Lecture 2 Notes: Quantum Gates

Quantum Computing 101

1 Bra-ket notation

It's nice to have consistent notation to refer to the state of our qubits. Remember that the state of n qubits can be thought of as a column vector of length 2^n, containing complex amplitudes. For example, the zero qubit is the vector $[1, 0]^T$, the 2-qubit state where the first qubit is 0 and the second is 1 can be represented by the vector $[0, 1, 0, 0]^T$, and so on. This works fine, but it can get cumbersome to carry around these vectors everywhere we go. Is there a simpler way to represent these?

Physicists like to use **bra-ket notation**. It's not important right now what a **bra** is, but a **ket** is another way to represent column vectors of amplitudes. For now, just go with it! The ket for the zero qubit $[1, 0]^T$ is $|0\rangle$, the key for the one qubit $[0, 1]^T$ is $|1\rangle$, the ket where the first qubit is 1 and the second is 1 is $|01\rangle$. Pretty natural, right?

Think about how bigger kets like $|1010\rangle$ would look like as a vector, and always remember that **kets are just a way of representing the column vectors of amplitudes**: there's nothing funny going on. You can also just think about kets as variables, so the ket $|\Psi\rangle$ could represent some arbitrary quantum state.

Just like you can add vectors and multiply vectors by scalars, you can do the same thing for kets. As an example, you can represent any one-qubit state (which, remember, is a length two vector), as

$$\begin{aligned} |\Psi\rangle &= \alpha\,|0\rangle + \beta\,|1\rangle \\ &= \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \end{aligned}$$

where $|\alpha|^2 + |\beta|^2 = 1$, as usual.

Similarly, any two-qubit state could be thought of as $\alpha_1\,|00\rangle + \alpha_2\,|01\rangle + \alpha_3\,|10\rangle + \alpha_4\,|11\rangle$.

2 Quantum circuits

Classical computers fundamentally rely on basic logical gates that act on sets of bits, like **AND**, **OR** and **NOT**, and some combination of these, put into a circuit, is enough to do any classical computation.

Quantum computers also rely on the idea of circuits, with gates acting on sets of qubits, except the gates aren't quite as simple as above: any unitary operator could be used to change the state of the qubits. In general, a quantum algorithm consists of two steps:

1. Prepare an input state of qubits (this is often the all-zeros state)

2. Do some sequence of operations on this qubit state, which could either be a unitary operation or a measurement of the state

The output of the algorithm is the measurements you've taken of the quantum states.

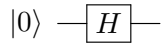
2.1 What gates do we use?

Any unitary transform can be used to evolve a quantum state, but that's not very useful for us. (There is a concept of a *universal set of quantum gates*, but that will come later). For now, let's talk about a few important gates.

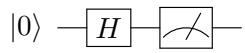
One of the most useful gates the the *Hadamard transform*,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Notice that when H is applied to $|0\rangle$, we get the state $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$. In terms of quantum circuits, this looks like

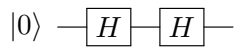


If we were to measure this qubit state, there would be a 50% chance of observing 0, and a 50% chance of observing 1. In quantum circuits, the act of observing is denoted with a weird eye-looking thing.



The output of this quantum circuit would be, essentially, a coin flip. The Hadamard transform "randomized" our bit.

What if we applied the Hadamard transform again to the state $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$? This would look like

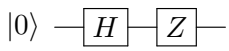


If you do the math, you'll notice that $H^2 = I$ (the identity matrix), so applying H twice actually just returns the original bit. Applying it once randomized it, applying it twice unrandomized it. What's going on?

Let's introduce one more gate, the **Pauli Z gate**. There are 4 Pauli gates, X, Y, Z and I, that all operate on a single qubit, but right now we just need Z.

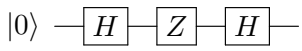
$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

What would happen if we took the $|0\rangle$, applied Hadamard, and then applied the Z operator?



This gives us the quantum state $[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]$, which is subtly different from what we got from a single Hadamard operation (note the negative sign). Still, if we were to observe the outcome of this circuit, it would be a random bit, just like a single Hadamard operation. So it seems like adding the Z operator didn't really do anything, right?

What if, instead, we did a Hadamard operation, then Z, then another Hadamard? If Z doesn't do anything, then you'd expect this to give you the same thing as before: whatever you gave it as input. Let's try it...



Doing the math, it turns out that this circuit **always** gives us $|1\rangle$ back. Not $|0\rangle$. So Z isn't useless at all. What does this mean? To me, it kind of shows that for these complex amplitudes, it's not just their magnitudes that matter: the actual values are important, and can lead to many different outcomes that you wouldn't expect just by looking at the magnitudes.

3 Hold up, why can't we just do this classically?

You might be looking at all of these matrix operations and scratching your head: why can't I just do this in, say, Python? I can do linear algebra there, and this is all just matrix-vector multiplications (and once in a while, some random sampling). And you're right – **everything here can, in theory, be done on a non-quantum computer**.

The key is that, on a classical computer, all of these operations take time (and space) that's *exponential in the number of qubits*. Hell, just storing an n-qubit state requires 2^n complex amplitudes. However, at least in theory, a quantum computer could store all of those amplitudes in just n qubits, and it could implement all of those unitary operators (which, for us, are exponentially expensive), *in linear time*. Quantum computers aren't any more powerful (in terms of computability) than your laptop: they just can do things a lot faster.