A Primer in Quantum Computing
Lesson 5: Quantum Algorithms, Deutsch's

Agustin Valdes Martinez

November 16, 2024

1 Recap

Last class, we introduced operations on qubits called *quantum gates*, which we learned to represent as just matrix-vector multiplication. We learned about the *Hadamard* gate, represented in matrix form as

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1}$$

It's an important operator because it takes states $|0\rangle$, $|1\rangle$, which are states of total certainty, into states of maximum uncertainty:

$$\hat{H}|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2}$$

$$\hat{H}|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{3}$$

The $|+\rangle$, $|-\rangle$ states are the "50-50" qubits we met in the first class, and the two form the *Hadamard basis*. What this means is that I should be able to write *any other* qubit as a superposition of $|+\rangle$ and $|-\rangle$, just as we've been doing with $|0\rangle$ and $|1\rangle$ (the *computational* basis).

I'll also list the Pauli matrices below for your convenience... do you remember how they act on $|0\rangle$, $|1\rangle$?

$$\hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \hat{Y} = \begin{pmatrix} 0 & -i \\ i & i \end{pmatrix}, \ \hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{4}$$

1.0.1 Quick Practice 1

Compute the resulting qubits

- $\hat{H}\hat{X}\hat{H}|+\rangle$
- $\hat{X}\hat{Y}\hat{Z}|0\rangle$

in two ways:

1. First multiplying all of the operators together, then acting on the qubit
2. Acting on the qubit successively (one operator at a time)

For each case, you should find that either method results in the right answer! Like we mentioned last time, matrix-matrix multiplication obeys associativity,

$$\hat{A}(\hat{B}\hat{C}) = (\hat{A}\hat{B})\hat{C} \tag{5}$$

but not necessary commutativity:

$$\hat{A}\hat{B} \neq \hat{B}\hat{A} \tag{6}$$

If it so happens that $\hat{A}\hat{B} = \hat{B}\hat{A}$, we say that $\hat{A}$ and $\hat{B}$ *commute*.

2 Our First Quantum Algorithm

So we have all of these quantum logic gates... what can we do with them? You might have this vague notion that if we string them together in a special sequence, we might be able to do something interesting to our qubit. This is exactly the idea behind quantum algorithms.

In case you don't know, an algorithm is a set of instructions that leads to some result. We will now discuss one of the earliest quantum algorithms developed which showed some advantage over classical algorithms: **Deutsch's algorithm**.

2.1 The Problem

Every algorithm intends to solve some problem, which we need to fully understand before seeing how, in this case, Deutsch's algorithm solves it. Suppose I hand you a function $f(x)$, **which takes in as input either 0 or 1**, and I tell you that it's either *constant* or *balanced*.

- Constant means all inputs map to the same output: $f(0) = f(1)$. This means either $f(0) = f(1) = 0$ OR $f(0) = f(1) = 1$.
- Balanced means one input maps to 0, and the other maps to 1. So $f(0) \neq f(1)$ and either $f(0) = 0, f(1) = 1$ OR $f(0) = 1, f(1) = 0$.

The problem is to figure out whether f(x) is balanced or constant by evaluating f(x) as few times as possible!

2.2 Oracles and Performance of Algorithms

To say one algorithm is better than another, we need a metric to compare them. A common choice for this problem is **the number of queries to an oracle**. What does this mean?

Imagine we have a black box that, when we query it, we get a result back. This is how we will evaluate $f(x)$: we assume there is a black box 'oracle' that, if I give it x, it returns $f(x)$.

Figure 1: Black box operator that takes in state x and acts on it, returning $f(x)$; we don't quite know what $f(x)$ is, but we can learn some of its properties.

We'll see that in the classical solution to this problem, it takes **two** queries to the oracle to know whether $f(x)$ is balanced or not. In the quantum case, it'll take only **one**! We will use the following oracle:

$$O_f|x\rangle = (-1)^{f(x)}|x\rangle \tag{7}$$

It might seem weird to not write it out as a matrix, as we have with all of our other operations/gates, but usually, oracles are written out this way to show how they act on some state $|x\rangle$. We'll get practice using this oracle in a bit.

2.3 Classic Solution

How might I solve this with a classical computer? Well, I will need to evaluate **both** $f(0)$ and $f(1)$ to determine whether $f(x)$ is constant or balanced. That's two queries to the oracle!

2.4 Quantum Solution - Deutsch's Algorithm

Again, a quantum algorithm is just a sequence of quantum logic gates we need to apply to an initial state to reach our solution: in this case, figure out whether $f(x)$ is balanced or constant. You'll see many variations of this algorithm online, but I think the simplest (and most elegant) one is as follows:

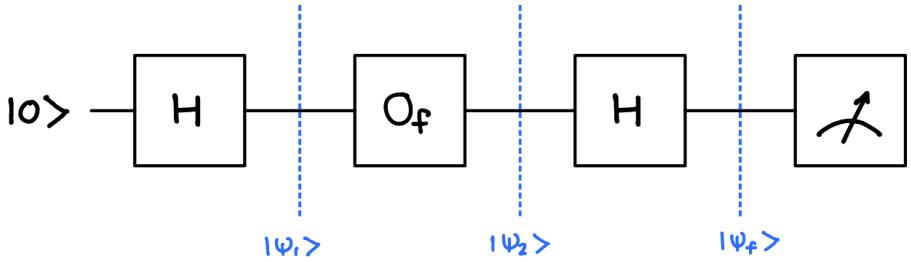

Figure 2: Deutsch's algorithm just involves 2 Hadamard gates, 1 query to the oracle, and a measurement at the end. We denote intermediate states in blue (like milestones in the algorithm) to make the calculation easier to follow.

We read quantum algorithms from left to right. So we start with our initial state $|0\rangle$, apply $\hat{H}$, query the oracle (Equation 15), apply $\hat{H}$ again, and perform a measurement. This measurement should encode, in some way, the answer to our question: is $f(x)$ balanced or constant? Let's solve for the final state milestone by milestone ($|\psi_1\rangle \to |\psi_2\rangle \to |\psi_f\rangle$):

$$|\psi_1\rangle = \hat{H}|0\rangle \tag{8}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{9}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{10}$$

$$|\psi_2\rangle = O_f|\psi_1\rangle \tag{11}$$

$$= \frac{1}{\sqrt{2}} O_f \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{12}$$

$$= \frac{1}{\sqrt{2}} O_f(|0\rangle + |1\rangle) \tag{13}$$

$$= \frac{1}{\sqrt{2}} (O_f|0\rangle + O_f|1\rangle) \tag{14}$$

$$= \frac{1}{\sqrt{2}} \left[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right] \tag{15}$$

Note that we apply the definition of the oracle (Equation 7) to go from lines 14 to 15.

$$|\psi_f\rangle = \hat{H}|\psi_2\rangle \tag{16}$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\left[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right] \tag{17}$$

$$= \frac{1}{2}\left[\left((-1)^{f(0)} + (-1)^{f(1)}\right)|0\rangle + \left((-1)^{f(0)} - (-1)^{f(1)}|1\rangle\right)\right] \tag{18}$$

Now we just have to measure our resulting qubit. In particular, we're after the probability of getting outcome $|0\rangle$; this is just the absolute-value-squared of the value in front of the $|0\rangle$ in Equation 26!

$$P_0 = \frac{1}{4}\left[(-1)^{f(0)} + (-1)^{f(1)}\right]^2 \tag{19}$$

Here's the punchline: If $P_0 = 0$, then $f(x)$ is ***balanced***. If $P_0 = 1$, then $f(x)$ is ***constant***. This might not be very obvious at first glance, so let's convince ourselves that this is true.

Suppose $f(x)$ is balanced; then $f(0) = 1, f(1) = 0$ OR $f(0) = 0, f(1) = 1$. Either way, we get

$$P_0 = \frac{1}{4}\left[(-1)^0 + (-1)^1\right]^2 = 0 \tag{20}$$

What this means is that if $f(x)$ is balanced, we will **never** measure the qubit in the $|0\rangle$ state. It also implies that **if we do** measure the qubit in the $|0\rangle$, then $f(x)$ is constant. Just like that, we've finished our first quantum algorithm... congrats!