11 Quantum Complexity Theory II

In this section we introduce basic techniques for performing quantum computations. We will apply them to two problems: a problem by Deutsch and Jozsa and the prime factoring problem.

11.1 Quantum bits

The simplest unit to apply quantum operations to is a single bit, called *quantum bit* or *qubit* in the following. A qubit can be realized in many different ways such as, for instance, by the polarization of a photon or the spin of an atom. If a qubit is realized with the help of the polarization of a photon, we may view the basis states $|0\rangle$ and $|1\rangle$ as a horizontally and vertically polarized photon. A photon going through a transparent tank of sugar water will have its polarization slowly rotated. The amount of rotation depends on the length of the tank and the density of sugar. By appropriately setting these parameters, the tank can be made to induce a 45 degree rotation on incoming photons. Thus, it will have the following effect:

- $|0\rangle$ will be transformed into $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and
- $|1\rangle$ will be transformed into $\frac{1}{\sqrt{2}}(-|0\rangle + |1\rangle)$.

The transformation induced on the basis states completely determines the matrix A. If $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ is shot through the tank, it will come out transformed into state $|\varphi'\rangle$ where

$$\begin{aligned}
|\varphi'\rangle &= A|\varphi\rangle \\
&= A(\alpha|0\rangle + \beta|1\rangle) \\
&= \alpha A|0\rangle + \beta A|1\rangle \\
&= \alpha\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) + \beta\left(\frac{1}{\sqrt{2}}(-|0\rangle + |1\rangle)\right) \\
&= \frac{1}{\sqrt{2}}(\alpha - \beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha + \beta)|1\rangle
\end{aligned}$$

Or, in more familiar matrix-and-vector style:

$$A = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad |\varphi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

and

$$|\varphi'\rangle = A|\varphi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}}(\alpha - \beta) \\ \frac{1}{\sqrt{2}}(\alpha + \beta) \end{pmatrix}.$$

It is easy to check that A is a unitary matrix.

11.2 Single-bit quantum gates

Next, we consider the problem of manipulating a single qubit. In particular, we are interested in whether any 1-qubit operation can be performed by a small set of quantum gates. In general, every 2×2 unitary matrix U is of the form

$$\begin{pmatrix} e^{i(\delta+\alpha/2+\beta/2)}\cos(\Theta/2) & e^{i(\delta+\alpha/2-\beta/2)}\sin(\Theta/2) \\ -e^{i(\delta-\alpha/2+\beta/2)}\sin(\Theta/2) & e^{i(\delta-\alpha/2-\beta/2)}\cos(\Theta/2) \end{pmatrix}$$

for some $\alpha, \beta, \delta, \Theta \in \mathbb{R}$ (recall that $e^{i\delta} = \cos\delta + i \sin\delta$). It is not difficult to show that this can be expressed as

$$\begin{pmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{pmatrix} \begin{pmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{pmatrix} \begin{pmatrix} \cos(\Theta/2) & \sin(\Theta/2) \\ -\sin(\Theta/2) & \cos(\Theta/2) \end{pmatrix} \begin{pmatrix} e^{i\beta/2} & 0 \\ 0 & e^{-i\beta/2} \end{pmatrix} .$$

The effect of the vector

$$R(\Theta) = \begin{pmatrix} \cos(\Theta/2) & \sin(\Theta/2) \\ -\sin(\Theta/2) & \cos(\Theta/2) \end{pmatrix}$$

is a Θ degree rotation of the state vector in the Hilbert space spanned by $|0\rangle$ and $|1\rangle$. Furthermore, let the phase vectors $P_0(\delta)$ and $P_1(\delta)$ be defined as

$$P_0(\delta) = \begin{pmatrix} e^{i\delta} & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad P_1(\delta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{pmatrix} .$$

$P_0(\delta)$ causes a δ degree rotation of the amplitude of $|0\rangle$ and $P_1(\delta)$ causes a δ degree rotation of the amplitude of $|1\rangle$. Looking at the decomposition of U into four matrices above, it can be seen that any unitary 2×2 matrix can be represented as the product of matrices of the form $R(\Theta)$, $P_0(\delta)$, and $P_1(\delta)$. Since it would be too difficult to construct a new device for R, P_0, or P_1 every time a new angle is needed, it would be highly desirable to be able to realize any rotation matrix by the use of a matrix with a fixed angle. The next proposition shows that this is possible.

Proposition 11.1 *There is an angle Θ_0 so that for any angle Θ and any $\epsilon > 0$ there is a $k = O(1/\epsilon^2)$ so that $|(k \cdot \Theta_0 \bmod 2\pi) - \Theta| \leq \epsilon$.*

Proof. Choose Θ_0 as

$$\Theta_0 = 2\pi \sum_{i \geq 0} \frac{1}{2^{2^i}} .$$

With this definition it holds for all $k \geq 0$ that multiplying Θ_0 with 2^{2^k} yields

$$(2^{2^k} \cdot \Theta_0) \bmod 2\pi = \left(2\pi \sum_{i \geq k+1} \frac{2^{2^k}}{2^{2^i}} \right) \bmod 2\pi$$

which is roughly $2\pi/2^{2^k}$. Hence, multiplying Θ_0 with $d \cdot 2^{2^k}$ for $d \in \{1, \ldots, 2^{2^k}\}$ yields angles $\Theta_1, \ldots, \Theta_{2^{2^k}}$ such that for any $\Theta \in [0, 2\pi]$ there is an i with $|\Theta - \Theta_i| \leq 4\pi/2^{2^k}$. Since $2^{2^{k+1}} = (2^{2^k})^2$, the proposition follows. $\square$

Hence, we obtain the following result.

Corollary 11.2 *Every unitary 2×2 matrix U can be realized up to an error of ϵ by $O(1/\epsilon^2)$ quantum operations of the form $R(\Theta_0)$, $P_0(\delta_0)$, and $P_1(\delta_0)$.*

11.3 Quantum registers

Quantum computations generally use more than just one qubit. We will show now how the mathematical formalism introduced in the previous subsection can adapted to the treatment of groups of qubits.

Definition 11.3 *A* quantum register *is an ordered set of a finite number of qubits. The* standard basis $\mathcal{B}$ *of an n-qubit quantum register is*

$$\mathcal{B} = \{|i\rangle : \ i \text{ is an } n\text{-bit binary string}\} .$$

Let $|\varphi_1\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ and $|\varphi_2\rangle = \beta_0|0\rangle + \beta_1|1\rangle$ be two qubits composing a 2-qubit quantum register. The state vector $|\psi\rangle$ of the register is defined as the *tensor product* of the states $|\varphi_1\rangle$ and $|\varphi_2\rangle$:

$$\begin{aligned} |\psi\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle &= \left(\sum_{i=0}^{1} \alpha_i |i\rangle\right) \otimes \left(\sum_{j=0}^{1} \beta_j |j\rangle\right) \\ &= \sum_{i,j=0}^{1} \alpha_i \beta_j (|i\rangle \otimes |j\rangle) . \end{aligned}$$

By definition, the tensor product maps $|i\rangle \otimes |j\rangle$ (where i and j are basis states) to $|ij\rangle$. This allows us to write $|\psi\rangle$ as

$$|\psi\rangle = \sum_{i,j=0}^{1} \alpha_i \beta_j |ij\rangle .$$

Similarly, let A and B be two unitary matrices corresponding to two apparatuses operating on $|\varphi_1\rangle$ and $|\varphi_2\rangle$ separately. The combined action of A and B on the joint state $|\psi\rangle = |\varphi_1\rangle \otimes |\varphi_2\rangle$ is defined as a 4×4 matrix C where

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{pmatrix} .$$

One can verify that the tensor product has the following property

$$(A \otimes B)(|\varphi_1\rangle \otimes |\varphi_2\rangle) = (A|\varphi_1\rangle) \otimes (B|\varphi_2\rangle)$$

and that it preserves unitarity.

So far, we seem to have only complicated the notation for a basically simple situation: two independent qubits are acted upon by two independent apparatuses. But the point in joining two qubits is specifically to allow them to be *dependent*. In fact, not all states of a 2-qubit quantum register can be expressed as the tensor product of single qubit states. An example of such a state is

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) .$$

If $|\psi\rangle$ is observed with the observable corresponding to the standard basis, the results "00" or "11" will be seen each with probability 1/2, but the results "01" or "10" will never be observed. When the state of an n-qubit register cannot be expressed as the tensor product of n qubit states, the register is said to be *entangled*. Similarly, not all 4×4 unitary matrices can be expressed as the tensor product of two 2×2 unitary matrices. One such matrix is

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

which represents the following mapping of the basis states of the register: if the register's state is such that the first qubit is 0, no action is performed; otherwise the value of the second qubit is negated. The above matrix performs the operation called *controlled not* on two quibits. This can be extended to arbitrary controlled unitary transformations U on a qubit by using the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{11} & u_{12} \\ 0 & 0 & u_{21} & u_{22} \end{pmatrix}$$

This matrix and the 2×2 matrix for manipulating a single qubit have been used as the basic building blocks to realize many complex unitary transformations including arbitrary transformations that can be performed by quantum Turing machines.

It is a simple matter to generalize what has been presented in this section to represent the state of an n-qubit register. The general state vector $|\psi\rangle$ of an n-qubit quantum register is

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

and the 2^n vectors $|i\rangle$ form the set of basis states of the register. This means that $|\psi\rangle$ is a vector in a 2^n-dimensional Hilbert space and operations are defined by $2^n \times 2^n$ unitary matrices. Observables for extracting information from the state vectors are partitions of the 2^n-dimensional Hilbert space. We are now ready to apply these notions of quantum mechanics to computation.

11.4 Computing with quantum registers

The original quantum computing model proposed by Deutsch was essentially a Turing machine, but with the added properties that tape cells and the head's state could be in quantum superposition. As might be imagined from the previous section, programming quantum Turing machines is quite difficult, especially because finding a suitable transition function corresponding to a unitary transformation is non-trivial. In classical complexity theory, uniform circuit families are also commonly used as a computing model. Turing machines and uniform circuit families are effectively equivalent in computing power in that they can simulate one another with negligible complexity overhead. This makes the use of one or the other a matter of taste. There exists a quantum equivalent to uniform circuit families: quantum gate arrays. They were introduced by Deutsch and studied extensively by many authors. Yao has shown that acyclic quantum gate arrays can simulate efficiently quantum Turing machines, thus making the use of one or the other a matter of choice. However, since quantum gate arrays allow a more natural way to introduce unitarity in computation, they are emerging as the standard quantum computing model.

11.4.1 Quantum gate arrays

Figure 1 represents a general quantum gate array. The initial (basis) state of the register is on the left and time flows from left to right. One might think of the particles composing the register as traveling through the different gates. At the right end is the observable that extracts information from the register after it has gone through all the gates.

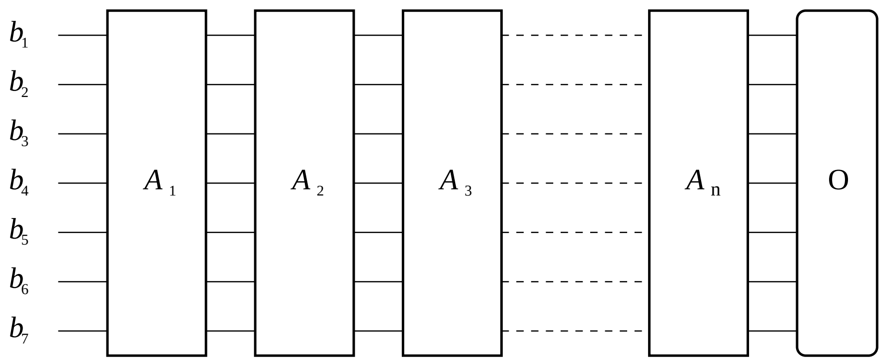

Figure 1: The layout of a general quantum gate array

The sequence of A_i's with observable $\mathcal{O}$ is what constitutes a quantum program. Formally speaking, the A_i gate should be of some well-defined form corresponding to some definition of elementary steps. For the purpose of this lecture, it is sufficient to consider any quantum gate acting on only one or two qubits to be such an elementary step. The reader is encouraged to consult [BBC+95] for more details on the notion of elementary quantum gates.

To illustrate the programming with quantum gate arrays, we will use a variant of the Deutsch-Jozsa Problem. First, we define two properties of functions from $\{0,1\}^n$ to $\{0,1\}$.

Definition 11.4 *A function $f : \{0,1\}^n \to \{0,1\}$ is called* non-balanced *if one of the two values of f has majority.*

Definition 11.5 *A function $f : \{0,1\}^n \to \{0,1\}$ is* non-constant *if there exist $x, y \in \{0,1\}^n$ such that $f(x) \neq f(y)$.*

Notice that most (but not all) functions from $\{0,1\}^n$ to $\{0,1\}$ have both properties simultaneously. The modified Detusch-Jozsa problem is described as follows:

Modified Deutsch-Jozsa Problem (MDJP):
Input: a computable function $f : \{0,1\}^n \to \{0,1\}$

Problem: to answer either "non-balanced" or "non-constant", but the answer must apply to f.

The original Deutsch-Jozsa problem dealt with strings rather than functions and was the first example of a problem which can be solved exponentially faster on a quantum computer than on a Turing machine. By recasting the original problem in the context of promise problems, Berthiaume and Brassard proved some early results in relativized quantum complexity theory. These results were improved upon first by Bernstein and Vazirani and then by Simon who proved the following theorem.

Theorem 11.6 *There exists an oracle relative to which there is a problem solvable in polynomial time (with bounded error probability) on a quantum computer, but any probabilistic Turing machine with bounded error probability claiming to solve this problem (using the oracle) will require exponential time on infinitely many inputs.*

Simon's theorem is the strongest argument in favor of the superiority of quantum computers over Turing machines. Moreover, the quantum gate array used in Simon's proof is similar to the one used by Shor for his factoring algorithm. In this subsection, we present a solution to the MDJP using quantum gate arrays. This will allow us later to outline Shor's factoring algorithm.

Consider now the MDJP. According to the Lecerf-Bennett theorem, for every Turing-computable function $f : \{0,1\}^n \to \{0,1\}$ there is a quantum gate array that performs the computation expressed in Figure 2.

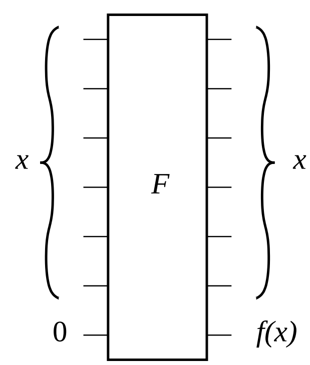

Figure 2: The gate for computing f.

Note that in the process of computing f it might be necessary to use additional qubits. However, at the end they can be reversibly set back to 0 so that we do not have to consider them here.

Suppose that the lowest qubit in the circuit is not in the basis state $|0\rangle$ but in some superposition $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ then, instead of $f(x)$, the lowest bit would be in the state

$$\alpha|f(x)\rangle + \beta|1 \oplus f(x)\rangle$$

where $\oplus$ is the XOR function.

Computing a function on an input is fine, but the rules of quantum mechanics allow much more. Recall that by linearity of quantum operations, if the input state is in quantum superposition $\frac{1}{\sqrt{2}}(|x,0\rangle + |y,0\rangle)$ then the F gate will compute the superposition of f on both values:

$$F\left(\frac{1}{\sqrt{2}}(|x,0\rangle + |y,0\rangle)\right) = \frac{1}{\sqrt{2}}(|x,f(x)\rangle + |y,f(y)\rangle) .$$

Assume there is a way to unitarily generate (through some matrix S_n) a superposition of all possible values of an n qubit register. That is, if the initial state of the register is all zero, then S_n transforms it in a superposition of all 2^n values of the first n qubits:

$$S_n|\underbrace{0\ldots0}_{n}\rangle = \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle .$$

We can see that by first applying S_n and then F, we can compute in one sweep all possible values for f in quantum superposition (see Figure 3).

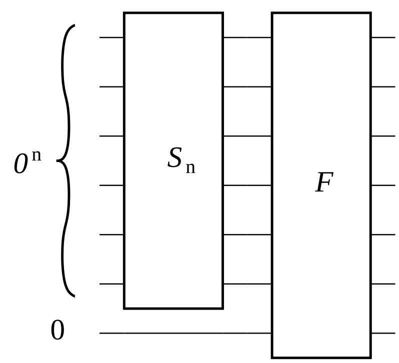

Figure 3: Computing f for all possible inputs.

$$F\,S_n|\underbrace{0\ldots0}_{n},0\rangle \rightarrow F\left(\frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i,0\rangle\right) \rightarrow \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i,f(i)\rangle .$$

The reader should take careful note of what is meant by the diagram shown in Figure 3. While the operator S_n acts on n qubits, its mathematical representation is a $2^n \times 2^n$ unitary matrix. Also, in the expressions following the diagram, it would be more accurate to use $S_n \otimes I$ (where I is the 2×2 identity matrix) since our gate array uses $n+1$ qubits.

We now show how to implement an S_n gate to achieve this form of quantum parallelism. Consider the unitary matrix

$$S_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} .$$

It is a simple matter to verify that S_1 is indeed unitary. Note also that $S_1^{-1} = S_1$. An S_1 gate is an elementary gate as it acts only on a single qubit: it transforms $|0\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and

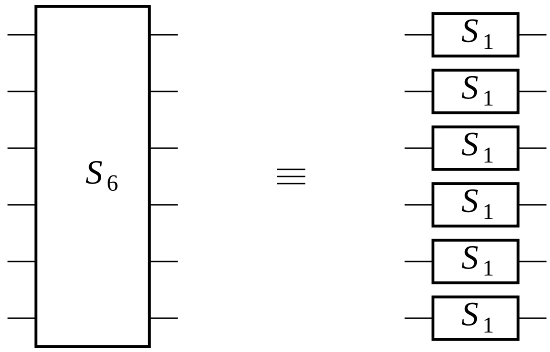

Figure 4: The quantum gate array for S_6.

$|1\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The desired S_n gate acts on a quantum register by sending each qubit individually into a separate S_1 gate (see Figure 4).

The unitary transformation induced by an S_n gate is given by the formula $S_n = \bigotimes_n S_1$. This has a nice recursive definition: if $n > 1$ then

$$S_n = \begin{pmatrix} S_{n-1} & S_{n-1} \\ S_{n-1} & S_{n-1} \end{pmatrix} .$$

Or in ket notation,

$$S_n|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{x \cdot i} |i\rangle$$

where the operation $x \cdot i$ can be seen here as the XOR of the bitweise AND of the strings x and i. Clearly, if x is set of 0^n, S_n performs the desired transformation.

With the conjunction of the S_n and F gates, a single computation produces all possible values of the function f for each input. But these values are in quantum superposition and we have seen that when observing it, only one output $|x, f(x)\rangle$ will be seen and the rest will be lost. To get some benefit from superpositions, a more subtle use of quantum parallelism is needed.

Consider the following unitary transformation:

$$P = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} .$$

If a qubit is set to 0 nothing happens. But if it is set to 1, then the amplitude is multiplied by -1. This gate "encodes" the value of the qubit into the sign of the amplitude. Now consider the following gate array:

(where the observable $\mathcal{D}$ will be defined shortly). From our definitions, we know that the state $|\varphi\rangle$ of the register just after the P gate is

$$|\varphi\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{f(i)} |i, f(i)\rangle .$$

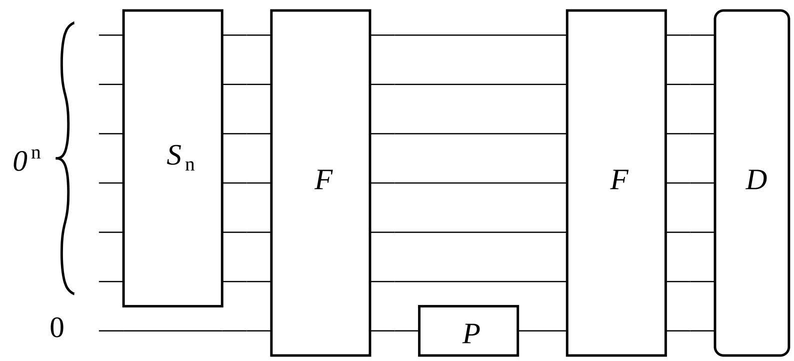

Figure 5: The quantum gate array for the MDJP.

When that state goes through the final F gate, the values for f are again computed and non-destructively combined using (in our case) the XOR function. Since $f(i) \oplus f(i) = 0$ for all $i \in \{0,1\}^n$, the final state before the observation is

$$|\varphi\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (-1)^{f(i)} |i, 0\rangle \ .$$

All the manipulations done so far had only one purpose: to transfer the values of f into the amplitudes relative to each of the basis states. The power of quantum computation resides in the interference of these amplitudes and the observable used to read the quantum states. We now define the observable. Consider $\mathcal{D} = \{E_a, E_b\}$, where the subspace E_a is the one-dimensional space spanned by

$$|\psi_a\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i, 0\rangle$$

and $E_b = (E_a)^{\perp}$, the orthogonal complement of E_a. Using $\mathcal{D}$ in the gate array above allows us to answer the MDJP, that is to determine without errors whether f is non-balanced or non-constant. To see this, recall that $\mathcal{D}$ will give the answer a or b with probabilities depending on the amplitudes of $|\varphi\rangle$ in the subspaces E_a and E_b. We must find the expression of $|\varphi\rangle$ in the basis defined by $\mathcal{D}$. This is easy since $\mathcal{D}$ has only two subspaces, one being one-dimensional. Let α and β be the projections of $|\varphi\rangle$ in E_a and E_b, then

$$|\varphi\rangle = \alpha|\psi_a\rangle + \beta|\psi_b\rangle$$

where $|\psi_b\rangle$ is a vector in E_b and, of course, $|\psi_a\rangle \perp |\psi_b\rangle$. Observing the final state $|\varphi\rangle$ with $\mathcal{D}$ will give the answer a or b with probability $||\alpha||^2$ and $||\beta||^2$ respectively. Since the observable has only two possible answers, $||\beta||^2 = 1 - ||\alpha||^2$. Also, finding the projection of $|\varphi\rangle$ in the one-dimensional subspace E_a is simple. We now compute the exact expression for α, the projection of $|\varphi\rangle$ into $|\psi\rangle$:

$$\alpha \;=\; \langle\psi_a|\varphi\rangle$$

$$= \left(\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \langle i, 0| \right) \left(\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{f(j)} |j, 0\rangle \right)$$
$$= \frac{1}{2^n} \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} (-1)^{f(j)} \langle i, 0|j, 0\rangle \ .$$

But since $\langle i, 0|j, 0\rangle = 1$ if and only if $i = j$ and zero otherwise, the expression for α simplifies to

$$\alpha = \frac{1}{2^n} \sum_{i=0}^{2^n-1} (-1)^{f(i)} \ .$$

We now look at the value of α for different functions f. If f is a balanced function, the sum for α will contain exactly as many 1's as -1's, so in this case $\alpha = 0$ and $\mathcal{D}$ will always give a b answer and never a. If f is a constant function, the value for α will either be 1 or -1, so in this case $\mathcal{D}$ always gives the answer a and never b. If f is of any other type, $\mathcal{D}$ will answer a or b with various probabilities.

To demonstrate that the above quantum gate array solves the MDJP, we need to take the above reasoning backwards. If the answer received from $\mathcal{D}$ is a, we know for certain that f could not have been a balanced function, so answering "non-balanced" is correct. Similarly, if $\mathcal{D}$ gives the answer b, then we know for certain that f could not have been a constant function, so answering "non-constant" is correct.

11.5 Shor's prime factorization algorithm

Every integer n has a unique decomposition into prime factors. However, finding this decomposition when n is large is a difficult computational problem. The best method known to date requires time $O(\mathrm{e}^{c(\log n)^{1/3}(\log\log n)^{2/3}})$ (see, for instance, Lenstra and Lenstra 93), which is exponential in the size (the number of digits) of n. Whether the factoring problem can be solved in polynomial time via classical computations is still unknown. Yet the faith in the hardness of this problem is such that the security of many classical cryptographic protocols is based on the impossibility of factoring efficiently.

Number theory offers another interesting problem: finding the order of an element. Given x and n, find an r (called the *order*) such that $x^r \equiv 1 \pmod n$. As with the factoring problem, no efficient algorithm is known for solving this problem. But while these problems appear very different, they are closely related. Miller has shown that, using randomization, one could solve the factoring problem *if* one had access to an oracle for finding the order of an element. His reduction works as follows: first, make sure that n is odd and not a prime (there are efficient primality testing algorithms). Then, use the following algorithm:

$x =$random($\{0, \ldots, n\}$)
$r =$order(x, n) // computes the order of x (mod n)
if r is odd or $x^{r/2} \equiv -1 \pmod n$ then *failure*
else return gcd($x^{r/2} - 1, n$)

Choosing a random number in the range $\{0, \ldots, n\}$, doing the modular exponentiation and finding the gcd (greatest common divisor) can all be done in polynomial time (see Knuth 81).

Let k be the number of odd prime factors of n. One can prove that, provided n is odd and non-prime, the above algorithm will return a prime factor of n with probability at least $1 - 1/2^{k-1}$. Repeating this algorithm a polynomial number of times will produce a complete factorization of n.

Shor's breakthrough was to discover an efficient quantum algorithm to find the order of an element. The factoring algorithm is simply Miller's reduction where the oracle is replaced by a call to this quantum algorithm. The next section gives a sketch of how to find the order of an element using quantum superpositions.

11.5.1 Finding the order of an element

We now describe Shor's algorithm to find the order r of an element $x \pmod{n}$. There are two distinct parts of the algorithm: the first is the quantum component, described next, which produces a value c. Thanks to appropriately chosen amplitudes, this c has a relationship to r such that a little (purely classical) post-processing in the second part can efficiently determine r. We describe the quantum component using quantum gate arrays. First, we need to find an m such that $n^2 \leq 2^m \leq 2n^2$. The gate array operates on a $2m$-qubit quantum register. Next, we need a gate such that on input $|a, 0\rangle$ it computes $|a, x^a \bmod n\rangle$. We know that modular exponentiation can be done classically in polynomial time. So by the Lecerf-Bennett theorem there exists a quantum gate E_n^x that efficiently implements this operation. This E_n^x gate is shown in Figure 6.

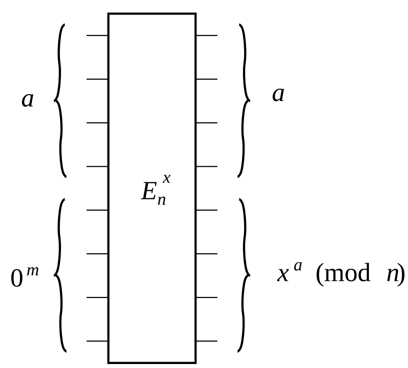

Figure 6: The gate E_n^x.

We only need one more quantum operation. Shor refined the S_n transformation used by Bernstein and Vazirani and Simon in the following way: instead of using phases that are $\pm 1/\sqrt{2^m}$, we now make use of the full spectrum of complex amplitudes. The transformation A_m sends a m qubit register in basis state $|a\rangle$ to

$$\frac{1}{\sqrt{2^m}} \sum_{c=0}^{2^m-1} e^{i(2\pi ac)/2^m} |c\rangle \ .$$

(Recall that for any $a + ib \in \mathbb{C}$ of norm 1, there exists an angle $\Theta \in [0, 2\pi]$ such that $a + ib = \cos\Theta + i\sin\Theta = e^{i\Theta}$.) This transformation is called the discrete quantum Fourier transform.

The fact that one can efficiently implement such a quantum gate is not immediately clear, if only for the fact that the amplitudes seem to require increasing precision as m grows large. However, Deutsch and Coppersmith independently found an efficient solution based on the Fast Fourier Transform algorithm (Knuth 81) which only requires $O(m^2)$ elementary quantum gates.

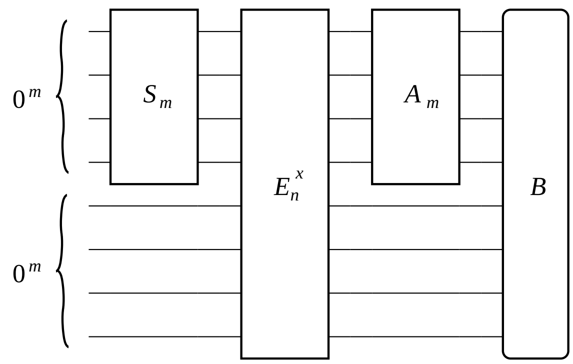

Figure 7: The gate array to find the order of an element.

The gate array for Shor's algorithm to find the order r of an element $x \pmod n$ is shown in Figure 7. S_n was defined in the previous subsection and only serves to generate a superposition of all possible values for the top half of the register. We then compute in quantum parallel the modular exponentiation of x for all these values and then apply the Fourier Transform A_m. The state of the register just prior to the observation is: (omitting the (mod n) in the ket for clarity)

$$\frac{1}{2^m} \sum_{a=0}^{2^m-1} \sum_{c=0}^{2^m-1} \mathrm{e}^{i(2\pi ac)/2^m} \left| c, x^a \right\rangle .$$

Since we are using the standard observable, the observation will yield any basis state $|c, x^k\rangle$ with probability

$$\left\| \frac{1}{2^m} \sum_{a:\, x^a \equiv x^k} \mathrm{e}^{i(2\pi ac)/2^m} \right\|^2 .$$

Shor proves that this probability vanishes everywhere except for basis states $|c, x^k\rangle$ such that there exists an integer d satisfying

$$\left| \frac{c}{2^m} - \frac{d}{r} \right| \leq \frac{1}{2^{m+1}}$$

where the probability is at least $1/(3r^2)$. This means that reading the final state of the register will yield with high probability a value c such that the fraction $c/2^m$ is close to d/r. Because $2^m > n^2$, there is only one fraction d/r that satisfies the above equality while keeping $r < n$. The algorithm for finding that fraction d/r from $c/2^m$ is the post-processing we referred to earlier and can be done efficiently by continued fraction expansion (see Knuth 81). This produces the r we needed.

For a more detailed study of Shor's algorithm including the necessary number theory which was left out here, see Shor 94. Shor's algorithm and Simon's theorem are two of the most important results in quantum complexity theory. Both are strong arguments in favor of the superiority of quantum computing models over classical ones. But a proof that quantum computing models are indeed stronger than classical ones seems still to be far away.

11.6 References

- A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolous, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phycial Review Letters A*, 1995.

- D. Deutsch and R. Jozsa. Rapid solutions of problems by quantum computation. In *Proc. of the Royal Society*, London, Vol. A439, pp. 553–558, 1992.

- D.E. Knuth. *The Art of Computer Programming*, Vol. 2, Addison Wesley, 1981.

- A.K. Lenstra and H.W. Lenstra. *The Development of the Number Field Sieve*. Springer Verlag, LNCS 1554, 1993.

- G.L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer Science* 13:300–317, 1976.

- P. Shor. Algorithms for quantum computing: discrete logarithms and factoring. In *Proc. of the 35th Symp. on Foundations of Computer Science*, pp. 124–134, 1994.

- D. Simon. On the power of quantum computation. In *Proc. of the 35th Symp. on Foundations of Computer Science*, pp. 116–123, 1994.

- A. Yao. Quantum circuit complexity. In *Proc. of the 34th Symp. on Foundations of Computer Science*, pp. 352–361, 1993.