ÉCOLE POLYTECHNIQUE **FILIÈRE MP**
OPTION INFORMATIQUE

CONCOURS D'ADMISSION 2007

COMPOSITION D'INFORMATIQUE

(Durée : 4 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
On attachera une grande importance à la concision, à la clarté, et à la précision de la rédaction.

⋆ ⋆ ⋆

Codage Reed-Solomon

Ce problème s'intéresse aux codes de correction d'erreurs de Reed-Solomon. On souhaite transmettre un message au travers d'un canal de communication *bruité* : les données émises sont modifiées par quelques erreurs lorsqu'elles sont reçues. Pour permettre néanmoins la transmission fiable d'un message, la donnée émise est un *codage* du message, plus long que le message lui-même. La *redondance* ainsi ajoutée permet de corriger ensuite des erreurs qui peuvent apparaître lors de la transmission. Les codes appelés codes de *Reed-Solomon* ont de nombreuses applications : les CDs, DVDs, les systèmes ADSL, ou encore de nombreuses sondes spatiales utilisent des codes bâtis autour des codes de Reed-Solomon.

Le préambule donne les définitions communes à l'ensemble de l'énoncé, et introduit les codes de Reed-Solomon. La partie I s'attache au calcul du codage d'un message. La partie II définit quelques fonctions de manipulation de polynômes. La partie III poursuit cette étude par la mise en œuvre d'algorithmes de meilleure complexité. Enfin, la partie IV permet d'améliorer la complexité du codage. On notera que le problème du *décodage* (retrouver un message original à partir d'une transmission bruitée) n'est pas traité dans ce problème.

Les parties I à III peuvent être traitées indépendamment. La partie IV repose sur les résultats de la partie III.

Préambule : Définitions et notations

Dans l'ensemble de l'énoncé, on fixe un nombre premier p inférieur à 2^{15} (de telle sorte que le produit de deux entiers modulo p soit toujours représentable par un simple entier positif en Caml ou en Pascal). Les calculs liés aux codes de Reed-Solomon seront réalisés dans le corps $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ des entiers modulo p. On rappelle que l'opération « modulo » est réalisée par l'opérateur `mod` en Caml et en Pascal, où on suppose également disposer de la fonction `inv` suivante pour calculer l'inverse modulo p de tout entier a vérifiant $0 < a < p$:

```
                (* Caml *)
(donné) inv : int -> int
```

```
                { Pascal }
(donné) function inv(a:integer)
    : integer
```

Toutes les questions ayant trait à la complexité des programmes écrits demandent comme réponse une borne supérieure (raisonnable) du type $O()$ sur le nombre d'opérations dans $\mathbb{F}_p$: chacune des opérations dans $\mathbb{F}_p$ (addition, multiplication, inversion) est considérée de complexité 1.

On manipulera des listes d'entiers modulo p en les identifiant avec des polynômes. Ainsi la liste de $(d+1)$ entiers $\langle a_0, \dots, a_d \rangle$ correspond au polynôme $A(X) = \sum_{i=0}^{d} a_i X^i$ de degré inférieur ou égal à d, à coefficients dans $\mathbb{F}_p$. On notera couramment $\deg A$ le degré d'un polynôme $A(X)$. Les listes d'entiers modulo p sont des valeurs du type `poly`, défini comme suit :

```
(* Caml *)
type poly = int list;;
```

```
{ Pascal }
type poly = ^polycoeff;
polycoeff = record
    coeff: integer;
    suivant: poly;
end;
```

En Pascal la liste vide est `nil`, et l'on pourra utiliser la fonction suivante pour construire des listes :

```
{ Pascal }
function nouveauPolynome(a : integer ; Q : poly) : poly;
var r : poly;
begin new(r); r^.coeff := a ; r^.suivant := Q; nouveauPolynome := r; end;
```

Pour définir le codage de Reed-Solomon, on fixe deux constantes entières k et n qui vérifient

$$0 < k < n < p.$$

Le codage de Reed-Solomon transforme un *message* en un *codage*. Le message est une liste $A = \langle a_0, \dots, a_{k-1} \rangle$ de k entiers modulo p, tandis que le codage est une liste $B = \langle b_0, \dots, b_{n-1} \rangle$ de n entiers modulo p. On peut aussi voir le message comme un polynôme $A(X)$ de degré inférieur ou égal à $k-1$. Le codage utilise comme paramètre une liste $\alpha = \langle \alpha_0, \dots, \alpha_{n-1} \rangle$ de n entiers modulo p, et est calculé comme suit :

$$\underbrace{\boxed{A = \langle a_0, \dots, a_{k-1} \rangle}}_{\text{message}} \longrightarrow \underbrace{\boxed{B = \langle b_0, \dots, b_{n-1} \rangle}}_{\text{codage}}, \text{ avec } b_i = A(\alpha_i) = \left(\sum_{j=0}^{k-1} a_j \alpha_i^j \right) \bmod p.$$

Dans les programmes qu'on écrira, on considérera les entiers k, n et p comme des constantes globales.

Partie I. Codage de Reed-Solomon, première version

Question 1 Écrire la fonction `valeur` qui prend comme arguments un polynôme U à coefficients dans $\mathbb{F}_p$ et un entier x ; et qui retourne la valeur $U(x)$ dans $\mathbb{F}_p$ de ce polynôme en cet entier x.

```
(* Caml *) valeur : poly -> int -> int
{ Pascal } function valeur(U : poly ; x : integer) : integer
```

Question 2 Écrire la fonction `codage` qui prend comme arguments la liste $\alpha = \langle \alpha_0, \ldots, \alpha_{n-1} \rangle$ et le polynôme $A(x)$; et qui retourne le codage de Reed-Solomon du message A.

```
(* Caml *) codage : poly -> poly -> poly
{ Pascal } function codage(α : poly ; A : poly) : poly
```

Question 3 Quelles sont les complexités des fonctions `valeur` et `codage` en fonction de n et k ?

Partie II. Polynômes

Dans cette partie, toutes les opérations retournent des polynômes à coefficients dans $\mathbb{F}_p$.

Question 4 Écrire la fonction `addition` qui prend comme arguments deux polynômes $U(X)$ et $V(X)$ à coefficients dans $\mathbb{F}_p$, représentés par des listes de ℓ_U et ℓ_V coefficients ; et qui retourne la somme de ces deux polynômes. Les coefficients du résultat sont aussi dans $\mathbb{F}_p$. Déterminer la complexité de cette fonction en fonction de ℓ_U et ℓ_V.

```
(* Caml *) addition : poly -> poly -> poly
{ Pascal } function addition(U : poly ; V : poly) : poly
```

Question 5 Écrire de même la fonction `soustraction` qui prend comme arguments deux polynômes $U(X)$ et $V(X)$ à coefficients dans $\mathbb{F}_p$, représentés par des listes de ℓ_U et ℓ_V coefficients ; et qui retourne la différence $U(X) - V(X)$. Déterminer la complexité de cette fonction en fonction de ℓ_U et ℓ_V.

```
(* Caml *) soustraction : poly -> poly -> poly
{ Pascal } function soustraction(U : poly ; V : poly) : poly
```

Question 6 Écrire la fonction `produitParScalaire` qui prend comme arguments un polynôme $U(X)$, représenté par la liste de ℓ_U coefficients, et un entier s dans $\mathbb{F}_p$; et qui retourne $s \times U(X)$. Donner la complexité de cette fonction en fonction de ℓ_U.

```
(* Caml *) produitParScalaire : poly -> int -> poly
{ Pascal } function produitParScalaire(U : poly ; s : integer) : poly
```

Question 7 Écrire la fonction `produit` qui prend comme arguments deux polynômes $U(X)$ et $V(X)$, représentés par des listes de ℓ_U et ℓ_V coefficients ; et qui retourne le produit de ces deux polynômes. Déterminer la complexité de cette fonction en fonction de ℓ_U et ℓ_V.

```
(* Caml *) produit : poly -> poly -> poly
{ Pascal } function produit(U : poly ; V : poly) : poly
```

Question 8 Soit un polynôme $U(X)$ de degré inférieur ou égal à d_U représenté par une liste de $\ell_U = (d_U + 1)$ entiers, et un polynôme non nul $V(X)$ de degré exactement égal à d_V, représenté par une liste de $\ell_V = (d_V + 1)$ entiers. Écrire la fonction `division` qui calcule le *quotient* de la division euclidienne de $U(X)$ par $V(X)$, noté $U \operatorname{div} V$. On rappelle que le résultat de la division euclidienne d'un polynôme $U(X)$ par un polynôme non nul $V(X)$ est l'unique couple $(Q(X), R(X))$ (quotient et reste) vérifiant :

$$\begin{aligned} &U(X) = Q(X)V(X) + R(X), \\ &\deg R < d_V. \end{aligned}$$

Déterminer la complexité de la fonction `division` en fonction de ℓ_U et ℓ_V.

```
(* Caml *) division : poly -> poly -> poly
{ Pascal } function division(U : poly ; V : poly) : poly
```

Question 9 Écrire la fonction `modulo` qui prend comme arguments deux polynômes $U(X)$ et $V(X)$; et qui calcule le reste de la division euclidienne de $U(X)$ par $V(X)$. Quelle est la complexité de cette fonction ?

```
(* Caml *) modulo : poly -> poly -> poly
{ Pascal } function modulo(U : poly ; V : poly) : poly
```

Dans la suite du problème, le reste de la division euclidienne de $U(X)$ par $V(X)$ est noté $U \bmod V$.

Partie III. Multiplication et division rapide

Cette partie s'attache à l'amélioration de la complexité des algorithmes de multiplication et de division de polynômes. La notation $\lfloor x \rfloor$ (respectivement $\lceil x \rceil$) désigne la partie entière inférieure (respectivement la partie entière supérieure) d'un nombre réel x. On remarque qu'on a, pour tout nombre entier d, l'équation $\left\lfloor \frac{d}{2} \right\rfloor + \left\lceil \frac{d}{2} \right\rceil = d$.

Multiplication Soient deux polynômes $U(X)$ et $V(X)$ représentés par les listes $\langle u_0, \ldots, u_{\ell-1} \rangle$ et $\langle v_0, \ldots, v_{\ell-1} \rangle$ ayant ℓ coefficients (le degré de U et V est donc au plus $\ell - 1$). On pose $e = \left\lfloor \frac{\ell}{2} \right\rfloor$ puis $U_b = \langle u_0, \ldots, u_{e-1} \rangle$ et $U_h = \langle u_e, \ldots, u_{\ell-1} \rangle$, de sorte que $U = U_b + X^e U_h$. On définit de même V_b et V_h. Lorsqu'on écrit le produit $U(X)V(X)$, on fait apparaître quatre produits impliquant les polynômes U_h, U_b, V_h, V_b :

$$\begin{aligned} U(X)V(X) &= (U_h X^e + U_b)(V_h X^e + V_b), \\ &= U_h V_h X^{2e} + \Big(U_h V_b + U_b V_h\Big) X^e + U_b V_b. \end{aligned}$$

Question 10 Donner une formule par laquelle le produit $U(X)V(X)$ s'exprime sous la forme $W_h X^{2e} + W_m X^e + W_b$, où les polynômes W_h, W_m et W_b se calculent avec seulement trois produits impliquant les polynômes U_h, V_h, U_b, V_b, $U_h + U_b$ et $V_h + V_b$.

Question 11 Écrire la fonction `produitRapide` qui prend comme arguments deux polynômes $U(X)$ et $V(X)$ représentés par des listes de ℓ coefficients ; et qui calcule leur produit, avec une complexité $O\left(\ell^{\log_2 3}\right)$ dans le cas où ℓ est une puissance de 2. Justifier cette formule de complexité.

```
(* Caml *) produitRapide : poly -> poly -> poly
{ Pascal } function produitRapide(U : poly ; V : poly) : poly
```

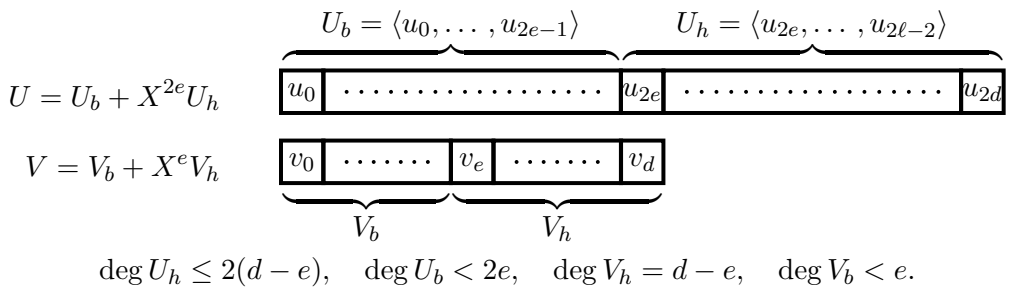
Division Il est possible d'obtenir une amélioration similaire de la division de polynômes. Soit un dividende $U(X)$ et un diviseur non nul $V(X)$, respectivement représentés par les listes $\langle u_0, \dots, u_{2\ell-2}\rangle$ et $\langle v_0, \dots, v_{\ell-1}\rangle$ d'entiers. On suppose que $v_{\ell-1}$ est non nul, de telle sorte que $\deg V$ est exactement égal à $d = \ell - 1$. Le degré de U est au plus égal à $2d = 2\ell - 2$.

Comme pour la multiplication, on sépare les entrées en « partie haute » et « partie basse ». On pose $e = \left\lfloor \frac{\ell}{2} \right\rfloor = \left\lceil \frac{d}{2} \right\rceil$, puis $U_h = \langle u_{2e}, \dots, u_{2\ell-2}\rangle$, $U_b = \langle u_0, \dots, u_{2e-1}\rangle$, $V_h = \langle v_e, \dots, v_{\ell-1}\rangle$, $V_b = \langle v_0, \dots, v_{e-1}\rangle$. Le découpage est représenté par la figure 1. Attention, selon la parité de d, on a $2e = d$ ou $2e = d + 1$.

On recherche le quotient Q, écrit sous la forme $Q = X^e Q_h + Q_b$ avec $\deg Q_h \leq d - e$ et $\deg Q_b < e$. Les formules suivantes donnent la valeur de Q_h et Q_b.

- Q_h est le quotient de la division de U_h par V_h.
- R_h est le reste de cette même division.
- Q_b est le quotient de la division de $S = \left\langle s_0, \dots, s_{2(d-e)}\right\rangle$ par V_h, où :

$$S = X^e R_h + u_{2e-1} X^{e-1} - Q_h V_b.$$



$$\deg U_h \leq 2(d-e), \quad \deg U_b < 2e, \quad \deg V_h = d - e, \quad \deg V_b < e.$$

FIG. 1: Découpage de U et V pour `divisionRapide`, où $e = \left\lfloor \frac{\ell}{2} \right\rfloor = \left\lceil \frac{d}{2} \right\rceil$

Question 12 Écrire la fonction `divisionRapide` qui prend comme arguments deux polynômes $U(X)$ et $V(X)$ représentés comme dans ce qui précède par deux listes de $2\ell - 1$ et ℓ coefficients, avec $v_{\ell-1}$ non nul ; et qui calcule leur quotient en utilisant cet algorithme. Quelle est la complexité de cette fonction dans le cas où ℓ est une puissance de deux ?

```
(* Caml *) divisionRapide : poly -> poly -> poly
{ Pascal } function divisionRapide(U : poly ; V : poly) : poly
```

Question 13 Écrire la fonction `moduloRapide` qui prend comme arguments deux polynômes $U(X)$ et $V(X)$ (toujours soumis à la contrainte $\deg U \leq 2 \deg V$) ; et qui calcule le reste de la division euclidienne de $U(X)$ par $V(X)$, plus rapidement qu'avec la fonction `modulo` de la question II.9. Quelle est la complexité de la fonction `moduloRapide` ?

```
(* Caml *) moduloRapide : poly -> poly -> poly
{ Pascal } function moduloRapide(U : poly ; V : poly) : poly
```

Partie IV. Codage par évaluation multi-points

L'objectif de cette partie est de proposer un autre algorithme pour calculer le codage de Reed-Solomon par une stratégie de type « diviser pour régner ».

On commence par construire un arbre binaire dont les nœuds sont étiquetés par des polynômes. Un *arbre de sous-produits* pour les polynômes $X - \alpha_0, X - \alpha_1, \ldots X - \alpha_{n-1}$ est défini comme suit : les $X - \alpha_i$ sont les étiquettes des feuilles de l'arbre (par ordre des indices i croissants dans l'ordre de lecture des feuilles de la gauche vers la droite), chaque nœud interne est étiqueté par le produit des polynômes étiquetant ses nœuds fils. Un arbre de sous-produits est dit *optimal* s'il est de hauteur h minimale ($2^{h-1} < n \leq 2^h$), et si les degrés des polynômes étiquetant deux fils d'un même nœud diffèrent d'au plus un. La figure 2 représente un tel arbre de sous-produits de hauteur $h = 3$ pour une famille de 6 polynômes $X - \alpha_0, \ldots, X - \alpha_5$.

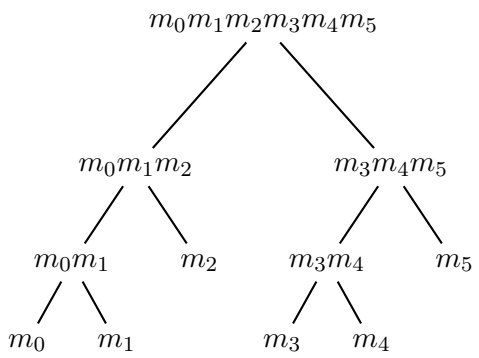


FIG. 2: Un arbre de sous-produits optimal pour $(m_0, \ldots, m_5)$, avec la notation $m_i = X - \alpha_i$.

Un arbre de sous-produits est représenté par une valeur de type `arbre`, défini comme suit :

```
(* Caml *)
type arbre = Vide
| Noeud of poly * arbre * arbre
```

```
{ Pascal }
type arbre = ^noeud;
  noeud = record etiquette: poly;
  filsG: arbre; filsD: arbre;
end;
```

L'arbre vide (sans aucun nœud) est représenté par `Vide` en Caml et par `nil` en Pascal.

Question 14 Écrire la fonction `arbreSousProduits` qui prend comme argument la liste $\langle \alpha_0, \alpha_1, \dots \alpha_{n-1} \rangle$; et qui retourne un arbre de sous-produits optimal pour la suite de n polynômes $X - \alpha_0, X - \alpha_1, \dots X - \alpha_{n-1}$. Dans le cas où on a $n = 2^h$, quelle est la complexité de la fonction `arbreSousProduits` ? On donnera deux résultats, selon que la multiplication est réalisée avec `produit` ou `produitRapide`.

```
(* Caml *) arbreSousProduits : poly -> arbre
{ Pascal } function arbreSousProduits(α: poly) : arbre
```

On souhaite maintenant calculer un second arbre, appelé *arbre des restes* de $A(X)$. Il est similaire à l'arbre de sous-produits calculé à la question précédente, mais les étiquettes des nœuds diffèrent : à un nœud d'étiquette m dans l'arbre des sous-produits, correspond un nœud d'étiquette $A(X) \bmod m$ dans l'arbre des restes. La figure 3 donne l'arbre des restes de $A(X)$ qui correspond à l'arbre de la figure 2.

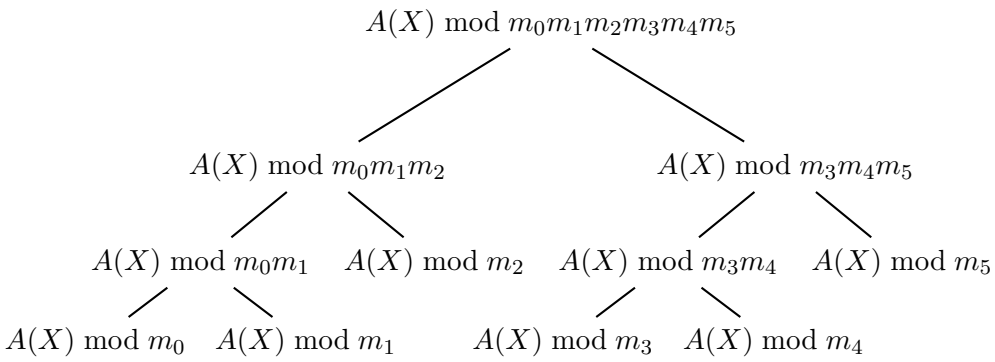


FIG. 3: Arbres des restes de $A(X)$ correspondant à la figure 2.

On fait l'observation suivante. Soient $(Q(X), R_1(X), R_2(X))$ trois polynômes à coefficients dans $\mathbb{F}_p$. Si on connaît $Q \bmod R_1R_2$, on peut calculer $Q \bmod R_1$ par l'opération :

$$Q \bmod R_1 = (Q \bmod R_1R_2) \bmod R_1.$$

Question 15 Écrire la fonction `arbreRestes` qui prend comme arguments l'arbre des sous-produits et un polynôme $A(X)$; et qui retourne comme résultat l'arbre des restes de $A(X)$. Quelle est la complexité de la fonction `arbreRestes` lorsqu'on utilise les fonctions de la partie III ? Quelle est la complexité lorsqu'on ne les utilise pas ?

```
(* Caml *) arbreRestes : arbre -> poly -> arbre
{ Pascal } function arbreRestes(T : arbre ; A : poly) : arbre
```

À présent, on se sert de l'arbre des sous-produits pour factoriser les évaluations du même polynôme $A(X)$ aux nombreux points $\alpha_0, \alpha_1, \ldots \ \alpha_{n-1}$. On remarque qu'on a pour tout i $(0 \leq i < n)$:

$$A(\alpha_i) = A(X) \bmod (X - \alpha_i)$$

Et on calcule ces valeurs à partir de

$$A(X) \bmod (X - \alpha_0)(X - \alpha_1) \cdots (X - \alpha_{n-1})$$

à l'aide de l'arbre des restes.

Question 16 Écrire la fonction `codageArbre` qui prend comme arguments la liste $\alpha = \langle \alpha_0, \ldots, \alpha_{n-1} \rangle$ et le polynôme $A(x)$; et qui retourne le codage de Reed-Solomon du message A en utilisant l'arbre de sous-produits correspondant à la suite des polynômes $X - \alpha_i$. Déterminer sa complexité.

```
(* Caml *) codageArbre : poly -> poly -> poly
{ Pascal } function codageArbre(α : poly ; A : poly) : poly
```

* *
*