# Algorithmique quantique : de l'exponentiel au polynômial

Michaël Monerau

Novembre 2008

**Résumé**

L'informatique quantique, même si elle n'en est encore qu'à ses premiers pas, porte en elle des promesses qui lui ont valu un engouement spontané de la part de la communauté scientifique. Même si la réalisation d'ordinateurs quantiques reste hypothétique à l'heure actuelle (sauf pour des miniatures), c'est un domaine de recherche qui a été très actif ces quinze dernières années, et pour cause : ces nouveaux concepts permettraient d'accélérer drastiquement des opérations qui coûtent beaucoup trop cher sur nos ordinateurs actuels (on passe d'une complexité exponentielle à une polynômiale, à une très faible probabilité d'erreur près).

Après avoir expliqué brièvement en quoi consiste le modèle de l'ordinateur quantique, nous essaierons de comprendre et d'apprivoiser la nouvelle puissance de calcul qui s'offre à nous par l'intermédiaire de deux des plus célèbres algorithmes quantiques : l'algorithme de factorisation de Shor et l'algorithme de recherche de Grover. Nous ne présenterons que brièvement le second, le cœur de ce travail étant l'algorithme de Shor.

## Table des matières

# 1 L'ordinateur quantique

## 1.1 L'ordinateur classique

Les ordinateurs que nous manipulons conservent et manipulent leur information à l'aide de bits : chaque bit peut valoir 0 ou 1. Avec des millions de tels bits, on peut stocker et traiter d'importantes quantités d'information. Une suite de bits est une donnée qui peut alors représenter un nombre en base 2, une adresse mémoire, un mot, ou une quelconque autre information qu'on sait conserver et relire sous forme de 0 et de 1.

Plus précisément, toute manipulation de ces bits passe par l'utilisation de portes logiques qui prennent en entrée un certain nombre de bits, disons $n$, et fournissent en sortie un résultat sur $m$ bits, qui est l'application d'une certaine fonction booléenne $f$ aux bits d'entrée.

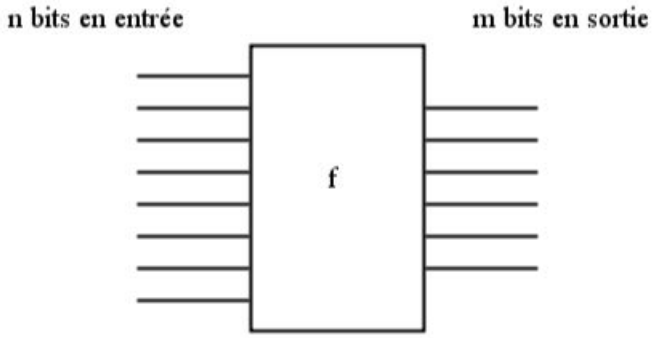


FIG. 1 – Porte logique d'une fonction booléenne $f : \{0,1\}^n \longrightarrow \{0,1\}^m$

On peut montrer avec quelques notions de logique propositionnelle élémentaires que de tels circuits peuvent toujours être décomposés en un réseau de portes **NAND** ( = **NON - ET**) à 2 entrées et 1 sortie. Par réseau, on entend un ensemble de portes logiques reliées entre elles (une entrée d'une porte est la sortie d'une autre, ou une des deux constantes 0 et 1), et où on a le droit de diviser un fil en deux, pour le donner en entrée à 2 portes par exemple.

Physiquement, on réalise ces circuits à l'aide de transistors et autres puces électroniques, où un bit à 0 correspond à un faible potentiel, et un 1 à un fort potentiel (le plus souvent).

## 1.2 L'ordinateur quantique

Le principe de base de l'ordinateur quantique est de changer le support de l'information. Au lieu d'un *bit*, on manipule un *qubit*. On définit l'état quantique d'un *qubit* comme un vecteur unitaire de $\mathbb{C}^2$.

Afin de représenter cet état, on se fixe une base orthonormée de $\mathbb{C}^2$ qu'on note $|0\rangle$ et $|1\rangle$. Un qubit $|\psi\rangle$ peut alors être dans un de ces deux états (on retrouve le *bit* usuel), mais surtout il peut aussi être dans une superposition de ces deux états :

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad \text{avec} \quad \alpha, \beta \in \mathbb{C}, \text{ et } |\alpha|^2 + |\beta|^2 = 1$$

Ainsi, là où on avait un nombre d'états possibles fini dans le cas du bit, un qubit peut prendre une infinité de valeurs. Mais bien entendu, cela ne veut pas dire qu'on peut stocker une infinité d'information dans un qubit. En effet, les lois de la mécanique quantique nous disent qu'on ne peut avoir accès directement à l'information stockée dans un qubit : une mesure n'est que probabiliste et peut modifier l'état du qubit, changeant le comportement normal du calcul. De plus, une mesure ne rendra qu'un état $|0\rangle$ ou $|1\rangle$, donc ne donnera pas accès à $\alpha$ et $\beta$ qui représentent les vrais porteurs de l'information. Il faudra donc utiliser des astuces pour exploiter tout de même cette puissance quantique.

On peut maintenant généraliser ce concept de qubit pour accéder au $n$-qubit. On se place alors dans l'espace $(\mathbb{C}^2)^{\otimes n}$ qui représente un système formé de $n$ qubits. Et alors tout état $|\psi\rangle$ du $n$-qubit peut se décomposer sur la base dite *de calcul* de $(\mathbb{C}^2)^{\otimes n}$ obtenue à partir de la base $(|0\rangle, |1\rangle)$ de $\mathbb{C}^2$ par produits

tensoriels successifs : $(|0\dots0\rangle, |0\dots01\rangle, \dots, |1\dots1\rangle)$. Par convention, on notera, pour $a \leq 2^n - 1$ entier, $|a\rangle$ le vecteur de la base correspondant à l'écriture binaire de $a$. On peut alors décomposer :

$$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle \quad \text{avec} \quad a_i \in \mathbb{C} \text{ et } \sum_{i=1}^{2^n-1} |a_i|^2 = 1$$

Si on cherche à avoir accès à l'état effectif de $|\psi\rangle$, on trouvera l'état $|i\rangle$ avec une probabilité $|a_i|^2$ d'après les lois de la mécanique quantique. De plus, le $n$-qubit sera dans l'état $|i\rangle$ après la mesure, modifiant donc éventuellement sa valeur pour la fixer sur un des vecteurs de la base de calcul. On remarque donc ici une première différence essentielle avec le calcul classique : la mesure peut changer l'état d'un $n$-qubit et ne nous donne même pas accès à son vrai état s'il est superposé au moment de la mesure.

Afin de réaliser des calculs sur ces $n$-qubits, il convient de définir l'analogue des portes logiques de l'informatique traditionnelle dans ce nouveau monde. Mais on rencontre deux problèmes de fond.

Premièrement, les opérations qu'on applique aux $n$-qubits doivent être inversibles (réversibles dans le langage de la physique) : avec la valeur d'un $n$-qubit de sortie, on doit pouvoir retrouver quel était le $n$-qubit d'entrée correspondant (cela impose au passage qu'on ne peut avoir une porte qui prendrait en entrée un $n$-qubit et qui rendrait un $m$-qubit avec $m \neq n$). Cela est dicté par les lois de la physique quantique. Ainsi, on ne peut pas implémenter de manière directe la porte logique **NAND** qui agirait sur un 2-qubit pour rendre un 1-qubit, ni même les usuelles **AND**, etc. Il va falloir ruser pour les implémenter de façon réversible et disposer ainsi au moins de la puissance du calcul classique.

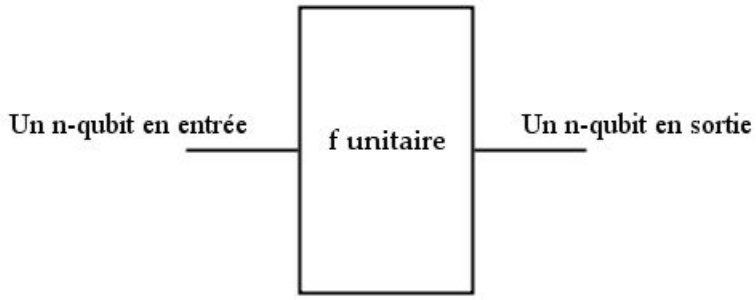


FIG. 2 – Porte quantique d'un opérateur unitaire (donc inversible) $f : (\mathbb{C}^2)^{\otimes n} \longrightarrow (\mathbb{C}^2)^{\otimes n}$

Deuxièmement, il est fondamentalement impossible de dupliquer un $n$-qubit : un circuit quantique ne peut comporter de bifurcations, c'est-à-dire de "dédoublement de fil", comme on se l'autorisait pour un circuit classique. Ceci est dû au théorème de non clonage quantique [NC00, p. 532]. On peut comprendre ce résultat intuitivement de la manière suivante : si on arrivait à dupliquer un état quelconque autant de fois qu'on veut, on aurait accès statistiquement aux $a_i$ ce qui n'est pas raisonnable. Là aussi, une astuce va nous sortir de ce mauvais pas.

## 1.3 Circuits et portes quantiques

Malgré ces deux problèmes, on peut tout de même agir sur des $n$-qubits à la manière de $n$-bits normaux. La fonction booléenne que représente une porte classique est remplacée par un opérateur unitaire de $(\mathbb{C}^2)^{\otimes n}$ : on donne un $n$-qubit $|\psi\rangle$ en entrée à un opérateur unitaire $U$, et il donne en sortie le $n$-qubit $U|\psi\rangle$. Il s'agit donc de trouver des opérateurs unitaires qui représentent les opérations logiques que l'on veut réaliser.

### 1.3.1 Porte de Toffoli

Tout d'abord, il serait satisfaisant qu'un ordinateur quantique ait une puissance de calcul au moins aussi importante qu'un ordinateur classique.

Pour cela, on construit une porte quantique particulière, appelée porte de Toffoli, qui permet d'obtenir une version quantique du **NAND** et de la bifurcation dans un circuit. Ainsi, on aura récupéré au moins la puissance de calcul d'un ordinateur classique d'après le résultat énoncé en 1.1.

La porte de Toffoli s'applique à 3 qubits en entrée ($a$, $b$, $c$) – ie. à un 3-qubit – et produit 3 qubits en sortie ($a'$, $b'$, $c'$). Son action est simple : si les deux premiers qubits sont dans l'état $|1\rangle$, elle remplace la valeur du troisième qubit par son complémentaire. Pour fixer les idées, donnons sa table de vérité dans la base de calcul (les 0 et 1 représentent les états $|0\rangle$ et $|1\rangle$ pour plus de clarté) :

| Entrée | | | Sortie | | |
|---|---|---|---|---|---|
| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |

Cette table donne la valeur de la porte appliquée aux vecteurs de base de $(C^2)^{\otimes 3}$, et elle caractérise donc bien l'opérateur unitaire sous-jacent.

À partir de cette porte quantique, on voit qu'on obtient les opérations "génératrices" du calcul logique classique : **NAND** et la bifurcation. En effet, si on fixe $c = |1\rangle$, alors $c' = |\mathbf{NAND}(\mathbf{a}, \mathbf{b})\rangle$. De même, si on fixe $a = |1\rangle$ et $c = |0\rangle$, alors $b' = c' = b$, et on a donc dupliqué $b$.

Mais si un ordinateur quantique peut bien simuler un ordinateur classique, il présente bien sûr une puissance de calcul beaucoup plus importante. Tout est dans les états dits "d'intrication" des $n$-qubits. En effet, comme les $n$-qubits peuvent être des combinaisons linéaires d'éléments de la base, on peut y stocker plus d'information que si on pouvait simplement se placer sur des vecteurs de la base (comme on le ferait si l'ordinateur était classique).

Malheureusement, les mesures ne renvoient qu'un vecteur de la base, ce qui exclut de mesurer exactement un état intriqué et d'en obtenir les différentes composantes sur chaque vecteur de la base de calcul. Cependant, comme on va le voir dans les algorithmes quantiques qui suivent, on peut tout de même utiliser cette latitude de calcul pour accélérer des algorithmes classiques.

### 1.3.2 Porte CNOT

La porte de Toffoli nous a permis de simuler de manière quantique la logique de **NAND**. Maintenant, nous allons imiter son caractère "générateur".

En effet, on dispose d'une porte de base qui a un rôle analogue à **NAND**, dans le cas classique dans le sens où elle est universelle parmi les opérateurs unitaires. Elle prend en entrée 2 qubits. Le premier est dit *de contrôle*, le second est la *cible*. Son action peut être décrite simplement : si le qubit de contrôle vaut $|0\rangle$, le qubit cible reste tel quel ; mais si le qubit de contrôle vaut $|1\rangle$, alors le qubit cible est changé en son complémentaire. Sa table de vérité est donc :

| Entrée | | Sortie | |
|---|---|---|---|
| $a$ | $b$ | $a'$ | $b'$ |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

et on nomme cette porte **CNOT** (Controlled-NOT). **CNOT** présente une propriété algébrique intéressante :

**[1.A] Proposition** ***(Universalité de CNOT et des portes sur*** $1$***-qubit)***

*À partir de* ***CNOT*** *et des portes quantiques agissant sur un* $1$*-qubit, on peut créer par composition n'importe quelle porte quantique agissant sur un* $n$*-qubit (c'est-à-dire obtenir n'importe quel opérateur unitaire de* $(C^2)^{\otimes n}$*).*

Contrairement au cas classique, il n'y a pas que **NOT** qui agisse sur un seul qubit. Il y en a même une infinité puisqu'il y a une infinité d'opérateurs unitaires sur $\mathbb{C}^2$. On dénote par exemple l'importante porte suivante, qu'on utilisera par la suite :

$$\text{La porte d'Hadamard : } H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Il est bon de noter ici que cela ne résoud pas *totalement* notre problème. En effet, si on peut simuler tout opérateur unitaire par un nombre fini de compositions de portes de bases, un ordinateur quantique effectivement construit ne peut embarquer qu'un nombre *fini* de portes quantiques, et donc ne peut avoir accès à *tout* opérateur agissant sur un qubit – on n'avait pas ce problème dans le cas classique puisqu'il suffisait d'embarquer la seule porte **NAND**.

Cependant, des résultats d'approximation montrent qu'on peut choisir, si on se fixe une précision, un sous-ensemble fini de portes de base pour approximer n'importe quel opérateur unitaire [Fim00].

## 1.4 Complexité d'un algorithme quantique

On parle "d'accélérer des algorithmes classiques", mais encore faut-il se donner un moyen de mesurer effectivement la complexité d'un algorithme quantique.

Rappelons que dans le cas d'un algorithme classique, on choisit de mesurer la complexité en temps : on estime le temps mis par l'algorithme sur une entrée de taille $n$, asymptotiquement. Cela revient à compter le nombre d'opérations "atomiques" réalisées au cours d'une exécution. Les algorithmes polynômiaux en $n$ sont considérés comme étant utilisables en pratique, alors que les exponentiels mettraient beaucoup trop de temps sur des entrées, même petites, pour être utiles.

Sur un ordinateur quantique, on mesure de même le temps d'exécution. On va utiliser le résultat de [1.A] : on considère que l'application de **CNOT** ou d'un opérateur agissant sur un 1-qubit se fait en temps constant. Une mesure prend également un temps constant.

Ainsi, lorsqu'on dispose d'un circuit quantique, on établit sa complexité en décomposant chaque opérateur en **CNOT** et en opérations de 1-qubit, puis on compte.

Dans toute la suite, les opérateurs agissant sur les $n$-qubits se décomposent en un nombre polynômial en $n$ de portes de base. Ainsi, compter un nombre polynômial de ces portes quantiques ou des portes de base est équivalent. On ne se souciera donc plus de ce détail de définition, et on comptera directement les opérateurs agissant sur des $n$-qubits.

L'aspect pratique des algorithmes polynômiaux à opposer aux exponentiels reste évidemment le même : un algorithme polynômial serait assez rapide si un ordinateur quantique l'exécutait, alors qu'un exponentiel serait trop long sur des entrées de taille usuelle.

Au moment où est écrit ce texte, on est parvenu à construire effectivement un ordinateur quantique de 7-qubit, mais pas plus. Il a factorisé 15 avec l'algorithme de Shor que nous allons décrire dans la suite. On est bien loin d'un vrai ordinateur disposant d'un 100-qubit par exemple, mais c'est déjà une belle avancée d'être parvenu à passer de la théorie à la pratique.

# 2 Algorithme de Shor

## 2.1 Description du problème & algorithme

### 2.1.1 Mise en situation

Actuellement, beaucoup de techniques de cryptologie sont basées sur un fait simple : nous ne connaissons pas de façon efficace (ie. polynômiale) de factoriser un nombre.

En particulier, la méthode RSA, utilisée dans les transmissions bancaires par exemple, repose sur ce fait : on fait transiter publiquement un très grand nombre qu'on sait être produit de deux nombres premiers. La connaissance d'un de ces nombres permettrait de casser la sécurité de la transmission, mais vu qu'on ne peut pas (dans l'état actuel des connaissances) trouver ce nombre en moins de centaines d'années, on considère que la transmission est sûre.

L'algorithme de Shor bouleverse totalement cette vision : en 1994, Shor trouve un algorithme quantique qui factorise un nombre en temps polynômial. Ainsi, un ordinateur quantique remettrait en cause une bonne partie de nos algorithmes de cryptologie. (Pas de panique cependant, il fournirait également une puissance d'encryptage plus importante !)

***Remarque 1.***
Dans un algorithme manipulant en entrée un nombre $N$, la taille de l'entrée est le nombre de bits de $N$, ie. $\log_2 N = O(\log N)$.

### 2.1.2 Squelette de l'algorithme

On peut décrire l'algorithme de Shor en plusieurs étapes :

---

Shor

**Entrée** : nombre $N$ à factoriser. On sait que $N$ n'est pas premier.
**Sortie** : 1 facteur non-trivial de $N$

1. Si $N$ est pair, retourner 2.
2. Déterminer si $N = a^b$ pour des entiers $a \geq 1$ et $b \geq 2$. Si c'est le cas, retourner $a$.
3. Choisir un entier $x$, $1 \leq x \leq N - 1$.
4. Si le $pgcd(x, N) > 1$, alors retourner $pgcd(x, N)$.
5. Sinon, chercher l'ordre $r$ de $x$ dans $\mathbb{Z}/N\mathbb{Z}$, c'est-à-dire que $r$ est le plus petit entier non-nul tel que $x^r \equiv 1 \mod N$.
6. Si $r$ est impair ou $x^{r/2} \equiv -1 \mod N$, alors recommencer avec un autre $x$.
7. Calcule $pgcd(x^{r/2} - 1, N)$ et $pgcd(x^{r/2} + 1, N)$. L'un des deux est un facteur non trivial.

---

Quelques justifications s'imposent ici pour voir que l'algorithme est bien polynômial en $\log N$, et qu'il calcule bien ce qu'on veut.

Les deux premières étapes sont là pour vérifier que $N$ est un entier impair avec plus d'un facteur premier. Ces étapes peuvent être réalisées en $O(1)$ et en $O(\log^3 N)$ respectivement. Cette propriété de $N$ sera essentielle pour l'étape 6.

L'étape 3 choisit un entier aléatoirement. Au vu de la quantité d'aléas présente par essence dans un ordinateur quantique, on se doute qu'obtenir un entier aléatoire n'est pas très compliqué. En effet, on peut faire simplement la manipulation suivante : on applique la porte d'Hadamard au 1-qubit $|0\rangle$. On a par définition :

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Il suffit alors de mesurer l'état de $H|0\rangle$ pour obtenir $|0\rangle$ ou $|1\rangle$ avec une probabilité 1/2. À partir de cela, il est facile de générer aléatoirement un nombre entre 1 et $N-1$ en $O(\log N)$ (on tire aléatoirement chaque chiffre de son écriture en base 2).

L'étape 4 retourne éventuellement directement un facteur non trivial. Le *pgcd* se calcule classiquement en $O(\log^3 N)$ à l'aide de l'algorithme d'Euclide.

Jusque là, nous n'avons pas utilisé la puissance particulière de l'ordinateur quantique (l'aléa peut être simulé sur un ordinateur classique). C'est dans l'étape 5 que tout se joue. En effet, on ne sait pas trouver l'ordre de $x$ en temps polynômial sur un ordinateur classique, alors qu'on le peut sur un ordinateur quantique avec une faible probabilité d'erreur. La clef de voûte de cette technique est la transformée de Fourier quantique qui se fait en $O(\log^3 N)$ contre $O(N \log N)$ classiquement. Nous allons étudier en détail cette étape dans la section suivante.

Dans l'étape 6, il y a aussi une probabilié d'erreur de l'algorithme, mais elle est très petite :

**[2.A] Théorème** ***(Probabilité d'échec)***
*Soit $N = \prod_{i=1}^{m} {p_i}^{\alpha_i}$ la décomposition en facteurs premiers de $N$. Alors avec les notations de l'algorithme :*

$$P(r \text{ est pair et } x^{r/2} \neq -1 \mod N) \geq 1 - \frac{1}{2^{m-1}}$$

**Preuve de [2.A].**
cf. [Sho99].

En particulier, cette probabilité d'erreur est inférieure à 1/2.

Dans l'étape 7, le calcul du *pgcd* se fait toujours en $O(\log^3 N)$. On sait par l'étape précédente que $x^{r/2} \neq -1 \mod N$, et on sait par définition de $r$ que $x^{r/2} \neq 1 \mod N$. Ainsi, aucun des deux facteurs n'est nul modulo $N$, et donc leurs *pgcd* avec $N$ donnent des facteurs non-triviaux, puisque leur produit divise $N$.

Finalement, on voit que ce qui permet à l'algorithme quantique d'être polynômial (à une petite probabilité d'erreur près) est simplement le fait de pouvoir calculer une transformée de Fourier efficacement.

Comme nous l'avions indiqué, cet algorithme a une (petite) probabilité d'erreur. Il faut distinguer cependant deux sources d'erreurs : dans l'étape 5 la probabilité d'erreur vient des lois de la mécanique quantique alors que dans l'étape 6, c'est purement arithmétique.

## 2.2 Algorithmes intermédiaires

Afin de réaliser l'étape 5, il nous faut construire 2 routines : l'exponentation modulaire et la transformée de Fourier quantique.

Nous décrivons ici l'algorithme tel qu'il a été énoncé par Shor, mais la version qu'on trouve plus souvent dans la littérature passe par un autre intermédiaire plus général et utile dans d'autres cas, qu'on appelle l'estimation de phase.

### 2.2.1 L'exponentiation modulaire

L'objet de ce paragraphe est de décrire un algorithme quantique polynômial pour l'exponentation : on prend en entrée $n$, $x$ et $a$ pour donner en sortie $(x^a \mod n)$. Ce sont des entiers, c'est-à-dire des vecteurs de la base de calcul.

On peut alors appliquer la méthode classique pour calculer les puissances, dite d'*exponentiation rapide*. Elle consiste à calculer les $x_i = (x^{2^i} \mod n)$ en mettant au carré successivement tant que $i \leq \log_2 a$. Ensuite, il reste à multiplier $x_{\lfloor \log_2 a \rfloor} = (x^{2^{\lfloor \log_2 a \rfloor}} \mod n)$ par quelques-uns des $x_i$ déjà calculés pour l'ajuster exactement à $(x^a \mod n)$.

L'implémentation quantique effective des opérations arithmétiques n'est pas triviale, mais elle n'est pas terrifiante. On se reportera à [Sho99] pour plus de détails.

Cette opération coûte donc clairement $O(\log n)$ multiplications de nombres plus petits que $n$. On dispose ensuite de l'algorithme classique qui multiplie deux entiers plus petits que $n$ en $O(\log^2 n)$. En fin de compte, la complexité est de $O(\log^3 n)$. En réalité, pour notre problème, on pourrait utiliser n'importe quel algorithme de multiplication, même naïf, tant qu'il reste polynômial en $\log n$. En particulier, on peut faire mieux que ce $O(\log^3 n)$ en faisant appel à des algorithmes de multiplication plus sophistiqués.

Pour que ce soit un algorithme exécutable sur un ordinateur quantique, il se doit d'être réversible comme on l'a déjà vu. Dans [Sho99], l'auteur décrit une méthode pour rendre réversible cet algorithme (et même tout algorithme). Nous ne nous attardons pas sur ce point et renvoyons à cette source pour plus de détails.

### 2.2.2 La transformée de Fourier quantique

Usuellement, la transformée de Fourier discrète prend $q$ complexes $(x_0, \dots, x_{q-1})$ où $q$ est fixé, et calcule $(y_0, \dots, y_{q-1})$ :

$$y_k = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} x_c \exp\left(\frac{2i\pi ck}{q}\right)$$

On transpose alors aisément ce calcul dans le cadre quantique. On se fixe un entier $q = 2^l$, $l \geq 1$, et $0 \leq a < q$. La transformation de Fourier agit sur $|a\rangle$ qui est un $l$-qubit de la manière suivante :

$$|a\rangle \quad \longmapsto \quad \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c\rangle \exp\left(\frac{2\pi ac}{q}\right) \tag{2.i}$$

En clair, on applique simplement l'opérateur unitaire qui a pour matrice $A_q$ dans la base de calcul où le terme $(a, c)$ de $A_q$ est, d'après (2.i) :

$$(A_q)_{(a,c)} = \frac{1}{\sqrt{q}} \exp\left(\frac{2\pi ac}{q}\right) \tag{2.ii}$$

Il s'agit donc de représenter cet opérateur comme une composition d'un nombre polynômial en $\log q$ de portes quantiques simples (voir §1.4).

Pour réaliser $A_q$, nous n'avons besoin que de quelques opérateurs de base.

Premièrement, on doit disposer des portes $H_j$ qui opèrent chacune sur le $j^{\text{ème}}$ qubit du $l$-qubit d'entrée et qui a pour matrice $H$ sur ce qubit (la porte d'Hadamard, cf. §1.3.2).

Ensuite, il suffit d'avoir les opérateurs $S_{j,k}$ qui opèrent chacun sur les $j^{\text{ème}}$ et $k^{\text{ème}}$ qubit d'entrée, avec $j < k$, et qui a pour matrice sur ces 2 qubits :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta_{j,k}} \end{pmatrix}, \text{ où } \theta_{j,k} = \frac{\pi}{2^{k-j}}$$

Et alors on obtient la porte $A_q$ en mettant bout à bout quelques-unes bien choisies de ces portes :

$$H_{l-1}\ ; S_{l-2,l-1} \xrightarrow{\text{puis}} H_{l-2}\ ; S_{l-3,l-1}\ ; S_{l-3,l-2} \xrightarrow{\text{puis}} H_{l-3} \dots H_1\ ; S_{0,l-1} \dots S_{0,1} \xrightarrow{\text{puis}} H_0$$

puis en prenant les qubits à l'envers dans la réponse de ce circuit (le premier qubit de sortie correspond en fait à la coordonnée de $A_q$ sur $|c\rangle$ pour $c = q-1$ et non pas $c = 0$, et ainsi de suite) – ceci se fait bien en un temps linéaire simplement avec un circuit de Swap puisqu'il faut intervertir $l/2$ couples.

Ainsi, on voit qu'on utilise $l + \sum_{i=1}^{l} i = O(l^2) = O(\log_2^2 q) = O(\log^2 q)$ portes, ce qui nous fait donc un algorithme polynômial en $\log q$. Reste à voir tout de même pourquoi cela calcule bien la transformée de Fourier qu'on attend.

Pour cela, regardons attentivement ce qu'il se passe sur un vecteur $|a\rangle = |a_{l-1} \dots a_0\rangle$, et plus précisément la coordonnée de $A_q |a\rangle$ sur un vecteur $|c\rangle = |c_{l-1} \dots c_0\rangle$. On s'attend à retrouver $(A_q)_{(c,a)}$.

Premièrement, regardons l'amplitude. Les $S_{j,k}$ ne la modifient pas, alors que chaque application de $H_j$ la multiplie par $\frac{1}{\sqrt{2}}$. Il y a $l = \log_2 q$ applications de $H_j$, et donc l'amplitude est multipliée par

$$(2^{-1/2})^{\log_2 q} = \sqrt{2^{-l}} = \frac{1}{\sqrt{q}}$$

Ainsi, le facteur multiplicatif dans (2.ii) est acquis.

Maintenant, regardons le changement de phase.

Pour commencer, regardons l'action des $H_j$. Pour $j$ fixé, $H_j$ a une action visible (multiplie par $-1$, ie. déphase de $\pi$) sur $|a\rangle$ seulement si $a_j$ vaut 1, et si on regarde le résultat sur une ligne où le $j^{\text{ème}}$ qubit de $c$ vaut 1. Mais comme on inverse les qubits en fin d'algorithme, cela se traduit par le fait que $c_{l-1-j} = 1$.

On peut voir cela en regardant l'action de $H_j$ sur un $l$-qubit global :

$$H_j = \underbrace{\mathbb{I}_2 \otimes \mathbb{I}_2 \dots \otimes \mathbb{I}_2}_{j-1 \text{ fois}} \otimes H \otimes \underbrace{\mathbb{I}_2 \otimes \dots \otimes \mathbb{I}_2}_{l-j-2 \text{ fois}}$$

Donc si on dispose de $|a\rangle = |a_{l-1} \dots a_0\rangle$, d'après les règles de calcul des produits tensoriels :

$$\begin{aligned} H_j |a\rangle &= \mathbb{I}_2 |a_{l-1}\rangle \otimes \mathbb{I}_2 |a_{l-2}\rangle \otimes \mathbb{I}_2 |a_{j+1}\rangle \otimes \dots H_j |a_j\rangle \otimes \mathbb{I}_2 |a_{j-1}\rangle \otimes \mathbb{I}_2 |a_0\rangle \\ &= |a_{l-1} \dots a_{j+1}\rangle \otimes H_j |a_j\rangle \otimes |a_{j-1} \dots a_0\rangle \end{aligned}$$

Ainsi, on peut diviser l'étude en deux cas :

| $\mathbf{a_j = 0}$ | $\mathbf{a_j = 1}$ |
|---|---|
| $H_j \|a\rangle = \frac{1}{\sqrt{2}} \|a\rangle + \frac{1}{\sqrt{2}} \|a_{l-1} \dots a_{j+1}\ 1\ a_{j-1} \dots a_0\rangle$ | $\begin{aligned} H_j \|a\rangle &= \frac{1}{\sqrt{2}} \|a_{l-1} \dots a_{j+1}\ 0\ a_{j-1} \dots a_0\rangle - \frac{1}{\sqrt{2}} \|a\rangle \\ &= \frac{1}{\sqrt{2}} \|a_{l-1} \dots a_{j+1}\ 0\ a_{j-1} \dots a_0\rangle + \frac{1}{\sqrt{2}} e^{i\pi} \|a\rangle \end{aligned}$ |

Ainsi, on remarque que le seul cas où on observe un déphasage du vecteur initial est lorsque $a_j = 1$ et qu'on regarde le résultat sur un vecteur dont le $j^{\text{ème}}$ qubit vaut $|1\rangle$. On traduit ceci par $c_{l-j-1} = 1$ dans notre cas. Le déphasage est de $\pi$, et c'est donc bien le résultat annoncé ci-dessus.

Observons à présent l'action de $S_{j,k}$. Par définition, on voit que $S_{j,k}$ déphase selon $|c\rangle$ de $\frac{\pi}{2^{k-j}}$ si $a_j\, c_{l-k-1} = 1$, et ne fait rien sinon.

On peut maintenant recoller les morceaux et voir ce que fait effectivement le total. On a un déphasage total $\Phi$ pour $A_q |a\rangle$ selon le vecteur $|c\rangle$ de la base de calcul de :

$$\begin{aligned} \Phi_{q,a,c} &= \sum_{j=0}^{l} \pi a_j\, c_{l-j-1} + \sum_{0 \le j < k < l} \frac{\pi}{2^{k-j}} a_j\, c_{l-k-1} \\ &= \sum_{0 \le j \le k < l} \frac{\pi}{2^{k-j}}\, a_j\, c_{l-1-k} \\ &\overset{k \leftarrow l-k-1}{=} \sum_{0 \le j+k < l} 2\pi \frac{2^j 2^k}{2^l}\, a_j\, c_k \\ &= \frac{2\pi}{2^l} \sum_{j=0}^{l-1} 2^j\, a_j \sum_{k=0}^{l-1} 2^k\, c_k \\ &= \frac{2\pi ac}{q} \end{aligned}$$

Or $(A_q)_{(a,c)} = (A_q)_{(c,a)} = \frac{1}{\sqrt{q}}\, e^{i\Phi_{q,a,c}}$, donc on a bien montré que $\boxed{(A_q)_{(a,c)} = \frac{1}{\sqrt{q}} \exp\left(\frac{2\pi ac}{q}\right)}$ □

Intuitivement, on voit bien où le calcul quantique permet de gagner du temps : lorsqu'on applique $H_j$ ou $S_{j,k}$, l'opérateur agit sur toutes les composantes en même temps. Plutôt qu'un long discours, voici la matrice de $H_2$ lorsque $l = 3$, où le vide représente des zéros. On voit bien que des calculs sont faits en parallèle (2 résultats superposés pour chaque composante en entrée ; plus les opérateurs s'appliquent au vecteur de départ, plus l'état est superposé, et donc "plus on parallélise") :

$$\mathbb{I}_2 \otimes H \otimes \mathbb{I}_2 = \left(\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & & & & \\ 0 & 1 & 0 & 1 & & & & \\ 1 & 0 & -1 & 0 & & & & \\ 0 & 1 & 0 & -1 & & & & \\ \hline & & & & 1 & 0 & 1 & 0 \\ & & & & 0 & 1 & 0 & 1 \\ & & & & 1 & 0 & -1 & 0 \\ & & & & 0 & 1 & 0 & -1 \end{array}\right)$$

Malgré tout, il serait abusif de dire que cet algorithme calcule **effectivement** la transformée de Fourier de $|a\rangle$. En effet, si on sait que $A_q |a\rangle$ est bien dans un état qui représente sa transformée de Fourier, on ne peut pas pour autant en extraire les différentes harmoniques, toujours à cause du problème de la mesure qui est probabiliste et qui projetterait le vecteur sur un vecteur de la base de calcul. Ainsi, elle ne peut servir que comme étape "intermédiaire" dans un autre algorithme. C'est précisément la manière dont on s'en sert dans l'algorithme de recherche de l'ordre comme on va le voir.

## 2.3 Recherche de l'ordre d'un élément

### 2.3.1 L'algorithme

Il est temps de faire effectivement l'algorithme pour l'étape 5 :

---

Order-Finding
**Entrée** : un entier $N$, et un entier $x$, $1 \leq x \leq N$.
**Sortie** : $r$ l'ordre de $x$ dans $\mathbb{Z}/N\mathbb{Z}$.

---

La taille de l'entrée est donc $\log N$.

Premièrement, choisissons $l$ tel que $q = 2^l$ soit la puissance de 2 telle que $n^2 \leq q < 2n^2$. On prend ensuite un $2l$-qubit, auquel on pense comme *deux registres d'un $l$-qubit chacun*, et que l'on place dans un état de superposition uniforme sur le premier registre :

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} \underbrace{|a\rangle}_{l\text{-qubit}} \underbrace{|0\rangle}_{l\text{-qubit}}$$

On peut réaliser cette superposition par exemple à l'aide d'une porte d'Hadamard généralisée $H_n = H^{\otimes n}$, en prenant $H_n |0\rangle$.

Ensuite, on utilise l'algorithme d'exponentiation modulaire pour calculer $x^a \mod N$ dans le second registre. Ce calcul peut être fait de façon réversible puisqu'on conserve $a$ dans le premier registre. On obtient donc le nouvel état :

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle |x^a \mod N\rangle$$

Maintenant, on applique la transformée de Fourier $A_q$ au premier registre. C'est bien une opération polynômiale en $\log N$ puisque $2 \log N \leq 2 \log q \leq \log 2 + 2 \log N$. D'après (2.i), on se retrouve à présent dans l'état :

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} \exp\left(\frac{2i\pi ac}{q}\right) |c\rangle |x^a \mod N\rangle \tag{2.iii}$$

À présent, les calculs sont terminés et on fait une mesure de l'état, récoltant un certain $|c\rangle\,|x^a \mod N\rangle$.

Il ne reste plus qu'une étape de gestion des données (qui peut être réalisée sur un ordinateur classique). On cherche $d$ et $r'$ tels que $\left|\frac{c}{q} - \frac{d}{r'}\right| \leq \frac{1}{2q}$ (on peut utiliser un algorithme de fractions continues pour cela, qui s'exécute en temps polynômial).

Si $d$ et $r$ (qu'on ne connaît pas) sont premiers entre eux, alors $r = r'$ et on a trouvé l'ordre de $x$. Sinon, il faut recommencer l'algorithme (pour savoir si on doit recommencer, on teste simplement si $x^{r'} \equiv 1 \mod N$).

Comme on va le voir ci-dessous, la probabilté de recommencer est telle que l'on est certain à une très faible probabilité d'erreur près que le calcul va terminer au bout de $O(\log\log r)$ exécutions. Et on sait que $r$ divise $N$ par le théorème de Lagrange, donc c'est aussi un $O(\log\log N)$.

Finalement, on a bien un algorithme qui s'exécute en temps polynômial et qui prétend trouver l'ordre de l'élément $x$. Reste à voir pourquoi et comment dans la section suivante.

### 2.3.2 Preuve de correction

On se place après la mesure et il faut montrer que les opérations qu'on fait ensuite mènent bien à $r$ avec bonne probabilité.

Pour plus de clarté, nous allons faire la preuve un petit peu à l'envers par rapport à [Sho99].

On définit $\{rc\}_q \in \mathbb{N}$ comme la quantité telle que $(rc \equiv \{rc\}_q \mod q)$ et $\{rc\}_q \in ]-q/2, q/2]$.

Supposons à présent qu'après la mesure, on ait :

$$-\frac{r}{2} \leq \{rc\}_q \leq \frac{r}{2} \tag{2.iv}$$

c'est-à-dire qu'il existe un entier $d$ tel que :

$$-\frac{r}{2} \leq rc - qd \leq \frac{r}{2}$$

ou encore

$$\left|\frac{c}{q} - \frac{d}{r}\right| \leq \frac{1}{2q} \leq \frac{1}{2n^2} \tag{2.v}$$

On connaît $c$ et $q$ qui sont fixés. Supposons qu'on ait deux fractions différentes $\frac{d}{r}$ et $\frac{d'}{r'}$ qui vérifient (2.v) avec $r, r' < N$. Alors on a :

$$\left|\frac{d}{r} - \frac{d'}{r'}\right| \geq \frac{1}{rr'} > \frac{1}{N^2}$$

et d'autre part

$$\begin{aligned}\left|\frac{d}{r} - \frac{d'}{r'}\right| &\leq \left|\frac{d}{r} - \frac{c}{q}\right| + \left|\frac{c}{q} - \frac{d}{r}\right| \\ &\leq \frac{1}{N^2}\end{aligned}$$

Ceci est absurde, donc il existe au plus une fraction ayant un dénominateur $r < N$ qui vérifie (2.v).

On admettra qu'on peut trouver une telle fraction en temps polynômial grâce à un algorithme utilisant les fractions continues, on la note $\frac{d_0}{r_0}$.

Dans cette situation, si $d$ et $r$ sont premiers entre eux, alors $\frac{d}{r}$ est irréductible et l'algorithme a donc donné la bonne fraction : $d = d_0$ et $r = r_0$, on a bien trouvé $r$. Dans le cas où $d$ et $r$ ne sont pas premiers entre eux, l'algorithme a donné une version réduite de $d/r$ et donc $d_0 < d$ et $r_0 < r$ : on n'a pas trouvé $r$ et il faut recommencer depuis le départ.

Si on accepte qu'on est dans le cas de (2.iv), regardons quelle probabilité on a d'être dans le cas où $d$ et $r$ sont premiers entre eux.

Pour voir que cette étude qui a pris comme postulat (2.iv) reflète bien la réalité, il faut estimer la probabilité qu'on a d'arriver dans la bonne situation. Comptons donc le nombre d'états $|c\rangle\big|x^k \mod N\big\rangle$ qui nous mènent dans le cas où on obtient bien la valeur de $r$ au bout du compte.

Il y a $\varphi(r)$ valeurs de $d$ telles que $d$ est premier avec $r$ par définition de $\varphi$, où $\varphi$ est la fonction d'Euler. Et pour chacune de ces valeurs de $d$, il y a une valeur de $c$ telle que (2.v) soit vérifiée. Pour chacune de ces valeurs de $c$, il y a $r$ valeurs possibles pour $(x^k \mod N)$ puisque $r$ est l'ordre de $x$.

Donc il y a $E = r\varphi(r)$ états possibles de $|c\rangle|x^k \mod N\rangle$ qui nous permettent de trouver $r$. Reste à voir dans quelle mesure la condition (2.iv) est vérifiée.

D'après (2.iii), la probabilité de mesurer un état $|c\rangle|x^k \mod N\rangle$ est :

$$P_{c,k} = \left| \frac{1}{q} \sum_{a:x^a \equiv x^k [N]} \exp\left(\frac{2i\pi ac}{q}\right) \right|^2$$

Et comme l'ordre de $x$ est $r$, cette somme porte en fait sur les $a$ tels que $a \equiv k \mod r$, puisque $x^{a-k} \equiv 1 \mod N$. Alors si on écrit $a = br + k$, on trouve :

$$\begin{aligned} P_{c,k} &= \left| \frac{1}{q} \sum_{b=0}^{\lfloor \frac{q-k-1}{r} \rfloor} \exp\left(\frac{2i\pi(br+k)c}{q}\right) \right|^2 \\ &= \left| \frac{1}{q} \sum_{b=0}^{\lfloor \frac{q-k-1}{r} \rfloor} \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right) \right|^2 \end{aligned}$$

Il s'agit maintenant d'évaluer cette probabilité. Pour cela, on va faire une estimation série-intégrale :

$$\begin{aligned} \Delta &= \left| \sum_{b=0}^{\lfloor \frac{q-k-1}{r} \rfloor} \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right) - \int_0^{\lfloor \frac{q-k-1}{r} \rfloor} \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right) \mathrm{d}b \right| \\ &= \left| \sum_{j=0}^{\lfloor \frac{q-k-1}{r} \rfloor} \left[ \exp\left(\frac{2i\pi j\{rc\}_q}{q}\right) - \int_j^{j+1} \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right) \mathrm{d}b \right] \right| \\ &= \left| \sum_{j=0}^{\lfloor \frac{q-k-1}{r} \rfloor} \left[ \exp\left(\frac{2i\pi j\{rc\}_q}{q}\right) \left( \int_0^1 1 - \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right) \mathrm{d}b \right) \right] \right| \end{aligned}$$

Or $|\{rc\}_q| \leq q/2$, et donc on fait "moins d'un demi-tour", et $\left|1 - \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right)\right| \leq \left|1 - \exp\left(\frac{2i\pi\{rc\}_q}{q}\right)\right|$ pour $b \in [0,1]$. Donc :

$$\begin{aligned} \Delta &\leq \left| \sum_{j=0}^{\lfloor \frac{q-k-1}{r} \rfloor} \left[ \exp\left(\frac{2i\pi j\{rc\}_q}{q}\right) \left(1 - \exp\left(\frac{2i\pi\{rc\}_q}{q}\right)\right) \right] \right| \\ &\leq \left| \left\lfloor \frac{q-k-1}{r} \right\rfloor \left(1 - \exp\left(\frac{2i\pi\{rc\}_q}{q}\right)\right) \right| \end{aligned}$$

Finalement, on a donc :

$$\frac{1}{q} \sum_{b=0}^{\lfloor \frac{q-k-1}{r} \rfloor} \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right) = \frac{1}{q} \int_0^{\lfloor \frac{q-k-1}{r} \rfloor} \exp\left(\frac{2i\pi b\{rc\}_q}{q}\right) \mathrm{d}b + O\left( \frac{\lfloor \frac{q-k-1}{r} \rfloor}{q} \left( \exp\left(\frac{2i\pi\{rc\}_q}{q}\right) - 1 \right) \right)$$

À l'aide d'un développement limité, on voit que si $|\{rc\}_q| \leq r/2$, alors le terme correctif est en $O(1/q)$. On l'ignorera donc à partir de maintenant. Et après changement de variable on obtient :

$$\frac{1}{r} \int_0^{\lfloor \frac{q-k-1}{r} \rfloor r/q} \exp\left(\frac{2i\pi u\{rc\}_q}{r}\right) \mathrm{d}u$$

Comme $k < r$, on peut approximer la borne supérieure de l'intégrale par 1 en ne faisant qu'une erreur en $O(1/q)$. Donc on se retrouve à traiter :

$$I = \frac{1}{r} \int_0^1 \exp\left(\frac{2i\pi u\{rc\}_q}{r}\right) \mathrm{d}u$$

Une étude de fonction immédiate montre que $|I| \geq \frac{2}{\pi r}$ pour $|\{rc\}_q| \leq r/2$. Et donc dans le cas où $|\{rc\}_q| \leq r/2$ (qui est (2.iv)), on a $P_{c,k} \geq \frac{4}{\pi^2 r^2} \geq \frac{1}{3r^2}$ pour $q$ assez grand, c'est-à-dire pour $N$ assez grand.

Finalement, on vient de montrer qu'asymptotiquement, la probabilité de mesurer un état tel que (2.iv) soit vérifiée est plus grande que $\frac{1}{3r^2}$. Le fait que ce soit asymptotique n'est nullement gênant : on applique l'algorithme de factorisation pour de grands nombres, sinon un test direct est faisable.

Si on en revient à la conclusion précédente, la probabilité d'être dans le cas où on trouve le bon $r$ est donc supérieure, asymptotiquement, à $\frac{E}{3r^2} = \frac{r\varphi(r)}{3r^2} = \frac{\varphi(r)}{3r}$. Or un théorème d'arithmétique nous dit qu'il existe une constante $\delta$ telle que $\frac{\varphi(r)}{r} \geq \frac{\delta}{\log\log r}$. Cela prouve que l'on trouve $r$ effectivement au moins une fraction $\frac{\delta}{3\log\log r}$ du temps. Donc en répétant l'expérience seulement $O(\log\log r) = O(\log\log N)$ fois, on est assuré d'une bonne probabilité de succès.

# 3 Algorithme de Grover

Le cœur de l'exposé reposant sur l'algorithme de Shor, nous ne donnerons qu'un aperçu rapide de l'algorithme de Grover en nous limitant à énoncer les (belles) idées qu'il fait intervenir.

## 3.1 Position du problème

On dispose d'un ensemble de $N$ éléments désordonnés, parmi lesquels on veut trouver un élément particulier (on peut facilement généraliser l'algorithme pour trouver $M \leq N$ éléments).

Sur un ordinateur classique, dans une telle situation, on ne peut trouver l'élément en moins de $N/2$ opérations en moyenne. Ainsi, la recherche est linéaire : on regarde chaque objet jusqu'à tomber sur le bon.

L'algorithme de Grover permet de trouver cet élémént avec une (très) forte probabilité de succès en temps $O(\sqrt{N})$.

## 3.2 Algorithme de Grover

Le point essentiel de l'algorithme de Grover est la parallélisation de calculs que permet les ordinateurs quantiques. Comme d'habitude, on n'obtient pas "tous" les résultats puisqu'ils ne sont pas mesurables, mais ils sont calculés dans l'état.

Tout d'abord, on se munit de deux registres. Le premier est de taille $\log N$, le deuxième est simplement un qubit. On place le premier registre dans un état de superposition (qu'on a déjà croisé) :

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$$

On définit la fonction indicatrice de la solution :

$$f(i) = \begin{cases} 1 & \text{si } i \text{ est l'élément cherché } i_0 \\ 0 & \text{sinon} \end{cases}$$

L'algorithme de Grover consiste à construire un opérateur unitaire $U_f$ tel que :

$$U_f(|i\rangle\,|j\rangle) = |i\rangle\,|j \oplus f(i)\rangle$$

où $\oplus$ désigne le **NAND** (ie. la somme modulo 2).

Plaçons le deuxième registre dans l'état $|\psi_2\rangle = H\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. Alors :

$$\begin{aligned} U_f(|i\rangle\,|\psi_2\rangle) &= \frac{U_f(|i\rangle\,|0\rangle) - U_f(|i\rangle\,|1\rangle)}{\sqrt{2}} \\ &= \frac{|i\rangle\,|f(i)\rangle - |i\rangle\,|1 \oplus f(i)\rangle}{\sqrt{2}} \\ &= (-1)^{f(i)}\,|i\rangle\,|\psi_2\rangle \end{aligned}$$

Du coup, si on applique ce calcul à notre état $|\psi_1\rangle$, on obtient :

$$\begin{aligned} U_f(|\psi_1\rangle\,|\psi_2\rangle) &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} U_f(|i\rangle\,|\psi_2\rangle) \\ &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{f(i)}\,|i\rangle\,|\psi_2\rangle \end{aligned}$$

Ainsi, on a *différencié* l'état recherché puisque c'est la seule coordonnée de $|R\rangle = U_f(|\psi_1\rangle\,|\psi_2\rangle)$ qui est négative. On a donc bien évalué la fonction $f$ en parallèle sur les $N$ valeurs. Bien entendu, comme toujours, on n'a toujours pas accès à l'élément recherché puisqu'une mesure ne nous donnerait aucune

information (on a une chance sur $N$ de tomber sur la bonne valeur, puisque toutes les coordonnées ont même module).

La suite de l'algorithme de Grover consiste à manipuler cet état $|R\rangle$ pour accentuer l'amplitude de la coordonnée négative au détrimens des positives. On peut montrer qu'au bout de $O(\sqrt{N})$ opérations, on a assez accentué la coordonnée cherchée pour que la probabilité de l'avoir dans une mesure soit excellente.

On se réfèrera à [Gro96] pour le premier article qui décrit l'algorithme, à [LMP03] et [NC00, p. 248] pour une version mieux expliquée accompagnée d'illustrations parlantes. [Gro01] explique la génèse de cet algorithme et le cheminement pour arriver jusqu'à lui.

## Références


[Fim00] Pierre Fima. Portes et circuits quantiques. Accessible sur le site du groupe de travail : http ://www.dma.ens.fr/edition/NotesGDT/GT_calculquantique, ≈ 2000.

[Gro96] L.K Grover. A fast quantum mechanical algorithm for database search. *Proc. 28th annual ACM Symposium on the Theory of computing (STOC)*, pages pp. 212–219, May 1996.

[Gro01] L.K Grover. From schrödinger's equation to the quantum search algorithm. Trouvable sur arXiv, 2001.

[LMP03] C. Lavor, L.R.U Mansur, and R. Portugal. Grover's algorithm : quantum database search. Trouvable sur arXiv, 2003.

[NC00] Michael L. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information.* Cambridge, 2000.

[Sho99] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2) :pp. 303–332, Jun. 1999.