Université Bordeaux 1
Mathématiques

MHT812 – Master
Année 2008–2009

Devoir Surveillé, 4 mars 2009

Durée 1h30. Documents interdits.

Exercice 1 – [Variation sur la FFT]

Soit $n = 2^k$, où $k \in \mathbb{N}$, une puissance de 2. Soit A un anneau commutatif contenant une racine primitive n-ième de l'unité ω et dans lequel 2 est inversible (on notera $1/2$ son inverse). Soient P et Q deux polynômes de $A[X]$ de degrés $< n$. On a vu un algorithme (FFT) permettant de calculer $P \star Q = PQ \bmod (X^n - 1)$ et donc PQ si $\deg P + \deg Q < n$, qui prend appui sur l'évaluation de P et Q en les ω^i, $0 \leqslant i \leqslant n-1$. On se propose ici d'étudier la variante suivante de cet algorithme.

Algorithme 1. Convolution rapide

Entrées: P, $Q \in A[X]$ de degrés $< n = 2^k$ et les puissances $\omega, \omega^2, \ldots, \omega^{n/2-1}$ d'une racine primitive n-ième de l'unité.

Sorties: $P \star Q = PQ \bmod (X^n - 1)$.

si $k = 0$ **alors**
  Retourner PQ
$P_0 \leftarrow P \bmod (X^{n/2} - 1)$, $P_1 \leftarrow P \bmod (X^{n/2} + 1)$,
$Q_0 \leftarrow Q \bmod (X^{n/2} - 1)$, $Q_1 \leftarrow Q \bmod (X^{n/2} + 1)$.
Appeler l'algorithme **récursivement** pour calculer R_0, $R_1 \in A[X]$ de degrés $< n/2$ tels que

$$R_0 \equiv P_0 Q_0 \bmod (X^{n/2} - 1), \quad R_1(\omega X) \equiv P_1(\omega X) Q_1(\omega X) \bmod (X^{n/2} - 1).$$

Retourner

$$\frac{1}{2}\Big((R_0 - R_1)X^{n/2} + R_0 + R_1\Big).$$

---

1) Montrer que $R_0 \equiv PQ \bmod (X^{n/2} - 1)$ et $R_1 \equiv PQ \bmod (X^{n/2} + 1)$.

2) En déduire que l'algorithme 1 calcule effectivement $P \star Q$.

3) Quelle est la complexité algébrique (nombre d'opérations dans A) de cet algorithme ?

Exercice 2 – [Suite de Fibonacci]

On rappelle que la suite de Fibonacci est définie par

$$F_0 = 0, \quad F_1 = 1, \quad \text{et} \quad F_{n+2} = F_{n+1} + F_n \text{ pour tout } n \geqslant 0.$$

1) Écrire un algorithme qui calcule F_n en $O(n)$ additions dans $\mathbb{N}$.

2) Estimer la complexité binaire de cet algorithme[1].

---

[1] On rappelle que $F_n = \frac{1}{\sqrt{5}}\Big(\Phi^n - (-\Phi)^{-n}\Big)$ où $\Phi = (1+\sqrt{5})/2 \approx 1.618$ est le nombre d'or.

3) Montrer que pour tout $k, n \in \mathbb{N}$, on a

$$F_{n+k+1} = F_n F_k + F_{n+1} F_{k+1}.$$

4) En déduire un algorithme [2] de calcul de F_n faisant appel à $O(\log n)$ opérations dans $\mathbb{N}$.

★ **5)** Estimer la complexité binaire de ce nouvel algorithme.

N.B.: On écrira ces algorithmes avec soin (pseudo-code ou code `Maple`) et on justifiera proprement la complexité à chaque fois.

---

[2] On pourra chercher une procédure calculant (F_n, F_{n+1}) à partir de $(F_{n/2}, F_{n/2+1})$ si n est pair et $(F_{(n-1)/2}, F_{(n+1)/2})$ si n est impair, procédure que l'on appliquera récursivement.