Polynômes irréductibles et corps finis

Introduction à la cryptographie, TP du 10 avril 2009

`www-fourier.ujf-grenoble.fr/~eiserm/cours#crypto`

OBJECTIFS

Dans ce projet on se propose d'implémenter quelques algorithmes pour les polynômes irréductibles sur un corps $\mathbb{Z}/n := \mathbb{Z}/n\mathbb{Z}$ où $n \geq 2$ est premier, dans l'objectif de construire le corps fini de cardinal n^d pour d donné.

Les programmes sont à rédiger en C/C++, ou Java, ou tout autre langage avec accord préalable. Les exemples explicités ci-dessus sont écrits en C++. Leur traduction en d'autres langages devrait être facile mais prendra un peu de temps.

On accordera un soin particulier à la rédaction du code source :
– Idéalement l'implémentation se fait dans un fichier source unique.
– Commencer par les commentaires usuels : auteur(s), date, objectifs.
– Commenter chaque fonction par sa spécification (pré- et postconditions).
– Veiller à la lisibilité : noms parlants, style cohérent, bonne indentation, ...
– Vos réponses aux questions annexes peuvent être jointes en commentaire.

Le fichier final — abondamment testé, vérifié, relu — est à envoyer jusqu'au 1er mai 2009 à `Michael.Eisermann@ujf-grenoble.fr`

1. CONVENTIONS ET PRÉPARATIONS

Pour l'implémentation nous aurons besoin de quelques bibliothèques :
– L'anneau quotient $A/(Q)$ d'un anneau euclidien A.
– Arithmétique des nombres entiers $\mathbb{Z}$ puis du quotient $\mathbb{Z}/n$.
– Arithmétique des polynômes $\mathbb{Z}/n[X]$ puis du quotient $\mathbb{Z}/n[X]/(Q)$.

En C++ de telles implémentations (`integer.cc` , `polynome.hh` , `quot.hh`) sont disponibles en ligne sous `~eiserm/Enseignement/crypto/tp-polynomes/`.

Le fichier `polynome.cc` explicite les fonctions à implémenter et rajoute quelques tests. Vous pouvez déjà le compiler avec `g++ -lgmpxx polynome.cc` et le tester.

2. POLYNÔMES IRRÉDUCTIBLES ET CORPS FINIS

2.1. Implémenter la puissance dichotomique modulaire en une fonction

```
Poly puissance( Poly a, Integer n, Poly m ) .
```

2.2. Implémenter l'algorithme d'Euclide-Bézout en trois fonctions

```
Poly pgcd( Poly a, Poly b )
Poly pgcd( Poly a, Poly b, Poly& u )
Poly pgcd( Poly a, Poly b, Poly& u, Poly& v )
```

qui calculent $d = \text{pgcd}(a,b)$, le pgcd unitaire (ou nul). La deuxième variante calcule en même temps u tel que $au \equiv d \pmod{b}$, alors que la troisième variante calcule u, v tel que $au + bv = d$.

2.3. Afin de déterminer si $P \in \mathbb{Z}/n[X]$ est irréductible, écrire une fonction efficace

```
bool irreductible( const Poly& p ) .
```

À titre d'exemple, lesquels des polynômes suivants sont irréductibles sur $\mathbb{Z}/5$: $a := 2+2X+2X^2+X^3$? $b := a+X$? $c := b+X$? Comment vérifier les réponses de votre fonction par un calcul indépendant (à la main) ?

2.4. Dans notre implémentation la multiplication et la division euclidienne sont de complexité quadratique dans le degré. Détailler la complexité des fonctions `puissance(a,n,m)`, `pgcd(a,b)` et `irreductible(p)`.

3. Polynômes irréductibles et corps finis

3.1. Écrire une fonction efficace

```
Poly irreductible_aleatoire( Degre d )
```

qui construit un polynôme unitaire irréductible aléatoire de degré d. Produire ainsi de polynômes irréductibles sur $\mathbb{Z}/5$ de degré $2,3,4,\ldots,100$.

Jusqu'à quel degré environ le temps de calcul vous semble-t-il raisonnable ? Empiriquement, combien d'essais faut-il environ pour trouver un polynôme irréductible ? Quelle en est la prévision théorique ?

3.2. Écrire une fonction efficace

```
Integer ordre(Poly q, const Poly& p)
```

qui détermine l'ordre de l'élément $\bar{Q}$ dans le groupe multiplicatif $K^\times$ du corps $K = \mathbb{Z}/n[X]/(P)$. (Penser à attraper le cas erroné où $\bar{Q} = 0$.)

À titre d'exemple déterminer l'ordre de $\bar{X}$ dans $\mathbb{Z}/5[X]/(2+X^2)$.

Ajouter d'autres exemples, avec des polynômes P de plus en plus grands. Pour quels degrés la méthode vous semble-t-elle suffisamment efficace ?

4. Racines primitives dans un corps fini

4.1. Écrire une fonction efficace

```
Poly racine_primitive(const Poly& p)
```

qui cherche une racine primitive aléatoire du corps $K = \mathbb{Z}/n[X]/(P)$ en implémentant la méthode vue en cours. Essayer de ne pas faire appel à la fonction `ordre(q,p)` mais d'implémenter un test plus spécifique et plus efficace.

À titre d'exemple trouver une racine primitive de $\mathbb{Z}/5[X]/(2+X^2)$.

Ajouter d'autres exemples, avec des polynômes P de plus en plus grands. Pour quels degrés la méthode vous semble-t-elle suffisamment efficace ?

4.2. On peut aussi chercher la plus petite racine primitive dans un ordre spécifié. Dans $\mathbb{Z}/n$ on teste successivement $1,2,3,\ldots$. Dans $\mathbb{Z}/n[X]/(P)$ où $\deg(P) > 1$ on teste successivement $\bar{X}, 1+\bar{X}, 2+\bar{X}, \ldots, 2\bar{X}, 1+2\bar{X}, \ldots, \bar{X}^2, \ldots$.

Pour cela implémenter une fonction optimisée

```
bool est_racine_primitive( q, p, cardinal, decomp )
```

qui détermine efficacement si $\bar{Q}$ est une racine primitive dans $\mathbb{Z}/n[X]/(P)$. Essayer de ne pas faire appel à la fonction `ordre(q,p)` mais d'implémenter un test plus spécifique et plus efficace.

À titre d'exemple trouver la plus petite racine primitive de $\mathbb{Z}/5[X]/(2+X^2)$.

Ajouter d'autres exemples, avec des polynômes P de plus en plus grands. Pour quels degrés la méthode vous semble-t-elle suffisamment efficace ?