# Lecture 12: Grover's Algorithm

March 7, 2006

We have completed our study of Shor's factoring algorithm. The basic technique behind Shor's algorithm, which we described in terms of phase estimation, can also be used to solve some other number-theoretic and group-theoretic problems as well (such as computing discrete logarithms and calculating orders of abelian groups). This family of problems represents one of the main reasons why there is such great interest in building quantum computers, as the associated quantum algorithms give an exponential advantage over the best known classical algorithms for these problems.

We will now study a different quantum algorithmic technique. It does not give as impressive an advantage over classical algorithms as we have for Shor's algorithm, but the technique is applicable to a much broader class of problems. The simplest general problem to which this technique can be applied is *search*; and although there are other applications of the technique, this will be the focus of our investigation. The technique was first considered by Lov Grover, and is known as Grover's Algorithm in the context of search.

## Unstructured search

We will be returning to the black-box context that was used to describe Deutsch's algorithm, the Deutsch-Jozsa algorithm, and Simon's algorithm. (The fact that Grover's algorithm works in this very general context represents one of its strengths, because in general there will not need to be any promises on the black box.) Suppose that we have a function

$$f : \{0,1\}^n \to \{0,1\}$$

that is implemented by a reversible transformation $B_f$ in the usual way:

$$B_f |x\rangle |a\rangle = |x\rangle |a \oplus f(x)\rangle$$

for all $x \in \{0,1\}^n$ and $a \in \{0,1\}$ The problem of *search* is simply to find a string $x \in \{0,1\}^n$ such that $f(x) = 1$, or to conclude that no such $x$ exists if $f$ is identically 0).

It is important to note that this searching problem is completely *unstructured*. There are no promises on the function $f$, so it is not possible to use binary search or any other fast searching method to efficiently solve the problem classically.

What is the best *classical* algorithm for solving the above search problem? As we have noted before, it is more complicated than one might initially think to rigorously prove lower bounds on algorithms. In this particular case it is not too hard, but because our focus is on the quantum algorithm for the problem we will only discuss informally the issue of classical complexity.

It is not hard to see that a deterministic algorithm would need to make $2^n$ queries to the black-box in the worst case (to distinguish the case where $f$ is identically 0 from any of the cases where there is a single $x$ for which $f(x) = 1$, for instance).

Probabilistically, a best strategy for an algorithm that makes $k$ queries is to simply choose $k$ distinct values of $x$ and to query the black-box at these $k$ values. In the case that there is a single value of $x$ for which $f(x) = 1$, and we require that our algorithm succeeds in finding this $x$ with probability at least $1 - \varepsilon$, then we must have $1 - \frac{k}{2^n} \leq \varepsilon$. This implies $k \geq (1 - \varepsilon)2^n$, so for constant error we therefore need $k = \Omega(2^N)$ queries to solve the problem.

In contrast, Grovers algorithm will solve the problem using $O(\sqrt{2^n})$ queries.

## Description of Grover's algorithm

Grover's algorithm is remarkably simple to describe. We will need to use the following two unitary transformations on $n$ qubits:

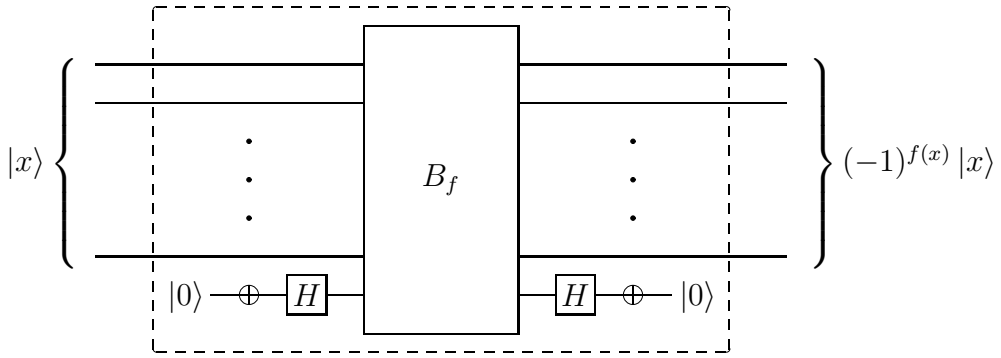
$$Z_f |x\rangle = (-1)^{f(x)} |x\rangle$$

and

$$Z_0 |x\rangle = \begin{cases} -|x\rangle & \text{if } x = 0^n \\ |x\rangle & \text{if } x \neq 0^n \end{cases}$$

for each $x \in \{0,1\}^n$.

Given the black-box transformation $B_f$, we can implement $Z_f$ using a single ancillary qubit using the phase kick-back phenomenon:




Just one query to $B_f$ is therefore required to implement $Z_f$.

The $Z_0$ transformation can be implemented in precisely the same way, except replacing the $B_f$ transformation by a reversible circuit (possibly using additional ancillary qubits) for computing the transformation

$$|x\rangle |a\rangle \mapsto |x\rangle |a \oplus (\neg x_1 \wedge \cdots \wedge \neg x_n)\rangle .$$

Such a circuit can be implemented using a classical Boolean circuit for computing $\neg x_1 \wedge \cdots \wedge \neg x_n$, along with the method we studied in Lecture 7 for constructing reversible circuits. Obviously no queries to $B_f$ are required to do this: $Z_0$ has nothing to do with $f$.

Now we are ready to describe the quantum part of the algorithm. Similar to Shor's algorithm and Simon's algorithm, we can view that there will be some classical post-processing after this algorithm is run, possibly several times.

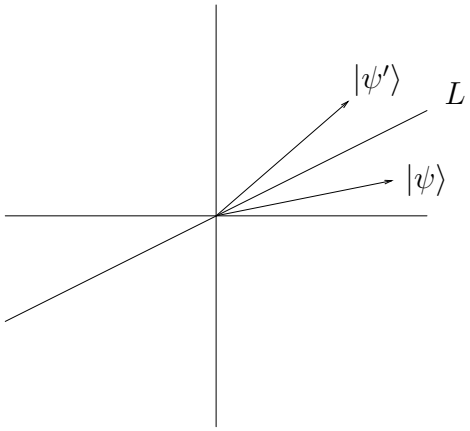
**Grover's Algorithm**

1. Let $\mathsf{X}$ be an $n$-qubit quantum register (i.e., a collection of $n$ qubits to which we assign the name $\mathsf{X}$). Let the starting state of $\mathsf{X}$ be $|0^n\rangle$ and perform $H^{\otimes n}$ on $\mathsf{X}$.
2. Apply to the register $\mathsf{X}$ the transformation
$$G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$$
$k$ times (where $k$ will be specified later).
3. Measure $\mathsf{X}$ and output the result.

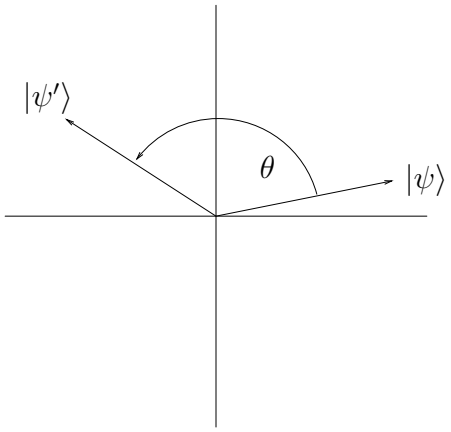
## Analysis of Grover's algorithm

As you would imagine, analyzing Grover's algorithm is more difficult than describing it. When analyzing it, it will help to think about *reflections* and *rotations* in the plane.

Here is a *reflection* of a vector $|\psi\rangle$ about the line $L$:

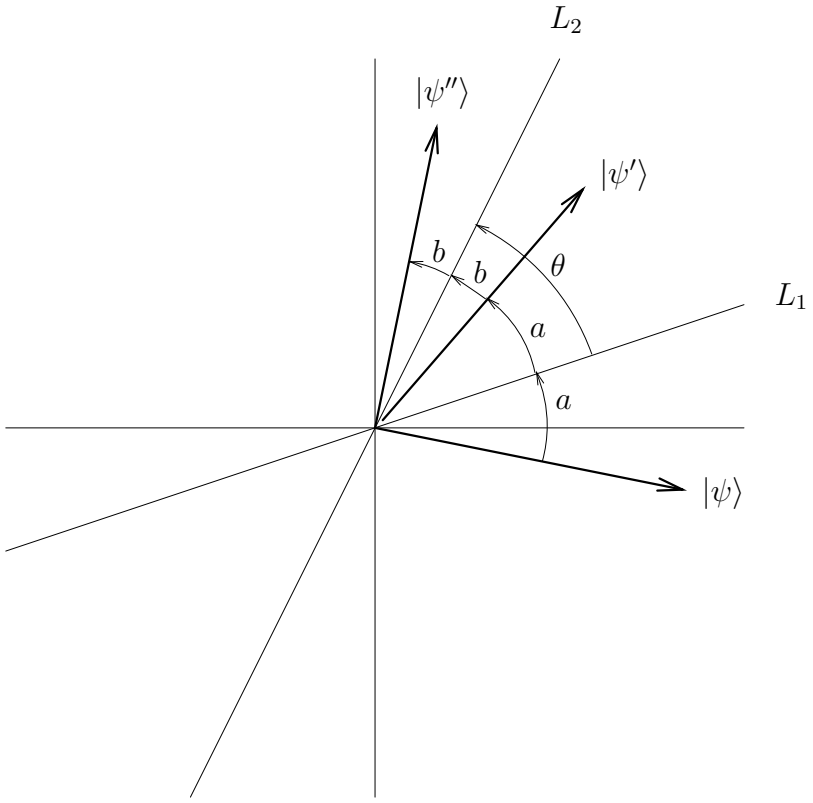



Rotations are about the origin, by some angle:



The effect of a given reflection and rotation might be the same for a particular vector $|\psi\rangle$, but they are always different transformations (e.g., reflections have determinant $-1$ while rotations have determinant $1$).

Now, suppose we have two lines $L_1$ and $L_2$, and the angle between them is $\theta$. Then for any choice of $|\psi\rangle$ if you first reflect $|\psi\rangle$ about $L_1$ and then reflect about $L_2$, the effect will be the same as rotating by an angle $2\theta$:




Why do we care about rotations and reflections? The answer is that

$$H^{\otimes n} Z_0 H^{\otimes n} \quad \text{and} \quad -Z_f$$

may be viewed as reflections, so

$$G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$$

is a rotation by a particular angle. This rotation will happen in the plane defined by two vectors that we will see shortly.

Define sets of strings $A$ and $B$ as follows:

$$A = \{x \in \{0,1\}^n \,:\, f(x) = 1\}$$
$$B = \{x \in \{0,1\}^n \,:\, f(x) = 0\}\,.$$

We will think if the set $A$ as the set of "good" strings $x \in \{0,1\}^n$; the goal of the algorithm is to find one of these strings. The set $B$ contains all of the "bad"strings $x \in \{0,1\}^n$ that do not satisfy the search criterion. Let

$$a = |A| \quad \text{and} \quad b = |B| .$$

The case where either $a$ or $b$ is zero will be handled as an easy special case, so assume for now that $a \neq 0$ and $b \neq 0$. Although the analysis will be completely general, you might imagine that an interesting case of the problem is where $a$ is very small (perhaps $a = 1$) and therefore $b$ is large (close to $N = 2^n$).

Next, define

$$|A\rangle = \frac{1}{\sqrt{a}} \sum_{x \in A} |x\rangle$$

$$|B\rangle = \frac{1}{\sqrt{b}} \sum_{x \in B} |x\rangle .$$

Notice that $|A\rangle$ and $|B\rangle$ are orthogonal unit vectors. What we will show is that the rotation mentioned above will occur in the space spanned by $|A\rangle$ and $|B\rangle$. This will simplify the analysis greatly because immediately before and after each application of $G$ in the algorithm, the state of $\mathsf{X}$ will have the form $\alpha |A\rangle + \beta |B\rangle$ for $\alpha$ and $\beta$ that will happen to be real numbers.

The state of the register $\mathsf{X}$ immediately after step 1 in the algorithm is given by

$$|h\rangle \stackrel{\text{def}}{=} H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle .$$

We can write

$$|h\rangle = \sqrt{\frac{a}{N}} |A\rangle + \sqrt{\frac{b}{N}} |B\rangle ,$$

so the state of $\mathsf{X}$ before step 2 is a vector in the subspace spanned by $\{|A\rangle , |B\rangle\}$.

Next let us see how $G$ affects states in the span of $|A\rangle$ and $|B\rangle$. In order to do this it will help to express $Z_0$ as follows:

$$Z_0 = I - 2 \, |0^n\rangle \langle 0^n| .$$

(If you forgot what vectors that look like $\langle\psi|$ mean, please refer back to the first couple of lectures of the course.) This means that

$$H^{\otimes n} Z_0 H^{\otimes n} = H^{\otimes n} (I - 2 \, |0^n\rangle \langle 0^n|) H^{\otimes n} = I - 2 \, |h\rangle \langle h| .$$

Here we are using two facts: (i) that $H^\dagger = H$, and (ii) that if $|\psi\rangle = U |\phi\rangle$ then $\langle\psi| = \langle\phi| U^\dagger$.

Now the action of $G$ on elements in the space spanned by $\{|A\rangle , |B\rangle\}$ can be determined by

examining its effect on $|A\rangle$ and $|B\rangle$:

$$\begin{aligned}
G|A\rangle &= -H^{\otimes n} Z_0 H^{\otimes n} Z_f |A\rangle \\
&= (I - 2|h\rangle\langle h|)(-Z_f)|A\rangle \\
&= (I - 2|h\rangle\langle h|)|A\rangle \\
&= |A\rangle - 2\langle h|A\rangle |h\rangle \\
&= |A\rangle - 2\sqrt{\frac{a}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) \\
&= \left(1 - \frac{2a}{N}\right)|A\rangle - \frac{2\sqrt{ab}}{N}|B\rangle
\end{aligned}$$

and

$$\begin{aligned}
G|B\rangle &= -H^{\otimes n} Z_0 H^{\otimes n} Z_f |B\rangle \\
&= -(I - 2|h\rangle\langle h|)Z_f|B\rangle \\
&= -(I - 2|h\rangle\langle h|)|B\rangle \\
&= -|B\rangle + 2\langle h|B\rangle |h\rangle \\
&= -|B\rangle + 2\sqrt{\frac{b}{N}}\left(\sqrt{\frac{a}{N}}|A\rangle + \sqrt{\frac{b}{N}}|B\rangle\right) \\
&= \frac{2\sqrt{ab}}{N}|A\rangle - \left(1 - \frac{2b}{N}\right)|B\rangle .
\end{aligned}$$

Indeed, we have that $G$ maps the subspace spanned by $\{|A\rangle, |B\rangle\}$ to itself.

We will determine in the next lecture that this action is in fact a rotation by some angle as we have speculated, and determine exactly what the angle is. Once this is done, the analysis of the algorithm will be fairly simple.