CMSC 657: Introduction to Quantum Information Processing
Lecture 12

Instructor: Daniel Gottesman

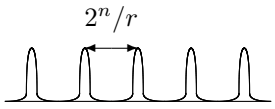
Fall 2024

1 Shor's Algorithm Continued

In the last lecture, we saw how Shor's algorithm works for the unrealistic case where $r|2^n$.

1.1 $r \not| 2^n$

In the case where $r|2^n$, the function is perfectly periodic on the domain and the Fourier transform gives us sharp spikes at multiples of $2^n/r$. When r does not divide 2^n, x^a is not perfectly periodic as there is a glitch when we wrap from $a = 2^n - 1$ to $a = 0$. However, if we choose n to be fairly large compared to r, the function will be *almost* periodic. Instead of getting sharp peaks at multiples of $2^n/r$, we get narrow peaks centered around multiples of $2^n/r$ (which are not integers now).



The larger an n we choose, the closer we get to something truly periodic and the narrower peaks we get after the Fourier transform. Now the measurement outcome at the end gives a random integer b between 0 and $2^n - 1$ which, with high probability, is close to $c2^n/r$ for some random c. We would like to find c/r. How can we do that?

The two numbers $b/2^n$ and c/r are both rational numbers but we should pick n big enough so that r (which is at most N) is much smaller than 2^n. Therefore, b is (with high probability) a special value for which $b/2^n$ has a good approximation with much lower denominator.

Since $b/2^n$ is rational, it has a finite *continued fraction* expansion:

$$\frac{b}{2^n} = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{c_3 + \ldots}}}. \tag{1}$$

There is an efficient algorithm to calculate continued fraction expansions: subtract off integer part, invert, repeat. When $b/2^n$ is sufficiently close to c/r, it turns out that they have very similar continued fraction expansions:

$$\frac{c}{r} = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \ldots}}. \tag{2}$$

The difference is that the continued fraction expansion for c/r ends earlier. This gives us a set of good candidates for c and r: We truncate the continued fraction expansion for $b/2^n$ in various places and evaluate

those truncated expansions as rational numbers. Now we repeat the whole procedure a few times. Each time we do it, we will get a new set of candidates for c and r. c will be different each time but r should mostly be the same, although we may occasionally get a b that is far from the peak or a case where c and r are not relatively prime. Neverthless, with only a few repetitions, the odds are high of seeing some potential values for r showing up repeatedly. It turns out that by picking $2^n \approx N^2$, we get sufficiently narrow peaks that, with high probability, the measurement outcome b at the end will be sufficiently close to $c2^n/r$ that we have a good chance of getting the right r out of this procedure.

A good chance does not mean correct, but if the r we deduce turns out to be even, we can go through the reduction we discussed before and get potential factors of N. Then we can test out potential factors by dividing them into N. This is something that can be done efficiently.

2 Grover's Algorithm for Unstructured Search

2.1 Unstructured Search

Unstructured Search: (Also called "database search" sometimes) This is an oracle problem $O : \mathbb{Z}_N \to \mathbb{Z}_2$. Find if $\exists\, x_0$ such that $O(x_0) = 1$. Elements x with $O(x) = 1$ are called "marked elements". Frequently we want to actually find x_0, not just determine if one exists.

The unstructured search problem is an abstraction of NP-complete problems. In particular, one could attempt to search through potential witnesses for an NP problem. The oracle is implemented as the verification function that checks witnesses. The marked elements are those witnesses which pass the verifier.

The idea of treating this as an unstructured search is we are trying to solve it without using any knowledge about the verifier except that it exists. The knowledge that the verifier has a poly-time circuit would be extra structure not used here, as would any additional structure possessed by a particular NP-complete problem. The intuition here is that while NP-complete problems might have additional structure, there is no real way of using that structure.

The lower bound on the classical query complexity, including the randomized query complexity, for unstructured search is that $\Omega(N)$ queries are needed. This makes sense because if only one random element is marked, we will have to try this many queries to find it with good probability.

There is a quantum algorithm, called Grover's algorithm, which solves unstructured search with $O(\sqrt{N})$ queries. The intuition is that classically random guessing gives a $1/N$ probability per guess of finding the marked element. In the quantum case, we guess coherently, giving an amplitude $1/\sqrt{N}$. Grover's algorithm ensures that the guesses will add up coherently so that we need only make $O(\sqrt{N})$ of them to find the marked element.

Grover's algorithm is very widely useful even though it only gives a polynomial speedup over classical computation. Basically anything that involves trying a long list of possibilities without learning from previous failures can be sped up using Grover's algorithm. This includes searching for witnesses for an NP-complete problem, searching for keys to a cryptosystem not completely broken by quantum computers, and speeding up a lot of other mundane search problems. It has some more exotic applications as well.

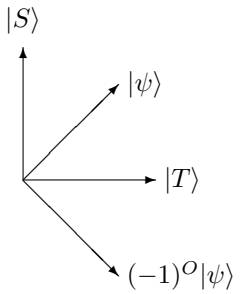
2.2 Grover's algorithm with a single marked element

First let us specialize to the case where there is exactly one marked element x_0. The unmarked elements can be treated symmetrically. It thus makes sense to work in a 2-dimensional Hilbert space spanned by

$$|S\rangle = |x_0\rangle \tag{3}$$

$$|T\rangle = \sum_{x \neq x_0} |x\rangle. \tag{4}$$

These two states are orthogonal. (Of course, we don't know initially what exactly this subspace is; but all the states we get will be in this subspace.)



What operations do we have available? Using 1 oracle call and phase kickback, we can map

$$|\psi\rangle = \alpha|S\rangle + \beta|T\rangle \mapsto (-1)^O|\psi\rangle = -\alpha|S\rangle + \beta|T\rangle. \tag{5}$$

$(-1)^O|\psi\rangle$ is the reflection of $|\psi\rangle$ over the $|T\rangle$ axis.

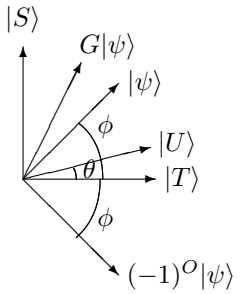
We can also do some operations without involving the oracle at all: For instance, we can create the state

$$|U\rangle = \sum_x |x\rangle = |S\rangle + |T\rangle. \tag{6}$$

We make $|U\rangle$ by applying $H^{\otimes n}$ to $|00\ldots0\rangle$.

We have one reflection, how about another? We can reflect over $|00\ldots0\rangle$ using the n-qubit controlled-sign gate F_0: $F_0|00\ldots0\rangle = |00\ldots0\rangle$, $F_0|a\rangle = -|a\rangle$ for $a \neq 00\ldots0$. (This requires a number of gates, though only poly(n), but no oracle calls.) Of course, $|00\ldots0\rangle$ is not in our subspace (unless x_0 happens to be all 0) but $H^{\otimes n}$ relates $|U\rangle$ to $|00\ldots0\rangle$. Therefore we can reflect over the state $|U\rangle$ using

$$(-1)^U = H^{\otimes n} F_0 H^{\otimes n}. \tag{7}$$



Now what happens if we perform these two reflections one after the other?

$$G = (-1)^U(-1)^O. \tag{8}$$

The angle between $(-1)^O|\psi\rangle$ and $|T\rangle$ is also ϕ, so the angle between $(-1)^O|\psi\rangle$ and $|U\rangle$ is $\phi+\theta$. Therefore, the angle between U and $G|\psi\rangle = (-1)^U[(-1)^O|\psi\rangle]$ is $\phi + \theta$ in the other direction. Thus, the angle between $G|\psi\rangle$ and $|\psi\rangle$ is 2θ.

The overall effect of this procedure is to rotate $|\psi\rangle$ an angle 2θ away from $|T\rangle$ and towards $|S\rangle$. This is progress! If we do it about $\pi/4\theta$ times, we should be very close to $|S\rangle$ (although actually we are still off by an angle θ).

What is θ? It is the angle between $\frac{1}{\sqrt{N}}|U\rangle$ and $\frac{1}{\sqrt{N-1}}|T\rangle$. (We are about to take an inner product, so we need to be careful about normalization for once.)

$$\cos\theta = \frac{1}{\sqrt{N(N-1)}}\langle U|T\rangle = \frac{N-1}{\sqrt{N(N-1)}} = \sqrt{\frac{N-1}{N}}. \tag{9}$$

Thus, $\sin\theta = \sqrt{1-\cos^2\theta} = 1/\sqrt{N}$. When N is large, we immediately see that we need about $\sqrt{N}$ queries.

To be more precise, we have the algorithm:

1. Initialize the state to $|\psi_0\rangle = |U\rangle = H^{\otimes n}|00\ldots0\rangle$. We have the angle $\phi_0 = \theta$.
2. Apply $G = (-1)^U(-1)^O$ M times. Each application of G uses 1 oracle call. After the kth query, we have the state
$$|\psi_k\rangle = G|\psi_{k-1}\rangle. \tag{10}$$
The angle between $|\psi_k\rangle$ and $|T\rangle$ is $\phi_k = \phi_{k-1} + 2\theta$.
3. Measure the register to get a candidate x_0.

Therefore, after k iterations, we have the state

$$|\psi_k\rangle = \cos\phi_k|T\rangle + \sin\phi_k|S\rangle \tag{11}$$
$$= \cos((2k+1)\theta)|T\rangle + \sin((2k+1)\theta)|S\rangle. \tag{12}$$

We want to end when $(2M+1)\theta$ is as close as possible to $\pi/2$. Thus, M should be an integer near $(\pi/4)/\sin^{-1}1/\sqrt{N} \approx (\pi/4)\sqrt{N}$.

Since we do not get exactly the $|S\rangle$ state, there is some chance that the measurement will not produce the marked state, so after measuring, we should use one more oracle call to verify that our proposed x_0 does in fact have $O(x_0) = 1$. Assuming it does, we end here; otherwise, we have to repeat the procedure. If we've tried many times without finding a marked element, we can instead conclude that there is none.

Notice that if we do too many iterations, the probability of getting a marked element starts to decrease again, eventually to close to 0.