5

Introduction to quantum computing

5.1 General remarks

It is easy to represent integers in terms of qubits in the same manner as for ordinary bits. Let us suppose that we wish to write an integer between 0 and 7 in a register of qubits. If this were a classical register, we would need 3 bits. In a system of base 2, a number between 0 and 7 can be represented in binary notation as a sequence of three digits 0 or 1. A classical register will store *one* of the 8 following configurations:

$$0 = \{000\}, \qquad 1 = \{001\}, \qquad 2 = \{010\}, \qquad 3 = \{011\},$$
$$4 = \{100\}, \qquad 5 = \{101\}, \qquad 6 = \{110\}, \qquad 7 = \{111\}.$$

A system of three qubits will also allow a number from 0 to 7 to be stored, for example, by making these numbers correspond to the following 8 states of three qubits:

$$\begin{array}{llll} 0 : |000\rangle, & 1 : |001\rangle, & 2 : |010\rangle, & 3 : |011\rangle, \\ 4 : |100\rangle, & 5 : |101\rangle, & 6 : |110\rangle, & 7 : |111\rangle. \end{array} \tag{5.1}$$

Here we have omitted the tensor product notation; for example, $|101\rangle$ is abbreviated notation for $|1_A \otimes 0_B \otimes 1_C\rangle$, where the qubits A, B, and C have their state vector in $\mathcal{H}_A$, $\mathcal{H}_B$, and $\mathcal{H}_C$, respectively. We use $|x\rangle$, $x = 0, \ldots, 7$, to denote one of the eight states of (5.1), for example, $|5\rangle = |101\rangle$. It is not difficult to generalize to the case of n qubits; representing a number less than $N = 2^n$ requires n qubits, and $|x\rangle$ denotes the state vector with

$$0 \le x \le 2^n - 1.$$

The basis of the Hilbert space $\mathcal{H}^{\otimes n}$ formed using orthonormal vectors $|x\rangle$ is called the *computational basis*. Since we can construct a linear superposition of the eight states (5.1), it can be concluded that the state vector of a system of three spins allows us to store $2^3 = 8$ numbers at the same time, while if n spins are used we can store 2^n numbers! However, if, for example, spins 1/2 are used for the physical support of the qubits, a measurement of the three spins along the axis Oz will necessarily give one of the eight states (5.1). We have at our disposal important virtual information, but when we try to materialize it in a measurement we can do no better than for a classical system: the measurement gives one of eight numbers, and not all eight at the same time! It is therefore necessary to go further in order truly to exploit the possibilities of a quantum computer and find algorithms which are specific to it. This will be explained later on in this chapter, and for now we shall give only a schematic description of the principle by which such a quantum computer functions.

A calculation performed on a quantum computer is shown schematically in Fig. 5.1, where n qubits are all prepared in the state $|0\rangle$ at time $t = t_0$. This is the preparation stage of the quantum system, and the initial state vector belongs to a Hilbert space of 2^n dimensions, $\mathcal{H}^{\otimes n}$. This initialization stage is not a unitary operation, but a projective measurement, and it is a dissipative process. The qubits then undergo a unitary quantum evolution described by a unitary operator $U(t, t_0)$ acting in $\mathcal{H}^{\otimes n}$ which performs the desired operations, for example, the calculation of a function. The experimental difficulty is to avoid any interaction with the environment, because then the phenomenon of decoherence would make the evolution nonunitary. As we have seen in Section 4.4, if there is an interaction with the environment, the unitary evolution occurs in a Hilbert space which is larger than $\mathcal{H}^{\otimes n}$, because it includes the degrees of freedom of the environment along with those of the qubits. Interactions with external classical fields are compatible

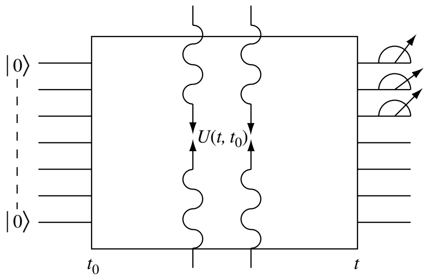


Figure 5.1 Schematic depiction of the basic principle of a quantum calculation. n qubits are prepared in the state $|0\rangle$. They undergo a unitary and deterministic evolution in the space $\mathcal{H}^{\otimes n}$ from time $t = t_0$ to time t described by a unitary operator $U(t, t_0)$ acting in $\mathcal{H}^{\otimes n}$. The wiggly arrows represent interactions with external classical fields. A measurement of the qubits (or a subset of the qubits, the first three in this figure) is made at time t. The diagrams are read from left to right, in the direction opposite to that of the operator products.

with unitary evolution and they are needed to manipulate qubits by Rabi oscillations.[1] Once the quantum evolution has been completed, a measurement is made on the qubits (or on a subset of qubits) at time t in order to obtain the result of the calculation. An important point is that *the state of the calculation cannot be observed* between t_0 and t, because any measurement would modify the unitary evolution: the box $U(t, t_0)$ of Fig. 5.1 is a black box which must not be tampered with. The qubits are measured at the entrance and at the exit of the box, but not inside it. Another essential point is that the unitary evolution is *reversible*:[2] if we know the state vector at time t, we can recover the state vector at time t_0 using $U^{-1}(t, t_0) = U(t_0, t)$.

5.2 Reversible calculation

The passage of the initial qubit state at $t = t_0$ to the final qubit state at t occurs via a reversible operation, and the algorithms of a quantum computer must necessarily be reversible. This is not the case with the algorithms used on classical computers, which are irreversible, and so the latter cannot be transposed directly to quantum computers. Most of the usual logic gates are irreversible, because they correspond to a transformation (2 bits $\to$ 1 bit), and the final state of a single bit does not allow the reconstruction of the initial two-bit state. For example, the NAND gate

$$x \uparrow y = 1 \oplus xy,$$

where $\oplus$ is mod 2 addition, gives the correspondence

$$(00) \to 1, \qquad (01) \to 1, \qquad (10) \to 1, \qquad (11) \to 0,$$

and knowledge of the final state does not permit reconstruction of the initial state. It is known that the NAND gate and the COPY operation are sufficient for constructing any logic circuit. An interesting question is whether or not all the usual logic operations can be performed reversibly on a classical computer.

[1] A note for physicists: the reason why the action of a classical field is compatible with unitary evolution is subtle, see Leggett (2002). Let us consider the states $|0\rangle$ and $|1\rangle$ of a spin 1/2 in a uniform magnetic field parallel to Oz. If the spin is initially in the excited state $|1\rangle$, then application of a π-pulse will send it in the ground state $|0\rangle$, and the external magnetic field will gain one photon. So, it appears that the transition has left a mark in the environment, a feature which should lead to decoherence. However, the (quantum) state of the field is a coherent state containing a very large number of photons and this state changes in a negligible manner when one adds one photon: the spin does not become entangled with the (quantum) state of the electromagnetic field. Of course, the energy difference between the two levels must be small enough, so that spontaneous emission is negligible. Otherwise the spin would also interact with the vacuum fluctuations of the *quantized* electromagnetic field, and its evolution would be no longer be unitary, see Exercise 4.6.7. All this has been confirmed in a beautiful experiment using neutron interferometry by Badurek *et al.* (1985); this experiment is thoroughly analyzed in Omnès (1994), Chapter 11.

[2] A second note for physicists: reversible evolution and invariance under time reversal should not be confused, as time reversal is represented in $\mathcal{H}$ by an anti-unitary operator, whereas $U^{-1}(t, t_0) = U(t_0, t)$ is unitary.

This question was initially only of theoretical interest, and was first raised by Landauer and Bennett, who wondered if it were possible to perform a calculation without energy dissipation. In fact, in spite of its abstract nature, information is necessarily carried by some physical support.[3] As a bonus, in following this line of inquiry Bennett was finally (after more than a century!) able to obtain a satisfactory solution of the paradox of the Maxwell demon (see Box 5.1). According to Landauer, a calculation involving irreversible operations like the loss of a bit of information in a NAND operation costs a thermodynamical entropy of at least $k_B \ln 2$ per bit, where k_B is the Boltzmann constant ($k_B = 1.38 \times 10^{-23}$ J/K), and therefore leads to the dissipation of an energy $\Delta E = k_B T \ln 2$ into the environment, where T is the absolute temperature of the computer. At present the problem is academic, because for an actual PC the energy dissipated per erased bit is already $\Delta E \sim 500 k_B T$ simply owing to electricity consumption, and so we are nowhere near $k_B T$. However, it is possible that this question will become of practical import some time in the future. The energy dissipated per logical operation has decreased by ten orders of magnitude in the past 50 years, so that it might be that the $k_B T$ limit becomes relevant in a little more than 10 or 15 years.

The real reason for the interest in reversible calculation is the possibility of transposing classical algorithms to quantum computing. As we have already mentioned, direct transposition is impossible, because quantum computing is reversible, and so the NAND operation must be replaced by an equivalent reversible operation. It is also necessary to find the equivalent of the COPY operation without coming into conflict with the no-cloning theorem when transposition is made to the quantum version of the gates. This can be done using two logic gates, the cNOT gate and the Toffoli gate (Fig. 5.2). If the bits entering the *control-NOT (cNOT)*

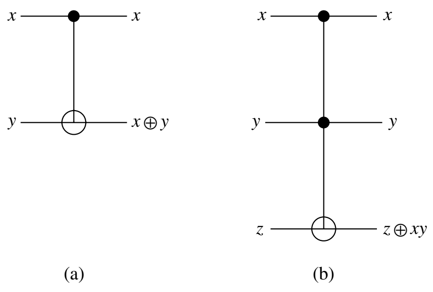


Figure 5.2 The cNOT gate (a) and the Toffoli gate (b). The black points represent the control bits and the circles represent the target bits.

[3] "Information is physical," according to Landauer, who went as far as deducing (debatably in the author's opinion) from this that mathematics and information science are branches of physics!

gate are (x, y), where x is the *control bit* and y is the *target bit*, the action of the cNOT gate on the target bit depends on the state of the control bit according to the scheme

$$\text{cNOT}: \quad (x, y) \rightarrow (x, x \oplus y). \tag{5.2}$$

Box 5.1: Maxwell's demon and the physical nature of information

In this box we show that it is impossible to ignore the fact that information must be carried by a physical support; otherwise, we come into conflict with the second law of thermodynamics. In 1871 Maxwell imagined the following device. A container filled with a gas at absolute temperature T is divided into two compartments of identical volume, separated by a wall in which a small hole is pierced (Fig. 5.3). A demon can observe the velocity of the molecules and open and close this hole by a door without expending energy. The molecules in the container have an average velocity of several hundred meters per second at ambient temperature ($T \simeq 300\,\text{K}$),

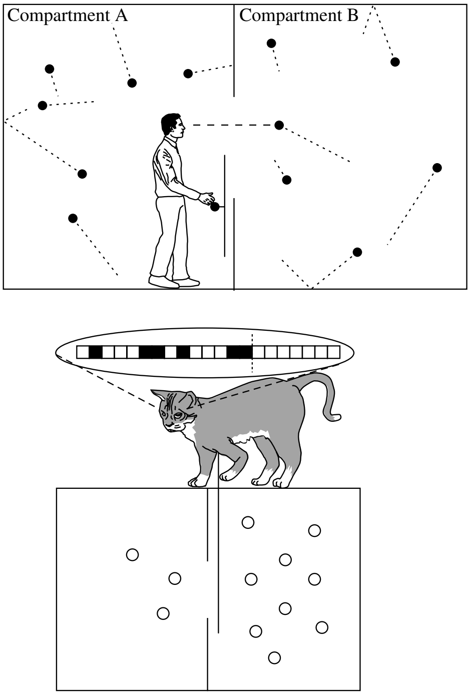


Figure 5.3 Maxwell's demon. The demon stores the position of the molecules in his memory.

but some are faster and others are slower. The demon opens the door when he sees a fast molecule traveling from the left-hand to the right-hand compartment, and also when he sees a slow molecule traveling from the right-hand to the left-hand compartment. Therefore, the average speed of the molecules in the right-hand compartment will increase, while that of the molecules in the left-hand compartment will decrease, with the total energy of the gas remaining constant. Since the average velocity is related to T and to the mass m of a molecule as

$$v \simeq \sqrt{\frac{k_B T}{m}},$$

the right-hand compartment will become warmer than the left-hand one. These two compartments can then be used as two heat sources at different temperatures to run a heat engine, thus making it possible to obtain work from only a single heat source, contradicting the second law of thermodynamics (equivalently, we could make a refrigerator without a motor, which is also forbidden by the second law).

In 1929, this problem was reduced to its bare essentials by Szilard, who considered a gas limited to a single molecule. This molecule can be localized in one or the other compartment without expending energy, and it does work by pushing a piston until it occupies the entire container, while taking energy from the outside in the form of heat. The expansion is done at constant temperature, and the work done is given by

$$W_0 = k_B T \int_{V/2}^{V} \frac{dV'}{V'} = k_B T \ln 2,$$

where V is the volume of the container. The operation can be performed n times so as to obtain an arbitrarily large amount of work $W = nW_0 = nk_B T \ln 2$, all using a single heat source.

The paradox was elucidated by Bennett in 1982. He noted that this device *does not function on a cycle*, which is the condition for the second law of thermodynamics to be valid, because the localization of the molecule in one or the other compartment during the n operations involves the assumption that this information will be stored in a memory of n bits. If we wish to erase the contents of this memory in order to restart from zero and perform a complete cycle, this would release into the environment an entropy of at least $nk_B \ln 2$, and therefore dissipate into the environment an energy of at least $nk_B T \ln 2$, which would convert all the work performed into heat.

Looking at this in more detail, we see that when the compartment in which the molecule is located has been determined, the information entropy of the system is one bit (corresponding to a thermodynamic entropy $k_B \ln 2$), because the position of the molecule and the contents of the memory are correlated. Once the expansion has occurred, the value of the information entropy is two bits, because the information about the compartment is lost. The information entropy of the environment must therefore decrease by one bit, that is, an energy $k_B T \ln 2$ equal to the work W_0 is supplied by the environment. When the memory is erased, we return to a one-bit entropy for the system, which means that the environment will receive at least one bit, because the entropy of the ensemble (system + environment) can only increase. If the operations are performed in a quasi-static manner, they are all reversible and we return to the starting point after a cycle. In contrast to the case where *deterministic* data are erased as in the thermodynamically irreversible NAND operation, in the case we are discussing in this Box it is *random* data which are reversibly erased.

The cNOT gate copies the bit x if $y = 0$ and gives $\neg x$ if $y = 1$, and is the reversible equivalent of the COPY operation. It is reversible because there is a one-to-one correspondence between the initial state and the final state. The cNOT operation is a simple permutation of the basis vectors (see (5.4)). It can be shown that using single-bit gates

$$x \to 1 \oplus x \quad \text{or} \quad x \to \neg x$$

and the cNOT gate, it is possible to construct only linear functions if we limit ourselves to classical operations. If (x, y) and (x', y') are the initial and final bits, one can show that

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$$

where $\alpha, \ldots, \mu$ are numerical coefficients. It is necessary to introduce an additional gate, the *Toffoli gate*, which is a gate with three entrance and three exit bits, two of them control bits (x, y) and one a target bit z:

$$\text{Toffoli}: \quad (x, y, z) \to (x, y, z \oplus xy). \tag{5.3}$$

The nonlinearity of the gate is obvious from the xy factor. If $z = 1$, the Toffoli gate performs the NAND operation reversibly. The Toffoli gate can be used to reproduce reversibly all the classical logic circuits: the Toffoli gate is a universal gate for all the reversible operations of Boolean logic.

5.3 Quantum logic gates

The most general quantum evolution is a unitary transformation in the 2^n-dimensional Hilbert space of n qubits, $\mathcal{H}^{\otimes n}$. The most general quantum logic gate is a $2^n \times 2^n$ unitary matrix operating in $\mathcal{H}^{\otimes n}$. A theorem of linear algebra which we state without proof will allow us to limit ourselves to operations on one and two qubits.

Theorem Any unitary transformation on $\mathcal{H}^{\otimes n}$ can be decomposed into a product of cNOT gates and unitary transformations on one qubit.

As already explained, an operation on individual qubits cannot produce a general unitary transformation of $\mathcal{H}^{\otimes n}$, because such an operation has the form of a tensor product:

$$U = U^{(1)} \otimes U^{(2)} \otimes \cdots \otimes U^{(n)}.$$

It is necessary to perform nontrivial operations on at least two qubits to obtain a general unitary transformation. The above theorem guarantees that this is sufficient. This theorem is an existence theorem; in general, it is easier to construct the quantum logic gates for a given problem without using this theorem explicitly. It is useful to give the 4×4 matrix representation of the cNOT gate. In terms of qubits, the cNOT operation corresponds to the transformation

$$|00\rangle \to |00\rangle, \qquad |01\rangle \to |01\rangle, \qquad |10\rangle \to |11\rangle, \qquad |11\rangle \to |10\rangle.$$

In the basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ the matrix representation then becomes

$$\text{cNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & \sigma_x \end{pmatrix}. \tag{5.4}$$

In this form it is clear that cNOT cannot be a tensor product (Exercise 5.10.1). The generalization of the cNOT gate is the *control-U (cU) gate*, where the matrix σ_x is replaced by a 2×2 unitary matrix U:

$$\text{cU} = \begin{pmatrix} I & 0 \\ 0 & U \end{pmatrix}.$$

The cU gate leaves the target bit unchanged if $x = 0$ and modifies it as $|y\rangle \to U|y\rangle$ if $x = 1$. The cU gate can be constructed starting from the cNOT gate (Fig. 5.4). It is necessary to find three unitary operators A, B, and C such that

$$CBA = I, \qquad C\sigma_x B\sigma_x A = U.$$

In quantum physics, the Toffoli gate may be constructed from cU gates and cNOT gates (Fig. 5.4 with $U = \sqrt{\sigma_x}$) and the equation

$$\sqrt{\sigma_x} = \frac{1}{1+\mathrm{i}} \begin{pmatrix} 1 & \mathrm{i} \\ \mathrm{i} & 1 \end{pmatrix},$$

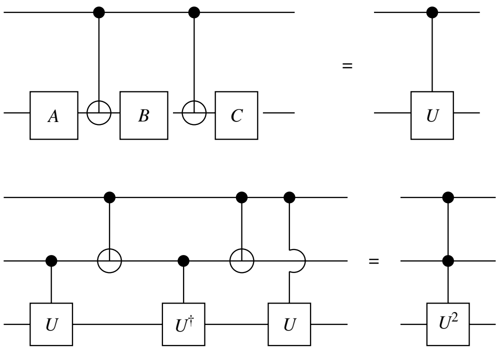


Figure 5.4 Construction of the cU gate and the Toffoli gate. The diagrams are read from left to right, and the products of operators act from right to left.

which is not possible in classical physics where the operation $\sqrt{\sigma_x}$ does not exist. In contrast to the classical case, it is not necessary to introduce the Toffoli gate explicitly to construct the ensemble of reversible logic circuits. We see from the results of Section 5.2 that if we have at our disposal a classical logic circuit allowing a function $f(x)$ to be calculated, then we can construct a quantum circuit using essentially the same number of gates. The justification of the circuits in Fig. 5.4 is left to Exercise 5.10.1.

Now that we know that there exists a quantum logic circuit which can evaluate a function $f(x)$, for example, by transposing a classical algorithm, we can state the basic ideas of *quantum parallelism*. We shall use two registers, an input register which stores x and an output register which stores the bits needed for $f(x)$. To simplify the discussion, we start with the case where the input register is a one-qubit register, as is the output register. We construct a transformation U_f (Fig. 5.5) which performs the operations

$$(x, y) \xrightarrow{U_f} (x, y \oplus f(x)). \tag{5.5}$$

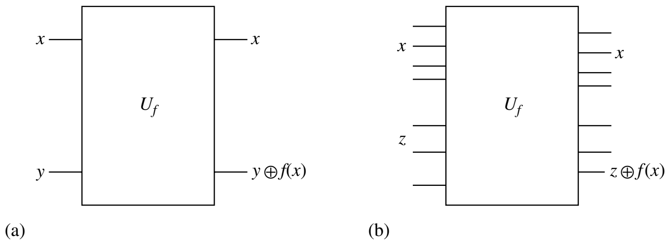


Figure 5.5 The construction of U_f: (a) 2 qubits, (b) $n+m$ qubits.

If the initial value is $y=0$, we have simply

$$(x,0) \xrightarrow{U_f} (x, f(x)).$$

One might ask why we do not simply perform the transformation $x \to f(x)$. The answer is that such a transformation cannot be unitary if the correspondence between x and $f(x)$ is not one-to-one, and so it is not suitable for a quantum algorithm. On the contrary, it is easy to convince ourselves that U_f is unitary, because its square is the identity:

$$(x, [y \oplus f(x)]) \xrightarrow{U_f} (x, [y \oplus f(x)] \oplus f(x)) = (x, y),$$

owing to $f(x) \oplus f(x) = 0$ for any $f(x)$. The operation U_f transforms one basis vector into another, and since $U_f^2 = I$ this correspondence can only be a simple permutation of the four basis vectors, and so it is a unitary transformation. In operator notation,

$$U_f|x \otimes 0\rangle = |x \otimes f(x)\rangle, \qquad U_f|x \otimes y\rangle = |x \otimes [y \oplus f(x)]\rangle. \tag{5.6}$$

Let us apply to the state $|0\rangle_x$ a Hadamard gate H (not to be confused with the Hamiltonian $\hat{H}$):

$$\boxed{H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}} \tag{5.7}$$

or

$$H|0\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle), \qquad H|1\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).$$

Now if the second qubit is in the initial state $|0\rangle$, the final state vector of the two qubits is the entangled state

$$|\Psi\rangle = U_f|H0 \otimes 0\rangle = U_f \frac{1}{\sqrt{2}} (|0 \otimes 0\rangle + |1 \otimes 0\rangle) = \frac{1}{\sqrt{2}} (|0 \otimes f(0)\rangle + |1 \otimes f(1)\rangle). \tag{5.8}$$

The state vector $|\Psi\rangle$ contains the information on $f(0)$ and $f(1)$ *simultaneously*, and the calculation of the vector $|\Psi\rangle$ *requires the same number of operations as that of* $U_f|0 \otimes 0\rangle$ *or* $U_f|1 \otimes 0\rangle$ *separately*: U_f is a unitary operator which does not depend on the state vector to which it is applied.

5.4 The Deutsch algorithm

Although $|\Psi\rangle$ in (5.8) contains information on $f(0)$ and $f(1)$ simultaneously, this does not give us any advantage over a classical computer if we wish to construct a table of values of $f(x)$ explicitly. However, it may happen that we need only information which does not require the construction of such a table. It is then possible that a quantum algorithm can exploit the information contained in $|\Psi\rangle$ to obtain the result using fewer operations than a classical algorithm. We shall explain how this works for the example of the Deutsch algorithm.

The Deutsch algorithm can be realized using the circuit of Fig. 5.6, with one-qubit input and output registers. The unknown function $f(x)$ takes the value 0 or 1 and we can ask the following question: do we have $f(0) = f(1)$ (a "constant" function) or $f(0) \neq f(1)$ (a "balanced" function)? If we were using a classical computer, we would have to calculate $f(0)$ and $f(1)$ and compare the two values. If we use a quantum computer, the question can be answered in a *single* operation. An equivalent problem is that of checking a coin: are the two sides different (a head *and* a tail) or are they the same (two heads or two tails)? The quantum computer allows us to make this comparison without looking at the two sides of the coin in succession.[4] This example is of course too elementary to be of any practical interest, but it gives the simplest illustration of quantum parallelism, and moreover it is a good warmup for the Grover algorithm of Section 5.6. The circuit of Fig. 5.6 gives the state $|\Psi\rangle$ at the entrance to the box U_f, the input register initially being in the state $|0\rangle$ and the output register in the state $|1\rangle$:

$$|\Psi\rangle = (H|0\rangle) \otimes (H|1\rangle) = \frac{1}{2}\left(|0\rangle + |1\rangle\right) \otimes \left(|0\rangle - |1\rangle\right) = \frac{1}{2}\left(\sum_{x=0}^{1}|x\rangle\right) \otimes \left(|0\rangle - |1\rangle\right). \tag{5.9}$$

We apply U_f (5.8) to this state with the following result:

1. if $f(x) = 0$, then $(|0\rangle - |1\rangle) \to (|0\rangle - |1\rangle)$,
2. if $f(x) = 1$, then $(|0\rangle - |1\rangle) = (|1\rangle - |0\rangle) \to -(|0\rangle - |1\rangle)$,

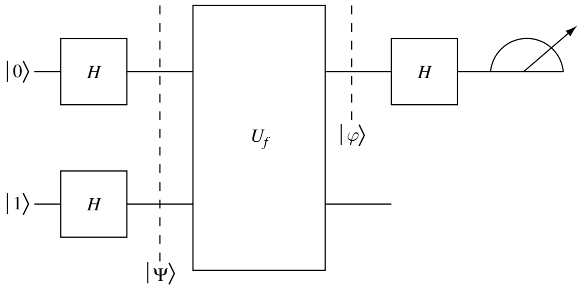


Figure 5.6 The Deutsch algorithm.

[4] Film-lovers may recall Hawks's *Only Angels Have Wings*, where Cary Grant says to Jean Arthur, "Heads you stay and tails you leave," and Jean Arthur, furious at being the subject of the flip of a coin, nevertheless checks the coin before leaving, and finds that it has two heads.

or, to summarize,

$$(|0\rangle - |1\rangle) \to (-1)^{f(x)}\,(|0\rangle - |1\rangle)\,. \tag{5.10}$$

The state $U_f|\Psi\rangle$ is then the tensor product

$$U_f|\Psi\rangle = \frac{1}{2}\left(\sum_{x=0}^{1}(-1)^{f(x)}\,|x\rangle\right)\otimes(|0\rangle - |1\rangle)\,. \tag{5.11}$$

The net result for the input register is

$$|x\rangle \xrightarrow{U_f} (-1)^{f(x)}|x\rangle. \tag{5.12}$$

In this particular case, the box U_f is called an *oracle*. Note that the input and output registers are unentangled after the oracle. The state of the qubit of the input register then is

$$|\varphi\rangle = \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right).$$

Before measuring the input register, we apply a Hadamard gate (see Fig. 5.6):

$$\begin{aligned} H|\varphi\rangle &= \frac{1}{2}\left[(-1)^{f(0)}\,(|0\rangle + |1\rangle) + (-1)^{f(1)}\,(|0\rangle - |1\rangle)\right] \\ &= \frac{1}{2}\left[(-1)^{f(0)} + (-1)^{f(1)}\right]|0\rangle + \frac{1}{2}\left[(-1)^{f(0)} - (-1)^{f(1)}\right]|1\rangle. \end{aligned} \tag{5.13}$$

If measurement of the qubit gives $|0\rangle$, then $f(0) = f(1)$, i.e., the function is a "constant" one. If it gives $|1\rangle$, then $f(0) \neq f(1)$ and the function is a "balanced" one. The important point is that quantum parallelism has allowed us to bypass the explicit calculation of the function $f(x)$; the measurement of a single qubit contains the two possible results. The generalization to several qubits is left to Exercise 5.10.2.

5.5 Generalization to $n + m$ qubits

The above discussion can be generalized to an n-qubit input register and m-qubit output register, where m is the number of bits needed to write $f(x)$. We take as an example the case $n = 3$ for the input register. Using the notation $|x\rangle$, the number x is one of the eight numbers (in binary notation)

$$|000\rangle, \quad |001\rangle, \quad |010\rangle, \quad |011\rangle, \quad |100\rangle, \quad |101\rangle, \quad |110\rangle, \quad |111\rangle.$$

The special magic of a quantum computer is that it allows us to make linear combinations of the vectors of the computational basis using the operator H, which in the particular case $n = 3$ gives

$$|\Psi\rangle := H^{\otimes 3}|000\rangle = \frac{1}{\sqrt{8}}\sum_{x=0}^{7}|x\rangle,$$

where $H^{\otimes 3}$ denotes the tensor product of the three operators H. In general,

$$H^{\otimes n}|0^{\otimes n}\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle.$$

Here x is compact notation for the binary representation of the number x and the state vector of the computational basis is $|x\rangle = |x_{n-1}\cdots x_1 x_0\rangle$, where $x_{n-1}, \ldots, x_1, x_0$ take the value 0 or 1. The operator U_f is defined by generalizing the definition (5.6) as[5] (Fig. 5.5(b))

$$U_f|x \otimes z\rangle = |x \otimes [z \oplus f(x)]\rangle,$$

where $\oplus$ is mod 2 addition *without carry-over*, for example,

$$1101 \oplus 0111 = 1010.$$

We recall that

$$|x\rangle = |x_{n-1}\cdots x_1 x_0\rangle, \qquad |z\rangle = |z_{n-1}\cdots z_1 z_0\rangle$$

with $x_i, z_j = 0$ or 1. This assures that $U_f^2 = I$ and that U_f, which is a simple permutation of the 2^{n+m} basis vectors, is unitary. If we take $|0^{\otimes m}\rangle$ as the initial state of the output register, then

$$U_f|x \otimes 0^{\otimes m}\rangle = |x \otimes f(x)\rangle.$$

If finally we apply H to the input register in the state $|0^{\otimes n}\rangle$ before U_f, the state vector of the final state will be, by linearity,

$$|\Psi_{\text{fin}}\rangle = U_f|(H^{\otimes n}\, 0^{\otimes n}) \otimes 0^{\otimes m}\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x \otimes f(x)\rangle. \qquad (5.14)$$

This state vector in principle contains the 2^n values of the function $f(x)$ (not necessarily all of them different). For example, if $n = 100$, it contains the $\sim 10^{30}$ values of $f(x)$: it is this exponential growth of states which leads to the miracle of quantum parallelism. A measurement can of course give only one of these values. As we have seen in the case of the Deutsch algorithm, it is nevertheless possible to extract useful information about the *relations* between the values of $f(x)$ for an ensemble of different values of x, of course at the price of losing the individual values. A classical computer, on the other hand, would have to evaluate $f(x)$ for all these values of x independently. In Section 5.7 we shall discuss this using the example of a quantum Fourier transform.

[5] Since later on we shall use y in a different context, here we denote the output register by z.

5.6 The Grover search algorithm

A quantum algorithm of more practical relevance than the Deutsch algorithm is the Grover search algorithm. This is an algorithm which performs a search for an entry in an *unstructured* data base, for example, a person's name in a telephone directory when the phone number is known. If N is the number of entries in the data base, a classical algorithm must on the average make $N/2$ attempts to find the name, as the only possibility is to check each entry one by one. The Grover algorithm allows the problem to be solved in $\sim\sqrt{N}$ operations.

The data base is stored using n qubits and we define the function $f(x)$, $x = \{0, 1, \ldots, 2^n - 1\}$, such that $f(x) = 0$ if $x \neq y$ and $f(x) = 1$ if $x = y$ is a solution: $f(x) = \delta_{xy}$. To simplify the argument, we assume that the value of y is unique. We define an operator O, the oracle, whose action in the computational basis is (see (5.12))

$$O|x\rangle = (-1)^{f(x)}|x\rangle. \tag{5.15}$$

As in the case of the Deutsch algorithm (see (5.11)), the auxiliary qubit (lowest qubit in Fig. 5.8) is unentangled with the other qubits after the oracle. The Grover operator G is defined as

$$G = H^{\otimes n}\, X\, H^{\otimes n}\, O = H^{\otimes n}\,(2|0\rangle\langle 0| - I)\, H^{\otimes n}\, O, \tag{5.16}$$

where

$$X|x\rangle = -(-1)^{\delta_{x0}}|x\rangle = (2|0\rangle\langle 0| - I)|x\rangle.$$

To simplify the notation, we shall introduce the vector $|\Psi\rangle$ already used in Section 5.5:

$$|\Psi\rangle = H^{\otimes n}\,|0^{\otimes n}\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle. \tag{5.17}$$

Taking into account $H^2 = I$, we find

$$H^{\otimes n}\,(2|0\rangle\langle 0| - I)\, H^{\otimes n} = 2H^{\otimes n}|0\rangle\langle 0|H^{\otimes n} - I = 2|\Psi\rangle\langle\Psi| - I$$

and so

$$G = (2|\Psi\rangle\langle\Psi| - I)\, O. \tag{5.18}$$

This construction can be used to draw the quantum logic circuit corresponding to G (see Fig. 5.8(b)).

The operator G can be interpreted as a rotation in a two-dimensional plane. Let $|\alpha\rangle$ be the normalized vector

$$|\alpha\rangle = \frac{1}{\sqrt{N-1}} \sum_{x\neq y} |x\rangle \tag{5.19}$$

($N = 2^n$), which can be used to write $|\Psi\rangle$ as

$$|\Psi\rangle = \sqrt{1 - \frac{1}{N}}\,|\alpha\rangle + \sqrt{\frac{1}{N}}\,|y\rangle. \tag{5.20}$$

We rewrite this equation as

$$|\Psi\rangle = \cos\frac{\theta}{2}\,|\alpha\rangle + \sin\frac{\theta}{2}\,|y\rangle, \tag{5.21}$$

where the angle θ is given by

$$\cos\frac{\theta}{2} = \sqrt{1 - \frac{1}{N}}.$$

According to (5.15), the action of the oracle on $|\Psi\rangle$ is

$$O(\lambda|\alpha\rangle + \mu|y\rangle) = \lambda|\alpha\rangle - \mu|y\rangle.$$

This is a reflection with respect to the direction of $|\alpha\rangle$ in the plane Π subtended by $|\alpha\rangle$ and $|y\rangle$ (Fig. 5.7). Moreover, $(2|\Psi\rangle\langle\Psi| - I)$ performs a reflection in Π with respect to the direction of $|\Psi\rangle$: if $\langle\Psi|\Phi\rangle = 0$,

$$(2|\Psi\rangle\langle\Psi| - I)(\lambda|\Psi\rangle + \mu|\Phi\rangle) = \lambda|\Psi\rangle - \mu|\Phi\rangle.$$

However, the product of two reflections is a rotation, and Fig. 5.7 shows that the angle taking us from $|\alpha\rangle$ to $G|\Psi\rangle$ is $3\theta/2$:

$$G|\Psi\rangle = \cos\frac{3\theta}{2}\,|\alpha\rangle + \sin\frac{3\theta}{2}\,|y\rangle. \tag{5.22}$$

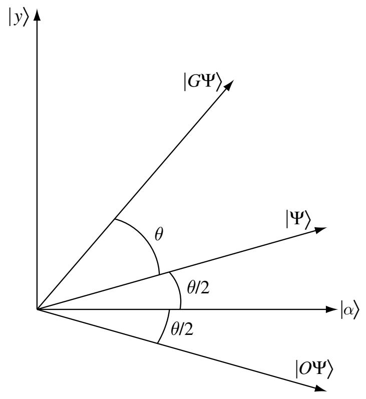


Figure 5.7 Schematic depiction of the rotations and reflections of the Grover algorithm.

The angle between $|\Psi\rangle$ and $G|\Psi\rangle$ is θ, and the angle between $|\alpha\rangle$ and $G|\Psi\rangle$ is $3\theta/2$; $G|\Psi\rangle$ is deduced from $|\Psi\rangle$ by rotation by an angle θ. Moreover, $G^2|\Psi\rangle$ is deduced from $G|\Psi\rangle$ by rotation by an angle θ. After k iterations of G, $G^k|\Psi\rangle$ is always in Π and is deduced from $|\alpha\rangle$ by rotation by an angle $(2k+1)\theta/2$:

$$G^k|\Psi\rangle = \cos\frac{(2k+1)\theta}{2}\,|\alpha\rangle + \sin\frac{(2k+1)\theta}{2}\,|y\rangle. \tag{5.23}$$

The effect of successive rotations is to make $G^k|\Psi\rangle$ come closer and closer to $|y\rangle$. The optimal value $k = k_0$ of k is determined using the following argument. We wish to have

$$\begin{aligned} 0 = \cos\frac{(2k+1)\theta}{2} &= \cos k\theta\,\cos\frac{\theta}{2} - \sin k\theta\,\sin\frac{\theta}{2} \\ &= \sqrt{1-\frac{1}{N}}\,\cos k\theta - \sqrt{\frac{1}{N}}\,\sin k\theta. \end{aligned}$$

We then find that $\tan k\theta = \sqrt{N-1}$, or $\cos k\theta = 1/\sqrt{N}$, and so

$$k_0 = \left[\frac{1}{\theta}\cos^{-1}\sqrt{\frac{1}{N}}\right] + 1,$$

where $[x]$ is the integer part of x. For $N \gg 1$ we have, comparing (5.20) and (5.21), $\theta \simeq 2/\sqrt{N}$ or

$$k_0 \simeq \frac{\sqrt{N}}{2}\cos^{-1}\sqrt{\frac{1}{N}} \simeq \frac{\pi\sqrt{N}}{4}. \tag{5.24}$$

It is therefore sufficient to apply the oracle $\sim\sqrt{N}$ times in order to have a very good chance of obtaining the result. To estimate the probability of this, we

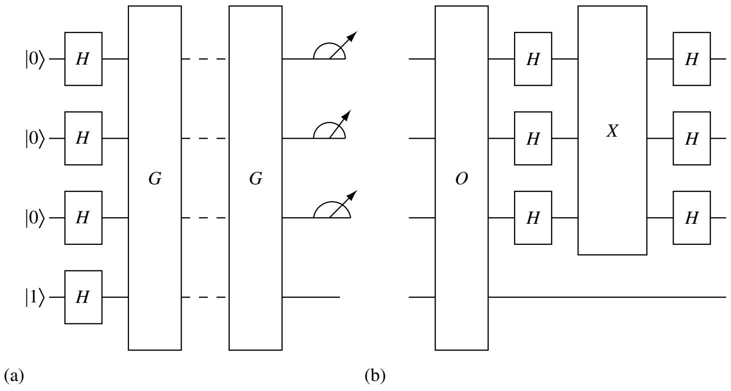


Figure 5.8 (a) Logic circuits of the Grover algorithm for $n = 3$. (b) The circuits of G. The action of the oracle O is $O|x\rangle = (-1)^{f(x)}|x\rangle$ and that of the box X is $X|x\rangle = -(-1)^{\delta_{x0}}|x\rangle$.

note that according to Fig. 5.7 the angle between $G^{k_0}|\Psi\rangle$ and $|y\rangle$ is less than $\theta/2$. The probability of error is therefore less than $\mathcal{O}(1/N)$. It can be shown that the Grover algorithm is optimal: it is not possible to find a faster algorithm than Grover's. If we count the quantum logic gates, the total number of operations of the Grover algorithm is actually $\simeq\sqrt{N}\ln N$. The circuit corresponding to the Grover algorithm is shown schematically in Fig. 5.8.

5.7 The quantum Fourier transform

The last algorithm we shall describe is that of Shor. As a preliminary, let us construct a quantum logic circuit for the Fourier transform. Let an integer x, $0 \le x \le 2^n - 1$, be written using n bits

$$x = 0, 1, \ldots, 2^n - 1,$$

and let $|x\rangle$ be a vector of the computational basis

$$|x\rangle = |x_{n-1}\cdots x_1 x_0\rangle, \qquad x_i = 0 \text{ or } 1.$$

We define a unitary transformation[6] $U_{\rm FT}$ whose matrix elements in the computational basis are

$$\langle y|U_{\rm FT}\, x\rangle = (U_{\rm FT})_{yx} = \frac{1}{2^{n/2}}\, \mathrm{e}^{2\mathrm{i}\pi xy/2^n}. \tag{5.25}$$

The transformation $U_{\rm FT}$ is physically realized in the box $U_{\rm FT}$ of Fig. 5.9(a), and, as we shall soon see, a possible circuit is that given in Fig. 5.9(b). If $|\Psi\rangle$ is a normalized linear combination of the vectors $|x\rangle$,

$$|\Psi\rangle = \sum_{x=0}^{2^n-1} f(x)|x\rangle, \qquad \sum_{x=0}^{2^n-1} |f(x)|^2 = 1, \tag{5.26}$$

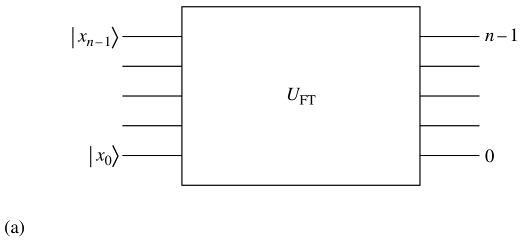
where $f(x) = \langle x|\Psi\rangle$, then the amplitude for finding at the exit from the box $U_{\rm FT}$ a state $|y\rangle$ of the computational basis (note that $|y\rangle$ denotes a state of the input register) is, from (2.17) setting $|\Phi\rangle = U_{\rm FT}|\Psi\rangle$,

$$\begin{aligned} a(\Phi\to y) = \langle y|\Phi\rangle &= \sum_{x=0}^{2^n-1} \langle y|U_{\rm FT}\, x\rangle\langle x|\Psi\rangle \\ &= \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} \mathrm{e}^{2\mathrm{i}\pi xy/2^n} f(x) = \tilde{f}(y), \end{aligned} \tag{5.27}$$

[6] In fact,

$$\sum_{y=0}^{2^n-1} (U^\dagger_{\rm FT})_{x'y}(U_{\rm FT})_{yx} = \sum_{y=0}^{2^n-1} (U^*_{\rm FT})_{yx'}(U_{\rm FT})_{yx} = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \mathrm{e}^{2\mathrm{i}\pi(x-x')y/2^n} = \delta_{x'x}.$$

The result is obtained upon noticing that the sum over y is a geometric series.



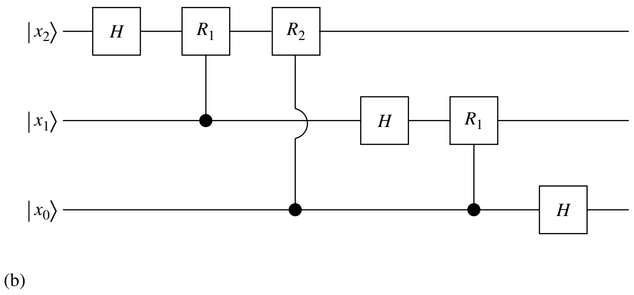


Figure 5.9 (a) The box U_{FT}. (b) A circuit constructing U_{FT} in the case $n = 3$.

where we have used the completeness relation $\sum_x |x\rangle\langle x| = I$ (Box 2.1). The probability amplitude $a(\Phi \to y)$ is just the discrete (or lattice) Fourier transform $\tilde{f}(y)$ of $f(x)$.

For constructing the box U_{FT} it is convenient to write $U_{\mathrm{FT}}|x\rangle$ as

$$U_{\mathrm{FT}}|x\rangle = \sum_{y=0}^{2^n-1} |y\rangle\langle y|U_{\mathrm{FT}}\,x\rangle = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} \mathrm{e}^{2\mathrm{i}\pi xy/2^n}\,|y\rangle. \tag{5.28}$$

We shall transform (5.28) so as to write it as a manifestly nonentangled state using a standard technique of fast Fourier transforms. Let

$$\begin{aligned} x &= x_0 + 2x_1 + \cdots + 2^{n-1}x_{n-1}, \\ y &= y_0 + 2y_1 + \cdots + 2^{n-1}y_{n-1}, \end{aligned} \tag{5.29}$$

be the binary decompositions of x and y and let us examine the factor $xy/2^n$ in the exponential of (5.28) in the case $n = 3$, $N = 2^3 = 8$. Using the fact that $\exp(2\mathrm{i}\pi p) = 1$ for integer p, we can replace the product $xy/8$ in the exponential of (5.28) by

$$\begin{aligned} \frac{xy}{8} &\to y_0\left(\frac{x_2}{2} + \frac{x_1}{4} + \frac{x_0}{8}\right) + y_1\left(\frac{x_1}{2} + \frac{x_0}{4}\right) + y_2\,\frac{x_0}{2} \\ &= y_0.x_2x_1x_0 + y_1.x_1x_0 + y_2.x_0, \end{aligned}$$

where we have introduced the notation (the binary representation of a number less than one)

$$.x_p x_{p-1}\cdots x_1 x_0 = \frac{x_p}{2} + \frac{x_{p-1}}{2^2} + \cdots + \frac{x_0}{2^p}. \tag{5.30}$$

Coming back to the general case, we see that it is now possible to factorize the sum over y into sums over $y_0, \ldots, y_{n-1}$, $y_i = 0$ or 1, $|y\rangle = |y_{n-1}\cdots y_1 y_0\rangle$:

$$\begin{aligned}U_{\rm FT}|x\rangle &= \frac{1}{2^{n/2}} \sum_{y_0,\ldots,y_{n-1}} \mathrm{e}^{2\mathrm{i}\pi y_{n-1}.x_0} \ldots \mathrm{e}^{2\mathrm{i}\pi y_0.x_{n-1}x_0} \ldots |y_{n-1}\cdots y_0\rangle \\ &= \frac{1}{2^{n/2}} \left(\sum_{y_{n-1}} \mathrm{e}^{2\mathrm{i}\pi y_{n-1}.x_0} |y_{n-1}\rangle\right) \cdots \left(\sum_{y_1} \mathrm{e}^{2\mathrm{i}\pi y_1.x_{n-2}\cdots x_0} |y_1\rangle\right) \\ &\quad \times \left(\sum_{y_0} \mathrm{e}^{2\mathrm{i}\pi y_0.x_{n-1}\cdots x_0} |y_0\rangle\right),\end{aligned}$$

or, expanding each quantity in parentheses,

$$\begin{aligned}U_{\rm FT}|x\rangle = &\frac{1}{2^{n/2}} \left(|0\rangle_{n-1} + \mathrm{e}^{2\mathrm{i}\pi.x_0}|1\rangle_{n-1}\right) \cdots \left(|0\rangle_1 + \mathrm{e}^{2\mathrm{i}\pi.x_{n-2}\cdots x_0}|1\rangle_1\right) \\ &\times \left(|0\rangle_0 + \mathrm{e}^{2\mathrm{i}\pi.x_{n-1}\cdots x_0}|1\rangle_0\right),\end{aligned} \tag{5.31}$$

which manifestly has the form of a tensor product. Let us give an example for $n = 2$:

$$\begin{aligned}U_{\rm FT}|x\rangle \equiv U_{\rm FT}|x_1 x_0\rangle &= \frac{1}{4}\left(|00\rangle + \mathrm{e}^{2\mathrm{i}\pi.x_1x_0}|01\rangle + \mathrm{e}^{2\mathrm{i}\pi.x_0}|10\rangle + \mathrm{e}^{2\mathrm{i}\pi(.x_1x_0+.x_0)}|11\rangle\right) \\ &= \frac{1}{4}\left(|0\rangle_1 + \mathrm{e}^{2\mathrm{i}\pi.x_0}|1\rangle_1\right)\left(|0\rangle_0 + \mathrm{e}^{2\mathrm{i}\pi.x_1x_0}|1\rangle_0\right).\end{aligned}$$

A possible logic circuit for performing this Fourier transform is shown in Fig. 5.9(b). The gate cR_d is defined by the operator R_d:

$$R_d = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\pi/2^d} \end{pmatrix}. \tag{5.32}$$

Let us study the circuit of Fig. 5.9(b). The action of the gate H is

$$H|0\rangle_2 = \frac{1}{\sqrt{2}}\left(|0\rangle_2 + |1\rangle_2\right), \qquad H|1\rangle_2 = \frac{1}{\sqrt{2}}\left(|0\rangle_2 - |1\rangle_2\right),$$

and so the action on the first bit $|x_2\rangle$ can be written as

$$H|x_2\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_2 + \mathrm{e}^{2\mathrm{i}\pi.x_2}|1\rangle_2\right). \tag{5.33}$$

We use $c_i R_d^j$ to denote the action on the bit j of R_d controlled by the bit i. Then

$$x_1 = 0: \qquad (c_1 R_1^2) H|x_2\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle_2 + e^{2i\pi.x_2}\,|1\rangle_2\Big),$$

$$x_1 = 1: \qquad (c_1 R_1^2) H|x_2\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle_2 + e^{2i\pi.x_2}\,e^{i\pi/2}|1\rangle_2\Big),$$

which can be written as

$$(c_1 R_1^2) H|x_2\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle_2 + e^{2i\pi.x_2x_1}\,|1\rangle_2\Big). \tag{5.34}$$

It is clear that the procedure is followed by

$$(c_0 R_2^2)(c_1 R_1^2) H|x_2\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle_2 + e^{2i\pi.x_2x_1x_0}\,|1\rangle_2\Big), \tag{5.35}$$

and after applying all the gates in Fig. 5.9(b) we obtain the state

$$|\Psi'\rangle = \frac{1}{\sqrt{8}}\Big(|0\rangle_0 + e^{2i\pi.x_0}|1\rangle_0\Big)\Big(|0\rangle_1 + e^{2i\pi.x_1x_0}|1\rangle_1\Big)\Big(|0\rangle_2 + e^{2i\pi.x_2x_1x_0}|1\rangle_2\Big).$$

The qubits are in the wrong order, and one may use SWAP gates to put them in the right one. However, one can also write by convention the computational basis in the order

$$|x\rangle = |x_0 x_1 \cdots x_{n-1}\rangle$$

that is, the first digit is x_0 and the last digit x_{n-1}, meaning that the number is read from right to left. Then one can avoid the SWAP gates altogether. The number of gates needed decomposes into n gates H and

$$n + (n-1) + \cdots + 1 \simeq \frac{1}{2}n^2$$

conditional gates cR_d, or $\mathcal{O}(n^2)$ gates.

5.8 The period of a function

The Shor factorization algorithm is based on the possibility of "rapidly," that is, in a time which is a polynomial in n, finding the period of a function $f(x)$, which in the Shor case is the function $b^x \bmod N$. Let us assume that we have a function $f(x)$ of period r, $f(x) = f(x+r)$, with

$$x = 0, 1, \ldots, 2^n - 1. \tag{5.36}$$

For the algorithm to be successful we must have, as we shall see, $2^n > N^2$. A classical algorithm uses $\mathcal{O}(N)$ elementary operations (the function $b^x \bmod N$ appears to be random noise over a period and does not give any key to what the period is), but the quantum algorithm we shall describe below uses only $\mathcal{O}(n^3)$ elementary operations. The variable x is stored in a register $|x\rangle$ and the function $f(x)$ in a register $|z\rangle$ corresponding to m qubits. We start from an initial state of $n+m$ qubits:

$$|\Phi\rangle = \frac{1}{2^{n/2}}\left(\sum_{x=0}^{2^n-1}|x\rangle\right)\otimes|0\cdots 0\rangle. \tag{5.37}$$

We then use the box U_f which calculates the function $f(x)$:

$$|\Psi_f\rangle = U_f|\Phi\rangle = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\otimes f(x)\rangle. \tag{5.38}$$

This requires $\mathcal{O}(n)$ operations. If we measure the output register and find the result f_0, the state vector of the input register after this measurement is given by the state vector collapse (see Section 2.4)

$$|\Psi_0\rangle = \frac{1}{\mathcal{N}}\sum_{x;f(x)=f_0}|x\rangle, \tag{5.39}$$

where the sum runs over the values of x such that $f(x)=f_0$, and $\mathcal{N}$ is a normalization factor. We shall assume that $f(x+s)=f(x)$ implies that $s=pr$, p integer, in other words, the function $f(x)$ never takes the same value twice in a period, which is the case for the function b^x mod N. The normalized vector $|\Psi\rangle$ of the input register, with $f(x_0)=f_0$ and x_0 being the smallest value of x such that $f(x_0)=f_0$, then is

$$|\Psi_0\rangle = \frac{1}{\sqrt{K}}\sum_{k=0}^{K-1}|x_0+kr\rangle, \tag{5.40}$$

where[7] $K\simeq 2^n/r$. *In reality, it is not necessary to measure the output register* (Box 5.2). At the exit from the box U_f of Fig. 5.10, the qubits of the input register are entangled with the qubits of the output register (see (5.38)), and if only the qubits of the input register are observed, it is necessary to take the trace over the output register in order to obtain the state operator of the qubits of the input register; the physical state of the qubits of the input register will in general be described by a state operator, and not by a vector of $\mathcal{H}^{\otimes n}$. Stated differently, the physical state of the input register is an *incoherent* superposition of the vectors $|\Psi_i\rangle$:

$$|\Psi_i\rangle = \frac{1}{\sqrt{K_i}}\sum_{k=0}^{K_i-1}|x_i+kr\rangle, \tag{5.41}$$

where $f(x_i)=f_i$ and x_i is the smallest value of x such that $f(x_i)=f_i$. Since the rest of the argument does not depend on x_i, we can just as well avoid measuring the output register. In other words, it is completely unnecessary to resort to state vector collapse.

[7] More precisely, $K=[2^n/r]$ or $K=[2^n/r]+1$, where $[z]$ denotes the integer part of z.

Box 5.2: What measurements are needed?

Formally, the *total* state operator (of the input and output registers) $\rho_{\rm tot}$ is, according to (5.38),

$$\rho_{\rm tot} = |\Psi_f\rangle\langle\Psi_f| = \frac{1}{2^n}\sum_{x,z}|x\otimes f(x)\rangle\langle z\otimes f(z)|.$$

The state operator of the input register is obtained by taking the partial trace over the output register (see (4.14):

$$\rho_{\rm in} = {\rm Tr}_{\rm out}\rho_{\rm tot} = \frac{1}{2^n}\sum_{x,z}|x\rangle\langle z|\,\langle f(z)|f(x)\rangle.$$

Let us suppose that the function $f(x)$ takes the value f_0 N_0 times, and the value f_1 N_1 times, with $N_0+N_1=2^n$. Then

$$\rho_{\rm in} = \frac{1}{2^n}\left(\sum_{x,z;f(x)=f(z)=f_0}|x\rangle\langle z| + \sum_{x,z;f(x)=f(z)=f_1}|x\rangle\langle z|\right),$$

because $\langle f(x)|f(z)\rangle = 1$ if $f(x)=f(z)$ and $\langle f(x)|f(z)\rangle = 0$ if $f(x)\neq f(z)$. This corresponds to an incoherent superposition of normalized vectors

$$|\Psi_0\rangle = \frac{1}{\sqrt{N_0}}\sum_{x;f(x)=f_0}|x\rangle, \quad |\Psi_1\rangle = \frac{1}{\sqrt{N_1}}\sum_{x;f(x)=f_1}|x\rangle$$

with the probabilities $\mathsf{p}_0 = N_0/2^n$ and $\mathsf{p}_1 = N_1/2^n$. In the case of our periodic function, the reduced state operator of the input register is

$$\rho_{\rm in} = \frac{1}{2^n}\sum_{i=0}^{r-1}\sum_{k_i,k_j=0}^{K_i-1}|x_i+k_ir\rangle\langle x_i+k_jr|.$$

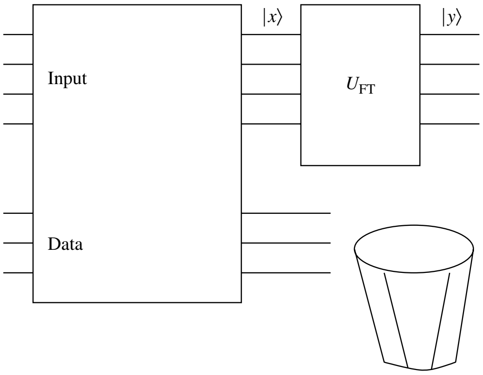


Figure 5.10 Schematic depiction of calculation of the period. The qubits of the output register are discarded.

The state vector (5.40) corresponds in (5.26) to the choice $f(x) = 1/\sqrt{K}$ if x has the form $x_0 + kr$ and $f(x) = 0$ otherwise. According to (5.27), the amplitude $a(\Phi_0 \to y)$, where $|\Phi_0\rangle = U_{\mathrm{FT}}|\Psi_0\rangle$, then is

$$a(\Phi_0 \to y) = \frac{1}{2^{n/2}} \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} \mathrm{e}^{2\mathrm{i}\pi y(x_0+kr)/2^n}, \tag{5.42}$$

and the probability of measuring the value y (that is, of finding the state $|y_{n-1} \cdots y_1 y_0\rangle$ of the computational basis at the exit from the box U_{FT}) is

$$\mathsf{p}(y) = |a(\Phi_0 \to y)|^2 = \frac{1}{2^n K} \left| \sum_{k=0}^{K-1} \mathrm{e}^{2\mathrm{i}\pi k r y/2^n} \right|^2. \tag{5.43}$$

We observe that $\mathsf{p}(y)$ is independent of x_0, and we could have started from any of the vectors $|\Psi_i\rangle$ of (5.41). We next use the geometric series[8]

$$\sum_{k=0}^{K-1} \mathrm{e}^{2\mathrm{i}\pi y k r/2^n} = \frac{1 - \mathrm{e}^{2\mathrm{i}\pi y K r/2^n}}{1 - \mathrm{e}^{2\mathrm{i}\pi y r/2^n}} = \mathrm{e}^{\mathrm{i}\pi(K-1)r/2^n} \frac{\sin(\pi y K r/2^n)}{\sin(\pi y r/2^n)}.$$

In the exceptional case that $2^n/r$ is an integer, and therefore $2^n/r = K$, we would find (recall that y is an integer)

$$\begin{aligned} \mathsf{p}(y) = \frac{1}{2^n K} \frac{\sin^2(\pi y)}{\sin^2(\pi y/K)} &= \frac{1}{r} \quad \text{if} \quad y = jK \\ &= 0 \quad \text{otherwise,} \end{aligned}$$

where j is an integer. We then find that $j/r = y/2^n$, which gives j and r if we are lucky and j/r happens to be an irreducible fraction. We cast $y/2^n$ into its irreducible form j_0/r_0, and then $j = j_0$, $r = r_0$. It is clear that the desired result has been obtained from constructive interference. In the general case we can write, always with integer j (but noninteger $2^n/r$!),

$$y_j = j\frac{2^n}{r} + \delta_j, \tag{5.44}$$

which gives the probability $\mathsf{p}(y_j)$:

$$\mathsf{p}(y_j) = \frac{1}{2^n K} \frac{\sin^2(\pi\delta_j K r/2^n)}{\sin^2(\pi\delta_j r/2^n)}. \tag{5.45}$$

In general, the function $\mathsf{p}(y)$ has sharp maxima when the value of y is close to $j2^n/r$. Using the bounds on $\sin x$,

$$\frac{2}{\pi}x \le \sin x \le x, \qquad 0 \le x \le \frac{\pi}{2},$$

[8] The problem is reminiscent of that of diffraction, for example, neutron diffraction by a crystal. If a is the distance between two sites ($a = 1$ in the text), the lattice cell is ra. The (quasi-) wavevector q can take the values $q = 2\pi p/2^n a$, $p = 0, 1, \ldots, 2^n - 1$ (p and $p' = p + 2^n$ are equivalent). Diffraction peaks are produced when q is an integer multiple of the reciprocal lattice cell $2\pi/ra$, or $q = j2\pi/ra$, $j = 0, 1, \ldots, r-1$.

we can show that the probability of reaching one of the values (5.44) if we require $|\delta_j| < 1/2$ is at least $4/\pi^2$:

$$\mathsf{p}(y_j) \geq \frac{4}{\pi^2}\frac{K}{2^n} \simeq \frac{4}{\pi^2}\frac{1}{r}.$$

Since $0 \leq j \leq r-1$ and $r \gg 1$, there is at least a 40% ($4/\pi^2 \simeq 0.406$) chance of finding one value of y_j close to $j2^n/r$. More precisely (see Exercise 5.10.3 for a specific example),

$$\left| y_j - j\frac{2^n}{r} \right| \leq \frac{1}{2}.$$

Since n and y_j are known (y_j is an integer, $0 \leq y_j \leq 2^n - 1$, the result of measuring the input register), we therefore have an estimate of the fraction j/r. Let us now show that measurement of y_j allows j and r to be determined (always with at least 40% probability). We let y_j vary by one unit, which gives

$$\left| (y_j \pm 1) - j\frac{2^n}{r} \right| \geq \frac{1}{2},$$

in contradiction with the preceding equation, and the (integer) value of y_j is determined by the condition $|\delta_j| < 1/2$. Owing to our choice $2^n > N^2$, which implies that $2^n > r^2$, we have obtained an estimate of j/r which differs from the exact value by less than $1/2r^2$:

$$\left| \frac{y_j}{2^n} - \frac{j}{r} \right| \leq \frac{1}{2^{n+1}}. \tag{5.46}$$

Since $r < N$, and since two fractions of denominator $\geq r$ must differ by at least $1/r^2$ unless they are identical,[9] we will then have a unique value of the fraction j/r. The value of j/r can be determined from the known value of $y_j/2^n$ by expansion in continued fractions, which gives the value of j/r as an irreducible fraction j_0/r_0. If we are lucky and j and r have no common factor, we will immediately obtain the value $r = r_0$ for r. The probability that two large numbers have no common factor is at least[10] 60%, and, with a probability $\sim 0.4 \times 0.6$ or once in four times, the method will directly give the period $r = r_0$, as can be

[9] Because

$$\left| \frac{n}{m} - \frac{p}{q} \right| \geq \frac{1}{mq}$$

unless the two fractions are identical.

[10] There is one chance in two that a number is divisible by 2, one in three that it is divisible by 3, ..., one in p that it is divisible by p, ... The probability for two large numbers to be divisible simultaneously by p is $1/p^2$, and the probability that they have no common factor is

$$\prod_{p=2}^{\infty}\left(1 - \frac{1}{p^2}\right) = \frac{1}{\zeta(2)} = \frac{6}{\pi^2}.$$

verified using a classical computer by comparing $f(x)$ and $f(x+r_0)$. If $f(x) \neq f(x+r_0)$, we can try out the first multiples of $r_0 : 2r_0, 3r_0, \ldots$, and if these trials give nothing, this indicates that the value of y_j was probably outside the interval $|\delta_j| \leq 1/2$. It is then necessary to repeat the entire procedure and measure another value for y_j. This procedure takes $\mathcal{O}(n^3)$ elementary operations, $\mathcal{O}(n^2)$ for the Fourier transform and $\mathcal{O}(n)$ for the calculation of b^x.

Determination of the period r is sufficient to crack the RSA code. In effect (Box 2.2), Eve has at her disposal the message encoded by Alice, b, and the numbers N and c, which are available publicly. She calculates d' as $cd' \equiv 1 \bmod r$ and then $b^{d'} \bmod N$:

$$b^{d'} = a^{cd'} = a^{1+mr} = a(a^r)^m \equiv a \bmod N,$$

because $a^r \equiv 1 \bmod N$ (Box 5.3), and Eve then recovers the original message a.

Box 5.3: The mathematics of RSA encryption

Let N be an integer and G_N be the set of integers $<N$ which have no common factor with N. If $a \in G_N$, then a and N have no common factor. G_N is closed under mod N multiplication, because if $a, b \in G_N$, then $ab \bmod N \in G_N$. In fact, the product ab can be written as

$$ab = x + qN, \qquad ab \equiv x \bmod N,$$

where x cannot have a common factor with N. If x had a common factor s with N, it would be possible to write

$$ab = s(x' + qN/s)$$

and ab would then have s as a factor, which is impossible. Furthermore, if $a, b, c \in G_N$ and $ab \equiv ac \bmod N$, then

$$a(b-c) = pN.$$

Since a has no common factor with N, $(b-c)$ must be a multiple of N, and because $b, c < N$, this means that $b = c$. As a result, if $b \neq c$, $ab \bmod N$ and $ac \bmod N$ will be different and the multiplication of the elements of G_N by a is a simple permutation of these elements. Since $1 \in G_N$, it follows that a possesses an inverse d in G_N, $ad \equiv 1 \bmod N$, and G_N is therefore a group. The order k of an element a of G_N is the smallest integer k such that $a^k \equiv 1 \bmod N$; the integer k is a divisor[11] of the order (the number of elements) of G_N. If N is a prime, the order of G_N is $(N-1)$, and then $\forall a < N$, and therefore also for any a not divisible by N

$$a^{N-1} \equiv 1 \bmod N,$$

[11] The order of a subgroup is a divisor of the order of the group: $1, a, a^2, \ldots, a^{k-1}$ form a subgroup of G_N.

because $(N-1)$ is a multiple of k. Let us take two primes p and q and an integer a which is divisible by neither p nor q; a^{q-1} is not divisible by p, so

$$[a^{q-1}]^{(p-1)} \equiv 1 \bmod p$$

and similarly a^{p-1} is not divisible by q

$$[a^{p-1}]^{(q-1)} \equiv 1 \bmod q,$$

that is,

$$a^{(p-1)(q-1)} = 1 + mp, \qquad a^{(p-1)(q-1)} = 1 + nq,$$

which implies that $mp = nq$ and so

$$a^{(p-1)(q-1)} = 1 + kpq, \qquad a^{(p-1)(q-1)} \equiv 1 \bmod pq.$$

We then deduce that

$$a^{1+s(p-1)(q-1)} \equiv a \bmod pq.$$

This last relation is valid whatever a, that is, even if a can be divided by p (or q), as the reader can easily check.

If c does not have a common factor with $(p-1)(q-1)$, then it has an inverse d in $G_{(p-1)(q-1)}$:

$$cd \equiv 1 \bmod (p-1)(q-1), \qquad cd = 1 + s(p-1)(q-1),$$

and if $b \equiv a^c \bmod pq$, then

$$b^d = a^{cd} \equiv a \bmod pq,$$

which gives the formula on which RSA encryption is based (Box 2.3).
The subgroups generated by a and b are the same because $b = a^c$. Let r be the order of this subgroup. We may assume that the integers $a, b \in G_{pq}$;[12] then r must be a divisor of $(p-1)(q-1)$, but since c has no common factor with $(p-1)(q-1)$ it cannot have a common factor with r. As a result, $c \in G_r$ and there exists a d' such that

$$cd' \equiv 1 \bmod r.$$

Since Eve knows r, she can calculate d' using $cd' \equiv 1 \bmod r$ and then $b^{d'} \bmod N$:

$$b^{d'} = a^{cd'} = a^{1+mr} = a(a^r)^m \equiv a \bmod N,$$

because $a^r \equiv 1 \bmod N$, and so Eve recovers the original message a.

[12] If this is not the case so that, for example, b and N have a common divisor (which is unlikely as b and N are very large numbers), Eve could first compute $\gcd(b, N)$. Then she would have directly factored N.

If we wish in addition to factorize N we must write

(i)

$$a^r - 1 = \left(a^{r/2} - 1\right)\left(a^{r/2} + 1\right) \equiv 0 \bmod N,$$

(ii)

$$a^{r/2} \not\equiv \pm 1 \bmod N.$$

If we are lucky, that is, if the two factors in (i) are integers and (ii) is satisfied, then the product of integers

$$\left(a^{r/2} - 1\right)\left(a^{r/2} + 1\right)$$

is divisible by $N = pq$. It is therefore necessary that p divides $\left(a^{r/2} - 1\right)$ and q divides $\left(a^{r/2} + 1\right)$ or vice versa. The values of p and q are obtained by seeking the greatest common divisors (gcds):

$$p = \gcd\left(N, a^{r/2} - 1\right), \qquad q = \gcd\left(N, a^{r/2} + 1\right).$$

If we are unlucky, we must start over, but the probability of success is greater than 50%. It is worth noting that the algorithm we have just described is a *probabilistic algorithm*: it does not work every time, but it has a good chance of working, and we can be sure that it will work after a small number of tries.

5.9 Classical algorithms and quantum algorithms

The theory of quantum algorithms raises doubts about some of the statements of the theory of classical algorithms when the subject of *algorithmic complexity* is considered, i.e., when we ask what resources are needed to perform a calculation. A general idea is that certain problems can be solved in a number of steps $\mathcal{N}$ which is a polynomial in the number of bits n that measures the size of the problem. For example, if we wish to multiply two numbers n binary digits long, the number of instructions needed is a polynomial in n. A much less trivial example is that of finding primes: how many steps are needed to show that a number is a prime? In 2002 it was proved that this problem is polynomial. However, experience suggests that other problems require a number of computational steps which grows more rapidly than any power of n for $n \gg 1$, for example, as $\exp n$, $\exp(n^{1/3})$, or $n^{\ln n}$. Such problems are often, somewhat inaccurately, termed "exponential."

Turing defined a class of machines, now known as *Turing machines*, which made it possible to study the concept of the complexity of a computational algorithm. He showed that there exist machines, called universal machines (and he also proposed a design for one), which are capable of simulating any other Turing machine. Since then it has been discovered that all computational models proposed for executing programs can be simulated by a Turing machine in a computational time which is a polynomial in the computational time of the simulated machine. This result suggested the following generalization. All machines are equivalent as regards the computational time (or the number of computational steps), up to a polynomial. If this idea is correct, the exponential or polynomial nature of a problem is preserved in going from one computational model to another, which leads to the idea of precisely defining the algorithmic complexity of a problem in terms of the number of instructions $\mathcal{N}$ required by a Turing machine to solve the problem. If $\mathcal{N}$ is a polynomial in n the problem is termed "tractable," and if $\mathcal{N}$ grows faster than any polynomial in n the problem is termed "intractable". The addition of two n-digit numbers is a tractable problem, and the factorization of a number into primes is believed to be intractable, although there is no formal proof of this. Two important complexity classes are the **P** class, the class of problems which are tractable, and the **NP** class, that of problems whose solution, if one can be found, can be *checked* in a polynomial time.[13] **NP** stands for "nondeterministic polynomial," which means that the corresponding class of problems can be solved using a branching algorithm, with instructions "go to both 1 and 2." The number of branches grows exponentially, which is the reason why finding the solution requires a nonpolynomial time, while exploring a single branch to check a solution requires only polynomial time. Naturally, $\mathbf{P} \subset \mathbf{NP}$, and there exists the celebrated conjecture $\mathbf{P} \neq \mathbf{NP}$, which to this day remains unproven. Numerous complexity classes have been identified using the definition based on the computational model of Turing machines, but independent of the actual model provided that the model can be simulated in a polynomial time by a Turing machine. In particular, one has identified **NP complete** problems, such as the traveling salesman problem: finding a polynomial algorithm for one of the **NP complete** problems would automatically imply a polynomial solution for all **NP** problems.

Up to now we have only discussed calculable problems. The *Church–Turing thesis*, which is universally agreed upon but is by its nature impossible to prove, states that *the class of functions which can be calculated by a Turing machine corresponds exactly to the class of functions which one would naturally consider to be calculable using an algorithm.* There exist properly identified problems which are not calculable, for which it is known that no algorithm exists. An example is the halting problem of a Turing machine: the function which associates with any program run on a Turing machine (a finite series of symbols) a 0 or a 1 according to whether the machine stops or does not stop is not a calculable function. Quantum computers also seem to be covered by the Church–Turing

[13] For example, it is intractable to decompose a number into primes, but it is tractable to check the product if the primes are known.

thesis: the functions which quantum computers can calculate are *a priori* the same as those calculable by Turing machines.

We have stated that the simulation of any model of a classical algorithm can be done up to a polynomial time on a Turing machine, and this result has become a sort of basic "axiom" of the theory of algorithmic complexity. This is the strong version of the Church–Turing thesis, which is stated as follows: *any computational model can be simulated on a probabilistic Turing machine with at most a polynomial increase in the number of computational steps*. Quantum computers are important in that they make this strong version questionable. In fact, if factorization is an intractable problem (as suggested by experience but unproven), then the Shor algorithm contradicts this strong version. Using a quantum computer it is possible to decompose a number into primes by a number of steps which is a polynomial in n, whereas a classical computer can only do this in an exponential number of steps. The power of the quantum algorithms is due to the fact that they can explore at the same time all the branches of a nondeterministic algorithm. As we have seen in the case of the Shor algorithm, it is a constructive interference of the different branches which allows us to select the right result. Unfortunately, factorization is not an **NP complete** problem, so that its polynomial solution does not imply that of all **NP** problems.

5.10 Exercises

5.10.1 Justification of the circuits of Fig. 5.4

1. Justify the upper circuit of Fig. 5.4. Show that the action of the cNOT gate on the tensor product

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$$

gives an entangled state.

2. Let us assume that qubits are measured immediately *after* a cU gate. Show that the probabilities of finding the target qubit in the state $|0\rangle$ or $|1\rangle$ and its final states are the same as if the control bit were measured *before* the gate and the target bit were transformed or not according to whether the control bit was in the state $|0\rangle$ or $|1\rangle$. This observation allows the two-qubit gate cU to be replaced by a one-qubit gate acting on the target bit, which is an enormous technical simplification. However, it works only at the end of a calculation, not for an intermediate cU gate!

3. Show that the lower circuit of Fig. 5.4 constructs the Toffoli gate, with $U = \sqrt{\sigma_x}$.

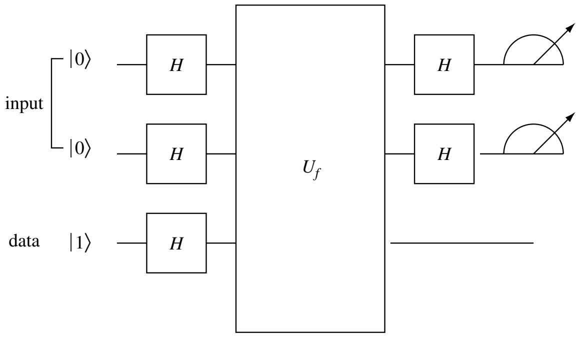


Figure 5.11 The Deutsch algorithm. The two qubits of the input register are initially in the state $|0\rangle$ and the qubit of the output register is in the state $|1\rangle$.

5.10.2 The Deutsch–Josza algorithm

The Deutsch algorithm (Section 5.4) can be generalized to the case where the input register contains two qubits and the output register contains only one qubit (Fig. 5.11). The two qubits of the input register are initially in the state $|0\rangle$, the qubit of the output register is in the state $|1\rangle$, and H is the Hadamard operator:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

The unknown function $f(x)$ is either

(i) $f(x) = \text{constant}$, or
(ii) $f(x) = x \bmod 2$.

1. Show that the global state vector $|\Psi\rangle$ before entering the box U_f is

$$|\Psi\rangle = \left(\frac{1}{2}\left[|0\otimes 0\rangle + |0\otimes 1\rangle + |1\otimes 0\rangle + |1\otimes 1\rangle\right]\right)\otimes\left(\frac{1}{\sqrt{2}}\left[|0\rangle - |1\rangle\right]\right),$$

where the quantity in the first set of parentheses is the state vector of the two qubits of the input register and that in the second is the state vector of the qubit of the output register.

2. Let us recall the action of the box U_f (where x is the input register and y is the output register):

$$U_f|x\otimes y\rangle = |x\otimes[y\oplus f(x)]\rangle,$$

where $\oplus$ is mod 2 addition. Write down the state vector $U_f|\Psi\rangle$ in the cases (i) and (ii).

3. Write down the final state vector $|\Phi\rangle$ of the qubits of the input register (that is, *after* application of the operator $H \otimes H$) in cases (i) and (ii). Show that measurement of these qubits allows the cases (i) and (ii) to be distinguished.

5.10.3 Grover algorithm and constructive interference

Let us write the state vector after application of the Hadamard gates in Fig. 5.8

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_x a_x^{(0)} |x\rangle \quad a_x^{(0)} = 1.$$

Show that the application of the operator GO gives

$$GO|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_x a_x^{(1)} |x\rangle$$

with

$$a_x^{(1)} = \frac{2}{N}\left(\sum_y (-1)^{f(y)} a_y^{(0)}\right) - (-1)^{f(x)} a_x^{(0)}.$$

Take, for example, $N = 16$ and show that $a_x^{(1)} = 3/4$ if $x \neq x_0$ and $a_x^{(1)} = 11/4$ if $x = x_0$. Constructive interference increases the probability for finding the final qubit state corresponding to $x = x_0$. Repeated application of $GO\ \sqrt{N}$ times allows the solution to stand out against a very small background.

5.10.4 Example of finding y_j

Let us take the example of Box 2.3, which shows that a possible period is $r = 3$. We choose $n = 4$, $2^n - 1 = 15$. What are the values of y_j such that $|\delta_j| < 1/2$ in (5.44)? Calculate the corresponding probability $\mathsf{p}(y_j)$ and show that the sum of these probabilities is greater than 40%.

5.11 Further reading

A popularized approach to quantum computing can be found in Johnson (2003). A great deal of information about circuits and quantum algorithms can be found in Nielsen and Chuang (2000), Chapter 5, Stolze and Suter (2004), Chapters 3, 5 and 8, Preskill (1999), Chapter 6, and Mermin (2003). The Shor algorithm is discussed in detail in Ekert and Josza (1996). Interesting references on general problems in information theory are Bennett (1987) and Landauer (1991a) and (1991b).