CS 294-2 Quadratic speedup for unstructured search - Grover's Algorithm 2/28/07
Spring 2007 Lecture 11

0.0.1 Unstructured Search

Here's the problem: You are given a boolean function $f : \{1,\dots,N\} \to \{0,1\}$, and are promised that for exactly one $a \in \{1,\dots,N\}$, $f(a) = 1$. Think of this as a table of size N, where exactly one element has value 1, and all the others are 0. Since we assume f can be computed classically in polynomial time, we can also compute it in superposition:

$$\sum_x \alpha_x |x\rangle |0\rangle \to \sum_x \alpha_x |x\rangle |f(x)\rangle$$

As we saw before, we can use circuit for f to put information about $f(x)$ in the phase by effecting the transformation:

$$\sum_x \alpha_x |x\rangle \to \sum_x \alpha_x (-1)^{f(x)} |x\rangle$$

Here is another way of creating this phase state:

$$\begin{aligned}
\sum_x \alpha_x |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) &\mapsto \sum_x \alpha_x \left(\frac{|x\rangle |f(x)\rangle - |x\rangle |\overline{f(x)}\rangle}{\sqrt{2}} \right) \\
&= \sum_x \alpha_x |x\rangle \left(\frac{|f(x)\rangle - |\overline{f(x)}\rangle}{\sqrt{2}} \right) \\
&= \sum_x \alpha_x |x\rangle (-1)^{f(x)} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)
\end{aligned}$$

Now, we might as well assume f is a black box or oracle. All we need to do is design an algorithm that finds $a : f(a) = 1$.

0.0.2 Grover's algorithm

Grover's algorithm finds a in $O(\sqrt{N})$ steps. Consider the N dimensional Hilbert space spanned by $|1\rangle, \dots, |N\rangle$. We wish to find $|a\rangle$. There is a state that we can create: $|u\rangle = \sum_x \frac{1}{\sqrt{N}} |x\rangle$. Consider the two dimensional subspace spanned by $|a\rangle$ and $|u\rangle$. Let $|e\rangle$ be the state orthogonal to $|a\rangle$ in this subspace. Let θ be the angle between $|u\rangle$ and $|e\rangle$. Then $sin\ \theta = 1/\sqrt{N}$ and therefore $\theta \approx 1/\sqrt{N}$. See Figure **??** for an illustration of these vectors.

$|a\rangle$ is the target, so we want to increase θ. But how?

One way to rotate a vector is to make two reflections. In particular, we can rotate a vector $|v\rangle$ by 2θ by reflecting about $|e\rangle$ and then reflecting about $|u\rangle$. This transformation is also illustrated in Figure **??**.

Each step of our algorithm is a rotation by 2θ (we discuss the implementation below). This means that we need $\frac{\pi/2}{2\theta}$ iterations for the algorithm to complete. Now, what's θ?

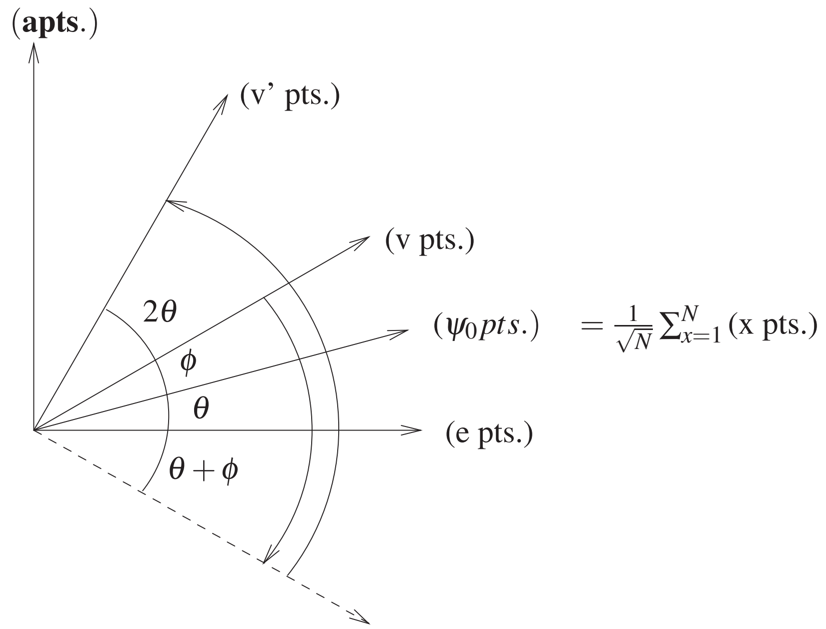

Figure 0.1: To rotate $|v\rangle$ by 2θ to $|v'\rangle$, we reflect around $|e\rangle$ and then reflect around $|\psi_0\rangle$.

$$\langle \psi_0 | a \rangle = \cos(\pi/2 - \theta) = \sin(\theta) = \frac{1}{\sqrt{N}}$$

Since $\sin\theta \approx \theta$, we know that $\theta \approx \frac{1}{\sqrt{N}}$. Thus, we need $O(\sqrt{N})$ iterations for the algorithm to complete. In the end, we get very close to $|a\rangle$, and then with high probability, a measurement of the state yields a.

How do you implement the two reflections?

1. Reflection about $|e\rangle$ is easy. We can reflect about the hyperplane orthogonal to $|a\rangle$ by flipping the phase of the component in the direction of $|a\rangle$; i.e. carry out the transformation

$$\sum_x \alpha_x |x\rangle \rightarrow \sum_x \alpha_x (-1)^{f(x)} |x\rangle$$

2. For the reflection about $|u\rangle$, we will actually reflect about $|u\rangle$ in the N dimensional space as follows: apply the Hadamard transform $H^{\otimes n}$ to transform $|u\rangle$ to $|0^n\rangle$. Now apply a phase flip if the register contents are anything other than $|0^n\rangle$. And apply the Hadamard transform to switch back from the Hadamard basis.

0.0.3 Another approach

Let's look at the search algorithm differently, with all superpositions. The rotation about $|u\rangle$, D, is an "inversion about the mean":

(a) For $N = 2^n$, D can be decomposed and rewritten as:

$$
\begin{aligned}
D &= H_N \begin{pmatrix} -1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} H_N \\
&= H_N \left(\begin{pmatrix} -2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} + I \right) H_N \\
&= H_N \begin{pmatrix} -2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} H_N + I \\
&= \begin{pmatrix} -2/N & -2/N & \cdots & -2/N \\ -2/N & -2/N & \cdots & -2/N \\ \vdots & \vdots & \ddots & \vdots \\ -2/N & -2/N & \cdots & -2/N \end{pmatrix} + I \\
&= \begin{pmatrix} -2/N+1 & -2/N & \cdots & -2/N \\ -2/N & -2/N+1 & \cdots & -2/N \\ \vdots & \vdots & \ddots & \vdots \\ -2/N & -2/N & \cdots & -2/N+1 \end{pmatrix}
\end{aligned}
$$

Observe that D is expressed as the product of three unitary matrices (two Hadamard matrices separated by a conditional phase shift matrix). Therefore, D is also unitary. Regarding the implementation, both the Hadamard and the conditional phase shift transforms can be efficiently realized within $O(n)$ gates.

(b) Consider D operating on a vector $\left|\alpha\right\rangle$ to generate another vector $\left|\beta\right\rangle$:

$$
D \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_i \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_i \\ \vdots \\ \beta_N \end{pmatrix}
$$

If we let μ be the mean amplitude, then the expression $2\mu - \alpha_i$ describes a reflection of α_i about the mean. Thus, the amplitude of $\beta_i = -\frac{2}{N}\sum_j \alpha_j + \alpha_i = -2\mu + \alpha_i$ can be considered an "inversion about the mean" with respect to α_i.

The quantum search algorithm iteratively improves the probability of measuring a solution. Here's how:

(a) Start state is $\left|\psi_0\right\rangle = \sum_x \frac{1}{\sqrt{N}}\left|x\right\rangle$

(b) Invert the phase of $\left|a\right\rangle$ using f

(c) Then invert about the mean using D

(d) Repeat steps 2 and 3 $O(\sqrt{N})$ times, so in each iteration α_a increases by $\frac{2}{\sqrt{N}}$

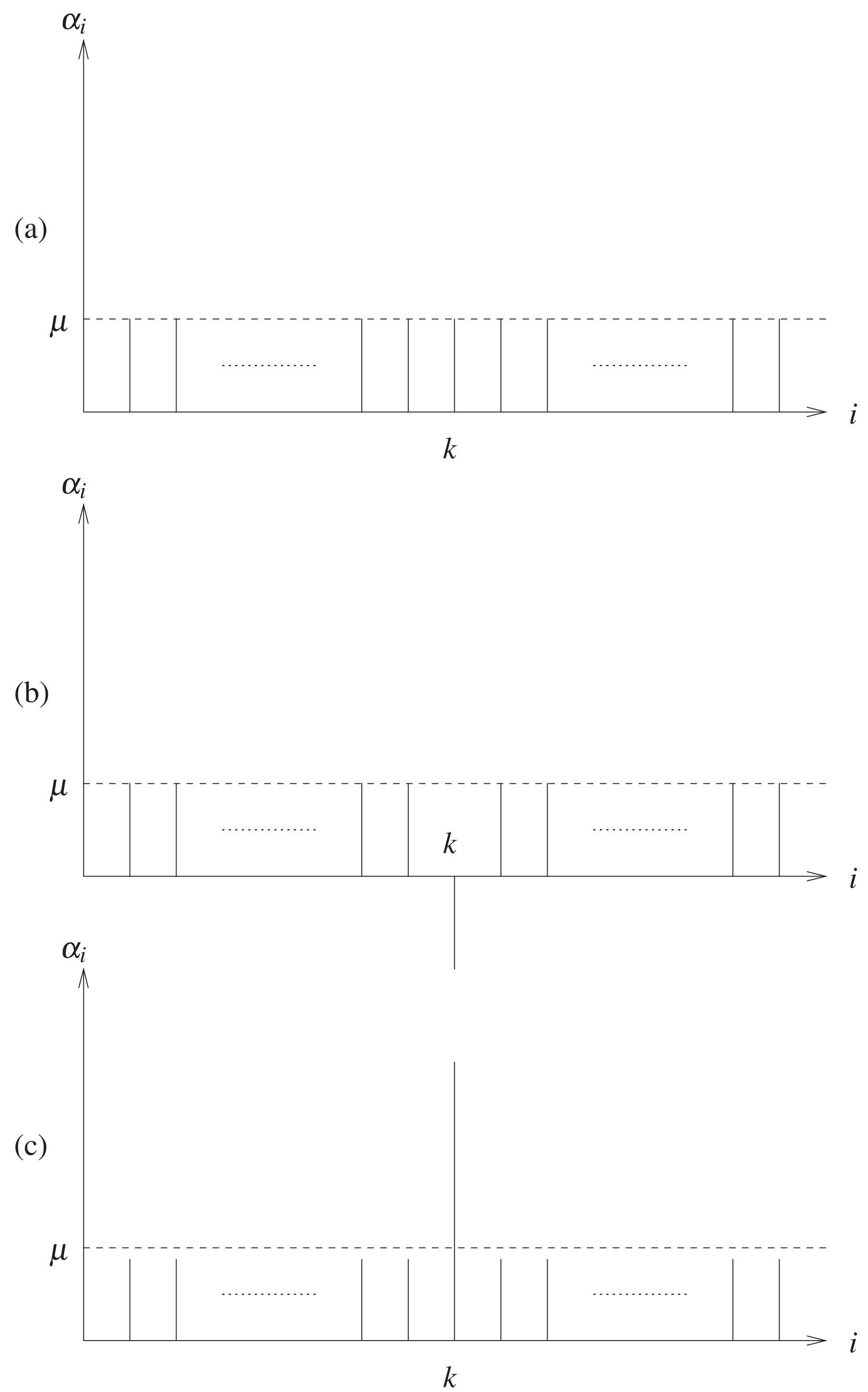

Figure 0.2: The first three steps of Grover's algorithm. We start with a uniform superposition of all basis vectors in (a). In (b), we have used the function f to invert the phase of α_k. After running the diffusion operator D, we amplify α_k while decreasing all other amplitudes.

This process is illustrated in Figure **??**.

Suppose we just want to find a with probability $\frac{1}{2}$. Until this point, the rest of the basis vectors will have amplitude at least $\frac{1}{\sqrt{2N}}$. In each iteration of the algorithm, α_a increases by at least $\frac{2}{\sqrt{2N}} = \sqrt{\frac{2}{N}}$. Eventually, $\alpha_a = \frac{1}{\sqrt{2}}$. The number of iterations to get to this α_a is $\leq \sqrt{N}$.

0.0.4 More applications

Grover's algorithm is often called a "database" search algorithm. This misnomer has been the cause of a lot of confusion, since essential that the algorithm be able to query in superposition....

But there are a number of applications of unstructured search:

(a) Find the minimum in $O(\sqrt{N})$ steps. Exercise.

(b) Approximately count elements, or generate random ones.

(c) $O(N^{1/3})$ algorithm for the collision problem....

(d) Speed up the test for matrix multiplication. In this problem we are given three matrices, A, B, and C, and are told that the product of the first two equals the third. We wish to verify that this is indeed true. An efficient (randomized) way of doing this is picking a random array r, and checking to see whether $Cr = ABr = A(Br)$. Classically, we can do the check in $O(n^2)$ time, but using a similar approach to Grover's algorithm we can speed it up to $O(n^{1.75})$ time.