Arithmétique modulaire, primalité

1 Exercices

Exercice 1.1 (Polynômes sans facteurs carrés)**.**
(*a*) Rappeler pourquoi pour tout corps K et tout $P \in K[X]$, P est sans facteurs carrés (dans sa décomposition en facteurs irréductibles dans $K[X]$) si et seulement si $\mathrm{pgcd}(P, P') = 1$.
(*b*) Soit $P = X^4 + 13X + 1 \in \mathbb{Z}/p\mathbb{Z}[X]$. Est-il sans facteurs carrés pour $p = 7$? $p = 11$?

Exercice 1.2 (Inverses dans des anneaux quotients de polynômes)**.**
Soit $P = X^3 + X + 1 \in \mathbb{Q}[X]$. Déterminer l'inverse de $X^2 + X + 1$ dans $\mathbb{Q}[X]/(P)$.

Exercice 1.3 (Écriture en base b)**.**
Décrire algorithmiquement comment écrire en base b un entier n. Quel est le coût de cette écriture ?

Exercice 1.4 (PGCD binaire)**.**
(*a*) Effectuer à la main l'algorithme du PGCD binaire pour 123 et 57 (on partira de leurs écritures binaires directement).
(*b*) Montrer que l'algorithme du PGCD binaire appliqué à 2 entiers de tailles respectives au plus m et n bits nécessite au plus $O(mn)$ opérations élémentaires du microprocesseur.

Exercice 1.5 (Algorithme d'Euclide étendu)**.**
(*a*) Rappeler comment une relation de Bézout entre deux entiers a et b premiers entre eux permet de calculer l'inverse de a modulo b (et inversement).
(*b*) Effectuer à la main l'algorithme d'Euclide étendu pour 123 et 57.
(*c*) En déduire que le pgcd de 123 et 57 est 3, puis calculer l'inverse de 19 modulo 41.

Exercice 1.6 (Calcul plus rapide de l'inverse modulaire)**.**
(*a*) Montrer que pour $a, n \in \mathbb{Z}$ premiers entre eux, on peut gagner environ un tiers des calculs pour celui de l'inverse de a modulo n par rapport au calcul des coefficients de Bézout pour a et n.
(*b*) Illustrer sur l'exemple de l'exercice précédent.

Exercice 1.7 (Théorème des restes chinois)**.**
(*a*) Pour a et b deux entiers premiers entre eux et des choix quelconques de $x, y \in \mathbb{Z}$, utiliser une relation de Bézout entre a et b pour calculer un entier n tel que $n \equiv x \bmod a$ et $n \equiv y \bmod b$. Quel est le coût du calcul complet ?
(*b*) Supposons de plus que a est de taille $O(k)$ et b de taille $O(1)$, montrer qu'alors le coût est en $O(k)$.
(*c*) Illustrer avec $a = 143$, $b = 5$, $x = 100$ et $y = 1$.

Exercice 1.8 (Déterminant d'une matrice entière par restes chinois)**.**
Soit $M \in M_n(\mathbb{Z})$ dont les coefficients sont en valeur absolue plus petits qu'une certaine borne A.
(*a*) Montrer que $|\det(M)| \leq (\sqrt{n}A)^n$ (utiliser la borne de Hadamard).
(*b*) Soit $M \in M_3(\mathbb{Z})$ à coefficients plus petits que 10 en valeur absolue. On a calculé le déterminant de M modulo 19, 23 et 29 et trouvé 1, 2 et 3. Que vaut le déterminant de M ?

(c) Combien de nombres premiers de taille proche de A faut-il utiliser pour reconstruire le déterminant de M en utilisant les restes chinois et la valeur du déterminant de M modulo ces nombres premiers ?
(d) Estimer le coût de l'algorithme si le coût de calcul du déterminant modulo un nombre premier est en $O(n^3)$.

Exercice 1.9 (Exponentiation rapide d'entiers modulaires)**.**
(a) Utiliser à la main l'algorithme de la puissance rapide (aussi appelé exponentiation rapide) pour calculer $\overline{3}^{1030}$ dans $\mathbb{Z}/19\mathbb{Z}$.
(b) Vérifier la validité de ce calcul en utilisant que 19 est premier.

Exercice 1.10 (Exponentiation naïve vs. exponentiation rapide d'entiers)**.**
Soit a un entier. Comparer le cout du calcul de a^n en utilisant la méthode de multiplication naive ($a \times a \times ... \times a$) et le même algorithme que l'exponentiation modulaire, mais sans modulo.

Exercice 1.11 (Résolution d'équations quadratiques dans les $\mathbb{Z}/n\mathbb{Z}$)**.**
(a) Résoudre dans $\mathbb{Z}/103\mathbb{Z}$ les équations du second degré :

$$x^2 + 3x + 5 = 0, \quad x^2 + 3x + 6 = 0.$$

(b) Pour $p > 2$ premier, montrer que pour tout $a \in (\mathbb{Z}/p\mathbb{Z})^*$, a est un carré si et seulement si $a^{(p-1)/2} = 1 \bmod p$. Si de plus $p = 3 \bmod 4$, montrer qu'alors une racine carrée de a modulo p est donnée par $a^{(p+1)/4} \bmod p$.
(c) Estimer le coût pour résoudre une équation de degré 2 dans un corps $\mathbb{Z}/p\mathbb{Z}$ où $p = 3 \bmod 4$ est un nombre premier de taille $O(n)$.

Exercice 1.12 (Générateurs des $(\mathbb{Z}/p\mathbb{Z})^*$)**.**
Estimer le coût moyen pour trouver un générateur de $(\mathbb{Z}/p\mathbb{Z})^*$ avec p un nombre premier de taille $O(n)$.

Exercice 1.13 (Irréductibilité de polynômes)**.**
(a) Le polynôme $P = X^4 + X + 2 \in \mathbb{Z}/3\mathbb{Z}[X]$ est-il irréductible ? 3 ?
(b) Estimer le coût d'un test d'irréductibilité de $P \in (\mathbb{Z}/p\mathbb{Z})[X]$ en fonction du degré de P et de la taille de p.
(c) Déterminer le coût moyen de construction d'un polynôme irréductible de degré n dans $(\mathbb{Z}/p\mathbb{Z})[X]$.

2 TP

2.1 Installation et utilisation de Xcas

2.1.1 Sans installation

Vous pouvez travailler avec Xcas avec la version web
`https://www-fourier.univ-grenoble-alpes.fr/~parisse/xcasfr.html`

ou sur une calculatrice compatible (Casio Graph 90, HP Prime, Numworks N0110, TI Nspire). La version web permet d'échanger très facilement des sessions par email ou de les publier sur des forums. Ces versions de Xcas sont compatibles entre elles (par exemple : menu Fich/Clone pour passer de Xcas PC à Xcas web).

2.1.2 Installation de Xcas sur votre machine (recommandé)

Vous pouvez installer Xcas sur votre ordinateur avec ce lien :
`https://www-fourier.univ-grenoble-alpes.fr/~parisse/install_fr`

2.1.3 Installation et lancement de Xcas sur votre session étudiante (recommandé)

Après des manipulations qui vous seront expliquées en séance, il suffira de lancer dans un terminal `~parisseb/bin/runxcas`

2.2 Principe de fonctionnement de Xcas

Xcas est un shell, un peu comme en Python : on tape une ligne de commande ou un programme dans un éditeur de programmes, on valide par Enter ou OK, et Xcas renvoie une valeur ou affiche un graphe ou indique une erreur.

- Pour vous familiariser avec les commandes de Xcas et leur syntaxe vous pouvez utiliser les menus de Xcas (Outil, Expression, Graphes, Cmds), les assistants mathématiques dans Xcas web, ou les menus sur calculatrices, ainsi que l'index des commandes (menuq Aide, Index dans Xcas).
- La touche de tabulation permet de compléter un début de nom de commande ou/et d'afficher l'aide sur cette commande. On peut recopier un exemple de commande de l'aide et modifier les valeurs des arguments. Ou suivez le tutoriel de Xcas (menu Aide, Débuter en calcul formel).
- Pour saisir une matrice (liste de listes de même taille), on peut utiliser la commande `matrix`, ou entrer les coefficients un par un en ligne de commandes (par exemple `[[1,a],[a,1]]`) ou avec le tableur (Tableur, nouveau tableur, donner un nom de variable et entrer les coefficients).
- Pour écrire un programme, vous pouvez choisir la syntaxe compatible Python de Xcas, la structuration d'un programme est alors identique à Python (`def/return, if/else, for/while`) (Xcas propose d'autres syntaxes : mots-clefs en français ou compatibles avec C/Javascript.). Pour vérifier dans quelle syntaxe Xcas fonctionne actuellement, vérifiez la configuration du logiciel dans la barre d'état, en particulier la syntaxe (compatible Python ou en français) et modifiez-la si nécessaire.
- Pour mettre au point un programme, vous pouvez utiliser le débugger qui permet d'exécuter en pas à pas un programme (commande `debug`).
- **Faites des sauvegardes régulièrement, faites-en systématiquement avant de tester un programme** (Xcas a un mécanisme de sauvegarde automatique en cas de crash, mais deux précautions valent mieux qu'une). Lorsque vous ouvrez une session sauvegardée, le contenu des variables est restauré, sauf si vous avez ouvert en mode de récupération. Notez que le menu Edit, Exeécuter session, permet de réexécuter toute la session.

2.3 Prise en main de Xcas

(*a*) Écrire le polynôme $(x+3)^7 \times (x-5)^6$ selon les puissances décroissantes de x.

(*b*) Simplifier les expressions suivantes :

$$\sqrt{3+2\sqrt{2}}, \quad \frac{1+\sqrt{2}}{1+2\sqrt{2}}, \quad e^{i\pi/6}, \quad 4\arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right)$$

(*c*) Factoriser :

$$x^8 - 3x^7 - 25x^6 + 99x^5 + 60x^4 - 756x^3 + 1328x^2 - 960x + 256$$

$$x^6 - 2x^3 + 1, \quad (-y+x)z^2 - xy^2 + x^2y$$

(dans quel anneau de polynômes ces factorisations sont-elles calculées par défaut ?)

(*d*) Calculer les sommes suivantes

$$\sum_{k=1}^{N} k, \quad \sum_{k=1}^{N} k^2, \quad \sum_{k=1}^{\infty} \frac{1}{k^2}$$

(*e*) Trouver les entiers n tels que le reste de la division euclidienne de $123n$ par 256 soit 17.
(*f*) Déterminer la liste des diviseurs de 45768. Factoriser 100 !
(*g*) Résoudre le système linéaire :

$$\begin{cases} x & + & y & + & az & = & 1 \\ x & + & ay & + & z & = & 2 \\ ax & + & y & + & z & = & 3 \end{cases}$$

Déterminer l'inverse de la matrice du système ci-dessus, puis la diagonaliser.

2.4 TP d'arithmétique

Exercice 2.1 (Vérification des calculs)**.**
Vérifier (ou faire) les calculs faits dans les exercices plus haut.

Exercice 2.2 (Programmation des algorithmes évoqués)**.**
Écrire les algorithmes mentionnés ou considérés dans les exercices de TD, à savoir :

— Le test si un polynôme est sans facteurs carrés de l'exercice 1.1.
— L'écriture en base b d'un entier écrit dans une base autre (par exemple 10) de l'exercice 1.3.
— L'algorithme d'Euclide étendu, le calcul d'inverse modulaire et le théorème des restes chinois pour deux entiers premiers entre eux des exercices 1.5 et 1.7 (bonus : l'étendre pour n entiers premiers entre eux deux à deux).
— L'exponentiation rapide d'entiers modulaires de l'exercice 1.9.

Exercice 2.3.
Tester le temps de calcul du produit de grands entiers.

Exercice 2.4 (Avantages de l'exponentiation rapide)**.**
Comparer le temps de calcul de $a^n \bmod m$ par la fonction `powmod` et celui donné en calculant a^n puis en prenant son reste modulo m.

Exercice 2.5 (Générateur de $(\mathbb{Z}/p\mathbb{Z})^*$ sous condition)**.**
Soit p premier tel qu'on connaît tous les facteurs premiers d de $p-1$.

(*a*) Montrer que $a \in \mathbb{Z}$ premier à p engendre $(\mathbb{Z}/p\mathbb{Z})^*$ si et seulement si $a^{(p-1)/d} \neq 1 \bmod p$ pour tout facteur premier d de $p-1$.
(*b*) Écrire un algorithme de recherche de générateur de $(\mathbb{Z}/p\mathbb{Z})^*$ basé sur cette propriété et comparer avec celle du coût moyen de l'exercice 1.12.

Exercice 2.6 (Test de primalité naïf et crible d'Eratosthène)**.**
(*a*) Écrire un algorithme testant si un nombre n est premier par division. Discuter sa complexité et expérimenter.
(*b*) Pour $n \geq 2$, écrire l'algorithme du crible d'Eratosthène renvoyant la liste des nombres premiers entre 2 et n (cet algorithme ne doit pas utiliser de division, seulement des additions). Estimer sa complexité.

Exercice 2.7 (Racine carrée modulo p)**.**
Soit p un nombre premier.

(*a*) Écrire un programme trouvant (si elle existe) une racine carrée d'un entier n modulo p.

(*b*) Si $p \equiv 3 \bmod 4$, utiliser l'exercice 1.11 pour un algorithme plus efficace.

Exercice 2.8 (Proto-test de Fermat et nombres de Carmichael)**.**

(*a*) Écrire un test de non-primalité basé sur le petit test de Fermat : $a^{p-1} = 1 \bmod p$ pour tout entier $a \in \zeta$ premier à p lorsque p est premier.

(*b*) Faire la liste des entiers entre 1 et 5000 qui ne sont pas premiers mais pour lesquels le test renvoie vrai (on les appelle nombres de Carmichael).

(*c*) Majorer la complexité de cette recherche.

Exercice 2.9 (Test de Miller–Rabin)**.**

(*a*) Programmer le test de Miller–Rabin.

(*b*) Vérifier qu'il détecte bien les nombres de Carmichael (cf. exercice 2.8) comme non premiers.

(*c*) Pour $n = 561$, trouver tous les entiers $a \in [1, 560]$ qui passent le test de Miller-Rabin

Exercice 2.10 (Certificat de primalité)**.**

Comprendre et détailler le certificat de primalité renvoyé par la commande Xcas ou PARI/GP

```
isprime(9856989898997789789,1)
```