INFORMATIQUE QUANTIQUE

Tuteurs : Rodin Vincent et Fronville Alexandra

Projet de TAL pour Master SIIA à l'UBO

Maxime Yonnet

07/06/2021

Introduction

Ce projet de Temps de recherche en Laboratoire a eu lieu du 26 Avril jusqu'au 14 Juin, le sujet était l'informatique quantique et la simulation sur Q# et Quantum++. Je travaillais en proche collaboration avec Kevin DRUART, et nous avions Vincent Rodin et Fronville Alexandra comme tuteur dont je tiens à remercier pour leur soutien et les aides qu'ils nous ont apportées. Dans ce projet de TAL, j'ai pu découvrir le fonctionnement de la mécanique quantique, puis comment il est appliqué en informatique quantique et les avantages que cela offre.

Sommaire

I. Qu’est-ce que l’informatique quantique

A. Introduction

L’informatique quantique tient une relation très proche de l’informatique classique, de par le fait que beaucoup de choses sont très ressemblantes. En informatique quantique on remplace les bits classiques par des Qubits et on remplace les portes classiques par des portes quantiques.

Ces intérêts sont multiples, cela peut aller d’optimiser des programmes de recherche, ou de codage/décodage. Le fait d’utiliser des qubits plutôt que des bits permet de stocker plus d’information par exemple avec deux qubits on obtient l’équivalent de quatres bits d’informations car les qubits peuvent être 1 et 0 à la fois. L’informatique quantique pourrait être utile pour simuler des choses avec un grand nombre de contraintes, par exemple simuler un avion en ajoutant la gravité et des forces mécaniques comme contraintes.

B. Qubits

En informatique quantique, comme dit juste précédemment, on utilise des Qubits contrairement à des bits en informatique classique. La différence est qu'un bit classique ne peut prendre uniquement la valeur 1 ou 0, alors qu'un Qubit n'a pas de valeur définie jusqu'à qu'on le mesure. En attendant, il a deux coefficients que l'on appellera α et β, ces deux coefficients permettent de calculer les probabilités d'obtenir une mesure du qubit égale à 0 ou à 1.

$$|\psi> = \alpha * |0> + \beta * |1>$$

En effet, lors de la mesure, un qubit va regarder ses coefficients et faire un tirage aléatoire en utilisant les coefficients au carré comme probabilités. A savoir que la somme des coefficients au carré doit être égale à 1, que l'on appelle ψ l'état d'un qubit et que l'on note |0> ou |1> les valeurs possibles d'un qubit.

$$\alpha^2 + \beta^2 = 1$$

Il est également possible de noter un qubit sous la forme de vecteur avec chaque ligne correspondant au coefficient d'avoir une valeur pour le qubit.

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha * |0> + \beta * |1>$$

Ici la première ligne correspond au coefficient d'avoir 0 comme valeur pour le qubit et la deuxième ligne correspond au coefficient d'avoir 1 comme valeur.

$$|1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix} et\ |0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

A savoir donc que l'état |1> est définit par la matrice dont la première ligne est à 0 et la seconde à 1 et que l'état |0> est l'état initial et est défini par la première ligne à 1 et la seconde à 0.

Lorsque l'on a plusieurs qubit, il est possible de les regrouper sous forme d'une seule matrice afin de pouvoir appliquer des portes que l'on verra plus tard, mais aussi simplifier la lecture. Pour ce faire on prend deux qubits que voici :

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha * |0> + \beta * |1>$$

$$\begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \gamma * |0> + \delta * |1>$$

Alors il est possible de regrouper les deux qubits dans une seule matrice afin de créer ce que l'on appelle un registre de qubit. Pour ce faire, il faut appliquer le produit tensoriel entre les deux vecteurs de qubits.

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{pmatrix} = \alpha*\gamma * |00> + \alpha*\delta * |01> + \beta*\gamma * |10> + \beta*\delta * |11>$$

Regrouper deux qubits donnera une matrice de dimension 4 lignes et 1 colonne. La première ligne correspond au coefficient auquel les deux qubits ont 0 comme valeur soit |00>, la seconde ligne correspond au coefficient auquel le premier qubit est à 0 et le second à 1 soit |01>. La troisième ligne correspond au coefficient pour lequel le premier qubit est à 1 et le second à 0 soit |10> et enfin la quatrième et dernière ligne correspond au coefficient pour lequel les deux qubits sont à 1 soit |11>. A savoir qu'il est possible de regrouper autant de qubit que souhaité et que la matrice aura 2^n lignes et 1 colonne.

C. Portes Quantiques et États superposés

De la même manière qu'en informatique classique, pour modifier un qubit, il faut appliquer une porte à celui-ci. Il en existe une multitude mais nous allons voir ici uniquement ceux qui sont primordiales et donc ceux qui sont les plus utilisés. A la différence d'une porte en informatique classique, une porte quantique doit avoir autant d'entrée que de sortie. Aussi, une porte quantique prend la forme d'une matrice carré ayant pour dimension 2^n avec "n" nombre de qubit requis dans le registre pour pouvoir être appliqué.

$$Pauli - X\ ou\ Not\ :\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Pour commencer voici la porte Pauli-X ou plus souvent appelée la porte not. Cette porte prend un unique qubit en entrée et permet d'inverser les coefficients pour la valeur |0> et |1>. Si l'on prend un qubit à sa valeur initial, soit un qubit avec le coefficient d'avoir |0> égal à 1 et son coefficient d'avoir |1> égal à 0. Alors pour appliquer la porte Not, il faut multiplier la matrice de la porte par la matrice du qubit comme ci dessous.

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Le résultat donne un qubit avec le coefficient d'avoir |0> égal à 0 et le coefficient d'avoir |1> égal à 1.

$$Porte\ Hadamard\ (ou\ H)\ :\ \frac{1}{\sqrt{2}} * \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Ensuite la seconde porte la plus utile se nomme la porte Hadamard ou bien porte H. Lorsqu'elle est appliqué sur un qubit dans son état initial, la porte H permet de faire qu'on a autant de chance d'avoir une valeur |0> qu'une valeur |1> pour le qubit, on dit alors que le qubit est en état de superposition.

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

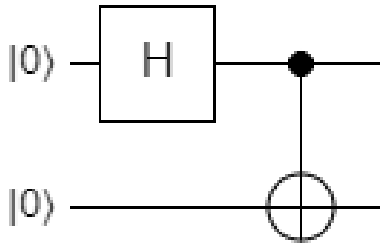
Après avoir appliqué la porte H sur le qubit, on peut voir que les coefficients sont à 1/sqrt(2) ce qui veut dire que les probabilités elles sont à 0.5, donc on est bien en état de superposition.

D. Intrication et états de Bell

Une des notions les plus importantes dans le monde quantique est l'intrication. L'intrication est le fait de lier plusieurs qubits de sorte à ce que mesurer un qubit nous permettent de deviner la valeur d'autres qubits. Si on prend l'état suivant, aussi appelé états de Bell :

$$\frac{1}{\sqrt{2}} * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} |00> + \frac{1}{\sqrt{2}} |11>$$

Alors on peut voir qu'il y a une chance sur deux d'obtenir les deux qubits à |0> et |1>. Cela signifie que si l'on mesure le premier qubit et que l'on tombe sur la valeur |0> alors on sait forcément que l'autre qubit aura la même valeur. De même si on mesure le second qubit à 1, on sait forcément que le premier qubit sera égal à 1.
Pour arriver aux états de Bell il faut appliquer les portes suivantes :



On applique la porte H que l'on a vu sur la première porte :

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Ensuite on doit appliquer une nouvelle porte, la porte CNOT.

$$\mathrm{CNOT} = cX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Cette porte est identique à la porte Not à l'exception que cette porte a besoin de deux qubits pour fonctionner, un qubit de contrôle et un qubit sur lequel on applique le not. Le qubit de contrôle permet de définir si l'on applique le not ou pas.

Comme cette matrice est de dimension 4 par 4 et donc qu'elle a besoin de deux qubits, il faut grouper sous un seul registre de qubits, deux qubits. Pour ce faire, on va faire le produit tensoriel entre le qubit que l'on vient de calculer et un qubit à l'état initial.

$$\frac{1}{\sqrt{2}} * \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} * \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Le résultat est équivalent à lorsque les qubits étaient séparés, c'est à dire que l'on a une chance sur deux que le premier qubit soit à |1> ou à |0> et l'autre qubit sera toujours à 0. Maintenant qu'on a un registre de deux qubits, on peut appliquer la porte CNOT.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \frac{1}{\sqrt{2}} * \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} |00> + \frac{1}{\sqrt{2}} |11>$$

On obtient alors le résultat vu tout à l'heure ou on a une probabilité d'avoir 1 à 0.5 et de même pour celle d'avoir 0.

II. Différences entre simulations et ordinateurs quantique

Actuellement avoir un ordinateur quantique est très rare et n'est pas à la portée de tous, c'est pour cela que certains développeurs ont créé un moyen de programmer des programmes quantiques fonctionnant sur des ordinateurs classiques. Nous allons voir les différences que cela apporte.

A. Différence Simulateur et ordinateur quantique

Comme dit précédemment, il existe deux façon de faire tourner un programme quantique, ou bien le faire tourner sur un ordinateur quantique, ou alors le faire tourner sur un simulateur quantique, c'est-à-dire sur un ordinateur classique qui simule le fonctionnement d'un ordinateur quantique.

L'avantage d'utiliser un simulateur est qu'il peut tourner sur n'importe quelle machine, mais aussi que les erreurs de calculs sont rares et que la limite du nombre de qubit est définie par la mémoire vive de l'ordinateur. Sauf que bien évidemment utiliser un simulateur quantique ne permet pas d'avoir la rapidité de calcul que fournit un ordinateur quantique.

B. Solutions existantes

Voici sous forme de tableau, une liste des différentes solutions pour programmer sur un ordinateur quantique ou sur un simulateur quantique.

Nom	Ibm	Amazon	Google	Q#/Q++	Qiskit
Simulateur	oui	oui	Non	oui	oui
Ordinateur	oui	oui	oui	non	non
Nombre de qubit simulé	5000	2000	X	X	X
Nombre de qubit réel	32	50	54	X	X

Les trois premiers sont des sites web qui permettent d'exécuter notre programme sur leur machine à eux, que ce soit simulateur ou ordinateur. Les deux derniers sont des langages de programmations que l'on peut utiliser ou bien sur l'un des sites, ou bien sur notre

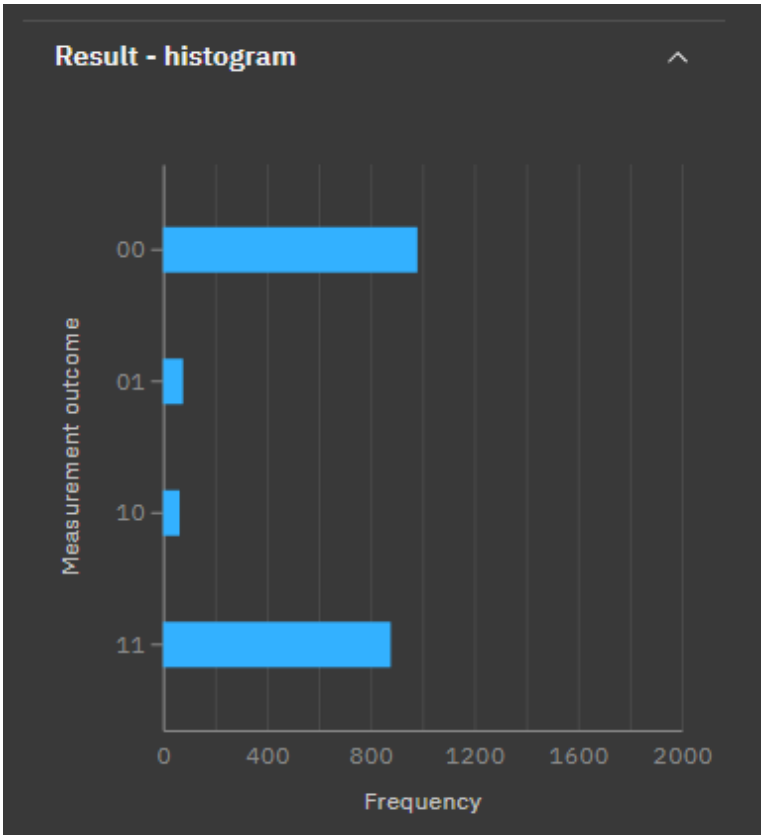
machine personnelle. Aussi certains sites sont gratuit et d'autres payant, j'ai mis en vert ceux qui sont gratuit d'utilisation.

Pour ma part j'ai utilisé ceux entouré en orange, soit IBM et Quantum#. J'ai aussi utilisé un autre site très utile qui est Quirk: Quantum Circuit Simulator, celui ci permet de simuler le comportement d'un programme quantique juste en regardant les probabilités et les coefficients, ce qui est utile pour faire des programmes étapes par étapes sans devoir les exécuter à la chaîne.

C. IBM

Le site gratuit qui semble le plus complet est celui d'IBM, celui-ci permet de faire des programmes jusqu'à 32 qubits sur des ordinateurs quantiques et 5000 sur des simulateurs quantiques.
Voici à quoi ressemble l'interface :

Cette partie de l'interface permet de créer les programmes souhaités, il suffit de faire glisser la porte sur le qubit voulu. On peut aussi voir qu'il existe bien plus de portes que l'on a vu tout à l'heure, mais pour des raisons de simplicité on va se concentrer sur les trois premiers.
A noter aussi qu'ici on retrouve l'algorithme des états de Bell de tout à l'heure.
Lors de l'exécution, le site nous permet de choisir la machine sur laquelle on souhaite exécuter le programme et aussi combien d'itération du calcul on souhaite faire.



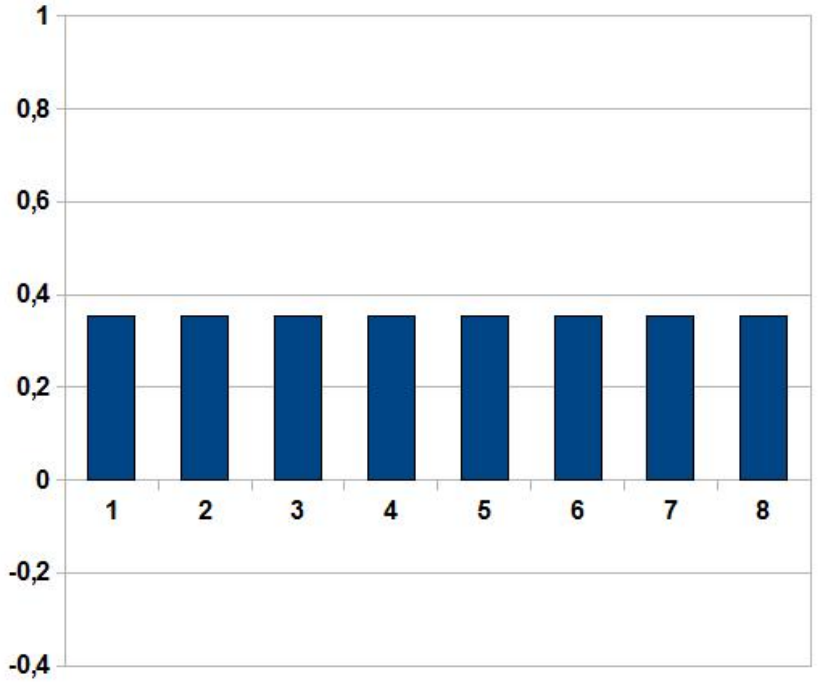
Une fois l'exécution faite, on peut trouver cette page qui nous indique les résultats. Ici on peut voir que sur 2000 exécutions on retrouve un grand nombre de fois 00 et 11. Les deux autres résultats étant des erreurs de l'ordinateur quantique.

III. Algorithme de Grover

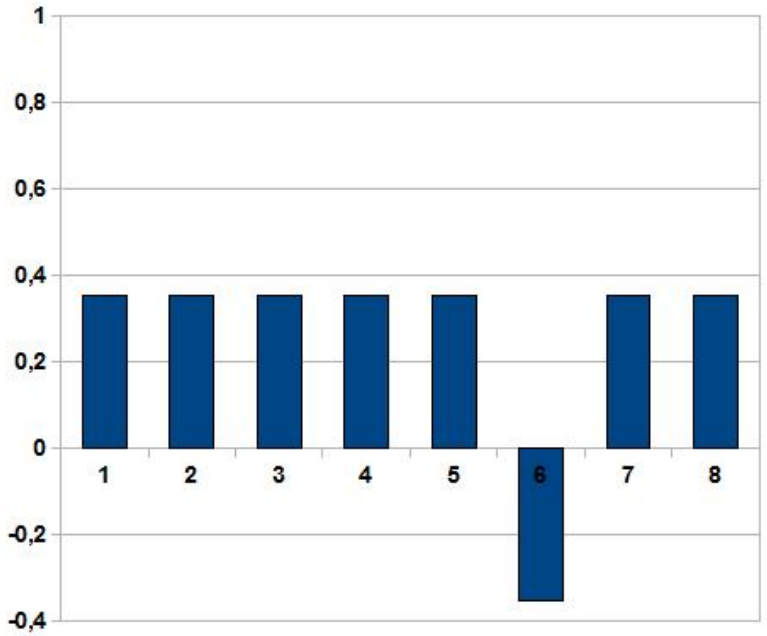
A. Utilité

L'algorithme de Grover est un des algorithmes les plus connus en informatique quantique. Il permet de résoudre des problèmes de recherche dans des listes non ordonnées. Pour des applications dans le monde réel, cela pourrait être recherché dans une base de données, ou bien une recherche inverse dans un annuaire téléphonique. Cela pourrait être aussi des problèmes plus complexes tel que le voyageur de commerce ou bien décodage de messages.
L'avantage de l'algorithme de Grover est qu'il est bien plus performant qu'un algorithme classique. Si l'on prend une liste de n éléments et que l'on cherche le dernier élément de celle-ci, en informatique classique il faudrait vérifier un par un chaque élément, ce qui veut dire qu'il faudrait entre 1 et n tours de boucle pour arriver au bon résultat et donc que la complexité de ce programme serait O(n), avec n étant le nombre d'éléments. Si maintenant on utilise l'algorithme de Grover pour résoudre ce problème, il faut √n itérations donc une complexité égale à O(√n).

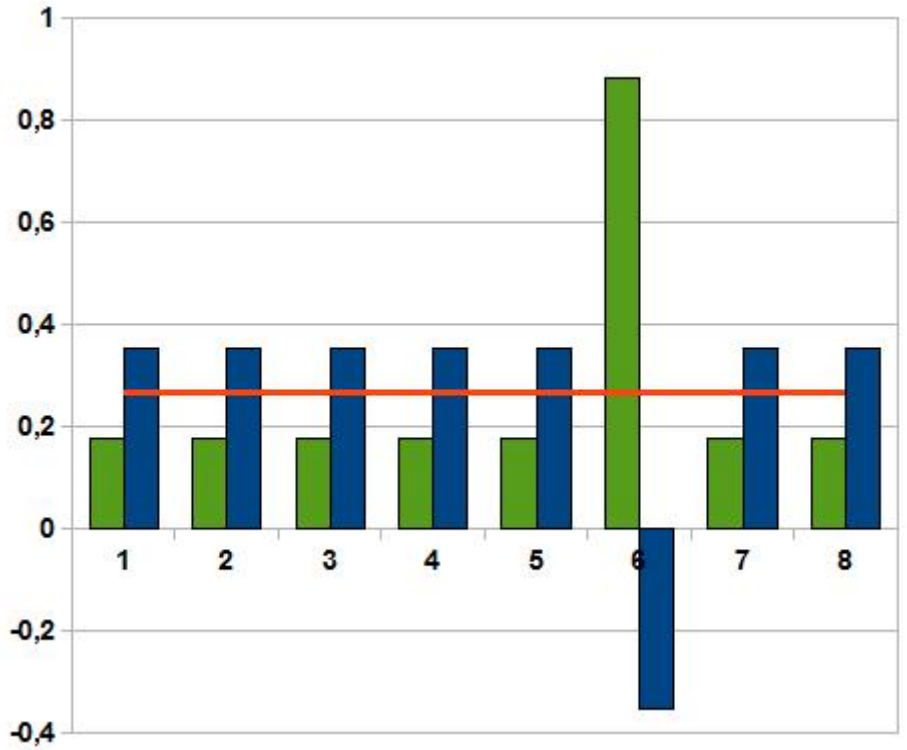
B. Fonctionnement



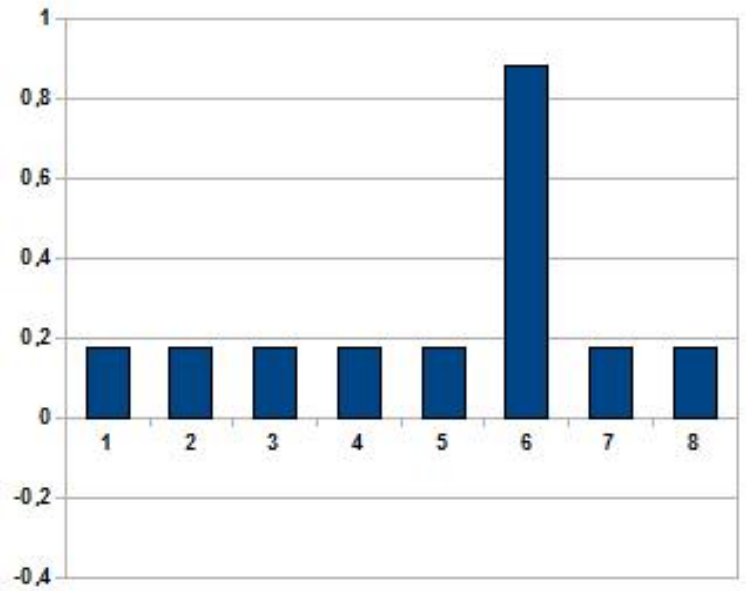
Pour commencer, il faut que tous les qubits soient en état de superposition, c'est-à-dire qu'on ait autant de chance d'avoir chacune des valeurs lors de la mesure.



Ensuite, il faut passer tous les qubits dans ce qu'on appelle l'oracle. Son but est de transformer le coefficient de la valeur que l'on cherche en valeur négative. Malheureusement comme la probabilité est égale au coefficient au carré, alors si l'on venait à mesurer ici, il n'y aurait aucune différence entre juste avant cette étape et maintenant. C'est pour cela qu'il faut passer à l'étape d'amplification.

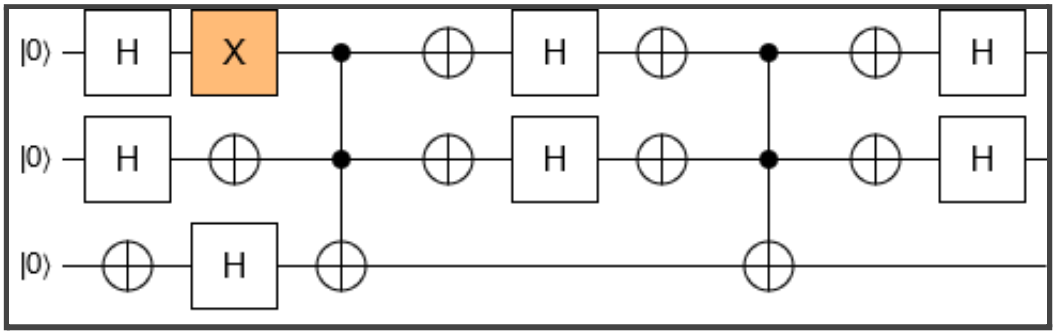


Cette étape permet d'augmenter le coefficient de tous ceux qui sont négatifs et de faire diminuer ceux qui sont positifs. Pour ce faire, on fait la moyenne de tous les coefficients et on obtient la valeur orange. Ensuite on fait la symétrie de chaque coefficient avec la moyenne, cela nous donne le résultat vert.



On remarque donc que désormais, le coefficient d'avoir la valeur que l'on cherche, 6, a été grandement augmenté. Il est possible d'appliquer cet algorithme une multitude de fois pour obtenir un résultat encore plus précis. Le nombre d'itération est égal à √n, avec n étant le nombre de valeur possible.

C. Réalisation



Si on imagine une liste de quatres éléments soit deux qubits où l'on cherche la valeur 0 c'est -à-dire la combinaison ou les deux qubits sont à 0. Voici à quoi ressemble le programme réalisé sur Algassert.

Les deux premiers qubits sont ceux qui simulent la valeur. Le troisième qubit est un ancilla qubit. Le but d'un ancilla qubit est de modifier l'état des autres qubits et dont le résultat n'est pas mesuré car il ne nous intéresse pas. Ici son but est de changer le signe des autres qubits lorsque l'oracle le veut. Lorsqu'on applique la port NOT suivi de la porte H, le résultat est le même que lorsqu'on applique uniquement la porte H à l'exception que le coefficient du qubit |1> est négatif.

Les deux premières portes appliquées aux deux premiers qubits sont des portes H pour arriver à l'état initial de superposition. Ensuite vient l'oracle composé de portes NOT et CCNOT. On connaît déjà la porte NOT, la porte CCNOT est comme une porte CNOT seulement il y a deux qubits de contrôle au lieu de un, c'est à dire qu'il faut appliquer le NOT uniquement lorsque les deux qubits de contrôles sont à 1. Donc ici comme on applique une porte NOT avant et après la porte CCNOT, cela a comme conséquence que le NOT est appliqué uniquement quand les deux qubits sont à 0. Comme le troisième qubit a un

coefficient négatif, lorsque la porte CCNOT applique la porte NOT sur le qubit au coefficient négatif, cela fait que les coefficients des qubits de contrôles deviennent négatifs. C'est ce qu'on appelle un "KickBack", c'est le fait de changer les qubits de contrôles à l'aide du qubit d'opération, ici le NOT. De ce fait lorsque les deux qubits sont à 0, la porte CCNOT transforment les coefficients des qubits en valeur négatif comme doit le faire l'oracle.
Ensuite vient deux portes H puis à nouveau une porte CCNOT entouré de deux portes NOT, et enfin deux portes H. Ces portes auront comme conséquences de faire des symétries afin de faire varier les coefficients selon la moyenne des coefficients et si les coefficients sont négatifs ou positifs. A la fin on obtient une probabilité plus grande d'obtenir la valeur que l'on cherche et il suffit d'exécuter cet algorithme plusieurs fois afin de faire encore augmenter la probabilité.

D. Calcul Théorique

Afin de vérifier que le programme fonctionne bien, j'ai réalisé le calcul matriciel de ce programme, il est disponible en annexe page 22.
On a vu précédemment que pour certaines portes matriciels, il fallait un certain nombre de qubits dans le registre, mais l'inverse est vrai aussi, pour un certain nombre de qubit dans un registre, il faut une matrice avec une certaine dimension pour fonctionner. De ce fait, il m'a fallu transformer certaines matrices afin qu'elles soient utilisables sur le registre. Pour transformer une porte quantique, il y a deux solutions.

La première fonctionne uniquement sur les portes à résultat binaire par exemple NOTCC, qui est une porte CCNOT inversée.

	000	001	010	011	100	101	110	111
000	1	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0
010	0	0	1	0	0	0	0	0
011	0	0	0	0	0	0	0	1
100	0	0	0	0	1	0	0	0
101	0	0	0	0	0	1	0	0
110	0	0	0	0	0	0	1	0
111	0	0	0	1	0	0	0	0

Pour trouver la matrice CCNOT, j'ai utilisé un tableau listant tous les états possibles des qubits, ensuite il suffisait juste de mettre des 1 lorsque la valeur de la ligne une fois passé dans la porte doit donner la valeur de la colonne. Par exemple, lorsque les qubits sont à 000, alors une fois passé dans la matrice, le résultat donne 000, donc le croisement entre les deux est à 1 et tous les autres sont à 0.

L'autre méthode fonctionne de façon plus générique, il suffit de faire un produit tensoriel entre les portes que l'on souhaite fusionner. Par exemple, si l'on souhaite appliquer la porte NOT à deux qubits mais laisser le troisième comme il est, il n'existe pas de porte qui permet de faire ça, donc il faut la créer nous même. Pour ce faire, on prend la porte NOT et on fait le produit tensoriel avec une autre porte NOT et enfin on refait un produit tensoriel entre le résultat et la porte Identité. Cette porte dont nous n'avons pas parlé jusqu'à présent, n'a aucun effet sur le qubit, et c'est exactement ce que l'on souhaite.

$$INOTNOT = I \otimes NOT \otimes NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

La matrice 8x8 que l'on peut voir est donc la porte INOTNOT, utilisée dans mes calculs, cette porte permet d'appliquer la porte NOT aux deux premiers qubits et de laisser inchangé le dernier qubit.

IV. Limites actuelles de l'informatique quantique

A. Limites matériels

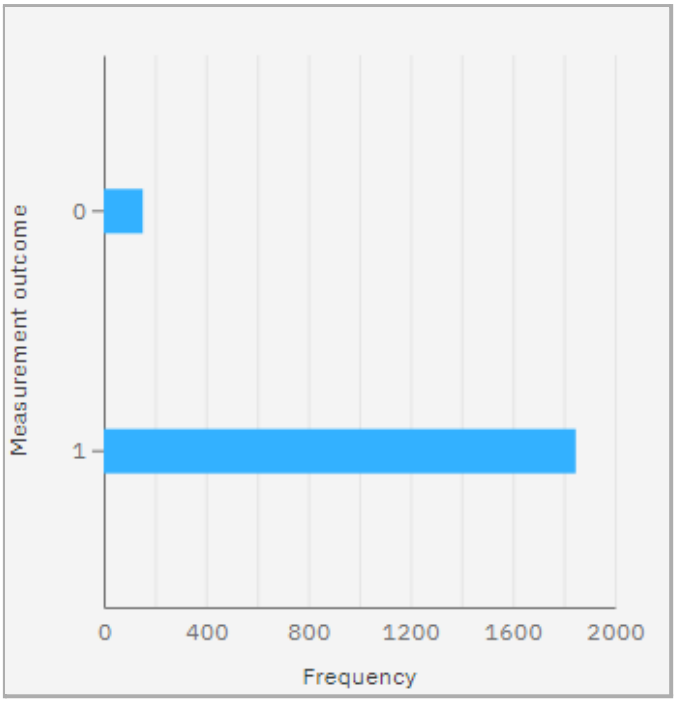
Actuellement la contrainte qui empêche le déploiement des ordinateurs quantiques sont les limites physiques. En effet, pour faire fonctionner un ordinateur quantique, il faut réussir dans un premier temps à le construire mais aussi à le maintenir à une température du 0 absolue. Dans le cas contraire, les qubits vont perdre leur état et créer des erreurs lors des mesures. Lorsque de telles erreurs se produisent, on appelle ça la décohérence quantique.

Avoir une complexité matérielle aussi grande implique donc que le prix des ordinateurs est très élevé, ce qui les rends moins accessibles de façon générale.

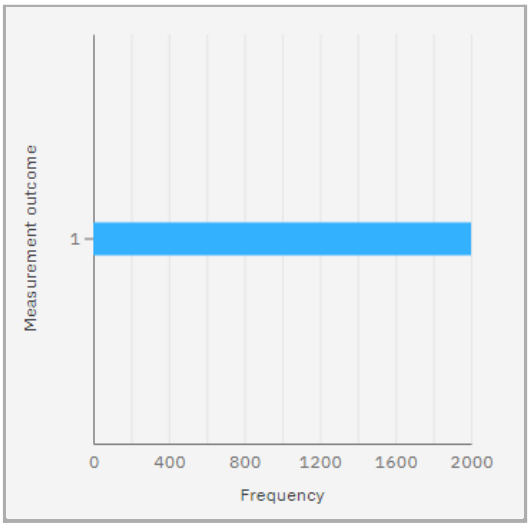
B. Erreurs ordinateur quantique

Comme on vient de le voir, un ordinateur quantique produit des erreurs. La question réside maintenant dans le fait de savoir si les ordinateurs classiques produisent eux aussi des erreurs.

Pour vérifier cela, on a fait un programme tout simple, contenant trois portes NOT, ce qui fait que le résultat lors de la mesure doit être 1.



Lorsqu'on a exécuté 2000 fois ce programme sur un ordinateur quantique on a pu remarquer qu'il y avait 153 fois le nombre 0 et 1847 fois le nombre 1. Cela représente donc 8% d'erreurs.

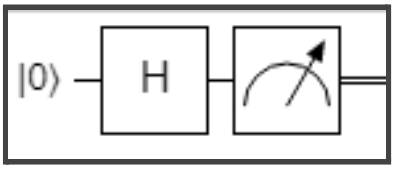


On a alors essayé d'exécuter ça sur plusieurs simulateur, que ce soit celui d'IBM, sur Q# ou Q++, mais le résultat était toujours le même. Il n'y avait aucune erreur.

On peut donc conclure qu'un simulateur quantique ne simule pas les erreurs et qu'on voit ici une limite des ordinateurs quantiques.

C. Pseudo-Aléatoire/Aléatoire

Après avoir analysé le taux d'erreur des ordinateurs quantiques, un autre point questionnable est l'aléatoire dans les simulateurs. En effet on sait que dans les ordinateurs quantiques, l'aléatoire est du vrai aléatoire, alors que dans les simulateurs quantiques, l'aléatoire est du pseudo aléatoire. La question est donc, est-ce que ce pseudo aléatoire peut imiter le vrai aléatoire ?



Pour vérifier cela, on a fait un programme au plus simple, avec une porte H puis une mesure juste après, de façon matricielle, le résultat devrait toujours être une fois sur deux 0 et l'autre fois 1. On a donc exécuté ça sur un simulateur et on a regardé les résultats.
Pour organiser les résultats, on a fait trois catégories, lorsqu'on mesure 100 fois d'affilés, puis 100 000 et enfin 1 000 000. Ensuite on a calculé l'écart type théorique de chaque colonne et on l'a comparé à l'écart type trouvé.

Nombre de tirage	100	100000	1000000
Tirage 1	59 / 41	50076 / 49924	4999551 / 500449
Écart type théorique	5	158	500
Écart type calculé	9	24	449

Ici on peut voir que lorsque le nombre de tirage est assez élevé, l'aléatoire est respecté et on peut comparer le pseudo aléatoire des simulateurs à l'aléatoire des ordinateurs quantiques.

V. Conclusion

Pour conclure, on peut voir que les ordinateurs quantiques sont compliqués à maîtriser, que leur résultat est parfois incertain, mais que leur but est réel. Un ordinateur quantique fait gagner du temps de calcul et ce dans n'importe quel domaine. C'est pour ces raisons qu'il est primordial d'avoir des simulateurs fonctionnels qui permettent d'imiter le comportement des ordinateurs quantiques lorsque ceux-ci n'auront plus aucune erreur. Ils nous permettent de nous entraîner jusqu'à ce que les ordinateurs quantiques soient prêts à l'usage et déployés partout.

VI. Sources

La composition de la matière
Science Étonnante
Porte quantique — Wikipédia
Algorithme de Grover — Wikipédia
Grover's Algorithm : Mathematics & Code | Quantum Untangled
Grover's Algorithm: Quantum Algorithms Untangled | by Shrey Biswas | Quantum Untangled
Symbolab Math Solver - Step by Step calculator
https://www.dcode.fr/produit-tensoriel
https://algassert.com/quirk
Grovers Algorithm — Programming on Quantum Computers — Coding with Qiskit S2E3
Dinner Party using Grover's Algorithm — Programming on Quantum Computers — Coding with Qiskit S2E5
Quantum science: The quest for quantum information technology expands
The Need, Promise, and Reality of Quantum Computing
Suivant Effacer de manière sûre un disque dur portes logiques
Intro to Quantum Logic Gates – Lahiru Madushanka

VII. Annexes

A. Grover's algorithm

$$qt = (q2 \otimes q1 \otimes q0) : \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{2} * \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{2\sqrt{2}} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$$NOTCC * qt : \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} * \frac{1}{2\sqrt{2}} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} = \frac{1}{2\sqrt{2}} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$

$$INOTNOT * qt : \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} * \frac{1}{2\sqrt{2}} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \frac{1}{2\sqrt{2}} * \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$$HHI * \frac{1}{2\sqrt{2}} * \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$INOTNOT * qt = \frac{1}{2\sqrt{2}} \begin{pmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

$$NOTCC * qt = \frac{1}{2\sqrt{2}} \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$INOTNOT * qt = \frac{1}{2\sqrt{2}} \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$IHH * qt = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$