Multiplication rapide de polynômes

Épreuve pratique d'algorithmique et de programmation
Concours commun des Écoles normales supérieures

Durée de l'épreuve: 3 heures 30 minutes

Juin/Juillet 2014

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Sur votre table est indiqué un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion!). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures!
Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple: $O(n^2)$, $O(n \log n)$,...
Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe***. Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

L'objectif de ce sujet est d'étudier divers algorithmes de multiplication de polynômes ainsi que leur complexité en fonction du nombre de termes des polynômes manipulés.
Pour expérimenter ces algorithmes, nous considérerons ici des polynômes dont les coefficients sont des entiers modulo $p = 12289$. En d'autres termes, nous travaillerons dans l'anneau des polynômes $(\mathbb{Z}/p\mathbb{Z})[t]$. Il est à noter que, comme p est premier, $\mathbb{Z}/p\mathbb{Z}$ est un corps : tous ses éléments non nuls sont inversibles modulo p.
Dans la suite, nous appellerons polynôme de taille n un polynôme de degré au plus $n-1$:

$$A(t) = \sum_{i=0}^{n-1} a_i t^i, \text{ où tous les } a_i \in \mathbb{Z}/p\mathbb{Z}.$$

Un tel polynôme sera représenté par la donnée du n-uplet $(a_i)_{0\leq i<n}$. De manière à ce que cette représentation soit unique, on veillera à ce que les coefficients a_i (qui sont en fait des classes d'équivalence modulo p) soient toujours représentés par l'unique entier $\tilde{a}_i \in \{0, \ldots, p-1\}$ de cette classe d'équivalence. Autrement dit, les coefficients des polynômes doivent toujours être réduits modulo p. Par exemple, le polynôme $2t-1$, de taille 2, sera représenté par le couple d'entiers $(12288, 2)$.

Ce sujet se divise en quatre parties. Les trois dernières, traitant chacune un algorithme de multiplication différent, sont relativement indépendantes les unes des autres.

1 Préliminaires

1.1 Génération de polynômes aléatoires

Avant toute chose, considérons les trois suites d'entiers positifs $(u_k)_{k\geq 0}$, $(v_k)_{k\geq 0}$ et $(w_k)_{k\geq 0}$ définies par

$$u_k = \begin{cases} \text{votre } u_0 \text{ (à reporter sur votre fiche réponse)} & \text{si } k = 0, \\ 17\,420 \times u_{k-1} \bmod 32\,003 & \text{si } k > 0, \end{cases}$$

$$v_k = \begin{cases} \text{votre } u_0 \text{ (le même)} & \text{si } k = 0, \\ 17\,420 \times v_{k-1} \bmod 32\,009 & \text{si } k > 0, \end{cases} \quad \text{et}$$

$$w_k = (u_k \cdot v_k) \bmod p, \quad \text{avec } p = 12289.$$

Question 1 *Donnez les valeurs (u_k, v_k, w_k) pour k valant* ***a)*** 5, ***b)*** 50, ***c)*** 500.

Nous notons alors $P_{n,k}$ le polynôme de taille n représenté par les coefficients $(w_{kn+i})_{0\leq i<n}$:

$$P_{n,k}(t) = \sum_{i=0}^{n-1} w_{kn+i} t^i.$$

1.2 Fonction de signature

Étant donnés deux entiers α et r, nous définissons aussi une fonction de hachage $\mathsf{Hash}_{\alpha,r}$ qui, à tout n-uplet d'entiers $A = (a_i)_{0\leq i<n}$, associe la quantité

$$\mathsf{Hash}_{\alpha,r}(A) = \left(\sum_{i=0}^{n-1} a_i \alpha^i \right) \bmod r.$$

Il est à noter que cette fonction de hachage peut être naturellement étendue aux polynômes, puisque tout polynôme de taille n peut être représenté de manière unique par un n-uplet d'entiers dans $\{0, \ldots, p-1\}$. Attention toutefois à bien effectuer tout le calcul de $\mathsf{Hash}_{\alpha,r}$ modulo r et non pas modulo p.
Nous définissons enfin une fonction de signature Sig qui, elle aussi, peut être appliquée indifféremment aux n-uplets ou aux polynômes de taille n :

$$\mathsf{Sig}(A) = (\mathsf{Hash}_{17,983}(A), \mathsf{Hash}_{23,991}(A), \mathsf{Hash}_{51,997}(A)).$$

Question 2 *Donnez la signature* $\mathsf{Sig}(P_{n,k})$ *des polynômes* $P_{n,k}(t)$ *suivants :*
a) $P_{10,0}(t)$, ***b)*** $P_{100,1}(t)$, ***c)*** $P_{1000,5}(t)$.

Remarque. Attention au dépassement de capacité des entiers machine lors du calcul des fonctions de hachage $\mathsf{Hash}_{\alpha,r}$. Pensez à réduire les résultats intermédiaires modulo r aussi souvent que nécessaire.

1.3 Opérations élémentaires

Étant donnés deux polynômes $A(t)$ et $B(t)$ (pas forcément de même taille) ainsi qu'un élément $c \in \mathbb{Z}/p\mathbb{Z}$ et un entier positif i, on définit les opérations classiques
- d'addition : $A(t) + B(t)$;
- de soustraction : $A(t) - B(t)$;
- de multiplication par un scalaire $c \cdot A(t)$; et
- de multiplication par t^i : $A(t) \cdot t^i$.

Question 3 *Calculez les polynômes suivants et donnez leur signature :*
a) $P_{15,1}(t) + P_{10,3}(t)$, ***b)*** $P_{100,1}(t) - P_{150,2}(t)$, ***c)*** $w_{42} \cdot P_{1000,1}(t)$, ***d)*** $P_{4000,1}(t) \cdot t^{17}$.

Question à développer pendant l'oral : En supposant que les polynômes $A(t)$ et $B(t)$ soient de taille m et n, respectivement, donnez le coût exact de chacune de ces opérations, en matière d'additions et de multiplications dans $\mathbb{Z}/p\mathbb{Z}$. (Une soustraction ou un calcul d'opposé dans $\mathbb{Z}/p\mathbb{Z}$ sera compté comme une addition.)
Dans la suite du sujet, lorsqu'aucune ambiguïté n'est possible, on évitera d'écrire explicitement la variable (t) des polynômes afin de ne pas alourdir les notations. Ainsi, par exemple, on notera $A + B$ la somme de deux polynômes $A(t)$ et $B(t)$.

2 Algorithme naïf

Question à développer pendant l'oral : Étant donnés deux polynômes $A(t)$ et $B(t)$ de taille m et n, respectivement, quelle est la taille de leur produit ? Écrivez un algorithme naïf qui calcule ce produit. Quelle est sa complexité en temps et en mémoire ? Combien d'additions et de multiplications dans $\mathbb{Z}/p\mathbb{Z}$ nécessite-t-il exactement ?

Question 4 *Calculez les produits de polynômes suivants et donnez leur signature :*
a) $P_{10,1}(t) \times P_{20,2}(t)$, ***b)*** $P_{200,1}(t) \times P_{100,4}(t)$, ***c)*** $P_{4000,1}(t) \times P_{5000,2}(t)$.

3 Algorithme de Karatsuba

Dans cette partie, les polynômes que nous cherchons à multiplier sont tous deux de taille $n = 2^\ell$, pour un entier $\ell \geq 1$. On note $A(t) = \sum_{i=0}^{n-1} a_i t^i$ et $B(t) = \sum_{i=0}^{n-1} b_i t^i$ ces polynômes.
L'algorithme développé par Karatsuba en 1960 pour multiplier deux tels polynômes repose sur une approche de type diviser pour régner. Pour cela, on commence par découper chaque polynôme en deux parties de taille $n/2$ en écrivant

$$A(t) = A_1(t) \cdot t^{n/2} + A_0(t), \quad \text{avec } A_0(t) = \sum_{i=0}^{n/2-1} a_i t^i \text{ et } A_1(t) = \sum_{i=0}^{n/2-1} a_{n/2+i} t^i, \text{ et}$$
$$B(t) = B_1(t) \cdot t^{n/2} + B_0(t), \quad \text{avec } B_0(t) = \sum_{i=0}^{n/2-1} b_i t^i \text{ et } B_1(t) = \sum_{i=0}^{n/2-1} b_{n/2+i} t^i.$$

Le produit $A(t) \times B(t)$ est alors donné par

$$A(t) \times B(t) = A_1 B_1 \cdot t^n + (A_0 B_1 + A_1 B_0) \cdot t^{n/2} + A_0 B_0.$$

Tel qu'écrit ci-dessus, le produit de deux polynômes de taille n requiert 4 produits de polynômes de taille $n/2$ (les produits A_0B_0, A_0B_1, A_1B_0 et A_1B_1), ce qui n'apporte aucun avantage par rapport à l'algorithme naïf.
Cependant, la méthode de Karatsuba permet d'effectuer le même calcul en seulement 3 produits de taille $n/2$, au prix de quelques additions et soustractions supplémentaires, en remarquant que

$$A_0B_1 + A_1B_0 = (A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1.$$

Question à développer pendant l'oral : Quels sont ces 3 produits de taille $n/2$? En remarquant qu'il est possible d'appliquer récursivement la méthode de Karatsuba pour calculer ces produits, écrivez un algorithme calculant le produit de deux polynômes par cette méthode. Quel est sa complexité en temps et en mémoire ? Indiquez le nombre exact d'additions et de multiplications dans $\mathbb{Z}/p\mathbb{Z}$ requises pour multiplier deux polynômes de taille $n = 2^\ell$.

Question 5 *Calculez les produits de polynômes suivants et donnez leur signature :*
a) $P_{2^{13},1}(t) \times P_{2^{13},2}(t)$, ***b)*** $P_{2^{14},1}(t) \times P_{2^{14},2}(t)$, ***c)*** $P_{2^{15},1}(t) \times P_{2^{15},2}(t)$.

Question à développer pendant l'oral : Peut-on appliquer l'algorithme de Karatsuba à des polynômes dont la taille n n'est pas une puissance de 2 ? Si oui, comment ? Donnez une borne supérieure de la complexité de cet algorithme en fonction de n. Et lorsque les deux polynômes n'ont pas la même taille ?

4 Méthode par transformée de Fourier rapide

4.1 Multiplication par évaluation-interpolation

La multiplication par transformée de Fourier rapide appartient à la famille des algorithmes de multiplication par évaluation-interpolation. Ces algorithmes reposent sur la propriété qu'un polynôme $A(t)$ de taille n défini sur $\mathbb{Z}/p\mathbb{Z}$ peut être représenté de manière

unique par son évaluation en n points distincts $(t_i)_{0\leq i<n}$ de $\mathbb{Z}/p\mathbb{Z}$: l'unicité de l'interpolation polynomiale nous permet en effet de retrouver les n coefficients du polynôme étant données les n valeurs $(A(t_i))_{0\leq i<n}$.
Il se trouve de surcroît que l'évaluation du produit de deux polynômes $A(t)$ et $B(t)$ en un point t_i est égale au produit de l'évaluation des deux polynômes en ce même point :

$$(A \times B)(t_i) = A(t_i) \times B(t_i).$$

Ainsi, lorsque l'on souhaite multiplier deux polynômes $A(t)$ et $B(t)$ de taille n, on peut

- évaluer ces polynômes en un nombre m de points distincts $(t_i)_{0\leq i<m}$ afin d'obtenir les m-uplets $(A(t_i))_{0\leq i<m}$ et $(B(t_i))_{0\leq i<m}$;
- calculer les m produits point-à-point $A(t_i) \times B(t_i)$ pour $0 \leq i < m$; et
- interpoler le polynôme $A(t) \times B(t)$ à partir des m produits précédents.

Bien entendu, tous ces calculs s'effectuent modulo p.

Question à développer pendant l'oral : Quelle est la taille du produit $A(t) \times B(t)$ lorsque $A(t)$ et $B(t)$ sont de taille n ? Déduisez-en la valeur de m, le nombre de points d'interpolation nécessaires pour déterminer uniquement tous les coefficients de $A(t) \times B(t)$ grâce à cet algorithme.
Il nous reste donc à trouver des points d'interpolation permettant d'effectuer efficacement ces étapes d'évaluation et d'interpolation.

4.2 Racines de l'unité

Pour rappel, les polynômes que nous considérons sont à coefficients dans $\mathbb{Z}/p\mathbb{Z}$, avec $p = 12289$.
Pour tout entier positif n, nous définissons alors $\mathcal{R}_n$, l'ensemble des racines n-ièmes de l'unité de $\mathbb{Z}/p\mathbb{Z}$, comme suit :

$$\mathcal{R}_n = \{\omega \in \mathbb{Z}/p\mathbb{Z} \mid \omega^n = 1\}.$$

Comme $\mathcal{R}_n$ est l'ensemble des racines dans $\mathbb{Z}/p\mathbb{Z}$ du polynôme $t^n - 1$, il ne peut y avoir plus de n racines de l'unité. De plus, si p ne divise pas n, alors ce polynôme n'admet aucune racine multiple.
Nous définissons aussi $\mathcal{P}_n$, l'ensemble des racines n-ièmes primitives de l'unité :

$$\mathcal{P}_n = \{\omega \in \mathcal{R}_n \mid \omega^d \neq 1 \text{ pour tout } d \text{ divisant } n,\ d \neq n\}.$$

Question à développer pendant l'oral : Calculez $\mathcal{R}_3$, l'ensemble des racines cubiques de l'unité dans $\mathbb{Z}/p\mathbb{Z}$. Lesquelles sont primitives ?

Question 6 *Pour chacun des couples (n, b) suivants, calculez la racine primitive $\omega \in \mathcal{P}_n$ telle que la différence $(\omega - b) \bmod p$ soit la plus petite possible. Si $\mathcal{P}_n$ est vide, renvoyez $\perp$.*

a) $(2^5, w_{10})$, ***b)*** $(2^{12}, w_{100})$, ***c)*** $(2^{13}, w_{1000})$.

Question à développer pendant l'oral : Montrez que toute racine n-ième primitive de l'unité $\omega \in \mathcal{P}_n$ permet de générer l'ensemble des racines n-ièmes de l'unité $\mathcal{R}_n$, c'est-à-dire que

$$\mathcal{R}_n = \{\omega^0, \omega^1, \ldots, \omega^{n-1}\}.$$

Si une telle racine primitive existe, quel est alors le cardinal de $\mathcal{R}_n$?

4.3 Transformée de Fourier discrète

Étant donné un polynôme $A(t) = \sum_{i=0}^{n-1} a_i t^i$ de taille n et ω une racine n-ième primitive de l'unité dans $\mathbb{Z}/p\mathbb{Z}$, la transformée de Fourier discrète de ce polynôme en ω, notée $\mathsf{TFD}_\omega(A)$, est l'évaluation de celui-ci en les n points distincts $(\omega^i)_{0 \leq i < n}$:

$$\mathsf{TFD}_\omega(A) = (A(\omega^i))_{0 \leq i < n}.$$

Dans le reste de cette partie, nous posons $\omega_{2^{12}} = 2014 \in \mathbb{Z}/p\mathbb{Z}$ et, pour tout entier $0 \leq \ell < 12$, nous définissons $\omega_{2^\ell} = \omega_{2^{12}}^{2^{12-\ell}}$. On pourra vérifier que $\omega_{2^\ell} \in \mathcal{P}_{2^\ell}$ pour tout $0 \leq \ell \leq 12$.

Question à développer pendant l'oral : Écrivez un algorithme qui, étant donné un polynôme $A(t)$ de degré $n = 2^\ell$, calcule de manière naïve $\mathsf{TFD}_{\omega_{2^\ell}}(A)$, la transformée de Fourier discrète de ce polynôme en ω_{2^ℓ}. Quelle est sa complexité en temps et en mémoire ?

Question 7 *Pour chaque polynôme $P_{n,k}(t)$ de taille $n = 2^\ell$ suivant, calculez $\mathsf{TFD}_{\omega_{2^\ell}}(P_{n,k})$, sa transformée de Fourier discrète en ω_{2^ℓ}, et donnez la signature de cette transformée :*
a) $P_{2^4,0}(t)$, ***b)*** $P_{2^8,1}(t)$, ***c)*** $P_{2^{12},2}(t)$.

Étant donné un polynôme $A(t)$ de taille n et une racine n-ième primitive de l'unité $\omega \in \mathcal{P}_n$, on peut considérer le polynôme de taille n associé aux n coefficients de la transformée de Fourier discrète de $A(t)$ en ω, que l'on note $\hat{A}(t)$:

$$\hat{A}(t) = \sum_{i=0}^{n-1} A(\omega^i) t^i.$$

En appliquant l'opération une seconde fois, mais en prenant cette fois-ci ω^{-1} (l'inverse multiplicatif de ω dans $\mathbb{Z}/p\mathbb{Z}$) comme racine primitive de l'unité, on obtient alors le polynôme de taille n

$$\hat{\hat{A}}(t) = \sum_{i=0}^{n-1} \hat{A}(\omega^{-i}) t^i,$$

dont les coefficients sont directement donnés par $\mathsf{TFD}_{\omega^{-1}}(\hat{A})$.

Question à développer pendant l'oral : Vérifiez que $\hat{\hat{A}}(t) = n \cdot A(t)$. Déduisez-en que les coefficients de l'interpolation d'un polynôme de taille n à partir de son évaluation en les racines n-ièmes de l'unité $(\omega^i)_{0 \leq i < n}$ sont donnés par

$$\mathsf{TFD}_\omega^{-1}(\hat{A}) = \mathsf{TFD}_{\omega^{-1}}(\hat{A})/n,$$

appelée transformée de Fourier discrète inverse de $\hat{A}(t)$ en ω. (Notez bien que la division par n ci-dessus est effectuée dans $\mathbb{Z}/p\mathbb{Z}$: elle correspond en fait à une multiplication par $n^{-1} \bmod p$, l'inverse multiplicatif de n modulo p.)

Indication. Afin de prouver le résultat précédent, remarquez que $\sum_{i=0}^{n-1} (\omega^k)^i \bmod p = 0$ pour tous $n > 0$, $k \neq 0$ non divisible par n et $\omega \in \mathcal{P}_n$.

Question 8 *Pour chaque polynôme $P_{n,k}(t)$ de taille $n = 2^\ell$ suivant, calculez $\mathsf{TFD}^{-1}_{\omega_{2^\ell}}(P_{n,k})$, sa transformée de Fourier discrète inverse en ω_{2^ℓ}, et donnez la signature de cette transformée :*
a) $P_{2^4,3}(t)$, ***b)*** $P_{2^8,4}(t)$, ***c)*** $P_{2^{12},5}(t)$.

4.4 Transformée de Fourier rapide

Si la transformée de Fourier discrète et son inverse présentées dans la section précédente permettent d'évaluer et d'interpoler un polynôme aux n racines n-ièmes de l'unité de $\mathbb{Z}/p\mathbb{Z}$, la complexité de l'algorithme naïf pour calculer ces transformées est trop élevée pour que cette approche présente un quelconque avantage par rapport à l'algorithme de multiplication classique ou à la méthode de Karatsuba.
Cependant, l'algorithme de transformée de Fourier rapide, initialement mis au point par Gauss en 1805 puis redécouvert en 1965 par Cooley et Tukey, permet de calculer la transformée de Fourier discrète d'un polynôme bien plus efficacement.
Soit n un entier pair et $A(t) = \sum_{i=0}^{n-1} a_i t^i$ un polynôme de taille n. On définit alors les deux polynômes $A^{(0)}(t)$ et $A^{(1)}(t)$ de taille $n/2$ par

$$A^{(0)}(t) = \sum_{i=0}^{n/2-1} a_{2i} t^i \qquad \text{et} \qquad A^{(1)}(t) = \sum_{i=0}^{n/2-1} a_{2i+1} t^i,$$

de sorte à ce que l'on ait $A(t) = A^{(0)}(t^2) + A^{(1)}(t^2) \cdot t$.
La transformée de Fourier discrète $\mathsf{TFD}_\omega(A)$, pour $\omega \in \mathcal{P}_n$, revient donc à calculer les évaluations $A(\omega^i) = A^{(0)}(\omega^{2i}) + A^{(1)}(\omega^{2i}) \cdot \omega^i$ pour tout $0 \leq i < n$.
Or, on peut remarquer que ω^2 est une racine $(n/2)$-ième primitive de l'unité et que, pour tout $0 \leq i < n/2$, $w^{2i} = w^{2(i+n/2)}$. Ainsi, $A^{(0)}(\omega^{2i}) = A^{(0)}(\omega^{2(i+n/2)})$ est le i-ième coefficient de $\mathsf{TFD}_{\omega^2}(A^{(0)})$, la transformée de Fourier discrète de $A^{(0)}$, de taille $n/2$, en $\omega^2 \in \mathcal{P}_{n/2}$. Il en va de même pour les évaluations $A^{(1)}(\omega^{2i})$.
Ainsi, si l'on note les $(n/2)$-uplets

$$\left(\hat{a}_i^{(0)}\right)_{0 \leq i < n/2} = \mathsf{TFD}_{\omega^2}(A^{(0)}) \qquad \text{et} \qquad \left(\hat{a}_i^{(1)}\right)_{0 \leq i < n/2} = \mathsf{TFD}_{\omega^2}(A^{(1)}),$$

obtenus en calculant deux transformées de Fourier discrètes de taille $n/2$, on obtient les coefficients de la transformée de $A(t)$ par $A(\omega^i) = \hat{a}^{(0)}_{i \bmod (n/2)} + \hat{a}^{(1)}_{i \bmod (n/2)} \cdot \omega^i$.
On remarque enfin que, si n est une puissance de 2, il est possible d'appliquer ce processus récursivement afin de calculer les transformées de taille $n/2$ de la même manière, et ainsi de suite, jusqu'aux transformées de taille 1 qui ne sont rien d'autre que l'identité.

Question à développer pendant l'oral : Grâce aux observations précédentes, retrouvez l'algorithme de calcul de transformée de Fourier rapide d'un polynôme de taille $n = 2^\ell$, en faisant l'hypothèse que $\mathcal{P}_{2^\ell}$ ne soit pas vide. Quelle est sa complexité en temps et en mémoire ?

Remarque. Il est possible d'économiser encore quelques opérations dans l'algorithme de transformée de Fourier rapide en remarquant que $\omega^{i+n/2} = -\omega^i$ pour tout $\omega \in \mathcal{P}_n$.

4.5 On recolle les morceaux

En utilisant les réponses aux questions précédentes et les algorithmes que vous avez mis au point, vous devriez a priori tout avoir pour calculer le produit de deux polynômes $A(t)$ et $B(t)$ de taille $n = 2^\ell$ avec $\ell < 12$. Pour mémoire, la marche à suivre est la suivante :

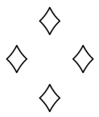
1. évaluer $A(t)$ et $B(t)$ en les $2^{\ell+1}$ racines de l'unité $\mathcal{R}_{2^{\ell+1}}$ en calculant leur transformée de Fourier discrète de taille $2^{\ell+1}$ en $\omega_{2^{\ell+1}}$ grâce à l'algorithme de transformée de Fourier rapide ;
2. effectuer les $2^{\ell+1}$ multiplications point-à-point $\hat{c}_i = A(\omega^i_{2^{\ell+1}}) \times B(\omega^i_{2^{\ell+1}})$;
3. interpoler le polynôme $C(t) = A(t) \times B(t)$ en les racines $(2^{\ell+1})$-ièmes de l'unité en appliquant la transformée de Fourier discrète inverse en $\omega_{2^{\ell+1}}$ à nouveau grâce à l'algorithme de transformée de Fourier rapide.

Question à développer pendant l'oral : Quelle est la complexité de cet algorithme, en temps et en mémoire ? Indiquez le nombre exact d'additions et de multiplications dans $\mathbb{Z}/p\mathbb{Z}$.

Question 9 *En utilisant cet algorithme de multiplication, calculez les produits de polynômes suivants et donnez leur signature :*
a) $P_{2^8,1}(t) \times P_{2^8,2}(t)$, ***b)*** $P_{2^{10},1}(t) \times P_{2^{10},2}(t)$, ***c)*** $P_{2^{11},1}(t) \times P_{2^{11},2}(t)$.

Question à développer pendant l'oral : De même que l'algorithme de Karatsuba peut être utilisé pour découper une multiplication de deux polynômes de taille $2n$ en trois multiplications de polynômes de taille n, l'algorithme de multiplication par transformée de Fourier rapide peut ici être utilisé pour découper une multiplication de deux polynômes de taille $2^{11}n$ en 2^{12} produits de polynômes de taille n, ainsi que des multiplications de polynômes de taille n par un scalaire. Expliquez comment adapter votre algorithme afin de pouvoir l'utiliser de manière récursive et ainsi multiplier des polynômes de plus grande taille. Quelle est la complexité en temps de cet algorithme ?

Question 10 *Calculez les produits de polynômes suivants et donnez leur signature :*
a) $P_{2^{16},1}(t) \times P_{2^{16},2}(t)$, ***b)*** $P_{2^{18},1}(t) \times P_{2^{18},2}(t)$, ***c)*** $P_{2^{20},1}(t) \times P_{2^{20},2}(t)$.



Fiche réponse type: Multiplication rapide de polynômes

$\widetilde{\mathbf{u_0}}$ **:** 42

Question 1

a) $(15207, 8469, 11652)$

b) $(22624, 13835, 2210)$

c) $(5765, 14177, 8555)$

Question 2

a) $(418, 975, 479)$

b) $(672, 405, 199)$

c) $(360, 901, 842)$

Question 3

a) $(427, 312, 246)$

b) $(40, 649, 116)$

c) $(739, 692, 810)$

d) $(199, 359, 979)$

Question 4

a) $(43, 500, 89)$

b) $(459, 572, 349)$

c) $(897, 396, 706)$

Question 5

a) $(833, 610, 142)$

b) $(249, 846, 142)$

c) $(863, 811, 472)$

Question 6

a) 1212

b) 11902

c) $\perp$

Question 7

a) $(932, 648, 526)$

b) $(728, 563, 327)$

c) $(68, 2, 705)$

Question 8

a) $(725, 326, 298)$

b) $(806, 644, 661)$

c) $(946, 445, 759)$

Question 9

a) $(148, 583, 153)$

b) $(959, 926, 915)$

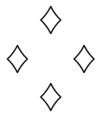
c) $(537, 70, 292)$

Question 10

a) $(660, 950, 930)$

b) $(952, 129, 945)$

c) $(2, 386, 811)$



Fiche réponse: Multiplication rapide de polynômes

Nom, prénom, u_0: .....

Question 1

a)

b)

c)

Question 2

a)

b)

c)

Question 3

a)

b)

c)

d)

Question 4

a)

b)

c)

Question 5

a)

b)

c)

Question 6

a)

b)

c)

Question 7

a)

b)

c)

Question 8

a)

b)

c)

Question 9

a)

b)

c)

Question 10

a)

b)

c)