CS294-4 Quantum Computing **Fall 2002**

Lecture 8: October 25

Lecturer: Umesh Vazirani *Scribes: Ronen Gradwohl, Jeremy Condit*

Searching an item in an unsorted database with size N costs a classical computer $O(N)$ running time. Can a quantum computer search for a needle in a haystack much more efficiently than its classical counterpart? Grover, in 1996, affirmatively answered this question by proposing a search algorithm that consults the database only $O(\sqrt{N})$ times. In contrast to algorithms based on the quantum Fourier transform, with exponential speedups, the search algorithm only provides a quadratic improvement. However, the algorithm is quite important because it has broad applications, and because the same technique can be used to improve solutions of NP-complete problems.

One might think of having better improvements over the search algorithm. However, it turns out that Grover's search algorithm is optimal. At least $\Omega(\sqrt{N})$ queries are needed to solve the problem. This note details two ways of thinking about the algorithm, as well as the lower bounds.

8.1 Quantum Search

8.1.1 The quantum oracle

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:

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

Another way we can implement f is put the answer register in superposition:

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

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

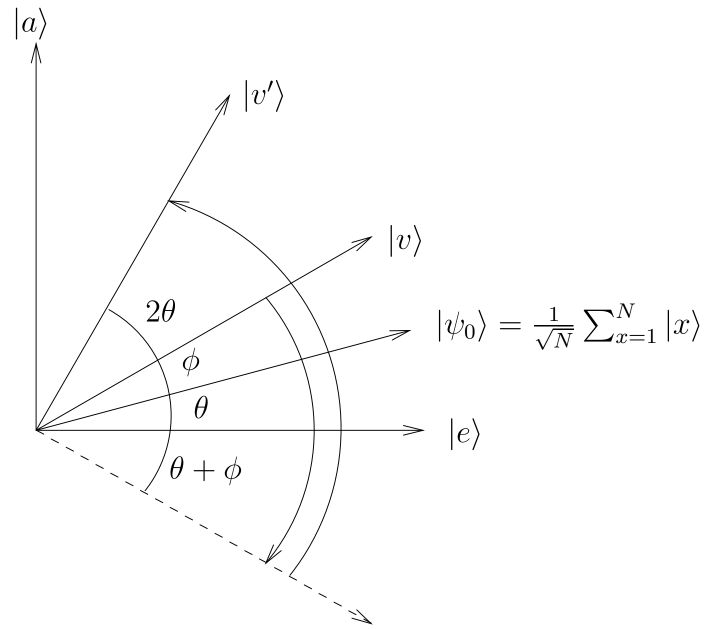

Figure 8.1: To rotate $|v\rangle$ by 2θ to $|v'\rangle$, we reflect around $|e\rangle$ and then reflect around $|\psi_0\rangle$.

8.1.2 Grover's algorithm

Grover's algorithm finds a in $O(\sqrt{N})$ steps. Consider the two dimensional subspace that consists of two states: $|a\rangle$ and $|\psi_0\rangle = \sum_x \frac{1}{\sqrt{N}}|x\rangle$. Let θ be the angle between $|\psi_0\rangle$ and $|e\rangle$, where $|e\rangle$ is the vector that is orthogonal to $|a\rangle$ in the direction of $|\psi_0\rangle$ in this subspace. See Figure 8.1 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 $|\psi_0\rangle$. This transformation is also illustrated in Figure 8.1.

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?

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

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. All we need to do is flip the phase of the component in the direction of $|a\rangle$; to accomplish this, we just use the second implementation of f that we showed earlier.

2. For the reflection about $|\psi_0\rangle$, we use the Diffusion operator D (assume $N = 2^n$), which works as follows. First, apply H_{2^n}, which maps $|\psi_0\rangle \mapsto |00\ldots0\rangle$. Then reflect around $|00\ldots0\rangle$. Finally, apply

H_{2^n} to return to the original basis. (Note that this is simply a reflection around the zero vector in the Hadamard basis.)

8.1.3 Another approach

Let's look at the search algorithm differently, with all superpositions.

Claim 8.1 *The Diffusion operator D has two properties:*

1. *It is unitary and can be efficiently realized.*
2. *It can be seen as an "inversion about the mean."*

Proof:

1. 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.

2. Consider D operating on a vector $|\alpha\rangle$ to generate another vector $|\beta\rangle$:

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

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.

$\blacksquare$

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

1. Start state is $|\psi_0\rangle=\sum_x\frac{1}{\sqrt{N}}|x\rangle$
2. Invert the phase of $|a\rangle$ using f
3. Then invert about the mean using D
4. Repeat steps 2 and 3 $O(\sqrt{N})$ times, so in each iteration α_a increases by $\frac{2}{\sqrt{N}}$

This process is illustrated in Figure 8.2.

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 $\leq\sqrt{N}$.

8.1.4 More applications

Grover's algorithm is often called a "database" search algorithm, where you can query in superposition. Other things you can do with a similar approach:

1. Find the minimum.
2. Approximately count elements, or generate random ones.
3. Speed up the collision problem.
4. 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.

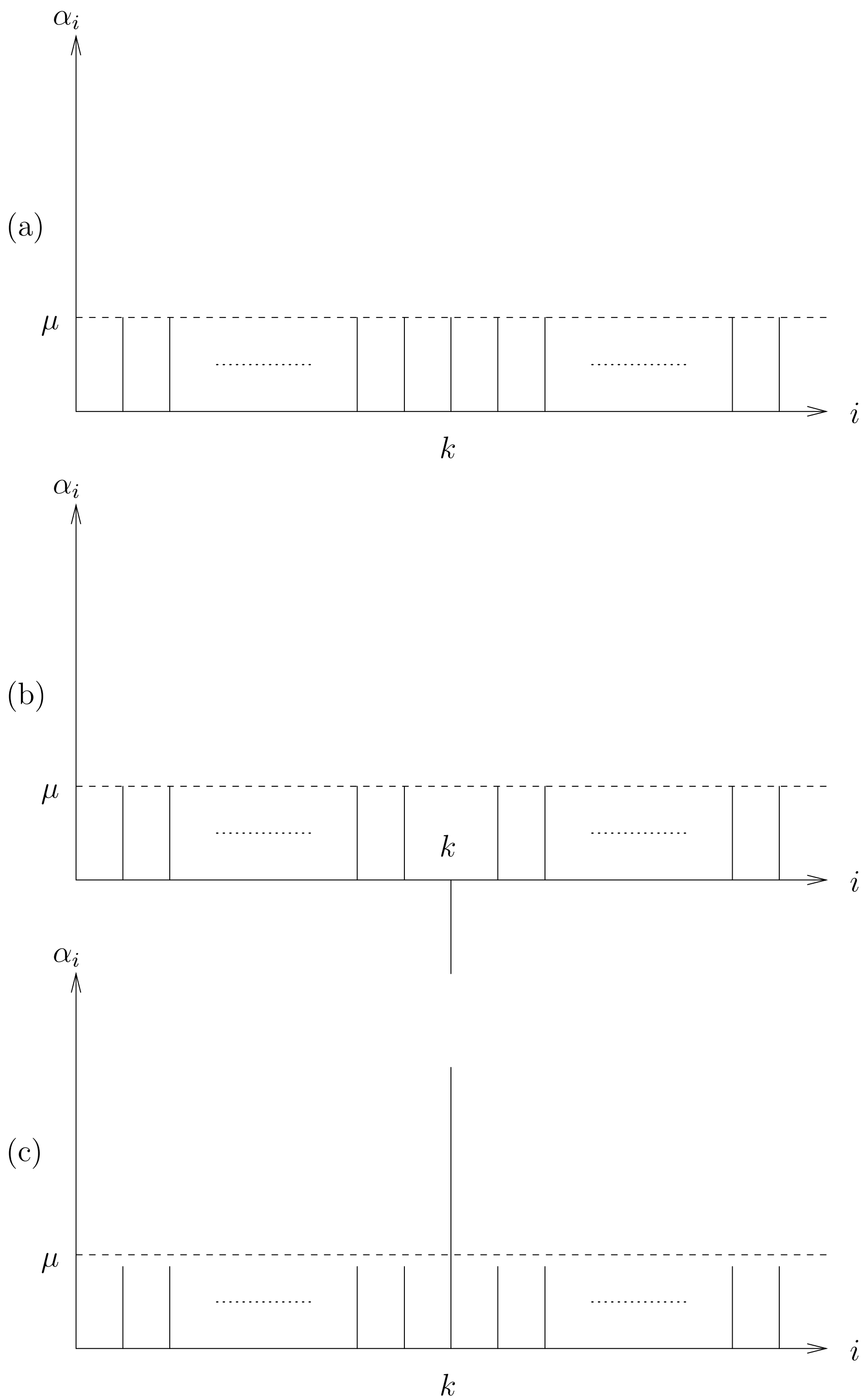

Figure 8.2: The first three steps of Grover's algorithm. We start with a uniform superposition of all basis vectors in (a). In (b), we have used the function f to invert the phase of α_k. After running the diffusion operator D, we amplify α_k while decreasing all other amplitudes.

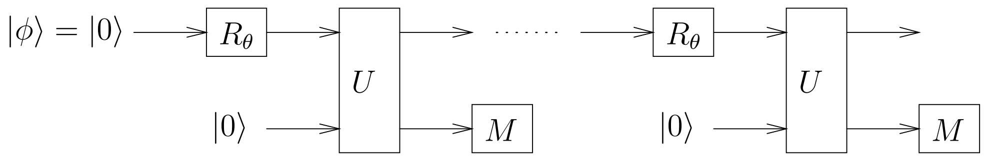

Figure 8.3: This figure shows the circuit used for finding a bomb. There are N steps, and in each step we rotate the control bit and run U.

8.2 Vaidman's Bomb

To illustrate some of the concepts behind Grover's algorithm, we'll briefly consider a problem known as Vaidman's bomb. In this problem, we have a package that may or may not contain a bomb. However, the bomb is so sensitive that simply looking to see if the bomb exists will cause it to explode. So, can we determine whether the package contains a bomb without setting it off? Paradoxically, quantum mechanics says that we can. In particular, we will demonstrate that there is a sequence of N measurements such that if the package contains a bomb, we will look with probability $1/N$, and if the package does not contain a bomb, we will look with certainty.

8.2.1 The Quantum Zeno Effect

To achieve this goal, we'll take advantage of a phenomenon known as the Quantum Zeno Effect (also referred to as the "watched pot" or the "watchdog" effect). Consider a quantum state consisting of a single qubit. This qubit starts at $|0\rangle$, and at every step we will rotate it toward $|1\rangle$ by $\theta = \pi/2N$. After one rotation, we have $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\beta = \sin\theta \approx 1/N$. After N steps, the state will be $|1\rangle$, so any measurement will return $|1\rangle$ with high probability.

Now what if we decide to measure the state after each rotation? After the first rotation, we will measure $|0\rangle$ with high probability, *but this measurement collapses the state back to* $|0\rangle$. Thus, each measurement has a high probability of yielding $|0\rangle$; the probability of getting $|1\rangle$ by the end is approximately $N\frac{1}{N^2} = \frac{1}{N}$, as opposed to the extremely high probability in the previous case.

Essentially, the Quantum Zeno Effect says that if we have a quantum state that is in transition toward a different state, making frequent measurements can delay that transition by repeatedly collapsing the qubit back to its original state.

8.2.2 Looking for the Bomb

To determine whether Vaidman's bomb exists without actually looking at it, we want to take advantage of the Quantum Zeno Effect. We'll have a control qubit that indicates whether or not we plan to look at the contents of the package, and we'll have a measurement that collapses this qubit back to $|0\rangle$ in the case that a bomb is present.

We'll assume that we have a device that can measure whether a bomb is present. We will model this device as a quantum circuit U that has one input (the control qubit $|\phi\rangle$) and one output (a qubit that is $|1\rangle$ if a bomb is definitely present). If there is no bomb, then U maps $|\phi\rangle|0\rangle \mapsto |\phi\rangle|0\rangle$; in other words, U behaves as the identity. If there is a bomb, then U maps $|0\rangle|0\rangle \mapsto |0\rangle|0\rangle$ (there's a bomb, but we didn't look) and $|1\rangle|0\rangle \mapsto |1\rangle|1\rangle$ (we looked at the bomb); that is, U behaves as a CNOT gate.

We want to figure out whether there is a bomb (i.e., we want to test U's behavior) without setting off the bomb very often. Figure 8.3 shows the circuit we will use. We initialize the control qubit $|\phi\rangle$ to $|0\rangle$. In each step of the algorithm, we rotate the control qubit toward $|1\rangle$ by θ and then run U; we'll execute the algorithm for N steps.

Consider the case where there is no bomb; our initial input is $|0\rangle|0\rangle$. If we rotate the control qubit by θ, the input to the first U gate is $(\alpha|0\rangle + \beta|1\rangle)|0\rangle$, and the output of U is the same state (since U is the identity). Measuring the output qubit always returns $|0\rangle$ and doesn't alter the state; thus, each step rotates the qubit further until $\beta = 1$ at the last measurement.

Now consider the case where there is a bomb. Once again, our initial input is $|00\rangle$. After the first rotation, the input to U is $(\alpha|0\rangle + \beta|1\rangle)|0\rangle$, and the output of U is $\alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle$. When we measure the last qubit, we have a $\beta^2 \approx 1/N^2$ probability of looking at the bomb and setting it off. Otherwise, we measure $|0\rangle$ for the output qubit, which means we didn't look at the bomb. However, *this measurement collapses the state back to* $|0\rangle|0\rangle$. Thus, subsequent steps in the algorithm will simply repeat this process. Overall, we only have a $N\beta^2 \approx 1/N$ chance of actually looking at the bomb.

8.2.3 Vaidman and Grover

To see the relationship to Grover's algorithm, consider a particularly unfortunate case where we have N packages, $N - 1$ of which contain bombs. We want to find the one package that does not contain a bomb, though we don't mind setting off a few of the bombs in the process. Grover's algorithm has a property similar to Vaidman's method where the amplitude of one target basis vector is amplified while all others are constantly diminished or reset.

The important thing to note is that it's highly counterintuitive to be able to search in $\sqrt{N}$ steps. By querying in superposition, we manage to search using fewer steps than there are locations to search!

8.3 Quantum Lower Bounds

In this section, we'll consider lower bounds for quantum algorithms. As it turns out, Grover's algorithm is optimal; you need $\Omega(\sqrt{N})$ queries to search on a quantum computer.

8.3.1 NP-Completeness

Consider SAT, the prototypical example of an NP-complete problem. An instance of this problem consists of a Boolean function $f(x_1, \ldots, x_n) = c_1 \wedge \ldots \wedge c_m$; the SAT problem asks you to determine whether there exists a satisfying assignment—that is, an input $(a_1, \ldots, a_n)$ such that $f(a_1, \ldots, a_n) = 1$. UNIQUE-SAT is a variant of SAT that poses the same problem with the restriction that f must have zero or one satisfying assignments, but no more. As it turns out, there is a randomized reduction from SAT to UNIQUE-SAT; thus, the two problems are equally hard.

We'll use the black box model when considering this problem. In this model, we know that either $f \equiv 0$ or there exists exactly one a such that $f(a) = 1$, where a is chosen uniformly at random. That is, f is treated as a black box; we can make queries to f, but we have no access to the Boolean formula itself.

In this model, UNIQUE-SAT looks very much like the problem that Grover's algorithm solves, since we can represent f as a table of 2^n entries where exactly zero or one entry is 1. Thus, when we consider the efficiency of searching on quantum computers, we're also considering the speed of NP-complete algorithms on quantum computers. What we'd like is an $O(\text{poly}(n))$ quantum algorithm for searching.

8.3.2 The Hybrid Argument

For the purposes of this discussion, we want to separate the quantum algorithm itself from the function f. We assume that the quantum algorithm is infinitely powerful (i.e., it can do any computation in one step) and focus instead on the number of queries it must make to f. All queries to f occur in superposition; that is, a single query on $\sum_x \alpha_x|x\rangle|0\rangle$ yields the output $\sum_x \alpha_x|x\rangle|f(x)\rangle$.

Theorem 8.2 *In the black box model, any quantum algorithm for determining whether there exist $x_1, \ldots, x_n$ such that $f(x_1, \ldots, x_n) = 1$ must make $\Omega(\sqrt{N})$ queries to f.*

Proof:

Consider any quantum algorithm A for solving this search problem. First do a test run of A on the function $f \equiv 0$. Let T be the number of queries that A makes to f, and let $\alpha_{x,t}$ be the amplitude with which A queries x at time t (that is, the query at time t is $\sum_x \alpha_{x,t}|x\rangle$). Now, define the query magnitude of x to be $\sum_t |\alpha_{x,t}|^2$. The expectation value of the query magnitude of x is $E_x\left(\sum_t |\alpha_{x,t}|^2\right) = T/N$. Thus $\min_x\left(\sum_t |\alpha_{x,t}|^2\right) \le T/N$. Let z be the input at which this minimum occurs; then by the Cauchy-Schwarz inequality, $\sum_t |\alpha_{z,t}| \le T/\sqrt{N}$. [1]

Let $|\phi_t\rangle$ be the states of A_f after the t-th step. Now run the algorithm A on the function g such that $g(z) = 1$ and for all $y \ne z$, $g(y) = 0$. Suppose the final state of A_g is $|\psi_T\rangle$. By the claim that follows, $|\phi_T\rangle - |\psi_T\rangle = |E_0\rangle + \ldots + |E_{T-1}\rangle$ where $|||E_t\rangle|| \le \sqrt{2}|\alpha_{z,t}|$. Using the triangle inequality and the inequality proved above, we have $|||\phi_T\rangle - |\psi_T\rangle|| \le \sum_t |||E_t\rangle|| \le \sqrt{2}\sum_t |\alpha_{z,t}| \le T\sqrt{2/N}$. This implies that the two states can be distinguished with probability at most $O(T/\sqrt{N})$ by any measurement. Thus any quantum algorithm that distinguishes f from g with constant probability of success must make $\Omega(\sqrt{N})$ queries.

■

Claim 8.3 $|\psi_T\rangle = |\phi_T\rangle + |E_0\rangle + |E_1\rangle + \ldots + |E_{T-1}\rangle$*, where* $|||E_t\rangle|| \le \sqrt{2}|\alpha_{z,t}|$*.*

Proof:

Consider two runs of the algorithm A, which differ only on the t-th step. The first run queries the function f on the first t steps and queries g for the remaining $T - t$ steps; the second run queries f on the first $t-1$ steps and g for the remaining $T - t + 1$ steps. After the first $t-1$ steps, both runs are in state $|\phi_t\rangle$. On the t-th step, one run queries f and the other queries g. The outputs of these queries differ only on the amplitude of the two basis vectors $|z\rangle|0\rangle$ and $|z\rangle|1\rangle$, so overall the output vectors differ by at most $\sqrt{2}|\alpha_{z,t}|$. Thus, at the end of the t-th step, the state of the first run is $|\phi_t\rangle$, whereas the state of the second run is $|\phi_t\rangle + |F_t\rangle$, where $|||F_t\rangle|| \le \sqrt{2}|\alpha_{z,t}|$. Now if U is the unitary transform describing the remaining $T-t$ steps (of both runs), then the final state after T steps for the two runs are $U|\phi_t\rangle$ and $U(|\phi_t\rangle + |F_t\rangle)$, respectively. The latter state can be written as $U|\phi_t\rangle + |E_t\rangle$, where $|E_t\rangle = U|F_t\rangle$. Since unitary transformations preserve length, we know that $|||E_t\rangle|| \le \sqrt{2}|\alpha_{z,t}|$. Thus, the effect of switching the queried function only on the t-th step can be described by an "error" $|E_t\rangle$ in the final state of the algorithm, where $|||E_t\rangle|| \le \sqrt{2}|\alpha_{z,t}|$.

[1] The Cauchy-Schwarz inequality says that for two vectors a and b of length T, $\left(\sum_t a_t b_t\right)^2 \le \left(\sum_t a_t^2\right)\left(\sum_t b_t^2\right)$. If we let $b_t = 1$ for all t, then we have $\left(\sum_t a_t\right)^2 \le T\sum_t a_t^2$. Thus, if $\sum_t |\alpha_{z,t}|^2 \le T/N$, then $\left(\sum_t |\alpha_{z,t}|\right)^2 \le T^2/N$.

We can transform the run A_f to A_g by a succession of T changes of the kind described above. Overall, the difference between the final states of A_f and A_g is $|E_0\rangle + |E_1\rangle + \ldots + |E_{T-1}\rangle$, where $\||E_t\rangle\| \leq \sqrt{2}|\alpha_{z,t}|$.

∎

Finally, it is useful to consider where this factor of $\sqrt{N}$ comes from. In the worst case, we query z with amplitude $1/\sqrt{N}$ at each time step. The vectors that indicate the differences at each step could all be orthogonal, in which case the total distance is the sum of the squares of each vector's length, which is about N. However, if all vectors are in the same direction, the total distance is the sum of the length of each vector, which is approximately $\sqrt{N}$. Grover's algorithm demonstrates that it is possible to align all of these vectors and achieve the factor of $\sqrt{N}$.

References

[1] Ambainis, A., **Quantum lower bounds by quantum arguments**, *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000.

[2] Bennett, C., Bernstein, E., Brassard, G., and Vazirani, U., **Strengths and weaknesses of quantum computation**, ISI Torino tech report (1994), SIAM J. Computing, 26, pp. 1510-1523 (1997).

[3] Hales,L. and Hallgren, L., **An Improved Quantum Fourier Sampling Algorithm and Applications**, *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.