{i-1}$ for all positive integers i. Then Γ_i is an ϵ_i^2-net for S_{ϵ_i}, where $\epsilon_i = k\epsilon_{i-1}^{3/2}$. Solving this recursion gives $\epsilon_i = (k^2\epsilon_0)^{(3/2)^i}/k^2$.

With these tools in hand, we are prepared to establish the main result.

Proof of the Solovay-Kitaev Theorem. It suffices to consider how to approximate an arbitrary $U \in \mathrm{SU}(2)$ to precision ϵ by a sequence of gates from a given universal gate set Γ.

First we take products of elements of Γ to form a new universal gate set Γ_0 that is an ϵ_0^2-net for SU(2), for some sufficiently small constant ϵ_0. We know this can be done since Γ is universal. Since ϵ_0 is a constant, the overhead in constructing Γ_0 is constant.

Now we can find $V_0 \in \Gamma_0$ such that $\|U - V_0\| \leq \epsilon_0^2$. Since $\|U - V_0\| = \|UV_0^\dagger - I\|$, we have $UV_0^\dagger \in S_{\epsilon_0^2}$. If ϵ_0 is sufficiently small, then $\epsilon_0^2 < k\epsilon_0^{3/2} = \epsilon_1$, so $UV_0^\dagger \in S_{\epsilon_1}$.

Since Γ_0 is an ϵ_0^2-net for SU(2), in particular it is an ϵ_0^2-net for S_{ϵ_0}. Thus by the above argument, Γ_1 is an ϵ_1^2-net for S_{ϵ_1}, so we can find $V_1 \in \Gamma_1$ such that $\|UV_0^\dagger - V_1\| \leq \epsilon_1^2 < k\epsilon_1^{3/2} = \epsilon_2$, i.e., $UV_0^\dagger V_1^\dagger - I \in S_{\epsilon_2}$.

In general, suppose we are given $V_0, V_1, \ldots, V_{i-1}$ such that $UV_0^\dagger V_1^\dagger \ldots V_{i-1}^\dagger \in S_{\epsilon_i}$. Since Γ_i is an ϵ_i^2-net for S_{ϵ_i}, we can find $V_i \in \Gamma_i$ such that $\|UV_0^\dagger V_1^\dagger \ldots V_{i-1}^\dagger - V_i\| \leq \epsilon_i^2$. In turn, this implies that $UV_0^\dagger V_1^\dagger \ldots V_i^\dagger \in S_{\epsilon_{i+1}}$.

Repeating this process t times gives a very good approximation of U by $V_t \ldots V_1 V_0$: we have $\|U - V_t \ldots V_1 V_0\| \leq \epsilon_t^2$. Suppose we consider a gate from Γ_0 to be elementary. (These gates can be implemented using only a constant number of gates from Γ, so there is a constant factor overhead if only count gates in Γ as elementary.) The number of elementary gates needed to implement a gate from Γ_i is 5^i, so the total number of gates in the approximation is $\sum_{i=0}^t 5^i = (5^{t+1} - 1)/4 = O(5^t)$. To achieve an overall error at most ϵ, we need $\epsilon_t^2 = (k^2\epsilon_0)^{(3/2)^t}/k^2 \leq \epsilon$, i.e.,

$$\left(\frac{3}{2}\right)^t > \frac{\log(k^2\epsilon)}{\log(k^2\epsilon_0)}. \tag{14}$$

Thus the number of gates used is $O(\log^\nu \frac{1}{\epsilon})$ where $\nu = \log 5 / \log \frac{3}{2}$.

At this point, it may not be clear that the approximation can be found quickly, since Γ_i contains a large number of points, so we need to be careful about how we find a good approximation $V_i \in \Gamma_i$ of $UV_0^\dagger V_1^\dagger \ldots V_{i-1}^\dagger$. However, by constructing the approximation recursively, it can be shown that the running time of this procedure is poly($\log \frac{1}{\epsilon}$). It will be clearer how to do this after we prove the lemma, but we leave the details as an exercise. □

It remains to prove the lemma. A key idea is to move between the Lie group SU(2) and its Lie algebra, i.e., the Hamiltonians generating these unitaries. In particular, we can represent any $A \in \mathrm{SU}(2)$ as $A = e^{i\vec{a}\cdot\vec{\sigma}}$, where $\vec{a} \in \mathbb{R}^3$ and $\vec{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$ is a vector of Pauli matrices. Note that we can choose $\|\vec{a}\| \leq \pi$ without loss of generality.

In the proof, the following basic facts about SU(2) will be useful.

(i) $\|I - e^{i\vec{a}\cdot\vec{\sigma}}\| = 2\sin\frac{\|\vec{a}\|}{2} = \|\vec{a}\| + O(\|\vec{a}\|^3)$

(ii) $\|e^{i\vec{b}\cdot\vec{\sigma}} - e^{i\vec{c}\cdot\vec{\sigma}}\| = \|\vec{b} - \vec{c}\| + O(\|\vec{b} - \vec{c}\|^3)$

(iii) $[\vec{b}\cdot\vec{\sigma}, \vec{c}\cdot\vec{\sigma}] = 2i(\vec{b}\times\vec{c})\cdot\vec{\sigma}$

(iv) $\|[\![e^{i\vec{b}\cdot\vec{\sigma}}, e^{i\vec{c}\cdot\vec{\sigma}}]\!] - e^{-[\vec{b}\cdot\vec{\sigma}, \vec{c}\cdot\vec{\sigma}]}\| = O(\|\vec{b}\|\|\vec{c}\|(\|\vec{b}\| + \|\vec{c}\|))$

Here the big-O notation is with respect to $\|\vec{a}\| \to 0$ in (i), with respect to $\|\vec{b} - \vec{c}\| \to 0$ in (ii), and with respect to $\|\vec{b}\|, \|\vec{c}\| \to 0$ in (iv).

Proof of Lemma. Let $A \in S_{\epsilon^2}$. Our goal is to find $U, V \in \Gamma$ such that $\|A - [\![U, V]\!]\| = O(\epsilon^3)$.

Choose $\vec{a} \in \mathbb{R}^3$ such that $A = e^{i\vec{a}\cdot\vec{\sigma}}$. Since $A \in S_{\epsilon^2}$, by (i) we can choose $\vec{a}$ so that $\|\vec{a}\| = O(\epsilon^2)$.

Then choose $\vec{b}, \vec{c} \in \mathbb{R}^3$ such that $2\vec{b} \times \vec{c} = \vec{a}$. We can choose these vectors to be orthogonal and of equal length, so that $\|\vec{b}\| = \|\vec{c}\| = \sqrt{\|\vec{a}\|/2} = O(\epsilon)$. Let $B = e^{i\vec{b}\cdot\vec{\sigma}}$ and $C = e^{i\vec{c}\cdot\vec{\sigma}}$. Then the only difference between A and $[\![B, C]\!]$ is the difference between the commutator and the group commutator, which is $O(\epsilon^3)$ by (iv).

However, we need to choose points from the net Γ. So let $U = e^{i\vec{u}\cdot\vec{\sigma}}$ be the closest element of Γ to B, and let $V = e^{i\vec{v}\cdot\vec{\sigma}}$ be the closest element of Γ to C. Since Γ is an ϵ^2-net for S_ϵ, we have $\|U - B\| \leq \epsilon^2$ and $\|V - C\| \leq \epsilon^2$, so in particular $\|\vec{u} - \vec{b}\| = O(\epsilon^2)$ and $\|\vec{v} - \vec{c}\| = O(\epsilon^2)$.

Now by the triangle inequality, we have

$$\|A - [\![U, V]\!]\| \leq \|A - e^{2i(\vec{u}\times\vec{v})\cdot\vec{\sigma}}\| + \|e^{2i(\vec{u}\times\vec{v})\cdot\vec{\sigma}} - [\![U, V]\!]\|. \tag{15}$$

For the first term, using (ii), we have

$$\begin{aligned}
\|A - e^{2i(\vec{u}\times\vec{v})\cdot\vec{\sigma}}\| &= \|e^{2i(\vec{b}\times\vec{c})\cdot\vec{\sigma}} - e^{2i(\vec{u}\times\vec{v})\cdot\vec{\sigma}}\| && (16)\\
&= O(\|\vec{b} \times \vec{c} - \vec{u} \times \vec{v}\|) && (17)\\
&= O(\|(\vec{b} - \vec{u} + \vec{u}) \times (\vec{c} - \vec{v} + \vec{v}) - \vec{u} \times \vec{v}\|) && (18)\\
&= O(\|(\vec{b} - \vec{u}) \times (\vec{c} - \vec{v}) + (\vec{b} - \vec{u}) \times \vec{v} + \vec{u} \times (\vec{c} - \vec{v})\|) && (19)\\
&= O(\epsilon^3). && (20)
\end{aligned}$$

For the second term, using (iii) and (iv) gives

$$\|e^{2i(\vec{u}\times\vec{v})\cdot\vec{\sigma}} - [\![U, V]\!]\| = \|e^{-[\vec{u}\cdot\vec{\sigma}, \vec{v}\cdot\vec{\sigma}]} - [\![U, V]\!]\| = O(\epsilon^3) \tag{21}$$

The lemma follows. □

Note that it is possible to improve the construction somewhat over the version described above. Furthermore, it can be generalized to SU(N) for arbitrary N. In general, the cost is exponential in N^2, but for any fixed N this is just a constant.

Reversible computation

Unitary matrices are invertible: in particular, $U^{-1} = U^\dagger$. Thus any unitary transformation is a reversible operation. This may seem at odds with how we often define classical circuits, using irreversible gates such as AND and OR. But in fact, any classical computation can be made reversible by replacing any irreversible gate $x \mapsto g(x)$ by the reversible gate $(x, y) \mapsto (x, y \oplus g(x))$, and running it on the input $(x, 0)$, producing $(x, g(x))$. In other words, by storing all intermediate steps of the computation, we make it reversible.

On a quantum computer, storing all intermediate computational steps could present a problem, since two identical results obtained in different ways would not be able to interfere. However, there is an easy way to remove the accumulated information. After performing the classical computation with reversible gates, we simply XOR the answer into an ancilla register, and then perform the computation in reverse. Thus we can implement the map $(x, y) \mapsto (x, y \oplus f(x))$ even when f is a complicated circuit consisting of many gates.

Using this trick, any computation that can be performed efficiently on a classical computer can be performed efficiently on a quantum computer: if we can efficiently implement the map $x \mapsto f(x)$ on a classical computer, we can efficiently perform the transformation $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$ on

a quantum computer. This transformation can be applied to any superposition of computational basis states, so for example, we can perform the transformation

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x, 0\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x, f(x)\rangle. \tag{22}$$

Note that this does not necessarily mean we can efficiently implement the map $|x\rangle \mapsto |f(x)\rangle$, even when f is a bijection (so that this is indeed a unitary transformation). However, if we can efficiently invert f, then we can indeed do this efficiently.

Uniformity

When we give an algorithm for a computational problem, we consider inputs of varying sizes. Typically, the circuits for instances of different sizes with be related to one another in a simple way. But this need not be the case; and indeed, given the ability to choose an arbitrary circuit for each input size, we could have circuits computing uncomputable languages. Thus we require that our circuits be *uniformly generated*: say, that there exists a fixed (classical) Turing machine that, given a tape containing the symbol '1' n times, outputs a description of the nth circuit in time poly(n).

Quantum complexity

We say that an algorithm for a problem is *efficient* if the circuit describing it contains a number of gates that is polynomial in the number of bits needed to write down the input. For example, if the input is a number modulo N, the input size is $\lceil \log_2 N \rceil$.

With a quantum computer, as with a randomized (or noisy) classical computer, the final result of a computation may not be correct with certainty. Instead, we are typically content with an algorithm that can produce the correct answer with high enough probability (for a decision problem, bounded above 1/2; for a non-decision problem for which we can check a correct solution, $\Omega(1)$). By repeating the computation many times, we can make the probability of outputting an incorrect answer arbitrarily small.

In addition to considering explicit computational problems, in which the input is a string, we will also consider the concept of *query complexity*. Here the input is a black box transformation, and our goal is to discover some property of the transformation by making as few queries as possible. For example, in Simon's problem, we are given a transformation $f : \mathbb{Z}_2^n \to S$ satisfying $f(x) = f(y)$ iff $y = x \oplus t$ for some unknown $t \in \mathbb{Z}_2^n$, and the goal is to learn t. The main advantage of considering query complexity is that it allows us to prove lower bounds on the number of queries required to solve a given problem. Furthermore, if we find an efficient algorithm for a problem in query complexity, then if we are given an explicit circuit realizing the black-box transformation, we will have an efficient algorithm for an explicit computational problem.

Sometimes, we care not just about the size of a circuit for implementing a particular unitary operation, but also about its *depth*, the maximum number of gates on any path from an input to an output. The depth of a circuit tells us how long it takes to implement if we can perform gates in parallel. In the problem set, you will get a chance to think about parallel circuits for implementing the quantum Fourier transform.

Fault tolerance

In any real computer, operations cannot be performed perfectly. Quantum gates and measurements may be performed imprecisely, and errors may happen even to stored data that is not being manipulated. Fortunately, there are protocols for dealing with faults that may occur during the execution of a quantum computation. Specifically, the *threshold theorem* states that as long as the noise level is below some threshold (depending on the noise model, but typically in the range of 10^{-3} to 10^{-4}, an arbitrarily long computation can be performed with an arbitrarily small amount of error.

In this course, we will always assume implicitly that fault-tolerant protocols have been applied, such that we can effectively assume a perfectly functioning quantum computer.