Hi!

Once again, thank you for your interest in the QC Mentorship program!

We decided to select participants based on how they will manage to do some simple “screening tasks”.

These tasks have been designed to:
- find out if you have the skills necessary to succeed in our program.
- be doable with basic QC knowledge - nothing should be too hard for you to quickly learn.
- allow you to learn some interesting concepts of QC.
- give you some choices depending on your interests.

What we mean by skills is not knowledge and expertise in QC. It’s the ability to code, learn new concepts and to meet deadlines.

What are we looking for in these applications?
- Coding skills – clear, readable, well-structured code
- Communication – well-described results, easy to understand, tidy.
- Reliability – submitted on time, all the points from the task description are met
- Research skills – asking good questions and answering them methodically

Also, feel free to be creative – once you finish the basic version of the task, you can expand it.

Choose tasks based on your interests, don’t try to pick the easiest one.

You need to do only 1 task. Feel free to do all of them, it might be a good learning opportunity, but it won’t affect admissions to the program :)

So here are the tasks:

Task 1 Statevector simulation of quantum circuits

For this task, you will implement a statevector simulator for quantum circuits from scratch. The goal is to demystify how to simulate a quantum computer and to demonstrate your familiarity with quantum circuits.

1) Naive simulation using matrix multiplication

Remember that [1, 0] = |0> is the most common representation of the single-qubit zero state, and analogously [0, 1] = |1>.

Most matrix representations of quantum gates you can find online follow this convention. For example, the X gate can be written as

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Using the Kronecker product and the *np.kron* function in *numpy* (we are using it as an example, but you can use any library you want to), you can create a vector of length 2^n representing an n-qubit quantum state, and matrix representation of X, H, and CNOT gates.

Hint: The single-qubit Identity matrix is

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Define a quantum circuit consisting of these gates and apply the gates sequentially to the statevector via matrix multiplication.

Plot the runtime of your code as a function of the number of qubits. How many qubits can you simulate this way?

2) Advanced simulation using tensor multiplication

Tensors are generalizations of vectors and matrices to any number of dimensions. Instead of representing an n-qubit state as a vector of length 2^n, it may be more natural to write it as an n-dimensional tensor of shape (2, 2, …., 2). The transformations between these two representations are naturally possible via *np.reshape* and *np.flatten*.

Using tensor multiplication and the *np.tensordot* (or *np.einsum*) function, you can apply a gate to the quantum state by multiplying the 1- or 2-qubit matrices with the state tensor along the corresponding qubit axes.

Define a quantum circuit consisting of the 1- and 2-qubit matrix representations of X, H, CNOT (same as above) and apply them sequentially to the quantum state tensor via tensor multiplication.

Plot the runtime of your code as a function of the number of qubits. How many qubits can you simulate this way? Compare your results to subtask 1).

3) Bonus question

1. How would you sample from the final states in the statevector or tensor representations?
2. And how about computing exact expectation values in the form <Ψ| Op |Ψ>?

Task 2 Noise, Noise, and More Noise

One of the main challenges in quantum computing is the noise in current devices. In this task, you will create a simple noise generator and assess its effect. You can use any framework you like (Qiskit, Cirq, etc..)

1. Noise Model

A standard way to represent the noise in a quantum circuit is through Pauli operators (σ_x, σ_y, σ_z). Build a function with input α, β and QuantumCircuit where:

- $\alpha \rightarrow$ Probability of having a random Pauli operator acting on the qubit after a one-qubit gate
- $\beta \rightarrow$ Probability of having a random Pauli operator acting on the qubit after a two-qubit gate
- QuantumCircuit → Quantum circuit where the noise will be added

The output should be the Quantum Circuit with Noise

2. Gate Basis

Quantum computers can implement only a set of gates that, with transformations, can represent any other possible gate. This set of gates is called the Gate Basis of the QPU. Build a function that transforms a general Quantum Circuit to the following gate basis: {CX,ID,RZ,SX,X}

3. Adding two numbers with a quantum computer

Build a function (quantum_sum) to add two numbers using the Draper adder algorithm. You will need the Quantum Fourier Transform (QFT). Many libraries offer a function to use it. For this task, you will need to build QFT from scratch.

4. Effects of noise on quantum addition

Now, we can combine all the functions. Transform the circuit used in the quantum_sum to the gate basis and add noise. Use different levels of noise and analyze the results.

- How does the noise affect the results?
- Is there a way to decrease the effect of noise?
- How does the number of gates used affect the results?

References:

[3] Quantum arithmetic with the Quantum Fourier Transform
https://arxiv.org/pdf/1411.5949.pdf

Task 3 Optimization

In the Bin Packing Problem (BPP), given a collection of items, the goal is to efficiently pack the items into the minimum number of bins, where each item has an associated weight, and the bins have a maximum weight. The problem can be found in different industries. For example, the supply chain management industry requires loading multiple packages onto a truck, plane, or vessel. In this task, you will solve the BPP using quantum computing.

1. From Integer Linear Programming (ILP) to Quadratic Unconstrained Binary Optimization (QUBO)

- Define the ILP formulation of the BPP. You can use Docplex or similar frameworks to do it.
- Create a function to transform the ILP model into a QUBO
- Test your function with specific instances (size small, medium, and big)

2. Create a Brute Force solver for the QUBO problem and solve the specific instances.

3. To solve the QUBO, use quantum annealing simulators. You can use the Dwave Ocean Framework. Here is an example.

4. Use a Quantum Variational approach to solve the QUBO.

- Create multiple Ansantz for tests.
- Build a function with input being the QUBO and Ansantz. Using a hybrid approach solved the QUBO.

5. Use QAOA to solve the QUBO.

- Create from scratch a QAOA function.

6. Compare and analyze the results.

- What is the difference between QAOA, Quantum Annealing, and Quantum Variational approaches with different Ansatz?
- How do the results compare with the brute force approach?

Task 4 Decomposition

A common challenge in the design and implementation of quantum circuits is that they become too extensive and complex due to the large number of qubits required.

The number of quantum operations and the interdependence between qubits can further complicate the problem, making some traditional optimization methods ineffective or insufficient for reducing the complexity. Additionally, as the number of qubits increases, the fidelity of the circuit may be affected, which implies a greater need for advanced techniques to maintain the precision of operations.

On the other hand, some quantum computing frameworks offer predefined optimization methods that may seem like a convenient option. However, they are not always the best alternative if one has a deep understanding of the quantum hardware structure. Customizing the circuits by taking advantage of the specific features of the quantum device in use can result in a more efficient design, better adapted to the physical limitations of the system. Therefore, having a solid understanding of the hardware can make the difference between a generic quantum circuit and one that is highly optimized for performance.

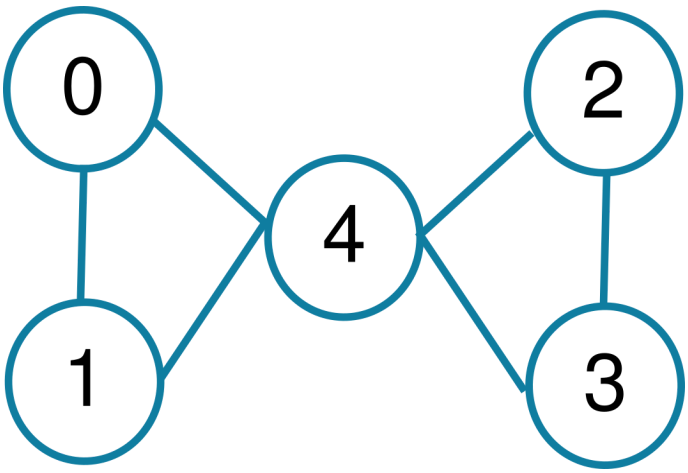
Challenge:

Consider the following code as input

```
import numpy as np

size = 5

state_values = [22,17,27,12]

state_vector = [0]*2**size

for s in state_values:
```

```
   print(np.binary_repr(s,size))

   state_vector[s] = 0.5

np.asarray(state_vector)
```

Find a quantum circuit that represents the state vector with a depth less than 50, using the basis_gates=[x,h,rz,cx] and the following architecture



Deadline
2 weeks from when you've submitted your application in your timezone.

Once you have finished a screening task, please submit your GitHub repository containing the code to this google form: https://forms.gle/zJdmwCT9115Hq5cu7 -- other forms of submission will not be accepted!

If you have any questions - please add comments to this document, or ask it in the QOSF slack workspace (invitation_link) in the #mentorship-applicants channel. We will be updating this document with more details and/FAQ to avoid confusion, so make sure to check it before asking :)

Have a nice day!

QOSF team

FAQ

Q: Can we use any quantum libraries or are we restricted to a particular set of tools?

A: Feel free to use whatever you like, just make sure that the tool doesn't solve the whole problem for you.
Regarding the language of choice, Python is definitely the preferred one, since this is the language that most of the mentors use.
You can do the task first in the language of your preference and then translate it to Python if that's more convenient for you.

Q: I am applying as a member of a team. How many tasks do we submit?

A: Each member of a team must submit their own screening task. This will help us judge the skill level of each individual team member and help us pair folks up with the right mentor.

Q: How should I submit the solution?

A: All the materials for the submission should be inside a GitHub repository. Please do not send us any loose files as attachments or in any other format. Please submit your GitHub repository to this google form once you've finished: https://forms.gle/C2KWaaTRXwWsjJbN7

Q: My team-mate wants to leave the team because he/she/they can't manage these along with exams. So will this affect our team status or anything like that?

A: Well, just let us know and you can continue as an individual/smaller team.

Q: Is it possible to make more than one task and send everything together?
A: Yes, you can. But you should specify which task you want to be evaluated. In other words, do it as an exercise but it does not affect your chances to enter the program.