Quantum Algorithms

Recall that given a classical circuit for computing a function f, we can design an equivalent reversible circuit, and therefore also a quantum circuit U_f, that outputs $xcdot f(x)$ on input x. In general, though, we don't need to feed U_f a classical state $|x\rangle$. If we feed U_f a superposition

$$\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle |0\rangle$$

then, by linearity,

$$U_f\left(\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle |0\rangle\right) = \sum_{x\in\{0,1\}^n} \alpha_x U_f(|x\rangle |0\rangle) = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle |f(x)\rangle$$

and we've computed $f(x)$ simultaneously for each basis state $|x\rangle$ in the superposition. Note that if we had not been able to eliminate the extra output junk(x) from the output of our circuit, we would have ended up with the quantum state $\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle |f(x)\rangle |\text{junk}(x)\rangle$ instead, which is a very different.

The procedure for converting classical circuits into quantum circuits U_f is a useful primitive which we will use extensively in quantum algorithms. A second primitive is the Hadamard transform H_{2^n}. Recall that H is the unitary transformation on one qubit defined by the matrix

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

In other words, H maps $|0\rangle$ onto $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, and $|1\rangle$ onto $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Given an n-qubit system, we'll often want to apply H separately to each of the qubits in turn. We call the resulting transformation $H_{2^n} = H^{\otimes n}$; it equals H tensored with itself n times. For all $n > 0$, $H^{\otimes n}$ can also be defined recursively by the matrix

$$\begin{pmatrix} \frac{1}{\sqrt{2}}H^{\otimes n-1} & \frac{1}{\sqrt{2}}H^{\otimes n-1} \\ \frac{1}{\sqrt{2}}H^{\otimes n-1} & -\frac{1}{\sqrt{2}}H^{\otimes n-1} \end{pmatrix}.$$

where $H^{\otimes 0} = \begin{pmatrix} 1 \end{pmatrix}$. (Incidentally, it is an exercise to show that for all $n \times n$ matrices U and V, if U and V are unitary then the tensor product $U \otimes V$ is also unitary.)

1 The Deutsch-Jozsa Algorithm

We start our study of quantum algorithms with the Deutsch-Jozsa algorithm.

Suppose we're given a Boolean function $f : \{0,1\}^n \to \{0,1\}$. We're promised that f is either identically zero (meaning that $f(x) = 0$ for all inputs $x \in \{0,1\}^n$), or else balanced (meaning that $f(x) = 0$ for precisely half of all inputs x, and $f(x) = 1$ for the other half). The challenge is to decide which is the case. To

do this, we can make queries of the form "Is $f(x)$ equal to 1?" for particular values of x. Our goal is to minimize the number of queries that have to be made.

The algorithm uses two quantum registers, the first having n qubits and the second having only one qubit. We initialize the system to the basis state $|0\cdots0\rangle|0\rangle$. Then we apply the Hadamard transform H_{2^n} to the first register (or equivalently, the Hadamard transform H to each qubit of the first register separately). This results in the superposition

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

We then compute $f(x)$ and store the result in the second register. How is this done? If f is a black-box oracle, then (by assumption) we don't need to worry about how it's done. If, on the other hand, f is given explicitly (say, as a Boolean circuit), then we've seen that we can compute f reversibly using Fredkin gates. In either case, the resultant superposition is

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

Next we apply the one-qubit transformation

$$\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$$

to the second register. This transformation, which is easily seen to be unitary, is a "phase flip": if $f(x)=0$ then it leaves the amplitude of $|x\rangle|f(x)\rangle$ alone, whereas if $f(x)=1$ then it inverts the amplitude. So we get

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

At this point we want to erase $f(x)$ from the second register, to allow for proper interference among the states. (What goes wrong if we don't erase $f(x)$ is left as an exercise.) We can't do this directly, since erasure is not a unitary operation. But since we have an oracle for f, we can simply compute f a second time and CNOT the result into the second register, so that the bit in that register is $f(x)\oplus f(x)=0$. Doing so, we obtain

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

The final step is to perform another Hadamard transform H_{2^n} on the first register. We could work out the result of this algebraically, but would rather reason about it to obtain more insight. We've seen that the Hadamard transform is its own inverse, and that

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

From these it follows that if f is identically zero, then applying H_{2^n} brings us back to $|0\cdots 0\rangle |0\rangle$. If, on the other hand, f is balanced, then the state

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

is a linear combination of $H_{2^n}(|y\rangle)$ for various values of y. However, none of these values can be $y = 0$, since the state is orthogonal to $H_{2^n}(|0\rangle)$ in the vector space Z_2^n (the two having inner product $2^{n-1} - 2^{n-1} = 0$).

Therefore, if f is zero, then $x = 0$ at the end of the computation, whereas if f is balanced, then $x \neq 0$. So by observing the $|x\rangle$ register, we can decide with certainty whether f is zero or balanced. We've done this using two queries to f and $\Theta(n)$ steps of computation.

Clearly, a deterministic classical algorithm requires $2^{n-1} + 1$ queries in the worst case: if 2^{n-1} bits have been queried and all turned out to be 0, we still need to query one more bit to decide whether the function is zero or balanced.

On the other hand, the problem admits an efficient randomized algorithm, as follows. Choose a value of x uniformly at random; if $f(x) = 1$ then conclude that f is balanced, otherwise repeat. If, after k iterations (for some constant k), we still haven't found an x such that $f(x) = 1$, then we halt and conclude that f is zero. This algorithm uses only $O(1)$ queries. However, has a nonzero probability of error, equal to 2^{-k}, or slightly less if we choose x values without replacement. This is because, even if f is balanced, the randomized algorithm has a nonzero probability of never seeing an x such that $f(x) = 1$.

What Deutsch and Jozsa showed is that a quantum computer can decide whether f is balanced, with certainty, using only two queries to f. What follows is an algorithm to accomplish this.