Transformation de Fourier rapide

Exercice 1 (*Racines primitives modulo p*)

Écrire une procédure `RPrim(p,n,m)` prenant en entrée un nombre premier p et deux entiers $m, n \geqslant 1$ et renvoyant `Fail`, si $m \nmid p^n - 1$, ou un couple (P, L) où P est un polynôme irréductible unitaire de degré n de $\mathbf{F}_p[X]$ et L est la liste des racines primitives m-èmes de l'unité dans $\mathbf{F}_q := \mathbf{F}_p[X]/(P)$, dans le cas contraire.

[On pourra, pour déterminer P, choisir au hasard un élément unitaire de degré n de $\mathbf{F}_p[X]$ et tester son irréductibilité avec `is_irreducible`. Pour obtenir les racines primitives m-èmes de l'unité de $\mathbf{F}_q$ on *ne calculera pas* l'ordre de tous les éléments un par un.]

Exercice 2 (*Transformation de Fourier rapide*)

Le but de cet exercice est d'implanter un algorithme de transformation de Fourier rapide dans un corps fini $\mathbf{F}_p$, p premier (ou plus généralement dans $\mathbf{Z}/n\mathbf{Z}$). Dans la première question on n'aborde que certains aspects de l'algorithme sur un exemple concret.

1. Soit $F = \mathbf{F}_{17}$. On considère $f = 5x^3 + 3x^2 - 4x + 3$ et $g = 2x^3 - 5x^2 + 7x - 2$ vus dans $F[x]$.
   (a) Montrer que $\omega = 2$ est une racine primitive 8-ème de l'unité dans F.
   (b) Pour $0 \leqslant j < 8$, calculer $\alpha_j := f(\omega^j)$, $\beta_j := g(\omega^j)$ et $\gamma_j := \alpha_j \cdot \beta_j$.
   [On pourra utiliser la commande `f.subs(x=a)` pour évaluer l'expression $f(x)$ en a. On rappelle aussi que si `f` est un polynôme, `f[i]` est son i-ème coefficient.]
   (c) Calculer le produit $f \cdot g$ en utilisant la transformation de Fourier discrète.
2. Écrire une procédure `fft(p,k,omega,a)` prenant en argument un nombre entier impair p, un entier $k \geqslant 0$, une racine primitive $n = 2^k$-ème de l'unité ω dans $R = \mathbf{Z}/p\mathbf{Z}$ et une liste $a = [a_0, a_1, \ldots, a_{n-1}]$ d'éléments de R et renvoyant $\mathrm{TFD}_\omega(\sum a_i X^i)$.
3. Avec les mêmes notations (et les mêmes entrées), écrire une procédure `inversefft` permettant de calculer $\mathrm{TFD}_\omega^{-1}(a) = (1/n)\mathrm{TFD}_{\omega^{-1}}(a)$.
4. Écrire une procédure `convolution(p,k,omega,f,g)` permettant de calculer le produit de convolution $f *_n g$ des polynômes $f, g \in \mathbf{F}_p[x]$ relativement à ω, racine primitive $n = 2^k$-ème de l'unité dans $\mathbf{F}_p$ (la procédure est testée dans la question suivante).
5. Soit $F = \mathbf{F}_{41}$. Donner, en utilisant l'exercice 1, la liste des racines primitives 8-èmes de l'unité dans F. Posons $\omega = 14$, $f = x^7 + 2x^6 + 3x^4 + 2x + 6 \in F[x]$ et $g = x^6 + 12x^5 + 35x^3 + 1 \in F[x]$. Calculer $f *_8 g$ par transformation de Fourier rapide.
6. Combiner les exercices 1 et 2 pour écrire une procédure `multpol(p,f,g)` calculant le produit d'éléments f et g de $\mathbf{F}_p[x]$ par transformation de Fourier rapide.

Exercice 3 (*Nombres premiers de Fourier*)

On étudie dans cet exercice certains corps $\mathbf{F}_p$ permettant (partiellement au moins) la transformation de Fourier rapide. Les nombres premiers p de la forme $1 + r2^s$ correspondants sont appelés *nombres premiers de Fourier*.

1. Soit $s \geqslant 1$. Soit p un nombre premier congru à 1 modulo 2^s. On note r l'entier tel que $p = 1 + r2^s$. Soit a un entier tel que $\left(\frac{a}{p}\right) = -1$. Montrer que la classe de a^r dans $\mathbf{F}_p$ est une racine primitive 2^s-ème de l'unité.

2. Écrire une procédure `RPrimS(p,s)` prenant en argument p et s comme ci-dessus et renvoyant une racine primitive 2^s-ème de l'unité dans $\mathbf{F}_p$.

3. Écrire une procédure `TroisPrem(s)` prenant en argument un entier $1 \leqslant s \leqslant 61$ et renvoyant, s'il en existe, des nombres premiers p_1, p_2, p_3 distincts compris entre 2^{63} et 2^{64} et des racines primitives 2^s-èmes de l'unité ω_i dans $\mathbf{F}_{p_i}$ pour $1 \leqslant i \leqslant 3$.

4. Quelle est la valeur maximale $s_{\max}$ de s pour laquelle la procédure ci-dessus trouve de telles données ?

Exercice 4 (*Multiplication rapide d'entiers*)

Cet exercice est une application de l'exercice 3 et de la transformation de Fourier rapide à la multiplication efficace des entiers.

Dans les ordinateurs modernes les nombres entiers naturels sont écrits en base 2 et on regroupe les chiffres par blocs de 64. Un nombre entier naturel n à au plus k blocs peut être identifié à un polynôme $P \in \mathbf{Z}[X]$ de degré $< k$ dont les coefficients sont dans l'intervalle $\{0, \ldots, 2^{64} - 1\}$ par la correspondance $n = P(2^{64})$. Pour simplifier on travaillera directement en base 2^{64}. (Voir les commandes `ZZ(L,b)` permettant de donner la valeur en base 10 d'un nombre dont la liste des chiffres en base `b` est `L`, et `a.digits(b)` permettant de donner la liste des chiffres de l'entier `a` dans la base `b`.)

Algorithme :

Supposons que deux entiers n et n' soient associés par le procédé ci-dessus à des polynômes P et P' tels que $d = \deg P + \deg P' < 2^s$ pour un entier $s \leqslant s_{\max}$ (on choisit de préférence le plus petit s possible). On utilise ensuite les couples (p_i, ω_i) pour $1 \leqslant i \leqslant 3$ pour calculer rapidement (par convolution rapide) la classe du produit PP' dans $\mathbf{F}_{p_i}[X]$. On utilise alors les restes chinois pour déterminer l'unique polynôme $Q \in \mathbf{Z}[X]$ (dont les coefficients sont compris entre 0 et $p_1p_2p_3 - 1$) se réduisant sur PP' modulo p_i pour tout i. On a alors l'égalité $Q = PP'$ dans $\mathbf{Z}[X]$ et le produit nn' est $Q(2^{64})$.

• Pourquoi a-t-on bien $Q = PP'$ dans $\mathbf{Z}[X]$?
• Implanter cet algorithme (on pourra écrire d'abord une procédure indépendante permettant de résoudre des systèmes de trois congruences (modulo p_1, p_2 et p_3)).