Extracted from:

Quantum Computing

Program Next-Gen Computers for Hard, Real-World Applications

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Quantum Computing

Program Next-Gen Computers for Hard, Real-World Applications

Nihal Mehta, Ph.D.
edited by Brian MacDonald

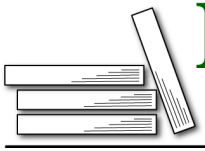
Quantum Computing

Program Next-Gen Computers for Hard, Real-World Applications

Nihal Mehta, Ph.D.

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Development Editor: Brian MacDonald
Copy Editor: L. Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

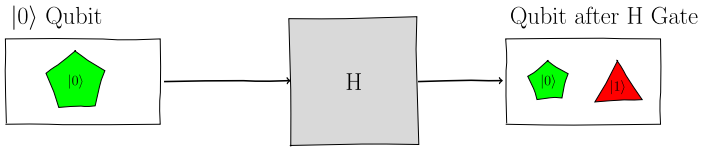
For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-720-1
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—August 2020

Putting Qubits in Blended States

The Hadamard (H) gate puts qubits in blended states of |0⟩ and |1⟩. It's the signature gate of quantum computing and forms the bedrock of quantum programs. For me, it was one of the two quantum gates that brought home the terrific potential of quantum computing. (The other is the Pauli-Z gate, which we'll cover in Chapter 5, Beam Me Up, Scotty—Quantum Tagging and Entangling, on page ?.)

The H gate is fundamentally a *qubelet splitter*, which is the basic mechanism to create blended qubits. For example, it takes a |0⟩ qubit, which is essentially a pentagon |0⟩ qubelet, and splits it into another pentagon |0⟩ and triangle |1⟩ qubelet, as shown in the following figure:



If you measure the blended qubit on the right, it'll collapse to |0⟩ roughly 50% of the time and will flop to |1⟩ the other times. You'll never actually see the qubit in a blended state.

The H gate is often the first quantum gate you declare in quantum programs. It puts qubits in superposition so that other quantum gates can act on them to modify their quantum states.

Joe asks:

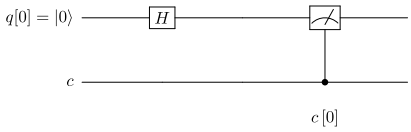
Why Is It Called the Hadamard Gate?

The oddly named gate is a tribute to the French mathematician Jacques Hadamard, who along with the German mathematician Issai Schur, is credited with formulating the tensor product of matrices,[a] a specialized matrix multiplication technique that's central to analyzing the superposition states in quantum circuits. We'll get to see how this works in Chapter 8, Giant Leap for Mankind—Multi-Qubit Programs, on page ?.

a. https://en.wikipedia.org/wiki/Hadamard_product_(matrices)

H Gate on |0⟩ Qubit

To see this gate in action, let's build the following quantum circuit:



This circuit has only a single quantum register, q[0] initialized with a |0⟩ qubit, and a single-bit classical register, c[0]. The Measure gate collapses the quantum state after the H gate acts on the qubit, and it records the corresponding binary value that the qubit settles down to in the classical register c[0].

On the IBM Quantum Computer, you can simply drag the H gate from the palette and place it on the wire representing q[0]. Then, drag the Measure gate and record the value of the collapsed qubit in c[0].

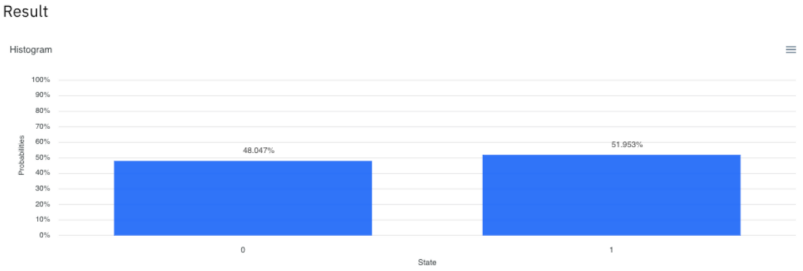
The code listing, excluding the header, for this circuit is as follows:

H_Gate.qasm

```
qreg q[1];
creg c[1];

// Put q[0] in superposition
h q[0];
measure q[0] -> c[0];
```

Declare the H gate, as shown on the highlighted line: the type of gate, in this case, h, followed by the qubit, q[0], that it acts on.

When we execute this program, the q[0] qubit is put into a superposition by the H gate. That is, it's now a bundle of pentagon |0⟩ and triangle |1⟩ qubelets. Thus, when this qubit is measured, it'll randomly collapse to the |0⟩ or |1⟩ idealized states, which correspond to the 0 or 1 binary states, with equal probability, as shown in the following output:



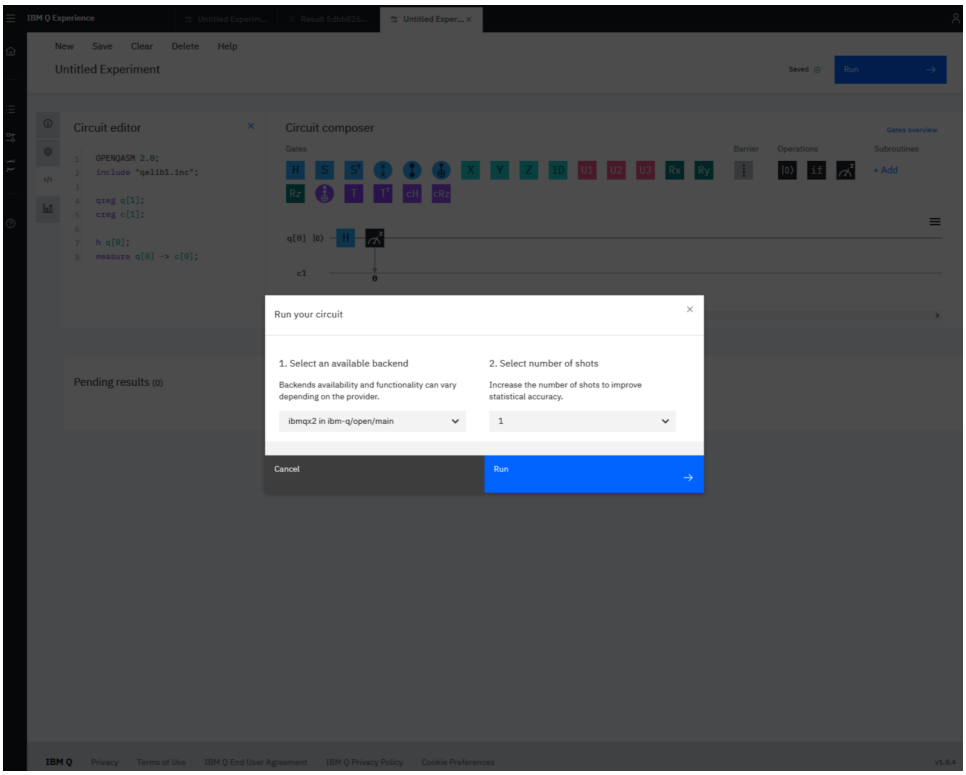
The output shows that the qubit, after its been operated on by the H gate, collapses roughly half the time to |0⟩ and about half the time to |1⟩, which are, in turn, recorded as 0 and 1 in the classical registers.

But what does the assertion "qubit collapses to $|1\rangle$ half the time" really mean? To see what's behind this claim, let's rerun this program a little differently than a standard execution.

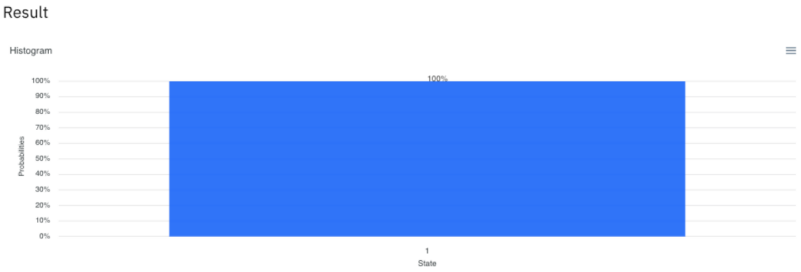
When a quantum program is typically invoked, it's rarely just run once. More likely, the program is run repeatedly. For each run, or shot, the following steps are executed:

- The qubits in the quantum register are operated upon by the quantum gates as specified in the quantum program.
- The `Measure` gates inspect the qubits and record the corresponding binary states in the classical register.
- The system keeps a running tally of the `1` and `0` values observed in the classical register after each shot. At the end of the specified number of runs, the system shows the number of times, as a percentage, a `1` or `0` was observed in each classical register element.

Let's make this a little less abstract by investigating what happens when we force our program to execute just once. To specify the number of times, or shots, you want to execute your program, click the Run button and on the dialog window that opens, specify the number of shots in the second drop-down list as shown on page 8.



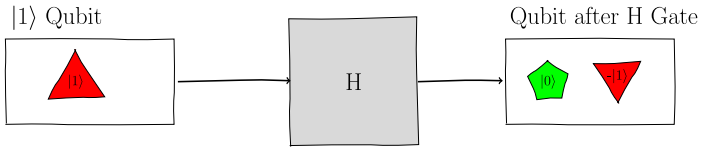
Then run your code. The solitary qubit in this circuit can only collapse once in a single run. So, you'll only find it in *either* a `0` or `1` binary state in the classical register at termination. Put another way, one of these two possibilities will occur with 100% probability. In my case, the qubit collapsed to $|1\rangle$ with the corresponding binary state `1`, as shown in the following figure:



This single shot run confirms that the qubit still only collapses to one of the two idealized quantum states—there's no bizarre two-headed bit created. In our subsequent quantum programs, we'll always execute them several times and keep count of the number of times each qubit collapses to $|0\rangle$ or $|1\rangle$ in each run.

H Gate on |1⟩ Qubit

The H gate acts almost identically on the |1⟩ qubit:



The blended qubit on the right still has an equal number of pentagon |0⟩ qubelets and triangle |1⟩ qubelets. But the triangle |1⟩ qubelets are inverted from those on the qubelets bundle after the H gate acted on a |0⟩ qubit. We label the quantum state on an inverted qubit with a negative sign. Thus, the operation of the H gate on a |1⟩ qubit is expressed as:

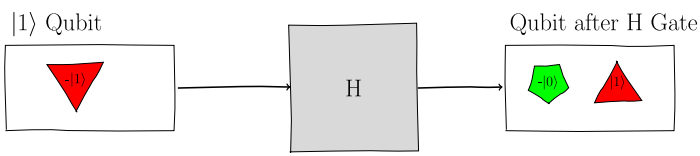
$$|1\rangle \mapsto |0\rangle \text{ and } -|1\rangle$$

When the qubit with the bundle of inverted triangle |1⟩ qubelets is inspected, a pentagon |0⟩ or |1⟩ qubelet will still be randomly picked with equal probability; in other words, examining a |1⟩ qubit immediately after it's operated on by the H gate will still be statistically equivalent had the H gate acted on a |0⟩ qubit.

It seems that the behavior of the H gate is the same whether we feed it a |0⟩ or a |1⟩ qubit—the probabilities of recording a 0 or 1 in the classical register are identical. But in quantum programming, we write quantum instructions that modify the quantum states of qubits. That is, the program works with qubelets, and it's not till the end that we collapse the qubits and record the corresponding binary states. So even though the H gate collapses the |0⟩ and |1⟩ qubits identically, it affects the pentagon |0⟩ qubelets and the triangle |1⟩ qubelets differently. (This state of affairs is akin to what John Wanamaker, a pioneering U.S. retailer, famously quipped: "Half the money I spend on advertising is wasted; the trouble is I don't know which half.") We'll have more to say about these types of collapses in Chapter 6, Designer Genes—Custom Quantum States, on page ?.

H Gate on |1⟩ Qubit with an Inverted |1⟩ Qubelet

The H gate can also act on a |1⟩ qubit in which the triangle |1⟩ qubelet is inverted on the left qubit:

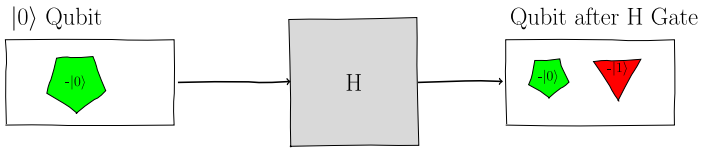


As we saw earlier, when the H gate acts on a |1⟩ qubit, an equal number of pentagon |0⟩ and triangle |1⟩ qubelets are created, but the triangle qubelets are inverted. But when a |1⟩ qubit in which the triangle |1⟩ qubelets are initially inverted is acted on by the H gate, the H gate splits the qubelets: the pentagon |0⟩ qubelets stay in their initial orientation (inverted), but the triangle |1⟩ qubelets are inverted from their initial orientation.

The chances of picking either a pentagon or triangle qubit is still roughly half, and so the qubit will still collapse to either |0⟩ or |1⟩ with approximately equal probability.

H Gate on |0⟩ Qubit with Inverted |0⟩ Qubelet

And, of course, a |0⟩ qubit could be an inverted pentagon |0⟩ qubelet, as shown in the following figure:



Here again, when the H gate operates on the inverted pentagon |0⟩ qubelet, the qubelet is split into another pentagon |0⟩ qubelet and triangle |1⟩ qubelet, but they're inverted.

Summary of Basic H Gate Operations

At first, the operation of the H gate may seem confusing. But you only need to keep in mind the following:

- The H gate is a qubelet splitter.
- When a triangle |1⟩ qubelet is split, the resulting triangle |1⟩ qubelet is inverted while the pentagon |0⟩ qubelet is not inverted.
- When an inverted qubelet is fed to the H gate, the operation of the H gate is inverted—a non-inverted qubelet is inverted and an inverted qubelet is non-inverted.

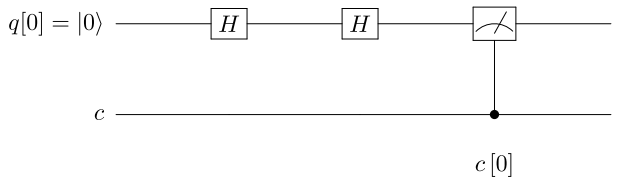
All this inverting and non-inverting of qubelets has no impact on the probability of which classical state is ultimately recorded in the classical register. This randomness, though, isn't yet helpful and, in practice, we don't collapse the qubit right away. In the next section, we'll see that inverted qubelets give a path to control the collapse of qubits and not have them flop about randomly. You'll learn to continue operating on the bundle of pentagon |0⟩ and triangle

|1⟩ qubelets till they cough up something useful—essentially, we'll tilt the odds to favor specific classical states when the qubits collapse. First we need to heighten our understanding of the superposition gates.

Back-to-Back H Gates: The First Hint of Taming Randomness

In the previous section, we saw the `H` gate operating on |0⟩ and |1⟩ qubits. When we measure a qubit that's just been fed to an `H` gate, though, all we ever get is one of the two classical states. We never see inverted qubelets. We can, however, indirectly confirm the existence of inverted qubelets in a qubit by not inspecting it immediately but by passing it to other gates before collapsing it.

So the next quantum circuit we'll look at is hooking up two consecutive `H` gates, as shown in the following figure:



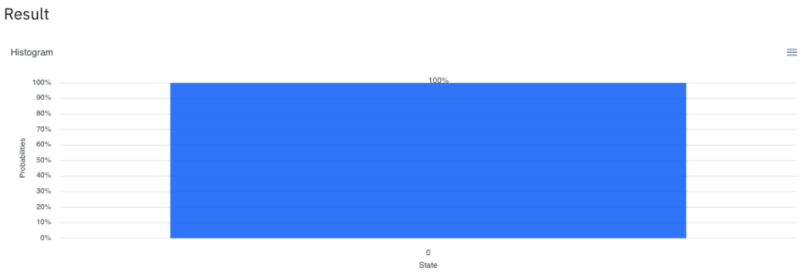
Note that the second `H` gate acts on a blended qubit that's been put in a superposition of |1⟩ and |0⟩ by the first `H` gate.

Here's the corresponding quantum program, excluding the header:

H_H_Measure.qasm

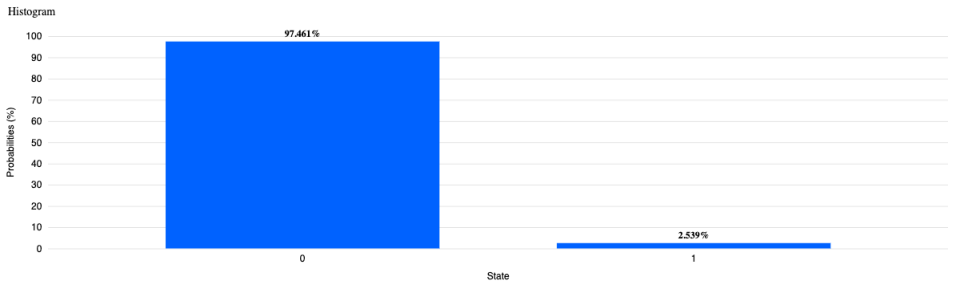
```
qreg q[1];
creg c[1];

// Back-to-back H gates
h q[0];
h q[0];

measure q[0] -> c[0];
```

We might be tempted to think that two coin tosses are just as random as a single toss—the coin still lands heads or tails with equal probability—so the collapse of the qubit after being operated on twice by the `H` gate should also collapse to `1` with roughly the same likelihood as `0`. But that isn't the case. The state of the collapsed qubit is shown in the figure on page 12.



In every shot, the qubit consistently collapses to `0`. We get the same behavior whether we run one shot, 1,024, or a million. In each case, the second toss counters the first and arrests the randomness so that, in effect, the coin always lands showing the same face. This circuit is the first clue that the collapse of a blended qubit isn't always random but can be controlled.

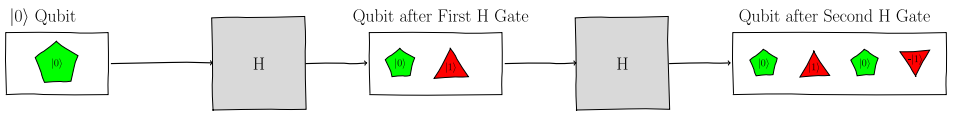
You'll see similar results on a real quantum computer, as shown in the following figure:



In almost every case, the qubit returns to the original state of $|0\rangle$. Because you're running on a real quantum computer, expect tiny disturbances so that in a small number of instances the qubit doesn't get back to $|0\rangle$.

Why Back-to-Back H Gates are Neutralizers

Analyzing this quantum circuit with the Qubelets Model gives us a clearer picture of what's going on with the qubit when it's consecutively operated on by two `H` gates. The quantum circuit is initialized with a $|0\rangle$ qubit, which is basically just a single pentagon $|0\rangle$ qubelet, as shown in the left qubit in this figure:



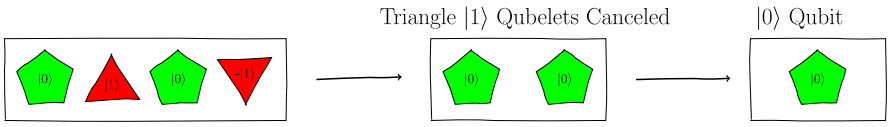
The middle qubit is the result of the first `H` gate on the left acting on the $|0\rangle$ qubit. The `H` gate splits the pentagon $|0\rangle$ qubelet in the $|0\rangle$ qubit on the left

into another pentagon |0⟩ qubelet and a triangle |1⟩ qubelet, as shown in the middle qubit in the same figure.

This blended qubit in the middle is then operated on by the second H gate on the right. This H gate splits the qubelets in the middle qubit as follows:

- The pentagon |0⟩ qubelet is split into another pentagon |0⟩ qubelet and a triangle |1⟩ qubelet.
- The triangle |1⟩ qubelet is split into a pentagon |0⟩ qubelet and an *inverted* triangle |1⟩ qubelet.

The two triangle |1⟩ qubelets, one inverted and the other non-inverted, in the right qubit in the figure cancel each other out, leaving two pentagon |0⟩ qubelets, as shown in the middle qubit here:



And such a qubit is equivalent to one with a single pentagon |0⟩ qubelet, as shown on the right. In other words, a |0⟩ qubit acted on by two back-to-back H gates just returns to being a |0⟩ qubit again. (In the exercises at the end of this chapter, you'll see that back-to-back H gates work identically on a |1⟩ qubit.)

The sequence of steps outlined on the right qubit take place instantaneously. There's no time lag. And, more importantly, even though we've drawn these transformations as happening after the gates, they really take place as the gate is operating on the qubit.

So, although a single H gate acts like a coin toss, back-to-back H gates never behave randomly. This decidedly oddball characteristic is a direct result of selectively inverting only the triangle qubelets. In fact, if the H gate was just like a fair coin toss, it wouldn't be terribly interesting from an algorithmic standpoint.

Back-to-back H gates by themselves, though, aren't very useful—you end up right where you started from. But, you learned how to analyze quantum circuits with the Qubelets Model. And it did demonstrate that randomness is controllable: the "selective negation" which wipes out qubelets gives us levers to collapse qubits in ways that solve Boolean logic expressions without having to sequentially test every possibility. We'll cover these techniques in Chapter 10, Quantum Search, on page ?, after we've investigated other ways to handle qubits.

Parallels with Quantum Mechanics

Thus far, we've talked about quantum mechanics concepts in the abstract and have not talked about how it came about and its motivations. So as to make quantum computing less science fiction and to give you confidence in the Qubelets Model, we've described experiments in Appendix 3, Quantum Mechanics with Qubelets, on page ?, whose behavior you can analyze with qubelets. You'll also see that the `H` gate is a direct derivative of these phenomena.