

Exercice I:

- 1) a) Programmez les algorithmes du pgcd récursif et itératif pour les entiers (cf. la section 2.1). Comment modifier votre programme pour en avoir une version pour les polynômes ?
- b) Testez aussi avec des valeurs assez grandes d'entiers. Vérifiez vos résultats avec les commandes internes d'xcas. (Trouvez la bonne commande dans les menus déroulants)
- c) On considère les polynomes suivant :

$$P = x^6 + 6x^5 + 6x^4 + 8x^3 + 42x^2 + 42x + 7, \quad Q = x^6 - 2x^5 + x^4 - x^3 - 2x^2 + x - 2$$

Donnez leur pgcd dans $\mathbb{Q}[x]$ puis dans $\mathbb{Z}/5\mathbb{Z}[x]$. (Avec votre programme, puis avec l'instruction toute prête de votre logiciel)

- 2) a) Créez une fonction **bezout** selon l'algorithme de bezout itératif (cf. 2.2). On le fera d'abord pour les entiers, puis pour les polynômes. Application avec

$$1 + 2x^3 + 5x^4 - x^7, 2 + x - 4x^6 + x^9$$

b) Dans votre logiciel trouvez l'instruction correspondante déjà implémentée, et vérifiez votre programme.

c) Les algorithmes d'Euclide, puissance rapide, ... sont à connaître. Remarquez qu'ils sont détaillés en html interactif et en pdf dans la documentation hors ligne d'Xcas :

(cf. menu **Aide>Manuels>Algorithmes**).

d) Montrer que $K = \mathbb{Z}/23\mathbb{Z}[x]/(x^{23} + x^2 + 1)$ est un corps. Quel est son nombre d'éléments ? (ranger le dans une variable **q**)

e) Quel est l'inverse de $a = 2x^5 + x + 1$ dans K ? (vérifiez)

f) En utilisant une instruction de puissance rapide du logiciel, retrouver l'inverse de a dans K .

g) $x + 3$ est il un carré dans K ?

3) a) Créez une fonction **F(n)** qui retourne le nième terme de la suite de Fibonacci. Comparez le nombre de tours effectués par votre algorithme pour calculer le pgcd de **F(n)** et **F(n+1)** avec celui obtenu pour des nombres de taille similaire. Majorez asymptotiquement en n le nombre de tours dans $gcd(a, b)$ avec $0 \leq a, b \leq n$.

Indication : Si $0 < y < x$ et que l'algorithme du $d = pgcd(x, y)$ se termine en k tours alors :

$$x > d.F(k+2), y > d.F(k+1)$$

b) Xcas possède une instruction permettant de résoudre certaines relations de récurrence. Rappeler nous rapidement la valeur de $F(n)$ par cette méthode et conclure sur la complexité du nombre de tours dans l'algorithme d'Euclide.

Exercice II:

1) Etudiez l'utilisation d'une fonction aléatoire uniforme sur $\mathbb{N} \cap [0, n[$ (et n grand ? et $[a, b[$?). Que pensez vous de la complexité d'un générateur (pseudo) aléatoire uniforme ? On pourra lire dans l'aide du menu : **Aide>Manuels>Algorithmes** la partie correspondant aux générateurs linéaires.

2) Donnez/Expliquez un ordre de grandeur pour qu'une boucle de k tours effectuant un calcul élémentaire soit faisable par un ordinateur (Ex : $10^5, 10^{10}, 10^{20}, 10^{100} \dots$).

3) Illustrez le problème de la factorisation d'entiers (expliquez et illustrez avec des exemples ce qui est facile et ce qui peut poser problème). En TP et en temps limité il vaut mieux se limiter aux temps inférieurs à 2min et expliquer le reste oralement. Pour les instructions longues il vaut mieux ne pas utiliser la fonction **time** sous xcas car les temps longs sont automatiquement affichés

1. <http://webusers.imj-prg/frederic.han/agreg>

et elle supporte mal les interruptions. NB1 : Les versions d'xcas 64 bits qui utilisent la librairie pari peuvent être bien plus performantes que d'autres pour factoriser un entier.

4) a) Etant donné des entiers e, r, n , il existe toujours une méthode naïve pour trouver (lorsque c'est possible) un entier a tel que $a^e = r[n]$. Par exemple, en utilisant une boucle trouvez a tel que $a^{77} = 27980 [69589]$. Cette méthode est-elle envisageable si n a 20 chiffres ?

b) Si n est un nombre premier que vaut $a^{n-1}[n]$? Par exemple avec `n:=nextprime(11**23)`, si e vaut 7 et r vaut 81625902815799101609288 trouvez facilement grâce à cette remarque la valeur de a telle que $a^e = r[n]$.

c) Illustrez sur un exemple la définition de ψ_e suivante et calculez son inverse en vous aidant de p et q (on prendra un exemple où l'ordinateur ne sait pas factoriser n). Vérifiez sur quelques valeurs qu'il convient et expliquez que (p, q) peut servir de secret même si n est publique.

Proposition 0.0.1 *On considère un entier $n = p \cdot q$ où p et q sont des nombres premiers impairs distincts. Pour tout entier e premier avec $\phi(n) = (p-1)(q-1)$, la fonction $\psi_e : (\mathbb{Z}/n\mathbb{Z})^\times \rightarrow (\mathbb{Z}/n\mathbb{Z})^\times, x \mapsto x^e [n]$ est bijective, et la connaissance de p et q permet de trouver sa réciproque en temps polynomial (en $\log n$).*

Remarque 0.0.2 *On ne connaît pas d'algorithme polynomial (en $\log n$) pour trouver les facteurs premiers d'un entier. De plus la donnée de $(n, \phi(n))$ est équivalente (en temps polynomial) à la donnée de (p, q)*

Si vous avez le temps, préparez une illustration complète.

Exercice III: (Cf Ecrit 2007)

1) Choisir un nombre premier p tel que² le problème du log discret soit difficile modulo p . On pourra éventuellement illustrer cette difficulté en chargeant dans xcas une fonction de pari qui calcule le log dans $\mathbb{Z}/p\mathbb{Z}$. (pour charger toutes les fonctions de pari³), on tape : `pari()`, elles deviennent alors accessibles ainsi : `pari_lenomdelafonctionpari`. et pour trouver le nom de la fonction qui vous intéresse regarder : Aide/Manuel/Pari-GP)

2) Choisir un entier b et un secret e et rendre public $(b, b^e[p])$. Commentez votre choix de b .

3) Remarquer que les lettres $A \dots Z$ ont des codes ASCII consécutifs. Comment obtenir les codes de chaque lettre d'un mot ? (et si l'on veut que A soit représenté par 0 et Z par 25 ?)

4) Votre voisin choisit un message, tire des k au hasard et code chaque lettre x avec $(b^k[p], x \cdot (b^e)^k)[p]$ (sans connaître e , seulement avec b et b^e)

5) a) Décoder le message grâce à e .

b) Préparer un petit exposé expliquant la méthode et son intérêt. (temps pour crypter, décrypter avec et sans e . Statistiques sur les lettres du message crypté...)

Exercice IV: Crible quadratique (mini texte)

1) Soit $b_1, \dots, b_k, n$ des entiers naturels, n étant impair. On note a_i l'entier congru à b_i^2 modulo n dont la valeur absolue est majorée par $n/2$ (le reste symétrique). On note $p_1, \dots, p_N$ l'ensemble des diviseurs premiers de tous les a_i , et $p_0 = -1$, et l'on pose $a_i = \prod_{j=0}^N p_j^{\alpha_{i,j}}$, ($\alpha_{i,j}$ pouvant être nul).

a) On suppose que dans $\mathbb{Z}/2\mathbb{Z}$, la matrice $(\alpha_{i,j})$ a un noyau non nul. En déduire un élément $c \in \mathbb{Z}$ exprimé en fonction des p_i et du noyau tel que c^2 soit un produit de a_i .

b) On note b le produit des b_i dont les indices correspondent à ceux du produit précédent. Montrer que si $b \neq \pm c[n]$, alors le $\text{pgcd}(b+c, n) \neq 1$ et $\text{pgcd}(b-c, n) \neq 1$.

2) Dans la pratique, il faut trouver des b_i tels que les a_i soient petits, pour qu'ils se décomposent rapidement, et que les p_i soient petits. De plus, pour avoir $b \neq \pm c[n]$, il faut que $a_i \neq b_i^2$ dans $\mathbb{Z}$, c'est à dire $b_i > \sqrt{n}$.

2. ou que vous supposez tel que

3. logiciel spécialisé en arithmétique dont la librairie est accessible depuis xcas

- a) Illustrez que si $\frac{u_k}{v_k}$ est une réduite du développement en fractions continues de $\sqrt{n}$, on a $|u_k^2 - nv_k^2| < 2\sqrt{n}$, donc que $\text{mods}(|u_k^2|, n) < 2\sqrt{n}$. (pour les fractions continues on peut aussi regarder dans : >Aide>Manuels>Algorithmes)
- b) Tester en prenant pour les b_i quelques u_i .

Chapitre 2

Exemples d'algorithmes itératifs et récursifs

Dans ce chapitre on va mettre l'accent sur l'écriture des algorithmes et leur justification (l'algorithme se termine et produit le bon résultat).

2.1 Deux versions de l'algorithme d'Euclide

Proposition 2.1.1 *Soient : $a \in \mathbb{Z}$, $b \in \mathbb{Z}$. On a pour tout $m \in \mathbb{Z}$:*

$$\text{PGCD}(a, b) = \text{PGCD}(b, a - mb) .$$

On peut en particulier appliquer la proposition précédente au cas où $m = q$ est le quotient dans la division euclidienne de a par b .

Algorithme: Euclide, forme récursive :

Entrée: Deux entiers relatifs : a, b ;
Sortie: Un entier pgcd de a et b ;
Fonction $\text{PGCD}(a, b)$;
si b est nul **alors**
 | retourner a ;
sinon
 | $r \leftarrow a \bmod b$ (reste de la division euclidienne) ;
 | retourner $\text{PGCD}(b, r)$;
fsi

Algorithme: Euclide, forme itérative :

Entrée: Deux entiers relatifs : a, b ;
Sortie: Un entier pgcd de a et b ;
Fonction $\text{PGCD}(a, b)$;
tantque b est non nul **faire**
 | $r \leftarrow a \bmod b$ (reste de la division euclidienne) ;
 | $a \leftarrow b ; b \leftarrow r$;
ftantque
retourner a ;

2.2 Algorithme d'Euclide étendu aux coefficients de Bezout

Proposition 2.2.1 *Soient : $a \in \mathbb{Z}$, $b \in \mathbb{Z}$, $d = \text{PGCD}(a, b)$. Il existe deux entiers u et v tels que $d = ua + vb$.*

Algorithme: Euclide étendu : version itérative

Entrée: 2 entiers a et b

Sortie: Une liste de 3 entiers : (u, v, d) tels que $a.u + v.b = d$

Fonction Bezout(a, b);

$A \leftarrow (1, 0, a)$; //On notera A_i le $i^{\text{ème}}$ opérande de la liste A ;

$B \leftarrow (0, 1, b)$; //On notera B_i le $i^{\text{ème}}$ opérande de la liste B ;

$reste \leftarrow b$;

tantque $reste \neq 0$ **faire**

$q \leftarrow$ le quotient de A_3 par $reste$;

$temp \leftarrow A - q \times B$;

$A \leftarrow B$;

$B \leftarrow temp$;

$reste \leftarrow B_3$;

ftantque

retourner A

Proposition 2.2.2 On considère deux entiers a, b et l'on pose : $r_0 = a$ et $r_1 = b$. On définit par récurrence tant que r_i est non nul : $r_{i+1} = r_{i-1} - q_i.r_i$ où q_i est le quotient de la division Euclidienne de r_{i-1} par r_i . On pose $u_0 = 1, u_1 = 0$ et $v_0 = 0, v_1 = 1$. Alors les suites (u_i, v_i) définies par

$$u_{i+1} = u_{i-1} - q_i.u_i \text{ et } v_{i+1} = v_{i-1} - q_i.v_i$$

tant que r_i est non nul, vérifient la relation suivante :

$$u_i.a + v_i.b = r_i$$

pour toute les valeurs de i où elles sont définies.

Exercice 2.2.3 Notons N le nombre d'itérations de l'algorithme d'Euclide. Alors les suites $(u_i)_{N \geq i \geq 1}$ et $(v_i)_{N \geq i \geq 1}$ sont de signe alterné et $(|u_i|)_{N \geq i \geq 1}$ et $(|v_i|)_{N \geq i \geq 1}$ sont croissantes

2.3 Exemples de coûts

Division euclidienne pour les entiers positifs

Théorème 2.3.1 Pour $a \in \mathbb{N}$, $b \in \mathbb{N}^*$, il existe un unique couple d'entiers (q, r) avec $a = bq + r$ et $0 \leq r < b$.

2.3.1 Méthode naïve :

Algorithme: Division euclidienne, méthode naïve :

Entrée: $a \geq 0$ et $b > 0$;

$q \leftarrow 0$; $r \leftarrow a$;

tantque $r > b$ **faire**

$r \leftarrow r - b$;

$q \leftarrow q + 1$;

ftantque

retourner q et r ;

Le nombre de boucle de cet algorithme est q ; à chaque étape il y a une soustraction et une comparaison; pour b fixé la complexité croît linéairement avec a .

Remarque 2.3.2 Si on mesure la taille de l'entier a par le nombre de chiffres de son développement en binaire : $t = \log_2(a)$, alors la complexité croît exponentiellement avec la taille de a .

2.3.2 Recherche dichotomique

Algorithme: Division euclidienne, recherche dichotomique :

Entrée: $a \geq 0$ et $b > 0$;

Sortie: Le quotient et le reste de la division de a par b

$n \leftarrow 0$;

tantque $2^n b \leq a$ **faire**

$n \leftarrow n + 1$;

ftantque

$\alpha \leftarrow 2^{n-1}$; $\beta \leftarrow 2^n$;

pour k **de** 1 **jusque** $n - 1$ **faire**

$\gamma = \frac{\alpha + \beta}{2}$;

si $\gamma b \leq a$ **alors**

$\alpha \leftarrow \gamma$;

sinon

$\beta \leftarrow \gamma$;

fsi

fpour

retourner $q = \alpha$ et $r = a - bq$;

2.3.3 Puissance rapide

Proposition 2.3.3 Soit $(G, *)$ un groupe commutatif dont on note la loi multiplicativement. On peut calculer a^n en $O(\log(n))$ opérations $*$.

Remarque 2.3.4 On distinguera deux situations assez différentes selon que le coût de $x * y$ est indépendant des éléments x, y de G ou pas. Donnez 2 exemples de G dans chaque situation.

Algorithme: Puissance rapide :

Entrée: Un élément g de G et un naturel n .

Sortie: Un élément de G : g^n

Fonction $\text{puiss}(g, n)$;

$u \leftarrow 1$; $v \leftarrow g$; // u, v : 2 variables locales ;

tantque $n > 1$ **faire**

si n *pair* **alors**

$v \leftarrow v * v$; $n \leftarrow n/2$;

sinon

$u \leftarrow u * v$; $v \leftarrow v * v$; $n \leftarrow (n - 1)/2$;

fsi

ftantque

retourner $u * v$;