Curso básico:

In about 30 minutes you’re going to know enough about qubits, superposition and entanglement to write your very first quantum program, and most importantly, have a reasonable idea of what it means. Sounds smashing.

Contents of this blog

* Back to Basics
* Measuring a Qubit
* Quantum Gates
* Important Gates
* Multiple Qubits
* Another Important Gate
* Bell State
* Writing a Quantum Program
* What’s next?
* Appendix
* Further resources

Back to Basics

It’s easy to forget that the programs we write are essentially just manipulating a bunch of 0s and 1s stored in our ‘classical’ computer. These are discrete, binary states.

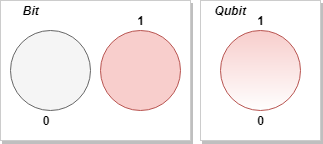
Quantum computers however operate on *continuous*states – that’s part of what makes them so powerful.

So instead of a classical bit being ‘on’ or ‘off’ like a light switch, in a quantum computer we have qubits, which are more like a dimmer switch, being any possible combination of ‘on’ and ‘off’ in between.

Just as bits are the fundamental object of information in classical computing, qubits (quantum bits) are the fundamental object of information in quantum computing. To understand this correspondence, let's look at the simplest example: a single qubit.

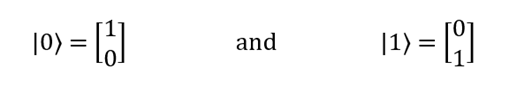
Representing a Qubit

While a bit, or binary digit, can have value either 00 or 11, a qubit can have a value that is either of these or a quantum superposition of 00 and 11.



The state of a single qubit can be described by a two-dimensional column vector of unit norm, that is, the magnitude squared of its entries must sum to 11. This vector, called the quantum state vector, holds all the information needed to describe the one-qubit quantum system just as a single bit holds all of the information needed to describe the state of a binary variable.

We can notate the two qubit states using *Dirac notation* which equates to these vectors:

[](https://msdnshared.blob.core.windows.net/media/2018/02/image93.png)

If we add the two states together, we can express any possible combination of |0〉 and |1〉, and in quantum mechanics this is called a superposition. The notation for that looks like this:

|*ψ*⟩ = *α*|0〉 + *β*|1〉

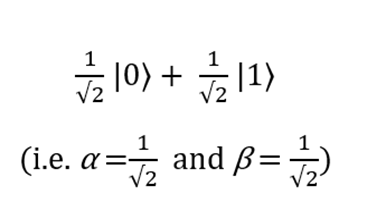
Here *α* and *β* are kind of like probabilities with the minor difference that they are complex numbers. It doesn’t matter if you think of them as real numbers; however, if you do, then remember that they'll sometimes be negative, and that the sum of their squares is always 1.

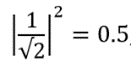
Measuring a Qubit

Now quantum states are weird. If you “look” at a qubit, it immediately collapses its state… a bit like looking at your bed after a long day. What I mean by that is, if we have a qubit that is in a superposition and we measure it, it’s either going to collapse to the |0〉 state or the |1〉 state (we can’t measure both at the same time!). After measurement, we effectively lose the prior values of *α* and *β*.

Because of that, when we measure a qubit, we talk about the result we get in terms of probabilities. In general, a qubit when measured will give ‘|0〉’ with probability |α|², and ‘|1⟩’ with probability |β|².

Let’s have a look at an example - say we had the following qubit:

[](https://msdnshared.blob.core.windows.net/media/2018/02/image94.png)

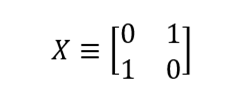
[](https://msdnshared.blob.core.windows.net/media/2018/02/image95.png)If we measure the qubit we are likely to get the outcome ‘0’ fifty percent of the time, because

 , and its post-measurement state will be |0〉 (i.e. *α* =1 and *β*=0). Similarly, we are likely to get the outcome ‘1’ fifty percent of the time, for the same reason, and its post-measurement state will be |1〉 (i.e. *α* =0 and *β*=1).

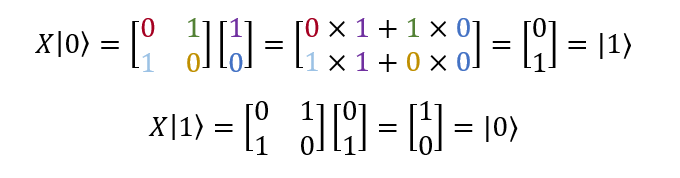
This can seem confusing at first. The main point here is that these probabilistic quantum states can be used for computation and in some cases, this is much more efficient than classical systems due to their ‘quantum weirdness’. Next, we’re going to look at how we can manipulate these qubits for computation just like you can with classical bits.

Quantum Gates

We’re back in familiar territory. In classical computing we use logic gates to operate on bits, and likewise we can use quantum gates to operate on qubits. Think of the NOT gate, which takes 0 → 1 and 1 → 0. A quantum NOT gate bears some resemblance to its classical brother, as it takes |0〉 → |1〉 and |1〉 → |0〉. So, a qubit in the state *α*|0〉 + *β*|1〉 becomes *α*|1〉 + *β*|0〉 after such a gate has operated on it. The not gate can be written as a matrix, *X*, which swaps the roles of 0 and 1 in the state



As we can see, X|0〉 = |1〉 and X|1〉 = |0〉:



Because |0〉 and |1〉 are defined in vector form as  and , you can think of the first column of X as the transformation applied to |0〉 and the second column as the transformation applied to |1〉.

Now this doesn’t seem all that different to what we’re used to. But I want to remind you of what we found in the previous section, that the measurement of a qubit is probabilistic. As we know from basic statistics, all probabilities sum to one. As a result, we have the condition that |α|²+|β|² =1 for a quantum state *α*|0〉 + *β*|1〉.

A consequence of this is that there are some constraints on what gates we can have in the quantum world. And one of them is that this normalization condition on the quantum states, |α|²+|β|² =1, should hold both before and after the gate has acted. In terms of matrices, this condition will hold if a matrix is *unitary*.

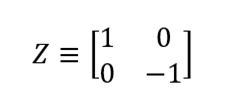
I’m about to tell you what unitary means in maths speak, if we read it *really quickly,* we can get to the next sentence. A gate is unitary if, where U†U=1, where U is obtained by transposing and then complex conjugating U, and 1 is the two by two identity matrix. In plain English, it means that the transformation doesn’t change the length of the vector. Keeping the length constant is the same as ensuring that all the probabilities add up to 100%. Having probabilities add up to 200% or 25% would make no sense and using unitary matrices guarantees at least this type of craziness is off the menu in the quantum world.

But we can breathe a sigh of relief – this is our only constraint. Some classical gates don’t have quantum equivalents for this reason, but there are some new ones that crop up too, and we’re going to look at some important ones next.

Important Gates

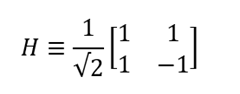
Z and Hadamard

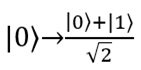
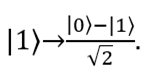
The following gates you’ll see in our first quantum program. The *Z*gate is nice and simple, it leaves |0〉 unaltered and makes |1〉 negative. That can be written as a matrix like this,



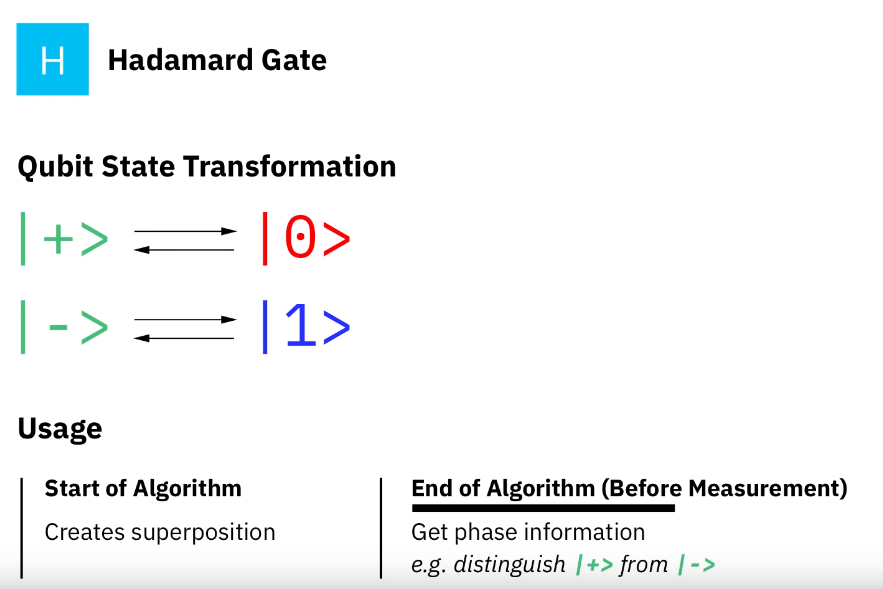
which transforms the qubit states from |0〉→|0〉 and from |1〉→ -|1〉. (Remember, the first column corresponds to the transformation applied to |0〉, and the second column to |1〉.)

Then we also have the *Hadamard*gate which creates a superposition between the |0〉 and |1〉 states, similar to those seen previously. And that can be written as a matrix like this,



which transforms the qubit states from  and 

If you’re interested in learning more about unitary matrices, and how to visualize these gates have a look at the Further Resources section where I’ve listed some useful material.



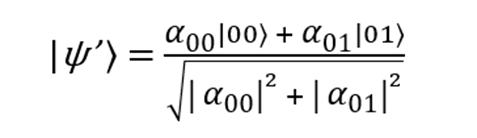
Multiple Qubits

Time for some more familiar stuff. In the classical world we can have multiple bits, such as 00, 01, 10 and 11. In quantum computing we can similarly have |00〉, |01〉, |10〉 and |11〉. We can describe the two qubits using a vector,



Just like before, the measurement results of ‘00’ for example, occurs with probability , ‘01’ with probability , and so on.

Now say we wanted to measure the first qubit (instead of both), the measurement results of 0 occurs with probability  . And remember measurement changes the state, so afterwards the vector would now have a value of

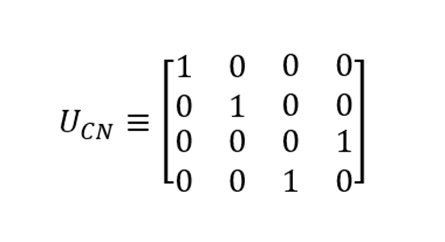


You can see on the top line we’ve got rid of all the terms whose first bit was 1, since they weren’t consistent with the measurement being 0. The square root in the denominator renormalizes the vector so that the sum of the squared amplitudes is still 1, which we require for it to be a valid quantum state.

Another Important Gate

You’ve seen the NOT gate, next up we’ve got the CNOT gate, which stands for *controlled*-NOT. It takes two input qubits, the first is the *control* qubit, and the second is the *target* qubit. As you might have guessed, if the control is |0〉, then the target qubit is unchanged. If the control is |1〉, then a NOT is performed on the target qubit.

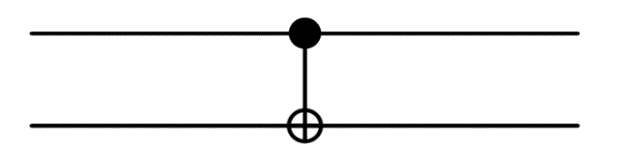
There are a few ways to think of CNOT. Like *X*, *Z* and *H,*we can show it in matrix form, *U*.



Looking at the columns of the matrix we can see that they correspond to the following transformations, |00〉→|00〉, |01〉→|01〉, |10〉→|11〉 and |11〉→|10〉. As with the other matrices we’ ve looked at ,

Is unitary, meaning .

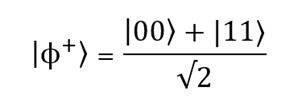
It can also be drawn like this, where the control bit is on top and the target bit is at the bottom:

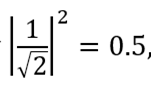


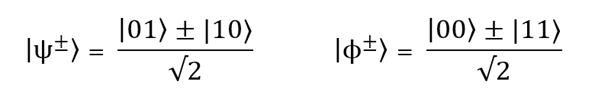
Looks like something you might find in the Tate Modern.

Bell States

These get a whole section of their own. The Bell states (there are four of them). You’ll see one of them (|ϕ+⟩) appear in the quantum program a little later, so here it is:

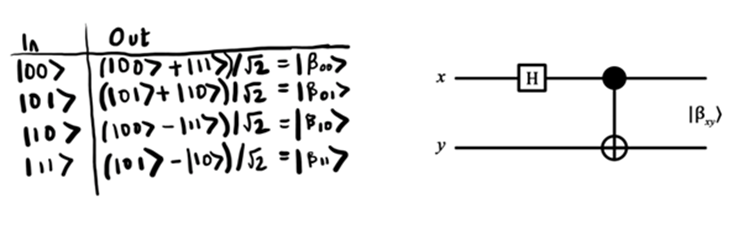


Suppose we were to measure the first qubit, we’re going to get |0〉 with probability  leaving the post-measurement state |*ψ’*⟩=|00〉, or |1〉 with the same probability, 0.5, and a post-measurement state |*ψ’*⟩=|11〉. In case you were interested, here’s the complete set of four Bell States (which represent the simplest examples of quantum entanglement):



Suppose instead that we had measured the *second* qubit - using the same logic it would leave the post-measurement state |00〉 or |11〉. If we then decided to measure the *first* qubit, the probability is no longer a half, we’re going to get |0〉 with probability 1 or 0 depending on the post-measurement state. The key point here is that these outcomes are correlated. This was first noticed by Einstein, Podolsky and Rosen (which is why you might see them called EPR pairs), with further progress made by John Bell.

One final thing, we can generate Bell states using a combination of a Hadamard gate followed by a CNOT. I think this is pretty cool. The Hadamard transforms the first qubit by putting it into a superposition, this is then used by the CNOT as a control to modify the target qubit. That is neatly described by the following circuit diagram:



If you want to see the working behind each of those transformations, take a look at the Appendix. For now, with our knowledge of qubit states, quantum gates, and Bell states, we’re ready to tackle our first quantum program.