量子計算的數學基礎 MA5501*

Chapter 7. Grover's Search Algorithm

§7.1 The Search Problem

The search problem: For $N = 2^n$, we are given an arbitrary $x \in \{0,1\}^N$. The goal is to find an i such that $x_i = 1$ (and to output 'no solutions' if there are no such i). We denote the number of solutions in x by t (that is, t is the Hamming weight of x). This problem may be viewed as a simplification of the problem of searching an N-slot unordered database. Classically, a randomized algorithm would need $\mathcal{O}(N)$ queries to solve the search problem. Grover's algorithm solves it in $\mathcal{O}(\sqrt{N})$ **queries**, and $\mathcal{O}(\sqrt{N}\log_2 N)$ other gates.

§7.1 The Search Problem

The search problem: For $N = 2^n$, we are given an arbitrary $x \in \{0,1\}^N$. The goal is to find an i such that $x_i = 1$ (and to output 'no solutions' if there are no such i). We denote the number of solutions in x by t (that is, t is the Hamming weight of x). This problem may be viewed as a simplification of the problem of searching an N-slot unordered database. Classically, a randomized algorithm would need $\mathcal{O}(N)$ queries to solve the search problem. Grover's algorithm solves it in $\mathcal{O}(\sqrt{N})$ **queries**, and $\mathcal{O}(\sqrt{N}\log_2 N)$ other gates.

§7.2 Grover's Algorithm

Let $O_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$ denote the $\pm$-type oracle for the input x, and R be the unitary transformation that puts a -1 in front of all basis states $|i\rangle$ whenever $i \neq 0$, and that does nothing to the basis state $|0^n\rangle$. The Grover iterate is $\mathcal{G} = H^{\otimes n} R H^{\otimes n} O_{x,\pm}$. Note that 1 Grover iterate makes 1 query, and uses $\mathcal{O}(\log_2 N)$ other gates.

Grover's algorithm starts in the n-bit state $|0^n\rangle$, applies a Hadamard transformation to each qubit to get the uniform superposition, applies $\mathcal{G}$ to this state k times (for some k to be chosen later), and then measures the final state. Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit.

§7.2 Grover's Algorithm

Let $\mathrm{O}_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$ denote the $\pm$-type oracle for the input x, and R be the unitary transformation that puts a -1 in front of all basis states $|i\rangle$ whenever $i \neq 0$, and that does nothing to the basis state $|0^n\rangle$. The Grover iterate is $\mathcal{G} = \mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}\mathrm{O}_{x,\pm}$. Note that 1 Grover iterate makes 1 query, and uses $\mathcal{O}(\log_2 N)$ other gates. Grover's algorithm starts in the n-bit state $|0^n\rangle$, applies a Hadamard transformation to each qubit to get the uniform superposition, applies $\mathcal{G}$ to this state k times (for some k to be chosen later), and then measures the final state. Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit.

§7.2 Grover's Algorithm

Let $\mathrm{O}_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$ denote the $\pm$-type oracle for the input x, and R be the unitary transformation that puts a -1 in front of all basis states $|i\rangle$ whenever $i \neq 0$, and that does nothing to the basis state $|0^n\rangle$. The Grover iterate is $\mathcal{G} = \mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}\mathrm{O}_{x,\pm}$. Note that 1 Grover iterate makes 1 query, and uses $\mathcal{O}(\log_2 N)$ other gates. Grover's algorithm starts in the n-bit state $|0^n\rangle$, applies a Hadamard transformation to each qubit to get the uniform superposition, applies $\mathcal{G}$ to this state k times (for some k to be chosen later), and then measures the final state. Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit.

§7.2 Grover's Algorithm

Let $\mathrm{O}_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$ denote the $\pm$-type oracle for the input x, and R be the unitary transformation that puts a -1 in front of all basis states $|i\rangle$ whenever $i \neq 0$, and that does nothing to the basis state $|0^n\rangle$. The Grover iterate is $\mathcal{G} = \mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}\mathrm{O}_{x,\pm}$. Note that 1 Grover iterate makes 1 query, and uses $\mathcal{O}(\log_2 N)$ other gates. Grover's algorithm starts in the n-bit state $|0^n\rangle$, applies a Hadamard transformation to each qubit to get the uniform superposition, applies $\mathcal{G}$ to this state k times (for some k to be chosen later), and then measures the final state. Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit.

§7.2 Grover's Algorithm

Let $\mathrm{O}_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$ denote the $\pm$-type oracle for the input x, and R be the unitary transformation that puts a -1 in front of all basis states $|i\rangle$ whenever $i \neq 0$, and that does nothing to the basis state $|0^n\rangle$. The Grover iterate is $\mathcal{G} = \mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}\mathrm{O}_{x,\pm}$. Note that 1 Grover iterate makes 1 query, and uses $\mathcal{O}(\log_2 N)$ other gates. Grover's algorithm starts in the n-bit state $|0^n\rangle$, applies a Hadamard transformation to each qubit to get the uniform superposition, applies $\mathcal{G}$ to this state k times (for some k to be chosen later), and then measures the final state. Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit.

§7.2 Grover's Algorithm

Let $\mathrm{O}_{x,\pm}|i\rangle = (-1)^{x_i}|i\rangle$ denote the $\pm$-type oracle for the input x, and R be the unitary transformation that puts a -1 in front of all basis states $|i\rangle$ whenever $i \neq 0$, and that does nothing to the basis state $|0^n\rangle$. The Grover iterate is $\mathcal{G} = \mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}\mathrm{O}_{x,\pm}$. Note that 1 Grover iterate makes 1 query, and uses $\mathcal{O}(\log_2 N)$ other gates. Grover's algorithm starts in the n-bit state $|0^n\rangle$, applies a Hadamard transformation to each qubit to get the uniform superposition, applies $\mathcal{G}$ to this state k times (for some k to be chosen later), and then measures the final state. Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit.

§7.2 Grover's Algorithm

The following figure illustrates the Grover algorithm.

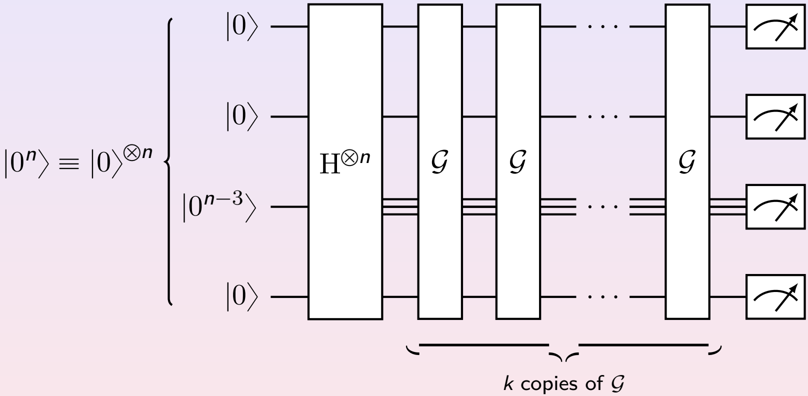

Figure 1: Grover's algorithm, with k Grover iterates

§7.2 Grover's Algorithm

To analyze this, define the following "good" and "bad" states:

$$|G\rangle = \frac{1}{\sqrt{t}} \sum_{\{i \mid x_i = 1\}} |i\rangle \qquad \text{and} \qquad |B\rangle = \frac{1}{\sqrt{N-t}} \sum_{\{i \mid x_i = 0\}} |i\rangle .$$

where $t = \#\{i \mid x_i = 1\}$. Then the uniform state over all indices edges can be written as

$$\begin{aligned} |U\rangle &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle = \frac{1}{\sqrt{N}} \Big(\sum_{\{i \mid x_i = 1\}} + \sum_{\{i \mid x_i = 0\}} \Big) |i\rangle \\ &= \frac{1}{\sqrt{N}} \Big(\sqrt{t}\,|G\rangle + \sqrt{N-t}\,|B\rangle \Big) = \sin\theta |G\rangle + \cos\theta |B\rangle , \end{aligned}$$

where $\theta = \arcsin\sqrt{\frac{t}{N}}$.

§7.2 Grover's Algorithm

To analyze this, define the following "good" and "bad" states:

$$|G\rangle = \frac{1}{\sqrt{t}} \sum_{\{i \mid x_i = 1\}} |i\rangle \qquad \text{and} \qquad |B\rangle = \frac{1}{\sqrt{N-t}} \sum_{\{i \mid x_i = 0\}} |i\rangle .$$

where $t = \#\{i \mid x_i = 1\}$. Then the uniform state over all indices edges can be written as

$$\begin{aligned} |U\rangle &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle = \frac{1}{\sqrt{N}} \Big(\sum_{\{i \mid x_i = 1\}} + \sum_{\{i \mid x_i = 0\}} \Big) |i\rangle \\ &= \frac{1}{\sqrt{N}} \Big(\sqrt{t}\,|G\rangle + \sqrt{N-t}\,|B\rangle \Big) = \sin\theta |G\rangle + \cos\theta |B\rangle , \end{aligned}$$

where $\theta = \arcsin \sqrt{\frac{t}{N}}$.

§7.2 Grover's Algorithm

The Grover iterate $\mathcal{G}$ is actually the product of two reflections (in the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$):

1. $\mathrm{O}_{x,\pm}$ is a reflection through $|B\rangle$: since $\langle G|B\rangle = 0$ and

$$\mathrm{O}_{x,\pm}\big(a|G\rangle + b|B\rangle\big) = -a|G\rangle + b|B\rangle\,.$$

2. $\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}$ is a reflection through $|U\rangle$: first the reflection through a unit vector $|\psi\rangle$ can be expressed as $2|\psi\rangle\langle\psi| - \mathrm{I}$ since

$$(2|\psi\rangle\langle\psi| - \mathrm{I})|\phi\rangle = \langle\psi|\phi\rangle|\psi\rangle - \big(|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle\big)$$

and note that $\langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto $\mathrm{span}(|\psi\rangle)$ and $|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto the space perpendicular to $|\psi\rangle$. Therefore, $\mathrm{R} = 2|0^n\rangle\langle 0^n| - \mathrm{I}$ so that

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

§7.2 Grover's Algorithm

The Grover iterate $\mathcal{G}$ is actually the product of two reflections (in the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$):

1. $\mathrm{O}_{x,\pm}$ is a reflection through $|B\rangle$: since $\langle G|B\rangle = 0$ and

$$\mathrm{O}_{x,\pm}\big(a|G\rangle + b|B\rangle\big) = -a|G\rangle + b|B\rangle\,.$$

2. $\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}$ is a reflection through $|U\rangle$: first the reflection through a unit vector $|\psi\rangle$ can be expressed as $2|\psi\rangle\langle\psi| - \mathrm{I}$ since

$$(2|\psi\rangle\langle\psi| - \mathrm{I})|\phi\rangle = \langle\psi|\phi\rangle|\psi\rangle - \big(|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle\big)$$

and note that $\langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto span$(|\psi\rangle)$ and $|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto the space perpendicular to $|\psi\rangle$. Therefore, $\mathrm{R} = 2|0^n\rangle\langle 0^n| - \mathrm{I}$ so that

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

§7.2 Grover's Algorithm

The Grover iterate $\mathcal{G}$ is actually the product of two reflections (in the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$):

1. $\mathrm{O}_{x,\pm}$ is a reflection through $|B\rangle$: since $\langle G|B\rangle = 0$ and
$$\mathrm{O}_{x,\pm}\big(a|G\rangle + b|B\rangle\big) = -a|G\rangle + b|B\rangle\,.$$

2. $\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}$ is a reflection through $|U\rangle$: first the reflection through a unit vector $|\psi\rangle$ can be expressed as $2|\psi\rangle\langle\psi| - \mathrm{I}$ since
$$(2|\psi\rangle\langle\psi| - \mathrm{I})|\phi\rangle = \langle\psi|\phi\rangle|\psi\rangle - \big(|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle\big)$$
and note that $\langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto span$(|\psi\rangle)$ and $|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto the space perpendicular to $|\psi\rangle$. Therefore, $\mathrm{R} = 2|0^n\rangle\langle 0^n| - \mathrm{I}$ so that
$$\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n} = \mathrm{H}^{\otimes n}(2|0^n\rangle\langle 0^n| - \mathrm{I})\mathrm{H}^{\otimes n} = 2|U\rangle\langle U| - \mathrm{I}\,.$$

§7.2 Grover's Algorithm

The Grover iterate $\mathcal{G}$ is actually the product of two reflections (in the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$):

1. $\mathrm{O}_{x,\pm}$ is a reflection through $|B\rangle$: since $\langle G|B\rangle = 0$ and

$$\mathrm{O}_{x,\pm}\big(a|G\rangle + b|B\rangle\big) = -a|G\rangle + b|B\rangle\,.$$

2. $\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}$ is a reflection through $|U\rangle$: first the reflection through a unit vector $|\psi\rangle$ can be expressed as $2|\psi\rangle\langle\psi| - \mathrm{I}$ since

$$(2|\psi\rangle\langle\psi| - \mathrm{I})|\phi\rangle = \langle\psi|\phi\rangle|\psi\rangle - \big(|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle\big)$$

and note that $\langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto span$(|\psi\rangle)$ and $|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto the space perpendicular to $|\psi\rangle$. Therefore, $\mathrm{R} = 2|0^n\rangle\langle 0^n| - \mathrm{I}$ so that

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

§7.2 Grover's Algorithm

The Grover iterate $\mathcal{G}$ is actually the product of two reflections (in the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$):

1. $\mathrm{O}_{x,\pm}$ is a reflection through $|B\rangle$: since $\langle G|B\rangle = 0$ and

$$\mathrm{O}_{x,\pm}\big(a|G\rangle + b|B\rangle\big) = -a|G\rangle + b|B\rangle\,.$$

2. $\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}$ is a reflection through $|U\rangle$: first the reflection through a unit vector $|\psi\rangle$ can be expressed as $2|\psi\rangle\langle\psi| - \mathrm{I}$ since

$$(2|\psi\rangle\langle\psi| - \mathrm{I})|\phi\rangle = \langle\psi|\phi\rangle|\psi\rangle - \big(|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle\big)$$

and note that $\langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto span$(|\psi\rangle)$ and $|\phi\rangle - \langle\psi|\phi\rangle|\psi\rangle$ is the orthogonal projection of $|\phi\rangle$ onto the space perpendicular to $|\psi\rangle$. Therefore, $\mathrm{R} = 2|0^n\rangle\langle 0^n| - \mathrm{I}$ so that

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

§7.2 Grover's Algorithm

Here is Grover's algorithm restated, assuming we know the fraction of solutions is $\varepsilon = t/N$:

1. Set up the starting state $|U\rangle = \mathrm{H}^{\otimes n}|0^n\rangle$.
2. Repeat the following $k = \mathcal{O}(1/\sqrt{\varepsilon})$ times:
   - (a) Reflect through $|B\rangle$ (that is, apply $\mathrm{O}_{x,\pm}$).
   - (b) Reflect through $|U\rangle$ (that is, apply $\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n}$).
3. Measure the first register and check that the resulting i is a solution.

§7.2 Grover's Algorithm

Geometric argument: There is a fairly simple geometric argument why the algorithm works. The analysis is in the 2-dimensional real plane spanned by $|B\rangle$ and $|G\rangle$. We start with $|U\rangle = \sin\theta|G\rangle + \cos\theta|B\rangle$: The two reflections ⓐ and ⓑ increase the angle from θ to 3θ, moving us towards the good state, as illustrated in Figure 2.

Figure 2: The first iteration of Grover: (left) start with $|U\rangle$, (middle) reflect through $|B\rangle$ to get $\mathrm{O}_{x,\pm}|U\rangle$, (right) reflect through $|U\rangle$ to get $\mathcal{G}|U\rangle$

§7.2 Grover's Algorithm

Geometric argument: There is a fairly simple geometric argument why the algorithm works. The analysis is in the 2-dimensional real plane spanned by $|B\rangle$ and $|G\rangle$. We start with $|U\rangle = \sin\theta|G\rangle + \cos\theta|B\rangle$: The two reflections ⓐ and ⓑ increase the angle from θ to 3θ, moving us towards the good state, as illustrated in Figure 2.

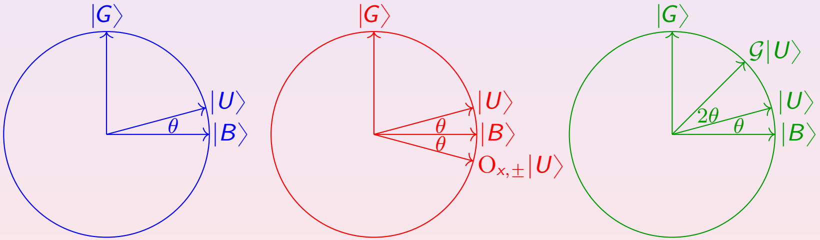

Figure 2: The first iteration of Grover: (left) start with $|U\rangle$, (middle) reflect through $|B\rangle$ to get $\mathrm{O}_{x,\pm}|U\rangle$, (right) reflect through $|U\rangle$ to get $\mathcal{G}|U\rangle$

§7.2 Grover's Algorithm

The next two reflections ⓐ and ⓑ increase the angle with another 2θ, etc. More generally, after k applications of ⓐ and ⓑ our state has become

$$\sin((2k+1)\theta)|G\rangle + \cos((2k+1)\theta)|B\rangle .$$

If we now measure, the probability of seeing a solution is $P_k = \sin^2((2k+1)\theta)$. We want P_k to be as close to 1 as possible. Note that if we can choose $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$, then $(2\widetilde{k}+1)\theta = \frac{\pi}{2}$ and hence $P_{\widetilde{k}} = \sin^2\frac{\pi}{2} = 1$. An example where this works is if $t = N/4$, for then $\theta = \pi/6$ and $\widetilde{k} = 1$. Unfortunately $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ will usually not be an integer, and we can only do an integer number of Grover iterations.

§7.2 Grover's Algorithm

The next two reflections ⓐ and ⓑ increase the angle with another 2θ, etc. More generally, after k applications of ⓐ and ⓑ our state has become

$$\sin((2k+1)\theta)|G\rangle + \cos((2k+1)\theta)|B\rangle\,.$$

If we now measure, the probability of seeing a solution is $P_k = \sin^2((2k+1)\theta)$. We want P_k to be as close to 1 as possible. Note that if we can choose $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$, then $(2\widetilde{k}+1)\theta = \frac{\pi}{2}$ and hence $P_{\widetilde{k}} = \sin^2\frac{\pi}{2} = 1$. An example where this works is if $t = N/4$, for then $\theta = \pi/6$ and $\widetilde{k} = 1$. Unfortunately $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ will usually not be an integer, and we can only do an integer number of Grover iterations.

§7.2 Grover's Algorithm

The next two reflections ⓐ and ⓑ increase the angle with another 2θ, etc. More generally, after k applications of ⓐ and ⓑ our state has become

$$\sin((2k+1)\theta)|G\rangle + \cos((2k+1)\theta)|B\rangle .$$

If we now measure, the probability of seeing a solution is $P_k = \sin^2((2k+1)\theta)$. We want P_k to be as close to 1 as possible. Note that if we can choose $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$, then $(2\widetilde{k}+1)\theta = \frac{\pi}{2}$ and hence $P_{\widetilde{k}} = \sin^2\frac{\pi}{2} = 1$. An example where this works is if $t = N/4$, for then $\theta = \pi/6$ and $\widetilde{k} = 1$. Unfortunately $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ will usually not be an integer, and we can only do an integer number of Grover iterations.

§7.2 Grover's Algorithm

The next two reflections ⓐ and ⓑ increase the angle with another 2θ, etc. More generally, after k applications of ⓐ and ⓑ our state has become

$$\sin((2k+1)\theta)|G\rangle + \cos((2k+1)\theta)|B\rangle .$$

If we now measure, the probability of seeing a solution is $P_k = \sin^2((2k+1)\theta)$. We want P_k to be as close to 1 as possible. Note that if we can choose $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$, then $(2\widetilde{k}+1)\theta = \frac{\pi}{2}$ and hence $P_{\widetilde{k}} = \sin^2\frac{\pi}{2} = 1$. An example where this works is if $t = N/4$, for then $\theta = \pi/6$ and $\widetilde{k} = 1$. Unfortunately $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ will usually not be an integer, and we can only do an integer number of Grover iterations.

§7.2 Grover's Algorithm

However, if we choose k to be the integer closest to $\widetilde{k}$, then our final state will still be close to $|G\rangle$ and the failure probability will still be small (assuming $t \ll N$):

$$\begin{aligned} 1 - P_k &= \cos^2((2k+1)\theta) = \cos^2((2\widetilde{k}+1)\theta + 2(k-\widetilde{k})\theta) \\ &= \cos^2\left(\frac{\pi}{2} + 2(k-\widetilde{k})\theta\right) \\ &= \sin^2(2(k-\widetilde{k})\theta) \leqslant \sin^2(\theta) = \frac{t}{N}, \end{aligned}$$

where we used $|k-\widetilde{k}| \leqslant 1/2$. Since $\arcsin(\theta) \geqslant \theta$, the number of queries is $k \leqslant \frac{\pi}{4\theta} \leqslant \frac{\pi}{4}\sqrt{\frac{N}{t}}$.

§7.2 Grover's Algorithm

However, if we choose k to be the integer closest to $\widetilde{k}$, then our final state will still be close to $|G\rangle$ and the failure probability will still be small (assuming $t \ll N$):

$$\begin{aligned} 1 - P_k &= \cos^2((2k+1)\theta) = \cos^2((2\widetilde{k}+1)\theta + 2(k-\widetilde{k})\theta) \\ &= \cos^2\left(\frac{\pi}{2} + 2(k-\widetilde{k})\theta\right) \\ &= \sin^2(2(k-\widetilde{k})\theta) \leqslant \sin^2(\theta) = \frac{t}{N}, \end{aligned}$$

where we used $|k - \widetilde{k}| \leqslant 1/2$. Since $\arcsin(\theta) \geqslant \theta$, the number of queries is $k \leqslant \frac{\pi}{4\theta} \leqslant \frac{\pi}{4}\sqrt{\frac{N}{t}}$.

§7.2 Grover's Algorithm

Algebraic argument: Let a_k denote the amplitude of the indices of the t 1-bits after k Grover iterates, and b_k the amplitude of the indices of the 0-bits so that $t|a_k|^2 + (N-t)|b_k|^2 = 1$ for all $k \in \mathbb{N}$. Initially, for the uniform superposition $|U\rangle$ we have $a_0 = b_0 = \dfrac{1}{\sqrt{N}}$.

Since

$$\mathrm{H}^{\otimes n} = \mathrm{H}_n \qquad \text{and} \qquad \mathrm{R} = \mathrm{diag}(1,-1,-1,\cdots,-1)\,,$$

$\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n} = \left[\frac{2}{N}\right] - \mathrm{I}$, where $\left[\frac{2}{N}\right]$ is the $N \times N$ matrix in which all entries are $\frac{2}{N}$; thus we find the following recursion:

$$a_{k+1} = \frac{2}{N}\big[-ta_k + (N-t)b_k\big] + a_k = \frac{N-2t}{N}a_k + \frac{2(N-t)}{N}b_k\,,$$

$$b_{k+1} = \frac{2}{N}\big[-ta_k + (N-t)b_k\big] - b_k = \frac{-2t}{N}a_k + \frac{N-2t}{N}b_k\,.$$

§7.2 Grover's Algorithm

Algebraic argument: Let a_k denote the amplitude of the indices of the t 1-bits after k Grover iterates, and b_k the amplitude of the indices of the 0-bits so that $t|a_k|^2 + (N-t)|b_k|^2 = 1$ for all $k \in \mathbb{N}$. Initially, for the uniform superposition $|U\rangle$ we have $a_0 = b_0 = \dfrac{1}{\sqrt{N}}$. Since

$$\mathrm{H}^{\otimes n} = \mathrm{H}_n \qquad \text{and} \qquad \mathrm{R} = \mathrm{diag}(1, -1, -1, \cdots, -1)\,,$$

$\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n} = \left[\frac{2}{N}\right] - \mathrm{I}$, where $\left[\frac{2}{N}\right]$ is the $N \times N$ matrix in which all entries are $\frac{2}{N}$; thus we find the following recursion:

$$a_{k+1} = \frac{2}{N}\big[-ta_k + (N-t)b_k\big] + a_k = \frac{N-2t}{N}a_k + \frac{2(N-t)}{N}b_k\,,$$

$$b_{k+1} = \frac{2}{N}\big[-ta_k + (N-t)b_k\big] - b_k = \frac{-2t}{N}a_k + \frac{N-2t}{N}b_k\,.$$

§7.2 Grover's Algorithm

Algebraic argument: Let a_k denote the amplitude of the indices of the t 1-bits after k Grover iterates, and b_k the amplitude of the indices of the 0-bits so that $t|a_k|^2 + (N-t)|b_k|^2 = 1$ for all $k \in \mathbb{N}$. Initially, for the uniform superposition $|U\rangle$ we have $a_0 = b_0 = \dfrac{1}{\sqrt{N}}$. Since

$$\mathrm{H}^{\otimes n} = \mathrm{H}_n \qquad \text{and} \qquad \mathrm{R} = \mathrm{diag}(1, -1, -1, \cdots, -1),$$

$\mathrm{H}^{\otimes n}\mathrm{R}\mathrm{H}^{\otimes n} = \left[\frac{2}{N}\right] - \mathrm{I}$, where $\left[\frac{2}{N}\right]$ is the $N \times N$ matrix in which all entries are $\frac{2}{N}$; thus we find the following recursion:

$$a_{k+1} = \frac{2}{N}\big[-ta_k + (N-t)b_k\big] + a_k = \frac{N-2t}{N}a_k + \frac{2(N-t)}{N}b_k\,,$$

$$b_{k+1} = \frac{2}{N}\big[-ta_k + (N-t)b_k\big] - b_k = \frac{-2t}{N}a_k + \frac{N-2t}{N}b_k\,.$$

§7.2 Grover's Algorithm

With $\theta = \arcsin\sqrt{t/N}$ as before, we have

$$\begin{bmatrix} a_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \cos(2\theta) & 2\cos^2\theta \\ -2\sin^2\theta & \cos(2\theta) \end{bmatrix} \begin{bmatrix} a_k \\ b_k \end{bmatrix}.$$

By matrix diagonalization, we find that

$$\begin{bmatrix} \cos(2\theta) & 2\cos^2\theta \\ -2\sin^2\theta & \cos(2\theta) \end{bmatrix} = \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix} \begin{bmatrix} e^{2i\theta} & 0 \\ 0 & e^{-2i\theta} \end{bmatrix} \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix}^{-1},$$

thus

$$\begin{bmatrix} a_k \\ b_k \end{bmatrix} = \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix} \begin{bmatrix} e^{2i\theta} & 0 \\ 0 & e^{-2i\theta} \end{bmatrix}^k \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix}^{-1} \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

$$= \frac{1}{\sqrt{N}} \begin{bmatrix} \sin(2k+1)\theta/\sin\theta \\ \cos(2k+1)\theta/\cos\theta \end{bmatrix}.$$

§7.2 Grover's Algorithm

With $\theta = \arcsin\sqrt{t/N}$ as before, we have

$$\begin{bmatrix} a_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \cos(2\theta) & 2\cos^2\theta \\ -2\sin^2\theta & \cos(2\theta) \end{bmatrix} \begin{bmatrix} a_k \\ b_k \end{bmatrix}.$$

By matrix diagonalization, we find that

$$\begin{bmatrix} a_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix} \begin{bmatrix} e^{2i\theta} & 0 \\ 0 & e^{-2i\theta} \end{bmatrix} \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix}^{-1} \begin{bmatrix} a_k \\ b_k \end{bmatrix};$$

thus

$$\begin{bmatrix} a_k \\ b_k \end{bmatrix} = \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix} \begin{bmatrix} e^{2i\theta} & 0 \\ 0 & e^{-2i\theta} \end{bmatrix}^k \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix}^{-1} \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}$$

$$= \frac{1}{\sqrt{N}} \begin{bmatrix} \sin(2k+1)\theta/\sin\theta \\ \cos(2k+1)\theta/\cos\theta \end{bmatrix}.$$

§7.2 Grover's Algorithm

With $\theta = \arcsin\sqrt{t/N}$ as before, we have

$$\begin{bmatrix} a_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \cos(2\theta) & 2\cos^2\theta \\ -2\sin^2\theta & \cos(2\theta) \end{bmatrix} \begin{bmatrix} a_k \\ b_k \end{bmatrix}.$$

By matrix diagonalization, we find that

$$\begin{bmatrix} a_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix} \begin{bmatrix} e^{2i\theta} & 0 \\ 0 & e^{-2i\theta} \end{bmatrix} \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix}^{-1} \begin{bmatrix} a_k \\ b_k \end{bmatrix};$$

thus

$$\begin{aligned} \begin{bmatrix} a_k \\ b_k \end{bmatrix} &= \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix} \begin{bmatrix} e^{2i\theta} & 0 \\ 0 & e^{-2i\theta} \end{bmatrix}^k \begin{bmatrix} \cos\theta & \cos\theta \\ i\sin\theta & -i\sin\theta \end{bmatrix}^{-1} \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} \\ &= \frac{1}{\sqrt{N}} \begin{bmatrix} \sin(2k+1)\theta/\sin\theta \\ \cos(2k+1)\theta/\cos\theta \end{bmatrix}. \end{aligned}$$

§7.2 Grover's Algorithm

Therefore, we obtain the following formulas for a_k and b_k:

$$a_k = \frac{1}{\sqrt{t}} \sin((2k+1)\theta) \quad \text{and} \quad b_k = \frac{1}{\sqrt{N-t}} \cos((2k+1)\theta)\,.$$

Accordingly, after k iterations the success probability (the sum of squares of the amplitudes of the locations of the t 1-bits) is the same as in the geometric analysis

$$P_k = t \cdot a_k^2 = \sin^2((2k+1)\theta).$$

Thus **assuming** t **is known** we have a bounded-error quantum search algorithm with $\mathcal{O}(\sqrt{N/t})$ queries.

§7.2 Grover's Algorithm

We now list (without proofs) a number of useful variants of Grover:

1. If we know t exactly, the algorithm can be tweaked to end up in exactly the good state. Roughly speaking, you can make the angle θ slightly smaller, such that $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ becomes an integer.
2. If we do not know t, then there is a problem: we do not know which k to use so we do not know when to stop doing the Grover iterates. Note that if k gets too big, the success probability $P_k = \sin^2((2k+1)\theta))$ goes down again! However, a slightly more complicated algorithm (basically running the above algorithm with systematic different guesses for k) shows that an expected number of $\mathcal{O}(\sqrt{N/t})$ queries still suffices to find a solution if there are t solutions.

§7.2 Grover's Algorithm

We now list (without proofs) a number of useful variants of Grover:

1. If we know t exactly, the algorithm can be tweaked to end up in exactly the good state. Roughly speaking, you can make the angle θ slightly smaller, such that $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ becomes an integer.
2. If we do not know t, then there is a problem: we do not know which k to use so we do not know when to stop doing the Grover iterates. Note that if k gets too big, the success probability $P_k = \sin^2((2k+1)\theta))$ goes down again! However, a slightly more complicated algorithm (basically running the above algorithm with systematic different guesses for k) shows that an expected number of $\mathcal{O}(\sqrt{N/t})$ queries still suffices to find a solution if there are t solutions.

§7.2 Grover's Algorithm

We now list (without proofs) a number of useful variants of Grover:

1. If we know t exactly, the algorithm can be tweaked to end up in exactly the good state. Roughly speaking, you can make the angle θ slightly smaller, such that $\widetilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ becomes an integer.
2. If we do not know t, then there is a problem: we do not know which k to use so we do not know when to stop doing the Grover iterates. Note that if k gets too big, the success probability $P_k = \sin^2((2k+1)\theta))$ goes down again! However, a slightly more complicated algorithm (basically running the above algorithm with systematic different guesses for k) shows that an expected number of $\mathcal{O}(\sqrt{N/t})$ queries still suffices to find a solution if there are t solutions.

§7.2 Grover's Algorithm

3. If we know a lower bound τ on the actual (possibly unknown) number of solutions t, then the algorithm in ② uses an expected number of $\mathcal{O}(\sqrt{N/\tau})$ queries. If we run this algorithm for up to three times its expected number of queries, then (by Markov's inequality) with probability at least $2/3$ it will produce a solution. This way we can turn an expected runtime into a worst-case runtime.

4. If we do not know t but would like to reduce the probability of not finding a solution to some small $\varepsilon > 0$, then we can do this using $\mathcal{O}(\sqrt{N \log(1/\varepsilon)})$ queries. The important part here is that the $\log(1/\varepsilon)$ is inside the square-root; usual error reduction by $\mathcal{O}(\log(1/\varepsilon))$ repetitions of basic Grover would give the worse upper bound of $\mathcal{O}(\sqrt{N}\log(1/\varepsilon))$ queries.

§7.2 Grover's Algorithm

3. If we know a lower bound τ on the actual (possibly unknown) number of solutions t, then the algorithm in ② uses an expected number of $\mathcal{O}(\sqrt{N/\tau})$ queries. If we run this algorithm for up to three times its expected number of queries, then (by Markov's inequality) with probability at least $2/3$ it will produce a solution. This way we can turn an expected runtime into a worst-case runtime.
4. If we do not know t but would like to reduce the probability of not finding a solution to some small $\varepsilon > 0$, then we can do this using $\mathcal{O}(\sqrt{N \log(1/\varepsilon)})$ queries. The important part here is that the $\log(1/\varepsilon)$ is inside the square-root; usual error reduction by $\mathcal{O}(\log(1/\varepsilon))$ repetitions of basic Grover would give the worse upper bound of $\mathcal{O}(\sqrt{N}\log(1/\varepsilon))$ queries.

§7.3 Amplitude Amplification

The analysis that worked for Grover's algorithm is actually much more generally applicable. Let $\chi : \mathbb{Z} \to \{0, 1\}$ be any Boolean function; inputs $z \in \mathbb{Z}$ satisfying $\chi(z) = 1$ are called solutions. Suppose we have an algorithm to check whether z is a solution. This can be written as a unitary O_χ that maps $|z\rangle$ to $(-1)^{\chi(z)}|z\rangle$. Suppose also we have some (quantum or classical) algorithm $\mathcal{A}$ that uses no intermediate measurements and has probability p of finding a solution when applied to starting state $|0\rangle$. Classically, we would have to repeat $\mathcal{A}$ roughly $1/p$ times before we find a solution.

§7.3 Amplitude Amplification

The amplitude amplification algorithm below only needs to run $\mathcal{A}$ and $\mathcal{A}^{-1}$ $\mathcal{O}(1/\sqrt{p})$ times:

1. Setup the starting state $|U\rangle = \mathcal{A}|0\rangle$.
2. Repeat the following $\mathcal{O}(1/\sqrt{p})$ times:
   a. Reflect through $|B\rangle$ (that is, apply O_χ).
   b. Reflect through $|U\rangle$ (that is, apply $\mathcal{A}\mathrm{R}\mathcal{A}^{-1}$).
3. Measure the first register and check that the resulting element x is marked.

§7.3 Amplitude Amplification

Defining $\theta = \arcsin\sqrt{p}$ and good and bad states $|G\rangle$ and $|B\rangle$ in analogy with the earlier geometric argument for Grover's algorithm, the same reasoning shows that amplitude amplification indeed finds a solution with high probability. This way, we can speed up a very large class of classical heuristic algorithms: any algorithm that has some non-trivial probability of finding a solution can be amplified to success probability nearly 1 (provided we can efficiently check solutions; that is, implement O_χ). Note that the Hadamard transform $\mathrm{H}^{\otimes n}$ can be viewed as an algorithm with success probability $p = t/N$ for a search problem of size N with t solutions, because $\mathrm{H}^{\otimes n}|0^n\rangle$ is the uniform superposition over all N locations. Hence Grover's algorithm is a special case of amplitude amplification, where $\mathrm{O}_\chi = \mathrm{O}_{x,\pm}$ and $\mathcal{A} = \mathrm{H}^{\otimes n}$.

§7.3 Amplitude Amplification

Defining $\theta = \arcsin\sqrt{p}$ and good and bad states $|G\rangle$ and $|B\rangle$ in analogy with the earlier geometric argument for Grover's algorithm, the same reasoning shows that amplitude amplification indeed finds a solution with high probability. This way, we can speed up a very large class of classical heuristic algorithms: any algorithm that has some non-trivial probability of finding a solution can be amplified to success probability nearly 1 (provided we can efficiently check solutions; that is, implement O_χ). Note that the Hadamard transform $\mathrm{H}^{\otimes n}$ can be viewed as an algorithm with success probability $p = t/N$ for a search problem of size N with t solutions, because $\mathrm{H}^{\otimes n}|0^n\rangle$ is the uniform superposition over all N locations. Hence Grover's algorithm is a special case of amplitude amplification, where $\mathrm{O}_\chi = \mathrm{O}_{x,\pm}$ and $\mathcal{A} = \mathrm{H}^{\otimes n}$.