MITRE's Research in Quantum Software Engineering and Workforce Education

Joe Clapis

Monthly Lecture + Education Series

GRIFFISS INSTITUTE

INNOVARE ADVANCEMENT CENTER

October 20, 2021

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

About MITRE’s Quantum Software Group

Paper Name	Use Case	Speedup	
A Quantum Pattern Recognition Method for Improving Pairwise Sequence Alignment	Aligning multiple sequences of data	Exponential	ht /s
Bayesian deep learning on a quantum computer	Machine Learning	Polynomial	ht /s
Quantum algorithm for visual tracking	Tracking an object through video frames	Quadratic	ht /P
An improved quantum algorithm for ridge regression	ML, analyzing data suffering from "multicollinearity" with linear regression	Exponential / Polynomial, Depends on Input	ht /T
~~Quantum image matching~~ (This paper is wrong, see next row)	Finding a smaller "target" image within a larger overall image	Quadratic?	ht /s
Analysis and improvement of the quantum image matching	Finding a smaller "target" image within a larger overall image	Quadratic	https://doi.org/10.1007/s11128-017-1723-7
FCM-based quantum artificial bee colony algorithm for image segmentation	Classifying pixels based on which region (e.g. foreground, background) they belong to	???	https://doi.org/10.1145/3240876.3240907
Image classification based on quantum K-Nearest-Neighbor algorithm	Image classification (selecting which of a pre-defined set of groups a picture belongs to)	Quadratic?	https://doi.org/10.1007/s11128-018-2004-9

SDK	MCX Mode	Backend	Circuit Creation Time	Compile Time	Width	NEQR Depth	QDP Depth	QFT Depth	Total Depth
QDK	---	Resource Counter	00:01.8	---	39	150,239	5	178	150,417
Qiskit	CCNOT Chain	Aer Sim	00:45.2	05:07.3	39	384,312	4	49	384,350
Qiskit	Single-Qubit	Aer Sim	03:54.6	18:38.7	30	1,940,132	4	49	1,940,173
Qiskit	CCNOT Chain	Rochester	00:53.9	4:34:26.7	53	1,027,772	20	1108	1,028,758
Qiskit	Single-Qubit	Rochester	DNF[16]						
Qiskit	CCNOT Chain	AQT Innsbruck	00:43.8	30:20.3	3				
Qiskit	Single-Qubit	AQT Innsbruck	02:53.0	2:50:15.7	3				

Join the Quantum Revolution

This course will help you develop practicable quantum software engineering skills and enable you to implement and analyze quantum algorithms

Get Started →

Origins

News room > News releases >

IBM Building First Universal Quantum Computers for Business and Science

- IBM unveils roadmap for commercial "IBM Q" quantum systems
- Releases API for developers to build interfaces between quantum computers and classical computers
- Following Watson and blockchain, quantum computing to deliver next powerful set of services on IBM Cloud platform

March 6, 2017

June 20, 2017

Introducing Forest 1.0

C Follow

Jun 2...

Today, I'm extremely excited to announce the public beta availability of Forest 1.0, the world's first full-stack programming and execution environment for quantum/classical computing.

Announcing the Microsoft Quantum Development Kit

December 11, 2017

Share

Microsoft Quantum

Microsoft Quantum Team

Just a few months back, Microsoft CEO Satya Nadella shared our vision of empowering the quantum revolution with bold investments towards a scalable end-to-end solution, revolutionary topological approach, and a global team. Today, we take the next step in this journey with the Microsoft Quantum Development Kit to help you get started with quantum development.

December 11, 2017

Initial Exploration

Quantum full-stack libraries

C++

- XACC - Extreme-scale programming model for quantum acceleration within high-performance computing.

JavaScript

- ~~Qiskit JS - Quantum information software kit for JavaScript (supported by IBM).~~

Python

- Cirq - Framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits.
- Forest - Rigetti's software library for writing, simulating, compiling and executing quantum programs.
- ~~Ocean - D-Wave System's suite of tools for solving hard problems with quantum computers.~~
- ProjectQ - Hardware-agnostic framework with compiler and simulator with emulation capabilities.
- Qiskit - Framework for working with noisy quantum computers at the level of pulses, circuits, and algorithms (supported by IBM).
- ~~Strawberry Fields - Xanadu's software library for photonic quantum computing.~~

Q#

- Q# - Microsoft's quantum programming language with Visual Studio integration.

List maintained by the Quantum Open Source Foundation: https://github.com/qosf/os_quantum_software

Initial Exploration

Quantum Software Framework Evaluation

This project contains the code written by MITRE during our evaluation of the most prominent quantum computing software frameworks in 2019. The evaluation is meant to gauge the software engineering experience in using each framework on a daily basis, to assess their strengths, weaknesses, and applicability to our work program.

Evaluation Criteria

Each framework is evaluated and scored according to these criteria:

- **Ease of Use** (learning curve, documentation, language features, control flow, development tooling, debugging support)
- **Maturity and Activity** (community size, frequency of updates, support, standard library functionality)
- **Flexibility and Modularity** (algorithm support, platform independence, compilation process, open source status)
- **Additional Criteria** (compiler and simulator performance, quantum error correction support, hardware tooling support, cost of access and training)

Source: https://github.com/jclapis/qsfe

Framework Analysis

Forest

```
# Hadamard the first three qubits - these will be the controls
for control in controls:
    program += H(control);

# pyQuil supports gates that are controlled by arbitrary many qubits, so
# we don't need to mess with Toffoli gates or custom multi-control implementations.
# We just have to chain a bunch of controlled() calls for each control qubit.
gate = X(target)
for control in controls:
    gate = gate.controlled(control)
program += gate

# Run the test
self.run_test("multi-controlled operation", program, controls + [target], 1000, valid_states)
```

```
operation MultiControl(Qubits : Qubit[]) : Unit
{
        let length = Length(Qubits);
        let controls = Qubits[0..length - 2];
        let target = Qubits[length - 1];

        // ApplyToEach is a helper function that basically runs the given operation
        // (in this case, the H gate) on each qubit in the given register.
        ApplyToEach(H, controls);

        // Controlled X means the controlled variant of the X gate, using the first
        // argument as the list of control qubits.
        Controlled X(controls, target);
}
```

Q#

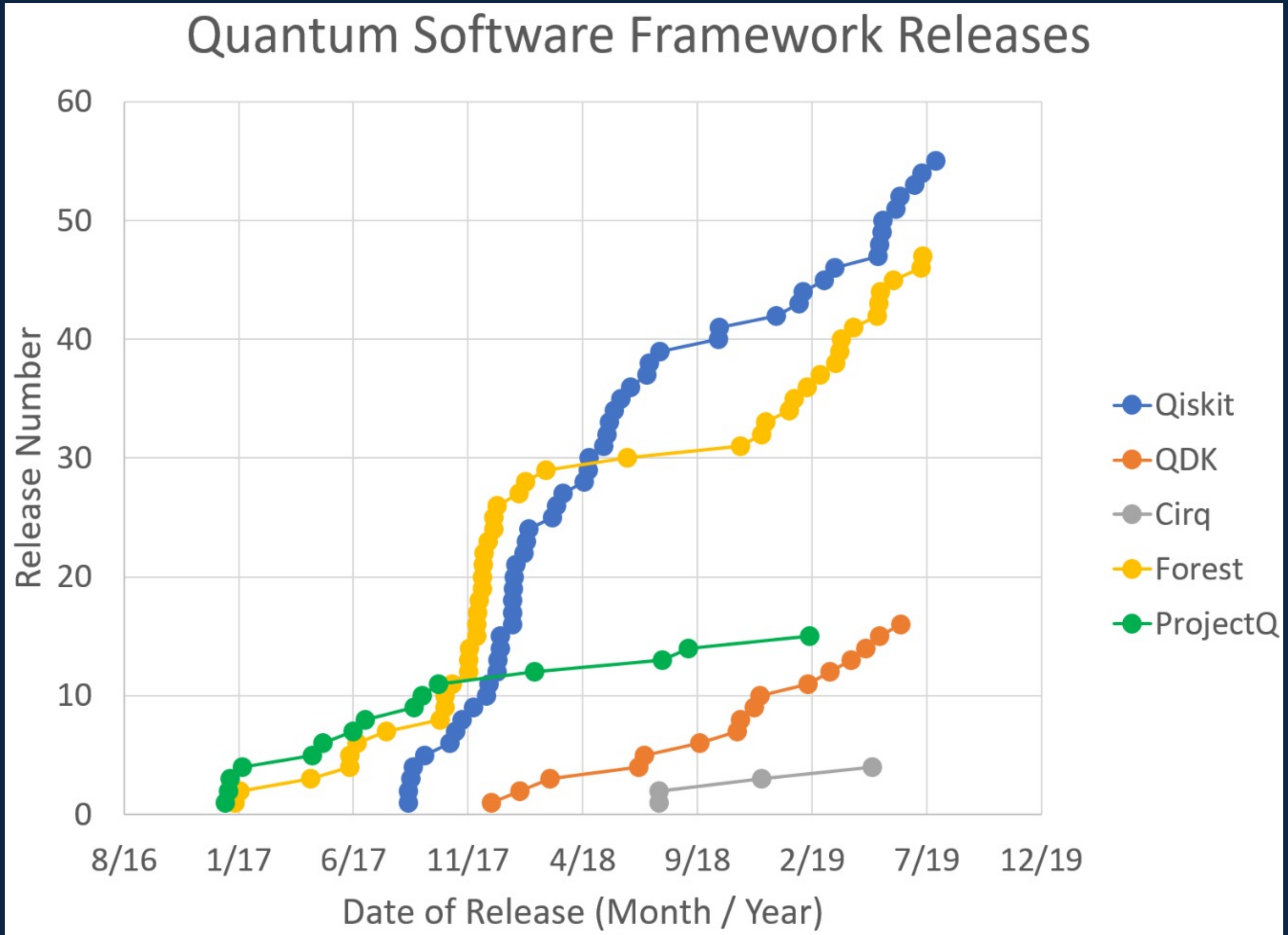
```
ancilla = QuantumRegister(1)
circuit.add_register(ancilla)
circuit.ccx(qubits[0], qubits[1], ancilla[0])
circuit.ccx(qubits[2], ancilla[0], qubits[3])
circuit.ccx(qubits[0], qubits[1], ancilla[0])

# Run the test
self.run_test("multi-controlled operation", circuit, 1000, valid_states)
```

Qiskit

MITRE

Framework Analysis



Evaluation Criterion	Qiskit	QDK	Cirq	Forest	ProjectQ	XACC
Installation Process	Excellent	Excellent	Excellent	Excellent	Excellent	None
Learning Curve	Excellent	Excellent	Good	Excellent	Good	Good
Background Theory Training Content	Good	Excellent	Poor	Fair	Poor	Poor
Documentation (Usage and Examples)	Good	Excellent	Fair	Excellent	Poor	Fair
Language Features	Fair	Excellent	Fair	Good	Excellent	?
Debugging Support	Good	Good	Good	Poor	Good	?
Community Size	Excellent	Good	Fair	Good	Poor	Poor
Frequency of Updates	Excellent	Good	Poor	Excellent	Poor	None
Support Availability	Excellent	Good	Fair	Excellent	Poor	Poor
Standard Library	Excellent	Excellent	Fair	Good	Fair	?
Control Flow	Good	Excellent	None	Good	Excellent	?
Quantum Hardware Platform Independence	Fair	N/A	Good	Poor	Excellent	Excellent
Open Source Status	Excellent	Excellent	Excellent	Excellent	Good	Good
Simulator Performance	Fair	Excellent	Good	Poor	Good	N/A
Noise Simulation and Error Correction	Excellent	Fair	Fair	Excellent	Poor	Good
Hardware Tooling Support	Excellent	Poor	Good	Good	Poor	?

Table 23. Summary of the frameworks' relative scores for each evaluation criterion.

Now that we know the tools…
What can we build with them?

Exploring Algorithms

Paper Name	Use Case	Speedup	Pub Link
A Quantum Pattern Recognition Method for Improving Pairwise Sequence Alignment	Aligning multiple sequences of data	Exponential	https://doi.org/10.1038/s41598-019-43697-3
Bayesian deep learning on a quantum computer	Machine Learning	Polynomial	https://doi.org/10.1007/s42484-019-00004-7
Quantum algorithm for visual tracking	Tracking an object through video frames	Quadratic	https://doi.org/10.1103/PhysRevA.99.022301
An improved quantum algorithm for ridge regression	ML, analyzing data suffering from "multicollinearity" with linear regression	Exponential / Polynomial, Depends on Input	https://doi.org/10.1109/TKDE.2019.2937491
~~Quantum image matching~~ (This paper is wrong, see next row)	Finding a smaller "target" image within a larger overall image	Quadratic?	https://doi.org/10.1007/s11128-016-1364-2
Analysis and improvement of the quantum image matching	Finding a smaller "target" image within a larger overall image	Quadratic	https://doi.org/10.1007/s11128-017-1723-7
FCM-based quantum artificial bee colony algorithm for image segmentation	Classifying pixels based on which region (e.g. foreground, background) they belong to	???	https://doi.org/10.1145/3240876.3240907
Image classification based on quantum K-Nearest-Neighbor algorithm	Image classification (selecting which of a pre-defined set of groups a picture belongs to)	Quadratic?	https://doi.org/10.1007/s11128-018-2004-9

MITRE

The "Paper Problem"

where the summation symbol denotes the sum over all possible positions.
system can be implemented by repeating the global unitary operator

$$\widehat{U} = \widehat{S}(\widehat{I} \otimes \widehat{C}),$$

where $\widehat{I}$ is the identity operator and $\widehat{C}$ is the coin operator appli
t steps is expressed by

$$|\psi\rangle_t = (\widehat{U})^t |\psi\rangle_0 = \sum_x \sum_v \lambda$$

and the probability of locating the walker at position x after t st

$$P(x, t) = \sum_{v \in \{0,1\}} |\langle x, v| (\widehat{U})^t$$

where $|\psi\rangle_{initial}$ is the initial state of the total quantum system.

(3) Add one qubit and rotate it from $|0\rangle$ to $\sqrt{1 - C_1^2 h^2(\pm\lambda_j, \alpha)}|0\rangle + C_1 h(\pm\lambda_j, \alpha)|1\rangle$ controlled on $|\frac{\pm\lambda_j}{N+M}\rangle$, where $h(\lambda, \alpha) := \frac{(N+M)\lambda}{\lambda^2+\alpha}$ and $C_1 = O\left(\max_{\lambda_j} h(\lambda_j, \alpha)\right)^{-1} = O(1/\kappa)$. As shown in appendix B, the maximum of $h(\lambda_j, \alpha)$ as well as C_1 depends on the actual choice of α, but $C_1 h(\lambda_j, \alpha) = \Omega(1/\kappa)$ for all possible α. Then we undo phase estimation and obtain

$$\sum_{j=1}^{R} \beta_j |\mathbf{u}_j, \pm\mathbf{v}_j\rangle \Big(\sqrt{1 - C_1^2 h^2(\pm\lambda_j, \alpha)}|0\rangle + C_1 h(\pm\lambda_j, \alpha)|1\rangle\Big). \quad (7$$

(4) Measure the last qubit to get $|1\rangle$ and project th
first register onto the $\mathbf{v}_j$ part. The final state of the firs
register approximates

$$|\phi_{\mathbf{w}}\rangle := \frac{\sum_{j=1}^{R} C_1 \beta_j h(\lambda_j, \alpha) |\mathbf{v}_j\rangle}{\sqrt{\sum_{j=1}^{R} C_1^2 \beta_j^2 h^2(\lambda_j, \alpha)}} \propto \mathbf{w}, \quad (8$$

2. Performing phase estimation of $e^{-i\tilde{X}t_0}$ on the first two registers, we obtain the whole state

$$\sum_{j=1}^{n} \beta_j \left(\frac{|\mathbf{w}_j^+\rangle |\lambda_j\rangle + |\mathbf{w}_j^-\rangle |-\lambda_j\rangle}{\sqrt{2}} \right) |0\rangle . \quad (11)$$

3. Performing a controlled rotation on the last register (qubit) conditioned on the eigenvalue register, we have

$$\sum_{j=1}^{n} \beta_j \Bigg(\frac{|\mathbf{w}_j^+\rangle |\lambda_j\rangle \left(\frac{C\lambda_j}{\lambda_j^2+\alpha} |1\rangle + \sqrt{1 - (\frac{C\lambda_j}{\lambda_j^2+\alpha})^2} |0\rangle \right)}{\sqrt{2}} + \frac{|\mathbf{w}_j^-\rangle |-\lambda_j\rangle \left(\frac{-C\lambda_j}{\lambda_j^2+\alpha} |1\rangle + \sqrt{1 - (-\frac{C\lambda_j}{\lambda_j^2+\alpha})^2} |0\rangle \right)}{\sqrt{2}} \Bigg) \quad (12)$$

Practicality Assessment

SCIENTIFIC REPORTS

OPEN

A Quantum Pattern Recognition Method for Improving Pairwise Sequence Alignment

Source: K. Prousalis and N. Konofaos, "A Quantum Pattern Recognition Method for Improving Pairwise Sequence Alignment," Scientific Reports, vol. 9, no. 7226, 2019, doi:10.1038/s41598-019-43697-3.

Practicality Assessment

	CUSHAW	SOAP2	Bowtie	BWA	QPR Method (by simulation)
Precision	90,00%	90,84%	93,47%	91,82%	94,24% (±1.48)
Recall	97,51%	92,85%	84,75%	97,22%	97,83% (±0.85)
Average time per read mapping (in *ms*)					
SE	1,154	2,044	1,343	3,721	0,00642
PE	2,898	2,310	3,013	3,929	
Approximated runtime (in *s*)					
SE	1026	2617	1085	4764	28,84
PE	3711	2958	3858	5031	

MITRE

Practicality Assessment

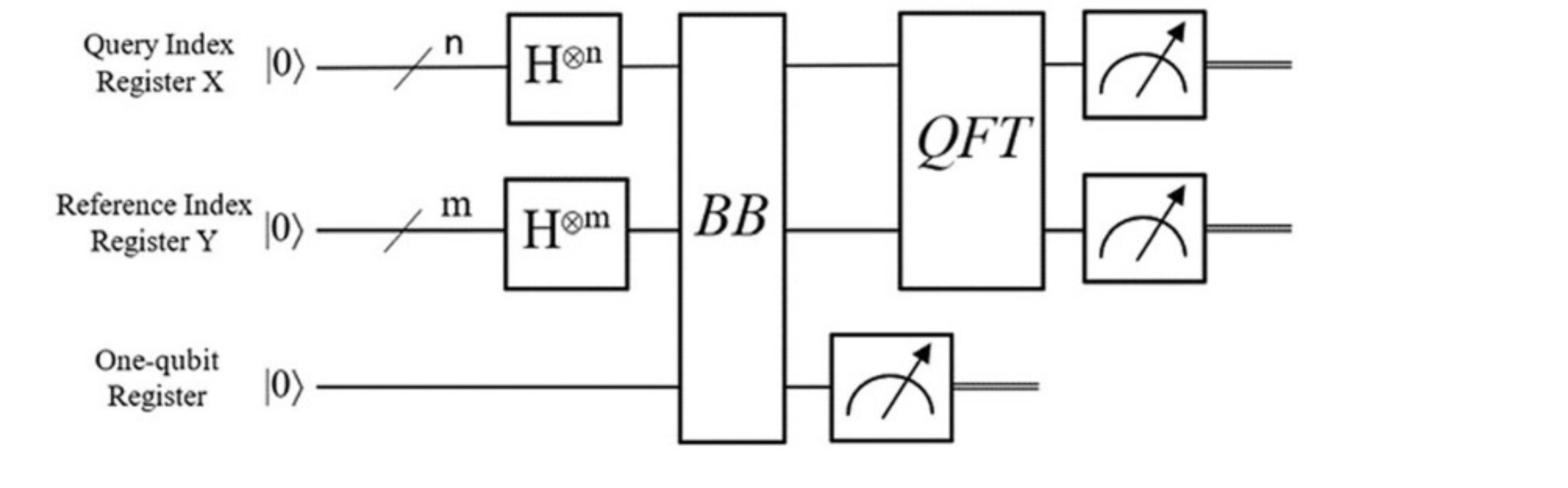


Figure 1. Circuit model of R. Schützhold's quantum pattern recognition method.

It is assumed that our aligner is a complex device which can handle large sequences on a lattice plane surface made by non-linear Kerr media and run R. Schützhold's QPR algorithm in a quantum computing system. The

Practicality Assessment

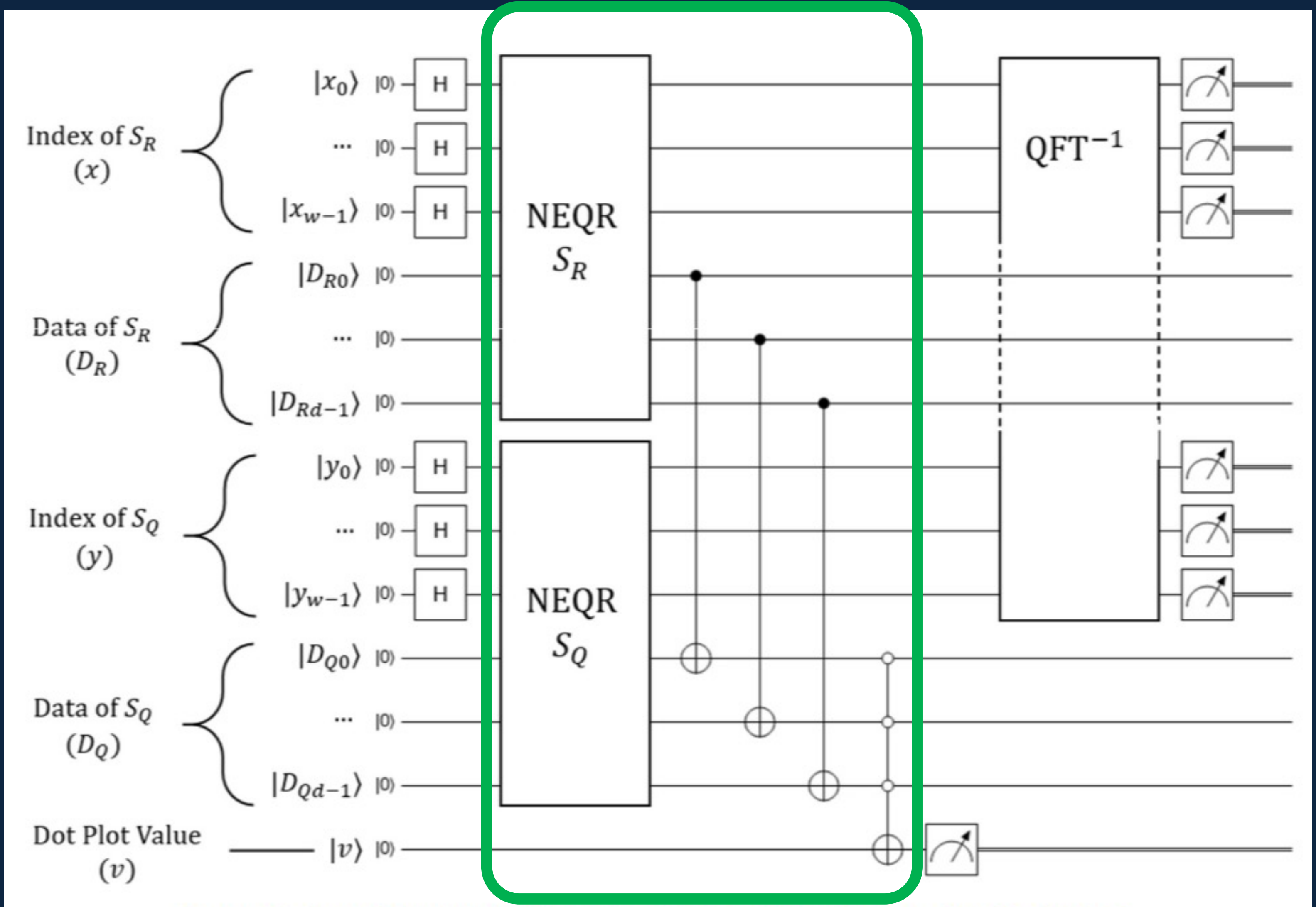


Figure 6. Complete circuit diagram of QPR, using QDP as the Black Box.

Source: J. Clapis, "A Quantum Dot Plot Generation Algorithm for Pairwise Sequence Alignment," 2021, arxiv: 2107.11346

Practicality Assessment

SDK	MCX Mode	Backend	Circuit Creation Time	Compile Time	Width	NEQR Depth	QDP Depth	QFT Depth	Total Depth
QDK	---	Resource Counter	00:01.8	---	39	150,239	5	178	150,417
Qiskit	CCNOT Chain	Aer Sim	00:45.2	05:07.3	39	384,312	4	49	384,350
Qiskit	Single-Qubit	Aer Sim	03:54.6	18:38.7	30	1,940,132	4	49	1,940,173
Qiskit	CCNOT Chain	Rochester	00:53.9	4:34:26.7	53	1,027,772	20	1108	1,028,758
Qiskit	Single-Qubit	Rochester	DNF[16]						
Qiskit	CCNOT Chain	AQT Innsbruck	00:43.8	30:20.3	39	761,606	34	506	762,002
Qiskit	Single-Qubit	AQT Innsbruck	02:53.0	2:50:15.7	30	9,524,360	34	506	9,524,819

Table 5. Resource estimates for dataset 3 (two 4096-element sequences).

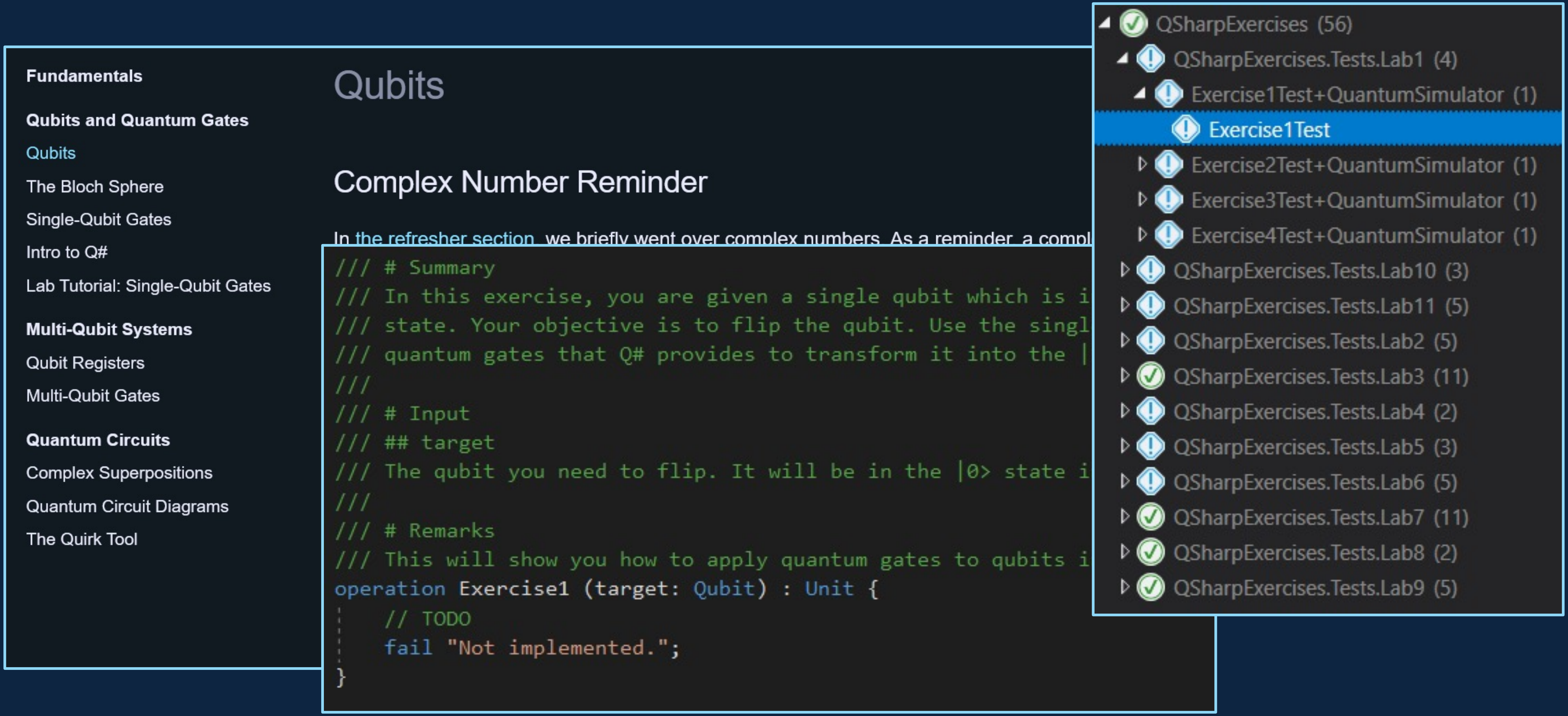
1. Understanding the quantum paradigm is difficult for conventional engineers

2. Deriving code from theory tends to require significant experience

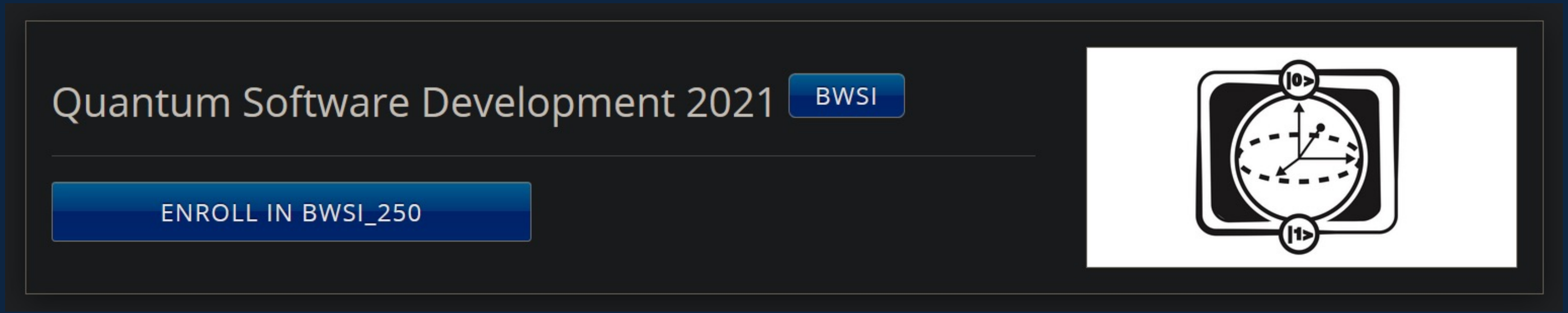
Growing the Workforce

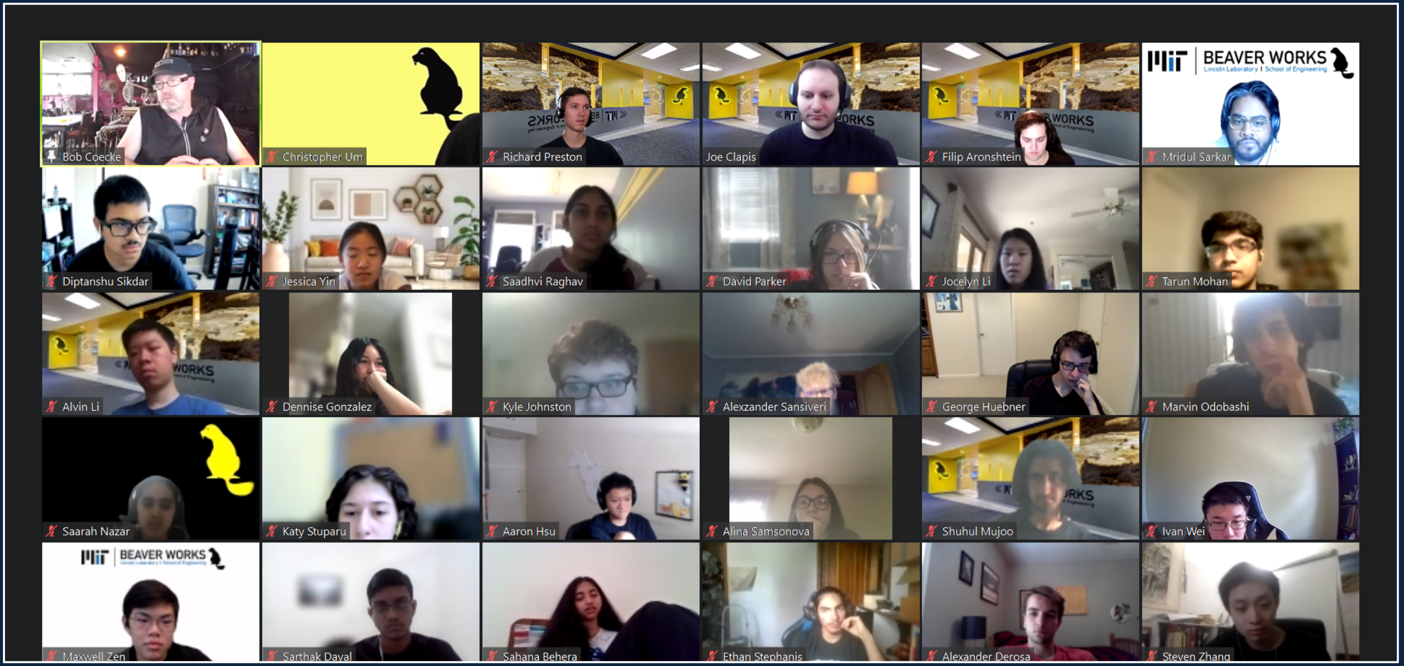
Background Math	Classical Computing	Qubits and Quantum Gates	Multi-Qubit Systems	Quantum Circuits	Quantum Protocols	Quantum Algorithms	Quantum Error Correction	Execution on Quantum Computers
Complex Numbers	Digital Information	Qubits	Qubit Registers	Complex Superpositions	Quantum Interference	Deutsch-Jozsa Algorithm	Bit-Flip Error Correction	Intro to Qiskit
Vectors	Endianness	The Bloch Sphere	Multi-Qubit Gates	Quantum Circuit Diagrams	Superdense Coding	Simon's Algorithm	Steane ECC	Cloud-Based Machines
Matrices	Digital Logic	Single-Qubit Gates		The Quirk Tool		Grover's Algorithm		Resource Estimation and Practicality Assessment
Bra-ket and Tensor Notation	Low-level Programming	Intro to Q#				Quantum Fourier Transform		Closing Thoughts and Next Steps
	High-level Programming	Lab Tutorial: Single-Qubit Gates				Shor's Algorithm		
	Visual Studio							

Growing the Workforce



Growing the Workforce





"The way all of [the instructors and TAs] interacted with the class made it feel like I was a part of the group and not being taught down to…

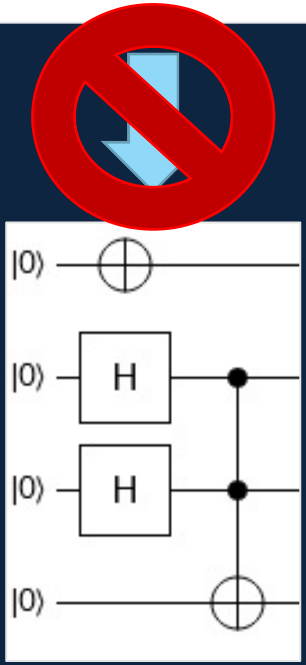
[I] now hope my future career lies in quantum computing!!"

MITRE

Bridging the Gap

$$|\psi\rangle = |1\rangle \otimes \sum \sum \frac{1}{\sqrt{n}} (|ij\rangle \otimes |i \cdot j\rangle)$$

Authors?
Engineers?



Software
Engineers

```
operation PrepareState(Qubits : Qubit[]) : Unit
{
    X(Qubits[0]);
    ApplyToEach(H, Qubits[1..2]);
    Controlled X(Qubits[1..2], Qubits[3]);
}
```

```
from pyquil.quil import Pragma, Program
from pyquil.api import get_qc
from pyquil.gates import *

qc = get_qc("9q-square-qvm")

program = Program()
program += X(0)
program += H(1)
program += H(2)
program += X(3).controlled(2).controlled(1)

ep = qc.compile(program)

print(ep.program) # here ep is of type Pyqui
```

Compilers

```
X 0
H 1
H 2
H 3
CNOT 1 3
T_ADJ 3
CNOT 1 2
…
```

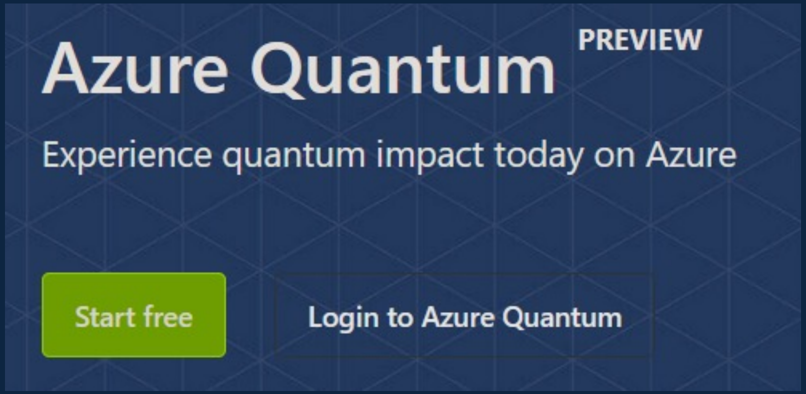
```
RZ(pi/2) 2
RX(pi/2) 2
RZ(2.510564325593089) 2
RX(pi/2) 2
RZ(-3*pi/4) 3
RX(pi/2) 3
RZ(0.9553166181245095) 3
…
```

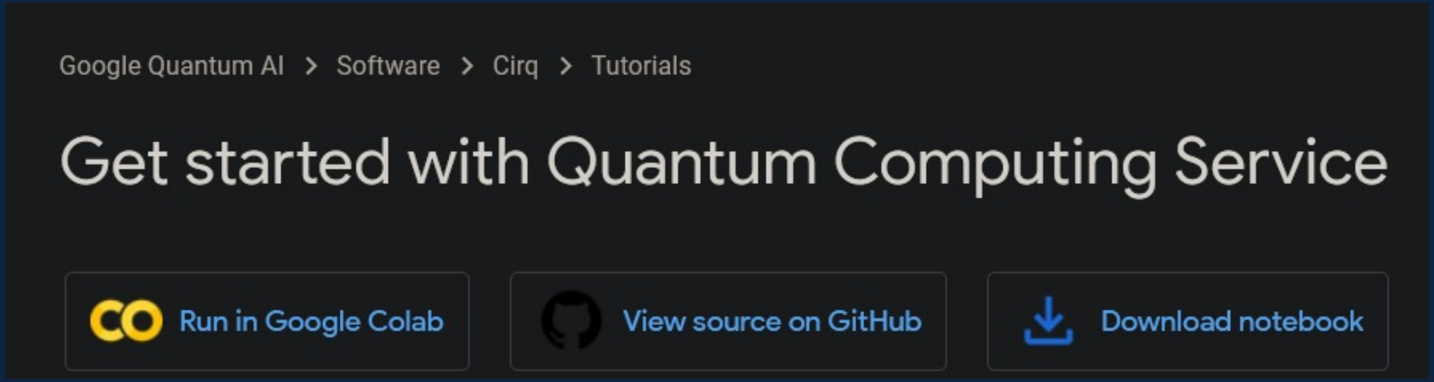
MITRE

Lessons Learned

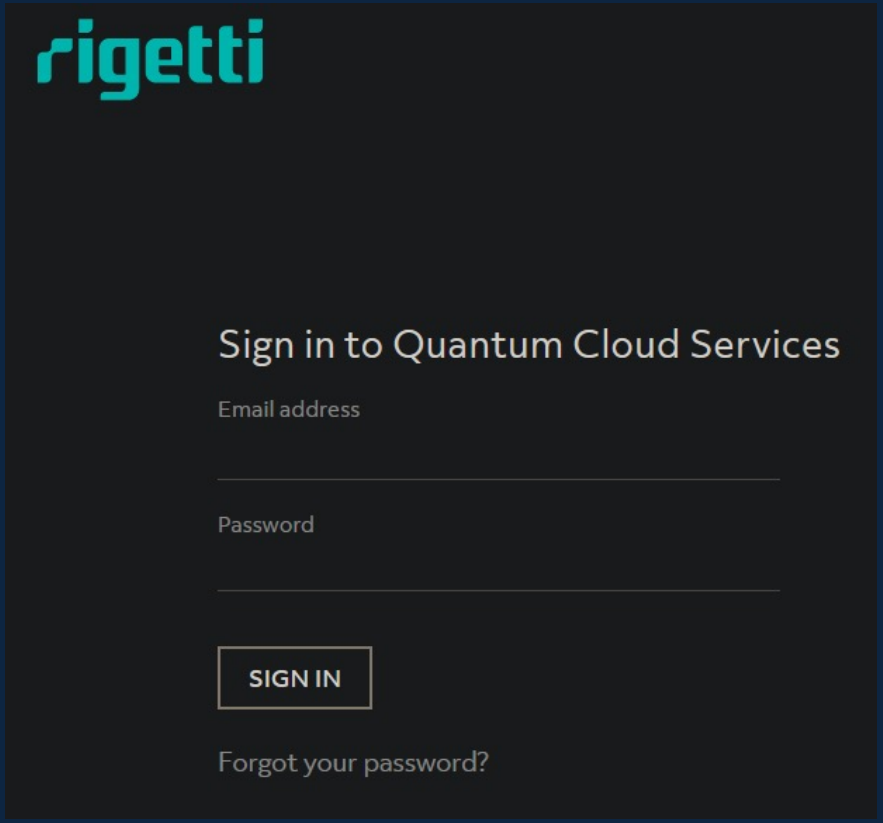
1. Use the right tool for the job.
2. Understand why a particular algorithm is useful in the first place.
3. Look for assumptions, omissions, and abstractions that prevent you from building and testing new algorithms.
4. Connect with experts that can provide access to hardware.
5. Help your colleagues when they get stuck.

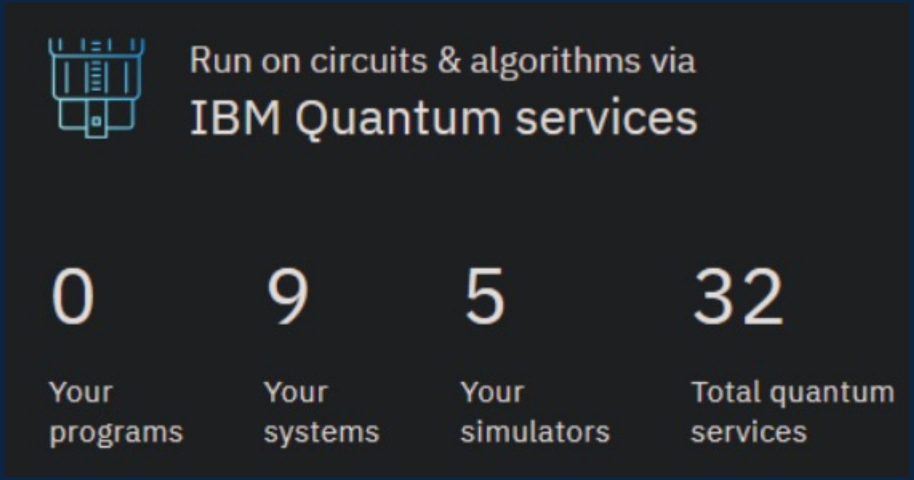
The Future


Azure Quantum
PREVIEW
Experience quantum impact today on Azure
Start free
Login to Azure Quantum


Google Quantum AI > Software > Cirq > Tutorials
Get started with Quantum Computing Service
Run in Google Colab
View source on GitHub
Download notebook


Amazon Braket
Get started with quantum computing


rigetti
Sign in to Quantum Cloud Services
Email address
Password
SIGN IN
Forgot your password?


Run on circuits & algorithms via
IBM Quantum services
0
Your programs
9
Your systems
5
Your simulators
32
Total quantum services

The Future

qRAM Library for Q#

Supported By UNITARY FUND

This library implements a variety of different proposals for memory for quantum computers, also commonly called qRAM.

> Want to learn more about what qRAM is? Check out the primer on memory for quantum computers in our docs!

Source: https://github.com/qsharp-community/qram

Quantum Algorithm Zoo

This is a comprehensive catalog of quantum algorithms. If you notice any errors or omissions, please email me at stephen.jordan@microsoft.com. (Alternatively, you may submit a pull request to the repository on github.) Your help is appreciated and will be acknowledged.

Source: https://quantumalgorithmzoo.org/

MITRE

Joe Clapis

jclapis@mitre.org

https://github.com/jclapis/

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™