CS 880: Quantum Information Processing 9/14/10

Lecture 5: More Elementary Quantum Algorithms

Instructor: Dieter van Melkebeek Scribe: John Gamble

Last class, we turned our attention to elementary quantum algorithms, examining the Deutsch, Deutsch-Jozsa, and Bernstein-Vazirani problems. In all three of these cases, we showed that quantum algorithm could do better than the best classical scheme. We then briefly introduced Simon's problem.

In this lecture, we will develop the quantum algorithm that solves Simon's problem in $\mathcal{O}(n)$ queries. We then begin analyzing Grover's quantum search algorithm.

1 Simon's Problem

In the following section, we analyze Simon's problem, a particular promise problem that is solvable exponentially faster on a quantum computer than on a classical computer. The problem statement is that, given some function $f : \{0,1\}^n \to \{0,1\}^n$ as a black box and the promise that either:

- f is one-to-one, or else
- there exists $s \neq 0^n$ such that $\forall x, y \in \{0,1\}^n$, $f(x) = f(y)$ if and only if $x \oplus y = s$,

determine into which case f falls, and if the second, determine s. Note that the second case corresponds to f being two-to-one, with each pair of inverse image points separated by exactly s.

Our analysis proceeds as follows. First, we discuss the classical complexity of this problem. Next, we will propose a quantum circuit to solve the problem. Finally, we examine the circuit to show that it has query complexity $\mathcal{O}(n)$, a result first obtained by D. R. Simon in 1994 [2].

1.1 Classical complexity

We first turn our attention to deterministic, classical algorithms. In order to solve the problem, we need to find the correct value of s, or show that there is no such $s \neq 0^n$ exists. Hence, we need to "scan" through all possible values of s as quickly as possible. Since there are 2^n choices for s, we have a trivial upper bound of $\mathcal{O}(2^n)$. Now, suppose that we have queried our function q times. Then, assuming that we have not repeated a query, we have tried $\binom{q}{2}$ pairs of inputs. Hence, we have eliminated at most $\binom{q}{2}$ possible choices of s. Since we need to check all possible values of s, to answer the problem with certainty, we need to satisfy

$$\binom{q}{2} \geq 2^n. \tag{1}$$

Applying the definition of the binomial formula, we find

$$\frac{q(q-1)}{2} \geq 2^n. \tag{2}$$

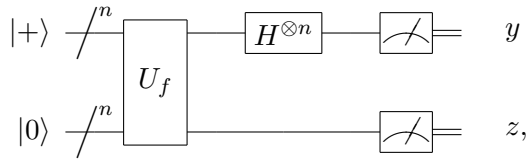
Hence, $q \gtrsim 2^{n/2}$, so that we require $\Omega\left(2^{n/2}\right)$ queries to solve the problem as a lower bound.

Probabilistically, the problem is not much different. We still need to rule out all possible values of s, of which there are 2^n. It can be shown that this also has a lower bound of $\Omega\left(2^{n/2}\right)$.

1.2 Simon's quantum algorithm

Now, we construct and analyze the quantum circuit that will enable us to solve Simon's problem in $\mathcal{O}(n)$ evaluations, exponentially faster than the classical case. Extending the general scheme that we used in the Bernstein- Vazirani algorithm, we consider

$$(a) \qquad (b) \tag{3}$$



where the $\not{}^n$ notation indicates the presence of n such wires. First, we initialize our system in the state

$$|+\rangle^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle |0\rangle^{\otimes n}, \tag{4}$$

where $N = 2^n$. Next, we apply the unitary operator corresponding to the application of f, giving us the state

$$|(a)\rangle = \frac{1}{\sqrt{N}} \sum_x |x, f(x)\rangle \tag{5}$$

at point (a). Recall from last time that acting $H^{\otimes n}$ on any computational basis state $|x\rangle$ gives

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{N}} \sum_y (-1)^{x \cdot y} |y\rangle . \tag{6}$$

Hence, at point (b) we have

$$\begin{aligned} |(b)\rangle &= \left(H^{\otimes n} \otimes I^{\otimes n}\right) |(a)\rangle \\ &= \frac{1}{\sqrt{N}} \sum_x \left(H^{\otimes n} |x\rangle\right) |f(x)\rangle \\ &= \frac{1}{N} \sum_{x,y} (-1)^{x \cdot y} |y, f(x)\rangle . \end{aligned} \tag{7}$$

We let $f(x) = z$ and define $\alpha_{y,z}$ by

$$|(b)\rangle = \sum_{y,z} \alpha_{y,z} |y, z\rangle , \tag{8}$$

where each sum runs over the usual $\{0,1\}^n$. We now perform a measurement, so we must analyze the form of $\alpha_{y,z}$ to determine the result. To do this, note that our function can either be one-to-one or two-to-one. If it is the latter, half of the $z \in \{0,1\}^n$ fall *outside* the range of f, and so the coefficients $\alpha_{y,z}$ are zero.

First, we consider the case where f is one-to-one. Then, we have

$$\alpha_{y,z} = \frac{(-1)^{x \cdot y}}{N}, \tag{9}$$

where x is the unique element such that $f(x) = z$. Next, consider the second case, where f is two-to-one. We have two sub-cases: either z is in the range of f or it is not. If it is, then

$$\alpha_{y,z} = \frac{(-1)^{x\cdot y} + (-1)^{(x\oplus s)\cdot y}}{N}, \tag{10}$$

where s is such that $f(x) = f(x \oplus s) = z$. Note that by the promise of the problem, we know that there will be exactly two elements in the inverse image, and that one can be obtained from the other by adding s. Finally, as mentioned before, if z is outside the range of f, then $\alpha_{y,z} = 0$. Now that we have worked out the form of $\alpha_{y,z}$, we can analyze the resulting outputs probabilistically, which we do in the next section.

1.3 Analysis of the measured output

Now that we know the possible values of $\alpha_{y,z} = 0$, we first need to use them to deduce what our circuit outputs, and with what probability. Specifically, we are interested in the top-half of our register, which, after measurement, will be in the computational basis state $|y\rangle$ with probability

$$\Pr(y) = \sum_z |\alpha_{y,z}|^2 . \tag{11}$$

Hence, from equation (9), if f is one-to-one we expect that for any y, $\Pr(y) = 1/N$. If f is two-to-one, then from equation (10) we get probability

$$\begin{aligned}
\Pr(y) &= \sum_z \begin{cases} \left|\frac{(-1)^{x\cdot y}+(-1)^{(x\oplus s)\cdot y}}{N}\right|^2 & \text{if } z \in \text{range}(f) \\ 0 & \text{if } z \notin \text{range}(f) \end{cases} \\
&= \frac{N}{2}\left|\frac{1+(-1)^{s\cdot y}}{N}\right|^2 \\
&= \begin{cases} \frac{2}{N} & \text{if } s\cdot y = 0 \\ 0 & \text{if } s\cdot y \neq 0 \end{cases},
\end{aligned} \tag{12}$$

where s is such that $f(x) = f(x \oplus s) = z$. In summary, if our function falls in the first case of our problem and is one-to-one, we get a uniform probability distribution across all possible outputs. However, if our function falls in the second case and is two-to-one, we get a uniform distribution across all outputs that are orthogonal to s. Hence, outputs that are not orthogonal to s are never observed.

By running our circuit repeatedly, we can record what output values we receive, and thus eventually determine s (or that we are in case one). However, we would like to know how many times we need to iterate before being certain of s. In order to accomplish this, we present the following lemma.

Lemma 1. *For any vector space $\mathcal{S}$ of dimension d over $\mathbf{GF}(2)$, pick d vectors $y_1, y_2, \ldots, y_d$ uniformly at random. Then, $\Pr(\text{span}(y_1, y_2, \ldots, y_d) = \mathcal{S}) \geq \delta$, where δ is a universal constant independent of $\mathcal{S}$.*

Proof. First, note that since we require a set of d vectors to span a space of dimension d, all must be chosen to be linearly independent. We then consider choosing the d vectors, one at a time,

uniformly at random. In the first step, we must only avoid choosing the zero vector (the string of all zeros), so our success probability at the first step is

$$P_1 = \frac{2^d - 1}{2^d}. \tag{13}$$

Now that we have fixed y_1, we need to make sure $y_2 \notin \text{span}\,(0, y_1)$, which has two members. Hence, the probability of the successfully choosing both the first and second vectors is

$$P_2 = \frac{2^d - 1}{2^d}\frac{2^d - 2}{2^d}. \tag{14}$$

To pick y_3, we need to require $y_3 \notin \text{span}\,(0, y_1, y_2)$, which has $2^{3-1} = 4$ members. Continuing like this, the probability for choosing all d vectors successfully is

$$\begin{aligned} P_d &= \frac{2^d - 1}{2^d}\frac{2^d - 2}{2^d}\cdots\frac{2^d - 2^{d-1}}{2^d} \\ &= \prod_{i=1}^{d}\left(1 - \frac{1}{2^i}\right) \\ &\geq \prod_{i=1}^{\infty}\left(1 - \frac{1}{2^i}\right). \end{aligned} \tag{15}$$

Taking the logarithm of both sides, we have[1]

$$\ln(P_d) \geq \sum_{i=1}^{\infty}\ln\left(1 - \frac{1}{2^i}\right). \tag{16}$$

Asymptotically, as $i \to \infty$, we note that

$$\ln\left(1 - \frac{1}{2^i}\right) \sim -\frac{1}{2^i}. \tag{17}$$

Hence, since

$$-\sum_{i=1}^{\infty}\frac{1}{2^i} = -1, \tag{18}$$

we know that the series in equation (16) converges to some finite value. So, there is a positive δ such that $P_d \geq \delta$, independent of d, as desired. □

Now, we return to the analysis of our quantum algorithm. We run the circuit $n-1$ times, each time receiving a random output y. If we are the second case (where f is two-to-one), the outputs are uniformly distributed amongst all basis states orthogonal to s. Hence, we have picked $n-1$ uniformly distributed vectors from a space of dimension $n-1$, so our lemma applies, and we know that our vectors span the space orthogonal to s with constant error.

Since we can easily check if the dimension of the space spanned by the output vectors is indeed $n-1$, in constant error we can determine a unique candidate for s. We can then check that this is

[1] Numerically, $\sum_{i=1}^{\infty}\ln\left(1 - \frac{1}{2^i}\right) \approx -1.24206$, providing a bound of $\delta \approx 0.288289$.

the correct value of s by checking $f(0) = f(s)$. If the equality holds then we know with certainty that we are in case two and have the correct value of s. Conversely, if the equality does not hold, then we know that we are in case one.

Hence, with constant, finite probability we will know that we have the correct answer. Otherwise, we will know that we need to start over, so we need to run our circuit $\mathcal{O}(1)$ times to boost our probability of success arbitrarily close to unity. This is an example of a *Las Vegas algorithm*, an algorithm that will succeed with finite probability, and will always alert us to failures. Las Vegas algorithms themselves are a special class of *Monte Carlo algorithms*, where we will still get a correct answer with finite probability, but we may not be able to tell if an answer is incorrect.

Exercise 1. *Consider a function $f : \{0,1\}^2 \to \{0,1\}$ with the promise that $\left|f^{-1}(\{1\})\right| = 1$. That is, we are promised that the inverse image set of $\{1\}$ has size one. Construct a quantum algorithm that determines $f^{-1}(\{1\})$ with zero error while using only one function query.*

2 Introduction to quantum search

In this section, we begin considering Grover's algorithm for quantum search, first formulated by L. K. Grover in 1996 [1]. In this problem, we are given some black-box function $f : \{0,1\}^n \to \{0,1\}$ and the size of the inverse image of 1, $t = \left|f^{-1}(1)\right|$. Our goal is to find an x such that $f(x) = 1$. Later on, we will consider relaxing the assumption that t is known, but for now will assume that it is given.

Before we start looking for quantum methods to solve the search problem, we will briefly investigate its classical complexity. Deterministically, the worst-case scenario is that f evaluates to zero for as many queries as possible before finally outputting one. Hence, we may need to try $N - t$ times before we know that there are only t options remaining, so they must evaluate to one. If we do not know the value of t, then we must try once more, since we would not be sure where to stop. If we allow randomization and select our trials uniformly at random, then we require $\Omega(N/t)$ function queries.

It turns out that the quantum algorithm will run in $\mathcal{O}\left(\sqrt{N/t}\right)$, and that we will be able to argue that this is optimal. Next time, we will talk about implementation in a circuit, but for now we will develop a conceptual picture for how such an algorithm could work by studying amplitudes of a quantum state. First, we consider initializing a quantum register of n qubits, corresponding to N computational basis states, in a uniform superposition,

$$|\psi\rangle = \sum_x \alpha_x |x\rangle , \tag{19}$$

where initially $\alpha_x = 1/\sqrt{N}$, as shown in the first panel of figure 1. Next, we apply phase kickback, that multiplies each α_x by -1 whenever $f(x) = 1$. An example of this for which $t = 3$ is depicted in the second panel (step (a)) of figure 1. After that, we reflect the value of each amplitude about the average (shown in the figure by a blue dashed line). If $t \ll N$, then the application of the phase kickback did not have much of an effect of the average value of $1/\sqrt{N}$, so the new values of the target state amplitudes are approximately $3/\sqrt{N}$, as depicted in the third panel of figure 1 (step (b)).

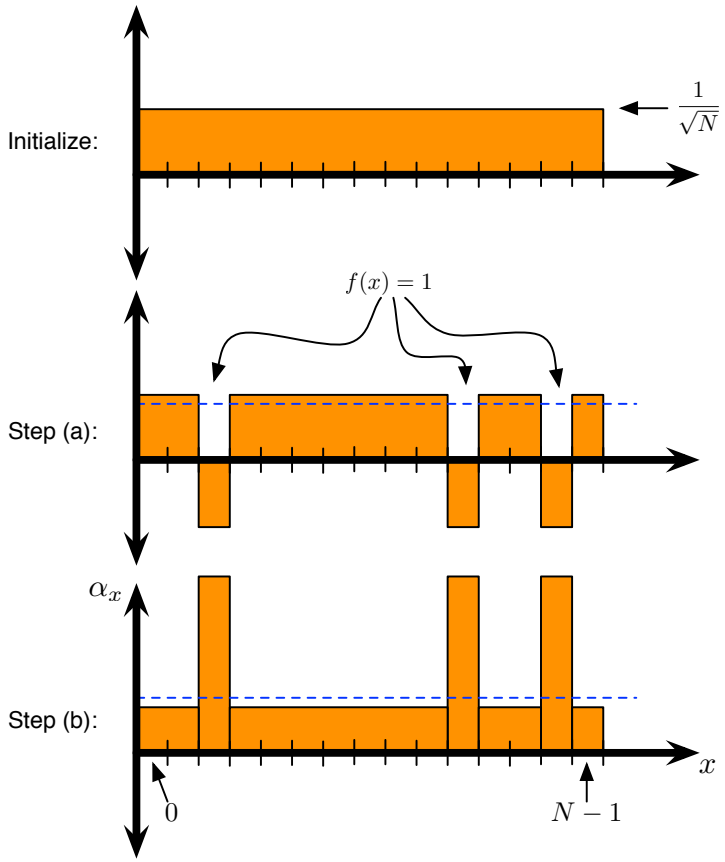


Figure 1: A graphical picture of Grover's search algorithm by studying quantum amplitudes. In the first panel, we initialize a quantum register of n qubits to a uniform superposition. In the second panel, we apply a phase kickback, assuming that f evaluates to one three times. In the third panel, we reflect all the amplitudes about the average amplitude.

By repeating steps (a) and (b), we can boost the amplitudes of the target states further. For instance, after applying (a) and (b) once more, the amplitudes on the target states are approximately $5/\sqrt{N}$. Note, however, that eventually it will not be a good approximation that the average does not move. In fact, eventually the amplitudes of the target qubits will become sufficiently high that the the average amplitude after an iteration of step (a) will be negative, causing our system return toward the uniform initial state.

It turns out that the repeated application of any unitary operator will eventually return us close to our original state. In order to see this, consider an arbitrary unitary operator U. Since we are dealing only with finite matrices, and hence spaces of finite volume, it follows that for any l and ϵ, there exist some $k > l$ such that

$$\left\| U^k - U^l \right\| \leq \epsilon. \tag{20}$$

That is, if we apply U enough, we will return arbitrarily close to a previous state. But now,

$$\begin{aligned} \left\| U^k - U^l \right\| &= \left\| U^l \left(U^{k-l} - I \right) \right\| \\ &= \left\| U^{k-l} - I \right\| \leq \epsilon, \end{aligned} \tag{21}$$

so there is some iteration that will bring U arbitrarily close to identity. Hence, we will need to develop a strategy for determining when to stop iterating and measure.

Next time, we will discuss the implementation of this process as a quantum circuit. We will then discuss how to determine the optimal stopping point so that we do not go back toward our initial state.

References

[1] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, New York, NY, USA, 1996. ACM.

[2] D.R. Simon. On the power of quantum computation. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:116–123, 1994.