TD 3 - Complexité

Informatique quantique pour la recherche opérationnelle, S5.

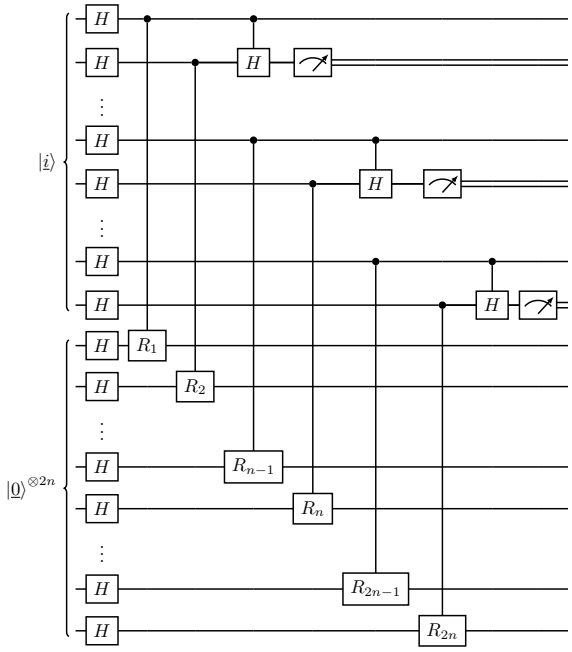
2023

Exercice 1 — *Complexité d'un circuit*

Quelle est la complexité de l'algorithme suivant ? On considèrera comme élémentaires les opérations classiques suivantes: affectation, Si, incrémenter une variable, Comparer deux entiers. On considèrera que créer ou appliquer une porte de taille fixe est aussi une opération élémentaire.

Entrées: Un entier $x \in \mathbb{N}$, avec $2n$ bits.
Sorties: Un entier y

1: Créer le circuit ci-dessous
2: $y \leftarrow 0$
3: **Pour** i de 1 à x **Faire**
4: Exécutez le circuit avec i en entrée
5: **Si** tous les qbits mesurés donnent $|0\rangle$ **Alors**
6: $y \leftarrow y + 1$
7: **Renvoyer** y



► **Correction**

On peut commencer par compter la complexité du circuit.

Le circuit a $4n$ qbits.

Ce circuit a $4n$ portes H, suivi de $2n$ portes R_i controlées, n portes H contrôlées et n portes de mesures. Donc $O(n)$ portes.

La profondeur du circuit est 4 si on considère qu'on peut mettre deux portes R_i contrôlées en parallèle, même si elles se croisent sur le circuit (si ce n'est pas le cas, on pourrait utiliser des portes SWAP pour remettre les qbits dans le bon ordre, mais cela augmenterait la profondeur du circuit ; on pourrait aussi simplement remarquer que sur le dessin, la profondeur est $2n + 3$).

Exécuter le circuit prend donc un temps $O(n)$ pour initialiser les qbits et $O(1)$ pour parcourir le circuit. Créer le circuit prend un temps $O(n)$.

Si on regarde l'algorithme, la ligne 1 se fait donc en $O(n)$; la ligne 2 en $O(1)$; la ligne 3 est une boucle qui exécute au pire $2^{2n}-1$ itérations (car x a $2n$ bits donc est inférieur à $2^{2n}-1$) ; une itération de la ligne 4 se fait en $O(n)+O(1)$; une itération des lignes 5 et 6 se fait aussi en $O(n)+O(1)$ pour mesurer et comparer les n qbits en sortie du circuit. Donc, au total, l'algorithme se fait en $O(n+1+2^{2n}(n+1+n+1)) = O(n2^{2n})$.

Exercice 2 — *Complexité de (ID)*

Soit $\varepsilon > 0$, deux matrices unitaires A et B de tailles $2^n \times 2^n$ sont dites ε-distinguables si, il existe $|\varphi\rangle$, on a $\|A|\varphi\rangle - B|\varphi\rangle\| \geq \varepsilon$.

Montrez que le problème (ID) suivant est dans QMA.

Soit C un circuit quantique et $\varepsilon > 0$, est-ce que la matrice unitaire de C est ε-distinguable de l'identité ?

▶ **Correction**

Commençons par reprendre la définition de QMA.

- Si (ID) est dans QMA, alors il existe un algorithme quantique $\mathcal{A}$ polynomial que Arthur peut exécuter. Cet algorithme prend en entrée un p-qbit que Merlin lui donne et une entrée (C, ε) de (ID). Le p-qbit de merlin est de taille polynomiale : si le circuit C et ε sont de taille $O(n)$ alors p vaut n ou n^2 ou n^c pour une certaine constante c.
  - Pour toute entrée (C, ε) de (ID) dont la réponse est OUI, il existe un p-qbit de Merlin $|\varphi\rangle$ tel que $\mathcal{A}$ répond OUI avec $|\varphi\rangle$ et (C, ε) en entrée avec une probabilité supérieure à $2/3$.
  - Pour toute entrée (C, ε) de (ID) dont la réponse est NON, pour tout p-qbit de Merlin $|\varphi\rangle$, $\mathcal{A}$ répond NON avec $|\varphi\rangle$ et x en entrée avec une probabilité supérieure à $2/3$.

Pour montrer que le problème est dans QMA, il suffit donc de trouver un qbit témoin que Merlin peut donner à Arthur quand la réponse est OUI, et de s'assurer qu'Arthur répondra souvent NON quand la réponse est NON. Le qbit témoin semble évident ici: si la réponse est OUI, C est distinguable de l'identité. Donc il existe $|\varphi\rangle$ tel que $\|C|\varphi\rangle - |\varphi\rangle\| \geq \varepsilon$. Merlin peut donc donner $|\varphi\rangle$. Mais comment Arthur peut il vérifier si $\|C|\varphi\rangle - |\varphi\rangle\| \geq \varepsilon$.

Pendant le TD, j'avais proposé la correction suivante:

Arthur utilise le circuit C plein de fois. Et pour chaque exécution, il mesure le résultat. Il construit donc petit à petit un vecteur de résultat $|\phi\rangle$. Le coefficient $\alpha_x = \langle \underline{x}|\phi\rangle$ devant $|\underline{x}\rangle$ dans $|\phi\rangle$ est le nombre de fois qu'il a mesuré x. Donc, par exemple, si Arthur mesure 10 fois $|\underline{0}\rangle$, 3 fois $|\underline{18}\rangle$ et 11 fois $|\underline{103}\rangle$, alors $|\phi\rangle = 10|\underline{0}\rangle + 3|\underline{18}\rangle + 11|\underline{103}\rangle$.

Une fois qu'il a fait cette exécution k fois, il normalise $|\phi\rangle$ et on obtient un vecteur qui, si C est l'identité, devrait ressembler à $|\varphi\rangle$. J'avais ensuite proposé de regarder, pour chaque $|x\rangle$ tel que $\alpha_x > 0$, le coefficient $\beta_x = \langle \underline{x}|\varphi\rangle$ devant $|\underline{x}\rangle$ dans $|\varphi\rangle$. On obtient un second vecteur $|\varphi\rangle'$ qu'on normalise et on compare $\big\||\phi\rangle - |\varphi\rangle'\big\|$ avec ε. Si C est loin de l'identité, alors $|\phi\rangle$ et $|\varphi\rangle'$ devrait rapidement s'éloigner l'un de l'autre au fur et à mesure qu'Arthur répète l'exécution du circuit C.

Mais cette technique ne peut pas marcher. En effet, contrairement à ce que j'ai dit en TD, on ne peut pas calculer β_x. J'ai supposé que c'était possible mais cela implique d'utiliser l'algorithme qui est dans le cours et qui permet de simuler des circuits quantiques. Et pour utiliser cet algorithme, on a besoin d'un temps exponentiel.

On pourrait à la place estimer $|\varphi\rangle$ en mesurant le qbit de Merlin à chaque fois qu'on mesure la sortie du circuit C. Mais les deux mesures sont décorrélées. Il serait difficile de comparer le résultat avec $|\phi\rangle$ car on peut facilement ne pas avoir de chance et tomber sur des qbits de bases $|\underline{x}\rangle$ différents de ceux qui nous intéressent (ceux où $\alpha_x > 0$). Imaginons par exemple que Arthur mesure, comme plus haut, $|\underline{0}\rangle$, $|\underline{18}\rangle$ et $|\underline{103}\rangle$. Alors on souhaiterais connaître les coefficients β_0, β_{18} et β_{103}. Pour connaître ces coefficients avec précision, il faudrait mesurer $|\varphi\rangle$ suffisamment de

fois pour obtenir des informations précises sur ces 3 coefficients. On peut rapidement voir qu'il faudrait un nombre exponentiel de mesures de $|\varphi\rangle$ et de $C|\varphi\rangle$ pour espérer pouvoir les comparer.

Bref, cet algorithme ne semble finalement pas être la bonne solution.

L'article qui a décrit et étudié ce problème est `Janzing, D., Wocjan, P., & Beth, T. (2005). " Non-identity-check" is QMA-complete. International Journal of Quantum Information, 3(03), 463-473. https://arxiv.org/pdf/quant-ph/0305050`. La technique utilisée est la suivante:

C est une matrice unitaire. A ce titre, elle a 2^n valeurs propres qui sont toutes de norme 1. Autrement dit, C est diagonalisable et s'écrit sous la forme diagonale $\begin{pmatrix} e^{i\theta_1} & 0 & \dots & 0 \\ 0 & e^{i\theta_2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & e^{i\theta_{2^n}} \end{pmatrix}$

Si C est la matrice identité, alors toutes ces valeurs propres sont égales à 1. Sinon, si C est ε-distinguable de l'identité, il existe 2 valeurs propres $e^{i\theta_k}$ et $e^{i\theta_j}$ dont l'écart est significatif. Cet écart doit être de sorte que $\varepsilon \leq \sqrt{2\left(1-\cos\left(\frac{\theta_k-\theta_i}{2}\right)\right)}$.

Merlin va donc envoyer 2 vecteurs propres de C. Nommons les $|\varphi_k\rangle$ et $|\varphi_j\rangle$. En passant ces vecteurs propres dans le circuit C, Arthur obtiendrait $e^{i\theta_k}|\varphi_k\rangle$ et $e^{i\theta_j}|\varphi_j\rangle$. On va ensuite comparer les valeurs propres obtenues. Pour cela, on va utiliser la transformée de Fourrier inverse pour extraire les phases. La transformée de Fourrier classique extrait des périodes dans les fonctions périodiques. La transformée de Fourrier quantique peut être utilisée pour extraire des périodes dans des circuits (autrement dit la phase que le circuit introduit dans un qbit). C'est une technique classique en informatique quantique dit de procédure d'estimation de phase. `https://qiskit.org/textbook/ch-algorithms/quantum-phase-estimation.html`

Une fois les phases extraites, il suffit de comparer ε et $\sqrt{2\left(1-\cos\left(\frac{\theta_k-\theta_i}{2}\right)\right)}$ avec un circuit quantique qui simule ce calcul (ou alors on mesure θ_k et θ_i après l'estimation de phase et on fait le calcul en classique).

Exercice 3 — *QMA* $\subset$ *PSPACE*

On rappelle qu'on peut simuler un circuit quantique avec un algorithme exponentiel qui utilise un espace polynomial.

Montrez que si un problème est dans QMA, alors il existe un algorithme exponentiel qui utilise un espace polynomial et qui résout ce problème.

▶ **Correction**

Commençons par reprendre la définition de QMA dans le cas général. Soit un problème Π dans QMA. Ce problème est un problème de décision, dont la réponse est OUI ou NON.

Il existe un algorithme quantique $\mathcal{A}$ polynomial que Arthur peut exécuter. Cet algorithme prend en entrée un p-qbit que Merlin lui donne et une entrée x de Π. Posons $|x| = n$ la taille de l'entrée. Le p-qbit de merlin est de taille polynomiale : il utilise $p = p(n)$ qbits où $p(n)$ est un polynome en n (par exemple n^2 qbits, n^4 qbits, n^{12} qbits, ...).

Pour toute entrée x de Π dont la réponse est OUI, il existe un p-qbit de Merlin $|\varphi\rangle$ tel que $\mathcal{A}$ répond OUI avec $|\varphi\rangle$ et x en entrée avec une probabilité supérieure à $2/3$.

Pour toute entrée x de Π dont la réponse est NON, pour tout p-qbit de Merlin $|\varphi\rangle$, $\mathcal{A}$ répond NON avec $|\varphi\rangle$ et x en entrée avec une probabilité supérieure à $2/3$.

La question est : est-ce que Arthur peut se passer à la fois de Merlin, de l'ordinateur quantique et de la possibilité de se tromper, et résoudre le problème avec un espace polynomial?

Simuler Arthur

On sait que Arthur peut simuler l'exécution de son algorithme quantique. Connaissant un qbit $|\varphi\rangle$ et x, il peut simuler $\mathcal{A}(|\varphi\rangle, x)$ avec un algorithme probabiliste $\mathcal{B}$ et obtenir en sortie OUI ou NON avec la même probabilité que $\mathcal{A}$.

Première question: connaissant $|\varphi\rangle$ et x et un simulateur de $\mathcal{A}$, comment passer d'un algorithme probabiliste à une algorithme déterministe ? L'algorithme probabiliste $\mathcal{B}$ utilise un nombre aléatoire r et, en fonction de la valeur de ce nombre, l'algorithme simule une mesure du circuit quantique de $\mathcal{A}$ et répond OUI ou NON. Au lieu de générer un nombre aléatoire, on pourrait à la place tester

tous les nombres aléatoires possibles. Pour chaque valeur possible de r, on exécute l'algorithme et on compte le nombre de fois qu'on obtient OUI et NON. On sait que r prend une valeur uniforme dans un intervalle donné. Cet intervalle n'est pas infini (en effet l'algorithme $\mathcal{A}$ est polynomial, donc on ne peut pas passer un temps trop grand pour construire r, donc r ne peut pas être trop grand). Donc on peut effectivement énumérer toutes les valeurs de r. Cette énumération prend un temps conséquent mais on ne s'en soucis pas, on veut juste un espace polynomial. Et pour chaque exécution, le seul espace nécessaire est celui nécessaire à l'exécution de la simulation de $\mathcal{A}$ et le stockage de r, ce qui est bien en espace polynomial.

Cet argument suffit simplement à montrer que BQP est dans PSPACE. Mais pour faire pareil avec QMA, il faut réussir à simuler Merlin. Et nous verrons que cela semble impossible.

Simuler Merlin

On peut donc bien se passer de probabiliste et de quantique. Mais peut-on se passer de Merlin? A la place de Merlin, il suffit de tester tous les qbits possibles $|\varphi\rangle$ que Merlin pourrait envoyer. Encore une fois, il y a beaucoup de qbits possibles mais ce n'est pas grave, ce qui est important c'est l'espace nécessaire au stockage de $|\varphi\rangle$ et de l'exécution du simulateur. Mais cette question n'est pas anodine.

Cas simple

Il exite une classe plus simple que QMA, nommée QCMA, où Merlin a une contrainte supplémentaire. Il ne peut envoyer qu'un qbit de base $\left|\underline{y}\right\rangle$ où $y \in [\![0; 2^n - 1]\!]$. Dans ce cas, le problème est facilement réglé, il suffit d'énumérer tous les y possibles et de simuler l'algorithme de $\mathcal{A}$ avec $\left|\underline{y}\right\rangle$ et x en entrée.

Cas pas simple

Combien de place prend le qbit de Merlin dans le cas général ? Classiquement, on se dirait "2^p" puisqu'il y a p qbits. Mais dans ce cas, on est embêté, on ne peut pas énumérer les p-qbits de Merlin en espace polynomial. Comment outrepasser cette difficulté ? En TD, j'ai proposé la solution suivante: énumérer tous les circuits possibles que Merlin pourrait utiliser pour générer $|\varphi\rangle$. Mais cela ne suffit pas. Merlin pourrait vouloir générer un qbit arbitrairement précis. Par exemple $\sqrt{\varepsilon}\,|\underline{0}\rangle + \sqrt{1-\varepsilon}\,|\underline{1}\rangle$, avec $\varepsilon = 2^{-2^n}$; et il n'est pas certain qu'il existe un circuit capable de générer ce qbit et qui soit de taille polynomiale.

Pour outrepasser la difficulté de la précision d'un qbit, on peut utiliser un résultat de l'exercice 4. Dans cet exercice, on peut montrer que, connaissant deux qbits $|\varphi\rangle$ et $|\phi\rangle$ et une porte quantique U, alors $\||\varphi\rangle - |\phi\rangle\| = \|U\,|\varphi\rangle - U\,|\phi\rangle\|$. Cela signifie que, si deux qbits sont éloignés d'une certaine distance δ, alors si on passe les qbits dans un circuit quantique, les sorties seront éloignées d'une même distance δ. Ainsi, on n'est pas embêté par la précision infinie de Merlin, il suffit de discrétiser les qbits, de sorte à énumérer les qbits au plus à une distance δ les uns des autres. La sortie de l'algorithme $\mathcal{A}$ d'Arthur sera alors assez "proche" de celle qu'il aurait eu avec le qbit de Merlin (il aurait une sorte à une distance δ du résultat attendu). Bien entendu, on pourrait avoir une sortie dont la mesure donne la réponse OUI avec une probabilité très légèrement inférieure à $2/3$ (au pire de l'ordre de $2/3 - \delta$). Mais dans ce cas, il suffirait de répéter l'algorithme pour faire réaugmenter cette probabilité au delà de $2/3$. Bref, la précision n'est pas un problème.

Si cet argument ne vous convient pas, on peut facilement se convaincre que la précision n'est pas un problème avec l'article de `Grilo, A. B., Kerenidis, I., & Sikora, J. (2015, August). QMA with subset state witnesses. In International Symposium on Mathematical Foundations of Computer Science (pp. 163-174). Springer, Berlin, Heidelberg. https://arxiv.org/pdf/1410.2882`. Dans cet article, les auteurs prouvent que, si Merlin est restreint aux qbits de la forme $\left|\overline{S}\right\rangle = \frac{1}{\sqrt{S}} \sum_{x \in S} |\underline{x}\rangle$ où $S \subset [\![0; 2^n - 1]\!]$, alors la définition de QMA ne change pas (sous-entendu, si la réponse de x est OUI, alors parmi les qbit que Merlin peut envoyer à Arthur pour qu'il réponde OUI avec une probabilité supérieure à $2/3$, il y a des qbits de cette forme là). Et ces qbits sont très simples.

Enumérer les qbits $\left|\overline{S}\right\rangle$

Mais peut-on générer tous ces qbits avec un circuit ? Pas sûr. Pour être tout à fait honnête, je ne sais pas. Mais d'après tout ce que j'ai pu lire, je ne pense pas.

Peut-on énumérer tous ces qbits en espace polynomial ? On pourrait croire que oui. Si on nous donne un sous-ensemble S, il est facile de générer le vecteur $\left|\overline{S}\right\rangle$. Mais ce vecteur utilise un espace qui est de l'ordre de $|S|$. Et $|S|$ peut, au pire être égal à 2^n. On utilise donc encore un

espace exponentiel. Est-ce grave? Ne peut-on pas se passer du stockage de S ? Encore une fois, on pourrait croire que oui. En effet, grâce à notre simulateur, on pourrait simuler la sortie de $\mathcal{A}$ pour chaque qbit de base $|\underline{x}\rangle$, et par linéarité, en adaptant l'algorithme de simulation du cours, on pourrait simuler la réponse de $\mathcal{A}$ pour $|\overline{S}\rangle$. Sauf qu'il y a beaucoup d'ensembles S possibles. Vraiment beaucoup. Il y en a 2^{2^n}. Notre algorithme aurait donc une complexité en temps de l'ordre de 2^{2^n}. Et il existe un théorème en théorie de la complexité : si un algorithme utilise un espace x et si x est au moins linéaire, alors l'algorithme a une complexité en temps au pire de l'ordre de $O(2^x)$. Donc un algorithme qui utilise un temps 2^{2^n} utiliserait nécessairement un espace au moins de l'ordre de 2^n.

Essayez de faire un générateur qui se contente juste d'afficher en console les sous-ensemble de $[\![0; 2^n - 1]\!]$ en python, sans jamais les stocker en mémoire, et vous verrez que malgré cela, vous ne serez pas capable de faire cet algorithme en un espace meilleure que $O(2^n)$.

Comment faire ?

Bref, on semble coincés. Enumerer tous les qbits possibles de Merlin ne semble pas la bonne piste. Pour montrer que QMA est dans PSPACE, les techniques utilisées sont les suivantes :

- Montrer que QMA est dans la classe PP qui est la classe des problème qu'on peut résoudre avec un algorithme probabiliste qui se trompe strictement moins d'une fois sur 2. PP est une extension de BPP où la probabilité peut être aussi proche de $1/2$ qu'on le souhaite. PP peut facilement être démontré comme inclu dans PSPACE. Cependant, montrer que QMA est dans PP est compliqué.

- On peut voir dans `https://quantumcomputing.stackexchange.com/questions/24470/how-can-i-show-that` qu'il s'agit (encore) d'une question de valeurs propres. On veut connaître la probabilité que Arthur dise OUI. Supposons que son algorithme $\mathcal{A}$ consiste juste à exécuter un circuit et à répondre OUI s'il mesure le premier qbit en sortie du circuit. La question devient: quelle est, en fonction de $|\varphi\rangle$ la probabilité que la mesure du premier qbit soit 1 ?

  Soit M la matrice du circuit d'Arthur. Alors la probabilité qu'il réponde OUI est

  $$\|(|1\rangle\langle 1| \otimes I)M|x\rangle|\underline{0}\rangle|\varphi\rangle\|^2$$

  La partie qui suit M est l'entrée : le qbit de merlin, l'entrée x et des qbits de travail. Les multiplier par M donne la sortie. $|1\rangle\langle 1|$ est la matrice $\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}$ qui projet le vecteur $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ sur $\begin{pmatrix}0\\ \beta\end{pmatrix}$. Donc multiplier la sortie du circuit par $(|1\rangle\langle 1| \otimes I)$ supprime tous les qbits dont la première composante n'est pas 0. La norme du vecteur résultant est la probabilité de mesurer 1 sur le premier qbit.

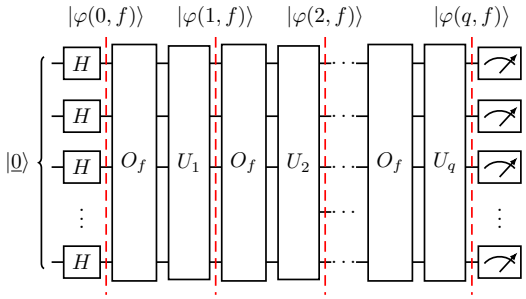
  On cherche donc la valeur de $|\bar{\varphi}N|\varphi\rangle|^2$ maximum pour tout $|\varphi\rangle$ où N est la matrice $\langle\underline{0}|\langle x|M^{-1}(|1\rangle\langle 1| \otimes I)M|x\rangle|\underline{0}\rangle$.

  Chose intéressante : si $|\varphi\rangle$ est un vecteur propre de N, de valeur propre α, alors $|\langle\varphi|N|\varphi\rangle|$ est $|\alpha|$. Et on peut montrer que N est diagonalisable, donc tout vecteur peut se décomposer en combinaison de vecteurs propres ; $|\langle\varphi|N|\varphi\rangle|$ est donc une combinaison linéaire des valeurs propres de N. Ainsi, il est impossible de faire mieux que la plus grande des valeurs propres de N. On cherche donc la plus grande valeur propre de N et on obtient alors immédiatement la probabilité d'acceptation maximum d'Arthur, sans avoir besoin ni de simuler Arthur ni de simuler Merlin.

  On a toujours une question : peut-on trouver cette valeur propre en espace polynomial ? A priori oui. Cette question est encore une fois malheureusement assez technique ; je n'ai pas trouvé d'argument simple pour expliquer comment faire. Vous trouverez des détails dans le lien précédent et les articles vers lequel il pointe.

Exercice 4 — *Finissons le cours*

On considère le circuit suivant où $U_1, U_2, \ldots, U_q$ sont des portes quantiques quelconques. Pour simplifier les calculs, on suppose qu'il n'y a pas d'autres qbits dans le circuit (dit autrement, tout autre qbit est un qbit de calcul qui n'intervient pas dans la suite).



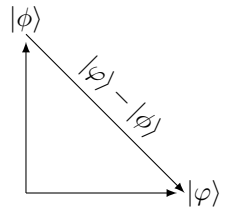
On suppose que, pour tout $s \in [\![0; 2^n - 1]\!]$ et pour toute fonction f telle que $f(s) = 1$ et $f(x \neq s) = 0$, ce circuit vérifie $|\varphi(q, f)\rangle = |\underline{s}\rangle$.

Dans la suite, on note f_s cette fonction. On note également zr la fonction nulle partout.

1. (a) Soient deux qbits $|\varphi\rangle$ et $|\phi\rangle$ orthogonaux. Montrez que $\||\varphi\rangle - |\phi\rangle\| = \sqrt{2}$.

▶ **Correction**

$|\varphi\rangle$ et $|\phi\rangle$ sont tous les deux de norme 1 et leur produit scalaire vaut 0.
On voit assez bien graphiquement que la norme de leur différence est $\sqrt{2}$.



$$\begin{aligned}\||\varphi\rangle - |\phi\rangle\| &= \sqrt{(\langle\varphi| - \langle\phi|) \cdot (|\varphi\rangle - |\phi\rangle)} \\ &= \sqrt{\langle\varphi|\varphi\rangle - \langle\phi|\varphi\rangle - \langle\varphi|\phi\rangle + \langle\phi|\phi\rangle} \\ &= \sqrt{1 - 0 - 0 + 1} \\ &= \sqrt{2}\end{aligned}$$

(b) Soit $s \in [\![0; 2^n - 1]\!]$. Montrez que $\frac{1}{2^n-1} \sum_{y \neq s} \big\||\underline{y}\rangle - |\underline{s}\rangle\big\| = \sqrt{2}$.

▶ **Correction**

On rappelle que, pour tout $y \neq s$, les qbits de base $|\underline{y}\rangle$ et $|\underline{s}\rangle$ sont orthogonaux. D'après la question précédente, on a donc $\big\||\underline{y}\rangle - |\underline{s}\rangle\big\| = \sqrt{2}$.

$$\begin{aligned}\frac{1}{2^n - 1} \sum_{y \neq s} \big\||\underline{y}\rangle - |\underline{s}\rangle\big\| &= \frac{1}{2^n - 1} \sum_{y \neq s} \sqrt{2} \\ &= \frac{1}{2^n - 1} (2^n - 1)\sqrt{2} \\ &= \sqrt{2}\end{aligned}$$

(c) Montrez que

$$\frac{1}{\sqrt{2}} \frac{2^n - 1}{2^n} \leq \frac{1}{2^n} \sum_{y=0}^{2^n-1} \big\||\underline{y}\rangle - |\varphi(q, zr)\rangle\big\|$$

Indice : Considérez s tel que $|\underline{s}\rangle$ minimise $\big\||\underline{y}\rangle - |\varphi(q, zr)\rangle\big\|$ et partez du résultat de la question précédente.

▶ **Correction**

Comme le suggère l'indice, on pose s qui minimise $\| |\underline{s}\rangle - |\varphi(q, zr)\rangle \|$.
Donc $\| |\underline{s}\rangle - |\varphi(q, zr)\rangle \| \leq \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\|$ pour tout $y \neq s$.
D'après l'inégalité triangulaire de la norme, on a:

$$\begin{aligned}
\frac{1}{2^n - 1} \sum_{y \neq s} \left\| |\underline{y}\rangle - |\underline{s}\rangle \right\| &\leq \frac{1}{2^n - 1} \sum_{y \neq s} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\| + \| |\varphi(q, zr)\rangle - |\underline{s}\rangle \| \\
&\leq \frac{1}{2^n - 1} \sum_{y \neq s} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\| + \| |\underline{s}\rangle - \varphi(q, zr)\rangle \| \\
&\leq \frac{1}{2^n - 1} \sum_{y \neq s} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\| + \left\| |\underline{y}\rangle - \varphi(q, zr)\rangle \right\| \\
&\leq \frac{2}{2^n - 1} \sum_{y \neq s} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\|
\end{aligned}$$

D'après la question précédente

$$\begin{aligned}
\sqrt{2} &\leq \frac{2}{2^n - 1} \sum_{y \neq s} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\| \\
\frac{1}{\sqrt{2}}(2^n - 1) &\leq \sum_{y \neq s} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\| \\
\frac{1}{\sqrt{2}} \frac{2^n - 1}{2^n} &\leq \frac{1}{2^n} \sum_{y \neq s} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\|
\end{aligned}$$

Puisque la norme est toujours positive, on peut rajouter $\| |\underline{s}\rangle - |\varphi(q, zr)\rangle \|$ dans la somme.

$$\frac{1}{\sqrt{2}} \frac{2^n - 1}{2^n} \leq \frac{1}{2^n} \sum_{y=0}^{2^n - 1} \left\| |\underline{y}\rangle - |\varphi(q, zr)\rangle \right\|$$

2. (a) Soient deux qbits $|\varphi\rangle$ et $|\phi\rangle$ et une porte quantique U. Montrez que $\| |\varphi\rangle - |\phi\rangle \| = \| U|\varphi\rangle - U|\phi\rangle \|$

▶ **Correction**

$$\begin{aligned}
\| U|\varphi\rangle - U|\phi\rangle \| &= \| U(|\varphi\rangle - |\phi\rangle) \| \\
&= \sqrt{(\langle\varphi| - \langle\phi|) \cdot U^{-1} \cdot U \cdot (\langle\varphi| - \langle\phi|)} \\
&= \sqrt{(\langle\varphi| - \langle\phi|) \cdot (\langle\varphi| - \langle\phi|)} \\
&= \| |\varphi\rangle - |\phi\rangle \|
\end{aligned}$$

(b) Montrez que, pour tout $i < q$ et $y \in [\![0; 2^n - 1]\!]$, on a

$$\| |\varphi(i+1, zr)\rangle - |\varphi(i+1, f_y)\rangle \| \leq \left\| |\varphi(i, zr)\rangle - O_{f_y} |\varphi(i, zr)\rangle \right\| + \| |\varphi(i, zr)\rangle - |\varphi(i, f_y)\rangle \|$$

▶ **Correction**

Pour obtenir $|\varphi(i+1, zr)\rangle$, on fait passer $|\varphi(i, zr)\rangle$ dans la porte O_{zr} puis dans la porte U_{i+1}. Puisque zr vaut 0 partout, alors O_{zr} est l'identité. Donc $|\varphi(i+1, zr)\rangle = U_{i+1} |\varphi(i, zr)\rangle$.

Pour obtenir $|\varphi(i+1, f_y)\rangle$, on fait passer $|\varphi(i, f_y)\rangle$ dans la porte O_{f_y} puis dans la porte U_{i+1}. Donc $|\varphi(i+1, f_y)\rangle = U_{i+1} O_{f_y} |\varphi(i, f_y)\rangle$.

$$\begin{aligned} \| |\varphi(i+1, zr)\rangle - |\varphi(i+1, f_y)\rangle \| &= \left\| U_{i+1} |\varphi(i, zr)\rangle - U_{i+1} O_{f_y} |\varphi(i, f_y)\rangle \right\| \\ &= \left\| |\varphi(i, zr)\rangle - O_{f_y} |\varphi(i, f_y)\rangle \right\| \end{aligned}$$

Par inégalité triangulaire

$$\begin{aligned} &\leq \left\| |\varphi(i, zr)\rangle - O_{f_y} |\varphi(i, zr)\rangle \right\| + \left\| O_{f_y} |\varphi(i, zr)\rangle - O_{f_y} |\varphi(i, f_y)\rangle \right\| \\ &\leq \left\| |\varphi(i, zr)\rangle - O_{f_y} |\varphi(i, zr)\rangle \right\| + \| |\varphi(i, zr)\rangle - |\varphi(i, f_y)\rangle \| \end{aligned}$$

On a utilisé 2 fois la question précédente, avec la porte U_{i+1} et avec la porte O_{f_y}.

3. Soit $y \in [\![0; 2^n - 1]\!]$, on pose $\alpha_y^i = \langle \varphi(i, zr) | \underline{y} \rangle$.

   (a) Soit $s \in [\![0; 2^n - 1]\!]$, que vaut $O_{f_s} |\varphi(i, zr)\rangle$?

   ▶ **Correction**

   Cette porte inverse le coefficient associé à $|\underline{s}\rangle$.

$$\begin{aligned} O_{f_s} |\varphi(i, zr)\rangle &= O_{f_s} \sum_{y=0}^{2^n} \alpha_y^i |\underline{y}\rangle \\ &= \sum_{y \neq s} \alpha_y^i |\underline{y}\rangle - \alpha_s^i |\underline{s}\rangle \end{aligned}$$

   (b) En déduire que $\| |\varphi(i, zr)\rangle - O_{f_s} |\varphi(i, z_r)\rangle \| \leq 2|\alpha_s^i|$.

   ▶ **Correction**

$$\| |\varphi(i, zr)\rangle - O_{f_s} |\varphi(i, z_r)\rangle \| = \left\| 2\alpha_s^i |\underline{s}\rangle \right\| = 2|\alpha_s^i|$$

   (c) En utilisant la question 2b, en déduire que

$$\| |\varphi(q, zr)\rangle - |\varphi(q, f_s)\rangle \| \leq 2 \sum_{i=0}^{q-1} |\alpha_s^i|$$

   ▶ **Correction**

   On peut montrer la relation par récurrence sur q.
   Puisque les qbits sont identiques au début de l'algorithme, on a

$$\| |\varphi(0, zr)\rangle - |\varphi(0, f_s)\rangle \| = 0$$

   Supposons la propriété vrai pour $q-1$. Montrons qu'elle l'est pour q. D'après la question 2b,

$$\| |\varphi(q, zr)\rangle - |\varphi(q, f_s)\rangle \| \leq \| |\varphi(q-1, zr)\rangle - O_{f_s} |\varphi(q-1, zr)\rangle \| + \| |\varphi(q-1, zr)\rangle - |\varphi(q-1, f_s)\rangle \|$$

   Par hypothèse de récurrence

$$\leq \| |\varphi(q-1, zr)\rangle - O_{f_s} |\varphi(q-1, zr)\rangle \| + 2 \sum_{i=0}^{q-2} |\alpha_s^i|$$

D'après la question précédente

$$\leq 2|\alpha_s^{q-1}| + 2\sum_{i=0}^{q-2}|\alpha_s^i|$$
$$\leq 2\sum_{i=0}^{q-1}|\alpha_s^i|$$

Par théorème de récurrence, la propriété est vraie pour tout q.

(d) En déduire que $\frac{1}{2^n}\sum_{y=0}^{2^n-1}\||\varphi(q,zr)\rangle - |\varphi(q,f_y)\rangle\| \leq \frac{2q}{\sqrt{2^n}}$

On rappelle que $\left|\sum_{i=1}^{n} x_i y_i\right| \leq \sqrt{\sum_{i=0}^{n} x_i^2}\sqrt{\sum_{i=0}^{n} y_i^2}$ pour tout $x, y \in \mathbb{R}^n$.

► **Correction**

D'après la question précédente, on a

$$\sum_{y=0}^{2^n-1}\||\varphi(q,zr)\rangle - |\varphi(q,f_y)\rangle\| \leq \sum_{y=0}^{2^n-1} 2\sum_{i=0}^{q-1}|\alpha_y^i|$$
$$\leq 2\sum_{i=0}^{q-1}\sum_{y=0}^{2^n-1}|\alpha_y^i|$$
$$\leq 2\sum_{i=0}^{q-1}\sum_{y=0}^{2^n-1}|\alpha_y^i \cdot 1|$$

On utilise l'inégalité de Cauchy Schwarz rappelée dans le sujet.

$$\leq 2\sum_{i=0}^{q-1}\sqrt{\sum_{y=0}^{2^n-1}|\alpha_y^i|^2}\sqrt{\sum_{y=0}^{2^n-1}|1|^2}$$
$$\leq 2\sum_{i=0}^{q-1}\sqrt{\sum_{y=0}^{2^n-1}|\alpha_y^i|^2 2^n}$$

On rappelle que $\alpha_y^i = \langle\varphi(i,zr)|\underline{y}\rangle$, donc $\sqrt{\sum_{y=0}^{2^n-1}|\alpha_y^i|^2} = \||\varphi(i,zr)\rangle\| = 1$

$$\leq 2\sum_{i=0}^{q-1}\sqrt{2^n} = 2q\sqrt{2^n}$$

$$\frac{1}{2^n}\sum_{y=0}^{2^n-1}\||\varphi(q,zr)\rangle - |\varphi(q,f_y)\rangle\| \leq 2q\frac{\sqrt{2^n}}{2^n} = 2q\frac{1}{\sqrt{2^n}}$$

4. Déduire de tout ça que $q = \Omega(\sqrt{2^n})$

► **Correction**

Ainsi, d'après la première partie et la dernière partie, on a

$$\frac{1}{\sqrt{2}}\frac{2^n-1}{2^n} \leq \frac{1}{2^n}\sum_{y=0}^{2^n-1} \left\| \left|\underline{y}\right\rangle - \left|\varphi(q, zr)\right\rangle \right\|$$

$$\frac{2q}{\sqrt{2^n}} \geq \frac{1}{2^n}\sum_{y=0}^{2^n-1} \left\| \left|\varphi(q, zr)\right\rangle - \left|\varphi(q, f_y)\right\rangle \right\|$$

On rappelle qu'on a supposé que l'algorithme produit à la fin exactement $\left|\underline{y}\right\rangle$, donc $\left|\varphi(q, f_y)\right\rangle = \left|\underline{y}\right\rangle$ et donc

$$\frac{1}{\sqrt{2}}\frac{2^n-1}{2^n} \leq \frac{2q}{\sqrt{2^n}}$$

$$\frac{1}{2\sqrt{2}}\frac{2^n-1}{2^n}\sqrt{2^n} \leq q$$

$$\frac{1}{4\sqrt{2}}\sqrt{2^n} \leq q$$

Donc $q = \Omega(\sqrt{2^n})$.