Quantum Programming Algorithms: Grover

Jens Palsberg

Apr 24, 2019

Outline

Hook: Our next quantum algorithm is Grover's algorithm. It solves a natural problem and does so faster on a quantum computer than we can ever do on a classical computer.

Purpose: Persuade you that you can understand this algorithm.

Preview:

1. We will make sure that everybody understands the problem.
2. We will show how Grover's algorithm iterates the application of a unitary matrix.
3. We will cover full details of why Grover's algorithm works.

Transition to Body: Now let us talk about the problem that this algorithm solves.

Main Point 1: We will make sure that everybody understands the problem.
[Most of you have done the homework on solving the problem on a classical computer]
[We can represent the black-box function in multiple ways]
[We can program the solution in any classical language]

Transition to MP2: Now let us look at how to solve the problem.

Main Point 2: We will show how Grover's algorithm iterates the application of a unitary matrix.
[First we need two particular unitary operations]
[Second we compose them in a particular way]
[Finally we iterate the use of the resulting matrix]

Transition to MP3: Now let us get into correctness.

Main Point 3: We will cover full details of why Grover's algorithm works.
[We can define a subspace with two dimensions]
[Each iteration performs a rotation on the subspace]
[After some iterations, a measurement gives what we are looking for, with high probaility]

Transition to Close: So there you have it.

Review: Grover's algorithm iterates to set up for a measurement that will give a good outcome with high probability.

Strong finish: Grover's algorithm is a powerful indication that a quantum computer can be faster than a classical computer. We will see more examples of that in this course.

Call to action: Look for other natural problems that may be solvable on quantum computers.

Detailed presentation

Hook: Our next quantum algorithm is Grover's algorithm. It solves a natural problem and does so faster on a quantum computer than we can ever do on a classical computer.

Purpose: Persuade you that you can understand this algorithm.

Preview:

1. We will make sure that everybody understands the problem.
2. We will show how Grover's algorithm iterates the application of a unitary matrix.
3. We will cover full details of why Grover's algorithm works.

Transition to Body: Now let us talk about the problem that this algorithm solves.

Main Point 1: We will make sure that everybody understands the problem.

[Most of you have done the homework on solving the problem on a classical computer]
Here is the homework problem.

> Grover's problem:
> Input: a function $f : \{0,1\}^n \to \{0,1\}$.
> Output: 1 if there exists $x \in \{0,1\}^n$ such that $f(x) = 1$, and 0 otherwise.
> Notation: $\{0,1\}^n$ is the set of bit strings of length n.
>
> The assignment:
> On a classical computer, in a classical language of your choice (such as C, Java, Python, etc), program a solution to Grover's problem. Treat the input function f as black box that you can call but cannot inspect in any way at all. Each solution will be code that includes one or more calls of f.

[We can represent the black-box function in multiple ways]
How do we keep the black-box function at an arms length such that we cannot get tempted to inspect it? In C we can use a function pointer. In Java we can use a lambda expression. In Python we can use an anonymous function. Other languages have similar constructs that will enable us to call the function, while giving us no way to inspect it.

[We can program the solution in any classical language]
You wrote the solutions to the homework in several languages. We need to make many calls to f: in the worst case, we need to try f on every bit string of length n. This means 2^n calls to f.

Transition to MP2: Now let us look at how to solve the problem.

Main Point 2: We will show how Grover's algorithm iterates the application of a unitary matrix.

[First we need two particular unitary operations]
Each of those operations map n qubits (plus helper qubits) to n qubits (plus helper qubits). If we

ignore the helper qubits, they the operations behave as follows. We use x to range over $\{0,1\}^n$.

$$\begin{aligned} Z_f\ |x\rangle &= (-1)^{f(x)}\ |x\rangle \\ Z_0\ |x\rangle &= \begin{cases} -|x\rangle & \text{if } x = 0^n \\ |x\rangle & \text{if } x \neq 0^n \end{cases} \end{aligned}$$

[Second we compose them in a particular way]
Define

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

[Finally we iterate the use of the resulting matrix]
Let X be a collection of n qubits that initially is $|0^n\rangle$. Grover's algorithm is:

1. Apply $H^{\otimes n}$ to X.
2. Repeat { apply G to X } $O(\sqrt{2^n})$ times.
3. Measure X and output the result.

Let us try Grover's algorithm on an example. Suppose $n = 2$. Let $f : \{0,1\}^2 \to \{0,1\}$ be defined by: $f(00) = 0$ and $f(01) = 0$ and $f(10) = 0$ and $f(11) = 1$.

Now we initialize two qubits to $|00\rangle$. Now let us execute the algorithm. How many times should we iterate in Step 2? Turns out that the correct number of iterations is 1, as we will see later.

$$\begin{aligned} & G\ H^{\otimes 2}\ |00\rangle \\ = \ & G\ \frac{1}{2}\ (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ = \ & -H^{\otimes 2}\ Z_0\ H^{\otimes 2}\ Z_f\ \frac{1}{2}\ (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ = \ & -H^{\otimes 2}\ Z_0\ H^{\otimes 2}\ \frac{1}{2}\ (|00\rangle + |01\rangle + |10\rangle - |11\rangle) \\ = \ & -H^{\otimes 2}\ Z_0\ \frac{1}{2}\ \frac{1}{2}\ ((|00\rangle + |01\rangle + |10\rangle + |11\rangle) + (|00\rangle - |01\rangle + |10\rangle - |11\rangle) + \\ & \qquad (|00\rangle + |01\rangle - |10\rangle - |11\rangle) - (|00\rangle - |01\rangle - |10\rangle + |11\rangle)) \\ = \ & -H^{\otimes 2}\ Z_0\ \frac{1}{4}\ (2|00\rangle + 2|01\rangle + 2|10\rangle - 2|11\rangle) \\ = \ & -H^{\otimes 2}\ Z_0\ \frac{1}{2}\ (|00\rangle + |01\rangle + |10\rangle - |11\rangle) \\ = \ & -H^{\otimes 2}\ \frac{1}{2}\ (-|00\rangle + |01\rangle + |10\rangle - |11\rangle) \\ = \ & H^{\otimes 2}\ \frac{1}{2}\ (|00\rangle - |01\rangle - |10\rangle + |11\rangle) \\ = \ & \frac{1}{2}\ \frac{1}{2}\ ((|00\rangle + |01\rangle + |10\rangle + |11\rangle) - (|00\rangle - |01\rangle + |10\rangle - |11\rangle) - \\ & \qquad (|00\rangle + |01\rangle - |10\rangle - |11\rangle) + (|00\rangle - |01\rangle - |10\rangle + |11\rangle)) \\ = \ & \frac{1}{4}\ (4|11\rangle) \\ = \ & |11\rangle \end{aligned}$$

So, when we measure, we will get 11 with probability 1. Thus, we found 11 after a single use of f.

Transition to MP3: Now let us get into correctness.

Main Point 3: We will cover full details of why Grover's algorithm works.

[We can define a subspace with two dimensions]
Define two sets of bit strings as follows:

$$\begin{aligned} A &= \{\ x \in \{0,1\}^n \mid f(x) = 1\ \} \\ B &= \{\ x \in \{0,1\}^n \mid f(x) = 0\ \} \end{aligned}$$

We can think of A as the *Awesome* bit strings and B as the *Bad* bit strings. For convenience, define

$$\begin{aligned} N &= 2^n \\ a &= |A| \\ b &= |B| \end{aligned}$$

Notice that $N = a + b$.

Let us focus on the main case of $(a \neq 0 \wedge b \neq 0)$. Define

$$\begin{aligned} |A\rangle &= \frac{1}{\sqrt{a}} \Sigma_{x \in A}\ |x\rangle \\ |B\rangle &= \frac{1}{\sqrt{b}} \Sigma_{x \in B}\ |x\rangle \end{aligned}$$

Lemma 1. *$|A\rangle$ and $|B\rangle$ are othogonal unit vectors.*

[Each iteration performs a rotation on the subspace]

Lemma 2.

$$\begin{aligned} G|A\rangle &= \left(1 - \frac{2a}{N}\right)|A\rangle\ -\ \frac{2\sqrt{ab}}{N}\ |B\rangle \\ G|B\rangle &= \frac{2\sqrt{ab}}{N}\ |A\rangle\ -\ \left(1 - \frac{2b}{N}\right)|B\rangle \end{aligned}$$

Proof. The proof has three main parts.

In the first part of the proof, define $|h\rangle$ to be the state of X after Step 1 of the algorithm.

$$|h\rangle\ =\ H^{\otimes n}\ |0^n\rangle\ =\ \frac{1}{\sqrt{N}}\ \Sigma_{x \in \{0,1\}^n}\ |x\rangle\ =\ \sqrt{\frac{a}{N}}\ |A\rangle\ +\ \sqrt{\frac{b}{N}}\ |B\rangle$$

Notice that in the third step, the renormalization is correct:

$$\left(\sqrt{\frac{a}{N}}\right)^2 + \left(\sqrt{\frac{b}{N}}\right)^2\ =\ \frac{a}{N} + \frac{b}{N}\ =\ \frac{a+b}{N}\ =\ 1$$

In the second part of the proof, notice that we can write

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

This is because Z_0 is the matrix that looks almost entirely like the identity matrix but has a -1 (instead $+1$) in the top-left corner. The above way of writing Z_0 enables us to calculate:

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

Here we used that $H^\dagger = H$ and that if $|\psi\rangle = U|\phi\rangle$, then $\langle\psi| = \langle\phi|\; U^\dagger$.

In the third part of the proof, we consider the two equations stated in the lemma. We begin with the equation for $G|A\rangle$.

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

In the third step, we use that we apply Z_f to $|A\rangle$, so for each vector in the sum for A, we have $f(x) = 1$, hence $(-1)^{f(x)} = -1$.

We continue with the equation for $G|B\rangle$.

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

In the third step, we use that we apply Z_f to $|B\rangle$, so for each vector in the sum for B, we have $f(x) = 0$, hence $(-1)^{f(x)} = 1$. □

Lemma 2 shows that G maps the subspace spanned by A and B to itself. Indeed, if we put $|B\rangle$ in the first row and first column, and we put $|A\rangle$ in the second row and second column, we see that we can represent G by the matrix:

$$M \;=\; \begin{pmatrix} -(1 - \frac{2b}{N}) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & 1 - \frac{2a}{N} \end{pmatrix}$$

Define $\theta \in (0, \frac{\pi}{2})$ to be the angle that satisfies:

$$\sin\theta \;=\; \sqrt{\tfrac{a}{N}} \qquad \text{and} \qquad \cos\theta \;=\; \sqrt{\tfrac{b}{N}}$$

Lemma 3. *G causes a rotation by an angle 2θ in the space spanned by A and B.*

Proof. The following matrix causes a rotation of θ:

$$R_\theta \;=\; \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

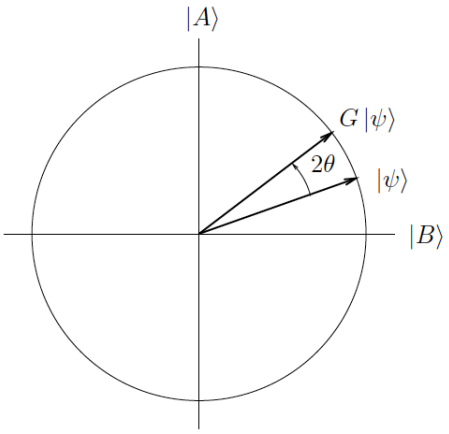
Notice that

$$R_\theta^2 \;=\; \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}^2 \;=\; \begin{pmatrix} \sqrt{\frac{b}{N}} & -\sqrt{\frac{a}{N}} \\ \sqrt{\frac{a}{N}} & \sqrt{\frac{b}{N}} \end{pmatrix}^2 \;=\; \begin{pmatrix} \frac{b-a}{N} & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & \frac{b-a}{N} \end{pmatrix} \;=\; \begin{pmatrix} -(1-\frac{2b}{N}) & -\frac{2\sqrt{ab}}{N} \\ \frac{2\sqrt{ab}}{N} & 1-\frac{2a}{N} \end{pmatrix}$$
$$=\; M$$

In the second-to-last step we used $b-a = -(a-b) = -((a-b+2b)-2b)$ and $b-a = (b-a+2a)-2a)$.

We conclude that M does two applications of R_θ so M causes a rotation that is twice the angle of the rotation caused by R_θ. □

We can illustrate the effect of G as follows:



[After some iterations, a measurement gives what we are looking for, with high probaility]

After Step 1 of the algorithm, the qubits are in the state

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

We want to rotate until we are as close to $|A\rangle$ as we can get. If we do k iterations in Step 2, then the state of the qubits is:

$$\cos((2k+1)\theta)\,|B\rangle \;+\; \sin((2k+1)\theta)\,|A\rangle$$

We are going for

$$\begin{aligned} & \sin((2k+1)\theta) \;\approx\; 1 \\ \iff\; & (2k+1)\theta \;\approx\; \frac{\pi}{2} \\ \iff\; & 2k+1 \;\approx\; \frac{\pi}{2\theta} \\ \iff\; & k \;\approx\; \frac{\pi}{4\theta} - \frac{1}{2} \end{aligned}$$

Let us focus on the case of $a = 1$, that is, we are looking for a unique bit string. Then

$$\begin{aligned} \theta &= \sin^{-1}\sqrt{\frac{1}{N}} \\ k &\approx \frac{\pi}{4\theta} - \frac{1}{2} = \frac{\pi}{4\frac{1}{\sqrt{N}}} - \frac{1}{2} = \frac{\pi}{4}\sqrt{N} - \frac{1}{2} \approx \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor \end{aligned}$$

In the example above, we have $N = 2^2$ so

$$k \approx \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor = \left\lfloor \frac{\pi}{4}\sqrt{4} \right\rfloor = \frac{\pi}{2} = 1$$

This justifies why we iterated the use of G a single time in the example.

The probability that we will measure and get a bit string x such that $f(x) = 1$ is

$$\begin{aligned} & | \sin((2k+1)\theta) |^2 \\ = & | \sin((2(\left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor) + 1) \sin^{-1}\left(\frac{1}{\sqrt{N}}\right) |^2 \end{aligned}$$

This quantity is always at least one half and it converges quickly to 1. So, we can repeat the algorithm a small number of times and evaluate f on the output each time; this will find the unique x such that $f(x) = 1$ with high probability.

If $a > 1$, we can use the same algorithm, and if it is unsuccessful, then we can increase k. This turns out to be an effective strategy.

If $a = 0$, then we can use the same algorithm, and if after a while, nothing has been found, we can be confident in concluding that $a = 0$.

Transition to Close: So there you have it.

Review: Grover's algorithm iterates to set up for a measurement that will give a good outcome with high probability.

Strong finish: Grover's algorithm is a powerful indication that a quantum computer can be faster than a classical computer. We will see more examples of that in this course.

Call to action: Look for other natural problems that may be solvable on quantum computers.