UFR Sciences
M2 Calcul Formel Martin WEIMANN

TP 2 : Autour de l'algorithme d'Euclide

Exercice 1 (division euclidienne) 1. Ecrire l'algorithme classique de division euclidienne dans $K[X]$ (on accède au monôme dominant avec `lt()` pour "leading term").

2. Tester votre procédure avec $A, B \in \mathbb{Q}[X]$ aléatoires de degrés respectifs 10 et 5, puis 100 et 10 et comparer vos résultats avec ceux obtenus par les commandes de Sage.

3. Quelle taille en moyenne ont les coefficients du reste R pour A, B de degrés respectifs 1000 et 10 (utiliser la commande `len(str(`R`))`) ?

Exercice 2 (multiplication dans un corps de nombres) Soit $\alpha = \sqrt[4]{2}$. On représente un élément x du corps de nombre $\mathbb{Q}[\alpha]$ par la liste de ses coefficients dans la base $(1, \alpha, \alpha^2, \alpha^3)$ de $\mathbb{Q}[\alpha]$ vu comme espace vectoriel sur $\mathbb{Q}$.

1. Ecrire un algorithme de multiplication dans $\mathbb{Q}[\alpha]$.

2. Calculer $(1 + \alpha + \alpha^2 + \alpha^3)^4$ avec votre programme

3. Reprendre le 2) avec les commandes Sage. On déclare l'anneau de polynôme $\mathbb{Q}[X]$ avec `R=QQ['X']` et `X=R.gen()` pour déclarer la variable, ou directement `R.<X>=PolynomialRing(QQ)`. On déclare le corps de nombre $\mathbb{Q}[\alpha]$ en considérant le quotient `K.<z>=R.quo(X^4-2)` .

4. Comment calculer $(1 + \alpha + \alpha^2 + \alpha^3)^{1000}$ efficacement (cf aussi exo 1 du TP4) ?

Exercice 3 (pgcd) 1. Ecrire un programme qui calcule le pgcd de deux entiers a et b. Votre algorithme s'adape-t-il aux polynômes ? Comparer avec la commande `gcd` de Sage.

2. Modifier la fonction pgcd pour qu'elle utilise cette fois le reste centré $-\frac{|b|}{2} < r \leqslant \frac{|b|}{2}$.

3. Écrire une fonction `f(N)` qui mesure le temps nécessaire au calcul du pgcd de deux entiers aléatoires d'au plus `N` bits (utiliser `randint`).

4. Tracer le graphe de f pour $N \in [1, 1000]$, sous forme d'un nuage de points (commande `point`), puis sous forme de ligne brisée (commande `line2d`). Comment semble évoluer le temps de calcul en fonction de la taille de l'entrée ?

5. Reprendre la question précédente en incluant cette fois le graphe pour le pgcd centré. Quelle méthode est préférable ?

Exercice 4 (Itérations dans l'algorithme d'Euclide) 1. Comparer le nombre moyen d'itérations utilisées par l'algorithme d'Euclide sur N paires d'entiers aléatoires $a, b < B$ avec la quantité $12 \log 2 \log B / \pi^2$.

2. Soit la suite de Fibonacci définie par $F_0 = 0$, $F_1 = 1$ et $F_{k+2} = F_{k+1} + F_k$. Vérifier expérimentalement que le nombre *maximal* d'itérations utilisées par l'algorithme d'Euclide sur N paires d'entiers aléatoires $a, b \leq F_{k+1}$ ne dépasse pas k puis vérifier que ce nombre d'itérations est précisément atteint pour $a = F_k$ et $b = F_{k+1}$. Ce fait est prouvé par le *théorème de Lamé*.

Exercice 5 (pgcd étendu et équation modulo N) 1. Programmer l'algorithme d'Euclide étendu en version itérative puis en version récursive. Comparer avec la commande `xgcd` de Sage.

2. Écrire une fonction qui prend en entrée trois entiers a, b et $N \geq 1$ et qui résout l'équation $ax + b = 0 \mod N$.

3. Reprendre la question 2. en utilisant cette fois l'arithmétique modulaire proposée par Sage (utiliser la commande `solve_mod`).

Exercice 6 (Restes chinois) On utilisera dans ce qui suit la commande `xgcd` de Sage.

1. Soient $n_1, n_2 \in \mathbb{N}$ premiers entre eux et $a_1, a_2 \in \mathbb{Z}$. Ecrire un programme qui résoud le système

$$\begin{cases} x \equiv a_1 \mod n_1 \\ x \equiv a_2 \mod n_2 \end{cases}$$

   Comparer avec la procédure `crt`() de Sage.

2. On ne suppose plus n_1 et n_2 premiers entre eux. On note $d = \text{pgcd}(n_1, n_2)$ et on considère u, v tels que $un_1 + vn_2 = d$.

   (a) Montrer que si $a_1 - a_2 \neq 0 \mod d$, alors il n'y a pas de solutions au système.

   (b) Montrer que l'on a les égalités

$$x := a_1 + u\frac{n_1}{d}(a_2 - a_1) = a_2 + v\frac{n_2}{d}(a_1 - a_2) = a_2 u\frac{n_1}{d} + a_1 v\frac{n_2}{d}$$

   (c) En déduire que si $a_1 - a_2 = 0 \mod d$, alors x est solution au problème, et que cette solution est unique modulo $n_1 n_2/d$. *Noter que l'on retrouve bien le théorème des restes chinois classique explicite lorsque* $d = 1$.

   (d) Ecrire un programme qui résoud le système pour n_1, n_2 entiers positifs quelconques.

3. Ecrire un programme `MultiChinois` qui, étant données deux listes d'entiers $a_1, \ldots, a_k$ et $n_1, \ldots, n_k$ avec les $n_i \in \mathbb{N}$ deux à deux premiers entre eux, résoud le système d'équations

$$\begin{cases} x &= a_1 \mod n_1 \\ x &= a_2 \mod n_2 \\ &\vdots \\ x &= a_k \mod n_k \end{cases}$$

   Vérifier que la procédure `crt`() de Sage fonctionne dans ce cas.

4. *Diviser pour régner.* Ecrire un programme `FastMultiChinois` en adoptant cette fois une stratégie "diviser pour régner".

5. *Complexité.* Comparer graphiquement les efficacités des algorithmes `MultiChinois` et `FastMultiChinois` en considérant $k \in \text{range}(1000, 3000, 20)$. On prendra les $a_i < 10000$ aléatoires et les n_i les k premiers nombres premiers. Comparer ensuite avec la procédure `crt` de Sage. Que constate-t-on ? Proposer une explication en regardant de plus près le code source de Sage en tapant `crt??` puis `CRT_list??`.

6. En supposant un algorithme d'Euclide étendu en $O(\text{M}(n)\log(n))$, montrer que `FastMultiChinois` a un coût en $O(\text{M}(n)\log^2(n))$ (donc quasi-linéaire avec la multiplication rapide).

7. *Application à l'interpolation polynomiale.* Utiliser l'un de vos programmes pour calculer l'unique polynôme $P \in \mathbb{Q}[X]$ de degré $k-1$ interpolant des valeurs $a_1, \ldots, a_k$ en des points $x_1, \ldots, x_k$ deux à deux distincts.