CIS 492/593 Hands-On Lab 2 Qiskit and Entanglement Fall 2023

In this hands-on lab, you will learn:

- Qiskit Terra and Qiskit Aer. The former is a module which handles quantum circuit construction, circuit analysis, and transformation, while the latter is a high performance simulator for quantum circuits.
- Hadamard gate
- two-qubit entanglement

Login to a workstation and open a terminal (either middle click the terminal icon on the left side bar or press CTRL-ALT-T). In the terminal window, type

```
cd QC
jupyter notebook
```

Inside the browser, click "New" and choose "Python 3 (ipykernel)". For each experiment, you need to put the experiment number as a comment (e.g. `# Experiment 1`) in the top of the code.

Experiment 1: Qiskit Programming

- In the cell, type

  `%load Hgate.py`

  and click "Run" to load the sample Qiskit program into the cell and then put the experiment number as a comment.

  Below is a brief description of the program. The code imports several modules at beginning. Then, it invokes the constructor `QuantumCircuit(1,1)` *to create a quantum circuit* `qc` *which contains 1 qubit and 1 classical bit. Next, we apply an H gate (i.e.* `qc.h(0)` *to the qubit 0. Note that the qubit is allocated using the array format and the first qubit is indexed with 0. Same for the classical bit. Finally, we use* `qc.measure(0,0)` *to measure the qubit 0 and put the result to the classical bit 0.*
  After building the circuit, the program calls the function `transpile()` *to compile the abstract circuit* `qc` *to low-level instructions and then uses the Aer simulator to execute the instructions 1000 times.*

- When calling `simulator.run()`, change `"shots=1000"` to `"shots=10000"` and also add

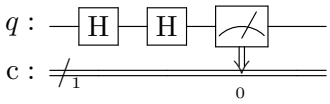
  `plot_histogram(counts)`

  at the end of the program. Click "Run" to execute the code in the cell. Record the numbers for `'0'` and `'1'`.
- Click the cell and click "Run" again. Do you get EXACTLY the same numbers for `'0'` and `'1'` as in the previous run? Explain your result. Moreover, what is the data structure of the variable `counts`?

Experiment 2: Apply two H gates in succession

If we consider the H gate to be analogous to a coin toss, two H gates in succession should be tossing the coin twice. We would still expect close to a 50/50 distribution of zeros and ones for the second flipping. However, quantum randomness is not simply like a classical random coin toss. Let's do the following experiment.

- Click the cell which contains the code in Experiment 1. Click the "Edit" button and select "Copy Cells". Click "Edit" again and choose "Paste Cells Below".
- Apply the H gates twice in a row, like the diagram below:



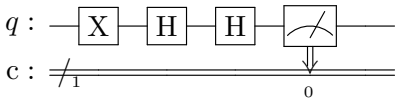
That is, you can add another `qc.h(0)` immediately after `qc.h(0)` in the code.
- Click "Run" to see and record the result. To verify, calculate by hand the state vector after applying the first H gate in your hard-copy lab report. Note that the qubit is initialized to be 0. *Hint:*

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} \\ \\ \end{bmatrix} =$$

Then, calculate the state vector after applying the second H gate.

Experiment 3: Apply two H gates in succession with initial value 1

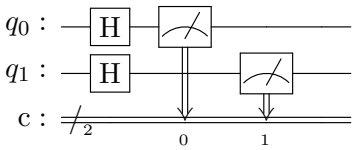
- Click the cell above (i.e. two H gates in succession). Click the "Edit" button and select "Copy Cells". Click "Edit" again and choose "Paste Cells Below".
- Instead of using the default value 0 for the qubit, we would like to initialize the qubit to 1 before applying the two qubits. To achieve this, you can add an X gate (i.e. the NOT gate) by using `qc.x(0)` before the first calling of `qc.h(0)`, like below:



- Click "Run" to see and record the result. Again, calculate by hand the state vector after applying the first H gate in your report. Then, calculate the state vector after applying the second H gate.

Experiment 4: Apply an H gate for each of two parallel qubits

- Click the cell which contains the code done in Experiment 1. Click the "Edit" button and select "Copy Cells". Click "Edit" again and choose "Paste Cells Below".
- Modify the code in the cell to build the circuit below:



Hint: The circuit needs 2 qubits and 2 classical bits when calling `QuantumCircuit()`. You also need to add a H gate on qubit 1. Lastly, use `qc.measure([0,1],[0,1])` to map the qubits 0 and 1 to the classical bits 0 and 1, respectively.
- Click "Run" to get and record the result. Explain.

- What is the initial state vector?

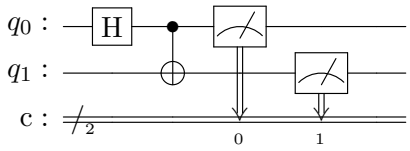
$$q_0 \otimes q_1 = \begin{bmatrix} \\ \end{bmatrix} \otimes \begin{bmatrix} \\ \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

Calculate the state vector after applying the parallel H gates. *Hint:*

$$\frac{1}{2} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} \\ \\ \\ \end{bmatrix} =$$

Experiment 5: Entangle two qubits

- Click the cell which contains the code done in Experiment 4. Click the "Edit" button and select "Copy Cells". Click "Edit" again and choose "Paste Cells Below".
- Modify the code in the cell to build the circuit below:



Hint: Replace the H gate on qubit 1 with a CNOT gate using qubit 0 as the control qubit and qubit 1 as the target qubit (i.e. `qc.cx(0,1)`).
- Click "Run" to get and record the result. Explain.
- To save the code in this cell into a file `entangle2.py`, put the following

```
%%writefile entangle2.py
```

as the FIRST line of the code. Click "Run".
- To check the file `entangle2.py` has been saved correctly, open a terminal and type

```
cd QC
cat entangle2.py
python3 entangle2.py
```

Experiment 6: Are they entangled?

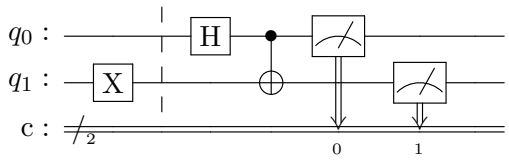
- Click the cell which contains the code done in Experiment 5. Click the "Edit" button and select "Copy Cells". Click "Edit" again and choose "Paste Cells Below". Remove the first line

```
%%writefile entangle2.py
```

- Add the following two lines of code

```
qc.x(1)
qc.barrier()
```

immediately after calling the function `QuantumCircuit()`, just like the diagram below:



The purpose of using the X gate on qubit 1 is to invert its initial value from 0 to 1. The `barrier()` (i.e. a dashed line in the circuit diagram) is a directive which tells the circuit transpiler not to optimize the code crossing the barrier separation line. Here we just use it for more readability.

- Click "Run" to see and record the result. Are these two qubits entangled? Explain. *Hint: See Textbook Page 59*
- Calculate the state vector (i.e. the tensor product of the two qubits) before applying the CNOT gate.

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} \quad \end{bmatrix} = \begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix}$$

Calculate the state vector after applying the CNOT gate. *Hint: similar calculation can be found on Textbook Page 67.*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix} =$$

Turnin

Click "File" and choose "Save as". Type "lab2" in the entry box and click "Save" to save your work today into the file `lab2.ipynb`. To turn in your file, use CTRL-ALT-T to open a terminal and type

```
ssh grail
```

and type your password to login to the server grail. Then, type

```
cd QC
turnin -c cis492s -p lab2 lab2.ipynb
```

to electronically submit your file `lab2.ipynb`.

If you finish early, you can start doing the "IBM QC Homework" posted on the CSU Blackboard to run Qiskit programs on a real IBM quantum computer.

Shutdown the jupyter notebook and logout the workstation.

Hand in your lab report before your leave.