6 Quantum search algorithms

Suppose you are given a map containing many cities, and wish to determine the shortest route passing through all cities on the map. A simple algorithm to find this route is to search all possible routes through the cities, keeping a running record of which route has the shortest length. On a classical computer, if there are N possible routes, it obviously takes $O(N)$ operations to determine the shortest route using this method. Remarkably, there is a *quantum search algorithm*, sometimes known as *Grover's algorithm*, which enables this search method to be sped up substantially, requiring only $O(\sqrt{N})$ operations. Moreover, the quantum search algorithm is *general* in the sense that it can be applied far beyond the route-finding example just described to speed up many (though not all) classical algorithms that use search heuristics.

In this chapter we explain the fast quantum search algorithm. The basic algorithm is described in Section 6.1. In Section 6.2 we derive the algorithm from another point of view, based on the quantum simulation algorithm of Section 4.7. Three important applications of this algorithm are also described: quantum counting in Section 6.3, speedup of solution of **NP**-complete problems in Section 6.4, and search of unstructured databases in Section 6.5. One might hope to improve upon the search algorithm to do even better than a square root speedup but, as we show in Section 6.6, it turns out this is not possible. We conclude in Section 6.7 by showing that this speed limit applies to most unstructured problems.

6.1 The quantum search algorithm

Let us begin by setting the stage for the search algorithm in terms of an *oracle*, similar to that encountered in Section 3.1.1. This allows us to present a very general description of the search procedure, and a geometric way to visualize its action and see how it performs.

6.1.1 The oracle

Suppose we wish to search through a search space of N elements. Rather than search the elements directly, we concentrate on the *index* to those elements, which is just a number in the range 0 to $N-1$. For convenience we assume $N = 2^n$, so the index can be stored in n bits, and that the search problem has exactly M solutions, with $1 \leq M \leq N$. A particular instance of the search problem can conveniently be represented by a function f, which takes as input an integer x, in the range 0 to $N-1$. By definition, $f(x) = 1$ if x is a solution to the search problem, and $f(x) = 0$ if x is not a solution to the search problem.

Suppose we are supplied with a quantum *oracle* – a black box whose internal workings we discuss later, but which are not important at this stage – with the ability to *recognize* solutions to the search problem. This recognition is signalled by making use of an *oracle*

qubit. More precisely, the oracle is a unitary operator, O, defined by its action on the computational basis:

$$|x\rangle|q\rangle \xrightarrow{O} |x\rangle|q \oplus f(x)\rangle\,, \tag{6.1}$$

where $|x\rangle$ is the index register, $\oplus$ denotes addition modulo 2, and the oracle qubit $|q\rangle$ is a single qubit which is flipped if $f(x) = 1$, and is unchanged otherwise. We can check whether x is a solution to our search problem by preparing $|x\rangle|0\rangle$, applying the oracle, and checking to see if the oracle qubit has been flipped to $|1\rangle$.

In the quantum search algorithm it is useful to apply the oracle with the oracle qubit initially in the state $(|0\rangle - |1\rangle)/\sqrt{2}$, just as was done in the Deutsch–Jozsa algorithm of Section 1.4.4. If x is not a solution to the search problem, applying the oracle to the state $|x\rangle(|0\rangle - |1\rangle)/\sqrt{2}$ does not change the state. On the other hand, if x is a solution to the search problem, then $|0\rangle$ and $|1\rangle$ are interchanged by the action of the oracle, giving a final state $-|x\rangle(|0\rangle - |1\rangle)/\sqrt{2}$. The action of the oracle is thus:

$$|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \xrightarrow{O} (-1)^{f(x)}|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\,. \tag{6.2}$$

Notice that the state of the oracle qubit is not changed. It turns out that this remains $(|0\rangle - |1\rangle)/\sqrt{2}$ throughout the quantum search algorithm, and can therefore be omitted from further discussion of the algorithm, simplifying our description.

With this convention, the action of the oracle may be written:

$$|x\rangle \xrightarrow{O} (-1)^{f(x)}|x\rangle\,. \tag{6.3}$$

We say that the oracle *marks* the solutions to the search problem, by shifting the phase of the solution. For an N item search problem with M solutions, it turns out that we need only apply the search oracle $O(\sqrt{N/M})$ times in order to obtain a solution, on a quantum computer.

This discussion of the oracle without describing how it works in practice is rather abstract, and perhaps even puzzling. It seems as though the oracle already *knows* the answer to the search problem; what possible use could it be to have a quantum search algorithm based upon such oracle consultations?! The answer is that there is a distinction between *knowing* the solution to a search problem, and being able to *recognize* the solution; the crucial point is that it is possible to do the latter without necessarily being able to do the former.

A simple example to illustrate this is the problem of factoring. Suppose we have been given a large number, m, and told that it is a product of two primes, p and q – the same sort of situation as arises in trying to break the RSA public key cryptosystem (Appendix 5). To determine p and q, the obvious method on a classical computer is to *search* all numbers from 2 through $m^{1/2}$ for the smaller of the two prime factors. That is, we successively do a trial division of m by each number in the range 2 to $m^{1/2}$, until we find the smaller prime factor. The other prime factor can then be found by dividing m by the smaller prime. Obviously, this search-based method requires roughly $m^{1/2}$ trial divisions to find a factor on a classical computer.

The quantum search algorithm can be used to speed up this process. By definition, the action of the oracle upon input of the state $|x\rangle$ is to divide m by x, and check to see if the division is exact, flipping the oracle qubit if this is so. Applying the quantum search algorithm with this oracle yields the smaller of the two prime factors with high probability.

But to make the algorithm work, we need to construct an efficient circuit implementing the oracle. How to do this is an exercise in the techniques of reversible computation. We begin by defining the function $f(x) \equiv 1$ if x divides m, and $f(x) = 0$ otherwise; $f(x)$ tells us whether the trial division is successful or not. Using the techniques of reversible computation discussed in Section 3.2.5, construct a classical reversible circuit which takes (x, q) – representing an input register initially set to x and a one bit output register initially set to q – to $(x, q \oplus f(x))$, by modifying the usual (irreversible) classical circuit for doing trial division. The resource cost of this reversible circuit is the same to within a factor two as the irreversible classical circuit used for trial division, and therefore we regard the two circuits as consuming essentially the same resources. Furthermore, the classical reversible circuit can be immediately translated into a quantum circuit that takes $|x\rangle|q\rangle$ to $|x\rangle|q \oplus f(x)\rangle$, as required of the oracle. The key point is that *even without knowing the prime factors of m, we can explicitly construct an oracle which recognizes a solution to the search problem when it sees one*. Using this oracle and the quantum search algorithm we can search the range 2 to $m^{1/2}$ using $O(m^{1/4})$ oracle consultations. That is, we need only perform the trial division roughly $m^{1/4}$ times, instead of $m^{1/2}$ times, as with the classical algorithm!

The factoring example is conceptually interesting but not practical: there are classical algorithms for factoring which work much faster than searching through all possible divisors. However, it illustrates the general way in which the quantum search algorithm may be applied: classical algorithms which rely on search-based techniques may be sped up using the quantum search algorithm. Later in this chapter we examine scenarios where the quantum search algorithm offers a genuinely useful aid in speeding up the solution of **NP**-complete problems.

6.1.2 The procedure

Schematically, the search algorithm operates as shown in Figure 6.1. The algorithm proper makes use of a single n qubit register. The internal workings of the oracle, including the possibility of it needing extra work qubits, are not important to the description of the quantum search algorithm proper. The goal of the algorithm is to find a solution to the search problem, using the smallest possible number of applications of the oracle.

The algorithm begins with the computer in the state $|0\rangle^{\otimes n}$. The Hadamard transform is used to put the computer in the equal superposition state,

$$|\psi\rangle = \frac{1}{N^{1/2}} \sum_{x=0}^{N-1} |x\rangle \, . \tag{6.4}$$

The quantum search algorithm then consists of repeated application of a quantum subroutine, know as the *Grover iteration* or *Grover operator*, which we denote G. The Grover iteration, whose quantum circuit is illustrated in Figure 6.2, may be broken up into four steps:

(1) Apply the oracle O.
(2) Apply the Hadamard transform $H^{\otimes n}$.
(3) Perform a conditional phase shift on the computer, with every computational basis state except $|0\rangle$ receiving a phase shift of -1,

$$|x\rangle \rightarrow -(-1)^{\delta_{x0}}|x\rangle. \tag{6.5}$$

(4) Apply the Hadamard transform $H^{\otimes n}$.

Exercise 6.1: Show that the unitary operator corresponding to the phase shift in the Grover iteration is $2|0\rangle\langle 0| - I$.

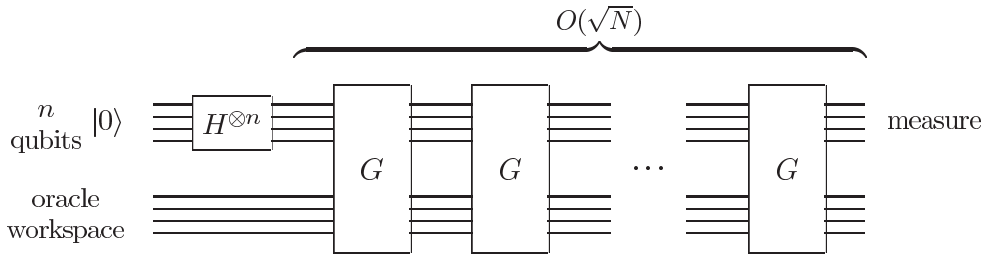

Figure 6.1. Schematic circuit for the quantum search algorithm. The oracle may employ work qubits for its implementation, but the analysis of the quantum search algorithm involves only the n qubit register.

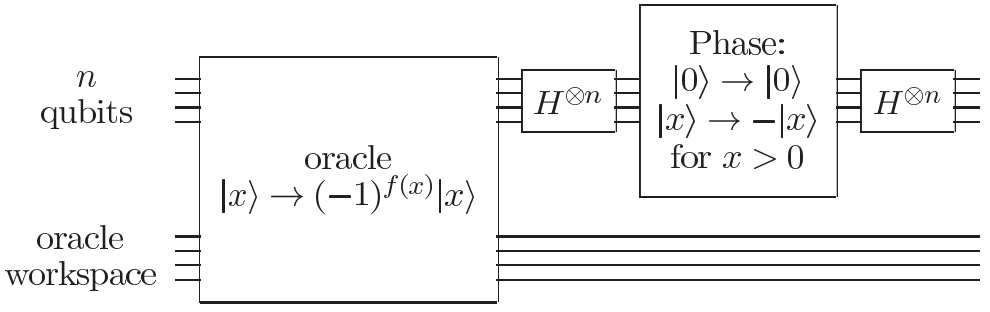

Figure 6.2. Circuit for the Grover iteration, G.

Each of the operations in the Grover iteration may be efficiently implemented on a quantum computer. Steps 2 and 4, the Hadamard transforms, require $n = \log(N)$ operations each. Step 3, the conditional phase shift, may be implemented using the techniques of Section 4.3, using $O(n)$ gates. The cost of the oracle call depends upon the specific application; for now, we merely need note that the Grover iteration requires only a single oracle call. It is useful to note that the combined effect of steps 2, 3, and 4 is

$$H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I\,, \tag{6.6}$$

where $|\psi\rangle$ is the equally weighted superposition of states, (6.4). Thus the Grover iteration, G, may be written $G = (2|\psi\rangle\langle\psi| - I)O$.

Exercise 6.2: Show that the operation $(2|\psi\rangle\langle\psi| - I)$ applied to a general state $\sum_k \alpha_k|k\rangle$ produces

$$\sum_k \left[-\alpha_k + 2\langle\alpha\rangle\right]|k\rangle\,, \tag{6.7}$$

where $\langle\alpha\rangle \equiv \sum_k \alpha_k/N$ is the mean value of the α_k. For this reason, $(2|\psi\rangle\langle\psi| - I)$ is sometimes referred to as the *inversion about mean* operation.

6.1.3 Geometric visualization

What does the Grover iteration do? We have noted that $G = (2|\psi\rangle\langle\psi| - I)O$. In fact, we will show that the Grover iteration can be regarded as a *rotation* in the two-dimensional space spanned by the starting vector $|\psi\rangle$ and the state consisting of a uniform superposition of solutions to the search problem. To see this it is useful to adopt the convention that $\sum_x'$ indicates a sum over all x which are solutions to the search problem, and $\sum_x''$ indicates a sum over all x which are not solutions to the search problem. Define normalized states

$$|\alpha\rangle \equiv \frac{1}{\sqrt{N-M}} \sum_x{}'' |x\rangle \tag{6.8}$$

$$|\beta\rangle \equiv \frac{1}{\sqrt{M}} \sum_x{}' |x\rangle \,. \tag{6.9}$$

Simple algebra shows that the initial state $|\psi\rangle$ may be re-expressed as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle \,, \tag{6.10}$$

so the initial state of the quantum computer is in the space spanned by $|\alpha\rangle$ and $|\beta\rangle$.

The effect of G can be understood in a beautiful way by realizing that the oracle operation O performs a *reflection* about the vector $|\alpha\rangle$ in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$. That is, $O(a|\alpha\rangle + b|\beta\rangle) = a|\alpha\rangle - b|\beta\rangle$. Similarly, $2|\psi\rangle\langle\psi| - I$ also performs a reflection in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$, about the vector $|\psi\rangle$. And the product of two reflections is a rotation! This tells us that the state $G^k|\psi\rangle$ remains in the space spanned by $|\alpha\rangle$ and $|\beta\rangle$ for all k. It also gives us the rotation angle. Let $\cos\theta/2 = \sqrt{(N-M)/N}$, so that $|\psi\rangle = \cos\theta/2|\alpha\rangle + \sin\theta/2|\beta\rangle$. As Figure 6.3 shows, the two reflections which comprise G take $|\psi\rangle$ to

$$G|\psi\rangle = \cos\frac{3\theta}{2}|\alpha\rangle + \sin\frac{3\theta}{2}|\beta\rangle \,, \tag{6.11}$$

so the rotation angle is in fact θ. It follows that continued application of G takes the state to

$$G^k|\psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right)|\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right)|\beta\rangle \,. \tag{6.12}$$

Summarizing, G is a rotation in the two-dimensional space spanned by $|\alpha\rangle$ and $|\beta\rangle$, rotating the space by θ radians per application of G. Repeated application of the Grover iteration rotates the state vector close to $|\beta\rangle$. When this occurs, an observation in the computational basis produces with high probability one of the outcomes superposed in $|\beta\rangle$, that is, a solution to the search problem! An example illustrating the search algorithm with $N = 4$ is given in Box 6.1.

Exercise 6.3: Show that in the $|\alpha\rangle$, $|\beta\rangle$ basis, we may write the Grover iteration as

$$G = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} , \tag{6.13}$$

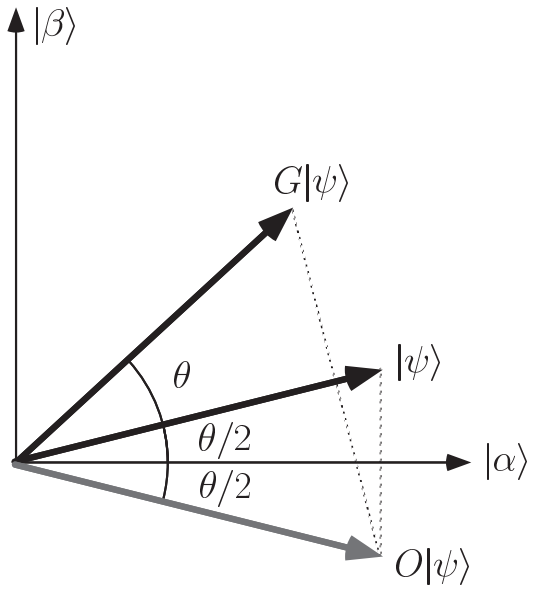

Figure 6.3. The action of a single Grover iteration, G: the state vector is rotated by θ towards the superposition $|\beta\rangle$ of all solutions to the search problem. Initially, it is inclined at angle $\theta/2$ from $|\alpha\rangle$, a state orthogonal to $|\beta\rangle$. An oracle operation O reflects the state about the state $|\alpha\rangle$, then the operation $2|\psi\rangle\langle\psi| - I$ reflects it about $|\psi\rangle$. In the figure $|\alpha\rangle$ and $|\beta\rangle$ are lengthened slightly to reduce clutter (all states should be unit vectors). After repeated Grover iterations, the state vector gets close to $|\beta\rangle$, at which point an observation in the computational basis outputs a solution to the search problem with high probability. The remarkable efficiency of the algorithm occurs because θ behaves like $\Omega(\sqrt{M/N})$, so only $O(\sqrt{N/M})$ applications of G are required to rotate the state vector close to $|\beta\rangle$.

where θ is a real number in the range 0 to $\pi/2$ (assuming for simplicity that $M \leq N/2$; this limitation will be lifted shortly), chosen so that

$$\sin\theta = \frac{2\sqrt{M(N-M)}}{N}. \tag{6.14}$$

6.1.4 Performance

How many times must the Grover iteration be repeated in order to rotate $|\psi\rangle$ near $|\beta\rangle$? The initial state of the system is $|\psi\rangle = \sqrt{(N-M)/N}|\alpha\rangle + \sqrt{M/N}|\beta\rangle$, so rotating through $\arccos\sqrt{M/N}$ radians takes the system to $|\beta\rangle$. Let $\mathrm{CI}(x)$ denote the integer closest to the real number x, where by convention we round halves down, $\mathrm{CI}(3.5) = 3$, for example. Then repeating the Grover iteration

$$R = \mathrm{CI}\left(\frac{\arccos\sqrt{M/N}}{\theta}\right) \tag{6.15}$$

times rotates $|\psi\rangle$ to within an angle $\theta/2 \leq \pi/4$ of $|\beta\rangle$. Observation of the state in the computational basis then yields a solution to the search problem with probability at least one-half. Indeed, for specific values of M and N it is possible to achieve a much higher probability of success. For example, when $M \ll N$ we have $\theta \approx \sin\theta \approx 2\sqrt{M/N}$, and thus the angular error in the final state is at most $\theta/2 \approx \sqrt{M/N}$, giving a probability of error of at most M/N. Note that R depends on the number of solutions M, but not

on the identity of those solutions, so provided we know M we can apply the quantum search algorithm as described. In Section 6.3 we will explain how to remove even the need for a knowledge of M in applying the search algorithm.

The form (6.15) is useful as an exact expression for the number of oracle calls used to perform the quantum search algorithm, but it would be useful to have a simpler expression summarizing the essential behavior of R. To achieve this, note from (6.15) that $R \leq \lceil \pi/2\theta \rceil$, so a lower bound on θ will give an upper bound on R. Assuming for the moment that $M \leq N/2$, we have

$$\frac{\theta}{2} \geq \sin \frac{\theta}{2} = \sqrt{\frac{M}{N}}, \tag{6.16}$$

from which we obtain an elegant upper bound on the number of iterations required,

$$R \leq \left\lceil \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rceil . \tag{6.17}$$

That is, $R = O(\sqrt{N/M})$ Grover iterations (and thus oracle calls) must be performed in order to obtain a solution to the search problem with high probability, a quadratic improvement over the $O(N/M)$ oracle calls required classically. The quantum search algorithm is summarized below, for the case $M = 1$.

Algorithm: Quantum search

Inputs: (1) a black box oracle O which performs the transformation $O|x\rangle|q\rangle = |x\rangle|q \oplus f(x)\rangle$, where $f(x) = 0$ for all $0 \leq x < 2^n$ except x_0, for which $f(x_0) = 1$; (2) $n + 1$ qubits in the state $|0\rangle$.

Outputs: x_0.

Runtime: $O(\sqrt{2^n})$ operations. Succeeds with probability $O(1)$.

Procedure:

1. $|0\rangle^{\otimes n}|0\rangle$ — initial state

2. $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$ — apply $H^{\otimes n}$ to the first n qubits, and HX to the last qubit

3. $\rightarrow \left[(2|\psi\rangle\langle\psi| - I)O \right]^R \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$ — apply the Grover iteration $R \approx \lceil \pi\sqrt{2^n}/4 \rceil$ times.

$\approx |x_0\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$

4. $\rightarrow x_0$ — measure the first n qubits

Exercise 6.4: Give explicit steps for the quantum search algorithm, as above, but for the case of multiple solutions ($1 < M < N/2$).

What happens when more than half the items are solutions to the search problem, that is, $M \geq N/2$? From the expression $\theta = \arcsin(2\sqrt{M(N-M)}/N)$ (compare (6.14)) we see that the angle θ gets smaller as M varies from $N/2$ to N. As a result, the number of iterations needed by the search algorithm *increases* with M, for $M \geq N/2$. Intuitively,

this is a silly property for a search algorithm to have: we expect that it should become easier to find a solution to the problem as the number of solutions increases. There are at least two ways around this problem. If M is known in advance to be larger than $N/2$ then we can just randomly pick an item from the search space, and then check that it is a solution using the oracle. This approach has a success probability at least one-half, and only requires one consultation with the oracle. It has the disadvantage that we may not know the number of solutions M in advance.

In the case where it isn't known whether $M \geq N/2$, another approach can be used. This approach is interesting in its own right, and has a useful application to simplify the analysis of the quantum algorithm for counting the number of solutions to the search problem, as presented in Section 6.3. The idea is to double the number of elements in the search space by adding N extra items to the search space, none of which are solutions. As a consequence, less than half the items in the new search space are solutions. This is effected by adding a single qubit $|q\rangle$ to the search index, doubling the number of items to be searched to $2N$. A new *augmented* oracle O' is constructed which marks an item only if it is a solution to the search problem and the extra bit is set to zero. In Exercise 6.5 you will explain how the oracle O' may be constructed using one call to O. The new search problem has only M solutions out of $2N$ entries, so running the search algorithm with the new oracle O' we see that at most $R = \pi/4\sqrt{2N/M}$ calls to O' are required, and it follows that $O(\sqrt{N/M})$ calls to O are required to perform the search.

Exercise 6.5: Show that the augmented oracle O' may be constructed using one application of O, and elementary quantum gates, using the extra qubit $|q\rangle$.

The quantum search algorithm may be used in a wide variety of ways, some of which will be explored in subsequent sections. The great utility of the algorithm arises because we do not assume any particular structure to the search problems being performed. This is the great advantage of posing the problem in terms of a 'black box' oracle, and we adopt this point of view whenever convenient through the remainder of this chapter. In practical applications, of course, it is necessary to understand how the oracle is being implemented, and in each of the practical problems we concern ourselves with an explicit description of the oracle implementation is given.

Exercise 6.6: Verify that the gates in the dotted box in the second figure of Box 6.1 perform the conditional phase shift operation $2|00\rangle\langle 00| - I$, up to an unimportant global phase factor.

6.2 Quantum search as a quantum simulation

The correctness of the quantum search algorithm is easily verified, but it is by no means obvious how one would dream up such an algorithm from a state of ignorance. In this section we sketch a heuristic means by which one can 'derive' the quantum search algorithm, in the hope of lending some intuition as to the tricky task of quantum algorithm design. As a useful side effect we also obtain a *deterministic* quantum search algorithm. Because our goal is to obtain insight rather than generality, we assume for the sake of simplicity that the search problem has exactly one solution, which we label x.

Our method involves two steps. First, we make a guess as to a *Hamiltonian* which