CMSC 657: Introduction to Quantum Information Processing
Lecture 13

Instructor: Daniel Gottesman

Fall 2024

1 Grover's algorithm continued

1.1 Grover's algorithm with multiple marked elements

When there are t marked elements, we can follow essentially the same analysis.

$$|S\rangle = \sum_{x|O(x)=1} |x\rangle \tag{1}$$

$$|T\rangle = \sum_{x|O(x)=0} |x\rangle \tag{2}$$

$$|U\rangle = \sum_{\text{all } x} |x\rangle = |S\rangle + |T\rangle. \tag{3}$$

Now

$$\cos\theta = \frac{1}{\sqrt{N(N-t)}}\langle U|T\rangle = \sqrt{\frac{N-t}{N}} \tag{4}$$

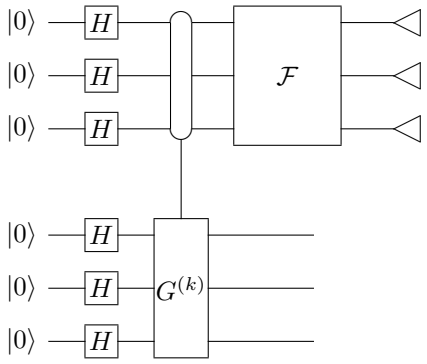
$$\sin\theta = \sqrt{\frac{t}{N}}. \tag{5}$$

We can run the same algorithm, but because θ is bigger, we should stop sooner, specifically after $M \approx (\pi/4)\sqrt{N/t}$ iterations. If we stop then, we will with high probability get a *random* x that satisfies $O(x) = 1$. There is no real way to control which one we get (since we don't know what they are).

If we know how many solutions there are, this works fine. But what do we do if we don't know how many solutions there are? We need to stop at the right time or the success probability will decrease. We don't have to know the exact number of solutions — if we have close to the right number, we will still get an $O(1)$ probability of success, although it won't be very close to 1. That is sufficient, since we can run it a $O(1)$ times and check the measured answer each time. The odds are good we will eventually get a marked element. To get to this point, we need some method of approximately counting the number of marked elements.

Let us imagine running Grover's algorithm for a variable time and plotting the amplitude of $|S\rangle$. It oscillates with period $\pi\sqrt{N/t}$. We can find t by finding the period. We already know how to do that: use Shor's algorithm!

Thus, what we want to do is to create an ancilla $\sum_k |k\rangle$ and run Grover's algorithm for k steps. Then do the Fourier transform on the ancilla and measure it, giving an estimate of the period.



We should let k run from 0 to $O(\sqrt{N/s})$ queries if we think there are at least s solutions. That way, we will be sure of getting at least one period in the worst case, the minimum for us to get any kind of estimate of the period. If you want a more accurate count, you can also run for longer, although it turns out that if you want an exact count, you need to use $O(N)$ queries in total. After counting, we can then run Grover's algorithm for roughly the correct number of steps to get a marked item.

If you just do this and s is much less than the true number t of solutions, you will end up using many more than $O(\sqrt{N/t})$ queries. We can get around this by first running the approximate counting algorithm with $s = N/2$. If the true number of solutions is much less than this, the amplitude of $|S\rangle$ will basically be flat (and very small), and period finding will probably find a period of 0 (which is what you get if you take the Fourier transform of a constant function). If that happens, run the approximate counting algorithm again with $s = N/4$, then $N/8$, and so on to $s = N/2^m$. Once we get $s \approx t$, we will likely find a non-zero period and can then find a marked element. In total, the number of queries we use is

$$O\left(\sum_{m=1}^{\log N/t} 2^{m/2}\right) = O\left(2^{(\log N/t)/2}\right) = O(\sqrt{N/t}). \tag{6}$$

1.2 Example: Finding Collisions

As mentioned, Grover's algorithm is quite flexible and can be used to solve things which are not quite as straightforward as an unordered search. As an example, consider finding collisions: Given a function $f(x)$ which is 2–1; that is, for all y, there are exactly two values (x_0, x_1) of x such that $f(x_0) = f(x_1) = y$. The goal is to find a collision, that is any pair (x_0, x_1) such that $f(x_0) = f(x_1)$. The collision problem is important, for instance, for breaking security of cryptographic hash functions. It can also be analyzed rigorously by consider f to be an oracle.

Classically, the oracle version of this problem takes $\Theta(N^{1/2})$ queries when there are N possible inputs. This follows from birthday-paradox arguments. Quantumly, it takes $\Theta(N^{1/3})$ queries. The algorithm to accomplish this uses Grover's algorithm as a subroutine.

Specifically, first make $N^{1/3}$ classical queries to the oracle, retrieving $N^{1/3}$ distinct values of $f(x)$ along with the appropriate input. (If two of these $f(x)$ values happen to be the same, then we can return those as a collision.) Make a list of those values. Then, on the remaining $N - N^{1/3}$ values of x, search using Grover's algorithm where an element x is marked if $f(x) = f(x')$ for some x' on the list. There are exactly $N^{1/3}$ marked elements in this case, so Grover's algorithm takes a time $O((N/N^{1/3})^{1/2}) = O(N^{1/3})$.

Note that Grover's algorithm by itself doesn't use much space, only $O(\log N)$, but this application to the collision problem uses $N^{1/3}$ space to store the initial list.

1.3 Amplitude Amplification

Another application of Grover's algorithm is as a subroutine of other quantum computations. Suppose we have some unitary U which acts as follows:

$$U|0\dots0\rangle = \sqrt{1-\epsilon}|\psi\rangle + \sqrt{\epsilon}|\phi\rangle = |U\rangle. \tag{7}$$

Here $\langle\psi|\phi\rangle = 0$, and we want to produce the state $|\phi\rangle$ more reliably.

Suppose we have some efficient measurement M which can reliably distinguish between $|\psi\rangle$ and $|\phi\rangle$. (A measurement always exists since they are orthogonal, but it might be hard to implement efficiently.) We can then purify M into a unitary V putting the result in an ancilla qubit, followed by a measurement of the ancilla (1 indicating $|\phi\rangle$). Then V followed by Z on the ancilla followed by $V^\dagger$ gives a phase of -1 to $|\phi\rangle$ and none to $|\psi\rangle$. This looks a lot like the $(-1)^O$ operation we had before.

We can also do the equivalent to $(-1)^U$: If we have $|U\rangle$ and perform $U^\dagger$ on it, we get back $|0\dots0\rangle$. A phase shift conditioned on all 0's, followed by U again, will thus give a -1 phase to $|U\rangle$ and no phase to any orthogonal state.

We can then run Grover's algorithm with alternating $(-1)^O$ and $(-1)^U$ steps like this to induce a rotation in the $|U\rangle/|\phi\rangle$ plane, resulting in an increase in the amplitude of $|\phi\rangle$. When ϵ is small, we can get the amplitude of $|\phi\rangle$ to be close to 1 in $\Theta(1/\sqrt{\epsilon})$ iterations.

1.4 Lower Bound on Unstructured Search

There is also a lower bound proven for the quantum query complexity showing that the quantum query complexity is $\Omega(\sqrt{N})$. Thus, the quantum query complexity is exactly $\Theta(\sqrt{N})$. In fact, it turns out that the constant in Grover's algorithm is optimal as well.

2 Hamiltonians and the Schrödinger Equation

The next (and last) quantum algorithm we will discuss is one of the first ones invented, and yet remains one of the most important practically. Historically, two people more-or-less independently had the idea that quantum computers could provide a computational speed-up over classical computers. One of them was David Deutsch, and his example was Deutsch's algorithm, which was improved by Jozsa to make the Deutsch-Jozsa algorithm we saw before. The other was Richard Feynman, who observed that in physics, it was very difficult for classical computers to simulate the behavior of quantum systems because of the exponentially large Hilbert space. He then pointed out that a quantum computer ought to be able to simulate other quantum systems well, which implied a computational speedup.

Most of the systems studied today by physicists have some level of quantumness in them, and quantum mechanics is also highly relevant in chemistry and other areas. In some cases, classical simulation is really sufficient, either because the system is not very quantum or because there are clever classical algorithms that work using specific properties of the system. Still, there are plenty of other cases where classical computers are insufficient and quantum computers would be needed. This would be relevant, for instance, in quantum chemistry calculations, in materials science and theoretical condensed matter, in nanoscience, and in high energy physics.

So, what is the task? The behavior of a *closed* quantum system (one without contact with the outside world) is described by the Schrödinger equation. The system has a linear operator called the *Hamiltonian* H, and Schrödinger's equation is

$$i\hbar\frac{d|\psi\rangle}{dt} = H|\psi\rangle. \tag{8}$$

Here, $|\psi\rangle$ changes continuously as a function of time t. (This is known as the "Schrödinger picture." There is an alternative approach where the time evolution is absorbed into the operators, which is called the "Heisenberg picture.") H is an operator that represents the *energy* of the system. H must be Hermitian so that the energies are real numbers. The eigenstates of H are states with a definite value of the energy, and

the eigenvalue of H for a specific eigenstate is the energy of that state. The *ground state* is the eigenstatestate with the lowest energy, which is important at low temperatures.

Open systems are ones that interact with the environment. They still have a Hamiltonian, but have additional terms describing how they interact with the environment. A full description of the behavior of the system would require understanding the behavior of the environment as well, but in some cases, you can approximate the behavior of the environment and get sensible equations to describe the evolution of the system by itself. Quantum algorithms to simulate the behavior of open quantum systems are not well-developed, so we will focus on the case of closed systems.