PRINCETON UNIVERSITY COS 522: COMPUTATIONAL COMPLEXITY

Lecture 23: Quantum Computation

Lecturer: *Sanjeev Arora* Scribe:*Zhifeng Chen, Jia Xu*

This lecture concerns quantum computation, an area that has become very popular recently because it promises to solve certain difficult problems —factoring and discrete logarithm— in polynomial time.

1 Quantum physics

Quantum phenomena are counterintuitive. To see this, consider the basic experiment of quantum mechanice that proves the wave nature of electrons: the 2-slit experiment. (See Figure 1.) A source fires electrons one by one at a wall. The wall contains two tiny slits. On the far side are small detectors that light up whenever an electron hits them. We measure the number of times each detector lights up during the hour. The results are as follows. When we cover one of the slits, we observe the strongest flux of electrons right behind the open slit, as one would expect. However, when both slits are open, we will see the "interference" phenomenon of electrons coming through two slits. In particular, at several detectors the total electron flux is *lower* when both slit are open as compared to when a single slit is open. This defies explanation if electrons behave like little balls, as implied in some high school textbooks. A ball could either go through slit 1 or slit 2, and hence the flux when both slits are open should be a simple sum of the fluxes when one of them is open.

The only explanation physics has for this experiment is that an electron does not behave as a ball. It should be thought of as *simultaneously* going through both slits at once, kind of like a wave. It has an *amplitude* for going through each slit, and this amplitude is a complex number (in particular, it can be a negative number). The chance of an electron appearing at a particular point p on the other side of the wall is related to

$$\text{amplitude of reaching } p \text{ via slit 1} + \text{amplitude of reaching } p \text{ via slit 2.}$$

Thus the points where the electron flux decreases when we open both slits are those where the two amplitudes have opposite sign.

"Nonsense!" you might say. "I need proof that the electron actually went through both slits." So you propose the following modification to the experiment. Position two detectors at the slits; these light up whenever an electron passed through the slit. Now you can test the hypothesis that the electron went through both slits simultaneously.

Unfortunately, when you put such detectors at the slits, the interference phenomenon disappears on the other side of the wall! The explanation is roughly as follows: the quantum nature of particles disappers when they are under observation. More specifically, a quantum system has to evolve according to certain laws, and "nonreversible" operations —such as observation from nosy humans and their detectors— are not allowed. If these nonreversible operations happen, the quantum state collapses. (One moral to draw from this is that

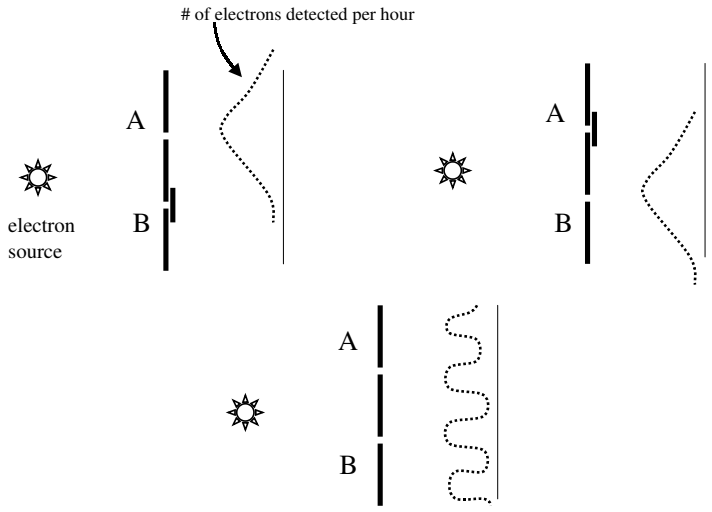


Figure 1: 2-slit experiment

quantum computers, if they are ever built, will have to be carefully isolated from external influences and noise, since noise tends to be an irreversible operation. Of course, we can never completely isolate the system, which means we have to make quantum computation tolerant of a little noise. This is a topic of ongoing research.)

2 Quantum superpositions

Now we describe a *quantum register*, a basic component of the quantum computer. Recall the classical register, the building block of the memory in your desktop computer. An n-bit classical register with n bits consists of n particles. Each of them can be in 2 states: up and down, or 0 and 1. Thus there are 2^n possible configurations, and at any time the register is in one of these configurations.

The n-bit quantum register is similar, except at any time it can exist in a *superposition* of all 2^n configurations. (Drawing inspiration from phenomena such as the 2-slit experiment, researchers have successfully implemented quantum registers using subatomic particles.) Each configuration $S \in \{0,1\}^n$ has an associated amplitude $\alpha_S \in \mathbf{C}$ where $\mathbf{C}$ is the set of complex numbers.

$$\alpha_S = \text{amplitude of being in configuration } S$$

Physicists like to denote this system state succinctly as $\sum_S \alpha_S |S\rangle$. This is their notation for describing a general vector in the vector space $\mathbf{C}^{2^n}$, expressing the vector as a linear combination of basis vectors. The basis contains a vector $|S\rangle$ for each configuration S. The choice of the basis used to represent the configurations is immaterial so long as we fix a basis once and for all.

At every step, actions of the quantum computer —physically, this may involve shining light of the appropriate frequency on the quantum register, etc.— update α_S according to some physics laws. Each computation step is essentially a linear transformation of the system state. Let $\vec{\alpha}$ denote the current configuration (i.e., the system is in state $\sum_S \alpha_S |S\rangle$)

and U be the linear operator. Then the next system state is $\vec{\beta} = U\vec{\alpha}$. Physics laws require U to be unitary, which means $UU^* = I$. (Here U^* is the matrix obtained by transposing U and taking the complex conjugate of each entry.) Note an interesting consequence of this fact: the effect of applying U can be reversed by applying the operator U^*: thus quantum systems are *reversible*. This imposes strict conditions on which kinds of computations are permissible and which are not.

As already mentioned, during the computation steps, the quantum register is isolated from the outside world. Suppose we open the system at some time and observe the state of the register. If the register was in state $\sum_S \alpha_S |S\rangle$ at that moment, then

$$\mathbf{Pr}[\text{we see configuration S}] = |\alpha_S|^2 \tag{1}$$

In particular, we have $\sum_S |\alpha_S|^2 = 1$ at all times. Note that observation is an irreversible operator. We get to see one configuration according to the probability distribution described in (1) and and the rest of the configurations are lost forever.

What if we only observe a few bits of the register —a so-called *partial observation*? Then the remaining bits still stay in quantum superposition. We show this by an example.

EXAMPLE 1 Suppose an n-bit quantum register is in the state

$$\sum_{s\in\{0,1\}^{n-1}} \alpha_s |0\rangle |s\rangle + \beta_s |1\rangle |s\rangle \tag{2}$$

(sometimes this is also represented as $\sum_{s\in\{0,1\}^{n-1}} (\alpha_s |0\rangle + \beta_s |1\rangle) |s\rangle$, and we will use both representations). Now suppose we observe just the first bit of the register and find it to be 0. Then the new state is

$$\frac{1}{\sqrt{|\alpha_s|^2}} \sum_{s\in\{0,1\}^{n-1}} \alpha_s |0\rangle |s\rangle \tag{3}$$

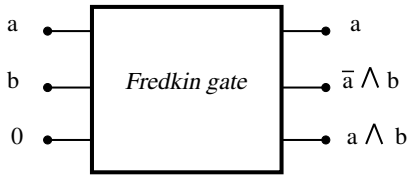
where the first term is a rescaling term that ensures that probabilities in future observations sum to 1.

3 Classical computation using reversible gates

Motivated by the 2nd Law of Thermodynamics, researchers have tried to design computers that expend —at least in principle— zero energy. They have invented *reversible computation*, which, using *reversible gates*, can implement all classical computations. We will study reversible classical gates as a stepping stone to quantum gates; in fact, they are simple examples of quantum gates.

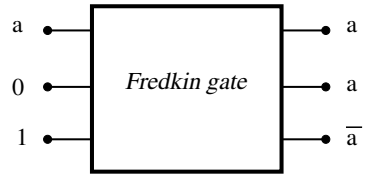
The *Fredkin gate* is a popular reversible gate. It has 3 Boolean inputs and on input (a, b, c) it outputs (a, b, c) if $a = 1$ and (a, c, b) if $a = 0$. It is *reversible*, in the sense that $F(F(a, b, c)) = (a, b, c)$. Simple induction shows that if a circuit is made out of Fredkin gates alone and has m inputs then it must have m outputs as well. Furthermore, we can recover the inputs from the outputs by just applying the circuit in reverse. Hence a Fredkin gate circuit is *reversible*.

The Fredkin gate is *universal*, meaning that every circuit of size S that uses the familiar AND, OR, NOT gates (maximum fanin 2) has an equivalent Fredkin gate circuit of size



Implementing AND

Figure 2:



Implementing NOT (and copy)

Figure 3: Implementing NOT and COPY with Fredkin Gate

$O(S)$. We prove this by showing that we can implement AND, OR, and NOT using a Fredkin gate some of whose inputs have been fixed 0 or 1 (these are "control inputs"); see Figure 2 for AND and Figure 3 for NOT; we leave OR as exercise. We also need to show how to copy a value with Fredkin gates, since in a normal circuit, gates can have fanout more than 1. To implement a COPY gate using Fredkin gates is easy and is the same as for the the NOT gate (see Figure 3).

Thus to transform a normal circuit into a reversible circuit, we replace each gate with its Fredkin implementation, with some additional "control" inputs arriving at each gate to make it compute as AND/OR/NOT. These inputs have to be initialized appropriately.

The transformation appears in Figure 4, where we see that the output contains some junk bits. With a little more work (see Exercises) we can do the transformation in such a way that the output has no junk bits, just the original control bits. Thus the reversible circuit has no effect apart from transforming the input bits into output bits. (We are assuming that the number of input bits equals the number of output bits.)

4 Quantum gates

A 1-input quantum gate (Figure 2) is represented by a unitary 2×2 matrix $U = \left(\begin{smallmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{smallmatrix}\right)$. When its input bit is 0 the output is the superposition $U_{00}|0\rangle + U_{01}|1\rangle$ and when the input is 1 the output is the superposition $U_{10}|0\rangle + U_{11}|1\rangle$. When the input bit is in the superposition $\alpha_0|0\rangle + \beta_0|1\rangle$ the output bit is a superposition of the corresponding outputs

$$(\alpha_0 U_{00} + \beta_0 U_{10})|0\rangle + (\alpha_0 U_{01} + \beta_0 U_{11})|1\rangle. \tag{4}$$

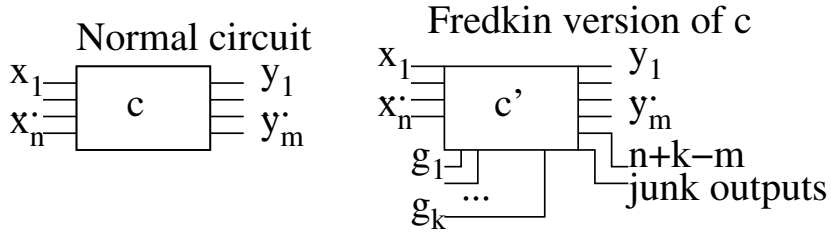


Figure 4: Converting a normal circuit c into an equivalent circuit c' of Fredkin gates. Note that we need additional control inputs

More succinctly, if the input state vector is (α_0, β_0) then the output state vector is

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = U \begin{pmatrix}\alpha_0\\ \beta_0\end{pmatrix}$$

If $|\alpha|^2 + |\beta|^2 = 1$ then unitarity of U implies that $|\alpha'|^2 + |\beta'|^2 = 1$.

Similarly, a 2-input quantum gate is represented by a unitary 4×4 matrix R. When the input is the superposition $\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$, the output is $\beta_{00}|00\rangle + \beta_{01}|01\rangle + \beta_{10}|10\rangle + \beta_{11}|11\rangle$ where

$$\begin{pmatrix}\beta_{00}\\ \beta_{01}\\ \beta_{10}\\ \beta_{11}\end{pmatrix} = R \begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{10}\\ \alpha_{11}\end{pmatrix}$$

In general, a quantum gate with k inputs is specified by a unitary $2^k \times 2^k$ matrix.

Example 2 A Fredkin gate is also a valid 3-input quantum gate. We represent it by an 8×8 matrix that gives its output on all 2^3 possible inputs. This matrix is a permutation matrix (i.e., obtainable from the identity matrix by applying a permutation on all the rows) since the output $F(a, b, c)$ is just a permutation of the input (a, b, c). Exercise 2 asks you to verify that this permutation matrix is unitary.

A quantum circuit on n inputs consists of (a) an n-bit quantum register (b) a sequence of gates $(g_j)_{j=1,2,\ldots}$. If g_j is a k-input gate, then the circuit specification has to also give a sequence of bit positions $(j,1),(j,2),\ldots,(j,k) \in [1,n]$ in the quantum register to which this gate is applied. The circuit computes by applying these gate operations to the quantum register one by one in the specified order. The register holds the state of the computation, and only one gate is applied at any given time.

Example 3 Suppose we have an n-bit quantum register in the state $\sum_{S\in 0,1^n} \alpha_S |S\rangle$. If we apply a 1-input quantum gate U to the first wire, the new system state is computed as follows. First "factor" the initial state by expressing each n-bit configuration as a concatenation of the first bit with the remaining $n-1$ bits:

$$\sum_{S' \in \{0,1\}^{n-1}} \alpha_{0,S'} \left|0S'\right\rangle + \alpha_{1,S'} \left|1S'\right\rangle. \tag{5}$$

(Formally we could express everything we are doing in terms of tensor product of vector spaces but we will not do that.)

To obtain the final state apply U on the first bit in each configuration as explained in equation (4). This yields

$$\sum_{S' \in \{0,1\}^{n-1}} (\alpha_{0,S'} U_{00} + \alpha_{1,S'} U_{10}) \left|0S'\right\rangle + (\alpha_{0,S'} U_{01} + \alpha_{1,S'} U_{11}) \left|1S'\right\rangle \tag{6}$$

We can similarly analyze the effect of applying a k-input quantum gate on any given set of k bits of the quantum register, by first "factoring" the state vector as above.

4.1 Universal quantum gates

You may now be a little troubled by the fact that the number of possible quantum gates with a single input is the set of all unitary 2×2 matrices, an uncountable set. That seems like bad news for the Radio Shacks of the future, who may feel obliged to keep all possible quantum gates in their inventory, to allow their customers to build all possible quantum circuits.

Luckily, Radio Shack need not fear. Researchers have shown the existence of a small set of "universal" 2-input quantum gates such that every circuit composed of S arbitrary k-input quantum gates can be simulated using a circuit of size $\text{poly}(k) \cdot S \log S$ composed only of our universal gates. The simulation is not exact and the distribution on outputs is only approximately the same as the one of the original circuit. (All of this assumes the absence of any outside noise; simulation in presence of noise is a topic of research and currently seems possible under some conditions.)

In any case, we will not need any fancy quantum gates below; just the Fredkin gate and the following 1-input gate called the Hadamard gate:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

5 BQP

DEFINITION 1 *A language $L \subseteq \{0,1\}^*$, is in* **BQP** *iff there is a family of quantum circuits (C_n) of size n^c s.t. $\forall x \in \{0,1\}^*$:*

$$x \in L \Rightarrow \mathbf{Pr}[C(x)_1 = 1] \geq \frac{2}{3}$$
$$x \notin L \Rightarrow \mathbf{Pr}[C(x)_1 = 1] \leq \frac{1}{3}$$

Here $C(x)_1$ is the first output bit of circuit C, and the probability refers to the probability of observing that this bit is 1 when we "observe" the outputs at the end of the computation.

The circuit has to be uniform, that is, a deterministic polynomial time (classical) Turing machine must be able to write down its description.

At first the uniformity condition seems problematic because a classical Turing machine cannot write complex numbers needed to describe quantum gates. However, the machine can just express the circuit approximately using universal quantum gates, which is a finite family. The approximate circuit computes the same language because of the gap between the probabilities 2/3 and 1/3 used in the above definition.

THEOREM 1
P ⊆ BQP

PROOF: Every language in **P** has a uniform circuit family of polynomial size. We transform these circuits into reversible circuits using Fredkin gates, thus obtaining a quantum circuit family for the language. □

THEOREM 2
BPP ⊆ BQP

PROOF: Every language in **BPP** has a uniform circuit family (C_n) of polynomial size, where circuit C_n has n normal input bits and an additional n^k input wires that have to be initialized with random bits.

We transform the circuit into a reversible circuit C'_n using Fredkin gates. To produce "random" bits, we feed $O(n^c)$ zeros into an array of Hadamard gates and plug their outputs into C'_n; see Figure 5.

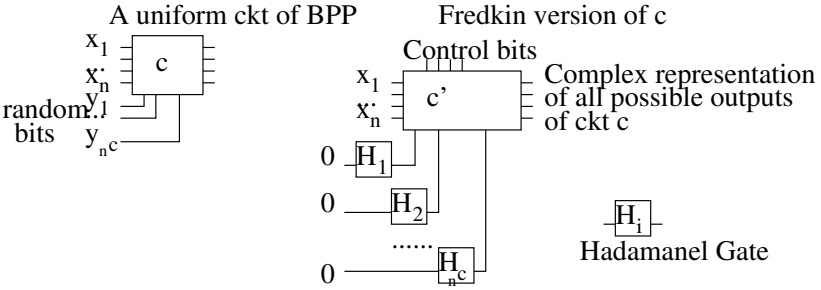


Figure 5: C'_n is a Fredkin circuits by replacing each normal gate of C_n with a respective quantum gate. An array of Hadamard gates produce random bits.

To see that this works, imagine fixing the first n bits to a string x in both circuits.

A simple induction shows that if we start with an N-bit quantum register in the state $\left|\vec{0}\right\rangle$ and apply the Hadamard gate one by one on all the bits, then we obtain the superposition

$$\sum_{S \in 0,1^N} \frac{1}{2^{N/2}} \left|S\right\rangle \tag{7}$$

Thus the "random" inputs of circuit C'_n get a uniform quantum superposition of all possible bit strings. Thus the output bit of C'_n is a uniform superposition of the result on each bit string and observing it at the end gives the probability that C_n accepts when fed a random input. □

6 Factoring integers using a quantum computer

This section proves the following famous result.

THEOREM 3 (SHOR 1994)
There is a polynomial size quantum circuit that factors integers.

We describe Kitaev's proof of this result since it uses eigenvalues, a more familiar and intuitive concept than the Fourier transforms used in Shor's algorithm.

DEFINITION 2 (EIGENVALUE) *λ is an* eigenvalue *of matrix M if there is a vector e (called the* eigenvector*) , s.t.:*

$$M \cdot e = \lambda e$$

Fact: If M is unitary, then $|\lambda| = 1$. In other words there is a $\theta \in [0, 1)$ such that

$$\lambda = e^{2\pi i\theta} = \cos(2\pi\theta) + i\sin(2\pi\theta).$$

Fact: If $M \cdot e = \lambda e$ then $M^k \cdot e = \lambda^k e$. Hence e is still an eigenvector of M^k and λ^k is the corresponding eigenvalue.

Now we are ready to describe Kitaev's algorithm and prove Theorem 3.
PROOF: Let N be the number to be factored. As usual, Z_N^* is the set of numbers mod N that are co-prime to N. Simple number theory shows that for every $a \in Z_N^*$ there is a smallest integer r such that $a^r \equiv 1 \pmod N$; this r is called the *order* of a. The algorithm uses the well-known fact that if we can compute the order of random elements of Z_N^* then we can factor N with high probability. The reason is that if $(a^r - 1) \equiv 0 \pmod N$, then $(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \pmod N$. If a is random, with probability $\geq \frac{1}{2}$, $a^{\frac{r}{2}} \neq 1 \pmod N, a^{\frac{r}{2}} \neq -1 \pmod N$ (this is a simple exercise using the Chinese remainder theorm) and hence $gcd(N, a^{\frac{r}{2}} - 1) \neq N, 1$. Thus, knowing r we can compute $a^{r/2}$ and compute $gcd(N, a^{\frac{r}{2}} - 1)$. With probability at least $1/2$ (over the choice of a) this method yields a factor of N.

The factoring algorithm is a mixture of a classical and a quantum algorithm. Using classical random bits it generates a random $a \in Z_N^*$ and then constructs a quantum circuit. Observing the output of this quantum circuit a few times followed by some more classical computation allows it to obtain r, the order of a, with reasonable probability. (Of course, we could in principle describe the entire algorithm as a quantum algorithm instead of as a mixture of a classical and a quantum algorithm, but our description isolates exactly where quantum mechanics is crucial.)

Consider a classical reversible circuit that acts on numbers in Z_N^*, and is described by $U(x) = ax \pmod N$. Then we can view this circuit as a quantum circuit operating on a quantum register. If the quantum register is in the superposition[1]

$$\sum_{x \in Z_N^*} \alpha_x |x\rangle,$$

[1]Aside: This isn't quite correct; the cardinality of Z_N^* is not a power of 2 so we cannot construct a quantum register whose only possible configurations correspond to elements of Z_N^*. However, if we use the nearest power of 2, everything we are about to do will still be approximately correct.

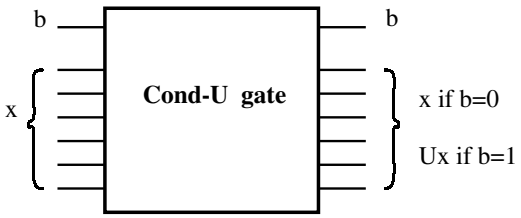


Figure 6: Conditional-U circuit

then applying U gives the superposition

$$\sum_{x \in Z_N^*} \alpha_x \,|\, ax \pmod N \rangle.$$

Interpret this quantum circuit as an $N \times N$ matrix —also denoted U—and consider its eigenvalues. Since $U^r = I$, we can easily check that each eigenvalue has the form $e^{2\pi i\theta}$ where $\theta = \frac{j}{r}$ for some $j \leq r$. The algorithm will try to obtain a random eigenvalue. It thus obtains —in binary expansion— a number of form $\frac{j}{r}$ where j is random. It turns out that the chance is pretty good that this j is coprime to r, which means that $\frac{j}{r}$ is an irreducible fraction. Even knowing only the first $2 \log N$ bits in the binary expansion of $\frac{j}{r}$, the algorithm can round off to the nearest fraction whose denominator is at most N (this can be done; easy number theory) and then it reads off r from the denominator.

Now we describe how to compute the first $2 \log N$ bits of a random eigenvalue of U. Assume for now that the algorithm has a quantum register whose state is a superposition, denoted $\vec{e}$, corresponding to a random eigenvector of U. (See below for details on how to put a register in such a state.) Then applying U gives the final state $\lambda\vec{e}$, where λ is the eigenvalue associated with $\vec{e}$. Thus the register's state has undergone a *phase shift* —i.e., multiplication by a scalar— although there is yet no direct way to measure λ.

Now define a conditional-U circuit (Figure 6), whose input is (b, x) where b is a bit, and cond-$U(b, x) = (b, x)$ if $b = 0$ and $(b, ax \pmod N)$ if $b = 1$. Again, notice that this can be implemented using classical Fredkin gates.

Using cond-U circuit and two Hadamard gates, we can build a quantum circuit shown in Figure 7. When this is applied to a quantum register whose first bit is 0 and the remaining bits are in a state $\vec{e}$, then we can measure the corresponding eigenvalue λ by repeated measurement of the first output bit.

$$\begin{aligned} |0\rangle\,|e\rangle &\xrightarrow{H_1} \frac{1}{\sqrt{2}}\,|0\rangle\,|e\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\,|e\rangle \\ &\xrightarrow{\text{cond-U}} \frac{1}{\sqrt{2}}\,|0\rangle\,|e\rangle + \frac{\lambda}{\sqrt{2}}\,|1\rangle\,|e\rangle \\ &\xrightarrow{H_2} \frac{1}{2}((1+\lambda)\,|0\rangle\,|e\rangle + (1-\lambda)\,|1\rangle\,|e\rangle) \end{aligned} \tag{8}$$

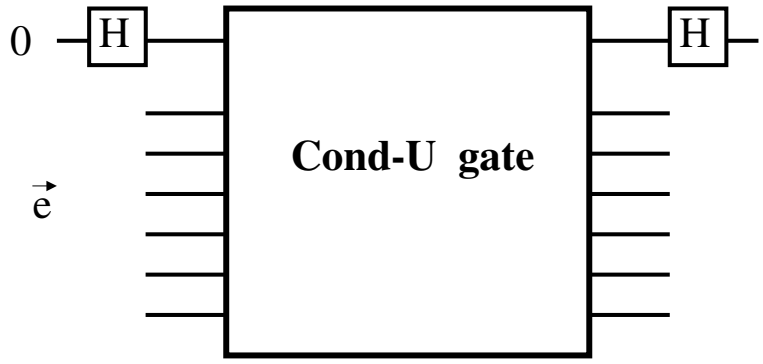


Figure 7: Basic block: Conditional-U gate and two Hadamard gates.

Thus the probability of measuring a 0 in the first bit is proportional to $|1 + \lambda|$. We will refer to this bit as the *phase bit*, since repeatedly measuring it allows us to compute better and better estimates to λ. Actually, instead of repeated measuring we can just design a quantum circuit to do the repetitions by noticing that the output is just a scalar multiple of $\vec{e}$ again, so we can just feed it into another basic block with a fresh phase bit, and so on (see Figure 8). We measure phase bits all at once at the end.

Are we done? Unfortunately, no. Obtaining an estimate to the first m bits of λ would involve a circuit with 2^m phase bits (this is a simple exercise about probabilistic estimation), and when $m = 2 \log N$, this number is about N, whereas we are hoping for a circuit size of poly($\log N$). Thus simple repetition is a very inefficient way to obtain accurate information about λ.

A more efficient technique involves the observation that U has a special property: powers of U also have small circuits. Specifically, $U^{2^k}(x) = a^{2^k} x \pmod N$, and a^{2^k} is computable by circuits of size poly($\log N + \log k$) using fast exponentiation.

Thus we can implement a conditional-U^{2^k} gate using a quantum circuit of size poly($\log N + k$). The eigenvalues of U^{2^k} are λ^{2^k}. If $\lambda = e^{2\pi i \theta}$ where $\theta \in [0, 1)$ (see Figure 9) then $\lambda^{2^k} = e^{2\pi i \theta 2^k}$. Of course, $e^{2\pi i \theta 2^k}$ is the same complex number as $e^{2\pi i \alpha}$ where $\alpha = 2^k \theta \pmod 1$. Thus measuring λ^{2^k} gives us $2^k \theta \pmod 1$ and in particular the most significant bit of $2^k \theta \pmod 1$ is nothing but the kth bit of θ. Using $k = 0, 1, 2, \ldots 2 \log N$ we can obtain the first $2 \log N$ bits of θ.

As in Figure 8, we can bundle these steps into a single cascading circuit where the output of the conditional-$U^{2^{k-1}}$ circuit feeds into the conditional-U^{2^k} circuit. Each circuit has its own set of $O(\log N)$ phase bits; measuring the phase bits of the kth circuit gives an estimate of the kth bit of θ that is correct with probability at least $1 - 1/N$. All phase bits are measured in a single stroke at the end.

To finish, we show how to put a quantum register into a state corresponding to an eigenvector.

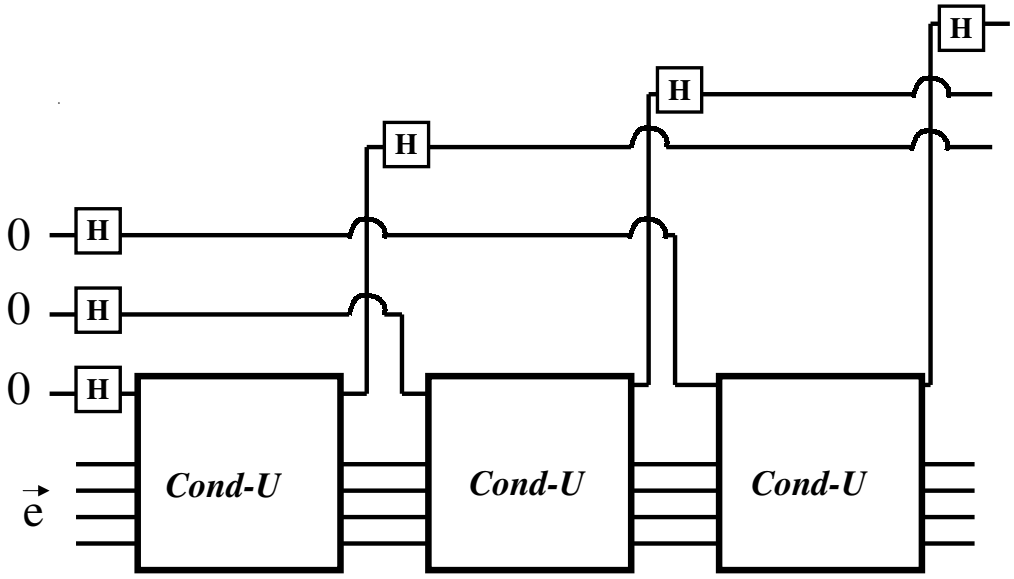


Figure 8: Repeating the basic experiment to get better estimate of λ.

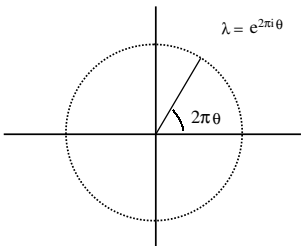


Figure 9: Eigenvalue $\lambda = e^{2\pi i\theta}$ in the complex plane.

6.1 Uniform superpositions of eigenvectors of U

Actually, we show how to put the quantum register into a uniform superposition of eigenvectors of U. This suffices for our cascading circuit, as we will argue shortly.

First we need to understand what the eigenvectors look like. Recall that $\left\{1, a, a^2, \ldots, a^{r-1}\right\}$ is a subgroup of Z_N^*. Let B be a set of representatives of all cosets of this subgroup. In other words, for each $x \in Z_N^*$ there is a unique $b \in B$ and $l \in \{0, 1, \ldots, r-1\}$ such that $x = ba^l \pmod N$. Then the following is the complete set of eigenvectors, where $\omega = e^{\frac{2\pi i}{r}}$:

$$\forall j \in \{0, 1, \ldots, r-1\}, \forall b \in B \qquad \vec{e}_{j,b} = \sum_{l=0}^{r-1} \omega^{jl} \left| \, ba^l \pmod N \right\rangle \tag{9}$$

The eigenvalue associated with this eigenvector is $\omega^{-j} = e^{-\frac{2\pi ij}{r}}$.

Fix b and consider the uniform superposition:

$$\frac{1}{r}\sum_{j=0}^{r-1} \vec{e}_{j,b} = \frac{1}{r}\sum_{j=0}^{r-1}\sum_{l=0}^{r-1} \omega^{jl} \left| \, ba^l \pmod N \right\rangle \tag{10}$$

$$= \frac{1}{r}\sum_{l=0}^{r-1}\sum_{j=0}^{r-1} \omega^{jl} \left| \, ba^l \pmod N \right\rangle. \tag{11}$$

Separating out the terms for $l = 0$ and using the formula for sums of geometric series:

$$= \frac{1}{r}\Big(\sum_{j=0}^{r-1} |b\rangle + \sum_{l=1}^{r-1} \frac{(\omega^l)^r - 1}{\omega^l} \left| \, ba^l \pmod N \right\rangle\Big) \tag{12}$$

since $\omega^r = 1$ we obtain

$$= |b\rangle \tag{13}$$

□

Thus if we pick an arbitrary b and feed the state $|b\rangle$ into the quantum register, then that can also be viewed as a uniform superposition $\frac{1}{r}\sum_j \vec{e}_{j,b}$.

6.2 Uniform superposition suffices

Now we argue that in the above algorithm, a uniform superposition of eigenvectors is just as good as a single eigenvector.

Fixing b, the initial state of the quantum register is

$$\frac{1}{r}\sum_j \left|\vec{0}\right\rangle |\vec{e}_{j,b}\rangle,$$

where $\vec{0}$ denotes the vector of phase bits that is initialized to 0. After applying the quantum circuit, the final state is

$$\frac{1}{r}\sum_j |c_j\rangle |\vec{e}_{j,b}\rangle,$$

where $|c_j\rangle$ is a state vector for the phase bits that, when observed, gives the first $2\log N$ bits of j/r with probability at least $1 - 1/N$. Thus observing the phase bits gives us whp a random eigenvalue.

Exercises

§1 Implement an OR gate using the Fredkin gate.

§2 Given any classical circuit computing a function f from n bits to n bits, describe how to compute the same function with a reversible circuit that is a constant factor bigger and has no "junk" output bits. (Hint: Use our transformation, then copy the output to a safe place, and then run the circuit in reverse to erase the junk outputs.)

§3 Verify that the Fredkin gate is a valid quantum gate.

§4 Verify all the number theoretic facts mentioned in the description of the factoring algorithm.