

Project #2

Student Names

SFUIDs:

Jiangpei Chen	301326516
Chenyu Ru	301323578
Nontawat Janpongsri	301311427
Luowen Zhu	301326420
Zhixin Huang	301326521

Purpose: This group-project is an extension of Project #1 on the specific topic described below.

Due: Friday, February 27th, 2020 (two weeks with TA submission instructions TBA)

Preliminaries:

- During the *first week*, do the assignment below *individually* resulting in a rough draft.
- During the *second week*, after the spring break, form groups of five people as close as possible. You are to prepare a *final draft* which synthesizes your individual efforts. **All rough drafts (with student names and SFUIDs) must be turned in with the final draft per group (which will include this document as a cover sheet).**
- **Note** that individuals who did *not* prepare a rough draft during the first week may be dealt with at the discretion of their team members (this is a key part of this exercise – how this is accomplished should be logged).

Task:

- Again, you are to follow the format of the following blog post:
<https://www.ybrikman.com/writing/2014/05/05/you-are-what-you-document/>
- Familiarize yourselves with **quantum computing** and **quantum algorithms** with a goal of redoing Project #1, but with a virtual quantum computing machine (like IBM's Q Experience). Most importantly, you are to extend the notion of Project #1 of **the co-processor concept** – think of a quantum computer as a co-processor as your underlying theme for this exercise. *You must explain why a quantum computer should be thought of as a co-processor and not just a stand-alone computer.*
- Submission instructions will be provided next week by the TA.
- This assignment is to be done in LaTeX.

IMPORTANT NOTES:

- I do not make attendance mandatory. However, it is the student's responsibility to find out what was discussed from your peers. Assignments and in-class activities are designed so that attendance is NOT mandatory. However, those that do not attend will find this project much more difficult than the last one.
- This project will form the basis of Project #3 – oral presentation by groups.

CMPT 376W Project 2

Jiangpei Chen 301326516 Chenyu Ru 301323578
Nontawat Janpongsri 301311427 Luowen Zhu 301326420
Zhixin Huang 301326521

Abstract

As the summary of the principle reference *You are what you document*, it summarized the documents into three different types. Written documentation, Code documentation, Community documentation. Each type of documentation solves a different problem, we will be following the structure of this document to explain quantum computing, especially virtual quantum computing machine.

1 Written documentation

As the summarize of our project 1, written documentation is the most straight forward document that we feel when we mention “documents”. There are different types under the category of written documents. I will use the guide (tutorials) and reference documents as examples to explain the quantum computing, quantum algorithm and more over on virtual quantum computing machine.

In order to have a sense of virtual quantum computing machine, first we need to understand basic quantum terminology and some related definition. After the base is built, then we can start using the documentation to illustrate a virtual quantum computing machine.

1.1 Guides and tutorials

First of all, I found some examples that can explain quantum computing and quantum algorithms.

Quantum Computing is the use of quantum-mechanical phenomena such as superposition and entanglement to perform computation. Wikipedia describes and summarizes quantum computing with the words above, then it mentioned the potential implementation of the quantum computing by using Qubits, and then it gives some introduction of the quantum algorithms.

https://en.wikipedia.org/wiki/Quantum_computing

Here are some other examples that gives a brief introduction of what quantum computing is.

<https://docs.microsoft.com/en-us/quantum/overview/what-is-quantum-computing?view=qsharp-preview>

What is quantum computing?

10/22/2019 • 5 minutes to read • 

There are some problems so difficult, so incredibly vast, that even if every supercomputer in the world worked on the problem, it would still take longer than the lifetime of the universe to solve.

Quantum computers hold the promise to solve some of our planet's biggest challenges - in environment, agriculture, health, energy, climate, materials science, and problems we've not yet even imagined. The impact of quantum computers will be far-reaching and have as great an impact as the creation of the transistor in 1947, which paved the way for today's digital economy.

Quantum computing harnesses the unique behavior of quantum physics to provide a new and powerful model of computing. The theory of quantum physics posits that matter, at a quantum level can be in a superposition of multiple classical states. And those many states interfere with each other like waves in a tide pool. The state of matter after a measurement "collapses" into one of the classical states.

Thereafter, repeating the same measurement will produce the same classical result. Quantum entanglement occurs when particles interact in ways such that the quantum state of each cannot be described independently of the others, even if the particles are physically far apart.

Quantum computing stores information in quantum states of matter and uses its quantum nature of superposition and entanglement to realize quantum operations that compute on that information, thereby harnessing and learning to program quantum interference.

Quantum computing might sound daunting, but with the right resources you can start building quantum applications today.

Figure 1: Microsoft document that explains quantum computing

Another example of explaining quantum computing

<https://quantum.country/qcvc>

1.2 Reference Documents

With the build of quantum computing and quantum algorithms, then we can go further with the discussion of virtual quantum computing.

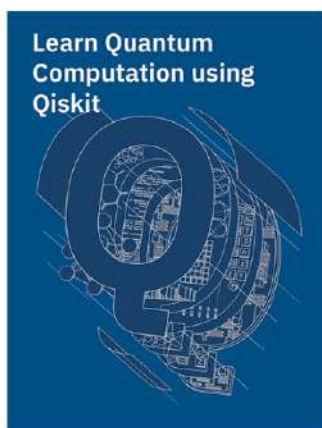
Here is one tutorial of the introduction of virtual quantum computing machine.

<http://swarm.cs.pub.ro/~agheorghiu/licenta/Quantum%20Computing%20Virtual%20Machine.pdf>

We used IBM's Q Experience as an example of the virtual quantum computing machine. Here are some reference documents that have specific and detailed information of the Q Experience quantum computing machine.

<https://quantum-computing.ibm.com/docs/> (IBM's Q Experience)

Learn Quantum Computation using Qiskit



Greetings from the Qiskit Community team! We initiated this open-source textbook in collaboration with IBM Research as a university quantum algorithms/computation course supplement based on Qiskit.

Traditional Quantum Computation Course







Learn Quantum Computation using Qiskit Textbook



Figure 2: <https://qiskit.org/textbook/preface.html> (Quantum Computation using Qiskit)

Also, there are some other virtual quantum computing machines on other platforms. Such as MicrosoftQDK.

Get started with the Quantum Development Kit (QDK)

10/23/2019 • 4 minutes to read •     +2

Welcome to the Microsoft Quantum Development Kit!

The QDK contains all the tools you'll need to build your own quantum programs and experiments with Q#, a programming language designed specifically for quantum application development.

To jump right in, you can head over to the [QDK installation guide](#). You'll then be guided through installing the Quantum Development Kit on Windows, Linux, or MacOS machines so that you can write your own quantum programs.

If you don't feel quite ready to start coding, but want to learn more about quantum computing and Q#, we encourage you to still read this article to get a feel for the resources at your disposal. In the [five questions about quantum computing](#) section, you'll find links to precisely what you're looking for!

Get started programming

The Quantum Development Kit provides many ways to learn how to develop a quantum program with Q#. To get up and running with the power of quantum, you can try out our *quickstarts*:

- The [quantum random number generator](#) is a "Q# Hello World" style application, providing a brief introduction to quantum concepts while letting you build and run a quantum application in minutes.
- [Quantum basics with Q#](#) guides you on writing a Q# program that demonstrates some of the foundational concepts of quantum programming. If you are not ready to start coding, you can still follow along without installing the QDK and get an overview of the Q# programming language and the first concepts of quantum computing.
- [Grover's search algorithm](#) offers an example of a Q# program to get an idea of the power of Q# for expressing the quantum algorithm in a way that abstracts the low-level quantum operations. This quickstart guides you through developing the program in a variety programming environments (with a Python host or with .NET language host and with Visual Studio or Visual Studio Code).

Figure 3: <https://docs.microsoft.com/en-us/quantum/welcome?view=qsharp-preview> (Microsoft QDK)

As well as the basic operation of the virtual quantum computing machine, like how to install, how to create quantum circuit. . . .

<https://quantum-computing.ibm.com/docs/qis-tut/> (Qiskit tutorials)

2 Code documentation

Quantum programming is the process of assembling sequences of instructions, called quantum programs, that are capable of running on a quantum computer.

In quantum computing, a qubit or quantum bit (sometimes qbit) is the basic unit of quantum information. So it determines quantum computer is different from normal computer, it can contain not only value 0 or value 1, but also contain some values which are in superposition state.

2.1 Quantum Code

Quantum programming is based on Q# language, here is an example of Q language from Microsoft.

Symbols

The name of a symbol bound or assigned to a value of type `T` is an expression of type `T`. For instance, if the symbol `count` is bound to the integer value `5`, then `count` is an integer expression.

Numeric Expressions

Numeric expressions are expressions of type `Int`, `BigInt`, or `Double`. That is, they are either integer or floating-point numbers.

`Int` literals in Q# are identical to integer literals in C#, except that no trailing "l" or "L" is required (or allowed). Hexadecimal and binary integers are supported with a "0x" and "0b" prefix respectively.

`BigInt` literals in Q# are identical to big integer strings in .NET, with a trailing "l" or "L". Hexadecimal big integers are supported with a "0x" prefix. Thus, the following are all valid uses of `BigInt` literals:

```
Q# Copy
let bigZero = 0L;
let bigHex = 0x123456789abcdef123456789abcdefL;
let bigOne = bigZero + 1L;
```

`Double` literals in Q# are identical to double literals in C#, except that no trailing "d" or "D" is required (or allowed).

Given an array expression of any element type, an `Int` expression may be formed using the `Length` built-in function, with the array expression enclosed in parentheses, `()`. For instance, if `a` is bound to an array, then `Length(a)` is an integer expression. If `b` is an array of arrays of integers, `Int[][]`, then `Length(b)` is the number of sub-arrays in `b`, and `Length(b[1])` is the number of integers in the second sub-array in `b`.

Given two numeric expressions of the same type, the binary operators `+`, `-`, `*`, and `/` may be used to form a new numeric expression. The type of the new expression will be the same as the types of the constituent expressions.

Figure 4: <https://docs.microsoft.com/en-us/quantum/language/expressions>

Boolean Expressions

The two `Bool` literal values are `true` and `false`.

Given any two expressions of the same primitive type, the `==` and `!=` binary operators may be used to construct a `Bool` expression. The expression will be true if the two expressions are equal, and false if not.

Values of user-defined types may not be compared, only their unwrapped values can be compared. For example, using the "unwrap" operator `!` (explained in the [Q# type model page](#)),

```
Q# Copy

newtype WrappedInt = Int;    // Yes, this is a contrived example
let x = WrappedInt(1);
let y = WrappedInt(2);
let z = x! == y!;            // This will compile and yield z = false.
let t = x == y;              // This will cause a compiler error.
```

Equality comparison for `qubit` values is identity equality; that is, whether the two expressions identify the same qubit. The state of the two qubits are not compared, accessed, measured, or modified by this comparison.

Equality comparison for `Double` values may be misleading due to rounding effects. For instance, `49.0 * (1.0/49.0) != 1.0`.

Given any two numeric expressions, the binary operators `>`, `<`, `>=`, and `<=` may be used to construct a new Boolean expression that is true if the first expression is respectively greater than, less than, greater than or equal to, or less than or equal to the second expression.

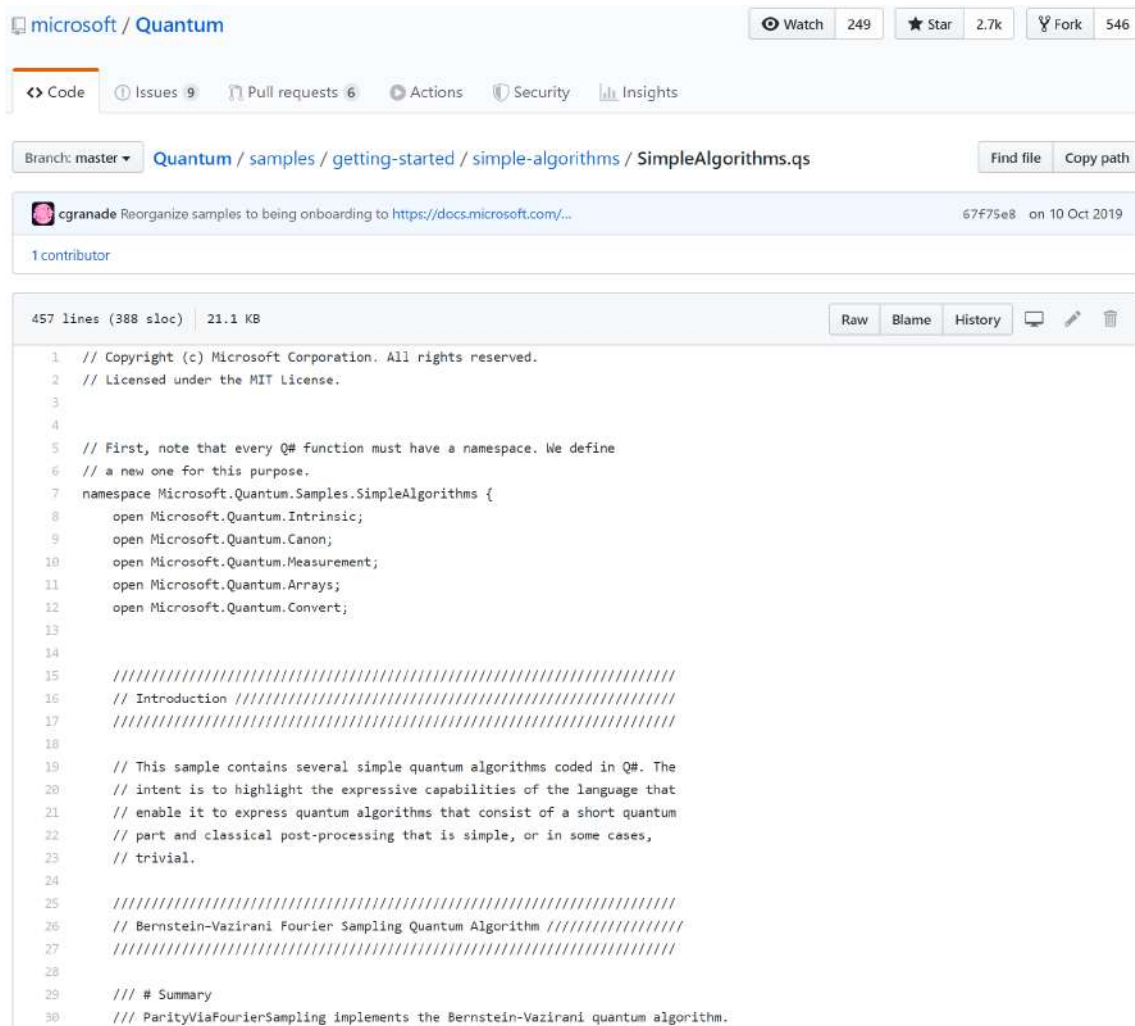
Given any two Boolean expressions, the `and` and `or` binary operators may be used to construct a new Boolean expression that is true if both of (resp. either or both of) the two expressions are true.

Given any Boolean expression, the `not` unary operator may be used to construct a new Boolean expression that is true if the constituent expression is false.

Figure 5: <https://docs.microsoft.com/en-us/quantum/language/expressions>

2.2 Quantum Algorithm

Just like modern computers, quantum computer also have to be based on quantum algorithm to solve problems. Here are some examples of quantum algorithm:



The screenshot displays the GitHub interface for the `microsoft / Quantum` repository. The repository has 249 watches, 2.7k stars, and 546 forks. The file `SimpleAlgorithms.js` is selected, showing its commit history and contributors. The file content is as follows:

```
1 // Copyright (c) Microsoft Corporation. All rights reserved.
2 // Licensed under the MIT License.
3
4
5 // First, note that every Q# function must have a namespace. We define
6 // a new one for this purpose.
7 namespace Microsoft.Quantum.Samples.SimpleAlgorithms {
8     open Microsoft.Quantum.Intrinsic;
9     open Microsoft.Quantum.Canon;
10    open Microsoft.Quantum.Measurement;
11    open Microsoft.Quantum.Arrays;
12    open Microsoft.Quantum.Convert;
13
14
15    ///////////////////////////////////////////////////////////////////
16    // Introduction ///////////////////////////////////////////////////////////////////
17    ///////////////////////////////////////////////////////////////////
18
19    // This sample contains several simple quantum algorithms coded in Q#. The
20    // intent is to highlight the expressive capabilities of the language that
21    // enable it to express quantum algorithms that consist of a short quantum
22    // part and classical post-processing that is simple, or in some cases,
23    // trivial.
24
25    ///////////////////////////////////////////////////////////////////
26    // Bernstein-Vazirani Fourier Sampling Quantum Algorithm ///////////////////////////////////////////////////////////////////
27    ///////////////////////////////////////////////////////////////////
28
29    /// # Summary
30    /// ParityViaFourierSampling implements the Bernstein-Vazirani quantum algorithm.
```

Figure 6: <https://github.com/microsoft/Quantum/blob/master/samples/getting-started/simple-algorithms/SimpleAlgorithms.js>

And the classical quantum computing algorithm for integer factorization, Shor's algorithm that can factorize big integers in polynomial-time.

Proceed as follows:

1. Initialize the registers to

$$\frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle = \left(\frac{1}{\sqrt{2}} \sum_{x_1=0}^1 |x_1\rangle \right) \otimes \cdots \otimes \left(\frac{1}{\sqrt{2}} \sum_{x_q=0}^1 |x_q\rangle \right),$$

where \otimes denotes the [tensor product](#).

This initial state is a superposition of Q states, and is easily obtained by generating q independent [qubits](#), each a superposition of 0 and 1 states.

We can accomplish this by initializing the qubits to the zero-position, and then applying the [hadamard gate](#) in parallel to each of the q qubits, where $2^q = Q$.

2. Construct $f(x)$ as a quantum function and apply it to the above state, to obtain

$$U_f |x, 0^q\rangle = |x, f(x)\rangle$$

$$U_f \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x, 0^q\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x, f(x)\rangle$$

This is still a superposition of Q states. However, the $q + n$ qubits, i.e, the q input qubits and $n (> \log_2(N))$ output qubits, are now entangled or not [separable](#), as the state cannot be written as a tensor product of states of individual qubits.

3. Apply the inverse [quantum Fourier transform](#) to the input register. This transform (operating on a superposition of $Q = 2^q$ states) uses a Q -th [root of unity](#) such as $\omega = e^{\frac{2\pi i}{Q}}$ to distribute the amplitude of any given $|x\rangle$ state equally among all Q of the $|y\rangle$ states, and to do so in a different way for each different x . We thus obtain

Figure 7: https://en.wikipedia.org/wiki/Shor%27s_algorithm

2.3 Quantum circuits

In quantum computing there is a special programming which is quantum circuits.

A *quantum circuit* is a model for quantum computation in which a computation is a sequence of quantum gates, which are reversible transformations on a quantum mechanical analog of an n -bit register. (https://en.wikipedia.org/wiki/Quantum_circuit)

Here is an example of quantum circuits based on IBM circuits composer experience:

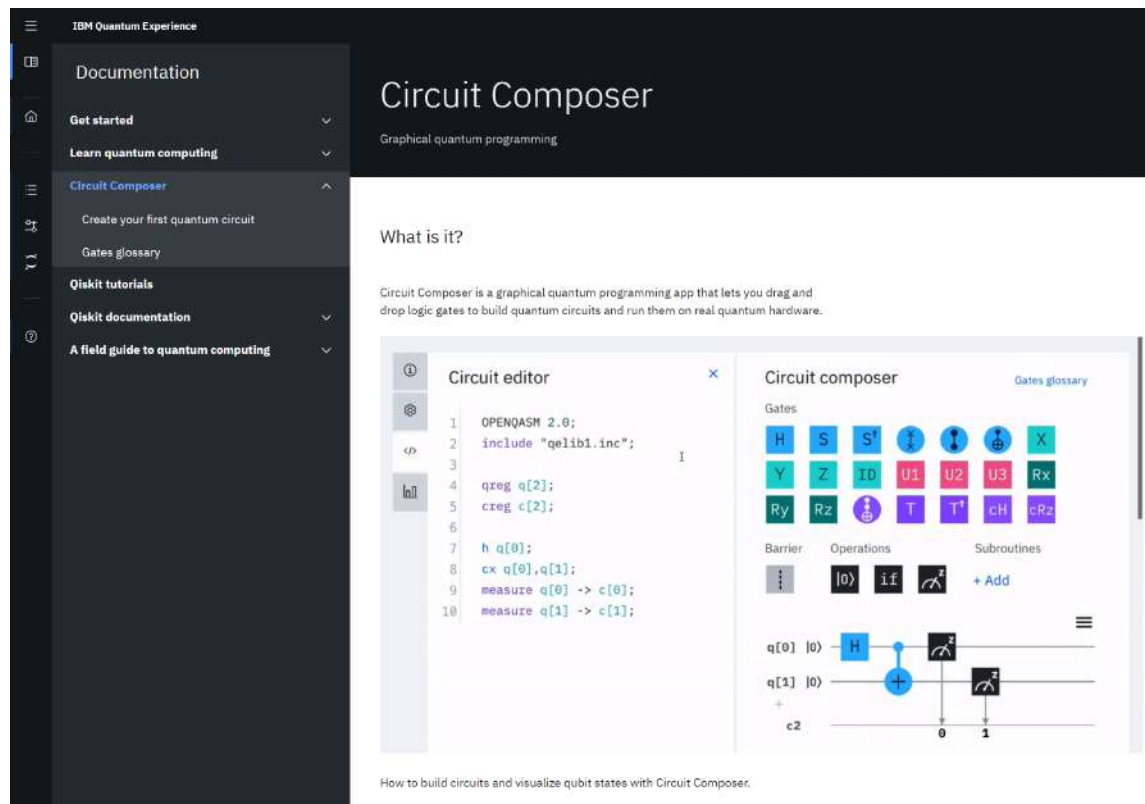


Figure 8: <https://quantum-computing.ibm.com/docs/circ-comp/>

3 Community documentation

Community documentation is an open source document. There are many forms and uses for the community documentation. One of the forms is QA. This form allows software developers to seek help from another software developer. Another form of community documentation is blog posts. This form is an open source documentation that is provided by the community to share their knowledge about the topic.

3.1 QA

As a new software developer who just started researching on quantum computing, reading the written documentation to understand the very basic terminology might be difficult. Thus, posting your question about quantum computing online can help you get a more understandable explanation to the problem you do not understand. Below is an example of QA form, explaining the basic of quantum computing.

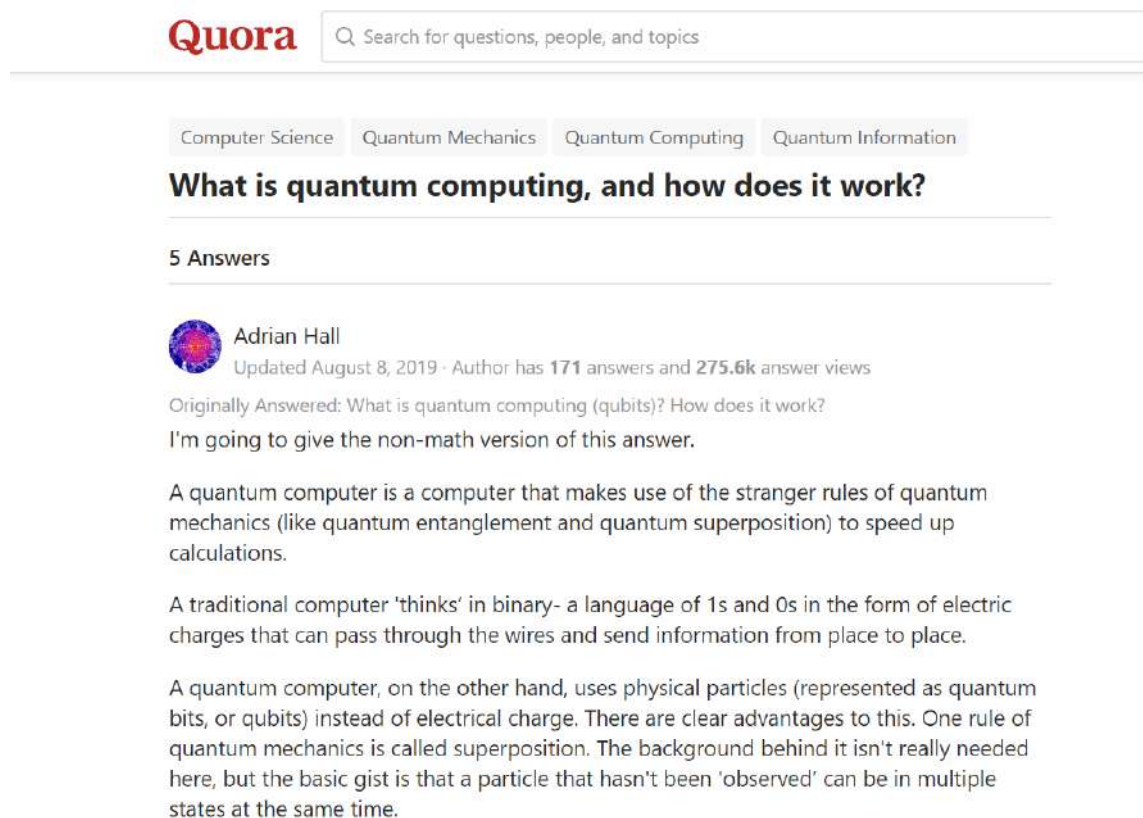



Figure 9: <https://www.quora.com/What-is-quantum-computing-and-how-does-it-work>

Therefore, to get the general idea of quantum computing as a coprocessor and not just a stand alone computer. We can simply post our question and get the explanation from a QA form. Below

is an example of an explanation received from a question about why quantum computing is not a stand alone computer.

Quora



Scott Alexander, PhD in CS and have worked as a sysadmin, application programmer, and researcher
Answered Oct 10, 2017 · Author has **243** answers and **1m** answer views

Quantum “computers” (or more precisely coprocessors) can solve certain problems more quickly than classical computers. They do this by using some of the strange properties of quantum mechanics. (Famously, Einstein referred to entanglement, a prime element of quantum computing, as “spooky action at a distance.” He said this because he didn’t believe that the universe could work this way. There is evidence to show that he was wrong.)

So quantum coprocessors are not just fast classical computers. Instead, they use a different way of performing the computation to allow them to answer certain questions quickly. Two of the best known / most useful quantum algorithms are Shor’s and Grover’s.

Shor’s algorithm factors numbers more quickly than any known classical algorithm. In particular, encryption algorithms like RSA rely on the idea that we cannot factor numbers quickly. If practical quantum computing becomes a thing, that will no longer be true and it will be trivial to decrypt information encrypted with such an algorithm. By contrast, using classical computers with the algorithms that we currently know, a faster classical computer would have to be ridiculously (perhaps impossibly) faster than current computers.

Grover’s algorithm finds an element of an unsorted set of data in sublinear time. For the classical algorithm, of course, you end up looking at every piece of data until you find the one you want. On average, you have to look at half the data before finding what you want. Grover’s finds the data without having to look at that much of the data. I can’t even imagine how I would approach trying to do that classically.

Figure 10: <https://www.quora.com/Is-a-quantum-computer-just-a-fast-computer>

3.2 Blog post

Since quantum computing is still in a developing stage, being able to get immediate update information can be crucial for software developers. Thus, with the help of a blog post from a company that works on quantum computing such as IBM. Software developers can now easily get the latest information about quantum computing. Below is a screenshot of IBM blog post page for quantum computing

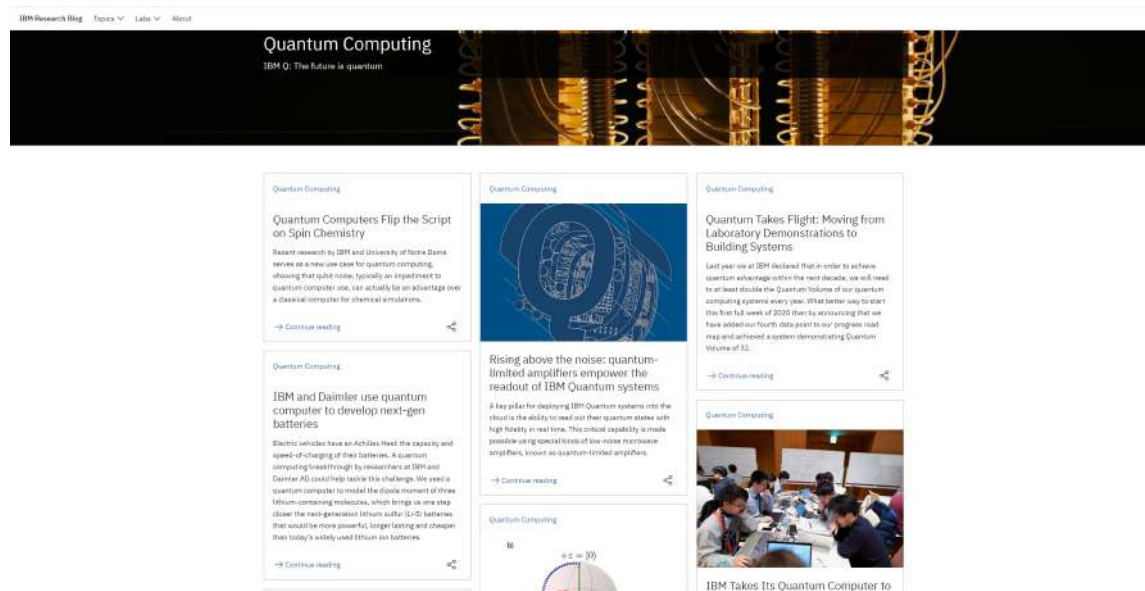


Figure 11: <https://www.ibm.com/blogs/research/category/quantcomp/>

4 Conclusion

In conclusion, there are quite a lot of advantages of quantum computer and quantum algorithms. Quantum computer uses qubits instead of bits compared with a normal computer. The normal bits can only store information as binary 0 and 1 states, while one qubit can store 2 states at the same time, which means if a quantum computer has 10 qubits, it can run 1024 states at the same time. That is equal to a normal computer use 1024 bits to run. If the quantum computer has large enough qubits like about 50, its efficiency is equal to thousands of normal computers run at the same time.

However, the quantum computer can not run without the quantum algorithm. Because like the normal computer, that we need to store the true and false value into register in order to perform calculation, qubits also need the quantum algorithm to run. For example the Shor's algorithm, it can use qubits to solve the discrete logarithm problem and the integer factorization problem in polynomial time. Also, the qubits use photons or some other particles as a physical medium and they need to have a very strict environment to keep their excited state.

Quantum computers can not run by itself. A reasonable quantum computer, should be implemented with a suitable quantum algorithm, and an environment that keeps photons or particles in excited state and so on. Therefore, it can be thought of as a co-processor.

Written document

I choose some Tutorials, walkthroughs, and guides as a sample of the written document of quantum computing.

MIT
Technology
Review


Sign inSubscribe

TopicsMagazineNewslettersEvents

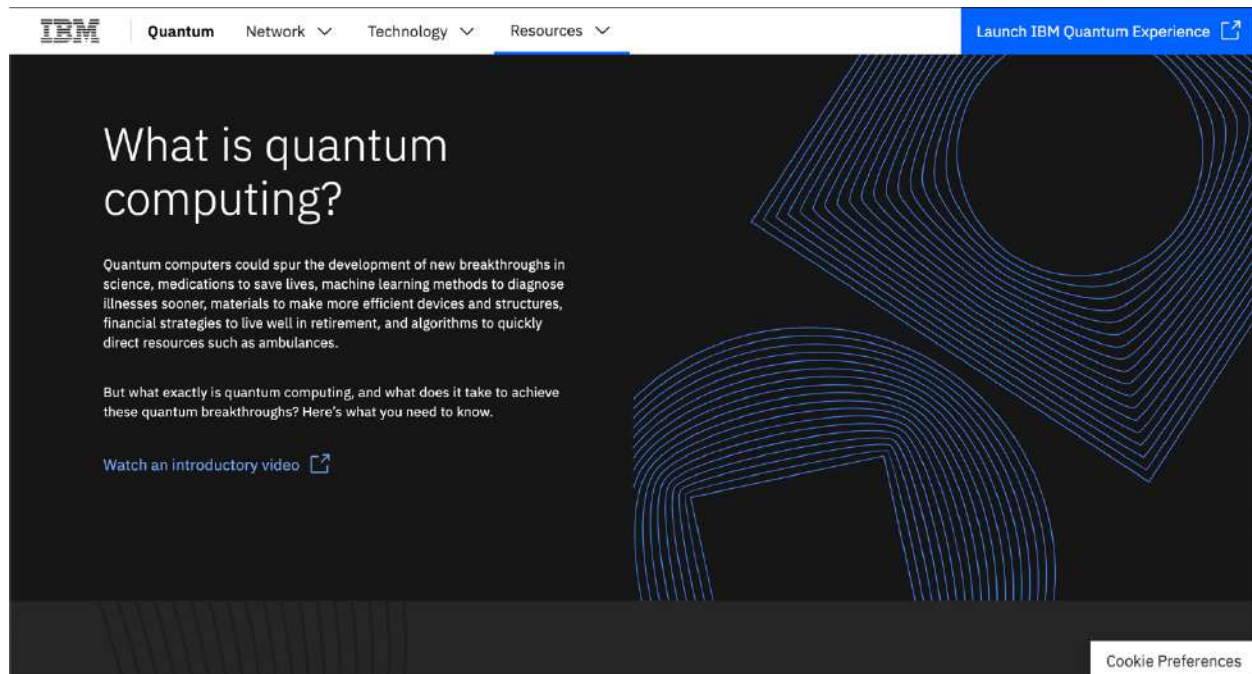
Computing / Quantum Computing

Explainer: What is a quantum computer?

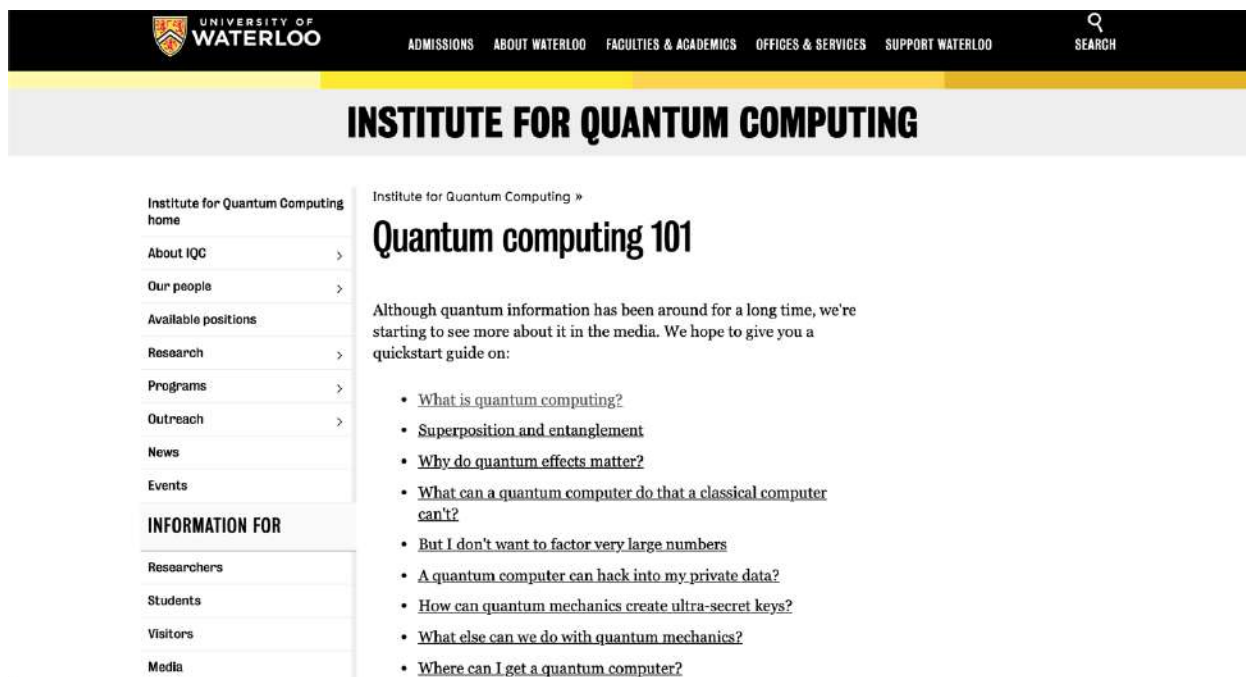
How it works, why it's so powerful, and where it's likely to be most useful first



<https://www.technologyreview.com/s/612844/what-is-quantum-computing/>



<https://www.ibm.com/quantum-computing/learn/what-is-quantum-computing/>



<https://uwaterloo.ca/institute-for-quantum-computing/quantum-computing-101#What-is-quantum-computing>

Code document

This is an example of how to create a quantum circuit.

The screenshot shows the IBM Quantum Experience web application. On the left is a dark sidebar with a menu containing items like 'Multiple qubits', 'Entanglement and Bell tests', 'GHZ states', 'Quantum algorithms', 'Basic circuit identities and larger circuits', 'Grover's algorithm', 'Deutsch-Jozsa algorithm', 'Learning parity with noise', 'Quantum phase estimation', 'Shor's algorithm', 'Quantum error correction', 'Quantum repetition code', and 'Stabilizer measurements'. The main content area is titled 'Qiskit example' and contains a Python code snippet for a single-qubit measurement circuit. Below the code, there is explanatory text about quantum gates and a small blue icon in the bottom right corner.

```
# single_q_measurement.py
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, execute, Aer

# Define the Quantum and Classical Registers
q = QuantumRegister(1)
c = ClassicalRegister(1)

# Build the circuit
single_q_measurement = QuantumCircuit(q, c)
single_q_measurement.measure(q, c)

# Execute the circuit
job = execute(single_q_measurement, backend = Aer.get_backend('qasm_simulator'),
shots=1024)
result = job.result()

# Print the result
print(result.get_counts(single_q_measurement))
```

As you may have guessed, we need to be able to put the qubit in other states. To do this, we require the concept of a *quantum gate*. A single-qubit quantum gate is a 2×2 unitary matrix (since quantum gates must be reversible and preserve probability amplitudes, the matrices must be unitary). The quantum state $|\psi\rangle$ after the action of the gate is found by multiplying the original quantum state by the gate

The screenshot shows the 'Documentation' section of the IBM Quantum Experience. The sidebar menu includes 'Get started' (with 'Create your first quantum circuit' and 'Code your first quantum circuit'), 'Learn quantum computing', 'Circuit Composer' (with 'Create your first quantum circuit' and 'Gates glossary'), 'Qiskit tutorials', 'Qiskit documentation' (with 'Installing Qiskit', 'Getting Started with Qiskit', and 'The Qiskit Elements'), and 'The Qiskit Elements'. The main content area is titled 'Create your first quantum circuit' and contains an introductory paragraph about quantum computing, a paragraph about the documentation's origin, and a 'Table of contents' with links to 'The state of a qubit', 'The quantum NOT gate', 'Next steps', and 'References'.

Create your first quantum circuit

Start your quantum journey with an introduction to the qubit and the quantum NOT gate, the two fundamental concepts you need to understand to quantum compute. Then, explore how to use Circuit Composer to create your first quantum circuit, run your circuit on IBM Quantum Experience, and interpret the results.

You do not need to be familiar with quantum computing or classical programming to follow along. It was adapted from the excellent introduction to quantum computing, *Quantum Computing for the Very Curious* [1].

Table of contents

- [The state of a qubit](#)
- [The quantum NOT gate](#)
- [Next steps](#)
- [References](#)

<https://quantum-computing.ibm.com/docs/guide/wwwq/the-qubit>

Community document

This is also an important documentation for a software. It is used for the people who involved with the project discuss questions and take some help.

Quantum Computing

Quantum Computers Flip the Script on Spin Chemistry

Recent research by IBM and University of Notre Dame serves as a new use case for quantum computing, showing that qubit noise, typically an impediment to quantum computer use, can actually be an advantage over a classical computer for chemical simulations.

[→ Continue reading](#)



Quantum Computing

IBM and Daimler use quantum computer to develop next-gen batteries

Quantum Computing



Rising above the noise: quantum-limited amplifiers empower the readout of IBM Quantum systems

A key pillar for deploying IBM Quantum systems into the cloud is the ability to read out their quantum states with high fidelity in real time. This critical capability is made possible

Quantum Computing

Quantum Takes Flight: Moving from Laboratory Demonstrations to Building Systems

Last year we at IBM declared that in order to achieve quantum advantage within the next decade, we will need to at least double the Quantum Volume of our quantum computing systems every year. What better way to start this first full week of 2020 than by announcing that we have added our fourth data point to our progress road map and achieved a system demonstrating Quantum Volume of 32.

[→ Continue reading](#)



StackExchange
Search on Quantum Computing...
Log in
Sign up

Home
Questions
Tags
Users
Unanswered

Explore our Questions

algorithm quantum-gate quantum-state qiskit programming quantum-information entanglement circuit-construction ibm-q-experience mathematics more tags

Active Hot Week Month

0 votes
0 answers
11 views

Turing Complete Equivalent for Quantum Computers - modeling singularity (e.g. black hole) with Q#

complexity-theory q# models

modified 3 hours ago Martin Vesely 2,484

1 vote
1 answer
8 views

Cirq: creating a new device with custom coupling topology?

cirq

answered 6 hours ago Craig Gidney 7,851

-1 votes
0 answers
11 views

How are algorithms for quantum computers written?

algorithm quantum-operation

asked 7 hours ago Navoneel Karmakar 1

0 votes
1 answer
14 views

Breaking cryptography with quantum computers. Is it really possible in 50 years?

quantum-information

answered 7 hours ago Martin Vesely 2,494

0 votes
1 answer
15 views

Qiskit Statevector_simulator returns different results

qiskit

modified 12 hours ago Stacia Wood 84

Microsoft Azure

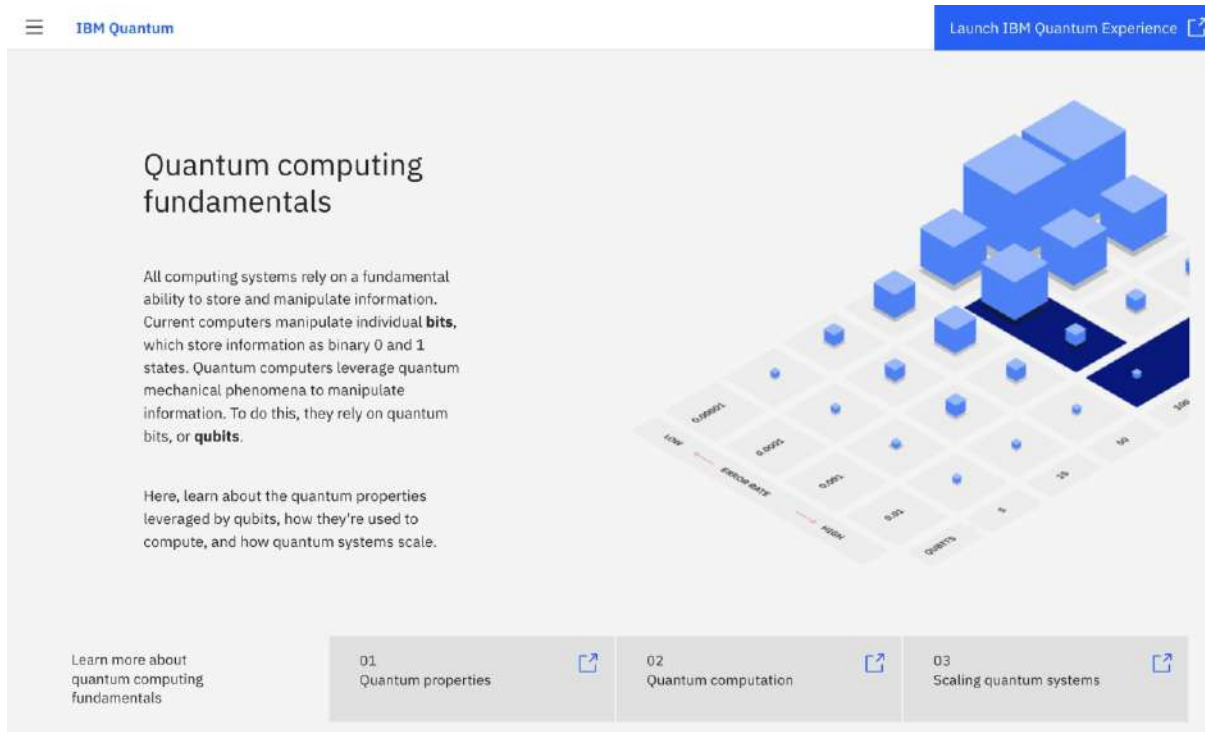
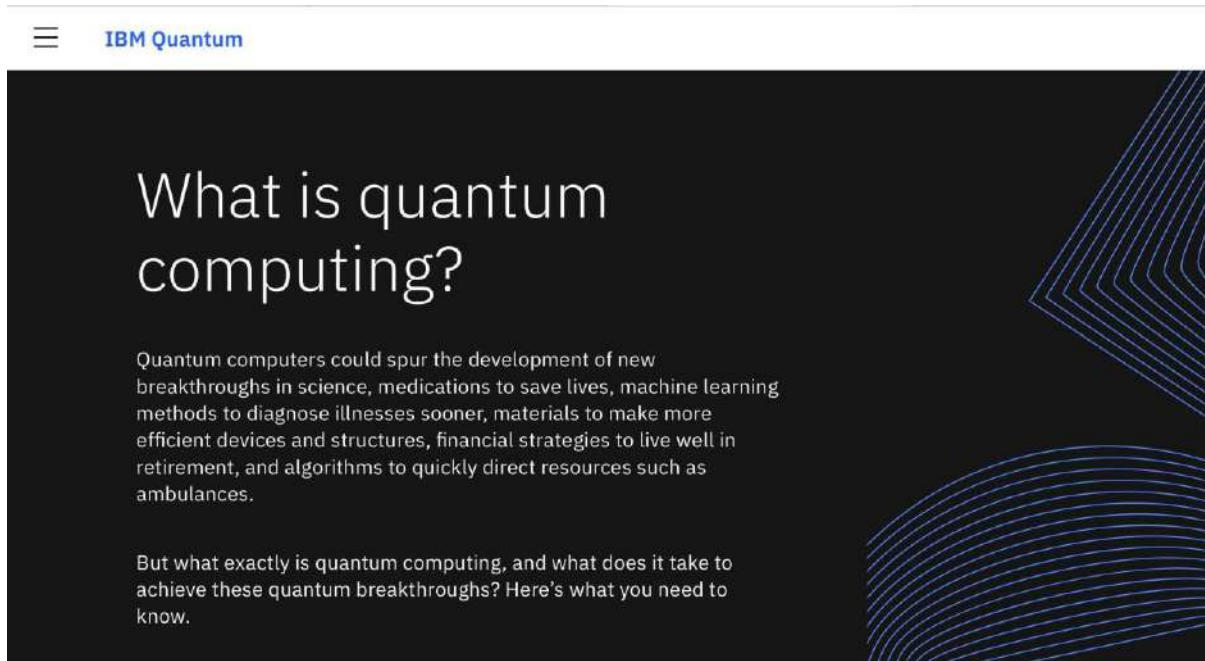
Build your next great app for free with Azure

Try Azure Free

Hot Network Questions

- ALTER VIEW drops Index from View
- Could a charity manufacture patented medications for free?
- I used a fork of a GitHub repository whose paper I cited – should I also cite the fork?
- Is a dystopian surveillance state computationally possible?
- Would lightning be an effective weapon against Cybermen?
- How to best deal with cliques at the table during "downtime"?
- How to press a touch sensitive elevator button without touching it

Written documentation



Code documentation

```
import numpy as np
from qiskit import(
    QuantumCircuit,
    execute,
    Aer)
from qiskit.visualization import plot_histogram

# Use Aer's qasm_simulator
simulator = Aer.get_backend('qasm_simulator')

# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)

# Add a H gate on qubit 0
circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)

# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])

# Execute the circuit on the qasm simulator
job = execute(circuit, simulator, shots=1000)

# Grab results from the job
result = job.result()

# Returns counts
counts = result.get_counts(circuit)
print("\nTotal count for 00 and 11 are:",counts)


# Draw the circuit
circuit.draw()
```

```
import numpy as np
from qiskit import(
    QuantumCircuit,
    execute,
    Aer)
from qiskit.visualization import plot_histogram
```



In more detail, the imports are

- `QuantumCircuit`: can be thought as the instructions of the quantum system. It holds all your quantum operations.
- `execute`: runs your circuit / experiment.
- `Aer`: handles simulator backends.
- `plot_histogram`: creates histograms.

Community documentation



Search


 

IBM Research Blog Topics Labs About


Quantum Computing

Quantum Computers Flip the Script on Spin Chemistry

Recent research by IBM and University of Notre Dame serves as a new use case for quantum computing, showing that qubit noise, typically an impediment to quantum computer use, can actually be an advantage over a classical computer for chemical simulations.

[→ Continue reading](#) 

Quantum Computing




Rising above the noise: quantum-limited amplifiers empower the readout of IBM Quantum systems

A key pillar for deploying IBM Quantum systems into the cloud is the ability to read out their quantum states with high fidelity in real time. This critical capability is made possible using special kinds of low-noise microwave amplifiers, known as quantum-limited amplifiers.

Quantum Computing

Quantum Takes Flight: Moving from Laboratory Demonstrations to Building Systems

Last year we at IBM declared that in order to achieve quantum advantage within the next decade, we will need to at least double the Quantum Volume of our quantum computing systems every year. What better way to start this first full week of 2020 than by announcing that we have added our fourth data point to our progress road map and achieved a system demonstrating Quantum Volume of 32.

[→ Continue reading](#) 

Quantum Computing

IBM and Daimler use quantum computer to

National Accelerator Laboratory, has joined the IBM Q Network. As a member organization, Q-FARM will collaborate with IBM to accelerate joint research in quantum computing and develop curricula to help prepare students for careers that will be influenced by this next era of computing across science and business.

→ Continue reading



Quantum Computing



Qiskit – Write once, target multiple architectures

Qiskit has the flexibility to target different underlying quantum

Quantum Computing

IBM and Wells Fargo Collaborate to Accelerate Innovation

IBM Research is embarking on a multi-year, collaborative effort with Wells Fargo focused on research and learning that is intended to enhance the company's artificial intelligence and quantum computing capabilities. Together with IBM Research, Wells Fargo plans to accelerate its learnings to inform innovation initiatives that reimagine the future of financial services in a way that is designed to deliver customer experiences that are simple, fast, safe and convenient.

→ Continue reading



Quantum Computing



IBM and the Unitary Fund Unite for Open Source Projects for Quantum Computing

We are pleased to announce our support to grow the community of quantum enthusiasts and explorers, by partnering with the Unitary Fund to provide funding for grants and priority access to certain IBM Q systems.

→ Continue reading



Quantum Computing



Rough Draft of Project 2

Jiangpei Chen
301326516

As the summary of the principle reference *You are what you document*, it summarized the documents into three different types. Written documentation, Code documentation, Community documentation. Each type of documentation solves a different problem, I will be following the structure of this document to explaining quantum computing, especially virtual quantum computing machine.

Written documentation

- Use some of the introduction documentation, such as tutorial document to introduce quantum computing and quantum algorithm
https://en.wikipedia.org/wiki/Quantum_computing
<https://docs.microsoft.com/en-us/quantum/overview/what-is-quantum-computing?view=qsharp-preview>
<https://quantum.country/qcvc>
- Then use the introduction tutorials to introduce some of the virtual quantum computing machine
<https://quantum-computing.ibm.com/docs/> (IBM's Q Experience)
<https://docs.microsoft.com/en-us/quantum/welcome?view=qsharp-preview> (Microsoft QDK)
<http://swarm.cs.pub.ro/~agheorghiu/licenta/Quantum%20Computing%20Virtual%20Machine.pdf>
- As well as the basic operation of the virtual quantum computing machine, like how to install, how to create quantum circuit....
<https://quantum-computing.ibm.com/docs/qis-tut/> (Qiskit tutorials)

Code documentation

- Using some of the code examples to illustrate how the virtual quantum computing machine works
<https://quantum-computing.ibm.com/docs/qis-tut/#qiskit-terra>
<https://quantum-computing.ibm.com/docs/qis-tut/#qiskit-aer>
<https://quantum-computing.ibm.com/docs/qis-tut/#qiskit-ignis>
- Some other example using Q#
- <https://docs.microsoft.com/en-us/samples/browse/?languages=qsharp&view=qsharp-preview>
<https://docs.microsoft.com/en-ca/samples/microsoft/quantum/measuring-qubits/>
- Some APIs
<https://docs.microsoft.com/en-us/qsharp/api/qsharp/microsoft.quantum.amplitudeamplification?view=qsharp-preview>(Microsoft)
<https://quantum-computing.ibm.com/docs/qiskit/apidoc/qiskit-terra/qiskit> (Qiskit)

Community documentation

- Some community documents that includes blogs and board discussions of quantum computing machine
<https://stackoverflow.com/questions/3163234/does-anyone-know-what-quantum-computing-is/3175933#3175933> (quantum computing)
<https://www.nature.com/articles/npjqi201523> (introduction quantum computing)

Draft
LUOWEN ZHU
301326420

I will use some written documentations, Code documentations and the Community documentations to introduce the quantum computer and the quantum algorithm.

Written documentation:

Here is the introduction of the quantum computer which is from the IBM company

<https://www.ibm.com/quantum-computing/learn/what-is-quantum-computing/>

And here is some tutorial about how to create quantum circuit

<https://quantum-computing.ibm.com/docs/start-iqx/drag-drop/first-circ>

Code documentation

Then use some code documentation about the quantum computing can understand thing deeply and here is a code about quantum

<https://quantum-computing.ibm.com/docs/start-iqx/code/first-circ>

Community documentation

After knowing the quantum computer form the written documentation and the code documentation, if you have more question about quantum computing, you can view the community documentation to find the answer.

<https://quantumcomputing.stackexchange.com/questions/tagged/ibm-q-experience>

<https://quantumcomputing.stackexchange.com/questions/tagged/qiskit>

CMPT 376 Project 2 Draft

Zhixin Huang

301326521

1. Written documentation

Code your first quantum circuit

Learn to code your first quantum circuit without downloading anything to your computer, by using Qiskit notebooks embedded within IBM Quantum Experience. While you can benefit from having some familiarity with [Python](#) and [quantum computing](#), you can get a sense of the big picture without those prerequisites.

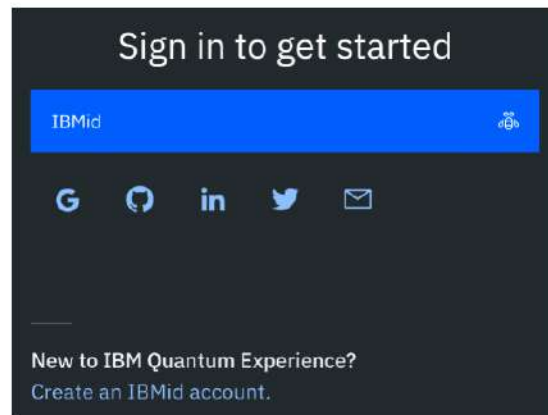
To code a quantum circuit with Qiskit, you follow three high-level steps:

- **Build:** design a quantum circuit that represents the problem you are considering.
- **Execute:** run experiments on different backends, either systems or simulators.
- **Analyze:** calculate summary statistics and visualize the results of experiments.

These instructions guide you through building a circuit with example code in a Qiskit Notebook, executing your program, and analyzing the results. You will then learn in greater detail how each component of the program functions.

Sign in to IBM Quantum Experience

1. Go to [IBM Quantum Experience](#).
2. Sign in or Create an IBMid account.



2. Code documentation

Code Snippet 1

```
# Build
#-----

# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)

# Add a H gate on qubit 0
circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)

# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])

# Execute
#-----

# Use Aer's qasm_simulator
simulator = Aer.get_backend('qasm_simulator')

# Execute the circuit on the qasm simulator
job = execute(circuit, simulator, shots=1000)

# Grab results from the job
result = job.result()


# Return counts
counts = result.get_counts(circuit)
print("\nTotal count for 00 and 11 are:",counts)


# Analyze
#-----

# Draw the circuit
circuit.draw()
```

```
Total count for 00 and 11 are: {'00': 479, '11': 521}
```

3. Community documentation

 Quantum Computing

SPONSORED BY 

[Home](#)
Questions
[Tags](#)
[Users](#)
[Unanswered](#)

Questions tagged [ibm-q-experience]

Ask Question

Use this tag for questions about IBM's quantum processors which are accessible online. DO NOT use this for questions about general Qiskit issues. As of July 2018, there are five processors on the IBM Q Experience: two 5-qubit processors, one 16-qubit processor and two 20-qubit processors (only available to hubs, partners, and members of the IBM Q network).

[Learn more...](#) [Top users](#) [Synonyms](#)

176 questions

Newest Active Bountied Unanswered More

Filter

1 vote

1 answer


32 views

Are cRz with angle π and cZ equal on the IBM Quantum Experience?

I was trying to implement the 2-qubit Grover algorithm without the auxiliary bit. Before finding the gate symbol for the controlled Z , I worked with the controlled R_x . The gate ...

qiskit circuit-construction ibm-q-experience grovers-algorithm

asked 2 days ago

 Daniel Müslig 175 • 5

10 votes

1 answer


2k views

Which subatomic particle does each company use in quantum computing?

Probably each company (Google, Amazon, Intel, IBM, Microsoft, D-Wave and so on) uses a mix of subatomic particles and technologies. I would like to know which particles/technologies are used by each ...

ibm-q-experience physical-realization d-wave google-sycamore

asked Feb 27 at 19:32

 Felipe Rojo Amador 123 • 5

4 votes

1 answer


300 views

How can I get the entire histogram in a 5 qubit program done in IBM quantum experience machine?

I have run a program on an IBM quantum experience machine which involves 5 qubits. Obviously when I measure the whole system I have 32 results but the machine only represents 20 in the histogram. In ...

programming ibm-q-experience

asked Feb 27 at 10:06

 user10202 41 • 1

0 votes

0 answers


35 views

How can I find the fidelity of preparation?

I want to know the fidelity (or error rate) of the preparation $|0\rangle$. How can I have it?

quantum-state qiskit ibm-q-experience fidelity

asked Feb 27 at 1:39

 Yongsoo 1

2 votes

1 answer


51 views

Why I am not getting approximate equiprobable states in the following circuit on IBM Q simulator?

I have the following circuit which consists of symmetric modules. I compiled the circuit in IBM Quantum experience backend - ibmq_qasm_simulator, 8192 shots. The result is not equiprobable. Why? Since ...

circuit-construction ibm-q-experience

asked Feb 25 at 14:10

 Adam Levine 393 • 10

Blog

The eight factors of happiness for developers

This week, #StackOverflowKnows outlaw wifi, GPU weakness, and neutrinos per...

Featured on Meta

The company's commitment to rebuilding the relationship with you, our community

The Q1 2020 Community Roadmap is on the Blog

Planned maintenance scheduled for Saturday, March 7, 2020 at 14:00 UTC (SAM...

Related Tags

qiskit • 78

programming • 42

quantum-gate • 24

quantum-state • 12

circuit-construction • 12

ibm • 11

algorithm • 8

error-correction • 6

measurement • 6

simulation • 6

[more related tags](#)

Hot Network Questions

Minimizing the expectation of the loss function

16 number puzzles on a blackboard

ALTER VIEW drops index from View

When low-level formatting a floppy, how does one ensure only existing existing sector markers are overwritten?

How to be an astrophysicist?

Do not vote, one vote will not reverse election results. What is wrong with this reasoning?