Ftl_quantum

Forty Two Lyon - Quantum

Ludovic Lemaire lulemair@student.42lyon.fr

Summary: Introduction to quantum programming.

Version: 1

Contents

Chapter I

Preamble

The following is a non-exhaustive list of resources dealing with quantum physics as a whole that we recommend you to read/watch before getting involved in the subject.

- Julien Bobroff's conference at USI on quantum levitation.
- David Louapre's videos about quantum physics.
- The book L'Univers à portée de main, by Christophe Galfard
- The book A Brief History of Time, From the Big Bang to Black Holes, by Stephen Hawking.

These resources will allow you to have some theoretical knowledge that will be very useful in the project. But also to demystify quantum physics, that it is now very well understood and that we use it every day.

Chapter II

Introduction

This project is an introduction to quantum programming. It will challenge you to create different quantum programs, and run them on a real quantum computer.

Chapter III

Objectifs

The final goal is to recreate a search algorithm on a quantum computer.
Don't panic, we are aware that quantum physics is a complex subject, take the time you need. The project will try to make you progress gradually before reaching the final goal.

Chapter IV

General guidelines

- You will have to develop your circuits and programs with Python and Qiskit.
- There are several languages/frameworks such as:
    - Qiskit (IBM).
    - Cirq (Google).
    - Q # (Microsoft).

`To facilitate learning, the subject will focus specifically on the IBM framework, which makes quantum computers available for free. It is possible (it is even recommended) to use a virtual machine or a container to create your working environment.`

- You can also use Jupyter Notebook, Google Colaboratory or equivalent if you want, if you use an online version, please push the .ipynb in your repo.
- You must return one file per exercise (.py or .ipynb).
- It goes without saying, but you have to develop everything yourself, you can't use a function that does a search algorithm for you, even partially. This rule applies to all exercises.
- Make sure you understand what you are doing! You will be asked questions during the evaluation.

Chapter V

Mandatory part

V.1 Prerequisites

- Create an account on IBMQ and get your token.

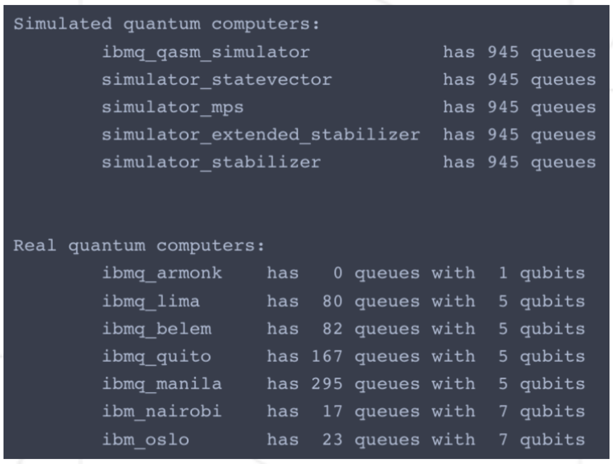
Don't put it on your repository! Make a system to retrieve it via a local file, or a simple C&P during the evaluation.

V.2 Exercise 1: Token

By using the IBMQ.get_provider function, write a program that will have to:

- List all available quantum simulators with their current queue.
- List all available quantum computers with their current queue and the number of qubits they have.

Example of an expected result



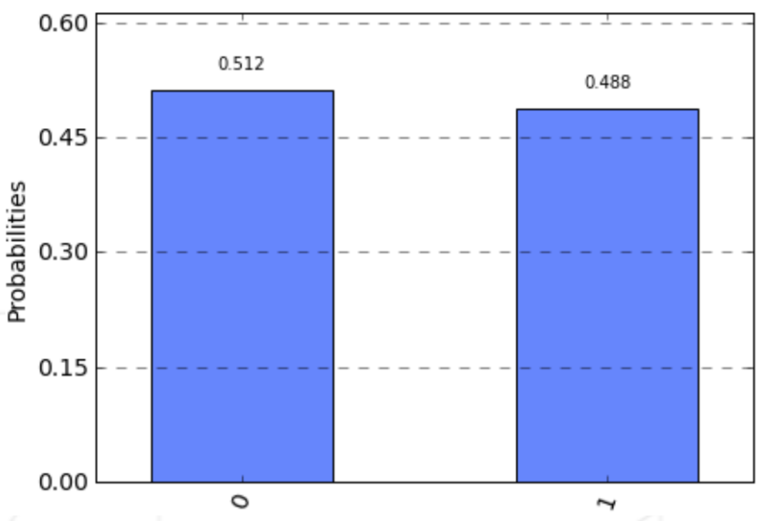
```
Simulated quantum computers:
        ibmq_qasm_simulator            has 945 queues
        simulator_statevector          has 945 queues
        simulator_mps                  has 945 queues
        simulator_extended_stabilizer  has 945 queues
        simulator_stabilizer           has 945 queues


Real quantum computers:
        ibmq_armonk    has   0 queues with  1 qubits
        ibmq_lima      has  80 queues with  5 qubits
        ibmq_belem     has  82 queues with  5 qubits
        ibmq_quito     has 167 queues with  5 qubits
        ibmq_manila    has 295 queues with  5 qubits
        ibm_nairobi    has  17 queues with  7 qubits
        ibm_oslo       has  23 queues with  7 qubits
```

V.3 Exercise 2: Superposition

Write a program (using Python and Qiskit) that will produce a quantum circuit with a single qubit to obtain this $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ state using the principle of quantum superposition.

The program should display a visual of the circuit, then run it on a quantum simulator with 500 shots and then display the results in a plot_histogram (*you can use a local simulator such as Aer if you wish*).

Expected result of the plot_histogram



Be sure you understand the principle of Quantum Superposition and the $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ notation before moving on.

V.4 Exercise 3: Entanglement

Write a program that will produce a quantum circuit with two qubits in order to obtain this $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ state using the principle of superposition and quantum entanglement.

The program must display the circuit, then run it on a quantum simulator with 500 shots and then display the results in a plot_histogram.

Be sure you understand the principle of quantum entanglement and the $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ notation before continuing.

V.5 Exercise 4: Quantum noise

Take the program from exercise 3, and modify it to run your circuit on a real quantum computer.

Your results between exercise 3 and 4 should be different, even though the circuit is identical. It's up to you to understand why.

From there, we strongly advise you to go back to what you have done, and understand the applied math/vector formulas behind each action, and test the different existing gates.

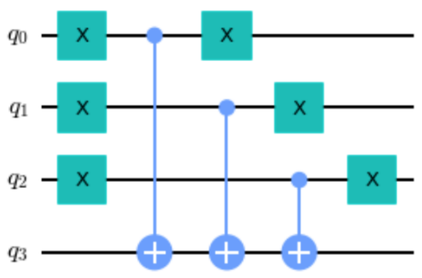
V.6 Exercise 5: Deutsch-Jozsa

You are about to create your first quantum algorithm: the Deutsch-Jozsa algorithm. Although it is not of great practical interest, it is one of the first quantum algorithms that is more efficient than a classical algorithm.

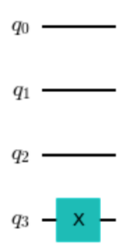
You have to recreate the Deutsch-Jozsa algorithm, it should work with a total number of 4 qubits.

Here are two examples of Oracle Constant and Oracle Balanced circuits.

Oracle Balanced



Oracle Constant



When applying your algorithm, your circuit should be able to determine whether the Oracle is Constant or Balanced, based on the measurement of your Qubits.

- Your Qubits must be 1 if the Oracle is Balanced.
- Your Qubits must be 0 if the Oracle is Constant.

During the evaluation, other Oracle examples (balanced and constant) will be given to test the validity of your algorithm.

V.7 Exercise 6: Search algorithm

The search algorithm, the ultimate goal of this project.
Your algorithm should search for one or more items that meet a given requirement among N unclassified items.

- On a traditional computer, the complexity of the problem is O(N).
- On a quantum computer, the complexity is reduced to O($\sqrt{(N)}$).

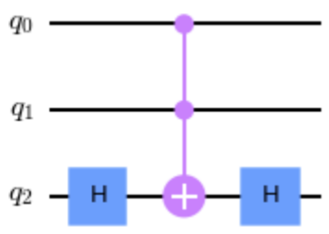
You will need to have 3 distinct parts:

- The initialization of states.
- The Oracle.
- Diffuser.

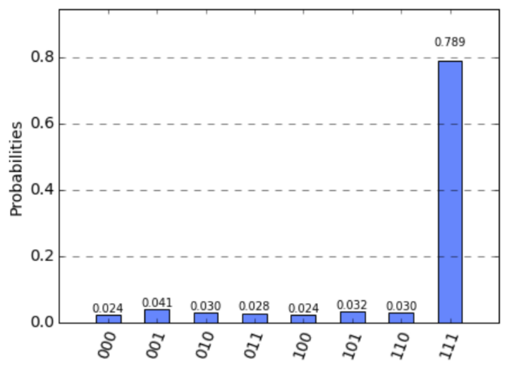
Your algorithm will take a Y number of qubits (minimum 2) and must not require any modification to work.

Similar to the Deutsch-Jozsa algorithm, several Oracle's will be provided during the evaluation to verify that your algorithm is working properly.

Here is an example of an Oracle running on 3 qubits:



And here is the expected answer by your algorithm for this oracle:



In addition to making your program containing the quantum search algorithm. During the evaluation, you will have to explain how and why your quantum algorithm is faster than a search algorithm on a classical computer.

Chapter VI

Bonus part

Only additional algorithms will be accepted as a bonus. Example of interesting algorithms:

- Bernstein-Vazirani.
- Simon.
- Shor.

⚠ The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

Chapter VII

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.