Université Bordeaux 1 **MHT812 – Master**
Mathématiques Année 2008–2009

Correction du Devoir Surveillé, 4 mars 2009

Exercice 1 – [VARIATION SUR FFT]

1) Par construction

$$P_0 \equiv P \pmod{X^{n/2}-1} \quad \text{et} \quad Q_0 \equiv Q \pmod{X^{n/2}-1}$$

donc

$$R_0 \equiv P_0Q_0 \equiv PQ \pmod{X^{n/2}-1}, \tag{1}$$

ce qui démontre la première affirmation. Similairement

$$P_1 \equiv P \pmod{X^{n/2}+1} \quad \text{et} \quad Q_1 \equiv Q \pmod{X^{n/2}+1}$$

donc

$$P_1Q_1 \equiv PQ \pmod{X^{n/2}+1}.$$

En outre,

$$R_1(\omega X) \equiv P_1(\omega X)Q_1(\omega X) \pmod{X^{n/2}-1};$$

en évaluant cette congruence en $X = \omega^{-1}Y$, on obtient

$$R_1(Y) \equiv P_1(Y)Q_1(Y) \pmod{(\omega^{-1}Y)^{n/2}-1}.$$

Or $\omega^{n/2} = -1$ car ω est une racine primitive de 1 d'ordre n, d'où

$$(\omega^{-1}Y)^{n/2} - 1 = -(Y^{n/2}+1).$$

On obtient alors

$$R_1(Y) \equiv P_1(Y)Q_1(Y) \equiv P(Y)Q(Y) \pmod{Y^{n/2}+1}. \tag{2}$$

2) On démontre la validité de l'algorithme par récurrence sur l'exposant k ; c'est clair pour $k = 0$, car dans ce cas $P \star Q = PQ$.

Soit $k \geq 1$ et supposons que l'algorithme calcule correctement la convolution pour les polynômes de degré $< 2^{k-1}$. Ainsi R_0 et R_1 sont correctement calculés. De (1) on tire par multiplication par $X^{n/2}+1$

$$R_0(X^{n/2}+1) \equiv PQ(X^{n/2}+1) \pmod{X^n-1}. \tag{3}$$

De même, de (2) on tire par multiplication par $X^{n/2}-1$

$$R_1(X^{n/2}-1) \equiv PQ(X^{n/2}-1) \pmod{X^n-1}. \tag{4}$$

En effectuant la différence entre (3) et (4) on obtient

$$(R_0 - R_1)X^{n/2} + R_0 + R_1 \equiv 2PQ \pmod{X^n-1},$$

d'où le résultat.

3) Notons c_n la complexité algébrique de cet algorithme sur des polynômes de degré $< n = 2^k$. On a $c_1 = 1$. Pour $n \geq 2$, l'algorithme

- effectue $2n$ opérations en ligne 4 : en effet, si $T = \sum_{i=0}^{n} a_i X^i$, alors

$$T \bmod (X^{n/2} \pm 1) = \sum_{i=0}^{n/2-1} (a_i \mp a_{i+n/2}) X^i;$$

- fait deux appels récursifs en degré $< n/2$ et effectue $3n/2$ multiplications par les ω^i dans le calcul de R_1, en ligne 5 ;
- effectue en ligne 6 deux additions en degré $< n/2$ puis une multiplication scalaire en degré $< n$, ce qui coûte en plus $n/2+n/2+n = 2n$ opérations.

Au total on obtient

$$c_n = 2c_{n/2} + \frac{11n}{2}.$$

Compte tenu de la valeur de c_1, on obtient par récurrence

$$c_n = n + \frac{11}{2} n \log_2 n = O(n \log n).$$

Exercice 2 – [Suite de Fibonacci]

1) Considérons par exemple l'algorithme suivant.

Algorithme 1. Calcul des F_n's

Entrées: $n \in \mathbb{N}$.
Sorties: F_n le n-ième nombre de Fibonacci.

```
si n = 0 alors
    retourner 0.
si n = 1 alors
    retourner 1.
u ← 0, v ← 1
pour i = 2 ... n faire
    w ← u + v ;   u ← v ;   v ← w.
Retourner v.
```

Cet algorithme calcule correctement F_n, utilisant avec $n-1$ additions dans $\mathbb{N}$ si $n \geqslant 2$.

2) On a $F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$ et donc

$$F_n \leqslant \frac{1}{\sqrt{5}}(\Phi^n + 1) \leqslant \Phi^n,$$

où $\Phi = (1+\sqrt{5})/2 \approx 1.618$ est le nombre d'or. En particulier,

$$\log_2 F_n \leq n \log_2 \Phi,$$

et la longueur de F_n est $\leqslant n\log_2\Phi+1$. Le coût de l'addition dans $\mathbb{N}$ de deux nombres de longueur $\leqslant \ell$ est majoré par 2ℓ. Si l'on note d_n la complexité binaire de l'algorithme proposé, on a

$$d_n \leqslant \sum_{j=1}^{n-1} 2j\cdot\big(\log_2\Phi+1\big) = (n-1)n\cdot\big(\log_2\Phi+1\big) = O(n^2).$$

3) Démontrons la relation par récurrence sur k. Pour $k=0$ on a

$$F_{n+0+1} = F_nF_0 + F_{n+1}F_1,$$

pour tout n car $F_0=0$ et $F_1=1$, et la propriété est vraie à l'ordre 0. Supposons la propriété vraie à l'ordre $k\geqslant 0$. Alors pour tout n

$$\begin{aligned}
F_{n+(k+1)+1} &= F_{(n+1)+k+1} \\
&= F_{n+1}F_k + F_{n+2}F_{k+1} && \text{(hypothèse de récurrence)}\\
&= F_{n+1}F_k + (F_{n+1}+F_n)F_{k+1} && \text{(Fibonacci)}\\
&= F_nF_{k+1} + F_{n+1}(F_k+F_{k+1}) \\
&= F_nF_{k+1} + F_{n+1}F_{k+2} && \text{(Fibonacci)}
\end{aligned}$$

et la propriété est vraie à l'orde $k+1$.

4) Si $n>0$ est pair, prenons dans la relation précédente n et k repectivement égaux à $\frac{n}{2}$ et $\frac{n}{2}-1$ puis $\frac{n}{2}$ et $\frac{n}{2}$. On obtient alors :

$$\begin{aligned}
F_n &= 2F_{\frac{n}{2}}F_{\frac{n}{2}+1} - F_{\frac{n}{2}}^2 \\
F_{n+1} &= F_{\frac{n}{2}}^2 + F_{\frac{n}{2}+1}^2.
\end{aligned} \tag{5}$$

De même si $n>0$ est impair, en prenant pour n et k respectivement $\frac{n-1}{2}$ et $\frac{n-1}{2}$ puis $\frac{n-1}{2}$ et $\frac{n+1}{2}$ on trouve :

$$\begin{aligned}
F_n &= F_{\frac{n-1}{2}}^2 + F_{\frac{n+1}{2}}^2 \\
F_{n+1} &= F_{\frac{n+1}{2}}F_{\frac{n-1}{2}} + 2F_{\frac{n+1}{2}}^2.
\end{aligned} \tag{6}$$

On peut alors écrire l'algorithme suivant.

Algorithme 2. Un autre algorithme pour le calcul des F_n

Entrées: $n \in \mathbb{N}$.
Sorties: (F_n, F_{n+1}) le n-ième et le $(n+1)$-ième nombres de Fibonacci.

1: **si** $n = 0$ **alors**
2: retourner $(0, 1)$
3: **si** $n = 1$ **alors**
4: retourner $(1, 1)$
5: **si** n pair **alors**
6: calculer (F_n, F_{n+1}) à partir de $(F_{\frac{n}{2}}, F_{\frac{n}{2}+1})$ par les formules (5) ;
7: **sinon**
8: calculer (F_n, F_{n+1}) à partir de $(F_{\frac{n-1}{2}}, F_{\frac{n+1}{2}})$ par les formules (6).
9: Retourner (F_n, F_{n+1}).

Notons e_n la complexité algébrique de cet algorithme. On appelle récursivement la procédure $O(\log_2 n)$ fois. Plus précisément, si $2^k \leqslant n < 2^{k+1}$ on l'appelle $k+1$ fois. À chaque étape on fait 7 opérations [1] dont 6 dans les formules en comptant la multiplication par 2 ! On a donc $e_n = O(\log_2 n)$.

★ **5)** Notons b_n la complexité binaire de l'algorithme précedent. Supposons que $2 \leqslant n < 2^{k+1}$. À chacune des $k+1$ étapes de calcul, on fait 7 opérations dont le coût est inférieur à celui que l'on aurait lors des $k+1$ premières étapes du calcul de $(F_{2^{k+1}}, F_{2^{k+1}+1})$; en effet, les entiers manipulés sont plus petits par croissance de (F_i).

A fortiori le coût total est inférieur à celui du calcul de $(F_{2^{k+1}}, F_{2^{k+1}+1})$. Pour ce dernier calcul, on calcule successivement les (F_{2^i}, F_{2^i+1}), i variant de 0 à $k+1$ en faisant à chaque étape 7 opérations de coût binaire majoré par $\mathsf{M}(l(F_{2^i}))$ où l désigne la longueur. Par ce qui a été vu plus haut ce coût est majoré par $\mathsf{M}(2^i \log_2 \Phi + 1) = O(\mathsf{M}(2^i))$. En sommant on obtient

$$b_n \leqslant O\Big(\sum_{i=0}^{k+1} \mathsf{M}(2^i)\Big) = O(\mathsf{M}(n)).$$

[1] l'étude de la parité de n et le calcul du nouvel indice - qui est le quotient de n par 2 - peuvent être assimilés à la division par 2.