1 TD

Exercice 0 Quel est le cout de l'écriture en base b d'un entier n ?

Exercice 1 Effectuer à la main l'algorithme du PGCD binaire pour 123 et 57. Montrer que l'algorithme du PGCD binaire appliqué à 2 entiers de taille au plus n bits nécessite au plus $O(n^2)$ opérations élémentaires (du microprocesseur).

Exercice 2 Effectuer à la main l'algorithme d'Euclide étendu pour 123 et 57. En déduire l'inverse de 19 mod 41.

Exercice 3 Montrer que le calcul de l'inverse de a modulo n peut se faire environ un tiers plus rapidement que le calcul des coefficients de Bézout pour a et n. Illustrer sur l'exemple de l'exercice 2.

Exercice 4 Soit P un entier de taille $O(n)$ et q un entier de taille $O(1)$ avec P et q premiers entre eux (par exemple P est le produit de n nombres premiers de taille $O(1)$). Pour déterminer C tel que $C = A \pmod P$ et $c = b \pmod q$ on résout $A + uP = b + vq$ puis on pose $C = b + vq = A + uP$. Montrer que le cout est en $O(n)$. Illustrer avec $P = 143, q = 5, A = 100, b = 1$ (toutes les opérations doivent avoir au moins un argument à 1 chiffre).

Exercice 5 Soit M une matrice à coefficients entiers de taille n dont les coefficients sont en valeur absolue plus petits que A.

1. Montrer que le déterminant de M est en valeur absolue plus petit que $(\sqrt{n}A)^n$.
2. Soit M une matrice de taille 3 à coefficients entiers plus petits que 10. On a calculé le déterminant de M modulo 19, 23 et 29 et trouvé 1, 2 et 3. Que vaut le déterminant de M ?
3. 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 ?
4. Estimer le cout de l'algorithme si le cout de calcul du déterminant modulo un nombre premier est en $O(n^3)$.

Exercice 6 Effectuer papier-crayon l'algorithme de la puissance rapide (aussi appelé exponentiation modulaire) pour calculer $\overline{3}^{1030}$ dans $\mathbb{Z}/19\mathbb{Z}$. Vérifiez en utilisant que 19 est premier.

Exercice 7 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 8 Résoudre dans $\mathbb{Z}/103\mathbb{Z}$ les équations du second degré :

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

Exercice 9 Estimer le cout pour résoudre une équation de degré 2 dans un corps $\mathbb{Z}/p\mathbb{Z}$ où p est un premier congru à 3 mod 4 de taille $O(n)$.

Exercice 10 Déterminer un générateur de $\mathbb{Z}/113\mathbb{Z}^*$.

Exercice 11 Estimer le cout moyen pour trouver un générateur de $\mathbb{Z}/p\mathbb{Z}^*$ avec p un premier de taille $O(n)$.

2 TP

Vous pouvez travailler avec Xcas en l'installant sur votre propre ordinateur
`https://www-fourier.univ-grenoble-alpes.fr/~parisse/install_fr`
ou la version web sans installation depuis votre navigateur
`https://www-fourier.univ-grenoble-alpes.fr/~parisse/xcasfr.html`
ou sur une calculatrice compatible (Casio Graph 90/35eii, 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)
Sur les PC de la salle Carism, vous pouvez utiliser la version de Xcas installée en tapant `xcas &` ou recompiler
`https://www-fourier.univ-grenoble-alpes.fr/~parisse/giac/giac-1.9.0.tar.gz`
`tar xvfa ~/Téléchargements/giac-1.9.0.tar.gz ; cd giac-1.9.0 ; ./configure --prefix=$HOME`
Puis compilez `make -j` et installez `make install`, puis créez un fichier `runxcas` contenant
`export LD_LIBRARY_PATH=$HOME/lib ; export XCAS_ROOT=$HOME/share/giac ; ~/bin/xcas &`
qui permet de lancer Xcas en tapant `sh runxcas`

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 (menu Outil, Expression, Graphes, Cmds), les assistants mathématiques dans Xcas web, ou les menus sur calculatrices, ainsi que l'index des commandes (menu 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 compatible avec C/Javascript. 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, 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.

2.1 Prise en main

1. Écrire le polynôme $(x+3)^7 \times (x-5)^6$ selon les puissances décroissantes de x.
2. Simplifier les expressions suivantes :
$$\sqrt{3+2\sqrt{2}}, \quad \frac{1+\sqrt{2}}{1+2\sqrt{2}}, \quad e^{i\pi/6}, \quad 4\mathrm{atan}(\frac{1}{5}) - \mathrm{atan}(\frac{1}{239})$$
3. Factoriser :
$$x^8 - 3x^7 - 25x^6 + 99x^5 + 60x^4 - 756x^3 + 1328x^2 - 960x + 256$$
$$x^6 - 2x^3 + 1, \quad (-y+x)z^2 - xy^2 + x^2y$$
4. Calculer les sommes suivantes
$$\sum_{k=1}^{N} k, \ \sum_{k=1}^{N} k^2, \ \sum_{k=1}^{\infty} \frac{1}{k^2}$$
5. Trouver les entiers n tels que le reste de la division euclidienne de $123n$ par 256 soit 17.
6. Déterminer la liste des diviseurs de 45768. Factoriser 100!
7. 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.

2.2 Arithmétique modulaire et primalité

Exercice 0 Vérifier à la machine les exercices calculatoires de la partie TD.

Exercice 1 Tester le temps de calcul du produit de grands entiers. Peut-on associer un algorithme du cours ?

Exercice 2 Programmer l'algorithme de Bézout et les restes chinois.

Exercice 3 Comparer le temps de calcul de $a^n \pmod m$ par la fonction `powmod` et la méthode prendre le reste modulo m après avoir calculé a^n. Implémentez l'exponentiation modulaire rapide.

Exercice 4 Soit p un nombre premier tel que la factorisation de $p-1$ soit connue. Écrire une fonction renvoyant un générateur de $\mathbb{Z}/p\mathbb{Z}^*$. Indication : prendre a au hasard, vérifier que $a^{(p-1)/d} \neq 1 \pmod p$ pour tous les diviseurs premiers d de $p-1$.

Exercice 5 Implémenter l'exercice 5 de TD pour calculer le déterminant d'une matrice à coefficients entiers.

Exercice 6 Écrire un programme naïf testant si p est premier par division. Discuter sa complexité et expérimentez.

Exercice 7 Écrire un programme calculant une racine carrée (si elle existe) d'un entier n modulo p pour p premier.

Exercice 8 Écrire un test de non primalité basé sur l'identité $a^{p-1} = 1 \pmod p$ pour p premier et a premier avec p. Déterminer les entiers non premiers plus petits que 1000 pour lesquels le test renvoie vrai. Majorer la complexité de cette recherche.

Exercice 9 Écrire un algorithme déterminant la liste des nombres de Carmichael (i.e. les entiers $n > 1$ tels que n n'est pas premier mais $a^{n-1} = 1 \pmod n$ si a et n sont premiers entre eux) plus petits qu'un entier N fixé. Estimer sa complexité.

Exercice 10 Programmer le test de Miller-Rabin. Vérifier qu'il détecte les nombres non premiers non détectés dans l'exercice 1. Pour $n = 561$, déterminez tous les entiers $a \in [1, 560]$ qui passent le test de Miller-Rabin, vérifiez qu'il y a (nettement) moins d'un quart de valeurs de a qui renvoient Vrai pour ce test.

Exercice 11 Détaillez le certificat de primalité renvoyé par la commande Xcas ou PARI/GP `isprime(9856989898997789789,1)`.