CSCE 785, Fall 2024, Homework 2
due Monday September 23

Written Exercises

Course notes Exercises 1.33, 1.34, 1.39, 1.41, 1.43.
Course notes Exercises 2.2, 2.3, 2.4, 2.6, 2.9, 2.14(2,3,4,8).

Programming Exercises

These exercises are to get your feet wet with Qiskit and IonQ—essentially “hello world” programs—and to work out any kinks in the process. Some of these exercises are just following directions on IonQ’s website with my own comments added. Qiskit is a software development kit (SDK), an extension of Python developed by IBM and probably the most popular high-level SDK for programming a QPU (Quantum Processing Unit.) There are other SDKs, including Cirq (Google), Braket (Amazon), and Strawberry Fields (Xanadu), but we will stick with Qiskit.

Unlike IBM, which supports superconducting QPUs, IonQ supports trapped ion QPUs. Each has its own advantages and disadvantages, but we will generally not dive into hardware peculiarities, but program at a higher, device-independent level.[1]

It is assumed that you have accepted an email invitation to get an IonQ account and that you have been added to the CSCE 785 project. If you have not yet completed the form found at `https://cse.sc.edu/resources/scquantum`, **do so immediately!**

For these instructions, I am assuming you are using a linux-like operating system, such as Ubuntu or MacOS. All of this can surely be done on MS Windows, but then you are on your own. Not having Windows myself, I can’t guarantee what works and what doesn’t. YMMV.

Make sure you have Python installed and up to date. You need at least version 3.11 or later, I believe (but why not have the latest?). Then follow these steps (thanks to Ryan Austin for helping with this):

1. Set up a directory structure and virtual environment:

[1]For the record, IBM’s devices support more qubits, but direct connections between their qubits are more limited than with IonQ’s devices.

```
mkdir ionq
cd ionq
python3 -m pip install virtualenv
python3 -m virtualenv venv
```

2. You will need to create an API key, and you must log into your account to do this (see Step 2 under `Quickstart` at `https://docs.ionq.com/`). Once you have it, add it to the virtual environment activation file:

   ```
   vim ./venv/bin/activate
   # Insert the following lines with your API key from the IonQ website:
   export IONQ_API_KEY="REPLACE-ME"
   export QISKIT_IONQ_API_TOKEN="REPLACE-ME"
   ```

   Use the same string for both environment variables. Alternatively, you can include it explicitly in every program you submit. Don't share your key with anyone (but submitting source code that includes it is ok).

3. Activate your python virtual environment and install dependencies:

   ```
   source ./venv/bin/activate
   pip install qiskit
   pip install qiskit-ibm-runtime
   pip install qiskit-aer
   pip install qiskit-ionq
   ```

Some of the above is also explained at `https://docs.quantum.ibm.com/guides/install-qiskit`. By the way, to get a list of all packages installed in your virtual environment with their versions, type `pip list`. To get out of your virtual environment, just type `deactivate`.

Programming Exercise 1

This exercise is to get you familiar with IonQ's text-only interface. It is relatively simple to set up. Do this in your virtual environment. If you've inserted definitions of your API key in your `activate` file, then you can do all of this without logging into your account.

1. Go to `https://ionq.com`.

2. Under the `Resources` tab, click `Documentation`, and scroll down to the `Quickstart` section.

3. Go through Step 3 up until the `Learning more` section in the `Learn how to use Qiskit` link. You can skip a lot at the beginning about the API key, the environment, and installing packages, because you've already done it. Start where it says, "`Start a script`."

4. Copy the snippets of Python code into python scripts and run them from the command line. Feel free to repeat runs of your programs to fix errors. This is best done on the simulator before submitting to a QPU. Once you are happy with your program, run it on whatever QPU they recommend (e.g., "aria-1"). When I did this, the job took about 5 minutes to run and produced output in a text-only format similar to the simulator.

5. Submit the final programs you run and their results as part of Homework 2.

Programming Exercise 2

This exercise uses a more sophisticated GUI (jupyter) to interact with IonQ. You'll be able to visualize the circuits you build and see the results graphically.

1. Working from your `ionq` directory, grab the `qiskit-getting-started` directory from Github:

   ```
   git clone https://github.com/ionq-samples/qiskit-getting-started.git
   ```

2. Install these in your virtual environment:

   ```
   pip install 'qiskit[visualization]'
   pip install jupyter
   ```

3. Staying in your virtual environment, fire up jupyter:

   ```
   jupyter notebook qiskit-getting-started &
   ```

   You should get a list (maybe as a tab in your browser) of files in the `qiskit-getting-started` directory.[2] Click on `bell-cat-states.ipynb` and then the `Open` button.

4. Read the `Setup` and do anything there that you haven't already done, such as installing `matplotlib` with `pip`. (You may have it already installed as a dependency of something else, but it doesn't hurt to try installing it again.)

5. The first cell with python code is out of date and needs editing. Change the line

   ```
   from qiskit import Aer
   ```

   to

   ```
   from qiskit_aer import Aer
   ```

[2] I had some problems with using Firefox on my (old) Macbook Air. The tab would open, but the page was completely blank. I had no problems using Chrome on the same machine, however.

If you included a definition of `QISKIT_IONQ_API_TOKEN` in your virtual envirnoment's `activate` file as described above, then change the line

```
provider = IonQProvider(token='My token')
```

to

```
provider = IonQProvider()
```

(If you didn't do this, then set the `token` argument to your API key, surrounded by single quotes).

6. Select the cell with the code you just altered (by clicking in the left margin), and hit the RUN icon (the triangle) to execute that code. If there are any errors, they will show up in pink below the cell. Otherwise, no news is good news.

7. Optionally select and run the next code cell to verify the two backends available (simulator and QPU).

8. The next input cell runs a program to build a simple quantum circuit, which is displayed below it. I suggest trying a few tweaks, such as changing the line `qc.h(0)` to `qc.h(1)` then re-running the code to see how the circuit changes. Also try swapping the two arguments to `qc.cx` and see what changes. Whatever you do, return the code to its original text afterwards.

9. Run the next cell to execute the circuit on IonQ's simulator. By default the simulation is noise-free, that is, there is no simulated noise model applied to the gates. The circuit is run 1000 times (1000 "shots"). Running the subsequent cell reveals a histogram showing the statistical results of the measurements. The circuit produces a quantum state called a *Bell state* where the qubit values are perfectly correlated (either 00 or 11) but each occurs with equal probability, as approximated by the histogram.

10. Run the next cell to execute the same circuit on an actual QPU. If you get a long error message that at the bottom says that harmony is retired and no longer accepting jobs, then you'll need to choose one of their other QPUs. Go back and change the line

    ```
    qpu_backend = provider.get_backend("ionq_qpu")
    ```

    to

    ```
    qpu_backend = provider.get_backend("ionq_qpu.aria-1")
    ```

    Then re-run. Other options include `aria-2` and `forte-1`.

11. You can run the next cell occasionally to check on the job status. When I did this, the job took less than 20 minutes to complete. (It may have taken a lot less; I went and did other things and only came back to check after 20 minutes, finding the job done.) There are other ways to check the status, as you saw going through Project 1.

12. Run the next cell to plot the histogram of results. You will most likely see some anticorrelation results (01 and/or 10) but with low frequency counts. This reflects the actual noise in the circuit.

13. Go through the rest of the notebook in a similar fashion, editing if needed to choose another QPU.

14. Save your notebook to submit as part of Homework 2. (Don't forget to shut down jupyter when finished to prevent zombie processes, and type `deactivate` at the shell prompt to get out of the virtual environment.)