QUANTUM INFORMATION & COMPUTATION

Nilanjana Datta, DAMTP Cambridge

1 Quantum algorithms for search problems

Searching is a fundamentally important task; many important computational problems can be thought of as searching tasks. For example factoring N can be viewed as a search amongst integers less than N for one that divides N exactly. Before discussing Grover's quantum searching algorithm we introduce the complexity class **NP** which contains many problems of urgent practical interest, and is fundamentally related to the notion of searching.

1.1 The class NP and search problems

We can intuitively think of **NP** as comprising problems that are "hard to solve" (i.e. no poly time algorithm known) but if a solution (or certificate of a solution) is given then its correctness can be "easily verified" (i.e. in poly time). Typically we are faced with a search over an exponentially large space of candidates seeking a "good" candidate, and given any candidate it is easy to check if it is good or not.

Definition of NP
NP ("**n**ondeterministic **p**oly time"):
a language is in **NP** if it has a *poly-time verifier* V. A verifier V for a language L is a computation with two inputs w and c such that:
(i) if $w \in L$ then *for some* c, $V(w, c)$ halts with "accept". Any such 'good' c is called a certificate of membership for w;
(ii) if $w \notin L$ then *for all* c, $V(w, c)$ halts with "reject".
V is a poly-time verifier if for all inputs (w, c) V runs in poly(n) time where n is the size of w. (Note that in this case c need only be poly(n) long too, since any single step of a computation can access only a constant number of new bits).

Intuitively (i) and (ii) say that you can certify membership of L (viz. (i)) in such a way that you cannot be tricked into accepting false w's (viz. (ii)) and checking of certificates can be done quickly/efficiently. Note the asymmetry – we are required to certify only membership, but not non-membership.

Alternative definition of NP
Imagine a computer that operates "nondeterministically" i.e. instead of sequentially implementing the steps of a single algorithm, at each step the computer duplicates itself and branches into two computational paths performing two steps (possibly the same) that are performed simultaneously in parallel (in contrast to a probabilistic choice of one or other step). Thus after m steps we have 2^m computers performing computations

in parallel. We require that all paths eventually halt (with "accept" or "reject") and the running time of this nondeterministic computation is defined to be the length of the longest path.

The computation is defined to:
(i) accept its input if *at least one* path accepts; and
(ii) reject its input if *all* paths reject,
(so all inputs are either accepted or rejected as (ii) is the negation of (i)). Then we have:
Proposition: NP is the class of languages that are decided by a nondeterministic computation with polynomial running time.

[Optional exercise: prove the proposition – given such a nondeterministic computation and input w, what is the verifier and certificate (if $w \in L$)? Conversely, given a verifier, what is the corresponding nondeterministic computation with acceptance conditions as above?]

Note that this notion of computation is "non-physical" for complexity considerations, in the following sense: although we have just a polynomial running time, we generally need to invest an *exponential* amount of physical resources to actually implement it viz. an exponential number of computers all running simultaneously, or alternatively, a single computer with exponential running time - being used to do an exponential number of computations i.e. all the paths, in succession.

The satisfiability problem SAT: given a Boolean formula $\phi(x_1, \ldots, x_n)$ with n variables and single bit output, we want to decide if there is an assignment $x_1 = b_1, \ldots, x_n = b_n$ with $\phi(b_1, \ldots, b_n) = 1$. Any such assignment is called a satisfying assignment for ϕ. A brute force evaluation of all 2^n possible assignments will surely decide this problem but this generally takes exponential ($O(2^n)$) time. More formally, if we encode the formula as a bit string using some specified representation of its basic symbols, each as a bit string, then inputs of size m could have $O(m)$ variables and hence the brute force algorithm runs in exponential time.

It is not known whether SAT is in **P** or not but it is easily seen to be in **NP** – if ϕ is satisfiable then the certificate c is any actual satisfying assignment and the verifier $V(\phi, c)$ simply evaluates $\phi(c)$ to check that it is 1. Clearly if ϕ is unsatisfiable we cannot be tricked into accepting it by this procedure!

Relation to searching: SAT illustrates a fundamental connection between **NP** and search problems – for any $\phi(x_1, \ldots, x_n)$ we have an exponentially large number of candidate assignments (possible certificates) and we want to know if a "good" (satisfying) assignment exists. Although it is not clear how to locate a good candidate "quickly", if we are given any prospective candidate we can check quickly if it is good or not. This is a general feature of very many practical problems e.g. scheduling/timetabling tasks, or more general simultaneous constraint satisfaction problems.

From the definitions we have the series of inclusions $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$ and $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{PSPACE}$, but it is not known whether either of **NP** and **BPP** is contained in the other or not. The most notoriously famous open problem of complexity theory is the question of whether **P** is equal to **NP** or not.

The unstructured search problem

Suppose we are given a large database with N items and we wish to locate a particular item. We assume that the database is entirely unstructured or unsorted but given any item we can easily check whether or not it is the one we seek. Our algorithm should locate the item with some constant level of probability (half say) independent of the size N. Each access to the database is called a query and we normally regard it as one computational step.

For classical computation we may argue that $O(N)$ queries will be necessary and sufficient: the good item has completely unknown location; if we examine an item and find it bad, we gain no further information about the location of the good item (beyond the fact that it is not the current one).

For quantum computation we will see that $O(\sqrt{N})$ queries are sufficient (and in fact that number is necessary too) to locate the good item i.e. we get a *quadratic* speedup over classical search. This speedup does not cross the polynomial vs. exponential divide (the "holy grail" of complexity theory) but it is still viewed as significant in situations where exhaustive search is the best known classical algorithm. At first sight we might have naively expected an exponential quantum speedup here: suppose $N = 2^n$ and recall that a quantum algorithm can easily access 2^n items in superposition (by use of only $n = \log N$ Hadamard operations) so we can look up the "goodness" of *all* items in superposition, with just *one* query! We may then hope that we could manipulate the resulting quantum state to efficiently reveal the good item. But the above-quoted result shows that this hope cannot be realised. Intuitively the ket corresponding to the good item occurs with only an exponentially small amplitude in the total superposition (as do all the bad items), and hence is very difficult to reliably distinguish by any physical process.

Databases are often actually structured, in a way that can facilitate the search. As an example suppose our N items are labelled by the numbers and we seek a particular one labelled k. Unstructured search (requiring $O(N)$ queries) corresponds to the database containing the numbers in some unknown random order. But if the items are structured by being presented in numerical order, then we can locate k with only $O(\log N)$ queries (in fact exactly $1 \times \log N$ queries) using a binary search procedure: each query of a middle item eliminates an entire half of the remaining database. This kind of structured search is common in practice e.g. the lexicographic ordering of names in a large phone book facilitating search for a given person's number. But suppose we were given a person's number and asked to determine their name. Then we would be faced with an essentially unstructured search requiring a lot more time!

In the following we will consider quantum algorithms for only *unstructured* search, in particular Grover's quantum searching algorithm which achieves this search in $O(\sqrt{N})$ queries. The issue of understanding which kinds of structure in a database can provide a good benefit for quantum versus classical computation is still largely open and a topic of current research. (One interesting known result is that in the case of a linearly ordered database (such as the phone book above) any quantum algorithm still requires $O(\log N)$ queries but the actual number of queries now is $k \log N$ with k strictly less than 1).

1.2 Grover's Quantum Searching Algorithm

Reflections and projections in Dirac ket notation
We first recall some elementary constructions from linear algebra (some of which we've already seen near the start of the course) that will be used in the discussion of Grover's algorithm. If $|\alpha\rangle$ is any unit length ket vector then

$$\Pi_{|\alpha\rangle} = |\alpha\rangle \langle\alpha|$$

is the operation of *projection onto* $|\alpha\rangle$, and

$$I_{|\alpha\rangle} = I - 2\,|\alpha\rangle \langle\alpha|$$

(with I denoting the identity operation) is the operation of *reflection in the subspace that is orthogonal to* $|\alpha\rangle$ (i.e. vectors in that subspace are left unchanged and general vectors have their component along $|\alpha\rangle$ reversed in sign). For any unitary operator U it is easy to check that

$$U\Pi_{|\alpha\rangle}U^\dagger = \Pi_{U|\alpha\rangle} \qquad UI_{|\alpha\rangle}U^\dagger = I_{U|\alpha\rangle}. \tag{1}$$

Example.
In the space of a single qubit let $\left|\alpha^\perp\right\rangle$ be any chosen unit vector orthogonal to $|\alpha\rangle$. Then any ket vector may be uniquely expressed as $|v\rangle = x\,|\alpha\rangle + y\left|\alpha^\perp\right\rangle$ and

$$\Pi_{|\alpha\rangle}\,|v\rangle = x\,|\alpha\rangle \qquad I_{|\alpha\rangle}\,|v\rangle = -x\,|\alpha\rangle + y\left|\alpha^\perp\right\rangle$$

so $I_{|\alpha\rangle}$ is reflection in the line defined by $\left|\alpha^\perp\right\rangle$. □

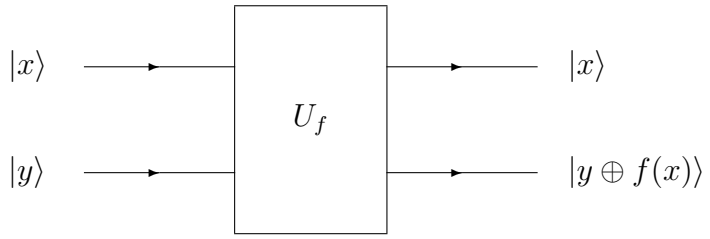
Grover's algorithm for unstructured search

We consider the fundamental problem of unstructured search for a unique item, and describe a quantum algorithm originally due to Lov Grover in 1996 which solves the problem with only $O(\sqrt{N})$ queries. We will give a simple geometrical derivation of the algorithm (different from Grover's original algebraic approach, cf Exercise Sheet 4) which clarifies its workings.

It will be convenient to take the size N of our search space to be a power of 2 *viz.* $N = 2^n$. Thus we can label the entries by bit strings (i.e. strings of 0's and 1's) of length n. Our search problem may then be phrased in terms of a black box promise problem as follows. We will replace the database by a black box which computes an n bit function $f : B_n \to B$. It is promised that $f(x) = 0$ for all n bit strings except exactly one string, denoted x_0 (the "marked" position that we seek) for which $f(x_0) = 1$. Our problem is to determine x_0. As usual we assume that f is given as a unitary transformation U_f on $n + 1$ qubits defined by

$$U_f\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle \tag{2}$$

Here the input register $|x\rangle$ consists of n qubits and the output register $|y\rangle$ consists of a single qubit. The symbol $\oplus$ denotes addition modulo 2. Pictorially we have



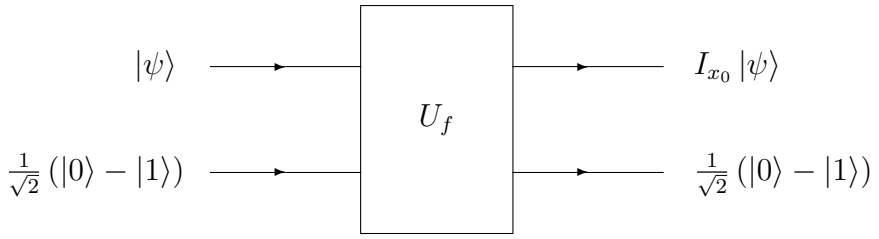
The assumption that the database is *unstructured* is formalised here as the standard oracle idealisation that we have no access to the internal workings of U_f – it operates as a "black box" on the input and output registers, telling us only if the queried item is good or not.

Instead of using U_f we will generally use a closely related operation denoted I_{x_0} on n qubits. It is defined by

$$I_{x_0}|x\rangle = \begin{cases} |x\rangle & \text{if } x \neq x_0 \\ -|x_0\rangle & \text{if } x = x_0 \end{cases} \tag{3}$$

i.e. I_{x_0} simply inverts the amplitude of the $|x_0\rangle$ component and so I_{x_0} is just the reflection operator $I_{|x_0\rangle}$ defined above. If x_0 is the n bit string $00\ldots0$ then I_{x_0} will be written simply as I_0.

A black box which performs I_{x_0} may be simply constructed from U_f by just setting the output register to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then the action of U_f leaves the output register in this state and effects I_{x_0} on the input register. Pictorially



Our searching problem becomes the following: we are given a black box which computes I_{x_0} for some n bit string x_0 and we want to determine the value of x_0 using the least number of queries to the box.

We will work in a space of n qubits with a standard basis $\{|x\rangle\}$ labelled by n-bit strings x. Let $\mathcal{B}_n$ denote the space of all n-qubit states. Let $H_n = H \otimes \ldots \otimes H$ acting on $\mathcal{B}_n$ denote the application of H to each of the n qubits separately.

Grover's quantum searching algorithm operates as follows. Having no initial information about x_0 we begin with the state

$$|\psi_0\rangle = H_n|0\ldots0\rangle = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle \tag{4}$$

which is an equal superposition of all possible x_0 values. Consider the compound operator

Q, called the **Grover iteration operator**, defined by

$$Q = -H_n I_0 H_n I_{x_0}. \tag{5}$$

Note that all amplitudes in $|\psi_0\rangle$ and all matrix elements of Q are *real* numbers so to analyse Q we will be able to use the geometrical interpretations of the projection and reflection operators described above in terms of *real* (rather than complex) Euclidean geometry.

In the next section we will explain the structure of Q and show that it has a simple geometrical interpretation:

(Q1): In the plane $\mathcal{P}(x_0)$ spanned by (the initially unknown) $|x_0\rangle$ and $|\psi_0\rangle$, Q is rotation through angle 2α where $\sin\alpha = \frac{1}{\sqrt{N}}$.

(Q2): In the subspace orthogonal to $\mathcal{P}(x_0)$, $Q = -I$ where I is the identity operation.

Thus by repeatedly applying Q to the starting state $|\psi_0\rangle$ in $\mathcal{P}(x_0)$ we may rotate it around near to $|x_0\rangle$ and then determine x_0 with high probability by a measurement in the standard basis. For large N, $|x_0\rangle$ and $|\psi_0\rangle$ are almost orthogonal and $2\alpha \approx 2\sin\alpha = \frac{2}{\sqrt{N}}$. Thus about $\frac{\pi}{4}\sqrt{N}$ iterations will be needed. Each application of Q uses one evaluation of I_{x_0} and hence of U_f so $O(\sqrt{N})$ evaluations are required, representing a square root speedup over the $O(N)$ evaluations needed for a classical unstructured search. More precisely we have $\langle x_0|\psi_0\rangle = \frac{1}{\sqrt{N}}$ so the number of iterations needed is the integer nearest to $(\arccos\frac{1}{\sqrt{N}})/(2\arcsin\frac{1}{\sqrt{N}})$ (which is independent of x_0).

Example: searching for "one in four".
A simple striking example is the case of $N = 4$ in which $\sin\alpha = \frac{1}{2}$ and Q is a rotation through $\pi/3$. The initial state is $|\psi_0\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$ and for any marked x_0 the angle between $|x_0\rangle$ and $|\psi_0\rangle$ is precisely $\pi/3$ too. Hence after one application of Q i.e. just *one* query, we will learn the position of any single marked item in a set of four with *certainty*! □

The iteration operator Q – reflections and rotations

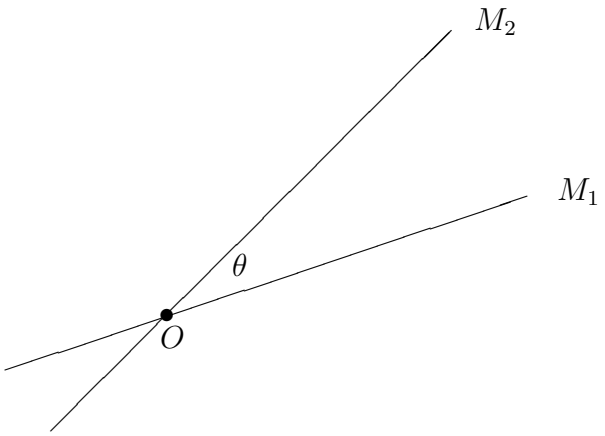
Using eq. (1) (and noting that $H = H^\dagger$) the Grover iteration operator can be written

$$Q = -I_{H_n|0\ldots0\rangle}\, I_{|x_0\rangle} = -I_{|\psi_0\rangle}\, I_{|x_0\rangle}.$$

Now for any $|\alpha\rangle$ and $|v\rangle$ we have $I_{|\alpha\rangle}\,|v\rangle = |v\rangle - 2\langle\alpha|v\rangle\,|\alpha\rangle$ i.e. $|v\rangle$ is modified by a multiple of $|\alpha\rangle$. Hence if $|v\rangle$ is in the (real) plane $\mathcal{P}(x_0)$ spanned by $|x_0\rangle$ and $|\psi_0\rangle = H_n\,|0\ldots0\rangle$ then both $I_{|x_0\rangle}\,|v\rangle$ and $I_{|\psi_0\rangle}\,|v\rangle$ will be in $\mathcal{P}(x_0)$ too – indeed $I_{|x_0\rangle}$ and $I_{|\psi_0\rangle}$ within this plane are just reflections in the lines perpendicular to $|x_0\rangle$ and $|\psi_0\rangle$ respectively. Hence Q also preserves the plane $\mathcal{P}(x_0)$ and its action is given by the following fact of Euclidean geometry.

Lemma: let M_1 and M_2 be two mirror lines in the Euclidean plane $\mathbb{R}^2$ intersecting at a point O and let θ be the angle in the plane from M_1 to M_2 (cf figure below). Then the

operation of reflection in M_1 followed by reflection in M_2 is just (anticlockwise) rotation by angle 2θ about the point O.



Proof of lemma: this is immediate, for example, from standard matrix expressions for rotations and reflections in $\mathbb{R}^2$. $\square$

Using the lemma we see that the action of $I_{H_n|0\rangle} I_{|x_0\rangle} = -Q$ in $\mathcal{P}(x_0)$ is a rotation through 2β where $\cos\beta = \langle x_0| H_n |0\rangle = \frac{1}{\sqrt{N}}$. For large N, $\beta \approx \pi/2$ and we have a rotation of almost π. It would be possible to use this large rotation as the basis of the quantum searching algorithm but we prefer a smaller incremental motion. We could use the operator $(I_{H_n|0\rangle} I_{|x_0\rangle})^2$ but there is another solution, explaining the occurrence of the minus sign in the definition of Q:

Lemma: for any 2 dimensional real v we have

$$-I_v = I_{v^\perp}$$

where $v^\perp$ is a unit vector perpendicular to v.

Proof: For any vector u we write $u = av + bv^\perp$. Then I_v just reverses the sign of a and $-I_v$ reverses the sign of b. Thus the action of $-I_v$ is the same as that of $I_{v^\perp}$. $\square$

Hence $Q = -I_{H_n|0\rangle} I_{|x_0\rangle}$ acting in $\mathcal{P}(x_0)$ is a rotation through 2α where α is the angle between $|x_0\rangle$ and a perpendicular state to $H_n |0\rangle$ i.e. $\sin\alpha = \langle x_0| H_n |0\rangle = \frac{1}{\sqrt{N}}$ as claimed in **(Q1)**.

To see the effect of Q on states orthogonal to $\mathcal{P}(x_0)$ suppose that $|\xi\rangle \in \mathcal{B}_n$ is orthogonal to both $H_n |0\rangle$ and $|x_0\rangle$. Then from the definitions of $I_{|x_0\rangle}$ and $I_{H_n|0\ldots0\rangle}$ we see that $I_{|x_0\rangle} |\xi\rangle = I_{H_n|0\rangle} |\xi\rangle = |\xi\rangle$ so $Q = -I$ in the orthogonal complement to $\mathcal{P}(x_0)$, as claimed in **(Q2)**.

Thus even though x_0 is unknown (but we are given a black box for I_{x_0}) we can construct a rotation operator Q in the plane spanned by the fixed starting state $|\psi_0\rangle$ and the unknown $|x_0\rangle$. Furthermore the angle between the starting state and $|x_0\rangle$ is independent of the value of x_0 (as the starting state is an equal superposition of all possible x_0 values)

so the number of iterations is independent of x_0 too.

1.3 Some further features of Grover's algorithm

Optimality

Grover's algorithm achieves unstructured search for a unique good item with $\frac{\pi}{4}\sqrt{N}$ queries. Is it possible to invent an even more ingenious quantum algorithm that uses fewer queries? Alas the answer is no. We'll just state (without proof):
Theorem Any quantum algorithm that achieves the search for a unique good item in an unstructured database of size N (with any constant level of probability, say half) must use $O(\sqrt{N})$ queries. □

Even more, it can be shown that $\frac{\pi}{4}(1-\epsilon)\sqrt{N}$ queries for any $\epsilon > 0$ are insufficient, so Grover's algorithm is optimal in a tight sense.

Searching with multiple good items

Suppose our search space contains $r \geq 1$ good items and we wish to find any one such item. Consider first the case that r is known. In this case we'll see that our previous algorithm still works; we just need to modify the number of iterations in a way that depends on r.

Let the good items be denoted $x_1, \ldots, x_r$ so now $f(x_i) = 1$ for $i = 1, \ldots, r$ and $f(x) = 0$ for all other x's. Using the same construction that gave I_{x_0} from U_f in the case of a single good item, we obtain the operator I_G (where G stands for "good") with action:

$$I_G |x\rangle = \begin{cases} |x\rangle & \text{if } x \neq x_1, \ldots, x_r \\ -|x\rangle & \text{if } x = x_1, \ldots, x_r \end{cases}$$

and we will use the iteration operator (cf eq. (5))

$$Q_G = -H_n I_0 H_n I_G = -I_{|\psi_0\rangle} I_G.$$

Let

$$|\psi_G\rangle = \frac{1}{\sqrt{r}} \sum_{i=1}^{r} |x_i\rangle$$

be the equal superposition of all good items. We can separate out the good and bad parts of the full equal superposition $|\psi_0\rangle$ writing:

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{\text{all } x} |x\rangle = \frac{\sqrt{r}}{\sqrt{N}} |\psi_G\rangle + \frac{\sqrt{N-r}}{\sqrt{N}} |\psi_B\rangle \tag{6}$$

where $|\psi_B\rangle = \frac{1}{\sqrt{N-r}} \sum_{\text{bad } x} |x\rangle$ is the equal superposition of all bad items and $|\psi_G\rangle$ and $|\psi_B\rangle$ are orthogonal states. Note that we can write $I_G = I - 2\sum_{i=1}^{r} |x_i\rangle \langle x_i| \, x_i$ which (on arbitrary vectors) is not of the form $I_{|\alpha\rangle}$ for any single vector $|\alpha\rangle$. But we still have:

Theorem: let $\mathcal{P}_G$ be the plane spanned by $|\psi_0\rangle$ and $|\psi_G\rangle$. Then the action of Q_G preserves this plane and within $\mathcal{P}_G$ this action is rotation through angle 2α where

$$\sin\alpha = \langle\psi_0|\psi_G\rangle = \sqrt{\frac{r}{N}}.$$

Proof: Clearly $I_{|\psi_0\rangle}$ preserves $\mathcal{P}_G$ since acting on any $|\psi\rangle$ it just subtracts a multiple of $|\psi_0\rangle$. For I_G we note that by eq. (6), $\mathcal{P}_G$ can also be characterised as the plane spanned by the orthogonal states $|\psi_G\rangle$ and $|\psi_B\rangle$. Now $I_G|\psi_G\rangle = -|\psi_G\rangle$ and $I_G|\psi_B\rangle = |\psi_B\rangle$ so for any state $|\psi\rangle = a|\psi_G\rangle + b|\psi_B\rangle$ in $\mathcal{P}_G$ the action of I_G is to subtract a multiple of $|\psi_G\rangle$ i.e. the result lies in the plane too. This also shows that *within* $\mathcal{P}_G$, I_G coincides with the operation $I_{|\psi_G\rangle}$ and $Q_G = -I_{|\psi_0\rangle}I_{|\psi_G\rangle} = I_{|\psi_0^\perp\rangle}I_{|\psi_G\rangle}$. Hence exactly as before, Q is a rotation through angle 2α where α is the angle between $|\psi_0^\perp\rangle$ and $|\psi_G\rangle$ i.e. $\sin\alpha = \langle\psi_0|\psi_G\rangle = \sqrt{r/N}$. $\square$

Now suppose that we start with $|\psi_0\rangle$ and repeatedly apply Q_G. The angle between $|\psi_0\rangle$ and $|\psi_G\rangle$ is β where $\cos\beta = \langle\psi_0|\psi_G\rangle = \sqrt{r/N}$. Each application of Q_G is a rotation through 2α where $\sin\alpha = \sqrt{r/N}$ so we need $\beta/(2\alpha) = (\arccos\sqrt{r/N})/(2\arcsin\sqrt{r/N})$ iterations to move $|\psi_0\rangle$ very close to $|\psi_G\rangle$. If $r << N$ then $|\psi_0\rangle$ and $|\psi_G\rangle$ are almost orthogonal ($\beta \approx \pi/2$) and $\alpha \approx \sin\alpha = \sqrt{r/N}$ so we need $\frac{\pi}{4}\sqrt{N/r}$ iterations.

The basic technique and use of the operator Q_G in the above result can be generalised to give the so-called principle of *amplitude amplification* (which we won't discuss in this course).

Searching with an *unknown* number of good items

Optional, not required for exam purposes.

We can also adapt the algorithm to work in the case that r is *unknown*. The apparent difficulty is the following: if we start with $|\psi_0\rangle$ and repeatedly apply the operator Q (in either case $r = 1$ or $r > 1$) we just rotate the state round and round in the plane of $|\psi_0\rangle$ and $|\psi_G\rangle$. The trick is to know when to stop i.e. when the state lines up closely with $|\psi_G\rangle$ in this plane. But if r is unknown then the rotation angle 2α of Q is unknown!

To illustrate the way around this problem we'll consider only the case where the unknown r is very small $r << N$. (General r values can be addressed by a more complicated argument along similar lines). We choose a number K *uniformly randomly* in the range $0 < K < \frac{\pi}{4}\sqrt{N}$, apply K iterations of Q, measure the final state and test if the result is good or not. For $r << N$ each iteration is a rotation through small angle $2\alpha \approx 2\sqrt{r/N}$ i.e. we have chosen a random angle in the range 0 to $\sqrt{r}\frac{\pi}{2}$ of $\sqrt{r}$ quadrants. Equivalently we can choose one of the $\sqrt{r}$ quadrants at random and then a random angle in it. Now think of $|\psi_0\rangle$ as the x-axis direction and $|\psi_G\rangle$ as the y axis direction (recalling that these states are almost orthogonal for $r << N$). If the final rotation angle is within $\pm 45^\circ$ of the y axis then the final state $|\psi\rangle$ has $|\langle\psi|\psi_G\rangle|^2 \geq \cos^2 45^\circ = 1/2$ i.e. we have probability at least half of seeing a good item in our final measurement. Now for every quadrant, half the angles are within $\pm 45^\circ$ of the y axis so our randomised procedure above, using $O(\sqrt{N})$ queries, will locate a good item with probability at least $1/4$. Repeating the

whole procedure a constant number of times, say $M = 10$ times, thus still using $O(\sqrt{N})$ queries, we will fail to locate a good item only with tiny probability $(3/4)^M = (3/4)^{10}$.

This case of unknown r is directly relevant to the consideration of computational tasks in NP, where rather than *locating* a good item we want instead to know whether a good item *exists* or not. Consider for example the task SAT: given a Boolean function f, does it have a satisfying assignment or not? f will generally have some unknown number $r \geq 0$ of satisfying assignments. We run the above randomised version of Grover's algorithm, say 10 times, checking each output x to see if $f(x) = 1$ or not. If they all fail we conclude that f is not satisfiable, which will be correct with high probability $1 - (3/4)^{10}$. In this way Grover's algorithm can be applied to any NP problem to provide a quadratic speedup over classical exhaustive search.