Algorithmes Arithmétiques II – TP 2 – Algorithme de Berlekamp–Massey

Julien Lavauzelle

9 octobre 2024

L'objectif de ce TP est d'implanter et d'étudier **l'algorithme de Berlelamp–Massey**, qui permet d'obtenir les polynômes de connexion minimaux de suite récurrentes linéaires (ou de manière équivalente, le polynôme de rétroaction minimal d'un LFSR). Par simplicité, nos suites/LFSR seront à coefficients dans le corps fini premier $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, où p est donc un nombre premier quelconque.

Pour ce TP, vous aurez besoin des fichiers `polynomial.py` et `sequence.py`, disponibles sur la page web du cours

`https://www.lvzl.fr/teaching/2024-25/aa.html`

— Le fichier `polynomial.py` contient une classe `Polynomial`, presque entièrement implantée, qui permettra de représenter des polynômes à une variable.
— Le fichier `sequence.py` contient le squelette d'une classe `LFSR` qui permettra de représenter un LFSR ; vous aurez à le compléter pour y ajouter votre implémentation de l'algorithme de Berlekamp–Massey.

Vous aurez également besoin des fichiers `matrix.py` et `timer.py` utilisés lors du TP1.

Manipulation de polynômes

Dans cette partie, l'objectif est de comprendre l'écriture de la classe `Polynomial` donnée dans le fichier `polynomial.py`, afin de pouvoir manipuler des polynômes dans $\mathbb{F}_p[x]$.

Question 1.– Lire attentivement la classe `Polynomial`. Puis, dans un fichier de test que vous aurez créé, simuler les instructions suivantes :
1. La création des polynômes $A(x) = 1 + 2x + x^3$ et $B(x) = 2 + x^2$ dans $\mathbb{F}_3[x]$.
2. Le calcul de la somme $A(x) + B(x)$ et du produit $A(x)B(x)$.
3. Le calcul du reste et du quotient de la division euclidienne de $A(x)$ par $B(x)$.
4. La création du polynôme x^{1000} dans $\mathbb{F}_3[x]$.
5. La création du polynôme nul.
6. La création d'un polynôme aléatoire de $\mathbb{F}_{101}[x]$ de degré 10.

On souhaite maintenant implanter une **méthode de Horner générique** pour le calcul de l'opération

$$(P(x), v, b) \longmapsto P(v)b$$

où $P(x)$ est un polynôme de $\mathbb{F}_p[x]$, et où (v, b) pourra être

— ou bien un couple d'éléments de $\mathbb{F}_p$;
— ou bien un couple formé d'une matrice carrée de taille n et d'un vecteur de taille n (car plus tard, on voudra typiquement évaluer le vecteur $P(\mathbf{V})\boldsymbol{b}$).

Algorithme 1 : Méthode de Horner pour le calcul de $P(v)b$.

Entrée : Un polynôme $P(x) = \sum_{i=0}^{d} p_i x^i \in \mathbb{F}_p[x]$, et un couple d'éléments (v, b) tel que le calcul de $P(v)b$ a du sens.
Sortie : La valeur de $P(v)b$.

```
res ← p_d · b
Pour tout i allant de d − 1 à 0 faire
    res ← v · res
    res ← res + p_i · b
Retourner res.
```

Rappelons la méthode de Horner dans l'Algorithme 1.

Question 2.– Implantez l'Algorithme 1 dans la méthode `horner_method_generic(self, value, b)` que vous trouverez à la fin de la classe `Polynomial`.

Puis, testez votre implantation dans votre fichier de test :

— avec le polynôme $P(x) = x^7 + 2x^4 + 1 \in \mathbb{F}_3[x]$ et les éléments $v = 1$ et $b = 2$ (vous devriez obtenir comme résultat la valeur $2 \in \mathbb{F}_3$) ;

— avec le polynôme $P(x) = x^4 + x^2 + x + 1 \in \mathbb{F}_2[x]$ et les éléments $v = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ et $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ (vous devriez obtenir comme résultat le vecteur $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$).

Remarque : pour créer un vecteur, on crée simplement une matrice avec une seule colonne.

Question 3.– Vérifiez expérimentalement que la complexité en temps de votre implantation de la méthode de Horner est linéaire en le degré du polynôme.

Génération de suites par des LFSR

Dans le fichier `sequence.py`, vous trouverez une classe `LFSR` qui permet de modéliser un LFSR.

Question 4.– Lire attentivement la classe `LFSR`, et bien comprendre le type et le rôle de ses attributs. En particulier, observer la présence d'un registre initial.

Rappelons qu'un LFSR formé d'un polynôme de rétroaction $P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_d x^d \in \mathbb{F}_p[x]$ et d'un registre $R = (r_0, \ldots, r_{L-1}) \in \mathbb{F}_p^L$ de longueur L permet d'engendrer une suite récurrente linéaire donnée par la relation :

$$p_0\, r_n + p_1\, r_{n-1} + \cdots + p_d\, r_{n-d} = 0, \quad \forall n \geq L\,.$$

Question 5.– Implanter la méthode `generate_sequence(self, length)`, qui doit retourner la liste des `length` premières valeurs que le LFSR `self` engendre à partir de son registre initial.

Puis, testez votre méthode en générant les séquences de longueur 12 avec :

1. le LFSR de polynôme de rétroaction $P(x) = 1 + x^2 + x^3 \in \mathbb{F}_2[x]$ initialisé avec le registre `[1, 0, 0, 1]` ; vous devriez obtenir la séquence :

   ```
   [1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0]
   ```

2. le LFSR de polynôme de rétroaction $P(x) = 1 + 3x + x^2 \in \mathbb{F}_5[x]$ initialisé avec le registre `[1, 1]` vous devriez obtenir la séquence :

   ```
   [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
   ```

Algorithme de Berlelamp–Massey

Nous avons maintenant tous les objets utiles pour implanter l'**algorithme de Berlekamp–Massey** que nous avons vu en cours et dont nous rappelons les détails dans l'Algorithme 2.

Question 6.– Implantez la fonction `berlekamp_massey(seq, p)`, qui a été préalablement déclarée en fin de fichier `sequence.py`. La fonction prend en entrée une suite `seq` d'éléments de $\mathbb{F}_p$, et retourne une suite de LFSR (ou de couples polynôme/entiers) minimaux qui engendrent successsivement les termes de la suite `seq`.

Puis, testez votre fonction avec :

1. la suite `[0, 1, 1, 1, 1, 1, 1, 1, 1, 1]` de longueur 10, sur $\mathbb{F}_2$; vous devriez obtenir les LFSR :
   - $(P_0, L_0) = (P_1, L_1) = (1, 0)$,
   - $(P_2, L_2) = (1, 2)$,
   - $(P_k, L_k) = (1 + x, 2)$ pour tout $k \geq 3$.
2. la suite `[1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1]` de longueur 12, sur $\mathbb{F}_2$; vous devriez obtenir les LFSR :
   - $(P_0, L_0) = (1, 0)$,
   - $(P_1, L_1) = (1, 1)$,
   - $(P_2, L_2) = (P_3, L_3) = (1 + x, 1)$,
   - $(P_k, L_k) = (1 + x + x^3, 3)$ pour tout $k \geq 4$.
3. la suite `[1, 0, 0, 16, 6, 0, 6, 9, 2, 15, 16, 8]` de longueur 12, sur $\mathbb{F}_{17}$; vous devriez obtenir les LFSR :
   - $(P_0, L_0) = (1, 0)$,
   - $(P_1, L_1) == (P_2, L_2) = (P_3, L_3) = (1, 1)$,
   - $(P_4, L_4) = (1 + x^3, 3)$,
   - $(P_5, L_5) = (1 + 6x + x^3, 3)$,
   - $(P_k, L_k) = (1 + 6x + 2x^2 + x^3, 3)$ pour tout $k \geq 6$.

Question 7.– Étudiez expérimentalement la complexité en temps de votre implantation de l'algorithme de Berlekamp–Massey, et proposez une conjecture sur sa complexité asymptotique théorique.

Algorithme 2 : Algorithme de Berlekamp–Massey.

Entrée : Une "suite tronquée" $\boldsymbol{s} = (s_0, \ldots, s_{n-1})$ de n éléments de $\mathbb{F}_p$.

Sortie : Deux suites $P = (P_0, \ldots, P_n)$ et $L = (L_0, \ldots, L_n)$ telles que (P_i, L_i) est un LFSR minimal qui engendre la suite $\boldsymbol{s}$ sur i termes.

$P_0 \leftarrow 1$ et $L_0 \leftarrow 0$
$i \leftarrow 0$
Tant que $i < n$ *et* $s_i = 0$ **faire**
    $i \leftarrow i + 1$
    $P_i \leftarrow 1$ et $L_i \leftarrow 0$
$i \leftarrow i + 1$
$P_i \leftarrow 1$ et $L_i \leftarrow i$
$j \leftarrow i - 1$
$\beta \leftarrow \sum_{k=0}^{j} P_j[k]\, s_{j-k}$ où $P_j[k]$ est le coefficient de degré k du polynôme $P_j(x)$
Tant que $i < n$ **faire**
    $\alpha \leftarrow \sum_{k=0}^{i} P_i[k]\, s_{i-k}$
    Si $\alpha = 0$
        $P_{i+1} \leftarrow P_i$ et $L_{i+1} \leftarrow L_i$
    Sinon
        $P_{i+1} \leftarrow P_i - \frac{\alpha}{\beta} \cdot X^{i-j} \cdot P_j$
        Si $L_i > i/2$
            $L_{i+1} \leftarrow L_i$
        Sinon
            $L_{i+1} \leftarrow i + 1 - L_i$
            $j \leftarrow i$
            $\beta \leftarrow \alpha$
    $i \leftarrow i + 1$
Retourner les listes P et L.