Algorithme de Shor : aménagements

On cherche à implémenter l'algorithme de Shor sur un ordinateur classique, c'est-à-dire non quantique. Pour cela on se base sur l'article Wikipédia correspondante. Ce document à pour but d'expliquer quelques points particuliers de l'algorithme, ainsi que de justifier quelques aménagements d'implémentation.

Étant donné qu'on cherche à réaliser un algorithme correct, chaque écart à l'article est précisément justifié.

1 Organisation de l'aglorithme

1.1 Dérécursification

L'article propose un algorithme en « go-to », c'est-à-dire en sautant d'une étape à une autre. L'implémentation la plus naturelle de ce type d'algorithme est récursive, néanmoins dans notre cas cela pose quelques problèmes, puisqu'on aurait une complexité spatiale non constante (inutilement), ce qui pourrait causer un dépassement de mémoire. On a donc préféré une boucle infinie, qui nous permet d'avoir un usage de mémoire constant.

1.2 Étapes 5 et 6 : négation

Les étapes 5 et 6 de l'article indiquent :

$$(r \text{ est impair}) \text{ ou } (a^{r/2} \equiv -1 \pmod N) \Rightarrow (\text{retour à l'étape 1})$$

Et dans le cas contraire, on accède à l'étape 7. On décide donc pour faciliter notre implémentation en boucle infinie, d'utiliser la contraposée. On sait que :

$$\text{non}(\text{retour à l'étape 1}) \iff (\text{passage à l'étape 7})$$

On en déduit :

$$\begin{aligned}(\text{passage à l'étape 7}) &\iff \text{non}\big((r \text{ est impair}) \text{ ou } (a^{r/2} \equiv -1 \pmod N)\big) \\ &\iff (r \text{ pair}) \text{ et } (a^{r/2} \not\equiv -1 \pmod N)\end{aligned}$$

Intéressons-nous à la deuxième partie. Supposons $a^{r/2} \equiv -1 \pmod N$, alors $a^{r/2} \equiv N - 1 \pmod N$, donc $a^{r/2}$ et $N - 1$ ont le même reste dans la division euclidienne par N. Or le reste de $N - 1$ dans la division euclidienne par N est clairement $N - 1$, on en déduit :

$$a^{r/2} \equiv -1 \pmod N \iff (a^{r/2} \mod N) = N - 1$$

2 Recherche de période

On cherche à trouver *le plus petit* $r \in \mathbb{N}$ tel que, avec $f : x \mapsto a^x \mod N$:

$$\begin{aligned} &\forall x \in \mathbb{N},\, f(x+r) = f(x) \\ &\iff a^{x+r} \equiv a^x \pmod N \\ &\iff a^x \cdot a^r \equiv a^x \pmod N \end{aligned}$$

Fixons $x \in \mathbb{N}$. Puisque $a \wedge N = 1$, a est inversible modulo N, notons α cet inverse. On a $a \cdot \alpha \equiv 1 \pmod N$, donc $\forall x \in \mathbb{N}, a^x \cdot \alpha^x \equiv 1^x \equiv 1 \pmod N$. a^x est donc inversible d'inverse α^x modulo N, et on en déduit :

$$\big(\forall x \in \mathbb{N}, f(x+r) = f(x)\big) \iff a^r \equiv 1 \pmod N$$

D'après le théorème d'Euler, en notant φ l'indicatrice d'Euler, on a : $a^{\varphi(N)} \equiv 1 \pmod N$, ce qui permet de prouver l'existence de r.

2.1 Algorithme classique

On a clairement $0 < r \leq \varphi(N) \leq N-1$. La recherche de r avec un algorithme classique itérant tous les r entre 1 et $N-1$ est donc un $O(N)$, ce qui réduit l'intérêt d'utiliser l'algorithme de Shor.

On note qu'on a, avec K élevé (100 000 pour la valeur approchée) :

$$\frac{1}{K}\sum_{i=1}^{K}\frac{\varphi(i)}{i} \approx 0.608 > \frac{1}{2}$$

On préférera donc partir de $N-1$ et décrémenter plutôt que partir de 0 et incrémenter.

2.2 Algorithme quantique

Le but est donc d'obtenir la valeur de r en $O(\log N)$.