Chapter 1

Quantum Computing

The fundamental goal of Science is to explain/predict natural phenomena using *simple* models. For example, a model explaining the behaviour of n charged particles should only involve $O(n)$ parameters, and not, say $\Omega(2^n)$. This requirement seems so natural and obvious that it is hardly discussed.

On the other hand, the behaviour of nature at microscopic levels seems to defy this modest expectation. Indeed, the main theory found useful at this level, Quantum Mechanics, seems to require $\Omega(2^n)$ parameters/variables to model the behaviour of an n particle system.[1] If you take such a system and perturb it a bit (e.g. by shining a laser on it), then turns out that typically all the $\Omega(2^n)$ parameters change. Note that the problem is not that Quantum Mechanics doesnt explain nature – that it does very well; only that it seems to need models with far too many parameters.[2] The first practical effect (rather than philosophical) of the high model complexity is that Quantum Mechanical systems are extremely expensive to simulate on a computer. Essentially exponential time is required to simulate a single step.

Quantum Computing aims turns this problem into an advantage. Suppose, as we said, when a laser is shone on a system of n particles it causes an exponential number of variables describing the system to get modified. *What if the modifications are similar to those needed in a certain computation A?* If so, we build the Quantum Mechanical system, shine lasers, measure the final state, and we would get the solution to our computation A. We would be using $O(n)$ hardware, but would get an exponential number of computations performed – exponential speedup – as great a bargain as any!

This is the big hope – by spending only on $O(n)$ hardware we could perhaps get $\Omega(2^n)$ speedup. There are however, many, many problems. The first is: sure the Quantum Mechanical system will effectively do many computations in a single step, but how many of those are actually useful? This problem also arises in parallel computing – but in a Quantum Computer it is much more restrictive. Nevertheless, people have been able to design some remarkable algorithms that can exploit the available parallelism. Most notable amongst these is the (linear hardware, polynomial time) algorithm for factoring integers. A second problem is that quantum computers appear difficult to build because quantum systems are very unstable. There is some progress on this too – recently a 3-4 bit quantum computer was configured and made to factor the number 15. This is a very tiny beginning, but nevertheless the research community (and others) are excited because the stakes are very high.

In this chapter we will study the mathematical model proposed for parallel computers. We will then study an algorithm for computing Fourier transforms. This algorithm is rather elegant and does exhibit exponential speedup. It is also important because it is at the heart of Shor's factoring algorithm – however we will not get to the factoring algorithm itself.

1.1 What is a Computational Model

Most people who learn to drive a car do not know how a four stroke engine works. Typically, it suffices to only have a vague idea that if petrol burns in a chamber it causes the gas inside it to expand, and this is in turn causes a piston to be pushed. There are of course many additional details needed to build a car (e.g. how to force the piston

[1] In contrast, the field due to a system of n charges can be completely described using an expression which contains $O(n)$ parameters giving the positions and magnitudes of the charges.

[2] This aspect has puzzled eminent scientists all the way from Einstein. It has caused Richard Feynman (known for his seminal contribution to Quantum Mechanics) to say, "I think I can safely say that nobody today understands quantum mechanics".

to move the wheels, or how to keep it moving periodically). But most people learn how to drive very well without knowing much of all this.

The description of quantum computing in these lectures is of the above kind. The goal is to provide a good mathematical description of the core computational model in quantum computing – which is what is needed to design algorithms. How the model is realized, i.e. what constitutes the hardware – is described very vaguely and superficially. The physical realization requires substantial understanding of physics – much more than can be given in 2-3 lectures. The core computational model, however, can be appreciated within the short time we have.

A computational model must describe how the computer inputs information from the user/environment, how the information is represented inside the computer, what operations can be used to alter (process) it, and how the answer is output to the user.

1.2 How information is represented

In classical computing, information is represented as collections of bits. Each bit can be either 0 or 1. A bit is realized physically as presence or abscence of charge on (current through) a capacitor (transistor).

In quantum computing, information is represented using collections of *qubits*. A single qubit has 2 base states, typically denoted 0,1. A qubit might be realized using an ion (or atom) which can either be un-excited (base state 0) or excited (base state 1). A qubit may be in the 0 state or in the one state, or it can be in a *superposition* of the 0 and one states. This superposition is a defining feature of Quantum systems. The state of a qubit is then described using a vector V of length 2, where $V[0]$ denotes the *amplitude* for the qubit to be in base state 0 (the ion to be un-excited), $V[1]$ the amplitude for being in base state 1 (ion being excited). $V[0], V[1]$ are complex numbers, but it is required that $|V|_2 = 1$, i.e. $|V[0]|^2 + |V[1]|^2 = 1$. With this constraint, it can informally be considered that $|V[0]|^2, |V[1]|^2$ denote the degree to which the qubit is in states 0 and 1 respectively.

It is natural, but incorrect, to associate amplitudes with any kind of uncertainty in our knowledge of the ion state: It is a fundamental property of quantum systems that they can exist in this "superposition" of base states, and the superposition given by complex valued amplitudes. It is also natural to expect that if amplitudes exist in nature that we should have some ways of measuring them. Even on this count, reality is stranger than our intuition developed out of macroscopic phenomenon. We will consider this in more detail in Section 1.5.

Now consider a collection of n qubits. If these qubits belong to independent non-interacting physical systems, then it is enough to describe them by their individual vectors. Qubits can however be made to interact – for example if the ions realizing the qubits are in close physical proximity, then they may get into a coupled (mechanical) oscillation. Many other ways of getting qubits to interact are also being considered by researchers. If qubits have interacted with each other, we need to describe them together, using a single global state vector. Each single qubit can have base-states 0 and 1; the collection's base states will be all possible n bit strings. Each (collective) base state will have an amplitude, and hence the collective state will consist of all these 2^n amplitudes. Specifically, the state is a vector of 2^n complex numbers $V[0..2^n - 1]$. Consider any pattern of n bits, say $b_{n-1}, b_{n-2}, ..b_0$. Let $b = b_{n-1}b_{n-2}..b_0$ denote the integer formed by concatenating these bits. Then $V[b]$ denotes the amplitude of the respective ions being in base-states $b_{n-1}, b_{n-2}, \ldots, b_0$. As before we require $|V|_2 = 1$.

If the exponential number of amplitudes required to describe a the state of a collection of qubits makes you intuitively uneasy – note that this is expected!

1.3 Input

It is possible to initialize the state vector of a collection of qubits to any base state.

Note that a base state is described by a vector consisting of a single entry with value 1 and all other entries 0. We will use $|i\rangle_n$ to denote a vector of length 2^n having 0s everywhere except in position i, which is 1. We will drop the subscript when it is obvious from the context.[3]

1.4 Computation

A computation takes the state vector of a collection of qubits, and transforms it. Physically realizable computations are unitary, i.e. for n qubits they can be represented as a $2^n \times 2^n$ matrix Q. If V is the state vector of the n qubit

[3]This notation is due to Dirac, and $|i\rangle$ is pronounced "ket". The transpose of this vector is written as $\langle i|$ and is referred to as "bra".

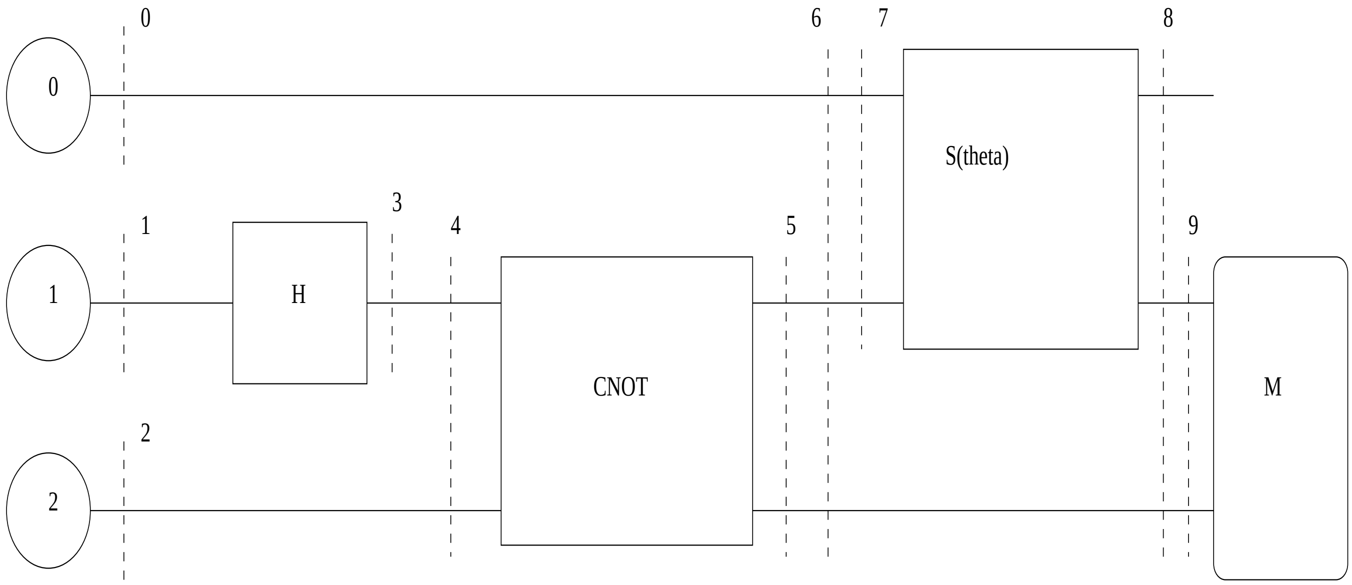

Figure 1.1: An example quantum computer.

collection before the operation, then after the operation the state vector becomes QV. It can be proved out that any unitary matrix Q can be realized (to sufficient accuracy) by performing a series of suitable physical operations on the physical realizations of the qubits. For example, shining light or applying magnetic fields and so on on the ions realizing the qubits.

While the operation corresponding to any unitary Q can be realized, the time required (or the number of laser pulses required) can be exponential in the number n of qubits. Thus, for the purposes of defining a mathematical model, it may be assumed that in unit time we can apply an operator to a fixed size collection of bits. Even though your system may consist of n qubits, your operations should only involve only some fixed value k of these at every step (typically $k \leq 3$). The operation will consist of applying any $2^k \times 2^k$ unitary matrix of your choosing to the state vector of the chosen k bits. It is reasonable to assume that such operations can be performed in constant time. A quantum algorithm then, consists of a sequence of T operations, in the tth of which some (predecided) fixed size subset of the n qubits are selected and operated upon using a certain (predecided) operator Q_t.

It is customary to view quantum computation using a left to right data flow, as shown in Figure 1.1. Each line (often called a wire because of the circuit analogy) represents a qubit at a certain time instant, with time increasing to the right. The boxes represent operations. The wires coming out represent the result of the operation. The subsequent operations operate on other sets of qubits. If an operation takes as inputs non-consecutive qubits, then it is customary to draw a box only on the consecutive qubits and indicate the rest by drawing a line from the box to the other qubits involved, and marking the involved qubits, see Figure 1.2 for an example of this. The operations are often called gates, again because of a circuit analogy. For constructing the state vectors, it is customary to number the qubits top to bottom, 0 to $n-1$.

We next show how this computer executes. For this we will need to discuss the state vectors of various groups of qubits – we have grouped these collections by dashed lines in Figure 1.1 and given a label L to each line. We will use ψ_L to denote the state vector for the qubit group L. Within each group, the qubits will be considered numbered top to bottom, starting at 0.

The execution of the computer begins by initializing the values of the qubits. As mentioned earlier, these can only be base states. This assignment will fix the state vectors for each qubit. Say we fix $\psi_0 = |1\rangle$, $\psi_1 = |0\rangle$, $\psi_2 = |1\rangle$. Following this, we must analyze the effect of each operator.

Analyzing the effect of the H operator is easy – since we know the state vector ψ_1 for the single qubit input to it. The H operator is an important single qubit operator and is called the Hadamard operator, whose matrix is:

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

From this we can compute ψ_3 as:

$$\psi_3 = H\psi_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Analyzing the effect of the CNOT operator is not straightforward. As we have described it, to compute the effects of an operator acting on a set S of qubits, we need to have the collective state vector of just the S bits. In this case we would need to have the state vector ψ_4 for qubits 1 and 2 at this step. This is obtained by a process called *entanglement* which we describe next. Given ψ_4 and the matrix for CNOT we can compute ψ_5. For computing the effect of S(theta), we first construct ψ_6 by entangling ψ_5 and ψ_0. For evaluating the effect of S(theta) we seem to need to compute ψ_7. You might think of this as *disentangling* qubit 2 from ψ_6 – such disentanglement is not possible in general. However by using properties of *superposition* which we discuss shortly, we will be able to compute ψ_8.

In fact using *entanglement* and *superposition* you should be able to analyze the action of any quantum computer on any input setting.

1.4.1 Entanglement and Disentanglement

Suppose a set P of p qubits and a set R of r qubits have so far evolved independently. Let T denote their union, with the i the bit of R being numbered i in T, while the ith bit of P numbered $r+i$ in T. If V_P, V_R, V_T denote the corresponding $V_T[i2^r + j] = V_P[i] * V_R[j]$. This rule is analogous to the rules for independent events in probability theory: if the elements of P are probabilities of a set of events which are independent a set of events whose probabilities are given by the vector R, then the probability that the ith event from P and the jth event from R happen together is $V_P[i]V_R[j]$. The process of forming V_T from V_P, V_R is called entanglement, and will be denoted using the symbol $\otimes$, i.e. $V_T = V_P \otimes V_R$. This operation is called the *tensor* product between the vectors.

Continuing our example, we have:

$$\psi_4 = \psi_2 \otimes \psi_3 == \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

The reverse operation (could be called disentanglement): getting back the the state vectors for P, R given the state vector for T cannot be performed in general. For example, consider the state vector $\frac{1}{\sqrt{2}}(|0\rangle + |3\rangle)$ which is only slightly different from ψ_4 above. You should be able to pursuade yourself that this cannot be written as a tensor product of two vectors.

However, if T is in a base state, then it can be disentangled. Suppose for example, that $V_T = |i2^r + j\rangle_{p+r}$ for certain i, j. Then it is easily checked that this is the entanglement of $V_P = |i\rangle_p$ and $V_R = |j\rangle_r$. In such a situation, if we have an operator Q that operates on set R, then its effect is easily modelled. We disentangle the given V_T, apply Q to the resulting V_R, and then entangle again. Specifically, the new state vector for T will be:

$$|i\rangle_p \otimes Q|j\rangle_r \tag{1.1}$$

To extend this to the general situation, we need the principle of superposition.

1.4.2 Superposition

The basic principle is that quantum operators are linear. This is not a surprise since we defined their effect using matrices; indeed we could define the effect as matrix multiplication simply because the operators are linear.

To explain this idea further, suppose U, V are valid state vectors for a collection of qubits. Then it can be seen that $\alpha U + \beta V$ is a valid state vector provided $|\alpha|^2 + |\beta|^2 = 1$. Suppose further that this system can be operated upon by an operator Q. Then the result for the superposed vector $\alpha U + \beta V$ is $Q(\alpha U + \beta V) = Q\alpha U + Q\beta V$.

So we begin by thinking of V_T as a superposition:

$$V_T = \sum_{k=0}^{2^{p+r}-1} V_T[k]|k\rangle_{p+r}$$

The result of applying Q on T is the same as applying Q on each $V_T[k]|k\rangle_{p+r}$ and then adding up the results. But $V_T[k]|k\rangle_{p+r}$ can be disentangled, and so it is easy to apply Q on it. Writing $k = i2^r + j$, and using equation 1.1

we see that the result of applying Q on $V_T[i2^r + j]|k\rangle_{p+r}$ is simply $V_T[i2^r + j]|i\rangle_p \otimes (Q|j\rangle_r)$. Thus, the new state vector is simply:

$$\sum_{i=0}^{2^p-1}\sum_{j=0}^{2^r-1} V_T[i2^r + j]|i\rangle_p \otimes Q|j\rangle_r$$

Noting that a matrix vector multiplication Ax can be written as $\sum_j A^j x_j$ where A^j is the jth column of A and x_j the jth element of x, the above is a matrix multiplication $Q'V_T$, where the kth column of Q' is $|i2^r + j\rangle_p \otimes Q|j\rangle_r$. It is left to you to ascertain that Q' is simply the block diagonal matrix consisting of 2^p copies of Q along the diagonal.

Notice the parallelism implicit here: although we just had one operator, we are getting 2^p matrix multiplications! This will be used with stunning effect in the FFT algorithm.

We now complete our example. Noting that the matrix for CNOT is:

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

It will be seen that multiplying this by ψ_4 leaves it unchanged, thus

$$\psi_5 = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

By entanglement,

$$\psi_6 = \psi_5 \otimes \psi_0 = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Note first that the matrix for S(θ) is defined only for $|\theta| = 1$ and is given by:

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

Now we can compute ψ_8 by multiplying a matrix with two copies of $S(\theta)$ along the diagonals. Alternatively observing that $\psi_6 = \frac{1}{\sqrt{2}}(|6\rangle + |7\rangle)$, we get

$$\psi_8 = \psi_6[1 \cdot 2^2 + 2]|1\rangle \otimes S(\theta)|2\rangle + \psi_6[1 \cdot 2^2 + 3]|1\rangle S(\theta)|3\rangle = \frac{1}{\sqrt{2}}|1\rangle \otimes S(\theta)^2 + \frac{1}{\sqrt{2}}|1\rangle \otimes S(\theta)^3 = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \theta \end{pmatrix}$$

Note that in above $S(\theta)^2, S(\theta)^3$ denote the second and third column of $S(\theta)$.

1.4.3 Hypercubic view

It is useful to consider the coefficients of the state vector to be residing on a binary hypercube. Specifically, imagine that $V[k]$ resides on node k of the $p+r$ dimensional hypercube. In this the ith hypercube dimension corresponds to the ith qubit in the collection. Now, the matrix multiplication $Q'V_T$ can be thought of as follows:

1. Only retain the edges along dimensions $0 \ldots r-1$, i.e. dimensions corresponding to the qubits on which the operator Q is to be applied.

2. At this point we have a collection of 2^p hypercubes each of r dimensions. Each has 2^r nodes. Consider each such hypercube as representing a state vector v, and within each such hypercube perform the multiplication Qv. Replace the values at the nodes by the new values.

The utility of this visualization is more apparent when the subset of qubits on which the operator operates is not contiguous. Even in that case we can think of the operation as involving matrix multiplication on smaller hypercubes (which are obtained by only retaining edges along dimensions corresponding to the numbers of the qubits on which the operation is being performed).

1.5 Output: Measurement

The state vector description of a qubit can thought of as a description of its "private" world. To see what the state is, you need to make a "measurement". A measurement is again done by shining a laser beam (or something else depending upon the physical realization). The notion of measurement in Quantum Mechanics is different from the standard notion. The measurement of a quantum state-vector V has the following effects:

1. A base state is revealed as the outcome of the measurement. The revelation is probabilistic: a base state $|i\rangle$ is revealed with probability $|V[i]|^2$.

2. The state vector gets set to the measured state. This is called a "collapse of the state vector".

Note that there is, unfortunately, no direct way to measure $V[i]$ themselves. If during a measurement the state vector collapses to $|i\rangle$ you may conclude that $|V[i]|$ must have had a large value. Of course state it is possible that $|i\rangle$ is the outcome of a measurement even if $|V[i]|$ is small, but with low probability.

As a metaphor you may think that a quantum system is like the human mind. It can have several thoughts at the same time (superposition), and if someone asks "what are you thinking", the daydreaming stops and one of the thoughts is revealed. Furthermore, after the question, the mind may forget the other thoughts and work more concentratedly on the one thought that was revealed.

1.5.1 Partial Measurement

We may also measure subsets of the n-qubits. The semantics of this is as you might expect: the probability of observing a base state s for the chosen qubits is the same as the probability of seeing that state had all the n qubits been measured. After the measurement, the measured qubits and the remaining qubits are disentangled. The measured qubits are in the measured ground state. The remaining qubits are in the superposition consistent with the measured state. More formally, suppose we measure the least significant r qubits of a state vector V for an $p+r$ qubit system. Then

$$Pr[|j\rangle_r] = \sum_i |V[i2^r + j]|^2$$

Given that $|j\rangle_r$ is measured, then if V' denotes the new state vector for the most significant p bits we have:

$$V'[i] = \frac{V[i2^r + j]}{\sum_i |V[i2^r + j]|^2}$$

1.6 Fourier Transform

The Fourier transform of a sequence of complex numbers $a = (a_0, a_1, \ldots, a_{N-1})$ is the sequence $A = (A_0, A_1, \ldots, A_{N-1})$ defined as follows. Let $P_a(x)$ denote the polynomial $\sum_i a_i x^i$. Then $A_i = P_a(\omega_N^i)$, where ω_N is a principal Nth root of 1. The standard algorithm is:

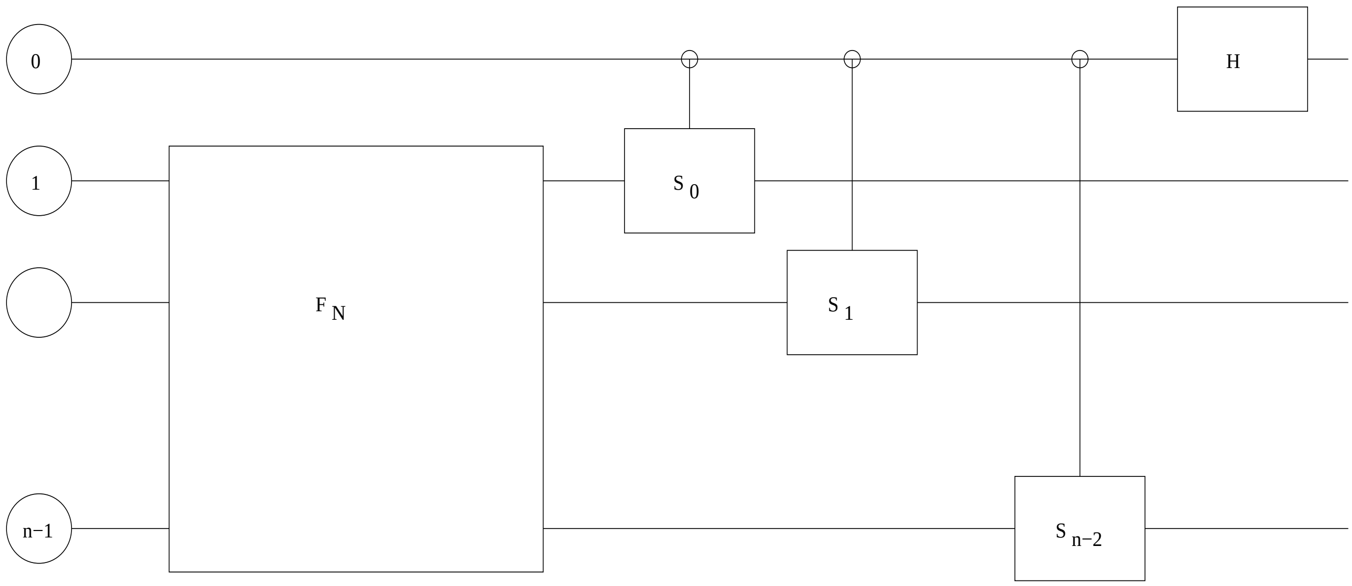

Figure 1.2: Quantum Fourier Transform Algorithm

1. If a has length 1, then we directly return the result. Else, we first partition a into its even and odd subsequences b and c (with $b_i = a_{2i}$ and $c_i = a_{2i+1}$) respectively, and define the corresponding $N/2$ degree polynomials $P_b(z) = \sum_{i=0}^{N/2-1} a_{2i}z^i$ and $P_c(z) = \sum_{i=0}^{N/2-1} a_{2i+1}z^i$.
2. We then recursively compute the Fourier transforms $B = (B_0, B_1, \ldots, B_{n/2-1})$ and $C = (C_0, C_1, \ldots, C_{N/2-1})$.
3. For $0 \le i < N/2$ we set $A_i = B_i + \omega_N^i C_i$ and $A_{i+N/2} = B_i - \omega_N^i C_i$.

Figure 1.2 basically implements the above algorithm. We use an $n = \log N$ qubit system. The state vector V of this system is assumed to be initialized to the input, i.e. $V[i] = a_i$. Initializing a set of qubits in this manner is hard for arbitrary a_i, but we will ignore this issue. After this each of the above steps will be implemented as a suitable matrix operator.

For this it is useful to note the following alternate expression of the Fourier transform:

$$\begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{N-1} \end{pmatrix} = F_N \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{pmatrix}$$

where F_N is a $N \times N$ matrix with $F_{jk} = \omega_N^{ij}$. This matrix is not unitary; however it is possible to show that the matrix $\frac{1}{\sqrt{N}} F_N$ is. In fact in many definitions, the scaling factor is included. From now on, we will also assume this.

Note that step 2 operates on the even indices and odd indices separately. In each the same operation (multiplication by matrix $F_{N/2}$) is performed. Thus this can be done by a single operation on the subcube induced by the most significant $n-1$ bits. This explains the first box in Figure 1.2. Notice that we dont need to perform the recursion twice – the inputs to the two recursions are conveniently superposed and so it suffices to perform the operation once on the superposition. The output produced from this will indeed be a superposition of the B, C vectors: the even nodes in the hypercube view holding B and the odd nodes C. In particular, hypercube vertex $2i$ holds B_i and $2i+1$ holds C_i.

The set of gates marked $S_0, \ldots, S_{n-1}$ together accomplish the task of computing $C_i\omega^i$, for each i, while leaving the B values alone. Each S_j gate is a $\mathrm{S}(\omega^{2^j})$ gate. Gate S_{j-1} operates on qubits j and 0. Thus we must consider 4 node subcubes induced by edges along these dimensions. Such subcubes will in general consist of nodes $x_{n-1} \ldots x_0$, for all the possible values for bits x_j and x_0, in particular

$$x_{n-1} \ldots x_{j+1} 0 x_{j-1} \ldots x_1 0, \quad x_{n-1} \ldots x_{j+1} 0 x_{j-1} \ldots x_1 1, \quad x_{n-1} \ldots x_{j+1} 1 x_{j-1} \ldots x_1 0, \quad x_{n-1} \ldots x_{j+1} 1 x_{j-1} \ldots x_1 1$$

These nodes will respectively store:

$$B_{x_{n-1}\dots x_{j+1}0x_{j-1}\dots x_1}, \quad C_{x_{n-1}\dots x_{j+1}0x_{j-1}\dots x_1}, \quad B_{x_{n-1}\dots x_{j+1}1x_{j-1}\dots x_1}, \quad C_{x_{n-1}\dots x_{j+1}1x_{j-1}\dots x_1}$$

From the matrix of $S(\theta)$ discussed earlier, only the last of the values will be multiplied by $\omega^{2^{j-1}}$, others will not change. Thus only those $C[x]$ having 1 in the $j-1$th bit (note that the lsb in the hypercube position only determined whether the node stores a B value or a C value) get multiplied by $\omega^{2^{j-1}}$. But this will happen for all j and all these multiplications will result in a total multiplication of exactly ω^x as needed. So after this, in hypercube node $2x+1$ we have $C_x\omega^x$ as needed. The hypercube node $2x$ will continue to hold B_x throughout.

It will be seen that the final step of adding in the even hypercube and subtracting in the even hypercube is precisely what the Hadamard gate does.

1.7 Reading material

There is excellent material on Quantum Computing on the web. Wikipedia has a good survey, as does `plato.stanford.edu/entr` Notes are also available for Quantum Computing courses, e.g. by Umesh Vazirani at Berkeley and John Preskill at Caltech.

The textbook "Algorithms" by Dasgupta, Papadimitriou and Vazirani[1] has a chapter on Quantum Computing.

Exercises

1. Argue by contradiction that the state $\frac{1}{sqrt2}(|0\rangle_2 + |3\rangle_2)$ cannot be disentangled.

Bibliography

[1] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms.* Tata McGraw-Hill Publishing Co., 2006.