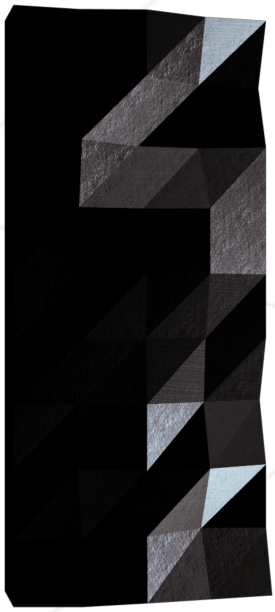


Ftl_quantum

Forty Two Lyon - Quantum

Ludovic Lemaire lulemair@student.42lyon.fr

Summary: Introduction à la programmation quantique.

Version: 1

Contents

Chapter I

Préambule

Voici une liste non exhaustive de ressources traitant de la physique quantique dans son ensemble que nous vous conseillons de lire/regarder avant de vous lancer sur le sujet.

- La conférence de Julien Bobroff à l'USI sur la lévitation quantique.
- Les vidéos de David Louapre traitant de la physique quantique.
- Le livre L'Univers à portée de main, de Christophe Galfard.
- Le livre Une brève histoire du temps, Du Big Bang aux trous noirs, de Stephen Hawking.

Ces ressources vous permettront d'avoir certaines connaissances théoriques qui vous seront vivement utiles dans le projet. Mais également de démystifier la physique quantique, que celle-ci est aujourd'hui très bien comprise et que nous l'utilisons au quotidien.

Chapter II

Introduction

Ce projet est une introduction à la programmation quantique. Il vous mettra au défi de créer différents programmes quantiques et de les exécuter sur un véritable ordinateur quantique.

Chapter III

Objectifs

L'objectif final est de recréer un algorithme de recherche sur un ordinateur quantique. Don't panic, nous sommes conscient que la physique quantique est un sujet complexe, prenez le temps qu'il vous faut. Le projet va essayer au maximum de vous faire avancer progressivement avant d'atteindre l'objectif final.

Chapter IV

Consignes générales

- Vous devrez développer vos circuits et programmes avec Python et Qiskit.
- Il existe plusieurs langages/framework tel que:
  - Qiskit (IBM).
  - Cirq (Google).
  - Q # (Microsoft).

Pour faciliter l'apprentissage, le sujet se concentrera spécifiquement sur le framework d'IBM, qui met gratuitement à disposition des ordinateurs quantiques. Il est possible (il est même conseillé) d'utiliser une machine virtuelle ou un container pour créer votre environnement de travail.

- Vous pouvez également utiliser Jupyter Notebook, Google Colaboratory ou équivalent si vous le souhaitez, si vous utilisez une version online, veuillez push les .ipynb dans votre repo.
- Vous devez rendre un fichier par exercice (.py ou .ipynb).
- Il en va de soi, mais vous devez tout développer vous même, vous ne pouvez pas utiliser une fonction qui fait un algorithme de recherche à votre place, même partiellement. Cette règle s'applique pour tous les exercices.
- Soyez sûr de comprendre ce que vous faites ! Des questions vous seront posées durant l'évaluation.

Chapter V

Partie obligatoire

V.1 Pré-requis

- Créez vous un compte sur IBMQ et récupérez votre token.

Ne le pushez pas dans votre dépôt ! Faites vous un système pour le récupérer via un fichier local, ou un simple C/C pendant l'évaluation.

V.2 Exercice 1: Token

En utilisant la fonction IBMQ.get_provider, écrivez un programme qui va devoir:

- Lister tous les simulateurs quantiques disponibles avec leur queue actuelle.
- Lister tous les ordinateurs quantiques disponibles avec leur queue actuelle ainsi que le nombre de qubits qu'ils disposent.

Exemple de résultat attendu

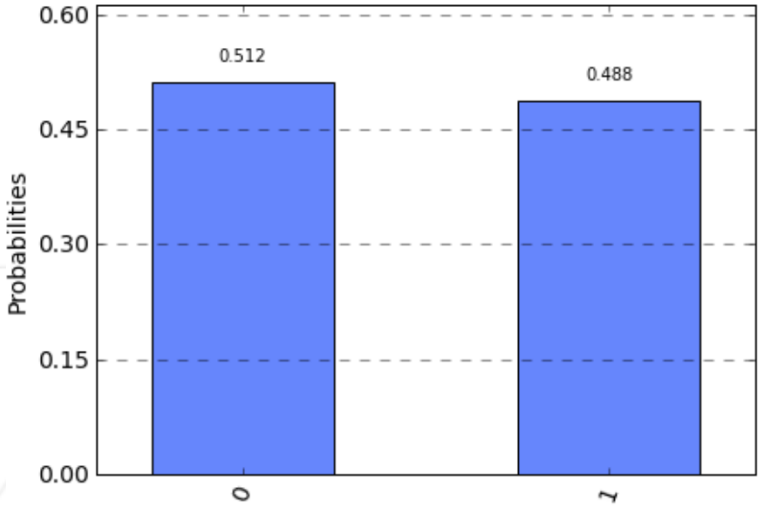
```
Simulated quantum computers:
        ibmq_qasm_simulator             has 945 queues
        simulator_statevector           has 945 queues
        simulator_mps                   has 945 queues
        simulator_extended_stabilizer   has 945 queues
        simulator_stabilizer            has 945 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 Exercice 2: Superposition

Écrivez un programme (grâce à Python et Qiskit) qui va produire un circuit quantique avec un unique qubit afin d'obtenir cet état $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ en utilisant le principe de superposition quantique.

Le programme devra afficher un visuel du circuit, puis le lancer sur un simulateur quantique avec 500 shots puis afficher les résultats dans un plot_histogram (*vous pouvez utiliser un simulateur localement tel que Aer si vous le souhaitez*).

Résultat attendu du plot_histogram



Soyez sûr de bien comprendre le principe de Superposition quantique ainsi que la notation $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ avant de continuer.

V.4 Exercice 3: Entanglement

Écrivez un programme qui va produire un circuit quantique avec deux qubits afin d'obtenir cet état $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ en utilisant le principe de superposition et d'intrication quantique.

Le programme devra afficher le circuit, puis le lancer sur un simulateur quantique avec 500 shots puis afficher les résultats dans un plot_histogram.

Soyez sûr de bien comprendre le principe de d'Intrication quantique ainsi que la notation $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ avant de continuer.

V.5 Exercice 4: Quantum noise

Reprenez le programme de l'exercice 3, et modifiez le pour lancer votre circuit sur un vrai ordinateur quantique.

Vos résultats entre l'exercice 3 et 4 devraient être différents, pour un circuit pourtant identique. À vous de comprendre pourquoi.

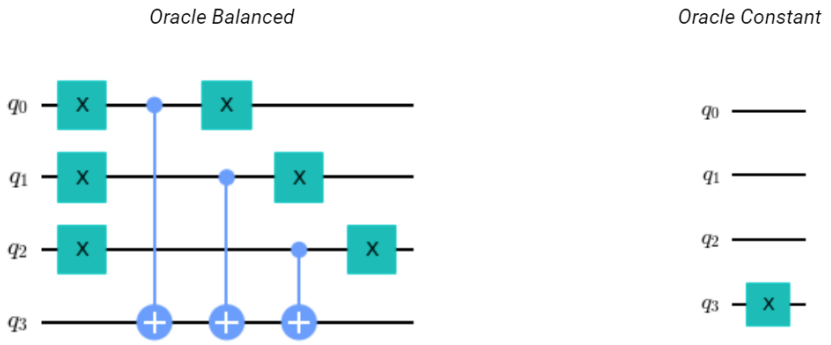
À partir de là, nous vous conseillons vivement de reprendre ce que vous avez fait, comprendre les formules mathématiques/vectorielles appliquées se cachant derrière chaque action, et tester les différentes Gate existantes.

V.6 Exercice 5: Deutsch-Jozsa

Vous allez créer votre premier algorithme quantique: l'algorithme de Deutsch-Jozsa. Bien qu'il ne soit pas d'un grand intérêt pratique, il s'agit d'un des premiers algorithmes quantiques qui est plus efficace qu'un algorithme classique.

Vous devez recréer l'algorithme de Deutsch-Jozsa, il devra fonctionner avec un nombre total de 4 qubits.

Voici deux exemples de circuits d'Oracle Constant et d'Oracle Balanced.



En appliquant votre algorithme, votre circuit doit pouvoir déterminer si l'Oracle est Constant ou Balanced, via la mesure de vos Qubits.

- Vos Qubits doivent être à 1 si l'Oracle est Balanced.
- Vos Qubits doivent être à 0 si l'Oracle est Constant.

Lors de l'évaluation, d'autres exemples Oracle (balanced et constant) seront évidemment fournis, pour tester la validité de votre algorithme.

V.7 Exercice 6: Algorithme de recherche

L'algorithme de recherche, l'objectif final de ce projet.
Votre algorithme doit permettre de rechercher un ou plusieurs éléments qui répondent à un critère donné parmi N éléments non classés.

- Sur un ordinateur traditionnel, la complexité du problème est de O(N) .
- Sur un ordinateur quantique, la complexité est réduite à $O(\sqrt{(N)})$.

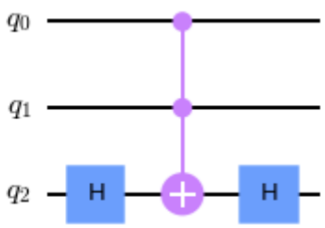
Vous devrez avoir 3 parties distinctes:

- L'initialisation des états.
- L'Oracle.
- Le Diffuser.

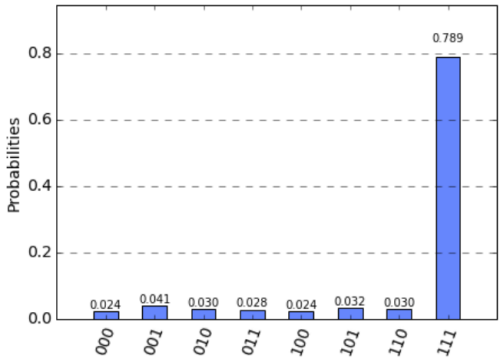
Votre algorithme prendra un nombre Y de qubits (minimum 2) et ne devra demander aucune modification pour fonctionner.

Comme pour l'algorithme de Deutsch-Jozsa, plusieurs Oracle seront fournis lors de l'évaluation pour vérifier le bon fonctionnement de votre algorithme.

Voici l'exemple d'un Oracle fonctionnant sur 3 qubits:



Et voici la réponse attendu par votre algorithme pour cet oracle:



En plus de rendre votre programme contenant l'algorithme de recherche quantique. Vous devrez, lors de l'évaluation, expliquer comment et pourquoi votre algorithme quantique est plus rapide qu'un algorithme de recherche sur un ordinateur classique.

Chapter VI

Partie Bonus

Uniquement des algorithmes supplémentaires seront acceptés en tant que bonus. Exemple d'algorithmes intéressant:

- Bernstein-Vazirani.
- Simon.
- Shor.

Les bonus ne seront évalués que si la partie obligatoire est PARFAITE. Par parfaite, nous entendons complète et sans aucun dysfonctionnement. Si vous n'avez pas réussi TOUS les points de la partie obligatoire, votre partie bonus ne sera pas prise en compte.

Chapter VII

Rendu et peer-évaluation

Rendez votre travail dans votre dépôt `Git` comme d'habitude. Seul le travail présent dans votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.