Université Bordeaux 1 **MHT839 – Master**
Mathématiques Année 2015–2016

Devoir Surveillé, 2 mars 2016
Corrigé.

Exercice 1 – [Interpolation d'Hermite]

1) *On cherche à trouver un polynôme P de $\mathbb{Q}[x]$ de degré inférieur ou égal à 5 tel que $P(1) = 0$, $P'(1) = -3$, $P''(1) = 2$, $P'''(1) = 42$, $P(2) = 11$ et $P'(2) = 45$.*

a) Sur votre copie, traduire ces contraintes en termes de problème de restes chinois.

Ce problème est équivalent au système suivant.

$$\begin{cases} P(x) \equiv P(1) + (x-1)P'(1) + \dfrac{(x-1)^2}{2}P''(1) + \dfrac{(x-1)^3}{6}P'''(1) \mod (x-1)^4 \\ P(x) \equiv P(2) + (x-2)P'(2) \mod (x-2)^2 \end{cases}$$

c'est-à-dire :

$$\begin{cases} P(x) \equiv -3(x-1) + 2\dfrac{(x-1)^2}{2} + 42\dfrac{(x-1)^3}{6} \mod (x-1)^4 \\ P(x) \equiv 11 + 45(x-2) \mod (x-2)^2 \end{cases}$$

b) Résoudre le problème en utilisant sage, par exemple la commande `crt`. *Sur votre copie, écrire la commande utilisée et le résultat obtenu.*

Sur sage, si l'on exécute les commandes

```
Qx.<x>=PolynomialRing(QQ)
crt([-3*(x-1)+(x-1)^2+7*(x-1)^3,11+45*(x-2)],[(x-1)^4,(x-2)^2])
```

on obtient le résultat $P(x) = x^5 - 3x^3 + x + 1$.

2) *Pour tout polynôme P et tout entier naturel i, on note $P^{(i)}$ la dérivée i-ème de P avec la convention : $P^{(0)} = P$.*

a) Écrire sur votre fichier ".sage" une fonction `Hermite1` *qui prend en entrée une liste $l = [a, b]$, où $a \in \mathbb{Q}$ et où b est elle même une liste $[b_0, \ldots, b_r]$ d'éléments de $\mathbb{Q}$, et qui en sortie rend le polynôme P de degré minimal tel que $P^{(i)}(a) = b_i$ pour tout $i \in [[0, r]]$.*

b) Écrire sur votre fichier ".sage" une fonction `Hermite` *qui en entrée prend une liste $l = [[a_0, b_0], \ldots, [a_s, b_s]]$ où les a_i sont des éléments de $\mathbb{Q}$ et les b_i des listes $[b_{i0}, \ldots, b_{ir_i}]$ d'éléments de $\mathbb{Q}$ et qui en sortie rend le polynôme de degré minimal P. tel que pour tout $i \in [[0, s]]$ et tout $j \in [[0, r_i]]$,*

$$P^{(j)}(a_i) = b_{ij}.$$

Cette fonction utilisera votre fonction `Hermite1` *de la question 2.a et la fonction* `crt` *de sage.*

Exercice 2 – [Convolution rapide]

Cet exercice est à rédiger sur votre copie.

Soit n une puissance de 2. Soit K un corps de caractéristique différente de 2 qui contient une racine primitive n-ème de l'unité notée ω. Soient f et g deux polynômes à coefficients dans K de degrés strictement inférieurs à n. On note $f *_n g$ le reste de la division euclidienne de fg par $x^n - 1$. On considère l'algorithme suivant.

Algorithme 1. *Convolution rapide*

Entrées: $\{\omega, \omega^2, \ldots, w^{n/2-1}\}$, $f, g \in K[x]$, $\deg f < n$, $\deg g < n$

Sorties: $f *_n g$

1: *Si $n = 1$, alors sortir fg*

2: *Calculer les restes respectifs f_0 et g_0 de la division de f et g par $x^{n/2} - 1$, ainsi que les restes respectifs f_1 et g_1 de la division de f et g par $x^{n/2} + 1$*

3: *Appeler récursivement l'algorithme pour calculer $h_0(x) = f_0(x) *_{n/2} g_0(x)$ et $h_1(\omega x) = f_1(\omega x) *_{n/2} g_1(\omega x)$*

4: *Sortir $(h_0 + h_1 + x^{n/2}(h_0 - h_1))/2$*

---

1) *Exécuter à la main cet algorithme avec les données $n = 4$, $f = 1 + x^3$ et $g = 1 + x + x^2$, le corps de base étant $K = \mathbb{C}$, puis vérifier le résultat trouvé.*

Comme $n = 4$, on peut prendre $\omega = i$. On calcule d'abord $f_0 = x + 1$, $f_1 = -x + 1$, $g_0 = x + 2$ et $g_1 = x$.

L'algorithme se rappelle lui-même avec $n = 2$, $\omega = -1$, f_0 et g_0. On calcule les restes de f_0 et g_0 modulo $x - 1$ et $x + 1$.

$$f_{00} = 2, f_{01} = 0, g_{00} = 3, g_{01} = 1.$$

L'algorithme se rappelle lui-même avec $n = 1$, $\omega = 1$ pour $f_{00} *_1 g_{00}$ et trouve $h_{00} = 2 * 3 = 6$ puis pour $f_{01} *_1 g_{01}$ et trouve $h_{01} = 0 * 1 = 0$.

L'algorithme se rappelle à nouveau pour le calcul de $f_1(ix) *_2 g_1(ix)$. Or, $f_1(ix) = -ix + 1$ et $g_1(ix) = ix$. Les restes de ces polynômes modulo $x - 1$ et $x + 1$ sont

$$f_{10} = -i + 1, f_{11} = i + 1, g_{10} = i, g_{11} = -i.$$

L'algorithme se rappelle avec $n = 1$, $\omega = 1$ pour $f_{10} *_1 g_{10}$ et trouve $h_{10} = (-i + 1) * i = 1 + i$ puis pour $f_{11} *_1 g_{11}$ et trouve $h_{11} = (i + 1) * (-i) = 1 - i$.

On revient au calcul de $f_0 *_2 g_0$. On trouve $h_0 = (h_{00} + h_{01} + x(h_{00} - h_{01}))/2 = (6 + 6x)/2 = 3 + 3x$.

De même, on calcule $f_1(ix) *_2 g_1(ix)$: $h_1(ix) = (h_{10} + h_{11} + x(h_{10} - h_{11}))/2 = (2 + 2ix)/2 = 1 + ix$. On en déduit que $h_1(x) = 1 + x$.

Enfin, on calcule $f *_4 g = (h_0 + h_1) + x^2(h_0 - h_1))/2 = (4x + 4 + x^2(2x + 2))/2 = x^3 + x^2 + 2x + 2$.

Pour vérifier le résultat obtenu, on calcule $fg = x^5 + x^4 + x^3 + x^2 + x + 1$, que l'on divise par $x^4 - 1$. Le reste obtenu est bien égal à $x^3 + x^2 + 2x + 2$ (pour obtenir ce reste, le plus simple est de remarquer que $x^4 \equiv 1 \mod x^4-1$ et $x^5 \equiv x \mod x^4-1$).

2) *Montrer que* $2fg \equiv (x^{n/2}+1)f_0g_0 - (x^{n/2}-1)f_1g_1 \mod x^n - 1$.

Comme $f_0g_0 \equiv fg \mod x^{n/2}-1$, on a aussi la congruence $(x^{n/2}+1)f_0g_0 \equiv (x^{n/2}+1)fg \mod x^n-1$. De même, $(x^{n/2}-1)f_1g_1 \equiv (x^{n/2}-1)fg \mod x^n-1$. En soustrayant ces deux congruences entre elles, on obtient le résultat escompté.

3) *Montrer que* $h_1(x) \equiv f_1(x)g_1(x) \mod x^{n/2}+1$.

Il existe un polynôme q tel que $h_1(\omega x) = f_1(\omega x)g_1(\omega x)+(x^{n/2}-1)q(x)$. Soit $y = \omega x$. Ce changement de variable donne $h_1(y) = f_1(y)g_1(y)+(\omega^{-n/2}y^{n/2}-1)q(\omega^{-1}x)$. Comme $\omega^{n/2} = -1$, on obtient $h_1(y) = f_1(y)g_1(y) - (y^{n/2}+1)q(\omega^{-1}x)$, donc $h_1(y) \equiv f_1(y)g_1(y) \mod y^{n/2}+1$.

4) *Montrer que l'algorithme "Convolution rapide" calcule bien* $f *_n g$. Si $n = 1$, c'est clair. Soit $n \geqslant 2$ une puissance de 2. On suppose que l'algorithme fonctionne à l'ordre $n/2$. Soient alors f et g de degrés strictement inférieurs à n. L'hypothèse de récurrence montre que $h_0 = f_0 *_{n/2} g_0$ et que $h_1(\omega x) = f_1(\omega x) *_{n/2} g_1(\omega x)$. D'après la question précédente, $h_1 \equiv f_1g_1 \mod x^{n/2}+1$, donc $(x^{n/2}-1)h_1 \equiv (x^{n/2}-1)f_1g_1 \mod x^n-1$ Comme $h_0 \equiv f_0g_0 \mod x^{n/2}-1$, on a aussi la congruence $(x^{n/2}+1)h0 \equiv (x^{n/2}+1)f_0g_0 \mod x^n-1$. On peut donc remplacer f_0g_0 et f_1g_1 par h_0 et h_1 dans la congruence

$$2fg \equiv (x^{n/2}+1)f_0g_0 - (x^{n/2}-1)f_1g_1 \mod x^n-1$$

obtenue à la question 2, ce qui donne

$$\begin{aligned} 2fg &\equiv (x^{n/2}+1)h_0 - (x^{n/2}-1)h_1 \mod x^n-1 \\ &\equiv h_0 + h_1 + x^{n/2}(h_0-h_1) \mod x^n-1. \end{aligned}$$

Cela prouve le résultat.

5) *Calculer la complexité algébrique de cet algorithme.*

Soit $C(n)$ cette complexité algébrique pour $f *_n g$. Pour calculer le reste de la division de f et de g par $x^{n/2}-1$, il suffit de remarquer que $x^{n/2+i} \equiv x^i$ pour tout entier naturel i et donc de remplacer les $x^{n/2+i}$ par x^i dans les expressions de f et g, ce qui se fait en temps linéaire. De même, la division par $x^{n/2}+1$ se fait en temps linéaire. Quand le polynôme $H_1(x) = h_1(\omega x)$ est calculé, on en déduit $h_1(x) = H_1(\omega^{-1}x)$ en temps linéaire. Dans le pas 4, la multiplication par $x^{n/2}$ est un décalage. Restent les sommes et la division par 2 qui sont de complexité linéaire. Comme l'algorithme se rappelle deux fois en taille $n/2$, on obtient $C(n) = 2C(n/2)+O(n)$, ce qui prouve que $C(n) = O(n \log n)$ d'après un résultat du cours.