Binome II - Exercice 2.10

17 janvier 2021

0.1 Motivations

La théorie de *la complexité* est le domaine des mathмatiques, et plus précisément de l'informatique théorique, qui étudie formellement la quantité de ressources (temps, espace mémoire, etc.) dont a besoin un algorithme pour résoudre un problème algorithmique.

Excercice 2.10

Dans cet excercice on s'intéresse au calcul du nombre arithmétique des opérations, ou *du temps*, nécessaire à l'exécution d'un algorithme. La complexité étudiée est donc *temporrelle*.

Soit u une fonction complexe et a-périodique. La T.F.D calcule les coefficients de Fourier de u à partir de N-échantillion $u(\frac{ka}{N}), k = 0, ..., N-1$.
En posant $w_N = e^{\frac{2i\pi}{N}}$ et $u_k = u(\frac{ka}{N}); k = 0, ..., n-1$. on a :

$$\widetilde{u}_n = \frac{1}{N} \sum_{l=0}^{N-1} u_l w_N^{-nl} = \frac{1}{N} \sum_{l=0}^{N-1} u_l e^{\frac{-2i\pi nl}{N}}.$$

Donc le calcul des $\widetilde{u}_n$, $n = 0, ..., N-1$. revient à évaluer le polynome $P(X) = \sum_{k=0}^{N-1} a_k X^k$ en $e^{\frac{2i\pi k}{N}}$, les racines N-ième de l'unitée. Avec $a_k = u_k$.
On a, pour chaque $n = 0, ..., N-1.$, le nombre opérations pour calcles les $\widetilde{u}_n$ est d'ordre $\mathcal{O}(N)$, donc la complexité du calcul de la Transformation de Fourier Discrète est d'ordre $\mathcal{O}(N^2)$.

Le but de cet excercice est de démunier cette complexité à l'ordre $\mathcal{O}(N \log(N))$.

0.2 L'algorithme de T.F.R de *Cooley-Tuckey*

On appelle $T(N)$ la complexité temporelle (le nombre des opérations arithmitiques), requis pour calculer la Transformation de Fourier Discrète.

Pour des raisons de simplification l'étude sera réduite sur l'ensemble $P_2 = \{2^n, n \in \mathbb{N}^*\}$.
On pose :

$$N = 2^n \ , \ \ Q(X) = \sum_{k=0}^{\frac{N}{2}-1} a_{2k} X^k, \quad \text{et} \quad R(X) = \sum_{k=0}^{\frac{N}{2}-1} a_{2k+1} X^k.$$

Pour k dans $\{0, ..., N-1\}$, on a :

$$\begin{aligned}
P(w_N^k) &= \sum_{l=0}^{N-1} a_l (w_N^k)^l \\
&= \sum_{l=0}^{\frac{N}{2}-1} a_{2l} (w_N^k)^{2l} + \sum_{l=0}^{\frac{N}{2}-1} a_{2l+1} (w_N^k)^{2l+1} \\
&= \sum_{l=0}^{\frac{N}{2}-1} a_{2l} ((w_N^k)^2)^l + w_N^k \sum_{l=0}^{\frac{N}{2}-1} a_{2l+1} ((w_N^k)^2)^l \\
&= Q((w_N^k)^2) + w_N^k R((w_N^k)^2).
\end{aligned}$$

N étant pair, donc $(w_N^k)^2, k = 0, ..., N-1.$ sont les racines $\frac{N}{2}$-ième de l'unitée. Donc, pour évaluer P aux racines N-ième de l'unitée, il suufit d'évaluer Q et R aux $(w_N^k)^2, k = 0, ..., N-1$.

L'évaluation de R et Q aux $(w_N^k)^2, k = 0, ..., N-1.$ nécessite $T(\frac{N}{2})$ opérations et puisque on effectue une multiplication et une somme pour atteindre

$$P(w_N^k) = Q((w_N^k)^2) + w_N^k R((w_N^k)^2).$$

D'où,

$$T(N) = 2N + 2T(\frac{N}{2}).$$

Montrons que : $T(N) = \mathcal{O}(N \log(N))$**.**

Démonstration. $N = 2^n, n \geq 2.$

On a :

$$\begin{aligned} T(2^n) = 2 * 2^n + 2T(\frac{2^n}{2}) \;\Rightarrow\; & \frac{T(2^n)}{2^n} = 2 + \frac{T(2^{n-1})}{2^{n-1}}, \text{ une suite arithmétique de raison 2.} \\ & \Rightarrow \frac{T(2^n)}{2^n} = \frac{T(2)}{2} + 2(n-1). \\ & \Rightarrow \frac{T(N)}{N} = \frac{T(2)}{2} - 2 + 2\log_2(N). \text{ on pose } c = \frac{T(2)}{2} - 2. \\ & \Rightarrow T(N) = cN + 2N\log_2(N). \\ & \Rightarrow T(N) = \mathcal{O}(N\log(N)). \end{aligned}$$

$\square$