Lecture 6: Simon's algorithm

February 2, 2006

The reason why we looked at the particular "black-box" problems from the previous couple of lectures is because they give examples in which quantum algorithms give some advantage over classical algorithms. These problems seem rather unnatural, and it is difficult to imagine any realistic setting in which one would need or want to solve these problems. But perhaps more importantly, so far they have not demonstrated any significant advantage of quantum algorithms over probabilistic algorithms.

The next problem we will study will still be in the category of completely artificial problems, but now the advantage of quantum over probabilistic is quite significant: a linear number of queries will be needed by the quantum algorithm whereas an exponential number will be needed for any classical probabilistic algorithm. Also, although the problem is artificial, it is surprisingly close to a very natural and important real-world problem as we will see.

Simon's problem

The input to the problem is a function of the form

$$f : \{0,1\}^n \to \{0,1\}^n$$

for a positive integer n. Access to this function is restricted to queries to the black-box transformation B_f as before. Similar to the Deutsch-Jozsa problem, the function f is *promised* to obey a certain property, although the property is slightly more complicated than before. The property is that there exists a string $s \in \{0,1\}^n$ such that

$$[f(x) = f(y)] \Leftrightarrow [x \oplus y \in \{0^n, s\}]$$

for all $x, y \in \{0,1\}^n$. The goal of the problem is to find the string s.

For example, if $n = 3$, then the following function is an example of a function that satisfies the required property:

x	$f(x)$
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

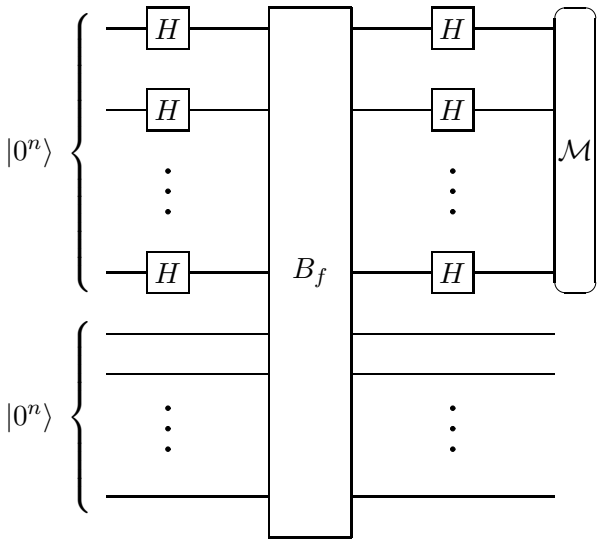
Specifically, the string s is 110. Every output of f occurs twice, and the two input strings corresponding to any one given output have bitwise XOR equal to $s = 110$.

Note that the possibility that $s = 0^n$ is not ruled out—in this case the function f is simply required to be a one-to-one function.

Intuitively this is a very hard problem classically, even if one uses randomness and accepts a small probability of error. We will not go through the proof of this fact, because (like many lower-bound proofs) it is harder than one might expect and I prefer that our focus remain on the quantum algorithmic aspects of this problem. However, the intuition is reasonably simple: if you want to solve the problem classically you need to find two different inputs x and y for which $f(x) = f(y)$. There is not necessarily any structure in the function f that would help you to find two such inputs—without any additional constraints on f you are as likely to find two such inputs by randomly guessing them as you would using any other method. But you would need to guess $\Omega(\sqrt{2^n})$ different inputs before being likely to find a pair on which f takes the same output.

Simon's algorithm

A quantum algorithm (called Simon's Algorithm) for solving this task consists of iterating the following quantum circuit and doing some classical post-processing:



Note that the B_f gate has a different (but very similar) form from before because the input and output of the function f are n-bit strings:

$$B_f \left|x\right\rangle \left|y\right\rangle = \left|x\right\rangle \left|f(x) \oplus y\right\rangle$$

where $\oplus$ denotes the bitwise XOR.

Before describing the classical post-processing, let us try to determine some properties of the measurement outcome in the above circuit. The circuit begins in state $\left|0^n\right\rangle \left|0^n\right\rangle$, and Hadamard

transforms are performed on the first n qubits. This produces the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0^n\rangle .$$

The state after the B_f transformation is performed is

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle .$$

This time there is no phase kick-back as there was in the previous algorithms we studied. Finally, n Hadamard transforms are applied, which results in state

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle |f(x)\rangle .$$

Here, we have used the formula

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$$

to obtain this expression.

Now, we are interested in the probability with which each string results from the measurement. Let us first consider the special case where $s = 0^n$, which means that f is a one-to-one function. We can write the state from above as

$$\sum_{y \in \{0,1\}^n} |y\rangle \left(\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |f(x)\rangle \right),$$

so the probability that the measurement results in each string y is

$$\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |f(x)\rangle \right\|^2 = \frac{1}{2^n}.$$

One way to see that the previous equation is true is to note that

$$\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |f(x)\rangle \right\|^2 = \left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |x\rangle \right\|^2$$

because the two vectors only differ in the ordering of their entries (as f is one-to-one). The value of the right-hand-side is more easily seen to be 2^{-n}. Thus, the outcome is simply a uniformly distributed n bit string when $s = 0^n$.

The second, more interesting, case is where $s \neq 0^n$. The analysis from before still works to imply that the probability to measure any given string y is

$$\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |f(x)\rangle \right\|^2 .$$

Let $A = \mathrm{range}(f)$. If $z \in A$, there must exist 2 distinct strings $x_z, x_z' \in \{0,1\}^n$ such that

$$f(x_z) = f(x_z') = z,$$

and moreover it is necessary that $x_z \oplus x_z' = s$ (which is equivalent to $x_z' = x_z \oplus s$). Now,

$$\begin{aligned}
\left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} \, |f(x)\rangle \right\|^2 &= \left\| \frac{1}{2^n} \sum_{z \in A} \left((-1)^{x_z \cdot y} + (-1)^{x_z' \cdot y} \right) |z\rangle \right\|^2 \\
&= \left\| \frac{1}{2^n} \sum_{z \in A} \left((-1)^{x_z \cdot y} + (-1)^{(x_z \oplus s) \cdot y} \right) |z\rangle \right\|^2 \\
&= \left\| \frac{1}{2^n} \sum_{z \in A} (-1)^{x_z \cdot y} \left(1 + (-1)^{s \cdot y} \right) |z\rangle \right\|^2 \\
&= \begin{cases} 2^{-(n-1)} & \text{if } s \cdot y = 0 \\ \\ 0 & \text{if } s \cdot y = 1. \end{cases}
\end{aligned}$$

In this calculation we have used the fact that

$$(x_z \oplus s) \cdot y = (x_z \cdot y) \oplus (s \cdot y)$$

which you can verify for yourself.

So, the measurement always results in some string y that satisfies $s \cdot y = 0$, and the distribution is uniform over all of the strings that satisfy this constraint. Is this enough information to determine s? The answer is yes, as we will discuss next.

Classical post-processing

When we run the circuit above, there are two cases: in the special case where $s = 0^n$, the measurement results in each string $y \in \{0,1\}^n$ with probability

$$p_y = \frac{1}{2^n};$$

otherwise, in the case that $s \neq 0^n$ the probability to obtain each string y is

$$p_y = \begin{cases} 2^{-(n-1)} & \text{if } s \cdot y = 0 \\ \\ 0 & \text{if } s \cdot y = 1. \end{cases}$$

Thus, in both cases the measurement results in some string y that satisfies $s \cdot y = 0$, and the distribution is uniform over all of the strings that satisfy this constraint.

Is this enough information to determine s? The answer is yes, provided that you repeat the process several times and accept a small probability of failure. Specifically, if the above process is run $n-1$ times, you will get $n-1$ strings $y_1, \ldots, y_{n-1}$ such that

$$\begin{aligned} y_1 \cdot s &= 0 \\ y_2 \cdot s &= 0 \\ &\vdots \\ y_{n-1} \cdot s &= 0. \end{aligned}$$

This is a system if $n-1$ linear equations in n unknowns (the bits of s), and the goal is to solve to obtain s. All of the operations are modulo 2 operations, so it is not exactly the type of system of linear equations you were used to in linear algebra, but the system can be efficiently solved using similar methods. You will only get a unique non-zero solution s if you are lucky and $y_1, \ldots, y_{n-1}$ are linearly independent. The probability that $y_1, \ldots, y_{n-1}$ are linearly independent is at least

$$\prod_{k=1}^{\infty} \left(1 - \frac{1}{2^k}\right) = 0.288788\cdots > \frac{1}{4}.$$

If you have linear independence, solve the system to get a candidate solution $s' \neq 0^n$, and test that $f(0^n) = f(s')$. If $f(0^n) = f(s')$, you know that $s = s'$ and the problem has been solved. If $f(0^n) \neq f(s')$, it must be the case that $s = 0^n$ (because if this were not so, the unique nonzero solution to the linear equations would have been s). Either way, once you have linear independence, you can solve the problem.

Now, repeat the entire process $4m$ times. The probability of not finding a linearly independent set during one of the iterations is less than

$$\left(1 - \frac{1}{4}\right)^{4m} < e^{-m}.$$

For example, if $m = 10$ this probability is less than $1/20000$.

The upshot is that for any constant $\varepsilon > 0$, Simon's algorithm can solve the problem we are considering with error probability at most ε using $O(n)$ queries to the black-box.